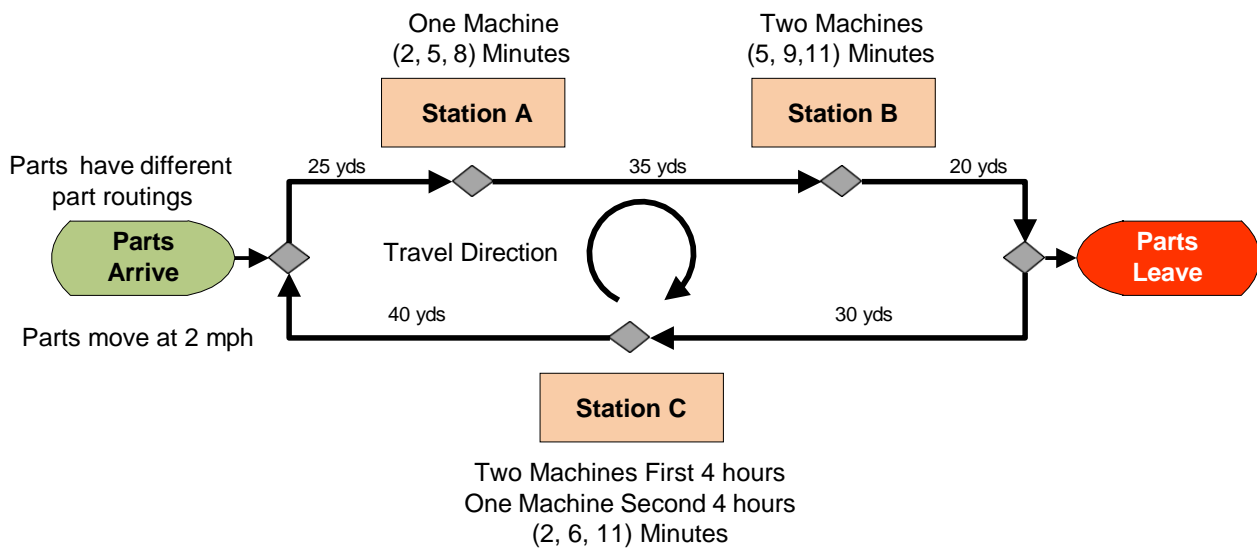


## Chapter 5

# Data-Based Modeling: Manufacturing Cell

Most simulation models employ **substantial** amounts of data in their development and execution. In SIMIO, employing a data table, as seen in the previous chapter, provides a convenient means to store and recall the data. In this chapter we will use tables more extensively by re-constructing a manufacturing cell consisting of several work stations.

A small manufacturing cell consists of three workstations through which four part types are processed. The workstations, A, B, and C, are arranged such that the parts must follow the one-way circular pattern as shown in the layout in Figure 5.1. For instance, **if a part is finished at Station C**, it would have to travel the distance to Station A, then the distance to Station B before it could travel to where it exits the system. We want to use a simulation of a five day week with eight hours of operation a day for a total of 40 hours.



**Figure 5.1: Workstation Layout**

All parts arrive at the “Parts Arrive” location and leave at the “Parts Leave” location. Travel between workstations is a constant two miles per hour. **Note SIMIO’s default speed for entities is *meters per second*.** The distances (yards) between workstations are shown in Table 5.1 and in Figure 5.1.

**Table 5.1: Distance between Workstations**

WorkStation Path	Distances (yds.)
Parts Arrive to Station A	25
Station A to Station B	35
Station B to Parts Leave	20
Parts Leave to Station C	30
Station C to Parts Arrive	40

Each part type arrives as follows.<sup>42</sup>

- Part 1’s arrive randomly with an average of 15 minutes between arrivals.
- Part 2’s have an interarrival time that averages 17 minutes with a standard deviation of 3 minutes.

<sup>42</sup> For the moment, we will let you think about the distributions that these parameters may represent.

- Part 3's can arrive with an interarrival time of anywhere from 14 to 18 minutes apart.
- Part 4's arrive in batches of five exactly 1 hour and 10 minutes apart.

*Question 1:* What distributions will you choose to model these four arrival processes?

---

Each of the part (entity) types is sequenced (routed) across the stations differently. In fact, not all part types are routed through all machines, as seen in the sequence order given in Table 5.2.

**Table 5.2: Part Sequences**

Step	1	2	3
Part 1	Station A	Station C	
Part 2	Station A	Station B	Station C
Part 3	Station A	Station B	
Part 4	Station B	Station C	

**Server Properties:**

- Station A has one machine with a processing time ranging from two to eight minutes and a likely (modal) time of five minutes.
- Station B has two machines, each with a processing time ranging between five and eleven minutes and a likely time of nine minutes.
- Station C has two machines that operate during the first four hours of each day and one machine operating during the last four hours of the day. Each has a processing time ranging between two and eleven minutes and a mode of six minutes.

*Question 2:* What distributions will you choose to model the three station processing times?

---

## Part 5.1: Constructing the Model

We will build the SIMIO model using four **SOURCES**, three **SERVERS** and several types of links.

**Step 1:** Insert four **SOURCES** named **SrcParts1**, **SrcParts2**, **SrcParts3**, and **SrcPart4** to create the four different part types (i.e., drag four **MODEL ENTITIES** onto the model canvas named **EntPart1**, **EntPart2**, **EntPart3**, and **EntPart4**).<sup>43</sup> Four **SOURCES** are necessary because each part has its own arrival process. Make sure to change the *Entity Type* property of the four **SOURCES** to produce the appropriate entity type.

**Step 2:** Insert three **SERVERS** (i.e., one for each workstation named **SrvStationA**, **SrvStationB** and **SrvStationC**) and one **SINK** named **SnkPartsLeave**. Be sure to name all the objects, including the entities, so it will be easy to recognize them later as seen in Figure 5.2.

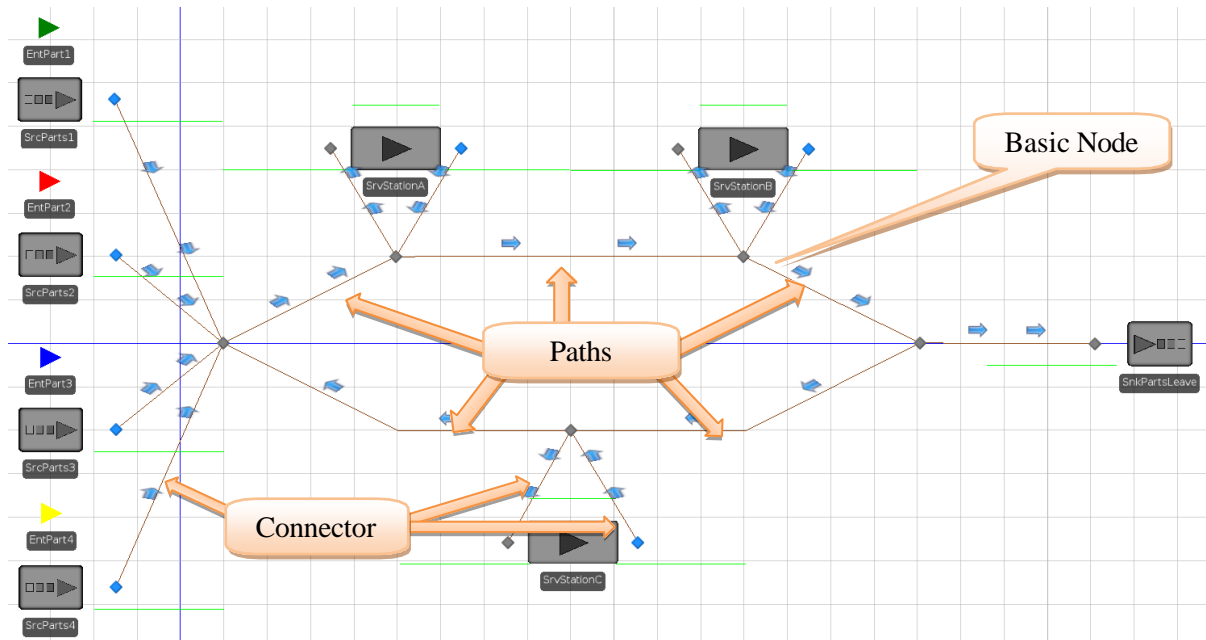
**Step 3:** Probably the best method to simulate the circular travel pattern would be to use a **BASICNODE** at the entry and exit point at each station.<sup>44</sup> This allows the entities to follow the sequence pattern without having to go through an unneeded process. A **BASICNODE** is a simple node to connect two different links while a **TRANSFERNODE** can connect different links as well but provides the ability to select destination, path, and/or require a **TRANSPORTER**.

---

<sup>43</sup>To do this efficiently, drag one **SOURCE** on to the facility and name it **SrcParts**. Then copy and paste it three times which will take advantage of the SIMIO naming scheme (i.e., it will create **SrcParts1**, **SrcParts2**, and **SrcParts3**). Now, rename the first source **SrcParts4**. Repeat this method for the four entities starting with **EntPart**.

<sup>44</sup> A **TRANSFERNODE** used to enter/exit from a station can disrupt the SIMIO internal sequencing of sequence steps if one is not careful.

**Step 4:** Now connect all the objects using **CONNECTORS** and **PATHS**. Note that because there is no time lapse or distance between the circular track and the server objects, we will use connectors. The distances between stations will be calculated using the five paths connecting each entry and exit node. Be sure that you draw the connectors and paths in the right direction – zoom in on your model to see the “arrows” as shown in Figure 5.2 (i.e., links are entering the input nodes of the **SERVERS** while leaving the output nodes).



**Figure 5.2: Initial Model**

**Step 5:** Change the properties of each entity so that they now travel at the desired speed of two miles per hour.<sup>45</sup> Also change the color of the entities so you can distinguish between the four part types.

**Step 6:** Change the properties of the four source objects so that they correspond to the correct parts and have the desired interarrival times and entities per arrival. We chose an Exponential, Normal, and Uniform as the distributions of interarrival times for part types one, two, and three respectively with the parameters given. For part type four, we chose a constant interarrival time of 70 minutes and *Entities Per Arrival* property set to 5 as seen in Figure 5.3.

Properties: SrcParts4 (Source)	
<input type="checkbox"/> Show Commonly Used Properties Only	
<b>Entity Arrival Logic</b>	
Entity Type	EntPart4
Arrival Mode	Interarrival Time
<input checked="" type="checkbox"/> Time Offset	0.0
<input checked="" type="checkbox"/> Interarrival Time	70
Units	Minutes
Entities Per Arrival	5

**Figure 5.3: Specifying Parameters for the SrcPart4**

**Step 7:** Set the distances for each of the five paths so they correspond to the distances between the stations. You will have to set the *Drawn To Scale* property for each to “False” in order to do this and then specify the logical length and units of type “Yards”.

<sup>45</sup> Select all four entities using the *Ctrl* button and specify the speed once for all entities.

**Step 8:** Set the processing times for all the stations using the Pert (BetaPert) distribution. In general, we prefer the Pert distribution to the Triangular distribution. It is specified like the Triangular (minimum, most likely, maximum), but has “thinner” tails, which, we believe, better represents the expected behavior of the processing time distribution. See Appendix A for more information.


## Part 5.2: Setting Capacities

We will need to set the capacities for the three workstations A, B, and C.

**Step 1:** Set the capacities for A and B to one and two respectively. For Station C, we will need to add a work schedule to accommodate the changes in capacity.




**Step 2:** Go to the “Data” tab and select “Schedules” and modify the default schedule (**StandardWeek**) and the default day pattern (**StandardDay**). Rename the work schedule to **ScheduleForC**<sup>46</sup> and leave the *Start Date* property alone to remain the default date the simulation will start in the *Run* setup.

SIMIO will repeat the work cycle over how many days are in the work pattern. Because our simulation will start at 12am and run for 40 hours we want to set the Days in *Work Schedule* property to 1 and let the cycle repeat after the first 24 hours.

Work Schedules		Day Patterns			
Name	Start Date	Description	Days	Day 1	
 ScheduleForC	6/25/2014	Standard Work Week Schedule	1	StandardDay	

**Figure 5.4: Setting up the WorkSchedule**

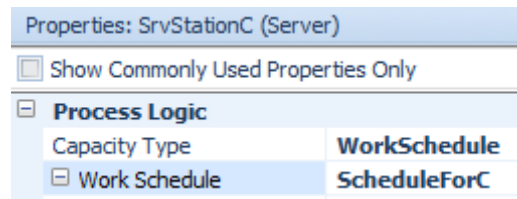
Now select the “Day Patterns” tab to modify the **StandardDay** day pattern to match the one in Figure 5.5. Change the *Start Time* property of the first row to 12:00 AM which will cause the *Duration* to change to match the *End Time*. Change it back to “4 hours” and click the *Enter*. After changing the value of the capacity to two, repeat this process for the second row using one as the capacity value. Next, add four new work periods by specifying the *Start Time*, the *Duration*, and the capacity value.

Work Schedules		Day Patterns					
Name		Description		Category			
▶  StandardDay		Standard 8-5 Work Day					
Work Periods							
		Start Time	Duration	End Time	Value	Cost Multiplier	Description
		12:00 AM	4 hours	4:00 AM	2	1	
		4:00 AM	4 hours	8:00 AM	1	1	
		8:00 AM	4 hours	12:00 PM	2	1	
		12:00 PM	4 hours	4:00 PM	1	1	
		4:00 PM	4 hours	8:00 PM	2	1	
▶ 8:00 PM 		4 hours	12:00 AM	1	1		
*							

**Figure 5.5: Modifying the Day Pattern to Represent Three Eight Hour Days**

**Step 3:** Go back to the “Facility” tab and set the *Capacity Type* property of Station C to “WorkSchedule” and add your work schedule, namely **ScheduleForC**.

<sup>46</sup> Double clicking the *Standard Week* name will allow you to change the name.



**Figure 5.6: Specifying the WorkSchedule for the StationC**

*Question 3:* For how many hours during a 24 hour day is the capacity of Station C equal to two?

---

*Step 4:* Save and run the model for 40 hours observing the path the different entities travel.

*Question 4:* How long is each part type in the system (enter to leave)?

---

*Question 5:* Why do you think the time in the system is large for all four part types?

---

### Part 5.3: Incorporating Sequences

The way the current model is set up, entities will branch at nodes based on link weights and the direction on the link. However, we want the entities to follow a desired sequence of stations as seen in Table 5.2. We would like to be sure that entities will continue along the circular pattern until they reach the node that takes them to their particular station. To accomplish this we will use “Sequence Tables” (from the “Data” tab).

*Step 1:* Click the “Add Sequence Table” button in the tab and rename the new table **SequencePart1** to correspond to the first entity type.

*Step 2:* Now add the sequence of nodes that you want Part 1 to follow (i.e., Station A, Station C and Exit). You only need to include the visited nodes at each station as well as the exit. Once the entities sequence has been set, it will always travel the shortest route to get to the first node on the list. The correct sequence for **Part1** is given in Figure 5.7.<sup>47</sup>

Sequence Part1	
	Sequence
1	Input@SrvStationA
2	Input@SrvStationC
▶ 3	Input@SnkPartsLeave
*	

Be sure to include the exit in the sequence

**Figure 5.7: Sequence for Part1**

*Step 3:* Next, add three more sequence tables to correspond to the remaining three entities (i.e., **SequencePart2**, **SequencePart3**, and **SequencePart4**). Set the sequences for each of the other three parts in their corresponding sequence tables as well. Make sure the last node is associated with the **SINK**. Note that the station’s names may appear differently in your sequence if you named them differently.

---

<sup>47</sup> Note the input nodes of the associated **SERVERS** and **SINK** are specified rather than the actual object. Note the names of the nodes are in the form **Input@ObjectName**.



**Step 4:** Now go back to the “Facility” tab and click on the **EntPart1** entity. Set the *Initial Sequence* property to the “**SequencePart1**” under the “*Routing Logic*” section which you just created for “Part 1” as seen in Figure 5.7. Do the same for each of the other entity types and their associated sequences.

Routing Logic	
Initial Priority	1.0
Initial Sequence	SequencePart1

**Figure 5.8: Specifying an Entity to follow a Particular Sequence**

**Step 5:** For each of the seven **TRANSFERNODES** (output nodes at each source and server), change the *Entity Destination Type* property to “By Sequence” for each one.<sup>48</sup> The first time an entity passes through a transfer node where its destination is “By Sequence”, the node sets the destination of the entity to the first row index of the **SEQUENCE TABLE**. After that, every time an entity passes through a transfer node with the “By Sequence” destination type the row index of the table is increased by one and the destination of the entity is set to the next node in the table.<sup>49</sup>

**Table 5.3: Definition of Basic and TransferNodes**

 <b>BasicNode</b>	Simple node to support connection between links. Often used for Input nodes of objects (e.g., <b>SINK</b> , <b>SERVER</b> , etc.).
 <b>TransferNode</b>	Supports connection between links but has the ability to select entity destination as well as request a transporter. Output nodes of objects ( <b>SOURCE</b> , <b>SERVER</b> , etc.) are <b>TRANSFERNODES</b>

**Question 6:** Will passage through a **BASICNODE** also change the row index of the **SEQUENCE TABLE**?

---

**Question 7:** Will passage through a **TRANSFERNODE** always change the row index of the **SEQUENCE TABLE**?

---

**Step 6:** It will also help to make sure they follow the correct sequence. Run the simulation for 40 hours at a slow speed to be sure each of the parts follows their sequence appropriately.

**Question 8:** Turn off the arrivals of all but one part type by changing the *Entities Per Arrival* property of the **SOURCE** object to zero for all but one **SOURCE**. Does it follow the expected sequence?

---

**Step 7:** If you turned off any source, turn them back on by changing the *Entities Per Arrival* property back to the original value. Save the current model and run it for 40 hours.

**Question 9:** How long is each part type in the system now (enter to leave)?

---

<sup>48</sup> To set them all at once, press the *Ctrl* key while selecting all seven **TRANSFERNODES** and then set the *Entity Destination* to “By Sequence.”

<sup>49</sup> If the current row index is on the last row and the entity enters a **TRANSFERNODE** that has “By Sequence” before reaching the current destination node, the row index will start over at one.

## Part 5.4: Embellishment: New Arrival Pattern and Processing Times

We now learn that the parts arrive in one stream of arrivals (i.e., with a mean arrival rate of 6 per hour, Poisson distributed<sup>50</sup>), instead of four. There are still four part types but the type is random, based on the percentages specified in in Table 5.4. The number of each that arrive doesn't change (meaning Part 4 arrives 5 at a time).

**Table 5.4: Part Type Percentage**

Part Type	Percentages	Number to Arrive
Part 1	25%	1
Part 2	35%	1
Part 3	25%	1
Part 4	15%	5

*Question 10:* When do you think a single source type of model is preferred over a multiple source model?

---

Previously, the processing times at the various stations were the same regardless of the part type. Now we discover the processing times depend on the particular part type as given in Table 5.5. In this table, the sequences are the same as before, but here the processing times have been added.

**Table 5.5: Entity Sequences and Processing Times (in Minutes)**

Step	1	2	3
Part 1	Station A (Pert(2,5,8))	Station C (Pert(2,6,11))	
Part 2	Station A (Pert(1,3,4))	Station B (Uniform(5 to 11))	Station C (Uniform(2 to 11))
Part 3	Station A (Triangular(2,5,8))	Station B (Triangular(5,9,11))	
Part 4	Station B (Pert(5,9,11))	Station C (Triangular (2,6,11))	

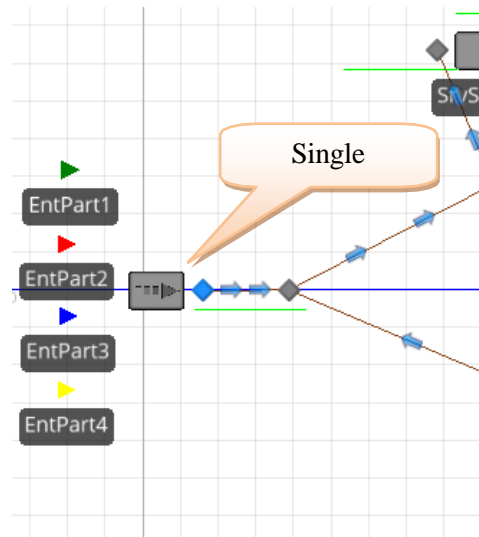
*Question 11:* Do you think that it is realistic that the processing time for Part 1 on Station A might be different for Part2 on that same Station A?

---

**Step 1:** Now delete all but one of the `SOURCES` referring to Figure 5.9 as an example renaming the one `SOURCE` as **SrcParts**. We will return to specifying this source after we specify the processing times.

---

<sup>50</sup> Remember that this Poisson arrival rate is equivalent to an Exponential interarrival time with a mean of 10 minutes. SIMIO wants the interarrival time, not the arrival rate – so be prepared to make the conversion.



**Figure 5.9: Manufacturing Cell Model**

**Step 2:** To model the processing times which are dependent on the part type, the processing times will be added to the original sequencing tables. When the entity is assigned a destination (i.e., a row in the sequence table is selected), the processing time column can be accessed for that a row. (Note, you could use a more complicated expression for the processing time by checking to see if this is a Part A, then make the processing time X, etc. but is not very easy if there is a large number of parts).

**Step 3:** Go to the “Data” tab and click on the **SequencePart1** table which is associated with Part 1 types. Since the processing times can be any distribution (i.e., any valid expression), add a new *Expression* type column from *Standard Property* drop down named **ProcessingTimes**. Make sure the column has the appropriate units by making the *Unit Type* be “Time” and the *Default Units* to be “Minutes.”

**Step 4:** Set the Processing times for **SrvStationA** (i.e., the first row) to a `Random.Pert(2,5,8)`, **SrvStationC** (i.e., the second row) to a `Random.Pert(2,6,11)` and finally the **SnkPartsLeave** or third row will remain zero as seen in Figure 5.10.

Sequence Part1	Sequence Part2	Sequence Part3	Sequence Part4
	Sequence	ProcessingTimes (Minutes)	
1	Input@SrvStationA	Random.Pert(2,5,8)	
2	Input@SrvStationC	Random.Pert(2,6,11)	
3	Input@SnkPartsLeave	0.0	
*			

**Figure 5.10: New Sequence Table for Part Type 1**

**Step 5:** Repeat the last two steps for the remaining three part sequence tables using the appropriate processing times from Table 5.5.

**Step 6:** In order to have a single source produce multiple types of entities, a new data table has to be created to specify the part type entities and the part mix percentages. Add a new data table called **TableParts** with four columns with the specified types as seen in Table 5.6.



**Table 5.6: Column Definitions**

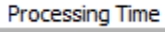
Column	Column Name	Column Type
1	<b>PartType</b>	<i>Entity</i>
2	<b>ProcessingTimes</b>	<i>Expression</i>
3	<b>PartMix</b>	<i>Integer (or Real)</i>
4	<b>NumberToArrive</b>	<i>Integer (or Real)</i>

**Step 7:** Next specify each entity as a new row with the appropriate processing times, part mix percentages and sequence table as shown in Table 5.7.

**Table 5.7: Content of TableParts**

Table Parts	Sequence Part1	Sequence Part2	Sequence Part3	Sequence Part4
	Part Type	ProcessingTimes (Minutes)	Part Mix	Number To Arrive
1	EntPart1	SequencePart1.ProcessingTimes	25	1
2	EntPart2	SequencePart2.ProcessingTimes	35	1
3	EntPart3	SequencePart3.ProcessingTimes	25	1
4	EntPart4	SequencePart4.ProcessingTimes	15	5

Specifying `SequencePart1.ProcessingTimes` states that the processing time will come from the column associated with `ProcessingTimes` in the table `SequencePart1` for the current row.

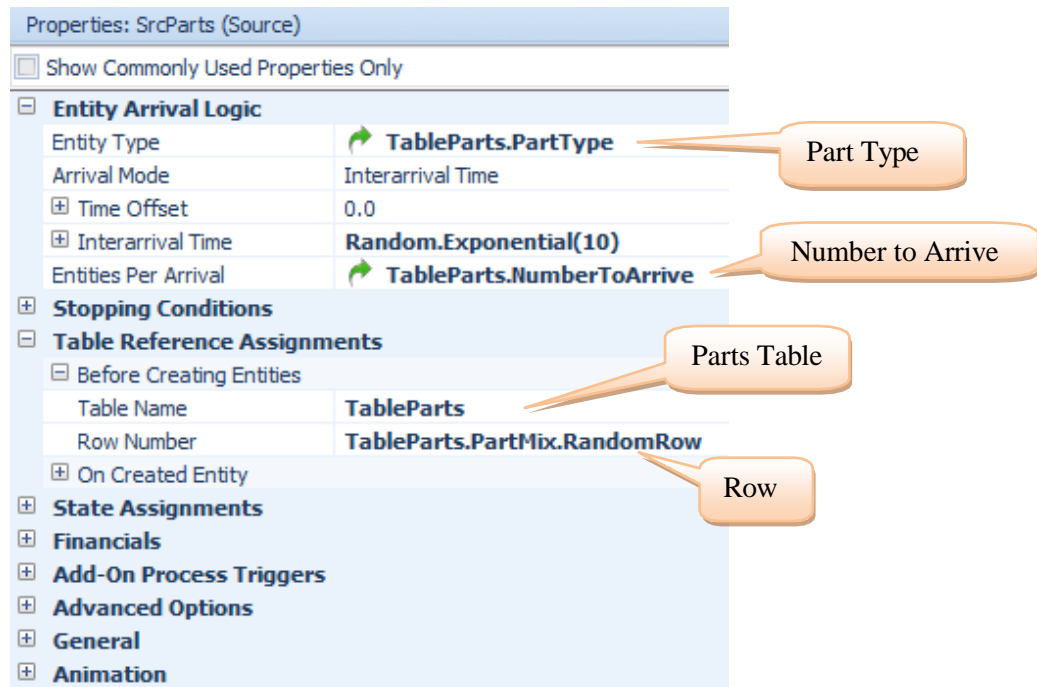
**Step 8:** In order to use the dependent processing times, change each of the processing time expressions in the three stations to be **TableParts.ProcessingTime**. This logic specifies that the processing time will come from the *ProcessingTimes* column of the data table **TableParts**. As the parts move through their individual sequence table (i.e., row to row), it will retrieve the processing time for the current row. The property is now a reference property type  **TableParts.ProcessingTime** which can only be changed back to a regular property by right clicking on the label (*Processing Time*) and resetting it.

**Step 9:** To generate multiple part types from the same source, the *Entity Type* needs to be changed from a specific entity type. In the `SOURCE`, specify the *Entity Type* to come from the **TableParts.PartType**. This will allow the *Entity Type* to come from one of the four entities specified in the table.

**Step 10:** Set the interarrival time of entities to be an `Exponential` with a mean of ten minutes (i.e., same as a Poisson arrival rate of six per hour). Set the *Entities Per Arrival* to **TableParts.NumberToArrive**.

**Step 11:** To determine which specific part type is created, the part mix percentages need to be utilized. The `SOURCE` has a property category called “Table Reference Assignment” which can assign a table and a row of that table to the entity.<sup>51</sup> There are two possible assignments: *Before Creating Entities* and *On Created Entity*. We want to do the assignment before the entity is actual created in order to create the correct entity type and the correct number to arrive. Therefore, specify the table shown in Figure 5.11.

<sup>51</sup> In later chapters we will utilize *Processes* and the *Set Row* step to assign a table to an entity.



**Figure 5.11: Making the Table Reference Assignment**

You specify which table (**TableParts**) and which row in the table to associate the object. In this case, the row is determined randomly by using the `PartMix` column and specifying that a random row should be generated (`TableParts.PartMix.RandomRow`).<sup>52</sup>

**Step 12:** Save and run the simulation model for 40 hours at a slow speed to be sure each of the parts follows their sequence appropriately and answer the following questions.

**Question 12:** How long is each part type in the system (enter to leave)?

---

**Question 13:** What is the utilization of each of the `SERVERS`?

---

**Question 14:** What is the overall time for all part types in the system?

---

## Part 5.5: Using Relational Tables

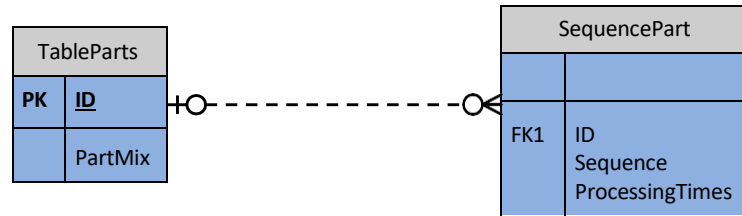
The original model utilized just four part types, however we may need to add additional parts. This would require adding a new entity, adding rows into the **TableParts** along with creating a new sequence table for each part. Each addition now requires design time to continue to update the model as well as adding sequence tables. Instead, can we drive these specifications from a spreadsheet that contains the number of parts and their sequences? Or is there an easier way to implement the creating and sequencing of parts?

One of the inventive modeling constructs in SIMIO is the use *relational data tables*. SIMIO has implemented limited database capabilities that can facilitate the modeling of relationships. Figure 5.12 shows a “relational diagram”<sup>53</sup> where the **TableParts** data table is related to the **SequencePart** Table via the **ID** column. **ID** is a

<sup>52</sup> SIMIO will normalize the numbers so the probability sums to one. Therefore, you may specify whole numbers 25, 25, and 50 rather than percentages 0.25, 0.25, and 0.5.

<sup>53</sup> More formally it is called an Entity Relationship Diagram, but the word “Entity” is use differently than it is used in SIMIO.

*primary key* (PK) in the **TableParts** which uniquely identifies each row in the table (i.e., cannot be any duplicates). The child table (i.e., **SequencePart**) inherits the primary key from the parent table (i.e., **TableParts**) which is denoted as a *foreign key* (FK). A particular part can have many rows (records) in the child table.



**Figure 5.12: Setting up a Relation between Table Parts and the Sequence Part**

**Step 1:** First, we will no longer need just four part types since we want to make this a more generic model. Delete all but one of the part type **ENTITIES** and rename the remaining entity **EntParts**. From the *Additional Symbols* section, add five additional symbols making sure to color each new entity as seen in the rotated Figure 5.13.<sup>54</sup>



**Figure 5.13: Adding Additional Symbols for the Part Entity**

**Step 2:** For the **SrcParts**, set the *Entity Type* property to **EntParts**.<sup>55</sup>

**Step 3:** From the *Data* tab, delete all the columns of the **TableParts** except for the *Part Mix* and *Number to Arrive* columns. Insert a new *Integer Standard Property* named **ID**. From the *Edit Column* section, you can move the new column to the left and set the column as a PK using *Set Column As Key* button in the same section as seen Figure 5.14. Set the **ID** for each of the part types to a unique number (i.e., 1, 2, 3, and 4).

Table Parts		Sequence Part		
	ID	Part Mix	Number To Arrive	
1	1	25	1	
2	2	35	1	
3	3	25	1	
4	4	15	5	

**Figure 5.14: Parent Table TableParts with the ID Primary Key**

**Step 4:** Select the **SequencePart1** sequence table and rename it **SequencePart**. From the *Add Column* section, select the *Foreign Key* button. Set the properties for the column to specify **TableParts.ID** as the primary table key to relate column too as seen in Figure 5.15. Change the name of the column to be ID.

<sup>54</sup> Note the symbols start with "0" index.

<sup>55</sup> Right click the property and reset it back from the reference property.

Properties: ID (Foreign Key Property)

☐ Show Commonly Used Properties Only

**Value**

Default Value: 1

Table Key: TableParts.ID

**Appearance**

Display Name: ID

**Operational Planning**

**General**

Name: ID

Description:

Required Value: True

**Figure 5.15: Specifying the Foreign Key Property.**

**Step 5:** Set the **ID** for all the current rows to be one.

**Step 6:** Select the **SequencePart2** tab and then highlight all the rows by selecting the first row using the row selector and dragging down. Then copy all the rows and transverse back to the **SequencePart** table. Select the last row and then paste in all the values. Change the IDs of this new part to two.

**Step 7:** Repeat for the last two part sequence tabs setting their **IDs** to three and four respectively as seen in Figure 5.16.

Table Parts	Sequence Part		
	Sequence	ProcessingTimes (Minutes)	ID
1	Input@SrvStationA	Random.Pert(2,5,8)	1
2	Input@SrvStationC	Random.Pert(2,6,11)	1
3	Input@SnkPartsLeave	0.0	1
4	Input@SrvStationA	Random.Pert(1,3,4)	2
5	Input@SrvStationB	Random.Uniform(5,11)	2
6	Input@SrvStationC	Random.Uniform(2,11)	2
7	Input@SnkPartsLeave	0	2
8	Input@SrvStationA	Random.Triangular(2,5,8)	3
9	Input@SrvStationB	Random.Triangular(5,9,11)	3
10	Input@SnkPartsLeave	0	3
11	Input@SrvStationB	Random.Pert(5,9,11)	4
12	Input@SrvStationC	Random.Triangular(6,9,11)	4
13	Input@SnkPartsLeave	0	4
*			

Row Selector

**Figure 5.16: The New Sequence Parts Table**

**Step 8:** Once the table is set up, select the **TableParts** table again and observe that each of the rows now has an expander (⊕) associated with it. Figure 5.17 shows the related records for “Part type 0.” If an entity is assigned a row from the **TableParts** table, it will automatically be assigned the related records in the **SequencePart** table.

Table Parts		Sequence Part		
	ID	Part Mix	Number To Arrive	
▶ 1	1	25	1	
Sequence Part				
	Sequence	ProcessingTimes (Minutes)	ID	
▶ 1	Input@SrvStationA	Random.Pert(2,5,8)	1	1
2	Input@SrvStationC	Random.Pert(2,6,11)	1	1
3	Input@SnkPartsLeave	0.0	1	1
*				
2	2	35	1	
3	3	25	1	
4	4	15	5	

**Figure 5.17: Seeing the Related Table Information**

**Step 9:** We need to modify the entity properties to utilize the new sequence table and id for changing the picture symbol. Under the *Animation* section change the *Current Symbol Index* to be “TableParts.ID-1” so the color will change appropriately.<sup>56</sup> Also make sure the *Initial Sequence Property* is set to “*SequencePart*” as seen in Figure 5.18. Note, the entity will only receive the related records and not the entire **SequencePart** table.

Properties: EntParts (ModelEntity)	
Show Commonly Used Properties Only	
Travel Logic	
Initial Desired Speed	2.0
Units	Miles per Hour
Initial Network	Global
Network Turnaround Method	Exit & Re-enter
Routing Logic	
Initial Priority	1.0
Initial Sequence	SequencePart
Financials	
Population	
Advanced Options	
General	
Animation	
Current Symbol Index	TableParts.ID-1
Random Symbol	False
Current Animation Index	ModelEntity.Animation
Default Animation Action	MovingAndIdle
Dynamic Label Text	

**Figure 5.18: Modifying the EntParts Property to Utilize the New Tables**

**Step 10:** For each of the three server stations, change the *Processing Time* to be the reference property *SequencePart.ProcessingTimes* rather than *TableParts.ProcessingTimes* since this column is now only in the related table.

**Step 11:** Save and execute the model.

**Question 15:** What is the utilization of each of the SERVERS now?

<sup>56</sup> If we had a lot of parts, we can utilize a status label to display the ID as the part moves through the network making this work for any number of parts. This concept will be explored in later chapters.

Question 16: What is the overall time for all part types in the system?

---

Question 17: Is there anything different in this model as compared to the one from the previous section?

---

## Part 5.6: Creating Statistics

SIMIO automatically produces statistics for various objects within the simulation. For example, there are statistics on the `MODEL_ENTITY` number in system and time in system or for any `SERVER` there is a scheduled utilization, a number in the `INPUT_BUFFER` station, the waiting time in the station, etc. However, in spite of their comprehensive nature, that may be insufficient. For example, our latest model has only one entity type, so the `MODEL_ENTITY` statistics are not separated by the part type. What if we wanted to know, on the average, how many Part 3s are in the system during simulation, and what is the average time these parts spend in the system?

The two major statistical categories are: `TALLY` statistics and `STATE` statistics. The `TALLY` statistic is based on observations like waiting time, time in system, cycle time, etc. The `STATE` statistic such as number in queue, number in inventory, etc., is a time-weighted statistic such that the time a value persists is used to compute the statistical value. While the `STATE` statistic may appear to be the more complicated, it is the easiest to specify in SIMIO. For example, let's add a `STATE` statistic to determine the average number of Part 3's in the system first and then add a `TALLY` statistic for the average times in the system.

**Step 1:** Before creating a `STATE` statistic, it is necessary to create a state variable that will hold the values for example the number of Part 3's in the system. SIMIO will monitor the value of the state variable through a `STATE` statistic and weight its value according to the time a particular value persists throughout the simulation. Since the number of Part 3's in the system is a model-based characteristic (not associated with particular entities), we need to define a model-based state variable. From the *Definitions*→*States* section, insert a new *Discrete, Integer* variable with the name **GStaPart3NumberInSystem** as seen in Figure 5.19

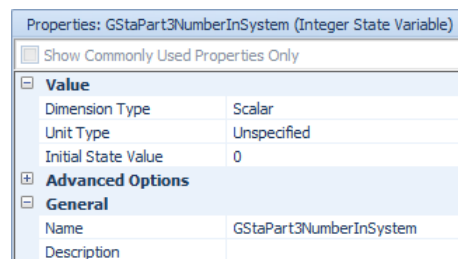


Figure 5.19: Discrete, Integer, Model-based State Variable

**Step 2:** The variable needs to be incremented when a Part 3 enters the system and then decremented when a Part 3 leaves the system. Let's use the *State Assignments* of the **SrcParts** for *Before Exiting* employing the *Repeating Property Editor* to increment the new state variable as shown in Figure 5.20.

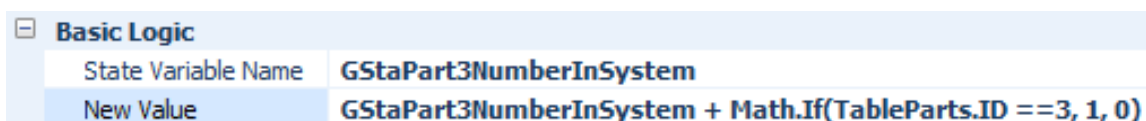
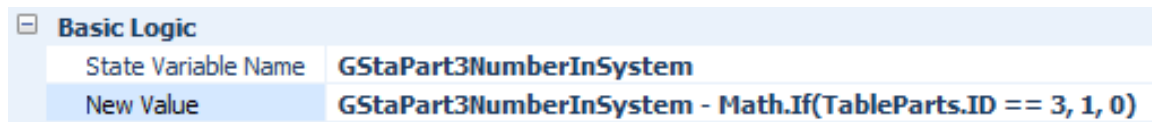


Figure 5.20: Incrementing the State Variable

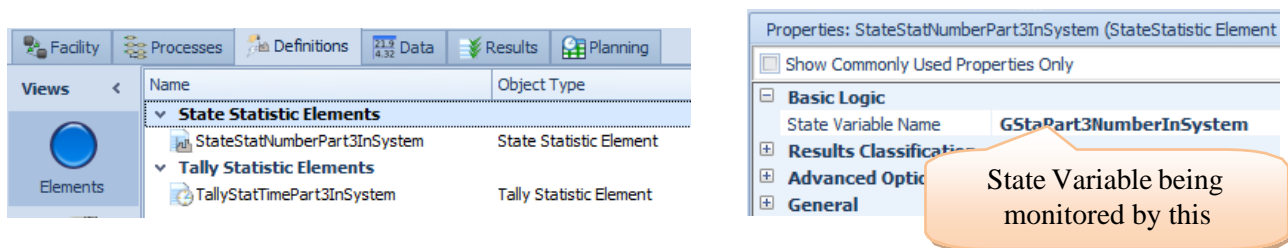
**Step 3:** The `Math.If()` function simply tests if **TableParts.ID** has the value of “3” which refers to Part 3’s then adds “1” to the state variable, otherwise the expression adds “0.”

**Step 4:** In a similar fashion, the state variable can be decremented in the *State Assignments* of the **SnkPartsLeave** for *On Entering* as shown in Figure 5.21.



**Figure 5.21: Decrementing the State Variable**

**Step 5:** Now all you need to do is to insert **STATESTATISTIC** from the *Definitions→Elements* section named **StateStatNumberPart3InSystem** that watches the state variable as seen Figure 5.22.

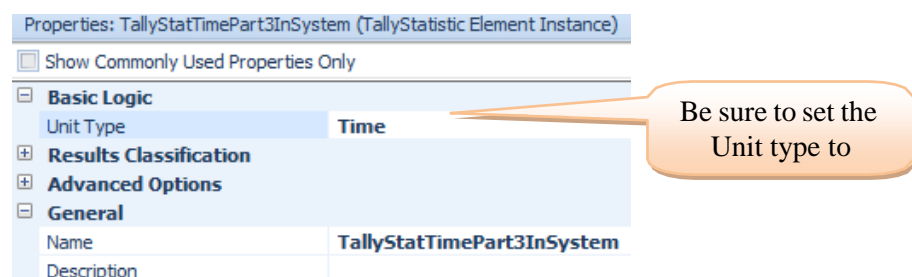


**Figure 5.22: Defining the State Statistic Element**

**Step 6:** Save your model and run your simulation for 40 hours.

**Question 18:** What did you get for the average and the maximum number of Part 3’s in the system?

**Step 7:** Now, we would like to collect time in system for the Part 3’s which is a **TALLY** statistic. The **TALLY** statistic needs to record individual observations. Tally statistics can be collected at any **BASIC** or **TRANSFER** node in their *Tally Statistics* property.<sup>57</sup> Unlike the **STATE** statistics, the **TALLY** statistic needs to be defined first. Using the *Definitions→Elements* section insert a Tally Statistic named **TallyStatTimePart3InSystem**. See Figure 5.23 for specifying the *Unit Type* property of “Time.”

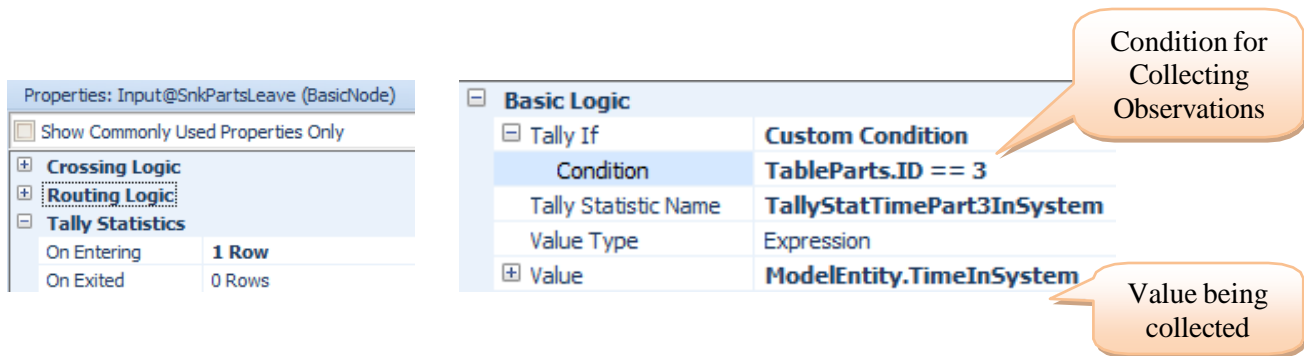


**Figure 5.23: Defining the Tally Statistic Element**

**Step 8:** Select the **Input@SnkPartsLeave** node (i.e., the one attached to the **SINK**). Under the *Tally Statistics* Section using the *On Entering* tally property, specify which observation (i.e., `ModelEntity.TimeInSystem`) to record using the *Repeating Property Editor*. Specify the collection of the tally statistic as shown in Figure 5.24.

<sup>57</sup> In later chapters we will use the *Tally* process steps to collect statistics anywhere.





**Figure 5.24: Specifying the Collection of Part 3 Time in System**

**Step 9:** Note that a *Custom Condition* causes only the Part 3’s time in system to be collected.

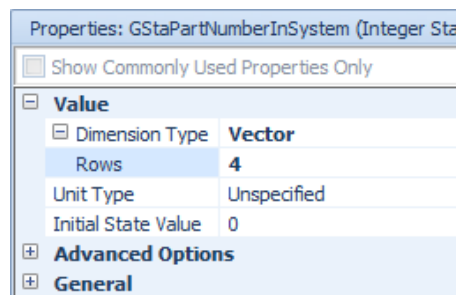
**Step 10:** Now run your simulation for the 40 hours.

**Question 19:** What did you get for the average, maximum, minimum, and number of observations for time in system for Part 3’s?

## Part 5.7: Obtaining Statistics for All Part Types

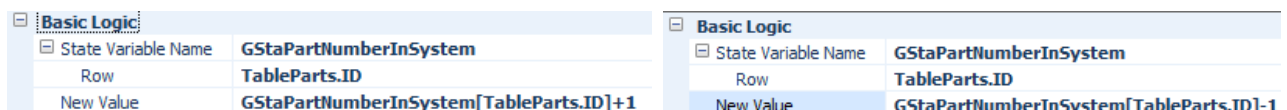
Since statistics like time in system and number in system would be useful for each part type, we will extend the statistics collection to all part types rather than just Part 3’s.

**Step 1:** First, let’s create the time in system statistics. Navigate to the *Definitions*→*States* section, and modify the definition of **GStaPart3NumberInSystem**. Rename the variable to **GStaPartNumberInSystem**. Set the *Dimension Type* property of the state variable to a “**Vector**” with “4” rows, as shown in Figure 5.25. This will create a vector of four state variables one for each part type.



**Figure 5.25: Defining the Number in System Vector**

**Step 2:** Next change the state assignments at **SrcParts** and **SnkPartsLeave** to increment and decrement the number in system for each part type respectively as shown in Figure 5.26. Note the use of the row (`TableParts.ID`) in the vector.<sup>58</sup>



**Figure 5.26: Incrementing and Decrementing the Number in System**

<sup>58</sup> The row index of the first row for Vectors, Matrices, and Data Tables is one while symbol indexes start at zero.



**Step 3:** You can now define state statistics for all the parts. Under *Definitions*→*States*, create three more state statistics named **StateStatNumberPart1InSystem**, **StateStatNumberPart2InSystem**, and **StateStatNumberPart4InSystem**. You need to specify the state variable for each of the statistics using the previously defined vector state variable. This specification is the same for each statistic, except for the row. The specification for Part 2's is shown in Figure 5.27. Modify all the four state statistics.

**Figure 5.27: Specification for StateStatNumberPart2InSystem**

**Step 4:** Now define the TALLY statistics for the other three part types (Parts 1, 2, and 4), just as was done for **TallyStateTimePart3InSystem** (see Figure 5.23 making sure the *Unit Type* property is equal to “Time”).

**Step 5:** Navigate to the *Data*→*Tables* section and select the **TableParts** DATA TABLE. Insert a new *Tally Statistic Element Reference* property named **TallyStatTimeInSystem**, which will allow us to specify a tally statistic for each part. Therefore each entity (i.e., part type) will know which statistic to update. There is little need to insert State statistics into the data table since we don't reference them within the model – only the state variable that the state statistics monitor. Fill in the tables as shown in Figure 5.28.

Table Parts		Sequence Part		
	ID	Part Mix	Number To Arrive	Tally Stat Time In System
1	1	25		1 TallyStatTimePart1InSystem
2	2	35		1 TallyStatTimePart2InSystem
3	3	25		1 TallyStatTimePart3InSystem
4	4	15		5 TallyStatTimePart4InSystem

**Figure 5.28: Data Table Including Tally Statistics**

**Step 6:** Once the Table row reference has been assigned, each MODEL ENTITY (i.e., part) will have an associated TALLY statistic which will eliminate using complicated deciding logic to determine which TALLY statistic to update. Select the **Input@SnkPartLeave** BASIC node in front of **SnkPartLeave** and under the *Tally Statistics* entry click on the *On Exited*.<sup>59</sup> Use the *Tally Statistic Name* will be based on the row that has been assigned to the entity (i.e., TableParts.TallyStatTimeInSystem). The observation *Value* will be the model entity time in system value as seen in Figure 5.29 to utilize the entity's tally statistic.

**Figure 5.29: Time in System Tally Statistics**

<sup>59</sup> The *On Entering* could be used as well.

**Step 7:** Now re-run the simulation and examine the results.

**Question 20:** Do the tally statistics on time in system for each part type seem correct?

---

**Question 21:** Do the state statistics for number in system for each part type seem correct?

---

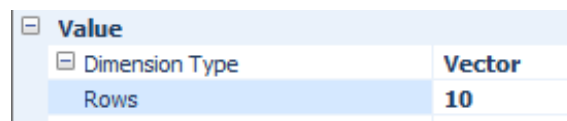
## Part 5.8: Automating the Creation of Elements for Statistics Collection

Adding intelligence to entities made it easy to model different routings as well as to calculate time in system and number in system statistics for each part type without having to create multiple sink and source objects corresponding to the individual parts. But we would still need to define the associated statistic objects and state variables. Such an approach becomes very cumbersome if we want to add many new part types.

An advantage of using related tables is that they may be linked (bound) directly to a spreadsheet containing many part types and sequences. Nevertheless, we may want statistics for time in system (i.e., cycle times) and number in system for each of the different part types. However, this situation will require a **TALLY** and **STATE** statistic for each part type. Having to define each statistic could negate the advantages of using related tables as individual tally and state statistic elements need to be created and then the correct statistic must be collected for the particular part. However, SIMIO allows for the automatic creation of state variables and statistic elements.

**Step 1:** If you are using the model from the previous section, you should delete all the **TALLY** and **STATE** *STATISTIC* elements, so these can be created automatically.

**Step 2:** Since the number of part types might change, the state variable **GStaPartNumberInSystem** will need to be modified to match the number of parts in the table each time. In our case, we will need to set it equal to maximum possible number of part types in the system (e.g., 10).<sup>60</sup> The size of the state variable can be tied to the number of rows and columns of a data table. From the *Definitions*→*States* section change the *Rows* property from “4” to “10”.



**Figure 5.30: Specifying the Size of the State Variable to Match the Table**

**Step 3:** Navigate to the *Data*→*Tables* section and select the **TableParts** DATA TABLE. Notice *Statistic Element Reference* property named **TallyStatTimeInSystem**. Remove the tally statistics from that column, so it looks like Figure 5.31. Note that the tally statistics references are missing.

Table Parts		Sequence Part		
	ID	Part Mix	Number To Arrive	Tally Stat Time In System
1	0	25	1	✗
2	1	35	1	✗
3	2	25	1	✗
4	3	15	5	✗

**Figure 5.31: Adding a Tally Stat Reference Column**

---

<sup>60</sup> “Matrix from Table” is another *Dimension Type* for state variables which will create a two dimensional matrix based on the number of rows and numeric columns of a table. Dynamically. However, it initializes the variable based on the table values.

**Step 4:** Once one specifies entries (i.e., names), the elements will be created. The element’s initial properties can be manually specified (e.g., *Category*, *Data Item*, *Unit Type*, etc.) from the *Definitions→Elements* section. However, since the number of part types can be dynamic we do not want to have to specify these properties each time, which would defeat the benefit of automatic specification of the statistical elements. Therefore, the elements that are created can also have their properties initialized by specifying another column in the table as their values. For our problem, the category and data item classification of the TALLY statistics can be specified as we want it to be the same as other time in system statistics (i.e., “FlowTime” and “TimeInSystem”). Therefore, insert two *Standard String* property columns named **Category** and **DataItem** as seen Figure 5.32.

Table Parts		Sequence Part							
	ID	Part Mix	Number To Arrive	Tally Stat	Time In System	Category	Data Item	Unit Type	State Stat Num In System
1	1	25	1	×		FlowTime	TimeInSystem	Time	×
2	2	35	1	×		FlowTime	TimeInSystem	Time	×
3	3	25	1	×		FlowTime	TimeInSystem	Time	×
4	4	15	5	×		FlowTime	TimeInSystem	Time	×

**Figure 5.32: Adding Properties to the Statistical Elements**

**Step 5:** For the *Unit Type* property of the TALLY statistic, insert a *Standard Enumeration* property column named **UnitType** into **TableParts**, as was done in Figure 5.32. An enumeration is a property whose value is specified from a list of potential values. In this case unit types can be “Unspecified”, “Time”, “Length”, etc. For the *Enumeration* column property, specify the *Enum Type* to be “UnitType” to pull its values from unit type enumeration list and the *Default Value* should be “Time” as seen in Figure 5.33.

Properties: UnitType (Enumeration Property)

☐ Show Commonly Used Properties Only

☒ **Logic**

Default Value	Time
Enum Type	UnitType

☒ **Appearance**

Display Name	UnitType
--------------	----------

☐ Captions

☒ **Operational Planning**

☒ **General**

Name	UnitType
------	----------

**Figure 5.33: Setting up the Enumeration Property Column**

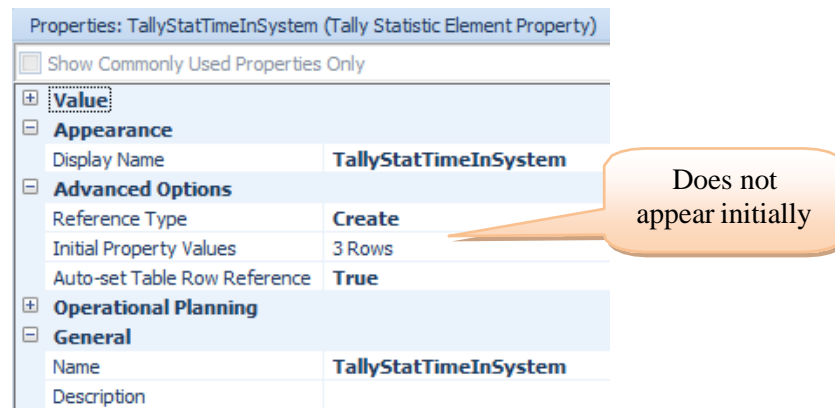
**Step 6:** Next, insert a *State Statistic Element Reference* property named **StateStatNumInSystem** as seen in Figure 5.32.

**Step 7:** Rather than creating each TALLY statistic to specify as entry values, SIMIO has the ability to automatically create elements (TALLY and STATE statistics, MATERIALS, etc.) specified from a table column. Therefore, each part row can have a specified TALLY statistic (i.e., name) that will automatically be created and used to keep track of its on time in system statistics. By default the *Element Reference* property column will be of a “Reference” type meaning the element would need to be already created in order to be specified as an entry in the table.

- Therefore, change the *Reference Type* property to “Create” which is explained in Table 5.8 and seen in Figure 5.34.
- Set the *Auto-set Table Row Reference property* to “True” as well.

**Table 5.8: Specifying Properties to Automatically Create the Element**

<i>Property</i>	<b>Description and Value</b>
<i>Reference Type</i>	“ <i>Reference</i> ” - It allows you to simply specify an element that is already been defined under the “Definitions” tab.
	“ <i>Create</i> ” - Entries in this column will now automatically create a new element of the specified type with the name of the entry. Note these elements will appear in the <i>Definitions</i> → <i>Elements</i> but they cannot be deleted from here. Any changes in names in either location (i.e., <i>TABLE</i> or <i>Definitions</i> → <i>Elements</i> ) will be reflected in both.
<i>Initial Property Values</i>	This can be used to initialize the element based on values in other columns of the table.
<i>Auto-set Table Row Reference</i>	If “True”, then the element that is pointed to by each row will automatically be given a table reference set to that row. This has to be case if your creating and initializing the element based on other columns

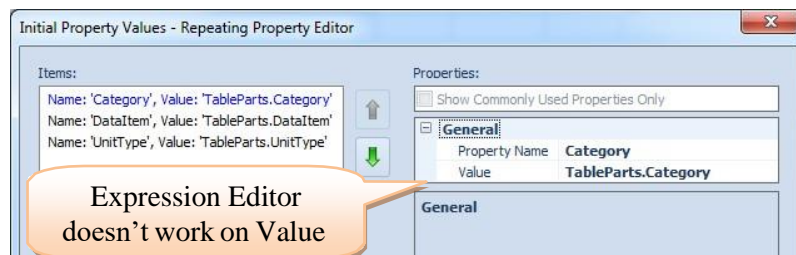


**Figure 5.34: Specifying Element Reference to Automatically Create the Element**

**Step 8:** After specifying the “Create”, the statistical elements be created but they will not have their properties referenced. In order to use the entries specified in these three new columns as the initial properties, select the **TallyStatTimeInSystem** column and then click the *Repeating Property* button of the *Initial Property Values* property. Next, add the following three properties and values as specified in Table 5.9.<sup>61</sup> To specify the initial properties use the repeating property window as shown in Figure 5.35.

**Table 5.9: Specifying the Initial Properties**

<b>Property Name</b>	<b>Value</b>
<i>Unit Type</i>	TableParts.UnitType
<i>Category</i>	TableParts.Category
<i>DataItem</i>	TableParts.DataItem



**Figure 5.35: Specifying the Initialization Properties of the TallyStat Reference Column**

<sup>61</sup> You will need to type in the values directly as this is not an expression editor.

**Step 9:** Select the **StateStatNumInSystem** and specify that this column will be created as well as seen in Figure 5.36.

**Figure 5.36: Having the State Statistics to be Automatically Created**

**Step 10:** Next we need to initialize the new state statistics in a similar fashion as the Tally statistic column. However, this time we will not setup additional columns to specify the parameters. Set the “Category” and “DataItem” properties to string values and the “StateVariableName” property will use the `TableParts.ID` to specify which state variable the statistic should monitor as seen in Table 5.10 and Figure 5.37.

**Table 5.10: Specifying the Initial Properties**

Property Name	Value
<i>StateVariableName</i>	<code>GStaPartNumberInSystem[TableParts.Id]</code>
<i>Category</i>	Content
<i>DataItem</i>	NumberInSystem

**Figure 5.37: Initial Parameters of the State Statistic Column**

**Step 11:** Add in the specifications for the properties of the four part types as shown in Figure 5.38.

Table Parts	Sequence Part						
	ID	Part Mix	Number To Arrive	Tally Stat Time In System	Category	Data Item	Unit Type
1	0	25	1	⊗	FlowTime	TimeInSystem	Time
2	1	35	1	⊗	FlowTime	TimeInSystem	Time
3	2	25	1	⊗	FlowTime	TimeInSystem	Time
4	3	15	5	⊗	FlowTime	TimeInSystem	Time

**Figure 5.38: Specifying the Properties of the Statistical Elements**

**Step 12:** You may now want to reset the *Reference Type* property to back to “Reference”. Then when you set the *Reference Type* property to back to “Create”, the statistical elements will be automatically be created as shown in *Definition→Tally Statistical Elements (Auto-Created)* section. Note how each statistic takes its *UnitType*, *Category*, and *DataItem* from the DATA TABLE.

**Step 13:** Now for every new row that is added, a new TALLY and STATE statistic will be created that is initialized from the entry values of the *Category*, *DataItem* and *UnitType* columns. For any rows that existed before the two columns were created, the values will not be initialized.<sup>62</sup> We can cut and paste values back and forth from SIMIO tables and Microsoft Excel™ spreadsheets. Select the entire table by highlighting the row selectors and then cut the rows (*Ctrl-x*). Next the paste them into a Microsoft Excel™ spreadsheet so you don't have to retype all of the information. Modify the names of the tally and state statistics, categories, data items and the unit types as seen in Figure 5.39.

	A	B	C	D	E	F	G	H
1	1	25	1	TallyStatP1	FlowTime	TimeInSystem	Time	StateStatP1NumInSystem
2	2	35	1	TallyStatP2	FlowTime	TimeInSystem	Time	StateStatP2NumInSystem
3	3	25	1	TallyStatP3	FlowTime	TimeInSystem	Time	StateStatP3NumInSystem
4	4	15	5	TallyStatP4	FlowTime	TimeInSystem	Time	StateStatP4NumInSystem

**Figure 5.39: Using Excel to Modify the Values**

**Step 14:** Copy the values in Excel and then select the empty row in the SIMIO data table by clicking on the row selector and then paste the values into the table which should now look like the one in Figure 5.31.



**Step 15:** Save and run the model for 40 hours.

**Question 22:** What is the average time in the system and number in system for each of the parts?

---

**Question 23:** Does the time in system for Part3 in TallyStatistic3 agree with the previous value computed?

---

**Step 16:** Now add a fifth part type by creating a sequence and update only the data table (don't create any elements) by copying the fourth part information changing the tally and state statistic column values as well as copy the fourth part sequences for the fifth part.

**Question 24:** Did SIMIO create the tally statistical element automatically?

---

**Question 25:** When you execute the model, did the flow time and time in system statistics for the fifth part type appear with the proper unit type, category, and data item?

---

## Part 5.9: Commentary

- The *Start Date* property of the Work Schedule intuitively would be when the schedule will start. So if one specified a future date then nothing would happen until that date. That is not the case. The *Start Date* property represents the particular day the first day of the pattern will start. If this date is in the future compared the start of the simulation, it will repeat backwards to the current date based on the pattern. For example, if we have a three day pattern named Day1, Day2, and Day3 and the work schedule start date is set to 09/17/15, schedule will follow the Day1 pattern for this date. If the simulation starts on 09/13/15, then the pattern for 09/13 is Day3, 09/14 is Day1,

---

<sup>62</sup> We can either modify the properties our self or we can delete the rows and recreate them. You can select the AutoCreate dropdown in each column value.

09/15 is Day2, 09/16 Day3 which makes Day1 be on 09/17 and Day2 on 09/18, etc. In order to have a work schedule not start until a particular day, exceptions to the work day have to be employed.

- Another way to look at SIMIO properties and states is that property values are established for each object in the Facility window at the beginning of the simulation execution and cannot be changed during “run-time”, whereas the value of states are also established at initialization but they can be changed during “run-time”. Objects that are created during run-time, such as `ENTITY`, have their properties established during an instance of run-time which is during the initialization of that instance, but they otherwise cannot be changed during the simulation execution.
- The use of relational tables offers a tremendous advantage over other simulation languages in being able to assign one table and then automatically be assigned all records of related tables. We will explore this feature more in later chapters. In later chapters, we will demonstrate how you can bind data tables to Excel spreadsheets which could facilitate the automatic creation of new parts and sequences.