Discrete Event Simulation Using Excel/VBA

David Elizandro and Jessica Matson
Tennessee Technological University

Abstract

The spectrum of discrete event simulation modeling courses in industrial engineering programs varies from an emphasis on learning concepts of discrete event simulation to modeling simple systems using a commercially viable simulation language. Often, when the emphasis is on learning a commercial language, much of the course becomes training at the expense of concepts. As a result the student has a basic understanding of the language and modeling but limited understanding of discrete event simulation concepts. In either case, modeling complex systems is problematic because students will lack sufficient knowledge of simulation concepts to understand nuances of the language or sufficient experience with constructs to be proficient with a commercial language. However, it is relatively easy to learn details of a simulation language for students who understand discrete event simulation and modeling concepts.

Excel/VBA is a ubiquitous software package with easy to use input and output features. Also, the statistical features of Excel overcome major limitations of modeling discrete event systems in a traditional procedural language. This paper examines a one-semester course in discrete event simulation that utilizes Excel and VBA to overcome limitations of traditional approaches for teaching simulation.

Introduction

Discrete event simulation is a tool that enables the user to compress time and study system performance characteristics. However, in order to perform simulation modeling, students must be able to define critical system component relationships as a function of time. Also, these relationships must be represented in software. In essence, students must learn concepts of discrete event simulation, acquire software development skills, and at the same time develop a global understanding of modeling concepts.

Students learn discrete event simulation concepts and terminology such as transactions, transaction attributes, and transaction movement in the model. System performance characteristics such as transient response and steady state cycle time, average queue lengths, time in the queue, and server utilization are expressed statistically using confidence intervals. Also a simulation language requires a student to rethink program control because segments of code operate "simultaneously." Such a course can easily overwhelm students.

Most discrete event simulation courses in industrial engineering emphasize modeling concepts. However, the discrete event simulation portion of the course varies from an emphasis on learning concepts of discrete event simulation to learning a commercially viable simulation language to model simple systems. Often, when the emphasis is on learning a commercial language, much of

the course becomes training at the expense of concepts. As a result the student has a basic understanding of the language and modeling but limited understanding of discrete event simulation concepts. In either case, modeling complex systems is problematic because students will lack sufficient knowledge of simulation concepts to understand nuances of the language or sufficient experience with constructs to be proficient with a commercial language.

The tradeoffs in educational approaches have been debated for many years among simulation education professionals. Kelton[3] presented the pros and cons of teaching the "classics," i.e., "any general-purpose procedural programming language that is not a simulation language at all" versus "high-level, icon-based simulation software." Nance and Sargent[4] noted that an "unsettling consequence" of modern simulation languages is that "users may have little understanding of how the model results are being produced" and "developers sometimes lack a sufficient understanding of the internal logic [of simulation programming languages] to enable the recognition of erroneous results produced by incorrect models." However, experience has shown that it is relatively easy to learn details of a simulation language for students who understand discrete event simulation and modeling concepts.

The following sections present a one-semester course in discrete event simulation in the Bachelor of Science in Industrial Engineering degree program at Tennessee Technological University that focuses on modeling and concepts of discrete event simulation. The course is based on previous work of Elizandro[1] and Starr[2]. In the Spring 2005 semester Excel/VBA tools will be incorporated into the course to facilitate learning discrete event simulation concepts and modeling concepts, and to serve as a first simulation language. Excel/VBA is a ubiquitous software package with easy to use input and output features. Also, the statistical features of Excel overcome major limitations of modeling discrete event systems in a traditional procedural language.

Simulation Course Environment

At Tennessee Tech, the discrete event simulation course is an industrial and systems engineering "tools" course scheduled in the spring semester of the junior year. Fall semester prerequisites for the simulation course have been Operations Research and Engineering Statistics. As of the Spring 2005 semester an Information Systems course will be a prerequisite for the course. The Operations Research and Engineering Statistics courses are typical foundation courses found in most ABET accredited industrial engineering programs, and a number of industrial engineering programs have recently added an Information Systems course to their undergraduate curriculum.

The Information Systems course was added to the senior year of the industrial engineering curriculum at Tennessee Tech University in the Fall 2002 semester. In the Information Systems course, students use Microsoft Excel, Access, and VBA to solve industrial engineering problems. Upon completion of the course, students have experience with constructs of the VBA language, ActiveX controls, and importing and exporting data between VBA and Excel. In the 2004-05 academic year, as a result of assessment data from senior exit interviews and capstone design juries, the course was moved to the fall semester of the junior year so that students could have these tools available for courses in the spring of the junior year as well as the senior year. The clear advantage to moving the course was that these tools could be used in subsequent courses.

The tradeoff is that the selection of laboratory assignments for the course becomes more restrictive because of the student's limited domain knowledge of industrial engineering topics. As reflected in this paper, the role and scope of Excel/VBA tools in the industrial engineering curriculum at Tennessee Tech continues to evolve.

In previous years, the programming experience of students in the simulation course was a three-credit freshman course in FORTRAN, C, or C++ with little reinforcement during the sophomore year. As a result, students had usually forgotten almost all of their programming skills. The primary language used in the simulation course has been Simnet. It is a process oriented simulation language with Source, Queue, and Facility blocks and FORTRAN constructs, for complex model logic[6]. Simnet is easy to learn, therefore, a minimum amount of course time is allocated to learning a simulation language. Because Simnet has been used as a teaching tool, advanced features of the language have not been covered. Constructs of Arena have been introduced during the last two weeks of the course.

An advanced course in simulation based on Siman/Arena has been offered as a technical elective. In this course, most of the advanced features of Siman are covered and there are several major projects. However, similar to the required simulation course, a significant portion of the course focuses on how concepts of discrete event simulation are implemented in Siman. Students who complete the simulation elective have had competitive submissions in the simulation competition sponsored by the Institute of Industrial Engineers.

Simulation Course Organization

As stated previously, the required simulation course is focused on concepts of discrete event simulation and modeling rather than language. In the first 25% of the course, students learn modeling terminology and the mathematics of simple queuing models. Using the Starr paper[2], the steady state response of a network of basic queuing systems is also examined. Thereafter, a Simnet program of the network has been used to evaluate transient and steady state system response.

In the next 25% of the course, Simnet programs and "pencil and paper" have been used to teach clock management, sequencing of "parallel" activities, processing random events, and data collection. The total of these concepts form the basic simulator logic. A Simnet program trace of a simple queuing model program has been the basis for learning the logic. To demonstrate the simulator logic, students are also required to perform a "pencil and paper" simulation of a simple queuing system model for a brief period of time.

The next 40% of the course has focused on basic features of the Simnet language and concepts of modeling using Simnet. Students complete four modeling assignments using Simnet. The assignment submission includes the program and a summary report that details the model development and an analysis of the system performance characteristics. Also to demonstrate understanding of relationships between system components modeled, students submit pseudo code for each primary event and its related secondary events.

The last 10% of the course has been an introduction to Arena with the fifth programming assignment a simple Arena program. Source material for the course has included Taha's[5] operations research textbook, Starr's paper on Jackson Networks[2], the Simnet language manual[6], and handouts on related topics.

Using definitions from Bloom's Taxonomy, the focus of the course is on the acquisition and synthesis of the concepts of modeling and discrete event simulation. Three exams and a final are used to assess the students' knowledge of these concepts. The course grade, which is a measure of that ability, is a weighted average of scores on examinations and laboratory assignments. Historically, a major problem in the course has been that the students' marginal programming competency limited their ability to grasp concepts and therefore their ability to synthesize the material.

Now that the Information Systems course is a prerequisite for the simulation course, students enroll in the course with knowledge of the constructs of the VBA language, ActiveX controls, and importing and exporting data between VBA and Excel. The course is being redesigned so that students can use their VBA skills to learn and reinforce discrete event simulation concepts and to model simple production systems.

Programming languages in the simulation course now include Excel/VBA, Simnet, and Arena. However, the role of each of these languages in the simulation course and the industrial engineering curriculum continues to evolve. Both Simnet and Arena have been used in the capstone projects. A recent graduate, who is also a contributor to this paper, chose Excel/VBA over Simnet and Siman/Arena to develop conveyor system modeling tools for his capstone project.

The focus and therefore the sequence of topics in the simulation course is the same as before. However, with the Information Systems course as a prerequisite, the presentation material for the simulation course has changed dramatically. The following section presents an overview of changes being made to the course.

Discrete Event Simulation Concepts

Tools similar to the Simnet language are being developed using VBA Classes. For example, a generalized Excel/VBA simulation program representing a network of queuing systems is now used to evaluate transient and steady state response of the network. The program is also used to examine the effects of various sampling rates and sampling intervals on estimates of the system's steady state response. Figure 1 is a view of the user interface for the simulation program and Figure 2 is an example of a performance metric for each queuing system in the network, and Figure 3 shows the Jackson network program output.

The Simnet program trace of a simple queuing model and "pencil and paper" simulation to present the basic simulator logic has also been replaced by a VBA program. The basic simulator logic described in the previous section is shown in Figure 4.

Table 1 presents VBA code of the simulator logic. Using the VBA debugger, students are able to view execution of the logic. The StepOver option of the debugger enables students to mask

event processing and data collection techniques in order to focus on the basic logic. Thereafter, details of event processing and data collection may be examined. The "pencil and paper" material on random number generators and random variables has been replaced with the statistical functions of Excel/VBA.

Definitions of discrete event simulation events used to develop models in VBA are:

> Primary Event: Events that are scheduled at a specific time. The list of primary events is time ordered on the Primary Events List.
>
> Secondary Event: Events that are not scheduled and therefore must reside on a Secondary Events List. Secondary events occur as a result of a primary event or another secondary event.

A customer scheduled for service completion is an example of a primary event. A customer waiting for service in a queue is a typical secondary event. The pseudo code for each primary event and related secondary event is now implemented in VBA. As shown in Table 1, an arrival event is specified in the basic simulator logic. Pseudo code and VBA for the arrival event are shown in Tables 2 and 3, respectively. In a similar fashion, students develop their models in VBA, add their events to the basic simulator logic, and validate the related event procedures.

"Trans" is an instance of a user-defined class, Transaction, defined in Table 4 and referenced in the Table 3 arrival procedure. In the arrival procedure, a transaction attribute is used to represent the transaction's current node in the network. Not only does the VBA class collection enable the user to maintain ordered primary and secondary events lists, it can also be used to maintain a list of model resources. The Primary Events List (PEL), SetOfSources, SetOfQueues, and SetOfFacilities referenced in Table 3 are instances of the VBA class collection. For example, the PEL is a time-ordered list of primary events.

When using a procedural language, students must understand how transactions move in the model. This concept is masked from users in most simulation languages. Figure 5 provides an overview of lists and transaction movements in a discrete event simulation model. In VBA, collections are used to represent the Primary and Secondary Events lists. Primary events initiate the movement and specify the destination of the transaction. For example path (a) represents scheduling a transaction for arrival in the model and path (c) could represent an arrival immediately scheduled for service completion.

Conclusions

A "tools" course in Information Systems emphasizing constructs of the VBA language, ActiveX controls, Excel, Access, and importing and exporting data between these systems to solve industrial engineering problems is now a prerequisite to the simulation course in the industrial engineering curriculum at Tennessee Tech University. This background has permitted modification of the simulation course to reflect the new tools that students bring to the course. The simulation course is being redesigned so that students can use their VBA skills to learn and reinforce discrete event simulation concepts and to model simple production systems.

Programming languages in the course are now Excel/VBA, Simnet, and Arena. Tools similar to the Simnet language are being developed using VBA classes.

Since the changes have occurred during the current academic year, assessment data are not yet available to determine the impact on learning. Over the next two years, annual end-of-year surveys, senior exit interviews, and capstone design project juries will be used to determine the effects of the change. Anecdotally, preliminary results of these changes are positive; however, the role of each of these languages in the simulation course as well as the industrial engineering curriculum continues to evolve.

References

1. D. Elizandro, "Another View of Simulation and Engineering Education," *Proceedings of the 1995 SCS Western Simulation Multiconference on Simulation in Engineering Education*, Vol. 27, Number 1, pp. 57-61.
2. Patrick J. Starr, "An Integrated Introduction to Simulation Using Deterministic Models, Queuing Results and Jackson Networks," *Proceedings of the 1992 SCS Western Simulation Multiconference on Simulation in Engineering Education*, Vol. 24, No. 2, pp. 235-239.
3. Kelton, W. D., "Teaching the Classics of Simulation to Beginners – Panel Contribution," Presentation for Simulation Education Track, 2003 Winter Simulation Conference, New Orleans.
4. Nance, R. E. and R. G. Sargent, "Perspectives on the Evolution of Simulation," *Operations Research*, Jan.-Feb. 2002, Vol. 50, No. 1, pp. 161-174.
5. H.A. Taha, *Introduction to Operations Research*, 7th Edition, Prentice-Hall, 2004.
6. H.A. Taha, *Simnet Users Manual*

Acknowledgements

The authors are please to recognize the contributions of Mr. Jacob Manahan and Mr. Clinton Thomas for their assistance with design and development of the software.

DAVID ELIZANDRO
David Elizandro is Professor of Industrial Engineering at Tennessee Tech University and a licensed P.E. He earned a B.S. in chemical engineering, M.B.A., and Ph.D. in industrial engineering from the University of Arkansas. He previously served as Dean of Mathematics, Science, and Engineering at University of the Incarnate Word, IME Department Chair at Tennessee Tech, and Head of Computer Science at Texas A&M University-Commerce.

JESSICA MATSON
Jessica Matson is Professor and Chairperson of the Industrial and Systems Engineering Department at Tennessee Technological University. She received her B.S. from Mississippi State University and her M.S. and Ph.D. from the Georgia Institute of Technology, all in industrial engineering. She has previously served on the faculty of Mississippi State University and the University of Alabama and is a licensed P.E. (Mississippi).

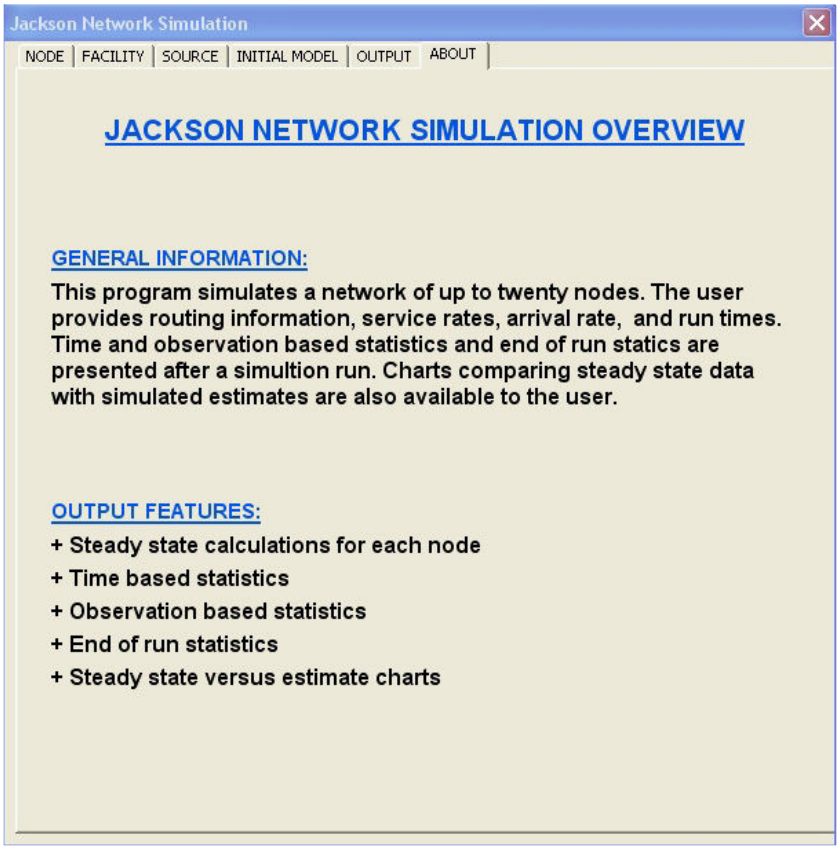

Figure 1: User Interface for Jackson Network Program

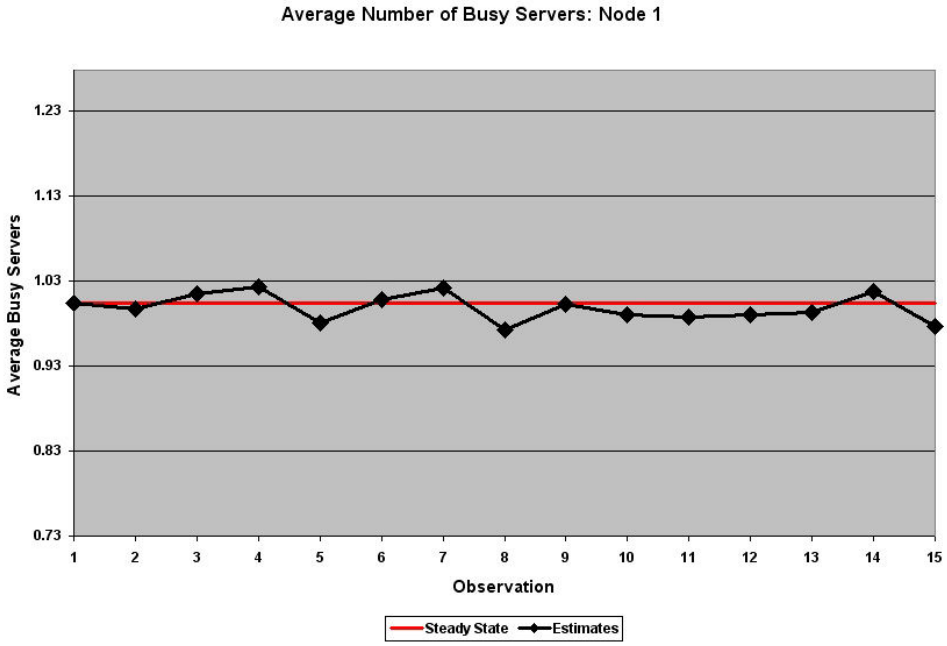

Figure 2: Jackson Network Program Output.

Queuing Model Output

Node	$c_i\mu_i$	λ_i	L_{si}	L_{qi}	W_{si}	W_{qi}
1	2	1.5	2.4642859	0.9642857	1.6428572	0.6428571
2	3	0.7957895	0.3610315	0.0957684	0.4536772	0.1203438
3	4	0.9789474	0.5050665	0.0155927	0.5159281	0.0159281
4	10	1.1873684	0.2391716	0.0016979	0.20143	0.00143
5	8	1.1633685	0.5816861	1.866E-06	0.5000016	1.604E-06
6	3	1.5789474	1.1111112	0.5847953	0.7037037	0.3703704

Figure 3: Jackson Network Program Output

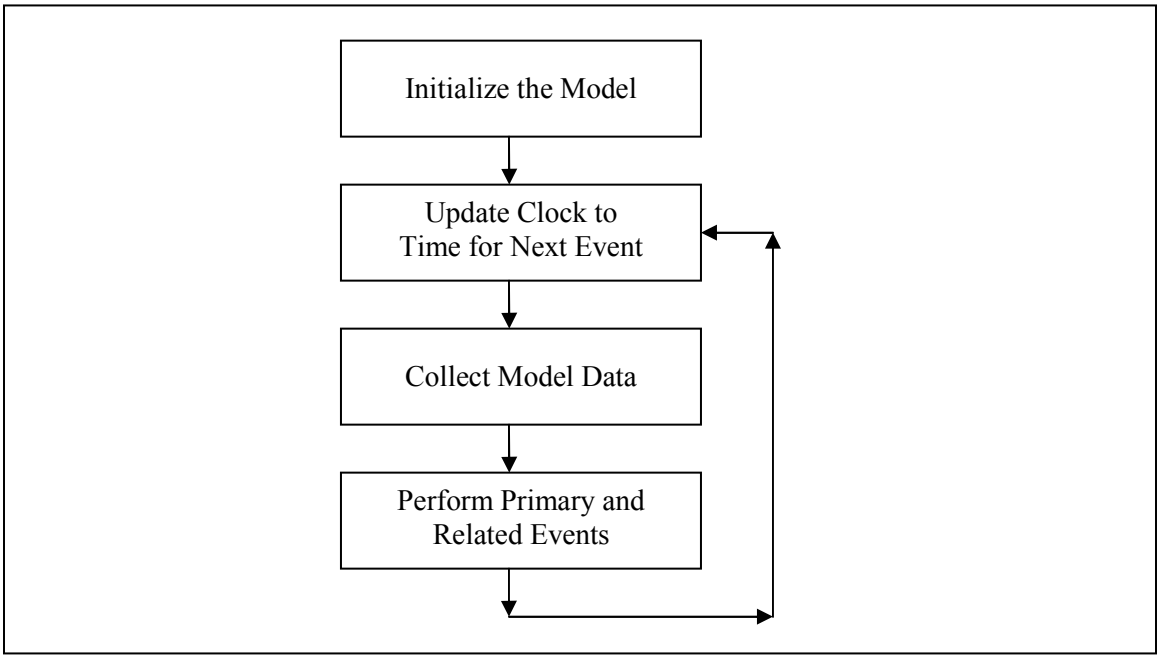

Figure 4: Basic Simulator Logic

Table 1: VBA of Basic Simulator Logic

```
Sub simulation(
  Call Initial                                  ‘ initialize the model
  While (CurrentTime < RunTime)
    Set NextEvent = PEL.Item(1)
    PEL.Remove 1
    PreviousTime = CurrentTime
    CurrentTime = NextEvent.TimeStamp           ‘ update the clock
    DeltaTime = CurrentTime - PreviousTime
    Call CollectData                            ‘ collect data
    EventType = NextEvent.EventType
    Select Case EventType                       ‘ perform event
      Case "A"
        Call arrival
      Case "SC"
        Call ServiceCompletion
      Case Else
    End Select
  Wend
  Call ShutDown                                 ‘ shut down the model
End Sub
```

Table 2: Arrival Pseudo Code

```
            ‘ find appropriate Server
Event Type = Service Completion
If Server = idle
Then
   Mark Node Server Busy
   Schedule Service Completion
   Insert into Primary Events List
Else
   Join Server Queue
End If
If Source = Origin
Then
   Schedule next arrival
End If
```

Table 3: Arrival Procedure in VBA

```
Sub arrival()
  Dim _
    duddy As Integer, _
    NodeID As Integer, _
    Trans As Transaction
        ' transaction at the node
        ' ready to be processed
        ' NodeID is in attribute #1
  AttrIndex = 1
  NodeID = NextEvent.AttrValue(AttrIndex)
  NextEvent.EventType = "SC"
  If SetOfFacilities.Item(NodeID).NServers > _
     SetOfFacilities.Item(NodeID).NBusyServers _
  Then                          ' server available
    SetOfFacilities.Item(NodeID).NumberIn = _
         SetOfFacilities.Item(NodeID).NumberIn + 1
    SetOfFacilities.Item(NodeID).NBusyServers = _
         SetOfFacilities.Item(NodeID).NBusyServers + 1
    NextEvent.TimeStamp = CurrentTime + _
         SetOfFacilities.Item(NodeID).ServiceTime
    Call InsertInPEL
  Else                          ' server(s) busy
    SetOfQueues.Item(NodeID).QueueTrans = NextEvent
  End If
  If NodeID = 1 _
  Then                          ' system arrival
     duddy = Source.NextTrans(Trans) ' schedule next system arrival
     Set NextEvent = Trans
     NextEvent.EventType = "A"
     NextEvent.Attr(AttrIndex) = 1
     NextEvent.TimeStamp = CurrentTime + _
          SetOfSources.Item(1).InterArrivalTime
     Call InsertInPEL
  End If
End Sub
```

Table 4: User Defined Transaction Class

```
Public TimeStamp As Single
Public EventType As String
Public TransID As Long
Public SourceID As String
Private A(1 To 10) As Variant
Public Property Let Attr(ByVal Index As Single, _
                         ByVal Value As Variant)
   A(Index) = Value        ' Assign attribute value
End Property
Public Property Get AttrValue(ByVal Index As Integer) As Variant
   AttrValue = A(Index)   ' retrieve attribute value
End Property
```

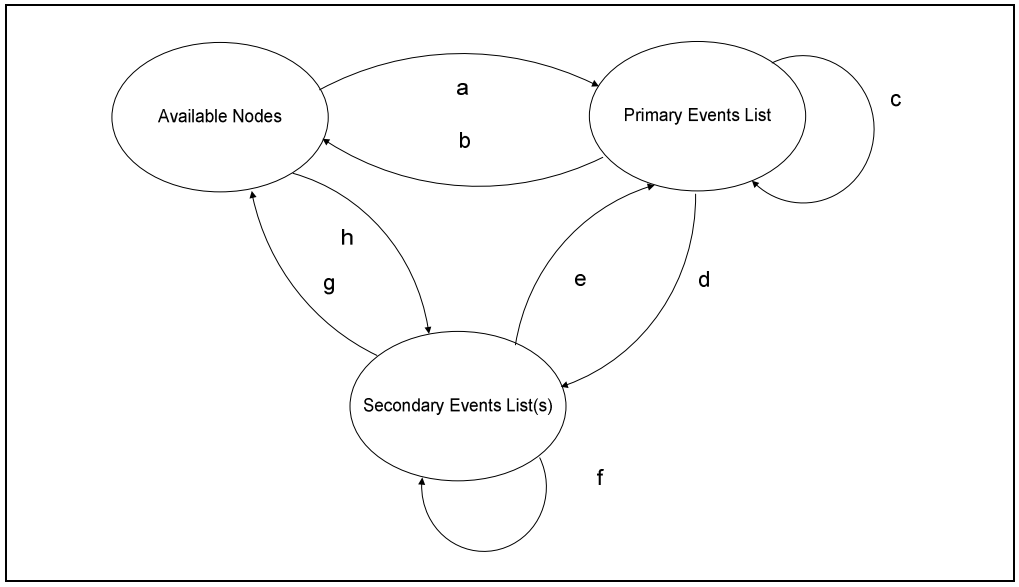

Figure 5: Transaction Flow in a Discrete Event Simulation Model