Presented at the 5th Euromicro Workshop on Real-Time Systems, 1993

A Petri Net Based Modelling and Synthesis Technique for Real-Time Systems

Zebo Peng and Anders Törne

Dept. of Computer and Information Science
Linköping University
S-581 83 Linköping, Sweden

Abstract

This paper presents a general approach to model, analyze and synthesize real-time systems. It describes a basic model for specifying real-time systems based on an extended timed Petri net notation, which can be used to study and verify several important characteristics of the modelled system. The model consists of a timed Petri net representing the controlling sub-system and an activity diagram representing the controlled sub-system. Designers can use both top-down and bottom-up synthesis techniques to carry out the design tasks based on the model. This paper presents mainly the application of the proposed method in modelling and synthesizing flexible manufacturing systems. Our ultimate goal is, however, to develop a general design environment for systematic modelling and design of large real-time systems.

1. Introduction

Real time systems are characterized by the fact that *severe* consequences might result if logical as well as timing correctness properties of the system are not satisfied [10]. Since the complexity of such systems is very high and the timing and logical characteristics of such systems intertwine, it is very difficult to analyze and design them. The design of real-time systems therefore requires the support of formal techniques and automatic/semi-automatic design tools. The objective of the research presented in this paper is to develop a *systematic* method and design aid tools to help designers to model and analyze a real-time system and to synthesize an implementation which satisfies given design constraints.

We are particularly interested in developing a systematic design environment that will allow designers to efficiently explore different design alternatives to a real-time problem and to generate an efficient implementation using iterative refinement or modular composition methods. Such an approach is very useful in designing flexible manufacturing systems (FMS), which can be programmed to accomplish different tasks. It is assumed that designers of a FMS system are given a set of basic components which must be configured and controlled to perform a specified task. The designers must synthesize an implementation configuration, and design the control program to satisfy given timing requirements or to optimize the performance of the system.

A critical step towards such a design environment is to develop a modelling technique to capture the different possible configurations and control strategies of the modelled system so that the different design decisions can be evaluated and different design alternatives explored and studied. Such a modelling technique should be able to capture a design at different level of abstractions. It must be able to model a partial design, when a completed implementation has not yet been developed, and a detailed final implementation.

Our approach is based on a Petri net based design modelling tool, which is used to model the basic operations, concurrency, synchronization, mutual exclusion, and sequential relations of the basic operations of real-time systems, especially flexible manufacturing systems. Our approach provides a basis to develop different design aids tools to explore different implementations. Once an implementation is selected, a mapping algorithm can be used to translate the controlling software into a real-time program. Therefore, no “reprogramming” is needed. Our approach can, therefore, be used to separate the end-users from the programming task. The current research follows the tradition of the ARAMIS project [11] [4] in emphasizing support for non-programmers in the process of developing and analyzing embedded real-time software as well as its controlled environment.

Petri nets [9] have widely been used for modelling real-time systems, especially real-time software [2]. Recently,

This work has been supported by CENIIT (the Center for Industrial Information Technology) at Linköping University.

they have also been used in modelling modern manufacturing systems such as flexible manufacturing systems [1] [3] [5] [12]. For example, stochastic Petri nets have been applied in performance evaluation of manufacturing systems [1]. Our approach, on the other hand, emphasizes the issue of using Petri net models for real-time system synthesis and provides a foundation for the development of a *systematic* approach to deal with the modelling, analysis, and synthesis aspects at the same time.

In most of the related work on using Petri nets for modelling flexible manufacturing systems, the behavior of the controlled sub-system (structure of the manufacturing cells) is annotated as part of the Petri net model, and the evaluation of alternative configuration of the controlled sub-system requires modification of the Petri nets. In the present paper, the controlled sub-system is separated from the Petri net model which is used only to specify the control function. This is a significant departure from previous application of Petri nets for modelling flexible manufacturing systems.

This paper describes some preliminary results of our current research activities aiming at the development of a systematic methodology for real-time systems. In particular, we will present the overview of the design methodology in section 2. The basic modelling technique based on Petri net will then be discussed in section 3. Section 4 will describe the application of the Petri net model in the analysis and synthesis process of real-time systems. In the paper, flexible manufacturing systems are used as an example to illustrate the proposed methodology, even though the methodology is applicable to other types of real-time systems as well.

2. A real-time system design methodology

We are developing a systematic design methodology which views a real-time system as consisting of a *controlling* sub-system and a *controlled* sub-system. This view can be applied at different levels of abstraction. A real-time system is consequently considered as a hierarchy of controlling/controlled sub-systems. In each of the abstraction levels, the controlled sub-system can be viewed as the environment with which the corresponding controlling sub-system interacts. The controlling sub-system consists of the software element together with its associated processing hardware. It interacts with the environment by sending control signals to, and receiving feedback information from, the environment.

In the context of a flexible manufacturing system, which is defined as a computer-controlled configuration of semi-independent manufacturing cells coupled with a material handling mechanism, the controlling sub-system is the computer hardware together with the real-time software used to send controlling signals to the manufacturing cells. A particular configuration of the manufacturing cells and the material handling mechanism forms a controlled sub-system. Flexibility of such systems is achieved by introducing sensors to provide information from the manufacturing process to the controlling sub-system and equipping the controlling function with adaptability, i.e., the system can be adapted to different applications as well as operational environments. The system must be able to operate efficiently with different level of resource availability, such as how many robots are available in the manufacturing cells that can be used to perform the tasks.

When designing a FMS system, it is desirable to develop at the very beginning a high-level behavioral specification of the system without detailed binding of the system operations to the particular physical configuration. Later when the desired properties of the system is ensured, design aid tools can be used to synthesize an implementation under given design constraints by exploring different scheduling, resource allocation, and implementation structures.

We have developed a Petri net based model, called modified timed Petri net (MTPN), which is to be used in the process of synthesizing real-time system implementations. It captures explicitly concurrent activities of the modelled system. It is formal and executable, which allows the designer to use verification and simulation techniques to perform qualitative and quantitative analysis of the systems. The main feature of the Petri net based model is that it can be used to describe the causal relations between basic operations of a system. Whenever additional sequencing constraints must be introduced, it can also be captured by the Petri net notation and its consequence can be studied using Petri net analysis techniques. Another feature of our approach is that once a flexible manufacturing system and its controlling software are captured in the Petri net based model, their implementation can be derived in a mechanized manner using net transformation methods.

The MTPN model is to be used as a modelling and synthesis tool in a real-time system design environment such as ARAMIS [11]. The ARAMIS system has been designed with a layered architecture with three basic levels: the task programming level, the control level and the physical level. At the task level the operator specifies what operations should be performed in the physical environment and under what conditions, using a graphical hybrid, rule based language [4]. The task program executes by setting reference values for the objects in the world model. The control level is responsible for keeping the real world in a state represented in the world model. The physical level is the actual connection to the real world. Each level of the ARA-

MIS architecture can be modelled as one level in the MTPN notation - a level consisting of a model of a controlling sub-system and an activity diagram modelling the controlled sub-system. If the underlying level in ARAMIS provides an implementation of the behaviour modelled by the MTPN controlled sub-system, then the behaviour of the total system at this level can be verified. The verification is primarily done to show the desired behaviour of the aggregate controlled sub-system together with the controlling sub-system at each level, and compare this with the specification at a higher level of abstraction.

3. The modified timed Petri net model

This section describes the modified timed Petri net notation which is used to model real-time systems. The MTPN notation is based on an extended timed Petri net model originally developed for representing digital hardware systems [6] [7]. The extended timed Petri net model consists of two separate but related parts, a control structure and a data path. The data path is represented as a directed graph with nodes and arcs. The nodes are used to capture data manipulation units (data storages, arithmetic operators, etc.). The arcs represent the connections of the nodes. The control part of the design representation is captured as a timed Petri net with restricted transition firing rules [6]. These two parts are related by the control signals coming from the control part to the data path, and the conditional signals traveling in the opposite direction.

In the MTPN notation, a Petri net is used to represent the control structure of a real-time system, the controlling sub-system. The controlled sub-system is modeled as an activity diagram. It describes the basic activities and the changes of states of the resources as the results of the activities in the controlled sub-system. An activity is represented by an arc in the activity diagram and controlled by a set of places in the control Petri net. When one of the control places holds a token, the activity will be activated. As the result of the activity, the state of some resources of the modelled system will change. The formal definition of the activity diagram is as follows.

Definition 1: An *activity diagram*, **D**, is a three-tuple, **D** = (**V**, **A**, **I**), where

$\mathbf{V} = \{V_1, V_2, ..., V_n\}$ is a finite set of *vertices* each of which represents a state, $S_i \in \mathbf{S}$, of a resource, $R_i \in \mathbf{R}$. $\mathbf{R} = \{R_1, R_2, ..., R_m\}$ is a finite set of resources. $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ is a finite set of states of the resources.

$\mathbf{A} \subseteq \mathbf{V} \times 2^{\mathbf{V}}$ is a finite set of *arcs* each of which represents an activity of the modelled system. An arc has a single source vertex and a subset of the vertex set as its destination including the empty set. Associated with each arc, there is a timing model representing the time needed for the activity, which can be deterministic or stochastic.

$\mathbf{I} \subseteq \mathbf{V}$ is the set of initial vertices which are *marked* initially.

Intuitively, an activity diagram specifies the basic components and the possible activities as well as the initial state of a controlled sub-system. The components are modelled as a set of resources. Each resource has a finite set of possible states. A resource/state pair defines a vertex of the activity diagram. When the vertex is marked, the resource is currently in the corresponding state. An activity, or operation, of the controlled sub-system will result in the transition of resources from the current state to a next state. When the destination vertex set of the arc, which represents the activity, has only one element or is empty, the activity will result in a deterministic state transition. Otherwise, one of the destination vertices of the arc will be marked as the result of the activity. The choice of which vertex to be marked is non-deterministic. This is used to model non-deterministic behavior of a real-time environment. With this mechanism, sensor-generated information can also be used to influence the controlling sequence.

It is assumed that the state transition will be controlled by control signals coming from the controlling sub-system which, together with the controlled sub-system, forms a real-time system defined as follows.

Definition 2: A *MTPN system,* Γ, is a seven-tuple, $\Gamma = (\mathbf{D}, \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{C}, \mathbf{G}, \mathbf{L}, M_o)$ where

$\mathbf{D} = (\mathbf{V}, \mathbf{A}, \mathbf{I})$ is an activity diagram;

$\mathbf{P} = \{P_1, P_2, ..., P_n\}$ is a finite set of *control places*;

$\mathbf{T} = \{T_1, T_2, ..., T_m\}$ is a finite set of *transitions*;

$\mathbf{F} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$ is a binary relation, called the *control flow relation*.

$\mathbf{C} : \mathbf{P} \rightarrow 2^{\mathbf{A}}$ is a mapping from control places to sets of arcs of the given activity diagram; an arc A_i is *controlled* by a control place P_j if $A_i \in \mathbf{C}(P_j)$.

$\mathbf{G} : \mathbf{V} \rightarrow 2^{\mathbf{T}}$ is a mapping from activity diagram vertices to sets of transitions; a transition T_i is *guarded* by a vertex V_j if $T_i \in \mathbf{G}(V_j)$.

$\mathbf{L} = \{L_1, L_2, ..., L_n\}$ is a finite set of time units each of which is assigned to a control place.

$M_o : \mathbf{P} \rightarrow \{0, 1\}$ is an *initial marking* function.

The definition of the MTPN model is based on a timed Petri net notation, which is used to model the controlling sub-system. The Petri net places are mainly used to capture the *control signal* concept. When a control place holds a token, a control signal will be sent to activate the corresponding activities in the controlled sub-system. This control relation is specified by the control mapping function **C**. If a place does not control any arc, it can be used to represent certain condition of the modelled system, such as the number of material units available at certain storage place. A transition, on the other hand, is used to represent start or completion of activities or the change of system conditions. The relation between the places and transitions is captured by the control flow relation **F**. To represent the situation where the control flow is influenced by some activities of the controlled sub-system, the guard concept is introduced; a transition may be guarded by a set of vertices, which is defined by **G**. The transition can be fired only if all its guarding vertices are marked. The behavior of a MTPN system is formally defined by the following execution rules.

Definition 3: Given $\Gamma = (\mathbf{D}, \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{C}, \mathbf{G}, \mathbf{L}, M_o)$ with $\mathbf{D} = (\mathbf{V}, \mathbf{A}, \mathbf{I})$, its *behavior* is defined as follows:

1. A function $M : \mathbf{P} \rightarrow \mathbf{N}$ is called a *marking* of Γ ($\mathbf{N} = \{0, 1, 2, ...\}$). A marking is an assignment of *tokens* to the places. Initially there is a token in each of the initial places, i.e., place P_i such that $M_o(P_i) = 1$.

2. A subset **W** of **V** is called the *currently marked set* of vertices. A vertex is *marked* if it is in the currently marked set. Initially, **W** = **I**.

3. A transition T is *enabled* at a marking M if and only if for every P_i such that $(P_i, T) \in \mathbf{F}$, we have $M(P_i) \geq 1$ and the time for the token to stay in P_i is longer than L_i.

4. A transition will *immediately* be *fired* when it is enabled and all of its guard vertices are marked.

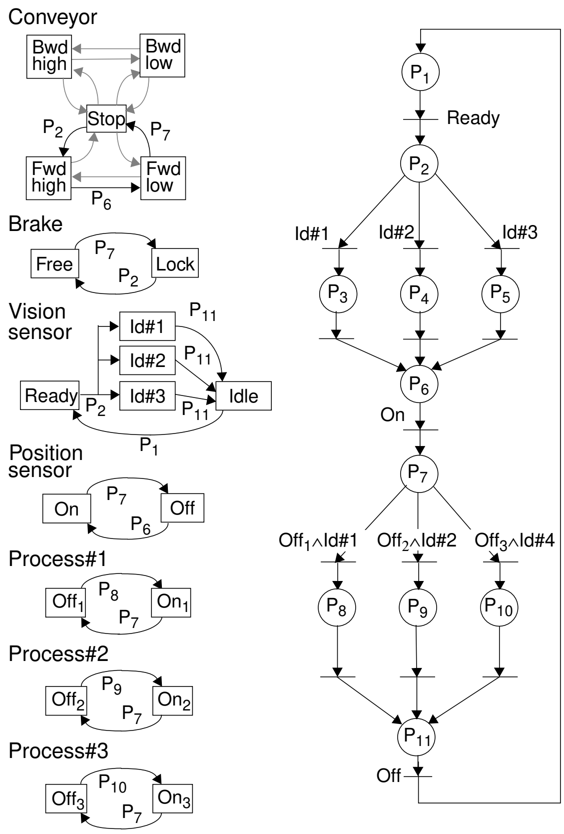
5. Firing an enabled transition T removes a token from each of its input places (i.e., P_i such that $(P_i, T) \in \mathbf{F}$) and deposits a token in each of its output places (i.e., P_j such that $(T, P_j) \in \mathbf{F}$), resulting in a new marking which is said to be *immediately reachable* from the old one. All markings which can be reached from M_o by firing a sequence of transitions constitute the set of *reachable* markings.

6. The deposit of a token into a place will trigger the activation of the activities associated with the arcs controlled by the place.

7. When an activity is activated and the source vertex of the arc representing the activity is in the currently marked set, the vertex will be removed from the set. When the activity is completed, one of the destination vertices will be marked, which results in a new currently marked set of vertices. If the source vertex is not in the currently marked set (which can be interpreted as the precondition of the activity is not satisfied), the activated activity will not occur.

In the MTPN model, timing information is attached to the Petri net places and the arcs of the activity diagram. The time attached to an arc models the time needed to complete an activity in the controlled sub-system. Non-deterministic characteristics of operation time will be handled by stochastic models with a upper bound and a lower bound. The time attached to a place is, however, a constant, which models the time needed to hold a control signal or the delay time needed to keep the controlled sub-system at certain state. In the case when a place controls a set of activities which must be completed during the time when the control signal is on, the time attached to the place must be larger than the worst case operation time of the set of activities.

Before application of the MTPN model is described, let us look at a simple MTPN example as illustrated in Fig. 1. We use a *graphical* representation of the controlling sub-system which follows the convention of Petri net graphs [9]. For the activity diagram, vertices are drawn as rectangles and arcs annotated with the places controlling them. The example describes a material transport/processing system. The system consists of a conveyor belt with a safety brake, a vision system, and three different manufacturing processes. There is also a position sensor which detects when material has reached the pickup position for each process. Initially, there is a token in place P_1 and the initial state of the controlled sub-system consists of the following vertices: *Conveyor/Stop*, *Brake/Lock*, *Vision sensor/Idle*, *Position sensor/Off, Process#1/Off*$_1$, *Process#2/Off*$_2$, and



a) Controlled sub-system b) Controlling sub-system

Fig. 1 A simple example

Process#3/Off$_3$. Note in this example, the conveyor has some extra functions which are not utilized. We assume that the design environment will provide a set of basic component models which can be used by the designers to construct their system models.

In the above example, the incoming material is classified by the vision system into three categories, each corresponding to a manufacturing process. The controlling sub-system first signals to the vision system to start identifying the incoming material (which actually consists of a "wait first for new material, then classify" sequence). When ready a control signal is sent to free the brake, start the conveyor belt at high speed in the forward direction, and ask the vision system for classification information. This information is used to select a suitable delay (associated with P_3, P_4, or P_5) to keep the conveyor running forwards, transporting the material to the appropriate position. When the delay time expires, low speed for the conveyor is ordered and the controlling sub-system waits for the position sensor to signal. The conveyor is then stopped and the brake is locked. The material should now be at the correct position for the intended manufacturing process, which is signalled to start. When the position sensor detects that the material has been removed from the conveyor, the cycle starts again.

Note that an activity arc can be used to model a detailed operation such as the engagement of a brake as in the above example, or the operation of a complicated sub-system. In the latter case, the sub-system can be, in turn, modelled by a MTPN model and the same analysis technique can be used to check desired characteristics of the modelled subsystem at the particular level of granularity. The hierarchical decomposition technique is, however, not discussed in this paper.

4. Application of the MTPN model

Once a real-time system is modelled by the MTPN notation, analysis techniques can be used to perform evaluation and verification of the modelled system. Both Petri net analysis and simulation techniques can be used for this purpose. In this section, we will first describe how safeness/boundedness analysis, timing constraint checking, and performance evaluation can be carried out. The application of the MTPN model for synthesis will then be discussed.

A MTPN model is unsafe if there exists an execution sequence that will lead to the activation of an activity before a previous activation of the activity has been completed. It is important to make sure that a FSM system is safe, since a unsafe situation represents, e.g., a violation of mutual exclusion. Safeness checking can be used to warn the designers of a faulty design. It can be carried out with the help of the reachability tree generation algorithm for Petri nets [9]. However, since we have introduced time into the Petri nets and used an immediate firing rule as defined in definition 3, the algorithm must be modified.

The first modification is to attach to each reachable marking a time interval representing the duration when the marking is to be valid in the modelled system. The second modification is that when generating the reachable markings, a set of transitions is simultaneously fired (if possible) instead of one single transition. Further, only markings which are reached by firing maximally firable sets of transitions are included in the reachable tree. A firable transition set is a set of transitions which are enabled by the current marking and all of their guarding vertices could simultaneously be marked. A maximally firable transition set is defined as a firable transition set that is not a proper subset of any of the firable transition sets of the current marking under the same assumption regarding the guarding vertices.

When the reachability tree is generated according to the modified algorithm, an inspection algorithm is used to check whether there is any possibility to have an unsafe situation. This includes checking the safeness of the Petri net places that control some activities, and calculating the time distances between different places controlling the same activity. If the minimal time distance is smaller than the maximal time of the activity, a unsafe situation may occur.

Boundedness properties of the modelled system can also be checked using a similar approach. A Petri net place is *k-bounded* if there are less than *k* tokens in it in any of the reachable markings. This property can easily be checked by inspecting the reachable tree generated with the modified algorithm, and be used to detect possible buffer size violation problem of the modelled system.

Another property which can be checked is timing constraint. A timing constraint can be expressed as the time needed for a token to travel from an initial place to a final place. Such a constraint is usually introduced due to the timing requirement imposed by the external environment or the next abstraction level in the design hierarchy. In the latter case, the initial place represents the activation of an activity in a higher level design and the final place the completion of the activity. The analysis will be based on the concept of a critical path between the two places, i.e., the longest time needed to finish the set of operations initialized from the given starting place and terminating at the given finishing place. It is analogues to the critical path analysis of hardware systems and similar algorithms can be used for this purpose [8].

Simulation technique can be used to carry out performance evaluation of the MTPN model. If the designers are provided with the set of simulation models of the basic components of FMS systems, the simulation task is reduced to the problem of executing the MTPN model following the execution rules given in definition 3. This can be done automatically. To estimate the performance of a FMS system, average execution time statistics will be collected, which gives more accurate estimation than using Petri net analysis techniques to obtain the worst case.

The second application of the MTPN model is to support synthesis of real-time implementations. Two basic strategies can be used: the top-down approach with iterative refinements and the bottom-up approach using modular composition methods. When using top-down synthesis technique, the designers start with the top level MTPN design and then refine each of the activities into the next level MTPN model if necessary. With the bottom-up approach, the designers start with the basic level of design and continue with the higher level by considering the lower level design as an activity to be used to build the higher level MTPN model.

When designing at a given level, only the minimal partial ordering relation among the basic activities of the system is specified with the MTPN notation at the very beginning. This means that the scheduling of the operations are not totally specified, and can be decided based upon different implementation configuration. One feature of MTPN is that it can be used to capture partial scheduling information. Therefore, the scheduling task can be carried out incrementally together with other design refinement, such as resource binding/allocation. This means that an iterative improvement approach is used to carry out the scheduling and resource allocation tasks which are traditionally done using two separate algorithms in two distinct steps. The designers can interactively guide the scheduling process and explore different alternatives until a satisfactory result has been reached.

5. Conclusions

We have described a systematic methodology for the modelling, analysis and synthesizing of real-time systems, especially flexible manufacturing systems. The main feature of the methodology is the use of a modified timed Petri net modelling technique which captures a real-time system during the synthesis process and allows the designers to explore different design alternatives. The model consists of a timed Petri net representing the controlling sub-system and an activity diagram representing the controlled subsystem. Both Petri net analysis algorithms and simulation techniques can be used to perform design verification and evaluation of the MTPN models and the results can be used to guide the top-down or bottom-up synthesis approaches to real-time system design.

The methodology is developed within the ARAMIS framework. The Petri net model of the controlling sub-system will eventually be implemented as real-time software used to control sensor-based flexible manufacturing systems.

References

[1] R. Y. Al-Jaar and A. A. Desrochers, “Petri-Nets in Automation and Manufacturing Systems”, In G. N. Saridis (Ed.) *Advances in Automation and Robotics 2*, JAI Press Inc., 1989.

[2] B. Berthomieu, and M. Diaz, “Modeling and Verification of Time Dependent Systems Using Time Petri Nets”, *IEEE Trans. on Software Engineering*, Vol. 17, No. 3, pp.259-273, 1991.

[3] M. Kamath, and N. Viswanadham, “Application of Petri Net based Models in the Modelling and Analysis of Flexible Manufacturing Systems”, *Proc. 1986 IEEE Inter. Conf. on Robotics and Automation*, pp. 312-317, 1986.

[4] P. Loborg, and A. Törne, “A Hybrid Language for the Control of Multimachine Environment”, *Proc. 4th Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems,* Hawaii, 1991.

[5] T. Murata, et. al., “A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation”, *IEEE Trans. Industrial Electronics*, Vol. 33, No. 1, pp.1-8, 1986.

[6] Z. Peng, *A Formal Methodology for Automated Synthesis of VLSI Systems*, Ph.D. Dissertation, No. 170, Dept. of Computer and Information Science, Linköping University, 1987.

[7] Z. Peng, “Semantics of a parallel computation model and its applications in digital hardware design,” in *Proc. 1988 International Conference on Parallel Processing*, pp.69-73, 1988.

[8] Z. Peng, K. Kuchcinski, and B. Lyles, “CAMAD: A Unified Data Path/Control Synthesis Environment”, in D.A. Edwards (Editor), *Design Methodologies for VLSI and Computer Architecture*, North-Holland, pp.53-67, 1988.

[9] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-hall, Englewood Cliffs, New Jersey, 1981.

[10] A. Stankovic, and K. Ramamritham, *Hard Real-Time Systems*, Tutorial, IEEE, 1988.

[11] A. Törne, “The Instruction and Control of Multi-Machine Environments”, in G. Rzevski (Editor) *Application of Artificial Intelligence in Engineering*, Springer-Verlag, pp.137-152, 1990.

[12] M. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, Norwell, MA, 1993.