7

Applications of Petri Nets to Human-in-the-Loop Control for Discrete Automation Systems

Jin-Shyan Lee and Pau-Lo Hsu

1. Introduction

For discrete automation systems, certain human operations may violate desired safety requirements and result in catastrophic failure. For such human-in-the-loop systems, this paper proposes a systematic approach to developing supervisory agents which guarantee that manual operations meet required safety specifications. In the present approach, Petri nets (PN) are applied to construct a system model and synthesize a desired supervisor. Applications to 1) a rapid thermal process (RTP) in semiconductor manufacturing controlled over the Internet (Lee and Hsu, 2003) and 2) a two-robot remote surveillance system (Lee et al., 2005) are provided to demonstrate the practicability of the developed supervisory control approach. The technique developed in this paper is significant in industrial applications.

1.1 General Review

Basically, an automated process is inherently a discrete event system (DES). The Petri net (PN) has been developed as a powerful tool for modelling, analysis, simulation, and control of DES. PN was named after Carl A. Petri (1962), who created a net-like mathematical tool for describing relations between the conditions and the events. PN was further developed to meet the need in specifying process synchronization, asynchronous events, concurrent operations, and conflicts or resource sharing for a variety of industrial automated systems at the discrete-event level. Starting in the late of 1970's, researchers investigated possible industrial applications of PN in discrete-event systems as in the survey/tutorial papers of Murata (1989), Zurawski and Zhou (1994), David and Alla (1994), and Zhou and Jeng (1998).
Recently, due to the rapid development of Internet technology, system monitoring and control no longer needs to be conducted within a local area. Several

Source: Manufacturing the Future, Concepts - Technologies - Visions , ISBN 3-86611-198-3, pp. 908, ARS/plV, Germany, July 2006, Edited by: Kordic, V.; Lazinica, A. & Merdan, M.

remote approaches have been proposed which allow people to monitor the automated processes from great distances (Weaver et al., 1999; Yang et al., 2002; Kress et al., 2001; Lee and Hsu, 2004; Huang and Mak, 2001).

Practically, to perform maintenance functions in hazardous environments without their exposure to dangers is a unique application of the remote technology. By conducting remote access using IP-based networks, an entire Internet-based control system is inherently a DES and its state change is driven by occurrences of individual events. The supervisory control theory provides a suitable framework for analyzing DES (Ramadge and Wonham, 1987, 1989; Balemi et al., 1993) and most existing methods are based on automata models. The calculus of communicating systems (CCS), which was invented by Robin Milner (1989), is another classical formalism for representing systems of concurrent processes. However, these available methods often involve exhaustive searches of overall system behaviours and result in state-space explosion design as system becomes more complex. On the other hand, PN is an efficient approach to model the DES and its models are normally more compact than the automata models. Also, PN is better suitable for modelling systems with parallel and concurrent activities. In addition, PN has an appealing graphical representation with a powerful algebraic formulation for supervisory control design (Giua and DiCesare, 1991; Moody and Antsaklis, 1998; Uzam et al., 2000).

1.2 Problem Statement

Typically, an Internet-based control system (remote access using IP-based networks) is a "human-in-the-loop" system since people use a general web browser or specific software to monitor and control remotely located systems. As shown in Figure 1 (a), the human operator is involved in the loop and sends control commands according to the observed status displayed by the state and/or image feedback. Research results indicate that approximately 80% of industrial accidents are attributed to human errors, such as omitting a step, falling asleep and improper control of the system (Rasmussen et al., 1994). However, the Internet-based control literature provides few solutions for reducing or eliminating the possibility of human errors. Therefore, solutions to reduce or eliminate the possibility of human errors are required in the Internet-based control systems.

1.2 Proposed Approach

In this chapter, we propose applying a supervisory design to the present remotely-controlled and human-in-the-loop system so as to prevent abnormal operations from being carried out.

As shown in Figure 1 (b), the supervisory agent acquires the system status and makes the decision to enable/disable associated events to meet the required specifications, typically safety requirements. The human operator is then only allowed to perform the enabled events to control the system. The role of a supervisory agent is to interact with the human operator and the controlled system so that the closed human-in-the-loop system meets the required specifications and to guarantee that undesirable executions do not occur.

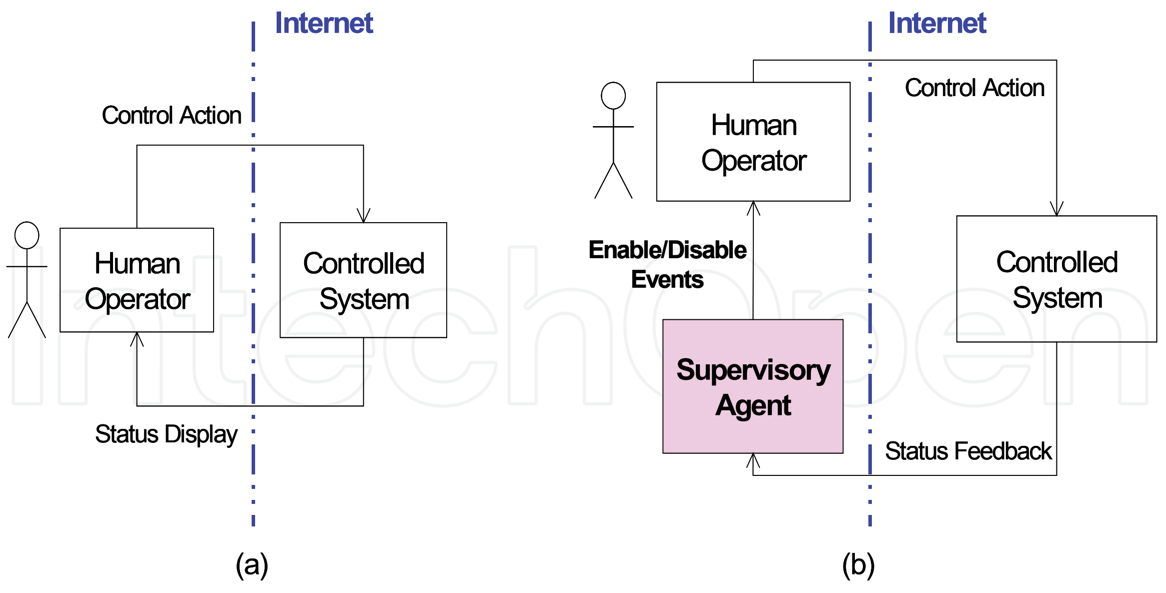


Figure 1. (a) Typical remote control system with the human in the loop. (b) The proposed remote supervisory control scheme

2. PN-Based Modelling

2.1 Basic PN Concepts

A PN is identified as a particular kind of bipartite directed graph populated by three types of objects. They are places, transitions, and directed arcs connecting places and transitions. Formally, a PN can be defined as

$$G = (P, T, I, O) \qquad (1)$$

where,

$P = \{p_1, p_2, ..., p_m\}$ is a finite set of places, where $m > 0$;
$T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$, where $n > 0$;
$I : P \times T \rightarrow N$ is an input function that defines a set of directed arcs from *P* to *T*, where $N = \{0, 1, 2, \ldots\}$;
$O : T \times P \rightarrow N$ is an output function that defines a set of directed arcs from *T* to *P*.

A marked PN is denoted as (G, M_0), where M_0: $P \rightarrow N$ is the initial marking. A transition *t* is enabled if each input place *p* of *t* contains at least the number of tokens equal to the weight of the directed arc connecting *p* to *t*. When an enabled transition fires, it removes the tokens from its input places and deposits them on its output places. PN models are suitable to represent the systems that exhibit concurrency, conflict, and synchronization.
Some important PN properties in manufacturing systems include boundedness (no capacity overflow), liveness (freedom from deadlock), conservativeness (conservation of non-consumable resources), and reversibility (cyclic behavior). The concept of liveness is closely related to the complete absence of deadlocks. A PN is said to be live if, no matter what marking has been reached from the initial marking, it is possible to ultimately fire any transition of the net by progressing through some further firing sequences. This means that a live PN guarantees deadlock-free operation, no matter what firing sequence is chosen. Validation methods of these properties include reachability analysis, invariant analysis, reduction method, siphons/traps-based approach, and simulation (Zhou and Jeng, 1998).

2.2 Elementary PN Models

At the modelling stage, one needs to focus on the major operations and their sequential or precedent, concurrent, or conflicting relationships. The basic relations among these processes or operations can be classified as follows.

1. *Sequential*: As shown in Figure 2 (a), if one operation follows the other, then the places and transitions representing them should form a cascade or sequential relation in PNs.
2. *Concurrent*: If two or more operations are initiated by an event, they form a parallel structure starting with a transition, i.e., two or more places are the outputs of a same transition. An example is shown in Figure 2 (b). The pipeline concurrent operations can be represented with a sequentially-connected series of places/transitions in which multiple places can be marked simultaneously or multiple transitions are enabled at certain markings.
3. *Cyclic*: As shown in Figure 2 (c), if a sequence of operations follow one after another and the completion of the last one initiates the first one, then a cyclic structure is formed among these operations.
4. *Conflicting*: As shown in Figure 2 (d), if either of two or more operations can follow an operation, then two or more transitions form the outputs from the same place.
5. *Mutually Exclusive*: As shown in Figure 2 (e), two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. A structure to realize this is through a common place marked with one token plus multiple output and input arcs to activate these processes.

In this chapter, PN models of the human behaviours will be constructed based on these elementary models.

2.3 System Modeling

The human behaviors can be modeled using the command/response concept. As shown in Figure 3, each human operation is modeled as a task with a start transition, end transition, progressive place and completed place. Transitions drawn with dark symbols are events that are controllable by the remote-located human through the network. Note that the start transition is a controllable event as “command” input, while the end transition is an uncontrollable event as “response” output.

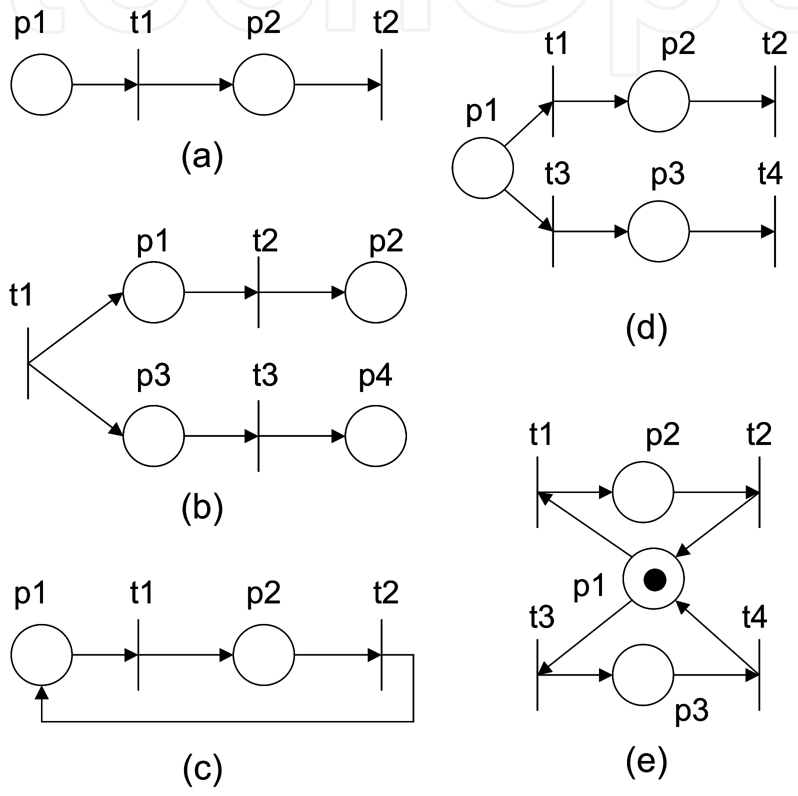


Figure 2. Basic PN models for (a) sequential, (b) concurrent, (c) cyclic, (d) conflicting, and (e) mutually exclusive relations

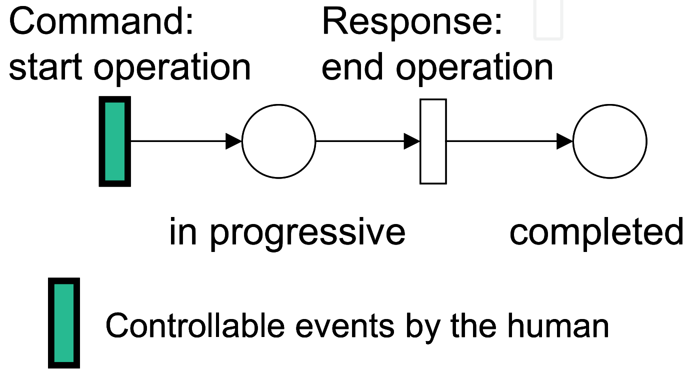


Figure 3. Modeling of human behavior using the command/response concept

3. PN-Based Supervisor Synthesis

3.1 Control Modes

For remote control via the Internet, we are interested in the following two control modes:

1. *Automatic mode*: When the system is in automatic control mode, the automatic controller autonomously controls the manufacturing process without user intervention (the human operator only needs to push a button to start the control cycle). Generally, an active sequence controller is used to automatically complete several operations in a certain order.
2. *Manual mode*: A system often must be open to manual control for various purposes, such as for test runs and fault diagnosis. Here, we examine the case in which the user can directly perform each operation. To ensure that safety constraints are not violated, the supervisory agent is on-line executed to acquire the system status and decide to either enable or disable specific operations.

3.2 Specification Types

The objective of the supervisor is to restrict the behavior of the system so that it is contained within the set of admissible states, called the specification. Two types of specifications are classified as follows:

1. *Explicit specifications for control sequences*: Generally, these specifications are "recipe-dependent". They are enforced by a sequence controller in automatic mode or by a human operator in manual mode so as to accomplish certain tasks in a desired logical order.
2. *Implicit specifications for safety requirements*: These specifications are "recipe-independent" and thus must always be obeyed throughout the whole operation of the system. Basically, these specifications are required to satisfy safety and liveness constraints. The safety specification prevents the system from performing undesirable actions, while the liveness specification ensures that a given behavior is repeatable. In automatic mode, these specifications can be effectively dealt with by the sequence controller. In manual mode, the supervisor enforces these specifications by restricting the commands available to human operators.

3.3 Supervisor Synthesis

PNs have been used to model, analyze, and synthesize control laws for DES. Zhou and DiCesare (1991), moreover, addressing the shared resource problem recognized that mutual exclusion theory plays a key role in synthesizing a bounded, live, and reversible PN. In mutual exclusion theory, parallel mutual exclusion consists of a place marked initially with one token to model a single shared resource, and a set of pairs of transitions. Each pair of transitions models a unique operation that requires the use of the shared resource.

Definition 1: Given two nets $G_1 = (P_1, T_1, I_1, O_1)$ and $G_2 = (P_2, T_2, I_2, O_2)$ with initial marking $M_{0,1}$ and $M_{0,2}$, respectively. The synchronous composition of G_1 and G_2 is a net $G = (P, T, I, O)$ with initial marking M_0:

$$G = G_1 \| G_2 \qquad (2)$$

where,

$P = P_1 \cup P_2$;

$T = T_1 \cup T_2$;

$I(p,t) = I_i(p,t)$ if $(\exists i \in \{1,2\})[p \in P_i \wedge t \in T_i]$, else $I(p,t) = 0$;

$O(p,t) = O_i(p,t)$ if $(\exists i \in \{1,2\})[p \in P_i \wedge t \in T_i]$, else $O(p,t) = 0$;

$M_0(p) = M_{0,1}(p)$ if $p \in P_1$, else $M_0(p) = M_{0,2}(p)$.

An agent that specifies which events are to be enabled and disabled when the system is in a given state is called a supervisor. For a system with plant model *G* and specification model *H*, the supervisor can be obtained by synchronous composition of the plant and the specification models:

$$S_G = G \| H \qquad (3)$$

where the transitions of *H* are a subset of the transitions of *G*, i.e. $T_H \in T_G$. Note that S_G obtained through the above construction, in the general case, does not represent a proper supervisor, since it may contain deadlock states from which a final state cannot be reached. Thus, the behavior of *S* should be further refined and restricted by PN analysis.

In this chapter, we adopt mutual exclusion concept to build the PN specification model and then compose it with the plant model to design the supervisor. The supervisor design procedure consists of the following steps:

Step 1) Construct the PN model of the human behaviors for system plants.
Step 2) Construct the PN model of specifications using the mutual exclusion concept for shared resources.
Step 3) Compose the behavior and specification models to synthesize the preliminary supervisor model.
Step 4) Analyze and verify the properties of the composed model.
Step 5) Refine the model to obtain a deadlock-free, bounded, and reversible model.

4. Agent-based Implementation

4.1 Agent Technology

The agent technology is a new and important technique in recent novel researches of the artificial intelligence. Using agent technology leads to a number of advantages such as scalability, event-driven actions, task-orientation, and adaptivity (Bradshaw, 1997). The concept of an agent as a computing entity is very dependent on the application domain in which it operates. As a result, there exists many definitions and theories on what actually constitutes an agent and the sufficient and necessary conditions for agency. Wooldridge and Jennings (1995) depicts an agent as a computer system that is situated in some environment, and that is capable of autonomous actions in this environment in order to meet its design objectives. From a software technology point of view, agents are similar to software objects, which however run upon call by other higher-level objects in a hierarchical structure. On the contrary, in the narrow sense, agents must run continuously and autonomously. In addition, the distributed multiagent coordination system is defined as the agents that share the desired tasks in a cooperative point of view, and they are autonomously executing at different sites. For our purposes, we have adopted the description of an agent as a software program associated to the specific function of remote supervision for the manufacturing system. A supervisory agent is implemented to acquire the system status and then enable and disable associated tasks so as to advise and guide the manager in issuing commands.

4.2 Client/Server Architecture

Figure 4 shows the client/server architecture for implementing the remote supervisory control system. On the remote client, the human operator uses a Java-capable web browser, such as Netscape Navigator or Microsoft Internet Explorer, to connect to the web server through the Internet. On the web server side, a Java servlet handles user authentication, while a Java applet is provides a graphical human/machine interface (HMI) and invokes the supervisory agent. In this chapter, we use Java technology to implement the supervisory agent on an industrial PLC, with a built-in Java-capable web server assigned to handle the client requests.

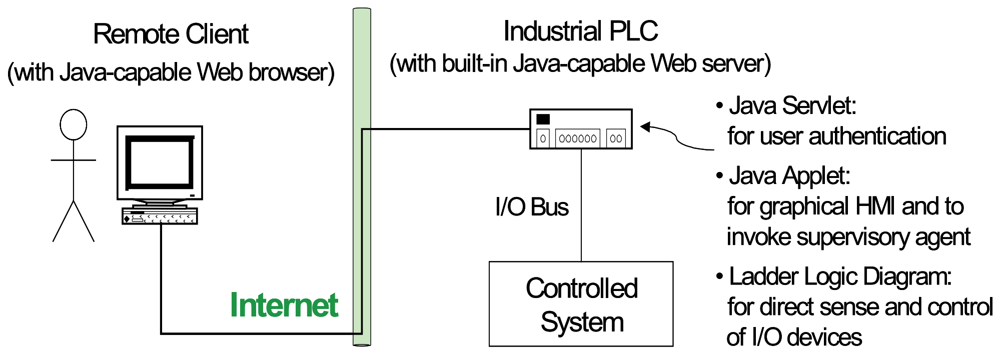

Remote Client
(with Java-capable Web browser)
Internet
Industrial PLC
(with built-in Java-capable Web server)
I/O Bus
Controlled System
• Java Servlet:
for user authentication
• Java Applet:
for graphical HMI and to invoke supervisory agent
• Ladder Logic Diagram:
for direct sense and control of I/O devices

Figure 4. Implementation architecture of the supervisory control system

4.3 Interactive Modeling

A sequence diagram of the UML (Booch et al., 1999) is applied to model client/server interaction in the remote control system. Within a sequence diagram, an object is shown as a box at the top of a vertical dashed line, called the object's lifeline and representing the life of the object during the interaction. Messages are represented by horizontal arrows and are drawn chronologically from the top of the diagram to the bottom.

Figure 5 shows the sequence diagram of the implemented remote supervisory control system. At the first stage, the *Remote Client* sends a hypertext transfer protocol (HTTP) request to the *Web Server*. Next, the *Web Server* sends an HTTP response with an authentication web page, on which the *Remote Client* can login to the system by sending a request with user/password. The *Web Server* then invokes a Java servlet to authenticate the user. If the authentication fails, the Java servlet will respond with the authentication web page again. On the other hand, if the authentication succeeds, the response of the Java servlet will be a control web page with a Java applet. The Java applet first builds a graphical HMI and constructs a socket on the specified port to maintain continuous communication with the server. Then, the Java applet acquires the system status through the constructed socket and displays it on the control web page iteratively by invoking the *Device Handler* to fetch the sensor states of *Device* objects. Finally, the supervisory agent called by the Java applet determines enable/disable control buttons on the HMI according to the current system status so as to meet the required specifications. Thus, the *Remote Client* can send an action command by pushing an enabled button to control the remote system through the constructed socket.

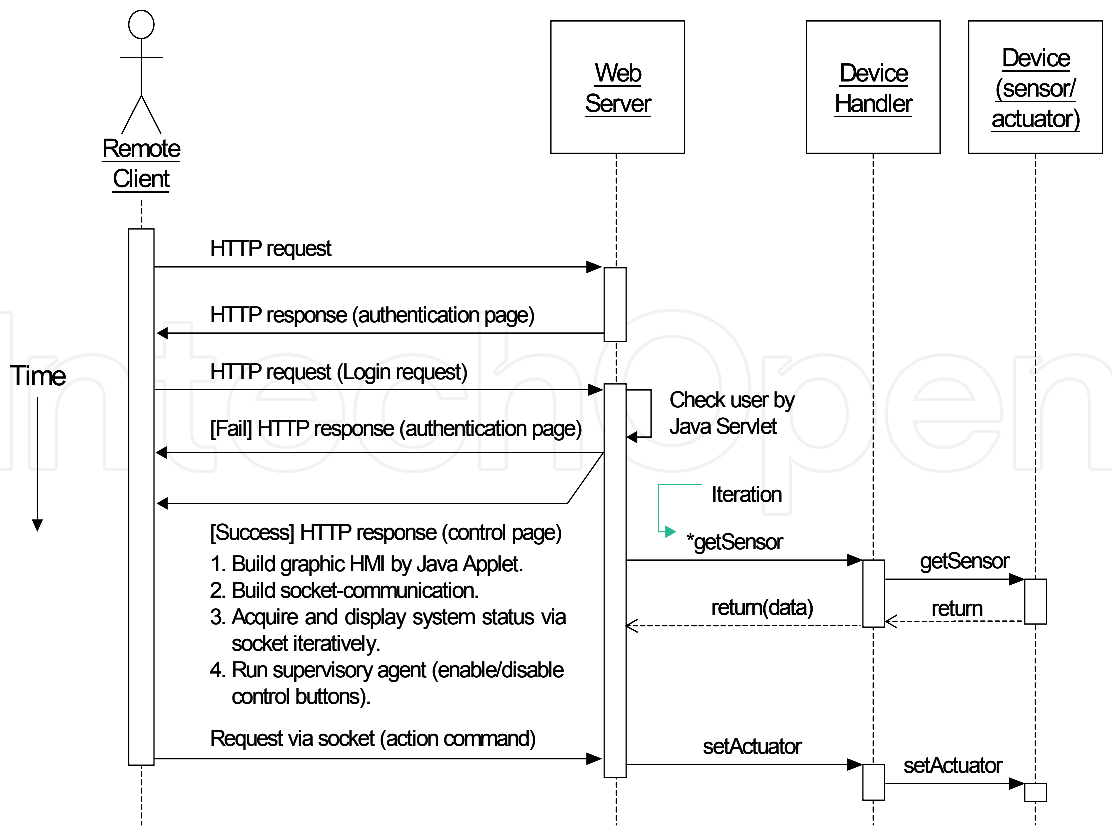

Remote Client
Web Server
Device Handler
Device (sensor/ actuator)
Time
HTTP request
HTTP response (authentication page)
HTTP request (Login request)
Check user by Java Servlet
[Fail] HTTP response (authentication page)
Iteration
[Success] HTTP response (control page)
1. Build graphic HMI by Java Applet.
2. Build socket-communication.
3. Acquire and display system status via socket iteratively.
4. Run supervisory agent (enable/disable control buttons).
*getSensor
getSensor
return(data)
return
Request via socket (action command)
setActuator
setActuator

Figure 5. Interactive modeling with sequence diagram

5. Application 1: A Rapid Thermal Process

5.1 System Description

A rapid thermal processor is a relatively new semiconductor manufacturing device (Fair, 1993). A schematic diagram of the RTP system is shown in Figure 6, which is composed of 1) a reaction chamber with a door, 2) a robot arm for wafer loading/unloading, 3) a gas supply module with a mass flow controller and pressure controller-I, 4) a heating lamp module with a temperature controller, and 5) a flush pumping system with a pressure controller-II.

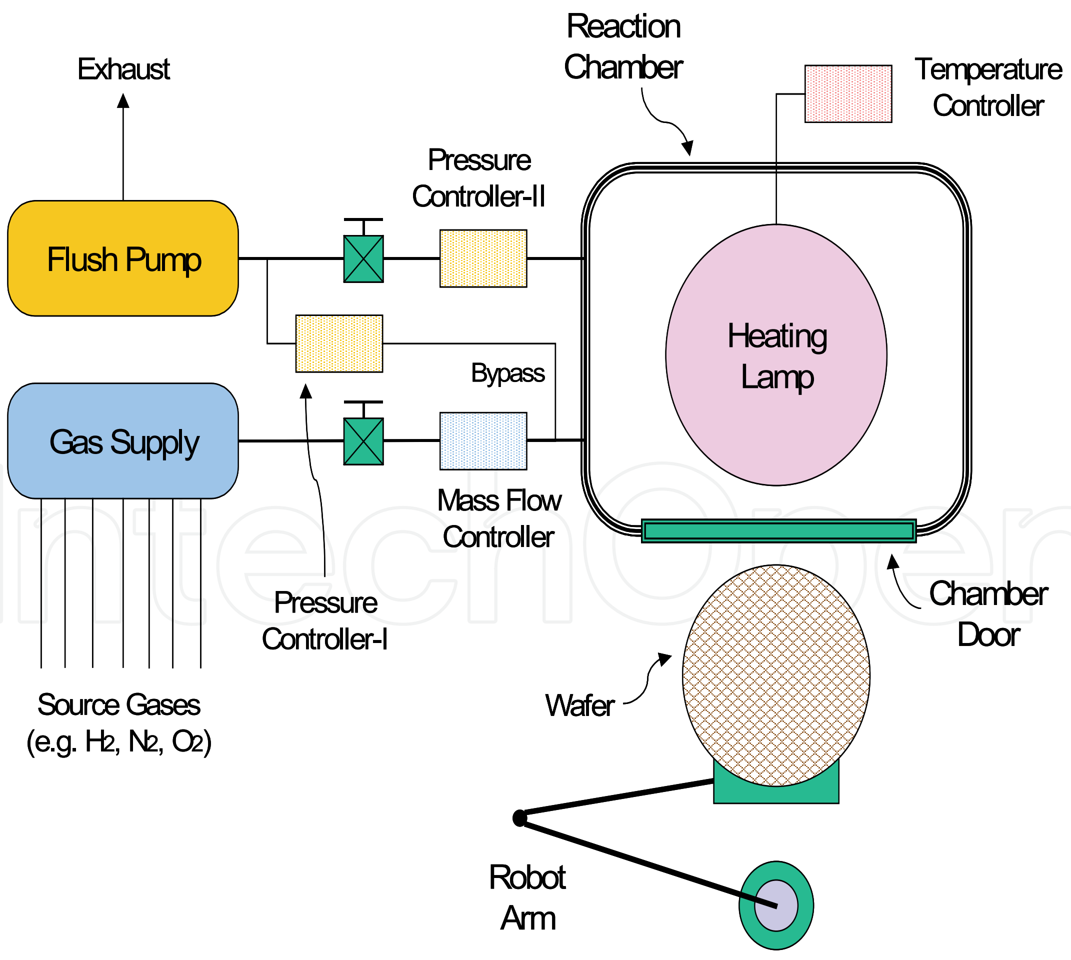


Figure 6. Schematic diagram of the RTP system

A realistic "recipe" of the hydrogen baking process, i.e. the explicit specification as mentioned in Section 3.2, is as follows:

Step 6) Load the raw wafer.

Step 7) Close the chamber door.

Step 8) Open the gas valve to supply gases with a desired gas flow rate and pressure of 2.8 liters per minute (lpm) and 0.5 Torr, respectively.

Step 9) Close the gas valve.

Step 10) Turn on the heating lamp to bake the wafer with a desired baking temperature and duration of 1000 $^{\circ}C$ and 4 seconds, respectively.

Step 11) Turn off the heating lamp to cool down the chamber to a desired temperature of less than 20 $^{\circ}C$.

Step 12) Turn on the flush pump with a desired pressure of less than 0.05 Torr.

Step 13) Turn off the flush pump.

Step 14) Open the chamber door.

Step 15) Unload the processed wafer.

The initial state of the components in the RTP is either closed or off, except that the door is open. The following safety specifications, i.e. the implicit specification mentioned in Section 3.2, must be enforced throughout system operation.

Spec-1: Wafer Loading is allowed only when no wafer is in the chamber.

Spec-2: Wafer Loading/unloading is allowed only when the door is open.

Spec-3: The gas valve must be closed when the flush pump is applied to the chamber.

Spec-4: The gas valve, heating lamp, and flush pump cannot be started when the door is open.

5.2 Automatic Controller Design

The specifications can be satisfied and involved in the sequence controller in the present automatic control mode. By applying the task-oriented concept, the PN model for the automatic control mode of the RTP is constructed as shown in Figure 7, which consists of 26 places and 20 transitions, respectively.

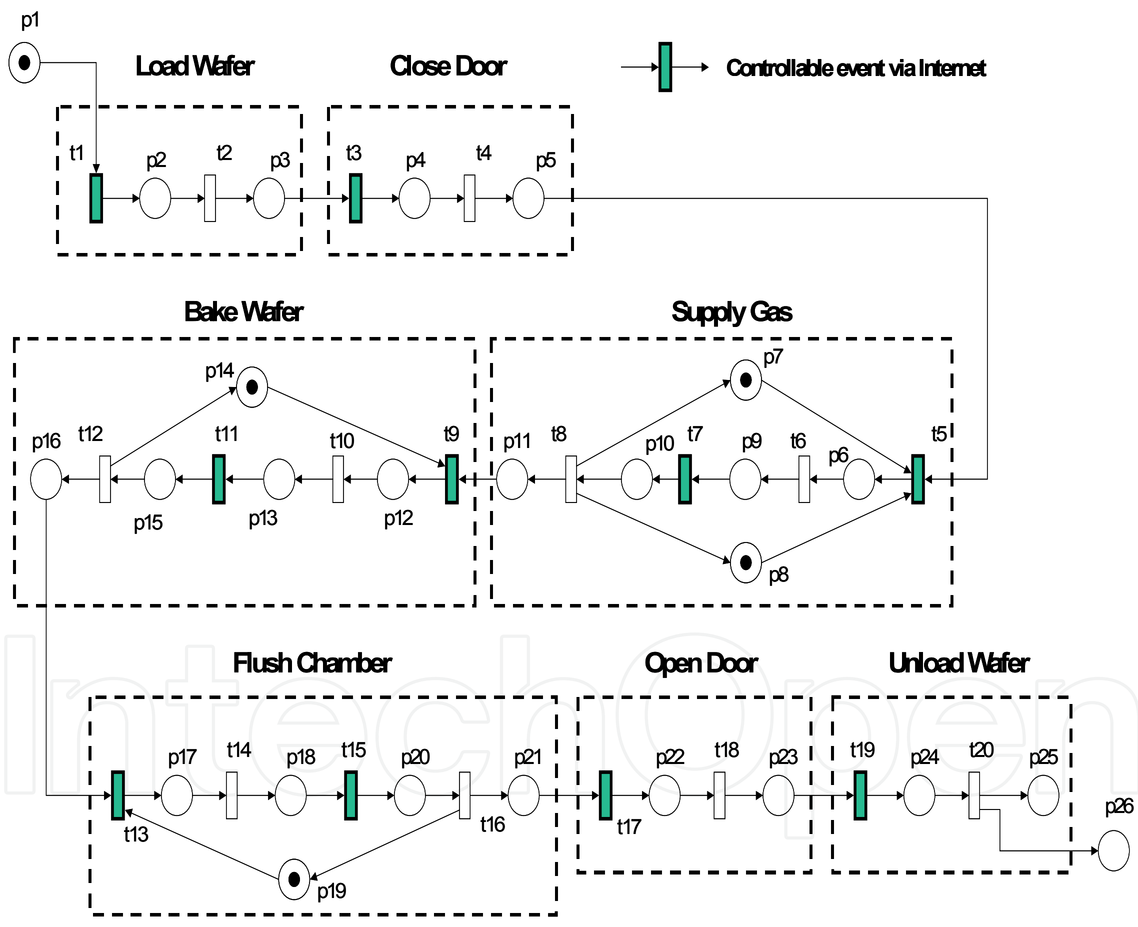


Figure 7. The PN model for automatic control of the RTP system

Corresponding notations are described in Table 1. Transitions drawn with dark symbols are events that are controllable by remote clients via the Internet.

Place	Description	Transition	Description
p1	Raw wafer buffer	t1	Cmd: start loading wafer
p2	Loading wafer	t2	Re: end loading wafer
p3	Loading wafer completed	t3	Cmd: start closing chamber door
p4	Closing chamber door	t4	Re: end closing chamber door
p5	Closing chamber door completed	t5	Cmd: start opening gas valve
p6	Opening gas valve	t6	Re: end opening gas valve
p7	Mass flow controller ready	t7	Cmd: start closing gas valve
p8	Pressure controller-I ready	t8	Re: end closing gas valve
p9	Opening gas valve completed	t9	Cmd: start turning on heating lamp
p10	Closing gas valve	t10	Re: end turning on heating lamp
p11	Closing gas valve completed	t11	Cmd: start turning off heating lamp
p12	Turning on heating lamp	t12	Re: end turning off heating lamp
p13	Turning on heating lamp completed	t13	Cmd: start turning on flush pump
p14	Temperature controller ready	t14	Re: end turning on flush pump
p15	Turning off heating lamp	t15	Cmd: start turning off flush pump
p16	Turning off heating lamp completed	t16	Re: end turning off flush pump
p17	Turning on flush pump	t17	Cmd: start opening chamber door
p18	Turning on flush pump completed	t18	Re: end opening chamber door
p19	Pressure controller-II ready	t19	Cmd: start unloading wafer
p20	Turning off flush pump	t20	Re: end unloading wafer
p21	Turning off flush pump completed		
p22	Opening chamber door		
p23	Opening chamber door completed		
p24	Unloading wafer		
p25	Unloading wafer completed		
p26	Processed wafer buffer		

Table 1. Notations for the PN of the RTP system in Figure 7

5.3 Supervisor Synthesis

For manual control mode, the plant model is formed by unconnecting each pair of transitions for the tasks in Figure 7. In the specification model, Spec-1 and Spec-2 are modeled as the pre-conditions of the associated operations, while Spec-3 and Spec-4 are built by using the mutual exclusion concept. The composed PN model of both the plant and specifications is shown in Figure 8, where A-J represent ten remote controllable tasks for the RTP system. The supervisory places **ps1-7** (**ps1** for Spec-1, **ps2-3** for Spec-2, **ps4** for Spec-3, **ps5-7** for Spec-4) are used to prevent undesired and unsafe operations on the part of the human operator. Corresponding notations for the supervisory places are described in Table 2. At this stage, the software package ARP (Maziero, 1990) is chosen to verify the behavioral properties of the composed PN model due to its graphical representation, ease of manipulation, and ability to perform structural and performance analyses. The ARP uses the reachability analysis to validate the PN properties. Results reveal that the present PN model is live and bounded. The liveness property means that the system can be executed properly without deadlocks, while the boundedness property means that the system can be executed with limited resources (e.g., limited buffer sizes).

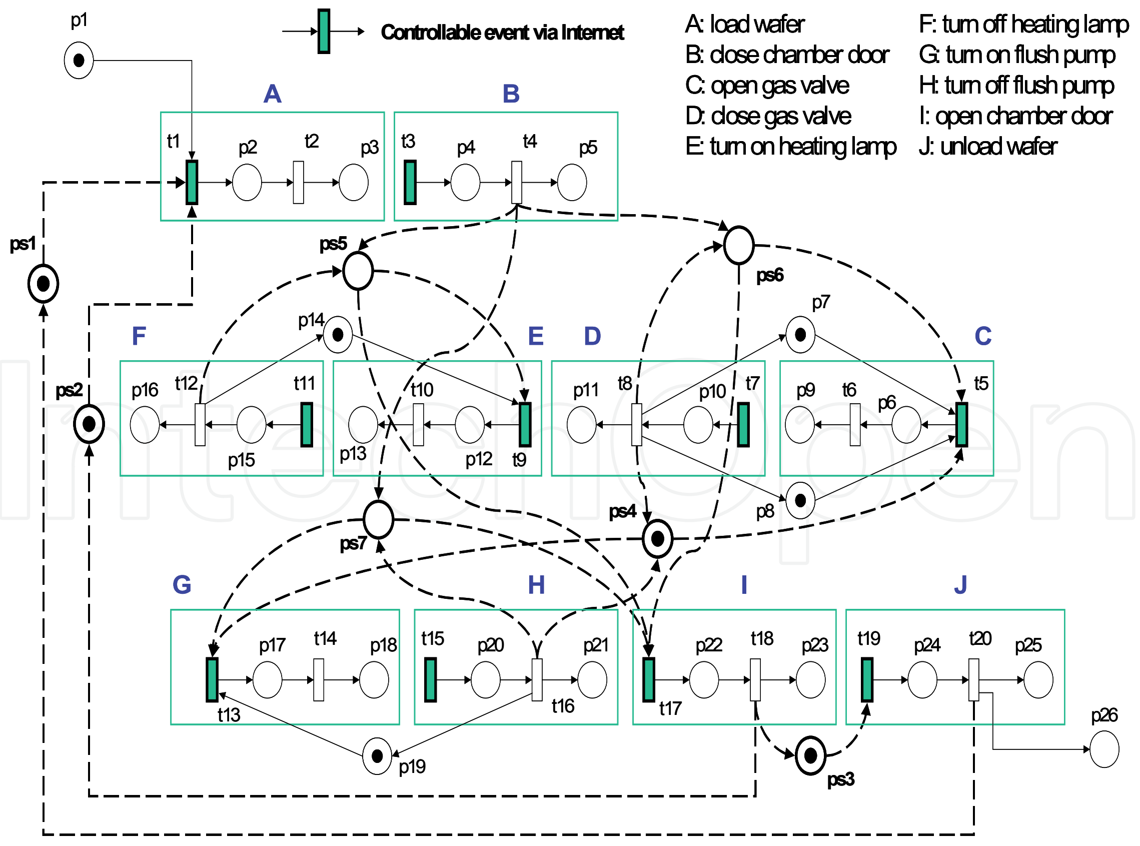


Figure 8. The composed PN model for manual control of the RTP system

Place	Description
ps1	Spec-1: chamber is empty
ps2	Spec-2: chamber door is open
ps3	Spec-2: chamber door is open
ps4	Spec-3: gas is closed/pump is off
ps5	Spec-4: door is closed/lamp is off
ps6	Spec-4: door is closed/gas is closed
ps7	Spec-4: door is closed/pump is off

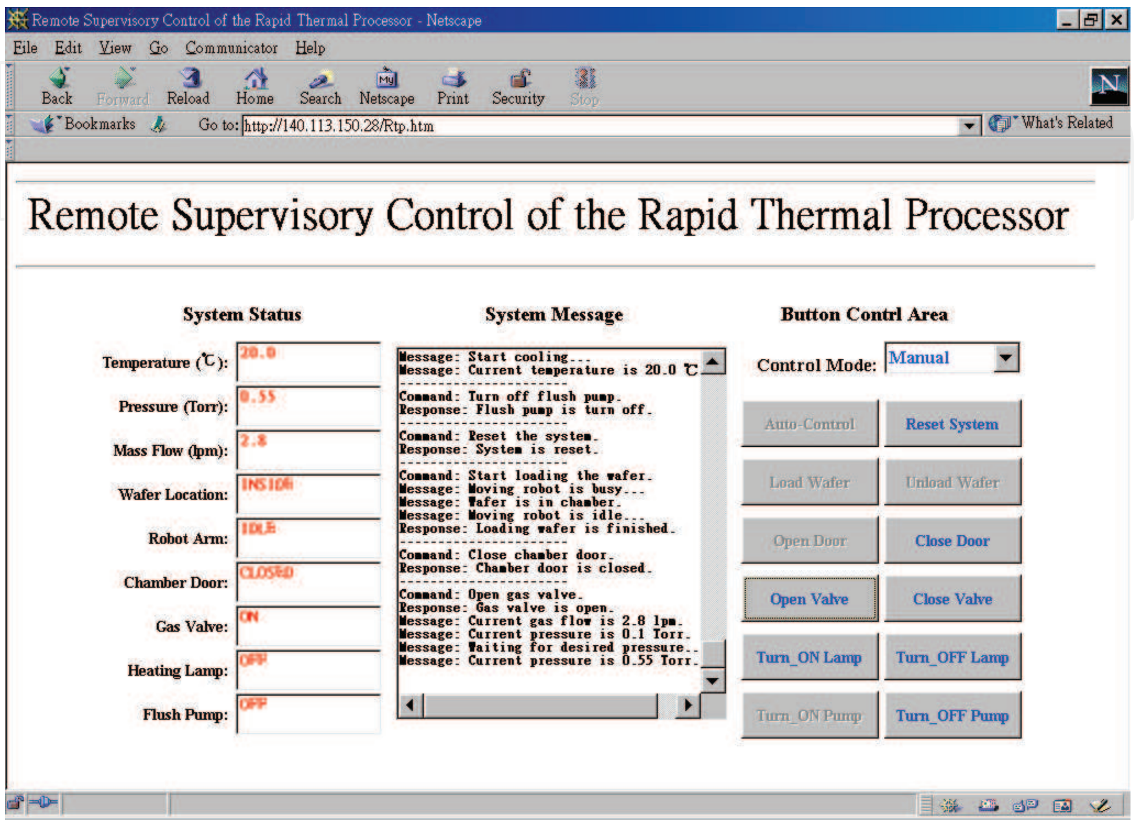
Table 2. Notations for supervisory places of PN in Figure 8

5.4 Implementation with Agent Technology

The system modeling and design developed in previous stages provide supervisory control models for implementation with agent technology. The developed supervisory agent is implemented on the Mirle SoftPLC (80486-100 CPU), an advanced industrial PLC with built-in Web server and Java virtual machine so that it can interpret the LLD, HTTP requests, and Java programs (Mirle Automation Corporation, 1999; SoftPLC Corporation, 1999).

The developed HMI, shown in Figure 9, is carefully designed to make its web pages more user-friendly and also to increase download speed by avoiding unnecessary images. Since the client users will be mainly operators and engineers, they will want effective information delivery and will not be interested in flashy graphics (Shikli, 1997).

The current system status is placed on the left, the system message is in the center, and the button control area is on the right. Figure 9 also shows the web pages for manual control mode after the **Open Valve** button has just been pushed (Step 3 in Section 5.1). In this situation, since one wafer is already in the chamber and the door is closed, the **Load Wafer** and **Unload Wafer** buttons are both disabled by the supervisory agent to meet Spec-1 and Spec-2. Moreover, the **Turn_On Pump** and **Open Door** buttons are disabled to meet Spec-3 and Spec-4, respectively. Thus, the safety requirements of the RTP processing are guaranteed as human operations are conducted.


Remote Supervisory Control of the Rapid Thermal Processor - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Go to: http://140.113.150.28/Rtp.htm What's Related

Remote Supervisory Control of the Rapid Thermal Processor

System Status

Temperature (℃):	20.0
Pressure (Torr):	0.55
Mass Flow (lpm):	2.8
Wafer Location:	INSIDE
Robot Arm:	IDLE
Chamber Door:	CLOSED
Gas Valve:	ON
Heating Lamp:	OFF
Flush Pump:	OFF

System Message

```
Message: Start cooling...
Message: Current temperature is 20.0 ℃
----------------------
Command: Turn off flush pump.
Response: Flush pump is turn off.
----------------------
Command: Reset the system.
Response: System is reset.
----------------------
Command: Start loading the wafer.
Message: Moving robot is busy...
Message: Wafer is in chamber.
Message: Moving robot is idle...
Response: Loading wafer is finished.
----------------------
Command: Close chamber door.
Response: Chamber door is closed.
----------------------
Command: Open gas valve.
Response: Gas valve is open.
Message: Current gas flow is 2.8 lpm.
Message: Current pressure is 0.1 Torr.
Message: Waiting for desired pressure..
Message: Current pressure is 0.55 Torr.
```

Button Contrl Area

Control Mode: Manual

Figure 9. Interactive web page in manual control mode at Step 3 of RTP processing (seven buttons are enabled)

6. Application 2: A Two-Robot Remote Surveillance System

6.1 System Description

Figure 10 shows a human-computer interactive system (HCIS), in which a human operator issues a command to trigger a human-controlled (semi-autonomous) robot and a computer controller automatically regulates a computer-controlled (fullly autonomous) robot both with the status feedback from the overall controlled system (i.e. both robots) through a network. Such HCIS can be applied as a remote surveillance system, which is composed of one human-controlled robot (simplified as Robot-h) and one computer-controlled robot (simplified as Robot-c). These two robots are placed on a floor with five rooms, and the moving directions for each robot are shown in Figure 11, respectively.

The Robot-h and Robot-c must traverse each doorway in the direction indicated. Moreover, in order to avoid possible collisions, Robot-h and Robot-c are not allowed simultaneously in the same room during the surveillance period. The initial states of the Robot-h and Robot-c are in R5 and R2, respectively.

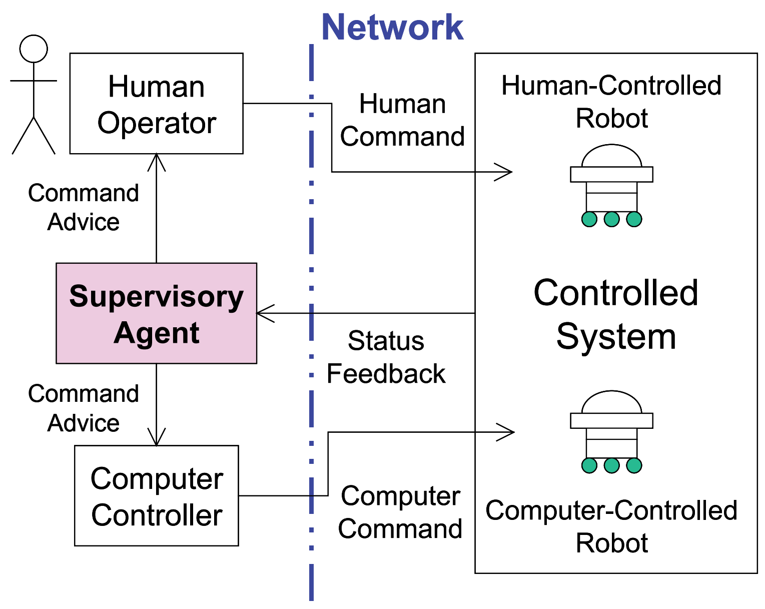


Figure 10. A two-robot human-computer interactive system

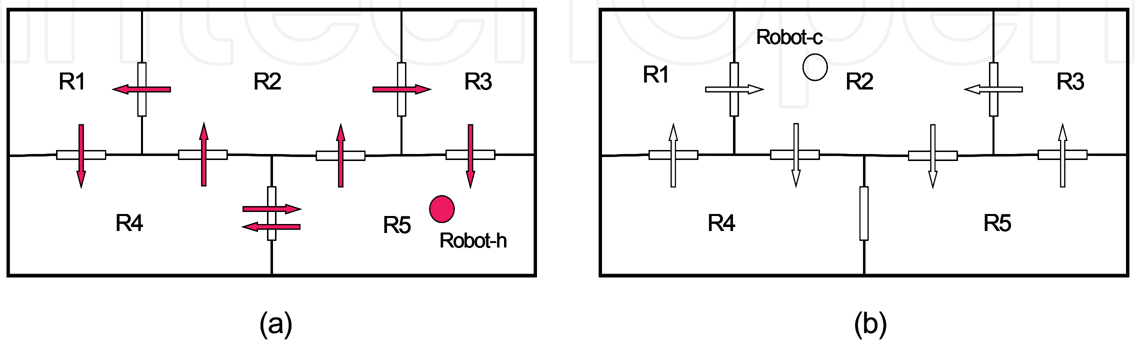


Figure 11. The schematic diagram of the two-robot remote surveillance system with the moving directions for (a) human-controlled robot, and (b) computer-controlled robot.

6.2 PN-Based System Modeling

By applying the command/response concept and based on the system description, the PN model for the human-controlled robot is constructed as shown in Figure 12 (a). It consists of 13 places and 16 transitions, respectively. On the other hand, for the computer-controlled robot, the PN model is directly built according to its located room, as shown in Figure 12 (b), which respectively consists of 5 places and 6 transitions. Corresponding notation of both the PN models is described in Table 3.

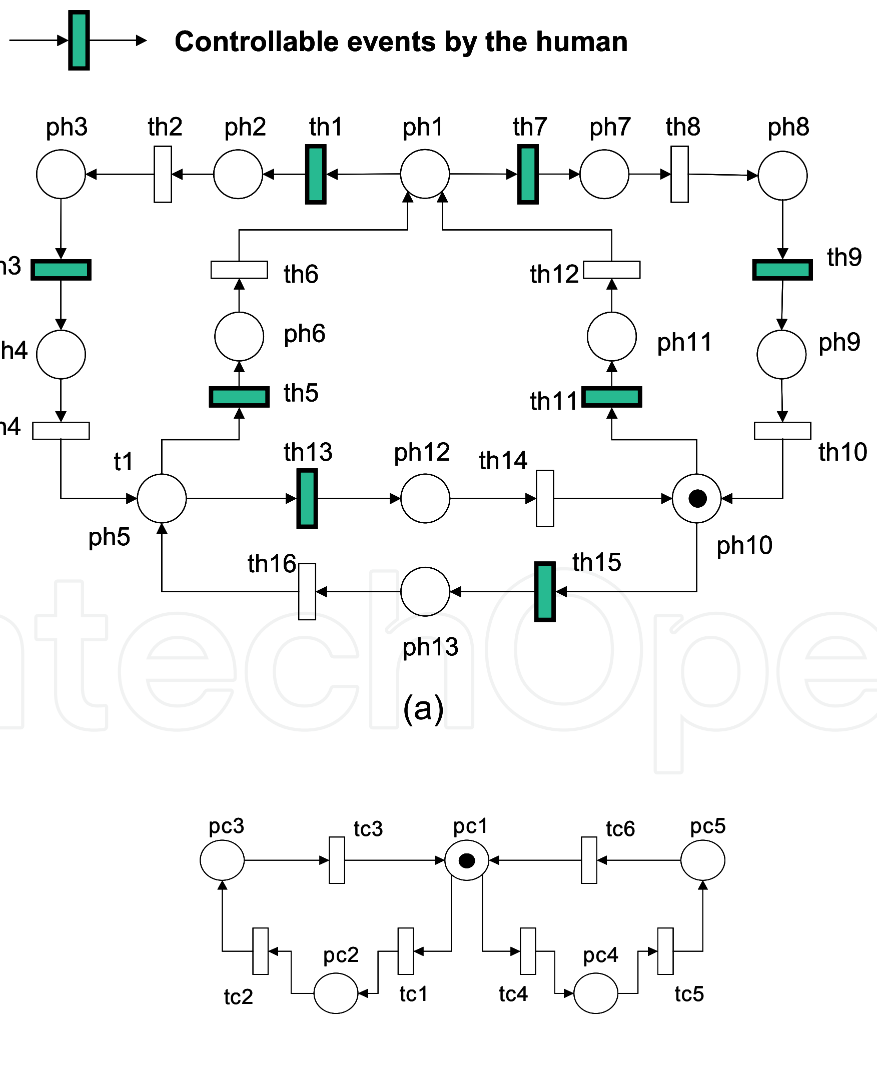


Figure 12. PN models of (a) human-controlled robot, and (b) computer-controlled robot

Place	Description	Transition	Description
ph1	Robot-h is in R2	th1	Cmd: start moving to R1
ph2	Moving to R1	th2	Re: end moving to R1
ph3	Robot-h is in R1	th3	Cmd: start moving to R4
ph4	Moving to R4	th4	Re: end moving to R4
ph5	Robot-h is in R4	th5	Cmd: start moving to R2
ph6	Moving to R2	th6	Re: end moving to R2
ph7	Moving to R3	th7	Cmd: start moving to R3
ph8	Robot-h is in R3	th8	Re: end moving to R3
ph9	Moving to R5	th9	Cmd: start moving to R5
ph10	Robot-h is in R5	th10	Re: end moving to R5
ph11	Moving to R2	th11	Cmd: start moving to R2
ph12	Moving to R5	th12	Re: end moving to R2
ph13	Moving to R4	th13	Cmd: start moving to R5
		th14	Re: end moving to R5
		th15	Cmd: start moving to R4
		th16	Re: end moving to R4
pc1	Robot-c is in R2	tc1	Move to R4
pc2	Robot-c is in R4	tc2	Move to R1
pc3	Robot-c is in R1	tc3	Move to R2
pc4	Robot-c is in R5	tc4	Move to R5
pc5	Robot-c is in R3	tc5	Move to R3
		tc6	Move to R2

Table 3. Notation for the Petri nets in Figure 12

6.3 PN-Based Supervisor Synthesis

The five rooms represent the resources shared by the two robots. Since more than one robot may require access to the same room, but in order to avoid collisions, each room can only be allowed to have one robot at a time, operations with collisions and deadlocks may thus occur. Hence, the objective is to design a supervisor to insure the whole system against these undesired situations. The required two main specifications are formulated as follows:

Spec-1: *Collision-free motions*: Robot-h or Robot-c moving to Room i is al lowed only when Room i is available, where $i = 1, 2, \ldots, 5$. Thus, we have five sub-specifications denoted as Spec-1.1 to Spec-1.5.

Spec-2: *Deadlock-free operations*: No deadlock states occur throughout system operation.

In the specification models, Spec-1.1 to Spec-1.5 are enforced by using the mutual exclusion concept. The composed PN model of both the systems and specifications is shown in Figure 13. The supervisory arcs are shown with dashed lines and the places showing the supervisory positions are drawn thicker than those showing the system positions. A supervisory place is modeled as an input place of the transitions that need such a resource, and as an output place of those that release this resource. Take an example of **ps1** that physically means Room 1 being available. It makes two transitions *th1* and *tc2* mutually exclusive. Intuitively, performance of *th1* is only allowed if Room 1 is available and *tc2* has not yet been fired. If *tc2* has been fired, *th1* cannot be executed until *tc3* is given to signal that Room 1 is available again. Thus, only one robot is allowed to be in Room 1 at any time, thereby avoiding the collision there.
The supervisory places **ps1** to **ps5** (for Spec-1.1 to Spec-1.5, respectively) are used to prevent the remote human operator and computer controller from issuing undesired commands leading to resource conflicts on the part of the system. The corresponding notation for the supervisory places (**ps1**-**ps5**) is described in Table 4.

Place	Description
ps1	Spec-1.1: R1 is available.
ps2	Spec-1.2: R2 is available.
ps3	Spec-1.3: R3 is available.
ps4	Spec-1.4: R4 is available.
ps5	Spec-1.5: R5 is available.
ps6	Spec-2.1: Robot-h is admitted into R1. Robot-c is admitted into R4.
ps7	Spec-2.2: Robot-h is admitted into R3. Robot-c is admitted into R5.

Table 4. Notation for the supervisory places in Figure 13

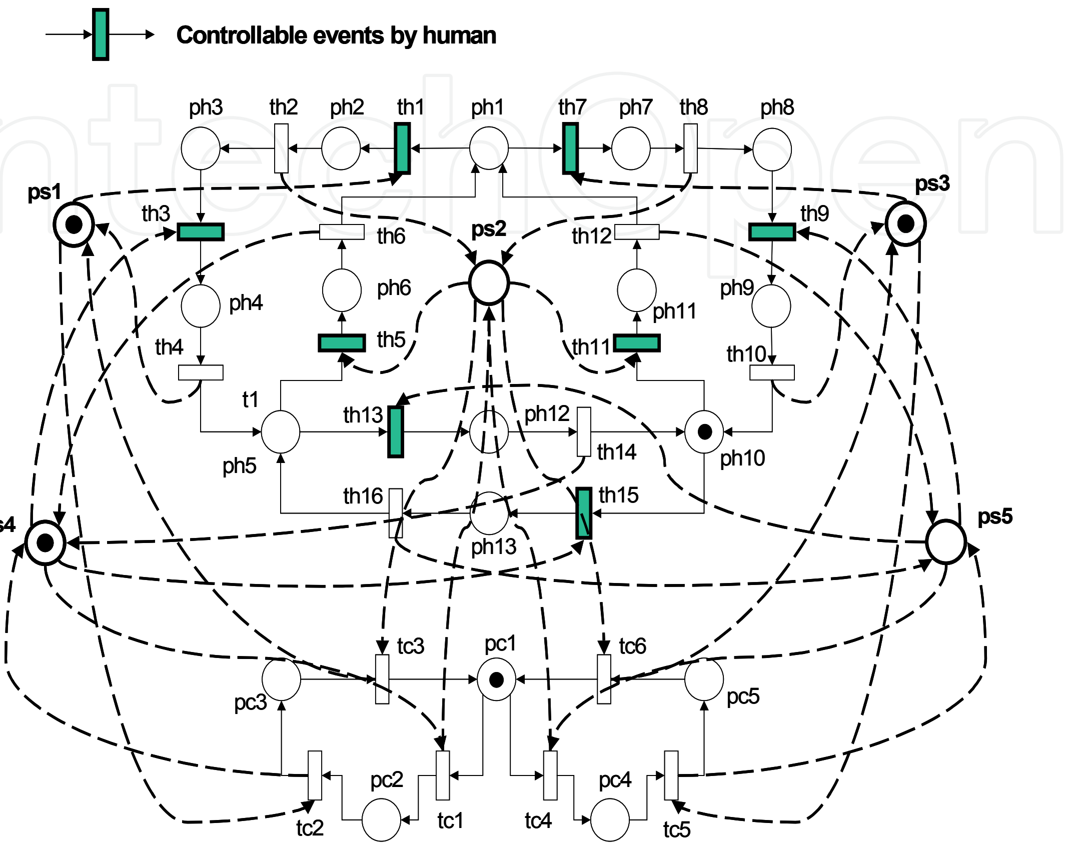


Figure 13. Preliminary composed PN model of the remote surveillance system

6.4 System Verification and Deadlock Resolution

Again, the software package ARP (Maziero, 1990) is used to verify the behavioral properties of the composed PN model using the reachability analysis. The validation result shows that two deadlocks occur with the marked places {ph3, pc2, **ps2**, **ps3**, **ps5**} and {ph8, pc4, **ps1**, **ps2**, **ps4**}, respectively. Figure 14 shows the real situations of the two deadlock states, of which the physical meaning is that if Room 1 (or Room 3) is occupied with Robot-h and Room 4 (or Room 5) is held by Robot-c, respectively, then no new events can be fired by the human or computer, and the system is deadlocked. Hence, for deadlock-free requirements, Spec-2 has two sub-specifications as follows:

Spec-2.1: Robot-h is allowed to enter Room 1 only when Robot-c is not in Room 4, and vice versa.

Spec-2.2: Robot-h is allowed to enter Room 3 only when Robot-c is not in Room 5, and vice versa.

As shown in Figure 15, **ps6** and **ps7** are further designed by using the mutual exclusion concept and then combined with the PN model in Figure 13. Take an example of **ps6**. It makes transitions *th1* and *tc1* mutually exclusive. That means either Robot-h moving to R1 or Robot-c moving to R4 is allowed to perform at a time. If *tc1* has been fired, *th1* cannot be executed until *tc2* is given to signal that Robot-c is not in R4.

Validation results (with **ps6** and **ps7**) reveal that the present PN model is deadlock-free, bounded, and reversible. The deadlock-free property means that the system can be executed properly without deadlocks, while boundedness indicates that the system can be executed with limited resources, and reversibility implies that the initial system configuration is always reachable. The corresponding notation for the supervisory places (**ps6** and **ps7**) is described in Table 4.

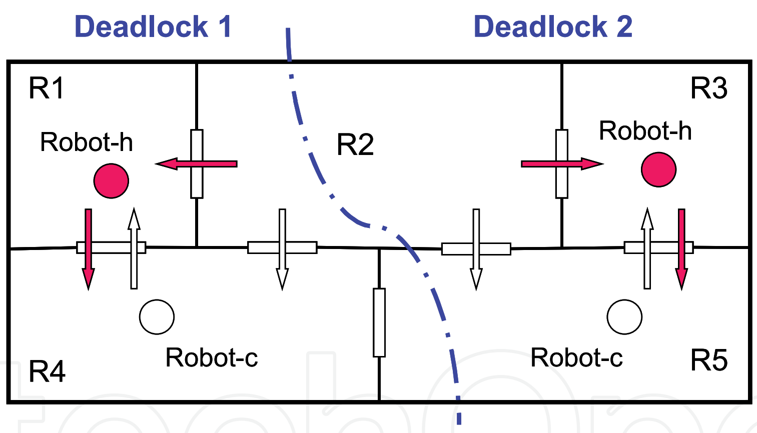


Figure 14. Two deadlock states of the PN model in the Figure 13

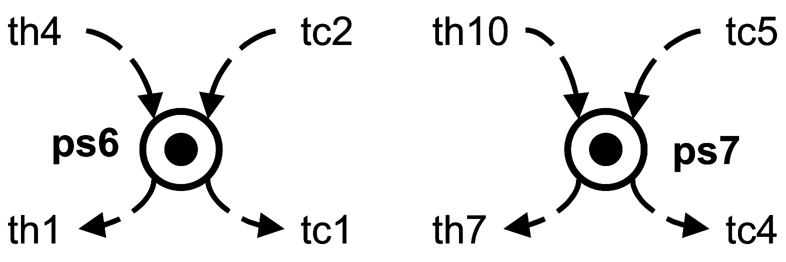


Figure 15. Supervisory places for the deadlock resolution

6.5 Discussions

On the part of the human-controlled robot, in the proposed supervisory framework, the human behavior is advised and restricted to satisfy the specifications so that the collision and deadlock are avoid during the surveillance period. As shown in Table 5, without supervisory control, the state space is 65, including the undesired collision and deadlock states. By using our proposed approach, in the preliminary supervision, i.e., only the collision-free specification (Spec-1.1 to Spec-1.5) is enforced, the state space reduces to 44. Finally, with the deadlock resolution, the state space is limited to 40 only. That means the undesired collision and deadlock states will be successfully avoided during the surveillance period. In this approach, the supervisor only consists of places and arcs, and its size is proportional to the number of specifications that must be satisfied.

Petri net models	Unsupervised system	Preliminary supervision (with deadlocks)	Complete supervision (deadlock-free)
Places	18	23	25
Transitions	22	22	22
State space	65	44	40

Table 5. Comparison between unsupervised and supervised systems

7. Conclusion

This chapter has presented a PN-based framework to design supervisors for human-in-the-loop systems. The supervisor is systematically synthesized to enforce the requirements. To demonstrate the practicability of the proposed supervisory approach, an application to 1) the RTP system in semiconductor manufacturing controlled over the Internet and 2) the two-robot remote surveillance system are provided. According to the feedback status of the remotely located system, the designed supervisory agent guarantees that all requested commands satisfy the desired specifications. On the part of human-controlled systems, the developed supervisor can be implemented as an intelligent agent to advise and guide the human operator in issuing commands by enabling or disabling the associated human-controlled buttons. Hence, for human-in-the-loop systems, the proposed approach would be also beneficial to the human-machine interface design.

Future work includes the extension of specifications to timing constraints, the multiple-operator access, and error recovery functions. Moreover, constructive definition of the synthesis algorithm should be investigated. Also, for the scalability of the supervisor synthesis, the hierarchical design can be further applied to more complex and large-scale systems.

8. References

Balemi, S.; Hoffmann, G. J.; Gyugyi, P.; Wong-Toi, H. & Franklin, G. F. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. Automat. Contr.*, Vol. 38, No. 7, pp. 1040-1059.

Booch, G.; Rumbaugh, J. & Jacobson, I. (1999). *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.

Bradshaw, J. M. (1997), Introduction to software agents, *Software Agents*, Bradshaw, J. M. Ed., Cambridge, MA: AAAI Press/MIT Press.

David, R. & Alla, H. (1994), Petri nets for modeling of dynamics systems– A survey, *Automatica*, Vol. 30, No. 2, pp. 175-202.

Fair, R. B. (1993), *Rapid Thermal Processing: Science and Technology*, New York: Academic.

Giua, A. & DiCesare, F. (1991), Supervisory design using Petri nets, *Proceedings of IEEE Int. Conf. Decision Contr.*, pp. 92-97, Brighton, England.

Huang, G. Q. & Mak, K. L. (2001), Web-integrated manufacturing: recent developments and emerging issues, *Int. J. Comput. Integrated Manuf.*, Vol. 14, No. 1, pp. 3-13, (Special issue on Web-integrated manufacturing).

Kress, R. L., Hamel, W. R., Murray, P. & Bills, K. (2001), Control strategies for teleoperated Internet assembly, *IEEE/ASME Trans. Mechatronics*, Vol. 6, No. 4, pp. 410-416, (Focused section on Internet-based manufacturing systems).

Lee, J. S. & Hsu, P. L. (2003), Remote supervisory control of the human-in-the-loop system by using Petri nets and Java, *IEEE Trans. Indu. Electron.*, Vol. 50, No. 3, pp. 431-439.

Lee, J. S. & Hsu, P. L. (2004), Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets, *IEEE Trans. Contr. Syst. Technol.*, Vol. 12, No. 2, pp. 293-302.

Lee, J. S.; Zhou M. C. & Hsu P. L. (2005), An application of Petri nets to supervisory control for human-computer interactive systems, *IEEE Transactions on Industrial Electronics*, Vol. 52, No. 5, pp. 1220-1226.

Maziero, C. A. (1990), *ARP: Petri Net Analyzer*. Control and Microinformatic Laboratory, Federal University of Santa Catarina, Brazil.

Milner R. (1989), *Communication and Concurrency*. Englewood Cliffs, NJ: Prentice Hall.

Mirle Automation Corporation (1999), *SoftPLC Controller User's Manual Version 1.2*. Hsinchu, Taiwan.

Moody, J. O. & Antsaklis, P. J. (1998), *Supervisory Control of Discrete Event systems Using Petri Nets.* Boston, MA: Kluwer.

Murata, T. (1989), Petri nets: Properties, analysis, and applications, *Proc. IEEE*, Vol. 77, No. 4, pp. 541-580.

Petri, C. A. (1962), *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2. English translation, *Communication with Automata*. New York: Griffiss Air Force Base, Tech.l Rep. RADC-TR-65--377, Vol. 1, pages 1-Suppl. 1. 1966.

Ramadge, P. J. & Wonham, W. M. (1987), Supervisory control of a class of discrete event processes, *SIAM J. Contr. Optimiz.*, Vol. 25, No. 1, pp. 206-230.

Ramadge, P. J. & Wonham, W. M. (1989), The control of discrete event systems, *Proc. IEEE*, Vol. 77, No. 1, pp. 81-98.

Rasmussen, J., Pejtersen, A. M. & Goodstein, L. P. (1994), *Cognitive Systems Engineering*. New York, NY: John Wiley and Sons.

Shikli, P. (1997), Designing winning Web sites for engineers, *Machine Design*, Vol. 69, No. 21, pp. 30-40.

SoftPLC Corporation (1999), *SoftPLC-Java Programmer's Toolkit*. Spicewood, TX.

Uzam, M., Jones, A. H. & Yücel, I. (2000), Using a Petri-net-based approach for the real-time supervisory control of an experimental manufacturing system, *Int. J. Adv. Manuf. Tech.*, Vol. 16, No. 7, pp. 498-515.

Weaver, A., Luo, J. & Zhang, X. (1999), Monitoring and control using the Internet and Java, *Proceedings of IEEE Int. Conf. Industrial Electronics*, pp. 1152-1158, San Jose, CA.

Wooldridge, M. & Jenkins, M. R. (1995), Intelligent agents: theory and practice, *Knowledge Engineering Review*, Vol. 10, No. 2, pp. 115–152.

Yang, S. H., Chen, X. & Alty, J. L. (2002), Design issues and implementation of Internet-based process control systems, *Contr. Engin. Pract.*, Vol. 11, No. 6, pp. 709-720.

Zhou, M. C. & DiCesare, F. (1991), Parallel and sequential mutual exclusions for Petri net modeling for manufacturing systems, *IEEE Trans. Robot. Automat.*, Vol. 7, No. 4, pp. 515-527.

Zhou, M. C. & Jeng, M. D. (1998), Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri net approach, *IEEE Trans. Semicond. Manuf.*, Vol. 11, No. 3, pp. 333-357, (Special section on Petri nets in semiconductor manufacturing).

Zurawski, R. & Zhou, M. C. (1994), Petri nets and industrial applications: a tutorial, *IEEE Trans. Ind. Electron.*, Vol. 41, No. 6, pp. 567-583, (Special section on Petri nets in manufacturing).