TEACHING SIMULATION BASICS THROUGH FLOWCHART SIMULATION THE EVENT SCHEDULING WORLD VIEW

Guilherme A B Pereira[(a)], Luís M S Dias[(a)], Hugo T C Rocha[(a)]

[(a)]Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho, Portugal

[(a)]gui@dps.uminho.pt, lsd@dps.uminho.pt, Hugo.TC.Rocha@azores.gov.pt

ABSTRACT

This paper refers to Event Scheduling World View, focusing on working explicitly with the foundational concepts of discrete event simulation, acting as an automatic generator of simulation programs, thus eliminating any programming effort and expertise. The main strategy is to enhance the utilization of flowcharts in modeling. Therefore, a graphical support tool (Microsoft Visio) is used to represent how the system really behaves and will also act as the source for the automatic generation of Visual Basic (VB) simulation programs. The software tool VBS (Visio Basic for Simulations) was created to read Visio flowcharts, interpret them and generate a VB simulation program.

Keywords: discrete event simulation, event scheduling world view, automatic generation of simulation programs, flowchart simulation

1. INTRODUCTION

(Banks 1998) defines simulation as "… the imitation of the operation of a real-world process or system over time..." and (Chase et al 2006) "…as a "…computer-based model used to run experiments on a real system…" These definitions call for the creation of a model that represents the behaviour of real processes or systems under analysis.

In this context, (Carson 2003) argues that a "…simulation model is a representation of a system or process … incorporates time and the changes that occur over time [and] … a discrete model is one that changes only at discrete points in time…"

However, (Schriber and Brunner 2008) convincingly support that "…A "black box" approach is often taken in teaching and learning discrete-event simulation software…". In fact, as far as discrete event simulation is concerned, teaching and learning approaches usually tend to neglect the full comprehension of simulation basic concepts. Again, (Schriber and Brunner 2008) state that "…the foundation on which the software is based is ignored or is touched on only briefly … The modeler therefore might not be able to think things through when faced with such needs as developing good approaches for modeling complex situations".

Most authors like (Pidd 1992), (Nance 1993), (Bennet 1995), (Zeigler 1976), (Ziegler et al 2000), (Brito and Teixeira 2001), (Guasch et al 2002), (Overstreet and Nance 2004) and (Sargent 2004) would refer three classical simulation approaches in Discrete Event Simulation - Event Scheduling World View, Process World View and Activities World View.

In the research here presented the authors concentrate their work on Event Scheduling World View, focusing on helping students construct their simulation "house", brick by brick, i.e., working explicitly with the foundational concepts of discrete event simulation – events, entities, resources, queues, randomness, future events schedule, simulation time advance, etc…

Event Scheduling World View essentially represents the behaviour of a system over time by means of defining specific events to occur at discrete points in time – these events, planned and executed, would mimic the real system.

Even though with a different approach as shown in (Dias et al 2008), in this work the authors again use Event Scheduling concepts, acting as an automatic generator of simulation programs, thus eliminating any programming effort and expertise. Previously (Dias et al 2005) have presented a similar procedure, but based in activity cycle diagrams.

2. EVENT SCHEDULING CONCEPTS

Under Event Scheduling paradigm one would define an event as an instantaneous action that might change the state of the system (Guasch et al 2002).

A change in the state of a system would refer to the state of the resources of that system. Each time a resource state changes from busy to free or otherwise, one could say that the state of the system would also change.

Therefore, every instant in time where the state of a system might change would be defined as an instant where a specific event has occurred.

At that time, the tasks involved in that event would have to be performed.

These tasks would reflect not only the implementation of the change in the system, as far as resources, queues and flow of entities are concerned (representing physical modifications in the real system) but also logical changes reflecting the planning of future events, recording statistics for future use and also

generating random variables to model stochastic behaviour of the real system.

The main strategy of this research work is to enhance the utilization of flowcharts in modeling, thus making it a great contribution to automatic generation of simulation programs, keeping it simple. Therefore, a graphical support tool is used to represent how the system really behaves and will also act as the source for the automatic generation of simulation programs.

Flowcharts are probably the most widely used graphical syntax in behaviour specification (Gilbreth and Gilbreth 1921). The first known mathematical formalization was made by (Nassi and Shneiderman 1973). It can be accepted as a universal visual language, and it can be easily assumed that every professional, in some technical work, has already used it.

(Pidd 1992) and (Tocher 1963), even support the view that when generic programming languages were replaced by specific purpose simulation languages the use of paper diagrams remained as a previous step to programming.

In fact, this paper emphasizes the importance of this step, by proposing a way for automatically translating "paper diagrams" (flowcharts) into a simulation program.

3. EVENT SCHEDULING IMPLEMENTATION

The Event Scheduling simulation philosophy, as previously mentioned, is based upon the identification of events that, together, would represent the mimic of the system under analysis. The identification of each event is complemented with the definition of the tasks to be performed each time an event occurs. These tasks, as far as a discrete simulation approach is concerned, would include:

1. Managing queues (removing/inserting entities from/in queues)
2. Managing resources utilization (either seizing or releasing resources)
3. Recording statistics (for future evaluation of performance indexes, i.e., average waiting time in queue, average queue length, average resource utilization, etc.)
4. Generating random variables
5. Managing future events schedule and simulation time advance

Basic Simulation Facility – BSF (Thesen 1978), constitutes a way of implementing such a representation model for the behaviour of a real system over time. BSF is based on the computer programming concept of managing files. Moreover a file, as far as a simulation systems is concerned could represent:

1. the behaviour of a queue, where

- Inserting an entity in a queue would be represented through the insertion of a record on that file
- Removing an entity from a queue would be represented through the removal of a record from that file

2. the state of a resource, where

- seizing a resource would be represented through the insertion of a record in that file
- releasing a resource would be represented through the removal of a record from that file

3. the future events list, where

- planning a future event would be represented through the insertion of a record in that file
- executing an event would be represented through the removal of a record from that file

This type of implementation, using an appropriate data structure to accomplish the above features would also be useful to record statistics, using the mentioned files – these tasks would simply involve using some fields of the records of those files in order to register the statistical information needed.

BSF includes the following four routines already developed:

1. INIT – essentially dedicated to the design and initialization of the data structure that supports the simulation
2. INSERT – basically dedicated to the insertion of a record into a file (e.g. the arrival of an entity to a queue, or seizing a resource or even the planning of a future event)
3. REMOVE – basically dedicated to the removal of a record from a file (e.g. the removal of an entity from a queue, or releasing a resource or even preparing the execution of a future event)
4. REPORT – essentially dedicated to the computation of simulation performance measures

These routines, and the associated philosophy, could be found (implemented) in various programming languages (Java, C, VB, Pascal, Excel VBA, etc.). Nevertheless, it is essential to develop a computer program, specifically dedicated to the system under analysis, which would invoke these routines, thus creating a mimic of the system. The development of this computer program, together with the correct utilization of the aforementioned routines, is usually better described (modeled) by the use of appropriate flowcharts for each event identified.

Next section presents a software tool based on flowcharts, built and founded on key issues of the Event

Scheduling philosophy, which automatically generates a simulation computer program to mimic the real system.

4. SOFTWARE TOOL DEVELOPED

The relevancy of creating a simulation software tool capable of generating a simulation program (VBS – Visio Basic for Simulations) is based on the following general principles/premises:

1. event scheduling world view definitely contributes to a better understanding of the foundations of simulation, i.e., the basic concepts of simulation
2. VBS would emphasize the utilization of a visual approach to deal with the representation of the real problem through the event scheduling paradigm
3. VBS would literally enable the automatic generation of a simulation program (VBA or Java program) thus with no programming effort at all

VBS would then use the well known graphical editor Microsoft Visio for incorporating the event scheduling flowcharts representing each event. This task would be accomplished through the creation of Visio Shapes that would reflect the different tasks involved in each event identified. Then VBA (Visual Basic for Applications) would interact with Visio, interpreting the shapes and tasks associated, as well as the sequence of shapes to be "executed". At the end, this means that the effort to run simulation experiments through this tool would be equivalent to building event scheduling flowcharts on a piece of paper.

This was the challenge of the work presented in this paper.

For this purpose, and according to the ideas presented in section 3, the software tool would have to implement the tasks for each event. In fact, through VBS a flowchart for each event will be created and a main flowchart will coordinate the execution of each event flowchart.

For each type of task, the software tool includes the following Visio Stencils and the corresponding Visio Shapes:

- For managing queues (Figure **1**) – Shape Insert (inserting an entity in a Queue) and Shape Remove (removing an entity from a queue). Additionally there is a shape for queue declaration.

  Parameters: Queue identification

Figure 1: Queue Stencil

- For managing resources utilization (Figure **2**) - Shape Seize (seizing a resource) and Shape Release (releasing a resource). There is also a shape for declaring each resource.

  Parameters: Resource identification

Figure 2: Resource Stencil

- For managing future events schedule and simulation time advance (Figure **3**) – Shape Plan (planning future event to occur, inserting information on the Future Events List), Shape Identify (identifying next event to occur, reading information from the Future Events List) and Shape Execute (transferring the execution of the program to the corresponding flowchart of the next event to be executed).

  Parameters: Identification of next event to occur and future instant for that event to occur

Figure 3: Event Stencil

- For generating random variables (Figure **4**) – Shape Random, generating a value from a statistical distribution with the respective distribution parameters.

  Parameters: Statistical Distribution and corresponding distribution parameters

Figure 4: Statistics Stencil

In Event Scheduling World View there is frequent need to identify if:

- a resource is free
- a queue is empty

The tool developed includes a shape for each of the situations above – see Figure **5**. The parameters include, respectively, the identification of the resource and the queue.

Figure 5: Decision Stencil

Additionally, every shape needs to be linked to other shapes. There are also a set of Link shapes for that purpose – see Figure **6**. These include the straight link between shapes, but also output decision links from decision shapes – True and False outputs following a decision point.

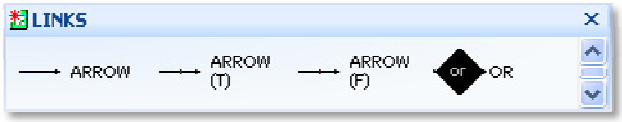

Figure 6: Links Stencil

Finally, and for software implementation purposes, there are also shapes for activating the main program as well as for activating each event. At the end of each event – after executing all the tasks for that event, a Return shape is used to direct program execution back to the main flowchart. This flowchart, at the end of the simulation, uses the shape Report to compute some statistical indicators for evaluating the system performance – Figure **7** below.

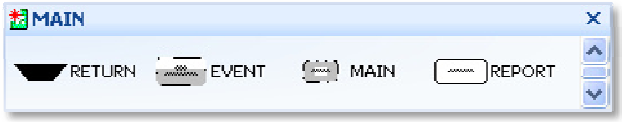

Figure 7: Main Stencil

The tool developed (VBS) promotes the interaction between Visio shapes and VBA (Visual Basic for Applications), that interprets each shape and corresponding parameters and generates computer code accordingly. Figure 8 and Figure **9** show general screenshots of the VBS application. At the end, a whole computer program is generated, compiled and executed.

Next section will present an application example, the corresponding interaction with the software tool and the computer code (VB) generated.

5. APPLICATION EXAMPLE

For a brief explanation of the tool developed, a simple example is presented.

This example represents a system that incorporates the arrival of entities. Once in the system, the entities would require the use of a first resource (*Resource_A*). Then, the entity would also require the use of a second resource (*Resource_B*). There is a first-in first-out type of queue associated with each resource. The entity will then leave the system.

The Event Scheduling philosophy would identify three events – Entity Arrival Event (Figure **10**); End of *Resource_A* Utilization (Figure **11**); End of *Resource_B* Utilization (Figure **12**). Figure **13** represents the main flowchart that integrates and coordinates the behaviour of each event occurring in the system.

Following each flowchart is the respective code, automatically generated by VBS, based on each shape and respective parameters.

VBA, by reading each flowchart, would recognize each shape and would interpret each parameter of each shape. Computer code generation would then follow in accordance.

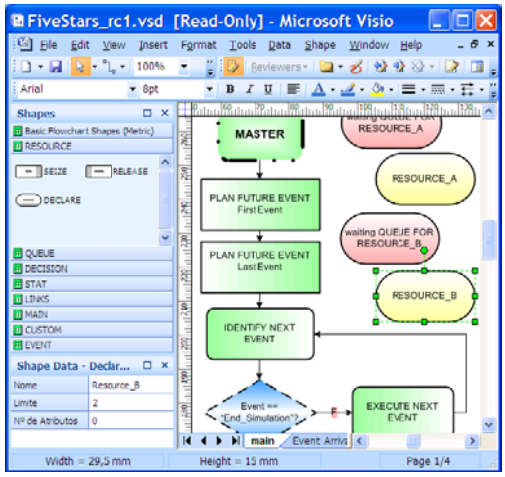

Figure 8: VBS Screenshot 1

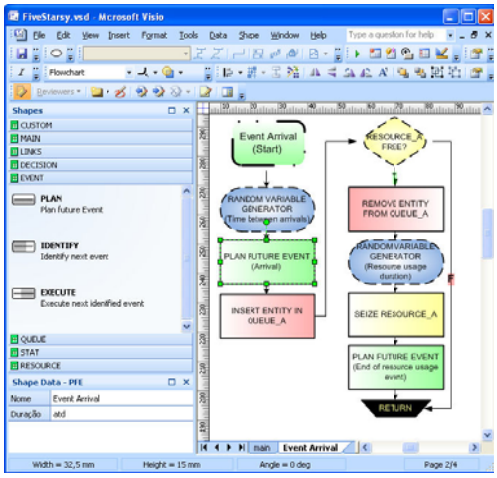

Figure 9: VBS Screenshot 2

For a better understanding of the relation between each shape and the corresponding computer code, let us follow the flowchart for Event Arrival (Figure **10**): first shape corresponds to the activation of the respective flowchart, meaning a call to the routine Sub Arrival. Then the shape Random generates line code *T=RVG("EXPO(5)")* – this means that the parameters of this shape were recognized as having an Exponential type of distribution with a mean of 5 (minutes). Shape Plan follows, indicating the planning of a future event – inserting a record in the future events list – line code *Insert("FutureEvents", Time+T, "Arrival")*. The entity that has just arrived in the system will be part of the

corresponding queue – line code *Insert ("QUEUE_A",Time)*.

Then, if *Resource_A* is free (next shape), the following actions need to be performed:

- removing entity from the queue
- generating duration of *Resource_A* utilization
- seizing *Resource_A*
- planning next event – end of *Resource_A* utilization
- end of flowchart

The above actions are equivalent to the following code generated by the application:

Table 1: Partial code for Event Arrival

```
If Size("RESOURCE_A")<Max_A Then
  Remove("QUEUE_A")
  T=RVG("normal(4,1)")
  Insert("RESOURCE_A",Time+T)
  Insert("FutureEvents", Time+T,
     "End of Resource_A Utilization")
End If
```

Having performed these steps, the full code for this flowchart is shown in Table **2**.

Table 2: Code for Event Arrival

```
Sub Arrival()
  T=RVG("EXPO(5)")
  Insert("FutureEvents", Time+T, "Arrival")
  Insert("QUEUE_A",Time)
  If Size("RESOURCE_A")<Max_A Then
    Remove("QUEUE_A")
    T=RVG("normal(4,1)")
    Insert("RESOURCE_A",Time+T)
    Insert("FutureEvents", Time+T,
       "End of Resource_A Utilization")
  End If
End Sub
```

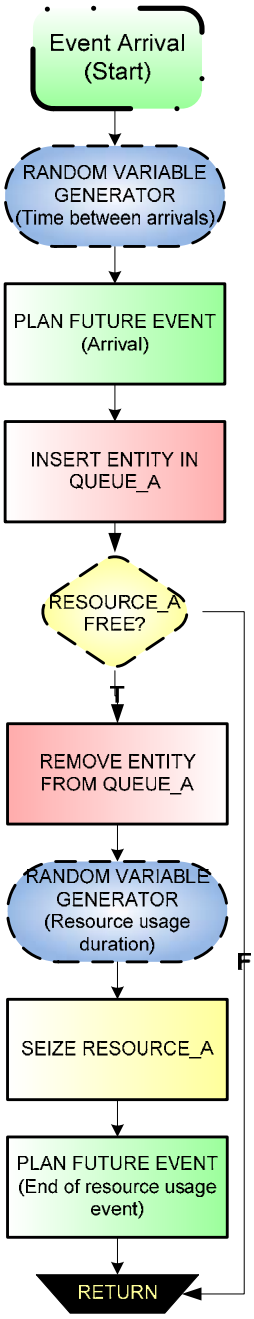

Figure 10: Entity Arrival Event Flowchart

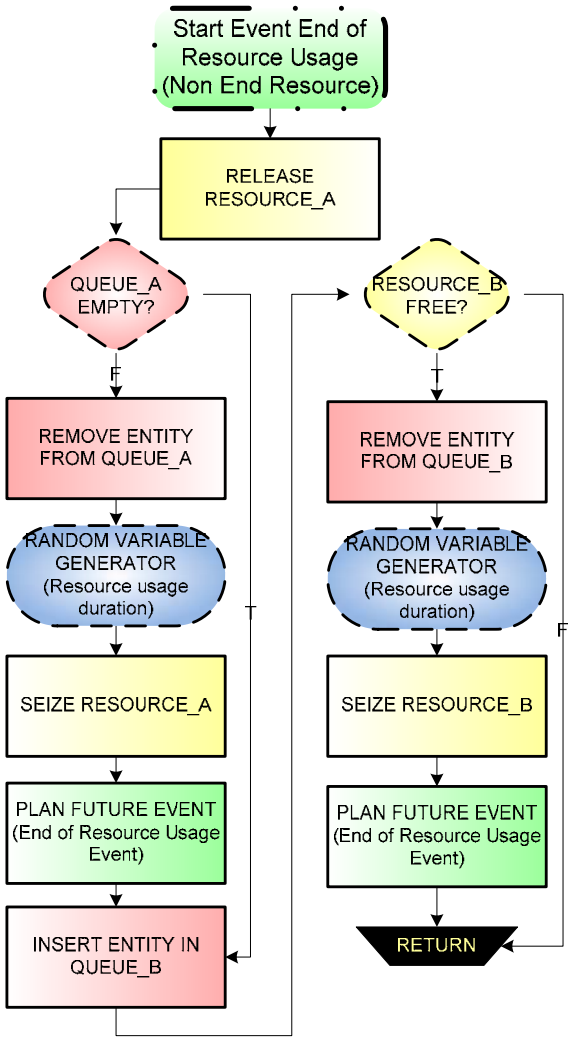

Figure 11: End of *Resource_A* Utilization Event Flowchart

Similar procedures would generate computer code of Table **3** for event represented in Figure 11.

Table 3: Code for Event End of *Resource_A* Utilization

```
Sub End_of_Resource_A_Utilization()
  Remove("RESOURCE_A")
  If Size("QUEUE_A")>0 Then
    Remove("QUEUE_A")
    T=RVG("normal(4,1)")
    Insert("RESOURCE_A",Time+T)
    Insert("FutureEvents", Time+T,
       "End of Resource_A Utilization")
  End If
  Insert("QUEUE_B",Time)
  If Size("RESOURCE_B")<Max_B Then
    Remove("QUEUE_B")
    T=RVG("normal(18,2)")
    Insert("RESOURCE_B",Time+T)
    Insert("FutureEvents", Time+T,
       "End of Resource_B Utilization")
  End If
End Sub
```

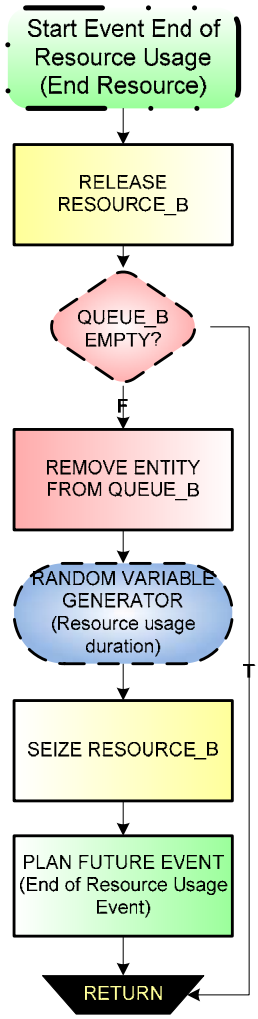

Figure 12: End of *Resource_B* Utilization Event Flowchart

Also for flowchart of Figure 12, the following code is generated.

Table 4: Code for Event End of *Resource_B* Utilization

```
Sub End_of_Resource_B_Utilization()
  Remove("RESOURCE_B")
  If Size("QUEUE_B")>0 Then
    Remove("QUEUE_B")
    T=RVG("normal(18,2)")
    Insert("RESOURCE_B",Time+T)
    Insert("FutureEvents", Time+T,
       "End of Resource_B Utilization")
  End If
End Sub
```

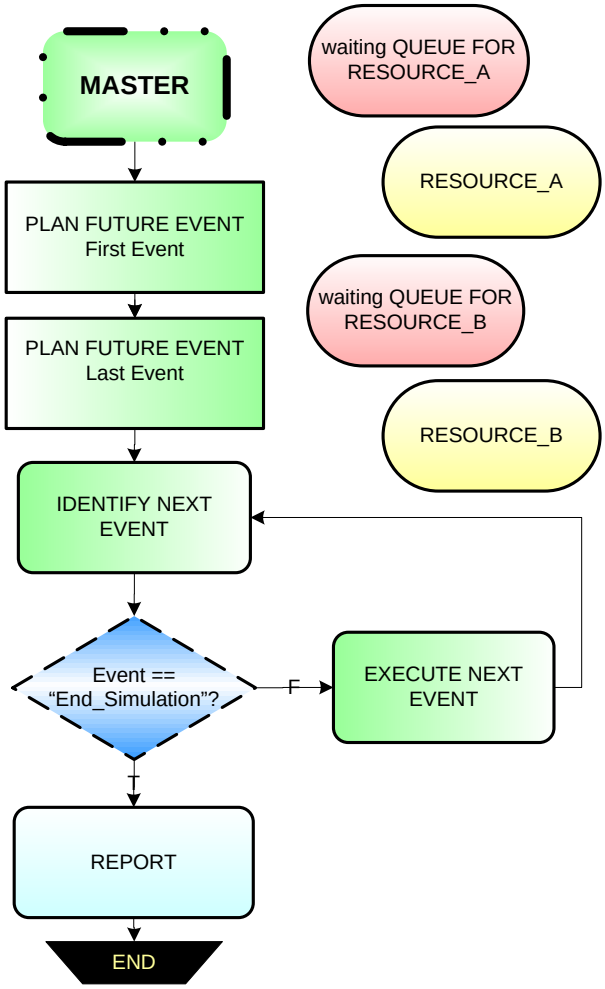

Figure 13: Main Flowchart

Finally for flowchart of Figure **13**, the code included in Table **5** is generated.

Table 5: Code for Main Flowchart

```
Main Program:
   INIT()
   Max_A=1
   Max_B=4
   Insert("FutureEvents", 0, "Arrival")
   Insert("FutureEvents",1000,"End_Simulation")
   DO
     Event=Remove("FutureEvents")
     If Event == "Arrival" Then Call Arrival
     If Event == "End of Resource_A Utilization"
        Then
            Call End_of_Resource_A_Utilization
     If Event == "End of Resource_B Utilization"
        Then
            Call End_of_Resource_B_Utilization
   UNTIL event == "End_Simulation"
   Call Report
```

This computer program is then ready for compiling and executing. At the end, the usual performance indicators are available – average time spent in a queue, average time spent in the system, average queue length, average number of resources busy, etc.

6. CONCLUSIONS

The software tool developed shows three particularly interesting features, namely:

- It is based on simple flowcharts that follow Event Scheduling Simulation philosophy
- It automatically generates a VB computer program to perform the mimic of the system under analysis
- It runs the model directly over the flowcharts, producing debugging trace files

These features, together, would contribute to

- the generalization and a better understanding of the use of simulation
- the comprehension of the foundations of simulation
- the automatic generation of simulation programs

In brief, it can be argued that the generalization and better understanding of the use of simulation would have been accomplished since the tool herein developed only requires i) expertise on a basic simulation approach: event scheduling, ii) incorporating simple flowcharts that define the system and its functioning rules.

Furthermore, these flowcharts, apart from providing an understanding of the system's behaviour, contribute to the comprehension of fundamental simulation concepts, such as entities, queues, resources and also to a very important simulation concept – the evolution of the state of the system over time.

Finally, these simple flowcharts are translated into the software tool by means of an automatic generation of a computer program that performs the mimic of the system and evaluates the corresponding efficiency measures. The simulation runs over the events' flowcharts, step by step, enabling the user to gradually assimilate concepts while validating his learnings.

REFERENCES

Banks, J., 1998. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley and Sons, Inc.

Banks, J., 2000. Introduction to Simulation, in Proceedings of the 2000 Winter Simulation Conference, eds., J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, pp. 9-16.

Bennett, B. S., 1995. *Simulation Fundamentals*, Prentice Hall, ISBN 0-13-813-262-3

Brito, A. E. S. C. and Teixeira, J. M. F., 2001. *Simulação por Computador – Fundamentos e implementação de código em C e C++*, Publindústria - Edições Técnicas, ISBN 972-98726-2-7

Carson, J. S., 2003. Introduction to Modeling and Simulation, Proceedings of the 2003 Winter Simulation Conference, eds., S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, pp. 7-13

Chase, R. B., Jacobs, F. R. and Aquilano, N. J., 2006. Operations Management for Competitive Advantage, McGraw-Hill/Irwin.

Dias, Luís M. S., Rodrigues, António J. M. G. and Pereira, Guilherme A. B., 2005. An Activity Oriented Visual Modelling Language with Automatic Translation to Different Paradigms, Proceedings of the 19th European Conference On Modelling And Simulation (ECMS 2005), Riga, Letónia. Ed. Yury Mercuryev et al, pp. 452-461

Dias, Luis M. S., Pereira, Guilherme A. B. and Oliveira, José A. V., 2008. Event Scheduling Made Easy: Basic Simulation Facility Revisited, in Proceedings of the European Modeling and Simulation Symposium, EMSS 2008, Lamezia, p. 688-692.

Gilbreth F.B. and Gilbreth L.M., 1921. Process Charts - First Steps in Finding the One Best Way to do Work. Presented at the Annual Meeting of The American Society of Mechanical Engineers, New York, USA.

Guasch, A., Pierra, M., Casanovas, J. and Figueras, J., 2002. Modelado y Simulación – Aplicación a procesos logísticos de fabricación y servicios. Edicions de la Universitat Politècnica de Catalunya, Barcelona.

Nance, R. E., 1993. A history of discrete event simulation programming languages, in the Second ACM SIGPLAN Conference on History of Programming Languages (Cambridge, Massachusetts, United States, April 20 - 23). HOPL-II. ACM Press, New York, NY, pp 149-175. DOI=10.1145/154766.155368

Nassi and Shneiderman, 1973. Flowchart techniques for structured programming, ACM SIGPLAN Notices, 12.

Overstreet, C. M. and Nance, R. E., 2004. Characterizations and Relationships of World Views, in *Proceedings of the 2004 Winter Simulation Conference*, ed. R .G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters, pp 279-287, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Pidd, M., 1992. Computer Simulation in Management Science. 3rd Ed. John Wiley & Sons, Inc.

Sargent, R. G., 2004. Some recent advances in the process world view, *Winter Simulation Conference,* pp 293-299.

Schriber, T. J. and Brunner, D. T., 2008. Inside discrete-event simulation softe«ware: how it works and why it matters, in *Proceedings of the 2008 Winter Simulation Conference, eds. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson and J. W. Fowles.*

Thesen, A., 1978. Computer Methods in Operations Research, New York: Academic Press.

Tocher, K. D., 1963. The Art of Simulation, UNIBOOKS – English Universities Press.

Zeigler, B. P., 1976. *Theory of Modeling and Simulation,* Wiley, New York.

Zeigler, B. P., Herbert, P. and Tag, G. K., 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, 2nd edition, Academic Press.

AUTHORS BIOGRAPHY

Guilherme A B Pereira was born in 1961 in Porto, Portugal. He graduated in Industrial Engineering and Management in the University of Minho, Portugal. He holds an MSc degree in Operational Research and a PhD degree in Manufacturing and Mechanical Engineering from the University of Birmingham, UK. His main research interests are Operational Research and Simulation.

Luís M S Dias was born in 1970 in Vila Nova de Foz Côa, Portugal. He graduated in Computer Science and Systems Engineering in the University of Minho, Portugal. He holds a PhD degree in Production and Systems Engineering from the University of Minho, Portugal. His main research interests are Operational Research, Simulation and Systems Visual Modeling.

Hugo T C Rocha was born in 1975 in Aveiro, Portugal. He graduated in Computer Engineering in the University of Minho, Portugal. During his degree he developed some research work in simulation. He currently works in the Center of Informatics for Finance of the Regional Government of Açores, Portugal and he is a professor of Information and Communication Technologies in the Professional School of Santa Casa da Misericórdia dos Açores. His main research interests are Informatics and Simulation.