Linking the Witness Simulation Software to an Expert System to Represent a Decision–Making Process

Stewart Robinson*, John S. Edwards** and Wu Yongfa***

*Operational Research and Systems Group, Warwick Business School, University of Warwick, Coventry, United Kingdom
**Operations and Information Management Group, Aston Business School, Aston University, Birmingham, United Kingdom
***Department of Management Engineering, South East University in Nanjing, Jiangsu, P.R. China

Expert systems, and artificial intelligence more generally, can provide a useful means for representing decision-making processes. By linking expert systems software to simulation software, an effective means of including these decision-making processes in a simulation model can be achieved. This paper demonstrates how a commercial off-the-shelf simulation package (Witness) can be linked to an expert systems package (XpertRule) through a Visual Basic interface. The methodology adopted could be used for models, and possibly software, other than those presented here.

Keywords: simulation, artificial intelligence, expert systems, decision making processes.

1. Introduction

The majority of simulation models contain at least one, and probably more, decision-making processes. These might be computer-based (e.g. a production control system), human (e.g. a branch manager in a bank) or a mixture of the two (e.g. computer based decision support systems often rely on human intervention to suggest potential improvements). Such decision-making processes are often modelled using the commands available in the simulation software. Some, however, have adopted artificial intelligence methods as a means of representation, for instance, Flitman and Hurrion (1987), O'Keefe (1989), Williams (1996), Lyu and Gunasekaran (1997) and Robinson et al. (1998). These give the advantage of providing a well developed framework for representing, and eliciting, knowledge (Doukidis and Angelides, 1994).

The purpose of this paper is to describe how a commercial off-the-shelf (COTS) simulation package, Witness (Lanner Group, 2001), can be linked to an off-the-shelf expert systems package, XpertRule (Attar Software, 2000), in order to provide a means of representing a decision-making process. The approach is demonstrated by reference to an example model. The aim is to illustrate how the two packages can be linked successfully. Albeit that the example presented is fairly simple, the approach provides the basis for further work on more complex problems.

The paper starts with a brief review of previous work in which simulations and expert systems have been linked. After this, the example simulation model is described and the representation of the decision-making process in XpertRule is explained. An overview of the methodology for linking Witness and XpertRule is given, as well as a detailed description of the Visual Basic code required to link the two software packages. A full listing of this code is provided in the appendix.

2. Previous Work Linking Simulations with Expert Systems

Linking a simulation with an expert system provides a number of technical challenges, to the extent that some have even relied upon a manual interface (Nissen, 1998). Flitman and Hurrion (1987) were among the first to propose the approach, linking a simulation written in

Fortran on one computer with a Prolog-based expert system on another computer. Similarly, O'Keefe (1989) developed a tool that linked GPSS with Prolog.

Some have taken the approach of developing architectures for linking expert systems with simulation that do not rely upon specific simulation or expert systems software. Artiba and Aghezzaf (1997) developed an architecture for linking expert systems, simulation, optimisation algorithms and heuristics for looking at complex production planning and scheduling problems. Zeigler et al. (1996) embed an expert systems shell within the DEVS environment.

Others have linked specific simulation packages with expert systems. Lyu and Gunasekaran (1997) achieve this by developing both a simulation model and expert system within SIMSCRIPT II.5 for modelling harbour unloading by a steel company. They note that this is only effective when the rules are sufficiently simple to be represented within the simulation software. Standridge and Steward (2000) link SLAMSYSTEM with an expert system written in C. Neither of these represent the use of a standard simulation software package with a standard expert systems package, as described in this paper.

The contribution of the current work is to show how commercial off-the-shelf software can be linked with reference to two specific packages, Witness and XpertRule. As the functionality of such software continues to grow, the possibility for using them in collaboration will increase. As such, both researchers and practitioners should find the method for linking such software described in this paper of interest.

3. Example Model

A model of a fictional truck loading bay (Figure 1) was developed in the Witness simulation package (Witness 2001 release 1.0). Trucks arrive at the truck park at an average interval of 10 minutes (based on a negative exponential distribution) and require loads of between 5 and 20 items (uniformly distributed). On arrival, the trucks are allocated to a loading bay, should a suitable one be available. In making this decision, account must be taken of the restrictions on the bay capacities. Trucks requiring more than 10 items must be allocated to bay 2 or 3, since bays 1 and 4 only have capacity for up to 10 items. Should a bay not be available, then the truck waits in the park until a suitable bay becomes available. Once a truck is allocated, it moves to the bay where it is loaded before departing from the system. Trucks take 1 minute to move to the loading bay where each item takes 1 minute to be loaded.

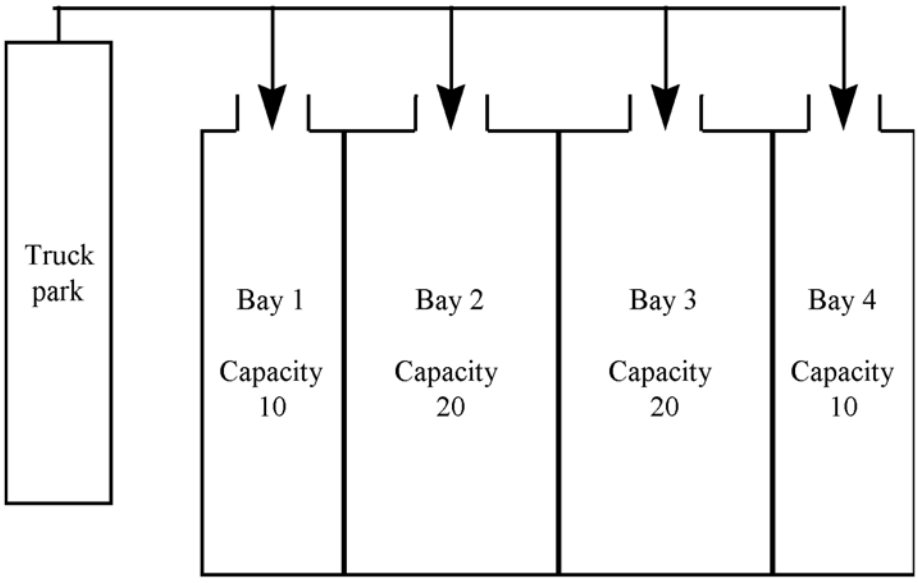


Fig. 1. Truck Loading Bay Example.

4. The Expert System

A rule-based expert system was developed in XpertRule (version 3.92, 32-bit) to represent the decision to allocate trucks to lanes. The resulting decision tree is shown in Figure 2. There are five key attributes to the decision: the size of the truck ('Truck_Size') in terms of the number of items to be loaded; and the current allocation of trucks to lanes, denoted by the attributes 'Lane1' to 'Lane4'. If no truck is presently allocated to a lane, the lane attribute is set to zero, otherwise it is set to the number of the truck that is allocated to that lane (note that truck numbers denote the sequence in which the trucks arrive in the simulation model). A third attribute 'Lane_for_Truck' reports the outcome of the decision made by the expert system. This is set to the number of the lane that a truck is to be allocated to, or to zero if no allocation can be made.

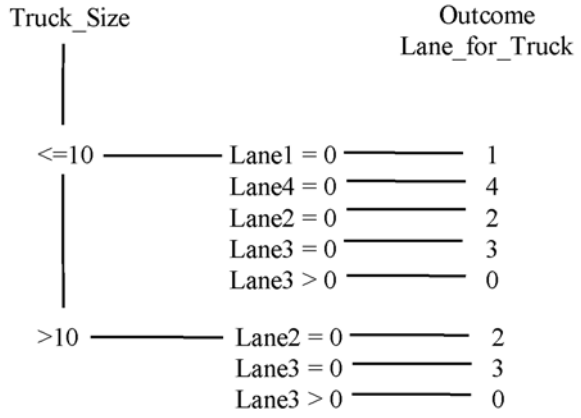


Fig. 2. Truck Lane Allocation Decision Tree.

The decision tree shows that the allocation decision rests primarily on the size of the truck. If this is less than or equal to 10 items, then the truck can be allocated to any one of the lanes. An attempt is made to allocate the truck to the smaller lanes first (lanes 1 and 4). Trucks that require more than 10 items can only be allocated to lanes 2 and 3.

In this example, because the decision is simple, the decision tree has been developed by a simple process of deduction as to what would entail a sensible decision-making process. Indeed, such is the nature of this decision that it could be coded directly in most simulation software. One useful extension to this approach, however, is to use the simulation as a means of knowledge elicitation. At a decision point the simulation would stop and wait for input. Expert decision-makers could enter their decisions at each decision point. In this way, a history of cases and decisions could be developed, which could then be used to induce rules within the expert system. In more complex examples, this approach has the significant advantage that example decisions can be obtained faster than real time with reproducible conditions. This approach may prove more reliable than interviewing experts, particularly when they are unable to express clearly how they make decisions. The latter is a well-known problem in developing knowledge-based systems; see for example Kidd (1987).

5. Methodology for Linking Witness with XpertRule

The simulation model and expert system were linked using Object Linking and Embedding (OLE) within Windows NT version 4.0. The obvious approach would have been to make calls to XpertRule from Witness as and when a decision point is reached (a truck arrives or a lane becomes free). This, however, is not possible, since Witness only acts as an OLE slave. As a result, it was necessary to develop a model controller in Visual Basic (Figure 3). The model controller initiates the run of the simulation model. At a point where an allocation decision is required, the simulation model automatically stops and waits until the model controller returns a decision and continues the simulation run. Once the model controller has

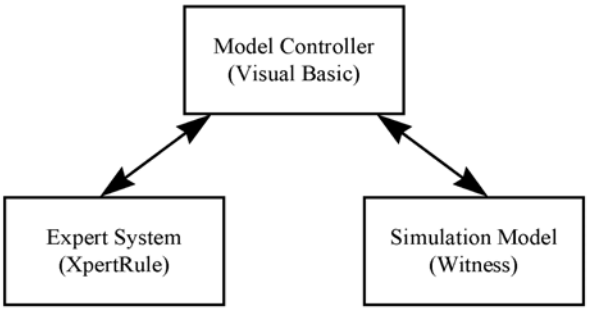


Fig. 3. Linking Witness to XpertRule.

detected that the model is not running, it extracts data from the model which it passes to the expert system for a decision. The decision is returned to the simulation model via the model controller. Some effort was required to ensure that this sequence of events was adherred to. The code required to achieve this is described below.

6. Witness Requirements

Within Witness, two specific actions are required. First to stop the model when a decision point is reached. This occurs either when a truck arrives in the truck park and so it will look for a lane to be allocated to, or when a truck leaves the loading bay and so a lane becomes free and a truck may be allocated to it. The model is stopped by issuing the 'Stop' command within the Witness actions language at these two decision points.

The second action that is required is to update the variables and attributes within the Witness model, so that if a truck has been allocated, it moves to its loading bay lane. This takes place once a decision has been taken and the model has been restarted from the model controller. To enable this, a dummy process ('machine') is defined within Witness. At the point when the model is stopped a toggle variable is set to 1, this triggers the dummy process to start by pulling a dummy part from the 'world'. As a result, the dummy process is the next event that will occur when the model restarts. In the actions on the dummy process, assuming a lane has been allocated to a truck (the Witness variable 'allocate_lane'>0), all the relevant variables and attributes are updated and the toggle variable is reset to zero.

7. Visual Basic Code

The Visual Basic code, for which a full listing is provided in the appendix, was developed in version 6.0 (32-bit). A flow chart outlining the code is shown in Figure 4. The numbers in brackets refer to the segments of code identified in the appendix. The details of the code, relating to each segment, are described below. Before proceeding, it should be noted that no specific modifications were needed to make the XpertRule application compatible with the model controller.

7.1. On Loading Visual Basic (VB): Sub-Routine Form_Load

The Form_Load routine (appendix segment 1) is invoked when the VB program is started. At this point, Witness and the Witness model 'truck.mod' are loaded via the shell command, the integer value referring to the mode in which the software is run, in this case minimised. Following this, an XpertRule object is created and the specific expert system ('truck.xra') is loaded. Note that the run-time version of XpertRule is loaded, known as XpertRun or 'xr32run'.

7.2. On Running the Model: Sub-Routine RunModel_Click

The 'Run Model' button invokes the RunModel_Click sub-routine. This performs the various functions that link the simulation and the expert system. It starts the Witness model running and detects when it has stopped. The variables required to make the decision in XpertRule are extracted from Witness and input to XpertRule. A decision is then obtained from XpertRule and returned to Witness.

Simulation Model Control
(appendix segment 2)

The variable 'continue' controls the main loop for this sub-routine. While it is set to 1, the Witness model continues to run until the next decision point, a decision is made and the model is run on. The second line of the code assigns the Witobj object to the loaded version of Witness, and the simulation is run through the Witobj.Run command. Once the Witness model is running, the VB code enters a DoEvents loop until it is detected that the Witness model is stopping (Modelstatus = 1) or has stopped (Modelstatus = 3). This implies that the simulation has reached a decision point, that is, a 'Stop' command has been issued in the Witness actions language.

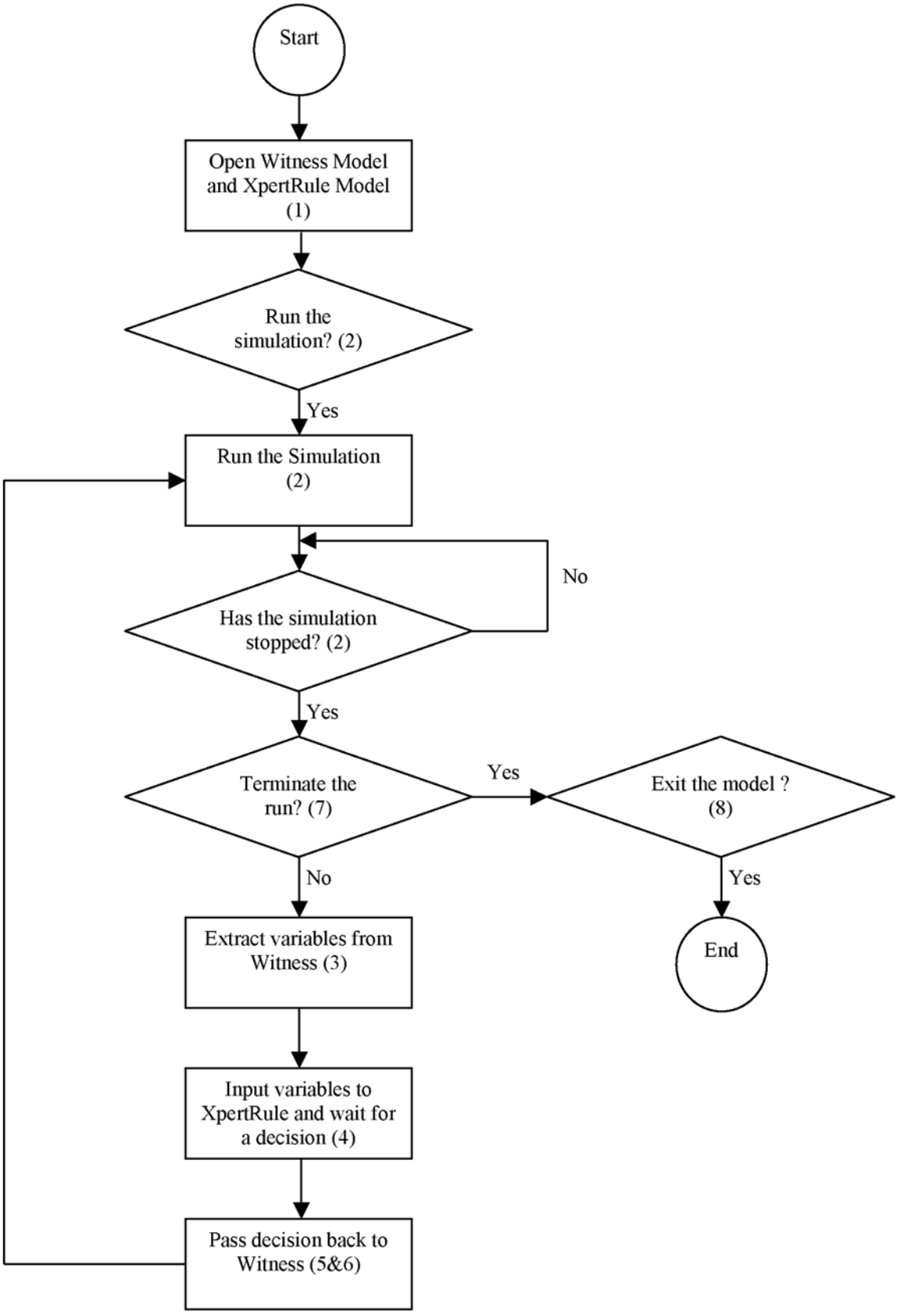


Fig. 4. Flow Chart of the Visual Basic Code (Segments of the Code Shown in the Appendix are Identified in Brackets).

Extract Variables from Witness (appendix segment 3)

The total number of trucks in the truck park is determined by using the Witness function 'NPARTS'. This is necessary because an attempt will be made to allocate a lane to each truck, in turn, that is waiting in the truck park. The allocation status of each lane is extracted from the Witness variable 'allocated', and the size of each truck in the truck park is obtained from the Witness attribute 'loadsize'.

Input Variables to XpertRule and Make a Decision (appendix segment 4)

For each truck, in turn, the five decision attributes (the allocation status of the four lanes and the size of the truck) are passed to XpertRule using the Poke method. The 'CONTINUE' command is issued to XpertRule in order to obtain an allocation decision. The VB code then enters a loop until XpertRule has reached a conclusion concerning the decision, that is, the software status is 'READY'. At this point the result is obtained from XpertRule. "Lane_for_Truck" will report a value between 1 and 4 if a lane has been allocated, otherwise it will be set to a value of zero if no allocation is possible.

Return Decision to Witness (appendix segment 5)

If a lane has been allocated to a truck (the decision is non-zero), then the decision is passed to Witness by setting two variables. The 'allocate_lane' variable holds the number of the lane to be allocated. The 'allocate_truck' variable holds the number of the truck that is to be allocated to that lane. Once an allocation has been made, no further allocations are possible. This is because an allocation decision is triggered by a single state change in the simulation, that is, a truck arrives or a lane becomes available. As a result, no further attempt is made to allocate any other trucks in the truck park, and the code jumps to the 'endalloc' label.

On Failure to Allocate a Truck (appendix segment 6)

If, having attempted to allocate all the trucks in the truck park to a lane, no allocation decision is possible, then return a zero decision to Witness.

7.3. Stopping the Model: Sub-Routine StopModel_Click

The while loop in the RunModel_Click subroutine will continue to loop between running the Witness model, stopping the model at a decision point, obtaining a decision and returning it to Witness. In order to interrupt this loop the variable 'continue' needs to be set to zero. This is done in the StopModel_Click sub-routine (appendix segment 7), which is invoked on clicking the 'Stop Model' button. It should be noted that, having clicked on the 'Stop Model' button, the Witness model will not stop running until it has reached the next decision point. The simulation model may be restarted by clicking on the 'Run Model' button.

7.4. Exiting the Model: Sub-Routine ExitModel_Click

Should the model user wish to leave the simulation altogether, then by clicking on the 'Exit Model' button the code within the ExitModel_Click sub-routine is invoked (appendix segment 8). This closes both the Witness and XpertRule packages.

8. Conclusion

Although the VB code presented here is aimed for a specific Witness simulation model and a specific XpertRule application, the basic structure would be similar for any Witness model or XpertRule application. Indeed, it should simply be a case of replacing the relevant variable and attribute names with those required for a different model, as well as possibly changing the For Next loop that attempts to allocate each truck in turn, to reflect the particular decision-making process. Although the model presented here is simple, a larger and/or a more complex model is only likely to result in the need to extract and pass more variables between the software. Indeed, continuing work, looking at human decision-making around maintenance operations in a Ford manufacturing plant, shows that this is the case (Robinson et al, 2001). Also, the code could probably be adapted for use with other simulation software or artificial

intelligence systems, as long as these could act as an OLE slave.

Another requirement might arise if more than one decision-making process is present within the same simulation model. This would require some adaptation to the approach described above. A separate XpertRule model would need to be developed for each decision-making process, and then run in parallel during the running of the simulation. In terms of the Visual Basic code, this would mean that n XpertruleObjects would need to be defined, where n is the number of decision-making processes. Depending on which decision is being taken, the relevant variables would have to be extracted from Witness, the correct object would have to be invoked and the relevant variables returned to Witness. Unless the variables involved in the decision are exactly the same, this would require separate segments of code for each decision-making process. The other requirement would be to identify, which decision-making process had caused the simulation to stop. This could simply be identified by setting a flag in Witness, to a value between 1 and n.

The linking of the simulation and expert system would have been made easier if Witness could act as an OLE client. This would have removed the need for the VB model controller. It would also result in the model running faster by removing the overhead of the model controller. The use of Witness could, therefore, be questioned. Why not use an alternative package that can act as an OLE client? For many, including the authors, this decision can only be considered when first deciding which simulation software to purchase. The costs of replacing an existing package are simply too great. The approach described above provides a useful method for linking Witness, and probably other simulation software that cannot act as OLE clients, to another software package.

Acknowledgements

The authors would like to acknowledge the financial support given by the British Council's Sino-British Friendship Scholarship Scheme and by the Lanner Group.

References

[1] ARTIBA, A. AND AGHEZZAF, E.H. (1997). "An Architecture of a Multi–Model System for Planning and Scheduling." *International Journal of Computer Integrated Manufacturing*, 10 (No. 5), pp. 380–393.

[2] ATTAR SOFTWARE (2000). *XpertRule KBS Reference Manual*. Attar Software, Leigh, UK.

[3] DOUKIDIS, G.I. AND ANGELIDES, M.C. (1994). "A Framework for Integrating Artificial Intelligence and Simulation." *Artificial Intelligence Review*, 8, pp. 55–85.

[4] FLITMAN, A.M. AND HURRION, R.D. (1987). "Linking Discrete–Event Simulation Models with Expert Systems." *Journal of the Operational Research Society*, 38 (No. 8), pp. 723–734.

[5] KIDD, A.L. (1987) *Knowledge Acquisition for Expert Systems: A Practical Handbook*. Plenum Press New York.

[6] LANNER GROUP (2001). *Witness 2001*. Lanner Group, Redditch, UK.

[7] LYU, J. AND GUNASEKARAN, A. (1997). "An intelligent simulation model to evaluate scheduling strategies in a steel company." *International Journal of Systems Science*, 28 (No. 6), pp. 611–616.

[8] NISSEN, M.E. (1998). "Redesigning Reengineering through Measurement–Driven Inference." *MIS Quarterly*, 22 (No. 4), pp. 509–534.

[9] O'KEEFE, R.M. (1989). "The Role of Artificial Intelligence in Discrete–Event Simulation." In: Widman, L.E., Loparo, K.A. and Neilsen, N.R. (eds). *Artificial Intelligence, Simulation and Modeling*. Wiley: New York, pp. 359–379.

[10] WILLIAMS, T. (1996). "Simulating the Man–in–the–Loop." *OR Insight*, 9 (No. 4), pp. 17– 21.

[11] ROBINSON, S., EDWARDS, J.S. AND YONGFA, W. (1998). "An Expert Systems Approach to Simulating the Human Decision Maker." *Winter Simulation Conference 1998* (D.J. Medeiros, E.F. Watson, M. Manivannan, J. Carson, eds.), The Society for Computer Simulation, San Diego, CA, pp. 1541–1545.

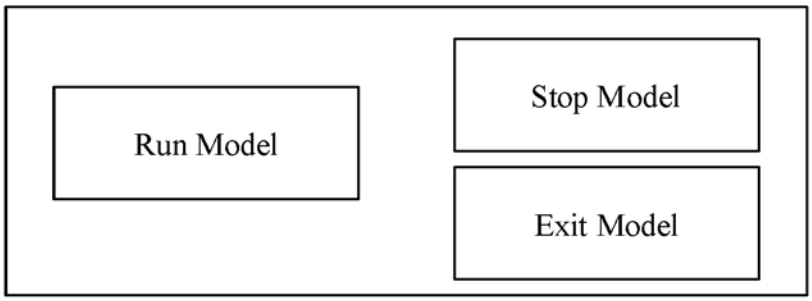
[12] ROBINSON, S., ALIFANTIS, A., EDWARDS, J.S., HURRION, R.D., LADBROOK, J. AND WALLER, T. (2001). "Modelling and Improving Human Decision Making with Simulation." *Winter Simulation Conference 2001* (B.A. Peters, J.S. Smith, D.J. Medeiros, M.W. Rohrer, eds.), The Society for Computer Simulation, San Diego, CA, pp. 913–920.

[13] STANDRIDGE, C.R. AND STEWARD, D. (2000). "Using Expert Systems for Simulation Modeling of Patient Scheduling." *Simulation*, 75 (No. 3), pp. 148–156.

[14] ZEIGLER, B.P., CHO, T.H. AND ROZENBLIT, J.W. (1996). "A Knowledge–Based Simulation Environment for Hierarchical Flexible Manufacturing." *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 26 (No. 1), pp. 81–89.

Appendix: Full Listing of Visual Basic Code

The Form



The Code

```
Dim Witobj, XpertruleObject As Object
Dim sheller As Integer
Dim load_size(100), allocated(4), truck_count As Integer
Dim continue, loop1, index As Integer
Dim result As String
```

```
Private Sub Form_Load()

' Load Witness
sheller = Shell("c:\witness\witness.exe c:\witness\truck.mod", 2)
' Load XpertRun
Set XpertruleObject = CreateObject("xrclient.xr32run")
Call XpertruleObject.StartApp("c:\witness\truck.xra -SERVER -MINIMIZED")

End Sub
```

Segment 1

```
Private Sub RunModel_Click()

Continue = 1
Set Witobj = GetObject(, "WITNESS.WCL")
' Run the Witness model
While continue = 1
 Witobj.Run
 ' Check whether the Witness model has stopped
 While Witobj.Modelstatus = 1 Or Witobj.Modelstatus = 3
  DoEvents
 Wend
 ' When the model has stopped if there are trucks waiting to be allocated
 ' obtain the required variables and attributes from Witness
 truck_count = Witobj.expression("NPARTS(PARK)")
 If truck_count > 0 Then
  For loop1 = 1 To 4
   index = loop1
   allocated(loop1) = Witobj.variable("allocated", index)
  Next
  For loop1 = 1 To truck_count
   load_size(loop1) = Witobj.expression("PARK@" + Str(loop1) + ":loadsize")
  Next
  ' For each truck in turn call XpertRule and make an allocation decision
  ' Continue until a truck is allocated to a lane
  For loop1 = 1 To truck_count
   Call XpertruleObject.Poke("Lane1", allocated(1))
   Call XpertruleObject.Poke("Lane2", allocated(2))
   Call XpertruleObject.Poke("Lane3", allocated(3))
   Call XpertruleObject.Poke("Lane4", allocated(4))
   Call XpertruleObject.Poke("Truck_Size", load_size(loop1))
   Call XpertruleObject.Command("CONTINUE")
  ' Loop until XpertRule has obtained a result
   Do
    Call XpertruleObject.Peek("?", a$)
   Loop Until a$ = "READY"
 ' Obtain result from XpertRule
   Call XpertruleObject.Peek("Lane_for_Truck", a$)
```

Segment 2

Segment 3

Segment 4

```
' If a decision is made return it to Witness
  If a$ <> "0" Then
   Witobj.variable("allocate_lane") = Val(a$)
   Witobj.variable("allocate_truck ") = Witobj.expression("PARK@" +Str(loop1)+":truckno")
   GoTo endalloc
  End If
 Next
' If all attempts to allocate have failed return a 0 result to Witness
 Witobj.variable("allocate_lane ") = 0
 Witobj.variable("allocate_truck ") = 0
endalloc:
 End If
Wend
End Sub
```

Segment 5

Segment 6

```
Private Sub StopModel_Click()

Continue = 0

End Sub
```

Segment 7

```
Private Sub ExitModel_Click()

If continue = 0 Then
 Call XpertruleObject.Command("EXIT")
 Witobj.WCL ("QUIT;")
End If

End Sub
```

Segment 8

Received: November, 2002
Revised: May, 2003
Accepted: May, 2003

Contact address:

Stewart Robinson
Operational Research and Systems Group
Warwick Business School
University of Warwick, Coventry
CV4 7AL
United Kingdom

John S. Edwards
Operations and Information Management Group
Aston Business School
Aston University
Birmingham
B4 7ET
United Kingdom

Wu Yongfa
Department of Management Engineering
South East University in Nanjing
Jiangsu
P.R. China

STEWART ROBINSON is a Senior Lecturer in Operational Research at Warwick Business School in the UK. He holds a BSc and PhD in Management Science from Lancaster University. Previously employed in simulation consultancy, he supported the use of simulation in companies throughout Europe and the rest of the world. His research interests are in finding ways to improve the use of operational research, and specifically simulation, within industry. He is author/co–author of two books on simulation. His current work involves an investigation into the use of expert systems to represent a human decision–maker in simulation models, the analysis of complex output behaviours from simulations, and developing an understanding of quality in relation to operational research and simulation studies.

JOHN S. EDWARDS is Professor of Operational Research and Systems at Aston Business School, Birmingham, UK. He holds MA and PhD degrees (in mathematics and operational research respectively) from Cambridge University. His principal research interests are knowledge–based systems and decision support systems, especially methods for their development. He has written more than 30 research papers on these topics, and two books, *Building Knowledge–based Systems* and *Decision Making with Computers*, both published by Pitman. His current work includes a study of the use of artificial intelligence techniques in management accounting.

WU YONGFA is a lecturer in Management Engineering at the South East University in Nanjing, P.R. China, where he earlier obtained a BSc in mathematics and MSc in management engineering. His research interests are in decision support systems, systems engineering and computer–based systems in management in general. Past research projects have included urban traffic management, and the textile industry. He has written several papers published in Chinese journals, and also a textbook on operational research/management engineering.