Discrete-Event Simulation of Queues with Spreadsheets: A Teaching Case

Article *in* Proceedings - Winter Simulation Conference · December 2006

DOI: 10.1109/WSC.2006.323053 · Source: DBLP

2 authors:

Marco Mesquita
University of São Paulo
30 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Alvaro Hernandez
University of São Paulo
1 PUBLICATION **7** CITATIONS
SEE PROFILE

Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

DISCRETE-EVENT SIMULATION OF QUEUES WITH SPREADSHEETS: A TEACHING CASE

Marco Aurélio de Mesquita

Dept. of Production Engineering
Escola Politécnica - Universidade de São Paulo
Av. Prof. Almeida Prado, 531
05508-900, São Paulo, SP, BRAZIL

Alvaro Euzebio Hernandez

Dept. of Production Engineering
Escola Politécnica - Universidade de São Paulo
Av. Prof. Almeida Prado, 531
05508-900, São Paulo, SP, BRAZIL

ABSTRACT

This paper describes the use of spreadsheets combined with simple VBA code as a tool for teaching queuing theory and discrete-event simulation. Four different cases are considered: single server, parallel servers, tandem queuing, and closed queuing system. The data obtained in the simulation run are conveniently stored in spreadsheets for subsequent statistical analysis. This approach was successfully deployed in a second one-semester course on management science for industrial engineers undergraduate students.

1 INTRODUCTION

Two common topics in a survey course on management science are queuing theory and discrete-event-simulation. Queuing theory generally refers to the development and implementation of analytical, closed-form models of waiting lines. Discrete-event simulation is usually taught by means of some dedicated simulation software.

Both approaches have been criticized. The former would be very mathematical and "closed-form models would not be able to analyze most of the complex systems that are encountered in practice" (Banks 1998). The latter would be prone to emphasize specific simulation software features instead of discrete-event simulation core concepts (Schriber and Brunner 1998).

In order to improve the learning experience on these subjects, discrete-event simulation of four simple queuing systems were deployed through the combined use of native spreadsheet functions and simple VBA code.

This paper presents the results of such process in the following order:

- Background.
- Multi-server queuing model.
- Two-Stage tandem queuing model.
- Cyclic queuing model.
- Discussion of the teaching experience.
- Conclusion.

2 BACKGROUND

Centeno (1996) explains the two main approaches for discrete simulation (event-driven and process driven): "Under event driven discrete simulation, the modeler has to think in terms of the events that may change the status of the system to describe the model. […] On the other hand, under the process driven approach, the modeler thinks in terms of the processes that the dynamic entity will experience as it moves through the system".

Several authors have written about the use of the process-driven approach based on spreadsheets to teach queuing theory (Grossman 1999, Evans 2000, Ingolfsson and Grossman 2002, and Hora 2003).

Banks (1998), employing a process-driven approach, generates a table, called "*ad hoc simulation* table", containing the relevant information about each customer in an single-server queuing system (e.g. arrival time, waiting time in queue, and total time in the system). Nonetheless, if one is interested in analyzing other variables, such as the total number of customers in the system at a given time, the "*ad hoc simulation* table" alone does not suffice. In such cases, a more elaborated table has to be constructed, placing the arrival and departure times in their proper chronological order, such as the example provided by Pritsker (1998), which is referred to as an "*event calendar*".

Chase et al. (1998) and Grossman (1999), also using process-driven approaches, implement queuing systems using only native spreadsheet functions, obtaining the respective "*ad hoc simulation*" tables. The former simulates tandem queues, the latter a multi-server queuing system. Ingolfsson and Grossman (2002), building on the previous work of Grossman, provides graphical interface in addition to basic tables, displaying the customers and servers statuses. An interesting feature of such approach is the possibility of replicating the simulation through the recalculating feature of MS Excel and observing the resulting changes. The graphical interface obtained in their implementation, in conjunction with the recalculating function,

does provide "a richer understanding of queues", as stated by these authors.

On the other hand, an instance of the event-driven approach to simulate single queuing systems can be found in Winston (2004). Albright (2001) presents a discrete-event simulator of multi-server queuing systems through the use of spreadsheet as interfaces by which the analyst can easily input system parameters and read system performance measures. In such implementation, the simulation and evaluation of performance measures are undertaken by pure VBA programming, which increases the complexity of the code.

We felt that a proper understanding of queuing systems and an even deeper understanding of discrete-event simulation could be attained if an event-driven simulation of queuing systems were implemented through some easy VBA programming in conjunction with spreadsheet functions, mainly when compared to the more complex logic of the spreadsheet modeling proposed by Ingolfsson and Grossman (2002).

3 MULTI-SERVER QUEUING MODEL

In order to model a multi-server queuing system, one has to take into consideration two main events: the arrival and the departure of each customer. The basic logic involved is to check which event occurs first as shown bellow. The simulation runs until the following conditions are both satisfied: the closing time is reached and there are no customers left in the system.

```
Sub MMc()
  Call Initialize
  Do
    Call FindNextEvent(Type, Server)
    Select Case Type
      Case 1
        Call Arrival
      Case 2
        Call Departure(Server)
    End Select
  Loop Until ClockTime > CloseTime
           And NumInSystem = 0
End Sub
```

The initialization process includes reading the parameters set-up (Table 1) and the generation of the first event (i.e., the arrival time of the first customer). The procedure *FindNextEvent* determines which event will occur next: either type 1, for an arrival, or type 2, for a departure. In the case of a departure, the server from which it takes place is also determined.

Table 1: Input Data

Arrival rate (customers per hour)	25
Mean service time (min)	5
Number of servers	3

Upon an arrival (during office hours), the customer either joins the queue or seizes the server, depending on server availability. In case there is more than one idle server, the customer picks one of them randomly. The last step is to define the next arrival time. The code for the arrival procedure is shown bellow.

```
Sub Arrival()
    ClockTime = NextArrival
    NumInSystem = NumInSystem + 1
    If NextArrival > CloseTime Then
        NextArrival = Infinite
        Exit Sub
    End If
    If NumInSystem > NumServer Then
        NumInQueue = NumInQueue + 1
    Else
        Server = ChooseServerIdle
        ServerBusy(Server) = True
        NextDeparture = ClockTime + INVEXP(Mu)
    End If
    NextArrival = ClockTime + INVEXP(Lambda)
End Sub
```

After a service completion, there are two possibilities: either the line is empty, in this case the server becomes idle, or there is at least one customer in line, in such case the customer who arrived first is picked up by the server.

```
Sub Departure(Server)
    ClockTime = NextDeparture(Server)
    NumInSystem = NumInSystem - 1
    If NumInQueue = 0 Then
        ServerBusy(Server) = False
        NextDeparture = Infinite
    Else
        ServerBusy(Server) = True
        NumInQueue = NumInQueue - 1
        NextDeparture = ClockTime + INVEXP(Mu)
    End If
End Sub
```

Despite the fact that in the current implementation the inter-arrival and service time are modeled by exponential random variates (which are easily generated), other distributions may be used.

The values taken on by the state-variables during the simulation run are written directly to the spreadsheets. There are three main tables: event table, customer table, and server table. These tables allow further statistical analysis through spreadsheet functions.

Although any table alone completely describes the simulation run, the way the information is displayed allows different views of the queuing phenomenon. Table 2 shows the event table for a given run. Besides the event calendar, it shows the main state-variables. Its purpose is to highlight the underlying discrete-event logic. Table 3 displays what happens to each customer (the usual output of the process-driven approach). Table 4 gives information about the server status throughout the simulation run (due to column width limitations, only data regarding one server is shown).

Table 2: Events

Time	Event	N(t)	Next Arrival	Next Departure		
				Serv. 1	Serv. 2	Serv. 3
9:00:00	Open	0	9:02:21	0:00:00	0:00:00	0:00:00
9:02:21	0	1	9:08:32	0:00:00	9:02:43	0:00:00
9:02:43	2	0	9:08:32	0:00:00	0:00:00	0:00:00
9:08:32	0	1	9:14:21	0:00:00	9:15:45	0:00:00
9:14:21	0	2	9:14:31	0:00:00	9:15:45	9:18:01
9:14:31	0	3	9:17:46	9:16:09	9:15:45	9:18:01
9:15:45	2	2	9:17:46	9:16:09	0:00:00	9:18:01
9:16:09	1	1	9:17:46	0:00:00	0:00:00	9:18:01
9:17:46	0	2	9:23:14	0:00:00	9:21:20	9:18:01
9:18:01	3	1	9:23:14	0:00:00	9:21:20	0:00:00
9:21:20	2	0	9:23:14	0:00:00	0:00:00	0:00:00

Key for event column: 0 - arrival; i – departure from server i (i > 0).

Table 3: Customers

Customer	Arrival Time	Service Starting Time	Server	Service Finishing Time
Open	9:00:00	---	---	---
1	9:02:21	9:02:21	2	9:02:43
2	9:08:32	9:08:32	2	9:15:45
3	9:14:21	9:14:21	3	9:18:01
4	9:14:31	9:14:31	1	9:16:09
5	9:17:46	9:17:46	2	9:21:20
6	9:23:14	9:23:14	3	9:26:03
7	9:23:40	9:23:40	1	9:29:56
8	9:27:13	9:27:13	2	9:28:40
9	9:27:27	9:27:27	3	9:34:06
10	9:28:09	9:28:40	2	9:31:37

Table 4: Server 1

Time	Status	Customer
9:00:00	Idle	---
9:14:31	Busy	4
9:16:09	Idle	---
9:23:40	Busy	7
9:29:56	Busy	11
9:35:30	Busy	15
9:38:52	Idle	---
9:42:52	Busy	19
9:43:58	Idle	---
9:44:38	Busy	22
9:54:33	Idle	---

Data from Table 2 and 3 are shown as Gantt charts in Figure 1 and 2 respectively. Usual performance measures (e.g. average time in system, percent idle time, and maximum queue length) can be easily calculated from these tables with the statistical spreadsheet functions.

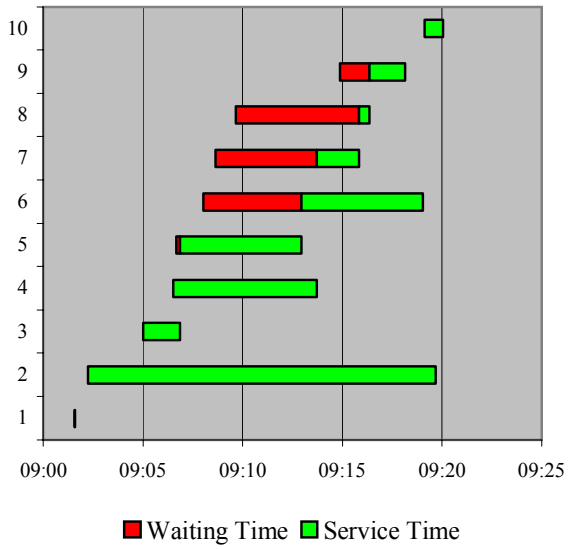

Figure 1: First Ten Customers Waiting and Service Times

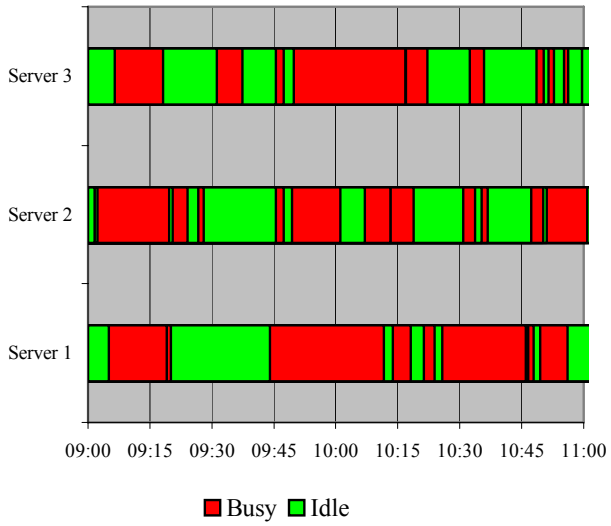

Figure 2: Status of Servers

In addition to the previous charts, it is possible and interesting to show the students how the number of customers in the system evolves over time, as exhibited in Figure 3.

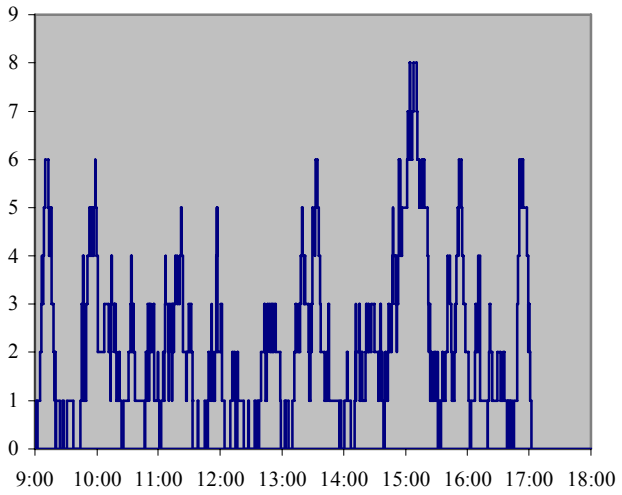

Figure 3: Number of Customers in the Queuing System

4 TANDEM SINGLE-SERVER QUEUING MODEL

The tandem queuing model presented in this section consists of two single servers. The main code below is very similar to the previous one. The arrival process is basically the same. However, the departure process should be handled differently on each server.

```
Sub TandemMM1()
  Call Initialize
  Do
    Call FindNextEvent(Type)
    Select Case Type
      Case 1
        Call Arrival
      Case 2
        Call DepartureFromServer1
      Case 3
        Call DepartureFromServer2
    End Select
  Loop Until ClockTime > CloseTime
           And NumInSystem = 0
End Sub
```

The departure process from server one and from server two are given next. The customer leaving server one goes to the second server. At this time, he can either joins the queue or seizes server 2 in case it is idle. At the same time, server one begins servicing a customer waiting in line or, if the line is empty, remains idle until a new customer arrives in the system. The departure process at server two is similar to the case of the previous multi-server model.

```
Sub DepartureFromServer_1
    ClockTime = NextDeparture(1)
    NumInStation(1) = NumInStation(1) - 1
    NumInStation(2) = NumInStation(2) + 1
    If ServerBusy(2) Then
        NumInQueue(2) = NumInQueue(2) + 1
    Else
        ServerBusy(2) = True
        NextDeparture(2) = ClockTime +
                                  INVEXP(Mu2)
    If NumInQueue(1) = 0 Then
        ServerBusy(1) = False
        NextDeparture(1) = Infinite
    Else
        NumInQueue(1) = NumInQueue(1) - 1
        ServerBusy(1) = True
        NextDeparture(1) = ClockTime +
                                  INVEXP(Mu1)
  End If
End Sub

Sub DepartureFromServer_2
    ClockTime = NextDeparture(2)
    NumInStation(2) = NumInStation(2) - 1
    If NumInQueue(2) = 0 Then
        ServerBusy(2) = False
        NextDeparture(2) = Infinite
    Else
        NumInQueue(2) = NumInQueue(2) - 1
        ServerBusy(2) = True
        NextDeparture(2) = ClockTime +
                                  INVEXP(Mu2)
    End If
End Sub
```

Tables 7, 8, and 9 are event, customer and server tables similar to those obtained in the previous model. Figure 4 presents the waiting and service time of the first ten customers on each server.

This model can be easily extended to any given number of servers. In order to accomplish such extension, one has to add intermediate servers, which will be all handled in the same way, but differently from the first and the last server in the tandem system.

Table 5: Events

Time	Event	$N_1(t)$	$N_2(t)$	Next Arrival	Next Departure	
					Serv. 1	Serv. 2
9:00:00	Open	0	0	9:06:00	0:00:00	0:00:00
9:06:00	0	1	0	9:19:10	9:08:46	0:00:00
9:08:46	1	0	1	9:19:10	0:00:00	9:10:53
9:10:53	2	0	0	9:19:10	0:00:00	0:00:00
9:19:10	0	1	0	9:26:24	9:27:13	0:00:00
9:26:24	0	2	0	9:47:39	9:27:13	0:00:00
9:27:13	1	1	1	9:47:39	9:32:26	9:31:52
9:31:52	2	1	0	9:47:39	9:32:26	0:00:00
9:32:26	1	0	1	9:47:39	0:00:00	9:34:22
9:34:22	2	0	0	9:47:39	0:00:00	0:00:00
9:47:39	0	1	0	9:52:50	9:51:11	0:00:00

Table 6: Customers

Customer	Arrival	Server 1		Server 2	
		Start	Finish	Start	Finish
Open	9:00:00	---	---	---	---
1	9:06:00	9:06:00	9:08:46	9:08:46	9:10:53
2	9:19:10	9:19:10	9:27:13	9:27:13	9:31:52
3	9:26:24	9:27:13	9:32:26	9:32:26	9:34:22
4	9:47:39	9:47:39	9:51:11	9:51:11	9:56:23
5	9:52:50	9:52:50	9:53:28	9:56:23	9:59:03
6	9:58:49	9:58:49	10:01:26	10:01:26	10:01:40
7	9:59:37	10:01:26	10:03:14	10:03:14	10:10:53
8	10:07:33	10:07:33	10:11:22	10:11:22	10:11:23
9	10:25:37	10:25:37	10:35:41	10:35:41	10:35:41
10	10:28:43	10:35:41	10:56:02	10:56:02	11:01:35

Table 7: Servers

Server 1		Server 2	
Time	Status	Time	Status
9:00:00	Idle	9:00:00	Idle
9:06:00	Busy	9:08:46	Busy
9:08:46	Idle	9:10:53	Idle
9:19:10	Busy	9:27:13	Busy
9:27:13	Busy	9:31:52	Idle
9:32:26	Idle	9:32:26	Busy
9:47:39	Busy	9:34:22	Idle
9:51:11	Idle	9:51:11	Busy
9:52:50	Busy	9:56:23	Busy
9:53:28	Idle	9:59:03	Idle
9:58:49	Busy	10:01:26	Busy
10:01:26	Idle	10:01:40	Idle

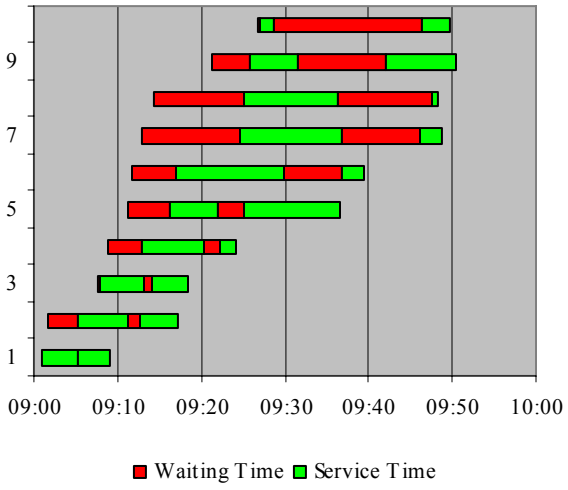

Figure 4: First Ten Customers Waiting and Service Time

5 CYCLIC QUEUING MODEL

Building up from the model presented in section 4, it is possible to model a closed queuing system in which a constant number of customers remains indefinitely in the system, moving sequentially from one station to the next and back to the first again (Kleinrock 1975).

Such model can be used to analyze *pull production systems* such as the CONWIP (Constant Work-in-Process). In this system, jobs flow sequentially through *m* working stations. Upon process completion at station *m*, the job exits the system and triggers a new job release at station one (Spearman et al. 1990).

A four-station assembly line, with identical patterns of process times (deterministic, normal and exponential distributions) and different numbers of jobs (or work-in-process level), was simulated.

Initially, it was considered a *balanced* line (i.e., all stations with the same average process time of 5 minutes). Figure 5 shows the throughput of the system (average number of jobs exiting the system per hour) as a function of the work-in-process level. One can easily grasp from this chart the corrupting influence of the variability of the process time over the throughput.

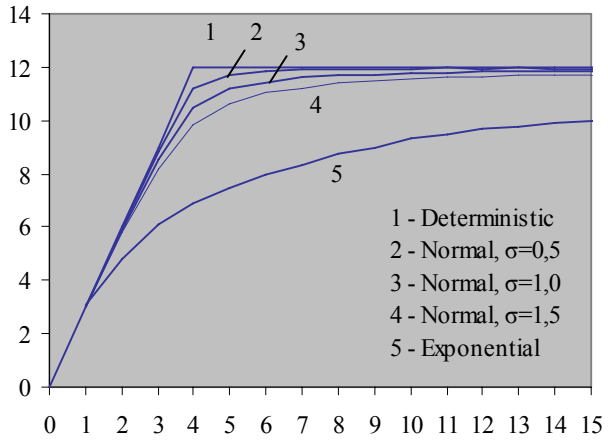

Figure 5: Throughput vs. WIP (m=4, μ=5min)

Figure 6 shows that the average lead time (i.e. the time it takes for the job to cross the system) increases as a function of the work-in-process level. Once again, the worst case takes place when the process time is exponentially distributed.

When simulating the case of *unbalanced* lines, with average process times of 5, 6, 5, and 4 minutes, the maximum throughput drops from 12 to 10 jobs/hour, which corresponds to the expected throughput of the second station (the bottleneck).

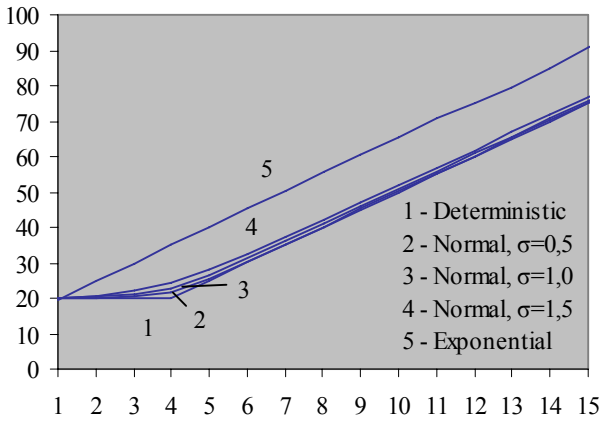

Figure 6:Lead Time vs. WIP (m=4, μ=5 min)

6 TEACHING EXPERIENCE

The authors teach the second one-semester management science survey course (out of two) in a production engineering curriculum comprising probabilistic modeling and discrete-event simulation.

The first third of the course is used to teach classic concepts on Markov chain and on queuing theory using regular text books like Winston (2004). The second third of the course is dedicated to teaching discrete-event simulation. The last part of the course comprises dynamic programming and decision analysis.

The discrete-event simulation module begins with the simulation of the classic M/M/1. The very first approach adopted is a conceptual, by hand process-driven simulation (the so-called *"ad hoc* simulation"). In the next step, the students are required to simulate the same M/M/1, this time using an event-driven approach, as suggested by Winston (2004), and implemented on an Excel/VBA workbook provided by the instructors.

Following these activities, the students are required to develop and implement the multi-server and tandem queuing models.

An interesting result is achieved when the exponential service time distribution is replaced by other types of distributions. For instance, through the use of normal distributions, the effect of the service time variance on performance measures can be easily shown.

The tandem queuing model, presented in section 4, is useful to show the effect of unbalanced workstations on customers waiting time.

By the same token, queuing networks are introduced. An instance of a cyclic queuing model is studied. The results achieved by its simulation (throughput and the cycle time) can be used to better analyze and design the work-in-process level.

The discrete-event simulation module is concluded with the introduction of a professional simulation software. This software is used in tutorial lectures where the students have the opportunity to implement the same models studied before. After that, more complex models using such software are demonstrated. Such experience reveals the great potential of the simulation technique in the analysis of real-world productive systems.

7 CONCLUSIONS

The widespread use of spreadsheets programs such as Microsoft Excel, either as a didactical or as a professional tool in the management science, is an indisputable fact. However, most authors, when referring to spreadsheet modeling, limit themselves to native spreadsheet functions, ruling out the implementation of macros through the use of some programming language. Nonetheless, we found out that it is to the students' advantage to blend simple VBA code (to handle the simulation itself) with spreadsheets native functions (for output data analysis).

In view of our previous teaching experience, the approach proposed in this paper allows not only a better understanding of queuing theory but also a better introduction to discrete-event simulation.

REFERENCES

Albright, S.C. 2001. *VBA for Modelers.* Pacific Grove: Duxbury.

Banks, J. 1998. Principles of Simulation. In *Handbook of Simulation:* principles, methodology, advances, applications, and practice. Ed. J. Banks. New York: John Wiley.

Centeno, M.A. 1996. An Introduction to Simulation Modeling. In *Proceedings of the 1996 Winter Simulation Conference,* ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, 15-22. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Chase, R.B., N. J. Aquilano, F. R. Jacobs. 1998. *Production and Operations Management: manufacturing and services*. 8th ed. Boston: Irwin/McGraw-Hill.

Evans, J.R. 2000. Spreadsheets as a Tool for Teaching Simulation. Informs Transactions on Educations. 1(1), p.27-37.

Grossman, T.A. 1999. Teachers' Forum: spreadsheet modeling and simulation improves understanding of queues. Interfaces. 29(3), p.88-103.

Hora, S.C. 2003. Spreadsheet Modeling of the G/G/c Queuing System without Macros or Add-Ins. Informs Transactions on Educations. 3(3), p.86-89.

Ingolfsson, A.; T. A. Grossman. 2002. Graphical Spreadsheet Simulation of Queues. Informs Transactions on Educations, 2(2), p.27-39.

Kleinrock, L. 1975. Queuing Systems. New York: John Wiley.

Pritsker, A. A. B. 1998. Principles of Simulation Modeling. In *Handbook of Simulation: principles, methodology, advances, applications, and practice*. Ed. J. Banks. New York: John Wiley.

Schriber, T.J.; and D. T. Brunner. 1998. How Discrete-Event Simulation Software Works. In *Handbook of Simulation: principles, methodology, advances, applications, and practice*. Ed. J. Banks. New York: John Wiley.

Spearman, M.L.; D. L. Woodruff; and W. J. Hopp. 1990. CONWIP: a pull alternative to kanban. International Journal of Production Research. 28(5), p.879-894.

Winston, W. L. 2004. Operations Research, 4th ed. Belmont: Thomson Learning.

AUTHOR BIOGRAPHIES

MARCO AURÉLIO DE MESQUITA is Assistant Professor of Operations Management at Polytechnic School of the University of São Paulo (USP), in Brazil. He holds a B.S. degree in Naval Engineering, an M.S. and a Ph.D. degree in Production Engineering from the Polytechnic School of the University of São Paulo. His research interests include modeling and simulation of production systems, inventory management and production scheduling. He is a member of the ABEPRO (The Brazilian Association for Production Engineering) and of the INFORMS. His e-mail is <marco.mesquita@poli.usp.br>.

ALVARO EUZEBIO HERNANDEZ holds a B.S. degree in Electrical Engineering, an M.S. and a Ph.D. degree in Industrial Engineering from the University of São Paulo. He also holds an M.S. degree in Engineering-Economic Systems and an MS degree in Operations Research from Stanford University. In addition, he holds an MBA from the Getulio Vargas Foundation and a JD from the University of Sao Paulo. He is an assistant professor at the Industrial Engineering Department at the Polytechnic School of the University of São Paulo. His e-mail is <alvarohernandez@usp.br>.