Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

MODELING AND SIMULATION IN HIGH SCHOOL EDUCATION - TWO EUROPEAN EXAMPLES

Henry Herper

Institute for Simulation and Graphics
Otto-von-Guericke University
D-39016 Magdeburg, GERMANY

Ingolf Ståhl

Department of Managerial Economics
Stockholm School of Economics
S-11383 Stockholm, SWEDEN

ABSTRACT

Discrete simulation is a suitable application area for several disciplines in high schools. One such discipline is Computer Science. Other disciplines are e.g. mathematics and business. The main principles of modeling information can be practiced in a suitable simulation project. For the implementation of the model on the computer, two different versions of Integrated Development Systems for GPSS have been developed mainly for the purpose of use in high schools. In Germany the version is WinGPSS, dedicated to Windows, and in Sweden WebGPSS, first implemented on the Web. Both systems use the same micro-GPSS simulator kernel. The paper discusses the goals with, and the experience gained from, the use of these GPSS versions in high schools. Some of the most recent developments of WinGPSS and WebGPSS are also presented.

1 THE GERMAN EXAMPLE (HENRY HERPER)

The goal of teaching Computer Science, in Germany called Informatics, in high schools is to transmit basic principles, methods, applications, ways of thinking and working within Computer Science, as well as to give an understanding of the societal importance of information systems. The contents and the methods should have a general and all-round educational character and precisely in this area support the independent acquirement of capabilities and skills.

Since the educational program in Germany is determined by the individual states (Bundesländer) and Computer Science is the most recently introduced subject, there are very different methods of teaching Computer Science in the different German states. The following account refers mainly to the education in the state of Sachsen-Anhalt (south-west of Berlin and Brandenburg). Here the teaching of Computer Science begins in grades 7 und 8, which as regards age can perhaps be compared to grades 8 and 9 in the US, since many students start at the age of seven in Germany. This early teaching is focused on basic work with the computer, involving word processors, spreadsheets and presentation programs. For this purpose there is each week one hour available in the curriculum. In grades 10 to 12 the students are offered a course in Computer Science, in which the students get the foundations of Computer Science. For this course two class hours per week are provided.

In the last few years, the focus of the teaching of Computer Science in Germany has shifted from programming languages to modeling of information, which includes all stages of solving a problem with a computer. It is here necessary to find problems that are suitable in the sense that the problems can be completely worked through to a solution in a course. In this basic education in Computer Science, the students learn a programming language. This language is usually object oriented. The coding of a program has proved important for motivating the students, since they can in this way test the completeness and correctness of their modeling approach.

After completing the basic education in Computer Science, which usually takes two years, students choose between optional courses from different areas of Computer Science. One of these courses is *Introduction to Modeling and Simulation*. One goal of this course is to confirm the knowledge and skills gained during the first two years of courses in Computer Science and to apply them to a new area. For this course some 32 classroom hours are available. The goal of the school authorities is that the students in this course should

- learn to use suitable abstraction techniques for the production of the simulation models,
- recognize that a model describes an excerpt from the real world, corresponding to the chosen level of abstraction,
- learn about insights to be gained by making inferences from the experiments with these models
- learn to work with a simulation tool and to implement simple computer models on their own,
- learn basic methods for visualizing the results and
- learn to interpret the results of a simulation run and evaluate them critically.

In these general goals it has not been stated whether the focus should be placed on continuous or discrete simulation. Traditionally, continuous simulation has had a strong position in Computer Science in German high schools. The students can implement simple continuous models using a general programming language, learnt during the previous years. However, with new simulation tools arriving, discrete event simulation is growing steadily in importance.

In cooperation with the department for the education of future teachers of Computer Science at the Otto-von-Gueriecke University in Magdeburg, the basics of a simulation course for high school students was developed and tested in a number of schools in Sachsen-Anhalt. These future teachers, who study the pedagogic principles of teaching Computer Science during three years, have not only learnt discrete events simulation, but have also participated in the test program. This course is based on discrete simulation with the use of a GPSS based simulation system with an Integrated Development Environment. This software will be discussed further below. Since the students can model e.g. service systems with which they are well familiar, there has been a high degree of motivation and acceptance for this kind of development of the course.

In the first part of the course, the students get acquainted with the concepts used in modeling. The concept *model* is already known from other subjects taught in school, especially within science. In Figure 1 we present some German school subjects in which models are discussed. One can systematize different model concepts and, on the basis of these, deduce concepts that are useful also within Computer Science.

The teacher presents different application areas of modeling and simulation to the students. In connection with this, the students will analyze and classify different models. The students learn that models are abstract pictures of a real or hypothetical system **with regard to a specific goal formulation**. The students also get to know different classification criteria and to recognize the connection between system attributes and the purpose of the modeling process.

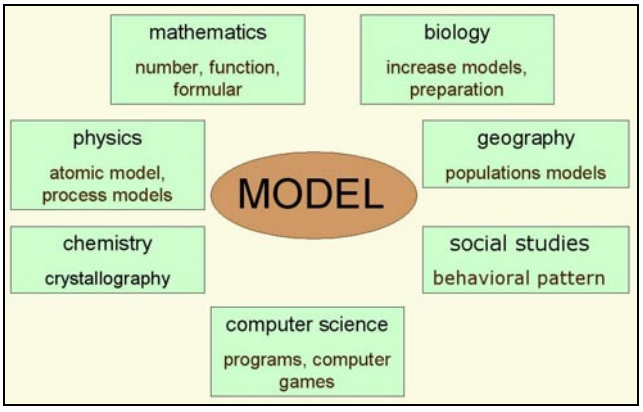

Figure 1: Models in High School Education

The teaching of **simulation** has been carried out in different ways. In many cases the focus on the course has been on an individual project of the student. This requires first of all that the student selects a suitable system to study and next formulates a concrete goal for the project.

The selection of the system to be modeled will to a large extent determine the success or failure of this course. An example of a successful simulation of a small service system is described in Herper (2001). There are hence some important points to think about when selecting the real system and the goal of the simulation study:

- The student should know the real system well,
- the description of the system must be available in a form that is easy for the student to understand,
- the project must be solvable within a limited time,
- the model must be such that it can be implemented with the available modeling and simulation packages,
- the solution of the simulation problem should imply some kind of "value added" for the student, so that the students feel motivated to do considerable work.
- learn basic methods for visualizing the results and
- learn to interpret the results of a simulation run and evaluate them critically.

The selection of the system can be done by the teacher and student in a dialog. Our experience from the educational efforts this far indicates that the students often suggest a system that is too complex. Here it is the task of the teacher to limit the complexity to something that can be realized within the stated time limits of the course. Examples of successful systems are school cafeterias, street crossings and small production systems.

As a first step, the students carry out a systems analysis study and do a delimitation of the problem that corresponds with the formulated goal of the study. Here the students learn a new way of working. They are from work within the sciences accustomed to problem formulations that contain a precise specification of all that is needed to get the desired solution. Even in simulation classes, the teachers have earlier often presented system descriptions in textual form, with a high degree of abstraction and with the necessary parameter values given.

Since in real projects, people doing simulation spend more than one third of the total time on the collection and analysis of input data, we have for our school projects regarded it as essential to allow a significant amount of time on this input phase also in the teaching in high schools. Project work has shown that the students have great problems in determining those attributes of a real system that are necessary for the modeling process. The necessary experience can only be gained through independent exercises.

The students are familiar with the different phases of systems analysis from the earlier software development projects. They have there learnt the methods of abstraction

and reduction as steps in the development of an abstract model. For the students, different sources of data for the analysis of a system are available.

In Figure 2 below, we present some forms for the description of the real system. The direct observations, including practical measurements, are the best methods for developing the students' necessary understanding of the system. If the students themselves have access to the real system, then the modeling phase can be more complete. First, the problem of defining the limits of the model must be tackled. The students then also have to make an explicit decision on when to collect the data.

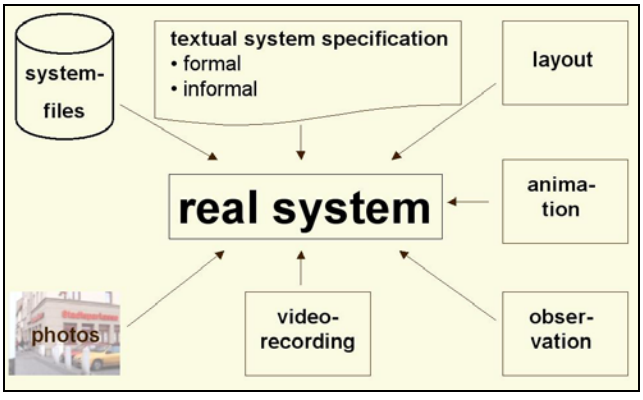

Figure 2: Different Kinds of Real System Specifications

The alternative that on the other hand leads to the least amount of modeling is the case when the system is described in text, with all the data given. In fact, textual descriptions often just give functions as regards e.g. the arrival times of cars. The interesting task of how to choose a suitable distribution based on e.g. many hundred observations is generally lost in the most common case of textual descriptions. In this case there is not very much modeling to be done by students, rather only coding. Intermediate forms are provided e.g. by videos or by animations of the system, produced by others. These forms can be used in cases when students do not have access to a real system or time is too short to study such a real system.

We can exemplify this by the case of a street intersection, where the problem is to find suitable times for the red and green lights. Textual descriptions of such a system are available in many simulation textbooks and coding can then be done right away. In the case when a real traffic situation is studied, students can stand in the street crossing, make a drawing of it, possibly based on a map, and then decide on how to limit the system, e.g. whether some adjacent street has to be taken into account, since this might affect the flow traffic at the street crossing. The students will also have to decide when they are going to get the data from the street crossing and for how long time. Data is collected in the form of e.g. the inter arrival times of the cars, the time it takes to drive a certain predefined distance, the length of the waiting lines of cars and the total times spent by the cars in the crossing.

In the case of using videos, all this data can also be collected, sometimes more conveniently and with greater certainty, since the students can run the video several times. Some videos, like those made in Magdeburg, also have the benefit that one has made some video shots also from above, so that it is easier to precisely measure e.g. the times needed for different cars to travel different distances. Thus the use of videos allows the students to practice several, but not all, modeling decisions.

An alternative to using video clips of different lengths is to use already made animations. There are quite a few such animations made e.g. of traffic systems in Proof Animation. An advantage compared to videos is that it is much easier to measure e.g. distances and times. Proof animation has a clear clock and the lengths of different paths can be easily established. In this way a little more modeling has already been done than in the case of videos, but definitely there is much more place for modeling than in the case of textual descriptions. It should here be mentioned that all versions of Proof can convert trace files into a format that allows them to be viewed with the royalty-free Proof Demo viewer of the Student Proof (Wolverine 2002). With low-cost or free student access to this software, one can distribute quite large animations, often at a lower total cost than with videos.

The students are next to produce verbal, but informal, descriptions of the system, like in pseudo-code. For some processes it has also been possible for the students to produce a more formal description. If the student has selected a model from the class of server systems, then the student can use concepts such as source, sink, transaction and service station. For the description of some processes, algorithms have been used. Some German students have learnt to use Nassi-Schneiderman diagrams and flow charts in the earlier courses. These methods cannot, however, describe the development of the process over time. The recognition of time as a system attribute and the modeling of different processes with regard to their time requirements put strong demands on the students as regards their ability to make abstractions. For the formal description of processes that are dependent on time, some students have used the modeling language UML, which they have studied in a preceding course. However, for small models it has been regarded as sufficient that the students use informal descriptions.

When it comes to the next step, namely of collecting data, the students have used some skills acquired in earlier courses in chemistry, physics and biology as regards how to carry out observations. In mathematics they have furthermore learnt the basic methods for selecting input probability distributions.

After the model has been built up in this fashion and data collected, the students are expected to do a first validation, i.e. to answer the question: "Does your model de-

scribe the system correctly and completely with regard to the level of abstraction that you have chosen?" For cases when the students have chosen a system that they know from earlier experience, they have usually been able to answer this question easily. The validation of model of a server system that is well known to the student leads to much smaller problems for the students than models in other areas of Computer Science, e.g. continuous simulation and data bases, etc., where the students do not have access to a corresponding real system.

The next step in the modeling process is the implementation of the model on the computer. For the coding process, it is important that the students can chose a suitable tool. It should be mentioned that the students at this stage have already studied a general programming language, during 2 hours a week for 20 weeks. Most often this has been an object-oriented language like Delphi, Java, Visual Basic or C++. It is, however, not enough to know a programming language, but it is also necessary to understand the basics of simulation. Since only around 8 hours can now be regarded as available for the coding work, it is impossible to do the coding of an interesting project regarding a service system in a general programming language. The implementation of models from service systems with a general programming language requires much more time.

The demands that can be put on a simulation system that can be used in high school education have been discussed in Herper und Ståhl (1999). The conclusion here was that the modern versions of GPSS, WebGPSS and WinGPSS, are the most suitable systems for this kind of high school education. A more detailed discussion of the characteristics of these systems, as well as the reasons for using them, will be discussed later in this paper. It here suffices to note that in the German simulation course discussed here, the WinGPSS system has been used several years with success for the coding of the student projects.

The next central point in this course is the planning, execution and evaluation of experiments. It is in this case important that the students learn to separate the test phase clearly from this experimental phase. For the execution of the experiments with a model, one must have a validated and verified model.

In the first experimental phase, the students learn the effect on the results of using different sequences of random numbers for a stochastic model. The students also learn about the methods of generating random numbers and their importance. The second phase of the experimental activity concerns the changes in the model parameters. Within the framework of the experimental planning, for each change of a parameter, one should establish the purpose as well as the expected changes in the results. It is important that the changes of the parameter are made within the area of validated values. The third phase concerns model experiments by changes in the model. One must then investigate whether the foreseen changes are possible as regards the real system. A change of the model requires a new verification and validation phase. Through this form of experimentation the students learn to see simulation as an **iterative** process.

The model formulation process, the test phase and the simulation experiments are next to be summarized into some form of documentation. The student is expected to make an oral presentation of the results, using e.g. PowerPoint, in which the student should demonstrate the relationships between the input parameters and the results. It is important that the student reveals a critical attitude to these results and analyses possible sources of errors. The student can here show that she has gained insights through the modeling and simulation process and can draw conclusions of interest to the real system from the experimental runs.

2 THE SWEDISH EXAMPLE (INGOLF STÅHL)

The start of the activities dealing with the teaching of discrete event simulation using WebGPSS started in 1993, when a Swedish high school student, D. Kudrén, approached me to learn more about GPSS. He had heard about GPSS from his father, who had attended a course on simulation with micro-GPSS that I had given in the executive program at the Stockholm School of Economics, the SSE. He wanted to use GPSS to make a simulation study of his school cafeteria as his senior year project work. It should here be mentioned that during the last year of high school in Sweden, when the students are 18 – 19 years old, they spend around one month on a project of their own choosing. Kudrén then went on to make really impressive project work on this school cafeteria.

Two years later the SSE was approached by a newly established Swedish foundation, the KK-foundation, for which one of its goal was to increase the general knowledge of Information Technology among young Swedes. One was investigating whether any Swedish university or college had developed some software for its teaching activities that, after some transformation, could be used in Swedish high schools. We had at the SSE since 1979 gradually developed micro-GPSS as a streamlined version of GPSS, basing the simplification of GPSS to a great extent on the feedback that we had received from the 300 students, to whom we were then each year giving a ten-classroom-hour course in GPSS. In fact, when students repeatedly made the same mistake in syntax, we always considered the alternative of simplifying the syntax.

With the school cafeteria work, mentioned above, in mind, I thought that it would be in line with the KK-foundation's goal to make a GPSS system available to Swedish high schools students to help them use simulation for their senior year project. There was indeed a great set of possible project areas for the high school students. Many Swedish high school kids work during summer vacations

and could hence do projects e.g. on hospitals, shops, gas stations, etc. Many have experience from family run small businesses. The KK-foundation agreed that it would indeed be suitable to entice students to do their project work on a real system that they knew well, at the same time using modern IT-technology.

To make micro-GPSS more suitable for this project work, some changes had, however, to be made. Micro-GPSS, like other earlier GPSS versions, is text-based and it was obvious that the high school students, only used to graphical software, would demand a GUI version. Secondly, in order to make GPSS more readily available to the students, it should be available on the Web. This thus become the starting point of WebGPSS. I shall below, in section 3, give some further details, but I shall here mention how WebGPSS has been used in this project work.

It should be stressed that, in contrast to the case in Germany, there has not been any central decision on including simulation in the curriculum. Instead, virtually all learning of the system has to be done as self-study. In order to make it as easy as possible for the students to learn to use WebGPSS on their own, four things were made:

1. The system is, as far as possible, self-explanatory, by e.g. having dialogs for the input of block operands, with understandable operand descriptions (see e.g. Figures 4 and 7 below).
2. The Web-system also contains an extensive Help-system.
3. The Web-system contains a number of tutorial lessons, at present around 30, each of 10 – 20 HTML pages.
4. The Web-system contains a number of program examples, at present 84. The programs are to some extent built up step by step, many relying on previous examples, with only one new feature added at a time. By building these programs themselves, the students can learn the basics of WebGPSS without access to a teacher.

It should be mentioned that these Web-based features in some cases have not been quite sufficient. We have found that students often prefer to have printed material to only electronic. Hence, we have provided also the tutorial lessons in Swedish in printed form (Ståhl 1999). There is also a more substantial textbook in English (Ståhl 2002). Furthermore, in order to introduce the students to the availability of the WebGPSS system and to give students a first hands-on training on the system, which, at least as regards some students, appears necessary to ensure that they can use WebGPSS on their own, we have developed a four-class-room-hours program, which we have run at a number of Swedish schools. Four hours appeared to be the maximum that we could get of unscheduled time in these schools. It should, however, be noted that students in four hours can progress fairly far. For example, after four hours students are able to write the program of "Boris vodka shop", presented in section 3.1. We have also had a one-day seminar with a number of teachers, in order to prepare the teachers to run similar short introductory programs.

It should also be mentioned that, although there is no organized teaching of modeling, we have made some text available on the Web for the students on the whole simulation process, including e.g. input/output analysis, experimentation etc. We also provide some animations, including some made by SSE students, for those students who do not have access to real cases, but also for general inspiration.

3 WINGPSS AND WEBGPSS

We have above noted that we in Germany and Sweden have used two GUI-based versions of a streamlined GPSS. We shall in this section first provide some additional comments on the reasons for this choice. We shall then discuss two of the major features of these systems, namely the multi-language aspect and the extensive error trapping/reporting system. Finally we shall present some aspects of the most recent developments of these systems.

3.1 Why GPSS?

As the high schools projects generally deal with some kind of service systems, the focus on a discrete events oriented simulation system is natural. According to McHaney (1996), for the implementation of such simulations, three types of system were most frequently used: 1. Block Based Systems (BBS), like GPSS, Arena/SIMAN and AweSim/SLAM (31 percent), 2. Animation Oriented Systems (AOS), like Witness (22 percent) and 3. General Programming Languages (GPL), like C++ (21 percent).

We have above mentioned why a GPL was not regarded as suitable in Germany. In Sweden, where we could not expect any knowledge of computer programming, a GPL was even less suitable. The reasons for not choosing an AOS are given in Herper and Ståhl (1999). We shall here bring only one to attention, which can be exemplified by the Boris vodka shop problem below. This is a problem that we usually have our students solve at the end of a 4-hour-session.

"At a store, run by Boris and Naina, customers arrive at rate of 7 ± 3 minutes. A customer first goes to Boris and chooses his bottle. This takes between 3 and 7 minutes. Next he goes to Naina to pay for the bottle. This also takes 3 to 7 minutes. Finally, he returns to Boris to pick up his bottle. This takes between 1 and 3 minutes. He then leaves the store. There is one waiting line in front of Boris and one in front of Naina. A customer returning to Boris to pick up his bottle has to start at the end of this line again. The store is closed after eight hours".

This example has been used in an experiment, carried out with a class of Latvian students with no prior experience of simulation. Half of them had four hours of an

AOS, the other half four hours of GPSS. At the end of each of these sessions, the students were asked to model the Boris problem. While none of the AOS students could do this, all the GPSS students could do so. We have also asked the vendors at the WSC during several years to solve the problem. The BBS vendors solved it in 5 minutes; the AOS have required more than 30 minutes. (For details see Ståhl 2002b.)

The reason for the difference can be explained by the simple logic of the diagram block in GPSS in Figure 3. We here see that the customers can come twice to Boris, i.e. Boris can be located in **two** different places in the program. In an AOS, a server like Boris must be in only **one** place and we must add complex logic to keep track of whether a customer comes the first or the second time to Boris.

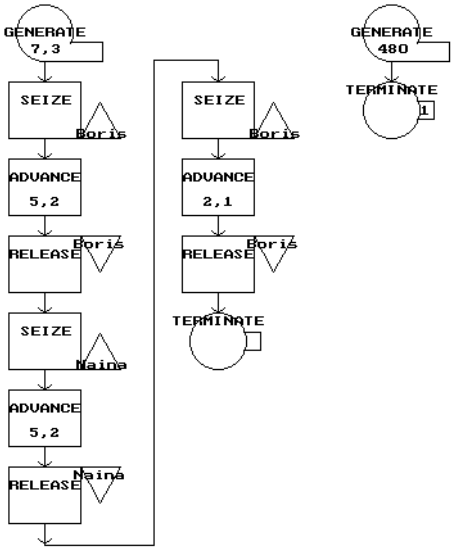

Figure 3: Block Diagram of Boris Vodka Shop in GPSS

When selecting among the Block Based Systems, GPSS has been the natural choice. Both in Germany and Sweden, GPSS has a long tradition of being used within education. For example, a 1997 survey showed that GPSS was the most used simulation system in education in Germany (Reinhard 1997).

We have, however, for the education in high schools not used the standard commercial version of GPSS, but in both Germany and Sweden used the simplified versions with a GUI. For the very limited time available for the whole course or the project, only a minimum of time can be spent on learning the software. We know from other experiments that a certain amount of material that in standard GPSS required 22 classroom hours, in WinGPSS or WebGPSS needed only 10 such hours.

The advantage over other BBS is similar. WebGPSS and WinGPSS appear to be the only systems that fulfill the criteria for suitable educational software discussed in the panel on teaching the "classics", i.e. GPSS, Arena and AweSim, to beginners (Ståhl et al. 2003). For example, in the case of a reasonably large simulation project, requiring e.g. 250 blocks and 500 simultaneously active transactions, the GPSS versions appear to provide the only free, or very low cost, alternative.

3.2 Multi-Language Focus

In most of Europe, one important feature, in which a simulation system for high school students distinguishes itself from a simulation system to be used only at the university level, is the importance of using the national language. At the university, where students are likely to run into simulation first after a couple of years of university studies, they will already be accustomed to reading literature in English. There has hence not been any problem using a completely English-based system e.g. for our teaching to the third-year college students in Sweden. When we were asked by the KK-foundation to produce a GUI and Web-based version of micro-GPSS for Swedish high schools, it was, however, regarded as very important to have the system in Swedish. This implies not only that we translated all the micro-GPSS output texts and error codes into Swedish, but we also wrote many lessons and HELP pages in Swedish.

With old micro-GPSS completely in English, we started in 1998 to reconstruct the micro-GPSS source code. Since we did not want to have two separate versions of the source code, one with English text and one with Swedish text, we broke out all commands that dealt with output text or error codes and put them into special files, which were then included first at compile time. These text files contain either ordinary text, like table headings, or error codes. We have one set of such text files for each language.

Although this separation of code and text took time to do, it soon paid off when we afterwards put micro-GPSS into German to be used with WinGPSS. The extra time for another language was not very large. We have now also contacted people to start working on French and Spanish versions of GPSS, since we believe that this translation effort is necessary in order to have any spread of GPSS into high schools in French and Spanish speaking countries.

As regards the two GUI systems, incl. the HELP systems, WinGPSS is still only in German, although we have heard a demand for an English version. WebGPSS, released in 1999, was originally only in Swedish, due to the requirement of the KK-foundation, but we started a translation into English in 1999, finished in the year 2000. There are tutorials, on WinGPSS in German, on WebGPSS in both English and Swedish, in all cases both in electronic and paper form (Ståhl and Herper 2003; Ståhl 1999; 2002).

3.3 Error Trapping and Reporting

We here want to bring the extensive error trapping and error code system of WebGPSS and WinGPSS to attention.

For an educational system, used mainly by novices, very prone to make errors, it is important that errors are trapped. In particular, it is critical that logical errors are avoided. In the construction of micro-GPSS, there was a focus on the "lead us not into temptation" principle, implying that constructions that by novices often lead to logical errors are not allowed and hence caught already as syntax errors. Many nasty logical errors that we have found students do in commercial versions of GPSS, like GPSS/H, are impossible to do in WinGPSS/WebGPSS. Furthermore, there is a very extensive system of error codes, containing over 500 different such codes. This has been accomplished largely with the help of our 6000+ GPSS students, who have been asked to report on all errors for which they have not found the error code understandable and helpful.

3.4 Development of WinGPSS Since 1999

The WinGPSS/WebGPSS systems are being continuously developed further. The feedback obtained in the education of the high school students, but also of the teachers, is being incorporated into the new versions. The WinGPSS and WebGPSS systems, as they were in their early form in 1999, are reported on in Herper and Ståhl (1999). Since this WSC paper is available on the web, we shall here mainly report on things that have been introduced or improved after 1999.

In WinGPSS, there has first of all been a substantial updating of all **block dialogs**, making the syntax very clear to the high school students. Figure 4 gives an example.

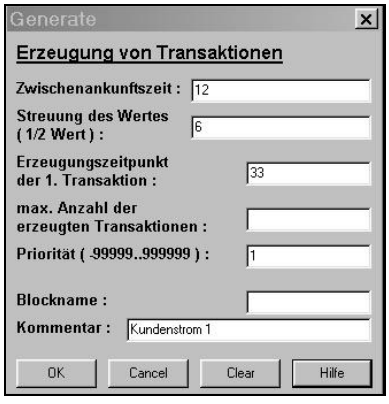

Figure 4: WinGPSS Dialog of the GENERATE Block

Furthermore, there has been continued work on the **Help system**. In all parts of Computer Science education, work with Help systems is demanded, since this constitutes a prerequisite for independent work on solving problems, not only with known tools, but also for learning to use new tools. For this reason, we have spent great efforts on a substantial Help system for WinGPSS. This should give support to the students when building the models, since in the Help system not only the syntax, but also examples of use, are presented for the different block types. The interpretation of the error codes and the correction of errors are in WinGPSS also supported by the Help system. The Help system of WinGPSS is, in fact, gradually being developed into a complete learning system.

Another important area of development of WinGPSS concerns so-called **block animation**. This development addresses the fact that students during the teaching of GPSS have experienced problems understanding how the simulation process works and how the transactions move through the system. In procedural programming languages, the commands are in general carried out sequentially, depending on the exact position in the program. The German students know about process control, e.g. from their earlier work with the debugger of the DELPHI IDE, but this kind of process control has little relevance for the GPSS programs. The critical factor is rather system time. Transactions move through the model, sometimes several at the same time, sometimes independent of each other, sometimes dependent on each other. It is also difficult for the students to understand how these parallel processes are carried out.

In order to visualize these movements of several transactions through the system, but also to support the validation of the models, we have for WinGPSS developed a simple block animation system integrated into the block diagram of WinGPSS, as illustrated by Figure 5.

The animation is produced from a trace file and the animation is hence a Post-Run-Animation system. A specific simulation run can be analyzed several times from different aspects. In the animation, every transaction is shown with its number, based on the order in which it was generated. In this way it is possible to follow the movement of a specific transaction during the run. We see the block statistics of both the current count of transactions present in a specific block, as well as the total number of transactions that have up to this time gone through the block. We also see the time of the simulation clock. Figure 5 provides a snapshot of the block diagram at time 259.9. We see e.g. next to the GENERATE block that 15 customers have arrived, and that customer

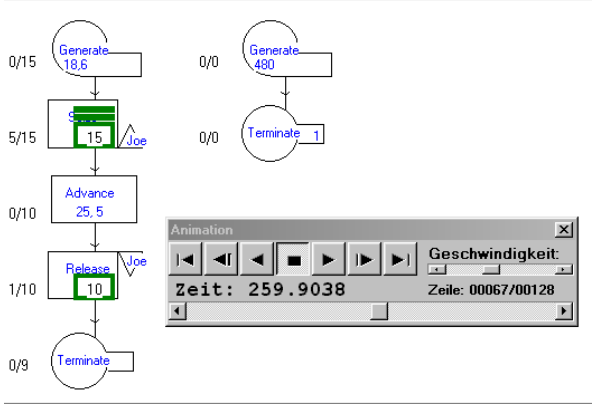

Figure 5: Block Animation of Joe's Barbershop

nr. 15 is waiting at Joe and, both from the figures and the symbols, that a total of 5 customers are waiting. We also see that the active transaction, customer nr. 10, is just going through the RELEASE block.

3.5 Development of WebGPSS Since 1999

One important change is that, while WebGPSS earlier was only available on the Web, in 2001 we also made a stand-alone version. The reason for this was that students encountered different technical problems running WebGPSS over the web. First of all, with a fairly large Java-based WebGPSS Applet (500 Kbytes), students with slow modems found it time-consuming running WebGPSS. Secondly, the WebGPSS Applet required that the SUN Java plug-in was installed the first time WebGPSS was run. In several schools, there was on the part of the system administrator a great reluctance to have this plug-in installed on the central server. The production of the stand-alone system was a fairly straightforward task and many students now run this version on their own computers.

The other important change in WebGPSS has taken place this year and deals with the number of block types. While micro-GPSS has a total of 22 block types, and WinGPSS also allows for all of these block types, we decided, when starting with WebGPSS in 1997, to include only 16 of these block types, shown to the left in Figure 6.

There were several reasons for this. Since WebGPSS was mainly intended for high school users, not getting very much classroom teaching, it was regarded as important not to overwhelm the high school students with details. We also wanted to limit the number of symbols in the symbol menu to make it readable, both on small screens (e.g. 12 inch) and with regard to what can be seen from the back row of the classroom when presenting the WebGPSS GUI in an introductory class with a computer projector. We also wanted to include Run and Start buttons in the symbol menu to make WebGPSS really user friendly and we wanted the students to be able to read the names of the blocks. Finally, we thought that the six block types left out would not be of interest to our high school students.

We have, however, found out that the functionality of some of these left out block types are of interest, not only to college students, but also to high school students in their project work. For example, the SPLIT block is of interest in many student projects. One interesting example, provided by R. Born, now put on our Swedish high school web, concerns the optimal stocking on a supermarket shelf of rapidly perishable foodstuff, like cottage cheese. Should the new cottage cheese be put in front, or in the back, leaving the old ones in front to be picked first by the customers? This is an example of a problem that many high school kids, working part-time in supermarkets, have personally encountered. The interesting program, which requires the use of the SPLIT block, shows that under realistic conditions the optimum is to place the new ones in front.

We have also found that the functionality of the PREEMPT and RETURN block types are of interest in student projects, like e.g. in emergency wards in hospitals, where less critical treatment is interrupted by the arrival of a critically damaged patience. Several high schools students have experience from summer work as hospital assistants, often in emergency wards.

Against this background, and not knowing what future demand could come from high school students, we have now decided to make the full functionality of micro-GPSS available in WebGPSS. We still wanted to keep the number of block symbols low, adding only a couple of block symbols to get a total of 18 symbols, namely the ones in the right hand column in Figure 6.

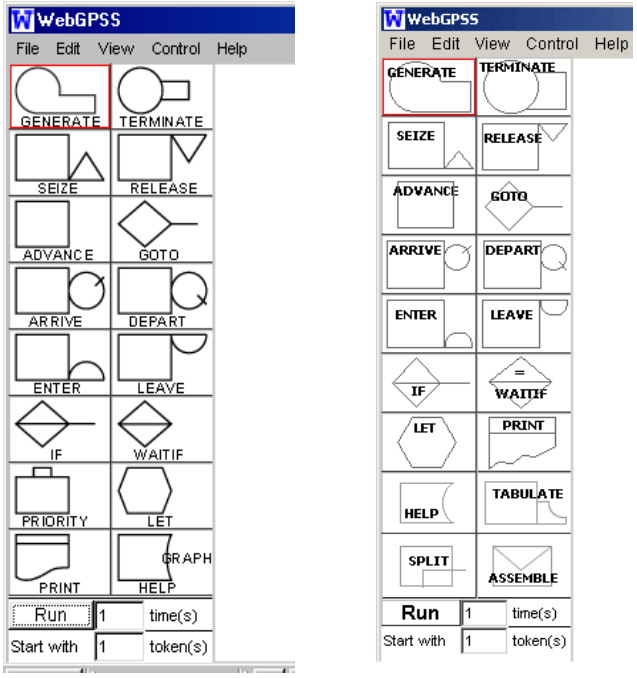

Figure 6: The Old and New Block Symbol Menus

Comparing the two symbol menus in Figure 6, we first notice that one block type in the old menu is missing in the new one, namely PRIORITY. We have now changed micro-GPSS so that we can use the LET block, e.g. with LET PRIORITY=1, to replace PRIORITY 1. Furthermore, there are three new block symbols in the new menu, TABULATE, SPLIT and ASSEMBLE. The question is then what happened to PREEMPT and RETURN. We have now allowed these to be handled by the ordinary SEIZE and RELEASE blocks. This can be seen by the revised operand dialog of the SEIZE block in WebGPSS in Figure 7. By clicking in the Preempt box, the SEIZE block turns into a PREEMPT block.

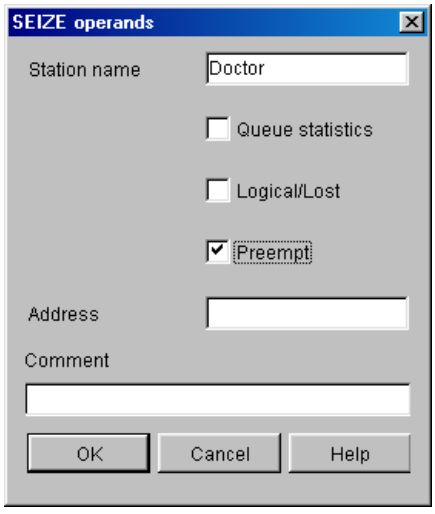

Figure 7: The New SEIZE Dialog

It should finally be mentioned that the SELECT block is now also handled by the LET block with a MIN (or MAX) expression. LET P$BESTQ=MIN(Q,1,6) can now replace the less understandable SELECT MIN 1,1,6,,Q of standard GPSS. Thus, we can now in WebGPSS with 18 block types handle all the 22 block types of micro-GPSS. More details on these changes are given in Ståhl (2003).

REFERENCES

Herper, H. and I. Ståhl. 1999. Micro-GPSS on the Web and for Windows - A Tool for Introduction to Simulation in High Schools. In P.A. Farrington, H.B. Nembhard, D.T. Sturrock and G.W. Evans (eds.) *Proceedings of the 1999 Winter Simulation Conference*, 298-306. SCS, Phoenix. Also available at the site <www.informs-cs.org/wsc99papers/042.PDF>.

Herper, H. 2001. Modellierung von Systemen - ein Applikationsgebiet im Informatikunterricht. In R. Keil-Slawik and J. Magenheim (eds.) *GI-Edition Lecture Notes in Informatics - Volume P-8*, 207-221.

McHaney, R. 1996. *Simulation Project Success and Failure: Some Survey Findings.* Working paper. Dept. of Management, Kansas State University, Manhattan, KA.

Reinhard, A. 1997. *ASIM Umfrage: Simulation in der Lehre.* At <www.fps.maschinenbau.uni-kassel.de/Forschung/Fabriksimulation/Sim_i_d_lehre>.

Ståhl, I. 1999. *Lektionstexterna i WebbGPSS.* Högskolan i Karlskrona/Ronneby.

Ståhl, I. 2002. *Simulation Made Simple with WebGPSS – A Tutorial*. SSE, Stockholm.

Ståhl, I. 2002b. Simulation Prototyping. In E. Yücesan, C. Chen, J. Snowdon and J. Charnes (eds.) *Proceedings of the 2002 Winter Simulation Conference*, 572-579. *SCS,* San Diego.

Ståhl, I. 2003. From 44 to 31 to 28 to 22 and now to 18 – Less becomes more in GPSS. In T. Schulze, S. Schlechtweg and V. Hinz. *Simulation und Visualisierung* 2003. SCS, Magdeburg.

Ståhl, I. and H. Herper. 2003. *Einführung in die Simulation mit Micro-GPSS.* Otto-von-Guericke-Universität, Magdeburg.

Ståhl, I., H. Herper, R. Hill, C. Harmonosky, J. Donohue and D. Kelton. 2003. Teaching the Classics of Simulation to Beginners. In S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, (eds.) *Proceedings of the 2003 Winter Simulation Conference.* SCS, New Orleans.

Wolverine. 2003. Using Proof Animation. 3rd ed. Wolverine Software Corp. Alexandria, VA.

AUTHOR BIOGRAPHIES

HENRY HERPER is in the Institute for Simulation and Graphics at the Otto-von-Guericke University, Magdeburg. His research interests include the modeling of logistical and manufacturing systems and the development of simulation tools for introduction to simulation. He is a member of ASIM and the GPSS-Users' Group Europe. You can reach him by e-mail at <henry@isg.cs.uni-magdeburg.de> and his web address is <wwwisg.cs.uni-magdeburg.de/isg/henry.html>.

INGOLF STÅHL is a Professor at the Stockholm School of Economics, Stockholm, and has a chair in Computer Based Applications of Economic Theory. He was visiting Professor, Hofstra University, N.Y., 1983-1985 and leader of a research project on inter-active simulation at the International Institute for Applied Systems Analysis, Vienna, 1979-1982. He has taught GPSS for twenty-five years at universities and colleges in Sweden and the USA. Based on this experience, he has led the development of the micro-GPSS and WebGPSS systems. He is also consultant in simulation to Swedish banks and industry. His email address is mailto:<ingolf.stahl@hhs.se>and the web address for his WebGPSS is <http://www.webgpss.com/>.