PAPER Special Section on Concurrent Systems and Hybrid Systems

Control of Batch Processes Based on Hierarchical Petri Nets

Tomoyuki YAJIMA[†a)], **Takashi ITO**[†], *Nonmembers*, **Susumu HASHIZUME**[†b)], *Member*, **Hidekazu KURIMOTO**[††c)], *and* **Katsuaki ONOGI**[†d)], *Nonmembers*

SUMMARY A batch process is a typical concurrent system in which multiple interacting tasks are carried out in parallel on several batches at the same time. A major difficulty in designing a batch control system is the lack of modeling techniques. This paper aims at developing a method of constructing batch control system models in a hierarchical manner and operating batch processes using the constructed models. For this purpose, it first defines process and plant specifications described by partial languages, next presents a procedure for constructing hierarchical Petri net based models, and states the verification of models based on reachability analysis. It also discusses the detection of faults and conflicts in batch processes based on place-invariant analysis.

key words: *batch control, discrete event system, concurrent system, hierarchical Petri net*

1. Introduction

Procedural control directs equipment-oriented actions to take place in an ordered sequence to carry out a process-oriented task. It is realized according to operating procedures to accomplish the task of a complete process. To carry out various tasks, it is necessary to design a procedural control system for efficiently sharing an available piece of equipment, utilities and production resources. Procedural control system design, however, takes considerable time and effort and is error prone as the complexity of a controlled system increases. A major difficulty in designing a procedural control system is the lack of modeling techniques.

A process whose output product appears in discrete batches or quantities is called a batch process. A batch process is a discontinuous process. Since the main objective of batch control is to sequence the process through a series of distinct states, procedural control plays an important role in batch processes. A batch plant consists of multiple interacting units to carry out tasks in parallel on several batches at the same time. One task may be carried out simultaneously with other tasks. In this situation, the two tasks are causally independent (i.e. a task may be carried out before, after or in parallel with the other). This concurrent nature creates some difficult modeling problems. Yamalidou et al. showed that several new modeling techniques are useful to represent the concurrent behavior of batch processes [1]. Wonham et al. gave some general foundations called the Supervisory Control Theory for the study of control issues for discrete event systems [2]. Sanchez applied the theory to batch control [3]. However, he did not deal with concurrent nature in batch processes, because his control system model was based on formal languages and automata. Although Tittus also applied the Supervisory Control Theory to the hierarchical batch control synthesis problem on the basis of Petri net models, the problem how to generate a control specification was not presented [4]. Viswanathan et al. developed a framework for synthesizing operating procedures for batch processes [5], [6]. Although they used a graphical modeling language called Grafchart, it cannot explicitly represent the states of batch processes. To develop a systematized approach for design of discrete event systems, we formulated a discrete event system design problem as that to construct a Condition/Event net (C/E net) whose net partial language equals to a given partial language as a specification [7]–[9]. A C/E net is a Petri net in which each place can contain at most one token. We also applied the C/E net construction problem to a batch control system design problem [10]. However, the problem how to generate a control specification still remained. To solve this problem, we discussed the information available to operating procedures synthesis and proposed a method of generating control specifications and operating procedures [11].

There has been a significant amount of work around standards for batch control. The ISA's SP88 Committee leads this work in the international arena [12], [13]. It is not the intention of the committee to generate code but to establish guidelines or a framework for batch control. The design of a large system proceeds to successively lower levels of the system. A hierarchical design approach involves first specifying a rough outline of a system and then successively refining the outline. This begins with a definition of desired system behavior on the basis of the operating objectives.

This paper aims at developing a method of constructing a batch control system model in a hierarchical manner and operating batch processes using the constructed model. In Sect. 2 we present a hierarchical modeling framework to represent the information about process and plant. In

Manuscript received April 5, 2004.
Manuscript revised June 14, 2004.
Final manuscript received July 9, 2004.

[†]The authors are with the Department of Chemical Engineering, Nagoya University, Nagoya-shi, 464-8603 Japan.
[††]The author is with the Department of Computer Science and Mathematical Informatics, Nagoya University, Nagoya-shi, 464-8601 Japan.
a) E-mail: yajima@nuce.nagoya-u.ac.jp
b) E-mail: hashi@nuce.nagoya-u.ac.jp
c) E-mail: kurimoto@nuce.nagoya-u.ac.jp
d) E-mail: onogi@nuce.nagoya-u.ac.jp

Sect. 3, we develop a method of constructing and verifying a hierarchical batch control system model. We present how to operate batch processes using the constructed model in Sect. 4 and show an example in Sect. 5.

2. Specification of Batch Control

To easily cope with the frequent modifications of products to be produced and batch plant configurations, a batch control system must be flexible. As stated above we classified the available information to batch control system design into the following three categories and presented a modeling framework for batch control systems [11].

(1) Process specification

The process information called a **recipe** contains the knowledge about process-related tasks to define the production requirements for a specific product, but not the knowledge about a plant to be used. In this paper, we define a recipe as a partially ordered set of tasks as shown in Fig. 1. Roughly speaking, a recipe is described by a graph consisting of a set of nodes representing tasks and a set of arcs representing precedence relations between two tasks. For example, a recipe r_1 implies that two tasks t_1 and t_2 are sequentially carried out and again t_1 is carried out. A recipe r_2 implies that two tasks t_{11} and t_{12} are concurrently carried out and then t_{13} is carried out.

The recipe at the highest level defines the rough outline of strategy for carrying out major tasks such as reaction, separation, heating, filtration, drying, etc. An upper-level task may be developed into a partially ordered set of lower-level tasks. Every lower-level recipe defines the refinement of an upper-level task. In Fig. 1 recipes r_2 and r_3 are refinements of tasks t_1 and t_2, respectively. We call a set of recipes a **process specification**.

(2) Plant specification

The plant information includes a potentially available piece of processing/storage equipment which is used for production. It contains the information about processing units, storage vessels, sensors and actuators. Since batch control handles logical aspects of operations, the behavior of a batch control system cannot be described by differential equations. Logical operations are carried out by controlling execution of operations. In this paper, we define a **function** as a partially ordered set of operations. Similar to a recipe, a function is described by a graph consisting of a set of nodes representing operations and a set of arcs representing precedence relations between two operations. We call a set of functions a **plant specification**. A plant specification can also be described in a hierarchical manner as shown in Fig. 2.

(3) Schedule information

The schedule information contains such information as when or in what order the products are to be produced, and what equipment is to be used. The tasks in a recipe are carried out by executing the operations in a function. The

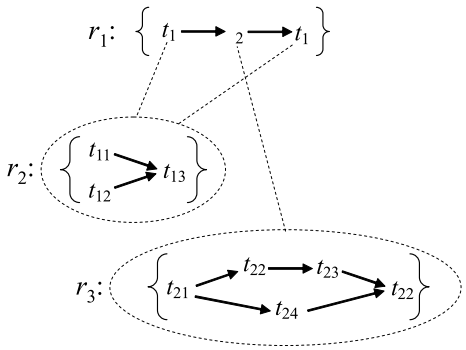


Fig. 1 Process specification.

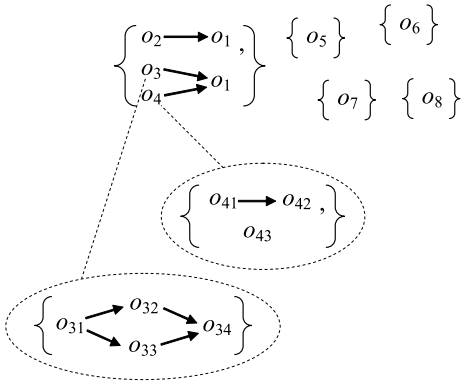


Fig. 2 Plant specification.

$t_{11} \mapsto o_5$
$t_{12} \mapsto o_3$
$t_{13} \mapsto o_4$
$t_{21} \mapsto o_6$
$t_{22} \mapsto o_2$
$t_{23} \mapsto o_7$
$t_{24} \mapsto o_8$

Fig. 3 Schedule information.

schedule information links the process specification to the plant specification. An example of schedule information is shown in Fig. 3. For example, $t_{11} \mapsto o_5$ implies that a task t_{11} in Fig. 1 is carried out by the execution of an operation o_5 in Fig. 2.

3. Modeling of Batch Control System

3.1 Representation of Process and Plant Models

If we view that the arrival of raw material, start of processing, completion of processing, discharge of product, etc. occur at asynchronous discrete instants of time, the evolution of a batch control system can be regarded as a discrete event system. A Petri net is a useful modeling tool to describe a discrete event system with interacting concurrent components. Therefore, we use a hierarchical Petri net to describe process and plant models.

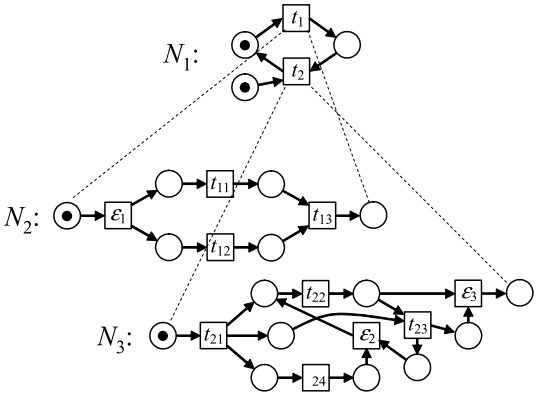


Fig. 4 Process model.

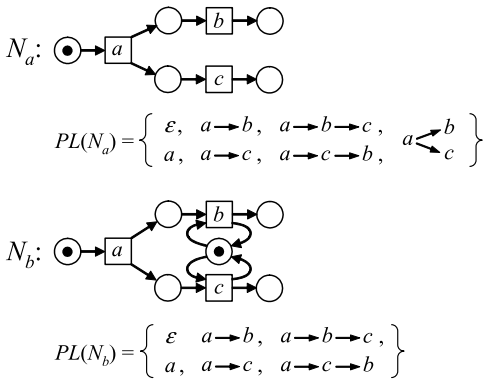


Fig. 5 Petri net partial languages.

When we describe recipes (functions) by Petri nets, tasks (operations) are modeled by transitions, and sequences of tasks (sequences of operations) are modeled by sequences of transitions. A hierarchical Petri net is an extended Petri net in which transitions can be developed into more detailed subnets in a hierarchical manner. An example of hierarchical Petri net is shown in Fig. 4. A transition t_1 in the upper-level Petri net N_1 is developed into a lower-level subnet N_2, and t_2 into N_3. When the firing of t_1 starts, the execution of N_2 starts. When the execution of N_2 is complete, the firing of t_1 is complete. The basic motivation of using a hierarchical Petri net is that it can correctly represent various logical behavior in a hierarchical manner.

Petri net languages have frequently been used to describe the behavior of Petri nets but they cannot correctly describe the concurrent Petri net behavior. The two transitions b and c in N_a shown in Fig. 5 can fire concurrently, while they cannot fire concurrently in N_b. However, the two nets N_a and N_b generate the same Petri net language $\{\varepsilon, a, ab, ac, abc, acb\}$ (ε : empty word) because Petri net languages force all transitions into a linear ordering. To describe the concurrent Petri net behavior, Grabowski applied **partial languages** to the description of Petri net behavior [14]. A partial language is a set of **partial words** each of which is a partially ordered multisets over a set of transitions. A set of all partial words generated by a Petri net N is called a **Petri net partial language**, denoted by $PL(N)$. The partial word $a\begin{matrix}\nearrow b\\ \searrow c\end{matrix}$ in $PL(N_a)$ in Fig. 5 shows the concurrent behavior. Figure 5 shows that the difference of the behavior of N_a from that of N_b can be represented by partial languages.

The recipes in Fig. 1 and the functions in Fig. 2 are essentially regarded as partial languages. If the desired behavior can be specified as a partial language, then it may possible to construct a Petri net whose partial language is the specified partial language.

Before showing a Petri net construction problem, we discuss the properties of Petri net partial languages [9].

i) If a Petri net N generates a partial word p, then it also generates a set of prefixes of p, p^{PREF} (partial words composed of leading transitions of p). For $PL(N_a)$ in Fig. 5, $a\begin{matrix}\nearrow b\\ \searrow c\end{matrix} \in PL(N_a)$, then $a, a\to b, a\to c \in PL(N_a)$. ∎

ii) If N generates a partial word p with n concurrent transitions $\{t_1, t_2, \cdots, t_n\}$, then it also generates a set of partial words each of which is obtained by forcing $\{t_1, t_2, \cdots, t_n\}$ into a linear ordering. For $PL(N_a)$, $a\begin{matrix}\nearrow b\\ \searrow c\end{matrix} \in PL(N_a)$, then $a\to b\to c, a\to c\to b \in PL(N_a)$. The partial word $a\begin{matrix}\nearrow b\\ \searrow c\end{matrix}$ is richest in concurrency. A set of the partial words richest in concurrency is called an activity of N, denoted by $FP(N)$. For N_a in Fig. 5,

$$FP(N_a) = \{\varepsilon, a, a\to b, a\to c,\ a\begin{matrix}\nearrow b\\ \searrow c\end{matrix}\}.$$

$PL(N_a)$ can be known from $FP(N_a)$. ∎

In addition to the above two properties, we denote the restriction of a partial word p over a set of transitions T' to T ($\subseteq T'$) by $p|T$. $p|T$ is a set of partial words obtained by masking the transitions contained in $T' - T$ ($T' - T$ denotes a difference set).

Based on the above discussion, we formulated a Petri net construction problem $P(X)$ in the following form [15].

Petri net construction problem $P(X)$:
Given a partial language X over a set of transitions T as a specification, construct a Petri net N such that

$$X^{PREF} = FP(N)|T \tag{1}$$

where

$$X^{PREF} = \bigcup_{p\in X} p^{PREF}$$

$$FP(N)|T = \bigcup_{p\in FP(N)} p|T$$ ∎

It should be noted that X^{PREF} is not equal to $FP(N)$, but to $FP(N)|T$. An extra transition which appears in N but does not in T is called an auxiliary transition. It may be possible to construct a Petri net such that $X^{PREF} = FP(N)|T$ by using auxiliary transitions even if we cannot construct a net such that $X^{PREF} = FP(N)$. The problem $P(X)$ implies that

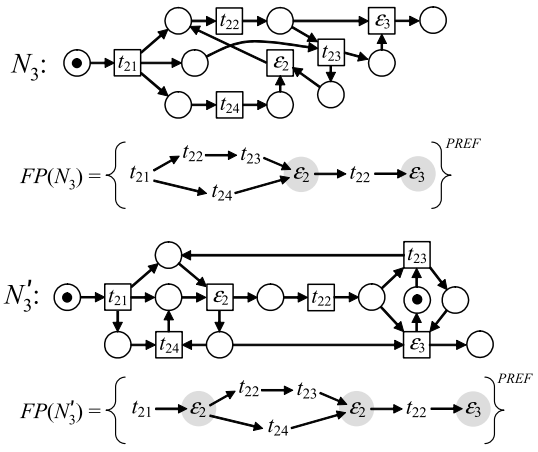


Fig. 6 Solutions of $P(r_3)$.

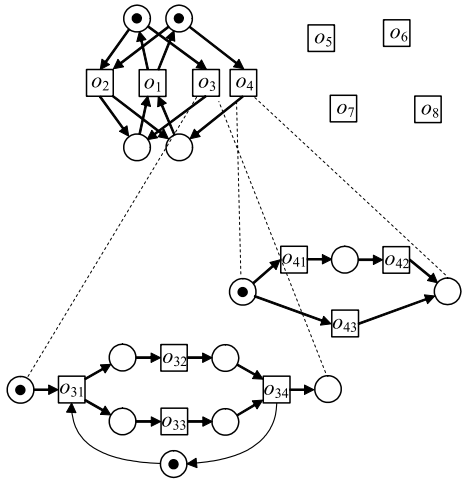


Fig. 7 Plant model.

we construct a Petri net which preserves only the sequentiality and concurrency written in the specification X when masking auxiliary transitions.

We showed the solvability of $P(X)$ and its solution techniques [7]–[9], [15]. The basic idea of solving the problem $P(X)$ is that we first divide X into small specification pieces, then construct subnets based on the divided pieces and last obtain a solution net by composing the subnets. The solution net of $P(X)$ is not always determined uniquely. For the recipe r_3 and the transition set $\{t_{21}, t_{22}, t_{23}, t_{24}\}$ in Fig. 1, both of the Petri nets, N_3 and N_3' shown in Fig. 6 are solutions of the problem $P(r_3)$. As shown in Fig. 6, N_3 and N_3' guarantee the specified behavior r_3 if we mask the auxiliary transitions ε_2 and ε_3.

The nets N_1 and N_2 in Fig. 4 also guarantee r_1 and r_2, respectively. Thus the hierarchical Petri net in Fig. 4 is a process model for the process specification in Fig. 1. The hierarchical Petri net in Fig. 7 is similarly a plant model for the plant specification in Fig. 2.

3.2 Construction of Batch Control System Model

As stated above, the linking of a process model to a plant model is done by deciding what equipment is to be used to execute the specific recipe. Since tasks are carried out by operations, the correspondence between tasks and operations defines batch control execution. Let N_R be a Petri net based process model and N_F a Petri net based plant model. The schedule information about equipment assignment is represented by the correspondence Γ from the transitions in N_R to those in N_F, namely

$$\Gamma : T(N_R) \to T(N_F) ,$$

where $T(N_R)$ and $T(N_F)$ are the sets of transitions in N_R and N_F, respectively.

A batch control system model is constructed as follows:

Batch control system model construction procedure:

Step 1: Construct a hierarchical Petri net based process model N_R and a plant model N_F by solving Petri net construction problems.

Step 2: Decide the correspondence $\Gamma : T(N_R) \to T(N_F)$ using schedule information about equipment assignment.

Step 3: Construct a Petri net $N_R \oplus N_F$ by piling the corresponding transitions in $T(N_R)$ and those in $T(N_F)$, where $\oplus$ stands for merging the transitions in N_R and N_F according to Γ.

Step 4: Eliminate unnecessary transitions, places and arcs from $N_R \oplus N_F$.

Step 5: Add subnets representing schedule information about production order, resource constraints, etc. to the hierarchical Petri net obtained in *Step* 4. ∎

Let N_C be a hierarchical Petri net based batch control system model obtained in *Step* 5. A system model N_C constructed from the process model in Fig. 4, the plant model in Fig. 7 and the schedule information in Fig. 3 is shown in Fig. 8.

3.3 Verification of Batch Control System Model

To correctly accomplish tasks written in recipes, verification of batch control system models is necessary. Execution of consistent operating procedures transfers the state of a batch process from a specified initial state to a desired final state. The state of a Petri net is given by its marking. To check the logical consistency of batch control, we apply the Petri net reachability analysis to the verification of batch control system models.

If a desired final marking $\boldsymbol{M}_f$ is reachable from a specified initial marking $\boldsymbol{M}_0$, then

$$\boldsymbol{M}_f = \boldsymbol{M}_0 + \boldsymbol{D}\boldsymbol{x} \tag{2}$$

where $\boldsymbol{D}$ is the incidence matrix of a Petri net based control

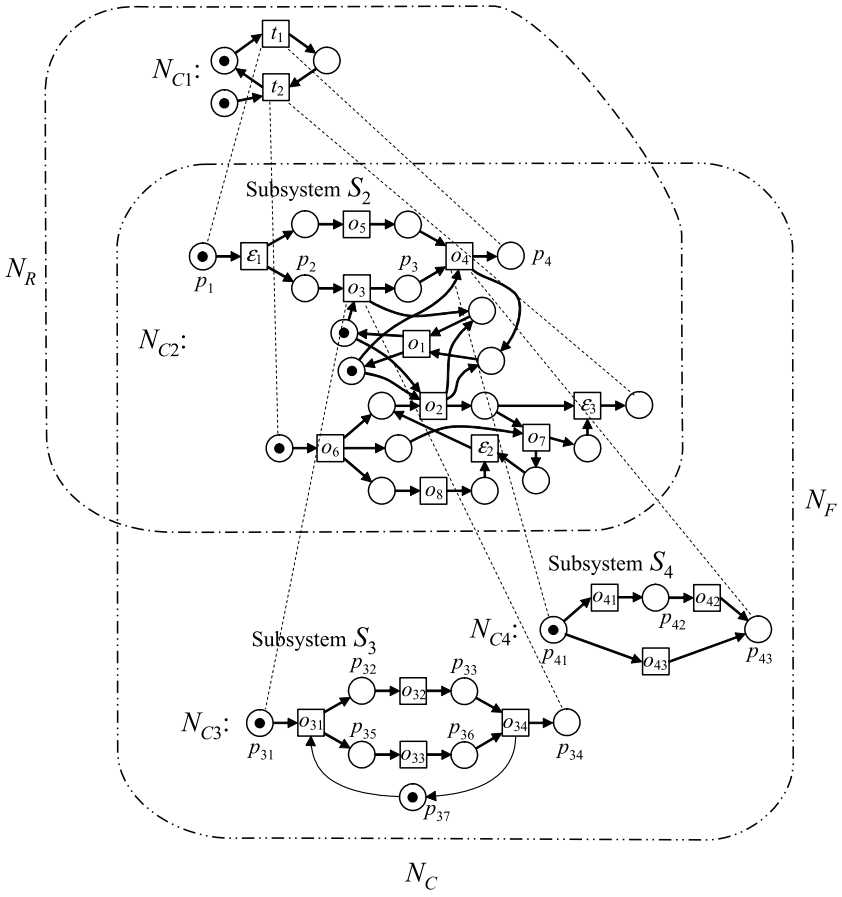


Fig. 8 Batch control system model.

system model N_C whose entry d_{ij} denotes the change of the number of tokens in the i-th place as the result of firing j-th transition, and $\boldsymbol{x}$ is a nonnegative integer vector whose entry x_j denotes the number of times that j-th transition must fire to transform $\boldsymbol{M}_0$ to $\boldsymbol{M}_f$. The existence of solutions, in nonnegative integers, of Eq. (2) is a necessary condition for reachability in N_C. The contraposition of this provides the following sufficient condition for nonreachability;

> "If Eq. (2) has no solutions in nonnegative integers, then $\boldsymbol{M}_f$ is not reachable from $\boldsymbol{M}_0$."

To start of batch productions, the initial state of a batch plant is specified. However, its desired final state is not necessarily specified in advance of batch productions. In this situation, all the entries of $\boldsymbol{M}_f$ are not necessarily known. By partitioning $\boldsymbol{M}_f$ in the form

$$\boldsymbol{M}_f = \begin{pmatrix} \boldsymbol{M}_{f1} \\ \boldsymbol{M}_{f2} \end{pmatrix}$$

Eq. (2) becomes

$$\begin{pmatrix} \boldsymbol{M}_{f1} \\ \boldsymbol{M}_{f2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}_{01} \\ \boldsymbol{M}_{02} \end{pmatrix} + \begin{pmatrix} \boldsymbol{D}_1 \\ \boldsymbol{D}_2 \end{pmatrix} \boldsymbol{x} \tag{3}$$

where $\boldsymbol{M}_{f2}$ is an unknown constant vector, and $\boldsymbol{M}_{01}$, $\boldsymbol{M}_{02}$, $\boldsymbol{M}_{f1}$, $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are known constant vectors and matrices. The solution $\boldsymbol{x}$ of Eq. (3) is obtained by solving the following homogeneous equations

$$\begin{pmatrix} \boldsymbol{D}_1 & \boldsymbol{0} & \boldsymbol{M}_{01} - \boldsymbol{M}_{f1} \\ \boldsymbol{D}_2 & -\boldsymbol{I} & \boldsymbol{M}_{02} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{M}_{f2} \\ \boldsymbol{1} \end{pmatrix} = \boldsymbol{0} \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix. Martinez's method can be used to find the solutions, in nonnegative integers, of Eq. (4). The consistency of the batch control system model N_C is verified by checking if Eq. (4) has a solution in nonnegative integers or not [11].

4. Control of Batch Processes

4.1 Synthesis of Procedural Controllers

A hierarchical Petri net has its own advantages as compared with the conventional procedural control programming languages such as Ladder diagrams, SFCs (Sequential Function Chart), decision tables, etc. It can explicitly describe the procedural control strategies and the hierarchical control structures. We presented a procedural controller synthesis method and developed C/E net based sequence controller synthesis support tool, CESeC [16]. Procedural controllers are synthesized by directly implementing a hierar-

chical Petri net based batch control system model N_C in personal computers.

A batch control system model N_C is executed by the synchronization of the controlled plant with the procedural controllers. A procedural controller sends control signals to the plant as a corresponding transition fires. It also enables a transition to fire as it receives sensor signals from the plant. There is no measure of time in a Petri net but tasks and operations represented by transitions take variable amounts of time. Therefore, information about lapse of time is necessary for synthesis of procedural controllers. In addition, information about input/output devices is also necessary for synthesis of controllers. CESeC helps a designer utilize time and input/output information.

4.2 Fault Detection of Batch Processes

A Petri net based system model N_C is not only used for batch control but also used for fault detection. For the incidence matrix $\boldsymbol{D}$ in Eq. (2), a nonnegative column integer vector such that

$$^t\boldsymbol{y}\boldsymbol{D} = \boldsymbol{0} \tag{5}$$

is called a place-invariant (p-invariant). A p-invariant $\boldsymbol{y}$ satisfies that

$$^t\boldsymbol{y}\boldsymbol{M} = {}^t\boldsymbol{y}\boldsymbol{M}_0 \tag{6}$$

where $\boldsymbol{M}_0$ is an initial marking and $\boldsymbol{M}$ is any reachable marking from $\boldsymbol{M}_0$. Every time any process variable is measured, the marking $\boldsymbol{M}$ in Eq. (6) is renewed. Basic idea of fault detection is to verify if Eq. (6) holds or not with every renewal of marking. A set of places corresponding to nonzero entries in $\boldsymbol{y}$ is called a support of $\boldsymbol{y}$. If Eq. (6) does not hold then a fault which concerns the process variables corresponding to the support of $\boldsymbol{y}$, may have occurred. Equation (6) is regarded as a constraint imposed on multiple process variables.

1) Existence of unobservable process variables

All process variables are not necessarily observable. By partitioning $\boldsymbol{M}$ into $\boldsymbol{M}^O$ corresponding to observable process variables and $\boldsymbol{M}^U$ to unobservable ones, namely $\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}^O \\ \boldsymbol{M}^U \end{pmatrix}$, Eq. (6) becomes

$$^t\boldsymbol{y}^O\boldsymbol{M}^O + {}^t\boldsymbol{y}^U\boldsymbol{M}^U = {}^t\boldsymbol{y}\boldsymbol{M}_0 \tag{7}$$

where $\boldsymbol{y}^O$ is a part of $\boldsymbol{y}$ corresponding to $\boldsymbol{M}^O$ and $\boldsymbol{y}^U$ a part of $\boldsymbol{y}$ corresponding to $\boldsymbol{M}^U$. We assume that $\boldsymbol{M}_0$ is known. If we estimate the upper and lower bounds of unobservable process variables, then the inequality,

$$\underline{\boldsymbol{M}^U} \le \boldsymbol{M}^U \le \overline{\boldsymbol{M}^U} \tag{8}$$

holds where $\overline{\boldsymbol{M}^U}$ is a vector of upper bounds and $\underline{\boldsymbol{M}^U}$ a vector of lower bounds. Inequality (8) derives that the first term in the left hand side of Eq. (7), $^t\boldsymbol{y}^O\boldsymbol{M}^O$ must satisfy

$$^t\boldsymbol{y}\boldsymbol{M}_0 - {}^t\boldsymbol{y}^U\overline{\boldsymbol{M}^U} \le {}^t\boldsymbol{y}^O\boldsymbol{M}^O \le {}^t\boldsymbol{y}\boldsymbol{M}_0 - {}^t\boldsymbol{y}^U\underline{\boldsymbol{M}^U} \tag{9}$$

Thus, if there exist unobservable process variables, Ineq. (9) instead of Eq. (6) is verified every time any observable process variable is measured. Faults of batch processes are detected by verification of Ineq. (9).

2) Detection of subsystem conflict

Figure 8 shows that the transitions o_3 and o_4 in N_{C2} are developed into the Petri nets N_{C3} and N_{C4}, respectively. Applying the p-invariant analysis to N_{C3} and N_{C4}, we obtain the following equations.

$$\boldsymbol{M}(p_{31}) + \boldsymbol{M}(p_{34}) + 2\boldsymbol{M}(p_{35}) + 2\boldsymbol{M}(p_{36}) + \boldsymbol{M}(p_{37}) = 2 \tag{10}$$

$$\boldsymbol{M}(p_{41}) + \boldsymbol{M}(p_{42}) + \boldsymbol{M}(p_{43}) = 1 \tag{11}$$

Equations (10) and (11) are the constraints imposed on the behavior of N_{C3} and that of N_{C4}, respectively. If Eq. (10) (Eq. (11)) does not hold then a fault may occur in the subsystem S_3 (S_4) modeled by N_{C3} (N_{C4}). Equations (10) and (11) are useful to detect a fault occurring in each subsystem. However, even if both of S_3 and S_4 are individually normal, the subsystem S_2 modeled by N_{C2} which involves S_3 and S_4 may exhibit abnormal behavior. Therefore, it is important to clarify the conditions for S_3 and S_4 to normally behave under the constraint imposed on S_2.

Applying the p-invariant analysis to N_{C2}, we obtain the following equation.

$$\boldsymbol{M}(p_1) + \boldsymbol{M}(p_2) + \boldsymbol{M}(p_3) + \boldsymbol{M}(p_4) = 1 \tag{12}$$

As stated in Sect. 3, when the firing of o_3 in N_{C2} starts, the execution of N_{C3} starts. When the execution of N_{C3} is complete, the firing of o_3 is complete. A token in p_2 moves to p_{31} with the start of o_3 firing and a token in p_{34} moves to p_3 with the completion of N_{C3} execution. The move of a token in p_3 is the same as that in p_2. Equation (12) indicates that two transitions o_3 and o_4 cannot fire at the same instant, that is two subsystems S_3 and S_4 cannot simultaneously work. From Eqs. (10)–(12) we obtain the following inequality.

$$\boldsymbol{M}(p_{34}) + 2\boldsymbol{M}(p_{35}) + 2\boldsymbol{M}(p_{36}) + \boldsymbol{M}(p_{37}) + \boldsymbol{M}(p_{42}) + \boldsymbol{M}(p_{43}) \le 2 \tag{13}$$

Inequality (13) is the constraint imposed on the cooperative behavior of S_3 and S_4. If S_3 and S_4 require common resources we must carefully decide their conflict resolution strategy. We devloped a method of finding out conflicts based on subsystem p-invariants.

5. Example

To show an example of batch control system design, we consider the plant shown in Fig. 9 which is often used for the benchmark plant in the field of process systems engineering. The details of the plant have been described in the report published by the JSPS's 143rd Committee on Process Systems Engineering [17].

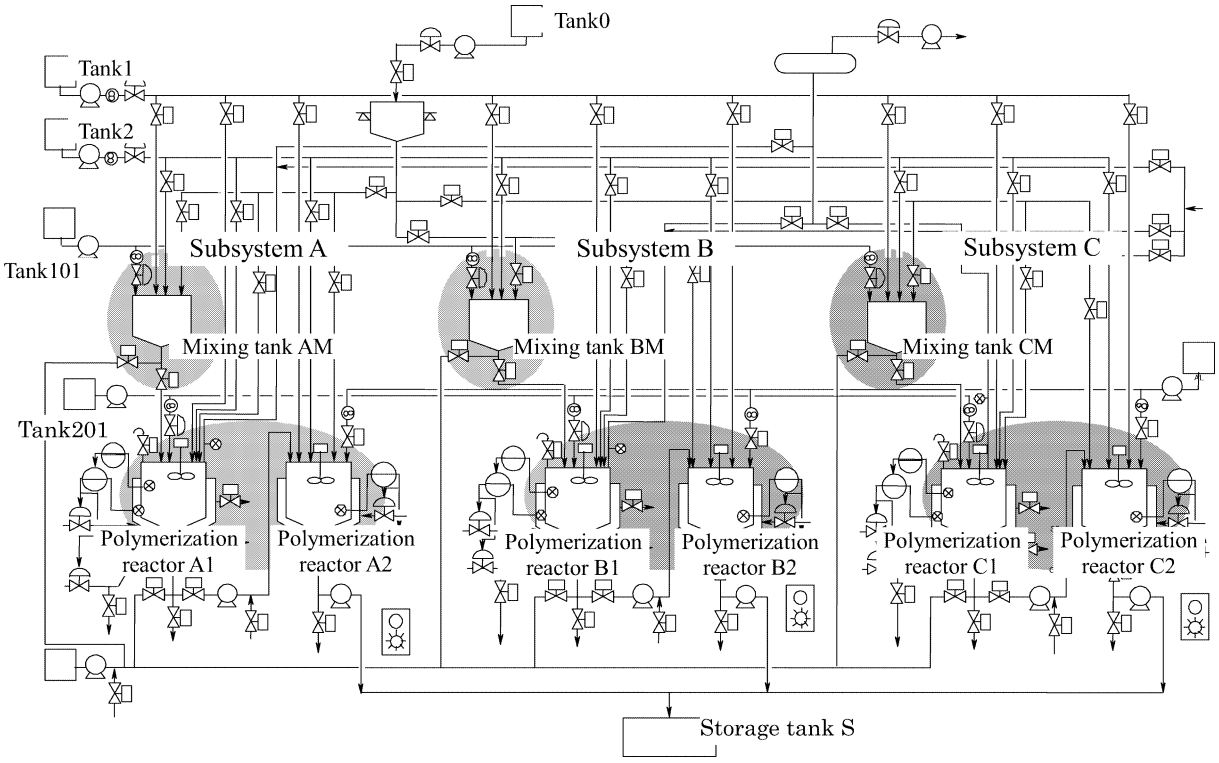


Fig. 9 Controlled plant.

The process information, plant information and schedule information are as follows:

Process information

Two kinds of product, P and Q are produced from two raw materials, R_1 and R_2. The outline of processes is as follows:

Production of P:
1: Mix R_1 and R_2 to produce the mixture M_P.
2: Charge M_P to the first polymerization reactor. After heating and stirring M_P, settle it to produce the intermediate product I_P.
3: Charge I_P to the second polymerization reactor. Heat and stir I_P to produce the final product P. Then transport P to a storage tank.

Production of Q:
1: Mix R_1 and R_2 to produce the mixture M_Q.
2: Charge M_Q to the first polymerization reactor. Heat and stir M_Q to produce the intermediate product I_Q.
3: Charge I_Q to the second polymerization reactor filled with R_2. Heat and stir I_Q to produce the final product Q. Then transport Q to a storage tank.

Plant information

The plant for production is illustrated in Fig. 9. It consists of three polymerization subsystems, A, B and C. Each subsystem has a mixing tank and two polymerization reactors. Every polymerization reactor is furnished with a temperature controller, heater and stirrer. Although the tanks and reactors carry out tasks in parallel on several batches at the same time, it is not possible to supply simultaneously one raw material to multiple tanks.

Schedule information

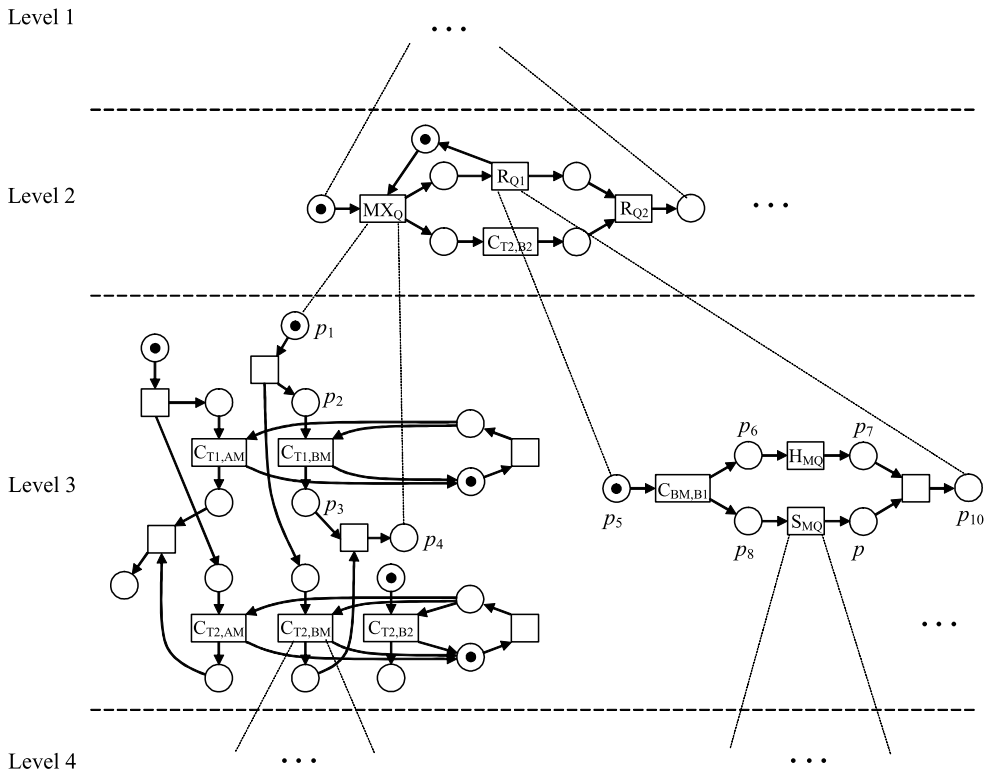
The raw materials R_1 and R_2 are stored in the feed tanks Tank1 and Tank2, respectively. The products P and Q are produced in the polymerization subsystems A and B, respectively.

Before designing a batch control system, we first built a plant simulator exhibiting the behavior of the plant in Fig. 9 using Visual Modeler (Omega Simulation Co., Ltd.) which is a dynamic simulation tool. The plant simulator contains 46 discrete variables, such as on-off state of relay switch, open-closed state of valve, etc. and 16 continuous variables, such as temperature, pressure, flow rate, etc.

We next constructed the process and plant models by solving the Petri net construction problems stated in Sect. 3 and then synthesized a batch control system model according to the procedure in Sect. 3. The Petri net based control system model is composed of 116 places and 84 transitions and structured with four hierarchical levels. The hierarchical Petri net in Fig. 10 is a part of the control system model.

1) Verification of control system behavior

To confirm the effectiveness of the proposed design approach, we examined the behavior of the controlled plant under the constructed control system model. As shown in Fig. 11 the control of the plant was executed by the synchro-



P_P (P_Q) : Production of final product P (Q)
MX_P (MX_Q) : Production of mixture M_P (M_Q)
R_{P1} (R_{Q1}, R_{Q2}) : Polymerization reaction in A1 (B1, B2)
H_{MP} (H_{MQ}) : Heating of M_P (M_Q)
S_{MP} (S_{MQ}) : Stirring of M_P (M_Q)
$C_{i,j}$: Charge of intermediate product from equipment i to equipment j

Fig. 10 Batch control system model.

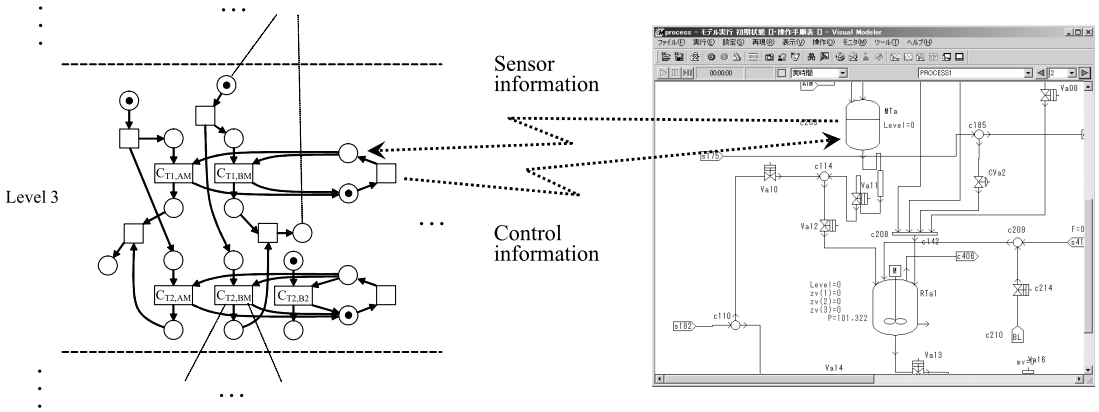


Fig. 11 Control of polymerization plant.

nization of the control system model with the plant simulator. A part of the control results is shown in Fig. 12. The chart at the second level shows that the two operations, R_{Q1} (polymerization reaction in the first reactor B1) and $C_{T2,B2}$ (charge of raw material R_2 to the second reactor B2) are carried out in parallel after the operation MX_Q (mixing of R_1 and R_2 in BM) is complete. The operation R_{Q1} is roughly divided into three operations, $C_{BM,B1}$ (charge of mixture M_Q to B1), H_{MQ} (heating of M_Q) and S_{MQ}(stirring of M_Q). The chart at the third level shows that $C_{BM,B1}$ is first carried out then the two operations H_{MQ} and S_{MQ} start at the same time. A series of simulation runs were repeatedly executed by set-

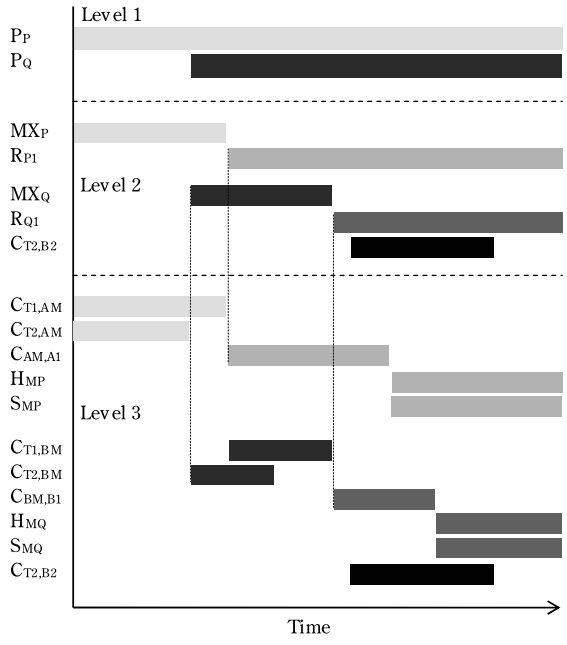


Fig. 12 Control results.

ting various physical quantities and we confirmed that the designed batch control system worked well.

2) Detection of subsystem conflict

We show an example of subsystem conflict detections stated in Sect. 4. The operations, MX_Q (mixing in BM) and R_{Q1} (polymerization reaction in B1) in Fig. 10 are developed into the detailed nets, respectively. Applying the p-invariant analysis to the Petri nets at the third level, we obtain the following equations.

Mixing in BM:

$$\boldsymbol{M}(p_1) + \boldsymbol{M}(p_2) + \boldsymbol{M}(p_3) + \boldsymbol{M}(p_4) = 1 \tag{14}$$

Polymerization reaction in B1:

$$2\boldsymbol{M}(p_5) + \boldsymbol{M}(p_6) + \boldsymbol{M}(p_7) + \boldsymbol{M}(p_8) + \boldsymbol{M}(p_9) + 2\boldsymbol{M}(p_{10}) = 2 \tag{15}$$

Equation (14) indicates that the raw materials must be charged to BM without interruption and Eq. (15) indicates that the operations with respect to B1 must be carried out with every product. Although Eqs. (14) and (15) are independent each other, we obtain the following inequality from them using the p-invariant of the Petri net at the second level.

$$2\boldsymbol{M}(p_2) + 2\boldsymbol{M}(p_3) + 2\boldsymbol{M}(p_4) + \boldsymbol{M}(p_6) + \boldsymbol{M}(p_7) + \boldsymbol{M}(p_8) + \boldsymbol{M}(p_9) + 2\boldsymbol{M}(p_{10}) \leq 2 \tag{16}$$

Inequality (16) prohibits the simultaneous execution of two operations, $C_{T1,BM}$ (inflow of raw materials to BM) and $C_{BM,B1}$ (outflow of intermediate product from BM). In this way the control constraints imposed on multiple subsystems are obtained from individual subsystem constraints by the p-invariant analysis.

6. Conclusion

A batch process is a discontinuous process with many interacting concurrent components. Although the design of a batch control system is one of the most important issues in the field of process systems engineering, it is a cumbersome and error-prone problem. In this paper, we have developed a method of constructing batch control system models in a hierarchical manner and operating batch processes using the constructed models. To attain correct and efficient batch managements, it is desired that the plan, design and control of batch processes are mutually investigated in close cooperation under a common model. We have shown that a Petri net is a candidate for a batch process common model.

References

[1] E.C. Yamalidou, E.P. Patsidou, and J.C. Kantor, "Modeling discrete-event dynamical systems for chemical process control—A survey of several new techniques," Computers and Chemical Engineering, vol.14, no.3, pp.281–299, 1990
[2] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems," Proc. IEEE, vol.77, no.1, pp.81–98, 1989.
[3] A. Sanchez, Lecture Notes in Control and Information Sciences 212, Springer-Verlag, London, 1996.
[4] M. Tittus and B. Lennartson, "Hierarchical supervisory control for batch processes," IEEE Trans. Control Syst. Technol., vol.7, no.5, pp.542–554, 1999.
[5] S. Viswanathan, C. Johnsson, R. Srinivasan, V. Venkatasubramanian, and K.E. Arzen, "Automating operating procedure synthesis for batch processes: Part I. Knowledge representation and planning framework," Computers and Chemical Engineering, vol.22, no.11, pp.1673–1685, 1998.
[6] S. Viswanathan, C. Johnsson, R. Srinivasan, V. Venkatasubramanian, and K.E. Arzen, "Automating operating procedure synthesis for batch processes: Part II. Implementation and application," Computers and Chemical Engineering, vol.22, no.11, pp.1687–1698, 1998.
[7] S. Hashizume, T. Suzuki, K. Onogi, and Y. Nishimura, "A construction problem of condition/event-nets and its solvability," Trans. SICE, vol.28, no.5, pp.632–639, 1992.
[8] S. Hashizume, A. Kaneshige, K. Onogi, and Y. Nishimura, "Control of discrete events using condition/event net models," Kagaku Kogaku Ronbunshu, vol.22, no.5, pp.1070–1078, 1996.
[9] Y. Nishimura, K. Onogi, and S. Hashizume, "Partial languages and their application to theory of discrete event systems," Trans. IEE Japan, vol.118-D, no.2, pp.158–163, 1998.
[10] K. Onogi, S. Hashizume, and Y. Nishimura, "Design of sequential control systems based on condition/event nets," Proc. International Symposium on Design, Operation and Control of Next Generation Chemical Plants (PSE Asia 2000), pp.261–266, Kyoto, Japan, Dec. 2000.
[11] T. Yajima, S. Hashizume, K. Onogi, and Y. Nishimura, "Establishment of operational procedures for batch control," Kagaku Kogaku Ronbunshu, vol.28, no.3, pp.262–267, 2002
[12] SP88 Committee, Batch Control Part 1: Models and Terminology, ISA-The Instrumentation, Systems, and Automation Society, North Carolina, 1995.
[13] SP88 Committee, Batch Control Part 2: Data Structures and Guidelines for Languages, ISA-The Instrumentation, Systems, and Automation Society, North Carolina, 2001.
[14] J. Grabowski, "On partial language," Fundamenta Informaticae, vol.4, no.2, pp.427–498, 1981.

[15] S. Hashizume, Y. Mitsuyama, Y. Matsutani, K. Onogi, and Y. Nishimura, "Construction of Petri nets from a given partial language," IEICE Trans. Fundamentals, vol.E79-A, no.12, pp.2192–2195, Dec. 1996.
[16] S. Hashizume, S. Yomogida, K. Ueda, T. Yajima, K. Onogi, and Y. Nishimura, "Modular synthesis of sequential control systems based on condition/event nets," Kagaku Kogaku Ronbunshu, vol.26, no.3, pp.443–449, 2000.
[17] 143rd Commitee on Process System Engineering, Technical Report of WS no.20 for Batch Process Modeling for Production Control, Japan Society for the Promotion of Science (JSPS), Tokyo, 1999.

Tomoyuki Yajima received B.E., M.E. and Dr.(Eng.) degrees in chemical engineering from Nagoya University in 1989, 1991 and 1994, respectively. He is currently a reseach associate at the Department of Chemical Engineering, Nagoya University. His research interests include design of batch control systems and simulation of macroreactor. He is a member of SICE and SCEJ (Society of Chemical Engineers, Japan).

Takashi Ito received B.E. and M.E. degrees in chemical engineering from Nagoya University in 2001 and 2003, respectively. He is currently a doctor course student at Graduate School of Engineering, Nagoya University. His research interests include batch control system framework and scheduling of real-time systems. He is a member of IPSJ and SCEJ (Society of Chemical Engineers, Japan).

Susumu Hashizume received B.E., M.E. and Dr.(Eng.) degrees in systems engineering from Toyohashi University of Technology in 1986, 1988 and 1992, respectively. From 1992 to 1998, he was a reseach associate at the Department of Productin System Engineering, Toyohashi University of Technology. He is currently an assistant professor at the Department of Chemical Engineering, Nagoya University. His research interests include design of hybrid systems and control of discrete event systems. He is a member of SICE, IPSJ and SCEJ (Society of Chemical Engineers, Japan).

Hidekazu Kurimoto received B.E., M.E. and Dr.(Eng.) degrees in chemical engineering from Nagoya University in 1979, 1981 and 1986, respectively. From 1986 to 1993, he was a reseach associate at the Department of Chemical Engineering, Nagoya University. He is currently an associate professor at the Department of Computer Science and Mathematical Informatics, Nagoya University. His research interests include quality systems architecture and management based on process thinking. He is a member of SICE and SCEJ (Society of Chemical Engineers, Japan).

Katsuaki Onogi received B.E., M.E. and Dr.(Eng.) degrees in chemical engineering from Nagoya University in 1973, 1975 and 1980, respectively. From 1978 to 1980, he was a reseach associate at the Department of Chemical Engineering, Nagoya University. From 1980 to 1996, he was an assistant professor, associate professor and professor at the Department of Productin System Engineering, Toyohashi University of Technology. Since 1996, he has been a professor at the Department of Chemical Engineering, Nagoya University. His research interests include control of discrete event systems and design of batch control systems. He is a member of SICE, ISCIE and SCEJ (Society of Chemical Engineers, Japan).