Simulation of Robot Arm Actions Realized by Using Discrete Event Simulation Method in LabVIEW

A. Anoufriev and Gy. Lipovszki
avanv007@hotmail.com; lipovszki@rit.bme.hu
Budapest University of Technology and Economics
Budapest 1111 Goldman Gy. tér 3.

Abstract: In the present paper a model for Robot Arm simulation is proposed. This model was developed using a new graphically programmable discreet event simulation system. This system was written in LabVIEW language and consists of several basic elements. Using these elements it is possible to construct new elements such as, for example, robot arm.

Key words: discrete event simulation, scheduling, robotics

1. INTRODUCTION

Choosing a modeling approach instead of conducting real experiments does not automatically mean a choice for simulation on the contrary. When one is able to build a mathematical model, then this must be given preference. It is possible to realize only if the relationships that compose the model are simple enough. It may use mathematical methods (such as algebra, calculus, or probability theory) to obtain exact information on questions of interests; this is called *analytic solution*.

Most of real world systems are too complex to allow realistic models to be evaluated analytically [7]. So these models must be studied by simulation.

In simulation, one uses computers to evaluate the model performance numerically, and data are gathered in order to estimate the desired behavior of the model.

Reasons for choosing to work with models rather than experimenting are usually based on cost aspects. Other reasons also come to mind, for example safety or the timespan of an experiment. Simulation should be used

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35492-7_50
G. L. Kovács et al. (eds.), *Digital Enterprise Challenges*

in those cases also when an experiment in reality is impossible or undesirable to perform.

Numerous discreet event simulation systems are existing in the world. These have advantages and disadvantages. In our frame system we tried to reduce the disadvantages of simulation environment.

1) The advantage of the present simulation system is a very high level of man-computer interactivity. Programming with graphical picture assigned icons and studying the result with the best suitable diagram type here is basic request. This programming style gives possibility to test very complex model situations that in text oriented programming is very hard or complicate.

2) The most of existing discrete event simulation systems use their own programming language for constructing models. It is increasing the time of study period but later on drastically reduce the development time of new models.

3) Finally, the main aim of this development was to construct a "relatively cheap" discrete event simulation system, for education purposes.

The Robot Arm model proposed in the present paper is connected with PUMA 560 robot. Detailed information about this robot (among other publications) may be found in J.Somlo, B.Lantos, P.T.Cat [2]. Real-time realization of robot control laws is possible using the OSA (Open System Architecture) robot control described in [5]. But all the mentioned may be used for other robots, too.

2. Description of the system

Let us formulate the task as follows. Develop the model of the PUMA 560 robot arm, which service three production lines. The robot takes a work piece from the Buffer on input and load machine tool on the output. The time for empty and charge motions is different and programmable.

Let us to describe a modeling environment of the Robot Arm model. A simple graphical interpretation is given on Fig.1. The model of the production system contains a number of model objects. Inputs of the production system are Source objects and outputs are Sink objects. Any number of objects can be between Source and Buffer as well as between Machine and Sink objects. Let us to describe some of the objects which are used in the system.

The **Source object** produces new entities with distribution functions given in interval. This subroutine is shown on Fig. 2. We can see the input parameters on left side of the picture and output parameters on the right side. Let us describe some of input and output parameters.

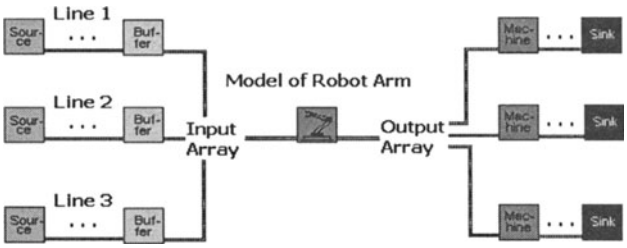

Fig. 1. Production System.

The "Container Object Name" and "Source Object Name" define given object relatively another elements of model. The "Source Object Name" is Name, which we give to this object and the "Container Object Name" is the so called, parent name.

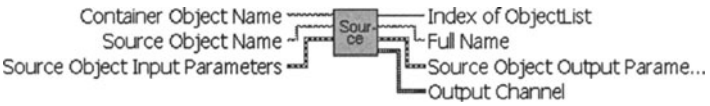

Fig. 2. Source object.

For example, if a factory has several workshops and every workshop has several production lines the "Full Name" of an element will be " 'Container Object Name' & 'Source Object Name'". The "Index of ObjectList" is an identification number of this object in a list of objects, which are used in the system. The "Output Channel" is output that entities go out through. The "Source Object Output Parameters" is output, which provides the user of the model with full information about processes inside subroutine. The "Source Object Input Parameters" are tuning up in work of the object. It is possible to program interval time between appearances of entities, the first creation time, etc.

The work of the **Sink object** is opposite to the work of the Source object. This element takes out entities from the system. This object is shown on Fig. 3.

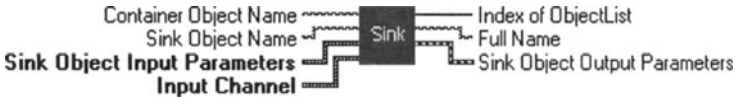

Fig. 3. Sink object.

The functions of the "Container Object Name", "Sink Object Name", "Index of ObjectList", "Full Name" and "Sink Object Output Parameters"

are similar to functions with same name in Source object. These inputs and outputs have the same functionality in every object.

The "Input Channel" is the entry point of this object where the entities are going in. The "Sink Object Input Parameters" are used for switch on/off this object in simulation.

The **Buffer object** collects entities as if a real buffer in production system collects work pieces. It is possible to program a Buffer "Capacity" as property of the "Buffer Object Input Parameters". The Buffer object is shown on Fig. 4.

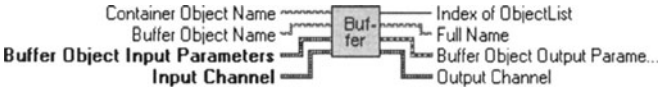

Fig. 4. Buffer object.

The **Machine object** is shown on Fig. 5. The main task of this object is to delay the motion of entity with calculated processing time. The calculation is performed by using stochastic distributions described above. The kind and parameters of distribution are input values of this object. The object calculates the next time of launching as an output parameter. More about probability distribution method are given bellow.

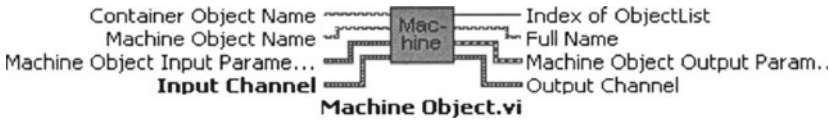

Fig. 5. Machine object.

In simulation, we use probability distribution functions to describe the stochastic character of system elements. Examples of stochastic quantities are: arrival of work piece in a system, length of the processes, breakdown behavior of a machine etc. .

There are two kinds of distributions existing: empirical and theoretical. If it is possible we use theoretical distributions. That is because there are usually not available historic data, and the theoretical distribution gives better results and easier to use. In this frame system five types of distribution functions were used: ***constant, exponential, normal, triangle and uniform distribution.***

The program of object calculates processing time or time of appearing new entity accordingly with one of the probability distributions used in the system. The advancing of the time of the process is not a trivial question in modeling (discrete event simulation).

Discrete simulator programs are using the "time slicing" or the "next event" technique for advancing the simulation clock.

Time slicing means that time passes in constant steps with other words this is a sampled time system. At each point of time, it is checked whether an event was overtaken or not. If the event was overtaken the system state is recalculated. A disadvantage of this method is that it is often slow; because it is evaluating every time step for the full system whether the system change or not. The next event technique does not have this disadvantage. The clock jumps from event to event.

Let us to explain advantages and disadvantages of these methods in an example [see 4]. The next event technique makes use of the fact that, in discrete processes, the state of the system changes erratically. If a product is processed at time 10 seconds, and the process takes 7.3 seconds, then we know the process will be finished at time 17.3 seconds. The next event simulator will determine at time 10 seconds (of simulation time) that the next event will take place at time 17.3 and (assuming that no events occur before that time) the clock will then jump from 10 to 17.3.

With the time slicing technique, time will pass according to fixed intervals and the simulation algorithm will continually ask: process ready?

For example:

t=11: Process is ready? Answer: No.

t=12: Process is ready? Answer: No.

.....

t=17: Process is ready? Answer: No.

t=18: Process is ready? Answer: Yes.

One can see that this method requests more calculation, and we can also see the size of the time steps determines the accuracy of the simulation. If one wants to simulate the processing time more accurately, than he should have taken steps of 0.1 seconds: i.e. even more calculation covers the period of 7.3 seconds. The advantage of the time slicing technique is the easier program organization.

In our frame system (simulator) the "next event" technique was used.

3. Robot Arm model

A new object for solving the given task was developed. This new object is the **Robot Arm**. It does not exist as a basic object, so we developed one

with given conditions. The Robot Arm (in future RA) model can open/close individually all its inputs/outputs. Having an input and an output strategy RA can get objects from the chosen input channel and is able to send it to a selected output channel. This task could not be solved by using the existing basic objects. So, we had to build their network and develop some new elements. This **Robot Arm** object can be used like another basic objects as base for constructing new objects. This program is shown on Fig. 6 .

Two inputs "Status of Next Line" (machine states connected to Robot Arm output) and "Contents of Previous Line" (content of Buffers connected to Robot Arm input) are used for choosing of input and output indexes.

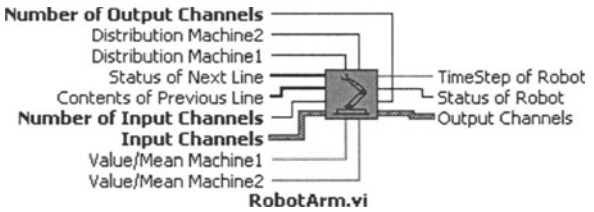

Fig. 6. Robot object.

The structure shown on Fig. 7 was used in **Robot Arm** object. This model contains a **Join**, a **Selector** and two **Machine** subroutines. Short descriptions of these objects is given bellow.

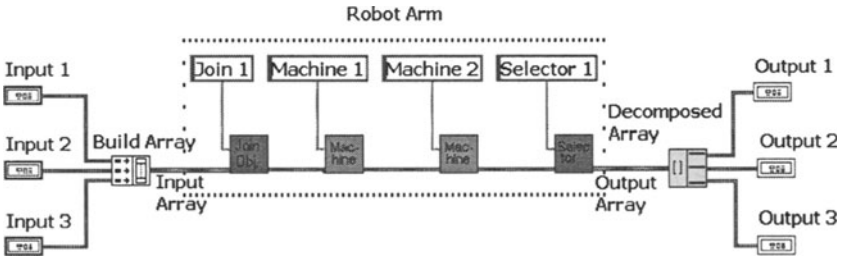

Fig. 7. Model of the robot.

The Machine1 calculate time for moving without work piece and the Machine2 for moving with work piece in a gripper. These calculations fulfilled by using stochastic distribution methods.

Let us describe the main objects of the model. The aim of **Join** is choosing from entities of inputs according with control policy described below. Join as all another basic objects is subprogram with input and output parameters (see Fig. 8).

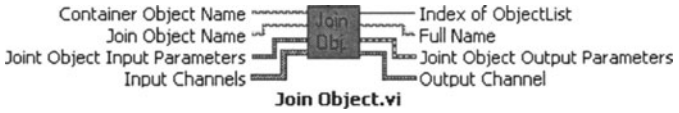

Fig. 8. Join object.

One can see the input parameters on the left side of the picture and output parameters on the right side of it. It is possible to switch off /on this subroutine from simulation.

Let us describe some of the input and output parameters. The main parameter is the "Input Channels". The entities go into Join object through this input. Every entity brings information about its number in object list and about its current channel. The "Output Channel" contains the entity from selected input channel. The input parameters of Join are "Capacity" (number of entities inside the Join object), the "Number of Input Channels" (dimension of input array) and the "Index of Input Channel". The Join object output parameters indicate the content, work and stage of this object during a simulation. Every object has its individual name. It gives us the possibility to use the same subroutine many times in the simulation model. The "Container Object Name" is a so-called parent name of object. The "Full Name" includes name of container and name of object. The "Index of Object List" is the identification number of object in list of objects used in the system. The work of **Selector object** is opposite to work of Joint. Selector takes the input and inserts it in output array accordingly with output control policy described bellow. Selector subroutine is shown on Fig. 9 .

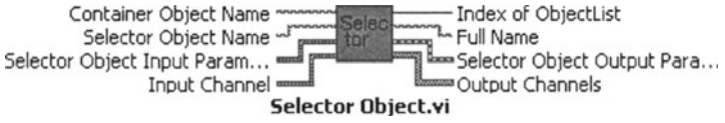

Fig. 9. Selector subroutine.

The input and output parameters are similar to a Join's parameters but it has an array of channels on exit and single channel on the input.

By using of these objects we started to develop the Robot Arm object.

Let us assume that the control policy for choosing of an input is based on *Work from the Buffer with the Largest Queue.*

The largest queue policy work in the model is the next. Let T_0=0. At time t=T_n let the robot choose that buffer to work which has the largest current level. If there is more than one buffer content which are not zero we find the maximum content buffer index. If there are more buffer with equal content we choose from sequence of "not empty buffers". After this process

the robot picks up the waiting entity and move it to an idle state machine tool. The choosing of machine tool fulfilled accordingly with one output choosing policy described bellow. This process is repeated before the production is stopped.

The following matrix determines the output choosing policy:

$$\begin{pmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nj} \end{pmatrix}.$$

The matrix has dimension $N \times J$ where N is a number of rows and J is number of columns. In our case $N=J=3$ and n corresponds to input and j corresponds to output. The robot chooses the output channel with non-zero element in the row. The output matrix, for example is hard determined as in a "ProductSelectingTable":

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

The program gives possibility to fill up control matrix accordingly with one of the rules specified below:

1. Specific channel: always send to channel j.
2. By percentage: for example 90 % of products go to channel j, the remaining percentage go to channel $j+1$ (Bernoulli distribution).
3. By entity name: if the entity name of the 1st object in the queue matches "EntityName" then send to channel j else $j+1$.
4. By "User Attribute" value (direct): the channel number is written directly on the label named "OutputName" of the 1st entity in the queue.
5. By label value (conditional): if the value on the "User Attribute" of the 1st entity in the queue is less than a given value then send to channel 1, equal then channel 2 else 3.
6. Round robin: all output channels are used in rotation. If channel is closed, then wait till open.
7. Largest queue: Send to the channel connected to the entity with the largest queue.
8. By lookup table: Send to the channel specified in row 1 column 2 of global table named "ProductSelectingTable".

Let us shortly describe one of the twelve output selecting methods enumerated above. Let the method that choose output would be based on Bernoulli distribution (5^{th} in the list of the methods). The essence of the method is that the system send X percent of all entities which arrive on three

inputs of the robot to the first output Y percent to the second and 100-(X+Y) to third output. X and Y are programmable values. Graphical interpretation of this method is shown on Fig. 10. The program generates values from 1 to 100 after arriving new entity to the robot's inputs. The probability of appearance every value from 0 to 100 is equivalent.

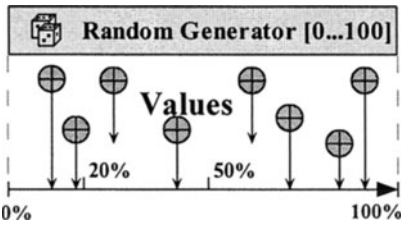

Fig. 10. Bernoulli distribution.

For example, let us send 20% of all entities to the output 1, 30% to the output 2 and leftover 50% to the output 3. It means that if generated value is less or equal 20 the model sends it to output 1, if one is more 20 and less or equal 50 the model sends it to output 2 and if generated value is more 50 the model sends it to output 3.

This strategy can be easily realized on LabView program-ming language (see Fig. 11).

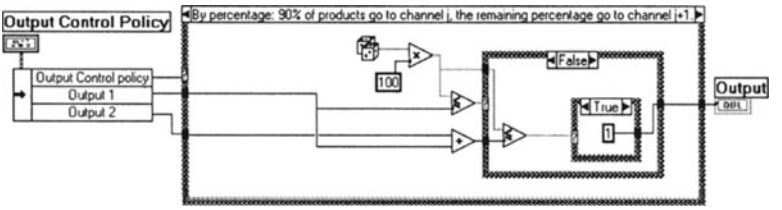

Fig. 11. Realization of Output Choosing Strategy.

The system gives information about the results in the form of Gantt diagram. The Gantt diagram is a XY graph. Time is shown along X-axis and on Y-axis are placed the states of objects in the system. The Gantt diagram gives us real information about stage of object in every time slice. An example of Gantt diagram is given on Fig. 12. Usually, Gantt charts are used in the production management for visualization the solution of scheduling problems.

The numbers along X-axis are time units and colored areas on the chart show periods when objects are busy. The program builds Gantt diagram for the simulated production period. The objects of system placed along Y-axis by the following way:

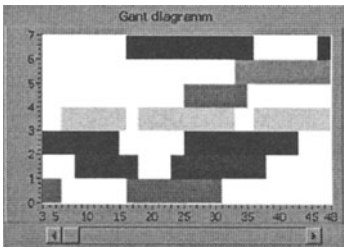

Fig. 12. Example of Gantt chart.

Nominations of Fig 12. can be seen in Table 1.

Table 1. Nomination of GANTT chat.

Position on Y axis	**1.1.1.1.1.1.1.1 Name of Object**
between 0-1	Machine1 (at the input of RobotArm)
between 1-2	Machine2 (at the input of RobotArm)
between 2-3	Machine3 (at the input of RobotArm)
between 3-4	RobotArm
between 4-5	Machine4 (at the output of RobotArm)
between 5-6	Machine5 (at the output of RobotArm)
between 6-7	Machine6 (at the output of RobotArm)

The Machine1 and Machine4 belong to the production line number one. The Machine2 and Machine5 belong to the production line number two and Machine3 and Machine6 are elements of the production line number three.

The scheduling problem formulation and solution can be **static** and **deterministic** [see 6]. When the number of jobs and their ready times are known and fixed our system is static. When processing times and all other

parameters are known and fixed the system is deterministic. Our system can solve **dynamic** and **stochastic scheduling** problems. It means that jobs can arrive randomly over a period of time (dynamic) and the processing times, times of appearances of new entities etc. can be uncertain (stochastic).

All these reasons can lead to situations when static and deterministic scheduling solutions which were produced by a manager are quite different from scheduling composed automatically by the program.

4. Planed Future Developments

The constructing of Robot Arm model is the first step of development. We are planning to develop all elements, which are needed for simulation production systems and flexible manufacturing systems. The model of transportation system and Robocar will be developed in the near future.

5. SUMMARY

In this article we introduced a new discrete event development system and its application in developing of new element named Robot Arm. We can use any number of objects as reference of the basic class objects. It is possible to develop new object by using the base objects or other (previously) developed objects. The first tests of the presented system have shown that the development system is working properly and development of a new object is not too hard. The system gives the possibility to build up Gantt diagrams and can be applied for verification of the solution of scheduling problems.

6. References

[1] Gy. Lipovszki "Diszkrét Esemény Szimulátor LabVIEW programnyelven" Budapesti Műszaki és Gazdaságtudományi Egyetem Rendszer- és Irányítástechnika Tanszék, Budapest, 2000 szeptember.

[2] J.Somlo, B.Lantos, P.T.Cat "Advanced Robot Control" Akadémiai Kiadó. Budapest 1997.

[3] Averill M. Law, W. David Kelton "Simulation Modeling and Analysis", Second edition, McGraw-Hill, Inc

[4] "Simulation. A pragmatic guide to discrete event simulation in business and engineering." F&H Ltd. The Nederland's. 1998

[5] Somlo Janos, Loginov Alexander, Sokolov Alexei "Optimal Robot Motion Planning and Realization Using LabView" INES'99 IEEE International Conference on Intelligent Engineering Systems, 1999 November 1-3, Stara Lesna, Slovak Republic.

[6] S. French "Sequencing and scheduling: An introduction to the Mathematics of the Job-Shop" Ellis Horwood Limeted, 1982.

[7] Ivan L. Ermolov, Philip R. Moore, Jury V. Poduraev "Modeling and visualization for mobile robots working in severe environment" IFAC Symposium on Manufacturing, Modeling, Management and Control. Patras Greece, July 2000.

Alexandre ANOUFRIEV was born in Moscow, Russia in 1976. He received M.Sc. degree in mechanical engineering from Budapest University of Technology and Economics (major in Robotics) and Bauman State Technical University (major in MECHATRONICS and control theory), in 1999 and 2000 respectively.

At present moment he is Ph.D. candidate at Budapest University of Technology and Economics. His current research interests are in Scheduling theory, Simulation and Optimization of Manufacturing processes, Production Planning.

György LIPOVSZKI was born in Miskolc, Hungary in 1950. He received M.Sc. degree in electrical engineering from Budapest University of Technology and Economics, at 1975.

He has received his Ph.D. degree at Budapest University of Technology and Economics in 1997. He is associate professor of Department Systems and Control Engineering at Faculty of Mechanical Engineering. His main research and education topic is continuous and discrete event simulation and control