Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

GPSS 50 YEARS OLD, BUT STILL YOUNG

Ingolf Ståhl

Stockholm School of Economics
Box 6501
SE- 11383 Stockholm, SWEDEN

James O. Henriksen

Wolverine Software
3131 Mount Vernon Avenue
Alexandria, VA 22305-2640, U.S.A.

Richard G. Born

OM&IS Department
Northern Illinois University
DeKalb, IL 60115, U.S.A.

Henry Herper

Institute for Simulation and Graphics
Otto von Guericke University
Magdeburg 39102, GERMANY

ABSTRACT

In 2011, GPSS, the General Purpose Simulation System, celebrates its 50th anniversary. At the 2001 Winter Simulation Conference there were two papers dealing with the 40th anniversary of GPSS. With these papers available on the Web, this paper will concentrate on the developments of GPSS after 2001. There are still three systems with the GPSS name that are sold, supported and improved: GPSS/H, GPSS World and the educational aGPSS systems family. There has also been a substantial development of the successor of GPSS/H, SLX. Finally, Proof Animation, which is closely connected to some of the GPSS systems, has been substantially improved during the last decade.

1 INTRODUCTION

GPSS, the General Purpose Simulation System, developed by Geoffrey Gordon at IBM, was made available to the public in the fall of 1961. Although it was not the very first computer software system for discrete events simulation, with GPS (the General Simulation Program) by Tocher having been released around 1958, it very soon became the dominant simulation software. Two indications of this are that the very first Winter Simulation conference, held in 1967, was called "Conference on Applications of Simulation Using GPSS" and that in 1972 GPSS was ranked in the tenth position among what were then judged to be the world's thirteen most important programming languages of any type (Sammett 1972). The popularity of GPSS, in particular within education, was further enhanced by the appearance in 1974 of T. Schriber's famous Red book, *Simulation Using GPSS* (Schriber 1974). After IBM stopped supporting GPSS in 1975, many private firms, as well as universities, developed their own GPSS systems. Most of them had GPSS in their names, but many others, while being built on the basis of GPSS, had other names. For this reason, it seems suitable to talk about a "GPSS family" of software systems.

With the advent of many other simulation systems, often with built-in animation systems, the relative importance of GPSS later decreased. Many of the systems in the GPSS family also ceased to be supported after a few years. However, recent surveys show that GPSS is still among the ten most used simulation systems (Dias, Pereira, and Rodrigues 2007). Within education, especially in Europe, its relative position is even stronger. With a substantial development taking place within the "GPSS family", to be discussed in this paper, the importance of GPSS is likely to continue.

With regard to competing systems, it should be noted that, in contrast to many other systems, animation is *not* mandatory in any of the GPSS family systems, although recent GPSS systems (see Section 4) allow the inclusion of animation. In most animation-based simulation systems, each permanent server, like a lathe, is represented only once, since it in the animation workspace, representing e.g. the factory floor, must be in only one place. Complex logic is then needed e.g. to get a product to access the lathe twice. In GPSS a certain machine can, however, be accessed (e.g., by SEIZE LATHE blocks) in many different places in the program, without extra logic, often making modeling much easier (Ståhl 2007).

The fundamental advantages of GPSS are discussed at length in two papers presented at the 2001 WSC, in connection with the 40th anniversary of GPSS, namely *GPSS Turns 40: Selected Perspectives* (Schriber *et. al*. 2001), and *GPSS - 40 Years of development* (Ståhl 2001), both available on the WSC web at http://www.wintersim.org.

In these two papers the development of GPSS during its first 40 years, i.e., up to 2001, is very well covered. For this reason, this paper shall focus on the development that has taken place *after* 2001. In this introduction, we shall present only a few aspects of the early history of GPSS that were missing in the 2001 papers. First of all, we want to present a picture of Geoffrey Gordon (from the IBM archives).

Figure 1: Photo of Geoffrey Gordon with an early GPSS block diagram

Next we want to present some statements made by him on what he regarded as the purpose and strengths of his GPSS. The following quotation from an article written by Gordon on his thoughts on the development of GPSS in the 1960s (Gordon 1978, p. 195) are of interest for the rest of this paper, since they bring out Gordon's interest in a simple system to be used by non-programmers, as well as his enthusiasm for the block diagram.

"Engineers and analysts would be able to use the program themselves, even if they were not trained in programming. Using the program was to be like using a special purpose machine; avoiding programming terminology and practices. This potential elimination of an intermediary in the form of a programmer who has to interpret the system designer's ideas was considered a highly desirable characteristic...To many users, particularly those on large projects; the language was to become somewhat restrictive. The emphasis placed on simplicity of use, however, was a major factor in the rapid acceptance of the program...The block diagram enhanced the illusion that the user was not programming, but describing the system. It also proved to be a great asset in teaching the program, and encouraged what would now be described as top-down programming...Block diagrams were also an asset in improving understanding between the various people who needed to know about the system. Even people who do not have experience in GPSS programming but do know what the system does, or should do, are usually able to follow how the system has been represented, when shown the block diagram with a commentary by the person who constructed it...There were times when the block diagram describing the logic of a system design was being used as a document of transmittal between groups engaged in the development of the system."

Next we want to show in Figure 2 below how the development of GPSS, starting with Gordon's 1961 system, has proceeded by the way of later IBM versions, to the three main groups of systems for which support and development have continued, namely GPSS/H, GPSS World and the educational aGPSS systems family. We can see this as a kind of "genealogy" of the GPSS systems. To the right of the name of the system, we have the first year of introduction and in most cases the number of block types. The number of block types gives a very simple measure of the complexity of the system.

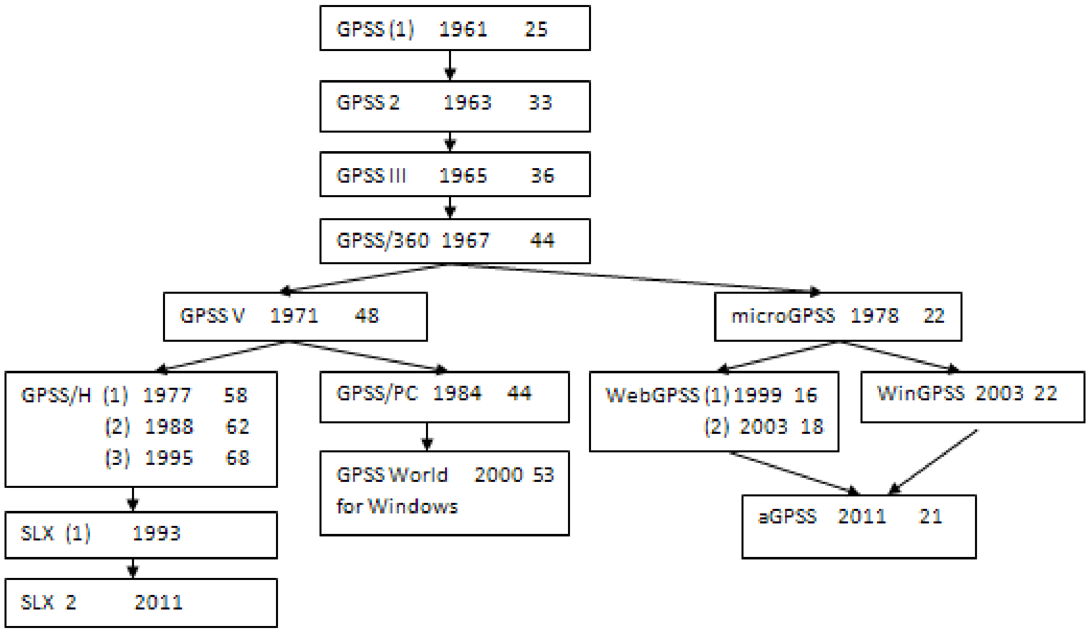

Figure 2: Genealogy of the GPSS systems

We see in Figure 2 that Gordon's original GPSS (1) was followed by GPSS 2, GPSS III and GPSS/360. As discussed at length in Ståhl (2001), GPSS 2 and GPSS (1) were quite similar, while GPSS III implied a big change from GPSS 2. GPSS/360 was, in turn, quite similar in basic structure to GPSS III. It should be stressed that GPSS/360 is the system described in the Red book (Schriber 1974). Since the primary goal of the first attempts in the late 70's to create a simple educational GPSS, later to be micro-GPSS, was that the system should be able to handle all of the 27 case studies in the Red book with roughly the same amount of code, the paths of the two commercial systems (GPSS/H and GPSS/PC) part from the path of the educational system already after GPSS/360. Both GPSS/H (Wolverine Software) and GPSS/PC (Minuteman Software) take the last IBM GPSS system, GPSS V, as the starting point. In Section 2 we shall discuss the developments in the GPSS/H, i.e. the Wolverine, path. In Section 3 we shall deal with the educational systems, today with the "family name" aGPSS.

As regards the GPSS/PC – GPSS World path, it shall suffice to note the following: while GPSS/PC is no longer supported, GPSS World for Windows, introduced in 2000, is still sold and supported. After 2001, when an Automatic Experiment Generator was added together with a Multiway ANOVA routine for analyzing user-designed experiments, new versions (the latest is 5.2.2) have mainly contained correction of bugs. No major development efforts are being made at present. The student version of GPSS World, can be downloaded free of cost at http://www.minutemansoftware.com. The use of GPSS World among students seems to be particularly strong in Germany and Russia. In the last few years several text books on GPSS World have appeared (Rothbauer 2003, Boyev 2004, Braznik 2006).

During the first 40 years of GPSS there were several GPSS similar systems, written in general programming languages like FORTRAN, Simula, APL, Pascal, etc. (Ståhl 2001). There has, to our knowledge, only been one new system of this type presented during the last ten years, namely Object GPSS on object Pascal, developed by A. Korolyov from Ukraine (anatoliygk@sti.lg.ua).

2 GPSS/H – WOLVERINE PATH

Wolverine Software's SLX product was initially developed as a follow-on to GPSS/H (Henriksen 1985 and 1993). While Wolverine continues to maintain and support GPSS/H, over the past ten years most of those efforts have taken the form of improvements to its operating environment, rather than architectural upgrades.

The major difference between SLX and GPSS/H is that GPSS/H is a very large, flat language, while SLX has a smaller number of constructs that are deeper. For example, GPSS/H has 7-8 blocks that can delay the flow of a transaction through a model. SLX has two such statements, "advance" for simple time delays, and "wait until" for arbitrarily complex state-based, time-based, and combined time- and state-based delays. From an architectural standpoint, GPSS/H is analogous to a large, shallow lake, while SLX is more like a swimming pool. While one can safely wade through the waters of GPSS/H, writing a large, complex model may require visiting many parts of the lake. To use SLX, one needs to know how to swim. For example, GPSS/H has servers and arrays, but lacks arrays of servers. SLX has server objects that can be aggregated into arrays and sets, and can be dynamically created and destroyed. Thus, to meaningfully use SLX, one must master the construction of objects and the use of arrays and sets and create and destroy verbs. Once these basic concepts have been mastered, one can combine them in unlimited combinations.

SLX features very powerful *extensibility* mechanisms that facilitate customization to meet users' needs. The developers of SLX initially viewed extensibility as a sort of "safety valve" mechanism to be used when all else failed. They wanted to provide a way for users to build their own statements in cases where no built-in statement existed that exactly matched a given requirement. The extensibility mechanisms have come to be very heavily utilized. Virtually all SLX users take advantage of them to some extent. Thus, extensibility mechanisms have evolved to be the tool of second resort in building SLX models, rather than being the tool of last resort. The tools of first resort are SLX's built-in features.

SLX was developed to be object-based rather than object-oriented. The reasons for taking this approach were (1) to keep the learning curve manageable and (2) to avoid trying to replicate all the capabilities of C++ and Java, ending up with a poor imitation. Over time, as the SLX user base became larger and more sophisticated, the demand for object-oriented features grew. In the 2009-2011 timeframe, a number of object-oriented features were incorporated into SLX. Most of these were inspired by C++ and Java, although the SLX counterparts to C++ and Java features are in general simpler and easier to use. SLX supports inheritance, Java-like interfaces, and other object-oriented features. Some of these features have turned out to be more powerful than was originally imagined.

Consider, for example, SLX sets. In SLX, sets are collections of objects. Prior to the existence of object-oriented features, sets were available in two forms. A set could contain either objects of a single class or objects of all classes (*universal* sets). A universal set might be used to represent a segment of a road in a transportation model. The road could contain car, truck and bus objects. One problem with such a set was that objects of unintended classes could be placed into the set. For example, a traffic light object could be placed in a road set. In object-oriented SLX, a third type of set was added, namely a set of an interface. An SLX interface (somewhat modeled on Java interfaces) is a set of component specifications (required variables and methods). Any class declared to implement an interface must supply each of the required components. One surprisingly useful construct is an empty interface, i.e., an interface that imposes no requirements. In a transportation model, one could define a "vehicle" interface, and declare that car, truck and bus classes implement this interface. Defining a road as a set of vehicles results in a set that can contain any mixture of car, truck, and bus objects, but no objects of any other class that is not declared to implement the vehicle interface. This provides very tight control over set membership in a way that is in between the previously available all-or-nothing-at all types of sets.

As was the case with extensibility mechanisms in the original SLX, the new object-oriented features of SLX have come to be far more heavily utilized than its developers ever anticipated. Thus software that originally evolved from GPSS, which was intended for use by non-programmers, now includes extremely sophisticated object-oriented features, while respecting many key aspects of its distinguished ancestor.

The student versions of GPSS/ and SLX, as well as that of Proof Animation (see Section 4), can be downloaded free of cost at http://www.wolverinesoftware.com.

3 EDUCATIONAL GPSS

While the path to the left under GPSS/360 in Figure 2 deals with commercial systems, the path to the right deals with systems developed with an educational purpose, often based on feed-back from students. In the mid-70's, when we (Ståhl and assistants) used GPSS/360 for teaching business students at the Stockholm School of Economics, we soon discovered that many aspects of this system caused the students problems, like unnecessary commas and unusual terms compared to programming languages like BASIC. Early on, we hence replaced e.g. TRANSFER and TEST with GOTO and IF. This was done in our text based system called **micro-GPSS**, running under DOS on PCs. With, as mentioned, Schriber's Red book as a target, we found out that we only needed 22 block types, to cover all of the 27 case studies in this book. These 22 block types were also found to be sufficient for the many student projects (of roughly one month's effort) that the students worked with in our courses (Ståhl 2003).

Micro-GPSS was first only used at different universities and colleges in Sweden, but in 1990 a text book *Introduction to Simulation with GPSS - on the PC, Macintosh and VAX* (Ståhl 1990), based on micro-GPSS, was published by Prentice Hall and many students in many countries then ordered the software. In the 1990s teaching of micro-GPSS started in the USA, by R. Born at Northern Illinois University, and in Germany by H. Herper, at Otto-von-Guericke-Universität. A smaller tutorial was developed (Ståhl 1996). This was later translated into German, with additional features (Ståhl and Herper 2003).

In 1999 the first tentative versions of two GUI-based systems, based on micro-GPSS, were presented, **WebGPSS**, with the client developed in Java and with the programs first developed on the client PC and then run on a remote server, and **WinGPSS**, developed in Delphi by H. Herper, A. Krüger and H. Schlifke, and running in Windows on student PCs. In both systems, blocks were chosen from a symbol menu leading to a block diagram. By clicking on a block here, a dialog with syntax explanations is opened to allow for the input of the operands of the block. The execution in both systems is carried out by the micro-GPSS "engine", which produces output files in ASCII format, which WebGPSS and WinGPSS then can turn into, e.g., graphs. These were the first GUI based GPSS versions that were available to be run on PCs. WebGPSS was made generally available on the Web, run on a Swedish server, and was used in several Swedish high schools, in particular for individual projects in the senior year.

WinGPSS is used in the German state of Sachsen-Anhalt, where high school students in their senior year choose between optional courses from different areas of computer science, one of which is *Introduction to Modeling and Simulation*. For this course some 32 classroom hours are available. The students in this course carry out simulation projects, using WinGPSS and sometimes also Proof Animation; see also Section 4 (Herper and Ståhl 2003).

The first version of WebGPSS of 1999 only allowed for 16 of the 22 block types of Micro-GPSS, but enough to handle the features in Ståhl 1996. The first version of WinGPSS allowed for all 22 blocks, but was more restricted in other aspects, e.g., as regards the control statements. The symbol menu of WinGPSS is seen to the very left in Figure 3, while the symbol menu of WebGPSS of 1999 is seen to the right of this WinGPSS menu.

After 2001, further developments were made in both WebGPSS and WinGPSS. The fact that WebGPSS ran on a server in Sweden, but was used by some students on far away continents led to problems. In 2004 we hence decided that we also had to have a stand-alone version of WebGPSS, first of all running under Windows. The server module was then also run on the user's computer. The prime focus during the years since then has been on Windows, but steps are being taken to also have the system run under Linux and the Macintosh. Because of this, we have considered the name **aGPSS** as more suitable. At present, a limited student version, but large enough for all programs in Born and Ståhl (2011), can be downloaded, free of cost, from the site http://www.webgpss.com.

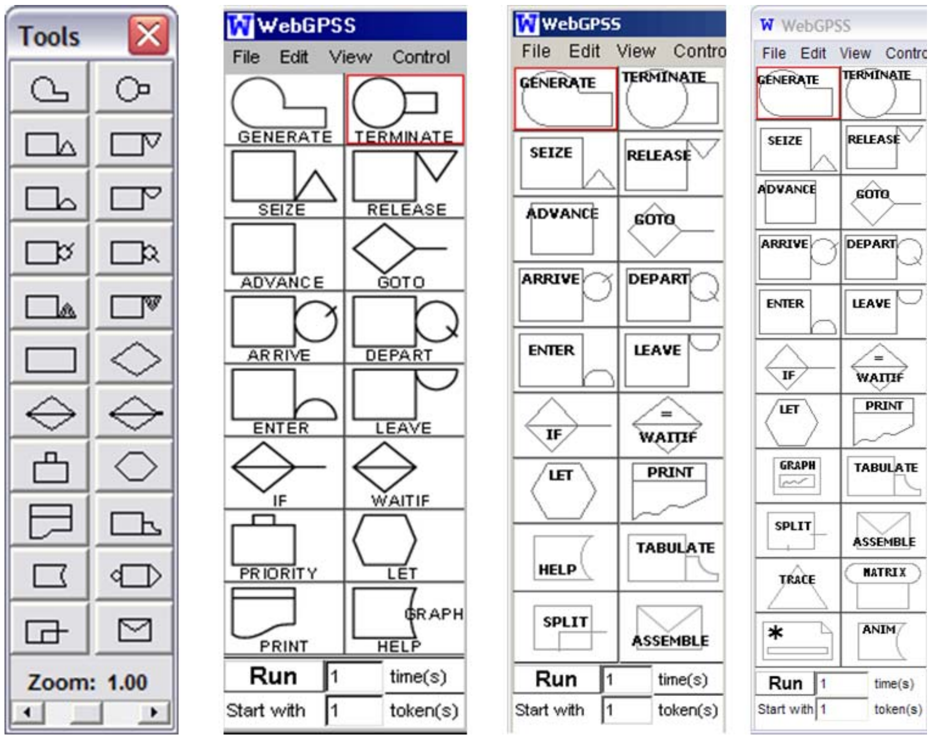

Figure 3: Symbol menus in WinGPSS and the three versions of WebGPSS/a GPSS

There have also been many other improvements of the WebGPSS/aGPSS system. In 2003, we extended the system so that it would have the same functionality as provided by the 22 blocks in micro-GPSS (Ståhl 2003). In Figure 3 we see the WebGPSS symbol menu of 2003 with a total of 18 blocks to the right of the original 1999 menu with 16 blocks. There are now three new blocks, TABULATE, SPLIT and ASSEMBLE. The reason that there are only 18 blocks (instead of 22) is first of all that the PRIORITY block has been dispensed with. The priority is now set by a stronger LET block, allowing also for an increase or decrease of priority. The functionality of the SELECT block is now also handled by the LET block. LET p$best=MIN,Q,1,6 conveys the idea that p$best is the shortest of 6 queues better than SELECT MIN 1,1,6,,Q of traditional GPSS. It should be noted that parameters are now given symbolic names. Finally, the PREEMPT and RETURN blocks are now part of the SEIZE and RELEASE blocks.

There has recently been one more change in the symbol menu, as shown by the menu to the very right in Figure 3. We have eliminated the HELP block. The HELP block was earlier used not only for HELP GRAPH, but also HELP TRACE and HELP MATIN. The word HELP has proved to be confusing to the students. We have now given GRAPH and TRACE their own symbols and instead of HELP MATIN we have MATRIX with its own symbol. We finally have kept the old HELP symbol in the lower right column to be used as a block, ANIM, for the interface between WebGPSS and Proof animation (see Section 4). Although there has hence been many changes in the block symbols over time, it can be noted that twelve of the blocks in the aGPSS menu of 2011 are the same as in GPSS and GPSS 2 of 1961-3, namely GENERATE, ADVANCE, SEIZE, RELEASE, ENTER, LEAVE, SPLIT, TABULATE, TERMINATE, ASSEMBLE, TRACE and PRINT.

Besides the changes in the block symbols, there have also been many other additions to the aGPSS system. We have as regards control statements added facilities for experiments and optimization, for warm up and for antithetic random numbers. aGPSS allows for running the model many times with an automatic calculation of the limits within which the universal average will lie with, for example, 95% probability. It also allows for a limited amount of optimization with a graphic representation of these confidence intervals for each of many alternatives. For the case of a comparison of two alternatives, it will determine if one alternative is better than the other one with 97.5% probability. New improved possibilities for printing and saving various output files in ASCII format have also been introduced.

One recently introduced improvement in the aGPSS system has been the possibility to place blocks in different columns. While WinGPSS has mainly used a drag-and-drop buildup of the block diagram, WebGPSS has used a point-and-click buildup, which has the advantage that it allows for a very quick buildup. The system would, except for GENERATE blocks, which are always placed at the top, place the block just below the last inserted block. The disadvantage was that all blocks in a segment were placed in one single column. The new system now allows the user to move the blocks to any desired position by the arrow keys. The information about the **relative** position of a block that is not to be placed just below the earlier block is in the computer code indicated by an extra part of the comment. Thus, e.g. the comment ! (Loc: -3,+1) of the block SEIZE fixer,Q in the model to the left in Figure 6 would place this block 3 steps up and 1 step to the right in comparison to the most recent block in the code, i.e. the first TERMINATE block. This methodology has the advantage that all information about the placement of the blocks can be in ASCII code, allowing for very compact programs.

It should finally be mentioned that a new collection of over 300 aGPSS programs has been compiled and that a new text book, focused on modeling business processes with aGPSS (Born and Ståhl 2011) has just been published. R. Born has produced a set of PowerPoint slides that accompanies this book and facilitates self-studies as well as teaching aGPSS. The GPHM system allowing for the translation of aGPSS programs into GPSS/H has been updated so that almost all programs in this book can be translated.

There have also been additions to the WinGPSS system. One is the introduction of simple block based animation. Students have during the teaching of GPSS experienced problems understanding how the simulation process works and how 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 now 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. With the new block diagram system of aGPSS in place, work is in progress on incorporating a similar system also into aGPSS, starting with a system based on Proof, described at the end of Section 4.

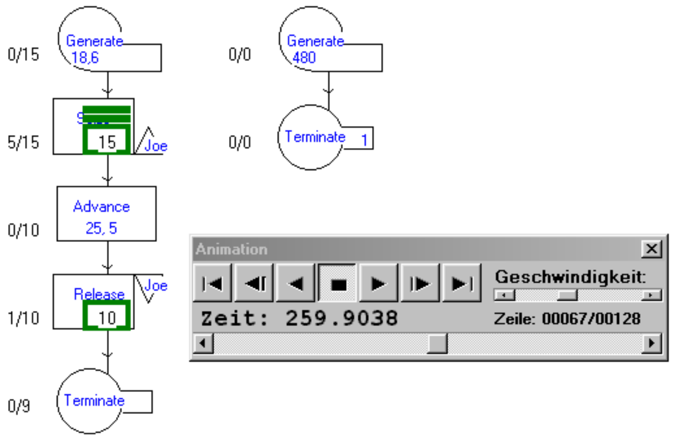

Figure 4: WinGPSS Visualization of Joe's Barbershop

4 ANIMATION WITH PROOF

Since the 40th anniversary of GPSS, Wolverine Software has introduced two new versions of its Proof Animation software family, Proof 5 (P5), a 2D animator, and Proof 3D (P3D). P5 was developed by re-hosting P4 under Microsoft's Direct3D. P5 uses a 3D environment with the Z-axis clamped at zero. The primary reasons for converting to Direct3D were (1) it allowed running animations in windows, and (2) DirectDraw, the Microsoft platform on which P4 was built was becoming obsolete. P4 required exclusive control of the full screen and was prone to "turf wars" with other software, fighting to maintain exclusive control. Switching one's focus back and forth between animations and underlying simulation programs was at best awkward. Under Direct3D, animations could run in windows, with seamless switching back and forth among other applications. The widespread availability of multi-core CPUs that has come about in recent years has been very beneficial to concurrent animation. On a dual core machine, Proof runs on one core, and a simulation application runs on the other.

The most recent member of the Proof family is P3D, a general-purpose, simulation-oriented animator. The most obvious contribution of P3D is that it extends P5 into the third dimension. However, in the development of P3D, a number of powerful generalizations were made to the architecture of Proof. For example, in P3D, objects can include sub-objects, and motion paths, whose scopes were always global under earlier versions of Proof, can be local to sub-objects. For example, a walking man can be constructed as a collection of arms, legs, a body, a neck, and a head. Arms and legs can easily be made to move along paths that are local to the body. All relative motion is performed on a top-down basis. For example, if the body moves, and legs move relative to the body, the trajectory of the legs with respect to the animation layout is the sum of two motions.

As mentioned above, Proof Animation (mainly P4) has during the last ten years been used together with WinGPSS in German high-schools. WinGPSS here uses the very simple interface that micro-GPSS has to Proof. WinGPSS has a HELP block that uses the following subroutines: Change, Color, Create, Destroy, and Move. Place, Space, Speed and Write for building up the special trace files are to be run by Proof. One example of the use of WinGPSS and Proof in the teaching of modeling in Germany can be illustrated by Figure 5. In the class room the students can study a very simple Lego railroad model, shown to the left in this figure. Their task is then to build an animation of this that is close to the Lego system. The Proof layout file might then look as in the right part of the figure.

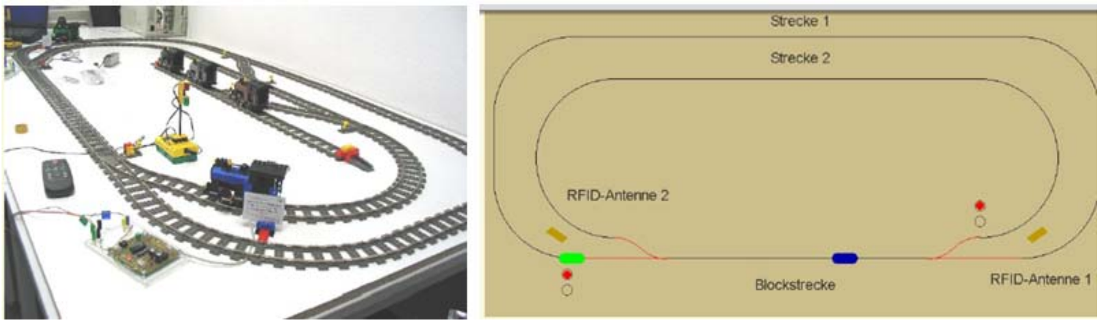

Figure 5: A Lego train circuit and its Proof .Lay file counter-part

As long as WebGPSS was run on the Web, sometimes from non-Windows systems, it seemed meaningless to produce long ATF-files. Hence, we have only recently found it suitable to have a WinGPSS similar interface also in WebGPSS/aGPSS. The only difference is that we now use the word ANIM instead of HELP.

Besides the more traditional user buildup of the program to produce the animation, we are also in the process of introducing a completely automatic buildup of .ATF files from any aGPSS program, although it has no ANIM block at all. The ideas were explained already in Ståhl (2000), but only now, when we have the new kind of block diagram with free location of blocks, has it appeared meaningful to implement this idea. To the left in Figure 6 we have the block diagram of the aGPSS model. This will automatically produce the paths of the Proof layout file to the right. It should be stressed that this kind of automatic animation is made possible by the fact that the layout files of Proof are ASCII files and can hence be easily constructed by another program.

It will also produce a special version of the GPSS code. In this extended program there will be commands for creating and destroying objects and placing the objects on the paths of the layout file. Thus, for example, **after** a GENERATE block with the address of a CLASS name, e.g. CIRCLE, the aGPSS processor, will insert three extra blocks: LET p$seg1=N$CIRCLE, ANIM CREATE,p$seg1,CIRCLE and ANIM PLACE,p$seg1,BLO1 and **before** a TERMINATE block, GPSS will insert two extra blocks, e.g., ANIM PLACE,p$seg1,BLO6 and ANIM DESTROY,p$num. Other blocks get similar extra blocks. Among these extra blocks there will sometimes be a block ADVANCE x$dum, which can give the class token, representing, e.g., p$seg1, a visible movement through the blocks. This very simple Proof program, which could be used as a very first introduction to animation, can then be improved by the student.

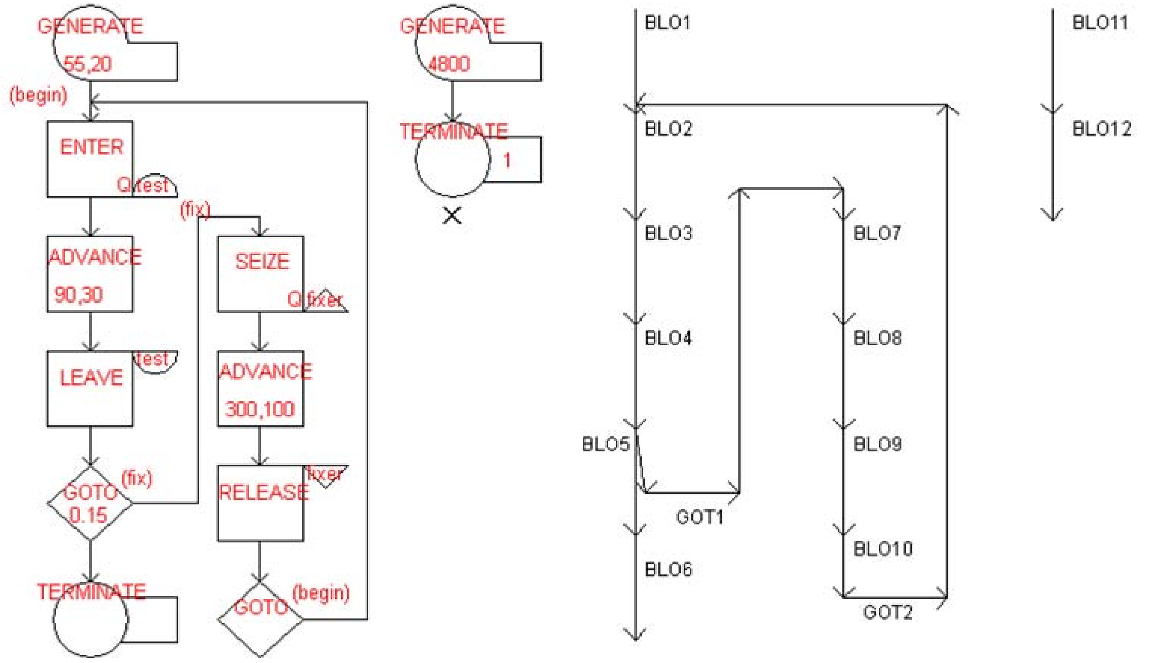

Figure 6: An aGPSS block diagram and its Proof path counterpart

5 SUMMARY AND CONCLUSIONS

In this paper we have given a very brief survey of the development of GPSS since its start in 1961, but with a strong focus on the last decade, which is not so well documented. From a technical development history point of view, it is interesting to note that out of the same software, GPSS/360 released in 1967, have grown two very different development paths. In both cases there has been a gradual development with continuous improvements. One path is focused on being able to solve any kind of problem, even the most complex ones and doing it in a very efficient manner. This has hence led to highly sophisticated software for sophisticated commercial clients. The other path is focused on making it very easy for students, in the last classes in high-school or in the first years in college, to be able to learn so much of a simulation system that they can carry out a simulation project that is of interest to people in the company involved.

We have noted that there has been much development activity along both of these paths during the last decade and that these activities continue. It is hence not unreasonable to predict that the GPSS family of software systems will be around for many more years.

ACKNOWLEDGEMENT

We thank George Rivas and the IBM archives for providing the photo of G. Gordon in Figure 1.

REFERENCES

Born, R., and I. Ståhl. 2011. *Modeling Business Processes with a General Purpose Simulation System – Part 1.* Gothenburg: Beliber.

Boyev, V. D. 2004. *Modeling of systems using GPSS WORLD. Textbook.* [In Russian] St. Peterburg: BChV-Peterburg.

Braznik, A. N. 2006. *Simulation Modeling – Potentials of GPSS WORLD.* [In Russian] St. Peterburg: Rennomé.

Dias, L., G. Pereira, and A. Rodrigues. 2007. "A Shortlist of the Most Popular Discrete Simulation Tools." *SNE - Simulation News Europe* 17(1).

Gordon, G. 1978. "The Development of the General Purpose Simulation System (GPSS)." *ACM SIGPLAN Notices* 13(8):183-198.

Henriksen, J. O. 1985. "The development of GPSS/85." In *Proceedings of the 18th Annual Symposium on Simulation*, edited by A. Miller, 61-67. Los Alamitos, CA: IEEE Computer Society Press.

Henriksen, J. O. 1993. "SLX. The successor of GPSS/H." In *Proceedings of the 1993 Winter Simulation Conference,* edited by G. Evans, M. Mollaghasemi, E. Russell and W. Biles, 502-509. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Herper, H., and I. Ståhl. 2003. "Modeling and Simulation in High School Education - Two European Examples." In *Proceedings of the 2003 Winter Simulation Conference,* edited by S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1973-1981. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rothbauer, H. 2003. *Angewandte Simulation mit GPSS World for Windows*. Berlin: Logos.

Sammett, J.E. 1972. "Programming Languages: History and Future." *Communications of the ACM* 15: 601-611.

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

Schriber, T., S. Cox, J. Henriksen, P. Lorenz, J. Reitman, and I. Ståhl. 2001. "GPSS Turns 40: Selected Perspectives." In *Proceedings of the 2001 Winter Simulation Conference*, edited by B. Peters, J. Smith, D. Madeiros and M. Rohrer, 565-576. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ståhl, I. 1990. *Introduction to Simulation with GPSS - on the PC, Macintosh and VAX.* Hemel Hempstead, UK: Prentice Hall International.

Ståhl, I. 1996. *Simulation Made Simple with micro-GPSS. A Short Tutorial with Eight Lessons*. Stockholm: SSE.

Ståhl, I. 2000. "Automatic Animation with a Block Based Simulation Language." In *Simulation und Visualisierung 2000,* edited by T. Schulze, P. Lorenz and V. Hinz, 165-182. Magdeburg: SCS Publishing House.

Ståhl, I. 2001. "GPSS – 40 years of Development." In *Proceedings of the 2001 Winter Simulation Conference*, edited by B. Peters, J. Smith, D. Madeiros and M. Rohrer, 577-585. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ståhl, I. 2003. From 44 to 31 to 28 to 22 and now to 18 – Less Becomes More in GPSS. In *Simulation and Visualisierung 2003,* edited by T. Schulze, S. Schlechtenweg and V. Hinz, 465-478. Magdeburg: SCS Publishing House.

Ståhl, I. 2007. "Teaching Simulation to Business Students – Summary of 30 Years' Experience." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-J. Hsieh, J. Shortle, J. D. Tew and R. R. Barton, 2327-2335. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

AUTHOR BIOGRAPHIES

INGOLF STÅHL is professor emeritus of the Stockholm School of Economics, but continues his work on aGPSS and its application to business problems. His email address is ingolf.stahl@hhs.se and his aGPSS system is at http://www.gpss.se and http://www.webgpss.com.

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation. He was the chief developer of the first version of GPSS/H, of Proof Animation, and of SLX. He is a frequent contributor to the literature on simulation and has presented many papers and has participated in many panel sessions at the Winter Simulation Conference. Mr. Henriksen has served as the Business Chair and General Chair of past Winter Simulation Conferences. He has also served on the Board of Directors of the conference as the ACM/SIGSIM representative. In 2006, he was designated a Titan of Simulation and gave the Titan talk at the Winter Simulation Conference. His email address is mail@wolverinesoftware.com.

RICHARD G. BORN is an associate professor emeritus of Northern Illinois University, but continues teaching simulation modeling of business processes and his work with Ståhl on aGPSS. His email address is rborn@niu.edu.

HENRY HERPER is at the Institute for Simulation and Graphics at the Otto-von-Guericke University, Magdeburg. His research interests include the development and evaluation of simulation tools and methodology for simulation education. His email address is henry.herper@ovgu.de.