*Proceedings of the 1996 Winter Simulation Conference*
*ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain*

# TEACHING THE FUNDAMENTALS OF SIMULATION IN A VERY SHORT TIME

Ingolf Ståhl

Stockholm School of Economics
Box 6501
S-113 83 Stockholm, Sweden

## ABSTRACT

This paper informs about the development of the micro-GPSS system on the basis of feed-back from some 5000 students during a period of two decades, providing a stream-lined simulation system which within ten class room hours of teaching make students prepared to write fairly advanced simulation programs of practical interest.

## 1. INTRODUCTION

Simulation has proved to be a very powerful tool not only in engineering but also in business administration. Against this background it is surprising that most business schools and quite a few engineering schools do not give their students any substantial amount of teaching in simulation. The main reason for this it that student time is an increasingly scarce resource in a very crowded curriculum. Often only a handful of hours can be spent on simulation. This, in turn, often leads to the teaching of simulation being limited to a very broad overview, without allowing for any "hands on" experience. Most teachers with this approach believe that the learning of a simulation language or package would take too much time. If any computer is used, it is limited to the input of data into an already existing model.

This approach has several draw backs compared to allowing the students to actually work with a simple simulation package on concrete problems. In this latter way the student can actively learn the whole process of doing simulation for a problem, from delimiting the actual problem, formulating the question to be answered by simulation, gathering data, outlining the simulation program graphically, coding the program, verifying, validating and documenting the program, 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 the implementation aspects. In particular, it is important to allow the students to be involved in some kind of modelling, since the modelling aspect, in my view, is of fundamental importance for understanding simulation.

To allow for such an approach to be feasible in a short time, a very simple simulation system is required. Let us already at this stage exemplify the time span that we are discussing, namely ten class room hours and all together at most a week of student time including reading and project work. (We are hence not discussing a full semester course, involving perhaps all together a month of student time.) The question is then what can be accomplished in such a very short time.

I shall in this paper discuss what we have done at two universities in Sweden: namely one business school, the Stockholm School of Economics (SSE), and one technical university, the Chalmer's Institute of Technology (CIT).

At SSE all 300 students will in their third year study simulation, having 10 class room hours of instruction in the micro-GPSS system to be discussed below, spending roughly 10 hours on reading and preparing shorter exercises and spending about 20 hours on a larger GPSS program. This course of ten hours, called *Integrative Computer Applications,* has the particular aim of integrating elements of Operations Research, Corporate Finance, Accounting and Statistics. The focus is on a student project, roughly 100 blocks in size, regarding some kind of cash flow forecast and involving decisions on the purchasing, production, inventory and borrowing policy of a smaller corporation. An example of such a project is presented in detail in Ståhl (1996a).

At the department of Transport Technology at CIT around one hundred students annually do roughly the same thing, also using micro-GPSS, with the main difference that the project work is more directly focused on transportation technology.

## 2. WHY GPSS?

The question then naturally arises why we in the teaching discussed above use a version of such an "old-fashioned" language as GPSS. I shall divide this discussion into two parts: a. Why we use a GPSS-based language, to be discussed in this section and b. why we use our own special version of GPSS, micro-GPSS, to be discussed in the next section. The discussion in this section hence applies to all versions of GPSS, be it GPSS/H, GPSS/PC, GPSS/R, HGPSS, GPSS/V, GPSS/VI, SIMPC, GRAMOS-GPSS, micro-GPSS, etc.

As I see it, the following factors are favorable for GPSS when it comes to choosing a simulation system for an educational task like that one specified above.

1. **Ease of learning**. The very short time available requires a special simulation language. We can hence directly rule out the alternative of doing simulation in a General Programming Language, like C++ or Pascal, or a simulation language that is built on a GPL, like Simula, SLX or YANSL. This point is further strengthened by the poor background in programming, in particular as regards the business students

2. **The GPSS World View**, involving temporary components, like customers, which move through the system to be served by various permanent servers provides a very problem oriented approach which has proved to have great benefits both from the modelling point of view and of making GPSS fun to learn.

3. The provision of a great amount of **automatic output**, like block statistics, server statistics, waiting line statistics, tables, etc. This automatic provision is very important since the novice will generally not know what kind of output is relevant. The study of this standard output is an important part of the learning process. Furthermore one avoids having to spend much of the initial time on print commands.

4. **The block diagrams** make it easy for students to study, discuss and document the logic of a program. The teaching can also be more visually oriented. One can start by looking at the main structure of model before looking at the details of the program syntax. The availability of standardized block symbols is a great advantage. Compared to other simulation systems with block symbols, GPSS has the advantage of having many blocks coming in pairs, with one block being the mirror picture of the block doing the "opposite" thing, e.g. SEIZE and RELEASE. The GPSS symbols are easy to read and generally clearly distinct from each other.

5. GPSS is a **"fun" language**. Simple things can be done in a simple manner. (As discussed below micro-GPSS focuses even more on this.) One important factor in this respect is the unity of program and experiment conditions which is an advantage for the novice, but perhaps a disadvantage for the professional user. Among other special features that are regarded by some as disadvantages, but are advantageous when it comes allowing the student to do interesting things at a very early stage, one can mention the concept of a facility, providing service to only one customer at a time and hence not needing a capacity definition.

6. GPSS allows for very **compact programs**. With only a few blocks one can produce quite interesting programs. The advantage of GPSS in this respect is self-evident when comparing with languages like SIMSCRIPT, MODSIM and SIMULA, but it appears that also compared with languages like SIMAN and SLAM, GPSS will generally lead to shorter programs. The shorter code has the advantage among other things of making the programs easier to "overview".

7. GPSS has, due to its long history, the great advantage that there are **many program examples and much literature** available, in fact over a dozen text books, (Ståhl 1993b and Schriber 1995), including what many teachers regard as the best text book ever on simulation, Schriber's famous Red Book from 1974. The availability of well-documented programs is important for the novice, when doing a project work, since he is then more likely to find a text book example to build upon.

8. There are reasons to believe that GPSS with its more than 35 years' tradition is **better debugged** than other simulation languages. The constructors of one GPSS version can use other GPSS versions for checking correctness. Furthermore, there has been a much higher degree of scientific scrutiny of GPSS than of other simulation languages. The internal mechanism of GPSS is better documented and therefore better understood. The user is therefore less likely to run into strange and unwanted effects in certain situations. (Schriber and Brunner 1995 and Ståhl 1996b).

9. Moderately limited versions of GPSS (allowing for programs of up to 150 blocks) are available at very **low cost** ($20 - $40) in several GPSS versions (e.g. GPSS/H, and micro-GPSS). The student therefore has the additional incentive of working in a simulation system that he knows he can run on his own PC and continue to use also after the simulation course is over.

10. GPSS is a **good starting point** for later moving to many other simulation systems The existence of GPSS similar packages in object oriented languages like Simula (e.g. GPSSS and DEMOS) makes a switch to some object oriented languages simpler. Since SIMAN contains many basic structures similar to that of GPSS, several of our GPSS students at CIT later learnt SIMAN much faster than their fellow students not having taken the GPSS course.

11. Finally it should be stressed that GPSS is a **General Purpose** Simulation System and that for simulation problems, not only focused on manufacturing, but e.g. combining physical and financial streams, and where one wants to investigate the full implications of uncertainty by many runs, GPSS is often more suitable than systems that are very good for animation of production systems, but otherwise fairly weak.

## 3. WHY MICRO-GPSS?

The development of the micro-GPSS system has been a gradual process, starting already in 1978. The first versions of micro-GPSS were virtually proper subsets of GPSS V, with the main difference being the internal mechanism (Ståhl 1996b). Having taught 200 - 300 students a year for almost 20 years, in courses with a decreasing amount of time allotted to them, but with the ambition of still providing the students with the same amount of simulation knowledge, we have, however, step by step changed the original system into the present one, simplifying it to make it easier to learn and use.

This development of micro-GPSS can be seen as a reaction against the development of increased complexity of the GPSS language over time. One measure of this increased complexity is the ever higher number of block types. GPSS I (1961) had some 20, GPSS III (1965) around 40, GPSS/360 (1967) 44, GPSS V (1970) 48, GPSS/H (version 1, 1977) 58 and GPSS/H (version 2, 1989) 62 block types. This development towards more and more block types has in turn lead to increased redundancy in the way that several blocks carry out the same job. This complexity has also made GPSS increasingly more difficult to learn and has lead to students learning a steadily decreasing share of the full language and therefore making more logical errors. Although many of the developers themselves (Gordon 1979 and Henriksen 1983 and 1985) have raised critique against this development, it has yet continued due to the developers' interest in compatibility with older versions of GPSS. Such compatibility is of interest to old users who have already done a lot of programming in earlier versions of the language, but it is of no interest to the novice who desires to have as easy a learning process as possible with regard to the goal of being able to do a certain kind of simulation.

During the process of development, we have spent much time reflecting on the design. Questions have arisen not only with regard to how to cut out a pure subset of GPSS/360, GPSS V or GPSS/H, but also whether or not we should change this or that feature of GPSS. I have in the process tried to formulate the general principles that have guided us in this work (Ståhl 1993a). The main ones are the following, divided into three groups, dealing with A. ease of learning, B. ease of use and C. safe programming.

### A. Ease of learning

A1. The main aim is that the students after a very short time shall be able to do simulation projects of practical value. First, as mentioned above, we have only a very limited time of teaching at our disposal.. This has lead to our aim that one should be able to learn most of micro-GPSS in less than ten hours so that one after that can do some student projects of practical interest. Some time should also be left for the issues of collection and evaluation of input data, the main principles of experimental design, statistical analysis of the output, aspects of implementation, etc., i.e. issues that are left out in many courses where all time is spent on the mechanics of a difficult-to-learn simulation language.

A2. As we want the students to focus on modelling (and experimentation), and not on syntax detail, the language should be such that one in the course does not have to learn a new block type every time that a new and different thing shall be done. It is from a pedagogical point of view often preferable that the new aspects can be handled using already known block types, even if the program thereby becomes a few blocks longer.

A3. When students frequently make the same mistake, one must always consider the alternative of changing the syntax instead of forcing them to learn strange features that, for example, might depend on hardware limitations of computers in the 60's or on pure mistakes made by developers in the early years of GPSS. Since we, in contrast to GPSS/H, are not bound to compatibility with earlier GPSS versions, we have in this way over the 20 years of teaching allowed our 5000 students to change the language syntax. Some examples are given in section 5 below.

A4. We have in micro-GPSS been very keen to keep and increase the "fun aspect" of GPSS. If the simulation system is fun to learn from the beginning, this will in itself provide strong incentives for further learning. It is here important that the language provides a possibility for the students to do interesting things after only a very short period of learning. Preferably the students should already after one or two class room hours be able to write some non-trivial simulation programs, i.e. students should be able to do simple things in a very simple fashion. One should not sacrifice the ease of introduction for the sake of having sophisticated features for the advanced user. One should in particular restrict what appears as unnecessary details, e.g. avoid commas that are not absolutely essential. (See section 5 below). It is important to “encourage users to forge ahead and experiment rather than present barriers that lead to discouragement” (Banks 1995). It should be mentioned that this positive aspect of learning has given the micro-GPSS courses increasingly favorable student ratings, much higher than for comparable computer courses.

A5. We have also developed micro-GPSS with the aim of facilitating the teaching in computer labs as well as facilitating self-studies in front of the computer, instead of learning by lectures in ordinary class rooms or text book studies. In this connection it is important that micro-GPSS allows several programs to be run in a stream, with both program listing and output presented one screen at a time, without the student having to leave the GPSS system.

A6. micro-GPSS was developed with the aim of being so simple that it could be completely covered in a pedagogical manner, with many examples in a book of reasonable size, say a maximum of 400 pp, and hence with a moderate price. This was accomplished with Ståhl (1990). The student should not have any need for a difficult-to-read manual in order to find features not covered in the text book or in class.

A7. For micro-GPSS it has been very important not to presuppose any pre-knowledge of programming. This is one difference between how micro-GPSS and GPSS/H are positioned. As mentioned in Henriksen (1985), the focus of GPSS/H is on persons with considerable programming experience. Most of our business students do not have any programming knowledge except possibly some very rudimentary knowledge of (Visual) BASIC. Even for my technical students at CIT, who some year earlier had programming in Pascal, this presupposition has appeared to be advantageous.

## B. Ease of use

B1. It should be very easy to do the coding of the programs. Since there are great variations between students as regards their preferences for input systems, one should provide a choice between conventional coding using an editor and coding by choices in a menu, either GUI based or text based. For easy coding by an editor, micro-GPSS has in contrast to other GPSS versions a **completely** free format so that students do not have to worry about the fact that certain words have to start in a certain column; every line can start in column 1. In contrast to GPSS/H, no distinction is made between upper-case and lower-case letters. As regards coding by use of a menu in an interactive dialogue mode, there are two systems for micro-GPSS, our own GPSSMENU and GPSSGUI for DOS and GPSSEDIT for Windows (Nywall 1992). Finally, these input systems aim at making syntax rules self-evident or easily available, so that the student does not have to refer to a text book..

B2. It must be easy to read the program listing and the output. micro-GPSS does not provide a lot of advanced output that the novice does not know how to read and would find confusing. In contrast to GPSS/H the output always fits into an 80 column screen. It should also be easy to read the extended program listing provided by the system. micro-GPSS provides a neat and easy-to-read program listing, with an automatic line-up of operators and operands, independent of the appearance of the original code

B3. Since it in teaching is important to focus on block diagrams, it is first of all essential that every block type has a corresponding block symbol. This is the case of micro-GPSS, but not of GPSS/H. Furthermore it is in this context important to have facilities for the automatic generation of block diagrams, in micro-GPSS by GPSSDIA, as well as systems for coding by clicking in a menu of block symbols (see B1).

B4. Since we also want to focus on experimentation, it is essential that it is very easy to make replications of the runs and also to make a statistical analysis of these repeated runs. Running a program 20 times with different random numbers is done by SIMULATE 20.

## C. Safe programming

C1. Closely related to ease of learning, but also to ease of use, is the principle of **safe programming.** We want to minimize the risk of logical errors, i.e. that that program produces unwanted and erroneous output. In this way a great amount of student time spent on debugging can be saved. If the simulation language is made as safe as possible with regard to logical errors, this also reduces the need for an extensive debugging system which, in turn, requires a lot of student time to learn. To secure safe programming we want to stress the "Lead us not into temptation" principle, implying that the simulation language should not be excessively permissive, allowing constructions that with a significant probability lead to logical errors. It is better that an unsuitable construction leads to a syntax error message and execution stops right away than have it lead to a difficult-to-find logical error. micro-GPSS is therefore much less permissive than ordinary GPSS; e.g. servers must have symbolic names, parameter usage is restricted, etc.

C2. It is also important that the simulation language has an extensive error trapping system with as clear error codes as possible. This error system is very developed in micro-GPSS. All of my 5000 students have been asked to report on errors with no or an unclear error message. This has lead to constant improvements.

C3. Closely connected with the idea of C1 is the aim that students should not run into surprises and unexpected logical errors due to not having learnt the full language. It is of special importance that the language does not have any reserved words, in particular reserved words that lead to strange logical errors. We must avoid problems such as, e.g. that SEIZE XID1 and SEIZE XJD1 in GPSS/H lead to completely different results. Unless the student knows that XID1 is a reserved word with special meaning in GPSS/H, he can make a serious logical error.

## 4. SELECTION OF MICRO-GPSS BLOCKS

On the basis of the principles discussed above we have developed a GPSS version with much simpler syntax. For example, instead of the 62 block types of GPSS/H we have arrived at only 22. These 22 block types can be seen in figure 1 below. The question is then how we arrived at these 22 block types. This is discussed in detail in Ståhl (1992b), but the main ideas are as follows:

With few block types there is less to learn. It is also in line with the "modelling" principle of point A2 above. Speaking, to the contrary, for the inclusion of more block types is the idea that short and compact programs in general are preferable to longer programs as regards solving a specific task. With a new block type, one can sometimes with one block do what would require several blocks to do with the earlier existing block types. This is one factor that has lead to the continued increase in the size of the GPSS language mentioned above.

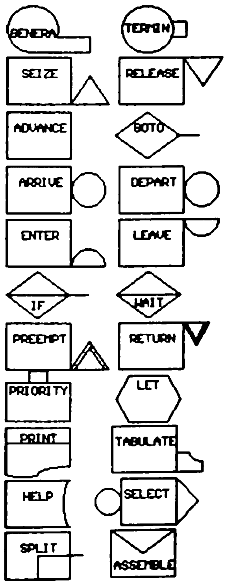

Figure 1: Menu of micro-GPSS blocks

A critical issue is then when to adopt a new block type, if this leads to shorter programs. Our general principle was that we started with a quite small "compulsory" core set of block types. We would extend this set with another block type only if this adoption would lead to significantly shorter programs as regards applications that appear to have a frequent usage. When establishing what kind of applications have a fairly frequent usage, the only alternative available to us was to look at the examples in existing GPSS text books. If applications were not dealt with in these text books they were not regarded to be in frequent usage. Furthermore, we did not adopt a new block type, like LOOP of "old" GPSS, if the use of one block of such a type could easily be replaced with two blocks of already adopted types.

Another consideration speaking for limiting the number of block types was our interest in easy-to-use programs for coding via a dialogue menu, with, for example, block symbols on which to click. With 62 block types like in GPSS/H, the menu itself would hardly fit readably into one computer screen. The 22 block symbols of micro-GPSS fit nicely into the left fourth of the screen as seen in figure 1, produced by GPSSGUI. A final consideration favoring the limitation of the number of blocks is the possibility for the micro-GPSS user to create his own new block types from one or several existing block types (Ståhl 1993c).

The set of 22 block types has proved to be quite powerful and micro-GPSS programs are in general not longer than corresponding programs in other GPSS dialects. The reason for this is that the increase in the number of blocks due to fewer block types is compensated by some new micro-GPSS features. The most important one is that for collecting statistics in a queue, e.g. in front of a facility, we can just add ',Q'. Thus the micro-GPSS block SEIZE SAL,Q does the same thing as the following blocks in ordinary GPSS: QUEUE SALQ, SEIZE SAL, DEPART SALQ.

We have hence been able to rewrite 99 percent of the programs in leading GPSS text books with almost the same amount of code (at most 20 percent difference). For example, for the 29 programs in Schriber's famous "red book" from 1974 the average number of blocks used is virtually the same (18.6 in Standard GPSS and 18.8 in micro-GPSS).

## 5. SOME OTHER MICRO-GPSS FEATURES

It should first be mentioned that of the 22 block types, five do not exist in "old GPSS", namely ARRIVE, GOTO, IF, LET and WAITIF. ARRIVE, the natural counterpart of DEPART, replaces QUEUE and has (almost) the same syntax; GOTO replaces TRANSFER, but with a simpler syntax. Instead of TRANSFER ,BYE (with BYE as B operand) and TRANSFER .4,,BYE (with BYE as C operand), micro-GPSS uses GOTO BYE and GOTO BYE,.4, with BYE always as A operand.

LET replaces ASSIGN and SAVEVALUE (and BLET in GPSS/H). IF replaces TEST and GATE blocks **with** an address, while WAITIF replaces TEST and GATE blocks **without** an address. In contrast to TEST, IF works with a "straight logic", namely that we proceed to the address in the C operand, if the stated condition is **true** (not false as in "old" GPSS). In micro-GPSS we write IF P1<0,BYE instead of TEST GE P1,0,BYE as in "old" GPSS versions, when we go to BYE if P1<0.

Among the many improvements as regards the block syntax , one can here mention that the PRINT statement is much more versatile in micro-GPSS, allowing us e.g. to write PRINT 'PROFITS ',X$PROF. Matrix handling as well as reading and writing the contents of the matrix to or from a file is also handled in a very compact and simple manner in micro-GPSS. Furthermore, the production of an output graph can be handled by a single block. Standard Numerical Attributes can be given easy-to-remember names. The SELECT block has a more understandable syntax.

Also as regards control statements, micro-GPSS requires a lot less learning. Instead of the 34 control statements of GPSS/H, micro-GPSS has only 13, where CAPACITY replaces STORAGE for the definition of capacity and SEEDS replaces RMULT, since it changes the seed, not the multiplier. SIMULATE has, as mentioned, an A operand determining the number of runs. Micro-GPSS also has facilities for automatic statistical analysis (e.g. of Student's *t*-distribution) and for optimization. It has several built-in statistical functions. Micro-GPSS also allows for animation by a very simple interface to Proof Animation (Ståhl 1992a). and for simple tracing and step by step animation through the block diagram.

The whole of micro-GPSS syntax is presented in Ståhl (1990). A shorter introduction to micro-GPSS is provided in Ståhl (1995) for which there is a tutorial diskette with restricted versions of the micro-GPSS system and GPSSDIA as well as 49 program examples. This contains roughly the material covered in the ten hour course presented above.

Due to these and several other simplifications, micro-GPSS is **much** easier to learn than traditional GPSS. We have thus with micro-GPSS been able to cover the same material in 10 hours that required 22 hours when using "old" GPSS. Micro-GPSS has also in tests compared favorably as regards learning time with other systems such as SIMAN and WITNESS.

## 6. TWO PROGRAM EXAMPLES

To give some idea of the simplicity of micro-GPSS we shall present two program examples. The first program, which will produce block statistics, facility statistics, queue statistics and a table on waiting times so that we, for example, can say what percentage of the 30 patients had to wait more than two hours for the doctor. I have not found any other simulation language which with so little code can do so much. The program is run 5 times with different random number streams. This is a program that students can write already after two hours of teaching.

```
simulate    5
qtable      doc,0,10,20
generate    18,6
seize       doc,q
advance     25,5
release     doc
terminate   1
start       30
end
```

Program 1

The next program will illustrate a problem that appears to be much more difficult to program in many "modern" simulation systems.

"At a small store customers arrive at a rate of every 4 to 10 minutes (7 $\pm$ 3 minutes; assume a rectangular distribution for all time data.) In the store there are two people working, Boris and Naina. Customers first go to Boris and choose the goods and find out how much they have to pay. This takes between 3 and 7 minutes. Next they go to Naina to pay for the goods and obtain a receipt. This also takes between 3 and 7 minutes. Finally, they return to Boris to pick up their goods after presenting the receipt, which is then stamped. This takes between 1 and 3 minutes. There is one waiting line in front of Boris and one in front of Naina. Customers returning to Boris to pick up the goods have to start at the end of this line again.

The program should be written so that times spent by customers in the store can be easily measured. How many customers will spend more than 15, 20, 25 minutes, etc. in the store? Assume that the store is closed after eight hours and that the mentioned statistics refer to customers having left the store at this closing time. Calculate also by repeated runs whether there is any significant risk (e.g. happening in one out of ten cases) that a customer has to spend more than an hour in the store."

```
simulate  20
qtable    store,0,5,20
generate  7,3
arrive    store
seize     boris
advance   5,2
release   boris
seize     naina
advance   5,2
release   naina
seize     boris
advance   2,1
release   boris
depart    store
terminate
generate  480
terminate 1
start     1
end
```

Program 2

This problem has been solved in micro-GPSS in ten minutes by students who have studied micro-GPSS for three hours. I have at earlier WSCs asked sales representatives of various modern animation based simulation packages to solve this problem and it has taken them over half-an-hour to do so in their system. The difference is that in GPSS you only need a simple sequence of three SEIZE-ADVANCE-RELEASE, while these other systems require Boris to be located at one spot and the program must then for each customer keep track of whether she comes to Boris for the first or the second time, something which is difficult, obviously not only for the novice.

## 7. HOW TO PROCEED FROM MICRO-GPSS

As discussed above, students can proceed fairly far in simulation after ten class room hours of micro-GPSS. The most clear proof that our business students after these studies can do fairly interesting simulation is given by the project work mentioned in section 1 above. The students here write a GPSS program involving on average one hundred blocks regarding the operations of a smaller corporation. An example of such a project with the micro-GPSS program is presented in Ståhl (1996a).

The question is then what long term benefits the students can have of this. Even if the students never do any more simulation there are clearly several valuable lessons to be learnt. One clear benefit is that simulation illuminates the connection between the actual physical processes and its counterparts in the accounting system. This has a pedagogical merit and has appreciably increased our business students' understanding of the relationship between accounting procedures and the physical and financial flows in a company. Another benefit is that the students in the course get a clearer understanding of concepts such as pseudo-random numbers, exponential, normal and Erlang distributions, experimental design, etc. than they could get by just reading about it. They also become better informed buyers of simulation services, knowing that simulation in certain situations can be a powerful, but yet fairly inexpensive, tool.

Many of our students have, however, proceeded in simulation. We have at SSE an optional longer course in simulation, focused on a larger project done in a Swedish corporation. In this course, the remaining parts of micro-GPSS are covered as well as Proof Animation with the micro-GPSS interface to this. There is also a focus on verification, validation, documentation and implementation. Quite a few of these student projects have been continued on a consultant basis or as a Master's thesis. It should be mentioned that all students have stayed with micro-GPSS. There has obviously not been any need to migrate to GPSS/H, in spite of the fact that we supply a program GPHM which can translate almost any micro-GPSS program into GPSS/H code.

Most of these projects have been fairly small in size, 200 - 300 blocks. The longest micro-GPSS program is 500 blocks. This gives an idea of how we see micro-GPSS positioned. Since micro-GPSS, as well as GPSS/H, in contrast to e.g. HGPSS (Claeys *et al.*, 1995), does not allow any hierarchical structure, we see micro-GPSS positioned as not only a pure teaching device, but also as suitable for rapid prototyping in simulation, for smaller models in line with Woolsey's "Quick and dirty" approach. When it comes to larger models than the 300 - 500 blocks discussed here, object oriented simulation languages like SIMULA or MODSIM III are clearly preferable. However, before committing oneself to a major effort in such a language a rapid prototype in GPSS can be a suitable starting point, as in cases like that of the Swedish ice-breaking operations (Jennergren *et al.* 1995).

## 8. FURTHER INFORMATION ON MICRO-GPSS

Micro-GPSS is available on the PC and Macintosh as well as on SUN and VAX workstations.

A demonstration diskette with scaled down versions of the micro-GPSS interpreter and GPSSDIA and with 50 program examples is available free of charge, together with a short tutorial, from the author. Professor R. Born (1995) has produced a very pedagogical computer slide show to accompany this tutorial.

More information on micro-GPSS and its use in the discussed ten hour course at the SSE is also available on the web at WWW.HHS.SE/5e/7784.htm.

## REFERENCES

Banks, J. 1995. Semantics of Simulation Software. In *OR/MS Today*, December 1995, pp. 38 - 40.

Born, Richard. 1995. *Computer Slide Presentation to Accompany Ingolf Ståhl's Simulation Made Simple with micro-GPSS.* Northern Illinois University. DeKalb.

Claeys, F. , H. Vangheluwe and G.C. Vansteenkiste. 1995. HGPSS: A Hierarchical Extension to GPSS. In *Proceedings of the 1995 European Simulation Multiconference*, ed. M. Snorek, M. Sujansky and A. Verbraek. Prague.

Gordon. G. 1979. The Design of the GPSS Language in Adams, R.N. and Dagramici, A. (eds) *Current Issues in Simulation*. Wiley, N.Y.

Henriksen, J.O. 1983. *State-of-the-Art GPSS.* Paper presented at the 1983 Summer Computer Simulation Conference, 1983, Vancouver, B.C, Canada.

Henriksen, J.O. 1985. *The Development of GPSS/85.* Paper presented at the 18th Annual Simulation Symposium, Tampa, Florida.

Jennergren, L.P., L. Lundh, U. Törnqvist and S. Wandel. 1995. Icebreaking Operations in the Northern Baltic. In Miser, H.J. (ed) *Handbook of Systems Analysis: Cases.* Wiley, N.Y.

Nywall, J. 1992. *GPSSEDIT Manual.* Mimeo. Karlstad College, Karlstad, Sweden.

Schriber, T. J. 1974. *Simulation Using GPSS*, Wiley, N.Y.

Schriber, T. J. 1995. Perspectives on Using GPSS. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K.Kang, W. Lilegdon and D. Goldsman, 110-117. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Schriber, T. J. and D. T. Brunner. 1995. Inside Simulation Software: How It works and Why It Matters. In *Proceedings of the 1995 Winter Simulation Conference*, ed. C. Alexopoulos, K.Kang, W. Lilegdon and D. Goldsman, 451-456. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Ståhl, I. 1990. *Introduction to Simulation With GPSS: On the PC, Macintosh and VAX,* Prentice Hall International, Hemel Hempstead, U.K., 1990.

Ståhl, I. 1992a, Animation with micro-GPSS and Proof In *Visualisierung und Präsentation von Modellen und Resultaten der Simulation,* ASIM, Heft. Nr. 31.

Ståhl, I. 1992b, *Principles behind the Design of an Easy-to-Learn Simulation Language: Implications for the Selection of Block Types.* EFI Research Paper 6485.

Ståhl, I. 1993a, Principles Behind the Design of an Easy-to-Learn Simulation Language. In Roberts, R.S. and S. Monroe (eds) *Simulation Applications in Business Management and MIS.* SCS, San Diego.

Ståhl, I. 1993b. GPSS Will Prevail - Some Reasons for the Resilience of the GPSS Simulation Ideas. In *GPSS-Users' Group Europe - Gruendungveranstaltung,* ASIM Heft nr. 36, Magdeburg.

Ståhl, I. 1993c. Recent Developments of micro-GPSS. In *GPSS-Users' Group Europe - Gruendundgveranstaltung*, ASIM Heft nr. 36, Magdeburg.

Ståhl, I. 1995. *Simulation Made Simple with micro-GPSS: A Short Tutorial with Seven Lessons,* Stockholm School of Economics, Stockholm.

Ståhl, I. 1996, Simulation of the Business Operations of a Small Furniture Company. In Javor, A., A. Lehman and I. Molnar (eds) *Modelling and Simulation,. ESM '96.* Budapest

Ståhl, I. 1996b. *Steps Towards a Better Internal GPSS Mechanism.* In this volume.

## AUTHOR BIOGRAPHY

**INGOLF STÅHL** is 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 research project on inter-active simulation at the International Institute for Applied Systems Analysis, Vienna, 1979-1982. He has taught GPSS for twenty years at universities and colleges in Sweden and the USA. He has on the basis of this experience led the development of the micro-GPSS system. He is also consultant in simulation to Swedish banks and industry.