Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

DEVELOPING SPECIAL PURPOSE SIMULATORS UNDER MICROSOFT WINDOWS

Kieran L. Coughlan
Paul J. Nolan

University College Galway
Galway, IRELAND

ABSTRACT

This paper discusses the benefits of using Microsoft Windows to construct special purpose simulators. Microsoft Windows has several features which makes the development of special purpose simulators easier than ever before. The development of a special purpose simulator for the Irish Electricity Supply Board (ESB) is described. This simulator models the network fault location and repair procedure and assists the ESB to find ways to reduce the length of power cuts. Visual Basic was used to design a user interface to a GPSS/H model and greatly assisted the development of the simulator.

1 INTRODUCTION

Simulation has the potential to be extremely useful in all aspects of the service and manufacturing industries. Experiments can be inexpensively performed on a computer model without the need for any actual changes to plant or procedures, obviously making it an invaluable management planning tool.

Simulation, however, has never achieved the level of penetration into management use that it deserves. Perception of simulation as an engineering tool has a certain amount to do with this, but Shannon, Mayer and Adelsberger (1985) summarised one of the major reasons:

> *"Today, in order to use simulation correctly and intelligently, the practicioner is required to have expertise in a number of different fields. This generally means separate courses in probability, statistics, design of experiments, modelling, computer programming and a simulation language. This translates to about 720 hours of formal classroom instruction plus another 1440 hours of outside study (more than one person-year of effort), and that is only to gain the basic tools. In order to become really proficient, the practicioner must then go out and gain real world experience (hopefully under the tutelage of an expert)."*

Clearly, this is considerable more time than managers in industry (the people who should be using simulation regularly) could or would be willing to spend. Nevertheless, managers can use simulation if a special purpose tool, fitted to their abilities and needs, can be provided.

2 SPECIAL PURPOSE SIMULATORS

A *special purpose simulator*, as distinguished from a simulation language or simulation system, can be defined as follows (Smith and Crain 1993):

> *"The special purpose simulator is a custom built analysis tool designed by an experienced model builder. The heart of the special purpose simulator is a data-driven model of a specific system set of similar systems. The end user is provided with a method to easily modify model parameters, run tests, and get results."*

Traditional simulation systems are usable only by trained experts, however, a special purpose simulator, developed by a consultant, can provide a user friendly interface, insulating the user from the inherent difficulties of the underlying simulation system. At the heart of a special purpose simulator is a *parameterised model* of the system being studied. The front end, or interface, can lead the user through the steps required to set up an experiment, i.e. setting the parameters for the model, and present them with the results in an accessible format afterwards. Models developed in this way are then essentially fixed and may not be modified by the user except through the available parameters. It is also thereafter up to the user to use their knowledge of the real system to set realistic parameters and to properly analyse results.

Special purpose simulators have been developed previously for industry. Seppanen (1990), Productive Systems Inc., describes a general methodology for developing special purpose simulators. These simulators were designed to be used by manufacturing engineers and managers with little or no simulation experience. Pro-

ductive Systems used a combination of SIMAN (Pegden 1982), a commercial spreadsheet package and BASIC programmes to construct their simulators.

Modern personal computer systems are easier to use than ever before, and are almost ubiquitous on managers' desk-tops. The potential therefore for providing special purpose simulators to industry has never been greater.

2.1 Computer User Interfaces

Computer user interfaces have undergone a considerable evolution in the few years since computers widespread introduction to businesses and academia. Early computers were precious resources which allowed no interactive use. Users submitted jobs for processing, usually in punch card form, and waited, often several days, for results. Without any form of interaction between the user and computer, the notion of an interface was somewhat redundant at that time.

Modern computer systems almost invariably offer a Graphical User Interface (GUI). GUI's offer a natural representation of tasks, objects and actions referred to as *direct manipulation* (Schniderman 1992). These systems offer:

- Visual representation (metaphor) of the world of action, i.e.: Objects and actions are shown.
  Associative reasoning is tapped.
- Rapid, incremental, and reversible actions.
- Typing replaced, in so far as possible, by pointing and selecting.
- Results of actions visible immediately.

Microsoft Windows is the most common Graphical User Interface in use world-wide, provided as standard on most new personal computers sold. Microsoft announced in early 1994 that over 50 million copies had been shipped at that time. Microsoft Windows is therefore a natural choice as the software platform for any modern special purpose simulator.

One advantage of adopting such a standard user interface is that a user with experience of other applications will find that it takes very little instruction to learn to use a new application. Many users, in fact, find the best way to learn a new application is by a short period of trial and error experimentation, rather than reading the user manual. Another distinct advantage is the broad availability of excellent software development tools.

Microsoft Windows has a very bright future. Windows based technologies like DDE (Dynamic Data Exchange) and OLE (Object Linking and Embedding) allow applications to work in a cooperative fashion never before seen in the PC world. Currently dominant only on the Intel based PC platform, Windows NT promises to place the Windows interface on an ever increasing number of hardware platforms. On the PC also, the next version of Windows, Windows 95 (Chicago) offers many attractive features such as an improved interface, true 32 bit operation, improved crash protection, better networking support and PnP (Plug and Play) to allow much easier hardware upgrading. All of these factors, combined with Windows huge user base, look set to assure its dominance and provide promising indications of its future development.

2.3 Software Development Tools

Programming for GUI-based software used to be a very involved process, requiring the user to do considerably more than coding the algorithm which described the core operation of their program. The programmer was required to produce a large volume of code simply to describe the appearance of their application and have it interface properly with the GUI environment, all before any real features were added.

All GUI systems have what is referred to as an application programming interface (API). An API is a set of programming functions available to assist the programmer in programming windowing applications. These APIs provide basic services for creating and displaying windows and widgets and monitoring user input.

Programming for GUI's used to involve calling API functions directly. Modern GUI software development tools tend to remove the need for the programmer to access the API functions directly. They automatically take care of many of the tasks, such as defining a window and making it appear on screen, which would have had to be done by the programmer previously.

Microsoft Visual Basic is an excellent example of a modern software development tool. Visual Basic essentially provides an abstraction of the MS Windows API, ensuring the programmer has no need to learn the many hundreds of API function calls.

Visual Basic allows the programmer to design the interface in a very user-orientated fashion. The appearance of the program's interface is resolved first. The interface objects, such as button, lists, menus, etc., are placed on the program's windows, as if with a drawing package. Up to this point no code needs to be written. This saves time and effort as compared to a traditional programming language, where many hundred lines of code would be necessary simply to invoke the windows and draw the widgets on them and the layout could not be designed interactively. For these reasons, Visual Basic makes an excellent software prototyping tool, as the look and feel of a final application can be designed very rapidly and presented to the end users for evaluation.

Once the programmer is satisfied that the user interface is clearly designed, they can proceed to actually coding the application. The programmer then attaches code fragments to the various objects (such as buttons and menus) which describe their handling of various system events (such as

a mouse click or key press). This general approach to programming is known as an *event-handling approach* and is characteristic of systems with graphical user interfaces where the program does not usually run directly from start to completion, but rather waits for user interaction before proceeding. In Visual Basic the code fragments mentioned above are in the BASIC programming language. Visual C++, also by Microsoft, is similar to Visual Basic but the code fragments are in the C++ programming language (Stroustrup 1992).

Another distinct advantage to the use of Visual Basic are the widely available, third-party, Visual Basic extensions (VBXs) (Udell 1994). VBXs are pre-programmed objects and extensions which function as reusable components. VBXs implement such diverse items as 3D-look buttons, scrolling spreadsheet-like grids and communication protocols. Programmers can add VBXs directly to a project and access to these extensions is identical to accessing any standard Visual Basic object. VBXs put considerable power within easy reach of the programmer.

A final advantage of Microsoft Visual Basic is that any changes and additions to the program being developed can be easily made as the software is being developed, allowing the programmer to derive the software in an incremental manner, starting with a few basic features and adding more over time as the need for them appears.

2.4 Simulation Engines

An essential part of any special purpose simulator is the simulation engine. This is the portion of the simulator which performs the actual simulation, as distinct from those portions which are concerned with presenting a friendly interface to the user.

The developer has two basic choices when picking a simulation engine. One obvious option is to write a simulation engine from scratch, either using the same software development tool which is being used to develop the user interface, or some other software development tool better suited to writing a simulation engine. The second choice is to use a robust and well established, commercially available, simulation package and integrate it with the user interface portion.

3 DEVELOPMENT OF A SPECIFIC EXAMPLE

In October 1990, the Irish Electricity Supply Board (ESB), the country's electrical utility, sponsored a project at University College Galway aimed at investigating ways of improving supply continuity in their electrical network. The initial work done on this project was described by Nolan, O'Kelly and Fahy (1993). It was decided that developing a special purpose simulator which modelled the fault location and repair procedure on the network would provide ESB engineers and managers with a very valuable tool for investigating the effects of procedural, personnel and equipment changes on supply continuity. Both the model and front end were developed in close cooperation with ESB engineers. Regular meetings were held throughout the development process and the system underwent an evolutionary process as new features were added to both the front-end and model. Validity of the model was demonstrated by a large number of tests performed with real network and fault data.

It was decided from the outset that the simulation system being developed should be personal computer based. This base offered the greatest design scope and would allow the ESB to distribute the final system as widely as possible. Microsoft Windows was also specified as the operating environment for the simulator and Microsoft Visual Basic as the software development tool for the front end.

With respect to the simulation engine, a number of different approaches were considered. Among the options available were several simulation systems developed "in-house" in UCG based on such simulation algorithms as the ABC method (Ellison and Tunnicliffe-Wilson 1984) and Petri Nets (Molloy 1982). Eventually it was decided to use a robust and well established, commercially available, simulation package. GPSS/H from Wolverine Software (Hendriksen and Crain 1989), was selected as the simulation engine.

GPSS/H is a modern, personal computer based, version of the industry standard GPSS simulation language. GPSS was first used on IBM mainframes in the 1960 and appears today in several different variations on a wide variety of hardware platforms. GPSS/H offers a superset of this original language, automatic statistics gathering, automatic report generation, improved execution speed and several other attractive features.

The specifications for the final tool were:

1. The tool should be easily applicable to real situations and capable of analysing specific area network data.
2. Although the tool will be used by computer literate people, it should be easy and straightforward to use.
3. As the tool will be applied to analyse individual area operations, a graphical display of network information would be required.
4. The model would be driven by actual fault data.
5. All different types of faults which occur on the network should be modelled.
6. The model should take account of varying customer density in an area.
7. The model should be capable of dealing with supply loss through normal voluntary network operation (repair, maintenance and new works) as

well as supply losses from faults.

8. The model should be able to simulate storm conditions using normal fault patterns for storm conditions.
9. It should also be possible to attempt to find answers to the following questions by performing "what-if" experiments with the model:
   A: What is the optimum size for an area?
   B: What is the optimum number of electricians in an area?
   C: What is the optimum location of electricians in an area?
   D: Should particular electricians specialize in repairing particular types of faults under storm conditions?
   E: What would be the effect of using a lower grade of personnel (known as linesmen) in the repair of certain faults?

Figure 1 shows the final layout of the simulator.

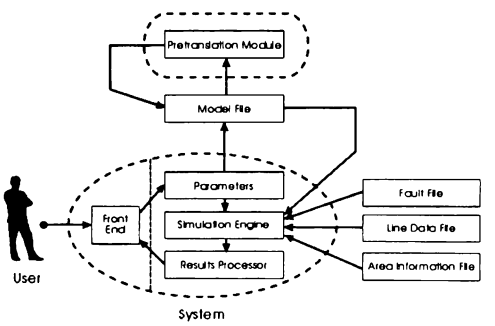

Figure 1: Simulator Architecture

3.1 Model Development

The simulator model models personnel of the ESB Distribution Department in a specific geographical area repairing the faults which occur there in a particular year.

The ESB's power lines fall under the responsibility of two different departments depending on their voltage:

Transmission Department:

Power lines coming from generating stations. This comprises lines of 400kV, 220kV and 110kV. This network feeds the distribution network.

Distribution Department:

High voltage (HV) power lines of 38kV and 10kV and the low voltage (LV) supply of 220V directly into the customer.

An area, in operational terms, is defined by the electrical sub-stations (where the voltage is transformed from 38kV down to 10kV), and the 10kV power lines contained within it. This area is controlled from an area office which is the base for all repair personnel.

The fault location and repair procedure on the high voltage (HV) network can be generalised into five stages:

1. The fault occurs and either triggers an alarm or is reported by a customer, depending on the type of fault.
2. Appropriate personnel (for the particular type of fault) are dispatched.
3. The personnel search for and locate the fault.
4. The fault is repaired.
5. Personnel return to base.

In the first stage, the electricians on duty become aware of the faults. A major branch here distinguishes between automatic (alarm) and customer-reported faults. In the case of customer reported faults, the number and geographical spread of the calls helps the electricians determine the nature of the fault they are dealing with.

The second stage describes dispatching personnel. Two types of repair personnel exist, electricians and linesmen, who are less qualified than electricians. A linesman, if one is available, will be sent in preference to an electrician if they are capable of repairing the particular fault, otherwise an electrician will be dispatched. Depending on the fault type, the personnel may travel first to the appropriate sub-station and check there for an internal fault before checking the power line in question.

Locating the fault is the most complex part of the operation and depending on the fault type, there are a variety of procedures and tools available to the electrician or linesman to assist him in this task. Some of the procedures used here include patrolling the line or sectioning the line (where portions of the line are switched out in an attempt to isolate the portion containing the fault).

Once the electrician or linesman has successfully located the fault and determined its exact nature, he may repair it. Finally, once the repair has been completed, the supply is restored. The fault is now considered over and the electrician returns to base.

The GPSS/H model at the heart of the simulator is based closely on the actual procedure described above. The following parameters can be set by the user:

1. The user can specify the number of electricians and linesmen.
2. The user can specify the average travel speed.
3. The user can freely define the area of operations by choosing a set of sub-stations and power lines.
4. The user can specify the base location.
5. The user can impose a two-day storm on a particular date.
6. It is possible for the user to optionally expand linesmens' responsibilities beyond their current level.
7. Voluntary outages, supply losses through maintenance or repair work, are included, and the user specifies the average number per week.

8. Low voltage outages are included and the user specifies the rate of low voltage to high voltage outages in up to four separate areas.
9. It is possible to specify electricians who specialise in a particular type of fault and a speed increase for them when repairing this fault type.
10. It is also possible to define emergency personnel. These are defined to be a number of personnel from other areas who can be called upon under specified conditions.

The model is run using a mixture of real and computer generated data. The ESB have extensive computer database records of all HV faults and this information is used as an input for the model.

The data for LV and voluntary outages is not computerised however and the system generates this data itself based on information the user supplies about these types of outages. Urban and rural areas have very different ratios of low voltage to high voltage faults. Urban areas have a large ratio of customers to line length and hence a large ratio of low voltage to high voltage faults, the situation is correspondingly reversed in rural areas. The user therefore may indicate up to four districts within the area and specify different LV:HV ratios in each.

The model also provides the facility to impose a two-day storm on the model starting on a user-specified date. The storm can be either of an electrical, wind or mixed type, each of these types generating a distinctive pattern of faults.

The user also specifies the conditions under which emergency personnel are called in. Once a certain number of HV faults are left unanswered (as during a storm when all personnel in the area are busy), the emergency personnel are called in. Once the number of unanswered faults falls below another specified threshold, the emergency personnel are released.

3.2 Front End

Once the simulator is run, the main screen shown in Figure 2 appears.

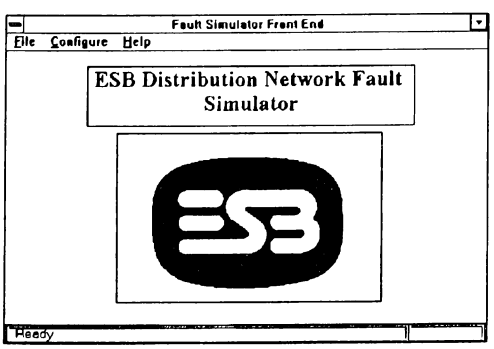

Figure 2: Main Screen

Full on-line hypertext help using the Windows help system is available within the simulator by selecting Help from the menu. The contents page of the help file is shown in Figure 3.

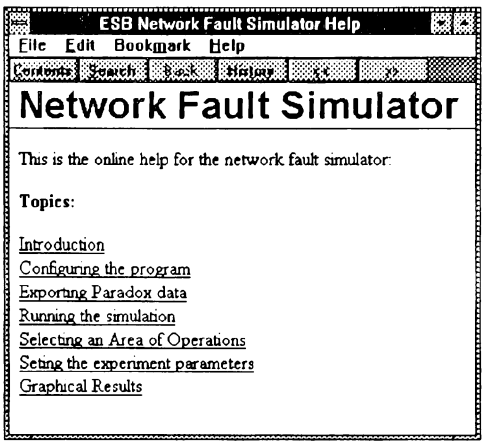

Figure 3: Online Help

The user chooses the Open option from the File menu and selects an input file for the simulation. The simulator scans the input file and, once it is satisfied it is in a valid format, ascertains which 38kv Stations and outlets (power lines) from these stations have faults pertaining to them contained within the data file. The user is then free to specify their area of operations by choosing, from the available stations and outlets, those to be included within the area. Figure 4 shows the Area of Operations screen. Placing the station or outlet within the area of operations is simply achieved by selecting it on the left in the available list and clicking on the ADD button to place it on the right hand list. Individual line extensions can be chosen independently of their stations and additionally, portions of the Backbone Line (BBL) can be added to the area by specifying a range of poles as seen in the lower portion of Figure 4. Four combinations are possible here:

From Start to End
From Specified Pole to End
From Specified Pole to Specified Pole
From Start to Specified Pole

Any elements accidently added to the right hand side can by removed by selecting them and clicking the remove button. The user clicks DONE once ready to proceed.

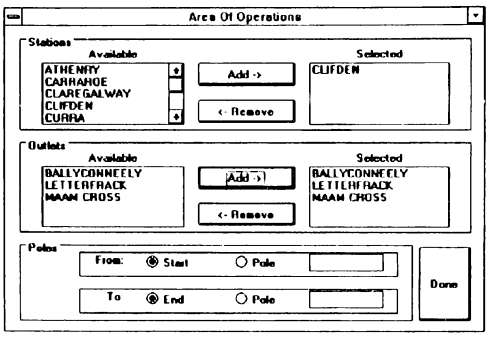

Figure 4: Area of Operations Screen

Once the area of operations has been specified by the user, a second window, as shown in Figure 5, opens showing a map of the selected area.

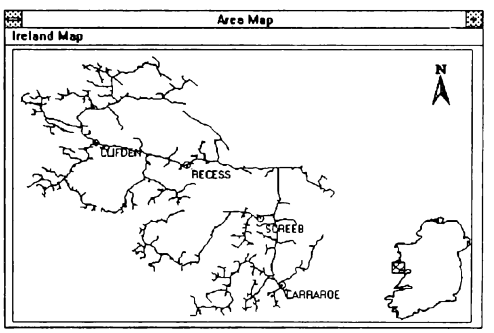

Figure 5: Map of the Area of Operations

The user is then asked to specify the rest of the simulation parameters. Figure 6 shows the Parameters screen. The base location can be entered directly as a set of grid co-ordinates or chosen from a list of known 38kV station locations. To impose a two-day electrical, wind or mixed storm at a given time, the relevant boxes are ticked and the start time in days and months is entered.

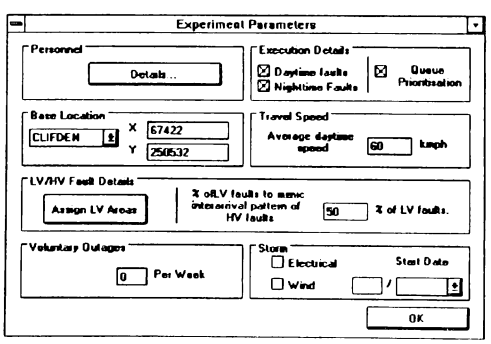

Figure 6: Parameters Screen

Clicking on the Personnel Details button presents the user with the dialog box shown in Figure 7.

Here the user can enter the details for normal and specialist electricians and linesmen. The user can also specify the numbers of emergency personnel and the conditions under which they are released or reclaimed. The percentage of LV faults which a linesman can answer is set here. It is also possible to specify a percentage repair speed increase for a specialist electrician when he is handling his own specialised fault type.

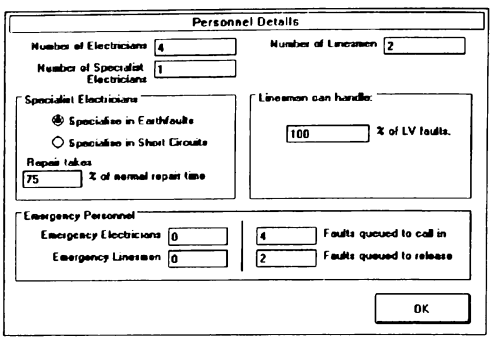

Figure 7: Expanded Personnel Parameters

Up to four areas with different ratios of Low Voltage (LV) to High Voltage (HV) faults may be specified by the user. Assigning the LV areas is done by pressing the LV Areas button visible in Figure 6. Provided the user does not choose to reload old settings, the map appears as in Figure 8 and the user may indicate the LV areas. The ratios cannot be set until after the translation stage because at this point the number of HV faults in these areas is unknown.

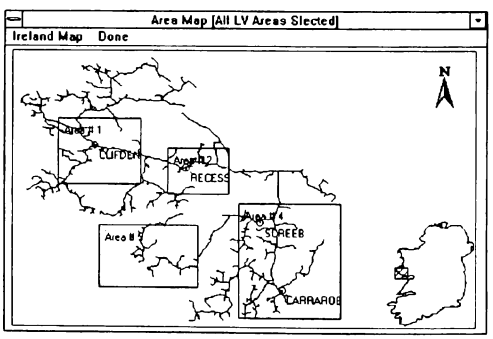

Figure 8: Assigning the LV Areas

After all the parameters have been entered the program then processes the input data to provide a processed input file for GPSS/H. While it processes the data a screen is shown tracing the program's progress through the data file.

Now that the program has established the number of HV faults in each area, the user can specify the ratio of HV to LV faults within those areas. This is accomplished using the form shown in Figure 9. The overall figure for the area is shown at the top so that the user can adjust the parameters in a realistic fashion.

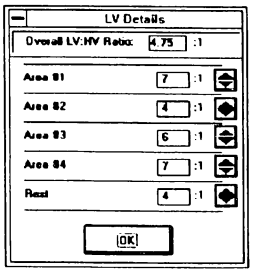

Figure 9: Setting the LV:HV Fault Ratio

The simulation model is then run. The program waits for the GPSS/H process to finish running. There is no interaction between the front end and the simulation engine while the simulation is being performed. Once the simulation is complete the program automatically moves on to processing the results.

The program analyses the output files from the simulation and displays the results graphically on the results screen. The two values used to assess continuity are the length of the outage and the number of customer hours lost. The number of customer hours lost are defined as the length of outage in hours, multiplied by the number of customers affected.

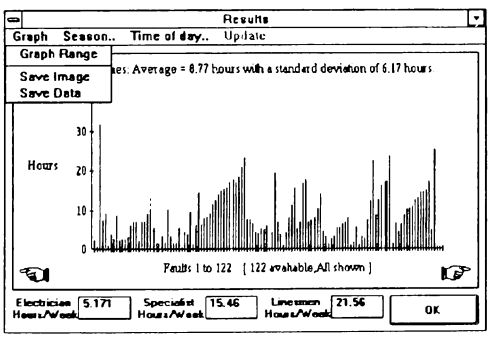

Figure 10: Results Screen

Six graphs are displayed on the results screen, a graph of outages times for each HV fault, a histogram of the distribution of outage times, and a graph of customer hours lost for each HV fault, a histogram of the distribution of customer hours lost for each HV fault, a pie-chart showing the winter/summer, day/night distribution of HV faults and finally a pie-chart showing the number of faults per month for the particular year. Switching forwards and backwards between graphs is accomplished by clicking intuitively on finger-shaped icons located at the bottom right and left corners of each graph. The results screen with the menu activated is shown in Figure 10.

The user is presented with three options in the menu. The first of these is an option to reset the range of the graph to enable the user to zoom in on particular sections. When viewing the outage times or customer hours lost graphs the user can specify a start and end value, leaving the either field empty causes its value to default to the limit. For the case of the frequency histograms the user can choose to discard out-liers after a specified value.

The other two options on the menu allow the user to save the results data. The first allows an image of the visible graph to be saved as a Windows metafile (a vector-based graphics format) for import into other applications (e.g. Microsoft Word). The second is enabled when viewing the outage times and customer hours lost graphs and allows the user to save these graphs' data to file in a numeric form again for import into another application (e.g. Microsoft Excel).

After the results screen, one final screen presents a text based report of the simulation. This can be seen in Figure 11. The report includes all relevant information about the particular simulation including the value of the experiment parameters, the area of operations, the location of any data saved to file, the results, etc.

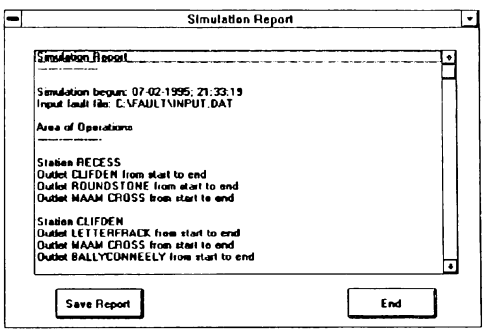

Figure 11: Report Screen

After examining the report the user presses the END button and is given the option of ending the program or performing another run.

The simulator can be usefully integrated with other Windows applications to provide extra functionality. The simulator produces graphs of the results of a simulation run which can be saved as Windows metafiles and imported into reports and other documents. The raw results data is also available and can be imported into other applications so that it can be presented in alternative formats. Microsoft Excel was used to produce graphs of cumulative hours lost over the year. To produce these results, the user could follow the following steps:

1. Import the file containing the start time of each fault into column one of the spreadsheet.
2. Import the file containing the customer hours lost per fault into the second column of the spreadsheet.
3. The two files are ordered in terms of the time the faults were cleared. Sort them in numerically ascending order based on column one to place them in true chronological order.
4. Fill the rest of column three with a formula which places in each row the sum of the value above it and the value

to its left (in column two). This gives you the cumulative customer hours lost in column three. [In Excel the formula would be =(C1+B2) for the first cell and then Copy could be used to transpose it to the other cells in column three]

5. Plot column three against column one.
   An example of this type of plot is shown in Figure 12.

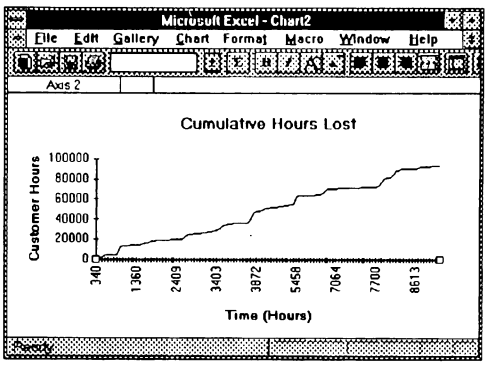

Figure 12: Cumulative Hours Lost Plot Produced in Excel

4 CONCLUSIONS

This paper has shown the merit of developing special purpose simulators under Microsoft Windows. Using a GPSS/H model with a Microsoft Windows front end resulted in a simulator which was shown to be both powerful and easy to use and also was comparatively quick to develop. The prime advantage of this and similar systems is that they empower users at a management level to use simulation as an everyday decision-making tool.

REFERENCES

Ellison, D., And Tunnicliffe-Wilson, J. C. 1984. *How to Write Simulations Using Microcomputers*. New-York: McGraw-Hill.

Hendriksen, J. D., and Crain, R. C. 1989. *GPSS/H Reference Manual*. Annadale, Virginia: Wolverine Software Corporation.

Molloy, M. K. 1992. Performance Analysis using Stochastic Petri Nets, *IEEE Trans. Computers*, C-31: 92-122.

Nolan, P. J., O'Kelly, M. E. J., and Fahy, C., 1993, Assessment of Ways of Improving the Supply Continuity in Electricial Power Systems - A Simulation Approach. In: *Proceedings of the 1993 Winter Simulation Conference*, eds. G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles, 1192-1200. La Jolla, California: Society for Computer Simulation.

Pegden, C. D. 1982. *Introduction to Siman*. State College, PA: Systems Modelling Corporation.

Seppanen, M. S. 1990. Special Purpose Simulator Development. In: *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R.P. Sadowski, and R.E. Nance, 67-71. La Jolla, California: Society for Computer Simulation.

Schniderman, B. 1992. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Second Edition. Massachusetts: Addison-Wesley.

Smith, D. S., and Crain, R. C. 1993. Industrial Strength Simulation Using GPSS/H. In: *Proceedings of the 1993 Winter Simulation Conference*, eds. G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles, 165-171. La Jolla, California: Society for Computer Simulation.

Stroustrup, B. 1986. *The C++ Programming Language*. New Jersey: Addison-Welsey.

Udell, Jon. 1994, ComponentWare. *BYTE*, 19: 46-56.

AUTHOR BIOGRAPHIES

KIERAN L. COUGHLAN received the B.E. and M.Eng.Sc. degrees in Mechanical Engineering at University College Galway. He is currently working as a computer support engineer in B. Braun Hospicare Ltd. in Co. Sligo, Ireland.

PAUL J. NOLAN is a Statutory Lecturer in Mechanical Engineering at University College Galway. He received the B.E. degree from University College Dublin and the M.Eng. and Ph.D. degrees from McMaster University (Canada) in Electrical Engineering. His research interests include discrete and continuous time simulation and control systems.