Learning Business Process Simulation Via Experiential Learning

Paul F. Schikora, Indiana State University, USA
Brian D. Neureuther, SUNY Plattsburgh, USA

ABSTRACT

The use of discrete event simulation as a process analysis and improvement tool is no longer limited to industrial engineering curricula. With advancements in desktop computing power, we have seen user-friendly simulation software packages become available (e.g. ProModel, Arena, ProcessModel). However, we have found it desirable that students still learn the very basic concepts behind these simulation models in order to better understand their development and use. We present a simple classroom game that teaches students the basic discrete-event simulation concepts and processes without requiring them to learn all the underlying mathematics and scientific theory.

Keywords: Discrete Event Simulation, Pedagogy, Operations Management, Games

INTRODUCTION

The use of discrete event simulation as a process analysis and improvement tool is no longer limited to industrial engineering curricula. With advancements in desktop computing power, we have seen user-friendly simulation software packages become available (e.g. ProModel, Arena, ProcessModel). Using graphical interfaces, and hiding much of the simulation science, they allow relatively advanced simulation techniques to be applied by practitioners as part of process improvement projects. This has made it more attractive to teach simulation by using a process improvement methodology in undergraduate business curricula, where extensive knowledge of the science of simulation is not necessary. However, we have found it desirable that students still learn the very basic concepts behind these simulation models in order to better understand their development and use.

In our experience teaching discrete event simulation to undergraduate business students, we have observed significant student difficulty understanding the basic event processing logic that forms the foundation of the methodology when taught through traditional lecture methods. We therefore searched for an alternative method of teaching these concepts and eventually developed the classroom game exercise we present here. There is significant literature addressing the efficacy of active learning techniques in general, and classroom games (see Cruishank & Telfer (2001)). The literature generally addresses overall student performance as a result of experiential activities in the classroom, though we found no works that specifically address the efficacy of such activities in teaching the topics of simulation. Several studies do look at the use of games in teaching technical or scientific courses. Hake (1998) compared student performance with interactive exercises and traditional lecture methods in introductory physics courses. He found that the students in the interactive courses showed considerably larger gains in conceptual knowledge, at both the secondary and post-secondary levels. Kumar & Lightner (2007) compared the use of classroom games in academia and corporate training systems. They found that games are more prevalent and widely accepted in corporate training centers. Student reaction to the introduction of a new game on the college classroom was positive, indicating that they learned a lot, the activity accomplished its goal, and a desire for more faculty to use such activities. In addition to aiding learning, games can be helpful in increasing retention. Though students may have achieved learning in particular context, they still need to practice the skill of abstracting what they know and applying it to industry practice (Alexander & Murphy, 1999).

In developing our game, we followed published advice for designing such activities (Garris et. al., 2002; Salies 2002). Specifically, we worked to match the difficulty of the game to the average undergraduate business student, providing both a clear statement of purpose for the game as well as rules by which to play it, and incorporate end-of-game debriefing to close the learning loop. The remainder of this paper presents a simple classroom game that teaches students the basic discrete-event simulation concepts and processes without requiring them to learn all the underlying math and scientific theory. It is interesting to note that the majority of classroom games in the literature take the general approach of using technology to mimic real-world, hands-on activities. In our game we reverse the approach, using a hands-on activity to simulate the use of technology. At the risk of sounding circuitous, we would describe our game as the simulation of a simulation. We will normally lead into the game with a lecture covering basic simulation concepts and terminology. The game is then typically played over two or three class periods, including post-game discussion. The discussion that follows assumes the reader is familiar with the science of discrete event simulation.

GAME STRUCTURE

The game is designed around Taco Casita, a fast-food operation in the student union. This system was chosen for a couple of reasons. First, the system's structure and process flow is familiar to all students, which eliminates the time needed to acquaint the students with the system. Secondly, the system has just the right amount of complexity to demonstrate multiple service activities, but is simple enough to simulate in a class activity.

The restaurant is a classic two-stage service operation that students everywhere will find familiar (an equivalent and ubiquitous example would be the Taco Bell chain of restaurants). The first stage involves order placement, payment, and drink service. With two registers, this first stage has a capacity to serve up to two customers at a time. After completing this stage, customers move to the second stage – which we call the Taco Assembly Line (TAL) – where their orders are prepared and served first come, first served (FCFS). Each stage has a queue. At stage 1, customers physically line up FCFS, while at stage 2 customers generally mill around until their order is served up FCFS, after which they leave the system. There are also several back-office activities that the customer can see, but does not interact with (e.g. ingredient preparation, utensil cleanup). A basic flowchart of the system is shown in Figure 1.

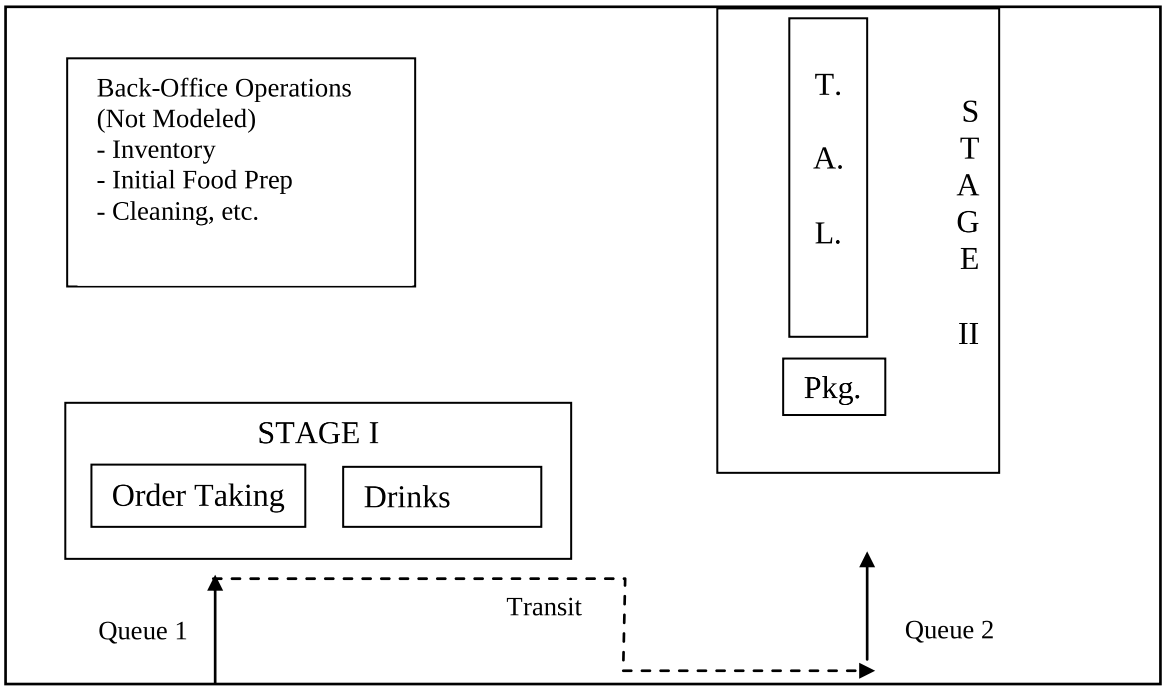


Figure 1: Simplified Process Flowchart

GAME METHODOLOGY

Prior to playing the game, we spend a class period discussing the system and how to model it for simulation purposes. In this activity, students work in small teams to sketch a process flowchart. In comparing and discussing students' work, the difference between logic flowcharts and process flowcharts is brought out, and the students are guided towards the flowchart shown in Figure 1. Several questions regarding modeling techniques are discussed, such as:

- Should the two order registers be modeled as two separate activities or one single activity? Does the drink station need to be modeled separately? Because the registers are identical, and a single server takes each customer's order and provides the drinks immediately thereafter, this is modeled as a single activity. While the real world system can have a capacity of two customers here (two cash registers), we simplify by modeling it as a single capacity activity.
- Once orders are placed, do orders need to be modeled as newly created entities that move through the TAL? This is what really happens, but there is a single TAL, and all orders are processed FCFS, in the same order that customers moved through Stage 1. Therefore, we can model just the customers, as though they are being served. This reinforces the idea of keeping things as simple as possible in building models while still accurately capturing the essence of the system.
- Do the back-office activities need to be modeled? Our interest in the system is with customer service measures, so for this activity we do not model them.

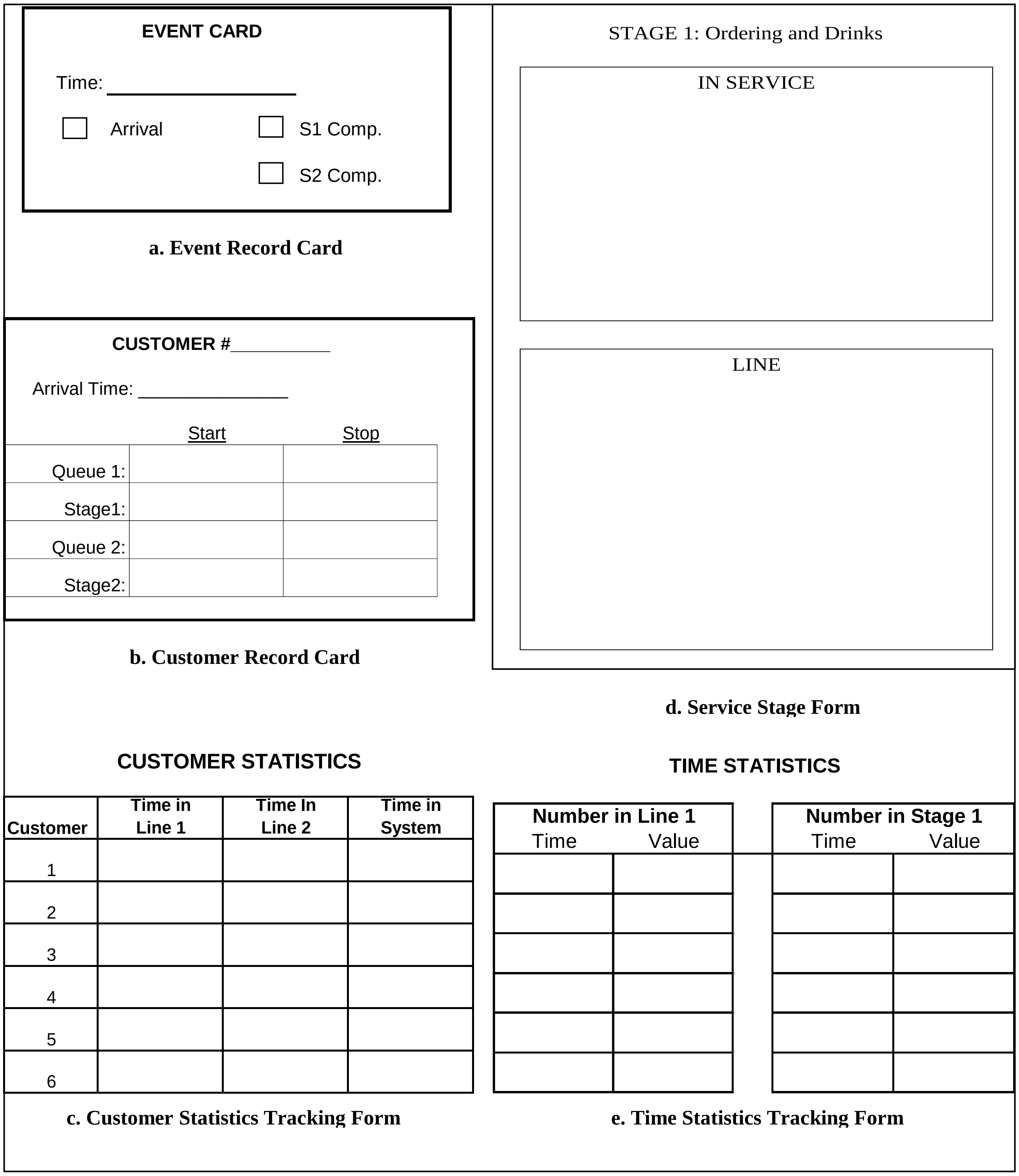
In the second class period, we introduce the game and get started playing. Game materials are decidedly low-tech, and include the following (see Figure 2 and Appendices A and B for samples):

- A handout with a summary of the system and modeling discussion from day 1. Also included are rules for playing the game – essentially an outline of the simulation logic (Appendix A).
- Sheets of paper (2) to represent each service stage and its queue (Figure 2d).
- Paper cards to represent customers, with spaces to record relevant attributes (Figure 2b).
- Paper cards for recording and storing events: customer arrivals and service completions (Figure 2a).
- Lists of random numbers to represent exponentially distributed inter-arrival and service times (Appendix B). The tables are generated in Excel. The default means are 80 seconds between arrivals and 50 seconds per service (these can be varied for learning purposes).
- Tables for recording customer statistics and time statistics (Figures 2c, 2e).

Various roles in the game are assigned to students, the specifics of which depend on the number of students in the class. Typically, students will be assigned the following roles:

- Event manager: generates events and maintains the event list. The event list is simply the collection of event cards that have been scheduled but have not occurred.
- Stage managers: generate new customer cards as needed at stage 1, and manage customers as they move through each stage, recording attributes on the customer card as needed.
- Statisticians: maintain tables of customer and time statistics.
- Time keeper: maintains the simulation clock and provides random times for events (pulled from list of random numbers that students are provided with).

The game itself begins with a brief discussion on initializing the model and the need to generate events to keep the simulation going. With the first arrival generated, we start the event processing loop that consumes the majority of game playing time. When each event is processed, students follow the appropriate event logic listed in the game handout (Exhibit A). As the events are processed, customer cards are moved through the simulated system, queuing up as needed. Relevant data for each customer is tracked directly on the customer card as events occur. Relevant time statistics are also updated at each event by recording changes in state variables, and the simulated time at which the changes occurred. For the first dozen or more events, we tightly manage the game to ensure the students are properly processing events and that everyone is performing their roles correctly. Once the students appear to have a good feel for how things should be happening, we let them run it on their own with distant guidance.



EVENT CARD

Time: ____________________

☐ Arrival ☐ S1 Comp.

☐ S2 Comp.

a. Event Record Card

CUSTOMER #__________

Arrival Time: _______________

	Start	Stop
Queue 1:		
Stage1:		
Queue 2:		
Stage2:		

b. Customer Record Card

STAGE 1: Ordering and Drinks

IN SERVICE

LINE

d. Service Stage Form

CUSTOMER STATISTICS

Customer	Time in Line 1	Time In Line 2	Time in System
1			
2			
3			
4			
5			
6			

c. Customer Statistics Tracking Form

TIME STATISTICS

Number in Line 1	
Time	Value

Number in Stage 1	
Time	Value

e. Time Statistics Tracking Form

Figure 2: Sample Game Materials

GAME TIMELINE

We typically dedicate three class periods to the entire exercise. Day 1 involves discussion of the real-world system, breaking it down in to relevant simulation terminology and developing the simplified process flowchart in Figure 1. On day 2, we introduce the game structure and processing rules, and begin playing the game. By the end of this day, the students have developed a good feel for the game rules, and we close with a mid-game discussion. Student questions and uncertainties about the game and its purpose will vary at this point, and the discussion is meant to clear up these issues before proceeding. On day 3, we pick up the game from where we left off on day 2, and continue playing for about ½ to ⅔ of the class period. Students typically run through the game quickly at this point, and by the end of the game have collected a reasonable amount of customer and time statistic data. We finish the day with further discussion of the exercise, comparing what they saw in the game with the real-world system, and explaining how the data captured would be used in a simulation project.

AN EXAMPLE

We now present an example of how the game would proceed for ten event processing iterations to better clarify how the game works. The procedures we will follow are in Appendix A. We will use the random numbers presented in Appendix B, an excerpt of which is shown in Table 1 for convenience.

Time Until Next Arrival (secs.)	Service Time Stage 1 (secs.)	Service Time Stage 2 (secs.)
11	72	76
101	87	50
8	201	63
12	21	152
140	38	71
24	26	76
97	55	32
77	96	145
46	121	62
143	109	48

Table 1. Example Random Times

Initialization

As shown in Appendix A, the first step in the game is to initialize the simulation. We initialize the system empty and idle following the instructions.

- Set clock to 0 seconds (Time-keeper simply records this time on a piece of paper)
- Record the number in line and in service at each stage as zero at the current time (zero) on the time statistics form (Figure 2e, Statistician(s) task).
- Generate and file the first arrival event, which gets the simulation going. The Event Manager would fill out an event card (Figure 2a), marking the event type as *Arrival* and the event time as the current time plus the first random arrival time from the random number list. The time of this event is 11 seconds (we will keep track of time in seconds for the sake of simplicity). That random number is now scratched off its list.

At the end of the initialization phase of the game, the state of the system is defined by Table 2.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
0	0	0	0

Table 2. System State After Initialization

There is one event in the event list: Customer #1 arrival at time 11. None of the statistics lists need to be updated.

Event Processing Loop

This loop comprises the vast majority of game time and is akin to running the simulation in a computer program. The general steps for processing an event are listed on the first page of Appendix A. The specific steps required for each type of event are listed on the second page of Appendix A. We will explain what happens in each iteration of the loop. Each iteration is akin to processing a single event in the simulation.

Iteration 1

The first step in each iteration is to pull the next event from the event list. As event cards are filed chronologically, the "next" event will be the first one in the stack of event cards. At this point, there is only one event in the event list (stack of event cards). This event is Customer #1 arrival at time 11, so the simulation clock (present time in the simulation) is advanced to time 11 (the Timekeeper updates this figure). We now process the event by going to the second page of Appendix A and performing the steps under *Customer Arrival*.

- Stage 1 Manager creates a new customer card (Figure 2b), marking it as Customer #1 and recording the arrival time as 11.
- Currently Stage 1 is idle (visual observation) so the Customer 1 card is placed at Stage 1, In Service (Figure 2d). The Stage 1 start time is recorded as the current time of 11 on the Customer 1 card. (The Queue 1 start and stop times can also be recorded as 11, or the students can simply enter dashes for those times as Customer 1 spends no time in Queue 1.)
- The Event Manager generates a new event card for Stage 1 service completion. That card is marked as *S1 Comp.* and the time recorded is the current clock time of 11 plus a random Stage 1 completion time from the random number list. The first appropriate random number is 72, so the time of the newly created S1 Comp. event is 83. The card is put into the event list (stack of cards). That first random number is then scratched off its list.
- The Event Manager generates a new card for the next customer arrival. The time of that event is the current time of 11 plus the next random number from the list of arrival times, which is 101. So the event card is marked as an *Arrival* with a time of 112. This card is filed in the event list. The time of 112 is later than the time of the only other event in the list, so this arrival event card is filed last in the list.
- The number of customers in Stage 1 service has changed from zero to one, so the Time Statistics list (Figure 2e) for the *Number in Stage 1* is updated with a new entry at the current time of 11 and a value of 1.

This concludes the processing of this event. At this point the state of the system is shown in Table 3.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	0	0	0

Table 3. System State After Iteration 1

Customer 1 is the only customer in the system. There are two events in the event list: Stage 1 Completion at time 83 and Customer Arrival at 112.

Iteration 2

We repeat the *Event Processing Loop* from the first page of Appendix A. The next event is a Stage 1 Service Completion at time 83, so the clock is advanced to time 83. We then follow the *Stage 1 Service Completion* logic from Appendix A.

- The Stage 1 stop time is recorded as the current time of 83 on the Customer #1 card.
- Customer 1 is moved to Stage 2, and since Stage 2 is idle, Customer 1 is put *In Service* at Stage 2. The time started Stage 2 for Customer 1 is recorded as the current time of 83. Dashes are entered for Queue 2 times on the Customer 1 card.
- The Time Statistic for *Number in Stage 2* is updated to a value of one at time 83.
- The Event Manager generates a new event for Stage 2 Completion based on the current clock plus a random service time. This event is scheduled at time 83 + 76 = 159. That event is filed in the event list. Since its time is later than the only other event in the list (Customer Arrival at time 112), this event is placed at the end of the event list.
- (Now we deal with Stage 1 because a customer has just left that stage.) There is no one in Queue 1, so Stage 1 now goes idle. The *Number in Stage 1* Time Statistic is now updated to a new value of zero at time 83.

This concludes the processing of this event. At this point the state of the system is shown in Table 4.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
0	0	1	0

Table 4. System State After Iteration 2

Customer 1 is the only customer in the system. There are two events in the event list: Customer Arrival at 112 and Stage 2 Completion at time 159.

Iteration 3

Again we repeat the *Event Processing Loop* from the first page of Appendix A. The next event is a Customer Arrival (#2) at time 112, so the simulation clock is advanced to time 112. As in Iteration 1, we now process the event by going to the second page of Appendix A and performing the steps under *Customer Arrival*.

- Stage 1 Manager creates a card for Customer 2 with the arrival time as 112.
- Currently Stage 1 is idle so the Customer 2 card is placed at Stage 1, In Service. The Stage 1 start time is recorded as the current time of 112 on the Customer 2 card. Dashes are entered for the Queue 1 start and stop times.
- The Event Manager generates a new event card for Stage 1 service completion with a time equal to the current clock of 112 plus a random service time: 112 + 87 = 199. The card is put in the correct place in the event list. That random number is then scratched off its list.
- The Event Manager generates a new event card for the next customer arrival. The time of that event is the current time of 112 plus the next random number from the list of arrival times, which is 8: 112 + 8 = 120. The event card is filed in the event list – since its time is earlier than the other cards in the list it is filed in the first position.
- The number of customers in Stage 1 service has changed from zero to one, so the Time Statistics list for the *Number in Stage 1* is updated with a new entry at the current time of 112 and a value of 1.

This concludes the processing of this event. At this point the state of the system is shown in Table 5.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	0	1	0

Table 5. System State After Iteration 3

Customers 1 and 2 are currently in the system. Both service stages are busy and both queues are still empty. There are three events in the event list: Customer Arrival at 120, Stage 2 Completion at time 159, and Stage 1 Completion at 199.

Iteration 4

The next event is a Customer Arrival (#3) at time 120, so the simulation clock is advanced to time 120 and we process the event as in Appendix A under *Customer Arrival*.

- Stage 1 Manager creates a card for Customer 3 with the arrival time as 120.
- Currently Stage 1 is busy so the Customer 3 card is placed at Stage 1, In Queue. The Queue 1 start time is recorded as the current time of 120 on the Customer 3 card. No other entries are made at this time.
- The Event Manager generates a new event card for the next customer arrival. The time of that event is the current time of 120 plus the next random number from the list of arrival times, which is 12: 120 + 12 = 132. The event card is filed in the event list – since its time is earlier than the other cards in the list it is filed in the first position.
- The number of customers in Stage 1 queue has changed from zero to one, so the Time Statistics list for the *Number in Queue 1* is updated with a new entry at the current time of 120 and a value of 1.

This concludes the processing of this event. At this point the state of the system is shown in Table 6.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	1	1	0

Table 6. **System State After Iteration 4**

Customers 1, 2, and 3 are currently in the system. Both service stages are busy, the stage 1 queue has one customer, and the stage 2 queue is still empty. There are three events in the event list: Customer Arrival at 132, Stage 2 Completion at time 159, and Stage 1 Completion at 199.

Iteration 5

The next event is a Customer Arrival (#4) at time 132, so the simulation clock is advanced to time 132 and we process the event as in Appendix A under *Customer Arrival*.

- Stage 1 Manager creates a card for Customer 4 with the arrival time as 132.
- Currently Stage 1 is busy so the Customer 4 card is placed at Stage 1, In Queue behind Customer 3. The Queue 1 start time is recorded as the current time of 132 on the Customer 3 card. No other entries are made at this time.
- The Event Manager generates a new event card for the next customer arrival. The time of that event is the current time of 132 plus the next random number from the list of arrival times, which is 140: 132 + 140 = 272. The event card is filed in the event list – since its time is later than the other cards in the list it is filed in the last position.
- The number of customers in Stage 1 queue has changed from one to two, so the Time Statistics list for the *Number in Queue 1* is updated with a new entry at the current time of 132 and a value of 2.

This concludes the processing of this event. At this point the state of the system is shown in Table 7.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	2	1	0

Table 7. **System State After Iteration 5**

Customers 1 - 4 are currently in the system. Both service stages are busy, the stage 1 queue has two customers, and the stage 2 queue is still empty. There are three events in the event list: Stage 2 Completion at time 159, Stage 1 Completion at 199, and Customer Arrival at 272.

Iteration 6

The next event is a Stage 2 Completion at time 159, so the simulation clock is advanced to time 159 and we process the event as in Appendix A under *Stage 2 Service Completion*.

- Stage 2 Manager records the Stage 2 Stop time for Customer 1 as the current time of 159.
- Since this customer is now complete, we record statistics for this customer on the *Customer Statistics* form (Figure 2c). Time in Queue 1 and Queue 2 is recorded as zero for Customer 1 (generically, the time in queue is the difference between the stop and start times in that queue). Time in System for Customer 1 is recorded as the difference between the Stage 2 Stop time and the Customer Arrival time: 159 – 0 = 159.
- No one is in Queue 2, so Stage 2 Service goes idle, and the Time Statistic for *Number in Stage 2* is updated with a value of 0 at time 159.
  This concludes the processing of this event. At this point the state of the system is shown in Table 8.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	2	0	0

Table 8. System State After Iteration 6

Customers 2 - 4 are currently in the system. Stage 1 Service is busy, Stage 2 Service is idle, the stage 1 queue has two customers, and the stage 2 queue is still empty. There are two events in the event list: Stage 1 Completion at 199, and Customer Arrival at 272.

Iteration 7

The next event is a Stage 1 Completion at time 199, so the simulation clock is advanced to time 199 and we process the event as in Appendix A under *Stage 1 Service Completion*.

- Stage 1 Manager records the Stage 1 Stop time for Customer 2 as the current time of 199.
- Customer 2 is moved to Stage 2. Stage 2 is idle, so Customer 2 is put *In Service* at Stage 2. The time started Stage 2 for Customer 2 is recorded as the current time of 199. Dashes are entered for Queue 2 times on the Customer 1 card.
- The Time Statistic for *Number in Stage 2* is updated to a value of 1 at time 199.
- The Event Manager generates a new event for Stage 2 Completion based on the current clock plus a random service time. This event is scheduled at time 199 + 50 = 249. That event is filed second in the event list (between the two events already in the list).
- (Now we deal with Stage 1 because a customer has just left that stage.) There are customers in Queue 1, so we move the next customer (#3) into service at Stage 1. We mark the Queue 1 Stop time and Stage 1 Start time for Customer 3 as the current time of 199. The *Number in Queue 1* Time Statistic is now updated to a new value of one at time 199
- The Event Manager generates a new event card for Stage 1 Service Completion. The time of that event is 199 + 201 = 400.

This concludes the processing of this event. At this point the state of the system is shown in Table 9.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	1	1	0

Table 9. System State After Iteration 7

Customers 2 - 4 are currently in the system. Stage 1 Service and Stage 2 Service are busy, the stage 1 queue has one customer, and the stage 2 queue is still empty. There are three events in the event list: Stage 2 Completion at 249, Customer Arrival at 272, and Stage 1 Service Completion at 400.

Iteration 8

The next event is a Stage 2 Completion at time 249, so the simulation clock is advanced to time 249 and we process the event as in Appendix A under *Stage 2 Service Completion*.

- Stage 2 Manager records the Stage 2 Stop time for Customer 2 as the current time of 249.
- Since this customer is now complete, we record statistics for this customer on the *Customer Statistics* form (Figure 2c). Time in Queue 1 and Queue 2 is recorded as zero for Customer 2. Time in System for Customer 1 is recorded as the difference between the Stage 2 Stop time and the Customer Arrival time: 249 – 112 = 137.
- No one is in Queue 2, so Stage 2 Service goes idle, and the Time Statistic for *Number in Stage 2* is updated with a value of 0 at time 249.

This concludes the processing of this event. At this point the state of the system is shown in Table 10.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	1	0	0

Table 10. System State After Iteration 8

Customers 3 - 4 are currently in the system. Stage 1 Service is busy, Stage 2 Service is idle, the stage 1 queue has 1 customer, and the stage 2 queue is empty. There are two events in the event list: Customer Arrival at 272 and Stage 1 Completion at 400.

Iteration 9

The next event is a Customer Arrival (#5) at time 272, so the simulation clock is advanced to time 272 and we process the event as in Appendix A under *Customer Arrival*.

- Stage 1 Manager creates a card for Customer 5 with the arrival time as 272.
- Currently Stage 1 is busy so the Customer 5 card is placed at Stage 1, In Queue behind Customer 4. The Queue 1 start time is recorded as the current time of 272 on the Customer 5 card. No other entries are made at this time.
- The Event Manager generates a new event card for the next customer arrival. The time of that event is 272 + 24 = 296. The event card is filed first in the event list.
- The number of customers in Stage 1 queue has changed from 1 to 2, so the Time Statistics list for the *Number in Queue 1* is updated with a new entry at the current time of 272 and a value of 2.

This concludes the processing of this event. At this point the state of the system is shown in Table 11.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	2	0	0

Table 11. **System State After Iteration 9**

Customers 3 - 5 are currently in the system. Stage 1 Service is busy, Stage 2 Service is idle, the stage 1 queue has 2 customers, and the stage 2 queue is empty. There are two events in the event list: Customer Arrival at 296 and Stage 1 Completion at 400.

Iteration 10

The next event is a Customer Arrival (#6) at time 296, so the simulation clock is advanced to time 296 and we process the event as in Appendix A under *Customer Arrival*.

- Stage 1 Manager creates a card for Customer 6 with the arrival time as 296.
- Currently Stage 1 is busy so the Customer 6 card is placed at Stage 1, In Queue behind Customer 5. The Queue 1 start time is recorded as the current time of 296 on the Customer 6 card. No other entries are made at this time.
- The Event Manager generates a new event card for the next customer arrival. The time of that event is 296 + 97 = 393. The event card is filed first in the event list.
- The number of customers in Stage 1 queue has changed from 2 to 3, so the Time Statistics list for the *Number in Queue 1* is updated with a new entry at the current time of 296 and a value of 3.

This concludes the processing of this event. At this point the state of the system is shown in Table 12.

Number in Stage 1	Number in Queue 1	Number in Stage 2	Number in Queue 2
1	3	0	0

Table 12. **System State After Iteration 10**

Customers 3 - 6 are currently in the system. Stage 1 Service is busy, Stage 2 Service is idle, the stage 1 queue has 3 customers, and the stage 2 queue is empty. There are two events in the event list: Customer Arrival at 393 and Stage 1 Completion at 400.

VARIATIONS ON THE GAME

By using a familiar service operation as the basis for the game, the entire exercise can take place within the classroom. We have on occasion tried to bring the game out of the classroom, bringing the students to the student union for discussion of the system, or playing of the actual game. We have met with varied success when we have tried this, as students can very easily be distracted by the various sights and sounds of the food court.

We typically use the game in classes of 6-10 students. With larger classes, you have at least a couple options to expand the game. With a dozen or more students, two or more teams can be established to play the game simultaneously. In this case, we recommend you provide different random number sets to the teams, and compare statistics after the game. This can demonstrate that simulation output measures themselves are random numbers, and lead to a discussion of the need for replications in a simulation project, and their relationship with confidence intervals. Another option with large classes is to let students take the place of simulated customers themselves. This will add some realism to the game structure, and should make the game more fun and interesting for the entire class. We have not had the opportunity to test this particular option however.

SUMMARY

Discrete-event simulation concepts, while quite logical, can be difficult to teach to undergraduate business students. We quickly learned that lecture simply does not cut it, though it can be useful in introducing some basic terminology and concepts. Feedback from students during the game's final discussion session has been generally positive. They express that they like the hands-on experience, and that the game is broken up across class sessions so we can address initial uncertainties they have. Not everyone gets the same meaning and knowledge out of the game, but the majority of students have commented that the game gives them a much better understanding of the basic concepts and terminology from the introductory lecture. This feedback is reinforced by student comments on end-of-course evaluations referencing the exercise.

EXHIBIT A

TACO CASITA GAME: Logic Handout

Simulation Logic

Initialization:

- Set clock to zero, or desired starting time
- Initialize any variables to the desired starting state of the system. Often idle and empty.
- Generate and file (schedule) initial event(s) needed to get simulation going
- Start Event Processing Loop

Event Processing Loop

- Get next scheduled event information
- Advance simulation clock to time of event
- Process the event
  - Steps based on logic related to real system
  - Modify variables/attributes, update time/customer stats as needed
  - Generate and file resultant future events
  - Discard the current event card
- Repeat the loop

Termination

- Simulation ends at some predetermined point
  - Some point in simulated time, or
  - Some number of entities processed
- Statistics updated as needed
- Desired performance measures computed
- Output analyzed

Event Processing

Customer Arrival

- Record arrival time
- If Stage 1 busy:
  - Put customer in Line 1
  - Record time started
- If Stage 1 idle:
  - Place customer in Stage 1 service
  - Record time started
  - Generate and file event card for Stage 1 service completion
- Generate event card for next arrival
- Update appropriate time stats

Stage 1 Service Completion

- Record Stage 1 end time on customer card
- Move customer to Stage 2
  - If Stage 2 busy:
    - Put customer in Queue 2
    - Record time started
    - Update number in Queue 2 statistic
  - If Stage 2 idle:
    - Place customer in Stage 2 service
    - Record time started
    - Update number in stage 2 service statistic
    - Generate and file event card for Stage 2 service completion
- If anyone in Line 1:
  - Move lead customer into service
  - Record customer’s Queue 1 end and Stage 1 start times
  - Update Number In Line 1 Statistic
  - Generate and file event card for Stage 1 Service Completion
- If nobody in Line 1:
  - Update number in Stage 1 service statistic

Stage 2 Service Completion

- Record Stage 2 end time on customer card
- Customer leaves system, record customer stats
- If anyone in Queue 2:
  - Move lead customer into service
  - Record customer’s Queue 2 end and Stage 2 start times
  - Generate and file event card for Stage 2 service completion
  - Update Number in Queue 2 statistic
- If nobody in Queue 2
  - Update Number in Stage 2 Service statistic

EXHIBIT B

TACO CASITA GAME: Random Times

Time Between Arrivals			Stage 1 Service			Stage 2 Service	
11	53		72	43		76	89
101	61		87	21		50	213
8	23		201	164		63	49
12	96		21	53		152	53
140	6		38	43		71	79
24	25		26	40		76	52
97	75		55	66		32	44
77	21		96	23		145	65
46	93		121	73		62	275
143	78		109	58		48	74
24	81		59	43		39	83
166	0		26	57		69	37
43	49		31	58		32	49
137	142		62	44		72	33
118	113		27	26		44	71
76	155		33	48		54	42
81	1		34	42		52	42
32	223		64	23		108	126
139	116		211	21		31	142
129	33		36	22		169	38
119	37		37	105		92	54
272	9		21	43		67	172
70	154		44	122		162	105
27	56		63	32		223	123
135	97		68	26		48	31
9	37		45	39		30	83
111	186		73	25		116	96
69	22		61	33		36	39
12	4		50	113		77	85
38	133		100	119		96	77
106	13		72	62		138	77
12	181		93	27		107	128
29	19		43	25		39	68
54	133		22	24		30	37
110	217		124	82		220	64
25	45		50	33		73	84
259	0		21	201		55	173
335	24		38	128		150	70
21	75		167	167		54	64
101	336		54	48		164	37
5	206		31	44		51	40
20	38		56	21		95	120
25	36		42	39		35	52

REFERENCES

1. Alexander, P. A. and P. K. Murphy (1999). Nurturing the seeds of transfer: A domain-specific perspective, *International Journal of Educational Research*, 31, 561-576.
2. Cruickshank, D. R. and R. Telfer (2001). Classroom games and simulations, *Theory into Practice*, 19:1, 75-80.
3. Garris, R., R. Ahlers & J. E. Driskell (2002). Games, motivation and learning: Simulation and gaming, *An Interdisciplinary Journal of Theory, Practice and Research*, 33:4, 441-467.
4. Hake, R. R. (1998). Interactive-engagement vs. traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses, *American Journal of Physics*, 66:1, 64-74.
5. Kumar, R. and R. Lightner (2007). Games as an interactive classroom technique: Perceptions of corporate trainers, college instructors and students, *International Journal of Teaching and Learning in Higher Education*, 19:1, 53-63.
6. Nemerov, L.G. (1996). Do classroom games improve motivation and learning? *Teaching and Change*, 3:4, 356-356.
7. Reuben, B. D. (1999). Simulations, games, and experience-based learning: the quest for a new paradigm for teaching and learning, *Simulation & Gaming*, 30:4, 498-505.
8. Salies, T. G. (2002). Simulations/gaming in the EAP writing class: Benefits and drawbacks, *Simulations & Gaming*, 33:3, 316-329.

NOTES