Implementation of a Remote Hierarchical Supervision System Using Petri Nets and Agent Technology

Jin-Shyan Lee and Pau-Lo Hsu, *Member, IEEE*

***Abstract*—For remote control systems, certain human operations may violate desired requirements and result in catastrophic failure. For such human-in-the-loop systems, this paper implements a hierarchical supervision system to guarantee that remote human-issued commands meet required specifications. In the presented scheme, Petri nets are applied to model, design, and verify both the local controller and the remote supervisor. Then, the agent technology is adopted to implement the supervisor as an intelligent agent for an online supervision of the remote control system. Hence, undesired resource conflicts and deadlock states can be avoided. An application to a flexible manufacturing system controlled over the Internet is provided to illustrate the developed approach. Implementation results show that by applying the presented hierarchical scheme, the supervisor has a more compact model with fewer states. Moreover, fewer request/response transmissions are consumed so that the effects of time delays and packet losses across the Internet could be moderated.**

***Index Terms*—Agent technology, discrete event systems, hierarchical supervision, Java, Petri nets, remote monitoring and control.**

I. Introduction

OVER the last decade, due to the rapid development of Internet technology, several approaches have been proposed to develop Web-based systems for remote monitoring and control of distributed manufacturing systems [1]–[8]. As compared with the traditional control, remote control allows people to monitor the processes of manufacturing systems from great distances and to perform maintenance functions in hazardous environments without exposure to dangers. Typically, an Internet-based control system 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 Fig. 1(a), the remote manager 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 [9]. However, the Internet-based control literature provides few solutions for reducing or eliminating the possibility of human errors. Basically, an Internet-based control system is a discrete-event system (DES), which is a dynamic system with state changes driven by occurrences of individual events. Moreover, supervisory control theory provides a suitable framework for analyzing DES [10]–[12] and its hierarchical scheme is a familiar approach to the design of large-scale DES to reduce design complexity [13]–[17].

This paper proposes a hierarchical scheme to design such systems for the supervision of remote-controlled processes. As shown in Fig. 1(b), we use a three-level architecture. In the command level, the abstract model is a simplified representation of the controlled system and is employed by the remote manager to make decisions for task allocation. Here, a task is a group of certain steps and the manager can send task requests to control the remotely located processes according to the displayed status. In this way, the manager exercises "virtual" control over the behavior of the abstract model. Actually, the manager sends a request for a decided task to the local controller, which really regulates the detailed operations of the task with event feedback in the control level. State changes in the system will eventually be conveyed in a summary form to the abstract model via the response channel. To avoid resource conflicts and deadlock, an agent is designed to acquire the system status and then enable and disable associated tasks so as to advise and guide the manager in issuing commands at the supervisory level. Thus, the human manager is only allowed to issue the enabled tasks, and the hierarchical loop is closed in this way.

Most existing design methods for a hierarchical supervision are based on automata models, which often involve exhaustive searches of overall system behavior and result in state-space explosion problems. One way of dealing with these problems is to model the DES with Petri nets (PNs) [18], [19]. PN modeling is normally more compact than the automata approach and is better suited for modeling concurrent systems. In addition, PNs have an appealing graphical representation with a powerful algebraic formulation, and have, therefore, generated intense interest among many researchers [20]–[24]. In this paper, PNs are used in designing both the remote supervisor and the local controller, yielding compact and graphical models for the hierarchical supervision.

We have found that the hierarchical supervision literature merely discusses how to implement various abstract supervisory models in real applications [13]–[15]. This paper demonstrates the feasibility and practicability of the proposed hierarchical

Manuscript received December 4, 2004; revised July 18, 2005. This work was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC 92-2917-I-009-005 and in part by the Ministry of Economic Affairs under the Embedded System Software Laboratory in Domestic Communication and Optoelectronics Infrastructure Construction Project. This paper was recommended by Associate Editor M. Jeng.

J. S. Lee is with the Information and Communications Research Laboratory, Industrial Technology Research Institute, Hsinchu 31040, Taiwan, R.O.C. (e-mail: jinshyan_lee@itri.org.tw).

P. L. Hsu is with the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan, R.O.C. (e-mail: plhsu@cc.nctu.edu.tw).

Digital Object Identifier 10.1109/TSMCC.2006.876056

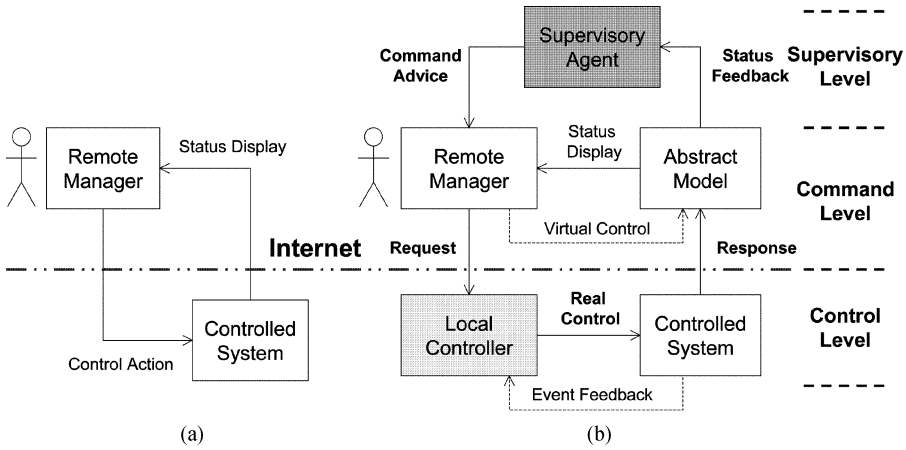

Fig. 1. (a) Basic remote control system over the Internet. (b) Proposed three-level architecture for hierarchical supervision.

supervision model by applying it to a flexible manufacturing system (FMS), where the local controller is implemented by ladder logic diagrams (LLDs) and the supervisory agent is implemented using agent technology on an industrial programmable logic controller (PLC) [25], [26]. When executing a number of concurrent operations, our approach ensures that remote control tasks via the Internet meet the given resource constraints and deadlock-free requirements with fewer packet transmissions required. Also, results show that the supervisor synthesis of the presented hierarchical scheme is less complex by far than a nonhierarchical one.

II. PN-Based Design For Hierarchical Supervision

This section first introduces the PN concept in production processes, and then, shows the different specifications for the command level and control level, separately, in remote supervisory control design. Finally, the PN-based design for the supervisor and the controller is introduced.

A. PN in Production Processes

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

$$\text{PN} = (P, T, I, O, M_0)$$

where

$P = \{p_1, p_2, \ldots, p_m\}$	is a finite set of places, where $m > 0$;
$T = \{t_1, t_2, \ldots, t_n\}$	is a finite set of transitions with $P \cup T \neq \oslash$ and $P \cap T = \oslash$, 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;
$M_0 : P \rightarrow N$	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 boundness (no capacity overflow), liveness (freedom from deadlock), conservativeness (conservation of nonconsumable 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 [20]. Validation methods of these properties include reachability analysis, invariant analysis, reduction method, siphons/traps-based approach, and simulation [22]. Among them, simulation is often used in real-world cases due to its convenience and effectiveness for engineers to validate the desired properties of manufacturing systems.

At the modeling 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 Fig. 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 Fig. 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 Fig. 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.

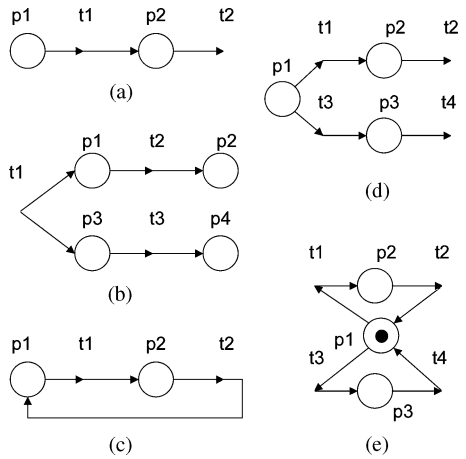

Fig. 2. Basic PN models for (a) sequential, (b) concurrent, (c) cyclic, (d) conflicting, and (e) mutually exclusive relations.

4) *Conflicting*: As shown in Fig. 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 Fig. 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.

B. Specification Separation

The objective of the hierarchical supervision is to restrict the behavior of the system so that it is contained within the desired states, called the specifications. The specifications are separated into two levels as follows.

1) Command-level specifications for recipes, resources, and liveness: These specifications require that the logical order of each recipe, resource constraints, and liveness requirement are satisfied throughout all operations of the system. The recipe specification indicates the sequence of tasks to be executed, and it can be modeled as a sequential flow. The resource specification presents the physical constraints of the limited resources, and shared resources can be adequately expressed in terms of mutual exclusion conditions. The liveness specification ensures that a given behavior is deadlock-free and repeatable, and it can be preserved by deadlock analysis with avoidance policies [24]. In the proposed hierarchical architecture, the supervisory agent enforces these specifications by restricting the task commands available to the remote manager.

2) Control-level specifications for detailed operations: These specifications are the detailed logical operations of each task. In the proposed hierarchical architecture, the control-level specifications are enforced by a local controller, which accomplishes certain operations of the requested task for the physical plant in a desired logical order.

To summarize, the system requirements are separated into the command-level specification, which results in nondeterministic sequences of tasks, and the control-level specification, which leads to detailed deterministic operations of each task. The proposed separation not only reduces the design complexity of the supervisor synthesis, as shown latter, but also makes the system design more flexible, since it avoids the need to redesign the local controller, as only the command-level specification varies.

C. Design of Supervisor to Meet Command-Level Specifications

PNs have been used to model, analyze, and synthesize control laws for DES. Zhou and DiCesare [27], moreover, addressing the shared resource problem, recognized that mutual exclusion theory plays a key role in synthesizing a live, bounded, 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 task that requires the use of the shared resource. In this paper, we adopt mutual exclusion theory to build the resource specification models and then compose them with the recipe models to design the supervisor. The supervisor design procedure consists of the following steps.

Step 1: Construct the PN model of the recipe specifications in the command level using the task-oriented approach.
Step 2: Build the PN model of the resource specifications using the mutual exclusion concept.
Step 3: Compose the recipe and resource models to yield the basic supervisor model.
Step 4: Analyze and refine the supervisor model to obtain a deadlock-free, bounded, and reversible model.

The PN recipe model is constructed using the task-oriented concept. Each task is modeled with a start transition, an end transition, a progressive place, and a completed place. Note that the start transition, as the "command" input is a controllable event, while the end transition, as the "response" output is an uncontrollable event. Obviously, the presented hierarchical scheme is endowed with task-based modularity in the command level.

D. Design of Local Controller to Meet Control-Level Specifications

The logical behavior of each task in the control level is a deterministic process. For the local controller design, the detailed PN models of each controllable task in the recipe are built to describe the detailed operations and follow the deterministic sequences in this stage. Applying the PN to design the controller leads to a unified PN-based approach to develop the hierarchical supervision, and thus facilitates the use of established PN analysis and implementation methods.

III. Implementation of Hierarchical Supervision

This section first describes the agent concept, and then shows the implementation architecture and interactive modeling of the hierarchical supervisory control system. Finally, the reasons of choosing implementation methods in Java technology are mentioned.

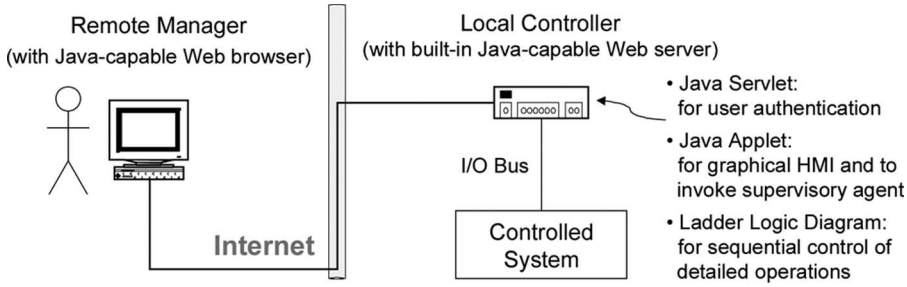

Fig. 3. Implementation architecture of the remote hierarchical supervision system.

A. 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 [28]. 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 [29] depict an agent as a computer system that is situated in some environment and that is capable of autonomous actions in this environment 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 with 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.

B. Client/Server Architecture

Fig. 3 shows the client/server architecture for implementing the remote hierarchical supervision system. On the remote client side, the manager uses a Java-capable Web browser, such as Netscape Navigator or Microsoft Internet Explorer, to connect to the local controller through the Internet. On the server side, a Java servlet handles user authentication, while a Java applet provides a graphical human/machine interface (HMI) and invokes the supervisory agent. In this paper, 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 [25], [26]. In addition, the LLD is used to implement the local controller on the same PLC so as to perform the detailed operations of the requested tasks. Our choice of using LLD to implement the local controller is due to its wide use in industry, while using Java to implement the supervisory agent is due to its object-orientation, portability, safety, and built-in support for networking and concurrency [30], [31]. The object-oriented programming is one where each small part of the program is considered as a separate object that can interact with other objects. The advantage of object-oriented software is that blocks of code can easily be reused in different parts of the program, or even in different programs. This reduces development time, and therefore, costs [32].

C. Interactive Modeling

A sequence diagram of the unified modeling language (UML) [33] is applied to model client/server interaction in the proposed remote hierarchical supervision 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.

Fig. 4 shows the sequence diagram of the implemented remote hierarchical supervision system. At the first stage, the *Remote Manager* sends a hypertext transfer protocol (HTTP) request to the *Local Controller*. Next, the *Local Controller* sends an HTTP response with an authentication Web page, on which the *Remote Manager* can login to the system by sending a request with user/password. The *Local Controller* 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 Java servlet's response 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 is called by the Java applet and run to enable/disable associated control buttons on the HMI according to the current system status so as to meet the required specifications. Thus, the *Remote Manager* can send a task command by pushing an enabled button to control the remote process through the constructed socket.

D. Java Implementation

In this paper, we have employed the Java servlet for authentication and Java applet for graphical HMI. A Java servlet [34] is a compiled code, dynamically loaded to process requests from a

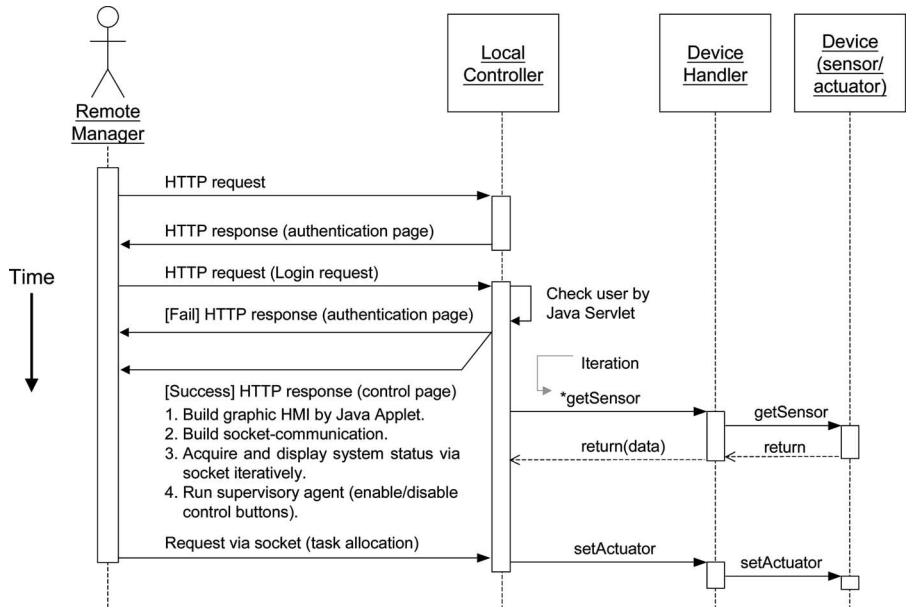

Fig. 4. Interactive modeling with sequence diagram for the implemented hierarchical supervision system.

Web server. It does not depend on browser compatibility due to running on the server side. Moreover, a Java server page (JSP) is a script and it is compiled into Java servlets during its first invocation and may call JavaBeans to perform processing on the server. A JavaBean is a portable, platform-independent component model, developed in collaboration with industry leaders. Since JSP with JavaBean requires the script translation, Java servlet has been selected for implementation due to its faster performance and easier debugging. On the other hand, a Java applet is a widely used program that can be embedded in a Web page. When you use a Java-enabled browser to view a page that contains an applet, the applet's code is transferred to your system and executed by the browser's Java virtual machine (JVM). This paper has adopted the Java applet for graphical HMI due to its plentiful availability of application programming interfaces (API) [35]. Also, most Web browsers (Navigator or Internet Explorer) provide the JVM to support Java applets. Moreover, as shown in Fig. 4, the TCP socket communication is used for data transmission due to its easier implementation. For distributed application development, the Java remote method invocation (RMI) or interface definition language (IDL) can be further applied [34].

IV. Remote Hierarchical Supervision of an FMS

A. Description of the Flexible Manufacturing System

Fig. 5 shows the remote-controlled FMS, which is composed of: 1) three processing machines; 2) three raw material suppliers; and 3) six automated conveyers. It is assumed that the raw materials are provided infinitely. The FMS corresponding to different products are specified in terms of recipes, i.e., the sequences of tasks to be carried out on discrete amounts of materials by employing all or part of the machines. This particular

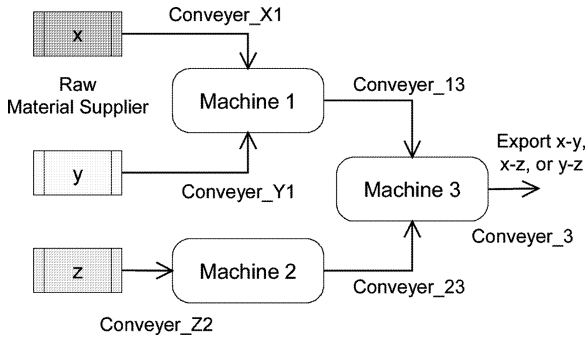

Fig. 5. Schematic diagram of the three-recipe FMS.

FMS has three recipes for three different products described as follows:

Recipe 1 (*Product x-y*): Load materials x and y to Machine 1 (M1) for processing. Then, convey x-y to Machine 3 (M3). After processing x-y in M3, unload the product.

Recipe 2 (*Product x-z*): Load materials x to M1 and z to Machine 2 (M2) for processing, and then convey x and z to M3. After processing x-z in M3, unload the product.

Recipe 3 (*Product y-z*): Load materials y to M1 and z to M2 for processing, and then convey y and z to M3. After processing y-z in M3, unload the product.

By applying the task-oriented concept, the PN model for the three recipes is constructed as shown in Fig. 6, which consists of 19 places and 22 transitions. Transitions drawn with dark symbols are events that are controllable by remote managers via the Internet. Corresponding notation is described in Table I.

B. Design of Supervisor to Meet Command-Level Specifications

The three machines represent resources shared between the different recipes. Since more than one recipe may require access

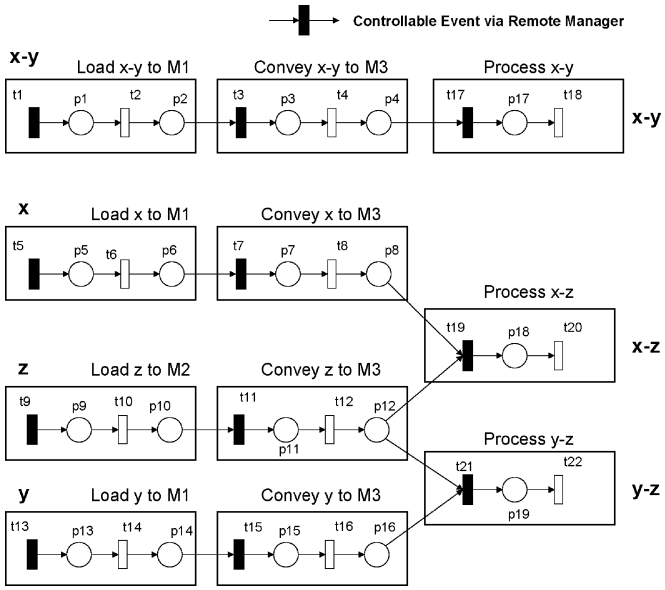

Fig. 6. Preliminary PN model of the three-recipe FMS.

TABLE I
NOTATION FOR THE PN OF THE FMS IN FIG. 6

Place	Description	Transition	Description
p1	Loading x-y to M1	t1	Cmd: start loading x-y to M1
p2	Loading x-y to M1 completed	t2	Re: end loading x-y to M1
p3	Conveying x-y to M3	t3	Cmd: start conveying x-y to M3
p4	Conveying x-y to M3 completed	t4	Re: end conveying x-y to M3
p5	Loading x to M1	t5	Cmd: start loading x to M1
p6	Loading x to M1 completed	t6	Re: end loading x to M1
p7	Conveying x to M3	t7	Cmd: start conveying x to M3
p8	Conveying x to M3 completed	t8	Re: end conveying x to M3
p9	Loading z to M2	t9	Cmd: start loading z to M2
p10	Loading z to M2 completed	t10	Re: end loading z to M2
p11	Conveying z to M3	t11	Cmd: start conveying z to M3
p12	Conveying z to M3 completed	t12	Re: end conveying z to M3
p13	Loading y to M1	t13	Cmd: start loading y to M1
p14	Loading y to M1 completed	t14	Re: end loading y to M1
p15	Conveying y to M3	t15	Cmd: start conveying y to M3
p16	Conveying y to M3 completed	t16	Re: end conveying y to M3
p17	Processing x-y in M3	t17	Cmd: start processing x-y
p18	Processing x-z in M3	t18	Re: end processing x-y
p19	Processing y-z in M3	t19	Cmd: start processing x-z
		t20	Re: end processing x-z
		t21	Cmd: start processing y-z
		t22	Re: end processing y-z

to the same resource, but each resource can only serve one recipe at a time, deadlock between different recipes may, thus, occur. The required specifications are as follows.

Spec-1: Raw material loading of x and y is allowed only when M1 is available.

Spec-2: Raw material loading of z is allowed only when M2 is available.

Spec-3: Material conveying to M3 is allowed only when M3 is available.

Spec-4: Liveness, i.e., no deadlock states, must be enforced throughout system operation.

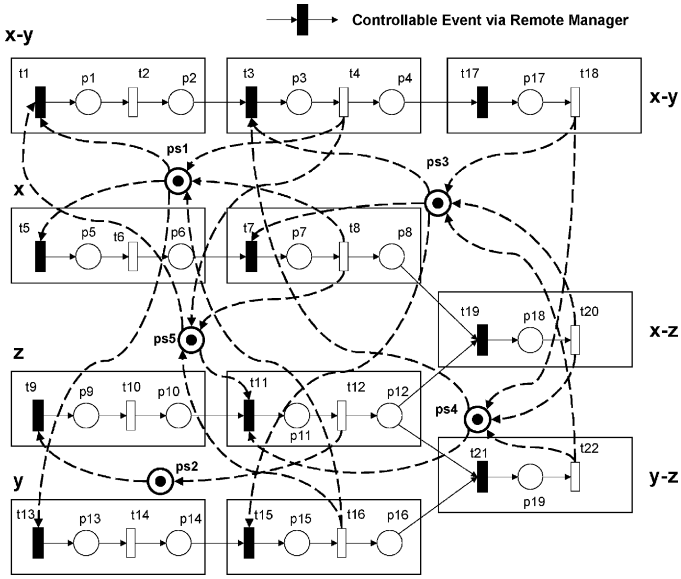

Fig. 7. Composed PN model of the three-recipe FMS.

TABLE II
NOTATION FOR THE SUERVISORY PLACES OF THE PN IN FIG. 7

Place	Description
ps1	Spec-1: M1 is available for x-y, x, or y.
ps2	Spec-2: M2 is available for z.
ps3	Spec-3: M3 is available for x-y, x, or y.
ps4	Spec-3: M3 is available for x-y, or z.
ps5	Spec-4: One token means x-y is not in M1 and
(2-bound)	z is not in M3. Another means x or y is in M3.

In the specification model, Spec-1 and Spec-3 are built by using the mutual exclusion concept, while Spec-2 is modeled as the precondition of the associated tasks. The composed PN model of both the recipe and specifications is shown in Fig. 7. The supervisory arcs are shown with dashed lines and the places showing the supervisory positions are drawn thicker than those showing the task positions. The supervisory places **ps1-4** (**ps1** for Spec-1, **ps2** for Spec-2, **ps3-4** for Spec-3) are used to prevent the remote manager from issuing undesired commands leading to resource conflicts on the part of the system. Corresponding notation for the supervisory places is described in Table II.

At this stage, due to its ease of manipulation, support for graphics import, and ability to perform structural and performance analyses, the software package ARP [36] is chosen to verify the behavioral properties of the composed PN model using the reachability analysis. The validation result (without **ps5**) shows that one deadlock occurs with the places p2, p10, p12, and **ps3** marked only. The physical meaning of the deadlock state is that if both M2 and M3 are occupied with z for the product x-z or y-z, while M1 is loaded for the product x-y, then no product can be completed and the system is deadlocked. Hence, for Spec-4, the **ps5** is further designed and added to the PN model, as shown in Fig. 7. Validation results (with **ps5**) reveal that the present PN model is live, bounded, and reversible. The liveness property means that the system can be executed properly without deadlocks, boundedness indicates that the system can be executed with limited resources, and reversibility implies

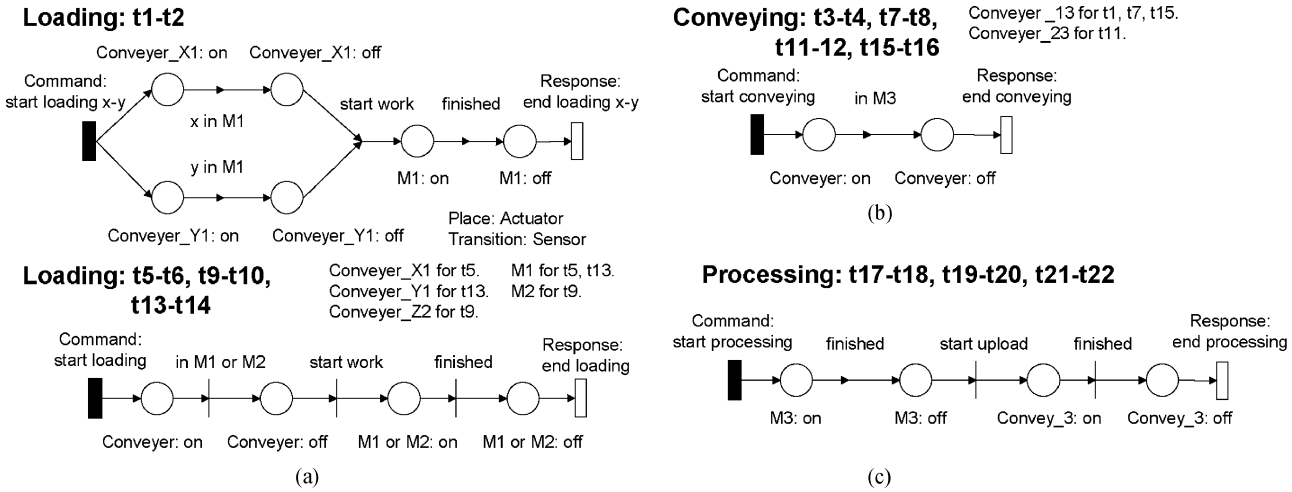

Fig. 8. PN models of (a) loading, (b) conveying, and (c) processing tasks for FMS.

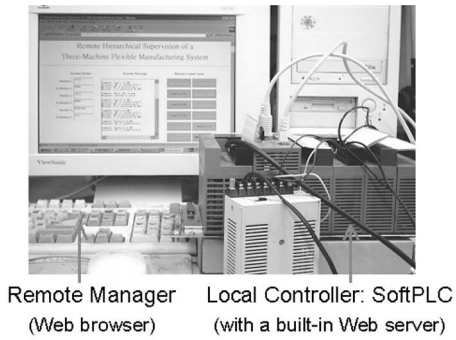

Fig. 9. Hardware setup during prototype development.

that the initial system configuration is always reachable. In this approach, the supervisor consists only of places and arcs, and its size is proportional to the number of specifications that must be satisfied.

C. Design of Local Controller to Meet Control-Level Specifications

As mentioned in Section II-D, the detailed operations of each task can also be designed and constructed with PN models. Fig. 8(a)–(c) shows the PN model of the tasks loading (from raw material supplier to M1 or M2 with processing), conveying (from M1 or M2 to M3), and processing (processed by M3 and unloaded), respectively.

D. Implementation of Remote Hierarchical Supervision

The system modeling and design developed in previous stages provide supervisory and control models for implementation of the present remote hierarchical supervision. The developed local controller and supervisory agent are implemented on the Mirle SoftPLC (80486-100 CPU), an advanced industrial PLC with built-in Web server and JVM so that it can interpret the LLD, HTTP requests, and Java programs [25], [26], as shown in Fig. 9.

The developed HMI, shown in Fig. 10, 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 efficient information delivery instead of flashy graphics [37]. 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. By pushing the enabled buttons, the remote manager can issue commands to start tasks operated by the local controller.

Fig. 10 also shows that M1 is available, and both M2 and M3 are occupied with the material z (the prestate of the mentioned deadlock in Section IV-B). In this situation, the button **Load X to M1** or **Load Y to M1** is enabled to meet Spec-1, while the **Load X-Y to M1** button is disabled by the supervisory agent to satisfy Spec-4, and the other buttons are disabled to meet Spec-2, Spec-3, and recipe specifications. The remote manager can only push the button **Load X to M1** or **Load Y to M1** to generate the product x-z or y-z, respectively. Thus, the desired requirements of the three-recipe FMS are guaranteed as the commands issued by the remote human manager are conducted.

E. Discussion

In the proposed hierarchical framework, the supervisor turns out to be more compact and simple, since it deals only with the command-level tasks, i.e., groups of operations. This greatly simplifies analysis and validation of the supervisor. The implementation of several elementary operations can be grouped into a single task performed by the local controllers. Separation of detailed control and supervision enables us to increase the conciseness of our design problem and makes the complexity manageable.

By comparison, as shown in Table III, using a conventional nonhierarchical approach [8] to the present three-recipe FMS, verification of the supervisor has to resolve all deadlock situations by searching the whole reachability graph, with the detailed control-level operations in a 2228-state space. However, by applying the proposed hierarchical framework, the

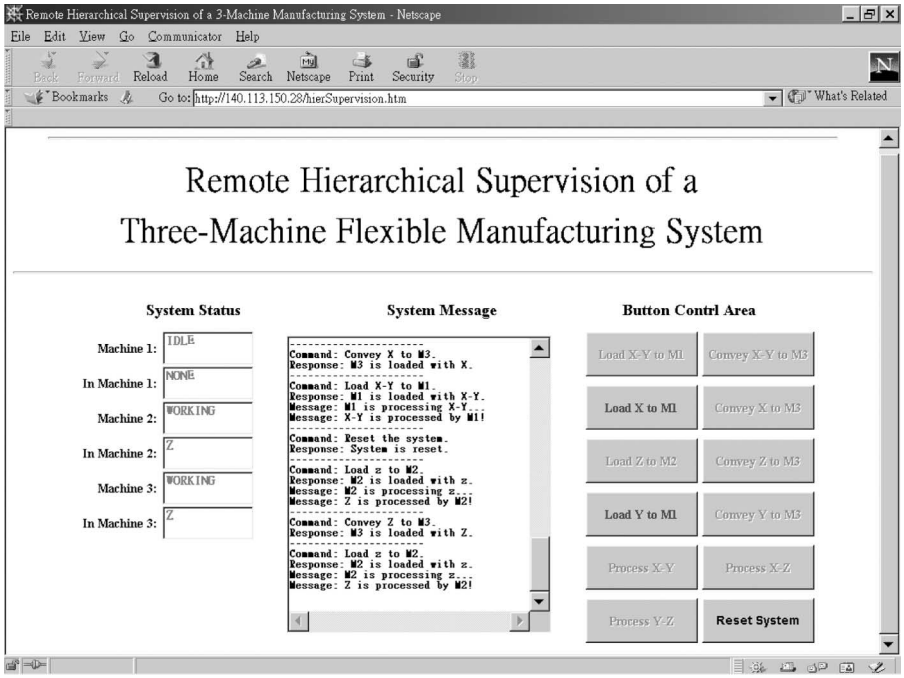

Fig. 10. Interactive Web page for remote supervision of the FMS by a Java applet (only three buttons are admissible).

TABLE III
COMPARISON BETWEEN THE NONHIERARCHICAL AND HIERARCHICAL APPROACHES

Index	Conventional nonhierarchical approach	Proposed hierarchical framework
Places	50	23
Transitions	48	22
State space	2228	248
Req/Resp transmissions for 30 products (10 each)	560	260

supervisor design has a more compact model with a 248-state space.

Moreover, to produce 30 products (ten x-y, x-z, y-z each), 560 request/response transmissions over the Internet are consumed in the nonhierarchical approach, while only 260 ones are required using the proposed hierarchical scheme.

V. CONCLUSION

This paper has presented a unified PN framework to design and implement a hierarchical supervisory system for remote-controlled processes. The supervisor is systematically synthesized using PNs to enforce the command-level specifications of resource constraints and liveness for the processes, and then is implemented with agent technology. The local controller in the lower level is also designed with PNs to meet the control-level specifications and is implemented by the LLD. To illustrate the proposed approach, an application to a three-recipe FMS controlled over the Internet is provided. According to the feedback status of the remotely located system, the designed Java-based supervisory agent guarantees that all requested commands from the human manager satisfy the requirements for multiple recipes, resource sharing, and deadlock avoidance, while the developed local controller performs the corresponding operations to meet the requested tasks. Moreover, results show that the supervisor synthesis of the presented hierarchical scheme is less complex than the conventional nonhierarchical one, and fewer packet transmissions are consumed so that the effects of time delays and packet losses across the Internet could be moderated.

Since the original supervisory control framework [10]–[14] is restricted to purely logical system models, for applications with time-based requirements (e.g., transmission delays), it is necessary to extend the present models with time specifications. Moreover, novel specifications for the error recovery due to the packet losses should be investigated in the future.

REFERENCES

[1] A. Weaver, J. Luo, and X. Zhang, "Monitoring and control using the Internet and Java," in *Proc. IEEE Int. Conf. Ind. Electron.*, San Jose, CA, 1999, pp. 1152–1158.
[2] A. Rovetta, R. Sala, X. Wen, and A. Togno, "Remote control in telerobotic surgery," *IEEE Trans. Syst., Man, Cybern. A*, vol. 26, no. 4, pp. 438–444, Jul. 1996.
[3] G. Q. Huang and K. L. Mak, "Web-integrated manufacturing: Recent developments and emerging issues," *Int. J. Comput. Integr. Manuf.*, vol. 14, no. 1, pp. 3–13, 2001.
[4] J. S. Lee and P. L. Hsu, "Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets," *IEEE Trans. Control Syst. Tech.*, vol. 12, no. 2, pp. 293–302, Mar. 2004.
[5] X. Feng, S. A. Velinsky, and D. Hong, "Integrating embedded PC and Internet technologies for real-time control and imaging," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 1, pp. 52–60, Mar. 2002.
[6] C. Batur, Q. Ma, K. Larson, and N. Kettenbauer, "Remote tuning of a PID position controller via Internet," in *Proc. Amer. Control Conf.*, 2000, pp. 4403–4406.
[7] J. S. Lee, M. C. Zhou, and P. L. Hsu, "An application of Petri nets to supervisory control for human-computer interactive systems," *IEEE Trans. Ind. Electron.*, vol. 52, no. 5, pp. 1220–1226, Oct. 2005.

[8] R. L. Kress, W. R. Hamel, P. Murray, and K. Bills, "Control strategies for teleoperated Internet assembly," *IEEE/ASME Trans. Mechatronics*, vol. 6, no. 4, pp. 410–416, 2001.
[9] J. Rasmussen, A. M. Pejtersen, and L. P. Goodstein, *Cognitive Systems Engineering*. New York: Wiley, 1994.
[10] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.
[11] ——, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.
[12] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin, "Supervisory control of a rapid thermal multiprocessor," *IEEE Trans. Autom. Control*, vol. 38, no. 7, pp. 1040–1059, Jul. 1993.
[13] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Autom. Control*, vol. 35, no. 10, pp. 1125–1134, Oct. 1990.
[14] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynam. Syst. Theory Appl.*, vol. 6, pp. 241–273, 1996.
[15] M. Tittus and B. Lennartson, "Hierarchical supervisory control for batch processes," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 5, pp. 542–554, Sep. 1999.
[16] F. Charbonnier, H. Alla, and R. David, "The supervised control of discrete-event dynamic systems," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 2, pp. 175–187, Mar. 1999.
[17] S. B. Gershwin, "Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems," *Proc. IEEE*, vol. 77, no. 1, pp. 195–208, Jan. 1989.
[18] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems Using Petri Nets*. Boston, MA: Kluwer, 1998.
[19] A. Giua and F. DiCesare, "Supervisory design using Petri nets," in *Proc. IEEE Int. Conf. Decis. Control,* Brighton, U.K., 1991, vol. 1, pp. 92–97.
[20] R. Zurawski and M. C. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 567–583, 1994.
[21] M. Uzam and A. H. Jones, "Discrete event control system design using automation Petri nets and their ladder diagram implementation," *Int. J. Adv. Manuf. Technol.*, vol. 14, no. 10, pp. 716–728, 1998.
[22] M. C. Zhou and M. D. Jeng, "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, Aug. 1998.
[23] J. S. Lee and P. L. Hsu, "A systematic approach for the sequence controller design in manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 25, no. 7–8, pp. 754–760, Apr. 2005.
[24] M. P. Fanti, B. Maione, and T. Turchiano, "Comparing diagraph and Petri net approaches to deadlock avoidance in FMS modeling and performance analysis," *IEEE Trans. Syst., Man, Cybern. Part B*, vol. 30, no. 5, pp. 783–798, Oct. 2000.
[25] *SoftPLC Controller User's Manual Version 1.2,* Mirle Automation Corporation, Hsinchu, Taiwan, 1999.
[26] *SoftPLC-Java Programmer's Toolkit,* SoftPLC Corp., Spicewood, TX, 1999.
[27] M. C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling for manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 7, no. 4, pp. 515–527, Aug. 1991.
[28] J. M. Bradshaw, "Introduction to Software Agents," in *Software Agents*, J. M. Bradshaw, Ed. AAAI Press/MIT Press, MA, 1997.
[29] M. Wooldridge and M. R. Jenkins, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.
[30] T. Hoshi, "Current and future Java technology for manufacturing industry," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.,* Tokyo, Japan, Oct. 12–15, 1999, vol. 6, pp. 404–409.
[31] E. Bertolissi and C. Preece, "Java in real-time applications," *IEEE Trans. Nucl. Sci.*, vol. 45, no. 4, pt. 1, pp. 1965–1972, Aug. 1998.
[32] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall, 1991.
[33] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley, 1999.
[34] J. Hunter and W. Crawford, *Java Servlet Programming*. Sebastopol, CA: O'Reilly, 1998.
[35] M. Campione and K. Walrath (1998) *The Java Tutorial: Object-Oriented Programming for the Internet*, 2nd ed. Reading, MA: Addison-Wesley. [Online] Available: http://java.sun.com/docs/books/tutorial/
[36] C. A. Maziero, *ARP: Petri Net Analyzer*. Florianopolis, Brazil: Control Microinformatic Lab., Federal Univ. Santa Catarina, 1990.
[37] P. Shikli, "Designing winning Web sites for engineers," *Mach. Design*, vol. 69, no. 21, pp. 30–40, 1997.

Jin-Shyan Lee received the B.S. degree in mechanical engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C., in 1997, and the M.S. and Ph.D. degrees in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1999 and 2004, respectively.

During July 2003–June 2004, he was a Visiting Researcher (supported by the National Science Council of Taiwan) at the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark. Since January 2005, he has been a Researcher at the Information and Communications Research Laboratory, Industrial Technology Research Institute, Hsinchu. His current research interests include PNs, discrete event systems, supervisory control, hybrid systems, remote monitoring and control, and wireless sensor networks. His research work has led to a number of papers in journals and international conference proceedings. He was invited to speak at North New Jersey IEEE Control Systems Chapter, Newark, and University of Rome "La Sapienza," Rome, Italy.

Dr. Lee is a member of the Technical Committee on Discrete Event Systems of the IEEE Systems, Man, and Cybernetics Society. He is a recipient of the SICE International Scholarship, and a finalist in both the Annual International Award and Young Author's Award at the 2004 SICE Annual Conference, Sapporo, Japan. He organized two special tracks on 1) Wireless Sensar Networks, and 2) Petri Nets and Discrete Event Systems at the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC), and one section on Computer Automated Multi-Paradigm Modeling at the 2004 IEEE International Conference on Computer-Aided Control System Design, both in Taipei, Taiwan.

Pau-Lo Hsu (M'92) received the B.S. degree from National Cheng Kung University, Tainan, Taiwan, R.O.C., the M.S. degree from the University of Delaware, Newark, and the Ph.D. degree from the University of Wisconsin-Madison, Madison, in 1978, 1984, and 1987, respectively, all in mechanical engineering.

Following two years of military service in King-Men, he was with San-Yang (Honda) Industry during 1980–1981 and with Sandvik during 1981–1982. In 1988, he was an Assistant Professor at the Department of Electrical and Control Engineering, National Chiao Tung University, Hsinchu, Taiwan, where he became a Professor since 1995 and served as the Chairman during 1998–2000. During 2002–2003, he was the President of the Chinese Automatic control Society. His research interests include mechantronics, CNC motion control, servo systems, network-based control systems, and diagnostic systems.