특별 기고 <해외편>

Modeling and Simulation of Intelligent Hierarchical Flexible Manufacturing

Tae Ho Cho* Bernard P. Zeigler** Hee Suk Seo***

Content

1. Introduction.

Many Researchers and practitioners have expressed the view that artificial intelligence(AI) may have significant application to solution of manufacturing problems. Expert systems have been developed for solving problems of varying complexities in several problem areas. There are potential applications of AI and expert systems at every stage of the manufacturing process. Applications that have achieved some success include planning, scheduling, controlling, maintenance and fault diagnosis[1] - [3]. Matsuo [4] has developed a knowledge-based intelligent crane scheduling system for controlling a stacker crane in a computer-integrated manufacturing environment. O'Grady[5] developed an intelligent cell control system for automated manufacturing. However, while there is an increasing literature on AI-based manufacturing tools[3],[6]-[13], the number of systems in daily use by manufacturing engineers is still small.

To deliver a working AI-based scheduler into a manufacturing environment requires attention to system issues. Simulation, already one of the most widely used methodologies in manufacturing design and analysis[14], [15], can be applied to test the effectiveness and efficiency of AI-based tools before they are introduced into existing environments.

However, the models used to study large-scale systems tend to be very complex, and writing simulations to execute them can be an extremely arduous task. This is especially true when system behaviors needed to test AI-based tools, as well as representations of the tools themselves, must be included in the model. Thus, it is understandable that current knowledge-based simulations of manufacturing systems typically address only a small part of the total system. Forexample, an expert system for stacker-crane control in a manufacturing environment[4] controls only a

* Professor of Sungkyunkwan Univ.
** Professor of Arizona of Univ.
*** Doctorial student of Sungkyunkwan Univ.

single crane moving among several workstations. Another simulation model for dispatch rule assessment[2] simply consists of two workstations with a queue attached to each. O'Grady's[5] application of the black board architecture for hierarchical control at four levels factory, shop, workcell and machine includes neither a manufacturing model nor an actual manufacturing system for testing and analysis of the AI technology.

In previous work we have developed a knowledge-based environment to support Modeling and simulation of manufacturing systems for purpose of testing AI-based production control tools[16], [17]. The objective of this article is to demonstrate the utility of such a simulation environment by illustrating its application to the verification of usefulness of an expert system for routing the batches of jobs, that are splitted[18] for improved lead time, within the context of multi-level hierarchical flexible manufacturing system(FMS). For the batch job splitting the technique of operation overlapping is studied. The rules in the routing expert system are written such that the advantages obtained by job splitting can be maintained throughout the entire routing process in the hierarchical manufacturing structure. The simulation is able to identify the operating regime in which operation overlapping succeeds in reducing manufacturing lead time and work in process.

Before proceeding, we briefly review the simulation environment concepts and features needed to explicate the operation overlapping case study.

2. Backgrounds.

Cho[16], [17] presented and approach to embedding expert systems within an object oriented simulation environment. The basic idea was to create classes of expert system models that can be interfaced with other model classes. An expert system shell was developed within Discrete Event System Specification implemented in Scheme language(DEVS-Scheme)[19], [20], a knowledge-based design and simulation environ -ment which combines artificial intelligence and system modeling concepts. The new shell enables interruptible and distributed expert systems to be defined as components of simulation models. This facilitates simulation modeling of knowledge-based controls for flexible manufacturing and many other autonomous intelligent systems. Moreover, the structure of a system can be specified using a recursive system entity structure (SES)[19], [21] and unfolded to generate a family of hierarchical structures using an extension of SES pruning called recursive pruning. This recursive generation of hierarchical structure is especially appropriate for design of multilevel flexible factories.

2.1 Expert System Shell for Simulation Environment.

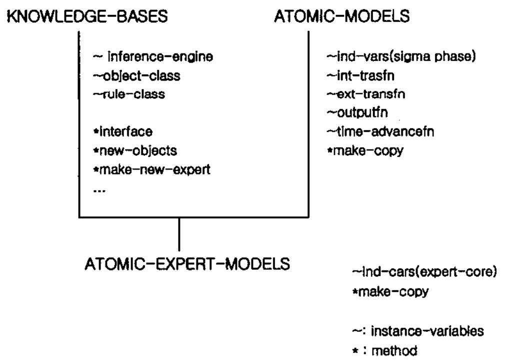
The Distributed Expert System Environment (DESE) is an environment in which expert system models can have distributed control (inference engine) and knowledge(rules and facts). The DESE consist of object classes, methods and other utility functions for creating distributed expert systems(DES's). The DES's created under the DESE provide the well known benefits of object oriented programming such as the ease of reuse, modularity, and extendibility [22] - [24].

The objects involved in the definitions of a DES are instances of knowledge-base, inference-engine, rules, and facts classes. An inference-engine instance refers to an associated knowledge-base instance which has links to the relevant rules and facts. In this way, the inferencing methods of the class inference-engines or its subclasses are generic(i.e., can be used for rules and facts stemming from a variety of classes). The DES is formed from instances of not only inference-engines class and knowledge-bases class but also subclasses of these classes depending on the problem domain. In this case the DES's are truly distributed in both control and knowledge. The DESE also provides a class for facts instances, in which the facts are created based on fuzzy logic. For more information refer to [16].

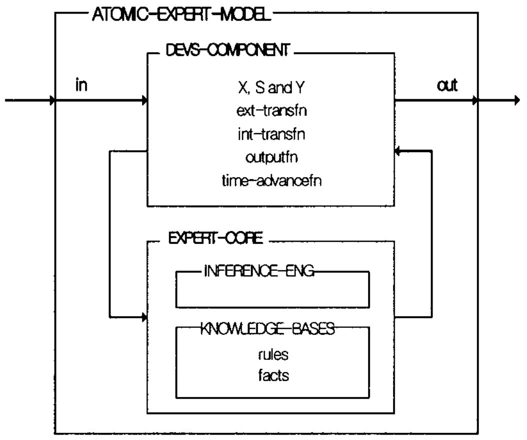
The interface of DESE to DEVS-Scheme is accomplished by creating classes called atomic-expert-models. This class(multiply) inherits methods and variables from knowledge-bases and atomic-models, an existing, fundamental class of DEVS-Scheme(Figure 1). Thus, as illustrated in (Figure 2), an instance of class atomic-expert-models has both a DES component (called as expert-core) and an atomic-model component (called as DEVS component). Such models can be duplicated by invoking the make-copy method which produces isomorphic copies[25].

The merger of DESE and DEVS-Scheme via inheritance extends the latter's facilities for hierarchical, modular discrete-event model construction to include design and testing of distributed expert system components. Since expert system components are treated no differently than other DEVS models, once the system entity and model bases have been constructed, it is relatively easy to generate alternative model architectures.

2.2 Interfacing DESE and DEVS-Scheme.

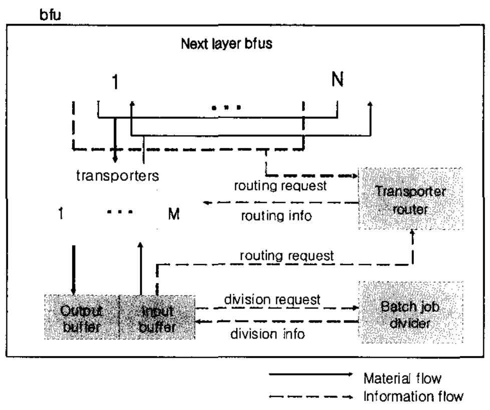


(Figure 1) Interface of DES to DEVS-Scheme



(Figure 2) Composition of atomic-expert-model

3. Intelligent Hierarchical FMS Modeling

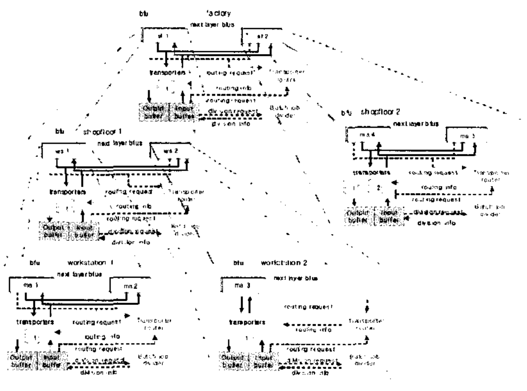


(Figure 3) Basic fractal unit architecture

Cho [16], [17] presented a design technique for generating recursive system entity structures. It is applied to represent the fractal architecture for flexible manufacturing introduced by Tirpak et. al [26]. Recall that the system entity structure (SES) directs the synthesis of models from components in a model base [19], [21].The SES is a knowledge representation scheme that combines the decomposition, taxonomy, and coupling relationships. An operation called recursive pruning was developed for generation hierarchical model structures with properties similar to fractals.

Tirpak et. al [26] argue that, cast in recursive form, a model of an FMS admits of a natural hierarchical decomposition of highly decoupled units with similar structure and control. Theobjective of such structuring is to manage the structural complexity and coordination of an FMS hierarchical by maximizing local functionality and minimizing global control. Moreover, the recurring components can be designed within the object-oriented paradigm so as to maximize reuse across levels. Thus the FMS fractal architecture model represents a hierarchical structure built from elements of a single basic design called a basic fractal unit(BFU). The design of the BFU incorporates a set of pertinent attributes that can fully represent any level in the hierarchy.

(Figure 3) depicts a BFU specifically designed to embody the elements which fully describe the structure of any level in the model hierarchy. Included within a BFU is a set of lower layer BFU's whose internal detail is hidden. The routing controller (transporter router) sees these units as stations to which transfer batches should be delivered. It is the responsibility of transporter routers within the lower layer BFU's to subsequently route the received batches.



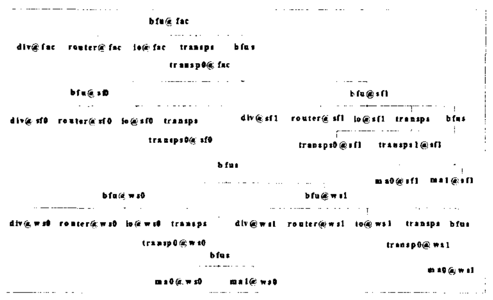
(Figure 4) Fractal architecture pruned to have five BFU

3.1 Fractal Architecture Modeling

(Figure 4) illustrates a three level fractal architecture. The top of the hierarchy is a factory. The factory is composed of two hop floors. The first shop floor(shopfloor 1) in turn has two workstations, workstation1 and workstation2. Whereas, the second shop floor has two machines without having any workstation. Workstation1 has two machines and workstation2 has just one machine.

3.2 Simulation Scenario

The simulation starts with the generator sending a transfer batch job to the buf@fac model which consists of job-id, job-type, batch-size and sequence schedule. The sequence schedule contains operations of the job to be performed in sequence, e.g., (a b c d e), (a b c d) and so on, here a, b, c, d ande represent different types of operations. The in-buf@fac (in-b model of buf@fac) converts the sequence schedule of operations to sequence schedule composed of machine lists based on an operation-type -to-machine table, e.g., a is converted to ma0@ws0, b to mal@sw0, c to ma0@wsl, d to ma0@sfl, and e to mal@sfl (Figure 5 is the pruned entity structure that generates the actual simulation model and shows where these machines belong).



(Figure 5) Pruned system entity structure of bfu

The divider then divides(forks) this transfer batch. The next section will study the performance of a divider working in accordance with division formula given by operation overlapping(1). These divided batches, or splitted batches, are routed to the next machine by router according to the schedule. The router routes the transporters in each BFU based on several dynamic factors. The divided batches are gathered (joined) later on at the same io model (out-b within io) where they were divided (forked). When all the machines in the schedule are traversed the batch job leaves the bfu@fac model (factory level bfu) and is sent to the transducer for recording statistic on the bfu@fac's performance (throughput, flow time, work in process (WIP) and average inter-arrival time).

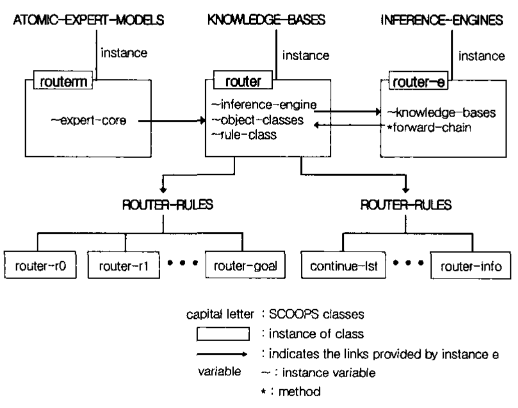
The size of a transfer batch is determined by a uniform distribution function. Its maximum and minimum batch sizes generated at generator is 500 and 100, respectively. The sequence schedule of job types are picked randomly with equal probability among the following schedules: (a b c d e), (a b d e), (a c d e), (b d e) and (a c e). Table 1 contains the processing times given to machines according to job types. The processing time per unit ranges from 2 to 15 and the setup time is ten times the processing time per unit. The remaining processing times are as follows (all values are in arbitrary time units):

- transporters: 100 (fac level),
  80 (shop floor level),
  30 (work station level)
- router : 1
- divider: 0

3.3 Transporter Router

The transporter router is an expert system model which controls touting of transporters in each BFU. It routes transporters among the lower layer BFU's, output buffer and input buffer. There are a total of five such DES models (one for each BFU) whose inferencing is based on their own sets of facts, responding to the different dynamic conditions encountered at each level(Figure 4).

The routing decision is made based on, for example, whether the sub-batch being routed is the first sub-batch or the second sub-batch. If it is the second sub-batch it should have higher priority than the first to get the advantages from the operation overlapping batch job division algorithm as explained in Section ?. There are other factors that affect the routing decision. The overall routing logic and the rule descriptions are shown in the Appendix. The DESE classes used to specify the router expert system are depicted in (Figure 6).



(Figure 6) Classes in the construction of expert router

〈Table 1〉 Processing and setup times of machines according to job types

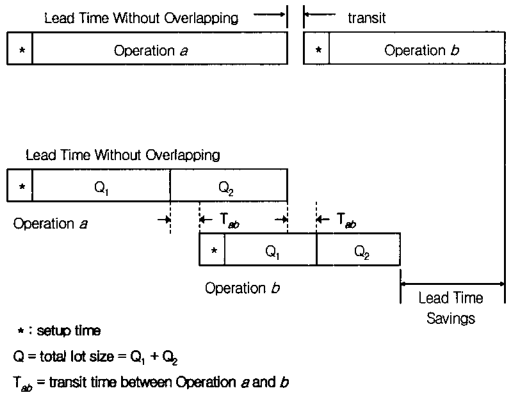
Machine name	Job type	Setup time	Processing Time per unit
ma0@ws0	job1	30	3
〃	job2	50	5
〃	job3	60	6
〃	job4	50	5
〃	job5	20	2
ma1@ws0	job1	30	3
〃	job2	100	10
〃	job3	100	10
〃	job4	50	5
〃	job5	40	4
ma0@ws1	job1	30	3
〃	job2	50	5
〃	job3	60	6
〃	job4	50	5
〃	job5	20	2
ma0@sf0	job1	50	5
〃	job2	100	10
〃	job3	100	10
〃	job4	100	10
〃	job5	80	8
ma1@sf1	job1	50	5
〃	job2	150	15
〃	job3	100	10
〃	job4	100	10
〃	job5	50	5

(All numbers are in the time units)

4. Example: Operation Overlapping

A simple task called operation overlapping will be studied within the fractal architecture framework. Operation overlapping is implemented by the batch job divider model in (Figure 3). The batch job divider divides a transfer batch into two sub-batches. The size of these two sub-batches are determined according to operation overlapping. The transporter router(expert system model) routes these divided sub-batches according to the routing logic contained in the rules of router model. The rules in the expert system are written considering the hierarchical nature of fractal architecture and the nature of sub-batches. For example the sub-batches from a batch should be routed consecutively to reduce setup time and this fact should be maintained in routing down the hierarchy as well ad within the same hierarchy.

Operation overlapping is a part of production activity control(PAC). The function of PAC is to have activities performed as planned[27]. The physical layout of a job shop usually groups equipment(e. g., manufacturing cell, workcell, workstation) used in performing similar functions in the same area. Typically, there are many different orders being processed in the plant at the same time and relatively few have the same routing. Scheduling of a shop floor, the first phase of PAC, involves allocating jobs to work centers for meeting due dates, maximizing machine utilization, minimizing setup times, minimizing lead times or any combination of these [13], [27], [28].


Lead Time Without Overlapping
transit
Operation a
Operation b
Lead Time Without Overlapping
Q1
Q2
Operation a
Tab
Tab
Q1
Q2
Operation b
Lead Time Savings
* : setup time
Q = total lot size = Q1 + Q2
Tab = transit time between Operation a and b

(Figure 7) Operation overlapping

Operation overlapping, schematically re -presented in (Figure 7), is a technique performed on the batches being transferred according to the given plans. It is used to reduce the lead time by dividing the lot into two or more batches and linking at least two successive operations directly(one is performed immediately after the other). Operation overlapping is a common practice in manufacturing lead time by the reduction of the individual operation throughput times. Usually the major savings from overlapping come from reducing the time a lot waits in a queue between operations often several times greater than total processing time. Also associated with manufacturing lead time reduction is the reduction in work in process(WIP) and the increased ability to meet product due dates. The disadvantages are the added cost of increased control required by doubling the number of batches and material movements. Melnyk [18] briefly explains operation overlapping and Fogarty et. al. [27] offer a rather detailed exposition with some examples.

Consider two serial operations, A and B. Then operation overlapping consists of the following:

1) A lot of parts if divided into at least two batches(transfer batches).
2) As soon as the first batch completes Operation A, it is moved to Operation B for immediate processing.
3) While Operation A is being performed on the second batch, Operation B is being performed on the first batch.
4) When Operation A has been completed on the second batch, it is moved immediately to Operation B.

If Operation B requires substantially less time per piece than Operation A, the first batch should be sufficiently large to avoid idle time at Operation B. Calculation of this minimum batch size is shown as Q1 in (Figure 7) [27].

$$Q1 >= (QPa\ \ Sb) / (Pb + Pa) \qquad (1)$$

where

Q	Total lot size = Q1 + Q2.
Q1	Minimum size of first batch.
Q2	Maximum size of second batch.
Sb	Setup time of Operation B.
Pa	Processing time per unit, Operation A.
Pb	Processing time per unit Operation B.

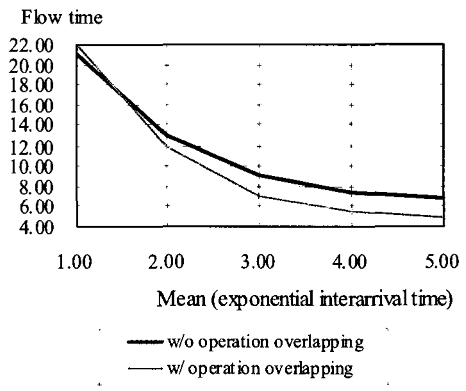
5. Result of Simulation

Two different simulations were performed: one with dividing the transfer batches according to operation overlapping and the other without division.

<Table 2> Average and 90% confidence intervals on flow times.

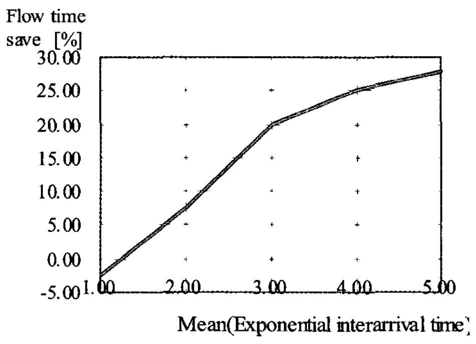
mean interarrival time (exponential distribution)	w/ overlapping avg. [conf. interval]	w/o overlapping avg. [conf. interval]
1000	21906 [14890, 28912]	21477 [15902, 27033]
2000	11467 [5205, 17728]	12528 [7093, 17962]
3000	7316 [2975, 11656]	9115 [5292, 12937]
4000	5555 [3951, 7158]	7448 [5642, 9253]
5000	4850 [4140, 5559]	6790 [6702, 6877]

The exponential distribution function was used for the job interarrival process[14]. To observe how performance is affected by the frequency of job arrivals we ran five mean interarrival times: 1000, 2000, 3000, 4000, and 5000. Figure 8 compares the observed lead time with, and without, operation overlapping. The 90% confidence intervals are shown in <table 2>.

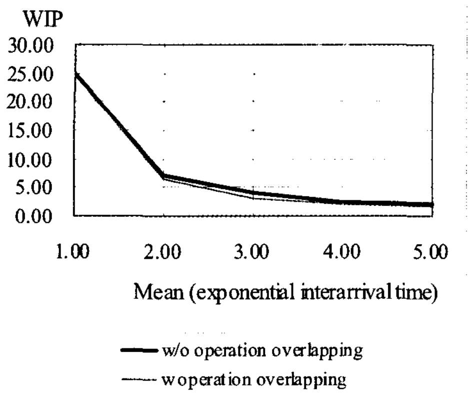


(Figure 8) Comparison of flow times

(Figure 9) shows the saving in flow time when operation overlapping is applied. The graph shows about 30% of saving in flow time when interarrival time is large. The flow time with operation overlapping becomes closer to the lead time without it, as interarrival frequency increases. The major reason for this convergence is that utilization of machines increases as arrival frequency increases, thus the advantage of increased utilization disappears, i.e., the operation overlapping is better conditional on the arrival rate. The WIP and production rate comparisons are shown in (Figure 10 and 11), respectively.

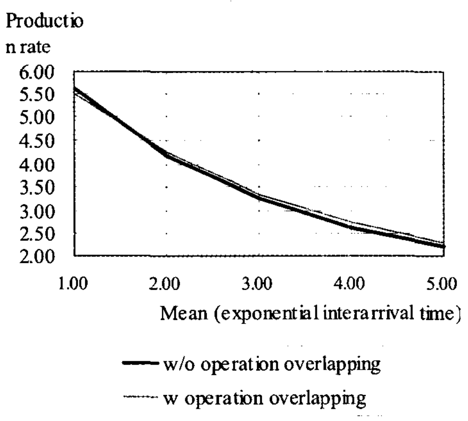


(Figure 9) Saving in flow time



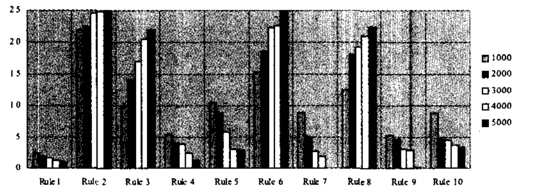
(Figure 10) Comparison of WIP.

(Figure 12) shows the statistics relating to when particular rule has been fired. Y-axis of the figure is the percentage of fired rulesand x-axis is the rule number. There are five different percentage values for each rule which represents the simulation results of five different interarrival times. The five different interarrival times are shown in the small box right side of (Figure 12).The important rules to be observed are rule number 5 and 7. The percentage of firing of rule 5 and rule 7 increas as the interarrival time decreases.



(Figure 11) Comparison of production rate.

These results are the major reasons for the reduction of the advantage of operation over-lapping. He increase in number of firing of rule 5 and rule 7 tells that more first sub-batches have to wait until the destination receives the complete batch, i.e., the first sub-batch and the second sub-batch. He statistics on other rules follow the simulation results shown in (Figure 8-11).



(Figure 12) % of fired rules with operation overlapping applie

6. Conclusion

The immediate advantage of applying operation overlapping with proper routing logic for the splitted batches is the reduction of flow times which in turn reduces WIP. For the fractal architecture, a second advantage is apparent: smaller transportation devices can be used. Since the sizes of transfer batches are reduced as one descends the hierarchy from factory to shop floor, work station and machine, smaller capacity transporters can be employed at lower levels. Without job splitting with operation overlapping the size of the batches remains unchanged as the batches are processed. This means that the transporters at the lower level must be as large as the ones at the factory level. The disadvantage is, however, the need for added control of dividing (fork) and combining(join) the batches at each BFU.

The case study reported here demonstrates the use of simulation in verifying the usefulness ofproposed expert system controls for factory activity. Without proper methodology for constructing multiple expert system models as shown in this paper the activity of writing simulation models is not only very difficult and much more time consuming but also the reusability and testability of the model decreases. It also illustrates how simulation investigation can delineate the region in which a proposed technique can yield the anticipated improvement in performance for certain arrival rates of input batches. It is a straightforward next step to incorporate this knowledge into the rules of the expert system itself, to determine the beneficial envelope as a function of the prevailing conditions of the factory.

References

[1] S. D. Wu, Artificial intelligence and scheduling application, Artificial Intelligence: Manufacturing Theory and Practice, S. T. Kumara, Ed., Norcross, GA, 1989.

[2] H. Pierreval and H. Ralambondrainy, A simulation and learning technique for generating knowledge about manufacturing systems behavior, Artificial Intelligence in Operational Research, G. I. Doukidis and R. J. Paul, Eds. New York: Macmillan, 1992.

[3] A. Kusiak, Expert systems and optimization in automated manufacturing systems, in Artificial Intelligence: manufacturing Theory and Practice , S. T. Kumara, Ed., Norcross, GA, 1989.

[4] H. Matsuo, J. S. Shang, and R. S. Syllivan, "A knowledge-based system for stacker crane control in a manufacturing environ -ment," IEEE Trans. Syst., Man, Cybern., vol. 19, pp. 932-945, Sept./Oct. 1989.

[5] P. J. O'Grady and K. H. Lee, An Intelligent Cell Control System for Automated Manufacturing. New York: Taylor & Francis, 1989.

[6] J. Erschler and P. Esquirol, "Decision-aid in job shop scheduling: A knowledge based approach," in Proc. 1986 IEEE Int. Conf. On Robotics and Automation, San Francisco, CA, pp. 1651-1656.

[7] M. S. Fox, "Constraint-directed search: A case study for job-shop scheduling," Ph.D.dissertation, Carnegie Mellon University, Pittsburgh, PA, 1983.

[8] S. Subranmanyam R. G. Askin, An expert system approach to and in flexible manufacturing system, "Flexible Manu -facturing Systems: Methods and studies, A. Kusiak, Ed. Amsterdam, The Netherlands: North Holland, 1986.

[9] R. J. Paul and G. I. Doukidis, Operational research approach to artificial intelligence in production planning and scheduling, Artificial Intelligence in Operational Research, G. I. Doukidis and R. J. Pauln, Eds. New York: Macmillan, 1992.

[10] P. Duchessi and R. M. O'Keefe, A knowledge-based approach to production planning, Artificial Intelligence in Operational Research,. G. I. Doukidis and R. J. Paul, Eds. New York: Macmillan, 1992.

[11] M. S. Steffen, "A survey of artificial

intelligence-based scheduling system," in Proc. 1986 Fall Industrial Engineering Conf., pp. 395-405.

[12] K. Kempf, B. Russell, S. Sidhu, and S. Barrett, "AI-based schedulers in manu-facturing practice: Report of a panel discussion," AI Mag., (Special issue), pp., 46-55, 1991.

[13] A. Kusiak, Intelligent Manufacturing Systems. Englewood Cliffs, NJ: Prentice Hall, 1990.

[14] M. L. Law and W. D. Kelton, Simulation Modeling & Analysis, 2nd ed. New York: McGraw-Hill, 1991.

[15] R. G. Askin and C. R. Standridge, Modeling and Analysis of Manufacturing Syste. New York: Wiley, 1993.

[16] T. H. Cho, "A hierarchical, modular simulation environment for flexible manufacturing system modeling,"Ph. D. dissertation, Univ. of Arizona, Tucson, 1993.

[17] B. P. Zegiler, T. H. Cho, and J. W. Rozenblit, "A knowledge-based environment for hierarchical modeling of flexible manufacturing system,"IEEE Trans. Syst. Man, Cybern. A, vol. 26, pp. 81-90, Jan. 1996.

[18] S. A. Melnyk and P. L. Cater, Production Activity Control. Home-wood, IL: Dow Jones-Irwin, 1987.

[19] B. P. Zeigler, Object-Oriented Simulation with Hierarchical, Modular Models. San Diego, CA: Academic, 1990.

[20] J. W. Rozenblit, J. W. Hu, T. G. Kim, and B. Zeigler, "Knowledge-based design and simulation environment (KBDSE): Foundation concepts and implementation,"J. Oper. Res. Soc., vol. 41, no. 6, pp. 82-90, 1990.

[21] B. P. Zeigler, Multifacetted Modeling and Discrete Event Simulation. Orlando, FL: Academic, 1984.

[22] A.Yonezawa and M. Tokoro, Object-Oriented Concurrent Programming. Cambridge, MA: MIT Press, 1987.

[23] S. E. Keene, Programming in Common List Object-Oriented Systems. Reading, MA: Addison-Wesley, 1988.

[24] S. R. Alpert, S. W. Woyak, H. J. Shrobe, and L. F. Arrowood, "Object-Oriented programming," IEEE Expert, pp. 6-7, 1990.

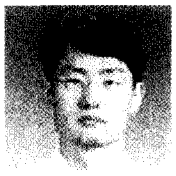
[25] T. G. Kim, "A knowledge-based environment for hierarchical modeling and simulation" Ph.D. dissertation, Univ. of Arizona, Tucson, AZ, 1998.

[26] T. M. Tirpak, S. M. Daniel, H. D. LaLonde, and W. J. Davis, "A note on a fractal architecture for modeling and controlling flexible manufacturing system," IEEE Trans. Syst., Man, Cybern, vol. 22, pp. 564-567, 1992.

[27] D. W. Fogarty, J. H. Blackstone, Jr, and T. R. Hoffmnann, Production & Inventory Management, 2nd. Ed. Cincinnati, OH: South-Western, 1991.

[28] P. Marchall, "The prospects for FMS in UK Industry," in Proc. 1st Int. Conf, on Flexible Manufacturing System, Brighton, U. K., 1982, pp. 71-76.

저자약력



Tae Ho Cho

Tae Ho Cho is associate professor of the School of Information and Communications Engineering, Modeling & Simulation Laboratory, Sungkyunkwan University, Suwon, South Korea.

Hee Suk Seo

Hee Suk Seo is doctorial student of the School of Information and Communications Engineering, Modeling & Simulation Laboratory, Sungkyunkwan University, Suwon, South Korea.

Bernard P. Zeigler

Bernard P. Zeigler is professor of the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85721 USA.