Proceedings of the 1984
Winter Simulation Conference
S. Sheppard, U. Pooch, D. Pegden (eds.)

Simulation Programming Using Spreadsheet Software

Richard M. Reese
Department of Computer Science
Stephen F. Austin State University
Nacogdoches, Tx 75962

ABSTRACT

This paper will examine the use of Spreadsheet software for sophisticated simulation projects. The advantages and disadvantages of using spreadsheet software, specifically Lotus*, for simulation programs will be illustrated through two simulation examples. Future directions for simulation languages are proposed. These include the need for simulation language to provide an easy method to observe the dynamics of the simulation, an integrated database and graphic facilities, and a more non-algorithmic approach to simulation languages.

INTRODUCTION

Spreadsheet software constitutes a useful, flexible, and powerful business analysis tool. This type of software is being used extensively by accountants and other business professionals to assist in the decision making process. In the past, limited simulation problems have been approached using spreadsheet software. While these simulations are important, they will not be addressed as they are somewhat limited in scope and are adequately covered in other references [7]. This paper will address the use of spreadsheet software for more sophisticated simulation problems. These problems need not be limited to business applications.

SPREADSHEET SOFTWARE

Spreadsheet programs are relatively easy to use and are for the most part non-algorithmic in nature. The user approaches a problem by first defining or setting up the various relationships between components of the problem through the use of mathematical expressions. Once these definitions have been established, the user enters the appropriate values corresponding to the problem's components and the spreadsheet will automatically calculate and display the results. Some spreadsheet programs, such as Lotus 123, also incorporate database and graphic facilities. Because of the flexibility and popularity of Lotus, examples used in this paper will be based upon Lotus [8].

The current state of the art limits what is possible using the non-algorithmic facilities of spreadsheet programs. As a result, the capabilities of most spreadsheet programs are enhanced with a limited algorithmic facility. The algorithmic facilities of Lotus are termed macros by Lotus. Special key indicators and /X commands are used to create macros. The /X commands are the more powerful facilities and include basic programming constructs such as simple sequence, go to, logical if, subroutine invocation, and return statements. The use of these constructs allow sophisticated simulation problems to be pursued. Table I and Table II list the special key indicators and the /X commands respectively.

There are several other features of Lotus which can be used to assist in the development of a simulation program. These include a random number generator, table processing capability, sorting facilities, and various built-in statistical functions. Lotus, however, lacks several basic simulation primitives including queue primitives, an event list facility, and scheduling primitives. The usefulness of Lotus as a simulation language is partially determined by how well Lotus can adapt to these limitations.

Two examples are provided to illustrate the potential of Lotus as a simulation language. The first example models a small grocery store and illustrates the use of Lotus's database and graphic facilities. The second example is a somewhat simpler program which simulates a discrete sequential electronic circuit.

Grocery Store Simulation

The grocery store simulation, consisting of three checkout stands, is illustrated by examining the primary screen of the Lotus implementation as shown in Figure 1. The customer enters the store, shops,

*Lotus 123 is a trademark of the Lotus Development Corporation.

~	Return key	{Up}	Up cursor
{Down}	Down cursor	{Left}	Left cursor
{Right}	Right cursor	{Home}	Home cursor
{End}	End key	{PgUp}	PgUp key
{PgDn}	PgDn key	{Esc}	Escape key
{Del}	Delete key	{Bs}	Backspace key
{GoTo}	Goto key	{Query}	Query command
{Calc}	Calculate key	{Graph}	Graph command
{?}	Help key		

Table I: Special Key Indicators

/XIcondition~...	If-Then
/XGlocation~	GoTo
/XClocation~	Call Subroutine
/XR	Return from subroutine
/XQ	Quit macro execution
/XMlocation	Process a user-defined menu
/XNmessage~location~	Display a message and accept entry

Table II: The /X Commands

and then enters the shortest of three queues. Upon completion of the checkout process, the customer leaves the store.

As the program executes, the affected values displayed upon the screen change, thus conveying to the user the current state of the simulation. In the simulation illustrated in Figure 1, the value of various simulation statistics are modified and displayed. These include, for example, the queue's current, average, and maximum size. This capability is an inherent characteristic of Lotus and does not require any special code.

The advantage of being able to observe the changes which take place during the execution of the program is that the dynamics of the simulation can convey information which may provide the user with insight and a better understanding of the system being modeled. This information can be used to help in checking the validity of the program implementation as well as observing the execution of an actual simulation run. While this can be achieved with traditional simulation languages in an interactive environment, it is not achieved as easily.

This dynamic view at the same time presents new problems. The changing of too many variables too quickly could leave the user somewhat confused. The user may not be able to keep track of the changes and may not be able to fully comprehend the dynamics of the simulation. This problem suggests the need for some mechanism to take "snapshots" as the simulation progresses. This technique is not a new one but should not be overlooked in a Lotus simulation. While the user is able to watch the simulation progress, a history is often useful. It is possible to build snapshots into the Lotus simulation.

At any point during the execution of the simulation the execution can be halted. The user may choose one of three options. The first is to examine the current state of the simulation and then resume execution. The second is simply terminate the simulation at that point. The third alternative is to resume execution in a single step mode. This allows a slower, but more careful examination of the execution of the model. This single step execution capability is just one feature which makes spreadsheet software an attractive simulation tool. This mode of operation is not readily available in most other simulation languages.

In addition to the single step execution capability, the user is also given the opportunity to change system parameters and to redirect the program's execution. While this is not necessarily considered to be a good programming practice, during the debugging phase it can prove to be a useful technique if properly exercised. The system modification capability can also be used to modify the results of a simulation run. The existence and use of this capability should be noted and carefully considered.

A very useful feature of Lotus is the database management facilities. These facilities permit the user to construct a database and then query the database in a convenient and easy-to-use fashion. For example, after a simulation run the user may desire to isolate specific entities which may exist within the system. The query facilities permit the selection of records as specified by the user. The user does not need to export the information to an intermediate file and then import the information into a database management system. The facilities are already present. In addition, queries may be made during the actual simulation run if so desired. Table III summarizes the database management commands available with Lotus.

/Data Query	
Find	Used to highlight selected records
Extract	Copies selected records
Unique	Extracts duplicate records
Delete	Deletes selected records
Criterion	Selects records
Input	Specifies input data
Output	Specifies output area
/Data Sort	
Data Range	Enter data range
Primary Key	Specify the primary key
Secondary Key	Specify the secondary key

Table III: Query and Sort Commands

The integrated graphic facilities of Lotus also permit the user to quickly and easily generate graphic displays to illustrate the results of the simulation. The user is given considerable control over the form and type of the display. The user can generate line, bar, stacked bar, XY, and pie graphs. Again, this facility is part of the system and the necessity of importing or exporting information between systems is avoided [2].

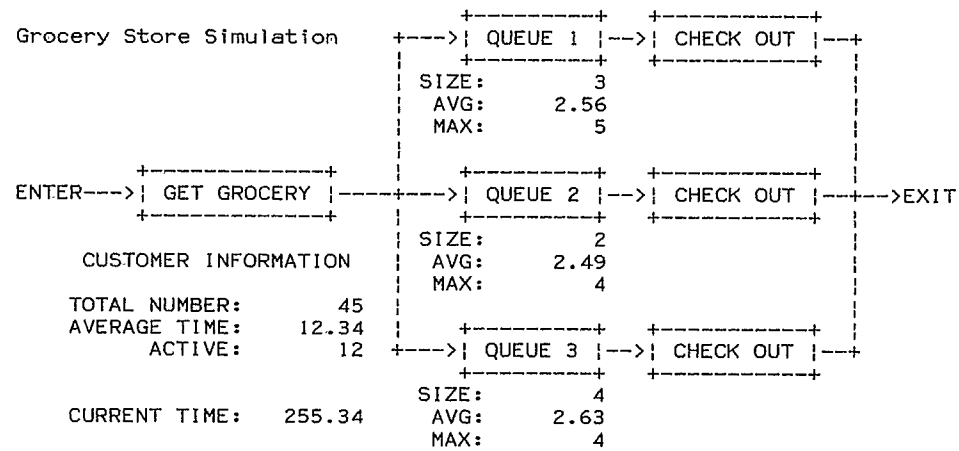

Figure 1: Grocery Store Simulation Primary Screen

Simulation Primitives

The primary simulation primitive used in the grocery store simulation was a simple queue. The queue was implemented by using a table of entities. Each entry in the table had specific attributes as shown in Figure 2. The queue was maintained by using Lotus's Sort command. Entities were enqueued by copying the relevant entities attributes to the last element of the table. The dequeue operation was implemented by copying relevant information from the first entry of the table to appropriate locations and then storing maximum values in the first table position. After both the enqueue and the dequeue operation, the table was sorted using necessary primary and secondary keys. This resulted in the proper ordering of the queue.

QUEUE 1

ENTITY NUMBER	ARRIVAL TIME	GROCERY TIME	ENTER LINE	EXIT LINE
3	1.12	2.32	4.34	
4	3.13	2.01	5.34	
5	3.13	1.95	6.34	
			99999	
			99999	
			99999	

Figure 2: Queue Table

While the implementation technique is rather primitive it did achieve the desired results. The Sort command provides the user with the flexibility needed to maintain queues. The user may establish necessary enqueueing criteria by sorting on appropriate primary and secondary keys. The two primary limitations of this technique are execution speed and a fixed queue size. The size of the queue is determined by the table size which is in turn determined by the microcomputer's memory limitations.

Scheduling was accomplished by entering into the appropriate tables the needed information. Macros were used to make these table entries and the random number generator was used to generate times. Rescheduling could be accomplished by changing the table times. The event list is maintained implicitly by the tables. The next scheduled event is determined by using the @MIN function on selected table entries. This function returns the lowest element of the list provided.

The random number generator is just one of several library functions. These include mathematical, logical, financial, date, statistical, database, and special functions. Table IV list the more useful simulation oriented functions.

@MIN	Chooses the minimum of arguments
@MAX	Chooses the maximum of arguments
@STD	Standard deviation of all list items
@VAR	Variance of all items in list
@RAND	Random number between 0 and 1
@IF(cond,x,y)	The value x if TRUE else y
@CHOOSE	Select argument value
@HLOOKUP	Horizontal table lookup
@VLOOKUP	Vertical table lookup

Table IV: A Sample of Lotus Library Functions

Another implementation detail concerns the method of choosing between queues. In the grocery store simulation, the shopper must choose between one of three checkout lines. The selection criteria for this simulation was that the shopper was to choose the shortest queue and in case of a tie choose the lowest numbered queue. This was implemented through the use of the @MIN function. Other more sophisticated selection criteria may also be supported.

Digital Simulation Model

Lotus can be used to easily simulate the action of a discrete digital circuit. A diagram of the circuit can be incorporated into the Lotus spreadsheet with a moderate amount of effort. Lotus was not designed to facilitate the use of graphics within the spreadsheet itself but rather separate from the spreadsheet. However, templates of the basic circuit symbol can be developed to suggest standard circuit components. Table V list some possible circuit symbols. A diagram of the digital circuit is not included due to space limitations. The scrolling capabilities of spreadsheets facilitate the viewing of large spreadsheet programs. This capability cannot be carried over to hardcopies. The circuit, however, illustrates a configuration consisting of two registers and the gates needed to perform an addition of the two registers. The circuit diagram is facilitated with standard fixed size columns and standard templates which can be copied and moved as needed. The drawing of lines between the templates can be a tedious task at times.

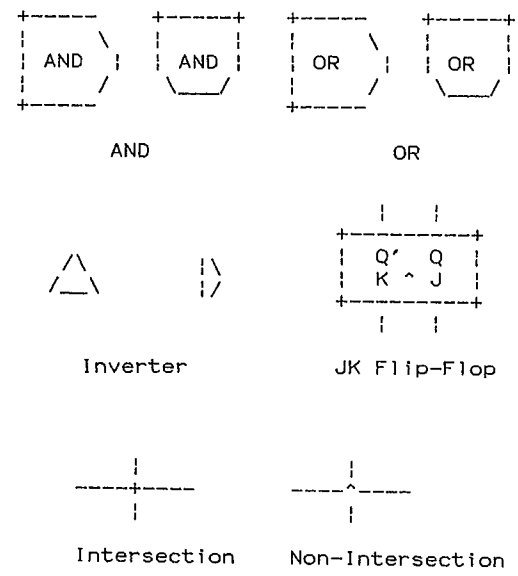

Table V: Circuit Symbols

The actual simulation formulas are easy to develop as Lotus provides several pertinent boolean functions. The modification of any bit value will result in an immediate and observable change in associated system components. This can be used to illustrate the structure and internal operation of the circuit to designers and students. The circuit illustrated is limited in scope but it does serve to illustrate the advantages of using spreadsheet software as a simulation tool.

CONCLUSIONS AND FUTURE RESEARCH

One of the major advantages of using spreadsheet software for simulation is that it allows the user to easily observe the execution of the simulation program. This can provide the user with a better understanding of the simulation model itself. Another advantage is the relative ease of generating a simulation for prototyping purposes. In addition, Lotus provides built-in database and graphic facilities.

The primary disadvantage of using spreadsheets as a simulation language is the speed of execution. Current microprocessors do not execute as fast as minicomputer and mainframe central processing units. The speed limitations will hopefully become less significant as new generations of faster microprocessors are introduced.

From a programming point of view, the lack of simulation primitives has required the implementer to reinvent the wheel. The lack of these primitives can be overcome as demonstrated in this paper. The implementation problems encountered are not significant when contrasted with the advantages obtained by using spreadsheets.

A major result of this research is is to suggest future direction for simulation languages. Simulation languages can be improved with the incorporation of single step execution modes [4], the automatic display of important system attributes, the integration of graphic and database facilities, and the incorporation of the non-algorithmic nature of spreadsheets. It is the non-algorithmic nature in particular from which language designers should learn from. The tedium of creating simulation programs can be reduced if the programmer is relieved of the burden of having to take care of all the little details which could be handled by the language system itself.

While spreadsheet software in its current state will not be the best choice for all simulation problems, it does have its uses. It can be used not only as the primary simulation language for a problem but also for rapid prototyping. The designer can quickly examine the problem and apply lessons learned to the full implementation.

There are numerous simulation languages which have proposed and integrated many of the above mentioned features [1, 3, 5, 6], but none which incorporate all of the features. A careful design must be undertaken to insure a useful and manageable language system. Spreadsheet software as discussed in this paper suggests a possible direction for future simulation languages.

Contemporary spreadsheet software is easy to use and manageable by individuals who have had little or no formal computer science education. While Lotus is lacking in basic simulation primitives, it is not difficult to compensate for these limitations. In addition, Lotus provides facilities such as an interactive execution capability, and various database and graphic functions which are easy to use and not commonly available in other simulation languages. Simulation language designers can learn from the spreadsheet example.

REFERENCES

1. Clema, J.K., "General Purpose Tools For System Simulation," Proc. 11th Annual Simulation Symposium, Tampa Florida, 37-60, March 15-17, 1978.

2. Donovan, J.J.; Jones, M.M.; and Alsop, J.W., "A Graphical Facility For An Interactive Simulation System," Information Processing, North Holland, Amsterdam, 1969, 593-596.

3. Greenberger, M.; and Jones, M., "On Line, Incremental Simulation," Simulation Programming Language, North Holland, Amsterdam, 1968, 13-32.

4. Sheppard, S.S. (Nelson S.S.), "CONSIM: A Study of Control Issues in Conversational Simulation," The Computer Journal, Vol 22, No 2, 119-126, May, 1979.

5. Sohnle, R.C.; Tartar, J.; and Sampson, J.R., "Requirements for Interactive Simulation Systems," Simulation, Vol 20, No 5, 145-152, May, 1973.

6. Strandridge, C.R., "The Simulation Data Language (SDL™*): Applications and Examples," Simulation, Vol 37, No 4, 119-130, October, 1981.

7. Trost, S.R.; and Pomernacki, C., VisiCalc For Science And Engineering, Sybex, Berkeley, 1983, 203 pages.

8. Lotus 123 User's Manual, Release 1A, The CSA Press, Belford, 1983, 362 pages.

---

*SDL is a trademark of Pritsker & Associates, Inc.