9. USING DOTNETSIM

CHAPTER OVERVIEW

In this chapter, the use of the DotNetSim prototype is demonstrated by modelling and simulating a single server system, which is a fine-grained component that occurs in a large number of DES systems. Parts, cars or customers arrive at the system and wait in a FIFO disciplined queue to be served by a single server. This is modelled into an Event Graph within the DotNetSim modelling environment by using its menu driven modelling commands. Prior to this, a demonstration is provided of how to create the Event Graph stencil with the modelling notation. Hence, the use of the DotNetSim modelling component is demonstrated in 5 major steps, namely (i) drawing, listing and redrawing the Event Graph (ii) describing the model (iii) reviewing the model (iv) reporting the model and (v) general operations on the model.

Simulations of the single server model are then run by the DotNetSim simulation engine which places the results in an Excel workbook. These are displayed and analysed within the DotNetSim output analysis environment. Additionally, other use cases are presented to illustrate the utilisation of features such as the parameterised and cancelling edges implemented by DotNetSim but not required by the single server model.

9.1. USE CASES

Given that Event Graphs are able to model the important features of DES systems by composing fine-grained and loosely coupled sub-models [107], we demonstrate the DotNetSim prototype by modelling a single server system which occurs in a large number of DES. Real life examples include a toll system, a call centre or a machine centre. These systems comprise an arrival process and a service process:

- Arrivals of cars, phone calls or parts occur one at a time; they are not affected by the current length of the queue; they queue on a FIFO basis and their inter-arrival time can be described by a probability distribution [127, 85] such as the negative exponential distribution. Arrivals are independent and occur one after the other, i.e. one arrival schedules the next one within the simulation.
- Service, for example, collecting the payment, answering a phone enquiry or processing a part, also occurs individually; the service time is unaffected by the length of the queue and it follows a probability distribution [127, 85] such as the uniform distribution. One service starts when the last finishes or when a customer arrives, hence services are delimited by a start event and a finish event. The start event schedules its finish and the latter schedules a new start if the queue is not empty.

This single server system can be modelled as a set of events e={run, arrival, start-service and finish-service} which transit the system throughout the states s={s1, s2, s3, s4}. The state of such a system is basically characterised by variables which represent the number of cars, customers or parts that arrive at the system (N), the number of arrivals queuing (Q) while waiting to be served and the status of the server (S) that follows the convention{0-busy, 1-idle}.

Each event transits the system from one state to the next and schedules future events. The run event schedules the first arrival; when an arrival occurs, the customer joins the queue so that N and Q are incremented by one. A service starts when the server is idle and there is at least one customer queuing. Starting the service decrements the queue and turns the server to busy. The end of a service toggles the status of the server and, when the service ends, if the queue is not empty, the next customer is served. Fig. 9.1 depicts the Event Graph and state transition of such a single server component.

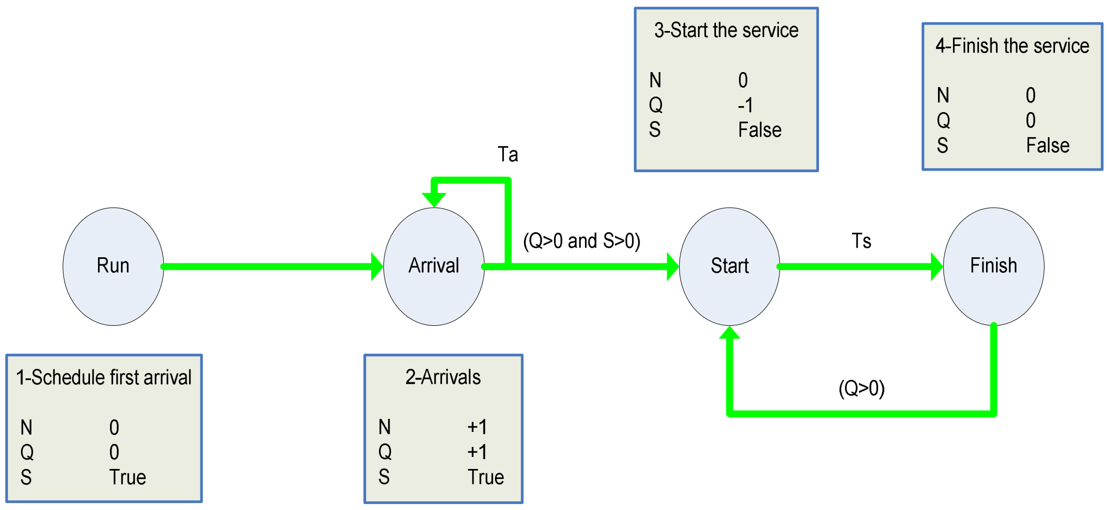


Fig. 9.1: Event Graph of a single server queue with the associated state transitions

This is the basic model for single server systems. However, additional variables may be derived from the state variables in Fig. 9.1 to collect time varying statistics on the performance of the system. Other events can be considered if the level of detail requires this.

This model component can easily be extended to a multiple server system in which the number of idle servers (S) is set to the number of existing servers. Also, it can be plugged into other multiple server systems to simulate, for example, a system in which a part is processed sequentially by a number of machines, creating a sequential set of queues.

9.2. CREATING THE EVENT-GRAPH STENCIL

Before using DotNetSim for modelling the Event-Graph stencil must be created by running the EGstencil.vsd (see Fig. 9.2). This starts by requiring the user to choose the data source from which the Stencil menu is generated. By double clicking Visio, the menu is generated from within Visio^{TM}, otherwise it is generated from the Excel file

C:\DotNetSim\Stencils\MenuStencil. The Event-Graph stencil can then be created and master shapes can be added to it by selecting the appropriate commands on the Stencil menu.

```
1> Run Visio
2> Select File – Open
3> C:\ DotNetSim\Stencils\EGstencil
```

Fig. 9.2: Commands to create the Event-Graph stencil

9.3. RUNNING AND CUSTOMISING THE DOTNETSIM MODELLING COMPONENT

By instantiating the EGmodelling template (see Fig. 9.3), we gain access to the functionality implemented by the DotNetSim modelling component described in chapter 6. Visio^{TM} starts up with only its File and the Event Graphs menus, which are the only tools a DES system requires for modelling. However, the user can activate other menus, toolbars and stencils by selecting Customising Visio on the Event Graphs menu at any time.

```
1> Run Visio
2> Select File – New – Choosing Drawing Type
3> Select Template on my computer from the Tasks Pane
4> C:\DotNetSim\EGmodelling
```

Fig. 9.3: Commands to instantiate the EGmodelling template

9.4. MODELLING A DES SYSTEM IN DOTNETSIM

Modelling a DES system within DotNetSim may be summarised into the following four broad stages described in detail in this section:

(i) The system is modelled into an Event Graph, which is drawn and redrawn until we get a reasonable representation of the system. Meanwhile, data on the events and the edges is added. Each sketch should be labelled to facilitate a clear understanding of the system logic. To proceed to the next stage, the version of the Event Graph accepted for use has to be read automatically so that the structure of the scheduling and the cancelling relationships between events is captured. Also, it may be automatically labelled according to the Event Graph modelling notation.

(ii) Data is collected on the dynamic behaviour of the system. General attributes, state variables, state transitions and simulation parameters have to be defined at this point.

(iii) Diagrammatical and modelling data are automatically placed into a relational database so that we can review the model's properties either in Visio^{TM} or Excel^{TM}.

(iv) Finally, the model is ready to be reported either within Visio^{TM} or in other Microsoft applications.

The Event Graphs menu unfolds these steps into a set of commands which are to be run in sequence so as to facilitate the modelling of a DES system.

9.4.1. DRAWING, LISTING AND REDRAWING EVENT GRAPHS

To illustrate the use of the DotNetSim modelling component, we will develop the Event Graph shown in Fig. 9.4. This will be done in two stages to illustrate different ways in which DotNetSim can be used to develop Event Graphs. First, we started with the user to draw the Event Graph in Fig. 9.5, which is Fig. 9.4 without event C.

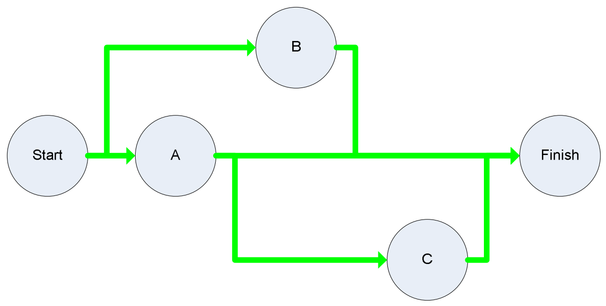


Fig. 9.4: Network of events as drawn automatically from an Excel descriptive list (C:\DotNetSim\WorkingFiles\Netw2.vsd)

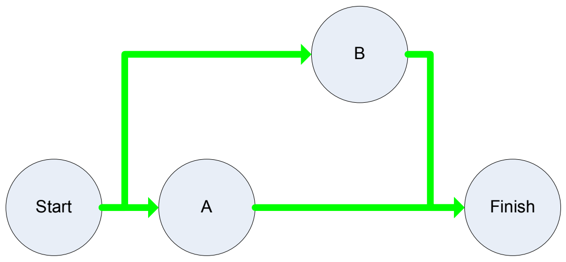


Fig. 9.5: Network of events as drawn, read and labelled in the DotNetSim modelling componen (C:\DotNetSim\WorkingFiles\Netw1.vsd)

Intuitively, we started drawing Event Graphs directly by dragging and dropping the events and edges from the stencil onto the drawing page. Then, the Event Graph was read and labelled (see section 6.2.3.) by selecting Reading the diagram – Reading the diagram sequence and Reading the diagram - Labelling the diagram on the Event Graphs menu.

Next, the diagram was converted into the table 9.1 by selecting Reporting the model - Listing diagram to Excel on the same menu. The C:\DotNetSim\Netw1.xls

Number	Level	Node/Edge	Name	Description	Origin	Destination
1	2	Event	Start	Starting		
2	2	Event	A	Op A		
3	1	Event	B	Op B		
4	2	Event	Finish	Finishing		
1		Schedule			1	2
2		Schedule			1	3
3		Schedule			2	4
4		Schedule			3	4

Table 9.1: Diagram's descriptive list saved in C:\DotNetSim\Netw1.xls

was previously created by selecting Utilities on the Event Graphs menu.

To illustrate the inter-changeability of graphical and Excel modelling, we will add

Event C to the Excel list of table 9.1. This leads to table 9.2 which, when re-drawn within the DotNetSim modelling component by selecting Drawing the diagram - Descriptive List in Excel on the Event Graphs menu, displays the Event Graph of Fig. 9.4.

Number	Level	Node/Edge	Name	Description	Origin	Destination
1	2	Event	Start	Starting		
2	2	Event	A	Op A		
3	1	Event	B	Op B		
4	3	Event	C	Op C		
5	2	Event	Finish	Finishing		
1		Schedule			1	2
2		Schedule			1	3
3		Schedule			2	5
4		Schedule			3	5
5		Schedule			2	4
6		Schedule			4	5

Table 9.2: The new descriptive list of the network (C:\DotNetSim\WorkingFiles \Netw2.xls)

9.4.1.1. DRAWING THE SINGLE SERVER EVENT GRAPH

The Event Graph of the single server system was drawn directly by dragging and dropping onto the drawing page the four events of the single server component and interconnecting them with scheduling edges. Lists of events and edges are displayed in Table 9.3 and Table 9.4.

Event Name	Description	Parameter
Run	Scheduling the first Arrival	---
Arrival	New arrival	---
Start	Start the service	---
Finish	Finish the service	---

Table 9.3: List of events and prompted properties

Edge	Delay time	Delay time Distribution	Condition	Priority
Run – Arrival	---	----	---	1
Arrival-Arrival	Ta	Negative Exponential	---	1
Arrival - Start	---	---	Q>0 and S>0	2
Start – Finish	Ts	Uniform	---	1
Finish – Start	---	---	Q>0	1

Table 9.4: List of edges and prompted properties. Ta and Ts are random variables that represent the arrival rate and service time

The single server Event Graph was then read and labelled as shown in Fig. 9.6.

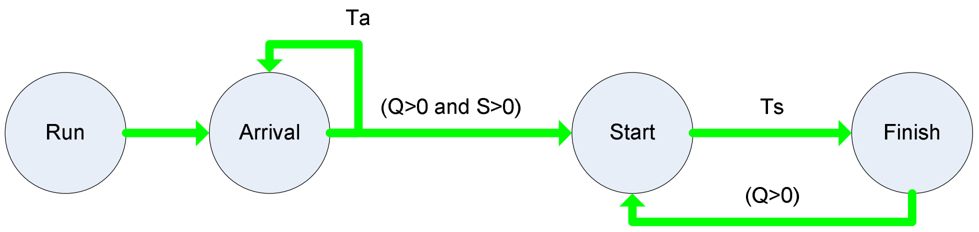


Fig. 9.6: Event Graph into which the single server system was modelled (C:\DotNetSim\WorkingFiles\SingSerSys.vsd)

Data on the probability distributions has now to be entered. By double clicking the edges labelled with the names of the random variables, the distribution parameters are prompted. Let us assume that the arrival rate is 8 arrivals per unit of time, the first arrival occurring after 3 units of time and a service time between 5 and 15 units of time.

9.4.1.2. DRAWING OTHER EVENT GRAPHS

To illustrate the use of the cancelling edges, we consider a single server system with failures [17], i.e. the server fails following a certain probability distribution. For example, the server is a computerised system that fails, causing critical errors following a probability distribution and has to be rebooted. Te and Tb are random variables representing the error times and the reboot times. The first failure is scheduled during the initialisation of the Event Graph. Fig. 9.7 shows the Event Graph devised within the DotNetSim modelling environment which models such a system.

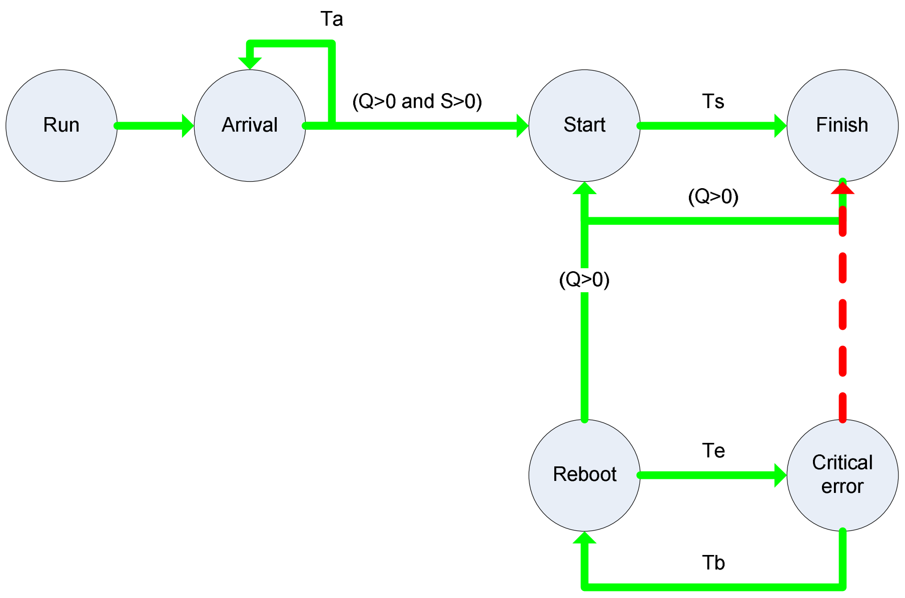


Fig. 9.7: Event Graph which models a single server system with failures (C:\DotNetSim\WorkingFiles\SingSerFailures.vsd)

Also, to illustrate the use of parameterised scheduling, a multiple server system [107] with N distinct servers is modelled. A service is allocated to the first server that is idle. The servers are numbered from 1 to N and their status is represented by one-dimensional array S[i] with $i \in \{0, 1, 2, 3, \dots, N-1\}$. The service time is represented by the one-dimensional Ts[i] with $i \in \{0, 1, 2, 3, \dots, N-1\}$ and may differ from server to server.

Fig. 9.8 depicts an Event Graph that models this system. The newly introduced event, Search, linearly looks for an idle server while an idle server (S[i]=1) has not been found and it is still possible to find one (i<N). Thus, the first idle server is allocated to the service. The parameter i represents the first idle server and passes through the parameterised scheduling edges.

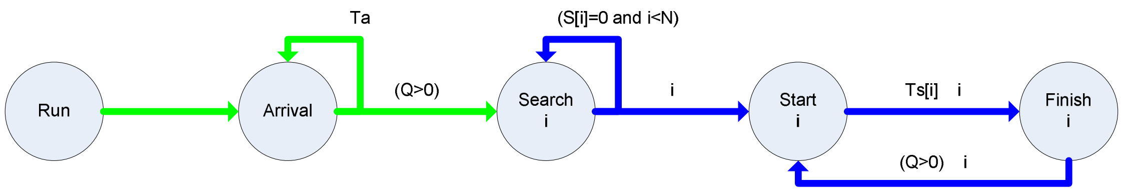


Fig. 9.8: Event Graph which models a server system with N distinct servers (C:\DotNetSim\WorkingFiles\MultSer.vsd)

9.4.2. DESCRIBING THE MODELS

General data on the DES systems and their dynamic behaviour is now to be entered (see section 6.2.4.) The command Describing the model splits this data into four groups, namely the general attributes, the state variables, the state transition and the simulation parameters. These are entered into user forms especially designed for this purpose.

9.4.2.1. DESCRIBING THE SINGLE SERVER

By executing the four sub commands listed above, we enter the following data (C:\DotNetSim\WorkingFiles\SingSerSys.vsd) :

(i) Model name: Single server system

Model description: Component of a DES system in which a single server provides a service on a FIFO basis.

(ii) State variables as in Table 9.5.

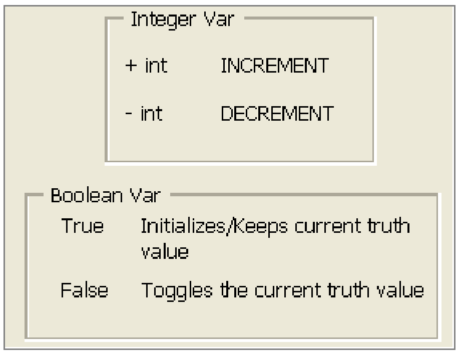
Variable	Description	Type	Max value
N	Number of arrivals	Int	500
Q	Length of the single queue	Int	500
S	Status of the server	Bool	1

Table 9.5: State variables for the single server system

(iii) State transition as in Table 9.6. The key for the state transitions is shown in Fig. 9.9 as it appears in the state transition's user form.

Event	N	Q	S	Interpretation
Run	0	0	True	Run initialises the number of arrivals and the queue length to 0, and sets the server as idle.
Arrival	+1	+1	True	A new arrival is enqueued, hence, it increments N and Q by one and keeps the server status unchanged.
Start	0	-1	False	Starting the service dequeues one element and toggles the server status. N is unchanged.
Finish	0	0	False	Finishing the service toggles the server status and does not affect N and Q.

Table 9.6: State transition triggered by each event of the single server system



Integer Var

+ int	INCREMENT
- int	DECREMENT

Boolean Var

True	Initializes/Keeps current truth value
False	Toggles the current truth value

Fig. 9.9: Key for entering the changes which the events trigger on the state variables.

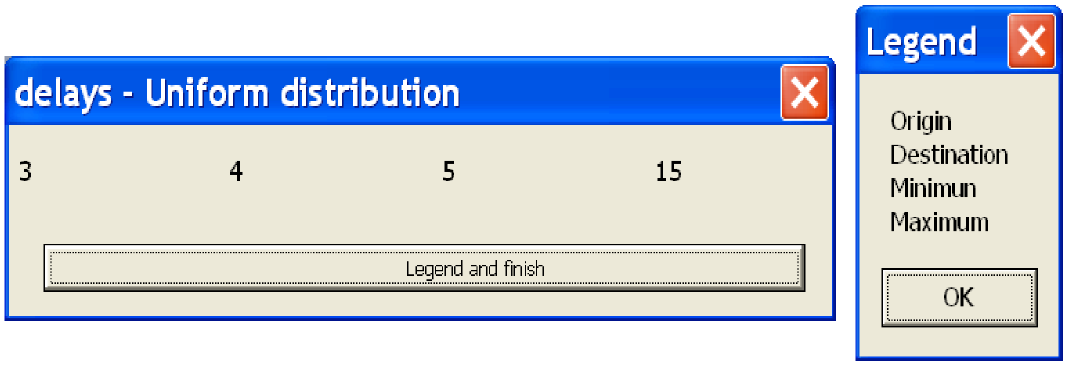
IV. Simulation parameters as in Table 9.7

Parameter	Value	Interpretation
Run length	100	Each simulation will emulate 100 units of physical time (e.g. 100 minutes)
Unit of time	Unit	The physical time is measured in units (e.g. Min).
Seed		The random number seed, i.e. the initial value for generating the streams of pseudo-random numbers. If the seed is omitted, the computer system's time, in milliseconds, is used to generate different stream numbers for each replication.
Replications	6	6 simulations will run in sequence.

Table 9.7: Parameters for simulating the single server system over time

9.4.3. REVIEWING THE MODEL

The single server Event Graph is now organised and automatically placed into a set of tables (see section 6.2.5.) which can be listed to review the model. By selecting Reviewing the model – Converting the model into a database - Visio tables Event Graphs menu we can, for example, list the scheduling edges through which passes a uniformly distributed delay time(as shown in Fig. 9.10). Given the objectives of the DotNetSim prototype's modelling component, the design of these outputs is not essential; hence, the legends are displayed in separate message boxes.



```
3            4            5            15
```

```
Origin
Destination
Minimun
Maximum
```

Fig. 9.10: An uniformly distributed delay time passes through the edge that links event 3 to event 4. The parameters are the minimum and maximum time

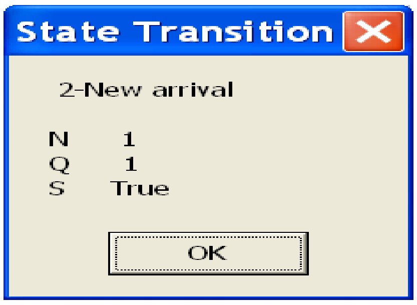
To facilitate the review of the model, these tables can be permanently displayed in an Excel workbook. This can be done in a three-stage process:

(i) Create an Excel workbook based on the EGmodel template (see C:\DotNetSim\Exceltemplates). This can be done by selecting Utilities – Excel files – Creating an Excel file on the Event Graphs menu.

(ii) Select Reviewing the model - Converting the model into a database – Excel tables on the Event Graphs menu.

(iii) Save the Excel workbook (see C:\DotNetSim\Workingfiles\SingSerMod).

A quick shortcut shows the state transition that each event triggers, i.e. by double clicking an event on the diagram, for example, the arrival, a message box pops up with the event number, description and the state transition it triggers (see Fig. 9.11)


State Transition

2-New arrival

```
N     1
Q     1
S     True
```

Fig. 9.11: State transition triggered by the arrival event

Reviewing the model – Highlighting variables changes is another reviewing feature that allows the user to trace a state variable along the Event Graph, i.e. the events throughout the diagram are coloured to show the changes they trigger on a state variable.

For example, Fig. 9.12 shows that the variable N, which represents the number of arrivals, is initialised by the Run event, incremented when an Arrival occurs and maintained unchanged by the other events.

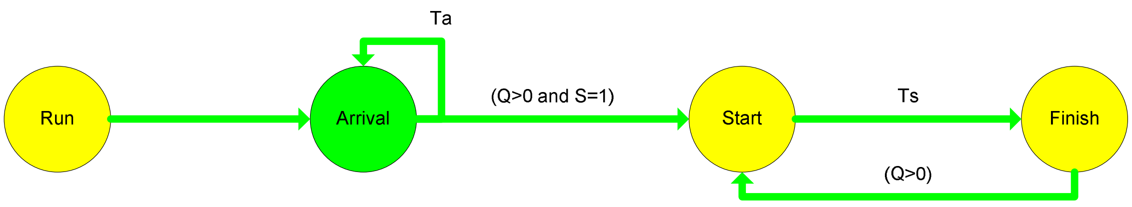


Fig. 9:12: Changes triggered on the number of arrivals in a single server system

9.4.4. REPORTING THE MODEL

The single server model is now to be reported (see section 6.2.6). The diagram can be converted into an Excel list, pasted into Word documents and PowerPoint presentations. The model can be summarised into an Excel workbook or a Word document.

9.4.4.1. REPORTING THE EVENT GRAPH

Listing the single server Event Graph in Excel^{TM} is a three stage process:

(i) Create an Excel workbook based on the EGdiagram template (see

C:\DotNetSim\ Exceltemplates). This can be done by selecting Utilities – Excel files – Creating an Excel file on the Event Graphs menu.

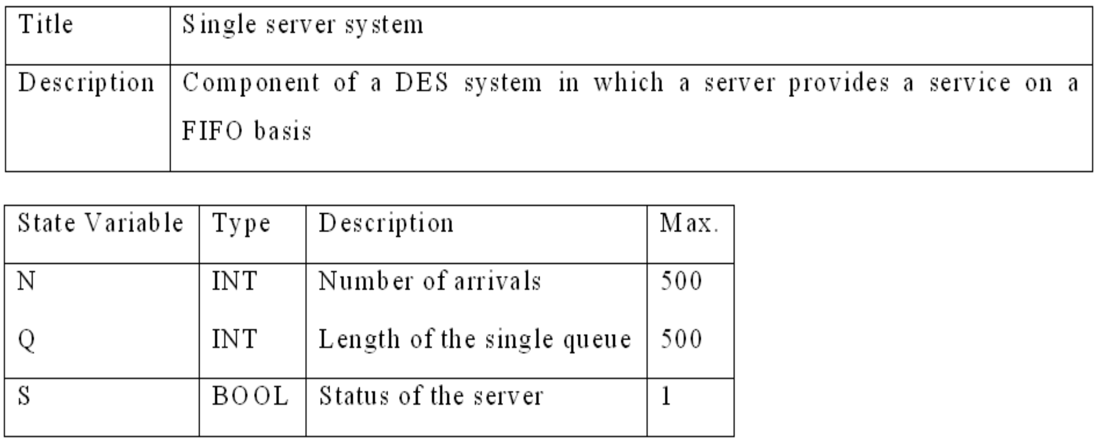
(ii) Select Reporting the model - Listing the diagram to Excel on the Event Graphs menu.

(iii) Save the Excel workbook (see C:\DotNetSim\Workingfiles\ SingSerDiag).

Pasting the Event Graph to Word documents is possible by selecting the Reporting the model - Pasting the diagram – To Word. For example, the subsequent New Word Document option pastes the single server Event Graph, to a new Word document. Similar usage pastes the single server Event Graph to PowerPoint presentations (see C:\DotNetSim\WorkingFiles\SingSerDiag)

9.4.4.2. SUMMING UP THE MODEL

Summarising the single server model into, for example, the active Word Document just requires the selection of Reporting the model - Summing up the model - To Word (see C:\DotNetSim\WorkingFiles\SingSerMod). Fig. 9.13 shows part of Word summary of the single server model.



Title	Single server system
Description	Component of a DES system in which a server provides a service on a FIFO basis

State Variable	Type	Description	Max.
N Q	INT INT	Number of arrivals Length of the single queue	500 500
S	BOOL	Status of the server	1

Fig. 9.13: Part of the summary of the single server model inserted into the active Word document (C:\DotNetSim\WorkingFiles\SingSerMod.Doc)

9.4.5. GENERAL OPERATIONS ON THE MODEL

Resizing or zooming the single server Event Graph and deleting the whole model (see section 6.2.7) is available through the General operations on the model command of the Event Graphs menu. Its subcommand Resizing diagram allows the user, for example, to define the length of edges and re-sizes the page width automatically. It also allows zooming out and in the diagram. For example, the edges of the single server event shown in Fig 9.14 were shortened to 1cm long.

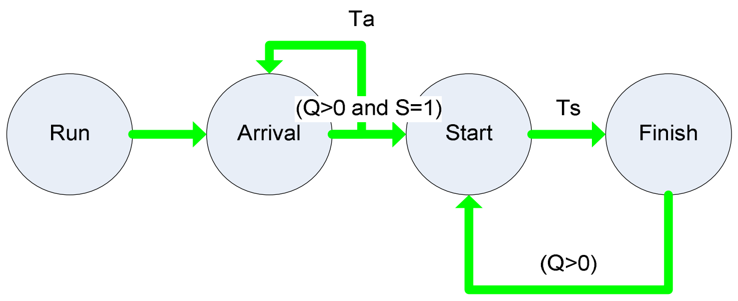


Fig. 9.14: The single server Event Graph with 1cm long edges

The commands General operations on the model - Resizing diagram – Pan and Zoom Window, allow, for example, the Arrival sub system to be selected in a Pan Window and zoomed as shown in Fig. 9.15.

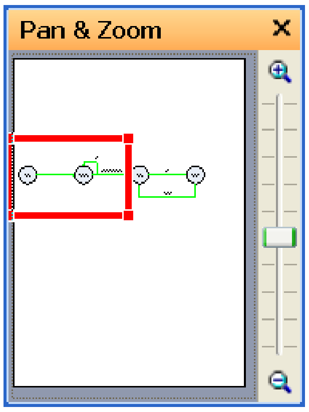


Fig. 9.15: Selecting and zooming in the Arrival subsystem

The commands General operations on the model - Resizing diagram – Zooming allows reduction or broadening of the view of the drawing page.

Deleting the single server model in order to re-start its definition is available by running commands General operations on the model – Deleting model. This is preceded by a confirmation of the operation.

9.5. SIMULATING DE MODELS

Given the data already specified, selecting Simulating the model on the Event Graphs menu will cause 6 simulation replications of the single server model to run, each simulating the model for 100 minutes. Control is transferred to the DotNetSim simulation engine (see section 7.2.) and a message informs the user that the simulation has started running. Eventually, the simulation results are placed on an instance of the SimOutAna Excel template for analysis (see section 7.2.3.).

Table 9.8 shows the system state at the end of each replication, as placed in the Replications worksheet (see singleserver.xls in appendix A) by the DotNetSim simulation engine. Each row shows the physical time, in milliseconds, that the system took to run the corresponding simulation, the last value assigned to the simulation clock and the values of the state variables (N, Q and S) at the end of each replication.

	A	B	C	D	E	F
1	Replication	Duration (millisec)	Current simulated time	N	Q	S
2	1	310	100	11	1	FALSE
3	2	280	100	8	0	TRUE
4	3	351	100	15	4	FALSE
5	4	300	100	12	3	FALSE
6	5	451	100	20	8	FALSE
7	6	411	100	15	3	FALSE

Table 9.8: Snapshot of the Excel worksheet where the DotNetSim simulation engine placed a summary of each replication (C:\DotNetSim\WorkingFiles\SingSerOut.xls)

Table 9.9 shows part of the sequence of events executed during the first 3 replications as placed in the Events worksheet by the DotNetSim simulation engine. Each pair of columns < Clock, Event> shows the simulated time at which the event, represented by its number, was executed. We are aware of the time that writing this data can take especially, when the simulation run length is long (see section 7.3.2.).

The state transition matrix, shown in Fig. 9.10, is also placed by the DotNetSim simulation engine in the Transition worksheet.

	A	B	C	D	E	F
1	Clock	Event	Clock	Event	Clock	Event
2	6	2	24	2	6	2
3	6	3	24	3	6	3
4	11	2	27	2	9	2
5	12	4	30	2	12	4
6	12	3	37	4	12	3
7	22	4	37	3	14	2
8	24	2	43	4	17	4
9	24	3	43	3	17	3
10	27	2	47	2	23	4
11	29	2	50	2	29	2
12	29	2	51	4	29	3
13	34	4	51	3	31	2
14	34	3	57	4	32	2
15	36	2	57	3	36	4

Table 9.9: Snapshot of part of the Excel worksheet where the DotNetSim simulation engine placed the sequence of the events executed in each replication (C:\DotNetSim\WorkingFiles\SingSerOut.xls)

	A	B	C	D	E
1			N	Q	S
2	1	Run	0	0	TRUE
3	2	Arrival	1	1	TRUE
4	3	Start	0	-1	FALSE
5	4	Finish	0	0	FALSE

Table 9.10: Transitions triggered by each event on the server component's state (C:\DotNetSim\WorkingFiles\SingSerOut.xls)

9.6. ANALYSING SIMULATION RESULTS

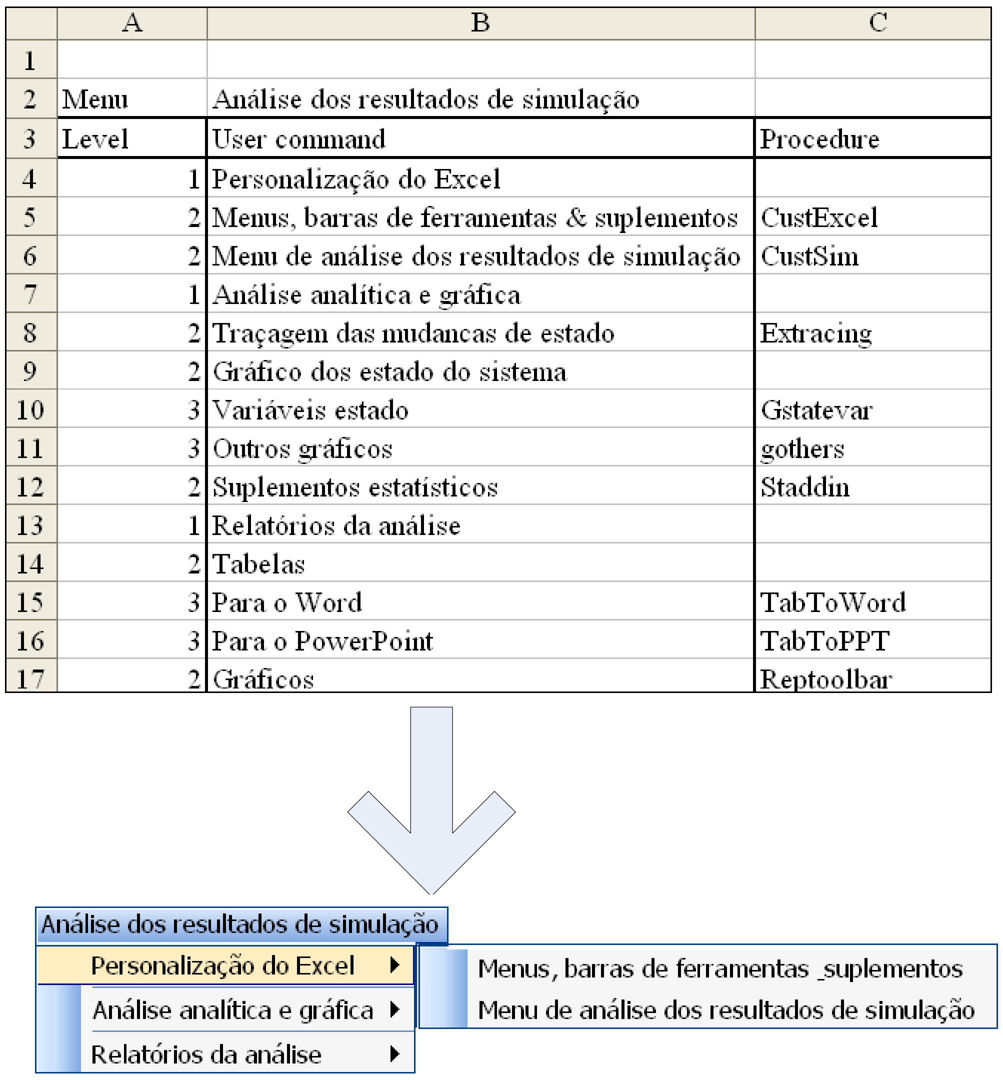
The simulation results placed on the Replications and Events worksheets of the SimOutAna's instance are now ready for analysis and reporting (see chapter 8). The states which the single server system passes through during the simulation runs can be reproduced and plotted into charts to check the correctness of the simulation results and gain insights into the system's dynamic behaviour. Statistical analysis can be carried out for further evaluation of the performance of the system.

Eventually, the analysis results are reported into Word documents and PowerPoint presentations.

9.6.1. SIMULATION OUTPUT ANALYSIS MENU

Prior to the analysis of the results, we may customise the Excel graphical user interface (see section 8.2.1.) to display the capabilities required by the output analysis of the single server system's simulation. Excel^{TM} starts up with only the File and Simulation Output Analysis menus, but the user can activate other menus, toolbars and add-ins by selecting the appropriate commands on the Simulation Output Analysis menu at any time (similar to section 9.3). For example, we may toggle the Excel tools menu, the formatting toolbar and the Analysis ToolPack add-in by selecting the commands Customising Excel – Menu, toolbars and add-ins on the Simulation Output Analysis

In addition, if, for example, we want to display the Simulation Output Analysis menu in Portuguese, we enter the new description of the user commands in the Customisation worksheet, select Customising Excel - Simulation Output Analysis menu on the Simulation Output Analysis menu and press the button New menu in the Customising Toolbar. Fig. 9.16 shows the simulation menu in Portuguese. By pressing the Restore built-in menu, the Simulation Output Analysis menu returns to its English version.



	A	B	C
1			
2	Menu	Análise dos resultados de simulação	
3	Level	User command	Procedure
4	1	Personalização do Excel	
5	2	Menus, barras de ferramentas & suplementos	CustExcel
6	2	Menu de análise dos resultados de simulação	CustSim
7	1	Análise analítica e gráfica	
8	2	Traçagem das mudancas de estado	Extracing
9	2	Gráfico dos estado do sistema	
10	3	Variáveis estado	Gstatevar
11	3	Outros gráficos	gothers
12	2	Suplementos estatísticos	Staddin
13	1	Relatórios da análise	
14	2	Tabelas	
15	3	Para o Word	TabToWord
16	3	Para o PowerPoint	TabToPPT
17	2	Gráficos	Reptoolbar

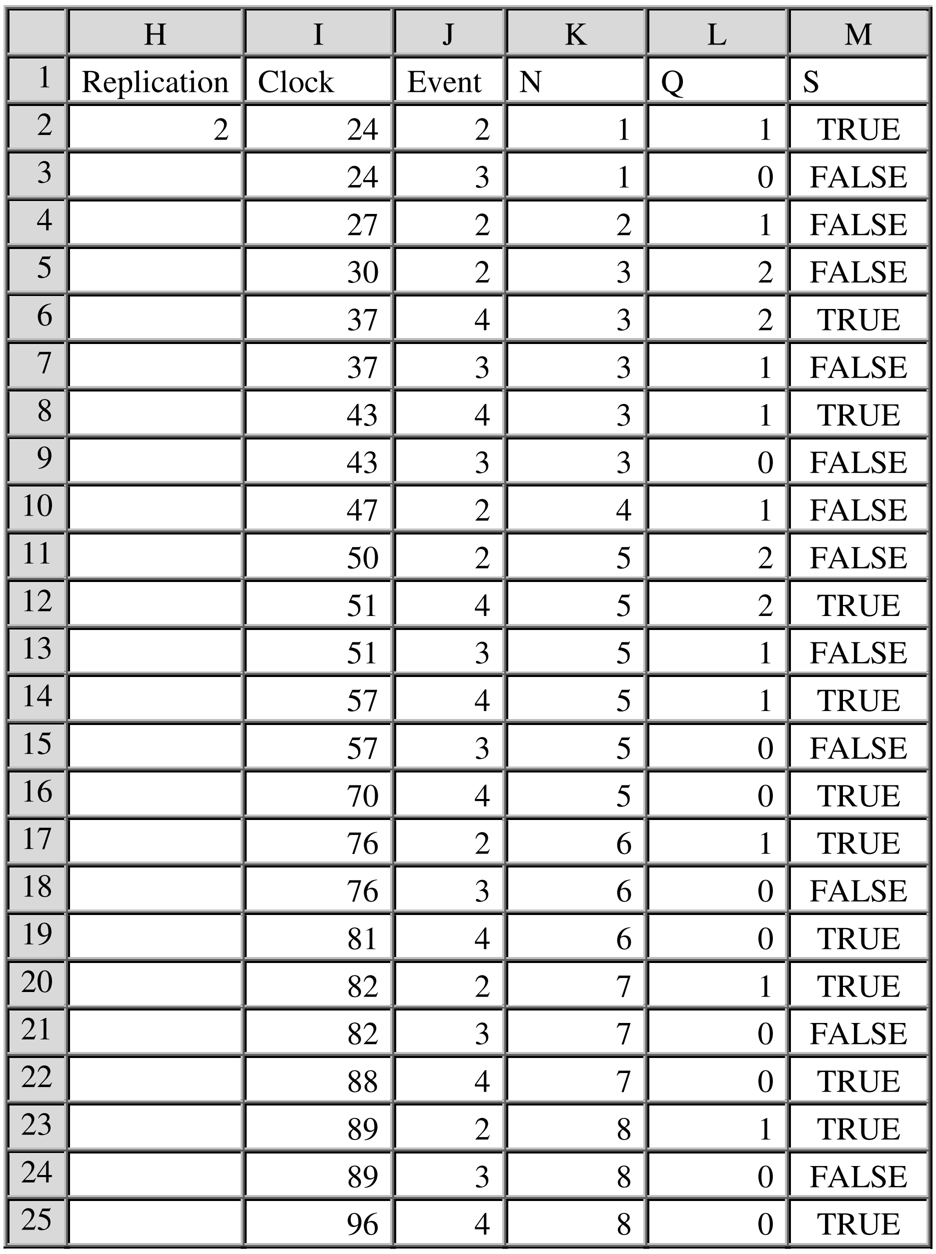
Análise dos resultados de simulação
Personalização do Excel
Análise analítica e gráfica
Relatórios da análise
Menus, barras de ferramentas _suplementos
Menu de análise dos resultados de simulação

Fig. 9.16: Generating the Simulation Output Analysis menu in Portuguese

9.6.2. ANALYSING THE SYSTEM'S STATE TRANSITION

The states through which the single server system has passed during one simulation may be reproduced by tracing the state variables after the execution of each event. For example, tracing the state transition of the single server system during the second replication is possible by selecting the commands Analytical and graphical analysis – tracing state changes on the Simulation Output Analysis menu. By entering the number of the replication and selecting the variables to be traced, the values taken by these variables during that replication are automatically displayed on the Transition worksheet (see section 8.2.2.1.).

Table 9.11 traces all the state variables during the during the second simulation run. The data is automatically offset three columns to the right of the state transition matrix.



	H	I	J	K	L	M
1	Replication	Clock	Event	N	Q	S
2	2	24	2	1	1	TRUE
3		24	3	1	0	FALSE
4		27	2	2	1	FALSE
5		30	2	3	2	FALSE
6		37	4	3	2	TRUE
7		37	3	3	1	FALSE
8		43	4	3	1	TRUE
9		43	3	3	0	FALSE
10		47	2	4	1	FALSE
11		50	2	5	2	FALSE
12		51	4	5	2	TRUE
13		51	3	5	1	FALSE
14		57	4	5	1	TRUE
15		57	3	5	0	FALSE
16		70	4	5	0	TRUE
17		76	2	6	1	TRUE
18		76	3	6	0	FALSE
19		81	4	6	0	TRUE
20		82	2	7	1	TRUE
21		82	3	7	0	FALSE
22		88	4	7	0	TRUE
23		89	2	8	1	TRUE
24		89	3	8	0	FALSE
25		96	4	8	0	TRUE

Table 9.11: State transitions of the single server system during the second simulation run (C:\DotNetSim\WorkingFiles\SingSerOut.xls)

The commands Analytical and graphical analysis – Plotting system's states – state variables on the Simulation Output Analysis menu plot these data in a Bar chart.

Fig. 9.17 shows a Bar chart that depicts the state variables of the single server system during the second simulation run.

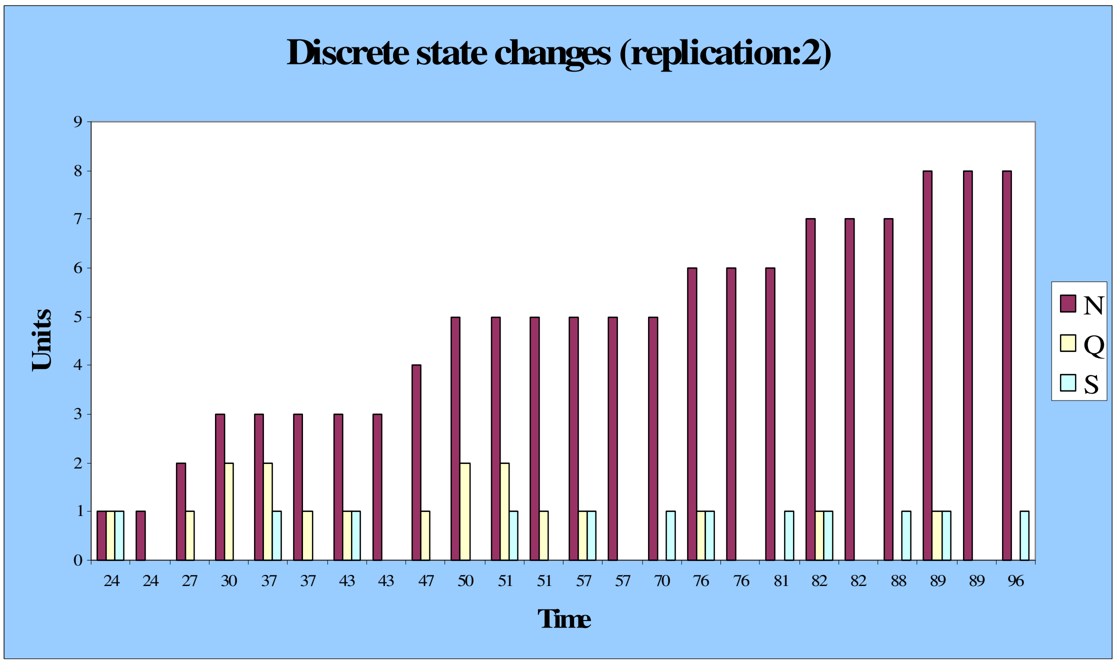


Fig. 9.17: Tracing the state variables of the single server system during the second simulation run (C:\DotNetSim\WorkingFiles\SingSerOut.xls)

This automatic chart can then be enriched by resorting to the chart's functionality built into the Excel^{TM}.

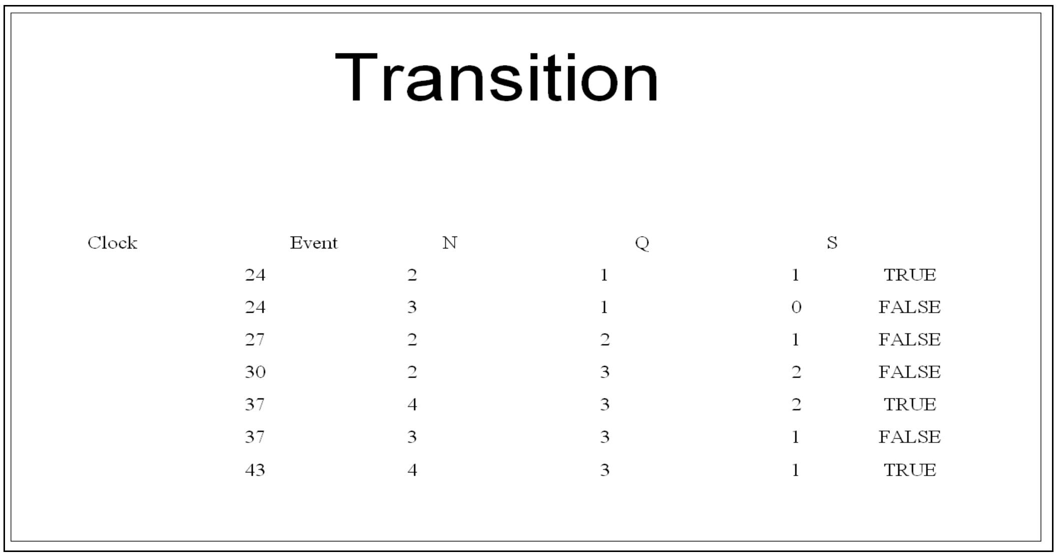
9.6.3. FURTHER ANALYTICAL AND GRAPHICAL ANALYSIS

Existing or newly developed user-defined functions and procedures can be gathered in Excel add-ins to support the analytical and graphical analysis of the simulation results (see section 8.2.2.2.). The add-ins may then be loaded when they are required. For example, we may gather, in the Simulation.xla, functions and procedures which calculate statistical measures such as the average system time required to run a simulation, the average number of events executed per run, the average number of arrivals, etc. This add-in can then be loaded and unloaded by selecting Analytical and graphical analysis–statistical add-ins on the Simulation Output Analysis menu. If it is the first time that this add-in is to be used, its full name C:\DotNetSim\WorkingFiles has to be typed on the prompted input box.

The simulation.xla functions and procedures can now execute. For example, to compute the average number of arrivals, the user has to execute the user-defined function Averages() and type the name of the state variable that represents the arrivals.

9.6.4. REPORTING THE SIMULATION ANALYSIS

The analysis of the outputs from the simulation of the single server system can be reported by inserting Excel tables and charts into Word documents or PowerPoint presentations. For example, subsets of the states which the single server system passed through during the second replication (see table 9.11) can be pasted into a PowerPoint slide, selecting the required elements and Reporting simulation analysis – Tables – To PowerPoint on the Simulation Output Analysis menu.



Transition

Clock	Event	N	Q	S	
	24	2	1	1	TRUE
	24	3	1	0	FALSE
	27	2	2	1	FALSE
	30	2	3	2	FALSE
	37	4	3	2	TRUE
	37	3	3	1	FALSE
	43	4	3	1	TRUE

Fig. 9.18: Snapshot of the PowerPoint slide created from within the Singleserver.xls

Also, the bar chart that plots the state variables along the second replication of the single server system can be pasted into the active or a new Word document by selecting Reporting simulation analysis – Charts on the Simulation Output Analysis menu, selecting the chart and pressing the To Word button of the Reporting graphs toolbar. Similarly (with the appropriate VBA statements), this bar chart can be pasted to PowerPoint slides.

9.7 THE CHAPTER IN CONTEXT

This chapter demonstrates the use of the DotNetSim prototype whose three coarse-grained components were discussed on the previous 4 chapters. A single server system is modelled and simulated within the DotNetSim prototype to demonstrate the functionality each component implements. Thus, the single server system is modelled in an Event Graph within the DotNetSim modelling component by following the menu driven modelling commands. Simulations of the single server model are then run by the DotNetSim simulation engine, which places the results in an instance of DotNetSim output analysis environment. The illustrative tools of the DotNetSim output analysis tools are then used to analyse and report the simulations results.

The next chapter summarises the thesis, reviews the major arguments and underlying research to enumerate and comment on its contributions to an alternative development strategy for DES software. This leads to a consideration of possible further research.