*Proceedings of the 2007 Winter Simulation Conference*
*S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.*

.

# TEACHING SIMULATION TO BUSINESS STUDENTS
# SUMMARY OF 30 YEARS' EXPERIENCE

Ingolf Ståhl

Department of Economic Statistics and Decision Support
Stockholm School of Economics
S-11383 Stockholm, SWEDEN

## ABSTRACT

I summarize my experience from having taught simulation to over 7000 students for over 30 years; to undergraduate, graduate and Ph. D. business students, executives and high school students, in five countries. I discuss how my students differed from other simulation students and my general teaching goals. I answer the question of why Discrete Events Simulation is important at a business school. I present the five main types of course modules that I have taught. I finally discuss my choice of DES software, explaining why I have chosen to use a streamlined GUI based version of GPSS, WebGPSS.

## 1 INTRODUCTION

In this paper I shall summarize my experience from having taught simulation to some 7000+ students, mainly in business, since 1976. Most teaching has been done at the SSE (the Stockholm School of Economics), but I have also taught simulation at NHH (a Norwegian business school) during 9 years, and at Hofstra University, NY, during two years. I have also given many smaller simulation courses, in Sweden at five universities, in Latvia and in Germany.

I have mainly been teaching simulation in three different forms: As a small part (e.g. 2 – 4 hours) of a Management Science or introductory computer course as a rapid introduction to simulation modeling, below called the "small course module"; as a substantial part (e.g. 1/3 with around 10 – 15 hours) of a course on some computer or OR methods, presenting the main ideas of a simulation, below called the "intermediate course module" and a "full semester course" of 30 – 40 hours, going slightly beyond the technical material of the intermediate course, but involving a substantial simulation project, often in a corporation. I have, however, not only been teaching ordinary university students, but also, on a "higher" level, executives in executive programs and, on a "lower" level taught and developed courses for high school students and their teachers. I have also taught simulation to doctoral students.

## 2 TYPES OF STUDENTS

The main bulk of my university students have been business students and hence business problems have been the main focus of my applications. I have, however, also taught engineering students, both in Sweden and the US. Most of these engineering students have been focusing on transportation, logistics, material handling and supply chain management.

I have taught very few students of computer science. The engineering students have hence in background and application focus been fairly similar to my business students. While most of my business students had no or little experience in programming, also many of my engineering students were not either experienced programmers. Almost all my students have hence been different from computer science students, e.g. in terms of programming background. My students have also been different from production students, focusing on manufacturing planning and plant layout, e.g. in engineering shops. These differences are of importance when I later in the paper discuss my choice of software for simulation.

## 3 GENERAL TEACHING GOALS

For all my teaching of simulation I have had some general goals. The basic goal has frankly been that I should be able to attract more students to my simulation courses, since I have preferred teaching simulation to other subjects. In order to attract students I have wanted my courses to be enjoyable, i.e. **fun**, but also to be helpful in future studies and to be helpful when landing a job.

In order to make my courses fun, I have tried to avoid overwhelming the students with difficult, often unnecessary, technical details. I have tried to ensure that students at a very early stage get to write simple programs that are not trivial. Furthermore, I have proceeded step by step, with simple examples in the beginning, so that no students are left behind at an early stage and loose the possibility to catch up. To ensure that students can use the simulation

software, all my teaching has been done in computer labs. The students have here, under my supervision, solved exercises of gradually increasing complexity.

I have furthermore always avoided pre-course knowledge requirements, e.g. of computing. Finally, students have not had to learn a new concept every time that a new and different thing was to be done. Students have liked to find that the new aspects can be handled using already known concepts, even if the programs thereby become slightly longer.

It should further be mentioned that I have in all my courses used a simulation package and allowed the students to do some modeling on their own. It is my belief that. as long as at least two classroom hours are available, one should try to teach some basic simulation modeling. This is better than giving a very broad overview, without any "hands on" experience or with just inputting data into a "black-box" model. A student can get some idea about both the potentials and restrictions of simulation only by doing some kind of simulation **modeling**. This will also make simulation fun. As mentioned, my students have done interesting things, in the form of non-trivial simulation programs, after only a very short period of learning.

This is all connected with my basic goal of giving my students at least some idea about discrete event simulation, DES, in order to create informed buyers of simulation services. Some knowledge of simulation modeling is e.g. important for making reasonably realistic time assessments of the work of a simulation specialist. Knowledge of simulation is also important for being able to "sell" the idea of making a larger simulation project to top-management.

It should be stressed that my goal has not been to teach a specific simulation system as such. The simulation software used is only a tool to allow the students to produce simulation models. I have regarded it as fairly unlikely that the students will continue to work with the specific software over a long time in the future. It is more important to have the students open to several types of simulation software, both GUI and text based. In this regard my students may differ from a computer science major, who in the future might be the specialist from whom business graduate would buy more advanced simulation modeling work.

## 4 WHY SIMULATION AT A BUSINESS SCHOOL?

I have often been asked, especially by people from other disciplines: "Why do you teach discrete event simulation, DES, at a business school? Why are you not content by just simulating financial flows in a spreadsheet?" The answer is that I have found DES, implying dynamic stochastic simulation, to be of great importance in a business curriculum for several reasons.

First, I have had DES replace, or at least complement, many analytical/optimization parts of Operations Research or Management Science methods, such as queuing theory, inventory theory, PERT/CPM and parts of Decision Analysis. My students have liked this, since this has implied a greater focus on solving problems, and fewer methods to be learnt and forgotten.

Furthermore, DES has provided my students with a better understanding of the physical processes in a firm. DES is one way of giving business students an introductory understanding of some problems in the areas of production economics, material handling, inventory management, etc. Hence, in an introductory course on production economics for business students, I had DES play an important role. Closely related to this is the demonstration of the connection between the physical activities and the consequential financial flows. For this kind of simulation, a general-purpose simulation system is of greater interest than a system focused entirely on manufacturing.

Stochastic simulation is required to handle the uncertainty that is the core of financial theory. We can just think of how we answer the following questions: How much will we sell next year: 100,000 units for certain or 80,000 - 120,000 units? When will this customer pay: Within 30 days for certain or with 80 percent probability within 60 days? What will the $/€ ratio be a year from now: 1.30 for certain or between 1.0 and 1.6? In all cases, the last answer, indicating uncertainty, seems more reasonable.

If all future payments could be forecast with certainty, all corporate debt would be as safe as government bonds. There would then not be any need for different types of financial instruments and hence no need for financial theory. Against this background, it seems strange that most simulation of the future financial position of a corporation, e.g. cash forecasts, is done using deterministic simulation, without any uncertainty, by ordinary spreadsheets. Instead most financial simulation **should** be stochastic.

Finally, there is a need for **dynamic** simulation in the form of DES, allowing us to follow each major payment, regardless of when it takes place. This can be illustrated by Figures 1 and 2 of a cash forecast of a small corporation.

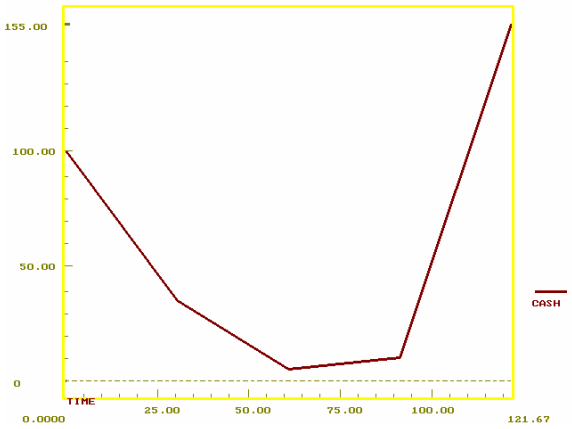


Figure 1: End of the Month Cash Graph

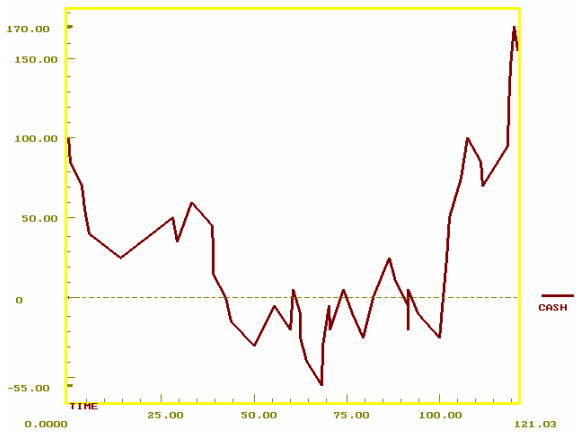


Figure 2: Dynamic Cash Graph

These diagrams have been produced by a simple GPSS simulation program for cash forecasting, presented as program PRO50 at <www.webgpss.com> and in Ståhl (2003b). The program deals with an importer who buys and sells certain machines. It pays the foreign producer in cash directly for each unit, but provides the customers with credit. Orders arrive according to an exponential distribution, while customers' payment times vary according to an Erlang distribution. Our students can write this type of program after ten hours of study.

We see that the two graphs give completely different impressions. From Figure 1 it appears that there would be enough cash for the corporation and hence not any liquidity problems. The financial problems, with negative cash several times, are clearly seen in Figure 2, where we can follow payment by payment. In Figure 1 these problems are not perceived at all, since by chance there is a cash surplus at each end of the month. This illustrates the need for a dynamic, discrete-events approach to corporate financial planning. As discussed in Ståhl (1993), the need for this type of dynamic simulation for cash flow forecasts is especially large, when a couple of hundred large payments constitute more than half of the payments of the company. This is true for many smaller corporations in areas such as mechanical engineering and construction.

Among other important applications of DES that are of interest in a business school I should mention simulation based costing, forecasts of diffusion of new computer models and lap-top based sales models (see also Section 5.3). I have also recently used DES to a large extent in a Decision Analysis course (Ståhl 2005). DES was used for doing difficult probability calculations, not only faster, but also with greater chance of a correct answer, as exemplified by the Car-Trip Example (Barton 2002). DES was also used for solving the bidding example of Samuelson and Bazerman (1985). Students tended to analytically, and erroneously, calculate the optimal bid to be in the interval of $50 to $75. Using DES, with many runs for each bid, it could be clearly seen that a bid of 0 is optimal.

## 5 TYPES OF COURSES

### 5. 1 Short Course Module

I have in many courses taught a short course module of 2 – 4 hours. After two hours my students modeled "Joe's barbershop" (see Section 5.5.2, Figure 4), producing full queuing statistics, including also histograms (Born and Ståhl 2004). By varying the parameters of uncertainty, the students studied the effect of uncertainty on queuing behavior. After four hours they modeled "Boris vodka shop", presented in section 6.2. This program produces surprising and revealing statistics on queuing behavior (Ståhl 2000).

### 5. 2 Intermediate Course Module

My mostly frequently run course module has been one of 10 – 15 hours. In this, the students built, as the main part of the examination, a simulation model of a small furniture company, based on a real case study (Ståhl 1996). This model involves decisions on production, inventory and pricing and produces profit/loss accounts and balance sheets as well as graphs of the development of cash. The students made several runs to insure both good profitability and a low probability of bankruptcy. This exercise has proved useful to the students e.g. in later courses in accounting and corporate finance.

### 5. 3 Full Course

In the full semester course of 30 – 40 hours, usually with the first 10 -15 hours almost the same as in the intermediate course module of the preceding section, I also made the students work through the whole simulation process as regards some concrete problem, usually in business. In this way, the student actively learnt the whole process, from formulating the question to be answered, delimiting the problem system, outlining the model graphically, gathering data, coding the program, verifying, validating and documenting, running the program a sufficient number of times, doing a statistical analysis for drawing significant conclusions, and presenting the results in a form suitable for a potential user, with a focus on implementation.

The students first made a reasonably valid simulation model of the present set-up. They gathered data (on items like arrival and service times and on waiting lines) from the real system and then compared the output data (e.g. on waiting lines) from the tentative model with real data to validate the model. If not validated, the model had to be adjusted. Finally, the students provided and tested a suggestion for an improvement of the system. This project determined the main part of the grade of the course.

An important aspect of this full course is hence that less time was spent on learning simulation software and more on the other aspects of the simulation process, men-

tioned above. The best way to learn all these aspects is through a simulation project carried out in business. Although the main part of the lessons in class might have been spent on learning simulation software, the main part of the student's total time for the course was spent on these other aspects, mainly in form of "learning by doing" in the project, but also by specific reading and by discussions with me.

I have in such courses had good experience of students, in groups of two or three, doing project work in different Swedish corporation. Many of the project programs have had continued use in the corporations. The work on the project, of e.g. three man weeks, has in many cases led to extended work in the form of a Master's thesis and/or in several cases landed the student a job in the project company. It should be recognized that in many companies the art of DES is little known and many medium-sized company have no expertise at all in this area. Hence, a reasonably good student could already after only 20 classroom hours do something useful in a company that no one in the company could do and hence be regarded by people in the company almost as a specialist. It was also advantageous that the simulation projects dealt with very down-to-earth problems.

I have guided some 500+ student projects. They have dealt with a great variety of subjects, for example in banking, telecommunications and retailing. Quite a few projects have involved "sales support simulation models", where the simulation model was run on a laptop, interactively with a client, in order to help to determine e.g. the optimal configuration of model for an Ericsson corporate switch board. Other examples of interesting projects are: Simulation based costing for a flexible manufacturing system at an IBM plant; The Stockholm Stock Exchange electronic system; Test of exits from the planned 2004 Olympic stadium in Stockholm; A logistics system for trucks on European roads; Inventories of cash and foreign currencies in banks; A local energy system based on biomass; Forecasting the diffusion process for new computers.

### 5. 4 Executive Education

During many years I was teaching a course on simulation in the executive program of the SSE. First the course was sold as a completely separate course for interested executives; later it became part of the Executive MBA program of the SSE. All the participants were working in a company, which sponsored them. The course has been two, or later three, days fulltime, with the lectures covering roughly the same material as the intermediate course discussed in section 5.2. One difference to the course given to ordinary university students is that the executive course was given on consecutive days, while the student course extended over several weeks. While the students could do exercises and reading as homework between the lectures, the executives would do the exercises in class and the lectures would proceed at a somewhat slower pace. The executives were also more negative to learning "strange software system details" than ordinary students.

The course was examined in the form of the participants developing a simulation model with relevance to the company in which they worked. This project was very much focused on producing a quick-and-dirty prototype simulation model which could possibly be the starting point of a more refined model, later to be developed with the aid of a true simulation expert, e.g. a computer scientist (Ståhl 2002). The main focus was hence on making the executives informed buyers of simulation services and on understanding of how one could get the results of the simulation model implemented.

This issue of implementation is a most important one and, although dealt with also in the full courses for student (in section 5.3), it was of particular importance in the executive classes. The main causes of failure of simulation lies in the implementation phase, namely in the fact that the ultimate decision maker does not trust the simulation results enough to dare to make decisions on the basis of these results. A main reason for the lack of such trust is that the decision maker does not understand the model.

Figure 3 below brings out the main aspects of the model implementation process. This process really deals with two models: the mental model of the decision maker and the simulation model produced by the model builder

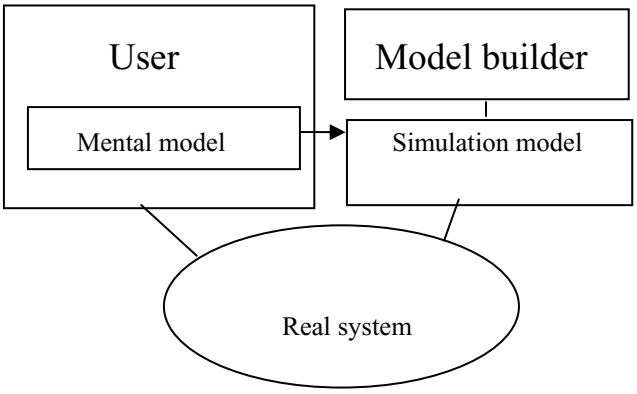


Figure 3: View of Implementation

The user is the buyer of the simulation model, i.e. the "problem owner", with special "domain" knowledge about the actual situation. The model builder, with special knowledge about simulation technique, but less knowledge about the problem situation than the user, can be an outside consultant or a staff officer, working as an inside consultant. The mental model of the user is important, since every person responsible for decisions regarding a system will in her head have a model or picture of the system, where certain simplifications have been made, implying that some factors in the real system are stressed. The computer model is like-

wise a simplification of the real system, where possibly other factors are included.

In order to have the simulation model accepted by the user, there must be a correspondence between the simplifications in the user's mental model and the simplifying assumptions of the simulation model. Secondly, the user must be **confident** that this correspondence really is at hand. Even if a correspondence exists between the two models, this is not sufficient for implementation, if the user does not perceive this correspondence, but fears that the simulation model might be at odds with the mental model. A prerequisite for action is hence that the user understands the model and its main assumptions and feels comfortable with them.

This implementation issue, although touched upon in the other courses, was of special importance in the executive courses, since the participants should all work on a simulation model to be implemented, at least as a first prototype, in the company they worked in. They could hence by their project work get to understand the roles of both the buyer of the simulation model and the provider of it.

## 5. 5 Simulation for High School Students

### 5.5.1 In Sweden

My activities regarding simulation in high schools grew out the executive classes described in the preceding section. The son of one of the participants in 1993 approached me to get help with a simulation study of his school cafeteria as his senior year project work, which all Swedish students do during their last year of high school in Sweden. Two years later the SSE was approached by the newly established KK-foundation, focused on spreading knowledge about IT among young Swedes, to see if we had developed any software that could be used in Swedish high schools.

As discussed below, we had at the SSE developed micro-GPSS as a streamlined, easy-to-learn version of GPSS. With the school cafeteria project work in mind, I thought that it would be in line with the KK-foundation's goal to make my GPSS system available to Swedish high school students to help them use simulation for their senior year project. There were a great set of possible project areas for high school students. Many Swedish high school kids work during summer vacations and could hence do projects on hospitals, shops, gas stations, etc. 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 a modern simulation system.

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 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.

In contrast to the case in Germany, discussed in Section 5.5.2, there has not been any central decision in Sweden 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, the system was given dialogs for the input of block operands, with understandable operand descriptions, an extensive Help-system, a number of tutorial lessons and around 100 program examples.

Furthermore, in order to introduce the students to 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 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 could progress fairly far. For example, after four hours students were able to write the program of "Boris vodka shop", presented below in section 6.2. We have also had one-day seminars with a number of teachers, in order to prepare the teachers to run similar short introductory courses.

### 5.5.2 In Germany

I have also taken part in a similar program for teaching simulation to high school students in Germany. Though the main work by far has been carried out by Dr. H. Herper, I have aided in the development of course material and also given a few lectures to teachers of simulation in the state of Sachsen-Anhalt at their annual meeting in Magdeburg,

In Sachsen-Anhalt, high school students in their senior year choose between optional courses from different areas of computer science. One of these courses is *Introduction to Modeling and Simulation*. For this course some 32 class-room hours are available. The students will in this course carry out a simulation project, similar to that described above in Section 5.3, but usually of a smaller size. As simulation software the students use WinGPSS, a system with a GUI and IDE developed in Magdeburg directly for Windows, but with the same core as WebGPSS, namely micro-GPSS. Program text is, just as in WebGPSS, built by the use of block symbols and block dialogs for the input of operands. This program text is then run by the micro-GPSS engine, producing tables and graphs (Herper and Ståhl 1999, 2003).

An important aspect of WinGPSS 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 order to visualize the movements of the transactions

through the system, but also at the same time to support the validation of the model, WinGPSS has a simple system for visualizing these movements integrated into its block diagram. 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. Figure 4 illustrates this dynamic visualization system for Joe's barbershop.

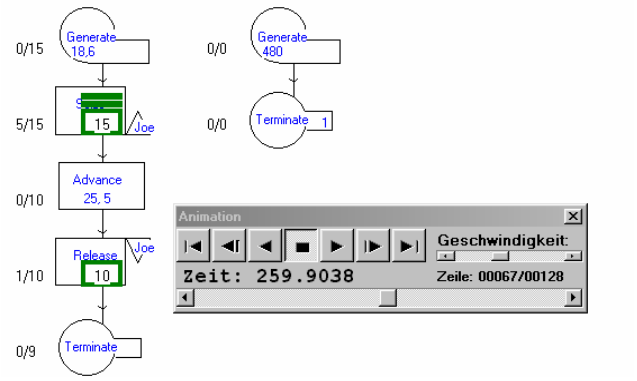


Figure 4: WinGPSS Visualization of Joe's Barbershop

## 6 CHOICE OF SOFTWARE

### 6.1 General Considerations

I shall in this section explain why all my teaching of simulation has been done using some form of GPSS software. This can be divided into two questions. Why did I start with GPSS and why have I stuck to GPSS?

The first question is the simpler one to answer. In 1976 when I started to teach simulation, the alternatives available were either a GPL, i.e. a General Programming Language, like FORTRAN or Algol 60, or a GPL based simulation package, like Simula. Due to the limited time available and the lack of programming knowledge among my business students, these alternatives could be ruled out.

To explain why I have stuck with GPSS, and, in fact, developed and simplified it in quite a substantial way, I need to provide some background thoughts, first on what I regard more generally as suitable characteristics of a simulation system for business students.

It is desirable that input can be made using a Graphical Users Interface, where the student from a menu of symbols can choose the building blocks of the program. The student should then for each building block be able to open a dialog for inputting the operands of this block. This dialog should reveal the main syntax of the block operands.

The system must provide most necessary statistics automatically, since the novice does not know what kind of statistics is of interest. It must be easy to read and understand the output. The system should, on the other hand, not provide a lot of advanced output that the novice would find confusing, but it should provide a single easy-to-read program listing, which is clear and compact and allows for short comments. This listing is also essential for making it easy for the teacher to correct and mark the student programs. The system should also automatically complement the program listing with a program logic diagram to make it easy for students to study, discuss and document the logic of a program.

To encourage replications, it should be very easy to make replications of the runs by just one command, easily available in the GUI. It is also desirable that the system can automatically carry out a statistical analysis of these repeated runs, e.g. of confidence levels. It should be easy to create functions based on empirical data and also easy to change these functions as new observations are made.

It is furthermore for a learning system of paramount importance that the system should minimize the risk of the student making logical errors. Students should not run into surprises and unexpected errors due to not having learnt the full system. The system should have an extensive error trapping system with as clear error codes as possible. An easy-to-learn system for debugging and program verification is also desirable.

### 6.2 Why BBS and not AOS?

When seeing that I have stuck to GPSS, many have asked why I have not instead used a more "modern" software system, in particular one with built in animation. To answer this question I think it is suitable to distinguish between two main types of simulation software, Block Based Systems, BBS, and Animation Oriented Systems, AOS.

GPSS is, just like Arena/SIMAN and AweSim/SLAM, a BBS, while systems like e.g. WITNESS, ProModel and Simul8 can represent the AOS. In both types of system the model consists of temporary entities, like customers, being served by permanent service stations, like a barber. There is a difference between the two types of systems as regards with which type of entities you start the detailed modeling. In the AOS you usually start placing the permanent servers, while you in the BBS start with the temporary entities, in particular by deciding on how and when they come into the system. The difference of the greatest importance is, however, that in the AOS, with mostly compulsory animation, each permanent server is in principle only represented once, since it in the animation work space, representing e.g. the factory floor, must be in only **one** place. In the BBS, with optional animation, a permanent server can be represented in **many** different places, since we here follow the temporary entities and, if different entities use the same machine, this usage of the machine can take place in different parts of the program.

We can exemplify the two types of systems with the following simple example, Boris vodka shop.

"At a store, run by Boris and Naina, customers arrive at rate of 7 $\pm$ 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 was 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 WSC during several years to solve the problem. BBS vendors solved it in 5 minutes; AOS vendors required more than 30 minutes. For details see Ståhl (2002).

The reason for the difference can be explained by the simple logic of the GPSS block diagram in Figure 5 compared to the more complex diagram in Figure 6 representing an AOS in a more general way.

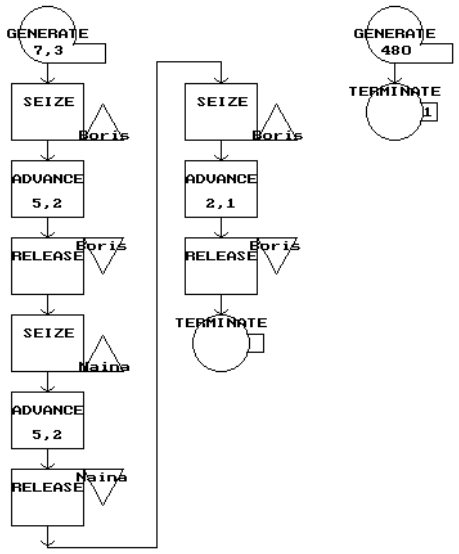


Figure 5: GPSS Block Diagram of Boris Vodka Shop

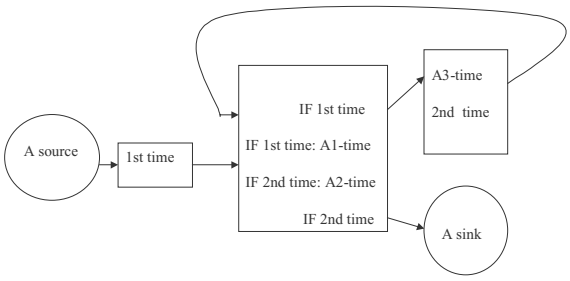


Figure 6: General AOS System for Boris Vodka Shop

The reason for the difference in complexity is that an AOS requires Boris to be located at **one** spot and the program must then for each customer keep track of whether he comes to Boris for the first or the second time. In the BBS, Boris is "seized" in two different places.

There are certain situations in which the AOS are very strong with regard to conveying the "gist" of the simulation. This refers mainly to manufacturing situations, where the animation of all processes can be carried out in proportional time. Proportional time implies that the time compression ratio (time-in-reality/time-on-the-screen) is **constant**, during at least a substantial part of the simulation. If, for example, one process takes 5 minutes in reality and 5 seconds on the screen, then a process taking 30 seconds in reality should in the case of proportional time take 0.5 seconds on the screen.

For many other animations, e.g. of service processes, one must "fake" the times in order to make a "pretty" simulation. An example is the classic barbershop, where a haircut in reality takes 20 minutes, while it takes 10 seconds to move from the waiting chair to the barber's chair. If a haircut in the animation takes 6 seconds, i.e. we have a time compression ratio of 1200/6=200, then moving between the chairs should take only 10/200=0.05 seconds on the screen. This is too short a time for the human eye to recognize as anything else but a sudden jump.

In situations where one can make a nice animation of physical processes with the same time compression ratio for all processes, an AOS has a great value in conveying important information. However, in situations where one is forced to choose between the three evils of either running a boring animation for a long time without much happening, distorting reality by having different time compression ratios or having the animation very jumpy, one might do just as well without animation.

Animation might furthermore not be very informative, when the simulation mainly deals with "invisible processes" such as message flows, cash flows and other accounting features, e.g. costs and profits. One must then choose symbols that often do not lead to much better model understanding than one could obtain by presenting graphs and text. This is often just as easily done in a BBS.

Since many problems of interest for business students have strongly different times for movement and have invisible processes, animation has been of much less interest for my teaching than for courses for manufacturing students, dealing e.g. with the movement of products in a workshop. Hence, I have preferred to stick to a BBS.

### 6.3 Development of Micro-GPSS and WebGPSS

I shall next explain why I have stuck with GPSS rather than move to Arena, which is used by many other teachers of simulation. I have studied Arena and some text books on it and it is an impressive system. However, for the very

specific purposes of my courses I regarded it as too complicated. As mentioned, I did not expect my business students to continue doing simulation in the system taught in the course. I just needed a tool with which they could do an interesting simulation project in a company. Continuing to simplify GPSS hence appeared as a better alternative.

I have personally not found it problematic that the first version of GPSS appeared already in 1961. WebGPSS has kept the original **goal** of G. Gordon of 1961, namely that people not used to programming should be able to produce simple simulation models, e.g. prototypes that can help solve basic decision problems. Furthermore, we have for 11 blocks kept the old GPSS words, namely GENERATE, TERMINATE, SEIZE, RELEASE, ADVANCE, ENTER, LEAVE, DEPART, SPLIT, ASSEMBLE and TABULATE (Ståhl 2001). We have for these block types also kept the original block symbols. In general, the use of block diagrams of the GPSS type with clearly distinguishable symbols, often in pairs with mirror-pictures of each other, has shown to have great pedagogical advantages and has also facilitated the students' documentation.

There is, however, a very great difference between WebGPSS of 2007 and the GPSS of the 1960s, and even that of GPSS/H or GPSSWorld. The most noticeable difference between WebGPSS and traditional GPSS is that, while programs for all other GPSS versions are constructed completely by using a text editor, WebGPSS programs are made by a GUI. By pointing and clicking on block symbols in a symbol menu, shown in Figure 7 below, a block diagram is constructed. This point-and-click method allows a faster build-up than the drag-and-drop method used by some other simulation systems.

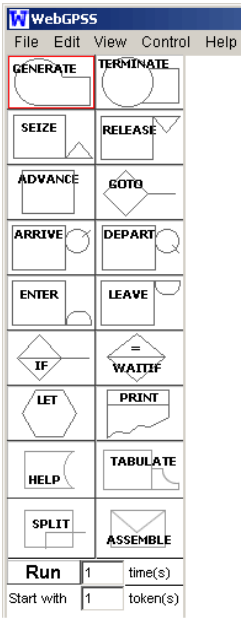


Figure 7: Block Symbol Menu of WebGPSS

By next double-clicking on the symbols in this block diagram, one can open an operands dialog for this block, where the operands are input in textboxes and fields with text describing the main syntax of the block.

The other main difference between the other GPSS versions such as GPSS V, GPSS/H or GPSS World is of course the much lower number of block types. While the number of block types in other systems has constantly been increased, with GPSS/H now having more than 70 such types, we have on the contrary continued to simplify the language; from having 22 block types in 1990, we can now with the same effect be content with only 16 block types of WebGPSS. (Ståhl 2003). As an example LET does the same job as ASSIGN, BLET, INITIAL, LET, PRIORITY, SAVEVALUE and SELECT of GPSS/H. It should be stressed that the limitation to just very few blocks does not make WebGPSS significantly less powerful than traditional GPSS. In fact, our original goal of making it possible to rewrite all of the 27 well-known case studies of Schriber's Red book (Schriber 1974) into WebGPSS with no longer code has been reached and 99 percent of all programs in GPSS text books have been rewritten for WebGPSS with roughly the same amount of code (Ståhl 2003).

As regards other improvements over traditional GPSS we can mention simplified collection of statistics, by e.g. SEIZE Joe,Q in WebGPSS compared to QUEUE JoeQ, SEIZE Joe, DEPART JoeQ in GPSS/H. WebGPSS has a straight IF logic, with IF Q$Joe=4,BYE instead of TEST NE Q$Joe,4, BYE in GPSS/H. WebGPSS has GOTO BYE and GOTO BYE,0.15 instead of TRANSFER ,BYE and TRANSFER .15,,BYE in GPSS/H. Many other features of traditional GPSS, which have been heavily criticized, like e.g. the logical error caused by having a GENERATE block prior to a SEIZE block, have also been eliminated. Many of the simplifications have been based on feed-back from students, who have found certain features difficult. In particular the error message system has thus been constantly updated, to catch errors early and provide clear error messages. Students have reported any unclear message.

A great improvement, inspired by my executives being reluctant to calculate cumulative probabilities manually, is the new system for defining empirical random functions. For each value, one just inputs the number of observations. WebGPSS first automatically transfers this into percentages and then cumulative probabilities. In contrast to other systems, the student does not have to do a lot of recalculations if one e.g. finds one more observation.

We have hence, since 1979, gradually developed our own easy-to-learn easy-to-use GPSS system, so that it fulfills the requirements discussed above in Section 6.1. The system was originally called micro-GPSS, but, after being put on the web, its name is WebGPSS. It is available both on the Web, at <www.webgpss.com> and as a free standing, at present more modern, version on a CD. It has for the students been of importance that they on the CD in-

cluded with their textbooks have had a software version large enough to carry out projects of a generally sufficient size (Ståhl 2003b).

Based on comparing the effect of teaching WebGPSS with that of some other systems and on very positive student evaluations, we have dared to make the claim: "In courses that devote a maximum of twenty classroom hours to simulation, WebGPSS is certain to surpass every other simulation system with regard to the power and the relevance of the programs that business students can write at the end of the course" (Ståhl 2003b).

## REFERENCES


Barton, R. 2002. Using simulation to teach probability: Panel. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C-H. Chen, J.L. Snowdon, and J.M. Charnes, 1816-1817. New York: ACM.

Born, R. and I. Ståhl. 2004. WebGPSS: The First Two Hours of Simulation Education**.** In *Proceedings of the 2004 Winter Simulation Conference,* ed. R.G. Ingalls, M.D. Rossetti, J.S. Smith, and B.A. Peters, 2066-2074. New York: ACM.

Herper, H. and I. Ståhl. 1999. Micro-GPSS on the Web and for Windows - A Tool for Introduction to Simulation in High Schools. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, 298-306. New York: ACM.

Herper, H. and I. Ståhl. 2003. Modeling and simulation in High School education – Two European examples. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P.J. Sanchez, D. Ferrin, and D.J. Morrice, 1973 – 1981. New York: ACM.

Samuelson, W. and M. Bazerman. 1985. Negotiating under the winner's curse. In *Research in experimental economics,* Vol. 3, ed. V. Smith, 105-137. Greenwich, CT: JAI Press.

Schriber, T.J. 1974. *Simulation Using GPSS*. New York: Wiley.

Ståhl, I. 1993. Discrete-event simulation for corporate financial planning. In *Proceedings of the 1993 Winter Simulation Conference,* ed. G.W. Evans, M. Mollaghasemi, E.C. Russell, and W.E. Biles, 1264-1169. New York: A.C.M.

Ståhl, I. 1996. Simulation of the business operations of a small furniture company. In *Modelling and simulation ESM96,* ed. A. Javor, A. Lehmann, and I. Molnar, 374-376. Budapest: SCS.

Ståhl, I. 2000. How should we teach simulation? In *Proceedings of the 2000 Winter Simulation Conference,* ed. J. Joines, R. Barton, K. Kang, and P. Fishwick, 1602-1612. New York: ACM.

Ståhl, I. 2001. GPSS – 40 years of development? In *Proceedings of the 2001Winter Simulation Conference,* ed. B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, 577-585. New York: ACM.

Ståhl, I. 2002. Simulation Prototyping. In *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yücesan, C-H. Chen, J.L. Snowdon, and J.M. Charnes, 572-579. New York: ACM.

Ståhl, I. 2003. From 44 to 31 to 28 to 22 and now to 18 – Less becomes more in GPSS. In *Simulation und Visualisierung* 2003, ed. T. Schulze, S. Schlechtweg, and V. Hinz, 465-478. Magdeburg: SCS.

Ståhl, I. 2003b. *Simulation Made Simple with WebGPSS – A Tutorial*. Stockholm: SSE.

Ståhl, I. 2005. Using discrete event simulation in the teaching of decision analysis. In *Proceedings of the 2005 Winter Simulation Conference,* ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2280-2289, New York: ACM.


## AUTHOR BIOGRAPHY


**INGOLF STÅHL** is Professor Emeritus at the Stockholm School of Economics, Stockholm, of a chair in Computer Based Applications of Economic Theory. He has taught GPSS for almost thirty years at universities in Sweden, Norway, Germany, Latvia and the USA. Based on this experience, he has led the development of the micro-GPSS and WebGPSS educational simulation systems. His email address is <ingolf.stahl@hhs.se>. His WebGPSS system is at <http://www.webgpss.com/>.