**ANNALS of the ORADEA UNIVERSITY.**
**Fascicle of Management and Technological Engineering, Volume VII (XVII), 2008**

# SIMULATION OF QUEUES IN MANUFACTURING SYSTEMS

**Marius NICA[1], Lucian, Macedon GANEA[2], Gheorghe DONCA[1],**
[1]. eng., drd., University of Oradea, [2]. prof., PhD., eng., University of Oradea
e-mail: mariusnica@yahoo.com

**Keywords** : queues, automated manufacturing lines, simulation softwares.

**ABSTRACT :** Using of modeling and simulation in manufacturing and assembling scheduling is a necessity required by the fact that these systems have to satisfy more and more divers queries from the market, like widening of product ranges, increasing quality and precise delivery time. A queue requires the setup of a space and the design of equipments in which the parts can be stored. In these cases the simulation can help in definig the dimensions of the storage spaces and in establishing the required equipments.

## 1. INTRODUCTION.

In the case of FMS parts that are going to be manufactured or assembled are not entering the system in a definite order, there is no prior knowledge of what kind, and how many parts will enter the system in a time interval. Also, the parts which are entering the system will follow different technological paths, producing different loads on the system servers (manufacturing or assembly units). There is a high probability that in some time intervals some servers are stopped and in front of others, queues can be formed.

A queue requires the setup of a space and the design of equipments in which the parts can be stored. In these cases the simulation can help in definig the dimensions of the storage spaces and in establishing the required equipments.

A discrete-event simulation, or event-based simulation, permits the system's state transitions to depend on asynchronous discrete incidents that are called events. By contrast, a simulation based solely on differential equations in which time is an independent variable is a time-based simulation because state transitions depend on time.

A detailed description and precise definition of discrete-event simulation are beyond the scope of this paper, more details can be find in [2] or [4].

In a discrete event simulation, there are some specific notions which have to be defined, among which the most important are : entities, events, queues and servers.

## 2. MODEL DESCRIPTION IN DISCRET-EVENT SIMULATION.

Discrete-event simulations involve discrete items of interest, which in an assembly line can be parts or sub-assemblies. These items are called entities. Entities can pass through a network of queues, servers (robots, CNC machines), gates (barriers and manipulators), and switches during a simulation. Each entity have its own attributes. In a discrete-event simulation, an event is an instantaneous discrete incident that changes a state variable(for example the arrival of a part in the system), an output, and/or the occurrence of other events. Events in a simulation can depend on each other. One event might be the cause of another event. For example, the arrival of the first part in a queue causes the queue length to change from 0 to 1.

One event might enable another event to occur, but only under certain conditions. For example, the completion of surface on a part enables the part's departure from the machne tool, but only if the subsequent machine tool is able to accept the arrival of that part. In this case, one event makes another event possible.

In case of simulations on a single processor machine, events that occur at the same time are called simultaneous events, even if they are processed sequentially. When simultaneous events are not causally related to each other, the processing sequence can significantly affect the simulation behavior [1],[3].

A queue stores entities for some time that cannot be determined in advance. The queue attempts to output entities as soon as it can, but its success depends on whether the next processing station accepts new entities.

The parameters of a queue include:

- the queue capacity, which is the number of entities the queue can store simultaneously;
- the queue rule, which determines which entity departs first if the queue stores multiple entities.

In a discrete-event simulation, a server stores entities for some length of time, called the service time, and then attempts to output the entity. During the service period, the block is said to be serving the entity that it stores. The service time for each entity is computed when it arrives, which contrasts with the inherent unknowability of the storage time for entities in queues. If the next block does not accept the arrival of an entity that has completed its service, however, then the server is forced to hold the entity longer or to pass the entity to another storage server which is used as a buffer.

Parameters of different servers include:

- the number of entities it can serve simultaneously, which could be finite or infinite;
- method of computing the service times of arriving entities.

A finite-capacity server does not accept new arrivals when it is already full. A queue can be placed before each finite-capacity server, establishing a place for entities to stay while they are waiting for the server to accept them.

Discrete event systems can be modeled in many ways, for example with Petri Nets or in a newer version of MATLAB program which has a toolbox named SimEvents. For the following application we tried to establish whether the SimEvents toolbox is suitable for modeling the presented system. SimEvents toolbox has a great variety of blocks which can model events, entities, servers and queues.

As an example of discrete-event model with SimEvents Toolbox [5], in figure 1. there is showen a system with infinite storage capacity and five identical servers. In the notation, the M stands for Markovian; M/M/5 means that the system has exponentially distributed interarrival and service times, and five servers.

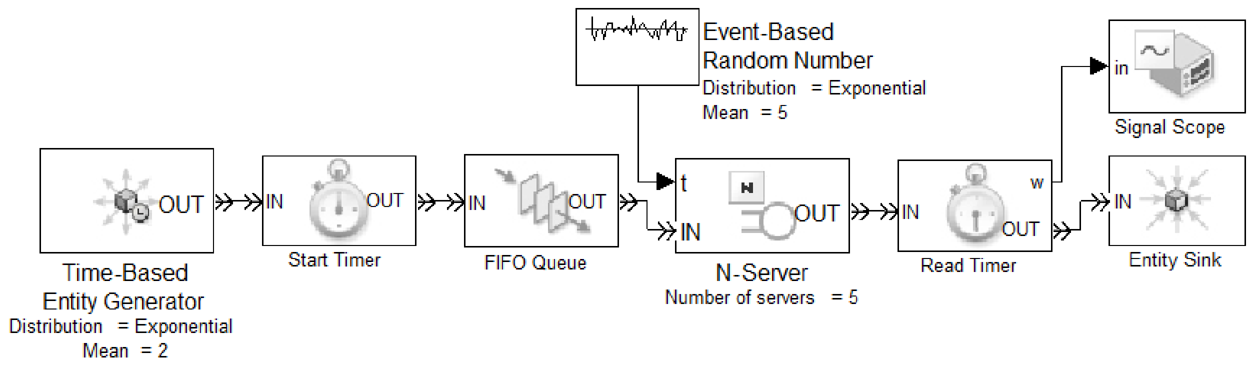

***Fig. 1. M/M/5 queue model in MATLAB SimEvents environment.***

## 3. CASE STUDY OF A ROBOTIC ASSEMBLY LINE

The application which is to be studied is presented in figure 2. This type of assembly system can be found in different industries such as machine building or in “End of line” applications in packaging systems field.

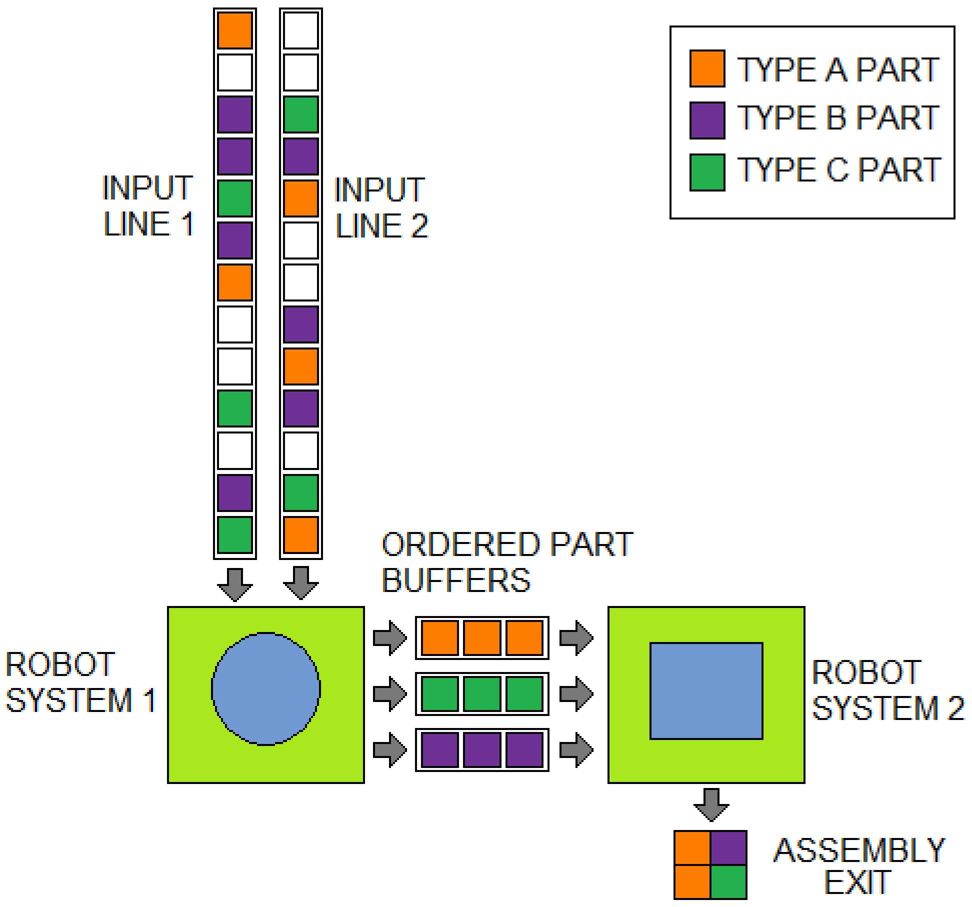

*Fig. 2. Robotic assembly system diagram.*

In a machine building application, the parts are entering the system from two input lines in a random order. The first server “ROBOT SYSTEM 1” is equipped with a camera and vision system which can recognize the type of part which is input. In the diagram the parts are represented in colors but they can be different as shape and dimensions. After recognition the robot from this system places the parts in three different buffers, ordered by their type. The second server “ROBOT SYSTEM 2” will take the parts from the buffers and assembles them in an assembly which exits the system. The advantage of this configuration is that the recognition and ordering are made simultaneously with the assembling and the processing time for the system is much lower than for a single robot system which would make the recognition and the assembly in the same server.

In a packaging application the entities will represent products and the task is to make groups of products which have to be packed in a box. This task can be done with the same system.

This application must be modeled in order to help the designer to establish the parameters for the robot systems, what time ratio is required in order to synchronize the two robotic systems, how long the input lines must be, and so on.

Our goal, in this paper is to establish if the SimEvents toolbox have suitable blocks and modeling possibilities to simulate this type of assembly system.

Identification of different application components with the SimEvents blocks is presented in the followings.

At first we have to generate the parts or products which are input randomly on the two input lines. A possibility to do this is to use the blocks presented in figure 3. The Set Attribute block will assign parameters for every generated entity so the entity can be identified anywhere in the system during the simulation.

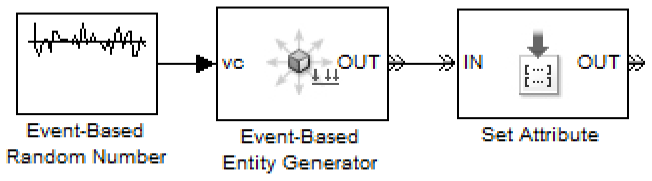

***Fig. 3. Random entity input and attribute assigning blocks.***

The input lines and the Ordered Part Buffers of the application can be modeled with the blocks shown in figure 4. The Priority Queue block can be set to simulate priorities in which the parts are living the input lines or the buffers. In a simple fashion, for the presented application we consider that the FIFO queue is suitable.

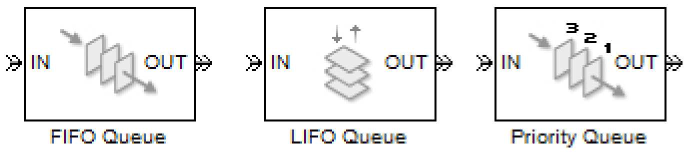

***Fig. 4. Types of block used for queue modeling.***

The assembly of parts can be modeled with blocks presented in figure 5. There is the possibility to model the sequential manner in which the robot take the parts and assembles them by using the Input Switch block.

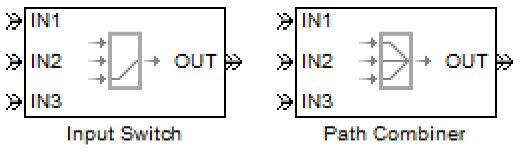

***Fig. 5. Part or product combining models.***

The two Robotic Systems of our application can be modeled with one of the servers presented in figure 6. For our purpose it would be better to use the single server option.

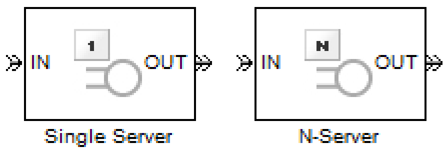

***Fig. 6. Servers blocks.***

In order to be able to analyze the simulation results, in different locations of the model, we will have to place Timing and Counter blocks (figure 7 and 8). These blocks will count the time passed between two events or will cont the processed entities giving valuable information about system operation parameters.

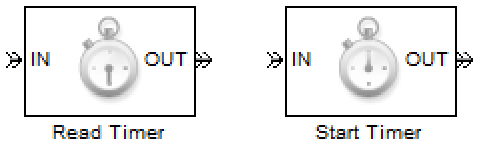

***Fig. 7. Timing blocks.***

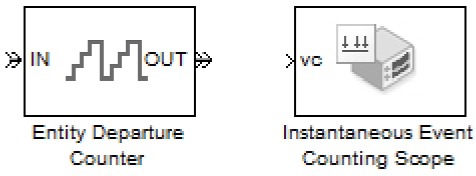

***Fig. 8. Counter blocks.***

Simulation results can be directly plot on a graph with a Signal Scope or stored to be processed later with the Simout block.

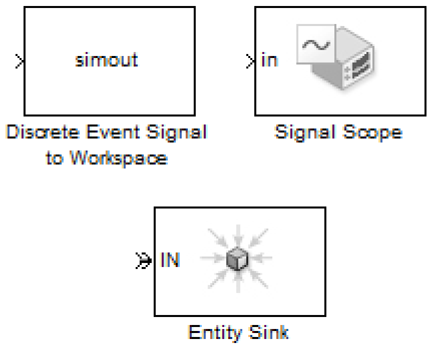

***Fig. 9. Results evaluation and assembly exit blocks.***

Finally every entity has to exit the system. In the model this is simulated using an Entity Sink block, otherwise we will have a countless accumulation of undesired entities.

## 4. CONCLUSIONS

Analyzing the simulation capabilities of SimEvents toolbox in accordance to the requirements to model the presented application we can conclude that this type of modeling is suitable and has all the capabilities to simulate the desired tasks.

In following papers we will present the results of simulations made with SimEvents Toolbox and how this results were used to optimize the described robotic system's operation.

## REFERENCES

[1] Banks, Jerry, John Carlson, and Barry Nelson. Discrete-Event System Simulation, Second Ed. Upper Saddle River, N.J.: Prentice-Hall, 1996.
[2] Cassandras, Christos G., and Stéphane Lafortune. Introduction to Discrete Event Systems. Boston: Kluwer Academic Publishers, 1999.
[3] Fishman, George S. Discrete-Event Simulation: Modeling, Programming, and Analysis. New York: Springer-Verlag, 2001.
[4] Law, Averill M., and W. David Kelton. Simulation Modeling and Analysis, 3rd Ed. New York: McGraw-Hill, 1999.
[5] ***, MATLAB, SimEvents Toolbox, Users Guide, 2007.