Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

EVALUATING EMBEDDED DECISION PROCESSES OF MANUFACTURING SYSTEMS THROUGH SIMULATION

S. Cem Karacal

Southern Illinois University
Mechanical & Industrial Engineering
Edwardsville, IL 62026, U.S.A.

ABSTRACT

This paper addresses the issues related to the decision processes of manufacturing system simulation. The manufacturing system is perceived in terms of intelligent entities capable of making non-programmed decisions and data driven entities capable of making programmed decisions. The system entities are classified according to the levels of the previously defined modeling formalism. The objective of the study is to investigate the effects of employing multiple layers of non-programmed control. An example make-to-order type of manufacturing system is simulated using the developed tool to observe the impacts of different layers on overall performance.

1 INTRODUCTION

The manufacturing systems are purposeful systems that contain several intelligent entities that make state dependent decisions. In traditional simulation modeling, the non-programmed decision making processes are implicitly represented in the model if they are represented at all. Therefore, there is a strong need for a tool that can simulate the underlying decision processes of manufacturing systems along with traditional simulation of the physical system. The simulation modeling formalism developed (Karacal and Mize, 1996) uses five levels, namely, *source*, *data*, *information*, *knowledge*, and *intelligence* anyone of which can contain entities. Each level is formalized through a generic construct that contains several sets and functions to explain relations/interactions among system entities. The information and intelligence levels are the main tools used to represent simulation dynamics and they correspond to *Data-Driven- Physical* and *Decision-Making* entity types of the formalism. These two entity types are used to encapsulate programmed and non-programmed control in a manufacturing system. The programmed control corresponds to a predefined sequence of actions based on condition checks and are carried out in a sequential fashion. The non-programmed control (decision making process) uses the knowledge of the current state of the system and selects an action by searching through the knowledge base to satisfy a given goal. These processes do not have a preset sequence of activities and actions are selected based on state depended decision rules encoded in their knowledge bases. The goals for each intelligent entity are defined as the production orders received from superior entities that contain quantity and timing information on a particular part or component.

The formalism is implemented using the Smalltalk object-oriented programming language (Karacal and Mize, 1997). It utilizes a grammar, based on concepts from formal language theory, to translate customer orders into component orders and then component orders to batch orders (Karacal, 1997a). The orders are broken into detailed shop, work-center, and batch orders in a hierarchical fashion as they are released by non-programmed (decision making) control entities to subordinate entities. The comparison of the developed system against a traditional simulation methodology using the Analytic Hierarchy Process yielded favorable results (Karacal, et. al., 1996).

2 EXAMPLE SYSTEM

A simple make-to-order manufacturing system is defined to illustrate the modeling methodology developed. The example model demonstrates the versatility of the information and insight that can be gained from simulation. In the developed framework, modeling begins by describing the physical configuration of the manufacturing system. This is the backbone of the simulation model and is the process of defining machines contained in each work-center, work-centers contained in each shop, and shops that form the total manufacturing system. The physical configuration of the system is represented with a tree structure to take full advantage of Smalltalk's inheritance property.

2.1 Physical System Configuration

During model development, as the physical entities are created, their corresponding default material ports, control objects, and communication channels are automatically created and linked to the physical entities. The model developer is given the opportunity to overwrite and customize these default structures as they are created. The physical configuration of the simple example system is illustrated with the hierarchical tree given in figure 1.

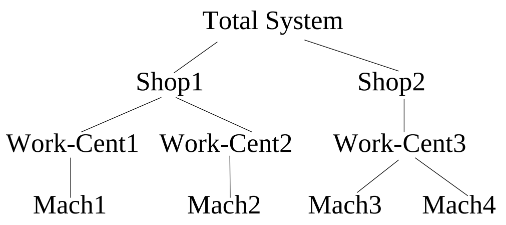

Figure 1: Physical Configuration of the Example Manufacturing System

The next step is the definition of some products, in terms of their product structures (illustrated in figures 2 and 3) and product data bases, that are manufactured in the example system. These product structure trees have small data dictionaries attached to each node that contain information on:

i) Name and location for the machine that produces this part, component, or product
ii) Mean operation time distribution
iii) Mean lead time distribution for the part, component, or product
iv) A factor that indicates how many of this item is required for each unit of its parent in the product structure tree.

After the product structure trees are defined they are linked to an object class that represents the total product data base for the manufacturing system.

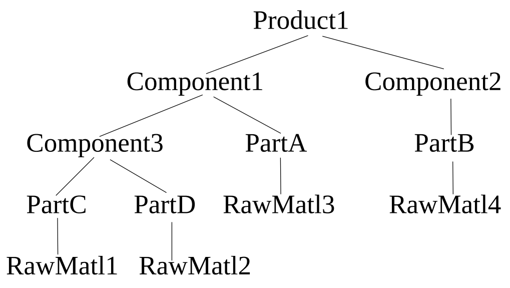

Figure 2: Product Structure for Product 1

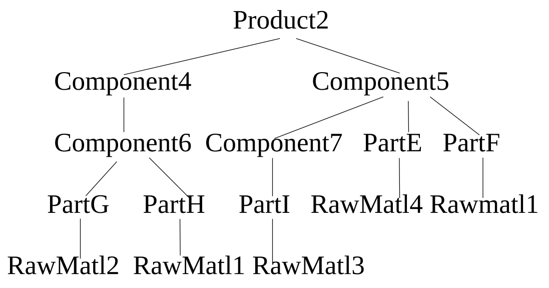

Figure 3: Product Structure for Product 2

2.2 Knowledge Bases

The next step in the modeling process is the definition of the knowledge bases for each control level defined in the hierarchy. Three generic object oriented knowledge bases and their inference rules are defined for system, shop, and work-center levels and called *systemPlanning, shopPlanning,* and *WorkCenterPlanning,* respectively. During the definition of these knowledge bases and inference rules, the simulation model developer can express the operational policies and system expertise for each specific non-programmed control entity. The knowledge bases and the rules defined for the example system modeled are rather simple ones and reflect typical behavior of hierarchical control levels of a make-to-order type of manufacturing.

System level non-programmed control has two knowledge base entities named systemPlanner and shops with several parameters are associated with each. Upon receiving the request from system level controller, *systemPlanning* knowledge base first checks the timing of the product order through a series of rules. If timing is critically close to due-date, it immediately releases the order. Otherwise, it evaluates a set of rules using order size, product type and customer identification to derive the importance level of the order. The next step is the determination of the timing priority for the order using order importance, lead time and slack time information. Then, the *systemPlanning* object acquires the present shop load and capacity knowledge from the model in real time while the simulation is running. The quantitative information on shop loads and capacities are retrieved by the system control object from the involved shop control objects, converted into symbolic form, and used in non-programmed decision making to find an action for the order on hand.

The conversion of quantitative information to symbolic knowledge takes place as follow:

1) The system controller object looks at the shop load in terms of a moving monthly time window. As product orders are released, the windows of the shops involved are loaded using the lead time and order quantity information. As orders completed and breakdowns or other problems are encountered, the time windows are

updated to reflect the changes during the simulation. When shop load status is requested for a particular shop, an expression that relates present load, unused shop time, and a flexibility factor is evaluated to find a numeric value at that instant of time. Depending on which predefined range this value falls into, one of the possible symbolic values such as high, low, etc. is returned to the knowledge base.

2) A symbolic value for shop capacity status is determined using another expression that relates the number of operational machines, the type of orders being processed at that instant of time, shop reliability, and total number of available machines in the shop. Again, depending on which range this numerical value falls into, a symbolic value such as full-capacity, low-capacity, etc. is returned to the knowledge base.

The *systemPlanning* knowledge base uses a set of rules to ascertain the possible values for the parameter action according to various combinations of order timing, shop load and capacity status situations along with the system level plan already on hand. In each case, the value assigned to action (either release or put the order at a specific place in the system plan) has an uncertainty value associated with it which is later utilized to revise the plan. The action is passed back to a system controller object that implements the action by updating the internal and external states. The action inference rules try to evenly spread the work load in terms of shop loads and order timing. The inference mechanism uses a simple non-monotonic reasoning mechanism based on goal regression that is explained in detail in (Karacal, 1997b).

The next layer of knowledge bases, an instance of *shopPlanning* knowledge base for each shop, are defined for non-programmed shop decisions. They are customized from their default structure by the model developer through definition of new parameters and rules. The two knowledge base entities defined for each are *shopPlaner* and *workCenters*, each with a set of parameters. These knowledge bases deal with component orders and their main design principles are similar to those in *systemPlanning*.

When they are interrogated, these knowledge bases asses a symbolic value for component order timing using due-date and lead time information from the interrogating shop control object. A set of rules classify the component on hand according to its position in the product structure tree. The symbolic knowledge regarding the load and capacity status of the work-centers involved in processing of the present component order are obtained in real simulation time. The methods used in deriving these symbolic values are very similar to the ones used in *systemPlaning* knowledge base, with the main difference being the weekly time window used. These knowledge bases search through a set of rules representing various combinations of work-center load and capacity status values, order timing, and the present shop plan to find a value for parameter action. This value is passed back to shop controller which in turn implements the action.

The last non-programmed decision level in the hierarchy is handled by a set of *workCenterPlaning* knowledge bases that deal with batch orders released to machine controllers. These are relatively simple structures that consist of two entities *workCenterPlaner* and *machines*, each with a set of parameters. They decide on what to do with each individual batch order using the real time symbolic values derived for order timing, machine status, machine reliability, and present work-center plan. The work-center control object in turn implements the action returned by its *workCenterPlaning* knowledge base.

When physical machines receive batch orders from work-center control objects, they translate them into a set of activity orders that are carried out sequentially based on a programmed control that may perform condition checks such as physical material being available at that particular location, etc. This is where non-programmed control links to programmed control and decision processes are connected to physical activities.

The other model relevant data are all specified by the model developer during model definition using the standard probability distributions provided in the system. The order inter arrival pattern is an exponential distribution. The customer types, order sizes, requested due-dates, are all generated from dicrete sample spaces defined for this manufacturing system. The processing times are assumed to be normally distributed and the means and standard deviations for each operation are obtained from the data base when a processing time random variable is needed. The MTTB and MTTR are assumed to have exponential distributions with a specific mean time for each machine type.

2.3 Experimentation

The described example with its system, shop, and work-center planing entities is simulated for three months period (43,200 min. with 480 min./day). A one factor four levels experiment is designed to compare the performance of the system with or without different levels of knowledge based control. Considering the make-to-order nature of the manufacturing system, the following three statistics are analyzed.

1) *Throughput time*: Time between the release of a product order by system controller and the completion of that order.

2) *Order lateness*: Time difference between the completion of an order and its due-date.

3) *Customer response time*: Time between the arrival of a customer to the system and the completion time for all the products requested by the customer in that order.

In addition, traditional simulation statistics such as machine utilizations and queue statistics are collected. Throughput time is selected as the main statistics of interest. The four experimental levels are defined as follow:

I) Manufacturing system without non-programmed control. The orders at all levels are translated into suborders and immediately released to the next level. The orders mostly accumulate at the machine control level where they are processed according to FIFO.

II) Only system level non-programmed control is employed using *systemPlaning* knowledge base. The lower level controls are programmed decisions and orders are processed using FIFO.

III) The system and shop control object both use their knowledge bases simultaneously.

IV) The system, shop, and work-center control objects all use their knowledge bases simultaneously.

The main idea behind this experimental design is to perform a tighter timing and status control around product, component, and batch orders as they move downward in the control hierarchy. The frequency of consultations with the relevant knowledge bases during the simulation is different for each control level. The work-center knowledge bases are the most frequently queried ones, shop knowledge bases next, and the system knowledge base is the least. Table I shows the average of five replications made for each level.

Table I: Average Throughput Times (TpT-in minutes) of Five Replications

Levels	1 No Know Base (KB)	2 System KB	3 System & Shop KB	4 System & Shop & Work-Cent. KB
Run#	1	2	3	4
TpT	9825	9335	8488	7024
Run#	5	6	7	8
TpT	12017	11645	11788	9712
Run#	9	10	11	12
TpT	13377	12747	12415	6212
Run#	13	14	15	16
TpT	12861	12259	12743	5509
Run#	17	18	19	20
TpT	9425	9182	8965	7623

Common random number seeds are used to reduce variance during pairwise comparison of the difference between throughput times (in minutes) of levels. The random number seeds are changed between replications but kept the same across the levels. Furthermore, to better observe the effect of non-programmed control, the manufacturing system is slightly overloaded.

The simulation data is analyzed as follow: First, level 1 is compared with level 2 by taking the difference between mean throughput times of runs 1-2, 5-6, 9-10, 13-14, and 17-18. Then, a 95% confidence interval is constructed for the true mean difference. The same procedure is applied for pairwise comparison of other levels. The conclusions are drawn based on relative location of the confidence interval with respect to zero.

2.4 Results

The following conclusions are drawn.

Levels 1 - 2: The confidence interval for the true mean difference does not contain zero and completely lies in the positive region. Therefore, the difference is statistically significant and the system yields smaller mean throughput values when system level non-programmed control is employed.

Levels 2 - 3: The confidence interval for the true mean difference contains zero, implying that there is not enough statistical evidence to claim that one level is better than the other. The length of the interval suggests that more replications needed to reach a conclusion. Therefore, we can not claim that the system performs better when shop and system level non-programmed control are employed simultaneously. One thing to bear in mind is that the performance of the system heavily depends on the particular knowledge bases defined. Since this pair wise comparison did not yield a preference between the levels, levels 2 and 3 are pairwise compared to level 4.

Levels 2 - 4: Although relatively wide, the confidence interval for the true mean difference does not contain zero and lies in the positive values region. Therefore, with the sample runs on hand we can claim that the throughput time is shorter when system, shop, and work-center levels use their knowledge bases simultaneously.

Levels 3 - 4: The confidence interval for the true mean throughput time difference completely lies on the positive region. Thus, the system yields shorter throughput times when all knowledge bases are employed simultaneously.

3 CONCLUSIONS

From the above statistical analysis, I concluded that system performs best when non-programmed control is used at all control level. This conclusion was somewhat expected and suggests that unless a sound non-programmed and programmed control scheme is applied at the shop floor level, the benefits that will be obtained from upper level non-programmed control will diminish and will not significantly impact the overall performance of the system. This observation closely coincides with real life situations (Vollman et. al. 1984). The other statistics collected such as customer response time and order lateness showed a similar trend. In general, throughput time can be improved by using shortest processing time (SPT) rule in programmed control, but since it does not take due-date information into account, it may severely deteriorate order lateness statistics. By using due-date, lead time, present state of the system together in the knowledge bases, we can define rules that make measured compromises among conflicting aspects of several system performance measures.

Although not statistically analyzed, traditional measures such as utilization statistics and queue statistics showed that as layers of non-programmed hierarchical control are added, queue lengths were shorter and machine utilizations gave smaller values due to better batch timing.

The other possible set of experiments such as making replications with only shop or work-center knowledge base or other combinations to see combined and/or stand alone effects of the knowledge bases is left as a future investigation area. Also, other programmed control rules such as LIFO, STP, other control heuristics, and more refined knowledge bases and inference rules will be investigated in the future.

Another major advantage of this modeling and control framework developed is the flexibility it brings into the simulation and modeling process. The operations of both physical and logical aspects of a manufacturing system modeled not only closely represents reality but are very easy to change. This system can easily be used as a demonstration tool to show that various local optimization rules and procedures of different control levels (or even different modules of the same level), when put together, may deteriorate total system performance. It can also be a very useful tool to shown how conflicting multiple objectives of different control levels interact and affect global performance.

REFERENCES

Karacal, S. C., and J. Mize. 1996. A Formal Structure for Discrete Event Simulation, Part I: Modeling Multiple Level Systems. *IIE Transactions*, 28, 753-760

Karacal, S. C., T. Beaumariage, Z. Karacal. 1996. Comparison of Simulation Environments Analytic Hierarchy Process. In *Proceedings of 1996 Winter Simulation Conference*, San Diego, CA

Karacal, S. C., and J. Mize. 1997. A Formal Structure for Discrete Event Simulation, Part II: Object Oriented Software Implementation for Manufacturing Systems. To appear in IIE Transactions.

Karacal, S. C. 1997a. A Goal Decomposition Language for Manufacturing System Simulation. In Proceedings of 1997 IE Research Conference, Miami Beach, FL.

Karacal, S. C. 1997b. A Formal Structure for Discrete Event Simulation, Part III: Knowledge Processing During Simulation, SIUE Industrial Engineering, Edwardsville, IL working technical paper 97-2.

Vollmann, T. E., W. Berry, D. Whybark. 1984. *Manufacturing Planning and Control Systems*. Richard D. Irwing publishing, Inc. Homewoods, IL.

AUTHOR BIOGRAPHY

S. CEM KARACAL is an assistant professor of Industrial Engineering at Southern Illinois University at Edwardsville. He received his B.S. degree in IE from Middle East Technical University, Ankara, Turkey, in 1982, and M.S. and Ph.D. degrees in IE from Oklahoma State University in 1986 and 1991, respectively. He has two years of experience in industry and consulting. His primary areas of interest are object-oriented modeling methodologies, AI applications in simulation, and semiconductor manufacturing scheduling. He is a member of IIE, SME, Alpha Pi Mu, and Tau Beta Pi.