Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

INTRODUCTION TO AWESIM

Jean J. O'Reilly
William R. Lilegdon

Symix Systems, Inc.
9200 Keystone Crossing, Suite 450
Indianapolis, IN 46240, U.S.A.

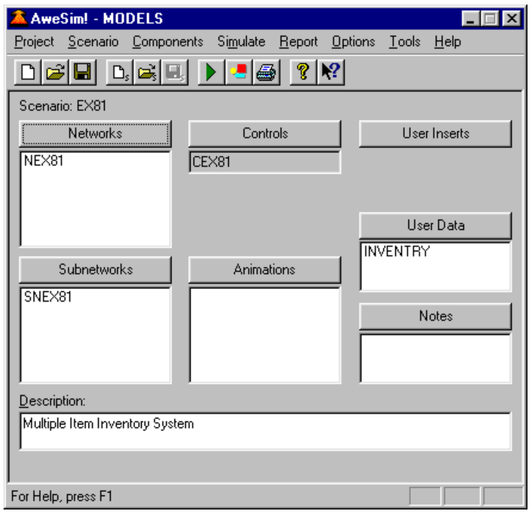
ABSTRACT

AweSim® is a general-purpose simulation system which takes advantage of the latest in Windows® technology to integrate programs and provide componentware. AweSim includes the Visual SLAM® simulation language to build network, subnetwork, discrete event, and continuous models. Network models require no programming yet allow user-coded inserts in Visual Basic or C. Discrete event and continuous models can be created using the object-oriented technology of Visual Basic, C or Visual C++ and can be combined with network models. This tutorial will demonstrate the process of using AweSim's componentware, describe examples of user interfaces that allow integration with other applications, and present a sample model.

1 INTRODUCTION

AweSim is a program that supports the range of tasks required to perform a simulation project. AweSim also provides integrating capabilities to store, retrieve, browse and communicate with externally written software applications. The most fundamental feature of the AweSim architecture is its openness and interconnectivity to databases, spreadsheets and word processing programs such as Microsoft Office. AweSim is built in Visual Basic and C/C++ and programs written in these languages are easily incorporated into its architecture. The details on the capabilities of AweSim are contained in the *AweSim User's Guide* (Symix Systems, 1997).

An AweSim project consists of one or more scenarios, each of which represents a particular system alternative. A scenario contains component parts. Software programs called builders are provided by AweSim to create each component. The AweSim executive window shown in Figure 1 describes an example scenario called EX81, which includes network, subnetwork, control and user data components. Component builders are accessed via the Components menu item or via the component's pushbutton.



Scenario: EX81

Networks: NEX81
Controls: CEX81
User Data: INVENTRY
Subnetworks: SNEX81

Description:
Multiple Item Inventory System

For Help, press F1

Figure 1: The AweSim Executive

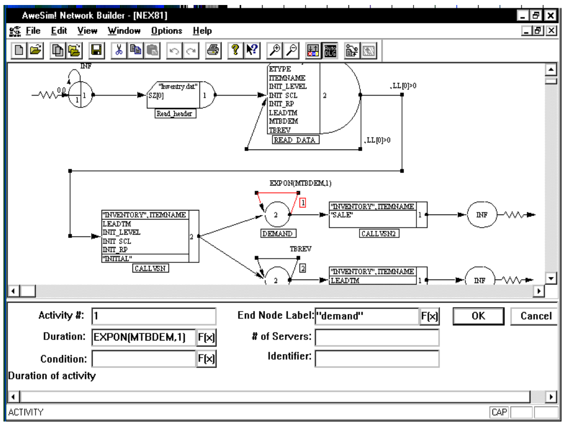
AweSim's Project Maintainer determines if a model translation or compilation is required. Each time the modeler requests a run, the Project Maintainer examines changes made to the current scenario to determine if any components have been modified and indicates whether translating tasks should be performed. The Project Maintainer then allows the user to specify whether these tasks should be done prior to performing the requested function. Multiple tasks may be performed in parallel while a simulation is executed in the background. The simulation modeler can switch between tasks by using a mouse to click in the appropriate window.

The use of the MS Windows interface simplifies learning and provides the foundation for combining application programs using that interface.

2 MODEL INPUT

AweSim incorporates the Visual SLAM modeling methodology. The basic component of a Visual SLAM model is a network, or flow diagram, which graphically portrays the flow of entities (people, parts or information, for example) through the system. A Visual SLAM network is made up of "nodes" at which processing is performed, connected by "activities" which define the routing of entities and the time required to perform operations. Visual SLAM nodes provide for such functions as entering or exiting the system, seizing or freeing a resource, changing variable values, collecting statistics, and starting or stopping entity flow based on status conditions. Entities may be given attributes, that is, characteristics such as type, size, or arrival time, to distinguish them and control their processing.

A network is built interactively in AweSim by selecting symbols from a graphical palette and dragging them to the desired location with the mouse. The symbol's parameters are specified by filling out a form, as illustrated by the ACTIVITY form shown in Figure 2. On-line error checking is performed upon completion of the form so that input errors can be corrected immediately. AweSim also facilitates model building by providing context-sensitive help and search capabilities.

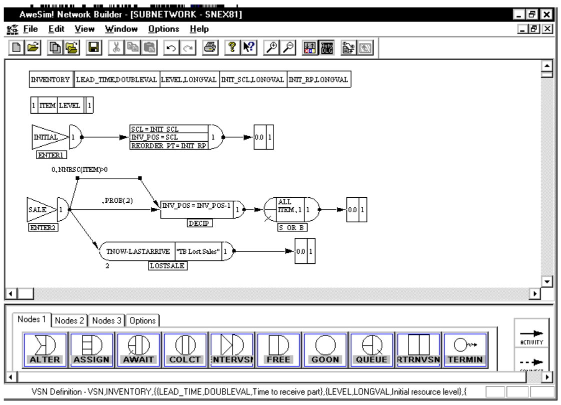

AweSim! Network Builder - [NEX81]
File Edit View Window Options Help
OK
Cancel
ACTIVITY
CAP

Activity #: 1
Duration: EXPON(MTBDEM,1)
Condition:
Duration of activity
End Node Label: "demand"
# of Servers:
Identifier:

Figure 2: The AweSim Network Builder

Subnetworks are reusable network objects, which are invoked from a calling network. Subnetworks provide for hierarchical models in which subnetwork instances are independent, encapsulated objects, called with a set of parameters. This is illustrated by the node named CALLVSN in Figure 2. Double-clicking on this node brings up the subnetwork builder for the named subnetwork, "INVENTORY". The subnetwork builder, shown in Figure 3, works just like the network builder but with a slightly different set of nodes.

In addition to the network and subnetwork builders, AweSim includes a forms-based builder for developing run controls (simulation run length, output options, when to clear statistics, queue ranking rules, etc.) and text builders for User Data files and Notes for documentation. The User Insert builder is either a text builder, if AweSim is installed for use with user inserts written in C, or the Microsoft Visual Basic development environment.



VSN Definition - VSN,INVENTORY,{(LEAD_TIME,DOUBLEVAL,Time to receive part},{LEVEL,LONGVAL,Initial resource level},{

Figure 3: Subnetwork Builder

3 MODEL OUTPUT

AweSim provides the ability to compare output from various scenarios both graphically and textually. A report "browser" allows alternative textual outputs to be compared side by side. Graphically, output may be viewed within AweSim in the form of bar charts, histograms, pie charts, and plots. Bar charts can be used to display the value of a statistic across scenarios. It is possible to view multiple windows of graphical output at the same time, as shown in Figure 4. Graphical and textual information from the AweSim database can be exported to other Windows packages such as EXCEL or Word for additional analysis and for documentation.

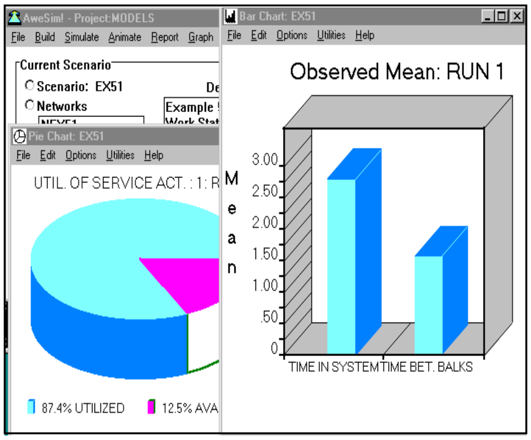


Figure 4: Multiple Report Windows

4 ANIMATION

With the AweSim animator, one may develop and display multiple animations for a single scenario (Figure 5).

For example, the modeler can create one animation of a system at an aggregate level and another at a department

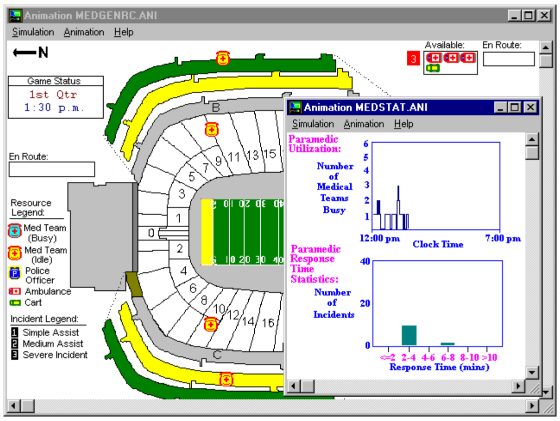
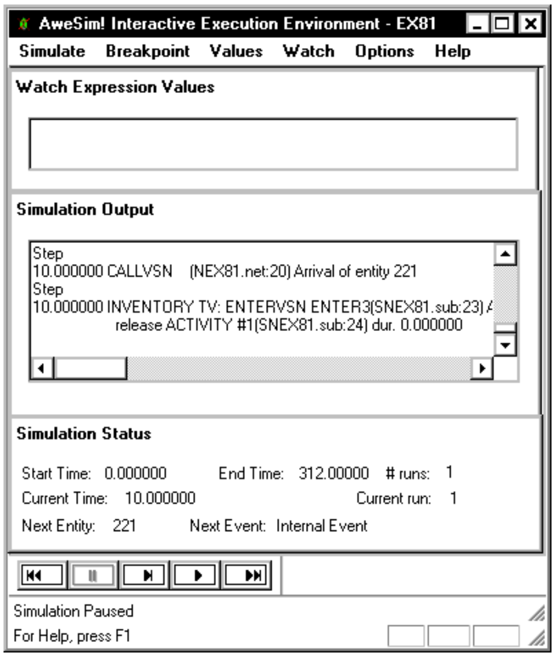


Figure 5: Animation with Multiple Windows

level, side by side in separate windows. The two views may then be displayed by associating the animations With the current scenario and running the simulation. Animation constructs, called actions, correspond to elements in an AweSim network model. For example, the ACTIVITY action shows movement of a symbol. It requires that the modeler define a symbol, a graphical path location where movement of the symbol will be shown, and the number of an activity in the AweSim model to which to tie the movement. Other actions are provided to display resource status, items in a queue, variable values, and other system status conditions. The symbols manipulated by the animator are of two types: graphical items one wants to display or move, and the background on which they will appear. These symbols are stored in standard Windows bitmap format, allowing them to be exchanged between programs using the Windows clipboard.

5 INTERACTIVE EXECUTION

The AweSim Interactive Execution Environment (IEE) provides an interface with an executing simulation. The modeler may examine, modify, save, or load the current system status using the IEE. The IEE supports the debugging of a model under construction and verification of a completed model. The analyst may use the IEE to develop and analyze alternative control strategies for a system or to demonstrate the operation of the model to non-modelers. The IEE control panel (Figure 6) lets you advance through the model step-by-step or by setting breakpoints. At any point you may examine variable values, statistics, or queue entries or save the system status in order to reload and experiment with alternative scenarios.


AweSim! Interactive Execution Environment - EX81
Simulate Breakpoint Values Watch Options Help

Watch Expression Values

Simulation Output

```
Step
10.000000 CALLVSN    (NEX81.net:20) Arrival of entity 221
Step
10.000000 INVENTORY TV: ENTERVSN ENTER3(SNEX81.sub:23) A
          release ACTIVITY #1(SNEX81.sub:24) dur. 0.000000
```

Simulation Status

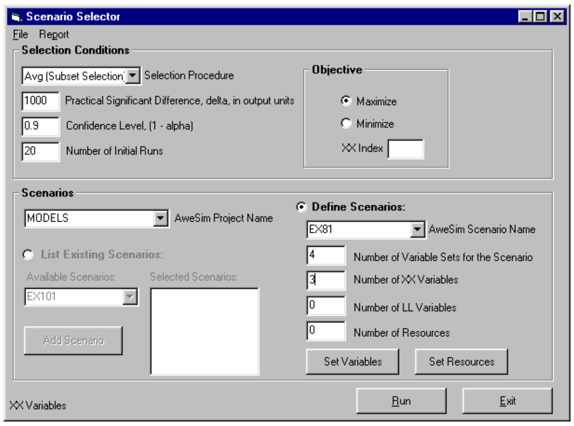
Start Time: 0.000000   End Time: 312.00000   # runs: 1
Current Time: 10.000000   Current run: 1
Next Entity: 221   Next Event: Internal Event

Simulation Paused
For Help, press F1

Figure 6: Interactive Execution Environment

6 SCENARIO SELECTOR

From the Tools menu item in the AweSim Executive Window, you can invoke the Scenario Selector function, which provides algorithms for screening a set of scenarios or for selecting the best scenario from a set of scenarios. The example shown in Figure 7 is invoking the Subset Selection algorithm to select a scenario that maximizes weekly profit (XX[5]). The Scenario Selector tool is described elsewhere in these proceedings (Pritsker et.al. 1999).



Selection Conditions
Avg (Subset Selection) Selection Procedure
1000 Practical Significant Difference, delta, in output units
0.9 Confidence Level, (1 - alpha)
20 Number of Initial Runs

Objective
[✓ Maximize] Minimize
XX Index

Scenarios
MODELS AweSim Project Name
List Existing Scenarios:
Available Scenarios: EX101
Selected Scenarios:
[✓ Define Scenarios:]
EX81 AweSim Scenario Name
4 Number of Variable Sets for the Scenario
3 Number of XX Variables
0 Number of LL Variables
0 Number of Resources

XX Variables

Figure 7: Scenario Selector

7 AN INVENTORY EXAMPLE

This example illustrates the use of subnetworks in an inventory system analysis (Pritsker and O'Reilly 1999).

Each subnetwork instance models the inventory level, sales events and review events for a specific inventory item. The main network initializes the model and schedules the sales and review events. This network is shown in Figure 2. It begins with a CREATE node which generates a single entity to read the data file INVENTRY.DAT. For each item to be modeled, the data file specifies the item number, name, initial level, stock control level, reorder point, lead time, mean time between demands, and time between inventory reviews (TBREV). The inventory status for each item is reviewed every TBREV weeks. If the inventory position, consisting of the amount on hand plus the amount on order minus the number of backorders, is less than the reorder point for the item, and order is placed to bring the inventory position back to the stock control level. The ETYPE, ITEMNAME, MTBDEM and TBREV variables are attributes of the item entity read from the file, while the remaining variables will be passed as parameters into the item subnetwork.

As each line of the data file is read, as long as the end of the file is not reached, an item entity proceeds to the CALLVSN node and a copy of the entity returns to the READ node to read data for the next item. The CALLVSN node invokes the subnetwork called INVENTORY for the instance ITEMNAME, which creates a new instance of the subnetwork for each item entity read from the file. A portion of the subnetwork is shown in Figure 3. It is headed by a VSN block which provides the subnetwork name and parameter list. The parameters of this subnetwork are the lead time, initial inventory level, stock control level and reorder point for an item. Below the VSN block is a RESOURCE block which defines the item as a resource whose initial level is a parameter of the subnetwork. Resource units will be allocated as sales are made, and the resource level will be replenished as orders are received. At the entry point into the subnetwork called INITIAL the item entity goes to an ASSIGN node to initialize the item's inventory control variables. Each instance of the INVENTORY subnetwork (i.e., each item) has its own SCL, INV_POS and REORDER_PT variables. After initialization, the item entity initiates two processes, DEMAND and REVIEW. At the GOON node called DEMAND, the demand entity schedules the next demand after a delay sampled from an exponential distribution with a mean of MTBDEM. A copy of the demand entity invokes the INVENTORY subnetwork for the instance ITEMNAME at entry point SALE. This entry point is also shown in Figure 3. At the SALE entry point, an item is sold if a unit is in stock. If not, there is a 20% probability that the item will be backordered. In either case, the inventory position is decreased by one and a unit of the item is allocated to the customer. If the customer is a lost sale, statistics on "Time Between Lost Sales" are updated.

The second branch from the CALLVSN node sends a review entity to the GOON node labeled REVIEW. Here the review entity schedules the next review after TBREV weeks and a copy of the entity invokes the INVENTORY subnetwork for the instance ITEMNAME at entry point "REVIEW". This portion of the subnetwork, shown in Figure 8, checks to see if the inventory position is less than or equal to the reorder point. If so, the order quantity is calculated and the inventory position is returned to the stock control level. An ACTIVITY then delays the receipt of the order by LEAD_TIME, which is a parameter of the subnetwork. After the delay, statistics on "Safety Stock" are updated and ORDER_QTY units are added to the inventory level.

8 INTEGRATION OF AWESIM WITH OTHER SOFTWARE

AweSim was designed to integrate easily with other Windows applications. AweSim is built on a relational database which is accessible with standard tools such as Dbase, Access, FoxPro and Excel. Input data is easily moved from an Excel worksheet to the AweSim input tables. Output data is stored in AweSim output tables, available for creating custom reports using a favorite tool. In addition to standard output data, raw data from the simulation can be stored in standard database or Excel format for analysis, manipulation, or presentation. Data used to create AweSim output graphics can also be exported "on the fly" to an output file for use in any tool accepting comma-delimited input.

An AweSim animation can use graphics created from other programs. As mentioned in the discussion of

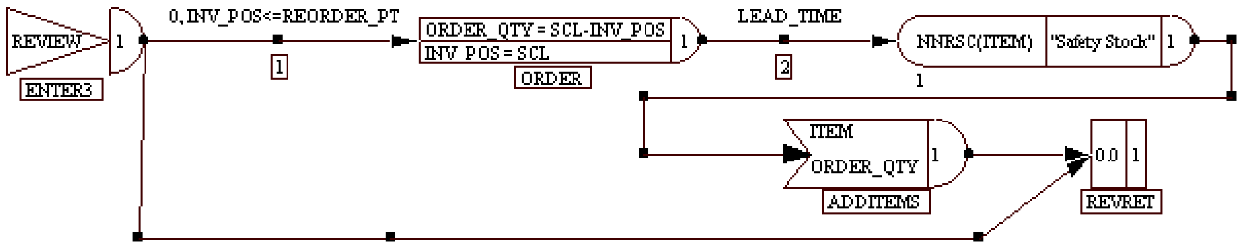


Figure 8: Inventory Subnetwork

animations, the graphical elements manipulated by the animator can be created using CAD, drawing or paint programs and loaded into AweSim by using the Windows bitmap format. The output charts and plots created by AweSim can be exported via the clipboard to other applications. For example, a pie chart created by AweSim may be copied to the clipboard and pasted into a word processing document describing the results of the model.

9 SUMMARY

AweSim is a new simulation support system which takes advantage of the latest Windows technology in order to provide a simulation support system able to interface with a variety of familiar tools. It incorporates the Visual SLAM modeling methodology. AweSim is distributed by Symix Systems, which offers regularly scheduled training classes as well as applications support.

REFERENCES

Pritsker, A.A.B., Goldsman, O., Nelson, B.N. and Opicka, T.L., "A Ranking and Selection Project: Experiences from a University-Industry Collaboration", *Proceedings, 1999 Winter Simulation Conference.*

Pritsker, A.A.B., and J.J. O'Reilly, *Simulation with Visual SLAM and AweSim*, John Wiley and Systems Publishing Corporation, 1999.

Pritsker, A.A.B., and J.J. O'Reilly, *Solutions Manual for Simulation with Visual SLAM and AweSim*, Systems Publishing Corporation, 1997.

Symix Systems, *AweSim! User's Guide*, West Lafayette, IN, 1997.

Symix Systems, *Visual SLAM Quick Reference Manual*, West Lafayette, IN, 1997.

AUTHOR BIOGRAPHIES

JEAN O'REILLY is a member of the technical staff at Symix Systems/Pritsker Division. She holds an M.S. in Applied Mathematics from Purdue University. Since joining Pritsker Corporation in 1978, Ms. O'Reilly has been involved in software development and in a variety of consulting projects. She is currently responsible for technical support, training and quality assurance for Symix simulation software.

WILLIAM LILEGDON is the Director of APS and Simulation Development for Symix Systems. He received a Bachelors Degree in Industrial Engineering from Purdue University. Since joining Pritsker Corporation he has worked in the consulting, development, support and training areas. Mr. Lilegdon led the development of the SLAM II/PC, SLAMSYSTEM, AweSim, and FACTOR/AIM software products. He has published numerous papers and articles on simulation products and their application and served as the General Chair of WSC '95.