INTRODUCTION TO DISCRETE EVENT SIMULATION

Delčo Jovanoski[1], **Gabriela Kostovska**[2]

[1] *University Ss. Cyril and Methodius, Faculty of Mechanical Engineering, Karpos II b.b.,*

Abstract: The paper describes some of the elementary principles of the discrete event simulation and goes on to discuss more general simulation issues that are of wider interest, such as the power of object-oriented simulation software packages, such as *eM-Plant* specialized simulation software. The paper concludes with observations on simulation experimentation and analysis of the results which serves as a basis for decision making processes.

Keywords: simulation modelling, discrete-event simulation, object-oriented, hierarchy model, object, simulation run, simulation experimentation.

1. Introduction

Modelling of systems, such as manufacturing systems, can be achieved using a number of tools and techniques one of which is simulation. The Oxford English Dictionary describes simulation as: "*The technique of imitating the behaviour of some situation or system (economic, mechanical etc.) by means of analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel."* Or put another way, simulation is the technique of building a model of a real or proposed system so that the behaviour of the system under specific conditions may be studied. One of the key powers of simulation is the ability to model the behaviour of a system as time progresses.

Discrete event simulation is one way of building up models to observe the time based (or dynamic) behaviour of a system. There are formal methods for building simulation models and ensuring that they are credible. During the experimental phase the models are executed (run over time) in order to generate results. The results can than be used to provide insight into a system and a basis to make decisions on.

2. Key principles of discrete event simulation

The process of building simulation models invariably involves some form of software. The software could either be a high level programming language or a specialized simulation software package in which the model is specified using default and user-defined data items.

Inside the software or model will be a number of important concepts, namely entities and logic statements. Entities are tangible elements found in the real world, e.g. for manufacturing these could be machines or trucks. The entities may be either temporary (e.g. parts that pass through the model) or permanent (e.g. machines that remain in the model). The concepts of temporary and permanent are useful aids to understanding the overall objective of using simulation, usually to observe the behaviour of the temporary entities passing through the permanent ones.

Logical relationships link the different entities together, e.g. that a machine entity will process a part entity. The logical relationships are the key part of the simulation model; they define the overall behaviour of the model. Each logical statement (e.g. “start machine if parts are waiting”) is simple but the quantity and variety and the fact that they are widely dispersed throughout the model give rise to the complexity. Another key part of any simulation system is the simulation executive. The executive is responsible for controlling the time advance. A central clock is used to keep track of time. The executive controls the logical relationships between entities and advance the clock to the new time. The process is illustrated in Fig. 1.

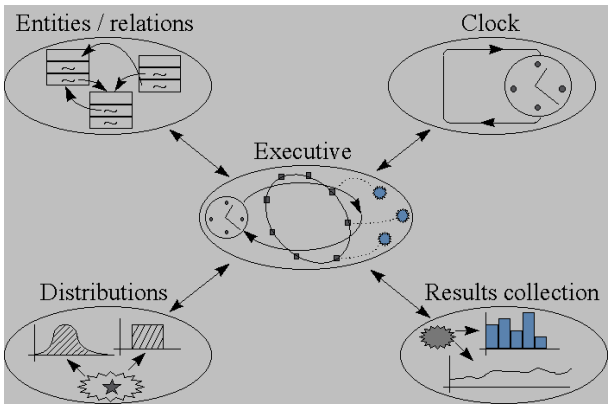

Fig. 1 Structure of a simulation system (adapted from Kreutzer, 1986)

The simulation executive is central to providing the dynamic, time based behaviour of the model. Whilst the clock and executive are key parts of a simulation system they are very easy to implement and are extremely simple in behaviour.

Two other elements that are vital to any simulation system are the random number generators and the results collation and analysis. The random number generators are used to provide stochastic behaviour typical of a real world. For example, machine scrap rates will rarely be fixed but will vary between certain ranges hence the scrap rate of a machine should be determined by a random distribution (probably a normal distribution).

The results collation and display provides the user a means of utilising the simulation tool to provide meaningful analysis of the new or proposed system. Simulation tools will typically display tabulated raw results and possess some graphing capabilities.

2.1. Mechanisms for time advancing

One of the central functions of a simulation system as described earlier is the simulation executive. The executive manages the passage of time and 'steps' the model into the future, executing the relevant logical relationships along the way. There are two basic approaches for controlling the time advance [2]:

- Time slicing
- Next event

The time slicing approach advances the model forward in time at fixed intervals, e.g. every 5 seconds. The executive moves the model between the time intervals regardless of whether anything will happen. With the next event approach the model is advanced to the time of the next significant event. Hence if nothing is going to happen for the next 3 minutes the executive will move the model forward 3 minutes in one go. The nature of the jumping between significant points in time means that in most cases the next event approach is more efficient and allows models to be evaluated more quickly.

There is a word of warning when using next event simulation software. Simulation software invariably has graphical displays to show the user the changing status of machines (running, idle, etc.) and the movement of parts. Because the software jumps between significant points in time, the jumps may be uneven with many jumps separated only by 5 seconds of simulated time followed by one or two jumps of 4 minutes say. The effect is that the series of snap shots shown by the graphical displays can be misleading and machines may appear broken down for long periods of time when in fact this is not the case. The situation is analogous to watching a video that is continually being speeded up and slowed down.

2.2. Mechanisms for describing logic

There are a number of different ways of representing the logic within a discrete event simulation model. These approaches can be used for modelling the same systems and will obtain the same results. The differences lie in the ease by which they can be understood and implemented as well as the efficiency of their computation. Three mechanisms will be briefly described followed by detailed explanation of one of them (see Pidd, 1992 for more details).
The approaches illustrated in Fig. 2 are: event, activity and process.

The event approach describes an event as an *instantaneous* change and such events are usually paired, e.g. start of machine loading, end of machine loading, etc. Activities describe *duration*, e.g. machine loading, and are therefore very similar to pairs of events. The process approach joins collections of events or activities together to describe the life cycle of an entity, in this case a machine.

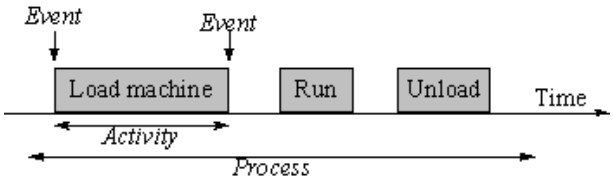

Fig. 2 Ways of describing model logic

The event approach is easy to understand and computationally efficient but is more difficult to implement than the activity approach. On the other hand whilst activity approach is relatively easy to understand it suffers from poor execution efficiency. The process is less common and requires more planning to implement properly though is generally thought to be efficient.

2.3. Example of the mechanism working

Fig. 3 illustrates the next event mechanism. The rows show the advance of time for a simple model involving one machine (cycle time 5) feeding a buffer followed by another machine (cycle time 12) that removes parts from the buffer to process them. Parts arrive every 6 time units. The units of time could be seconds, minutes, hours, etc. depending on the model.

Clock time	*Event*	Machine 1	Buffer	Machine 2	Output
0	-	*idle*	-	*idle*	0
6	*1st part arrives*	**busy**	-	*idle*	0
11	*part to buffer*	*idle*	1	*idle*	0
11	*buffer to m/c2*	*idle*	-	**busy**	0
12	*2nd part arrives*	**busy**	-	**busy**	0
17	*part to buffer*	*idle*	1	**busy**	0
18	*3rd part arrives*	**busy**	1	**busy**	0
23	*part to buffer*	*idle*	2	**busy**	0
23	*part complete*	*idle*	2	*idle*	**1**
23	*buffer to m/c2*	*idle*	1	**busy**	1
24	*4th part arrives*	**busy**	1	**busy**	1

Fig. 3 Passage of time in next event simulation

The model starts from the common starting point know as ‘empty and idle’; all entities are idle and there are no parts in the system. The next most significant time is 6 when the first part arrives. The executive jumps straight to this time. When the first part arrives the first

machine starts processing it. At time 11 (5 units later) the executive will cause the first machine to place its processed part in the buffer. Immediately the second machine takes the part and starts processing it. The events may occur at the same time, as well as there being significant times between events. The model unfolds over time with parts arriving, being processed on machine1 and placed in the buffer. As would be expected parts accumulate in the buffer since machine2 is slower.

For a graphical display the machines would be shown as icons changing colour when running. According to a graphical display it would appear that machine2 is busier than machine1. If the figures for the busy time are added up for each machine (machine1: 16 vs. machine2: 13) it is apparent that machine1 was busier. This is one of the problems noted in section 2.1. that can occur when the graphical displays of next event simulation are taken too literally.

Note that in a practical situation variations would be attached to many aspects of the model (cycle time variations, machine breakdowns, scrap, machine efficiency variations, etc.) and that the model would be simulated for significantly longer to be able to have confidence that the results are credible.

3. Object-oriented simulation

Object-oriented techniques have received a lot of publicity in recent years and the use of these techniques is becoming increasingly common. Interestingly the roots of object-oriented techniques can be traced back to the *SIMULA* simulation programming language developed in the 1960's [4].

The power of object oriented techniques lie in the ability to produce 'modular' code (known as classes) that can be "easily" modified and reused [5]. Libraries of the classes can be built up and used to create simulation models. The ability to contain software complexity into classes and to be able to realistically represent entities from the real world in computer model makes object-oriented techniques ideally suited to simulation which is inherently complex.

There are many object-oriented simulation packages available commercially; one such package founded on object-oriented principles and *Simple++* programming language is the ***eM-Plant*** simulation software package.

4. Building a model using *eM-Plant*

eM-Plant is standard software for object-oriented, graphical and integrated modelling for discrete event simulation and visualization of systems and business processes. *eM-Plant* is an object-oriented program so the user may enjoy all advantages of hierarchical structures, inheritance, polymorphism and the reuse of objects. This software package enables the simulation and optimization of production systems and processes. With eM-Plant user can optimize the material flow, resource utilisation and logistics for all levels of plant planning, from global production facilities, through local plants, to specific lines.

eM-Plant allows the creation of computer models of logistic systems (e.g. production) to explore the systems' characteristics and optimize performance. The computer model enables users to run experiments and what-if scenarios without disturbing an existing production system – or when used in the planning process – long before the real system is installed. Extensive analysis tools, statistics and charts let users evaluate different manufacturing scenarios and make fast, reliable decisions in the early stages of production planning.

4.1. Modelling manufacturing processes using libraries of standard and specialized components

eM-Plant enables creation of well-structured, hierarchical models of production facilities, lines and processes. This is achieved through its powerful object-oriented architecture and modelling capabilities. Simulation models can be created in short time by using predefined objects from the following library classes:

- Material flow: containing different material flow objects, from processing units and buffers, through simulation clock, to movable units (e.g. entities and transporters)
- Information flow: containing objects like tables, variables, methods and generators.
- User interface: provides objects for graphical and visual presentation of the simulation results as well as for creating user defined dialogs.

The functionality of the library can be extended by creating user-defined objects which will incorporate certain best practices and allow closer conceptualisation of the real system being modelled. Furthermore, users can load objects from variety of add-in programs compatible with *eM-Plant* as well as import CAD models.

The key advantage of this simulation package is that has built in programming language, *SimTalk*, which extends the ways of modelling considerably. Each object has basic properties providing many useful features. Depending on the model, there might be a need for more detailed or completely different properties. Those can be programmed by the user in the

programming language *SimTalk*. This feature of *eM-Plant* combines productivity and flexibility. The instructions that the program is to execute are entered into the object *Method* that is fully integrated into *eM-Plant*. In addition, *Method* object can be combined with other material flow objects to form models of great complexity.

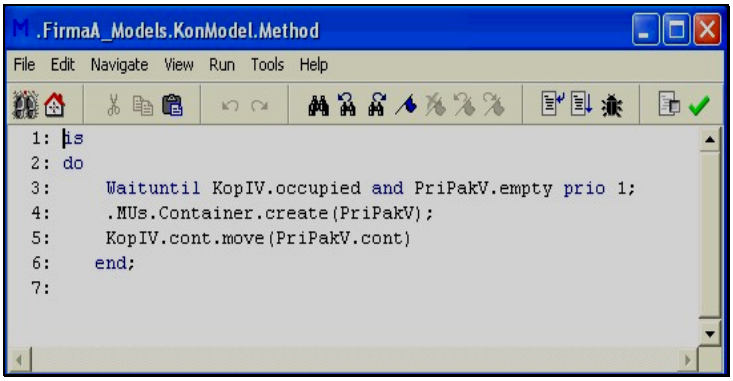

Fig. 4 Programming in *eM-Plant*

SimTalk provides control structures for defining and managing the behaviour of the different entities in the model and once again contribute to the approximation of the model to the existing or planned real system.

On Fig. 5 an example of a model build in *eM-Plant* is being presented. The model is created, simulated and analysed in the scope of a larger research on the features and practical utilisation of the *eM-Plant* simulation package.

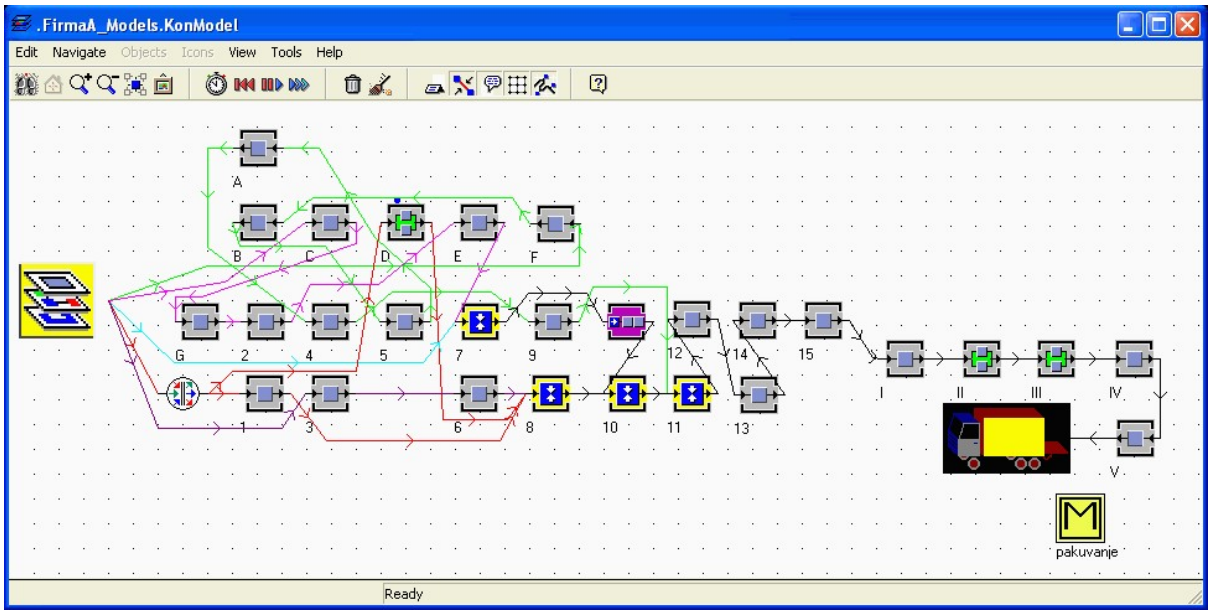

Fig. 5 *eM-Plant* model of a processing line

4.2. Simulation runs and animation

Each simulation in *eM-Plant* requires an *EventController* object which coordinates and synchronizes the different events taking place during simulation run. When a movable unit enters a station *eM-Plant* computes the time it takes to process it is calculated and enters it into the *EventController*. The *EvenController* moves along the time line and interprets messages relating to the event entered (refer to section 2.1). After the processing time has elapsed, the *EventController* passes on to the station that has to initiate an exit event. The movable unit is than moved on to the successor. There again, the exit time is calculated and passed to the *EventController*. This process is repeated cyclically, for all movable units in the simulation model.

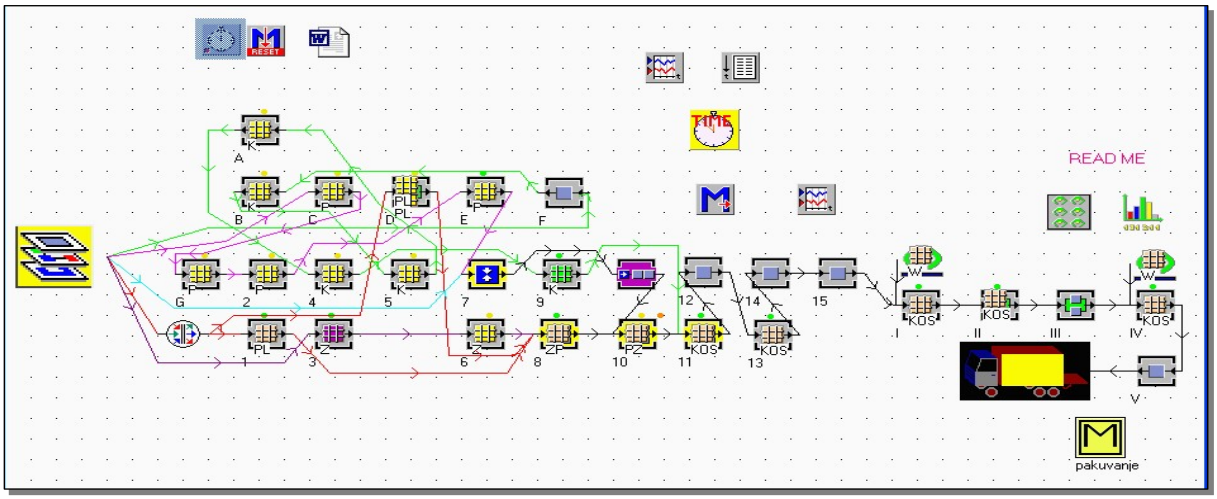

Fig. 6 Snapshot of *eM-Plant* model in a simulation run

Execution of the first simulation run can be regarded as an opportunity for testing the functionality of the complete model. While building the model test simulations are being executed, but only on limited parts of the model.

4.3. Statistics collection

A typical task of simulation runs is to determine the distribution of failure times, waiting times or blockages, and material flow throughout the processing stations. When interpreting the simulation run, statistical data collected by the objects can be utilized. The standard statistics of objects in *eM-Plant* provide data to enable making valid statements concerning the process considered.

During a simulation run a resource is either paused or not paused. When not paused, the resource is either available or not available. Not available signifies that a resource is either

failed or blocked. In the available state the resource is either processing or waiting. So, altogether there are five non-overlapping states for a resource. The statistics collection period is defined as sum of these five states.

In addition, it is possible to visually determine the resource state during the simulation run due to the animation feature built in the program. Namely, when the animation mode is active a 'lamp' occurs on each station's icon indicating the current state. This indicating lamp can be in one of the following six colours: red ("failed"), blue ("paused"), green ("working"), yellow ("blocked"), brown ("setup"), and light blue ("no_entry").

For graphical presentation of the collected statistical data, the use of the interface objects is recommended. Most commonly used are *Chart* and *Plotter* objects. By using such display objects the change of the considered data can be followed as time progresses (dynamic presentation).

eM-Plant is discrete simulation system. It is based on a variety of different times, such as processing times, machine down times, etc. The program allows true to life modelling with fixed times as well as with statistical distribution times.

5. Simulation experimentation and results

The area of experimentation and results analysis for simulation models will be briefly introduced here. The area is well developed and a range of rules and guiding methods can be found in the literature, e.g. [2, 6, 7, 8]. Many of the techniques developed are there to ensure that dangerous mistakes are not made when analysing and interpreting the results. Put simply the power of modern simulation software to generate large quantities of data can leave the user with the false sense of security that the results generated are credible and truly representative of the system under study. Like all modelling techniques care needs to be exercised. There are a number of phases for checking a simulation model prior to experimental analysis:

- Verification - "… the accuracy of transforming a problem formulation into a model (specification) … deals with building the model right … "
- Validation - "… model behaves with satisfactory accuracy consistent with the study objectives … deals with building the right model"

The above are essential checks performed prior to analysis of the model and are used to establish what is known as model credibility. In addition and in order to gain confidence that the results being compiled represent the average and the range of conditions that are likely, the model is run several times. Each time the random number generators are set to provide

different sequences of random numbers, e.g. the breakdown patterns of machines are different and the points at which material is scrapped is different.

6. Conclusion

This paper has described some of the basic concepts relevant to newcomers to the area of discrete event simulation and other relevant areas such as object-oriented simulation (the example being *eM-Plant*) were described. The use of discrete event simulation for modelling manufacturing systems (and other systems) has shown benefits to many companies. The evidence for this is both anecdotal and from that presented in the engineering literature.

7. References

[1] W. Kreutzer: *Systems Simulation – Programming Styles & Languages,* Addison-Wesley, 1986.

[2] J. Banks, J.S. Carson, B.L. Nelson: *Discrete-Event System Simulation,* Prentice-Hall Inc. 1996, pp. 59-87.

[3] M. Pidd**:** *Computer Simulation in Management Science***,** Wiley, 1992.

[4] G. Booch: *Object oriented design with applications,* Benjamin/Cummings Publishing Company, Inc. 1991.

[5] J.P. Shewchunk, T.C. Chung: An approach to object-oriented discrete event simulation of manufacturing systems, *Proceedings Winter Simulation Conference*, Phoenix, Arizona, USA: IEEE: 302-311 (1991).

[6] A.C. Chung: *Simulation Modeling Handbook: A Practical Approach,* CRC Press LLC. Boca Raton, Florida, US, 2004, pp.160-233.

[7] M. Laguna, J. Marklund: *Business Process Modelling, Simulation and Design,* Prentice-Hall, N. J., 2005, pp. 326-366.

[8] S. Robinson: *Simulation: The Practice of Model Development and Use,* John Wiley & Sons Ltd, England, 1964, pp. 137-198.