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

TEACHING SYSTEM MODELING, SIMULATION AND VALIDATION

Jörg Desel

Lehrstuhl für Angewandte Informatik
Katholische Universität Eichstätt
85071 Eichstätt, GERMANY

ABSTRACT

Simulation is used in the design process of dynamic systems. The results of simulation are employed for validating a model, and they are helpful for the improvement of the design of a system with respect to both, qualitative and quantitative properties. The paper concentrates on these aspects and applications of simulation in education, advocates its presence in student curricula, presents building blocks of education modules for simulation and validation with respect to both content and method, discusses requirements for simulation and validation education, and finally suggests the integration of simulation teachware in virtual classrooms and distance learning environments. Modeling and simulation is almost necessarily based on modeling languages with precise semantics. In education as well as in practice, suitable computer tools should be employed. We suggest Petri nets with sequential semantics and partial order semantics as a modeling language. The contribution is based on experiences from several university courses on system modeling and simulation with Petri nets, including practical training. Moreover, relevant concepts from recent distance learning projects are mentioned.

1 INTRODUCTION

The handling of abstraction and of models is generally claimed to be one of the central issues of computer science. The same topic is also central in engineering and science. However, little attention is paid to the process of model creation and validation. This observation holds for research as well as for education. In industrial practice computer scientists have to create models from existing or planned reality as one of the first steps in system design. A thorough knowledge of theoretical topics which are usually taught to computer science students - like model semantics, verification techniques, transformation of models within one or between several modeling languages - is a necessary prerequisite for this work but it does not really give information about how to create and validate models. In other words: We do not teach students to build the right model but only consider theoretical issues within the world of models in detail. Simulation of a system model is an approved method for validation if the simulation results can be compared with the intended behavior of the system.

As observed in (Denning 2000), experimental methods in computer science have not yet received the deserved general recognition in universities, although these methods have frequently lead to important developments of theories and of practical systems in practice. The creation of models and their simulation are the core issues in experimental methods. Simulation by playing with different values of parameters is particularly useful for the improvement of systems with respect to quantitative properties such as performance.

This contribution does not try to provide suggestions for any kind of simulation education but concentrates on the aspects of simulation based model validation and system evaluation (w.r.t. quantitative measures), motivated in the above paragraphs. It is organized as follows. In Section 2 we glimpse at application scenarios where simulation is used for validation purposes in early phases of system design. We will refer to these scenarios in subsequent sections. Section 3 presents building blocks that we consider essential for simulation skills. Consequently, it is argued that these topics are to be taught in education of simulation and validation. Concrete modeling languages for executable models are presented in Section 4. We report on experiences teaching these models and according tools in Section 5. Finally, Section 6 mentions some ideas about implementation of simulation matters in software for distance learning in an academic context, before some short conclusions end the paper.

2 SIMULATION AS A MEANS FOR MODEL VALIDATION

Any simulation technique can be taught separately for its own. However, for a proper understanding of simulation techniques and their right applications it is reasonable to embed the presentation of the technique into a larger

context, where simulation is applied for solving a problem. This statement implies in particular that we suggest not to collect simulation techniques in dedicated courses on simulation but rather provide single simulation techniques where needed. In this section we provide scenarios showing how and where simulation can be applied to system validation and performance improvement.

This contribution concentrates on models of discrete dynamic systems. The behavior of such a system is constituted by its set of runs, containing event occurrences and dependencies between event occurrences. In most cases, a system (which can be hardware, an algorithm, a set of rules or a combination thereof) is constructed with the goal to support or to generate a specific set of runs. Its model should exhibit a comparable behavior, i.e., if the runs of the model do not relate to the intended runs of the system then something is wrong with the relationship between the system and the model. The creation of runs from a model and analysis of these runs will be referred to as simulation. Thus, by simulation of models and comparison of runs, information about the validity of a model with respect to a system is obtained. If the model includes time parameters, then simulation yields information of the performance of the system. More detailed, we distinguish four scenarios.

1. For a given system, a model is generated. It is validated by simulating runs and comparing these runs with the known or intended runs of the system. If the model's behavior differs from the system's behavior then either the design of the model contains flaws (that might remain hidden without inspecting runs) or the model is too abstract for representing the relevant behavior. Figure 1, restricted to the two upper rows, illustrates this principle of validation by simulation.
2. For a given model (i.e., a specification), a system is to be constructed. The validity of the model is checked via simulation of the model. Only if its behavior meets given requirements for the system's behavior or corresponds to the needs of the users, the specification is correct with respect to the application. An illustrating figure looks like the one shown in Figure 1 except that the uppermost horizontal arrow (annotated by "Build Model") should show in the reverse direction and can be interpreted as "Create System".
3. For a given set of runs (i.e., a specification) a model is generated. From this model, a system is created. In general, the model might have more runs than those given by the specification. Consequently, the same holds for the system. Some of these additional runs are admissible with respect to the application, some are not. Simulation of the model and interpretation of its runs yields an improved specification. In addition to the modification of 2., the architecture of Figure 1 would need an additional arc from "Simulated Behavior" to "Model".
4. A system might be perfectly valid and correct but terribly inefficient. When the relevant performance parameters are already included in the model then simulation serves for a performance prognosis of the system. In particular, questions about quantitative consequences of changing performance parameters can be efficiently answered using simulation. If the system to be constructed has certain performance constraints then simulation of the model helps to tune the system such that these requirements are met. In Figure 1, the third row illustrates this aspect of simulation application in system design. If for performance reasons a model is significantly changed then in general the prior validation step has to be repeated.

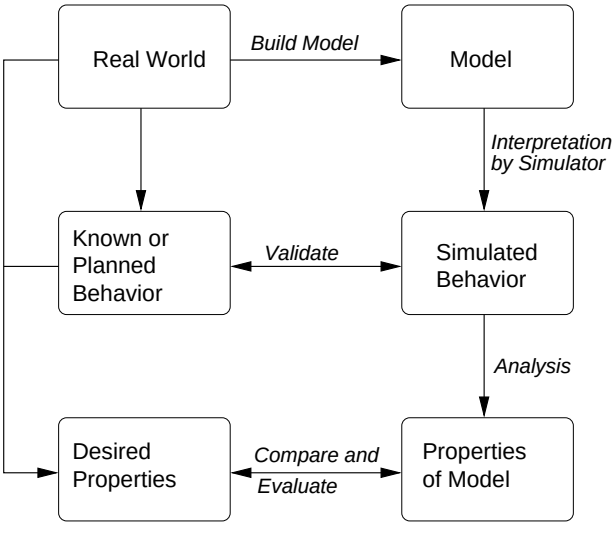

Figure 1: Application of Simulation for Validation

3 BUILDING BLOCKS OF SIMULATION AND VALIDATION EDUCATION

The scenarios of the previous section showed that simulation can be applied in different ways in the system design process. Usually several persons playing different roles are involved in this process. Designing a model, validating a model with respect to users of the current application, improving a model by tuning performance parameters and implementing a model are the most prominent tasks that need different types of expertise. Moreover, further important tasks on a meta-level are: designing a suitable modeling language, developing suitable modeling and simulation tools, embedding the validation steps in a management process etc. When talking about simulation and validation

in education, one has to be aware of the diversity of different skills necessary to perform the above mentioned tasks. This does not necessarily imply that every student must be able to fulfill all the above roles before reaching his or her final degree, but it does mean that without knowledge on the context of simulation including requirements of every part of a design process, it is not possible to properly understand an isolated task. So we advocate that a curriculum should include all following essentials of simulation and validation at least up to some degree:

3.1 Modeling is a Process Incorporating Machines and Humans

Modeling is a process incorporating machines and humans, and so is system design. At several steps, simulation can help to validate a models on different levels of abstraction in cooperation with the user. Here simulation does not only mean construction of runs (like program test) but also analysis of runs with respect to relevant parameters. It is important to keep in view that the simulation results are evaluated by humans. So their value highly depends on the quality of the interfaces of simulation programs. More importantly, the job of the system designer requires high social and communication skills. Only if he or she can filter the right collection of requirements from a set of partly imprecise, contradictory or incomprehensible ideas received within communication processes, and only if he or she can convince the relevant persons that the developed design is the optimal solution to the problem (often employing results from simulation) a system design project can be a success.

In education, all phases of a modeling and simulation process including communication of simulation results have to be exercised.

3.2 Simulation has Diverse Objectives

As emphasized above, simulation is used for system validation with respect to correctness of the model. This concerns in particular causal dependencies of events. In a next step, a correct model can be simulated for analysis of time, cost and other quantitative parameters. If the behavior can be improved, the model is changed and the first step of validation has to be performed again. Other parameters include for example the steady-state-behavior. Several different modeling languages are used, which can have strong interrelations. Simulation techniques can also be quite different, even if the general aim of simulation sketched above remains the same.

Students have to learn that simulation is not just one technique but a collection of techniques following a common philosophy. Whereas it is important to understand the philosophy, there is no point in including all reasonable objectives of simulation in one course. Instead we consider it important to find a precise answer to the question of simulation objectives before selecting techniques and tools.

3.3 Systems Can Run Automatically or Interactively

Systems can run automatically or interactively, and so should their models. For simulating interactive behavior, the user can simulate the interaction. So simulation needs to consider the role of the modeler as well as the user's role.

In education, students have to learn what kind of information is the input and the output to a simulation tool and to experience user interfaces. It is a good exercise to develop concepts for an own simulation tool including interfaces, taking the specific users and objectives into account.

3.4 Simulation Tools Differ with Respect to Diverse Quality Criteria

Simulation can be done manually, but for complex systems tools are necessary. Existing tools provide many different features, supporting different aspects of simulation. The above aspects lead to quality criteria for software tools, depending on the respective users and application domain. Important additional quality criteria are the functionality and the efficiency of a simulation tool.

For selecting the optimal tool in practice, one has to learn how simulation tools work, why some problems are more complex than others, and why some algorithms are faster than others. So a basic knowledge on the fundamentals of modeling languages and algorithms is necessary. This includes a rough understanding of mathematics, data structures and algorithm science.

3.5 Simulation Tools Can be Integrated in Software Architectures

Simulation can be viewed as a component in a software architecture which can be composed with other components. For example, a look-ahead simulation of an interactive system might detect undesirable states. Then a control component excludes runs of the system leading to these states. This concept can be used for automatic and for interactive systems (yielding intelligent assistant systems). For this kind of application, simulation input and output must respect suitable interfaces.

We do not consider this topic a *must* in education. However, apparently component based system design is the current trend in software engineering, and simulation can be viewed a very important component.

3.6 Simulation is Based on Content

As for all techniques, simulation can only be understood within a concrete application domain. In turn, each domain has specific simulation aspects. Examples include Distri-

buted Algorithms (fairness assumptions are essential), Data Base Transactions (concurrency control), Operation Systems (each run should never end), and Workflows (each run should eventually end). Clearly, this point closely relates to 3.2. The difference is that this building block concerns details of an application domain to derive the relevant objectives for simulation.

Since scientists will have to learn details of application domains in practice, it is a good exercise to do the same at least once in the university.

4 A MODELING LANGUAGE

We do not aim at defining all the best languages for modeling and simulation in this paper. Instead, this section presents one modeling language and its semantics, i.e., the set of runs of a model. Since simulation is based on (properties of) runs, a precise understanding of the runs of a model is a prerequisite for any simulation approach. The following section will report on experiences in education using the modeling language presented here.

The language we suggest for modeling and simulation of dynamic systems is given by Petri nets (for an introduction to Petri nets and its semantics see Desel 1999). Petri nets provide graphical means for specifying models that support an easy understanding. At the same time, Petri nets have a solid mathematical basis and there exists a rich theory on their semantics, their analysis, their simulation and their application in numerous domains. Moreover, many tools supporting Petri nets have been developed.

The core idea of modeling with Petri nets is to concentrate on local entities of objects, on local entities of events that can change the state of objects, and on the interrelations between objects and events. Objects and events constitute vertices of a Petri net graph, the interrelations are represented by directed arcs. Thus it is possible to represent large and complex systems in a readable way, provided each single event only depends on or changes a limited number of objects and each object can only be changed by a limited number of events. By avoiding global states as an elementary ingredient of the model, the size of a model only grows proportional to the size of a system, which allows to construct Petri nets in a compositional way. Global states are defined as a derived concept. Each set of coexisting local states can be viewed a global state.

Figure 2 shows an example of a very simple Petri net representing a process in a car factory.

The local objects are represented by circles. In this example, each object can be in one of two states: marked or unmarked. Viewing an object as a condition, these states read as *true* or *false*. In the example, initially exactly one condition is true (marked, i.e., carries a token), the others are false (unmarked).

The behavior of a Petri net is given by the occurrence rule. A transition, drawn as a square and representing an action, can occur if and only if all its pre-conditions are fulfilled. Its occurrence sets the pre-conditions of the transition to false and the post-conditions to true. In the example, the transition "Start" has only one pre-condition which is initially marked. Its occurrence leads to a subsequent marking where the only true conditions are "Parts-body-available" and "Parts-chassis-available".

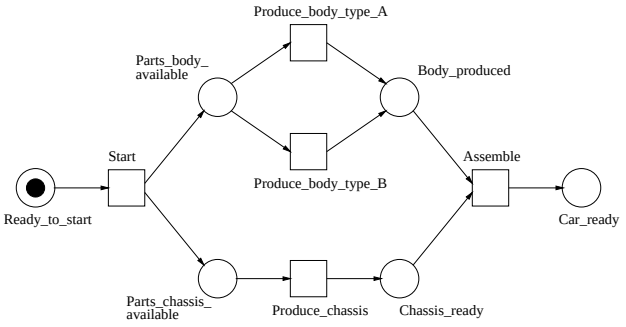

Figure 2: A Petri Net

The subsequent occurrences of transitions define an occurrence sequence which can be viewed as a single run of the model. Most simulation tools for Petri nets construct and explore occurrence sequences. The set of all occurrence sequences is known as the sequential semantics of a Petri net. One (of four) maximal occurrence sequence of the example is "S P_b t_A P_c A" (where the names of the transitions are abbreviated in an obvious way).

Sequential semantics has the disadvantage that concurrency is not explicitly represented. Instead, concurrent transitions, i.e., transitions that occur independently, are arbitrarily ordered. So concurrency yields an explosion of the number of runs. Moreover, concurrent transitions occurrences are not distinguished from alternative, mutually exclusive transition occurrences. Semantics based on partial orders do not exhibit these problems. A partially ordered run of a Petri net is defined as another Petri net, where concurrent actions are represented by transitions that are not connected by means of a directed path. Figure 3 shows two partially ordered runs of the net of Figure 2. Each of these runs corresponds to two occurrence sequences.

Transitions can be equipped with durations, time intervals or with stochastic time distributions. For each run, sequential or partial-order-based, the consumed time can be computed. Similarly, cost parameters associated to transitions yield total costs for runs.

We use Petri nets (more precisely, Place/Transition nets) as a modeling language. Runs of a model can be occurrence sequences or partially ordered runs. Occurrence sequences can be visualized in the Petri net model, partially ordered runs are visualized as in Figure 3. This visualization supports the validation. We concentrate on properties of runs concerning performance analysis. So, in our setting, Figure 1 translates to Figure 4.

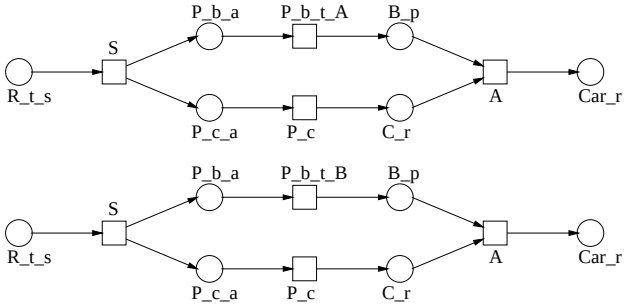

Figure 3: Partially Ordered Runs

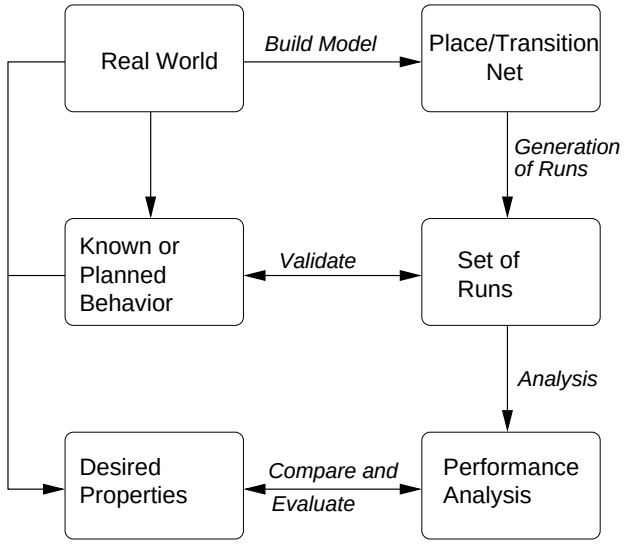

Figure 4: Simulation of Petri Nets

In classes, students have to understand the mathematics and the application of the model, its semantics and preferably a part of the underlying theory.

5 TEACHING SIMULATION

It is very hard to learn aspects of simulation without any practicing. Our experience from several courses on Petri nets, on workflow and on analysis techniques have shown that the combination of lecturing and practical exercises with computer tools works very well and helps students to deepen their understanding and to detect misunderstandings. These courses took place at the University of Karlsruhe, at the International University in Germany (in Bruchsal) and at the Catholic University in Eichstätt. The students had different backgrounds. In most cases, computer science was not the primary subject. So there was no deep general background on mathematics and computer science.

It turned out that the translation of a case given in form of natural language into a model was difficult for almost all students, no matter if they had prior knowledge on computer science or not. The understanding of the system dynamics was not a problem, even if the formal definitions were an obstacle for students from other areas. Perhaps surprisingly, the handling of the tools including import and export of models, parameters and diverse representation of simulation results was easy for all students after a relatively short time of vocational adjustment. The ability to interpret results of simulation with respect to the model or to the system took some more time. After some experience, patterns of relations were recognized. For example, variations of throughput-time, resource utilization, role allocation etc. were used in a systematic way to improve a model.

The cyclic process of building, simulating, and improving models was performed in groups of two up to four persons, ending with presentations of the final model and simulation results. In most cases the topic had been discussed extensively within the groups. Moreover, playing with alternatives and finding out good design solutions with the help of simulation apparently was a challenge and fun for the students. So here the combination of lecturing (including introduction to tools), group work and working with tools was very successful.

Here are some words about the used tools. As a simulation tool for sequential runs we employed EXPECT, developed in the group of Wil van der Aalst at the Technical University in Eindhoven (The Netherlands) (van der Aalst and Waltmans 1990 and 1991). This tool is tailored to work in combination with the analysis tool WOFLAN (van der Aalst et al. 1997) and the commercial tools COSA (a workflow management tool) and PROTOS (a business process modeling tool). The students had to use all these tools in combination for given case studies. Main results that had to be presented at the end of a course was a correct design and a distribution of resources to activities allowing for performance values that are either optimal or meet given performance requirements.

For partially ordered runs, we used VIPtool, developed in the group of this author (Desel 1999 and 2000). The combination of partially ordered runs and performance is desribed in (Desel and Erwin 2000). The architecture of the VIPtool together with links to the above mentioned tools as applied in the courses is shown in Figure 5.

6 DISTANCE LEARNING

Currently we are working on the integration of lectures on modeling and simulation in a computer based learning environment. The goal is to transfer the positive experience of combining lecturing, practicing and group work in a web based education tool. Central components of this tool are content modules that can be combined in different but restricted ways. Moreover, these modules can be used in different ways, for assistance of conventional lectures (overhead transparencies, scripts) as well as for explorative learning and guided tours (Desel et al. 1999). “Toy” simulation tools can be integrated in such an environment as

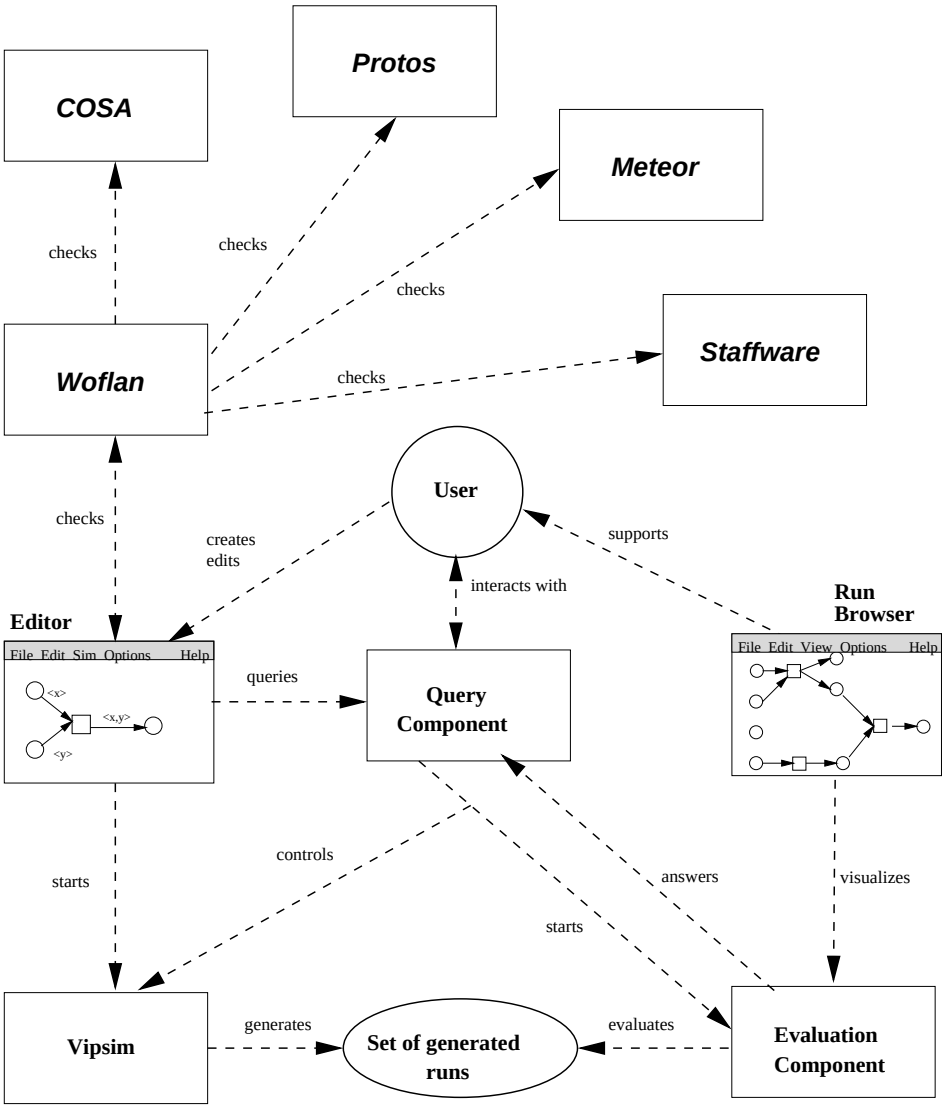

Figure 5: Architecture of Tools

applets, but for serious applications the students have to switch to "real" simulation tools. These tools can easily be started from the education tool but there is no general way to provide simulation results back to the education tool. Current research is on investigating such interfaces, in particular to support the common work on a simulation example by groups located at different sites.

This work is done in the context of several running governmental research projects on virtual universities, namely ViKar (Virtueller Hochschulverbund Karlsruhe), VIROR (Virtuelle Universität Oberrhein) and vhb (Virtuelle Hochschule Bayern). The overall aim of these projects is to feature the construction of learning modules that allow to be accessed via the internet (distance learning). A further activity within these projects is the use and the improvement of other new technologies in education. For example, lectures on simulation have been synchronously transferred to other universities via mbone (for a desription of this and further new media in education see Jiménez-Peris et al. 2000).

Whereas in these projects it turned out that the success of computer based teaching systems often suffers from lacking attractiveness compared to conventional educational settings, things look more positive in the specific domain of modeling and simulation. As mentioned above, modeling and simulation is integrated in a cyclic development process where several persons and tools are involved. Moreover, semantics of modeling languages and features of involved tools are not very easy to understand for students. So, by its very nature, modeling and simulation needs theory, tool knowledge, group work and tool usage - a combination that can be provided by a computer based environment. For simulation of interactive systems including social systems, different students at different locations

can play the role of the distributed environment. Taking the success of distributed internet games into account, students should have a very high motivation to apply these techniques for learning modeling and simulation issues.

7 CONCLUSIONS

Aspects of simulation education suggested in this paper range from experiences based on several courses to visions for distance learning. The main contribution is meant to be a collection of important criteria that have to be observed in courses for presence education and also when moving to education supported by new media. The detailed parts of the paper are restricted to applications of simulation in system development and particularly in validation and to a particular modeling language supported by a set of tools. The more general statements, however are claimed to be relevant for simulation education in general.

REFERENCES

van der Aalst, W.M.P. and A. W. Waltmans, 1990. Modelling flexible manufacturing systems with EXSPECT. In *Proceedings of the 1990 European Simulation Multiconference*, ed., B. Schmidt, 330-338. Simulation Councils, Inc., Erlangen.

van der Aalst, W.M.P. and A. W. Waltmans, 1991. Modelling logistic systems with EXSPECT. In *Dynamic Modelling of Information Systems*, ed., H.G. Sol and K.M. van Hee, 269-288. Amsterdam: Elsevier.

van der Aalst, W.M.P., D. Hausschildt and H. M. W. Verbeek, 1997. A Petri-net-based tool to analyze workflows. In *Proceedings of the Conference on Petri Nets in System Engineering (PNSE'97)*, ed., B. Farwer, D. Moldt and M.-O. Stehr, 78-90. Fachbereich Informatik, Technical Report FBI-HH-B-205/97 Universität Hamburg, Germany.

Denning, P. J., 2000. Computing the profession. In *Computer Science Education in the 21st Century*, ed., T. Greening, 27-46. New York, Berlin, Heidelberg: Springer.

Desel, J., 1999. Validation of system models using partially ordered runs. In *Proceedings of the 13th European Simulation Multiconference,* ed., H. Szczerbicka, 295-302. Simulation Councils, Inc., Erlangen.

Desel, J., M. Klein and W. Stucky. 1999. Virtuelle Kurse durch Wiederverwendung didaktischer Lehrmodule. Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Bericht 395, Universität Karlsruhe, Germany.

Desel, J., 2000. Validation of process models by construction of process nets. In *Business Process Management,* ed., W.M.P. van der Aalst, J. Desel and A. Oberweis, Lecture Notes in Computer Science 1806: 110-128, Berlin, Heidelberg, New York: Springer.

Desel, J. and T. Erwin. 2000. Modeling, simulation and analysis of business processes. In *Business Process Management,* ed., W.M.P. van der Aalst, J. Desel and A. Oberweis, Lecture Notes in Computer Science 1806: 129-141, Berlin, Heidelberg, New York: Springer.

Jiménez-Peris, R., C. Pareja-Flores, M. Patiño-Martínez and J. Á. Velázquez-Iturbide. 2000. New technologies in computer science education. In *Computer Science Education in the 21st Century*, ed., Tony Greening, 113-136. New York, Berlin, Heidelberg: Springer.

ACKNOWLEDGMENTS

The tool kit used in the mentioned courses was developed by Wil van der Aalst and his group at the Technical University in Eindhoven. Actually, during a visit of him at the University of Karlsruhe I experienced this kind of simulation education the first time. Thomas Erwin supplies technical and organizational support for the classes since several years, and his remarks helped to improve the present paper.

AUTHOR BIOGRAPHY

JÖRG DESEL is a Professor of applied Computer Science at the Catholic University in Eichstätt, Germany. He received his diploma degree from the University in Bonn, his doctoral degree from the Technical University in Munich, and his habilitation degree from the Humboldt-University in Berlin. He is a member of the Gesellschaft für Informatik and chairman of its special interest group on Petri nets. He has served as a chairman for the track on simulation validation methodologies at the European Simulation Multiconference 99. His email and web addresses are <joerg.desel@ku-eichstaett.de> and <www.informatik.ku-eichstaett.de>.