Evolution Structure of a Process and Resource Models-based Simulation for the Supply Chain Management

PYOUNG YOL JANG
Science & Technology Policy Institute (STEPI)
26th Fl., Specialty Construction Center 395-70, Shindaebang-dong, Seoul 157-714
REPUBLIC OF KOREA

Abstract: Many commercial and academic simulation tools have been released to capture the operating strategies necessary to enhance the efficiency of the supply chain. However, the supply chain's dynamic nature caused due to the concurrent flow of various parts as well as sharing of different types of resources requires inevitable hand-woven codes to fully implement all the activities in modeling simulation. This prohibits the system analysts from rapidly and effectively modeling and analyzing the supply chain. Generally, the supply chain consists of various types of subsystems like manufacturing system, transportation system, and distribution. In particular the manufacturing system controls the production of the subassembly, assembly, and final product, which plays an essential role in the whole supply chain. Hence, the objective of the paper is to address the evolution structure of a process and resource models-based simulation useful for rapid supply chain analysis in the manufacturing system. This research will help overcome the disadvantages of all the existing simulators with regard to supply chain control and support automated solutions for complex scheduling, planning, and design problems.

Keywords: Supply chain management, Simulation, Manufacturing, Process and resource models

1 Introduction

Many commercial and academic simulation tools have been released to capture the operating strategies necessary to enhance the efficiency of the supply chain. However, the supply chain's dynamic nature caused due to the concurrent flow of various parts as well as sharing of different types of resources requires inevitable hand-woven codes to fully implement all the activities in modeling simulation. This prohibits the system analysts from rapidly and effectively modeling and analyzing the supply chain. Generally, the supply chain consists of various types of subsystems like manufacturing system, transportation system, and distribution. In particular the manufacturing system controls the production of the subassembly, assembly, and final product, which plays an essential role in the whole supply chain. A manufacturing system designed for discrete part manufacturing consists of various machining and material handling devices, such as CNC machine tools, robots, conveyor belts, and AS/RS. The manufacturing control software must be able to efficiently and reliably manage and coordinate those devices in order to ensure the completion of production orders placed from the business system. In particular, once receiving the process plans associated with the parts to be produced, the manufacturing control software is responsible for resolving several problems, such as process routing selection, resource allocation, workpiece scheduling, processing instructions downloading, progress monitoring, and errors detection and recovery [2]. Additionally, it must be able to cope with the dynamic reconfiguration of a shop floor, the concurrent part movement, and the unpredictable device failures. Although many methodologies have been proposed to resolve the above problems, more practical and efficient tools necessary to operate the manufacturing system in real-time have yet to appear. Recently, simulation became a promising tool useful for manufacturing sytem design and control due to its powerful and realistic capability of problem settlement.

To this end, many simulation languages have been evolving to meet various requirements of the manufacturing system design and control. General-purpose languages such as FORTRAN, BASIC, COBOL, PASCAL, C were initially used. However, these languages could not easily support modeling the manufacturing system, since the users should define the complex simulation logic for dynamic and concurrent part flows. They are usually used in education to understand the mechanism of simulation and in some special purpose simulation areas, such as, heat transfer analysis and atomic reaction analysis.

The characteristics of the first-generation and the second generation simulation software were described by the previous research [4]. The first-generation of simulation software including GPSS, SIMULA, SIMSCRIPT, SIMAN, and SLAM. was developed for general-purpose simulation [8][9], they have many defects in simulating manufacturing system control. A typical disadvantage is that they do not easily support the material handling properties of AGV, crane, robot, forklift, truck, cart, conveyor, etc. Furthermore, they have complex grammars to learn for programming the manufacturing system control environment. They model a system in a process-oriented view, implying that the sequential processes of the parts are hard-coded and the resources required to fulfill the processes are specified if necessary. This implies that the users cannot effectively figure out the resources' properties and their distributed layout. A prototype developed for manufacturing is MAP/1 [10], which provides text based simulation models.

The second-generation incluing WITNESS, FACTOR/AIM, AutoMod, ProModel, and SIMFACTORY has a user-friendly interface and support some features for the manufacturing system like material handling systems. The graphical animation makes modeling simple. They model systems in a resource-oriented view, implying that they view simulation models as the specification and arrangement of resources and process sequences are hidden within and across resources. This property prohibits the user from understanding part routings. The characteristics of the first and second-generation simulation software for manufacturing system control are summarized in Table 1.

To overcome the first and the second generation simulation software, the process and resource models-based approach has been proposed [4]. The objective of the paper is to address the evolution structure of a process and resource models-based simulation useful for rapid supply chain analysis in the manufacturing system. To this end, the paper describes the following: (1) the evolution structure of a process model, which represents complex and flexible process plans for producing parts, (2) the evolution structure of a resource model, which represents the characteristics and distributed relationships of various resources, (3) the evoluation structure of a 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.

Table 1. Characteristics of the first and second generation simulators

Characteristics	First generation	Second generation
Major viewpoint	Process-oriented view	Resource-oriented view
Process plan	Linear process plan	Not easy to represent
Material handling	Mostly not embedded	Mostly embedded
User interface	Not good	Good
Simplicity of usage	Not good	Good

2 Related Work

A commercial tool for generating the WITNESS simulation models from the process model is the PROSIM developed by Knowledge Based Systems, Inc. [5]. The PROSIM is the extended version of IDEF3 process capture method. The IDEF3 focuses on the abstract capture of knowledge about the processes occurring within a system [7]. It can capture and describe not what happens at this or that particular time in a system, but instead what fundamentally occurs in a system: the dynamic patterns that occur again and again among elements of a system. One major motivation behind the IDEF3 development was the need to speed up business process modeling and to capture the dynamics of business activities and process descriptions. The PROSIM is designed to enhance the productivity of business systems analysis, to facilitate design data life-cycle management, to support the project management process, and to facilitate the system requirement definition process. Since it is not designed to model the FMS operations, however, various manufacturing-specific processes and their precedence relationships cannot be appropriately expressed. For example, in the PROSIM grammar, a fan-out AND junction implies the splitting of objects flowing through, while a fan-in AND junction implies the assembly of objects. This semantic is not applicable to the feature-based machining activities in discrete-part manufacturing. Another disadvantage is the lack of resource representation. The PROSIM simply arranges all the fixed objects on the target simulator environment.

A program generator is the tool to aid in the production of computer-coded representations of a logical model. For example, a job shop simulation program generator (JSSPG) produces the simulation model written in SIMSCRIPT according to the questionnaire [3]. The DRAFT family receives the entity cycle diagram and then produces SIMSCRIPTII.5 codes [6]. A discrete-event simulation generator for operational systems (SGOS) acquires the operation network and equations and then generates the SIMAN code [11]. Recently, a simulation generator for dual card, kanban-controlled flow shops was suggested [1].

3 Framework

A combined process and resource models-based approach was proposed by Jang *et al.*[4]. This research adopts the concept diagam of this previous research as shown in Figure 1 [4]. The process plans related to the parts to be machined are represented as the process model, which is represented in an AND/OR graph form, in which a process node contains required resources and related decision-making rules, and an edge denotes the precedence relationships among processes. The resource properties and layout are represented as the resource model that is represented as the set of arranged icons of corresponding resources. The simulator engine reads the two models and then runs simulation by managing events on the basis of integrated view of the two models.

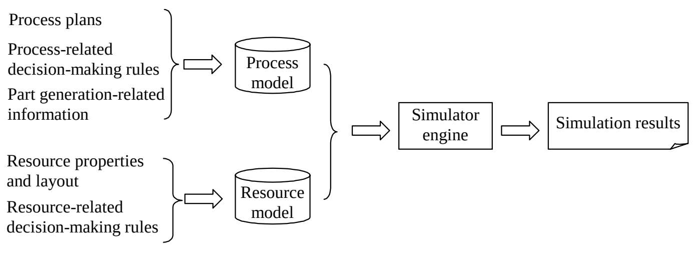

Process plans
Process-related decision-making rules
Part generation-related information
Process model
Resource properties and layout
Resource-related decision-making rules
Resource model
Simulator engine
Simulation results

Figure 1. Concept diagram of the process and resource models-based approach (Jang *et al.*, 2005 [4])

4 Evolution of the Process Model

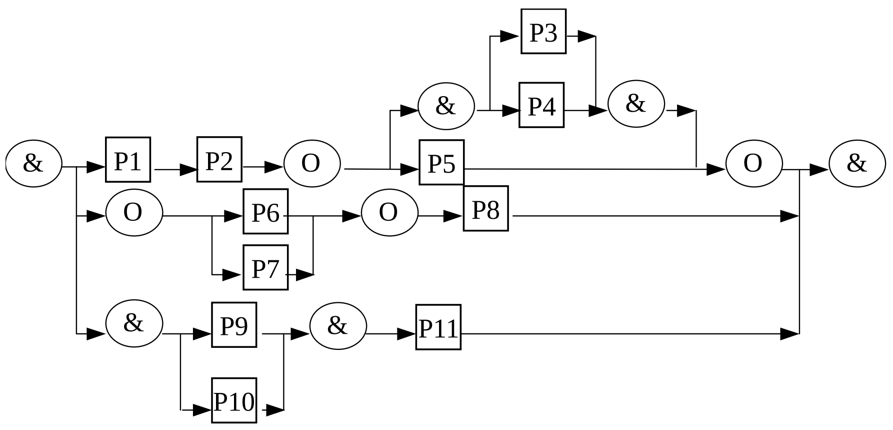
The serialization of processes surrounded with the AND junctions is called the process sequence problem. When the simulator engine hits the AND junction while scanning a process model, it selects directly the next process to be executed instead of serializing all the nodes following the AND junction. The user needs to specify a particular process sequence rule. For example, the 'minimum traveling time' rule selects the next process to be executed that needs the minimum travel time of the part from the current location. The process that can be executed by the same machine tool as the current location could be selected. The 'minimum set-up time' rule selects the next process that has the minimum set-up time required to prepare for the process.

As the simulator engine encounters the OR junction, it selects the specified number of paths or processes, which is defined as the path selection problem. If the number of paths is greater than 1, the selected paths are grouped with AND junctions, which can be resolved by the process sequencing rule. Hence, the user needs to specify the two particular rules associated with the path selection problem and the process sequence problem, respectively. For example, the 'maximum flexibility' rule selects the paths that have the largest number of AND junctions, and hence most processes can be sequenced later on. The 'resource load balancing' rule selects the paths that balance mostly the resource loading, and hence the paths that contain the more processes assigned to the less loaded machine tools.

The internal data structure of the process model consists of the tables for storing individual heads, processes, AND junctions, and OR junctions, and the sets for storing the precedence relationship among processes. In particular, the simulator engine maintains the three kinds of sets, such as the process candidate set, the process alternative set, and the process order set. The process candidate set, expressed as square bracket [ ], contains all the processes to be executed between two AND junctions without any precedence constraints being violated. According to the process sequence rule specified by the users, a candidate process is selected from the set. The process alternative set, expressed as brace { }, contains all the paths between the corresponding OR junctions. The simulator engine then selects the specified number of paths according to a particular path selection rule and then picks the next process to be executed according to a particular process sequence rule. The process order set, expressed as round bracket ( ), contains the sequenced processes. Every path following the junction must be mapped to a process order set. It may contain an element. Only the first process to be executed will be selected. It should be noted that a set might contain other sets as members. An exemplary process model and its corresponding set are shown in Figure 2.

The process model and its related set shown in Figure 2 is resolved according to the above procedures as shown in Figure 3. It starts with invoking the AND_resolution() procedure, since the set belongs to the process candidate set. The first elements of the each process order set are checked whether the three ‘if’ conditions are satisfied. The first element of the first process order set, P1, does not meet any ‘if’ condition. The simulator engine then checks the first element of the second process order set, {P6, P7}. Since it meets the second ‘if’ condition, the OR_resolution() procedure is invoked. Assume that the specified number of processes to be selected is 1 and process P6 is selected by the rule specified in the OR junction. Set {P6, P7} is replaced by process P6 and process P7 is removed from the set. The simulator engine checks again for the first element of the third process order set, [P9, P10]. Because it satisfies the first ‘if’ condition, the function Candidate() is invoked. Assume that it returns process P9. Process P10 is removed and inserted into the stored process set. Finally, the first element of the third process order set satisfies the third ‘if’ condition, one process is selected out of process, P1, P6, and P9 according to the process sequence rule specified in the AND junction. Assume that P9 is selected. Process P9 is then removed from the set, and its corresponding process P10 stored in the stored set replaces process P9. The bold process symbols in the selected process imply the executed process and the non-bold process symbols imply the dummy process that is selected for decision-making in the set resolution.

The simulator engine continues to take some actions necessary to perform process P1. For example, the simulator engine selects a machine tool according to the resource selection rule specified in the process model. If the selected resource is idle and is different from the current part location, the simulator engine will try to find material transport resources specified in the resource model. If the selected resource is not idle, the simulator engine determines whether the part is moved to the buffer or stays at the current location. If the selected resource is the current part location, the simulator starts to perform process P1. At this moment, the resolution procedure stops temporarily and looks up the next event in the event list. The simulator engine continues to resolve the set if the popped event is associated with the selection of the next process.



[(P1, P2, {([(P3), (P4)]), (P5)}), ({(P6), (P7)}, P8), ([(P9), (P10)], P11)]

Figure 2. Sample process model and its corresponding set

Functions	Set	Selected Process	Stored Process Set	Future events
	[(**P1**, P2, {([(P3), (P4)]), (P5)}), (**{(P6), (P7)}**, P8), (**[(P9), (P10)]**, P11)]		< >	
OR_resolution()	↘ {(P6), (P7)} ⟶	P6	< >	
	[(**P1**, P2, {([(P3), (P4)]), (P5)}), (**P6**, P8), (**[(P9), (P10)]**, P11)]		< >	
Candidate()	↘ [(P9), (P10)] ⟶	P9	<P10>	
AND_resolution()	[(**P1**, P2, {([(P3), (P4)]), (P5)}), (**P6**, P8), (**P9**, P11)]	**P9**	<P10>	Create future events for P9
AND_resolution()	[(**P1**, P2, {([(P3), (P4)]), (P5)}), (**P6**, P8), (**P10**, P11)]	**P1**	< >	Create future events for P1
AND_resolution()	[(**P2**, {([(P3), (P4)]), (P5)}), (**P6**, P8), (**P10**, P11)]	**P2**	< >	Create future events for P2
	[(**{([(P3), (P4)]), (P5)}**), (**P6**, P8), (**P10**, P11)]		< >	
OR_resolution()	↘ {[(P3), (P4)], (P5)} ⟶	P5	< >	
AND_resolution()	[(**P5**), (**P6**, P8), (**P10**, P11)]	**P6**	< >	Create future events for P6
AND_resolution()	[(**P5**), (**P8**), (**P10**, P11)]	**P5**	< >	Create future events for P5
AND_resolution()	[(**P8**), (**P10**, P11)]	**P10**	< >	Create future events for P10
AND_resolution()	[(**P8**), (**P11**)]	**P8**	< >	Create future events for P8
AND_resolution()	[(**P11**)]	**P11**	< >	Create future events for P11
	[ ]		< >	

Figure 3. Sequence of fired rules after process P1

5 Evolution of the Resource Model

Since the resources in the manufacturing system have interactions to complete the assigned part, the relationship among resources must be closely investigated and then the appropriate parameters captured in defining the resource model are extracted. A resource can be classified into the three types according to its functionality: processing, storage, transport resource which was proposed by Jang *et al.* [4]. The processing resource represented as the machine in the resource model performs the given process, such as milling, drilling, boring, etc. The storage resource can be further classified into buffer and AS/RS. While a buffer stores temporarily the parts in process, an AS/RS stores the parts and raw materials for a sufficiently long time. It is assumed that a part comes out of an AS/RS and it enters the AS/RS after all processes are finished. The transport resource can be further classified into material handler and material transporter. The former includes a robot, which can pick, move, and put parts. The latter includes an AGV and a conveyor, which can only be used to move parts. In other words, the material handler needs to serve the material transporter in order to load and unload parts.

6 Evolution of the Simulator Engine

Initially, the simulator engine examines the head symbols of all the parts in the process model and then obtains the first arrival time of each part to create 'part_arrival' events to be added in the event list. It also inspects the resource model and then obtains the first breakdown time of each resource to create the 'break' events to be added in the event list. The simulator engine then starts the simulation cycle: 1) to obtain the first occurring event, 2) to advance the simulation clock with respect to the event, 3) to execute the event, and 4) to generate the associated future events and put them in the event list. The simulation cycle will be repeated either until the specified simulation time is over or until the predetermined number of parts is produced. The major events used in the simulator engine, their associated future events, and their descriptions are summarized in Table 2.

Although the events are mostly associated with the status of resources, the part that is finished on a particular machine tool actively searches for the next machine according to its process model, instead of being searched by the empty machines. Hence, the part that cannot find the next machine is added to the part list. The machine that has just finished and sent out a part will try to find a part from the part list without regard to the current location of the part. In other words, the part that is finished on a machine looks for the next machine, but it stays at the current machine or moves to the buffer and then is added to the part list, while the machine that becomes empty looks for a part from the part list.

7 Conclusion

This research addresses the evolution structure of a process and resource models-based simulation useful for rapid supply chain analysis in the manufacturing system. This will help overcome the disadvantages of the existing simulators with regard to supply chain control and support automated solutions for complex scheduling, planning, and design problems.

Table 2**.** Major event list

Event	Associated future events	Description
part_arrival	part_arrival, pick_done,	A part arrives at the system, which inserts the next arrival in the event list
pick_done	put_done	A robot moves and picks a part up
put_done	setup_done, process_done, receive_done, conveyor_done, AGV_done	A robot moves and puts a part down
robot_break	robot_break, robot_up	Breakdown occurs in a robot
setup_done	process_done	The machine setup is finished
process_done	pick_done	A process is finished
machine_break	machine_break, machine_up	Breakdown occurs in a machine
tool_break	tool_break, tool_up	Breakdown occurs in a tool
retrieve_done	pick_done	An AS/RS retrieves a part
AS/RS_break	AS/RS_breakdown, AS/RS_up	Breakdown occurs in an AS/RS
conveyor_done	pick_done	Transport through a conveyor is finished
conveyor_break	Conveyor_break, conveyor_up	Breakdown occurs in a conveyor
AGV_done	pick_done	Transport through an AGV is finished
AGV_break	AGV_break, AGV_up	Breakdown occurs in an AGV

References:

[1] Christenson, K. R. and Dogan, C. A., "A simulation generator for dual-card kanban-controller flow shops", *International Journal of Production Research*, Vol. 33, pp. 2615-2631, September 1995.

[2] Cho, H., *An Intelligent Workstation Controller for Computer Integrated Manufacturing*, Ph. D. Dissertation, Texas A&M University, 1993.

[3] Ginsberg, A. S., Markowitz, H. M., and Oldfather, P. M., "Programming by questionaire", *Rand Memorandum RM-4460-PR*, The Rand Corporation, Santa Monica, California, April, 1965.

[4] Jang, P. Y., Jones, A., and Cho, H., "A Combined Process/Resource Models-based Approach to Shop Floor Simulation", *WSEAS Transactions on Computers*, Vol. 4, No. 9, pp. 1062~1072, 2005.

[5] KBSI, *PROSIM User's Manual*, Version 2.2.1, Knowledge Based Systems Inc., 1996.

[6] Mathewson, S. C., "The application of program generator software and its extensions to discrete event simulation modeling", *IIE Transactions*, Vol. 16, pp. 3-18, March 1984.

[7] Mayer, R. J., Cullinane, T. P., deWitte, P. S., Knappenberger, W. B., Perakath, B., and Wells, M. S., *IDEF3 Process Description Capture Method Report*, KBSI, Texas, 1992.

[8] Pegden, C. D., *Introduction to SIMAN*, Systems Modeling Corporation, Pennsylvania, 1982.

[9] Pritsker, A. A. B., *Introduction to Simulation and SLAMII*, Systems Publishing Corporation, 1986.

[10] Rolston, L. J., "Modeling flexible manufacturing system with MAP/1", Annals of Operations Research, Vol. 3, pp. 189-204, December, 1985.

[11]Yuan, Y., Dogan, C. A., and Viegelahn, G. L., "A flexible simulation model generator", Computers and Industrial Engineering, Vol. 24, pp. 165-175, 1993.