I. System Description

Creating and running models with Ents is fairly straightforward after some basic concepts have been explained.

1. The opening window.

When the application starts up there are two windows present: the simulation window and a palette that contains symbols for various network blocks. Figure 1 shows a typical arrangement. The main menu (not shown) is typically placed at its default spot, the upper left hand corner of the screen.

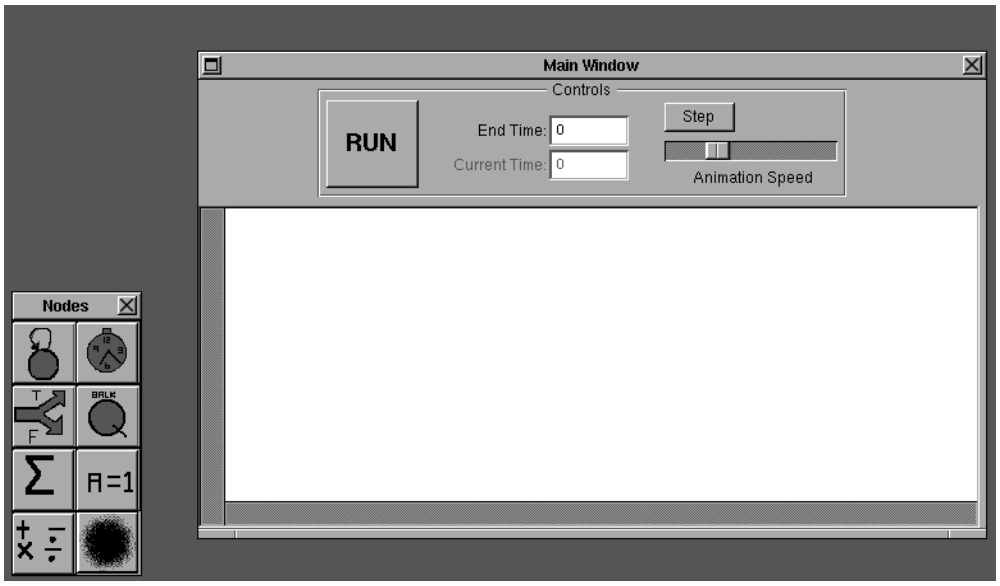


Figure 1. The Opening Screen.

To add a node to the model, click on the appropriate symbol with the mouse and drag it over to the simulation window, then release the button. The node will be added to the window. If you attempt to release the node in an inappropriate area—the background or the window of another application, say—the node will appear to travel back to the palette and no action will be taken.

2. Network symbols.

Ents defines eight network blocks from which simulation networks can be built. The model is created by placing blocks into a window, connecting them by drawing lines with a mouse, and then setting parameters for individual nodes. A network block that represents an activity is shown in Figure 2.

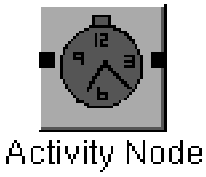


Figure 2. A network block

Each block type has its own icon; in this case, a clock face that represents the passage of time. The block's name is below the node. It can be edited by clicking on the text with the mouse and typing in new information. The node can be dragged to a new location on the background by clicking on it with the mouse and then, with the mouse button still down, moving to the desired location.

The parameters of the network node, such as the activity duration and the random number stream, can be viewed and set by double clicking on the icon. A panel to display and set the various network parameters will come up, and the user can select and specify values in the normal NeXT Step ways.

This node has two *connectors*, the small black boxes on the right and left of the icon. Entities arrive through the connector on the left hand side, and are sent to the next node through the connector on the right hand side. No source or destination nodes are shown for the node in figure 2.

To specify the next node an entity will go to, click down with the mouse somewhere in the right hand connector box and then drag, with the mouse button still down, to another connector. A line is drawn from the connector box to the current mouse location to let the user know something is happening.

When the mouse is over another connector box that is capable of accepting the entity, the receiving connector box will highlight. If the user releases the mouse button while over the accepting connector the line will attach to it; if not, the line will disappear. The line signifies that an entity path exists between the two nodes.

A connector is either an *inlet connector* or an *outlet connector*. An outlet connector specifies what node to send an entity to next. It may have one and only one line connecting it to another node, since the entity can be sent to only one node. Once a line has been drawn to an outlet connector no further associations are allowed. The user must first delete the existing link and then create a new one.

An inlet connector can have many lines connecting it to other nodes, since many blocks might need to send entities to the same node.

Outlet connectors must be linked to inlet connectors, and vice versa. This is just common sense—it serves no purpose to have two outlets connected to each other. The program will disallow any attempts to make such a connection.

The concepts above are applicable to all the process classes defined in Ents. Specific information on what the nodes do is described below.

i. Create nodes. The create nodes manufacture new entities at a user-selected interval. The symbol for the create node is shown in Figure 3.

Figure 3. The create node.

The create node only sends entities on to other nodes and does not receive them. It has one outlet connector but no inlet connector.

The inspector panel for the create node is shown in Figure 4. As with the other nodes, the inspector panel is called up by double clicking on the icon of the block in question in the simulation window.

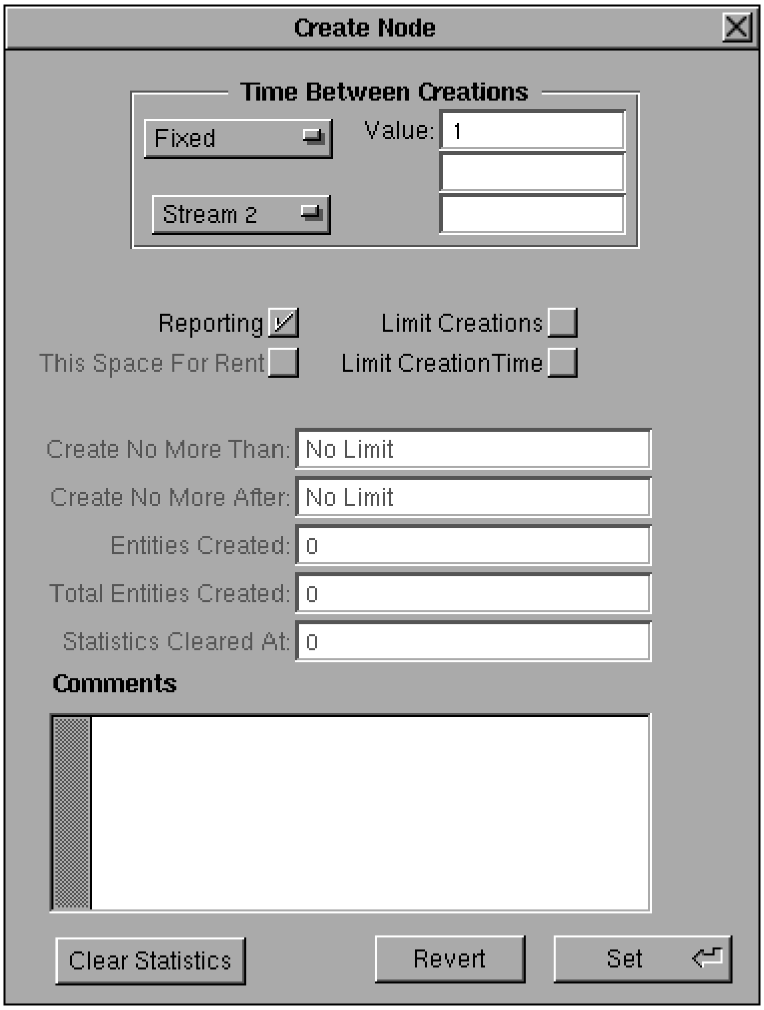


Figure 4. Create node inspector panel.

The inspector uses the usual NeXT Step conventions, including the close box in the upper right hand corner, buttons, fields, and pop-up lists. Fields that cannot be edited are denoted by light grey text. These only display information and cannot be changed.

By default the node creates an entity every time unit, though the parameter can be changed by editing the field at the top of the window. To select a random number distribution other than Fixed, click on the button in the *Time Between Creations* box and then drag to one of the other distributions. Exponential, normal, fixed, and uniform distributions may be selected from the list. The names for the distribution's parameters will appear in the form to the right, where the user can enter new values. The random number stream can also be selected in a similar manner from another pop-up list.

Negative values for time between creations are meaningless, and are therefore disallowed. If the user attempts to enter a negative number he will be alerted by a beep and the cursor will not advance to the next field. Restrictions on other random number parameters are also enforced so that the user cannot enter negative values for the normal distribution's variance or other obviously incorrect data.

If the reporting check box is on the node's summary statistics will be printed to a window if the user decides to print a report. If node statistics are of no interest, switch the check box off by clicking in it.

The create node can stop creating entities after either a specific time in the simulation or after a certain number of entities have been created. To enable either option, click on the appropriate check box and then edit the field in the form below it. By default there are no limits on entity creation.

Any comments or notes about the node can be entered in the scrolling text field at the bottom of the panel.

Once the create node parameters are set to your satisfaction, click on the *Set* button in the lower right corner of the panel to save the settings. The *Revert* button restores the node to the state it was in the last time the *Set* button was pressed. *If the inspector panel is closed without pressing the* Set *button the parameters will not be changed.*

A count of the entities created and the total entities created are displayed in the form. The *Entities Created* field refers to the number of entities created since the statistics were last cleared, while *Total Entities Created* is the running count of entities created since the simulation began. It is the *Total Entities Created* field that the entity creation limit refers to.

ii. The destroy node. The destroy node is the counterpart to the create node—

where the create node makes instances of new entities, the destroy node disposes of them and frees the computer system resources taken up by the entity representation. A destroy node does not forward entities to any other node, and therefore has only an inlet connector. The icon for the destroy node, the late, lamented black hole from release 1.0 of the operating system, is shown in Figure 5.

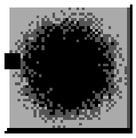


Figure 5. The destroy node.

The inspector panel for the destroy node is shown in Figure 6.

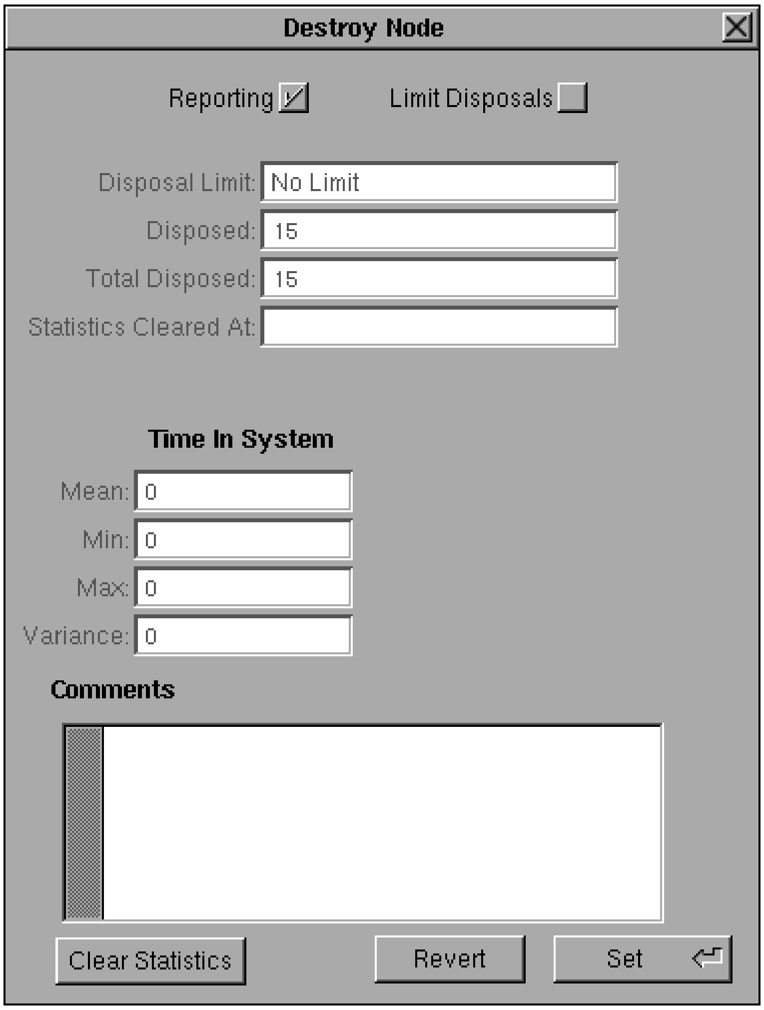


Figure 6. Destroy node inspector panel.

The destroy node inspector panel is quite similar to the other inspector panels and operates in an identical way. The *Disposed* and *Total Disposed* fields refer to the number of entities received since the last time the statistics were cleared and since the simulation began, respectively. Some statistics on the entities disposed by the system are shown in the *Time In System* form in the middle of the panel. The mean, variance, minimum, and maximum amount of time the entities disposed by this node have spent in the system are shown in fields that cannot be edited.

The simulation can be stopped after a specific number of entities have passed through the destroy node by selecting the *Limit Disposals* check box. The *Disposal Limit* field will become active and a new value can be entered. When the disposal limit has been equalled by the total number of entities disposed the simulation will stop execution.

iii. The assign node. Entities are objects which travel through the network and have attribute/value pairs that describe the entity. Instances of entities are manufactured at create nodes and disposed of (usually) at destroy nodes. The entities are given user-specified attribute/value pairs at assign nodes, the icon for which is shown in Figure 7.

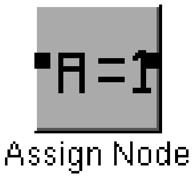


Figure 7. The assign node.

Entities have a few predefined attributes assigned by the system and an unlimited number of user-defined attributes.   The system-defined attributes cannot be redefined or assigned new value, but user-defined attributes can be added or changed at will. The restriction on system-defined attributes ensures that they always return the values expected of them.

The attribute names are text strings that can be any length and can include spaces or other characters; capitalization is not important, so “This Attribute” is equivalent to “THIS ATTRIBUTE” or “tHiS attRIBute.” While any text string can be used as an attribute name, it is unwise to use text that might be interpreted as a number, such as “12” or “1.4”. Though case is not important, the capitalization originally used is preserved to enhance readability.

The values associated with an attribute name are double-precision floating point variables.

For example, an entity might have attribute/value pairs as follows:

Time In System	10.4
TNOW	145.7
Creation Time	135.3
Repair Trips	2.0
Regular Trips	5.0

The first three attributes, Time in System, TNOW, and Creation Time, are system-defined. They always return the time the entity has spent in the system since it was created, the current simulation clock time, and the simulation time at which the entity was created. If the Time In System and TNOW attributes had been examined at another point in the simulation they might well have contained different values, while for a given entity the time of creation will always be the same. Repair Trips and Regular Trips are user-defined attribute/value pairs. The modeler is responsible for assigning and updating their values.

It should be emphasized that the attribute/value pairs are unique to each entity. Another entity might have different attributes and values, and changing the value of an attribute in one entity has no effect on the attribute in another entity.

The inspector panel for the assign node is shown in Figure 8.

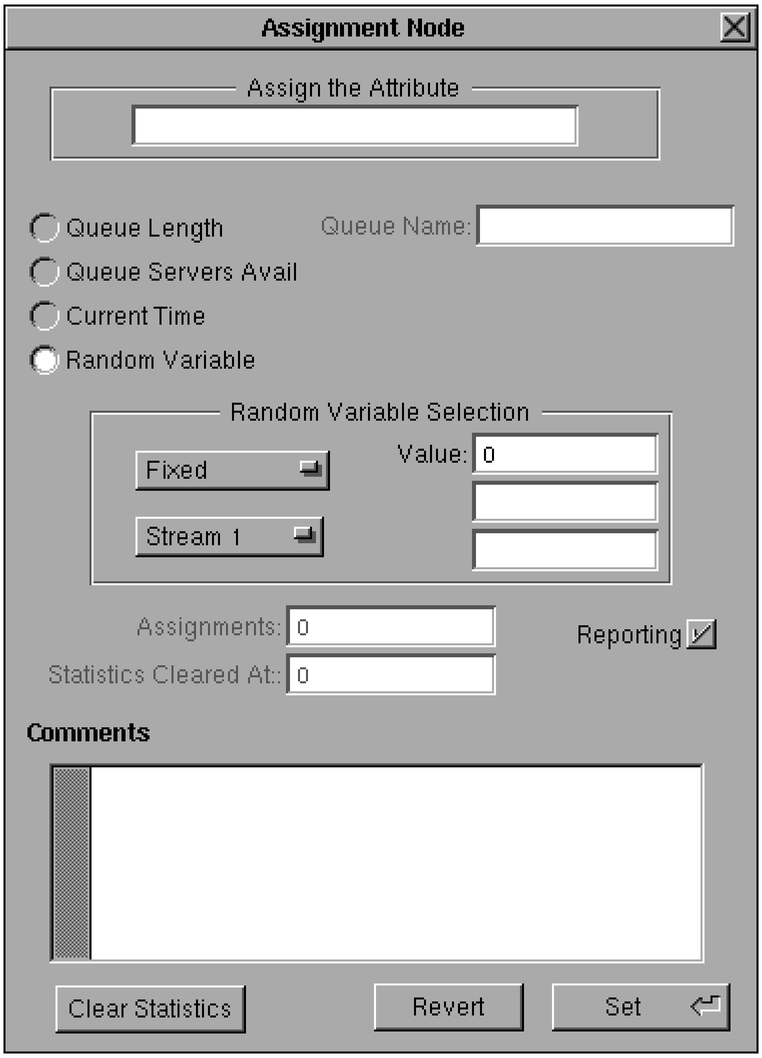


Figure 8. Assign node inspector panel.

The name of the attribute is entered in the field at the top of the panel. The user selects which of several values to assign to the attribute based on the state of the so-called “radio buttons” at the top left of the panel.

To assign a random variable to an attribute, click on the random variable radio button. All the other radio buttons will switch off and the selected button will switch on, just as a push-button car radio works when selecting a new station. Enter the parameters for the random variable in the field, and then make the selection permanent by pushing the *Set* button.

The current system time—the value of the system clock when the entity arrives at the node—can be assigned by clicking on the current time radio button. The attribute will retain the same value until it is reassigned, unlike the TNOW system assigned variable, which always reflects the current simulation time.

The attribute can also be assigned the length of a queue or the number of queue servers available. When either of these radio buttons is pressed, the *Queue Name* field becomes active. Enter the name of the queue you want to monitor. (The queue name is the text that appears underneath the node icon.)

The queue must already exist, and must be uniquely named. The inspector panel will refuse to accept a queue name that does not exist or which is used by two queues. The assign node is also checked at runtime for ambiguous name references; if another queue with the same name has been added after assignment node was specified, an alert panel will warn the user and prevent the program from running. Queues that are not mentioned in assignment nodes are allowed to have identical names.

The *Queue Length* and *Servers Available* assignment options are useful primarily for decision making. Often the entity will be routed to a different network branch when all the servers are busy or when the line exceeds a certain length[1].

When an entity arrives, the node checks to see if the attribute already exists. If it does, the attribute's value is replaced by the value shown in the inspector window. If the entity does not have the attribute it is created automatically and added to the list of attribute/value pairs. There are no limits on the number of attributes an entity may have, aside from computer system restrictions.

iv. The activity node. The activity node delays an entity by a specified amount of time. Any number of entities may be undergoing an activity at one time. An example of an activity might be the delay associated with moving a part from one station to another via conveyor belt. There is essentially no limit on the number of entities that can be engaged in the activity, and the process takes some amount of time

[1] Routing to a different node when the line becomes too long can also be acheived with the balking feature of queue nodes.

to complete.

The icon for the activity node is a clock face, shown in Figure 9. An activity node has one inlet connector on the left hand side of the node icon and one outlet connector on the right hand side.

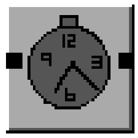


Figure 9. The activity node.

The inspector panel for activity nodes is shown in Figure 10.

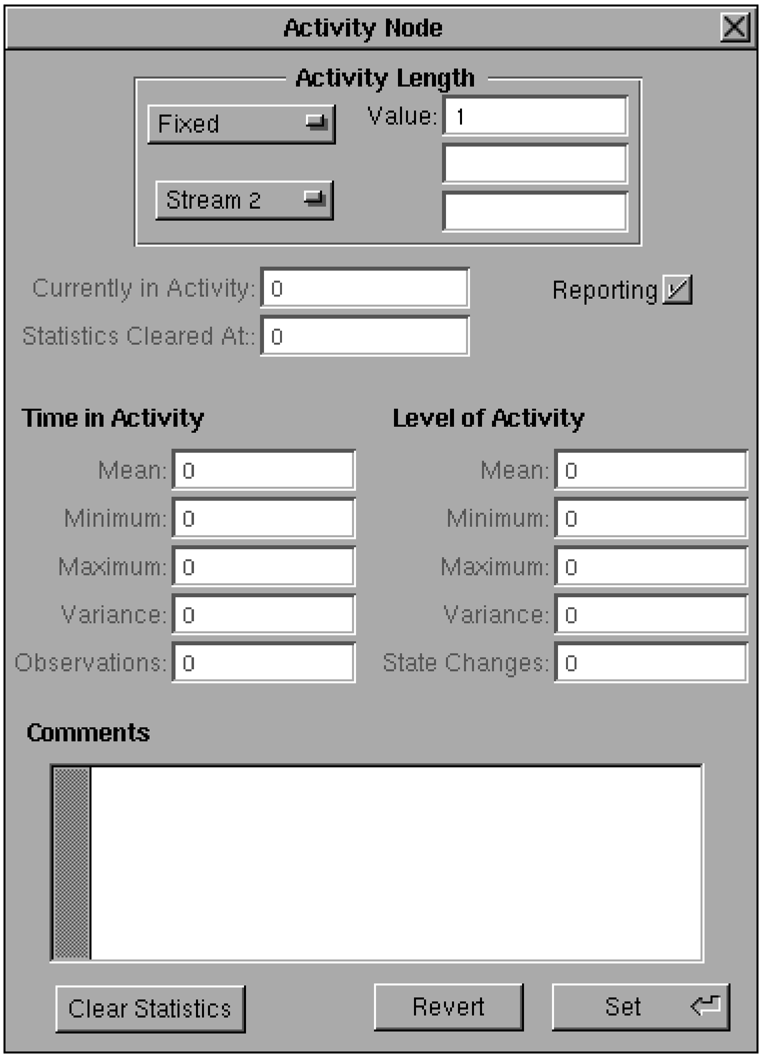


Figure 10. Activity node inspector panel.

The length of the activity can be set by selecting a random number distribution from the pop-up list and then entering the distribution's parameters. Either constant numbers or entity attributes can be entered as random number distribution parameters. If the latter option is used, the entity will be examined and the value of the attribute looked up and used as the parameter. If an entity arrives without the attribute shown or a value that is obviously invalid (a negative variance for the Normal distribution or a fixed length activity time of less than zero, for example), the simulation will stop with an error message.

Constant numeric values and entity names are distinguished by first attempting to resolve the data entered into the field as a number. If a negative number is entered the field will disallow entry; if a positive number is entered the field will accept the data. If the data cannot be interpreted as a number it is assumed to be an attribute that will be present in all entities that arrive at the node. It is for this reason that giving an attribute a name that might be construed as a number is a bad idea; an attribute named "13" would always be resolved as the value 13, and the value associated with the name would never be referenced.

The random number stream used by the random variable generator is selected through the other pop-up list in the box.

Reporting for the node can be turned on or off via the labeled check box.

Assorted statistics and status information on the node are displayed in greyed-out text, since they reflect a system state that cannot be changed by the user. The number of entities currently engaged in the activity is shown near the top of the panel, and statistics on the time the entities have spent in the activity and the *level* of the activity, or average number of entities occupied in the activity over time, are shown in two forms near the bottom of the panel. This is a statistic based on a time-persistent variable.

v. Branch nodes. Branch nodes route the entity based on the values of its attributes. For example, all entities that have a time of creation less than 500.0 might be sent to one branch of the network, while those created after that time would be sent to a branch that did special processing for late arrivals. The icon for the branch node is shown in Figure 11.

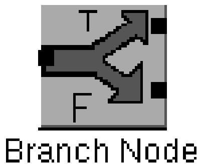


Figure 11. The branch node.

The branch node has one inlet connector and two outlet connectors. The upper outlet connector routes the entity to one branch if a boolean test is true, and the lower connector routes the entity to another branch if the test is false.

The inspector panel for branch nodes is shown in Figure 12.

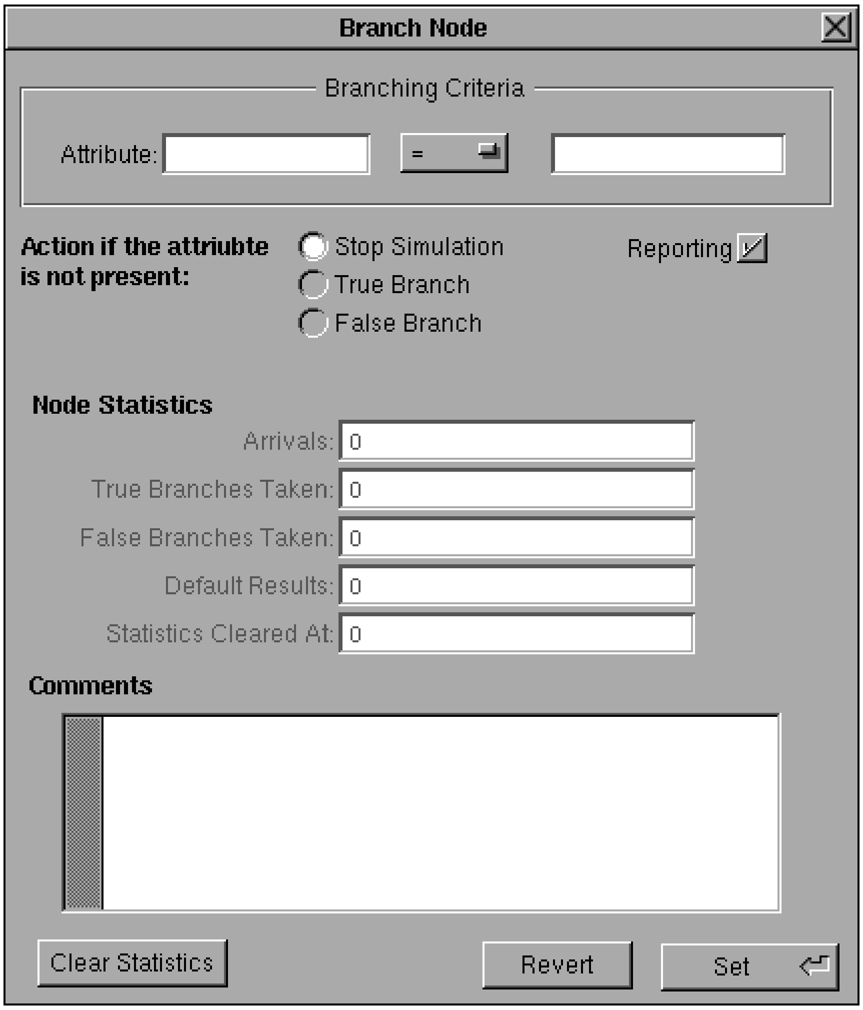


Figure 12. Branch node inspector panel.

The branch node performs a single boolean test on the value of an attribute given by the user. The name of the attribute is entered into the field at the upper left hand side of the panel, while the value it is being compared against is entered in the field to its right. The comparison operator, one of =, <=, >=, >, <, or <>, is selected from the pop-up scrolling list between the two fields.

The value on the right hand side may be either a constant value or another attribute contained by the entity. For example, the user might specify a left-hand attribute of TNOW, a comparison operator of <, and a right hand value of 550. If the current clock time is less than 550 the entity will be routed to the true branch. Alternatively, the right hand field could contain another entity attribute. The value of the right-hand attribute will be compared to the value of the left-hand attribute, and the entity routed accordingly. Suppose that an entity has been assigned attributes of "Max line length" and "Current line length," and that the comparison operator is "≥". Entering the names of the attributes in the left-hand and right-hand fields will ensure that the entity is routed to the true node when the value contained in "Max line length" is greater than or equal to the value contained in "Current line length."

The inspector panel differentiates between values and attribute names in the same manner that the activity node does: by first attempting to resolve the entry as a number, and then by assuming it is an attribute. Therefore attribute names that might be interpreted as numbers should not be used.

An ambiguity exists if an entity arrives at the branch node without one or both of the attributes referenced in the boolean statement: how can we determine the truth of a statement if we have no data upon which to base a comparison? As a safety measure the user can state what should occur if this situation occurs. The simulation can stop, the default state; the entity can take the true branch, or the entity can take the false branch. Any of these options can be selected by clicking on the appropriate radio button.

Statistics are kept on the number of entities arriving at the node, taking the true branch, the false branch, and default branches (an entity arrived without an attribute).

vi. The arithmetic node. The arithmetic nodes performs arithmetic operations on attribute values. An attribute might have one added to its pre-existing value whenever it comes to the node, for example, or it might be multiplied by another entity attribute. The icon for the arithmetic node is shown in Figure 13.

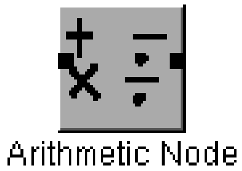


Figure 13. The arithmetic node.

The arithmetic node has one inlet connector and one outlet connector.

The inspector panel for the arithmetic node is shown in Figure 14.

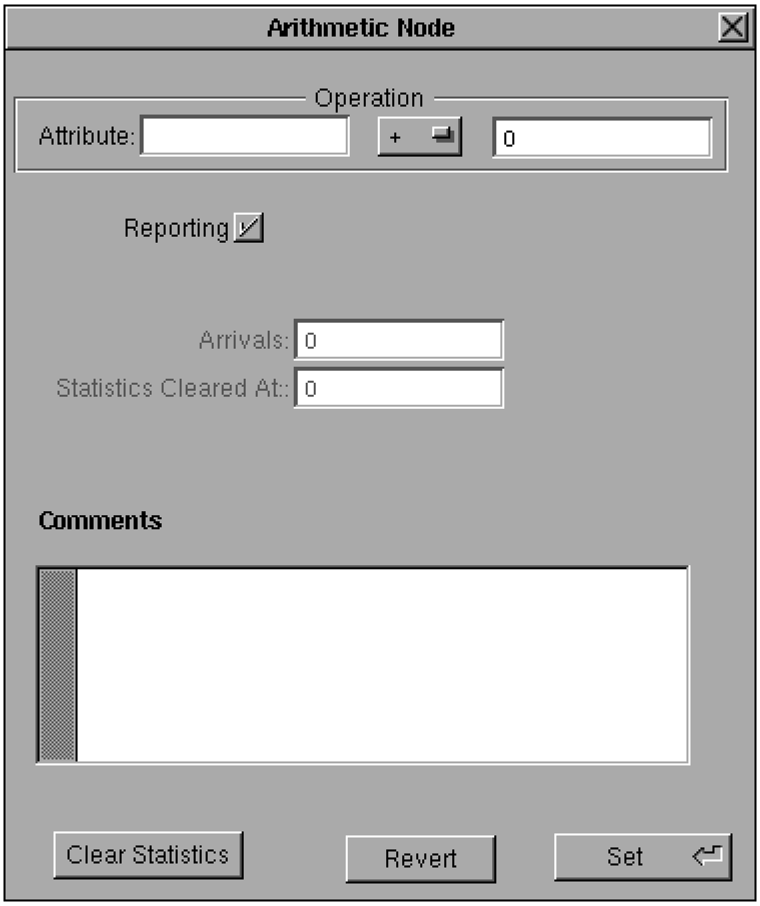


Figure 14. Arithmetic node inspector panel.

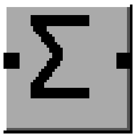
As with the assignment node, the name of the attribute is put into the field at the upper left of the inspector panel. The operation to be performed on the attribute,

+, -, /, or *, is selected from the scrolling pop-up list button. The attribute that the result will be placed in is entered into the left hand field, and a number or attribute is entered into the right hand field. As before, the value on the right hand side is first resolved as a number; if that operation fails the string is resolved as an attribute name. This allows either constant values or attribute values to be used on the right hand side of the expression.

If the left hand attribute did not exist prior to the entity's arrival at the node the attribute is created with a value of zero, and then the arithmetic operation is performed; if the right hand attribute name does not exist in the entity its value is assumed to be zero.

Suppose an arithmetic node has is specified such that "Loopcount" is in the left hand field, the operator is "+", and the number "1" in the right and field. If an entity arrives without the attribute "Łoopcount," it will be created and added to the entity's attribute list with an initial value of zero. Then operation will be performed; in this case, one will be added to zero, and the attribute will have a value of one. If the entity returns to the node the value of "Łoopcount" will be incremented by one each time from its existing value: 2, 3, 4….

vii. The statistics node. The statistics node collects summary data on a given entity attribute. The icon for the statistics node is shown if Figure 15.



Statistics Node

Figure 15. The statistics node.

The statistics node has one inlet connector and one outlet connector.

The inspector panel for the statistics node is shown in Figure 16.

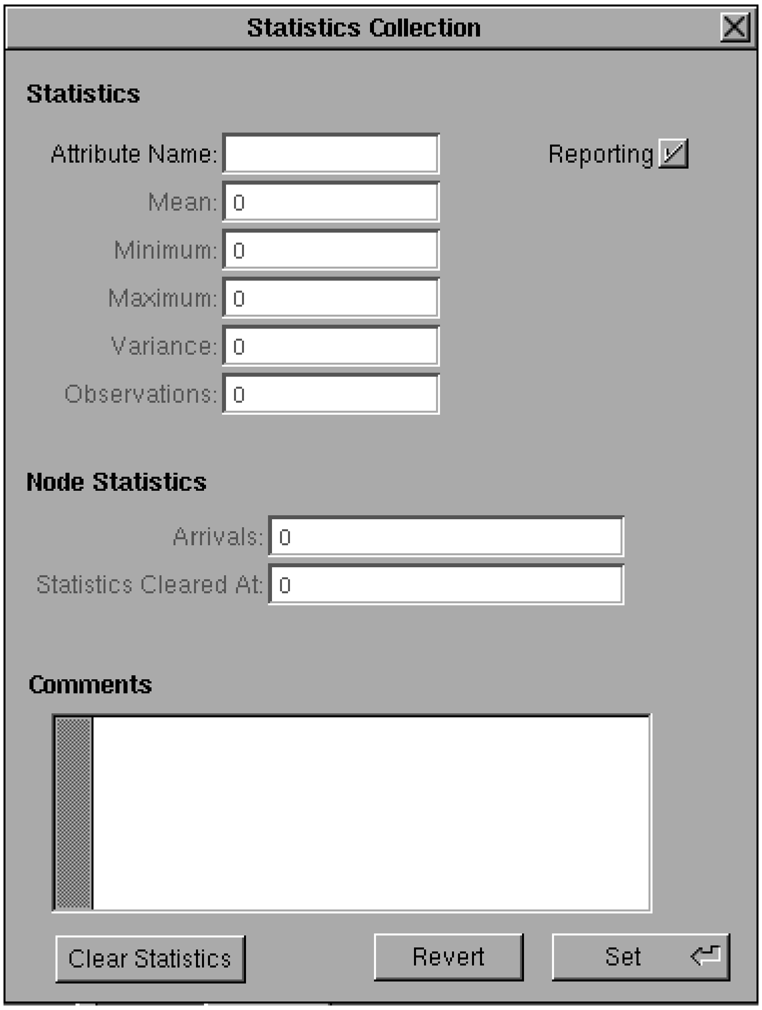


Figure 16. Statistics node inspector panel.

The attribute name that observation statistics are to be collected on is specified in the field at the top of the inspector panel. All entities that have an attribute by that name and that pass through this node contribute their data to the summary statistics. If an entity arrives at the node that does not contain the attribute, it is passed on without further action.

The mean, minimum, maximum, count and variance are calculated and shown from those entities that do contain an attribute with the name shown in the form. The statistics can, as usual, be reset to the zero state by pressing the *Clear Statistics* button.

Occasionally the user might want to confirm that no entities arrived at the node without an attribute by the name shown in the inspector panel. This can be done by comparing the number of arrivals to the node with the observation count in the statistics form. If such an entity passed through the node the number of arrivals will exceed the number of observations in the summary statistics.

viii. The queue node. This process class is one of the most complex and useful node types. Many business simulations become, in effect, a network of interacting queues, activities, and decision actions that are too complex to be solved by classical queuing theory techniques. The icon for the queue node is shown in Figure 17.

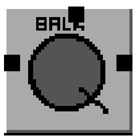


Figure 17. The queue node.

The queue node has one inlet connector and two outlet connectors. The connector at the right of the node is the normal outlet connector, where entities are sent when the complete service. The connector at the top of the icon is the balk connector.

Balking is a user-selected option. If an entity arrives and finds that the line is too long it immediately balks and goes to the node pointed to by the balk connector. If balking is not enabled the connector is not required to point to another node. *This is the only exception to the rule that all connectors must point to another connector.*

The inspector panel for the queue node is shown in Figure 18.

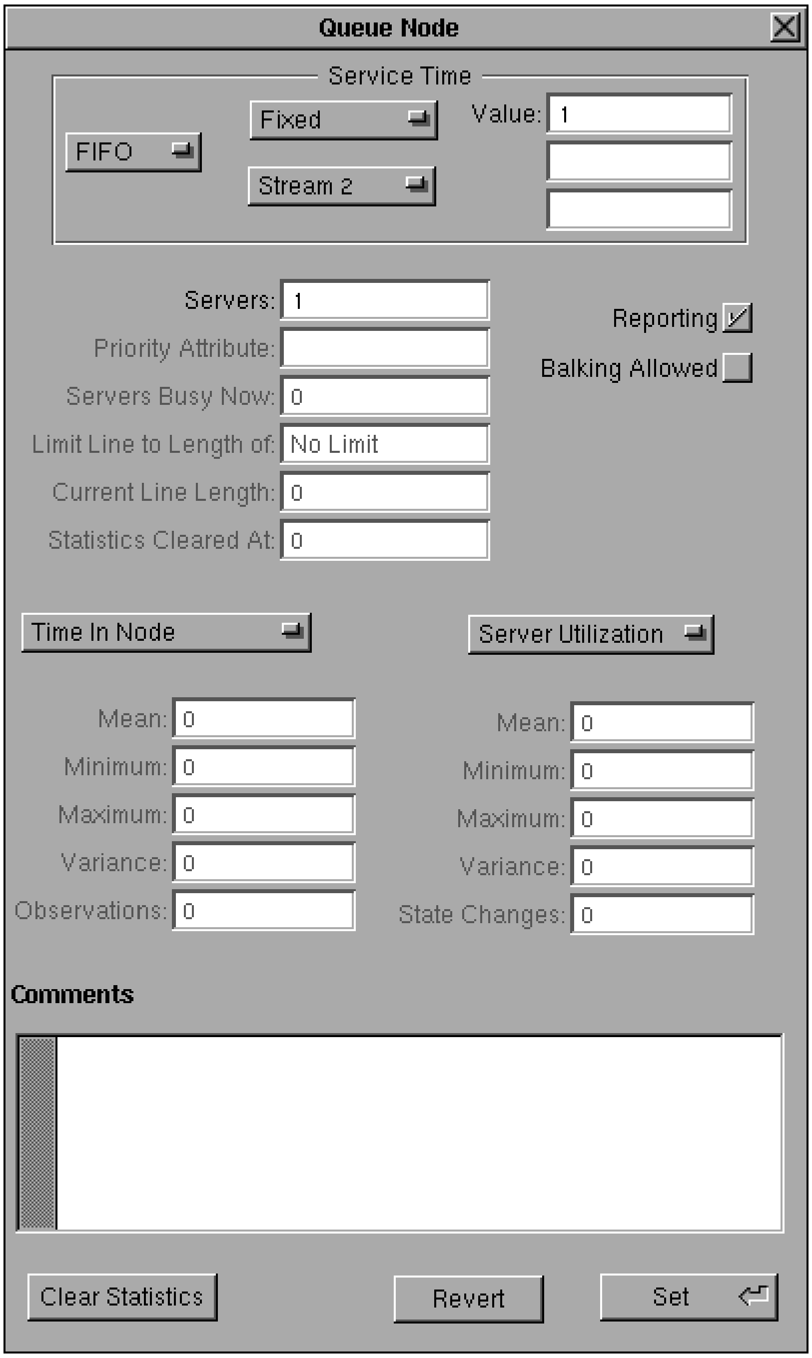


Figure 18. Queue node inspector panel.

The entity will arrive at the node and check for an available server. If no server is available the entity goes into a line and waits until one becomes free, then enters service. The time the entity spends in service is a random variable; afterwards the entity is sent on to the next node.

Balking is enabled by clicking in the marked check box. The maximum line length field becomes active, and the user may enter the largest size the line can grow to. For example, a drive-up teller window might have physical or logical constraints on the number of cars that can wait for service at one time: limited space for cars in the line or impatient customers, perhaps. If balking is enabled the balk output connector must point to another node. An error is reported by the program when it attempts to run if no balking connection exists and balking is enabled.

The length of service random variable is selected in the usual way from the pop-up lists and field entries in the box at the top of the inspector. The queue discipline can be selected from the pop-up list to the left; the options are FIFO, LIFO, and Priority. If the priority option is selected the priority attribute field is activated in the form below it and the user is allowed to enter the attribute name.

If the queue is in priority order the line is arranged in ascending, FIFO order: the entities that have the highest attribute value are placed first in line, and if any ties occur they are broken by putting the later arrival further back in the line. If an entity arrives at a priority queue node without the attribute the line is ordered by, the entity is placed at the back of the line in FIFO order. New entities with the attribute will always be placed in front of older entities without the attribute.

The number of servers is set through the *Servers* field, which can be set to any non-negative integer. If servers are added while a line for service exists, the waiting entities are taken in order from the line and placed into service immediately. If the number of servers is reduced below the number already busy the servers will be removed when the entity they are occupied with finish service.

The number of servers that are busy and the length of the current line are shown in greyed-out text to signify that the user cannot directly change these parameters.

Several summary statistics on the performance parameters of the queue can be shown in the two forms towards the bottom of the inspector panel. Observation based statistics are displayed in the left hand form and statistics based on time persistent variables in the form to the right. Each form can display statistics for a number of variables by selecting the pop-up list above the form. Observation based statistics that are collected automatically include *Time in Node*, *Time in Line*, *Time Being Serviced*, and *Time Between Balks*. *Time in Node* refers to the cumulative waiting time and service time, while *Time in Line* and *Time Being Serviced* are a more detailed breakdown of the same performance data. The remaining statistics are self explanatory.

The statistics based on time-persistent variables are displayed in the form to the right. These include *Server Utilization*, *Entities in Line*, and *Entities in the Node*.

Server Utilization is the average number of servers busy, over time. This is particularly useful information for use in determining how many servers are required to efficiently serve customers. High utilization is the usual design objective when optimizing a model, but this also has a side effect of increasing line length and customer waiting time.

The average number of entities in line and the average number in the node are two other summary statistics that are useful in optimizing models. The *Entities in Line* statistic refers to the time-averaged number of entities waiting for service, generally a parameter that should be minimized. The *Entities in Node* statistic is the time-averaged number of entities both waiting and undergoing service.

3. Running simulations. Building and running a simulation is simple. A typical window arrangement is shown in Figure 1. Drag the node objects over from the palette, drop them into the simulation window, and then hook up the objects by dragging between connectors with the mouse. The simulation window can be made larger to view models that take up more space, and the background upon which the network nodes rest can be scrolled with the sliders on the edge of the window. The background will scroll automatically when the user drags an existing node beyond the right hand or upper edges of the visible region.

All the connectors with the possible exception of queue node balk connector should point to another node. Enter the time to stop the simulation by typing in the text data entry field in the upper center of the simulation window, and then press the *Run* button. The simulation will execute until it reaches the end time or until one of the nodes in the model meets the simulation termination criteria.

The program checks for obviously incorrect data before running the simulation. If a node has incorrect or incompletely specified data an alert panel will pop up with the name of the node and a description of the problem.

The simulation window is in a so-called *modal loop* when the model is running and ignores input to menus, panels, or windows other than the running simulation window. The active simulation window still accepts input, however; the simulation can be stopped by pressing the *Stop* button, a node inspector panel can be opened, and the speed of the animation can be changed by moving the slider (animation and other features are discussed below). Allowing a model to run in background while the user continues to work in other windows would require multiple threaded execution of the application. The NeXT supplies the tools to do this, but this feature has not been implemented in the current version of the program.

The application's main menu is shown in Figure 19. For the most part the menu contains standard NeXT Step items that will be familiar to any NeXT user.

Figure 19. The main menu.

The menu items with arrow heads to their right signify submenus, and those ending with a "…" signify that a dialog box or some other panel will come up when the item is selected. Submenus can be "torn off" from the main menu and made into stand alone menus; all the menus can be dragged to more convenient locations on the screen. Most NeXTs are set up to place the main menu in the upper left hand corner of the screen when the application starts.

The menu that is most concerned with simulation proper is the *Simulation* submenu, a submenu of the main menu. It is shown in Figure 20.

Figure 20. The *Simulation* submenu.

The *Run* and *Step* menu items are alternatives to the buttons in the simulation window. The have keyboard equivalents of command-r and command-N, respectively.

A textual summary of the model's performance can be shown in a window at any time by selecting the *Report* menu item. All nodes that have the reporting check box enabled in their inspector windows will have their relevant statistics summarized and tabulated. Animation and execution tracing are discussed in the debugging section below.

Many models have a so-called "warm-up period" during which the model reaches steady-state conditions. The statistics gathered while the model is in the transient state can bias the estimate of the true steady-state system behavior, if ascertaining steady-state behavior is in fact the objective of the modeler. The statistics gathered by each node can be cleared by bringing up the inspector panel and pressing the *Clear Statistics* button, but doing this for each of the nodes in a large model would be very tedious. All the node statistics can be cleared at once by selecting the *Clear Statistics…* item from the *Simulation* menu, which has exactly the same effect as bringing up the inspector panel for each node and clicking the *Clear Statistics* button. An alert panel will come up to confirm the user's intent and then continue with the operation.

The model can be reset to its original, time zero state by selecting the *Reset…* item from the *Simulation* menu in a similar manner. It is necessary to reset the simulation to make topology changes or to add nodes to the model. Changing the parameters of existing nodes does not require a reset.

If the palette window disappears, a new one can be opened or the old one brought to the front by selecting the *Palette…* item from the main menu.

The above submenus and menu items are the only unique additions to the main menu. The remaining menus and menu items are standard NeXT Step application features which should be familiar to most users. For example, the model can be saved to disk at any time by selecting the *Document* menu item. This will present a standard NeXT Step save menu, which can call up the usual file saving and opening panels. The model will be saved with its current state—including the current simulation clock, event list, and node statistics.

The *Windows* menu item in the main menu shows all the application's open windows in a submenu. The Windows menu is another of the standard NeXT Step menu items that should be familiar to NeXT users.

The *Services* menu provides access to NeXT "Services," or cooperating programs that provide some utility function. While the services available will differ from system to system depending upon how the system administrator has set up the computer, it is almost certain that the user will be able to look up words in the Digital Webster on-line dictionary and to send text to another user by electronic mail. All the user needs to do is select an area of text with the mouse and then select the corresponding menu item from the *Services* menu. A program called nxyplot, available from NeXT-related electronic bulletin boards at no charge, can take textual numeric data that has been selected with the mouse and create charts and plots.

4. Output features. Ents uses the normal NeXT Step conventions for printing. Simply click on the Print… item in the main menu and a dialog box comes up that allows the user to pick a printer or file to which to send output.

5. Debugging. It is the rare analyst indeed who can always correctly specify the simulation model the first time. Ents includes debugging tools that let the user trace through program execution and entity flow.

Clicking on the *Trace Execution* menu item of the *Simulation* menu brings up a text window that prints a record of every event executed by the simulation engine. The event time, event type, and name of the node at which the event occurred are printed to the window for every event that is executed.

The *Step* menu item in the same menu and the *Step* button in the simulation window single-step through the event list. Both have the same effect—dispatching a single event and then stopping the simulation for examination. Stepping through the model execution is most useful when the simulation is in the execution tracing mode.

One particularly powerful debugging tool is the animation capability. When animation is turned on from the *Simulation* menu the movement of entities between nodes is shown by a small sphere that travels from one icon to another. A few minutes examination of a running simulation is usually all that's needed to determine that a model is routing entities incorrectly. The animation can be sped up or slowed down by adjusting the animation speed slider.

II. Application Examples

The specifications of the simulation process nodes were discussed in detail in the previous chapter, but no simulation is made of nodes working in isolation. At least a few nodes need to be used in a typical model, and increasing the scale from single nodes to an interconnected network presents hurdles that users often find difficult to overcome. This chapter gives two examples of modeling entire systems with Ents: a TV repair line and a Quarry operation, two examples that have been used throughout the literature.

1. TV repair problem.

One of the canonical examples used in simulation texts is that of a TV repair line, originally developed by Schriber (1974). Pritsker and Pegden (1979) restate the critical parameters of the system and model it using SLAM. Pritsker and Pegden's description is repeated here, and then the model will be developed step by step in Ents.

A television inspection and repair line has units arrive to its entry point at a rate uniformly distributed between 3.5 and 7.5 minutes. Two inspectors work at the entry point side by side, each examining sets as they arrive. An inspector takes between 6 and 12 minutes, uniformly distributed, to check over a TV. If the set passes scrutiny it is sent on to the packing department and out of the system of interest, but if it fails it is sent to an adjustment station where another worker tunes it. Once that operation has been completed the adjusted set is sent back to the incoming inspection station. On average 85% of the units pass inspection and 15% are in need of an adjustment; the adjustment takes 20-40 minutes, again uniformly distributed. Sets that have been adjusted are just as likely as new arrivals to be found defective. A schematic diagram of the system is shown in Figure 21.

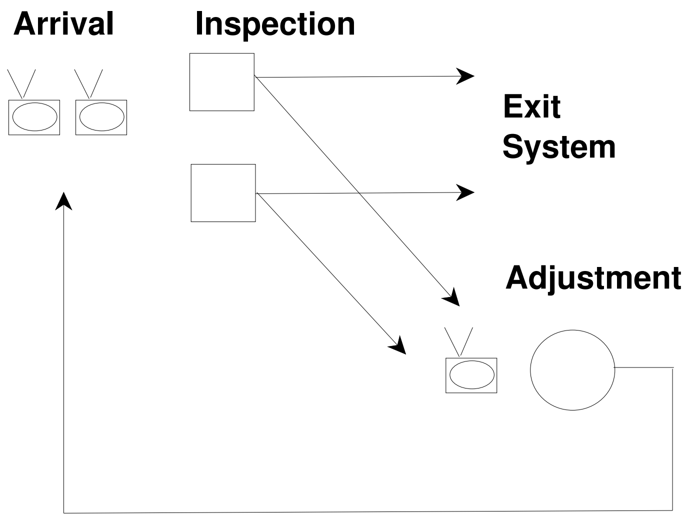


Figure 21. TV Inspection and Repair Line schematic diagram.

The TV sets arrive, are inspected, and either sent out of the system or to the adjustment station. At the adjustment station they are fixed, perhaps, and then sent back to the arrival station to be inspected with the other new TVs.

A model of this system is characterized by an entity creation mechanism to correspond to the entry of new TVs into the inspection line; a queue with two servers acting in parallel related to the inspection station; another queue for the adjustment station, this one manned by a single person; and a way to send some of the entities to the adjustment queue and the rest out of the system model. The creation mechanism, queues, and entity disposal parts of the model are easy to implement since they have exact equivalents in the defined process classes. Finding a way to route the entities is slightly more difficult. Two Ents nodes are needed to do this, one to assign an attribute and one to dispatch entities based on the attribute value.

From the opening screen shown in Figure 1, drag over a create node, two queue nodes, an assign node, a branch node, and a destroy node from the palette. Arrange them in the simulation window as shown in Figure 22. More meaningful captions can be placed on the nodes by clicking on the text with the mouse and entering new names.

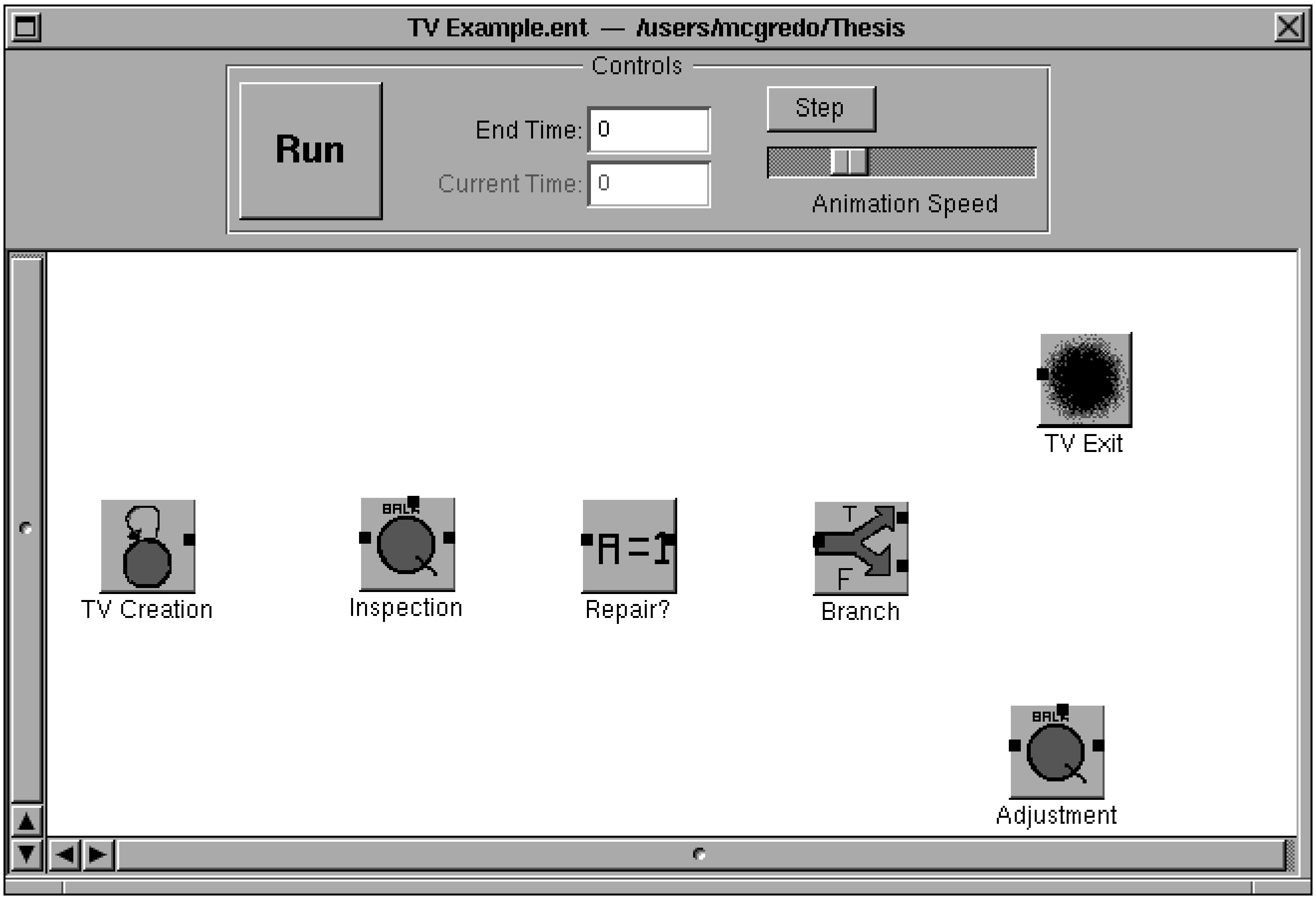


Figure 22. Initial node layout for TV inspection.

Using the mouse, specify the routes an entity can take as it travels through the model by connecting the nodes to each other. At the create node, click on the connector box and drag out a line to the inspection queue. When the pointer makes contact with the inlet connector the small box will highlight, signifying that a connection is possible. Release the mouse and the line will become permanent. Specify the other routes so that the entity travels from the create node to the inspection queue, the assignment node, and the branch node. Draw a line between the true outlet connector and the destroy node, and from the false branch connector and the adjustment queue. From the adjustment queue the entities return to the inspection station, so draw another line from the queue output connector to the inlet connector of the adjustment queue. There is no need to connect any of the queue balk nodes, since balking will not be allowed in this model. The network should resemble that shown in Figure 23. when finished. The network topology has been completely specified.

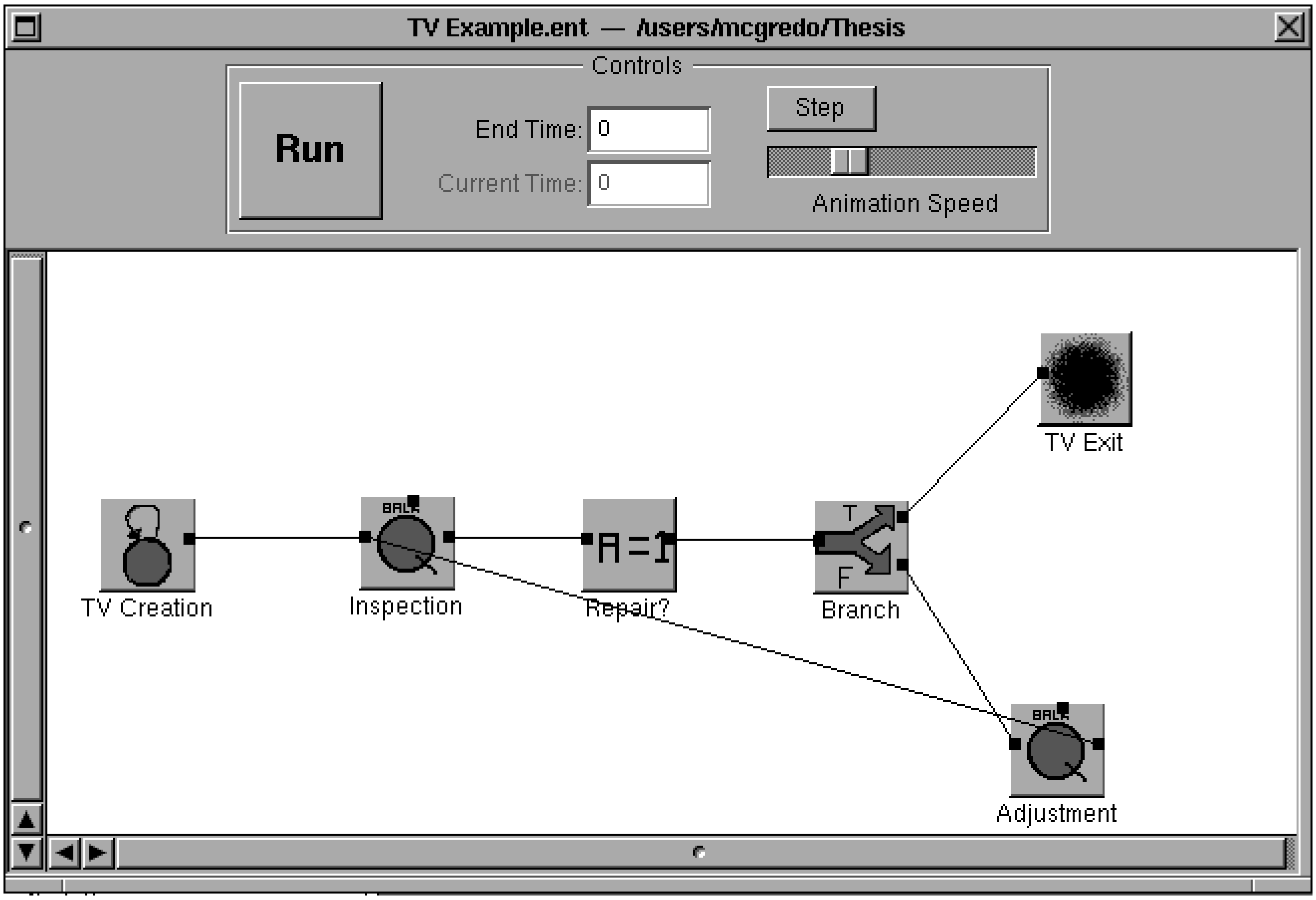


Figure 23. The connected TV inspection model.

The next step is to correctly set the parameters for the various nodes. The nodes have reasonable default values, but these need to be changed to match the requirements of our this system.

Double click on the TV creation node. An inspector panel in which the time between creations can be set appears. Change it to a uniform distribution of 3.5 to 7.5 minutes.

The inspection queue node needs two servers, but by default it has only one. That needs to be changed. Double click on the inspection queue icon and edit the inspector panel so that two servers are present and the service time is uniformly distributed between 6 and 12 minutes. Click on the *Set* button to make the changes permanent. The inspector panel should look as Figure 24. does.

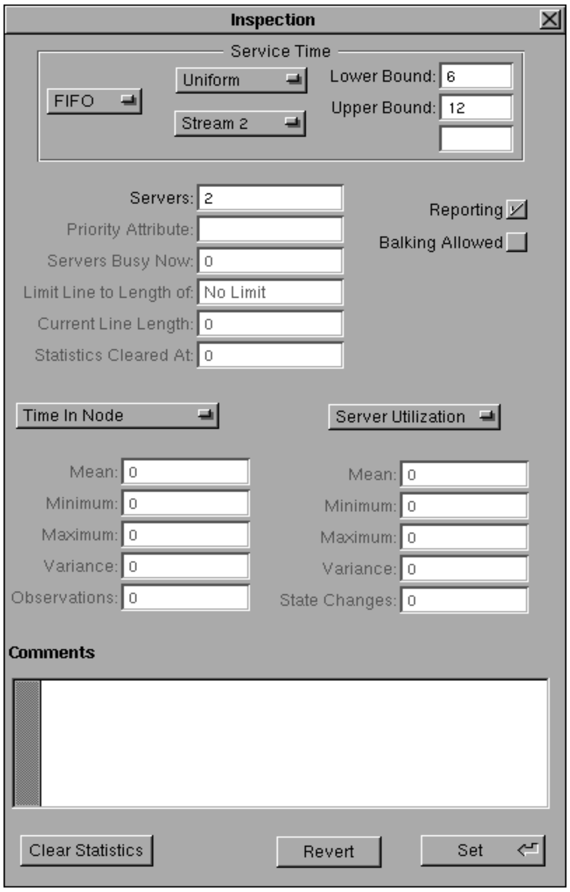


Figure 24. TV inspection queue panel.

The assignment node will put a random variable into an entity attribute and then the branch node will examine the attribute and route the entity accordingly. Since the requirement is that 85% of TV sets to exit the system and 15% of them to be sent to the adjustment station, the we can assign a random variable from the uniform (0,1) distribution to the attribute. The branch node will examine the attribute and route the entity either out of the system or to the adjustment queue.

Double click on the assignment node, which is named “Repair?” in Figure 23, and edit the inspector panel so that an attribute named “Branch RV” is assigned a random variable from the Uniform (0,1) distribution. Click on the *Set* button to make the changes permanent and then close the inspector panel.

Open the inspector panel for the branch node and enter the attribute name "Branch RV" in the left hand side of the form, then put the value 0.15 in the right hand portion of the form. Select the comparison operator "≥" from the pop-up list and then click on the *Set* button. When entities arrive at the branch node, the value associated with the attribute "Branch RV" will be compared to 0.15; if it is greater, as it should be 85% of the time, the entity will be routed to the disposal node.

Finally, edit the adjustment queue's inspector panel so that the service time is uniformly distributed between 20 and 40 minutes.

Enter "480" in the simulation window's *End Time* field. New models often have subtle bugs that aren't readily apparent, so—just to be careful—turn on the animation feature so that the paths the entities take are shown on the screen. Under the *Simulation* menu, click on the menu item that turns the animation on. Then, click on the *Run* button in the simulation window and watch the simulation execute.

The first 100 minutes or so of model execution time shows that most of the entities are going where expected. According to the specification 15% of the entities should be traveling to the adjustment node, and that looks about right.

Model verification and validation by "just watching the simulation run" is a dangerous habit to get into. It often gives rise to a false sense of security and discourages detailed scrutiny of the model. A serious simulation effort would include a much more in-depth examination of the model's correctness and its correspondence to the system being studied; however, many bugs and pitfalls that are not readily apparent in a textual simulation tool can be quickly discovered and eliminated by viewing the flow of entities through the network.

Some summary statistics on the model's performance can be viewed by double clicking on the nodes of interest and viewing the inspector panels. When the simulation finishes the inspection node shows that, on average, 1.8 servers were busy, and that TVs waited for just over two minutes to be inspected, with a maximum of 11 minutes. The branch node routed about 12% of its entities to the adjustment station. The 82 entities that exited the system spent an average of 16 minutes being inspected, adjusted, and moved, according to the summaries in the TV Exit node inspector[2].

A comparison between the results Ents produced and the results from Pritsker and Pegden's SLAM model are shown in Appendix B.

This example is not intended to serve as a complete analysis of the system's performance, but merely as a brief example to illustrate the operation of the program. The statistical results should be viewed with caution since the simulation has probably not run long enough nor been repeated often enough to obtain statistically valid results. The "warm-up period," the time during which the system has not yet reached its steady state, is probably also skewing the statistics.

To get a text report on the performance of all the nodes, select the Report item from the Simulation menu. A new window will open and a summary of each node's performance will be printed out. The text can be cut and pasted into documents from other applications in the usual NeXT Step way.

2. Quarry operations problem.

Pritsker and Pegden (1979) present another example, this time based on the operations of a quarry, and create a simulation model using SLAM. The same system is modeled here using Ents.

In a quarry, trucks are loaded from three shovels and deliver ore to a single crusher machine. Each shovel has three trucks assigned to it, one with 50 tons of capacity and two that can carry 20 tons. The trucks are loaded at the shovel, travel to the crusher and dump their load, and then return to the same shovel. A schematic diagram of the system is shown if Figure 25.

---

[2] The exact values for all the results quoted here will vary with the random number streams used and other factors.

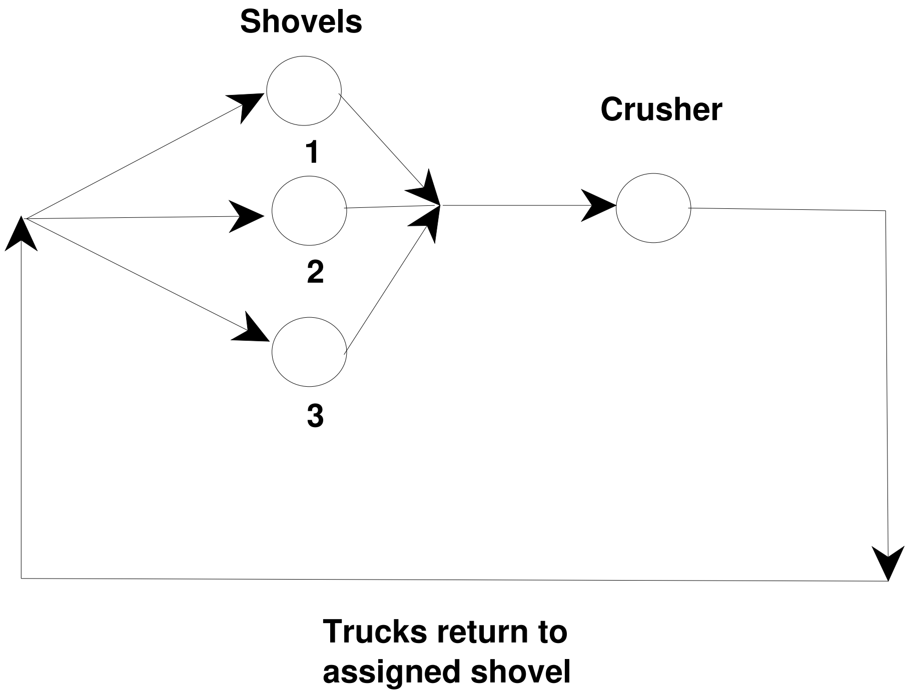


Figure 25. Schematic diagram of quarry operations.

The trucks do not all travel at the same speed or get loaded as quickly. The larger trucks take longer to fill, take longer to dump, and travel more slowly, but carry more ore. The time it takes a truck to perform each step in the operation is shown in Table 2. (Loading and dumping times are exponentially distributed random variables.)

Table 2. Quarry operation times (in minutes).

Truck Size	Loading	Travel	Dumping	Return
20 Ton	EXP(5)	2.5	EXP(2)	1.5
50 Ton	EXP(10)	3	EXP(4)	2

The shovels load the trucks on a first-come, first-served basis, but the crusher takes any 50 ton trucks that are waiting before it will unload any 20 ton trucks.

The process can be modeled rather easily with 4 queues and two activities: one queue for each of the shovels and a queue for the crusher, and activity nodes for traveling to and returning from the crusher. The entities in this model are the trucks which travel back and forth from the shovels to the crusher. However, some complexities arise: the trucks should be present at time zero and never exit the system. They must return to the same shovel from which they came, and depending on their size, take differing amounts of time to perform operations . These features require that several nodes be added to the model that perform initialization functions on the truck entities: creating them, assigning attributes that identify them as 50 ton or 20 ton trucks, and so on. A few branching nodes also need to be added to route the entities back to the shovel they are assigned to.

SLAM can add entities to a model at time zero with attribute/value pairs already defined through the use of the ENTRY statement. Ents has no counterpart to that facility, so several creation and assignment nodes need to be used to achieve the same effect.

Two creation nodes will feed into each of the shovel queues. One of the creation nodes will create two truck entities and then stop production, while the other creation node will produce just one entity. Once created the entities will have an attribute that reflects the truck size assigned (twenty to or fifty ton), and then be assigned an attribute to reflect its assigned shovel: one, two or three . Each shovel will require two creation nodes, for a total of six creation nodes for the entire model.

The truck entities also need to be assigned attributes that correspond to the times for the various quarry operations, depending on their carrying capacity. An attribute for each of the operations in Table 6 is needed; the attributes will be used as parameters to the random number generators in the activity and queue nodes.

The nodes required to initialize all the truck entities are shown in Figure 26. The create nodes have a time between creations of zero and stop producing entities after the required number of trucks of each type have been produced: two for twenty-ton truck nodes, one for fifty-ton truck nodes. The entities are immediately assigned an attribute to reflect the carrying capacity of the truck created, and then assigned a shovel number. The network feeds into a branch node, where the entities are separated based on truck size: twenty-ton trucks take the upper branch, and fifty-ton trucks take the lower branch. In each branch the attributes “loading time,” “travel time,” “dumping time,” and “return time” are assigned fixed numbers based on the values shown in Table 6. “Loading time,” for example, is assigned a fixed value of 5 in the upper branch; the value of that attribute is later used as the expected value in the shovel queue’s exponential random number generator.

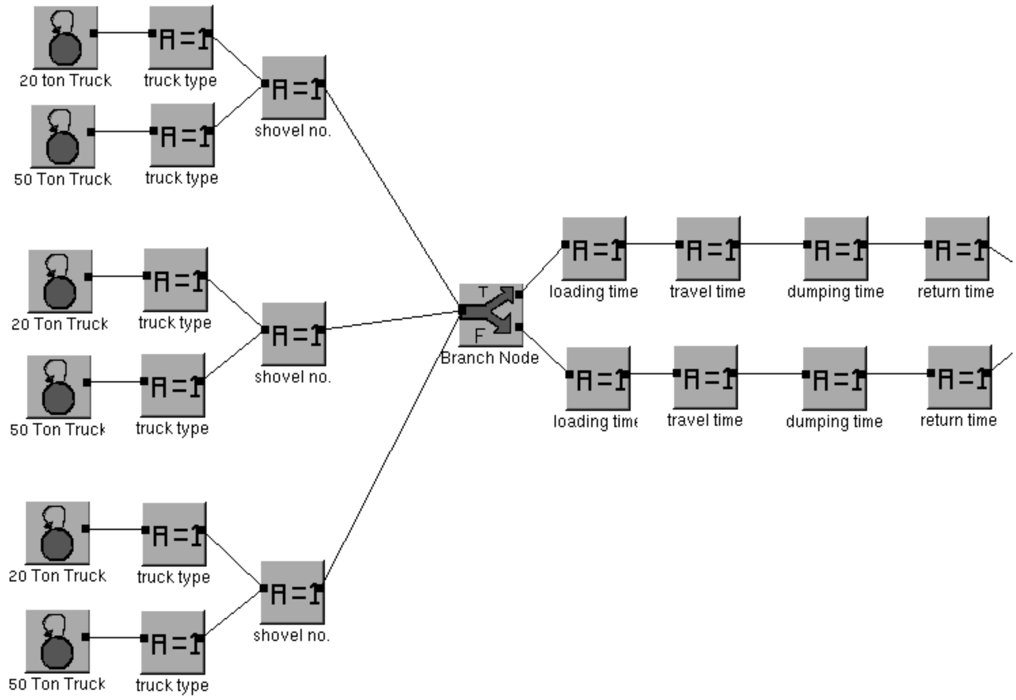


Figure 26. Initializing truck entities.

Once the entities have been initialized the real work of the simulation can begin. The nodes for this portion of the network are shown in Figure 27. (The two inputs to the first branch node are from the upper and lower branches of the initialization phase.)

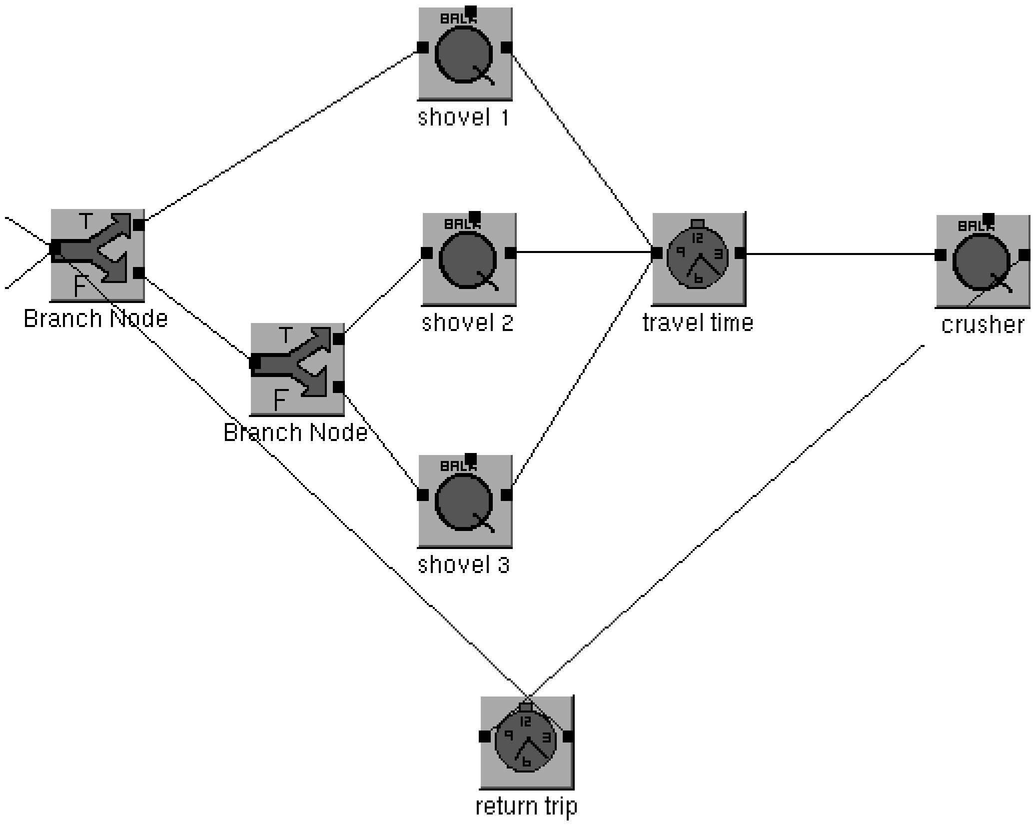


Figure 27. Quarry operations model.

The network feeds into a single branch node that routes entities based on the “shovel number” attribute. If the attribute has the value of one, it is sent to shovel queue one; otherwise, it is sent to another branch node. In that node if the attribute has a value of two, it is sent to shovel queue two. All the remaining entities must have a “shovel number” attribute of three, so they are sent to the third shovel.

At the shovels the service time is specified to be exponential with a mean of “loading time,” the attribute assigned earlier in the initialization phase of the network. All the entities will have a “loading time” attribute with a value of 5.0 for the twenty-ton trucks or 10.0 for the fifty-ton trucks. The node looks up the value associated with that attribute and uses it to calculate a random service length.

When the entity has completed service it enters an activity node . The length of the activity is a fixed or constant value, “travel time,” another attribute that was assigned a value in the initialization phase.

The truck entities wait for service at the Crusher queue node. The queue is ordered by the attribute “truck capacity” so that fifty-ton trucks are served before any twenty-ton trucks, and the service time is set to be exponential with a mean of “dumping time.” The entities then travel to another activity node, where they are delayed by a fixed amount “return time,” then routed back to the original branch node.

To follow the events as they occur, select the *Trace Execution* item from the *Simulation* menu. A window similar to what appears in Figure 28 will be created. As the program runs, each event dispatched by the executive is displayed in the window. At each node, the system clock time, the type of event, and the attribute/value pairs associated with the entity are shown.

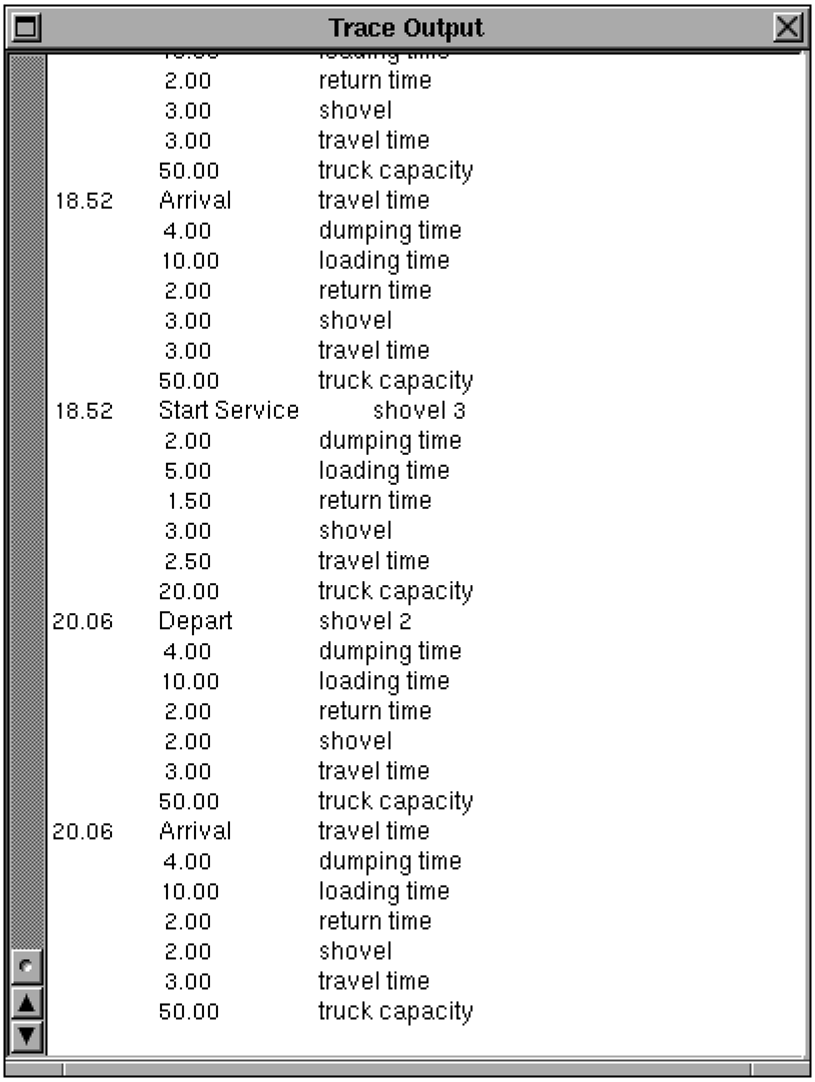


Figure 28. Tracing program execution.

The model's performance can be examined by double-clicking on a node and examining the summary statistics, or by selecting the *Report* item from the *Simulation* menu. The inspector panel for the return trip activity node after the simulation had run for 480 minutes is shown below in Figure 29.

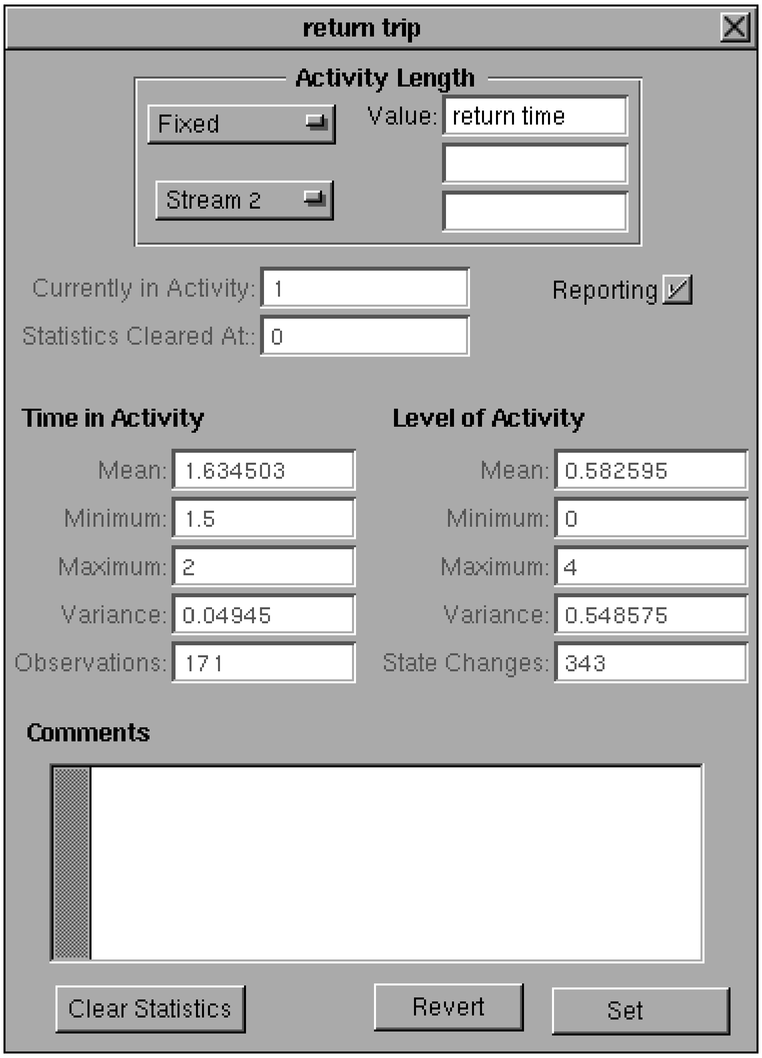


Figure 29. Return trip inspector panel.

On average, 0.59 trucks are returning to their shovels at any one time, and the trip takes a mean of 1.6 minutes.

The results of the SLAM simulation and the Ents simulation are compared in Appendix B.