# A BUSINESS COURSE IN SIMULATION MODELING

**Richard G. Born, Northern Illinois University, rborn@niu.edu**
**Ingolf Ståhl, Stockholm School of Economics, ingolf.stahl@hhs.se**

## ABSTRACT

*The economic and improved decision-making value of discrete-event simulation has been employed by businesses for years, though not at a large scale in every area of business. The lack of rapid growth in the use of simulation modeling is due, in part, to the cost of many simulation software packages in addition to the difficulty of learning them and the complexity of developing even the simplest of models of business systems with them. This paper discusses a modern version of GPSS (General Purpose Simulation System) known as WebGPSS, which was designed to overcome these problems as well as satisfy the special interests and needs of the business student. Students working with WebGPSS quickly see that it provides business value, allows sharing of information for decision making, is practical, and promotes the idea that simulation is fun.*

**Keywords:** decision support, simulation modeling, discrete-event simulation, WebGPSS

## INTRODUCTION

WebGPSS [6] is the most modern implementation of micro-GPSS, a streamlined version of GPSS, the General Purpose Simulation System, which originated more than 40 years ago. WebGPSS has been in existence for about eight years and has been used by students in the United States [1] and in Sweden, Norway and Latvia [4]. In addition, I. Ståhl, developer of WebGPSS and a co-author of this paper, has taught GPSS simulation modeling to approximately 8000 students. WebGPSS has been used, in courses and in self-study, by students from more than sixty countries.

A variety of features of WebGPSS make it particularly appropriate for use in the business world. A few of them will be discussed here. First, it has a Graphical User Interface that allows building models graphically in the form of block diagrams for which the user supplies operands by double-clicking on the blocks within the block diagram. The use of block diagrams also facilitates the task confronting the model builder when explaining the model to the user of the results, facilitating sharing and synchronization of knowledge. Second it is available both on the Web (thus the name WebGPSS) and as a stand-alone version on a CD. The Web-based version, available free of charge at www.WebGPSS.com, is the most restrictive version as regards program size, but is large enough to run all of the programs in the 256-page WebGPSS textbook [2]. Third, WebGPSS has many pedagogical simplifications, including fewer blocks, making it much easier to learn than traditional GPSS. Fourth, it has a very extensive error-trapping and reporting mechanism with more than 500 error code descriptions, many of which have been improved over the years by suggestions from students who have worked with the software. Fifth, programs that are small, but powerful, can be written in a short time, as the programs are very similar to the real-world business problems they are designed to solve. Program code is created automatically, and no knowledge of computer programming is required. In addition, relevant statistics are produced automatically, and histograms and several types of graphs, including continuous, discrete, and scatter, are easily created to improve visualization of simulation results. Sixth, WebGPSS allows one to set up simulation experiments, with provision for both text output in the form of confidence intervals, and a histogram of result variable values. Seventh, WebGPSS has a complete teachware package [3] containing hundreds of PowerPoint slides designed for teaching all of the material covered in [2]. Some of the features of this teachware package can be found in [5].

Simulation software systems can be grouped into two general categories: Block Based Systems (BBS) and Animation Oriented Systems (AOS) [2]. The former include GPSS systems such as WebGPSS, ARENA/SIMAN, and SLAM/AweSim. The latter include ProModel, Witness, Automod, and SIMUL8 to mention just a few. The main difference between these two categories is that in the AOS each server is, in principal, represented only once, since it appears in only one place in the animation space. In a BBS such servers can be found in many places. The transactions are followed as they move through the blocks, and if different transactions use the same server, such usage can easily take place in different parts of the same program model. AOS systems are better when a detailed animation is required for a physical system, such as the flow of automobiles in

the assembly line of a factory. For other systems in which the advantage of animation is limited, BBS are often preferable because programming is simpler and repeated runs are simpler and faster. Many systems studied by the business student fall into this category, e.g., service, inventory and financial systems.

The seven features of WebGPSS discussed early in the Introduction, coupled with the advantages of BBS for the business student, have formed the basis of the decision to use WebGPSS in a business course in simulation modeling. One of the authors, R. Born, has made use of micro-GPSS, the simulation engine underlying WebGPSS, since 1991, and WebGPSS since 2002. In the past 17 years this author has taught approximately 1000 business students simulation modeling at both the graduate and undergraduate levels. Most of the students taking these courses have been students in the areas of operations management and information systems, but there have also been some from the M.S. in Accountancy program.

This paper discusses three simulation models representative of the kinds of business systems that can be simulated with WebGPSS. The first is a simulation of a small, family-owned retail shop. The second is a financial model showing how one can simulate cash flow for a company. The third shows how one can apply simulation modeling to project time planning, eliminating many of the restrictions of other analytic approaches.

## RETAIL SHOP SIMULATION

Figure 1 shows the floor layout for a small, family-owned meat market. To aid traffic flow in the shop, there is a separate entrance and exit. Because of its popularity, the shop is quite busy, with customers entering the shop with inter-arrival times that are uniformly distributed from 60 to 120 seconds apart. They then browse and shop in aisle 1 for 10 to 60 seconds, uniformly distributed on that interval. After this they get in line to buy meat. There are three butchers available to service customers at the meat counter. Service takes between 2 and 6 minutes, uniformly distributed on that interval. After leaving the meat counter, customers browse and shop in aisle 2, again for 10 to 60 seconds, with times uniformly distributed. Finally, the customers get into the checkout line, where one clerk is available. The clerk spends between 1 and 2 minutes checking out each customer, uniformly distributed on that interval. The meat market is to be simulated for a three-hour time period, and queuing statistics are to be compiled for both the meat counter line and the checkout line. In addition, statistics and a histogram should be prepared that will provide information on the number of customers spending various total times in the shop in two minute intervals (i.e., 0-2 minutes, 2-4 minutes, etc.).

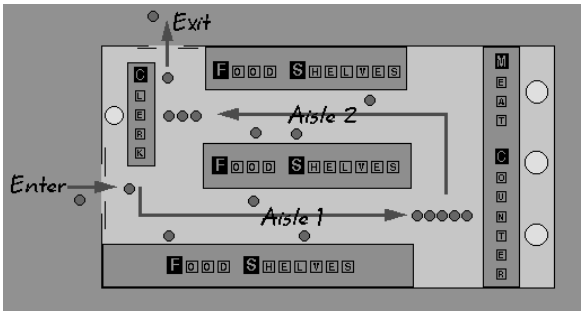


**Figure 1.** Floor Layout of the Butcher Shop

Figure 3 on the next page shows the WebGPSS window after constructing the model, accomplished by clicking on symbols in the symbol menu to the left, thereby placing them in the block diagram layout area to the right. This block diagram consists of two segments, each beginning with a GENERATE block. The left segment is the customer segment, and the right segment is the stop segment, which controls when the simulation stops. Operands that appear in each of the blocks in the block diagram are entered by double-clicking on the block and then supplying the values in the operands window that opens. An example of an operands window is shown in Figure 2 below for the GENERATE block in the customer segment. Customer arrivals between 60 and 120 seconds apart correspond to 90±30 seconds, when expressed in terms of Average inter-arrival time (IAT) and ½IAT spread. The comment field serves to document the block in ordinary language and is automatically added to the program code.

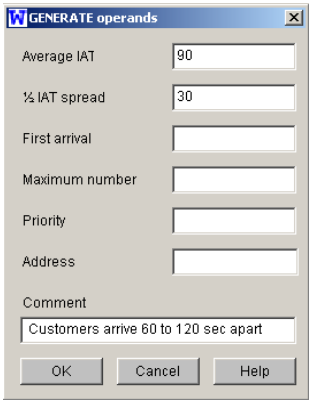


**Figure 2.** GENERATE Operands for Customers

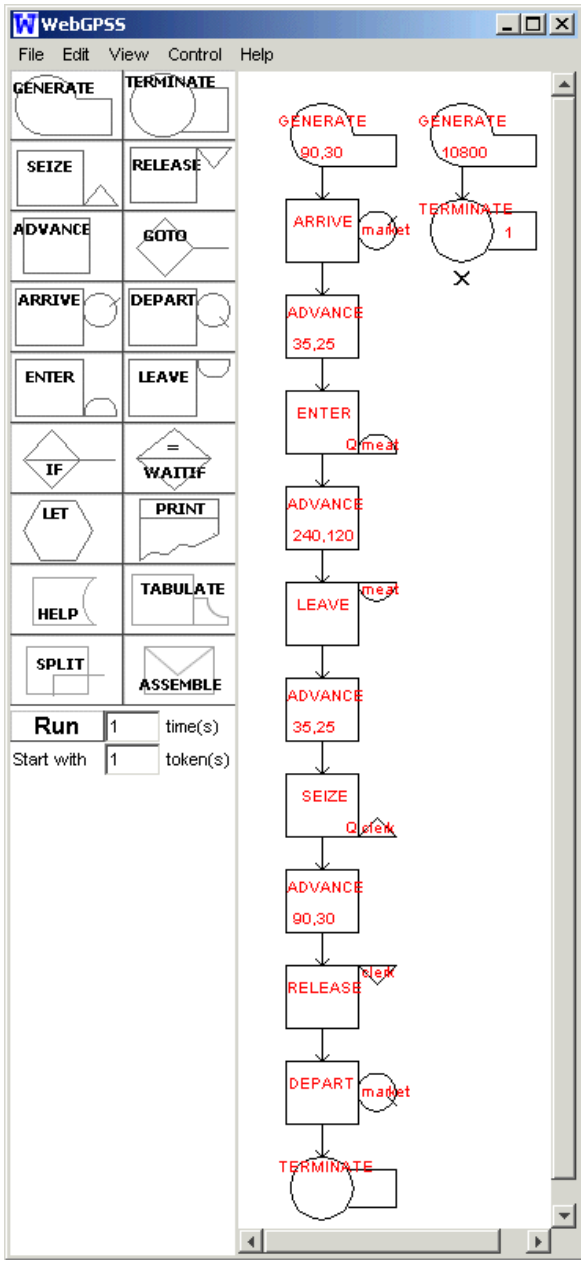


**Figure 3.** Butcher Shop Block Diagram

After being created in the GENERATE block, customers move through the blocks in the customer segment by following the arrows as shown in the block diagram. The ARRIVE block makes the customer a member of a statistical queue (or AD set) with the name MARKET, with the time of entry into this block noted. The block *ADVANCE 35,25* delays the customer for a time uniformly distributed on the interval 35±25 seconds (i.e., from 10 to 60 seconds), while the customer shops in aisle 1. In the *ENTER meat,Q* block, the customer attempts to get service from a set of butchers working in parallel (i.e., a storage in GPSS terms). If a waiting line forms, it is a joint waiting line processed in a FIFO fashion by the butchers. The Q in the block indicates that WebGPSS is to collect queuing statistics on the waiting line that forms in front of the butchers. The number of butchers is defined by specifying the capacity of the storage *meat*, as shown in Figure 4 below.

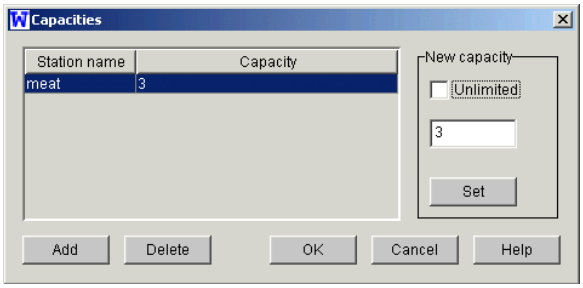


**Figure 4.** Capacities Window Defining the Number of Butchers in the Shop

After obtaining a butcher, the *ADVANCE 240,120* block is where the customer is serviced by the butcher for a time uniformly distributed on the interval 240±120 seconds (i.e., from 2 minutes to 6 minutes). Note that, in a WebGPSS model, the time unit can be whatever the modeler wants it to be. In order to avoid dealing with fractions of time units, it is common to let the time unit be the smallest unit used in the original description of the problem to be solved. After being serviced by a butcher, the customer enters the *LEAVE meat* block, which frees the butcher to begin serving another customer if one is waiting. The next block in the customer segment, *ADVANCE 35,25*, is where the customer shops in aisle 2. The customer then tries to get service from a cashier in the block *SEIZE clerk,Q*. SEIZE can be used whenever service can be given to only one customer at a time. If the cashier is busy, then the customer waits and is served in a FIFO fashion. The Q in the SEIZE block indicates that WebGPSS is to collect queuing statistics on the waiting line that forms in front of the cashier. The block ADVANCE 90,30 delays the customer while being serviced by the cashier during a time uniformly distributed on the interval 90±30 seconds, i.e., from 1 to 2 minutes, as stated in the original description of the problem. The RELEASE block frees the cashier to serve another customer. The *DEPART market* block causes the customer to leave the AD set *market*, the time of entry to the block is noted, and the total time spent in the AD set is calculated. In this model the AD set thus measures the total time that the customer spends in the butcher shop. The TERMINATE block removes the customer from the system.

In order to obtain statistics and a histogram of the total time customers spend in the butcher shop, the modeler defines a queue table control statement by clicking on *Control* from the main WebGPSS menu. The *Queue tables* window, after being filled out by the modeler, then appears as shown in Figure 5. Here we see that a Queue table is being defined for the AD set *market*, that the width of each class is 120 seconds (i.e., 2 minutes), and that WebGPSS is to also produce a histogram of the results.

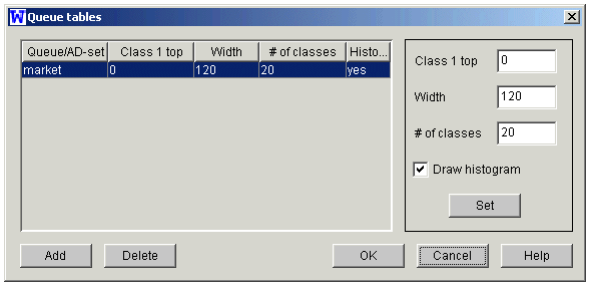


**Figure 5.** The Queue Table Window Defining Statistics and a Histogram for Total Time in the Shop

The stop segment, shown to the right in the block diagram of Figure 3, controls when the simulation stops. The block *GENERATE 10800*, creates a transaction at time 3 hours (i.e., 10,800 seconds) after the simulation begins. This transaction then moves into the block *TERMINATE 1*. This block removes the transaction from the system and decreases the termination counter by 1, or in other words, removes one token from the number that existed at simulation start, specified in the *Start with* textbox shown below the symbol menu to the left in Figure 3. When the termination counter becomes zero or goes negative, the simulation stops. Since the termination counter in this model began with the value 1, this transaction arriving at 3 hours will stop the simulation, and the *Results* window will automatically open.

The Results window includes: an extended program listing of the program code produced automatically by WebGPSS; block statistics; station statistics for the cashier and butchers; queue statistics for the cashier and butcher; and AD statistics, a table and histogram for the AD set MARKET. While space considerations here do not allow displaying all of the results in this paper, Figure 6 shows the extended program listing, Figure 7 shows the queue and AD statistics, and Figure 8 shows the histogram of total times spent by customers in the butcher shop.

```
        SIMULATE  1

meat    CAPACITY  3
        QTABLE    market,0,120,20,G
        GENERATE  90,30       ! Customers arrive 60 to 120 sec apart
        ARRIVE    market      ! Compute total time in market
        ADVANCE   35,25       ! Spend 10 to 60 sec in Aisle 1
        ENTER     meat,Q      ! Try to get butcher service
        ADVANCE   240,120     ! Butcher needed for 2 to 4 min
        LEAVE     meat        ! Butcher service done -- free butcher
        ADVANCE   35,25       ! Browse Aisle 2 for 10 to 60 sec
        SEIZE     clerk,Q     ! Try to check out of market
        ADVANCE   90,30       ! Checkout takes from 1 to 2 min
        RELEASE   clerk       ! Done paying--free the clerk
        DEPART    market
        TERMINATE

        GENERATE  10800       ! Run market for 3 hours
        TERMINATE 1           ! Close the meat market

        START     1
        END
```

**Figure 6.** Program Code for the Butcher Shop

| Queue or AD set | (1) Maximum contents | (2) Average contents | (3) Total entries | (4) Zero entries | (5) Percent zeros |
|---|---|---|---|---|---|
| MARKET | 9 | 6.07 | 120 | 0 | 0.00 |
| MEAT | 3 | 0.29 | 120 | 67 | 55.83 |
| CLERK | 5 | 1.48 | 118 | 10 | 8.47 |

| Queue or AD set | (6) Average time/trans | (7) $Average time/trans | (8) Current contents |
|---|---|---|---|
| MARKET | 546.13 | 546.13 | 5 |
| MEAT | 26.37 | 59.71 | 0 |
| CLERK | 135.55 | 148.10 | 2 |

$Average time/trans=average time/trans excluding zero entries

**Figure 7.** Queue/AD Statistics for the Butcher Shop

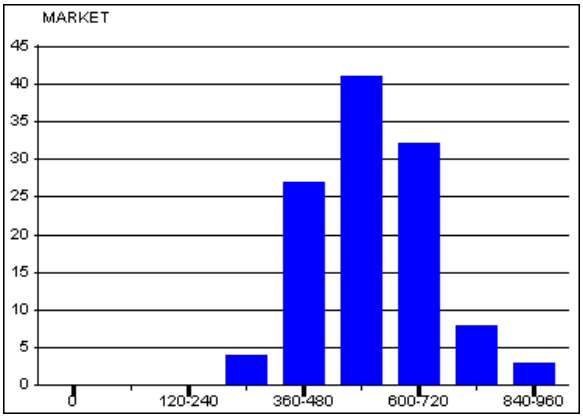


**Figure 8.** Histogram of Time Spent in Butcher Shop

We note when viewing the results that the maximum number of customers ever waiting for a butcher was 3, and that the average time spent by a customer waiting for the clerk was 135.55 seconds, suggesting an additional clerk may be advisable. The histogram shows us that, for example, 32 customers spent between 600 and 720 seconds (10 to 12 minutes) in the shop.

## CASH FLOW SIMULATION

This simulation model involves an import firm that buys computer-controlled machines from a producer, writes specialized programs to control the machines, and then sells the programmed machines to its customers [2]. Of particular interest is studying the cash flow of this import firm and how cash flow is affected by the price the firm sets per machine for its customers.

The firm initially has $100,000 in cash. It buys each unit from the producer on consignment, paying the producer $15,000 cash per unit at the time the import firm sells the unit to a customer. After programming the unit, the import firm charges its customers $25,000 per unit, but is free to change the price. The customers are provided with a credit, and asked to make full payment within 30 days. With some customers paying right away while others pay significantly later than 30 days, the average time to make payment is about 45 days. Though one can easily take into account interest on this credit, this is disregarded in this example due to space limitations for this article. Each customer order is assumed to be for a single unit.

Management of the import firm has estimated the number of units sold annually for a variety of prices in accordance with the chart show in Figure 9. The higher the price the fewer units it sells annually, as customers are then more likely to buy from a competitor.


**Figure 9.** Import Firm's Annual Sales Estimates

Figure 10 shows the WebGPSS window after constructing the model. This block diagram consists of two segments. On the left in the block diagram layout area is the customer segment and on the right is the stop segment, with both segments beginning with a GENERATE block. The unit of time for this model is *days*.

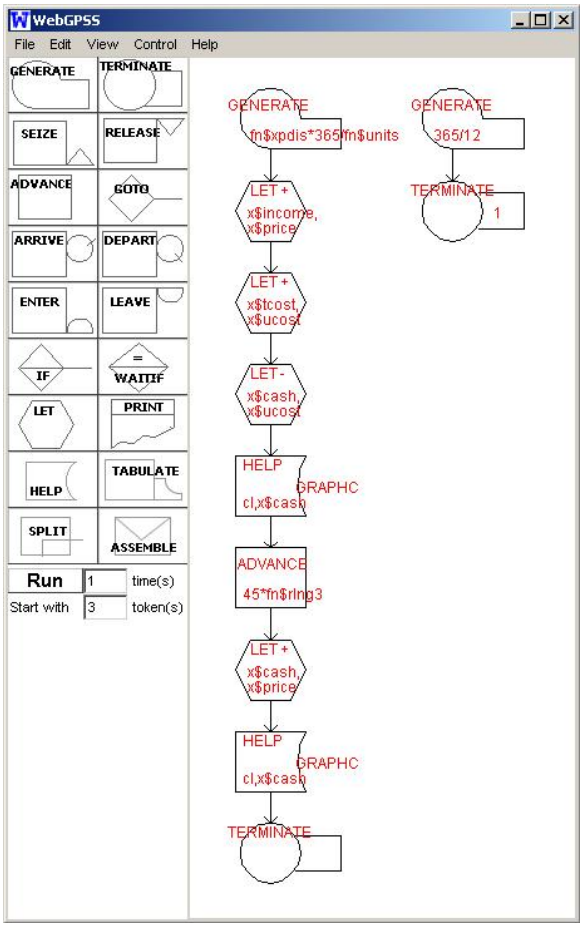


**Figure 10.** Import Firm Block Diagram

First, a discussion of the customer segment is presented. Customer inter-arrival times are assumed to be exponentially distributed with a mean time that is related to the price the firm charges its customers for a machine. This is accomplished by the block:

*GENERATE fn$xpdis*365/fn$units*

The function *fn$units* is a user-defined function, easily set up in WebGPSS, that computes the annual sales as a function of price, as illustrated in Figure 9. Dividing 365 by the annual sales then gives the mean inter-arrival time, in days, between customer orders. The function *fn$xpdis* is a WebGPSS built-in function that provides random, negative exponential arrivals with a mean of 1, thus adjusting the mean time for the exponential arrivals accordingly.

Following the GENERATE block are three LET blocks, that correspond to assignment statements in ordinary programming languages such as Visual Basic. User defined names beginning with *x$* essentially correspond to ordinary variables in the

traditional programming languages. With this fact in mind, the block *LET+ x$income,x$price* increases the income by the price of a unit machine. The value of *x$price* will be interactively input by the user when running the simulation model. The next block, *LET+ x$tcost,x$ucost*, increases the total cost of all machines sold by the unit cost that must be paid to the producer. The value of *x$ucost* is initialized at $15 (thousand), in accordance with the problem statement. Note that all monetary units in the model are in thousands of dollars. The next block is a LET block in decrease mode, *LET- x$cash,x$ucost*, that decreases the amount of cash on hand by the amount that must be paid to the producer because of the sale. The value of x$cash is initially set to $100 (thousand) in accordance with the problem statement.

Since the amount of cash on hand has just decreased, the next block, *HELP GRAPHC,cl,x$cash*, saves the current values of the simulation clock (*cl*) and cash on hand (*x$cash*) so that WebGPSS can later display a graph showing how the amount of cash changes with time. The *C* in GRAPHC implies that a continuous graph will be plotted.

The block *ADVANCE 45*fn$rlng3* delays the customer until the customer is ready to make payment. It makes use of a WebGPSS built-in Erlang distribution whose shape factor is 3. The random distribution *fn$rlng3* has a mean of 1, so multiplying by 45 makes the mean 45 days, as was specified in the problem statement. The Erlang function is ideal for representing the bill payment behavior of customers. Some people pay early, but only a few pay right away, i.e., as soon as they receive their bill. While the most frequent time to make payment is very near the due date of 30 days, some people pay much later than the due date. Figure 11 clarifies this behavior.

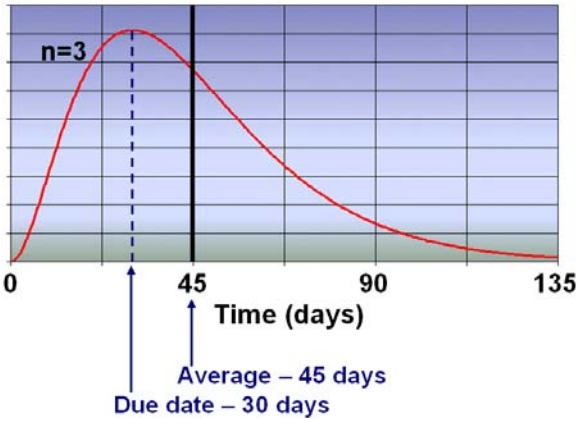


**Figure 11.** Bill Payment Behavior of Customers

Since the customer is now ready to make payment for the machine, the block *LET+ x$cash,x$price* increases the current amount of cash by the price the customer pays for the machine. Since the amount of cash has again changed, the HELP GRAPHC block below the ADVANCE block records the current clock reading as well as the current value of the cash on hand for WebGPSS to use later in displaying the cash flow graph. The TERMINATE block finally removes the customer from the system.

The stop segment consists of just two blocks. The block GENERATE 365/12 creates transactions with inter-arrival times of one month, i.e., 365/12 days apart. Noting in Figure 10 that the simulation starts with 3 tokens, this implies that the TERMINATE 1 block will cause the simulation to stop when three months have elapsed.

WebGPSS automatically initializes *x$* variables to the value 0. Non-zero initial values for variables are easily set by clicking on *Control* from the main menu, and selecting *Start values* from the dropdown menu. Figure 12 shows the Start values window after the modeler has completed the required entries. The cash on hand is initialized at $100 (thousand) and the unit cost of a machine is initialized at $15 (thousand) dollars. The user-defined prompt *What is the machine price?* causes WebGPSS to ask the user to enter the desired value when the simulation is run. This capability of interactive input allows the modeler to develop a model for an end-user, and then let the end user experiment with different values, improving the end-user's decision-making ability.

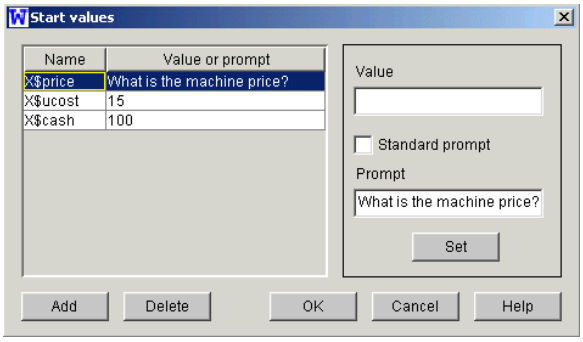


**Figure 12.** Start Values for the Import Firm Simulation

When the user runs the simulation by clicking the run button shown in Figure 10, an Input window appears asking the user to enter the price of a machine. Figure 13 shows what this input window looks like after the user has typed in 25, representing a price of 25 thousand dollars per machine.

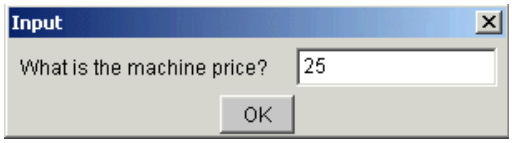


**Figure 13.** Interactive Input of Machine Price

The simulation results include a continuous graph of the amount of cash on hand as a function of the simulation clock time, as shown in Figure 14. It is noted that there is a fairly large amount of negative cash flow from roughly day 40 to day 80. The import firm would clearly find that it would have to borrow to make ends meet.

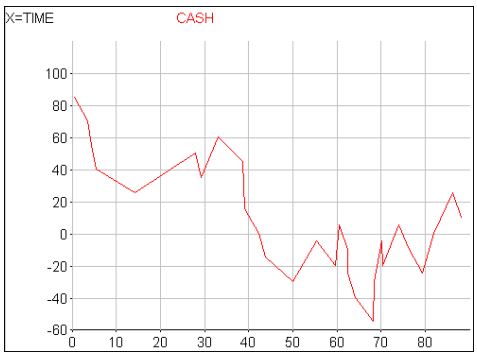


**Figure 14.** Import Firm Cash Flow with Machine Price Set at $25,000

Intuition suggests that raising the price of the machine is a possible way to avoid negative cash flow. A higher price will provide more cash each time a customer pays for the product. A higher price will also result in fewer units sold, lowering the number of customers who receive credit at any one time and decreasing the frequency with which the firm must pay its supplier for the units it buys on consignment. Figure 15 verifies this intuitive deduction for a machine price of $33,000.

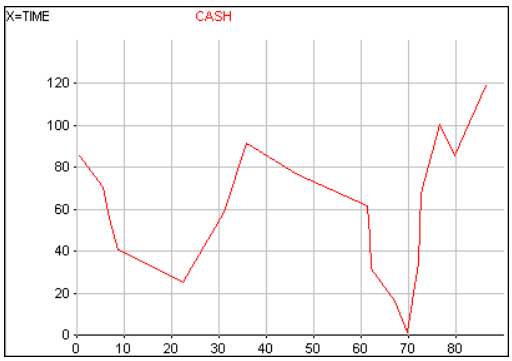


**Figure 15.** Import Firm Cash Flow with Machine Price Set at $33,000

## PROJECT TIME PLANNING SIMULATION

Problems that deal with project time planning are excellent candidates for GPSS solutions. Projects like building a house or a bridge, for example, are characterized by consisting of a number of tasks. Some tasks can be done in parallel with other tasks. Certain tasks cannot start until work on other tasks is complete. The time required for a given task is generally not fixed, but rather stochastic. Of particular interest is the time that the whole project will take. With random activity times, this means that one would like to determine the probabilities for various total project times.

Traditional methods for dealing with project time planning include PERT (Program Evaluation Review Technique) and CPM (Critical Path Method). Some of these methods require a fixed time for each activity, and some allow stochastic times, but are often limited to a distribution known as the Beta distribution. They do not always give correct estimates of the distribution of total project time, and those that do are generally limited to estimating time variation only on the critical path, the longest time path through the project network. Lastly, resource limitations for each of the tasks are generally not considered explicitly. The beauty of GPSS solutions to project time planning problems is that they do not have any of the above limitations.

The problem to be considered here involves the construction of houses. We assume that a contractor is planning to build a total of 100 houses in a new subdivision. The contractor is interested in determining the total time required to build the 100 houses as well as a distribution of the construction times for individual houses. For simplicity, it will be assumed that the construction of a house involves just four tasks. The first task, process A, is building the frame of the house. Processes B and C, plumbing and electrical wiring, respectively, can begin only after the frame has been built, but can be done concurrently with one another. Finally, there is a process D, painting, which can be done only after the plumbing and electrical wiring tasks have been completed. Figure 16 provides a pictorial view of the building of a house and indicates the nature of the time distributions required for each of the four tasks. Note from the figure that building the frame is assumed to be exponentially distributed with a mean of 32 hours. Plumbing and painting are uniformly distributed with distributions of 28±5 and 30±15 hours, respectively. Lastly, painting is normally distributed with a mean of 22 hours and a standard deviation of 4 hours. Note that the WebGPSS built-

in distribution *fn$snorm* has a mean 0 and standard deviation 1.

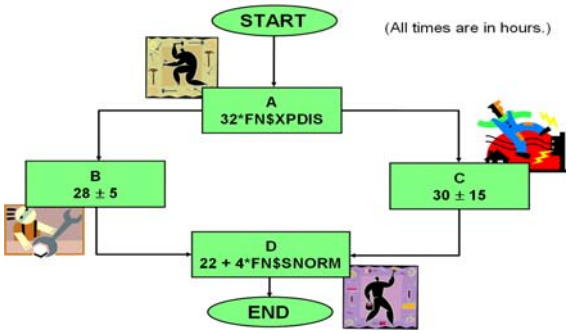


**Figure 16.** Tasks in Building a House

For each of the processes shown in Figure 16, it is assumed that there is only one worker available, although the model can readily be adjusted to allow for more workers. When a worker has completed his part of the house, he can, if it is possible, start doing the same kind of work on the next house. As an example, the carpenter can start construction on the next house as soon as he has completed his work on the previous house.

This model requires the use of the SPLIT and ASSEMBLE blocks of WebGPSS, shown in Figure 17. The SPLIT block allows one to create copies of transactions already existing in the simulation. The A operand specifies the number of copies, and the B operand is the address of the block to which the copies are to go. The transaction entering SPLIT from the top simply exits the SPLIT block at the bottom. SPLIT is useful when one wants to provide for some concurrent activities, e.g., allowing both the plumber and electrician to do their work once the frame of the house has been built by the carpenter.

The ASSEMBLE block merges several transactions coming from the *same* original, back into *one* transaction. The *A* operand is the number of transactions that are merged into one transaction, and merger does not take place until the number of transactions specified in the *A* operand have entered the ASSEMBLE block.

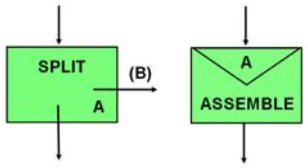


**Figure 17.** The SPLIT and ASSEMBLE Blocks

The block diagram for the contractor's house construction project time planning simulation is shown in Figure 18 and will be discussed on the next page.

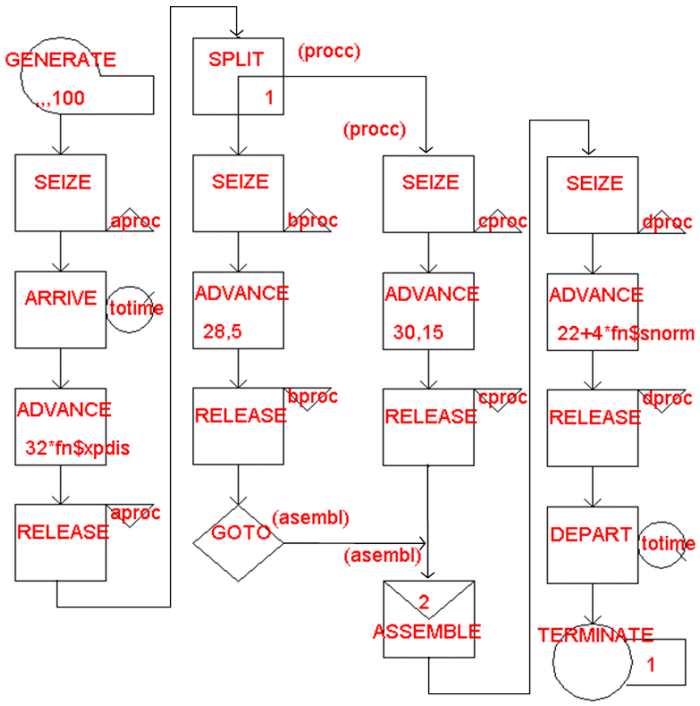


**Figure 18.** Block Diagram for the House Construction Project Time Planning Simulation

The GENERATE block creates 100 transactions at simulation time zero. These transactions represent 100 houses ready to be built. Construction of a house begins when it can seize the carpenter (*aproc*), who is responsible for building the frame of the house. The ARRIVE block marks the moment the house begins construction. In the block ADVANCE 32*fn$xpdis, the carpenter builds the frame, and then is freed by the RELEASE block that follows. The SPLIT block then creates a copy of the transaction that goes to the electrician (*cproc*), while the original goes straight through the SPLIT to the plumber (bproc). When the plumber and electrician have completed their tasks and are freed by their corresponding RELEASE blocks, the original transaction and the copy are merged into one transaction after both of them reach the ASSEMBLE block. The merged transaction then attempts to get service from the painter (*dproc*). When the painter is freed via its RELEASE block, the DEPART block then marks the moment the house is complete and the time to construct the house is computed by WebGPSS. The transaction finally enters the block *TERMINATE 1* where the transaction is removed from the system and the termination counter is decreased by 1. If the simulation starts with 100 tokens, the simulation will then stop when the last token is removed, causing the termination counter to become zero, i.e., when the 100th house has been completed and the contractor's job in the subdivision is done.

A queue table is defined for the AD set *totime*. The width of each time class is set to 100 hours, and the checkbox requesting a histogram is checked. When the simulation is run, the resulting histogram then provides a distribution of the times required to build the houses, as shown in Figure 19.

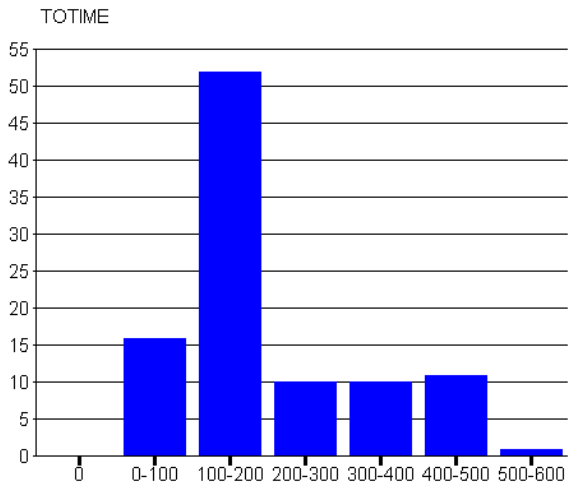


**Figure 19.** Distribution of Times to Build Houses

There is clearly a great deal of variation in the time required to construct a house. A majority (52) of the 100 houses take between 100 and 200 hours to build. Almost a third of the houses (31) take between 200 and 500 hours to build, and one house even requires between 500 and 600 hours.

A good question at this point relates to how much of this variation is due to randomness in the times of the four processes. One can investigate this by making four small changes to the model. The random times in the four advance blocks are simply replaced by their deterministic values, namely, 32, 28, 30, and 22, respectively. When the new simulation is run, it is found that each one of the houses requires exactly 84 hours to build. These 84 hours correspond to the sum of the times along the so-called critical path (longest time path in a PERT diagram), consisting of the processes A(32), C(30), and D(22). The differences between the stochastic and deterministic results are phenomenal. Stochastic time variations clearly play a very important role and should be taken into consideration if one is interested in obtaining realistic and useful results.

## ADDITIONAL WebGPSS CAPABILITIES

While we have carefully looked at three examples of how simulation modeling can be used to solve some typical business problems, there are many features of WebGPSS for which time and space here does not allow discussion. There are five additional blocks in the symbol menu of Figure 3 that have not been discussed. The GOTO block allows for sending transactions to a specific block unconditionally or with a given probability. The IF block allows transferring transactions to a specific block based either on the status of a server or a condition involving the values of a large variety of system variables. The WAITIF block looks at the status of a server and has the capability of causing a transaction to wait if and as long as a specific condition (e.g., busy or not busy) regarding the status is true. The PRINT block allows one to get printouts of the values of variables and system statistics *during* a simulation. This is useful when the automatically produced statistics at the end of a simulation are not enough for the required analysis. The TABULATE block allows one to gather statistics on the values of variables whenever a transaction enters a TABULATE block. There are also several other control features. One of the most important of these is the ability to design experiments that run a simulation model many times, automatically obtaining confidence intervals for the value of the result variable.

## CONCLUDING COMMENTARY

The authors believe that in the coming years, there will be a significant increase in the role that simulation plays in the business world. The main reason for this is the extremely rapid development of computer technology, which, as seen by Moore's Law, means doubling in speed and capacity roughly every 18 months. This, in turn, implies that computers, for a given price, have increased in power more than a thousand times in the past two decades. This implies that if simulation earlier, because of high computing costs and time requirements, was used as a method of last resort, simulation can now be regarded as a first alternative for solving a large number of business problems.

Because of these developments, computer simulation has been able to replace many analytic and optimization techniques such as queuing theory, inventory theory, PERT/CPM, and decision theory. It is quite appealing to cover many problem areas with **one** single easy-to-use method instead of relying on teaching and learning a great many more complex mathematical models. Simulation also allows one to place a much greater focus on **modeling**. Simulation plays a major role in production planning, providing a better understanding of the physical processes in a firm. Closely related to this is that one can, with simulation, demonstrate the connection between physical activities and financial flows, as was shown by the cash flow simulation discussed in this paper.

Furthermore, stochastic simulation is necessary when handling uncertainty that forms the core of financial theory. This is illustrated by thinking about how one would want to answer the following questions:

- How many units will we sell next year for a given product or service? 100,000 units for certain or 80,000 to 120,000.
- When will this customer make payment to us? Within 30 days for certain or within 60 days with a probability of 80%.
- What will be the $/€ ratio a year from now? 1.5 for certain or between 1.2 and 1.8.

In all cases, the last answer, indicating uncertainly, seems more reasonable.

Why not use only Excel rather than a special simulation language such as WebGPSS when doing dynamic simulations, where we want to see how things change over time? The primary reason is that most dynamic simulations, as shown in the cash flow of Figure 14 for example, are completely impossible to do in ordinary Excel without resorting to Visual Basic. One reason is that standard Excel, generally limited to 256 columns, cannot allow showing all time points. Of the greatest importance is that Excel is built for "pulling" data from a cell, and not assigning values to cells. Furthermore, the Excel RAND function is in many regards inferior to the built-in random number generators of most simulation languages. Finally, in Excel, IF constructs are limited to single cells and looping has to be done by copying. Most special simulation languages allow for more general IF and looping constructs like general programming languages.

For business courses that devote a full semester or less to simulation, WebGPSS has been found to be the most suitable software. From WebGPSS one can next easily proceed to commercially available simulation packages like e.g. GPSS/H, GPSSWorld, SLX and SIMAN/ARENA.

## REFERENCES


1. Born, R. (2003). "Teaching Discrete-event Simulation to Business Students: The Alpha and Omega". In *Proceedings of the 2003 Winter Simulation Conference.* Eds. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 1964-1972. Piscataway, New Jersey: IEEE. www.informs-sim.org/wsc03papers/254.pdf
2. Born, R. and I. Ståhl. (2007). *Simulation Made Simple with WebGPSS*. Gothenburg, Sweden: Beliber.
3. Born, R. and I. Ståhl. (2007). *WebGPSS Slide Presentation.* DeKalb, IL: R. Born. Available for evaluation on request from R. Born at rborn@niu.edu.
4. Ståhl, I (2007). "Teaching Simulation to Business Students - Summary of 30 Years' Experience". In *Proceedings of the 2007 Winter Simulation Conference.* Eds. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1973-1981. Piscataway, New Jersey: IEEE. www.informs-sim.org/wsc07papers/291.pdf
5. Schriber, T., I. Ståhl, J. Banks, A. Law, A. Seila, and R. Born. (2003). "Simulation Textbooks—Old and New" (Panel). In *Proceedings of the 2003 Winter Simulation Conference*. Eds. S. Chick, P.J. Sánchez, D. Ferrin and D. J. Morrice, 1952-1963. Piscataway, New Jersey: IEEE. www.informs-sim.org/wsc03papers/253.pdf
6. Ståhl, I. (2003). Simulation Made Simple with WebGPSS—A Tutorial. Stockholm: Stockholm School of Economics.