

Chapter 2 – Elements of a Simulation Model

Various elements of a simulation model are discussed in this section, all in reference to Sample Model 1 in Figure 2.1 below.

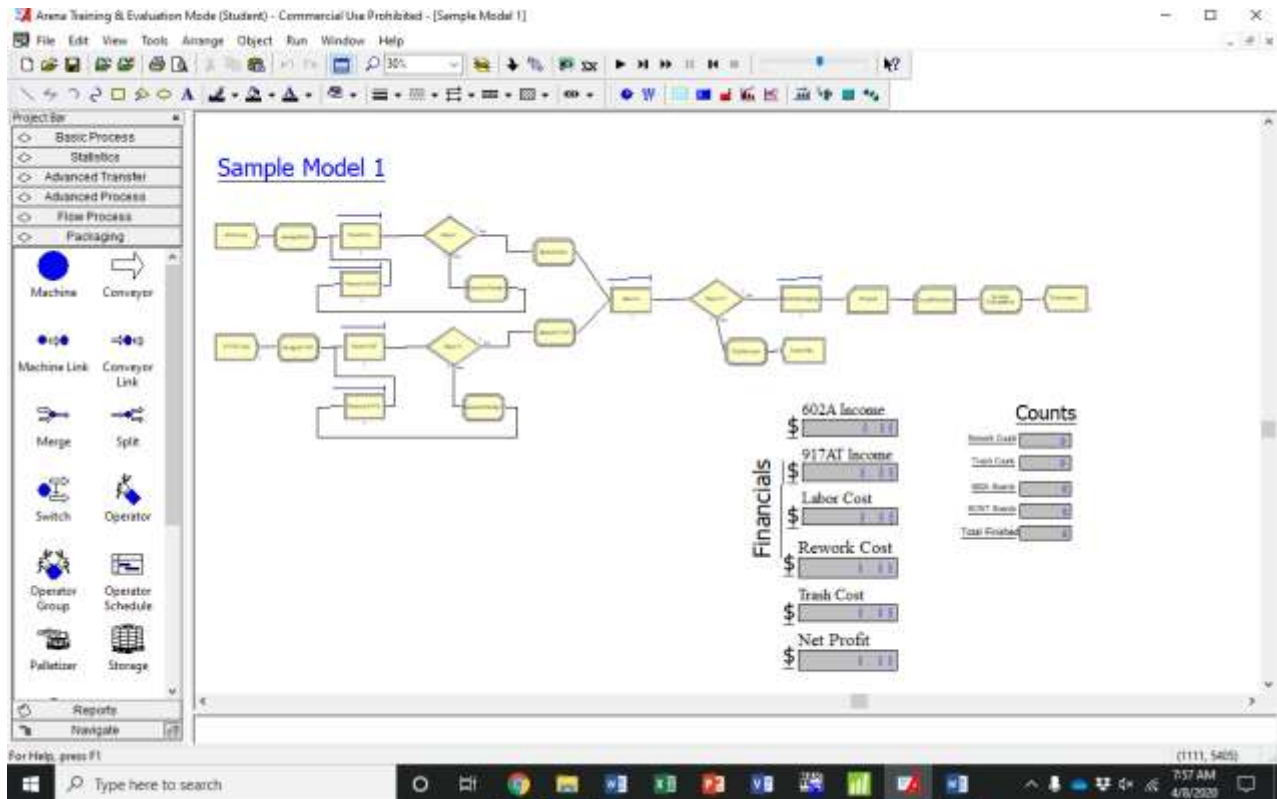


Figure 2.1. Sample Model 1.

2.1 Entities

Entities are the dynamic objects in a simulation. They are created by the analyst, move around, and are then disposed as they leave the system. It is also possible to have entities that never leave the system, they keep circulating within the system. All entities are created by either the analyst or automatically by the Arena software.

Most entities represent real things in a simulation. You can have different kinds of entities in the model, such as different kinds of parts, each possibly requiring different processing times and routes, perhaps having different priorities within the system.

You can add “fake” entities that do not pertain to anything tangible but rather account for certain modeling operations, such as creating a “breakdown” to model

machine failures. Alternatively, you can add “break” entities that arrive periodically to take a server off duty.

Determining what your entities are is one of the first things the analyst should do in construction a simulation model.

2.2 Attributes

An attribute is a common characteristic of all entities, but with a specific value that can differ from one entity to another. Consider the specific value as a label, or tag, that is attached to an entity but the value on the tag can differ across entities to characterize them individually. The value of the label, or tag, does not change for the entity during the model run. As an example, an attribute for an entity could be an arrival time, a rework priority, a due date, color, and so on, all of which are unique to that entity. It is up to the analyst to determine what attributes your entities need, name the attributes, assign values to the attributes, and then use the attributes in your model. An attribute is a *local* variable, meaning that they are local to each individual entity.

2.3 Variables

A variable (also known as a *global* variable) is a variable that reflects a characteristic of the system, regardless of the number of or what kinds of entities may be in your model. With global variables, the value of the variable can change during the simulation run. For example, a variable can represent something that changes during the simulation, such as the number of reworked parts going through a rework process or the number of completed units that exit the system, which is incremented by a part when it enters the area and decrements when it leaves the area.

Whereas the analyst can have many different variables in a model, each variable is unique. Arena has two types of variables: Arena built-in variables (such as number of parts in queue, current simulation clock time, status of a machine (busy or idle), etc.) and user-defined variables (such as mean throughput time, utilization rates, labor cost, net profit, etc.). Unlike attributes, a variable is not attached to any specific entity, but rather pertains to the system at large. A variable is accessible by all entities, and the value of a variable can be changed by any entity.

2.4 Resources

Entities often compete for service from *resources*, such as personnel, machines, equipment, or space in a storage area. In everyday life, a resource, such as a worker, seizes an entity, performs some activity, and then releases the entity to the next operation when finished. However, in the Arena world, it is the entity that seizes a

resource, undergoes some activity, and then the entity releases the resource when finished.

A resource can represent a group of several individual servers, each of which is referred to as a unit of that resource. An example would be several identical “parallel” billing clerks at a health insurance company. The number of available units of the resource can be changed during the simulation to represent different work shifts, break times, etc. If a resource has multiple units, or a variable number of units, the resource utilization will be the time-average number of units of the resource that are busy divided by the time-average number of units of the resource that are available.

2.5 Queues

The purpose of a *queue* is to hold an entity that needs to seize a resource in a temporary waiting area, but the resource is currently tied up with another entity. All queues have names and the names are typically the name of the process followed by a .queue, such as Assembly.queue. Queues can take on any one of four different priority rules: first in, first out; last in, first out, lowest attribute value, and highest attribute value. The analyst would select the appropriate priority rule in the Queue data module.

2.6 Statistical Accumulators

As the simulation progresses, Arena keeps track of various intermediate *statistical accumulators*, such as the number of parts produced so far, the total of the waiting times in queue so far, the total time in the system by all parts that have exited the system so far, and so on. All of the accumulators should be initialized to 0 at the start of the simulation. This is done by going to Run > Setup > Replication Parameters, then checking off the boxes for Statistics and System under the heading Initialize Between Replications (see Figure 2.2). Whether the analyst is running one replication or multiple replications, this sets the value of statistical accumulators back to zero for each replication.

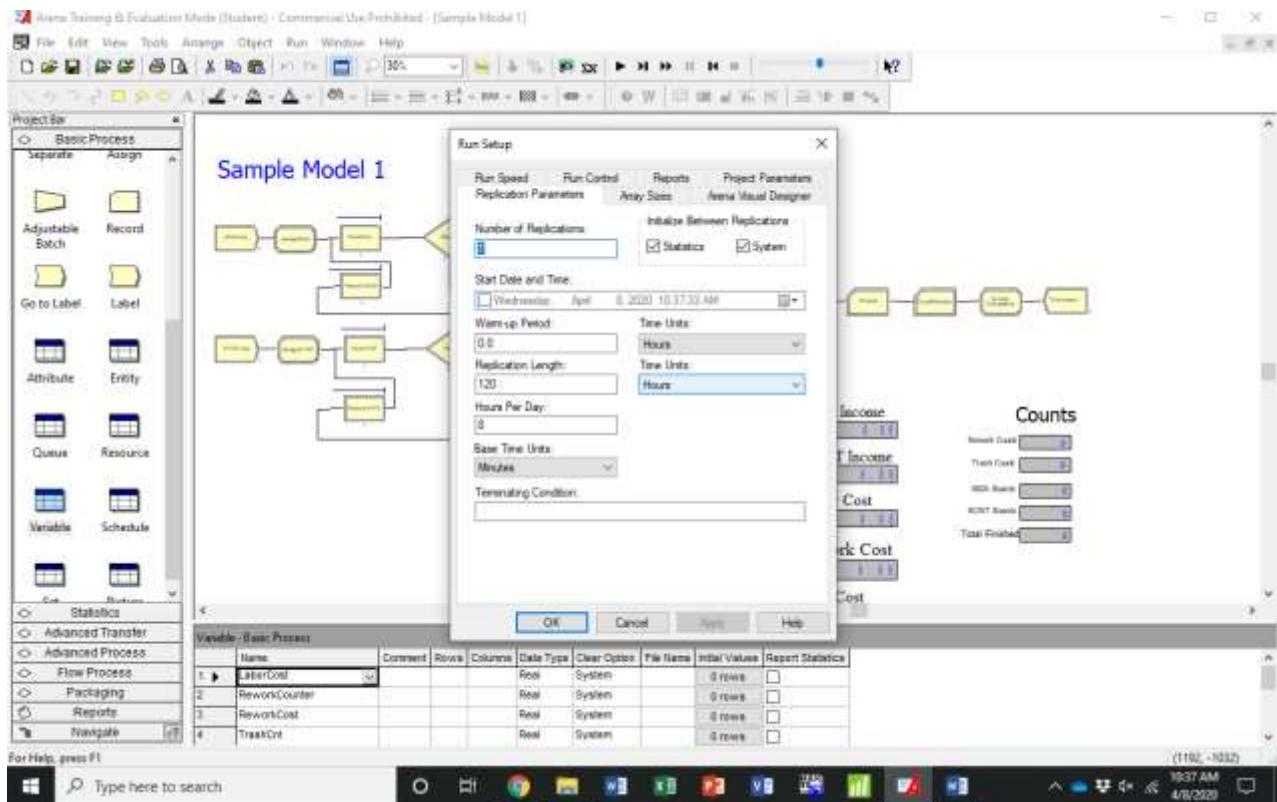


Figure 2.2. Dialog box for Initialize Between Replications in Run > Setup

2.7 Events

An *event* is something that happens at an instant of simulated time that may change attributes, variables, or statistical accumulators. Examples of events include: 1) the Arrival of a new part that enters the system; 2) the Departure of a part that exits the system, and 3) a Terminating Condition, such as the end of the simulation run.

Another example is when a part leaves a queue and enters a machine center for processing. This only occurs if the machine center is currently empty or when a previous part has finished processing on that machine center and exits the machine, thus allowing the next part in queue to replace it in the machine center.

Arena keeps track of events that are supposed to happen in the simulated feature in an *event calendar*. When the model runs, a record of information pertaining to a future event is placed on the event calendar. The event record contains information on the entity involved, the event time, and the kind of event it will be. Arena stores each newly scheduled event on the event calendar so that the next event is at the top of the event calendar. When it is time to execute the next event, the top record is removed from the calendar and the information in this event record is used to execute the appropriate logic. Then the next scheduled event moves to the top of the event calendar. This event action sequence is repeated until the end of the simulation run.

2.8 Simulation Clock

The variable called simulation clock stores the current value of simulated time during the simulation run. The simulation clock in Arena does not track real time when nothing changes between events. Rather, the simulated time lurches forward from the time of one event to the time that the next event is scheduled to happen. How each event is executed depends on the kind of event it is as well as the model state at that time, but could include the updating of variables and statistical accumulators, altering entity attributes, removing the top event on the event calendar, and placing newly scheduled event records onto the event calendar.

2.9 Starting and Stopping Times

The analyst must determine the starting and stopping times for your model. Arena cannot do this for you. For example, the analyst must determine the appropriate starting conditions (such as starting at time 0 or build in a delayed start time), how long a simulation run should last (such as the run length in seconds, minutes, hours, or days), and whether the simulation run should stop at a particular time (such as designating the run length) or whether the simulation run should stop when something specific happens (such as designating a terminating condition such as a specific time at the end of a work day). Assumptions should be consistent with what you are modeling.