CHAPTER 4

Modeling Basic Operations and Inputs

In Chapters 2 and 3, we introduced you to a simple processing system (Model 3-1), conducted a hand simulation (Chapter 2), and examined an Arena model (Chapter 3). In this chapter, we'll embellish this simple system to represent a more realistic environment and develop a complete model of that system, including specification of the input probability distributions.

Section 4.1 describes this more complicated system—a sealed electronic assembly and test system. We then discuss how to develop a modeling approach, introduce several new Arena concepts, build the model, and show you how to run it and view the results. By this time, you should start to become dangerous in your modeling skills. In Section 4.2, we'll enhance the model by introducing new modeling concepts and give you alternate methods for studying the results. Section 4.3 shows you how to dress up the animation a little bit. In Section 4.4, we'll take up the issue of how you specify quantitative inputs, including probability distributions from which observations on random variables are "generated," to drive your simulation. When you finish this chapter, you should be able to build some reasonably elaborate models of your own, as well as specify appropriate and realistic distributions as input to your models.

4.1 Model 4-1: An Electronic Assembly and Test System

This system represents the final operations of the production of two different sealed electronic units in Figure 4-1. The arriving parts are cast metal cases that have already been machined to accept the electronic parts.

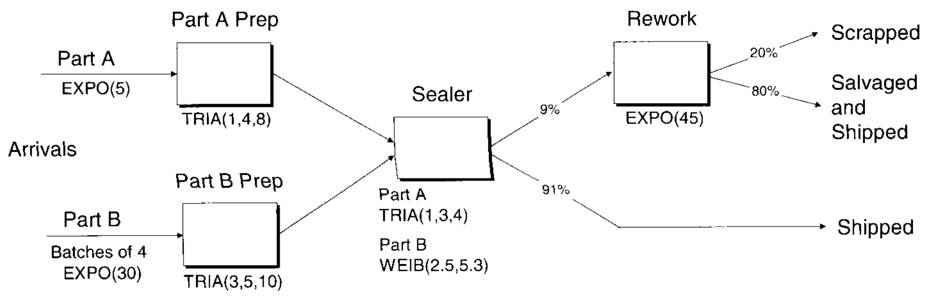


Figure 4-1. Electronic Assembly and Test System

The first units, named Part A, are produced in an adjacent department, outside the bounds of this model, with interarrival times to our model being exponentially distributed with a mean of 5 minutes. Upon arrival, they're transferred to the Part A Prep area. At the Part A Prep area, the mating faces of the cases are machined to assure a good seal, and the part is then deburred and cleaned; the process time for the combined operation at the Part A Prep area follows a TRIA(1, 4, 8) distribution. The part is then transferred to the sealer.

The second units, named Part B, are produced in a different building, also outside this model's bounds, where they are held until a batch of four units is available; the batch is then sent to the final production area we are modeling. The time between the arrivals of successive batches of Part B to our model is exponential with a mean of 30 minutes. Upon arrival at the Part B Prep area, the batch is separated into the four individual units, which are processed individually from here on. The processing at the Part B Prep area has the same three steps as at the Part A Prep area, except that the process time for the combined operation follows a TRIA(3, 5, 10) distribution. The part is then sent to the sealer.

At the sealer operation, the electronic components are inserted, the case is assembled and sealed, and the sealed unit is tested. The total process time for these operations depends on the part type: TRIA(1, 3, 4) for Part A and WEIB(2.5, 5.3) for Part B (2.5 is the scale parameter β and 5.3 is the shape parameter α; see Appendix D). Ninety-one percent of the parts pass the inspection and are transferred directly to the shipping department; whether a part passes is independent of whether any other parts pass. The remaining parts are transferred to the rework area where they are disassembled, repaired, cleaned, assembled, and re-tested. Eighty percent of the parts processed at the rework oven are salvaged and transferred to the shipping department as reworked parts, and the rest are transferred to the scrap area. The time to rework a part follows an exponential distribution with mean of 45 minutes and is independent of part type and the ultimate disposition (salvaged or scrapped).

We want to collect statistics in each area on resource utilization, number in queue, time in queue, and the cycle time (or total time in system) by shipped parts, salvaged parts, or scrapped parts. We will initially run the simulation for four 8-hour shifts, or 1,920 minutes.

4.1.1 Developing a Modeling Approach

Building a simulation model is only one component of a complete simulation project. We will discuss the entire simulation project in Chapter 12. Presume for now that the first two activities are to state the study objective and define the system to be studied. In this case, our objective is to teach you how to develop a simulation model using Arena. The system definition was given above. In the real world, you would have to develop that definition, and you may also have to collect and analyze the data to be used to specify the input parameters and distributions (see Secion 4.4). We recommend that the next activity be the development of a modeling approach. For a real problem, this may require the definition of a data structure, the segmentation of the system into submodels, or the development of control logic. For this problem, it only requires that we decide which Arena modules will provide the capabilities we need to capture the operation of the system at an appropriate level of detail. In addition, we must decide how we're going to model the different processing times at the sealer operation. To simplify this task, let's separate the model into the following components: part arrival, prep areas, sealer operation, rework, part departure, and part animation. Also, we'll assume that all entities in the system represent parts that are being processed.

Because we have two distinct streams of arriving entities to our model, each with its own timing pattern, we will use two separate Create modules (one for each part type) to generate the arriving parts.

We also have different processing times by part type at the sealer operation, so we'll use two Assign modules to define an attribute called `Sealer Time` that will be assigned the appropriate sealer processing time after the parts are generated by the Create modules. When the parts are processed at the sealer operation, we'll use the time contained in the `Sealer Time` attribute for the processing time there.

Each of the two prep areas and the sealer operation will be modeled with its own Process module, very much like the Process module used in the simple processing system of Model 3-1. There is an inspection performed after the sealer operation has been completed, which results in parts going to different places based on a "coin flip" (with just the right bias in the coin). We'll use a Decide module with the pass or fail result being based on the coin flip. The rework area will be modeled with Process and Decide modules, as it also has a pass or fail option. The part departures will be modeled with three separate Record and Dispose modules (shipped, salvaged, and scrapped) so we can keep corresponding cycle-time statistics sorted out by shipped vs. salvaged vs. scrapped. All of these modules can be found on the Basic Process panel.

4.1.2 Building the Model

To build the model, you need to open a new model window and place the required modules on the screen: two Create, two Assign, four Process, two Decide, three Record, and three Dispose modules.

Your model window should now look something like Figure 4-2, assuming you've made the Connections or used the Auto-Connect feature (Object menu) while placing the modules in the appropriate sequence. At this point, you might want to use the *File/Save* function to save your model under a name of your choosing.

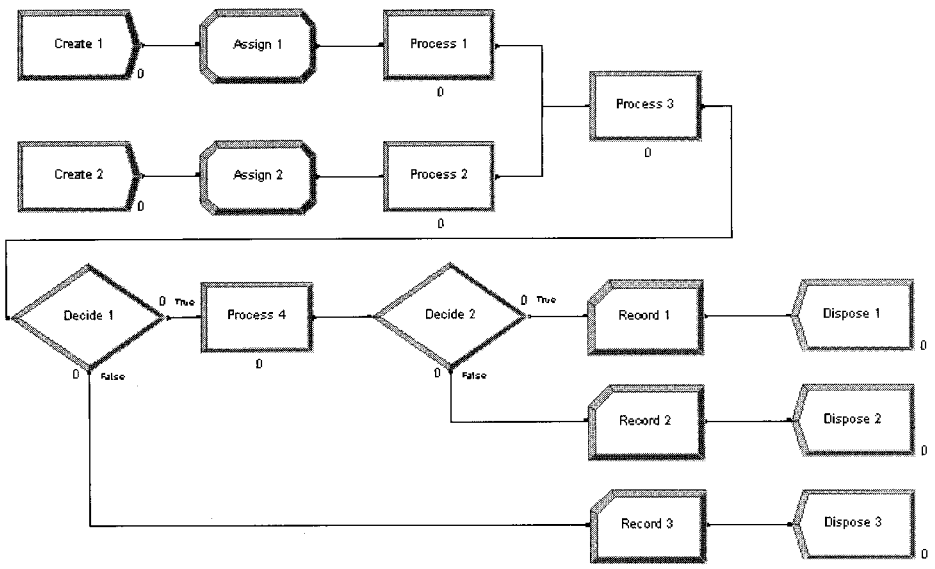
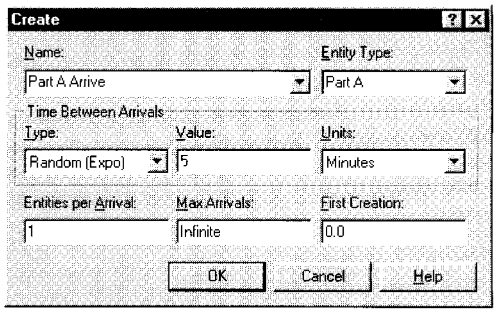


Figure 4-2. Model Window of Placed Modules

Now let's open each module and enter the information required to complete the model. Start with the Create 1 module that will create the arriving Part A entities. Display 4-1 (the "Display" device was described in Section 3.4.4) provides the information required to complete this module. Note that this is very similar to the Create module used in Model 3-1. We've given the module a different Name and specified the Entity Type as `Part A`. The Time Between Arrivals is Random (i.e., an exponential distribution) with a Value (i.e., mean) of 5, and the units are set to minutes. The remaining entries are the default options. We can now accept the module by clicking OK.



Name: Part A Arrive
Entity Type: Part A
Time Between Arrivals
Type: Random (Expo)
Value: 5
Units: Minutes
Entities per Arrival: 1
Max Arrivals: Infinite
First Creation: 0.0

Name	`Part A Arrive`
Entity Type	`Part A`
Type	`Random (Expo)`
Value	`5`
Units	`Minutes`

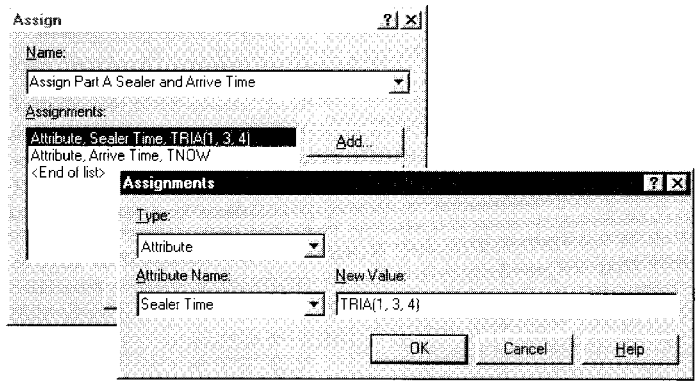
Display 4-1. The Completed Part A Create Dialog

The Create module for the Part B arrivals is very similar to that for Part A, as shown in Display 4-2 (we'll skip the graphics since they're almost the same as what you just saw), except we have filled in one additional field (Entities per Arrival) to reflect the batch size of 4. Recall that the Part B entities arrive in batches of four. Thus, this entry will cause each arrival to consist of four separate entities rather than one.

Name	`Part B Arrive`
Entity Type	`Part B`
Type	`Random (Expo)`
Value	`30`
Units	`Minutes`
Entities per Arrival	`4`

Display 4-2. The Completed Part B Create Dialog Entries

Having created the arriving parts, we must next define an attribute `Sealer Time` and assign it the sealer processing time, which is different for each part type. We'll assign these values in the Assign 1 and Assign 2 modules that we previously placed. The Part A assignment is shown in Display 4-3. We've defined the new attribute and assigned it a value from a TRIA(1, 3, 4) distribution. We've also defined an attribute, `Arrive Time`, which is used to record the arrival time of the entity. The Arena variable TNOW provides the current simulation time, which in this case is the time the part arrived or was created.


Assign

Name: Assign Part A Sealer and Arrive Time

Assignments:
Attribute, Sealer Time, TRIA(1, 3, 4)
Attribute, Arrive Time, TNOW
<End of list>

Assignments

Type: Attribute

Attribute Name: Sealer Time

New Value: TRIA(1, 3, 4)

Name	`Assign Part A Sealer and Arrive Time`
Type	`Attribute`
Attribute Name	`Sealer Time`
New Value	`TRIA(1, 3, 4)`
Type	`Attribute`
Attribute Name	`Arrive Time`
New Value	`TNOW`

Display 4-3. Assigning the Part A Sealer Time and Arrival Time

The assignment to the `Sealer Time` and `Arrive Time` attributes for Part B is shown in Display 4-4. Although four entities are created in the previous module for each arrival, they'll each be assigned a different value from the sealer-time distribution in the following Assign module.

Name	`Assign Part B Sealer and Arrive Time`
Type	`Attribute`
Attribute Name	`Sealer Time`
New Value	`WEIB(2.5, 5.3)`
Type	`Attribute`
Attribute Name	`Arrive Time`
New Value	`TNOW`

Display 4-4. Assigning the Part B Sealer Time and Arrival Time

Having completed the two part-arrival modules and the assignment of the sealer time, we can now move to the two prep areas that are to be modeled using the two Process modules previously placed. The completed dialog for the `Prep A Process` area is given in Display 4-5.

The Process module has four possible Actions: Delay, Seize Delay, Seize Delay Release, and Delay Release. The Delay action will cause an entity to undergo a specified time Delay. This Action does not require a Resource. This implies that waiting will occur and that multiple entities could undergo the Delay simultaneously. Since our prep area requires the use of a machine or Resource, we need an Action that will allow for waiting, queueing until the prep resource is available, and delaying for the processing time. The Seize Delay Action provides the waiting and delaying, but it does not release the Resource to be available for the next entity. If you use this Action, it is assumed that the Resource would be Released at a later time at another module. The Seize Delay Release option provides the set of Actions required to model our prep area accurately. The last action, Delay Release, assumes that the entity previously Seized a Resource and will cause a Delay, followed by the Release of the Resource.

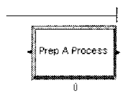
You might notice that when you select one of the last three options, a list box appears in the empty space below the Action selection box. You will need to click on the Add button to enter the Resource information.

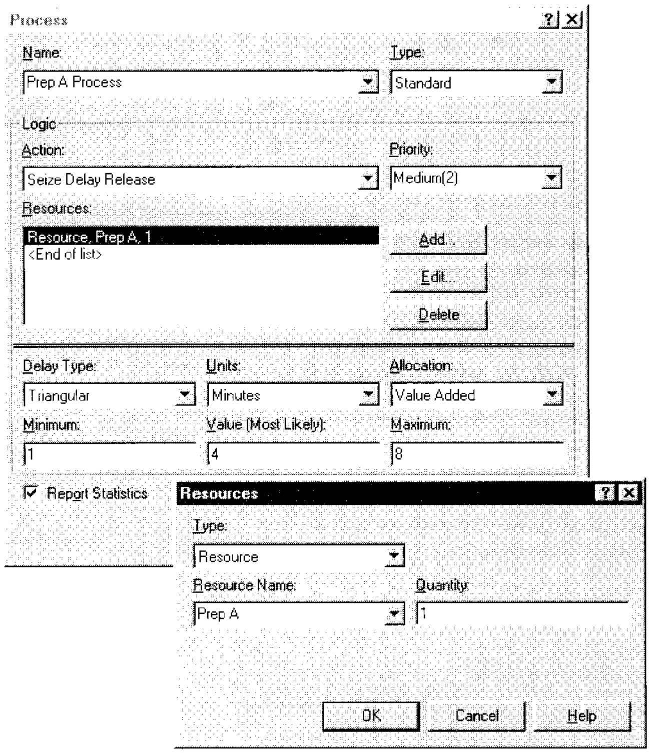
In entering data, we strongly urge you to make use of pull-down lists whenever possible. The reason for this caution is that once you type a name, you must always match what you typed the first time. Arena names are not case-sensitive, but the spelling and any embedded blanks must be identical. Picking the name from the list assures that it is the same. If you type in a slightly different name, Arena will give you an error message the first time you check or attempt to run the model.

Also note that when you place a module, Arena automatically provides default names and values. These default names are the object name (module, resource, etc.) with an appended number. The appended number is incremented for each additional name, if a unique name is required; e.g., Process 1, Process 2, and so on. There are two reasons for this. The first is a matter of convenience—you can accept the default resource name, or you can change it. The second reason is that all names for any objects in Arena must be unique, even if the object type is different. Otherwise, Arena could not determine which object to associate with a name that had been used more than once.

To help you, Arena does a lot of automatic naming, most of which you won't even notice. For example, if you click on the Queue data module, you'll see that Arena also assigned the name `Prep A Process.Queue` to the queue at this prep area. In most cases, you can always assign your own names rather than accepting the default names.

You might also notice that when you select either of the two actions that include a Seize and then accept the module, Arena will automatically place an animated queue (a horizontal line with a small vertical line at the right) near the associated Process module. This will allow you to visualize entities waiting in the queue during the simulation run. If you click on this queue, the queue name will be displayed.





Process

Name: Prep A Process
Type: Standard

Logic
Action: Seize Delay Release
Priority: Medium(2)
Resources:
Resource, Prep A, 1
<End of list>

Delay Type: Triangular
Units: Minutes
Allocation: Value Added
Minimum: 1
Value (Most Likely): 4
Maximum: 8

[✓ Report Statistics]

Resources

Type: Resource
Resource Name: Prep A
Quantity: 1

Name	`Prep A Process`
Action	`Seize Delay Release`
Resources	
Type	`Resource`
Resource Name	`Prep A`
Quantity	`1`
Delay Type	`Triangular`
Units	`Minutes`
Minimum	`1`
Value (Most Likely)	`4`
Maximum	`8`

Display 4-5. Prep A Process Dialog

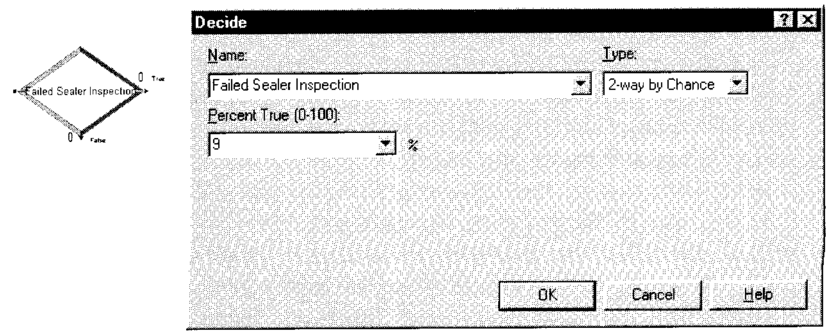
The second Process module is filled out in an almost-identical fashion, with the exception of the name (`Prep B Process`), the resource name (`Prep B`), and the parameters for the process time (`3, 5, 10`). We have not included a display for this module.

The next step is to enter data for the sealer operation, which is the third Process module we placed. The entries for the dialog are shown in Display 4-6. Note that in the upstream Assign modules we defined the attribute `Sealer Time` when the arriving parts were created. When an entity gains control of, or *seizes*, the resource, it will undergo a process delay equal to the value contained in the `Sealer Time` attribute.

Name	`Sealer Process`
Action	`Seize Delay Release`
Resources	
Resource Name	`Sealer`
Quantity	`1`
Delay Type	`Expression`
Units	`Minutes`
Expression	`Sealer Time`

Display 4-6. The Sealer Dialog

The inspection following the sealer operation is modeled using the first Decide module. We'll accept the default Type, `2-way by Chance`, as we only have a pass or fail option. The dialog requires that we enter a Percent True, and it provides two ways to leave the module—True or False. In this case, we will enter the Percent True as `9` percent. This will result in 9 percent of the entities (which we'll treat as the failed items) following the True branch, and 91 percent (the passed items) following the False branch.[1] Parts that pass are sent to Shipping, and parts that fail are sent to Rework. The data for this Decide module are shown in Display 4-7. By the way, if you've been building this model as we've moved through the material, now would be a good time to click on the Save button—you never know when somebody might bump the power switch!



Name: Failed Sealer Inspection
Type: 2-way by Chance
Percent True (0-100): 9 %

(Display 4-7 continued on next page)

[1] We could have just as well reversed the interpretation of True as fail and False as pass, in which case the Percent True would be `91` (and we'd probably change the module Name to `Passed Sealer Inspection`).

Name	`Failed Sealer Inspection`
Percent True	`9`

Display 4-7. The Sealer Inspection Dialog

The remaining Process module will be used to model the rework activity. The data for this Process module are in Display 4-8. This module is very similar to the Prep A and B Process modules with the exceptions of the Name, Resource Name, and Expression.

Name	`Rework Process`
Action	`Seize Delay Release`
Resources	
Resource Name	`Rework`
Quantity	`1`
Delay Type	`Expression`
Units	`Minutes`
Expression	`EXPO(45)`

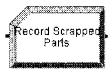
Display 4-8. The Rework Process Dialog

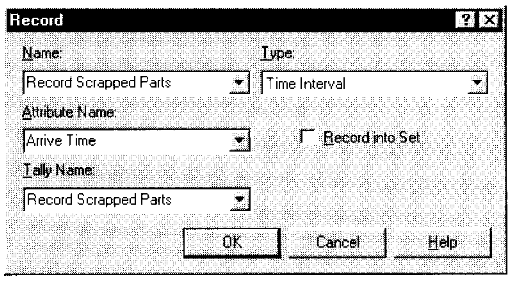
The final Decide module is used to separate the salvaged and scrapped parts following rework. The data for this Decide module are in Display 4-9. We've chosen the True branch to represent the scrapped parts (20 percent) and the False branch to represent the salvaged parts.

Name	`Failed Rework Inspection`
Percent True	`20`

Display 4-9. The Rework Inspection Dialog

Having defined all of the operations, we now need to fill in the Record and Dispose modules. Remember that as part of the simulation output, we wanted to collect statistics on resource utilization, number in queue, and time in queue at each of the operations. These three statistics are automatically collected whenever you use a Process module with an Action option that requires a Resource. We also wanted statistics on the cycle time separated by shipped parts, salvaged parts, and scrapped parts. The Record module provides the ability to collect these cycle times in the form of Tallies. The completed dialog for the scrapped parts tally is shown in Display 4-10. We picked the Type `Time Interval` from the pull-down list. The Tally Name defaults to the module name. This will cause Arena to record as a Tally statistic the time between the arrival (`Arrive Time`) of the part to the system and the time that it arrived at this Record module.





Name	`Record Scrapped Parts`
Type	`Time Interval`
Attribute Name	`Arrive Time`
Tally Name	`Record Scrapped Parts`

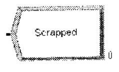
Display 4-10. The Scrapped Parts Tally Dialog

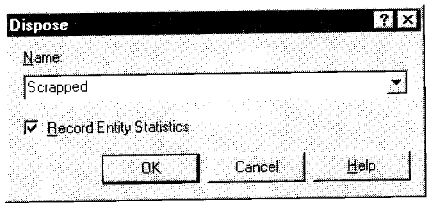
The remaining two Record modules are named `Record Salvaged Parts` and `Record Shipped Parts`. We have not bothered to include displays on these modules as they are completely analogous to the `Record Shipped Parts` module.

The final three modules Dispose of the entities as they leave the system. For this model, we could have directed all entities to a single Dispose module. However, one of the features of a Dispose module is the inclusion of an animation variable, which appears near the lower right-hand portion of the module. This animation variable will display the current count for the total number of entities that have passed through this module during the run and allow the viewer to discern the proportion of entities that have taken each of the three possible paths through the system.

The data for the `Scrapped` Dispose module are shown in Display 4-11. We have defaulted on the check in the box entitled Record Entity Statistics. However, if we had wanted to keep entity flow statistics only on the parts that were shipped, including the salvaged parts, then we could have unchecked this box (leaving checks in the remaining two Dispose modules). Doing so would have caused only the selected parts to be included in the automatic entity-flow statistics.

The two other Dispose modules, `Salvaged` and `Shipped`, are filled out in a similar way.




Dispose

Name: Scrapped

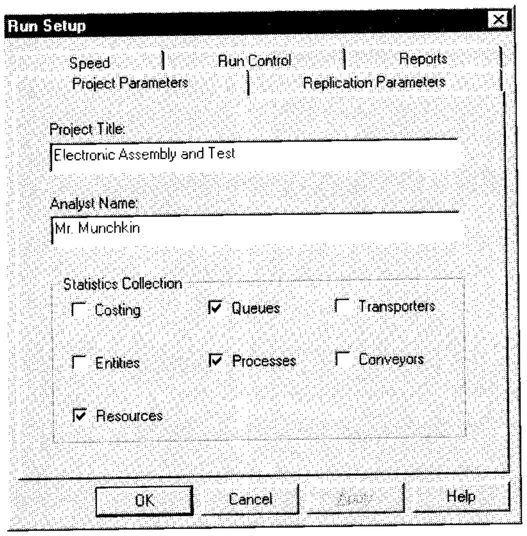
☑ Record Entity Statistics

Name	Scrapped

Display 4-11. The Scrapped Dispose Dialog

You're nearly ready to run the model. Although it has taken you some time to get this far, once you get accustomed to working with Arena, you'll find that you could have accomplished these steps in only a few minutes.

The model could actually run at this point, but once started, it would continue to run forever because Arena doesn't know when to stop the simulation. You establish the run parameters for the model by selecting the *Run/Setup* menu option. The Run Setup dialog has five tabs that can be used to control the simulation run. The data for the first tab, Project Parameters, are shown in Display 4-12. We've entered the project title and analyst name so they will appear on the output reports. In the statistics collection area, we have *unchecked* the Entities selection as we do not need those data for our analysis. You might try running the simulation with this box checked in order to see the difference in the output reports.


Run Setup

Project Title: Electronic Assembly and Test

Analyst Name: Mr. Munchkin

Statistics Collection

☐ Costing ☑ Queues ☐ Transporters

☐ Entities ☑ Processes ☐ Conveyors

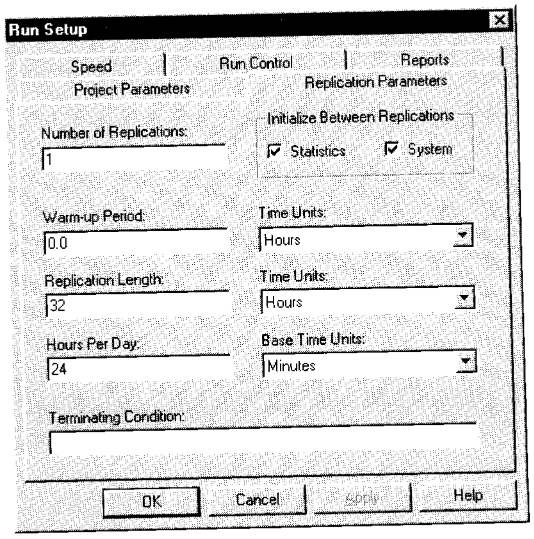
☑ Resources

(Display 4-12 continued on next page)

Project Title	Electronic Assembly and Test
Analyst Name	Mr. Munchkin
Statistics Collection	
Entities	*uncheck*

Display 4-12. The Run Setup Project Parameters

You also need to specify the run length, which is done under the Replication Parameters tab. We've set the Replication Length to `32` hours (four 8-hour shifts), the Base Time Units to `Minutes`, and defaulted the remaining fields. The completed dialog is shown in Display 4-13. We've also accepted the defaults for the remaining three tabs: Speed, Run Control, and Reports. You might want to look at these tabs to get an idea of the options available.



Initialize Between Replications: Statistics ✓ System ✓
Number of Replications: 1
Warm-up Period: 0.0
Time Units: Hours
Replication Length: 32
Time Units: Hours
Hours Per Day: 24
Base Time Units: Minutes
Terminating Condition:

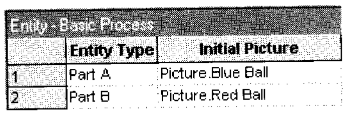
Replication Length	32
Base Time Units	Minutes

Display 4-13. The Run Setup Replication Parameters

Before we run our newly created model, let's give it one final tweak. Since we have two different part types, it might be nice if we could distinguish between them in the animation of our model. Click on the Entity data module found in the Basic Process panel and note that the initial picture for both parts is `Picture.Report`. When we run our model, all of our parts will use this same icon for displaying entities on the animation.

Now click on the Initial Picture cell for Part A, and use the pull-down list to select a different picture. We've chosen the blue ball for Part A and the red ball for Part B, as shown in Display 4-14. This will allow us to distinguish easily between the two parts in the animation. If you're interested in seeing what these icons look like, you can select *Edit/Entity Pictures* from the menu bar at the top of your screen to open the Entity Picture Placement window, which will allow you to see the icons. We'll show you later how to use this feature in more detail.

Entity



Entity - Basic Process

	Entity Type	Initial Picture
1	Part A	Picture.Blue Ball
2	Part B	Picture.Red Ball

Initial Picture (Part A)	`Picture.Blue Ball`
Initial Picture (Part B)	`Picture.Red Ball`

Display 4-14. The Entity Data Module

Your final model should look something like Figure 4-3.

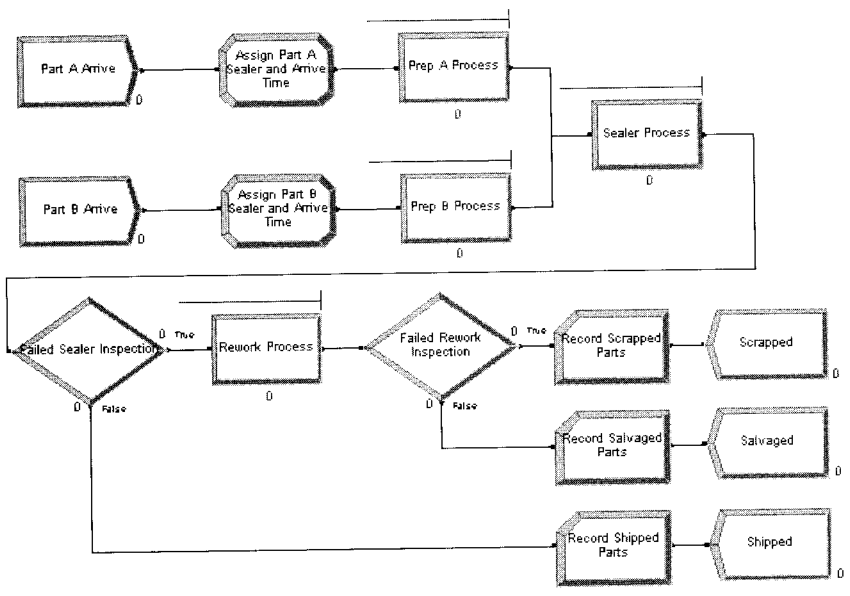


Figure 4-3. The Final Model 4-1

4.1.3 Running the Model

Before running your model, you might want to check it for errors. You can do this by clicking the Check button (✓) on the Run Interaction toolbar, the *Run/Check Model* option, or the F4 key on the keyboard. With a little luck, the response will be a small window with the message "No errors or warnings in model." If you have no luck at all, an error window will open with a message describing the error. If this occurs, you might want to select the Find option, if the button is enabled. This feature attempts to point you to where Arena thinks the error might be. We suggest you intentionally insert an error into your model and try these features. As you build more complex models, you might just find yourself using these features quite often.

If your model check results in no errors, you're now ready to run the simulation. There are four ways to run a simulation, but we'll only talk about three of them here. The first way is to run the simulation with the animation. Use the Go button (▶) on the Run toolbar, the *Run/Go* option, or the F5 key. If you've not yet checked your model or if you've made a change since the last check, Arena will first check the model, then initialize the model with your data, and finally run it. You'll notice during the run that Arena hides some of the graphics so that your focus is on the animation. Don't worry, though. They'll return when you end the run (or you can check to see that they're still there by using the *View/Layers* option).

If you leave the status bar active (at the bottom of the screen), you can tell what Arena is doing. Toward the right of this bar are three pieces of information: the replication number, the current simulation time, and the simulation status.

After the simulation starts to run, you may want to speed up or slow down the animation. You can do this while the model is running by pressing the "<" key to slow it down or the ">" key to speed it up. If you press one of these keys, the current Animation Speed Factor is displayed at the far left of the status bar. You can also increase or decrease the animation speed factor in the Speed tab of the Run Setup dialog (the *Run/Setup* menu option). This option can also be used to enter an exact speed factor.

During the simulation run, you can also pause the simulation using the Pause button (⏸) on the Run toolbar, the *Run/Pause* option, or the Esc key. This temporarily suspends the simulation, and the message "User interrupted" will appear on the status bar.

While you're in Pause mode, you might want to double-click on one of the entities that is visible in the animation. An Entity Summary dialog lists the values of each of the entity's attributes. This can be a very useful feature when trying to debug a model. You can also use the Step button (⏯) on the Run toolbar to move entities through the system one step at a time. You can continue the simulation run at any time with the Go button.

This method of running a simulation provides the greatest amount of information, but it can take a long time to finish. In this case, the time required to complete the run depends on the Speed Factor. You can skip ahead in time by Pausing and then selecting the Fast-Forward button (⏩) on the Run toolbar, or *Run/Fast-Forward*. This will cause the simulation to run at a much faster speed by not updating the animation graphics. At any time during the run, you can Pause and return to the animation mode, or you can Zoom In (+), Zoom Out (–), or move about in the simulation window (arrow keys or scroll bars).

Using Fast-Forward will run the simulation in much less time, but if you're only interested in the numerical simulation results, you might want to disable the animation (computation and graphics update) altogether. You do this with the *Run/Run Control/ Batch Run* option.

The *Run/Run Control* option also allows you to configure a number of other runtime options. For now, select the *Batch Run* option (No Animation). Note that if you revisit this option, there is a check to the left. Accept this option and click the Run button. Note how much faster the simulation runs. The only disadvantage is that you must terminate the run and reset the animation settings in order to get the animation back. If you have large models or long runs and you're only interested in the numerical results, this is the option to choose since it is even faster than Fast-Forward.

While you're building a model, you should probably have most of the toolbars visible and accessible. However, when you're running a model, many of the toolbars simply consume space because they are not active during runtime. Arena recognizes this and will save your toolbar settings for each mode. To take advantage of this, pause during the run and remove the toolbars that you don't want to have active while the simulation is running. When you end the run, these toolbars will reappear.

4.1.4 Viewing the Results

If you selected the *Run/Go* menu option (or the Go button, ▶), you might have noticed that, in addition to the blue and red balls (our Part A and B entities) moving through the system, there are several animated counters being incremented during the simulation run. There is a single counter for each Create, Process, and Dispose module and two counters for each Decide module. The counters for the Create, Dispose, and Decide modules are incremented each time an entity exits the module. In the case of the Process modules, the counter is the number of entities that are currently at that module, including any entities in the queue waiting for the resource and the entity currently in process. If you selected the *Run/Fast-Forward* menu option (or the Fast-Forward button, ⏩), these counters (and the entities in the queues) will be updated at the end of the run and whenever you pause or change views of the model. The final numbers resulting from our simulation are shown in Figure 4-4.

If you run the model to completion, Arena will ask if you want to see the results. If you select *Yes*, you should get a window showing the Category Overview Report (the default report). When the first page of the report appears, you might find it strange that the message "No Summary Statistics Are Available" is displayed. The system summary statistics are entity and costing statistics, which we elected not to collect when we *unchecked* the Entities selection in the Project Parameters tab of the Run Setup dialog (see Display 4-12). At some point, you might want to change these selections and view the difference in the reports.

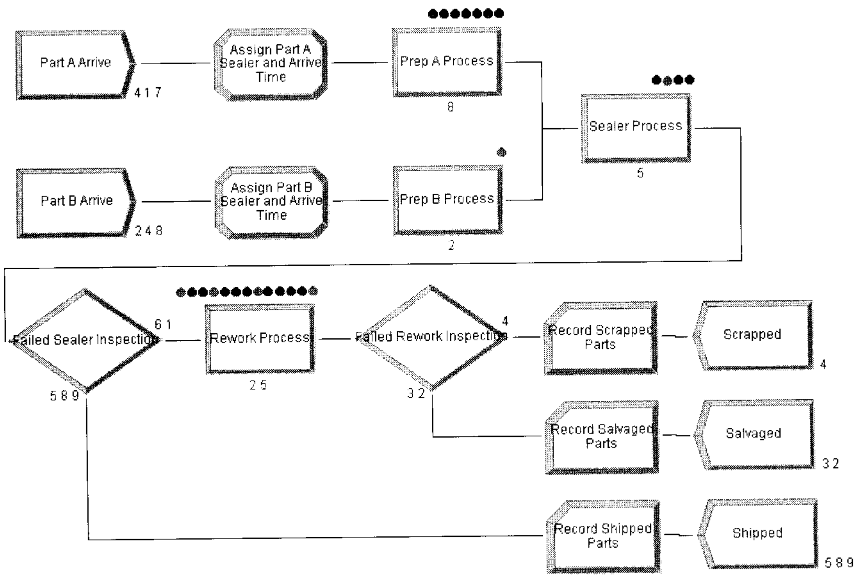


Figure 4-4. The Animated Results for Model 4-1

Recall that you can navigate through the report using the tree listing under the Preview tab at the left side of the report window or by using the arrow buttons (⏮◀ ▶⏭) in the upper-left corner of the report window. This report will provide statistics by the categories selected in the Run Setup dialog (Project Parameters tab, Statistics Collection area). For our model, you will find sections on Process, Queue, Resource, and User Specified. The User Specified section was automatically created by Arena because we included Record modules in our model to collect statistics on the cycle times sorted by departure type.

As in Chapter 3, you will find three types of statistics in our report: *tally*, *time-persistent*, and *counter*. A fourth statistic (*outputs*) is available when multiple replications are made (we'll talk about them later, when we do multiple replications). The tally statistics here include several process times, queue times, and the interval times collected by our Record modules. The time-persistent statistics include number waiting in queue, resource usage, and resource utilization. Counters include accumulated time, number in, number out, and number of times used.

The tally and time-persistent statistics provide the average, 95% confidence-interval half width, and the minimum and maximum observed values. With the exception of the half-width column, these entries are as described in Chapters 2 and 3, and should be self explanatory.

At the end of each replication, Arena attempts to calculate a 95% confidence-interval half width for the steady-state (long-run) expected value of each observed statistic, using a method called *batch means* (see Section 6.3.3). Arena first checks to see if sufficient data have been collected to test the critical statistical assumption (uncorrelated batches) required for the batch-means method. If not, the annotation "Insufficient" appears in the report and no confidence-interval half width is produced, as can be seen for several of the results. If there are enough data to test for uncorrelated batches, but the test is failed, the annotation "Correlated" appears and once again there's no half width, which is the case for the remainder of the results. If there were enough data to perform the test for uncorrelated batches *and* the test were passed, the half width (the "plus-or-minus" amount) of a 95% confidence interval on the long-run (steady-state) expected value of the statistic would be given (this does not happen for any of the results in this run). In this way, Arena refuses to report unreliable half-width values even though it could do so from a purely computational viewpoint. The details and importance of these tests are further discussed in Section 6.3.3.

Usage

Number Busy	Average	Half Width	Minimum Value	Maximum Value
Prep A	0.9038	0.061087502	0.00	1.0000
Prep B	0.7575	(Insufficient)	0.00	1.0000
Rework	0.9495	(Insufficient)	0.00	1.0000
Sealer	0.8595	0.043516684	0.00	1.0000

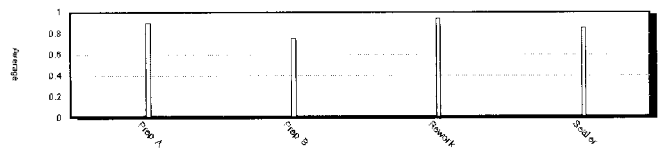


Figure 4-5. The Arena Resource Usage Summary Report: Model 4-1

Trying to draw conclusions from this single short run could be misleading because we haven't yet addressed issues like run length, number of replications, or even whether long-run steady-state results are appropriate (but we will, in Section 6.3). However, if you look closely at the results (Figure 4-5), you should note that the rework resource is busy almost 95% of the time, and the rework process has 25 parts (24 in the queue and one in process) at the end of the simulation. This number is displayed on our animation at the end of the run (see Figure 4-4). It can also be calculated by taking the difference between the Number In and Number Out counters found in the Process section of our report. This implies either that the rework area doesn't have enough capacity to handle its work, or there is a great deal of variability at this station. We'll address this issue in the next section.

4.2 Model 4-2: The Enhanced Electronic Assembly and Test System

Having constructed and run our model, the next activity would be to verify and validate that the model really represents the system being studied (see Section 2.7). For this example, that's fairly easy. We can examine the logic constructs we selected from the modules we used and compare them to the problem definition. With much larger and more complex systems, this can become a challenging task. An animation is often very useful during the verification and validation phases because it allows you to view the entire system being modeled as it operates. If you ran the model we developed and viewed its animation, you should have noted that it appeared to operate quite similarly to the way we described the system. Although verification can be very difficult, complete validation (the next activity) can sometimes be almost impossible. That's because validation implies that the simulation is behaving just like the real-world system, which may not even exist. And even if the system does exist, you have to have output performance data from it, as well as convince yourself and other nonbelievers that your model can really capture and predict the events of the real system. We'll discuss both of these activities in much more detail in Chapter 12.

For now, let's assume that as part of this effort you showed the model and its accompanying results to the production manager. Her first observation was that you didn't have a complete definition of how the system works. Whoever developed the problem definition looked only at the operation of the first shift. This system actually operates two shifts a day, and on the second shift, there are two operators assigned to the rework operation. This would explain our earlier observation when we thought the rework operation might not have enough capacity. The production manager also noted that she has a failure problem at the sealer operation. Periodically, the sealer machine breaks down. Engineering looked at the problem some time ago and collected data to determine the effect on the sealer operation. They felt that these failures did not merit any significant effort to correct the problem because they didn't feel that the sealer operation was a bottleneck. However, they did log their observations, which are still available. Let's assume that the mean uptime between failures was found to be 120 minutes and that the distribution is exponential (which, by the way, is often used as a realistic model for uptimes if failures occur randomly at a uniform rate over time). The time to repair also follows an exponential distribution with a mean of 4 minutes.

In addition, the production manager indicated that she was considering purchasing special racks to store the waiting parts in the rework area. These racks can hold 10 assemblies per rack, and we would like to know how many racks to buy. Our next step is to modify the model to include these three new aspects, which will allow us to employ some additional Arena features.

In order to incorporate these changes into our model, we'll need to introduce several new concepts. Changing from a one- to a two-shift operation is fairly easy. In Model 4-1, we set our run length to four 8-hour shifts and made no attempt to keep track of the day/shift during the run. We just assumed that the system conditions at the end of a shift were the same at the start of the next shift and ignored the intervening time. Now we need to model explicitly the change in shifts, because we have only one operator in the first shift and two in the second shift for the rework process.

We'll add this to our model by including a Resource Schedule for the rework resource, which will automatically change the number of rework resources throughout the run by adjusting the resource capacity. While we're making this change, we'll also increase the run length so that we simulate more than just two days of a two-shift operation. We'll model the sealer failures using a Resource Failure, which allows us to change the available capacity of the resource (much like the Resource Schedule), but has additional features specifically designed for representing equipment failures. Finally, we'll use the Frequencies statistic to obtain the type of information we need to determine the number of racks that should be purchased.

4.2.1 Expanding Resource Representation: Schedules and States

So far we've modeled our resources (prep area, sealer, and rework) as a single resource with a capacity of 1. You might recall that we defaulted all of this information in the Resource module. To model the additional rework operator, we could simply change the capacity of the rework resource to 2, but this would mean that we would *always* have two operators available. What we need to do is to schedule one rework operator for the first shift (assume each shift is 8 hours) and two rework operators for the second shift. Arena has a built-in construct to model this, called a *Schedule*, that allows you to vary the capacity of a resource over time. A resource Schedule is defined by a sequence of time-dependent resource capacity changes.

We also need to capture in our model the periodic breakdowns (or failures) of the sealer machine. This could be modeled using a Schedule, which would define an available resource capacity of 1 for the uptimes and a capacity of 0 for the time to repair. However, there is a built-in construct designed specifically to model failures. First, let's introduce the concept of *Resource States.*

Arena automatically has four Resource States: *Idle*, *Busy*, *Inactive*, and *Failed.* For statistical reporting, Arena keeps track of the time the resource was in each of the four states. The resource is said to be Idle if no entity has seized it. As soon as an entity enters the Process module and seizes the resource, the state is changed to Busy. The state will be changed to Inactive if Arena has made the resource unavailable for allocation; this could be accomplished with a Schedule changing the capacity to 0. The state will be changed to Failed if Arena has placed the resource in the Failed state, which also implies it's unavailable for allocation.

When a failure occurs, Arena causes the entire resource to become unavailable. If the capacity is 2, for example, both units of the resource will be placed in the Failed state during the repair time.

4.2.2 Resource Schedules

Before we add our resource schedule for the rework operation, let's first define our new 16-hour day. We do this in the Replication Parameters tab of the *Run/Setup* menu, by changing the Hours Per Day from 24 to 16. While we're in this dialog, let's also change our Time Units for the Replication Length to Days and the Replication Length itself to 10 days.

You can start the definition of a resource schedule in either the Resource or Schedule data module. We'll start with the Resource data module. When you click on this module from the Basic Process panel, the information on the current resources in the model will be displayed in the spreadsheet view of the model window along the bottom of your screen. Now click in the Type column for the Rework resource row and select `Based on Schedule` from the pull-down list. When you select this option, Arena will add two new columns to the spreadsheet view—Schedule Name and Schedule Rule. Note that the

Capacity cell for the Rework resource has been grayed out because the capacity will instead be based on a schedule. In addition, the cells for two new columns are also grayed out for the other three resources because they have a fixed capacity. Next you should enter the schedule name (e.g., `Rework Schedule`) in the Schedule Name cell.

Finally, you need to select the Schedule Rule, which determines the specific timing of when the capacity defined by the schedule will actually change. There are three options for the Schedule Rule: Wait (the default), Ignore, and Preempt. If a capacity decrease is scheduled to occur and the resource is idle, all three options immediately cause the resource unit(s) to become Inactive. But if the resource is currently allocated to an entity, each responds differently: (The three options are illustrated in Figure 4-6.)

- The Ignore option immediately decreases the resource capacity, ignoring the fact that the resource is currently allocated to an entity. When the resource is released by the entity, it is placed in the Inactive state. However, if the resource capacity is increased again (i.e., the scheduled time at the lower capacity expires) before the entity releases the resource, it's as if the schedule change never occurred. The net effect is that the time for which the capacity is scheduled to be reduced may be shortened with this option.
- The Wait option, as its name implies, will wait until the entity releases the resource before starting the actual capacity decrease. Thus the reduced capacity time will always be modeled correctly, but the time between these reductions may increase.
- The Preempt option actually preempts the resource by taking it away from the controlling entity and starts the capacity reduction. The preempted entity is held internally by Arena until the resource becomes available, at which time the entity will be reallocated the resource and continue with its remaining processing time. This provides an accurate way to model schedules and failures because, in many cases, the processing of a part is terminated at the end of a shift and when the resource fails. However, there are several special rules that govern the way entities can be preempted.

This brings us to the question of when to use each of the rules. While there are no strict guidelines, a few rules of thumb may be of help. First, if the duration of the scheduled decrease in capacity is very large compared to the processing time, the Ignore option may be an adequate representation. If the time between capacity decreases is large compared to the duration of the decrease, the Wait option could be considered. Generally, we recommend that you examine closely the actual process and select the option that best describes what actually occurs at the time of a downward schedule change or resource failure. If the resource under consideration is the bottleneck for the system, your choice could significantly affect the results obtained. For this model, we've selected the `Ignore` option because, in most cases, an operator will finish his task before leaving and that additional work time is seldom considered.

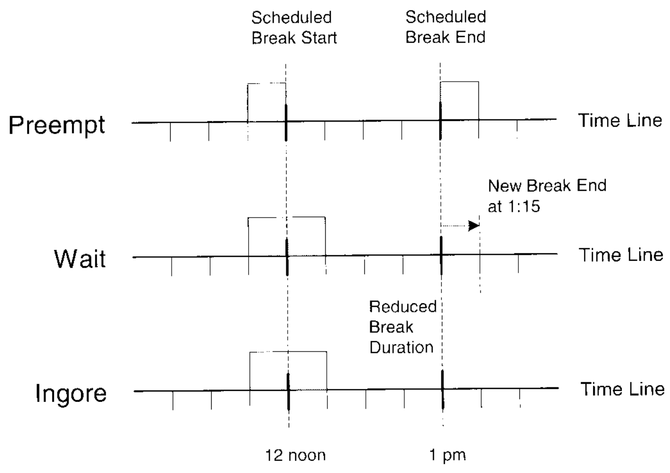
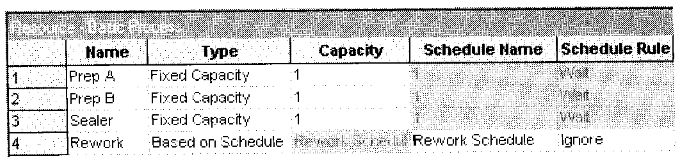


Figure 4-6. The Ignore, Preempt, and Wait Options

The final spreadsheet view of the first four columns is shown in Display 4-15 (there are actually additional columns to the right, which we don't show here). There is also a dialog form for entering these data, which can be opened by right-clicking on the rework cell in the Name column and selecting the *Edit via Dialog* option.

Resource


Resource - Basic Process

	Name	Type	Capacity	Schedule Name	Schedule Rule
1	Prep A	Fixed Capacity	1	1	Wait
2	Prep B	Fixed Capacity	1	1	Wait
3	Sealer	Fixed Capacity	1	1	Wait
4	Rework	Based on Schedule	Rework Schedul	Rework Schedule	Ignore

Rework Resource	
Type	`Based On Schedule`
Schedule Name	`Rework Schedule`
Schedule Rule	`Ignore`

Display 4-15. The Resource Data Module: Selecting a Resource Schedule

Now that you've named the schedule and indicated the schedule rule, you must define the actual schedule the resource should follow. One way to do this is by clicking on the Schedule data module and entering the schedule information in the spreadsheet view. A row has already been added that contains our newly defined Rework Schedule. Clicking in the Durations column will open the Graphical Schedule Editor, a graphical interface for entering the schedule data. The horizontal axis is the calendar or simulation time. (Note that a day is defined as 16 hours based on our day definition in the Run Setup dialog.) The vertical axis is the capacity of the resource. You enter data by clicking on the *x-y* location that represents day one, hour one, and a capacity value of one. This will cause a solid blue bar to appear that represents the desired capacity during this hour; in this case, one. You can complete the data entry by repeatedly clicking or by clicking, holding, and then dragging the bar over the first eight hours. Complete the data schedule by entering a capacity of two for hours nine through 16. It's not necessary to add the data for Day 2, as the data entered for Day 1 will be repeated automatically for the rest of the simulation run. The complete schedule is shown in Figure 4-7. You might note that we used the Options button to reduce our maximum vertical axis (capacity) value from ten to four; other things can be changed in the Options dialog, like how long a time slot lasts, the number of time slots, and whether the schedule repeats from the beginning or remains forevermore at a fixed-capacity level.

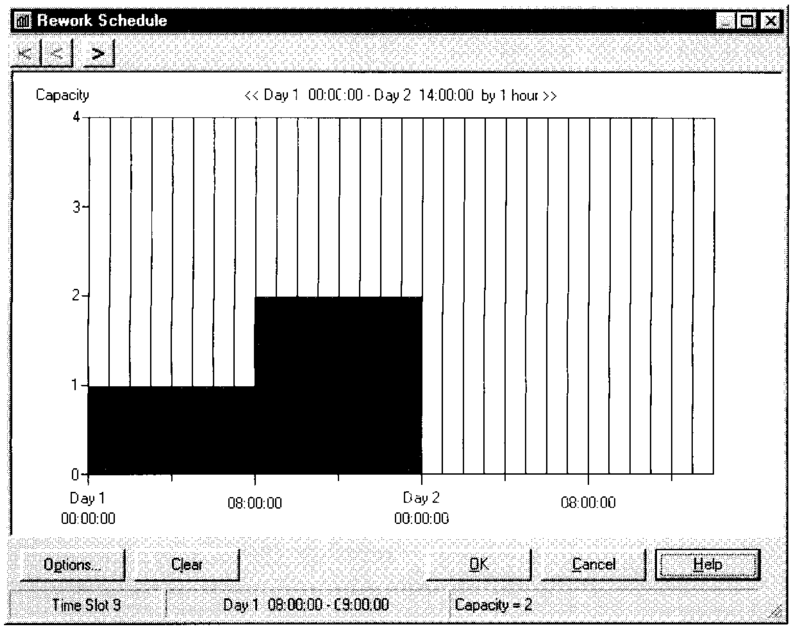
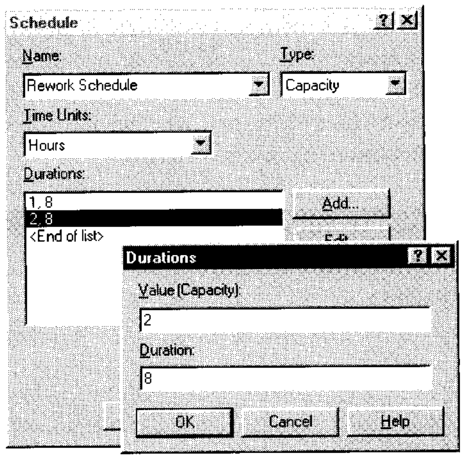


Figure 4-7. The Graphical Scheduling Editor: The Rework Schedule

You can also enter these data manually by right-clicking in the Durations column in the Schedule module spreadsheet view and selecting the *Edit via Dialog* option. If you select this option, first enter the schedule name, then click on the Add button to open the Durations window. Here you define the (Capacity, Duration) pairs that will make up the schedule. In this case, our two pairs are `1, 8` and `2, 8` (Display 4-16). This implies that the capacity will be set to 1 for the first 480 minutes, then 2 for the next 480 minutes. This schedule will then repeat for the duration of the simulation run. You may have as many (Capacity, Duration) pairs as are required to model your system accurately. For example, you might want to include operator breaks and the lunch period in your schedule. There is one caution, or feature,[2] of which you should be aware. If, for any pair, no entry is made for the Duration, it will default to infinity. This will cause the resource to have that capacity for the entire remaining duration of the simulation run. As long as there are positive entries for all durations, the schedule will repeat for the entire simulation run.

Schedule



Name	Rework Schedule
Value	1
Duration	8
Value	2
Duration	8

Display 4-16. The Schedule Data Module Dialog

If you use the Graphical Schedule Editor to create the schedule and then open the dialog, you will find that the data have been entered automatically. Note that you cannot use the Graphical Schedule Editor if you have any time durations that are not integer, or if any entries require an Expression (e.g., a time duration that's a draw from a random variable).

4.2.3 Resource Failures

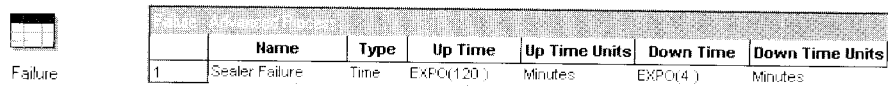
Schedules are intended to model the planned variation in the availability of resources due to shift changes, breaks, meetings, etc. Failures are primarily intended to model random events that cause the resource to become unavailable. You can start your failure definition in either the Resource or the Failure data module. The Failure data module can be found in the Advanced Process panel. Since we started with the Resource module in developing our schedule, let's start with the Failure data module for our Sealer failure.

[2] This is called a feature if you do it intentionally and an error if you do it accidentally.

Start by clicking on the Advanced Process in the Project Bar and then clicking on the Failure data module. The spreadsheet view for this module will show no current entries. Double-click in the open area below the column headers to add a new row. Next click in the Name column and enter a failure name, like `Sealer Failure`. After entering the Failure name, select whether the failure is Count-based or Time-based using the pull-down list in the Type cell. A Count-based failure causes the resource to fail after the specified number of entities have used the resource. This count may be a fixed number or be generated from any expression. Count-based activities are fairly common in industrial models. For example, tooling replacement, cleaning, and machine adjustment are typically based on the number of parts that have been processed rather than on elapsed time. Although these may not normally be viewed as "failures," they do occur on a periodic basis and prevent the resource from producing parts. On the other hand, we frequently model failures as Time-based because that's the way we've collected the failure data. In our model, the problem calls for a Time-based failure. So click on the Type cell and select the `Time` option. When you do this, the form of the spreadsheet will change. One cell is deleted and three new cells are added to reflect the different data requirements between the two options. Our Up Time and Down Time entries are exponential distributions with means of 120 and 4, respectively. We also need to change the Up Time Units and Down Time Units from `Hours` to `Minutes`.

The last field, Uptime in this State Only, allows us to define the state of the resource that should be considered as "counting" for the uptimes. If this field is defaulted, then all states are considered. Use of this feature is very dependent on how your data were collected and the calendar timing of your model. Most failure data are simply logged data; e.g., only the time of the failure is logged. If this is the case, then holidays, lunch breaks, and idle time are included in the time between failures, and you should default this field. Only if your time between failures can be linked directly to a specific state should this option be chosen. Many times equipment vendors will supply you with failure data based on actual operating hours; in this case, you would want to select this option and specify the Busy state. Note that if you select this option you must also define the Busy state using the StateSet data module found in the Advanced Process panel.

The final spreadsheet view for our Sealer Failure is shown in Display 4-17.


Failure

Failure - Advanced Process

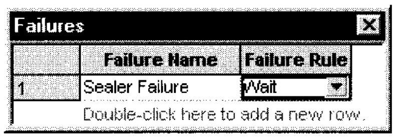
	Name	Type	Up Time	Up Time Units	Down Time	Down Time Units
1	Sealer Failure	Time	EXPO(120)	Minutes	EXPO(4)	Minutes

Name	`Sealer Failure`
Type	`Time`
Up Time	`EXPO(120)`
Up Time Units	`Minutes`
Down Time	`EXPO(4)`
Down Time Units	`Minutes`

Display 4-17. The Sealer Failure Spreadsheet View

Having completed the definition of our Sealer Failure, we now need to attach it to the Sealer resource. Open the Resource data module (back in the Basic Process panel) and click in the Failures column for the Sealer resource row. This will open another window with the Failures spreadsheet view. Double-click to add a new row and, in the Failure Name cell, select `Sealer Failure` from the pull-down list. We must also select the Failure Rule—Ignore, Wait, or Preempt. These options are the same as for schedules, and you respond in an identical manner. Returning to our rules of thumb for choosing the Fail When option, because our expected uptime (120 minutes) is large compared to our failure duration (4 minutes), we'll use the Wait option. The final spreadsheet view is shown in Display 4-18. If you have multiple resources with the same failure profile, they can all reference the same Failure Name. Although they will all use the same failure profile, they will each get their own independent random samples during the simulation run.

Resource


Failures

	Failure Name	Failure Rule
1	Sealer Failure	Wait

Double-click here to add a new row.

Sealer Resource Failure	
Failure Name	`Sealer Failure`
Failure Rule	`Wait`

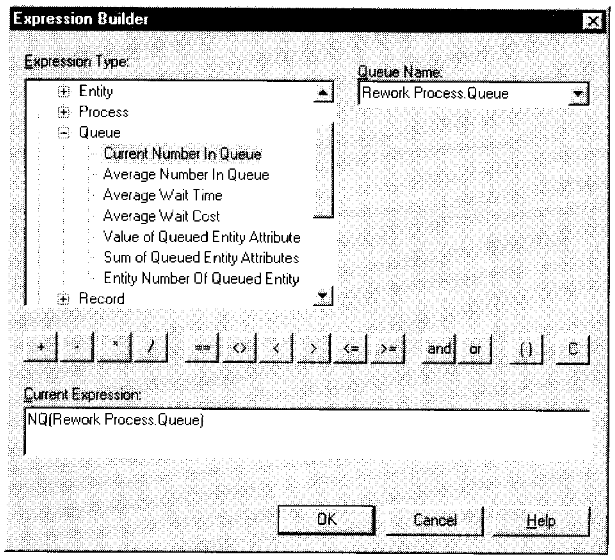
Display 4-18. The Sealer Resource Data Module: Failures Spreadsheet View

4.2.4 Frequencies

Frequencies are used to record the time-persistent occurrence frequency of an Arena variable, expression, or resource state. We can use the *Frequencies* statistic type to obtain the information we need to determine the number of racks required at the Rework area. We're interested in the status of the rework queue—specifically, how many racks of 10 should we buy to assure that we have sufficient storage almost all of the time. In this case, we're interested in the amount of time the number in queue was 0, greater than 0 but no more than 10, greater than 10 but no more than 20, etc.

Frequency statistics are entered using the Statistic data module, which can be found in the Advanced Process panel. Clicking on this data module will open the spreadsheet view, which is currently empty. Now double-click to add a new row. We'll first enter the name as `Rework Queue Stats`. Next we need to select `Frequency` from the Type pull-down list and default on the `Value` entry for the Frequency Type. You might note that when we selected the Type, the Report Label cell was automatically given the same name as the Name cell, `Rework Queue Stats`.

We now need to develop an Expression that represents the number in the rework queue. To request this information, we need to know the Arena variable name for the number in queue, NQ. We can get the name of the queue, `Rework Process.Queue`, from the Queue data module found in the Basic Process panel. Thus, the Expression we want to enter is `NQ(Rework Process.Queue)`. At this point, you should be asking, "Did you expect me to know all that?" To an experienced (and thus old) SIMAN user, this is obvious. However, it is clearly *not* obvious to the new user. Fortunately, Arena provides an easy way to develop these types of expressions without the need to know all the secret words (e.g., NQ). Place your cursor in the blank Expression cell, right-click, and select the Build Expression option. This will open the Arena Expression Builder window shown in Display 4-19. Under the Expression Type category Basic Process Variables, you will find the sub-category Queue. Click on the + sign to expand the options and then select Current Number In Queue. When you do this, two things will happen: the Queue Name option will appear at the right and the Current Expression at the bottom will be filled in using the queue name shown. In our case, the Current Expression was `NQ(Prep A Process.Queue)`, which is not yet what we want. Now use the pull-down option for the Queue Name to select the `Rework Process.Queue`. Now when you press the OK button, that expression will automatically be entered into the field from which the Expression Builder was opened.



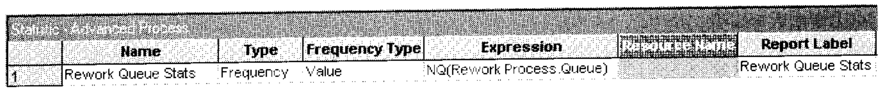
Queue Name: Rework Process.Queue

Current Expression: NQ(Rework Process.Queue)

Display 4-19. The Expression Builder Dialog

You can right-click on any field in which an expression can be entered to open the Expression Builder. For example, we could have used the Expression Builder to find the expression for the current simulation time (TNOW). You can also build complex expressions by using the function buttons in the Expression Builder.

The spreadsheet view to this point is shown in Display 4-20.



Statistic - Advanced Process

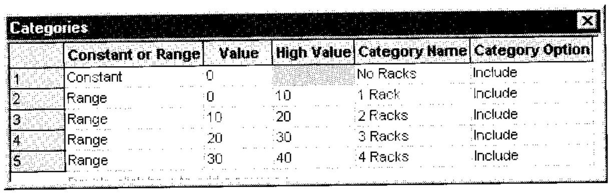
	Name	Type	Frequency Type	Expression	Resource Name	Report Label
1	Rework Queue Stats	Frequency	Value	NQ(Rework Process.Queue)		Rework Queue Stats

Statistic

Name	Rework Queue Stats
Type	Frequency
Frequency Type	Value
Expression	NQ(Rework Process.Queue)

Display 4-20. The Statistic Data Module

The last step in setting up the rework queue statistics is to build the categories that define how we want the values displayed, done in the Categories column at the far right. Display 4-21 shows the entries for the first three categories. The first entry is for a queue size of a constant 0; the next entry is for one rack, two racks, etc. For now, we will only request this information for up to four racks. If the queue ever exceeds 40 parts, Arena will create an out-of-range category on the output report. In the case of a Range, note that Value is not included in the range, but High Value is. Thus, for instance, Value = 10 and High Value = 20 defines a range of numbers (10, 20]; i.e., (strictly) greater than 10 and less than or equal to 20.



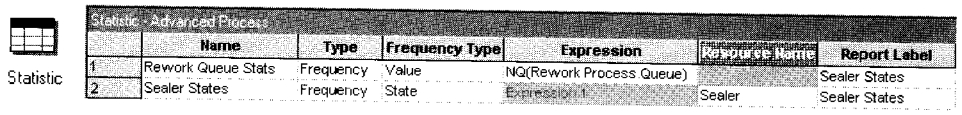
Categories

	Constant or Range	Value	High Value	Category Name	Category Option
1	Constant	0		No Racks	Include
2	Range	0	10	1 Rack	Include
3	Range	10	20	2 Racks	Include
4	Range	20	30	3 Racks	Include
5	Range	30	40	4 Racks	Include

Constant or Range	Constant
Value	0
Category Name	No Racks
Constant or Range	Range
Value	0
High Value	10
Category Name	1 Rack
Constant or Range	Range
Value	10
High Value	20
Category Name	2 Racks

Display 4-21. Categories for Rework Queue Stats Frequency Statistic

Before leaving the Statistic data module, we might also want to request additional information on the Sealer resource. If we run our current model, information on the utilization of the Sealer resource will be included in our reports as before. However, it will not specifically report the amount of time the resource is in a failed state. We can request this statistic by adding a new row in our Statistic data module as shown in Display 4-22. For this statistic, we enter the Name and Type, `Sealer States` and `Frequency`, and select `State` for the Frequency Type. Finally, we select the `Sealer` resource from the pull-down list in the Resource Name cell. This will give us statistics based on all the states of the Sealer resource—Busy, Idle, and Failed.



	Name	Type	Frequency Type	Expression	Resource Name	Report Label
1	Rework Queue Stats	Frequency	Value	NQ(Rework Process.Queue)		Sealer States
2	Sealer States	Frequency	State	Expression 1	Sealer	Sealer States

Name	`Sealer States`
Type	`Frequency`
Frequency Type	`State`
Resource Name	`Sealer`

Display 4-22. The Statistic Data Module for the Sealer States

Before you run this model, we recommend that you check the *Run/Run Control/Batch Run (No Animation)* option, which will greatly reduce the amount of time required to run the model. Although slower, an alternative is to select the *Run/Fast-Forward* (⏩) option. This would also be a good time to save your work. Note that you can still pause the run at any time to determine how far you've progressed.

4.2.5 Results of Model 4-2

Table 4-1 gives some selected results from the Reports for this model (rightmost column), as well as for Model 4-1 for comparison. We rounded everything to two decimals except for the two kinds of utilization results, which we give to four decimals (in order to make a particular point about them).

The results from this model differ from those produced by Model 4-1 for several reasons. We're now running the simulation for ten 16-hour days (so 160 hours) rather than the 32 hours we ran Model 4-1. And, of course, we have different modeling assumptions about the Sealer and Rework Resources. Finally, all of these facts combine to cause the underlying random-number stream to be used differently (more about this in Chapter 11).

Going from Model 4-1 to Model 4-2 didn't involve any changes in the Prep A or Prep B parts of the model, so the differences we see there are due just to the differences in run length or random bounce. The difference for the Prep B queue results are pretty noticeable, so either this area becomes more congested as time goes by, or else the results are subject to a lot of uncertainty—we don't know which (all the more reason to do statistical analysis of the output, which we're not doing here).

Table 4-1. Selected Results from Model 4-1 and Model 4-2

Result	Model 4-1	Model 4-2
Average Waiting Time in Queue		
Prep A	14.62	19.20
Prep B	29.90	51.42
Sealer	2.52	7.83
Rework	456.35	116.25
Average Number Waiting in Queue		
Prep A	3.17	3.89
Prep B	3.50	6.89
Sealer	0.86	2.63
Rework	12.95	3.63
Average Time in System		
Shipped Parts	28.76	47.36
Salvaged Parts	503.85	203.83
Scrapped Parts	737.19	211.96
Utilization of Resource		
Prep A	0.9038	0.8869
Prep B	0.7575	0.8011
Sealer	0.8595	0.8425
Rework	0.9495	0.8641
Scheduled Utilization of Resource		
Prep A	0.9038	0.8869
Prep B	0.7575	0.8011
Sealer	0.8595	0.8425
Rework	0.9495	0.8567

For the Sealer, the queue statistics (both average waiting time and average length) display considerably more congestion for Model 4-2. This makes sense, since we added the Failures to the Sealer in this model, taking it out of action now and then, during which time the queue builds up. The utilization statistics for the Sealer are not much different across the two models, though, since when it's in the Failed state the Sealer is not available, so these periods don't count "against" the Sealer's utilizations.

Unlike the Sealer, the Rework operation seems to be going much more smoothly in Model 4-2. Of course, the reason for this is that we added a second unit of the Rework resource during the second eight-hour shift of each 16-hour day. This increases the capacity of the Rework operation by 50% over time, so that it now has a time-average capacity of 1.5 rather than 1. And accordingly, the utilization statistics of the Rework operation seem to have decreased substantially (more on these different kinds of utilizations below).

Looking at the average time in system of the three kinds of exiting parts, it seems clear that the changes at the Sealer and Rework operations are having their effect. All parts have to endure the now-slower Sealer operation, accounting for the increase in the average time in system of shipped parts. Salvaged and scrapped parts, however, enjoy a much faster trip through the Rework operation (maybe making them feel better after failing inspection), with the net effect being that their average time in system seems to decrease quite a lot.

Now we need to discuss a rather fine point about utilizations. For each Resource, Arena reports two utilization statistics, called *Scheduled Utilization* and simply *Utilization*. To understand what these are, and how they might differ, we need a little bit of notation. Let $B(t)$ be the number of units of a particular Resource that are busy at time t, and let $M(t)$ be the number of units of that Resource that are available (busy or not) at time t. If the Resource has a fixed Capacity, then $M(t)$ is a fixed constant for all t, but if the Resource capacity follows a variable Schedule, then $M(t)$ will vary with t. Of course, $0 \le B(t) \le M(t)$ at all times t. If the Resource is not available at time t (e.g., if it's failed), then $M(t) = 0$, which then forces $B(t) = 0$. Let $U(t) = B(t)/M(t)$ whenever $M(t) > 0$; thus $0 \le U(t) \le 1$, and $U(t)$ represents what might be called *instantaneous utilization* of the Resource. What Arena calls simply the *Utilization* of the resource is just the (time) average of this instantaneous utilization over the simulation:

$$\text{Utilization} = \frac{\int_0^T U(t)\,dt}{T} = \frac{1}{T}\int_0^T \frac{B(t)}{M(t)}\,dt,$$

where T is the length of the simulation. In words, Utilization is the (time) average of the ratio of the number busy to the number available. If the number of units available of the Resource indeed changes according to a Schedule that might represent a staffing plan, then this Utilization measures how well the staffing plan tracks demand over time. Sometimes, though, it might be helpful to have an overall measure of total capacity vs. total demand and worry about the precise timing of the staffing later, and to this end Arena also reports the *Scheduled Utilization*, which is the ratio of the average number busy to the average number available:

$$\text{Scheduled Utilization} = \frac{\int_0^T B(t)\,dt \Big/ T}{\int_0^T M(t)\,dt \Big/ T} = \frac{\int_0^T B(t)\,dt}{\int_0^T M(t)\,dt}.$$

If the Resource has a fixed Capacity, then it's easy to show that Utilization and Scheduled Utilization will be the same (see Exercise 4-19); check that this is so in Table 4-1 for all four Resources in Model 4-1 and for all but the Rework Resource in Model 4-2. (In Model 4-2, the Sealer's availability changes due to the Failures, but these periods are eliminated from both utilization calculations, so both result in the same number.) However, in Model 4-2 the Rework Resource's capacity is on a variable Schedule, which is why you see a difference between its Utilization (0.8641) and Scheduled Utilization (0.8567). Arena actually reports the average number busy (1.2851 for Rework) and the average number scheduled (1.5 for Rework), and Scheduled Utilization is the ratio of these two averages, 1.2851/1.5 = 0.8567, the same[3] as in the reports. Which utilization measure to use depends on whether you want to know how your overall capacity *can* accommodate demand (Scheduled Utilization), or how well your Schedule tracks time-dependent demand (Utilization). If Utilization is a lot higher than Scheduled Utilization, it could be that you have enough overall capacity but you're not scheduling it quite right. It's not the case that Utilization is always higher than Scheduled Utilization, or the reverse, as addressed in Exercise 4-20.

The new frequency statistics are not part of the normal report. You must click on the Frequencies selection found in the Reports panel of the Project Bar. The results are given in Figure 4-8.

Rework Queue Stats	Number Obs	Average Time	Standard Percent	Restricted Percent
1 Rack	52	119.96	64.98	64.98
2 Racks	12	42.8210	5.35	5.35
No Racks	41	69.4722	29.67	29.67

Sealer States	Number Obs	Average Time	Standard Percent	Restricted Percent
BUSY	697	11.6044	84.25	84.25
FAILED	68	4.1861	2.97	2.97
IDLE	640	1.9173	12.78	12.78

Figure 4-8. The Arena Frequencies Report: Model 4-2

The first section shows the statistics we requested to determine the number of racks required at the rework process. There were never more than 20 in the rework queue (i.e., there are no data listed for four racks), and there were more than 10 only 5.35% of the time. This might imply that you could get by with only one rack, or at most, two. The Sealer statistics are still included in the main overview report section, but a more complete form of the utilization can be found in the Frequencies report. In this case, it provides the results in percentages for the states Busy, Idle, and Failed.

One last note is worth mentioning about frequency statistics. For our results, the last two columns, Standard and Restricted Percent, have the same values. It is possible to exclude selective data from the last column. For example, if you exclude the Failed data for the sealer Frequency, the Standard Percent would remain the same, but the Restricted Percent column would only have values for Busy and Idle, which would sum to 100.

[3] To make this situation even more complicated, there can be a further discrepancy between (average number busy)/(average number available) and the Scheduled Utilization in the reports. This is due to the way that Arena accounts for whether a Resource is allocated or not for the period following a capacity decrease when all units of the resource were busy. If the Schedule Rule is Wait this discrepancy will not be present, but if it is Ignore or Preempt the discrepancy can occur. We used Ignore in Model 4-2, so a discrepancy was possible; our numbers happened to be such that it didn't show up in four decimal places of accuracy, though.

4.3 Model 4-3: Enhancing the Animation

So far in this chapter we've simply accepted the default animation provided with the modules we used. Although this base animation is almost always sufficient for determining whether your model is working correctly, you might want the animation to look more like the real system before allowing decision makers to view the model. Making the animation more realistic is really very easy and seldom requires a lot of time. To a large extent, the amount of time you decide to invest depends on the level of detail you decide to add and the nature of the audience for your efforts. A general observation is that, for presentation purposes, the higher you go in an organization, the more time you should probably spend on the animation. You will also find that making the animation beautiful can be a lot of fun and can even become an obsession. So with that thought in mind, let's explore some of what can be done.

We'll start by looking at the existing animation. The current animation has three components: entities, queues, and variables. The entities we selected using the Entity data module can be seen when they travel from one module to another or when they're waiting in a queue. For each Process module we placed, Arena automatically added an animation queue, which displays waiting entities during the run. Variables were also placed automatically by Arena to represent the number of entities waiting or the number of entities that have exited a module.

As suggested by how they came to exist in the model—namely, they were added when you placed the module—the animation constructs are "attached" to the module in two respects. First, their names, or *Identifiers*, come from values in the module dialog; you can't change them directly in the animation construct's dialog. Second, if you move the module, its animation objects move with it. If you want the animation to stay where it is, though, just hold the Shift key when you move the module handle.

Sometimes it's helpful to "pull apart" the animation to a completely different area in the model window, away from the logic. If you do so, you might consider setting up some Named Views (see Section 3.4.13) to facilitate going back and forth. If you want to disconnect an animation construct completely from the module it originally accompanied, *Cut* it to the *Clipboard* and *Paste* it back into the model. It will retain all of its characteristics, but no longer will have any association with the module. An alternative method is to delete the animation construct that came with the module and add it back from scratch using the constructs from the Animate toolbar.

You might even want to leave some of the automatically placed animation constructs with the modules and just make copies of them for the separate animation. If you decide to do this, there are some basic rules to follow. Any animation construct that provides information (e.g., variables and plots) can be duplicated. Animation constructs that show an activity of an entity (e.g., queues and resources) should not be duplicated in an animation. The reason is quite simple. If you have two animated queues with the same name, Arena would not know which animated queue should show a waiting entity. Although Arena will allow you to duplicate an animated queue, it will generally show all waiting entities in the last animated queue that you placed.

We'll use the "pull apart" method to create our enhanced animation. Let's start by using the zoom-out feature, *View/Zoom Out* (or the – key), to reduce the size of our model. Now click on a queue and use the *Edit/Cut* (or Ctrl-X) option to cut or remove it from the current model and then use the *Edit/Paste* (or Ctrl-V) option to place the queue in the general area where you want to build your animation (click to place the floating rectangle). Repeat this action for the remaining queues, placing them in the same general pattern as in the original model. You can now zoom in on the new area and start building the enhanced animation. We'll start by changing our queues. Then we'll create new pictures for our entities and add resource pictures. Finally, we'll add some plots and variables.

4.3.1 Changing Animation Queues

If you watched the animation closely, you might have noticed that there were never more than about 14 entities visible in any of the queues, even though our variables indicated otherwise. This is because Arena restricts the number of animated entities displayed in any queue to the number that will fit in the drawn space for the animation queue. The simulation may have 30 entities in the queue, but if only 14 will fit on the animation, only the first 14 will show. Then as any entity is removed from the queue, the next undisplayed entity in the queue will be shown. While the output statistics reported at the end will be correct, this can be rather deceptive to the novice and may lead you to assume that the system is working fine when, in fact, the queues are quite large. There are three obvious ways (at least to us) to avoid this problem: one is to watch the animation variable for the number in queue, the second is to increase the size of the animation queue, and the third is to decrease the size of the entity picture (if it remains visually accurate).

Let's first increase the size of the queue. Figure 4-9 shows the steps we'll go through as we modify our queue. We first select the queue (View 1) by single-clicking on the Rework queue, `Rework Process.Queue`. Notice that two handles appear, one at each end. You can now place your cursor over the handle at the left end, and it will change to cross hairs. Drag the handle to stretch the queue to any length and direction you want (View 2). If you now run the simulation, you should occasionally see a lot more parts waiting at Rework.

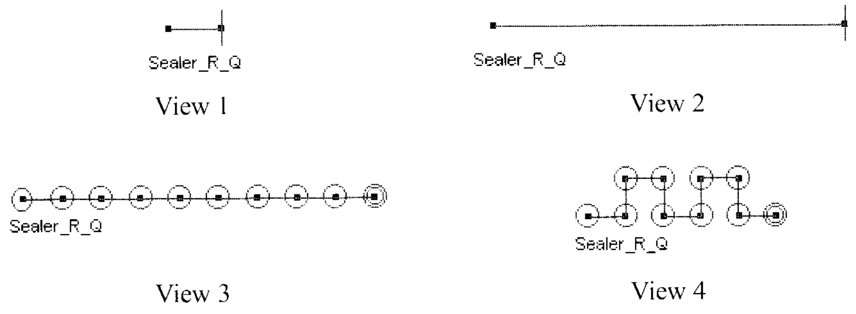
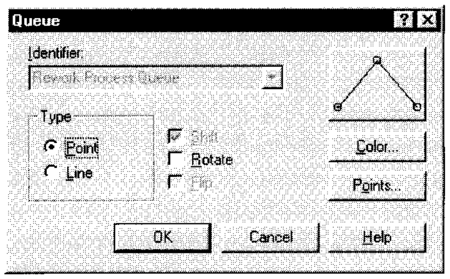


Figure 4-9. Alternate Ways to Display a Queue

We can also change the form of the queue to represent the physical location *point* of each entity in it. Double-click on the selected queue and the Queue dialog will appear as in Display 4-23.



Identifier: Rework Process Queue

Type: [✓ Point] Line

[✓ Shift] Rotate Flip

Type	
Point	*select*

Display 4-23. The Queue Dialog

Select the Point Type of the queue and click on the Points button. Then add points by successively clicking on the Add button. We could change the rotation of the entity at each point, but for now we'll accept the default of these values. When you accept these changes, the resulting queue should look something like the one shown in View 3 of Figure 4-9. Note that the front of the queue is denoted by a point surrounded by two circles. You can then drag any of these points into any formation you like (View 4). If you want all these points to line up neatly, you may want to use the Snap option discussed in Chapter 3. Arena will now place entities on the points during an animation run and move them forward, much like a real-life waiting line operates.

For our animation, we've simply stretched the queues (as shown in View 2 of Figure 4-9) for the Prep A, Prep B, and Sealer areas. We did get a little fancy with the Rework queue. We changed the form of the queue to points and added 28 points. This allowed us to align them in three rows of ten to represent three available racks as shown in Figure 4-10 (this was much easier to do with the Snap option on).

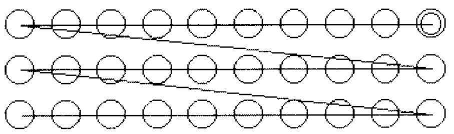


Figure 4-10. The Rework Queue with 30 Points

4.3.2 Changing Entity Pictures

Now let's focus our attention on our animation entities. In our current animation, we arbitrarily select blue and red balls for the two kinds of entities. Let's say we want our entities to be similar to the balls but have the letter "A" or "B" displayed inside the ball. You create new pictures in the Entity Picture Placement window, as shown in Figure 4-11, which is opened using the *Edit/Entity Pictures* menu option. The left side of this window contains entity pictures currently available in your model, displayed as a list of buttons with the picture and its associated name. The right side of this window is used for accessing *picture libraries*, which are simply collections of pictures stored in a file. Arena provides several of these libraries with a starting selection of icons; you might want to open and examine them before you animate your next model (their file names end with *.plb*).

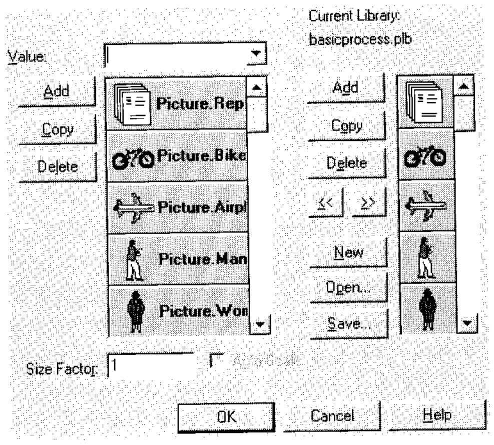


Figure 4-11. The Entity Picture Placement Window

There are several ways to add a new picture to your animation. You can use the Add button on the left to draw a new picture for the current list, or you can use the Copy button (on the left) to copy a picture already on the current list. If you use the Add function, your new entry will not have a picture or a name associated with it until you draw it and give it a name in the Value field above the list. If you use the Copy function, the new picture and name will be the same as the picture selected when you copied.

To add a picture from a library to your current entity picture list, highlight the picture you want to replace on the left, highlight the new selection from a library on the right, and click on the left arrow button (<<) to copy the picture to your picture list. You can also build and maintain your own picture libraries by choosing the New button, creating your own pictures, and saving the file for future use. Or you can use clip art by using the standard copy and paste commands.

For this example, as for most of our examples, we'll keep our pictures fairly simple, but you can make your entity and resource pictures as fancy as you want. Since the blue and red balls were about the right size, let's use them as our starting point. Click on the

`Picture.Blue Ball` icon in the current list on the left and then click Copy. Now select one of these two identical pictures and change the name (in the Value dialog) to `Picture.Part A` (now wasn't that obvious?). Note that as you type in the new name it will also change on the selected icon. To change the picture, double-click on the picture icon. This opens the Picture Editor window that will allow you to modify the picture drawing. Before you change this picture, notice the small gray circle in the center of the square; this is the *entity reference point*, which determines the entity's relation with the other animation objects. Basically, this point will follow the paths when the entity is moving, will reside on the seize point when the entity has control of a resource, etc.

We'll change this picture by inserting the letter "A" in the center of the ball and changing to a lighter fill color so the letter will be visible. When you close this window, the new drawing will be displayed beside the Picture.Part A name. Now repeat the same procedure to make a new picture for Part B. Your final pictures should look something like Figure 4-12.

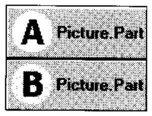


Figure 4-12. The Final Entity Pictures

Before we close this window, you might notice that the "A" and "B" do not appear in the picture name because there is not enough room. However, if you click on one of the pictures, the full name will be displayed in the Value field at the top. Also note that there is a Size Factor field in the lower left-hand portion of the window (see Figure 4-11). You can increase or decrease your entity picture size by changing this value. For our animation, we have increased the Size Factor from 1 to 1.5.

The final step is to assign these new pictures to our parts so they will show up in the animation. You do this by clicking on the Entities data module and entering the new names in the Initial Picture cell for our two parts. You might note that your new names will not appear on the pull-down list, so you will need to type them. However, once you have entered the new names and accepted the data, they will be reflected on the pull-down list.

4.3.3 Adding Resource Pictures

Now that we've completed our animated queues and entities, let's add resource pictures to our animation. You add a resource picture by clicking on the Resource button (✱) found in the Animate toolbar. This will open the Resource Picture Placement window, which looks very similar to the Entity Picture Placement window. There's very little difference between an entity picture and a resource picture other than the way we refer to them. Entities acquire pictures by assigning a *picture name* somewhere in the model. Resources acquire pictures depending on their state. In Section 4.2.1, we discussed the four automatic resource states (Idle, Busy, Failed, and Inactive). When you open a Resource Picture Placement window, you might notice that the picture for each of the four default states is the same. You can, however, change the drawings used to depict the resource in its various states, just like we changed our entity pictures.

First we need to identify which resource picture we are creating. You do this by using the pull-down list in the Identifier field to select one of our resources (e.g., Prep A). Now let's replace these pictures as we did for the entity pictures. Double-click on the Idle picture to open the Picture Editor window. Use the background color for the fill, change the line thickness to 7 points (from the Draw toolbar), and change the line color. Note that the box must be highlighted in order to make these changes. The small circle with the cross is the *reference point* for the resource, indicating how other objects (like entity pictures) align to its picture; drag this into the middle of your box. Accept this icon (by closing the Picture Editor window) and return to the Resource Picture Placement window. Now let's develop our own picture library. Choosing the New button from the Resource Picture Placement window opens a new, empty library file. Now select your newly created icon, click Add under the current library area, and then click on the right arrow button. Choose the Save button to name and save your new library file (e.g., `Book.plb`).

We'll now use this picture to create the rest of our resource pictures. Highlight the Busy picture on the left and the new library picture on the right and use the left arrow button to make your busy picture look just like your idle picture. When the animation is running, the entity picture will sit in the center of this box, so we'll know it's busy. However, you do need to check the Seize Area toggle at the bottom of the window for this to happen. Now copy the same library picture to the inactive and failed states. Open each of these pictures and fill the box with a color that will denote whether the resource is in the failed or inactive state (e.g., red for failed). Now copy these two new pictures to your library and save it. Your final resource pictures should look something like those shown in Figure 4-13. We also increased our Size Factor to 1.5 so the resource size will be consistent with our entity size.

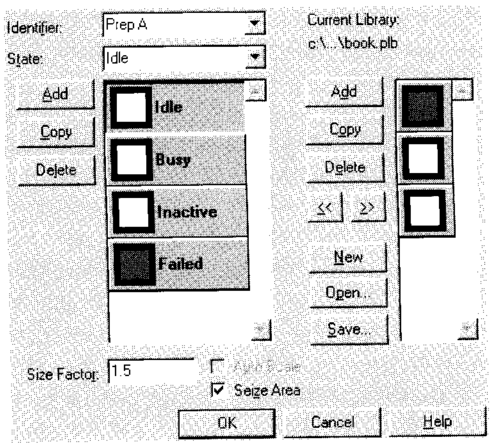

Identifier: Prep A
State: Idle
Current Library: c:\...\book.plb
Size Factor: 1.5
☑ Seize Area

Figure 4-13. The Final Resource Pictures

When you accept the resource pictures and return to the main model window, your cursor will be a cross hair. Position this cursor in the approximate area where you want to place the resource and click. This places the new resource on your animation. (By the way, have you remembered to save your model recently?) Your new resource icon may be larger than you want, so adjust it appropriately by dragging one of its corner handles. The resource picture also contains an object that appears as a double circle with a dashed line connected to the lower left portion of the resource picture—the *Seize Area*. This Seize Area is where your entity will sit when it has control of the resource; drag it to the center of your resource picture. Now run your animation to see if your entity and resource pictures are the sizes you want. If not, you can adjust the position of the seize area by pausing the run, displaying seize areas (using the *View/Layers* menu option), and dragging it to the desired location. After you've fine-tuned its position, you'll probably want to turn off the display of the seize area layer before ending the run.

Once you're satisfied with your animation of the Prep A resource, you'll want to add animations for the rest of the resources. You could repeat the above process for each resource, or you can copy and paste the Prep A resource for the Prep B and Sealer pictures. Once you've done this, you'll need to double-click on the newly pasted resource to open the Resource Picture Placement window and select the proper name from the pull-down list in the Identifier field.

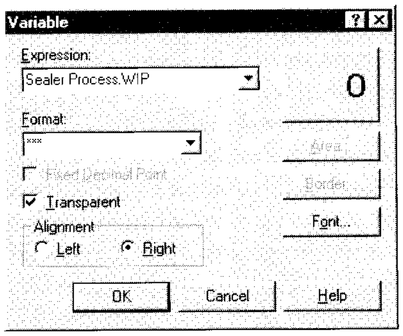
The picture of the rework resource will have to be modified because it has a capacity of 2 during the second shift. Let's start by doing a copy/paste as before, but when we open the resource picture placement window, edit the Idle picture and add another square (Edit/Duplicate or Copy/Paste) beside or under the first picture. This will allow room for two entities to reside during the second shift. Copy this new picture to your library and use it to create the remaining rework pictures. Rename the resource (Rework) and close the window.

The original resource animation had only one seize area, so double-click on the seize area, click on the Points button, and add a second seize area. Seize areas are much like queues and can have any number of points, although the number of points used depends on the resource capacity. Like a queue, seize areas can also be shown as a line. Close this window and position the two seize-area points inside the two boxes representing the resource.

You should now have an animation that's starting to look more like your perceived system. You may want to reposition the resources, queues, stations, etc., until you're happy with the way the animation looks. You also could add text to label things, place lines to indicate queues or walls, or even add a few potted plants.

4.3.4 Adding Plots and Variables

The last thing we'll do is add some additional variables and a plot to our animation. Variables of interest are the number of parts at each process and the number of parts completed. We could use the Variable object from the Animate toolbar (▣) for this, but it's much easier to just copy and paste the variables that came with our model. First copy and paste the four variables that came with our process modules; put them right under the resource picture we just created. Then resize them by highlighting the variable and dragging one of the handles to make the variable image bigger or smaller. You can also reformat, change font, change color, etc., by double-clicking on the variable to open the Variable window shown in Figure 4-14.


Variable

Expression: Sealer Process.WIP

Format: ***

☐ Fixed Decimal Point

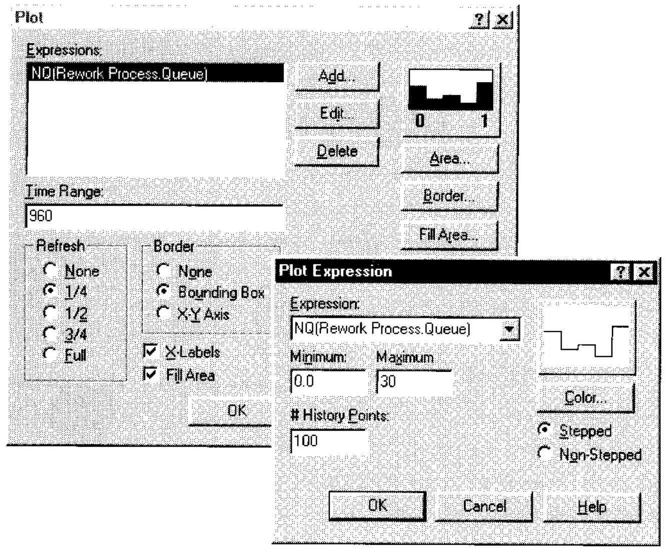
☑ Transparent

Alignment: ○ Left [✓ ● Right]

Figure 4-14. The Variable Window

We then repeated this process for the three variables that came with our Dispose modules. Finally, we used the Text tool from the Animate toolbar to label these variables.

Now let's add a plot for the number in the rework queue. Clicking on the Plot button ( ) from the Animate toolbar will open the Plot window. Use the Add button to enter the expression `NQ(Rework Process.Queue)`. Recall that we used this same expression when we created our Frequencies data. We also made a number of other entries as shown in Display 4-24. The # of History Points tells Arena the maximum number of possible "corner" points it will have on the plot at any time. If this number is too small, the left portion of your plot will not show any data. If this happens, simply increase this number until it's large enough. Finally, we changed the Area and Fill Area colors (punch the buttons) to make our plot look stunningly attractive.



Plot

Expressions: NQ(Rework Process.Queue)

Time Range: 960

Refresh: ( ) None [✓ 1/4] ( ) 1/2 ( ) 3/4 ( ) Full

Border: ( ) None [✓ Bounding Box] ( ) X-Y Axis

[✓ X-Labels]

[✓ Fill Area]

Plot Expression

Expression: NQ(Rework Process.Queue)

Minimum: 0.0 Maximum: 30

History Points: 100

[✓ Stepped] ( ) Non-Stepped

Plot Expression	
Expression	`NQ(Rework Process.Queue)`
Maximum	`30`
Plot	
Time Range	`960`
Refresh – 1/4	*select*
X-Labels	*check*
Fill Area	*check*

Display 4-24. The Plot Window

After you've accepted these data, you may want to increase the size of the plot in the model window and move it somewhere else on the animation. Finally, we used the Text tool from the Animate toolbar to add some information for our plot. You could also add some lines, drawings, or clip art to jazz up your animation.

Your animation should now be complete and look something like the snapshot shown in Figure 4-15, which was taken at about simulation time 7067. The numerical results are, of course, the same as those from Model 4-2 since we changed only the animation aspects to create the present Model 4-3.

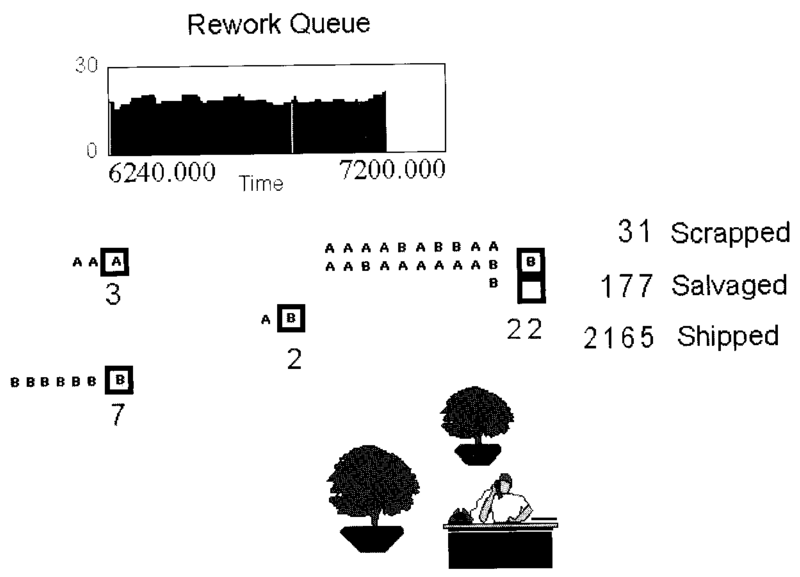


Figure 4-15. The Final Animation: Model 4-3

4.4 Input Analysis: Specifying Model Parameters and Distributions

As you've no doubt noticed, there are a lot of details that you have to specify completely in order to define a working simulation model. Probably you think first of the logical aspects of the model, like what the entities and resources are, how entities enter and maybe leave the model, the resources they need, the paths they follow, and so on. These kinds of activities might be called *structural* modeling since they lay out the fundamental logic of what you want your model to look like and do.

You've also noticed that there are other things in specifying a model that are more numerical or mathematical in nature (and maybe therefore more mundane). For example, in specifying Model 4-2, we declared that interarrival times for Part A were exponentially distributed with a mean of 5 minutes, total processing times for Part B Prep followed a triangular (3, 5, 10) distribution, the "up" times for the sealer were draws from an exponential (120) distribution, and we set up a Schedule for the number of rework operators at different times. You also have to make these kinds of specifications, which might be called *quantitative* modeling, and are potentially just as important to what happens as are the structural-modeling assumptions.

So where did we get all these numbers and distributions for Model 4-2 (as well as for practically all the other models in this book)? OK, we admit that we just made them up, after playing around for a while, to get the kinds of results we wanted in order to illustrate various points. We get to do this since we're just writing a book rather than doing any real work, but unfortunately, you won't have this luxury. Rather, you need to observe the real