Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp407-412)

Shop Floor Control Simulation with a Combined Process/Resource-oriented Approach

PYOUNG YOL JANG
Innovation Economy Department
Science & Technology Policy Institute (STEPI)
26th Fl., Specialty Construction Center 395-70, Shindaebang-dong, Seoul 157-714
REPUBLIC OF KOREA
http://www.stepi.re.kr/jangpy

Abstract: Shop floor control systems (SFCS) are used to make real-time planning and scheduling decisions to optimize the efficiency of manufacturing shops. These shops exhibit a non-linear, dynamic evolution. While simulation models are often used in the aforementioned decisions, Numerous commercial simulation languages typically support one of two views: a process-oriented view or a resource-oriented view. Process-oriented languages view systems as networks in which the nodes represent tasks, connected to processes. The emphasis in this view is on the flow of entities through the processes; the processes themselves and the resources that perform them are hidden. Resource-oriented languages view systems as a collection of resources that perform operations on parts as they flow through the system. The emphasis is on the resources; the processes they perform are hidden. In this paper, we argue that neither view is sufficient for planning and scheduling decisions. We present a new type of simulator that combines both views.

Keywords: Simulation, Shop floor control, Process and resource models, Decision-making problems

1 Introduction

Shops involved in the fabrication of discrete parts typically have a software system that must be able to plan, schedule, monitor, and control various machining and material handling devices. That system, called the shop floor control system (SFCS), does this through a series of decisions that 1) ensures the completion of production orders and 2) optimizes one or more performance measures [4]. Simulation has been used to support these decisions because it can model and analyze systems, like a manufacturing shop, that operate in an environment that is highly dynamic and unpredictable.

To utilize simulation efficiently and effectively in these areas, simulation must satisfy following four requirements: 1) It must be able to represent and utilize efficiently the nonlinear process plan which is a primary input to the SFCS, 2) It must be able to represent and deal with the various decision-making problems encountered in operating a shop floor, 3) It's modeling must be easy to build and understand, and 4) It's modification must be done rapidly and effectively. However, the existing simulation languages do not fully satisfy the above four requirements. They require hand-woven program codes to represent and utilize the nonlinear process plan, and to deal with decision-making problems (1,2). They supports simulation modeling either in a process-oriented view or in a resource-oriented view, which prohibits the users from building, understanding, and modifying simulation models easily and effectively (3,4). Process-oriented view models systems as networks in which the nodes represent tasks, connected to processes. It describes the sequence of activities or processes that the entities go through or experience. The emphasis in this view is on the flow of entities through the processes; the processes themselves and the resources that perform them are hidden. Resource-oriented view models systems as a collection of resources that perform operations on parts as they flow through the system. The emphasis is on the resources; the processes they perform are hidden.

We believe that a simulator capable of both process-oriented and resource-oriented views would be beneficial for the following reasons. First, nonlinear process routings, including both AND and OR branches, can be modeled more easily. Second, all of the important information about the parts and the resources would be visible and hence changeable easily in the model. Third, the decisions made by the SFCS, which require up-to-date information about both the parts and the resources, can be better analyzed. Finally, a separation of the data required to

run the model from the physical model of the shop itself can be achieved more easily. This would provide the SFCS with the ability to analyze many different scenarios before making a decision.

The objective of the paper is to describe a new process and resource models-based intelligent simulator (PRISM) which satisfies the aforementioned four requirements. To this end, the paper describes the following: (1) the process model, which represents complex and flexible process plans for producing parts, (2) the resource model, which represents the characteristics and distributed relationships of various resources, (3) the simulator engine, which advances the simulation clock and manages the evolution of the simulation by investigating various pieces of information specified in the process and resource models, (4) a comparison of PRISM with other simulation tools in terms of effectiveness and efficiency.

2 Related Work

There are many commercial simulators on the market today; some of them are general-purpose simulators and some are designed specifically to model and analyze manufacturing systems. In general, the existing simulation languages can be classified into two generations. The first generation of simulation languages supported a process-oriented view of systems. They began to appear in the late 60s, and some are still in use today. Most of the early simulators, including GPSS [5], SIMAN [7], SIMSCRIPT [6], and SLAM [8] are examples of process-oriented languages. These languages use a complex grammar and model systems as networks in which the nodes represent queues and the branches represent the paths connecting those queues. Each queue is connected to a process, and the emphasis is on the flow of entities through those processes -- not the processes themselves; the processes themselves and the resources that perform them are hidden. This means that the impact of the resources' properties and their distribution layout are hidden from the user. Consequently, those languages cannot support easily the dynamic decisions related to the concurrent movement of parts.

The second generation began to appear in late 80's. They include sophisticated, icon-based user interfaces and links to general-purpose programming languages such as C, C++, and VISUAL BASIC. Recent simulators, including WITNESS [1], ProModel [9], AutoMod [2], and SIMFACTORY [3], are resource-oriented languages. They view a system as the specification and arrangement of resources that perform a number of different operations on entities as they flow through the resources. The emphasis is on the resources, not the entity flow. The process sequences are hidden within and across resources, as are the part characteristics, which make it complicated to build and modify complex simulation models.

3 Framework

The concept for PRISM is shown in Figure 1. The process plans related to the parts to be machined are expanded and represented as the process model, while the resource properties and layout are represented as the resource model. The process model is encoded as an AND/OR graph form, in which a process node associated with a machining feature in the process plan contains required resources and related decision-making rules. The resource model is represented as the set of arranged icons of corresponding resources, in which each icon contains the functionality of its related resource, related decision-making rules and reachability relationships to other resources. The simulator engine of the PRISM reads the two models and then runs the simulation by managing events on the basis of an integrated view of the two models.

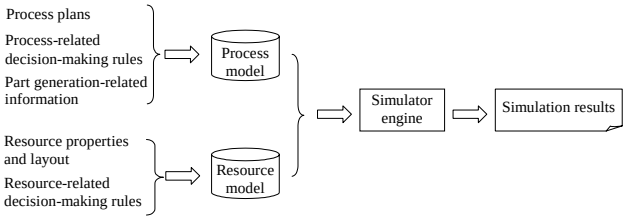

Figure 1. Concept diagram of the PRISM

We reviewed all the decision-making problems encountered in a SFCS and then classified them into the two groups according to their characteristics: 1) process model-related decision rules and 2) resource model-related decision rules as shown in Figure 2. We discuss the definitions of the problems in Sections 4 and 5. To resolve each decision-making problem, a decision-making rule can be selected and applied by the user.

Problems encoded in the Process Model	Problems encoded in the Resource Model
Process sequence problem	Part selection problem in a machine
Path selection problem	Transport selection problem
Resource selection problem	AGV location problem
Part buffering problem	Part selection problem in an AGV
	Robot location problem
	Part selection problem in a robot

Figure 2. Process and resource related problems

4 Process Model

When it arrives at the SFCS, a part carries a process plan that must contain the various processes to produce a finished part from the raw material. It must also contain temporal precedence relationships among these processes. In the past, a process plan was represented as a linear sequence of processes or resources. Recently, as noted above, many researchers have argued for allowing alternative processes and resources. We incorporate these alternatives and model the resulting process plan as an AND/OR graph in which a node is one of three kinds, *head* (represented as ellipse), *process* (rectangle), *junction* (circle), and an edge is denoted as *link* (arrow).

The *head* node contains the information needed for the creation of a part for simulation, such as first arrival time, inter-arrival time distribution, and lot size of the associated part. The *process* node represents a single machining operation performed on a part, such as turning, milling, or drilling. This node contains the information contents on the service resources, such as machine tool, cutting tool, and fixture. The *junction* node denotes the part flow logic. There are AND junctions (displayed as "A") and OR junctions (displayed as "O"), each of which consists of fan-out and fan-in pairs.

The attributes of the *process* node are process name, resources, and decision rules as shown in Figure 3. Two decision problems are possible at in every process step in the process plan: resource selection and part buffering. If alternative resources are specified for a process, the simulator must choose a primary and secondary resource set from the resource database. This is done using user-specified rules such as minimum processing time, minimum transport time, or minimum waiting time. For example, if the minimum-transport-time rule is selected, the simulator engine selects the machine tool located nearest to the current part's location. The part-buffering problem is associated with the determination of the next destination for the part when no resource specified for the next process is available. In that case, the simulator engine may (1) send the part to the buffer, (2) let it stay at the current location, or (3) send it to the material handler. Figure 4 illustrates the sequence of rules fired.

Figure 3**.** Properties of the symbol *process*

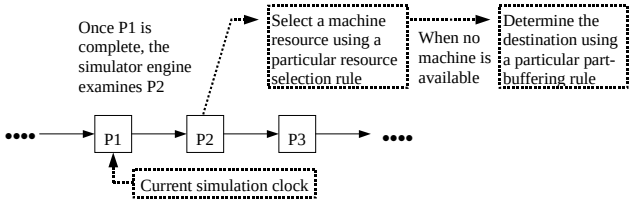

Figure 4. Sequence of fired rules after process P1

The AND junction means that all the processes between the starting A and ending A must be performed. The OR junction means that one and only one of the processes or paths between the starting OR and the ending OR junctions should be selected and then executed

When the simulator engine hits the initial AND junction, it does not immediately sequence all of the ensuing nodes. Instead, it chooses only the next node in the sequence. It does this one-at-a-time selection until the final AND junction is reached. This procedure, called the process sequence procedure, uses both the current state information and the rules imbedded in the definition of an AND junction. For example, if the rule is "minimum traveling time", then the process selected is the one to be executed on the machine located nearest to the current location. When the simulator engine encounters an initial OR junction, it prioritizes all of the possible paths or processes, then selects the one with the highest priority. This procedure, called the path selection procedure, is also based on the current state and the rules imbedded in the OR junction. For example, the maximum-flexibility rule will select that path with the largest number of AND junctions, because that path has many processes that can be sequenced later.

5 Resource Model

A process model contains no information about the resources on which processes are executed. PRISM

represents and stores this information, which includes such characteristics as resource type, resource properties, resource location, number of resources, and distances between pairs of resources, represents and stores this information in a separate model called the Resource Model.

There are three resource classes: processing, storage, and transport. A processing resource (MACHINE), typically a machine tool or a human being, performs various machining or inspection processes. A storage resource, typically buffer (BUFFER) or a storage facility (AS/RS), houses parts and materials for varying amounts of time.

Since all resources must work together to manufacture parts, we must specify their interactions. We assume that all interactions involve a material handler, because it alone picks up from one resource and puts down at another resource. A processing resource executes each "machine" process step in the process model. Since resource may have a breakdown, breakdown intervals and repair times are also included. There are also two decision-making problems: transport selection and part selection. The transport selection problem is to determine the transport path and all required transport resources needed to move the part from its current location to its next location. A path is chosen from the available candidates identified from the directed graph. The choice is made using the imbedded rules such as "Minimum transport time" or "Non-busy transport". The part selection problem is to choose the next part to be processed from among those waiting in the buffer or other storage locations. Many rules can be used for resolving the part selection problem, such as "Minimum processing time", "Minimum transport time", "Earliest due date", or "First in, first out". Figure 5 illustrates the procedure for part selection.

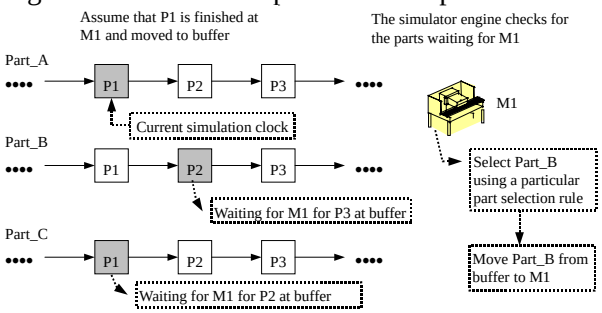

Figure 5. Conceptual situation of the part selection

6 Simulator Engine

While a simulation is running, the simulator engine retrieves the information needed to generate future events and make decisions from the process and resource models. The detail information flow directed to the simulator engine from two models is illustrated in Figure 6.

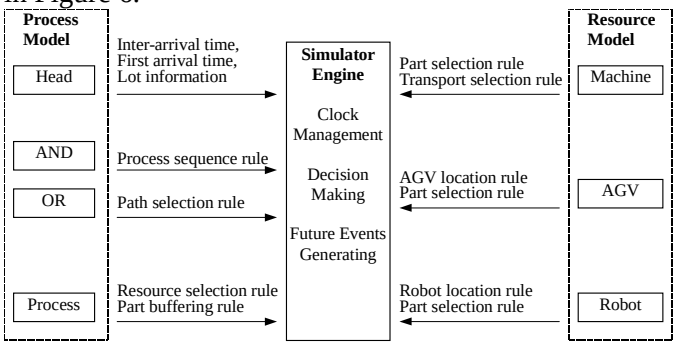

Figure 6. Information flow to the simulator engine

All the decision-making problems have precedence relationships among them, as shown in Figure 7. The main sequence of control flow is next process selection problem (path selection and process sequence problems) -> resource selection problem -> transport selection problem -> transport selection problem (robot and AGV location problem). When the processing resource or transport resource is busy, the part buffering problem and part selection problem in machine, robot, and AGV should be resolved.

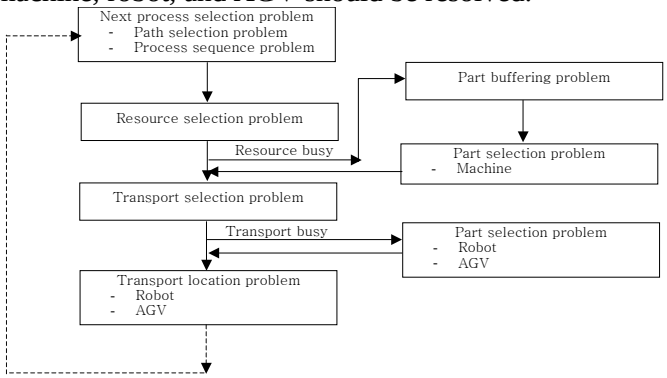

Figure 7. Relationships among decision-making

Based on the precedence relationships among the decision-making problems, the overall control flow of the simulator engine is shown in Figure 8. First, the simulator engine starts the operation by reading the process model and resolves path selection or process sequence problem to select the next process with the pre-specified rule. Then, the simulator engine reads the resource model and resolves the resource selection problem to select the next processing resource (machine). Based on the result, the simulator engine

resolves the transport selection problem to select the transport resource(s). To the end, the simulator engine generates the corresponding events and inserts them into the event list.

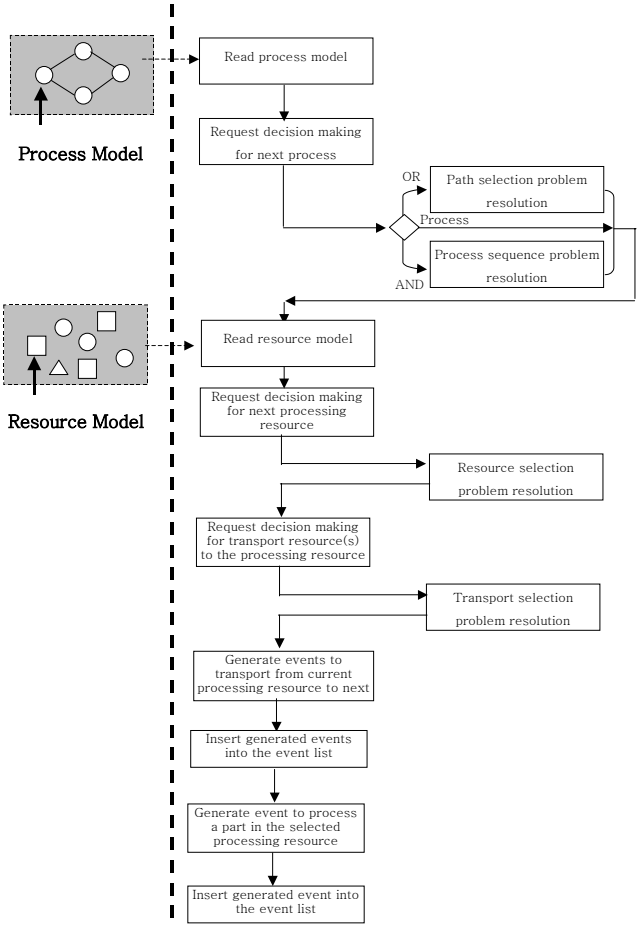

Figure 8. Overall control flow of the simulator engine

PRISM's user interface has four different windows: process model, resource model, simulator engine, and statistics results. The process and resource model windows are used to construct the process and resource models. The simulator engine window shows the evolution of the simulation as the clock advances. The exemplary process and resource model is shown in Figure 9.

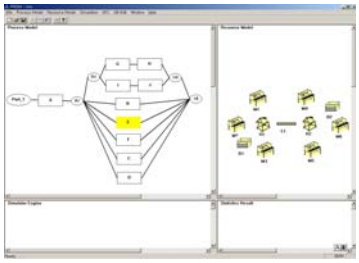

Figure 9. Constructed process and resource model

After running the simulation, the AND and OR junctions are resolved by the pre-specified process sequence and path selection rules. The job (Part_1) arrives at time 10 and proceeds to the "A" process, which is executed on machine M1 selected by the resource selection rule of process "A"("Min. waiting time" in this case). To move to machine M1, it resolves the transport selection problem by pre-specified rule in machine M1 ("Min. transport time" in this case) and selects robot R1. At time 12, it finishes the transportation and starts machining on machine M1. At time 22, it finishes machining and resolves the AND and OR junction by the process sequence and path selection rules. The simulator reads the process and resource models and uses the pre-specified rules to select the next process and resource -- in this case, process "D" (by the process selection and path selection rules: "Non-busy resource" in this case) on machine M2 (by the resource selection rule: "Min. waiting time") and robot R1 (by the transport selection rule: "Min. transport time") for the transportation from machine M1 to M2. This continues until the simulation ends. The resulting sequence of processes – "A"->"D"-> "G"-> "H"->"C"->"B"->"F"->"E"- and a detailed log of the progress and status of the simulation of a single job is shown in Figure 10.

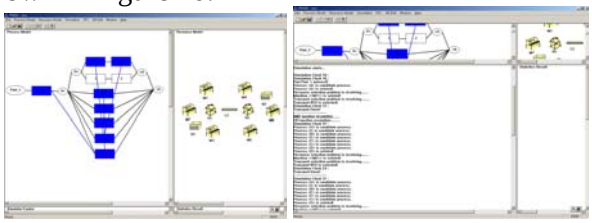

Figure 10. progress and status of simulator engine

The developed PRISM satisfies the aforementioned four requirements as follows: 1) The nonlinear process plan can be efficiently represented and utilized using the proposed symbols, 2) The decision-making problems can be identified and their resolution rules can be specified in the process and resource models, 3) It is easy to build, understand, and modify the simulation model since PRISM supports both process and resource views of SFCS behaviors. Furthermore, a detailed simulator engine scheme is insulated from the user.

7 Comparison and Conclusion

Because it integrates both a process view and a resource view, PRISM has several advantages over languages that support only one of these views:

1) Grammar: PRISM has a very simple grammar. The user just picks and places process and resource icons, and then defines their properties.
2) Model dynamics: PRISM represents processes and resources separately, which simplifies the construction of process plans and the capture of resource properties and shop layout. Since the sequence of blocks only represents the flow of entities, the process-oriented languages cannot represent the layout of resource. Resources (can be defined with Element Template in SIMAN/Arena) are just defined for the requiring blocks (can be defined with Block Template in SIMAN/Arena). Hence it is necessary to design the layout of resources further for the detail animation of a shopfloor (with Animation Template in SIMAN/Arena).
3) Part flow characteristics: PRISM captures part flow characteristics in the directed graph, which is generated automatically from the process and resources models. In languages that support a resource view only, the logic to control part flow is hidden in the resource descriptions.
4) Nonlinear process plan: PRISM is designed to define non-linear process plans, those with alternatives for both processes and resources, easily. In process-oriented languages, such plans must be implemented in user code, which is written in a commercial programming language or a language provided with the package. In some cases, it is procedural code attached as attributes to a part entity; and, in other cases, the plans are stored in external databases. In resource-oriented languages, each resource has its own PUSH and PULL logic to receive and send parts. In WITNESS, one of popular simulation packages with resource-oriented view, the part sequence logic is implemented in the "Machine" resources whose input-rule and output-rule are defined with the WITNESS Rule Editor. The WITNESS Rule Editor provides rules such as WAIT, PUSH and PULL, SEQUENCE, SELECT, IF, etc, along with user-defined ATTRIBUTES, VARIABLES and FUNCTIONS.

For example, Figure 11 shows a simple process plan and shop floor with 5 processes and 5 machines. If each process can be executed on any machine, the arrows show the possible movements of parts among the resources. In this case, the number of possible paths through the shop is 5! =120 and number of possible resource selections is $5^5 = 3125$. To represent such a number of part sequence logic may need a lot of user-woven codes when the users model the logic in the resource model only or in the process model only.

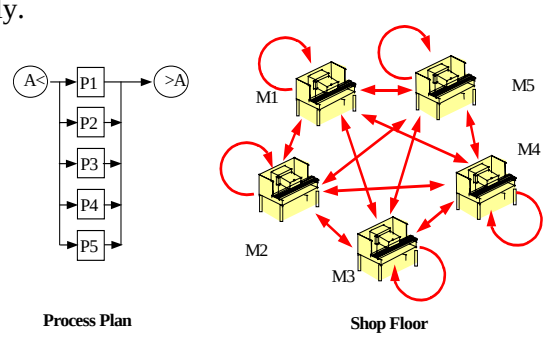

Figure 11. Exemplary process and resource model

This paper describes PRISM, a new simulation paradigm for the SFCS. This paradigm integrates process models and resource models. The major contribution of the paper is that the use of the two modeling views provides a much more tractable simulation modeling environment, which enables the system analysts to perform various off-line simulations rapidly and effectively to estimate the impacts of various control strategies of a shop floor. The analysts have only to capture the process model of produced parts and to construct the resource model of shop resources' properties and layout.

References:

[1] AT&T ISTEL, *Witness User Manual, Release 7.0*, Visual Interactive Systems, UK, 1995.
[2] AutoSimulations, Inc., *AutoMod II Users Manual*, Bountiful, Utah, 1989.
[3] CACI Products Company, *SIMFACTORY II.5 User's and Reference Manual*, Version 2.0, LaJolla, Calif., 1990.
[4] Cho, H., *An Intelligent Workstation Controller for Computer Integrated Manufacturing*, Ph. D. Dissertation, Texas A&M University, 1993.
[5] Gordon, G., *The Application of GPSS V to Discrete System Simulation*, Prentice-Hall, Englewood Cliffs, N. J., 1975.
[6] Law, A. M. and Larmey, C. S., *Introduction to Simulation Using SIMSCRIPT II.5*, CACI Products Company, La Jolla, Calif., 1984.
[7] Pegden, C. D., *Introduction to SIMAN*, Systems Modeling Corporation, Pennsylvania, 1982.
[8] Pritsker, A. A. B., *Introduction to Simulation and SLAMII*, Systems Publishing Corporation, 1986.
[9] Production Modeling Corporation of Utah, *ProModel User's Manual*, Orem, Utah, 1989.