A Development Methodology for Heterarchical Control of Flexible Manufacturing Systems

한 영 근

생산기술연구원 생산시스템개발센터

ABSTRACT

This paper presents a methodology for development of a heterarchical control system in Flexible Manufacturing Systems (FMS) environment. A Petri net based model is applied for development of control software. A real-time scheduling methodology for the heterarchical system is also developed and it is used as a software entity in the overall architecture. The partition of generic and specific modules in control software development is emphasized. The control system performs its control tasks in two stages: (1) generation of generic control code and distribution of them to each controller entity prior to system execution; (2) generation of specific control code according to job arrival and its process plan. The control software and the scheduling algorithm are evaluated by a simulation program.

I. Introduction

Today's manufacturing environment is forced to introduce new and updated products more frequently to enlarge their share of market. In recent years, the advance of computer technologies has had a great effect on the progress of Flexible Manufacturing Systems (FMS). FMS has been adopted in an effort to increase productivity and to shorten system response time to product variation in a discrete-parts production industry [13].

The control system of automated manufacturing systems coordinates and directs the parts handling and processing activities that transform raw materials into finished products [4]. Development of control system software is very complicated and needs a thorough understanding of not only manufacturing systems, but also computer programming and communication. Ayres presented that software development costs account for between 40% and 60% of the total costs for FMS [2].

One of the major reasons that software costs are quite high is because the software development is usually achieved in an *ad hoc* approach. Without formal development methodologies and tools, the software developed for a particular application is almost always implementation-specific, cannot be transferred to different, but similar systems [12] and cannot address speedy modification of the system configuration according to product changes.

In this research, the control means the composition of the control in a narrow sense and the scheduling. *Control* of manufacturing systems, in a narrow sense, implies interfacing with all subordinates, initiating start-up and shutdown, issuing commands to perform assigned activities, using feedback to monitor the execution of those activities, and recovering errors [11]. *Scheduling* is the activity for generating/updating expected start and finish times in the selected plan. Although both the control and the scheduling are considered in this research, these are separated functionally in the overall

control system. The control module can be arranged independently in the early stage without the consideration of the specific parts and the schedule, since it has a generic form. The scheduling module is part-specific, and once it is established for required jobs, then it can be plugged into the control module. The reason that the scheduling task is included in the control system originates from the inherent *part-centric* view of the heterarchical control system, i.e. the part entity issues messages to negotiate among other entities and to reserve one of them. In order to design the heterarchical control system, therefore, a certain level of scheduling tasks, for example deciding a machine on which a part will be processed among alternatives, is required to accomplish the control activities effectively.

The main objective of this research is to develop a FMS control system under the heterarchical architecture. The proposed system in this study is called ***HETerarchical COntrol Systems*** (**HETCOS**). The major goal of HETCOS is to develop more systematic, generic and flexible methodology for control systems than the existing systems and to minimize user's intervention. In order to achieve these goals, the research focuses on the following concepts:

- Applying *Enhanced Heterarchical Architecture* (EHA) and *Heterarchical Petri Nets* (HPN)
- Developing software to generate control code with specific FMS configuration and job orders on the base of the HPN model.
- Developing a real-time scheduler for EHA.
- Evaluating HETCOS by simulation software to check its feasibility.

II. Heterarchical Control Architeccture

As a fundamental building block, control architecture directly influences the flow of control and monitoring information and interaction of the manufacturing process entities. Many researchers have concentrated on the hierarchical architecture for FMS. In the hierarchical architecture, the control problems can be partitioned in order to bound the complexity of any module in the hierarchy to controllable limit. Control decisions are operated top-down, with status reports operating bottom-up.

Technological advances in distributed computing and communication network have enabled the consideration of more decentralized control architecture than the hierarchical architecture. Recently, to overcome the disadvantages of the hierarchical architecture such as low fault-tolerance and difficulties in dynamic modification, several authors have proposed a heterarchical approach [1][3][5][6][10][14]. The heterarchical architecture has distributed locally autonomous entities that communicate with other entities without the master/slave relationship. The local autonomy requires that the global information is minimized.

In the heterarchical system, parts, processing machines, and material handling devices are represented by system entities. The entities are not necessarily submissive and simply observe local rules. Part entities have knowledge about the processing required, processing machine entities have knowledge about the processing that they can perform, and material handling entities have knowledge about the source and destination of moving. The cooperation between entities is organized via a reservation procedure. The heterarchical control is a kind of part-centric control system in which the part entities broadcast processing requirements over communication networks and machine entities similarly broadcast processing availability. When the part and the machine entities negotiate and once a reservation is made, the part is transported to the machine and the processing starts.

In this research, the Enhanced Heterarchical Architecture (EHA) proposed by Han [8] is applied as a development platform. The basic concept of EHA is the same as the conventional heterarchical architecture, but a scheduler entity and a data-base entity are added in order to eliminate shortcomings which comes from no global information.

III. FMS Control Operation

The control action in this research is partitioned into two stages: *Preparation Stage (Stage I) and Execution Stage (Stage II)*. In Stage I, the control system prepares FMS execution only with input of FMS configuration and layout data. In this stage, heterarchical Petri net (HPN) models are constructed, generic control codes are generated, and the generic codes are distributed to each entity ahead of processing, while part-specific data are not considered. HPN was proposed to model the heterarchical properties [8][9]. The input data of Stage I are as follows:

- Type and number of part processing machines
- Type and number of material handling robots and their relationships with part processing machines (cell formation)
- Type and number of material transport devices and transpor-tation distances between stations
- Buffer capacity in each part processing machine

Stage II starts when the first part arrives at the loading station of FMS. The operation data for the part should be entered into the database entity by human users. With the input of the operation data, the part entity and the scheduler entity generates part-specific code for sending messages and reserving other entities and begins to execute FMS operations. The operation data for the input

of Stage II include the following:

- Process plan files (e.g. operation description, processing time, alternative routes, tool data, pointer to NC program files, etc.)
- NC program files
- User specified requirement for the part (e.g. due dates)

Several design principles are constructed and considered in developing the heterarchical control system:

- Master/slave relationship between entities should be removed.
- Global information should be minimized and control actions should be performed only by exchanging local information.
- Part entities should lead the control function between entities, since the heterarchical manufacturing system is a "part-oriented" approach. The part entity decides a machine, sends messages, and reserves the machine.
- If a fault situation happens, it should generate an action to help to escape from the fault and recover to normal conditions.
- By collecting only local information of other entities, the scheduling task should be performed.
- By dividing the control actions into two parts, generic part and specific part, the modification of models according to the dynamic change of FMS environment should be minimized.

The control system consists of two modules: *generic module* and *specific module*. Although it is impossible to develop a comple-tely generic control system, certain section of software execute static and identical functions regardless of the system specifics such as job orders and parts. This section can be made generic. For example, a processing machine entity performs part loading, processing, and part unloading operations regardless of kinds of parts. Specific module is created specifically and dynamically for a system based on the configuration of the system. Once the generic module is constructed ahead of FMS operations, the specific module according to the current system configuration is plugged in the overall control system. This separation can be helpful to modify the control system fast and easily in dynamic manufacturing environment and to enhance system flexibility, because users can change only the specific module.

Generic control tasks of hardware entities, e.g. part processing machine (PPM), material handling robot (MHR), and material transport (MT) entity, belong to the generic module and part-specific control tasks of software entities, e.g. part entity, scheduler entity, and database entity, belong to the specific module. Among the control functions of the specific module, a small portion of them can be represented as generic activities. For example, the operation of reserving a part processing machine entity is common for all kinds of part entities, although determining a specific PPM entity to which it will send a reservation message is a specific task. Therefore, only the generic section of the specific module can be represented as a generic model and it is called *semi-generic* model.

Figure 1 shows task flows of Stage I. With the information of the FMS configuration, generic *heterarchical Petri net (HPN)* modeling is performed on the base of the heterarchical architecture. Also, semi-generic HPN models for the part entity, the scheduler entity, and the database entity are constructed. The HPN models will be used for input of a *Control Code Generator (CCG)*. CCG generates control code from HPN models by some *translation rules*. Output of CCG in the specific module is semi-generic stationary codes which will become specific codes in Stage II. The format of CCG output is a form of pseudo-codes for each entity. The pseudo-codes should be converted again to machine-specific commands by post-processors in order to be input of machine controller units such as PLC and CNC.

In Stage II, each entity executes control tasks by communicating and cooperating with others as shown in Figure 2. Upon the messages of part arrival, the part entity starts control actions in real-time. Since part entities have pointers to required process plans and NC files, they can request and retrieve the data from the database entity while the system is operating. The scheduler entity produces schedules to sequence jobs on each PPM entity by the request from part entities. Software for the database entity and the scheduler entity is under development in the current stage of this research.

IV. Heterarchical Scheduling

The main reason that scheduling is added to the original heterarchical architecture is to overcome shortcomings of

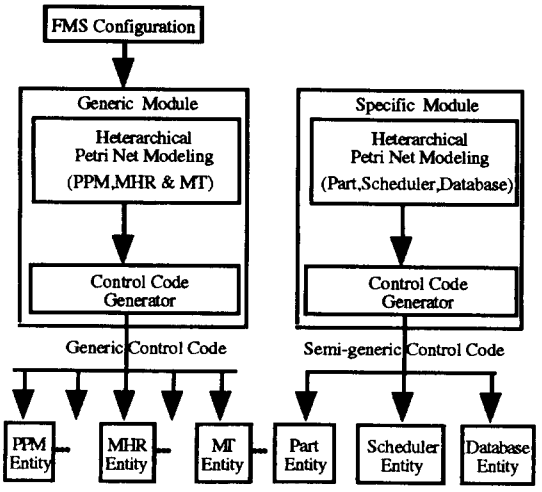

Figure 1. Stage I OF HETCOS

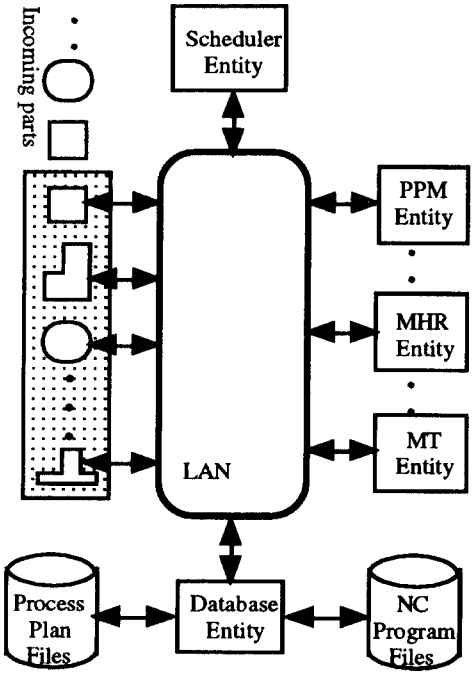

Figure 2. Stage II of HETCOS

local optimization. In this study, although the scheduling is actually combined with the control, the control system still ensures flexibility, because the largest portion of the control is modeled in the generic module and the scheduler is modeled in the specific module. Therefore, introducing new parts with different routes would not require regeneration of the whole models, but modify only the specific module.

Several approaches of scheduling in manufacturing field have been proposed. In the traditional centralized or hierarchical control architecture, the high level master scheduler uses large amounts of global information to generate an optimal or a near-optimal schedule for processing parts. Using their global information including the assumptions about the current status of other entities in the system makes fault-tolerance, one of the most important issues in the development of complex manufacturing systems, difficult. On the contrary, the fully distributed heterarchical control architecture is made up of loosely coupled, highly autonomous entities retaining minimal global information [7]. Therefore, it is believed that real-time scheduling with more fault-tolerance can be realized in the heterarchical control systems.

In this study, scheduling algorithm is developed based on two dispatching rules: EDD (Earliest Due Date) and EEFT (Estimated Earliest Finish Time). *Earliest Due Date* rule gives priority to the job the due date of which is the earliest. *Estimated Earliest Finish Time* rule gives assignment to the machine which could possibly complete the job in the earliest time. This rule is based on look-ahead estimation.

The scheduling task in this thesis is partitioned into two steps. Problem definitions of these two steps are as follows:

- *Performance Measure: Minimizing maximum lateness or maximum tardiness*
- *Scheduling Problem Definitions:*
  - *Step 1: Decide a PPM which will process this job (part point of view)*
  - *Step 2: Decide a part among parts in input buffer (PPM point of view)*

The scheduling algorithm of Step 1 is performed at *the scheduler entity* in the enhanced heterarchical architecture. When a new part arrives at the loading station of FMS or a part is removed from a PPM to the output buffer after finishing process, the part entity request the scheduler entity to decide the machine which will process the next operation of the part. Once the scheduler entity receives the request from the part entity with operation data (e.g. candidate PPMs, processing time, due date), it requests estimation of finish time to all the candidates machines. Before requesting the estimation of finish time, the scheduler entity should provide the estimated arrival time of the part at the input buffers of each candidate PPM. If PPM estimates the expected finish time without this data, it is possible that the real finish time is widely different from the estimated one, since the status of the input buffer, e.g. the number of parts waiting in the input buffer, may be changed while the part is transported to the input buffer.

Material Transport (MT) entities such as cart and AGV are not always available at the time when a part entity is about to reserve a machine. To provide the estimated arrival time of the part at the input buffers of each candidate PPM, the scheduler entity should request MT entities to send time estimation required for arriving at the current part location. With this arrival times of MTs, the scheduler entity calculates the transport times to each candidate PPM according to the distance from the current location to each PPM. Once all data needed for estimating finish time are provided, each PPM estimates the finish time of the part processing assuming that the part will be dispatched according to EDD rule. The estimation can be obtained through the heterarchical Petri net (HPN) model, since the delay times of each place of HPN were already modeled. The PPM entities will send the estimation results to the scheduler entity, and then the scheduler entity will decide the machine which can finish the job in the earliest time. To get a reliable estimation, the algorithm includes robot handling times, but the communication times are neglected, since they are little relative to the other time data. The algorithm of EEFT is as follows :

1) Part (P_i) entity requests scheduler (SC) entity to decide the next machine.
2) SC requests Material Transport (MT) entities to send estimation of arrival times at current P_i location.
3) MTs calculate and send the arrival times, $T_{mt,k}$ according to their current location and the part location, where k=1,....,nt and nt is total number of MTs.
4) SC selects a MT which has minimum value (T_{mt}) of $T_{mt,k}$. $T_{mt} = min\ T_{mt,k}$
5) SC calculates each $T_{tr,j}$, where $T_{tr,j}$ is a time required for transporting from current part location to each candidate machine j according to distance data, where j=1,...,np and np is total number of candidate machines.
6) SC calculates arrival time ($T_{arr,j}$) at input buffer of machine j,

$$T_{arr,j} = T_{mt} + T_{tr,j} + 2T_h$$

where T_h is a robot handling time (part current location --> MT and MT --> input buffer) and is assumed to be constant.
7) j = 1
8) Do while j <= np,
9) SC requests all candidate PPMs to send estimation of finish time of this part on the prediction that P_i is arriving at the PPM j when time is $T_{arr,j}$.
10) If the input buffer of PPM j is expected to be full at $T_{arr,j}$, PPM j sends "not_available" message and go to 8.
Else, PPM j Calculates

$$T_j = \sum_{n=1}^{q_{ji}} T_{jn} + (2q_{ji} + 1)T_h + T_{P,j}$$

where, q_{ji} : Total number of parts in PPM j's input buffer which will be processed ahead of P_i

T_j : Estimated finish time for processing P_i on PPM j

T_{jn} : Processing time of n-th part which will be processed ahead of P_i, where n=1,....,q_{ji}.

$T_{p,j}$: Processing time of the operation of P_i.

11) PPM j sends T_j to SC.
12) SC selects a PPM which has the minimum value of T_j.
13) SC sends this result to P_i.

The schematic diagram of Step 1 scheduling (EEFT) is shown in Figure 3.

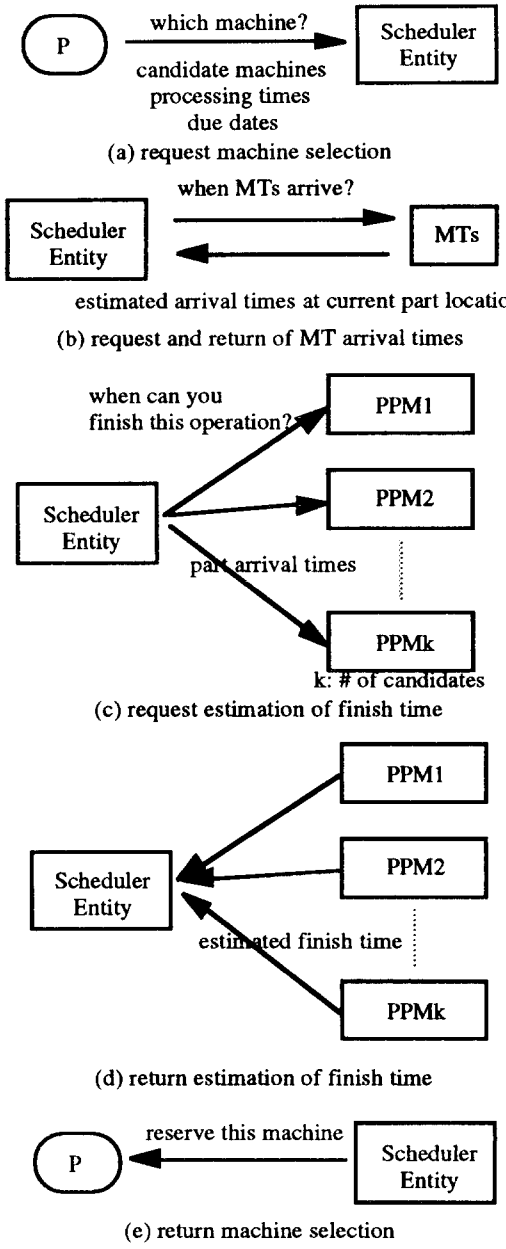

Figure 3. Schematic Diagram of Step 1(EEFT) Scheduling Task Flow

V. Control Code Generator

HPN model is a formal model of the behavior of the heterarchical controller. We are now interested in developing software implementation which exhibits the behavior described by the formal model and in how HPN models are converted as running systems. In this research, rules for translating the HPN model to the control code are developed. The *translation rules* are obtained by examining the patterns of the HPN models. Han [8] presented the translation rules in detail.

Control code generator (CCG) generates appropriate control pseudo-code from the HPN models by the translation rules. Input files of CCG are FMS configuration files and process plan files. In the first stage, FMS configuration is used for generating generic code of PPM, MHR, and MT entities. Generated generic codes are distributed to each hardware entity prior to FMS execution. Also, "semi-generic" codes for the part entity, the scheduler entity, and the database entity are prepared. This "semi-generic" code has no part-specific code, but has only generic commands derived from HPN models. The generic control codes of system entities are presented in Han's thesis [8].

The second stage starts when a new part arrives at the loading station of FMS. Its process plan is plugged into the semi-generic code of the part, the scheduler entity, and the database entity, and then the reservation and the scheduling tasks begin.

Once the generic control codes are generated, there is no need to revise the overall control software when the control environment changes. In order to address the changes, it is only required to add or remove control code of the added or removed entities. Also, the specific code of part entity and scheduler can be plugged in the generic code. This is a basic concept of the heterarchical control system and ensures the flexibility and the low cost to develop control software.

VI. System Evaluation

In order to evaluate the control system presented in this paper, a FMS based on CIM Lab of the Pennsylvania State University was designed. The FMS consists of three prismatic machining cells and two rotational machining cells as shown in Figure 4. Cell formation seems to mean an existence of the master/slave relationship between the cell and each entity. However, since there is no cell level controller in the heterarchical system, we can say that no master/slave relationship exists. Each cell is served by a material handling robot. Part entities are transported by six unidirectional carts. The FMS configuration data is formatted as an input file.

In order to evaluate the control system, a simulation program mainly for the heterarchical architecture was developed. Overall actions of each entity during control operation can be simulated by the simulation program. Simulation for the heterarchical system should be performed on the distributed and multi-tasking computer in order to evaluate the distributed and parallel nature of the system, but at the current stage of this study, a Power Macintosh 7100 without multi-tasking operating system is being used for emulating the parallel processing. The simulation program, first of all, starts with emulation of control codes generated in CCG.

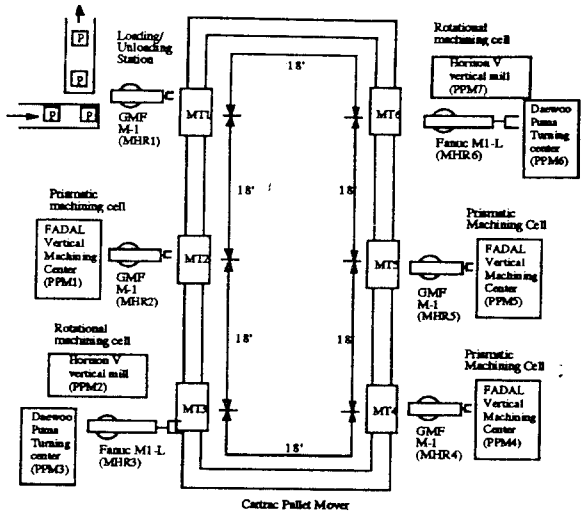

Figure 4. Model FMS

The FMS configuration data is formatted as an input file. With the input file, Stage I of the control system generates the generic code of PPM, MHR, and MT entities, and semi-generic code of Part, Scheduler, and Database entities. As the first part arrives at the loading station, Stage II begins to plug the process plan into the semi-generic code of the part entity. With the help of the real-time scheduler and database handler, parts are dispatched to each cell and finally to each PPM. With the input of FMS configuration, process plan, and generic code of each entity, the simulation program displays entity states in parallel, according to the progress of simulation clock.

At the end of one simulation run, the program provides maximum lateness and machine utilization. Several experiments were completed with some parts and their process plan. Random occurrence of communication error, minor machine breakdown, and major machine breakdown were inserted during simulation. By the simulation, we verified that the control system developed in this research performs the expected control tasks such as reservation, dispatching, sending messages, and so on. Also, it was checked that the system can solve the fault situation according to the HPN model.

VII. Conclusions

A formal development methodology for heterarchical FMS control systems has been presented. The methodology applies an enhanced heterarchical architecture, a formal heterarchical Petri net modeling scheme, a control code generator based on HPN models, and a set of generic control entity models to assist the software

implementation of control systems. The partition of generic and specific modules in the software development supports a convenient restructuring method which is adaptable to dynamic changes in the manufacturing environment. The methodology has been evaluated through a full-scale example implementation of control systems and simulation.

References

[1] Anstiss, P., "The Implementation and Control of Advanced Manufacturing Systems," *Control and Programming in Advanced Manufacturing*, IFS Publications Ltd., UK, 1988

[2] Ayres, R. U., "Technology Forecst for CIM," *Manufacturing Review*, Vol. 2, No. 1(1989), pp. 43-52.

[3] Bakker, H., "DFMS: A New Control Structure for FMS," *Computers in Industry*, Vol. 10(1988), pp.1-9.

[4] Dilts, D. M., N.P. Boyd, and H.H. Whorms, "The Evolution of Control Architecture for Automated Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 10, No. 1(1991) pp. 947-965.

[5] Duffie, N.A. and R.S. Piper, "Non-Hierarchical Control of A Flexible Manufacturing Cell," *Robotics and Computer-Integrated Manufacturing*, Vol. 3, No. 2(1987), pp.175-179.

[6] Duffie, N.A., R. Chitturi, and J. Mou, "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities," *Journal of Manufacturing Systems*, Vol. 7, No. 4 (1988), pp.315-328.

[7] Duffie, N. A., "Synthesis of Heterarchical Manufacturing Systems," *Computers in Industry*, Vol. 14 (1990), pp. 167-174.

[8] Han, Y.G., *Heterarchical Control For Flexible Manufacturing Systems Using Petri Nets*, Ph.D. Thesis, The Pennsylvania State University, 1995.

[9] Han, Y.G. and I. Ham, "Heterarchical Petri Net Modeling for FMS Control," *Proceedings of KSME Spring Conference*(1995).

[10] Hatvany, J., "Intelligence and Cooperation in Heterarchic Manufacturing Systems," *Robotics and Computer-Integrated Manufacturing*, Vol. 2, No. 2 (1985), pp.101-104.

[11] Joshi, S. B., R. Wysk and A. Jones, "A Scaleable Architecture for CIM Shop Floor Control," *Proceedings of Cimcon '90, National Institute of Standards and Technology*(1990), pp. 21-33.

[12] Smith, J.S., *A Formal Design and Development Methodlogy for Shop Floor Control in Computer Integrated Manufacturing*, Ph.D. Thesis, The Pennsylvania State University, 1992.

[13] Tombak, M. and A. De Meyer, "Flexibility and FMS: An Empirical Analysis," *IEEE Transactions on Engineering Management*, Vol. 28, No. 10(1988), pp. 1197-1210.

[14] Veeramani, D., B. Bhargava, and M.M. Barash, "Information System Architecture for Heterar-chical Control of Large FMSs," *Computer Integrated Manufacturing Systems*, Vol. 6, No. 2(1993), pp.76-92.