

國立交通大學

電機與控制工程學系

博士論文

以 Petri Net 設計之自動化程序遠端監控系統



Design of the Remote Supervision System for
Automated Processes via the Petri Net Approach

研究生：李俊賢

指導教授：徐保羅博士

中華民國九十三年七月

以 Petri Net 設計之自動化程序遠端監控系統

Design of the Remote Supervision System for
Automated Processes via the Petri Net Approach

研究生: 李俊賢

Student: Jin-Shyan Lee

指導教授: 徐保羅 博士

Advisor: Dr. Pau-Lo Hsu

國立交通大學

電機與控制工程學系



A Dissertation

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao-Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

July 2004

Hsinchu, Taiwan, Republic of China.

中華民國九十三年七月

以 Petri Net 設計之自動化程序遠端監控系統

研究生:李俊賢

指導教授:徐保羅 博士

國立交通大學

電機與控制工程學系

摘要

近年來，由於網際網路的快速發展，使得自動化程序之即時監控與管理不再受限於局部的區域來執行。對於以網際網路為基礎的遠端製造系統，本文闡述其一系列以 Petri net 為基礎，在程序控制，遠端監控，與網路管理系統上之設計與實現的方法，以達成系統之正常安全與運作。

對於日益複雜的製造系統，傳統之階梯圖程序控制設計，不但變的相當複雜，而且對於製程變動的彈性處理也更加困難。有鑑於此，本文先提出一套以法則為基礎的評估方法，來驗證 Petri net 在程序控制器設計上優於階梯圖的特性。之後，本文提出一套以 Petri net 為基礎，而以階梯圖實現之系統化設計方法，來發展製造系統之程序控制器。

在遠端監控系統中，本文提出一個以監督器 (supervisor) 監控人類行為的架構，來預防與禁止不正當的遠端人為操作。本文使用 Petri net 來塑造命令層中真實系統的抽象模型，合成出監督器，並進一步使用 Java 技術將監督器實現成一智慧型代理人 (intelligent agent)。藉由遠端受控系統的狀態回饋，我們發展的監控代理人會防止在違反安全規格下的命令，藉以降低及減少人為失誤所產生的影響。此外，在上述的監控系統中，為了降低監控器的合成複雜度，我們提出一個階層式的遠端監控架構，使得監督器的合成需有較少的狀態空間，藉以降低設計與實現的複雜度。

此外，對於大型遠端監控系統中各種不同的感測、致動、與控制元件，為了管理網路中各元件所收發的大量監控訊息，本文整合 Petri net 於統一建模語言(unified modeling language, UML)中，以系統化地從建模，設計，分析，驗證，來實現簡易網路管理協定 (simple network management protocol, SNMP) 代理人。本研究所提出之遠端網路管理方法，已經成功地應用在台灣大哥大的行動交換機房之環境安全遠端監控系統上。



Design of the Remote Supervision System for Automated Processes via the Petri Net Approach

Student: Jin-Shyan Lee

Advisor: Dr. Pau-Lo Hsu

Department of Electrical and Control Engineering
National Chiao-Tung University

ABSTRACT

Applications of the Internet technology become more popular in the modern industry. This thesis proposes the systematic design and implementation of remote supervision systems for automated processes via the Petri nets (PN) approach to achieve 1) the sequence controller, 2) the supervisor, and 3) the device management system, respectively.

As automated systems become more complex, traditional ladder logic diagram (LLD) design of sequence controllers becomes more difficult and inflexible. Thus, this thesis presents a rule-based evaluation to adequately compare the LLD and PN, and verify the superiority of PN. Then, since LLD is still widely used today in real industry, this thesis proposes a PN-based method systematically leading to the final LLD implementation for the sequence controller design.

In remote control systems, to prevent abnormal operations of humans, a remote supervisory scheme is proposed so that undesirable human operations are prohibited. PN is employed to synthesize both the remote supervisor and the local controller, and the Java technology is employed to implement the intelligent agent for on-line supervision. According

to the status feedback through the Internet, the developed supervisory agent provides allowable commands for operators and disables those operations that violate safety specifications. The possibility of human errors can be thus reduced. Moreover, to reduce the complexity of mentioned supervisory system design, this thesis further proposes a hierarchical structure with a smaller state-space size in supervisor synthesis so as to reduce the design complexity.

Furthermore, to integrate diverse network elements and construct a large-scale and distributed systems for remote supervision systems, this thesis integrates the PN into the unified modeling language (UML) to achieve modeling, design, analysis, verification, and implementation of simple network management protocol (SNMP) agents in the present framework. The developed management system has been successfully applied to a mobile switching center of Taiwan Cellular Corporation for the remote supervision and management of its various environmental devices.



ACKNOWLEDGMENT

博士論文的完成，首先要感謝的是指導教授，徐保羅博士在課業與研究上的指導，以及生活態度上的相授，在此表達我最深誠的敬意與感謝。

謝謝 New Jersey Institute of Technology 的 Meng-Chu Zhou 教授，在我美國訪問研究一年期間 (7/1/2003-6/30/2004) 的指導與照顧。感謝 NASA 系統工程師 Peter Graube 在第二章上的建議，盟立自動化蔡宗憲博士在2001年暑期實習期間的指導與照顧。感謝口試委員：鄭芳田教授 (成功大學製造工程所)、黃漢邦教授 (台灣大學機械系)、傅立成教授 (台灣大學電機系)、鄭慕德教授 (台灣海洋大學電機系)、梁高榮教授 (本校工業工程與管理系)、以及本系胡竹生教授等師長在論文上的指導與建議。

感謝師門的學長、同學與學弟妹們，在生活及研究上的相互幫助及砥礪。工研院的沈里正、葉賜旭、王安平，中科院的呂龍騰、黃財富、與中正理工的蒙天德等學長。碩士班88同梯畢業的明潔、裕淵、永生、與建邦等同學。碩士班學弟妹們：89梯的清穩、志銘、育憲與生虎，90梯的智迪、旭原、明炫、信宏、啟信與宜霖，91梯的鎮洲、信銘、育修、正義與松德，92梯的豪聲與致成，93梯的政衍、政宏與伊婷，94梯的景文、尚玲與議寬，以及現役博士班學弟鎮洲與琮政。此外，謝謝系辦林滿足、李蜀梅、陳英芝小姐，及施德喜先生在事務上的幫忙。

感謝國科會千里馬計劃，在美國訪問研究上的支持，以及教育部在博士班期間，出席國際會議上的補助。特別謝謝志剛立川夫婦、曉峰、景功、天志、叢哲夫婦、梅梅夫婦等友人，以及NJIT台灣同學會，在我美國訪問期間生活上的照顧。

謹將此論文獻給我最敬愛的父親 李金柱先生與母親 陳玉雲女士，雅琪姐與俊德弟，大姑姑 李金枝女士及其家人，以及貼心的女友彥蓉及其家人，因為有您們的支持與關懷，我才能夠無後顧之憂地，繼續往前邁進。

感謝所有曾經幫助過我與默默祝福我的朋友，謝謝您們。

TABLE OF CONTENTS

	Page
ABSTRACT (CHINESE)	i
ABSTRACT (ENGLISH)	iii
ACKNOWLEDGMENT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
<i>CHAPTER 1</i>	
INTRODUCTION	1
1.1. General Review	2
1.2. Problem Statement	5
1.3. Proposed Approach	7
1.4. Organization of Thesis	8
<i>CHAPTER 2</i>	
EVALUATION OF LADDER LOGIC DIAGRAMS AND PETRI NETS FOR SEQUENCE CONTROLLER DESIGN	9
2.1. Introduction of Petri Nets	10
2.2. The Rule-Based Comparison	13
2.3. Example: A Stamping Process	18
2.4. Discussions	28
2.5. Summary	29



CHAPTER 3
DESIGN OF THE SEQUENCE CONTROLLER IN
MANUFACTURING SYSTEMS **30**

- 3.1. Simplified Petri Net Controller 30
- 3.2. The IDEF0/SPNC/TPL/LLD Approach 33
- 3.3. Example: A Stamping Process 41
- 3.4. Summary 42

CHAPTER 4
REMOTE SUPERVISION FOR
HUMAN-IN-THE-LOOP SYSTEMS **45**

- 4.1. A Novel Supervisory Structure 45
- 4.2. Design of the Supervisor Using PN 46
- 4.3. Implementation of the Supervisor Using Java 50
- 4.4. Example: A Rapid Thermal Process 54
- 4.5. Summary 62

CHAPTER 5
HIERARCHICAL SUPERVISION FOR
MANUFACTURING SYSTEMS **63**

- 5.1. Proposed Hierarchical Structure 63
- 5.2. Design of the Hierarchical Supervision System 65
- 5.3. Example: A Three-Recipe Flexible Manufacturing System 67
- 5.4. Discussions 74
- 5.5. Summary 75

CHAPTER 6	
SNMP-BASED MANAGEMENT SYSTEM	76
6.1. Integration of UML and PN	76
6.2. Requirements of SNMP Agents	78
6.3. UML-Based Modeling for SNMP Agents	81
6.4. Example: A Mobile Switching Center	85
6.5. PN Modeling and Analysis	89
6.6. Architecture Design and Implementation	93
6.7. Discussions	94
6.8. Summary	96
CHAPTER 7	
CONCLUSIONS	97
7.1. Summary of Contributions	97
7.2. Future Research	98
REFERENCES	101
VITA	110
PUBLICATION LIST	113



LIST OF TABLES

2.1.	Basic elements in LLD and PN.	10
2.2.	IF-THEN rules for LLD and PN.	15
2.3.	Comparison of LLD and PN for the sequence: A+, A-.	17
2.4.	IF-THEN formats of LLD and PN in Fig. 2.4-2.8.	26
3.1.	Notations for the stamping process.	44
4.1.	Notations for the PN of the RTP system in Fig. 4.6.	58
4.2.	Notations for supervisory places of PN in Fig. 4.7.	59
5.1.	Notations for the PN of the FMS in Fig. 5.3.	70
5.2.	Notations for supervisory places of the PN in Fig. 5.4.	72
5.3.	Comparison between the nonhierarchical and hierarchical schemes.	74
6.1.	Notations of the PN for the SNMP-based management system in Fig. 6.7.	92

LIST OF FIGURES

1.1.	Architecture of the proposed remote supervision system in this thesis.	2
2.1.	Basic PN models for (a) sequential, (b) concurrent, (c) cyclic, (d) conflicting, and (e) mutually exclusive relations.	13
2.2.	The LLD and PN for the sequence: A+, A-.	17
2.3.	The stamping system.	19
2.4.	LLD and PN for Sequence_1.	21
2.5.	LLD and PN for Sequence_2.	22
2.6.	LLD and PN for Sequence_3.	23
2.7.	LLD and PN for Sequence_4.	24
2.8.	LLD and PN for Sequence_5.	25
2.9.	Required basic elements in the basic element approach.	27
2.10.	Required rules and logical operators in the IF-THEN transformation.	27
2.11.	The increase ratio for the basic element approach.	28
2.12.	The increase ratio for the IF-THEN transformation.	28
3.1.	The comparison between the PN and the SPNC via a simple process. (a) Ordinary PN. (b) SPNC. (c) Comparison results.	31
3.2.	The icon definition of the SPNC.	32
3.3.	Design procedure of the IDEF0/SPNC/TPL/LLD approach.	34
3.4.	The IDEF0 scheme.	35

3.5.	The transformation from the IDEF0 to the SPNC.	37
3.6.	The transformation from the SPNC to the TPL.	38
3.7.	The transformation from the TPL to the LLD.	40
3.8.	The transformations of the IDEF0/SPNC/TPL/LLD approach.	40
3.9.	The stamping system.	41
3.10.	Design of the sequence controller using IDEF0/SPNC/TPL/LLD approach.	43
4.1.	(a) Typical remote control system with the human in the loop. (b) The proposed remote supervisory control scheme.	46
4.2.	(a) A general model for door and valve components. (b) The mutual exclusion specification model. (c) The composed supervisor for the door-valve system.	50
4.3.	Implementation architecture of the remote supervisory control system.	52
4.4.	Interactive modeling with sequence diagram for the remote supervisory control system.	53
4.5.	Schematic diagram of the RTP system.	55
4.6.	The PN model for automatic control of the RTP system.	57
4.7.	The composed PN model for manual control of the RTP system.	57
4.8.	Interactive web page for remote control of the RTP system by a Java applet (only Auto-Control button is admissible in the automatic control mode).	60
4.9.	Interactive web page in manual control mode at Step 3 of RTP processing (seven buttons are enabled).	61
4.10.	The hardware setup during prototype development.	61

5.1.	Proposed three-level architecture for hierarchical supervision.	64
5.2.	Schematic diagram of the three-recipe FMS.	68
5.3.	Preliminary PN model of the three-recipe FMS.	69
5.4.	Composed PN model of the three-recipe FMS.	71
5.5.	PN models of (a) loading, (b) conveying, and (c) processing tasks for FMS.	72
5.6.	Interactive Web page for remote supervision of the FMS by a Java applet (only three buttons are admissible).	73
6.1.	The systematic development procedure for SNMP agents.	79
6.2.	The simple network management protocol (an extension of Stallings, 1993).	80
6.3.	Functional analysis with the use-case diagram.	82
6.4.	Interaction analysis with the sequence diagrams for (a) the Request scenario and (b) the Trap scenario.	83
6.5.	The SNMP-based remote monitoring and control system.	86
6.6.	The class diagram of the SNMP-based monitoring and control system.	88
6.7.	The PNs of the SNMP-based monitoring and control system.	91
6.8.	Architectural design with the deployment diagram.	95
6.9.	The hardware setup during prototype development.	96

Chapter 1

Introduction

Recently, with the rapid development of information technology on industrial applications, remote monitoring, control, and management are critical to increase safety and flexibility of modern manufacturing processes in real operations. Some issues in e-automation are extensively discussed like: the integration of the high level message management and the fundamental layer sequence control, the effect of human errors in remote control, and the efficient message management among various devices on the networks, etc.

Generally, an automated system implements a sequence controller to regulate local processes. Also, a supervisor is required to assure normal operations, and a device management system is required to administer the various elements efficiently and flexibly. In this thesis, a remote supervision system will be developed for automated processes, as shown in Fig 1.1. The design goals of the present remote supervision system are as follows:

- 1) to develop the sequence controller to regulate the processes.
- 2) to develop the supervisor to monitor the human behaviors.
- 3) to develop the device management system to integrate diverse elements on networks.

The developed approaches in this thesis have been studied on a stamping process, a rapid thermal process in semiconductor manufacturing, a three-recipe flexible manufacturing system, and an environmental monitoring system in mobile switching centers, respectively.

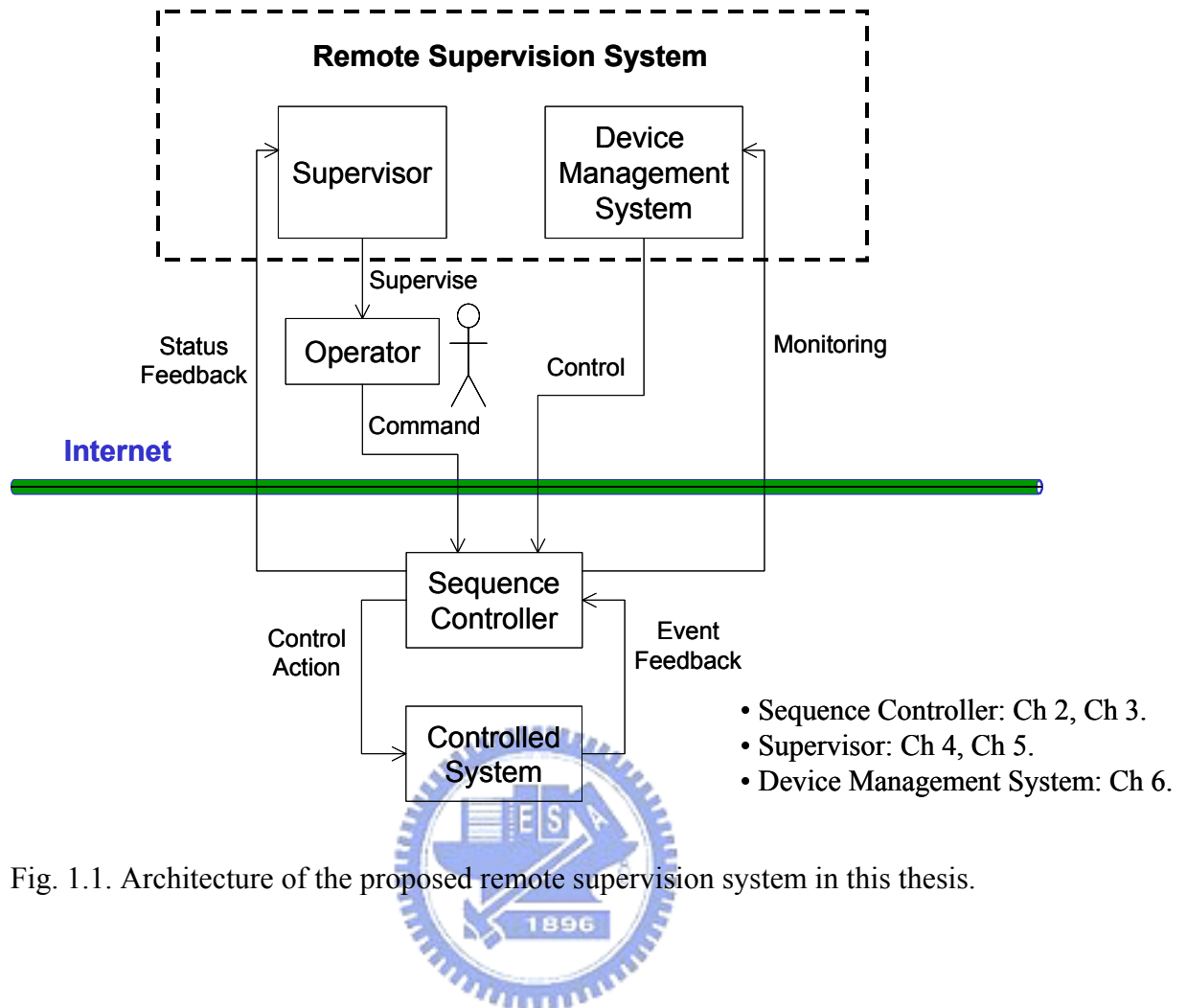


Fig. 1.1. Architecture of the proposed remote supervision system in this thesis.

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 modeling, 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 and results can be found in the survey/tutorial papers of Murata (1989), Zurawski and Zhou (1994), David and Alla (1994), and Zhou and Jeng (1998).

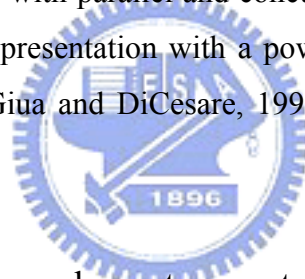
1.1.1 Systematic design of sequence controllers

A sequence controller that deals with the discrete events plays an important role in automated manufacturing systems (Tilbury and Khargonekar, 2001; Frey and Litz, 2000). Basically, the ladder logic diagram (LLD) of the industrial standard IEC1131-3 (International Electrotechnical Commission, 1993) has been widely used in real applications to conduct the control sequences and usually implemented with a programmable logic controller (PLC). The PLC has the advantages of reliability, robustness, and direct programming. The I/O procedures of the PLC are specified by the LLD and automated machines thus perform repetitive operations in sequence. For some simple controlled systems, it is easy to program the LLD with heuristic approaches. However, as systems become more complex, the controller design and the LLD implementation become even more difficult. In addition, because the LLD is usually programmed only to control the process, corresponding qualitative analysis and performance characteristics of the PLC controlled processes are seldom discussed in practice. Since product specifications are varied frequently, LLD programs of machining processes need to be modified and maintained usually with significant efforts. Hence, researchers are pursuing a systematic and efficient approach for the design and implementation of the sequence controller. Based on the PN, Liang and Hong (1994) proposed a hierarchy transformation method to design and implement controllers on a G2 expert system. Uzam and Jones (1998) introduced an extended PN method to analyze a target system and then implemented it via LLD. Feldmann, et al. (1999a, 1999b) used the colored PN to form the structured text (ST) for PLC implementation. In the past few years, the PN approach still attracted more attentions as a potential tool for designing sequence controllers.

1.1.2 Development of supervisory systems

Recently, due to the rapid development of Internet technology, system monitoring and control no longer needs to be conducted within a local area. Several 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; Huang and Mak, 2001; Batur et al., 2000). 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 behavior 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 modeling 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).



1.1.3 Management of diverse elements on networks

For large-scale and long-distance distributed systems, a reliable management system for all devices and components on the network is crucial to guarantee normal operations. It allows for reliably monitoring the status of processes, correctly detecting abnormal conditions, efficiently activating emergency mechanisms, and proactively reporting alarms. In general, the components of remote management systems can be classified into 1) the agent side and 2) the manager side. Some vendors build their web server software into their agent-side devices and the manager-side users may thus directly monitor them using web browsers through the hypertext transfer protocol (HTTP). However, as numerous devices are networked in automated manufacturing systems, the massive monitoring and control messages from all devices becomes increasingly difficult to handle. In general, straightforward integration with all Web access points is apparently not efficient. One approach to

manage diverse network elements is to use the simple network management protocol (SNMP). It is a standard protocol now widely supported by most device vendors for their products such as routers, bridges, and printers (Stallings, 1993). Aicklen and Main (1995) used SNMP to manage a variety of network elements. Cardoso and Monteiro (1998) applied the SNMP to monitor and control the industrial network. Kunes and Sauter (2001) provided a modular and extendible gateway to connect the high-level Internet and low-level fieldbus for SNMP network management.

1.2. Problem Statement

Although a lot of efforts in the past two decades have been put on the development of sequence controllers, supervision systems, and management systems for automated manufacturing processes with Internet technology, some critical issues still exist in the remote supervision system as discussed in the following:

1. Requirement of adequate evaluation for sequence controller design

Although PN has been studied to design sequence controllers with a potential in its flexibility, it is still argued that whether the PN approach is superior to the traditional LLD design for industrial practitioners. Hence, an adequate comparison is required. In the past, the “basic element” approach was developed to compare the complexity and flexibility between LLD and PN designs (Venkatesh et al., 1994a; Zhou and Twiss, 1998). However, the basic elements of these two designs are inherently different and hence, it may lead to unreliable comparison results.

2. Requirement of systematic sequence controller implementation

In practice, PLC engineers still widely prefer to use LLD for real implementation. However, it is not straightforward to construct the LLD models from a given sequence. Some researchers have attempted to transform PN into LLD (Peng and Zhou, 2001). However, those resultant LLD are usually more complex as compared to that programmed directly by engineers. A systematic approach from a given specification to achieve the final LLD implementation is

thus required.

3. Requirement of supervisory systems for human error prevention

Typically, an Internet-based control system is a “human-in-the-loop” system. The human operator is involved in the loop and use a general web browser or specific software to monitor and control remotely located systems according to the observed status, usually displayed by the state and/or image feedback. However, human operators may send incorrect or improper commands during the operation and 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). Therefore, solutions to reduce or eliminate the possibility of human errors are required in Internet-based control systems.

4. Requirement of reducing the complexity of supervisor synthesis

PN can represent the remote control system with a more compact model. However, during the synthesis of the supervisor, the complexity exponentially increases in the state-space size of the subsystems and specifications. This computational expense often makes the supervisor synthesis infeasible, especially for large-scale manufacturing systems.

5. Requirement of management for different networked devices

To design a remote monitoring and control structure through the network, efficient management to handle the massive information flow and represent data from different devices in a uniform format is required. Although using SNMP is a feasible approach to manage diverse network elements, in present industrial applications, many basic components such as sensors, actuators, and PLCs do not support SNMP for remote applications yet. Thus, for those without SNMP functions, a systematic approach to model and implementation SNMP function is required.

1.3. The Proposed Approach

To deal with the above problems, corresponding approaches are proposed in this thesis as follows.

1. Improved evaluation of LLD and PN

A rule-based approach for the LLD and PN evaluation via the IF-THEN transformation is proposed in this thesis. By converting both the LLD and PN into the same IF-THEN format, a unified comparison is then conducted with the same measure, which is the sum of 1) the number of IF-THEN rules, and 2) the number of logical operators, for both LLD and PN.

2. Systematic design of sequence controllers

A systematic approach to the LLD implementation of the sequence controller in manufacturing systems is introduced in this thesis. By defining the sensor state into the PN to form a simplified Petri net controller (SPNC), a more compact LLD structure through the token passing logic (TPL) is obtained. Typically, the sensor state is used to trigger sequences in manufacturing. The integration definition language 0 (IDEF0) can be used to obtain the SPNC model through the material flow diagrams and information flow diagrams in sequence. Thus, the proposed IDEF0/SPNC/TPL/LLD approach, including the IDEF0, SPNC, and TPL tools, leads to the LLD for general PLC implementation.

3. Supervisory control of human behaviors

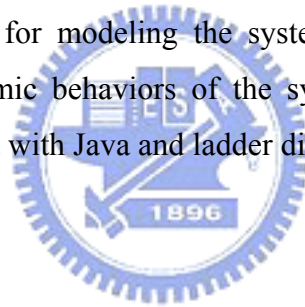
In this thesis, a supervisory scheme is proposed for the remotely controlled, human-in-the-loop system. The role of a supervisory agent is to interact with the human operator and the controlled system so that the closed human-in-the-loop system meets the required specifications. In the supervision system, the supervisory agent acquires the system status and makes the decision to enable/disable associated events to meet the required safety specifications. The human operator is then only allowed to perform the enabled events to control the system, and hence, the supervisory agent guarantees that undesirable manually executions never occur.

4. Hierarchical supervision of processes

In the present design of supervisory systems, PN can be used to design both the supervisor at the upper level and the local controller at the lower level. This thesis proposes a hierarchical supervision system resulting in a smaller state-space size through the supervisory synthesis. The proposed design guarantees that remote commands meet resource constraints and deadlock-free specifications. Also, fewer request/response transmissions are required for Internet communication. As a result, the effects of time delays and packet losses could be moderated.

5. Modeling and implementation of SNMP agents

A new approach to the development of SNMP agents for managing diverse network elements in manufacturing processes is proposed. The unified modeling language (UML) is adopted for modeling the system, and then the PN model is applied to analyze the dynamic behaviors of the system. In real applications, the present design is implemented with Java and ladder diagrams on the industrial PLC.



1.4. Organization of Thesis

This thesis is organized as that: the improved evaluation of LLD and PN is presented in Chapter 2. Then, Chapter 3 introduces the IDEF0/SPNC/TPL/LLD approach for the sequence controller design. The basic supervisory control scheme for the remote-controlled processes is proposed in Chapter 4, and Chapter 5 extends it to a hierarchical scheme. For device management, Chapter 6 proposes an integrated approach including UML modeling and PN analysis to develop the SNMP agents. Finally, conclusions and recommendations for further research are provided in Chapter 7.


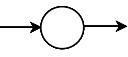









Chapter 2

Evaluation of Ladder Logic Diagrams and Petri Nets for Sequence Controller Design

Sequence controller designs play a key role in advanced manufacturing systems. Traditionally, the ladder logic diagram (LLD) has been widely applied to programmable logic controllers (PLC), while recently the Petri net (PN) has emerged as an alternative tool for the sequence control of complex systems. The evaluation of both approaches has become crucial and has thus attracted attention.

Practically, only a limited amount of research comparing these approaches has been reported, because suitable comparison criteria are difficult to identify. Boucher et al. (1990) studied the sequence control of a manufacturing system and reported that using PN makes the controller more tractable than using LLD. However, they have not formally quantified the comparison between LLD and PN to design sequence controllers. Venkatesh et al. (1994a, 1994b) proposed the number of “basic elements”, which are nodes and links in the LLD and PN, as a quantified measure to compare their design complexity and response time. They claimed that PN offers a better solution than LLD, especially in adaptability as specifications change. Based on the basic element approach, Zhou and Twiss (1995, 1998) further compared the LLD and PN in terms of the understandability, flexibility and the ability to perform correctness verification. They also reported that the PN displays better results. However, note that while basic elements in the LLD stand for push buttons, limited switches, relay coils, timers, counters, solenoids and lines, they are places, transitions and arcs in the PN. Since both nodes and links in the LLD and PN have different physical meaning, as shown in Table 2.1, analysis of LLDs and PNs simply by using the number of basic elements as the comparison measure may lead to an incoherent comparison.

Table 2.1. Basic elements in LLD and PN.

Basic elements	LLD	PN		
Nodes	Push button		Place	
	Normally open contact/switch			Transition
	Normally closed contact/switch			
	Relay coil			
	Timer			
	Counter			
	Solenoid			
Links	Line		Normal arc	
			Inhibitory arc	

In this chapter, an improved approach towards evaluating the LLD and PN methods is proposed via the IF-THEN transformation. By converting both the LLD and PN into the same IF-THEN formats (Looney and Alfize, 1987), a unified comparison is then achieved based on the same measure, which is the sum of 1) the number of IF-THEN rules, and 2) the number of logical operators for both LLD and PN. An example of five sequences with increasing complexity for a stamping process is provided to illustrate the proposed approach. We find that the proposed evaluation approach yields more reasonable results. Also, the realistic comparisons provided in this chapter support the superiority of the PN approach.

2.1. Introduction of Petri Nets

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

$$G = (P, T, I, O), \quad (2.1)$$

where,

$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places, where $m > 0$;

$T = \{t_1, t_2, \dots, t_n\}$ 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 $N = \{0, 1, 2, \dots\}$;

$O: T \times P \rightarrow N$ 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).

2.1.1 Elementary PN Models

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. 2.1 (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. 2.1 (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. 2.1 (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 Fig. 2.1 (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. 2.1 (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.

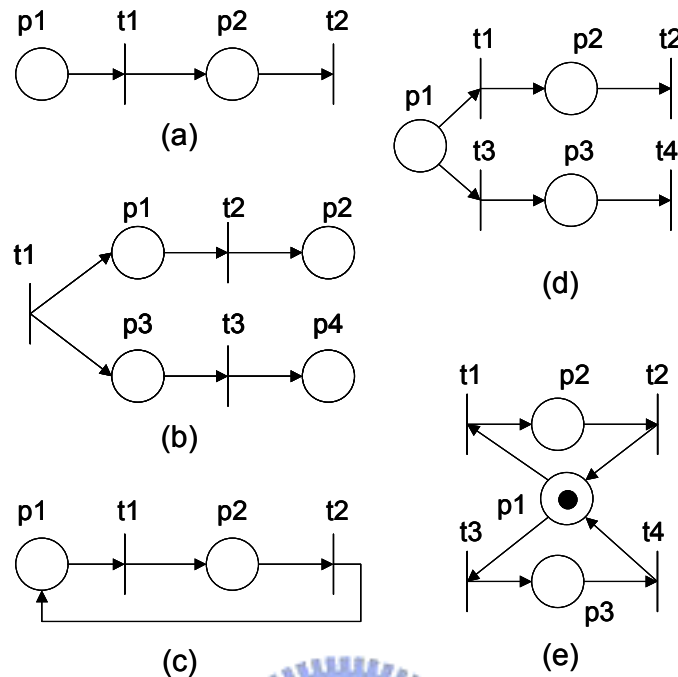


Fig. 2.1. Basic PN models for (a) sequential, (b) concurrent, (c) cyclic, (d) conflicting, and (e) mutually exclusive relations.

2.2. The Rule-Based Comparison

Two of major factors for comparison of LLD and PN for sequence control are identified as design complexity and response time (Venkatesh et al., 1994a). Design complexity is defined as the complexity associated in designing the control logic for a given specification. Response time is termed as the scan time in LLD or the execution time in PN. The major factor for design complexity is the physical size of the control logic model, whereas the response time is influenced by not only the physical size, but also the hardware of implementation. For simplicity, this chapter focuses on the comparison of the control logic models. The proposed approach includes two steps as follows:

Step 1) Transform both the LLD and PN into the same IF-THEN format.

Step 2) Evaluate the LLD and PN based on the number of a) rules and b) logical

operators.

In general, control models use smaller number of IF-THEN rules and logical operators are easier to understand, debug, check and maintain. Moreover, they may have a shorter response time. Thus, the proposed approach based on the unified rule-based format to compare the corresponding design complexity and response time for different LLD and PN structures.

2.2.1 IF-THEN Formats

Basically, compound IF-THEN rules, which involve both the conjunctive and disjunctive connectives in their antecedent or conclusion part, can be categorized into the following basic four types (Looney and Alfize, 1987).

Type 1: IF (A and B) THEN C, or expressed as $(A \cap B) \rightarrow C$,

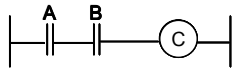
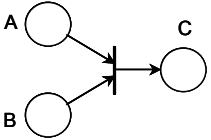
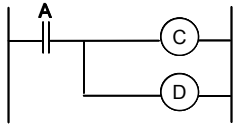
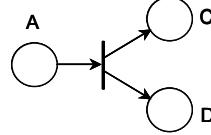
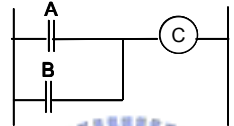
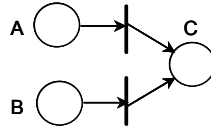
Type 2: IF A THEN (C and D), or expressed as $A \rightarrow (C \cap D)$,

Type 3: IF (A or B) THEN C, or expressed as $(A \cup B) \rightarrow C$,

Type 4: IF A THEN (C or D), or expressed as $A \rightarrow (C \cup D)$.

The Type 2 rule can be broken into two simple rules $A \rightarrow C$ and $A \rightarrow D$. Similarly, the Type 3 rule is equivalent to the two simple rules $A \rightarrow C$ and $B \rightarrow C$ because the truth of either A or B (or both) implies the truth of C. In practice, since the Type 4 rule does not achieve the specific implication and often causes conflict problems, it is generally not suitable for real applications in the sequence control. The IF-THEN rules excluding Type 4 for the LLD and PN transformations are shown in Table 2.2. Note that the timers and counters can also be expressed in the basic rules. For example, the condition A may represent delaying the desired time units, and the status C may express that a counter increases or decreases one unit.

Table 2.2. IF-THEN rules for LLD and PN.

IF-THEN rules	LLD	PN
IF A and B, THEN C $A \cap B \rightarrow C$		
IF A, THEN C and D $A \rightarrow C \cap D$		
IF A or B, THEN C $A \cup B \rightarrow C$		

2.2.2 Unified Comparison Measures

Based on the IF-THEN rules, two measures are proposed to evaluate PN and LLD as follows.

Measure 1: The number of IF-THEN rules.

Measure 2: The number of logical operators, including the conjunction (AND), disjunction (OR), block and implication.

The summation of Measure 1 and Measure 2 can be recognized as a new measure for evaluating different structures. By transforming both the LLD and PN to the same IF-THEN formats, comparisons with a unified measure can then be made. Basically, models use smaller number of IF-THEN rules and logical operators are easier to understand, debug, check and maintain. Moreover, they often have a shorter response time. Therefore, the sum of Measure 1 and Measure 2 properly signifies the design complexity and response time for the process represented in either LLD or PN structures.

2.2.3 A Preliminary Comparison

A simple example we use to illustrate the proposed approach is shown in Fig. 2.2, which a piston performs a forward stroke and then retracts. In this figure, the specification A+ indicates a forward stroke and A- indicates return stroke sequentially. Both the LLD and PN controllers as shown in Fig. 2.2 can be either represented by the basic elements or transformed into the same IF-THEN format, as listed in Table 2.3. Results show that the number of basic elements for the LLD and PN are 34 and 22, respectively. However, the basic elements in LLD and PN are physically different, as mentioned before, and the comparison based simply on the number of basic elements for different structures is apparently inappropriate. On the other hand, the results obtained from the IF-THEN transformation indicate that the LLD programming needs 4 IF-THEN rules and 14 logical operators, while the PN only needs 5 IF-THEN rules and 6 logic operators. Therefore, the number of IF-THEN rules and logical operators for LLD and PN is 18 and 11, respectively. Although the results of both approaches indicate that the PN offers a better solution than LLD, the present IF-THEN transformation provides more reasonable results when evaluating different structures in sequence controller design. Furthermore, the degree of programming flexibility can be analyzed by observing the increase ratio of either the number of basic elements or the number of present rules/operators as sequences become more complex.

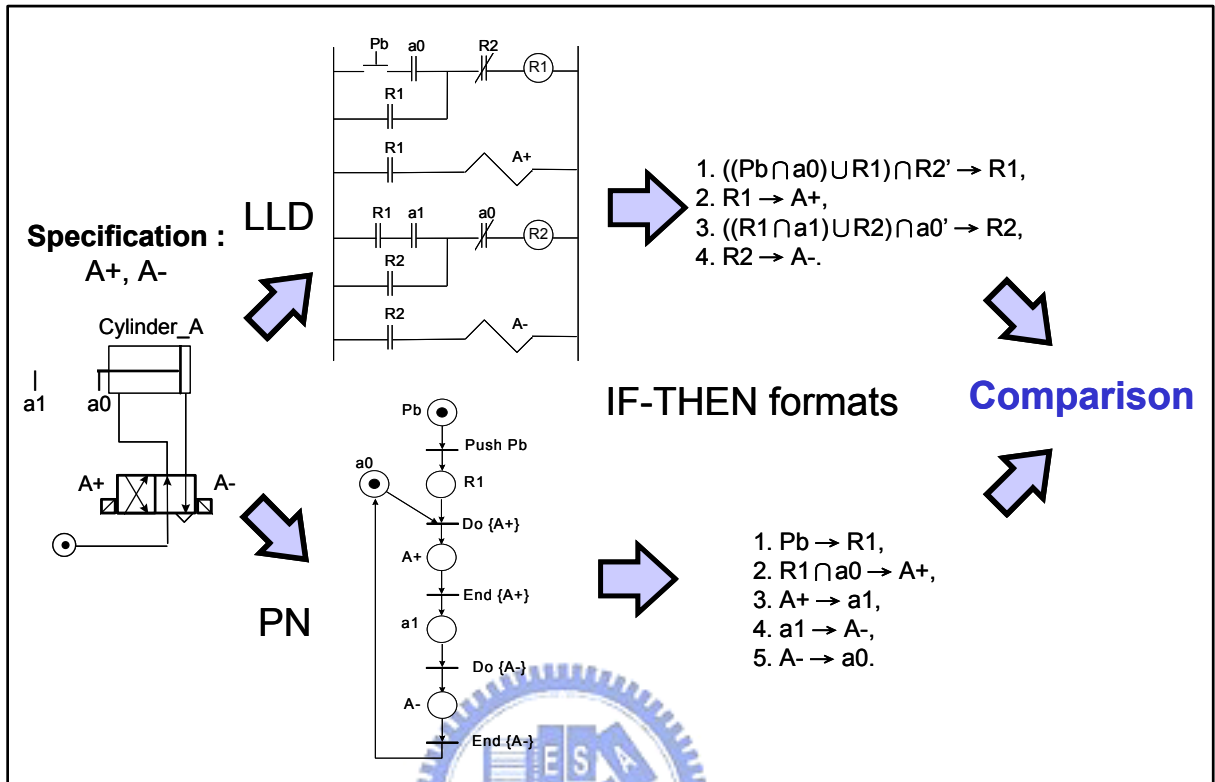


Fig. 2.2. The LLD and PN for the sequence: A+, A-.

Table 2.3. Comparison of LLD and PN for the sequence: A+, A-.

Comparison measures	LLD		PN	
Basic elements	Push button	1	Place	6
	NO contact	7	Transition	5
	NC contact	2	Normal Arc	11
	Relay	2		
	Solenoid	2		
	Line	20		
	Total	34	Total	22
IF-THEN rules	Rule	4	Rule	5
	Operator	14	Operator	6
	Total	18	Total	11

2.3. Example: A Stamping Process

To illustrate the proposed approach, we use an industrial process for automatic mark stamping and examine how the specifications change as we consider five increasingly complex sequences.

2.3.1 System Description

As shown in Fig. 2.3, a mark stamping system consists of three cylinders which are operated by four-port and two-way solenoid valves. Each cylinder has two normally open limit switches. For example, when the end of pusher_A contacts the limit switch a0, then a0 is closed, meaning that pusher_A is at the end of its return stroke. The whole system includes 7 input sensors corresponding to 6 limit switches and one push button for starting the system, and 6 output actuators corresponding to 6 solenoid valves. In the stamping process, pusher_A moves the workpiece from a store onto the worktable. Then, the workpiece is stamped by stamper_B and afterwards is ejected by thrower_C. The logical sequence of the stamping system is A+, B+, {A-, B-}, C+, and C-, where {A-, B-} represents two concurrent actions as the pistons of both pusher_A and stamper_B perform return strokes simultaneously. Five sequences with increasing complexity are considered here as follows:

Sequence_1: START, A+, B+, {A-, B-}, C+, C-

Sequence_2: START, A+, B+, 10 sec, {A-, B-}, C+, C-

(Sequence_1 with one 10-sec timer added)

Sequence_3: START, 3 [A+, B+, 10 sec, {A-, B-}, C+, C-]

(Sequence_2 with one 3-time counter added)

Sequence_4: START, 3 [A+, B+, 10 sec, {A-, B-}, C+, C-], 30 sec, 2 [A+, B+, 10 sec, {A-, B-}, C+, C-]

(Sequence_3 with one 30-sec timer and one 2-time counter added)

Sequence_5: Sequence_4 with one emergency stop added.

The complexity of these five sequences increases as specified above.

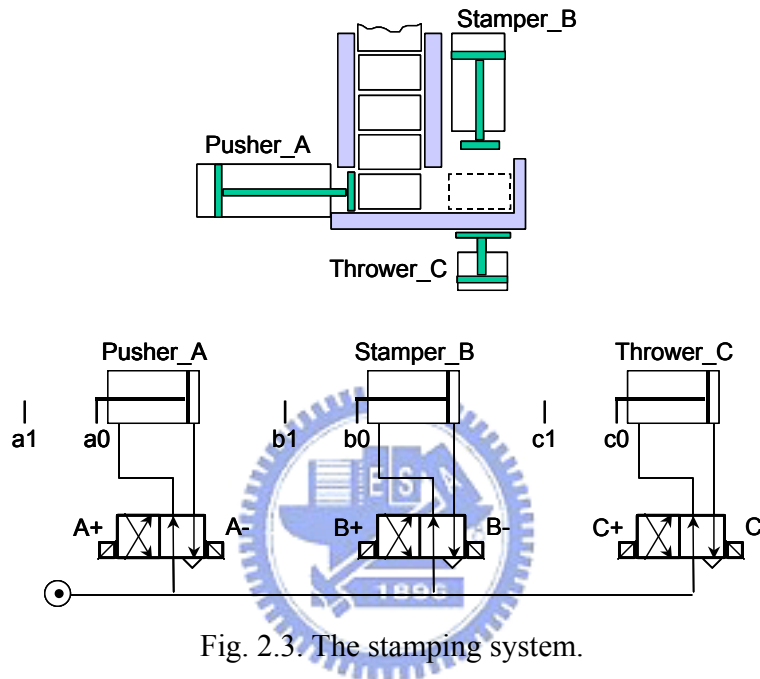


Fig. 2.3. The stamping system.

2.3.2 Sequence Controller Design

In order to solve the interlock problem, the LLD programs are usually developed with the assistance of the cascaded method which divides the required sequence into groups (Pessen, 1989). Possible contradictory solenoid signals can be thus avoided. On the other hand, since PN is a concurrent operation, it can be verified to avoid the interlock logic problem via the simulation (Zhou and Venkatesh, 1998). The LLD and PN for the Sequence_1-Sequence_5 are shown in Fig. 2.4-2.8. Although the sequences compared here only consider a typical cylinder-actuating system, similar analysis can be extended to general industrial applications such as motors, pumps, heaters and conveyors.

2.3.3 Comparison of LLD and PN

Table 2.4 shows the IF-THEN formats of the LLD and PN in Fig. 2.4-2.8. The required basic elements in the basic element approach, and the required rules and logical operators in the IF-THEN transformation for the five sequences are shown in Fig. 2.9-2.10, separately. For these five sequences, the increase ratio, which is the normalized measure based on Sequence_1 corresponding to the increasing sequence complexity, is also shown in Fig. 2.11-2.12 for the two approaches. In general, a larger ratio indicates that the design is less flexible when subjected to changes in sequence control. All results indicate that the PN is superior to LLD in terms of design simplicity, response time and flexibility responding to the specification changes.



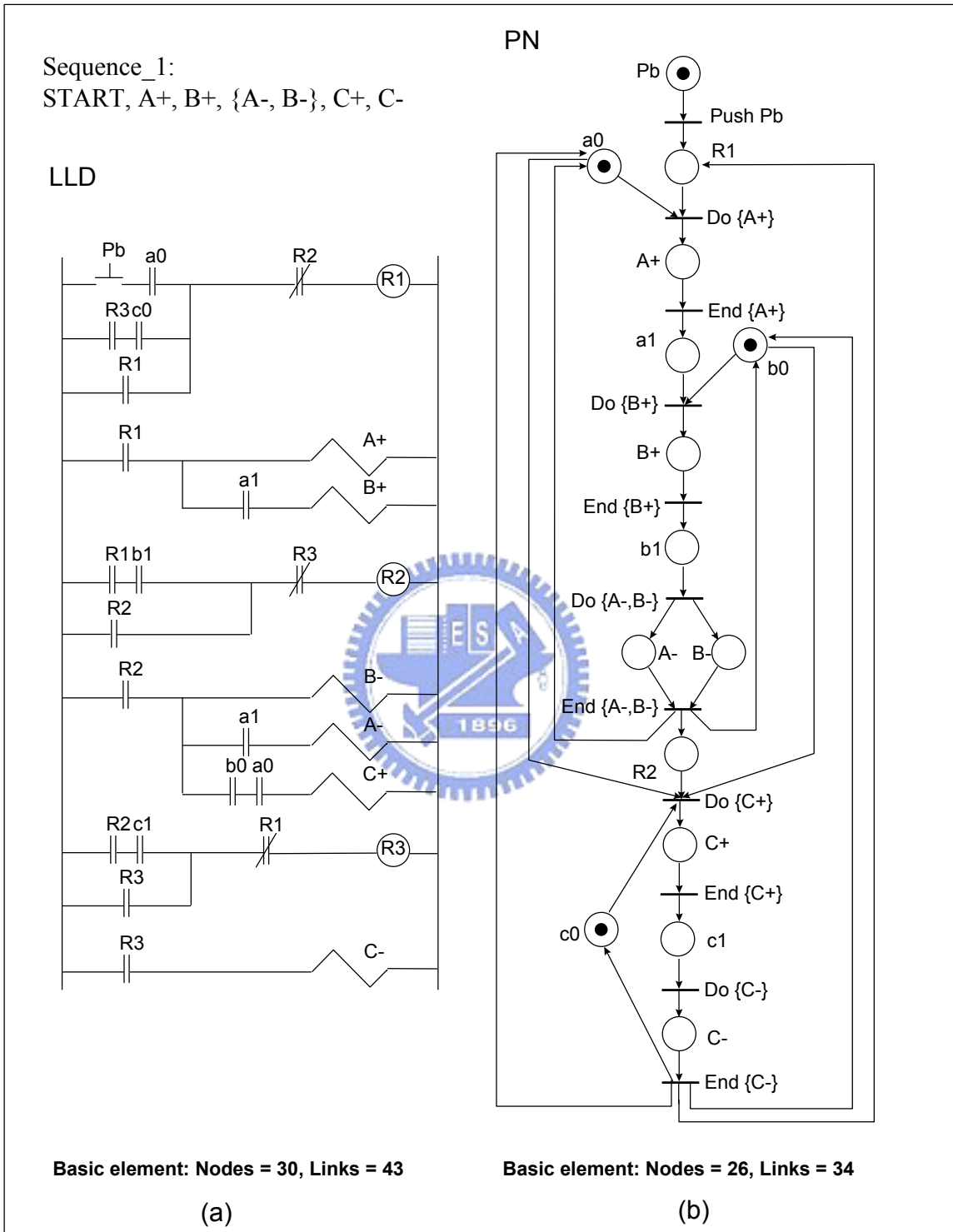
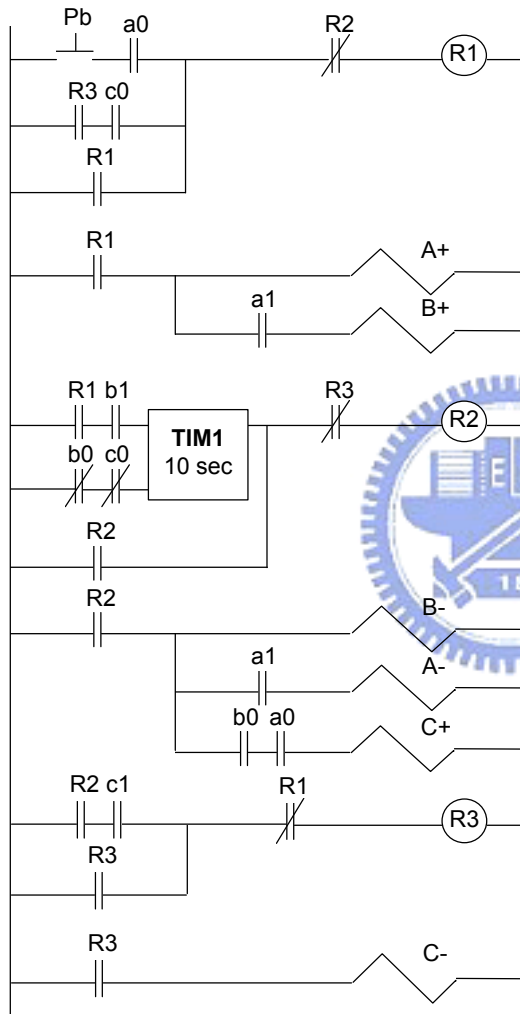


Fig. 2.4. LLD and PN for Sequence_1.

Sequence_2:
 START, A+, B+, 10 sec, {A-, B-}, C+, C-
 (Sequence_1 with one 10-sec timer added)

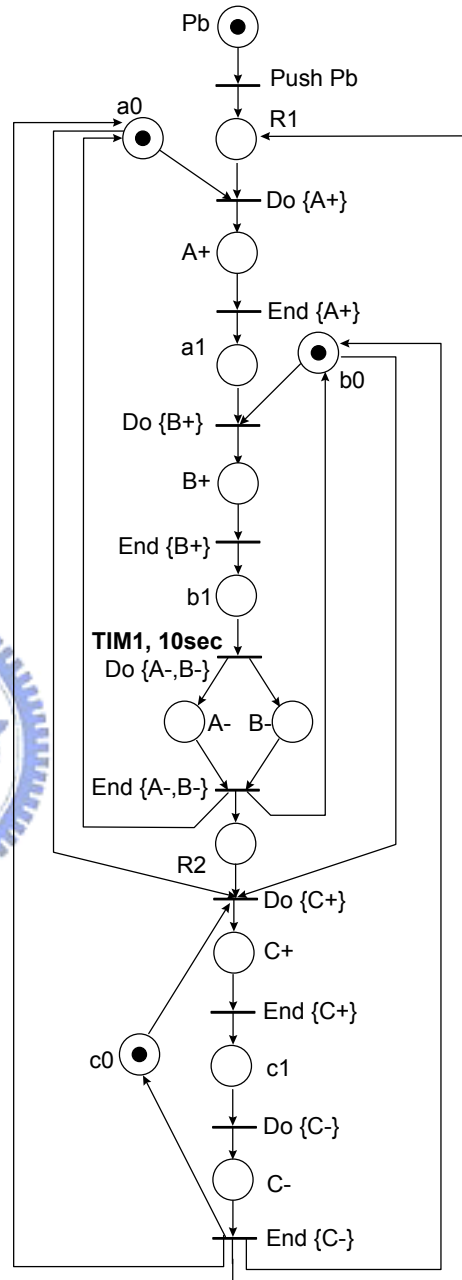
LLD



Basic element: Nodes = 33, Links = 47

(a)

PN



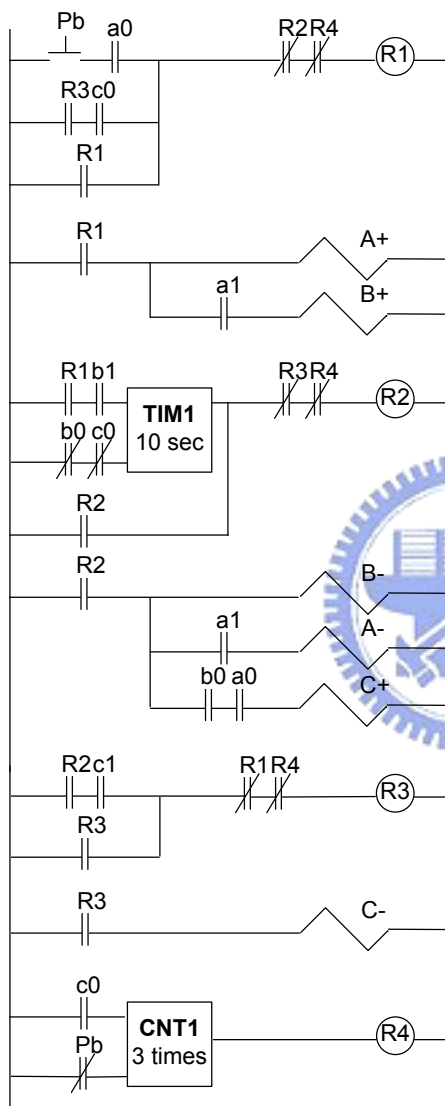
Basic element: Nodes = 26, Links = 34

(b)

Fig. 2.5. LLD and PN for Sequence_2.

Sequence_3:
 START, 3 [A+, B+, 10 sec, {A-, B-}, C+, C-]
 (Sequence_2 with one 3-time counter added)

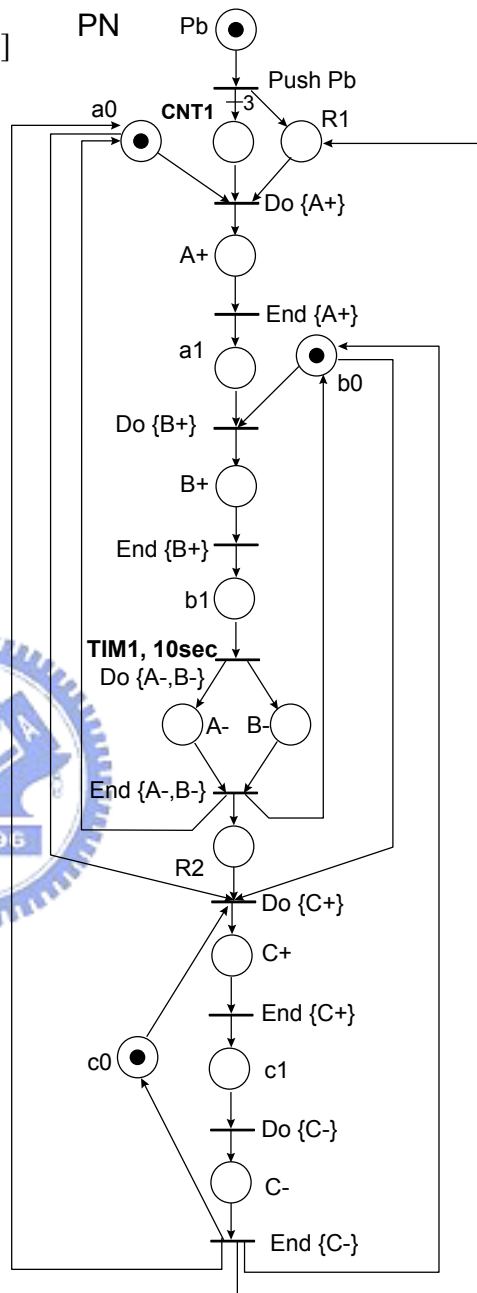
LLD



Basic element: Nodes = 40, Links = 56

(a)

PN



Basic element: Nodes = 27, Links = 36

(b)

Fig. 2.6. LLD and PN for Sequence_3.

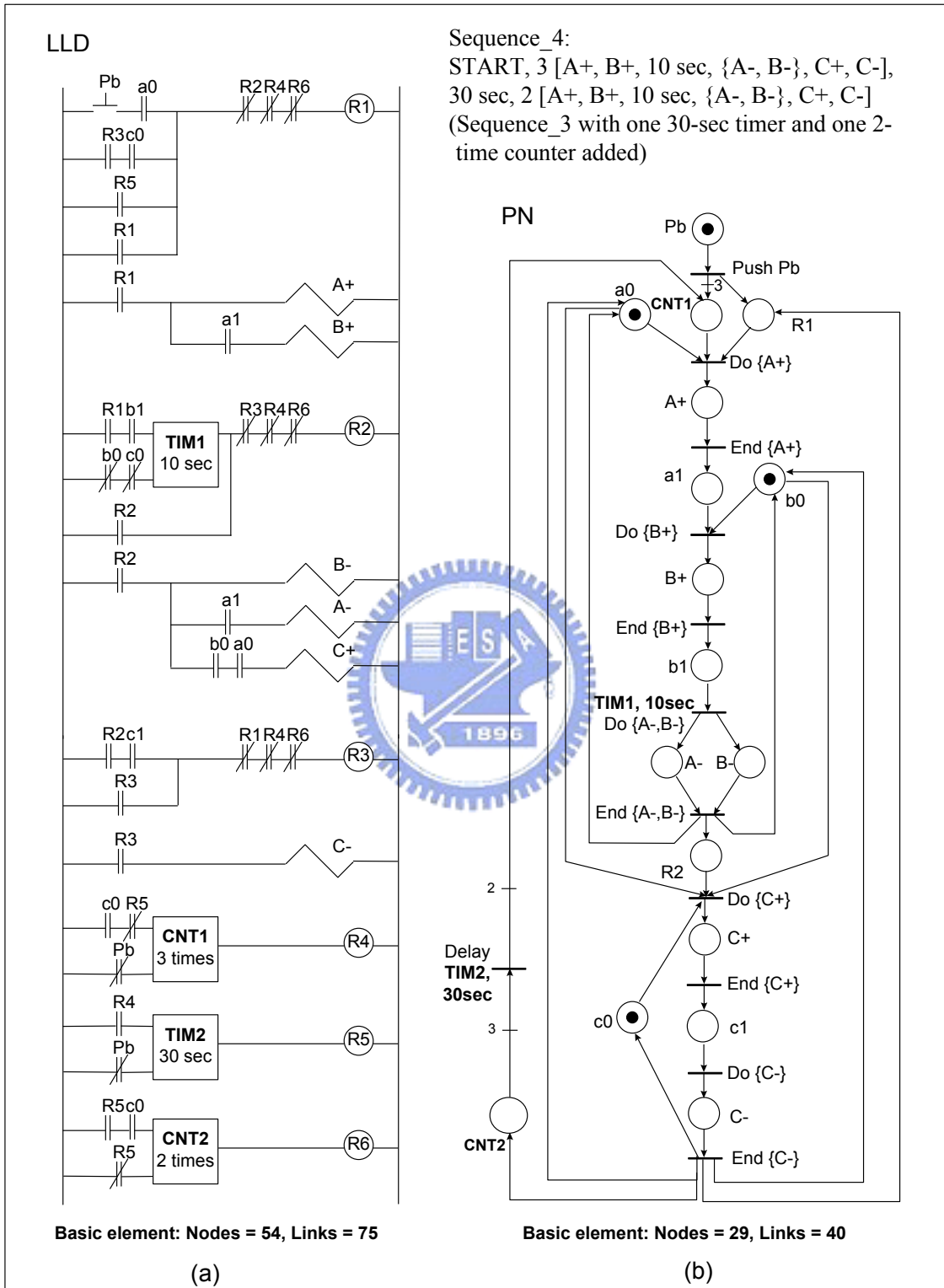


Fig. 2.7. LLD and PN for Sequence_4.

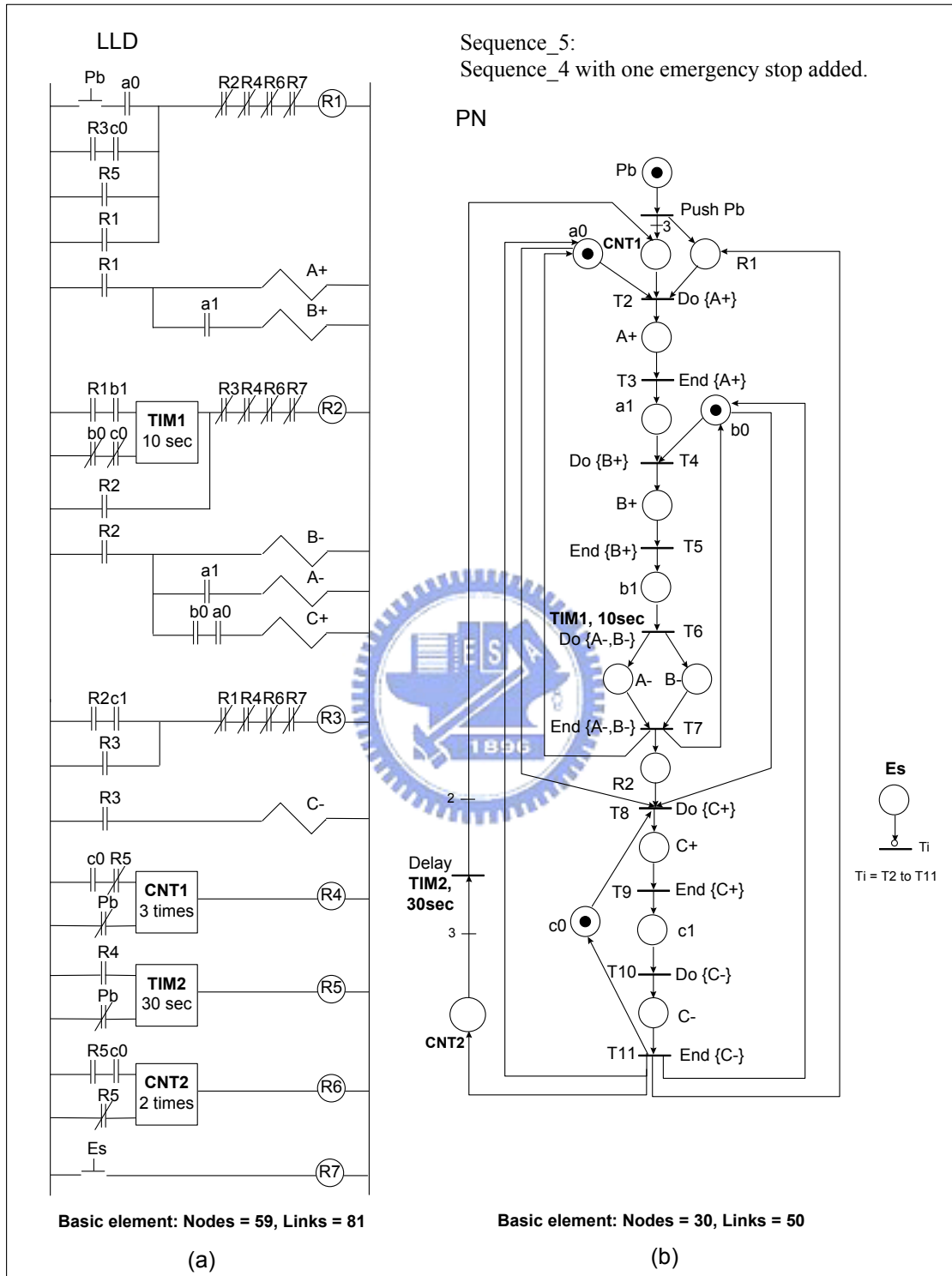


Fig. 2.8. LLD and PN for Sequence_5.

Table 2.4. IF-THEN formats of LLD and PN in Fig. 2.4-2.8.

	LLD	PN
Seq_1	<ol style="list-style-type: none"> 1. $((Pb \cap a0) \cup (R3 \cap c0) \cup R1) \cap R2' \rightarrow R1$, 2. $R1 \rightarrow A+$, 3. $R1 \cap a1 \rightarrow B+$, 4. $((R1 \cap b1) \cup R2) \cap R3' \rightarrow R2$, 5. $R2 \rightarrow B-$, 6. $R2 \cap a1 \rightarrow A-$, 7. $R2 \cap b0 \cap a0 \rightarrow C+$, 8. $((R2 \cap c1) \cup R3) \cap R1' \rightarrow R3$, 9. $R3 \rightarrow C-$. <p>Rules = 9, Operators = 31</p>	<ol style="list-style-type: none"> 1. $Pb \rightarrow R1$, 2. $a0 \cap R1 \rightarrow A+$, 3. $A+ \rightarrow a1$, 4. $a0 \cap b0 \rightarrow B+$, 5. $B+ \rightarrow b1$, 6. $b1 \rightarrow A-\cap B-$, 7. $A-\cap B- \rightarrow a0 \cap b0 \cap R2$, 8. $R2 \cap a0 \cap b0 \cap c0 \rightarrow C+$, 9. $C+ \rightarrow c1$, 10. $c1 \rightarrow C-$, 11. $C- \rightarrow a0 \cap b0 \cap c0 \cap R1$. <p>Rules = 11, Operators = 23</p>
Seq_2 (Seq_1 with one timer added)	<ol style="list-style-type: none"> 1. $((Pb \cap a0) \cup (R3 \cap c0) \cup R1) \cap R2' \rightarrow R1$, 2. $R1 \rightarrow A+$, 3. $R1 \cap a1 \rightarrow B+$, 4. $((R1 \cap b1 \cap TIM1) \cup R2) \cap R3' \rightarrow R2$, 5. $b0' \cap c0' \rightarrow (RST)TIM1$, 6. $R2 \rightarrow B-$, 7. $R2 \cap a1 \rightarrow A-$, 8. $R2 \cap b0 \cap a0 \rightarrow C+$, 9. $((R2 \cap c1) \cup R3) \cap R1' \rightarrow R3$, 10. $R3 \rightarrow C-$. <p>Rules = 10, Operators = 34</p>	<ol style="list-style-type: none"> 1. $Pb \rightarrow R1$, 2. $a0 \cap R1 \rightarrow A+$, 3. $A+ \rightarrow a1$, 4. $a0 \cap b0 \rightarrow B+$, 5. $B+ \rightarrow b1$, 6. $b1 \cap TIM1 \rightarrow A-\cap B-$, 7. $A-\cap B- \rightarrow a0 \cap b0 \cap R2$, 8. $R2 \cap a0 \cap b0 \cap c0 \rightarrow C+$, 9. $C+ \rightarrow c1$, 10. $c1 \rightarrow C-$, 11. $C- \rightarrow a0 \cap b0 \cap c0 \cap R1$. <p>Rules = 11, Operators = 24</p>
Seq_3 (Seq_2 with one counter added)	<ol style="list-style-type: none"> 1. $((Pb \cap a0) \cup (R3 \cap c0) \cup R1) \cap R2' \cap R4' \rightarrow R1$, 2. $R1 \rightarrow A+$, 3. $R1 \cap a1 \rightarrow B+$, 4. $((R1 \cap b1 \cap TIM1) \cup R2) \cap R3' \cap R4' \rightarrow R2$, 5. $b0' \cap c0' \rightarrow (RST)TIM1$, 6. $R2 \rightarrow B-$, 7. $R2 \cap a1 \rightarrow A-$, 8. $R2 \cap b0 \cap a0 \rightarrow C+$, 9. $((R2 \cap c1) \cup R3) \cap R1' \cap R4' \rightarrow R3$, 10. $R3 \rightarrow C-$, 11. $c0 \cap CNT1 \rightarrow R4$, 12. $Pb' \rightarrow (RST)CNT1$. <p>Rules = 12, Operators = 40</p>	<ol style="list-style-type: none"> 1. $Pb \rightarrow R1 \cap (SET)CNT1$, 2. $a0 \cap R1 \cap CNT1 \rightarrow A+$, 3. $A+ \rightarrow a1$, 4. $a0 \cap b0 \rightarrow B+$, 5. $B+ \rightarrow b1$, 6. $b1 \cap TIM1 \rightarrow A-\cap B-$, 7. $A-\cap B- \rightarrow a0 \cap b0 \cap R2$, 8. $R2 \cap a0 \cap b0 \cap c0 \rightarrow C+$, 9. $C+ \rightarrow c1$, 10. $c1 \rightarrow C-$, 11. $C- \rightarrow a0 \cap b0 \cap c0 \cap R1$. <p>Rules = 11, Operators = 26</p>
Seq_4 (Seq_3 with one timer and one counter added)	<ol style="list-style-type: none"> 1. $((Pb \cap a0) \cup (R3 \cap c0) \cup R1) \cap R2' \cap R4' \cap R6' \rightarrow R1$, 2. $R1 \rightarrow A+$, 3. $R1 \cap a1 \rightarrow B+$, 4. $((R1 \cap b1 \cap TIM1) \cup R2) \cap R3' \cap R4' \cap R6' \rightarrow R2$, 5. $b0' \cap c0' \rightarrow (RST)TIM1$, 6. $R2 \rightarrow B-$, 7. $R2 \cap a1 \rightarrow A-$, 8. $R2 \cap b0 \cap a0 \rightarrow C+$, 9. $((R2 \cap c1) \cup R3) \cap R1' \cap R4' \cap R6' \rightarrow R3$, 10. $R3 \rightarrow C-$, 11. $c0 \cap R5 \cap CNT1 \rightarrow R4$, 12. $Pb' \rightarrow (RST)CNT1$, 13. $R4 \cap TIM2 \rightarrow R5$, 14. $Pb' \rightarrow (RST)TIM2$, 15. $R5 \cap c0 \cap CNT2 \rightarrow R6$, 16. $R5' \rightarrow (RST)CNT2$. <p>Rules = 16, Operators = 52</p>	<ol style="list-style-type: none"> 1. $Pb \rightarrow R1 \cap (SET)CNT1$, 2. $a0 \cap R1 \cap CNT1 \rightarrow A+$, 3. $A+ \rightarrow a1$, 4. $a0 \cap b0 \rightarrow B+$, 5. $B+ \rightarrow b1$, 6. $b1 \cap TIM1 \rightarrow A-\cap B-$, 7. $A-\cap B- \rightarrow a0 \cap b0 \cap R2$, 8. $R2 \cap a0 \cap b0 \cap c0 \rightarrow C+$, 9. $C+ \rightarrow c1$, 10. $c1 \rightarrow C-$, 11. $C- \rightarrow a0 \cap b0 \cap c0 \cap R1 \cap (SET)CNT2$, 12. $CNT2 \cap TIM2 \rightarrow (SET)CNT1$. <p>Rules = 12, Operators = 29</p>
Seq_5 (Seq_4 with one emergency stop added)	<ol style="list-style-type: none"> 1. $((Pb \cap a0) \cup (R3 \cap c0) \cup R1) \cap R2' \cap R4' \cap R6' \cap R7' \rightarrow R1$, 2. $R1 \rightarrow A+$, 3. $R1 \cap a1 \rightarrow B+$, 4. $((R1 \cap b1 \cap TIM1) \cup R2) \cap R3' \cap R4' \cap R6' \cap R7' \rightarrow R2$, 5. $b0' \cap c0' \rightarrow (RST)TIM1$, 6. $R2 \rightarrow B-$, 7. $R2 \cap a1 \rightarrow A-$, 8. $R2 \cap b0 \cap a0 \rightarrow C+$, 9. $((R2 \cap c1) \cup R3) \cap R1' \cap R4' \cap R6' \cap R7' \rightarrow R3$, 10. $R3 \rightarrow C-$, 11. $c0 \cap R5 \cap CNT1 \rightarrow R4$, 12. $Pb' \rightarrow (RST)CNT1$, 13. $R4 \cap TIM2 \rightarrow R5$, 14. $Pb' \rightarrow (RST)TIM2$, 15. $R5 \cap c0 \cap CNT2 \rightarrow R6$, 16. $R5' \rightarrow (RST)CNT2$, 17. $ES \rightarrow R7$. <p>Rules = 17, Operators = 56</p>	<ol style="list-style-type: none"> 1. $Pb \rightarrow R1 \cap (SET)CNT1$, 2. $a0 \cap R1 \cap CNT1 \cap ES' \rightarrow A+$, 3. $A+ \cap ES' \rightarrow a1$, 4. $a0 \cap b0 \cap ES' \rightarrow B+$, 5. $B+ \cap ES' \rightarrow b1$, 6. $b1 \cap TIM1 \cap ES' \rightarrow A-\cap B-$, 7. $A-\cap B-\cap ES' \rightarrow a0 \cap b0 \cap R2$, 8. $R2 \cap a0 \cap b0 \cap c0 \cap ES' \rightarrow C+$, 9. $C+ \cap ES' \rightarrow c1$, 10. $c1 \cap ES' \rightarrow C-$, 11. $C-\cap ES' \rightarrow a0 \cap b0 \cap c0 \cap R1 \cap (SET)CNT2$, 12. $CNT2 \cap TIM2 \rightarrow (SET)CNT1$. <p>Rules = 12, Operators = 39</p>

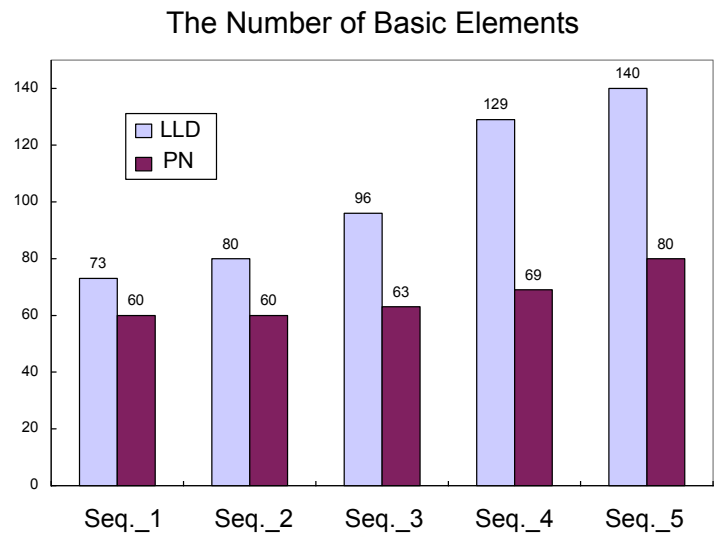


Fig. 2.9. Required basic elements in the basic element approach.

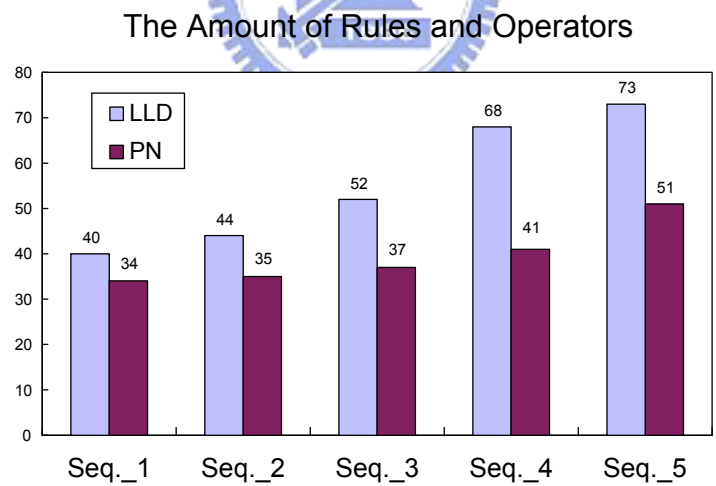


Fig. 2.10. Required rules and logical operators in the IF-THEN transformation.

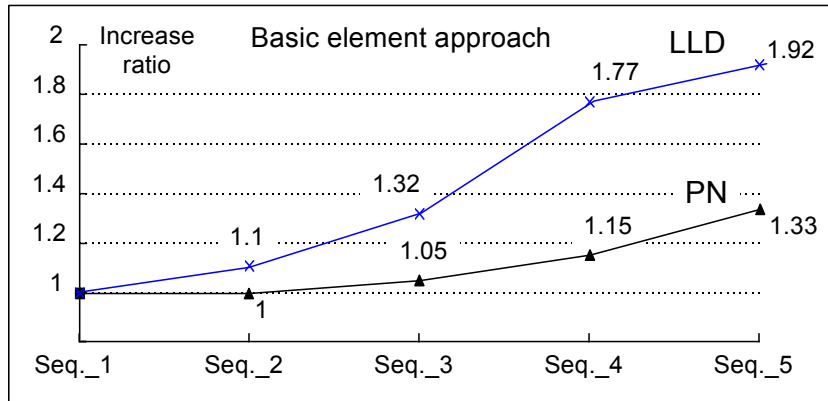


Fig. 2.11. The increase ratio for the basic element approach.

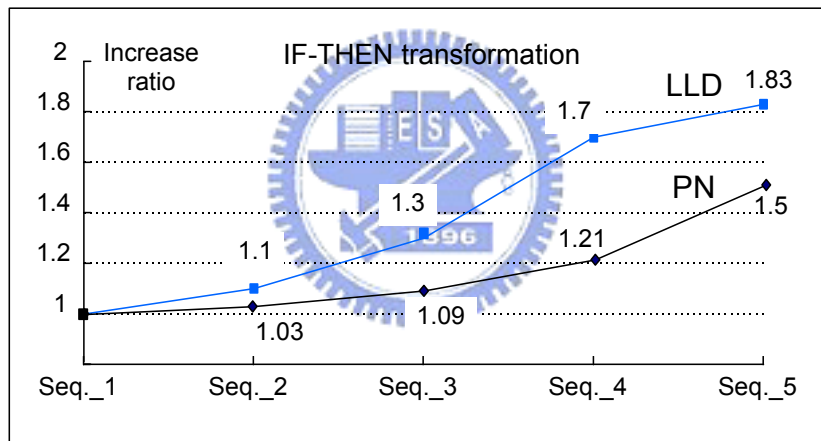


Fig. 2.12. The increase ratio for the IF-THEN transformation.

2.4. Discussions

This chapter presents a novel and unified approach for evaluating the computational burden and complexity subject of sequence programming for different structures. Because the basic elements for LLD and PN structures possess different physical meanings, results using the basic element approach are not adequate to conclude which design structure is more efficient. By applying the proposed IF-THEN transformation

approach, we obtain the same IF-THEN rules and logical operators for both LLD and PN structures, and thus the results in Fig. 2.10 show conclusively that the PN structure design is more efficient.

Furthermore, by applying the IF-THEN transformation, results indicate that the PN structure also leads to a lower increase ratio than the LLD structure, as shown in Fig. 2.12. Thus, design via the PN structure is more flexible when the specification changes. Similar trend can also be observed using the basic element approach as shown in Fig. 2.11. Therefore, the PN structure for sequence control design will become more valid for large-scale processes.

Although both the basic element approach and the IF-THEN transformation present similar results in terms of increase ratios for given sequence changes as shown in Fig. 2.11-2.12, a comparison indicates that the basic element approach overestimates the complexity of LLD, and underestimates that of PN. For example, comparing Sequence_1 with Sequence_2, which adds a timer to Sequence_1, results of the basic element approach indicate that both sequences require the same number of basic elements by using the PN, as shown in Fig. 2.9. This is obviously misleading. On the other hand, evaluation results with the present IF-THEN transformation properly indicate that the complexity of PN increases from 34 to 35, as shown in Fig. 2.10. Therefore, the proposed IF-THEN transformation is more realistic for evaluating sequence control design than the basic element approach.

2.5. Summary

In this chapter, we have proposed a unified comparison approach to adequately evaluate the LLD and PN by using the IF-THEN transformation. Thus, more realistic and reasonable results can be obtained to analyze the design complexity and flexibility to specification changes for different structures. Results show that the PN is simpler and more flexible than LLD in realization of sequence controllers. Hence, based on the given example, PN might be a promising solution for modern industrial control systems.

Chapter 3

Design of the Sequence Controller in Manufacturing Systems

In the previous chapter, a comparison between the ladder logic diagram (LLD) and Petri net (PN) has been provided. However, in real industrial environments, most industrial PLC users still prefer to program in LLD. Hence, this chapter presents a systematic approach to the LLD implementation of the sequence controller in manufacturing systems. Basically, the simplified Petri net controller (SPNC) is employed in the present approach (Lee, 1999). By employing the IDEF0, the SPNC model can be built through the material flow diagram and the information flow diagram. Then, the LLD can be transformed from the SPNC through the token passing logic (TPL). The proposed approach, including the IDEF0, SPNC, and TPL tools, leads to the standard IEC1131-3 LLD for PLC implementation. Finally, an application of a stamping process is provided to illustrate the design procedure of the developed approach.

3.1. Simplified Petri Net Controller

In this section, we propose a simplified Petri net controller (SPNC) by introducing sensor states into the ordinary PN. The SPNC is applied to simulate the manufacturing system and to lead the IDEF0 to LLD in the proposed IDEF0/SPNC/TPL/LLD approach.

3.1.1 Formal Definition

Fig. 3.1 (a) shows an ordinary PN model for pushing a button to trigger a process. By using the ordinary PN approach in controlling manufacturing processes, to deal with multiple sensor readings makes the net structure become more complicated and difficult to analyze. Therefore, by introducing the sensor state into the PN to form an SPNC, the

net structure becomes more simplified for implementation. From the control point of view, as shown in Fig. 3.1 (b), the sensor state in the SPNC replaces the reading sensor model such as push buttons or limit switches within the ordinary PN. Note that the condition of sensor states may change depending on the practical situation. Thus, as sensors increase in processes, the net structure of the SPNC is greatly simplified, as shown in Fig. 3.1 (c). Then, it becomes easy to model and implement the sequence controller through the SPNC defined as:

$$\text{SPNC} = (P, T, A, S, M_0) \quad (3.1)$$

where,

$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places, where $m > 0$;

$T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$, where $n > 0$;

$A \subseteq \{P \times T\} \cup \{T \times P\}$ is the set of arcs between the places and transitions,

$S = \{s_1, s_2, \dots, s_n\}$ is the set of sensor states, and

$M_0 : P \rightarrow 1$ is the initial marking.

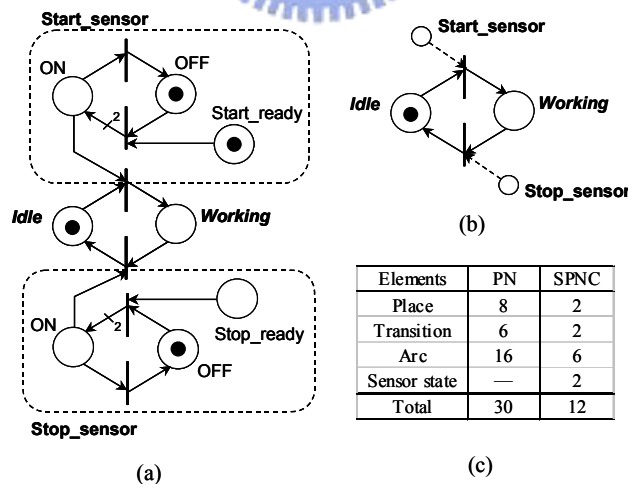


Fig. 3.1. The comparison between the PN and the SPNC via a simple process. (a) Ordinary PN. (b) SPNC. (c) Comparison results.

3.1.2 Graphical Representation

As shown in Fig. 3.2, the SPNC consists of three kinds of nodes: 1) the place, drawn as a circle, 2) the transition, drawn as a bar, and 3) the sensor state, drawn as a smaller circle with a hidden arrow. The arcs, represented by directed arrows, are either from a place to a transition or from a transition to a place. In modeling, the marking conditions of places represent the status of the system and the transitions represent events. A transition has a set of input and output places, which represent the pre-conditions and post-conditions of the event, respectively. A sensor state, associated with its transitions, represents the sensor readings as a firing condition which triggers a manufacturing sequence. The sensor state is a Boolean variable that can be 0 in which case the related transition is not fired, or 1 in which case the related transition is fired if it is enabled. The marking of the SPNC is represented by the number of tokens in each place, drawn as black dots. The presence of a token in a given place means that the associated condition is true or that the actions associated with this place are taken.

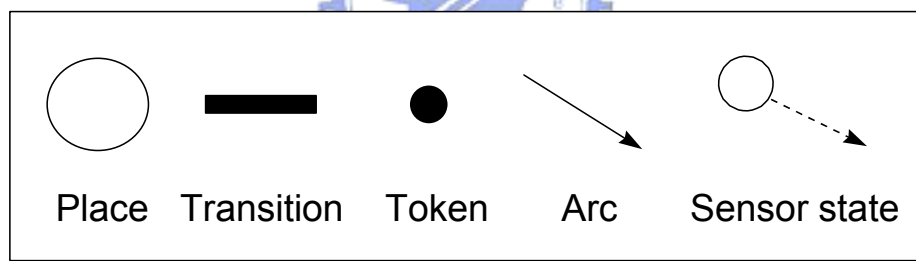


Fig. 3.2. The icon definition of the SPNC.

3.1.3 Dynamic Behavior

The dynamic behavior of a system is simulated by the distribution of tokens in places as the enable transitions fire. The flow of tokens in the SPNC is governed according to the following rules:

1) Enabling rule:

A transition is said to be enabled, if all its input places are marked.

2) Firing rule:

Furthermore, the enabled transition is fired if all its sensor states are true. When an enabled transition fires, it removes one token from all its input places and deposits one token into all its output places at the same time.

3.1.4 Comparison with Other Models

The behavior of the proposed SPNC is similar to the sequential function chart (SFC). However, since SFC is derived from PN with some modifications and simplifications, theoretical results of PN cannot be directly applied to SFC (Miyazawa et al., 1997). Since the present SPNC is an extension of the PN by introducing sensor states, SPNC allows formal analysis of various properties, such as the safety, liveness, and reversibility for the process (David and Alla, 1994). Moreover, SFC only offers the method for depicting sequences of control system without providing any mechanisms to perform the functional analysis. Note that in the present IDEF0/SPNC/TPL/LLD approach, by applying the IDEF0 for functional analysis and information flow design, the SPNC model can be transformed from the information flow diagram.

Furthermore, compared with other extended PN applications such as Interpreted PN (Moalla, 1985), Automation PN (Uzam and Jones, 1998), or Signal Interpreted PN (Frey, 2000), which use external events to model sensor readings, the present SPNC simply applies the sensor states to model the firing conditions. Also, the present IDEF0/SPNC/TPL/LLD approach obtains the PLC programs systematically, from the design specifications through the SPNC, and to the final LLD. Since the PN model is inherently concurrent, whereas the LLD is typically scan-based, the sequential specification must be determinate and deterministic in the present approach. Also, the mono-marked restrictions design is required in the proposed SPNC to guarantee the safety of the sequence in practice.

3.2. The IDEF0/SPNC/TPL/LLD Approach

In this section, the integrated IDEF0/SPNC/TPL/LLD approach, including the IDEF0,

SPNC, and TPL tools, is proposed to systematically obtain the LLD for PLC implementation. The design procedure of the IDEF0/SPNC/TPL/LLD approach, depicted in Fig. 3.3, consist of five stages and each stage is described as follows.

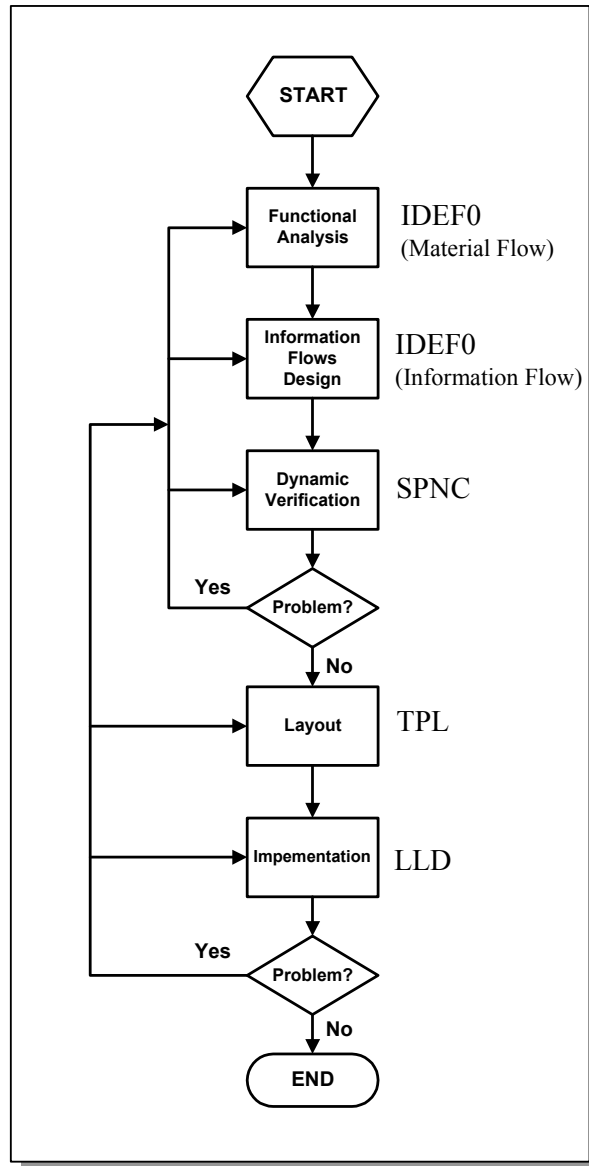


Fig. 3.3. Design procedure of the IDEF0/SPNC/TPL/LLD approach.

3.2.1 Functional Analysis Stage: IDEF0

With the given specifications, the purpose of functional analysis is to realize the functions and operations of the system and then generate the control signals for the next stage. At this stage, each function of the manufacturing system has to be specified with a top-down hierarchically decomposing process by using the IDEF0 (Prabhaka, 1993). IDEF0 is an activity-oriented modeling approach and its representation of a manufacturing process consists of an ordered set of boxes representing activities performed by the system. The inputs are those items transformed by the activity and the outputs are the results of the activity, as shown in Fig. 3.4. The mechanisms, drawn as supporting arrows, represent resources such as machines, computers and operators, etc. The decomposition process continues until there is sufficient in detail on the basic activities to serve the purpose of sequence control. A functional model of the material flow diagram is obtained at this stage.

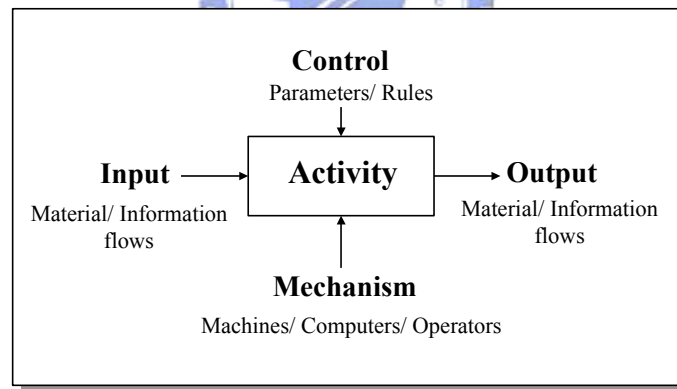


Fig. 3.4. The IDEF0 scheme.

3.2.2 Information Flow Design Stage: IDEF0

At this second stage, the information flow is used to control the material flow in a manufacturing system. The information flow diagram is constructed from the material flow diagram with static analysis, again using the IDEF0. In the information flow diagram, the input and output commands are designed to enable the activity and to

change the machine status after firing, respectively. Because the mechanisms will be assigned within the I/O ports at the layout stage later, the supporting arrows for mechanisms are omitted here to simplify the information flow design. The sensor readings representing the conditions to fire the activity are drawn as control signal arrows. A controllable model of the information flow diagram is obtained at this stage.

3.2.3 Dynamic Verification Stage: SPNC

The information flow diagram only represents system activities and their interrelationships. Since it does not show direct logical and dynamic dependencies between activities, a dynamic SPNC model, transformed from the information flow diagram, is applied to verify the dynamic behavior of the system. The transformation from the information flow diagram into the SPNC model is based on the following steps:

- Step 1) An activity box in the information flow diagram is transformed into a transition of the SPNC.
- Step 2) The input and output commands are transformed into input and output places, respectively.
- Step 3) The control signals of the sensor readings are transformed into sensor states.
- Step 4) The initial marking of the SPNC is set according to the initial condition of the system.

An example is shown in Fig. 3.5. The activity of the information flow diagram is transformed into the transition **T1**. The input command **I1** and output command **I2** are transformed into the input place **P1** and output place **P2**, respectively, and the control signal **control** is transformed into the sensor state **S1**. When the SPNC model is obtained, the correctness of the sequence order can be verified by studying the behaviors via computer simulations. Also the properties of the PN such as the safety, liveness, and reversibility can be analyzed to identify the dynamic behavior.

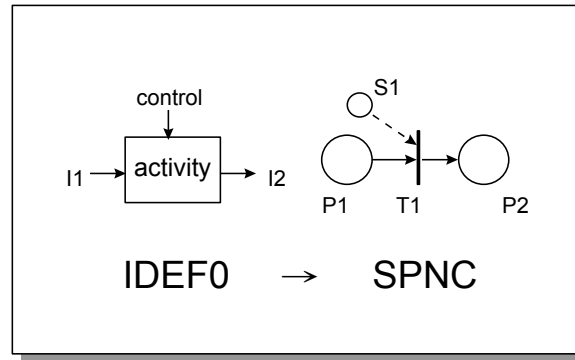


Fig. 3.5. The transformation from the IDEF0 to the SPNC.

3.2.4 Layout Stage: TPL

To simplify the conversion of the SPNC into the LLD, the token passing logic (TPL) is employed in this stage (Uzam and Jones, 1998). The attractive feature of the TPL is that it facilitates the direct conversion of a SPNC into a generic form of control logic, which may be implemented with low-level languages such as LLD, or with high-level languages such as C. This is achieved by adopting the SPNC concept of using tokens as the main mechanism for controlling the flow of the control logic. At this stage, the SPNC model is transformed into the TPL model to assign the I/O ports for actions and sensor readings. For applications in a variety of industrial PLC hardware, the TPL is defined as follows:

$$\text{TPL} = (M, T, A, in, out, time) \quad (3.2)$$

where,

$M = \{M_1, M_2, \dots, M_m\}$ is a finite set of memory bits,.

$T = \{T_1, T_2, \dots, T_n\}$ is a finite set of transitions,

$A \subseteq \{M \times T\} \cup \{T \times M\}$ is the set of arcs between the memories and transitions,

$in = \{in_1, in_2, \dots, in_n\}$ is the set of sensor inputs,

$out = \{out_1, out_2, \dots, out_m\}$ is the set of actuator outputs, and

$time = \{time_1, time_2, \dots, time_m\}$ is the set of delay timers.

The transformation from the SPNC model into the TPL form is based on the following steps:

- Step 1) The transition of the SPNC is transformed into a transition of the TPL.
- Step 2) The place is transformed into a memory bit.
- Step 3) The sensor state is transformed into a sensor input.
- Step 4) For the action with a place, besides the memory bit, an actuator output is assigned.
- Step 5) For the delay time with a place, besides the memory bit, a delay timer is assigned.

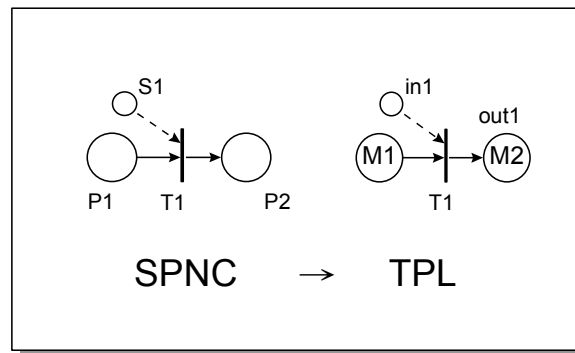


Fig. 3.6. The transformation from the SPNC to the TPL.

An example is shown in Fig. 3.6. The places **P1** and **P2** are transformed into the memory bits **M1** and **M2**, respectively, and the sensor state **S1** is transformed into the sensor input **in1**. Assume there is an action with **P2**, the actuator output **out1** is assigned. Hence, each place whose capacity is limited to one within the SPNC corresponds to a memory bit in the TPL. The token flow is then simulated by setting and resetting these memory bits. Thus, each place within the SPNC has at least one associated memory bit in the TPL. The sensor state within the SPNC corresponds to a sensor input contact in the TPL. To simulate the firing of a transition, if the memory bits associated with input places are set and the sensor inputs of the transition yield “true”, the memory bits at the input places are reset and the memory bits at the output places are set simultaneously. Moreover, the actions and delays within the SPNC are assigned to appropriate memory

bits within the TPL by using the actuator outputs and delay timers, respectively. By using the TPL, the I/O ports for the sensor readings and actuator outputs are assigned and the layout for implementation in LLD can be completed. The TPL bridges the gap between SPNC and LLD and provides a simple way of developing PLC controllers.

3.2.5 Implementation Stage: LLD

In order to convert the TPL model into LLD code for real time implementation, a direct mapping is used from the TPL to the LLD by maintaining the enabling and firing rules at this stage. The transformation from the TPL model into the LLD format is based on the following steps:

- Step 1) *Initial condition setting*: the token in the SPNC is mapped to the corresponding internal relay with the SET command.
- Step 2) For each transition, the input memory is mapped to a conditional contact and an internal relay with the RST command and the output memory is mapped to an internal relay with the SET command.
- Step 3) The sensor input is mapped to a conditional contact for the associated transition.
- Step 4) The output relay is assigned to send the command to perform the operation.
- Step 5) The delay timer is assigned to perform the delay.

An example is shown in Fig. 3.7. For transition **T1**, the input memory **M1** is mapped to a conditional contact and an internal relay **M1** with the RST command and the output memory **M2** is mapped to a internal relay **M2** with the SET command. The sensor input **in1** is mapped to a conditional contact **X1** and the actuator output **out1** is mapped to the output relay **Y1**. By integrating initial condition and setting all transitions, the LLD for sequence control is thus completed.

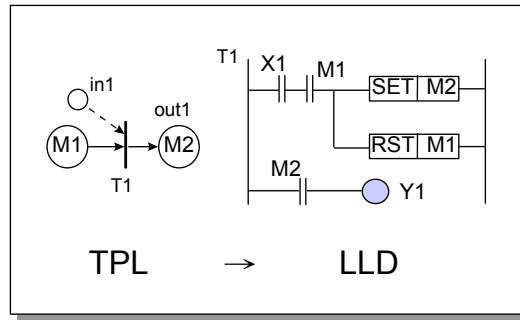


Fig. 3.7. The transformation from the TPL to the LLD.

In the proposed IDEF0/SPNC/TPL/LLD approach, the material flow diagram and the information flow diagram are obtained by using the IDEF0 technique for functional analysis and information flow design. Then, the information flow diagram is transformed into the SPNC model to verify its dynamic behavior. Subsequently, the SPNC model is converted into a TPL model for implementation layout. Finally, the IEC1131-3 LLD for implementation on PLC controller is obtained using a direct mapping from the TPL into LLD. Fig. 3.8 summarizes the transformations in the proposed IDEF0/SPNC/TPL/LLD approach.

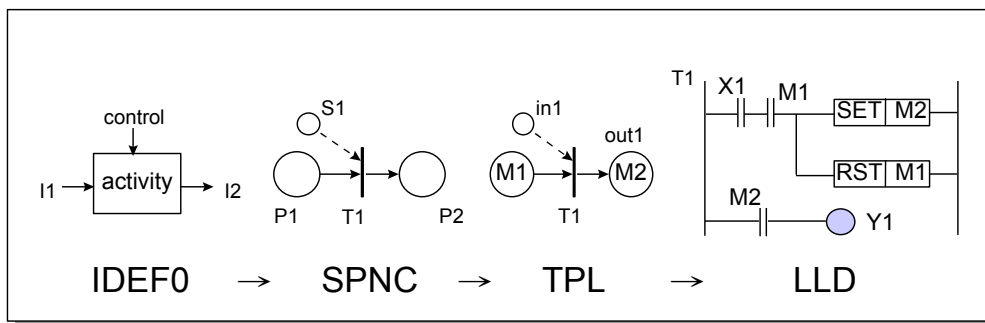


Fig. 3.8. The transformations of the IDEF0/SPNC/TPL/LLD approach.

3.3. Example: A Stamping Process

To demonstrate the viability of the developed approach, an application to a stamping process is provided.

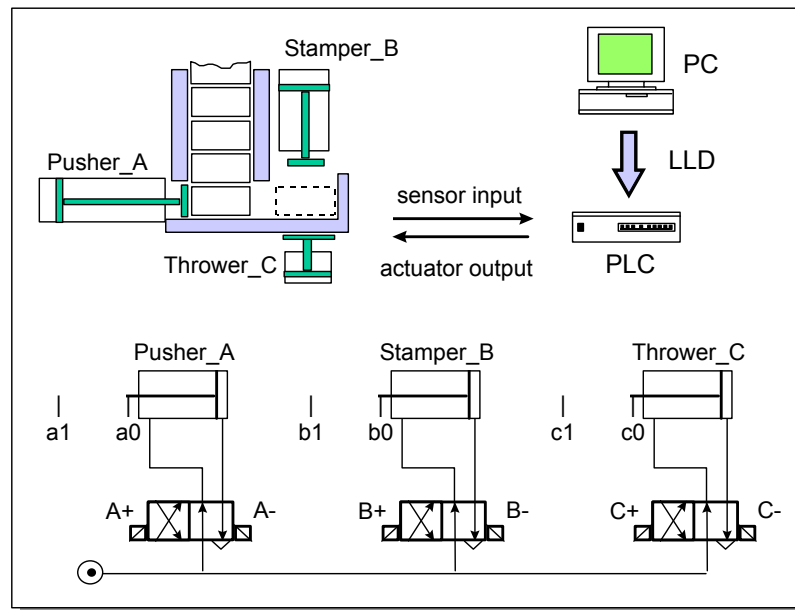


Fig. 3.9. The stamping system.

3.3.1 System Description

As shown in Fig. 3.9, a stamping system consists of three cylinders which are operated by four-port and two-way solenoid valves. Each cylinder has two normally open limit switches. For example, when the end of pusher_A contacts limit switch a0, a0 is then closed. This indicates that pusher_A is at the end of its return stroke. The whole system has 7 input sensors corresponding to 6 limit switches, one push button for starting the system and 6 output actuators corresponding to 6 solenoid valves. In the stamping process, pusher_A moves the workpiece from a store onto the worktable. Then the workpiece is stamped by stamper_B and afterwards is ejected by thrower_C. Thus, the sequence of the stamping system is A+, B+, {A-, B-}, C+, C-, where the plus and the minus signs mean a piston performing forward strokes and return strokes, respectively.

{A-, B-} represents two concurrent actions as the pistons of both pusher_A and stamper_B perform return strokes simultaneously.

3.3.2 Sequence Controller Design

Through the use of the proposed IDEF0/SPNC/TPL/LLD approach, as shown in Fig. 3.10, the LLD code for real time implementation on PLC controllers was systematically generated. First, by using the IDEF0 technique, the material flow diagram and the information flow diagram were obtained. Then, to verify its dynamic behavior, the information flow diagram has transformed into the SPNC model. Subsequently, the SPNC model was converted into a TPL model for layout. Finally, the LLD for implementation with PLC controllers was obtained by a direct mapping from the TPL. This LLD code is written for Mitsubishi FX2 PLCs which meet IEC1131-3. Table 3.1 gives the notations used in the IDEF0/SPNC/TPL/LLD together with their descriptions.



3.4. Summary

In this chapter, we have proposed a systematic IDEF0/SPNC/TPL/LLD approach to the PLC-based sequence controller design in manufacturing systems. To obtain the LLD for PLC implementation, the SPNC is defined by introducing the sensor states into the ordinary Petri net and leads to meaningfully simplified process modeling. Moreover, the IDEF0 technique is employed to construct the SPNC model through the material flow diagram and information flow diagram. Starting from the basic sequential specification, the proposed approach includes IDEF0, SPNC, and TPL, and systematically leads to the standard IEC1131-3 LLD for PLC implementation. An application of a stamping process is provided to demonstrate the viability of the developed approach.

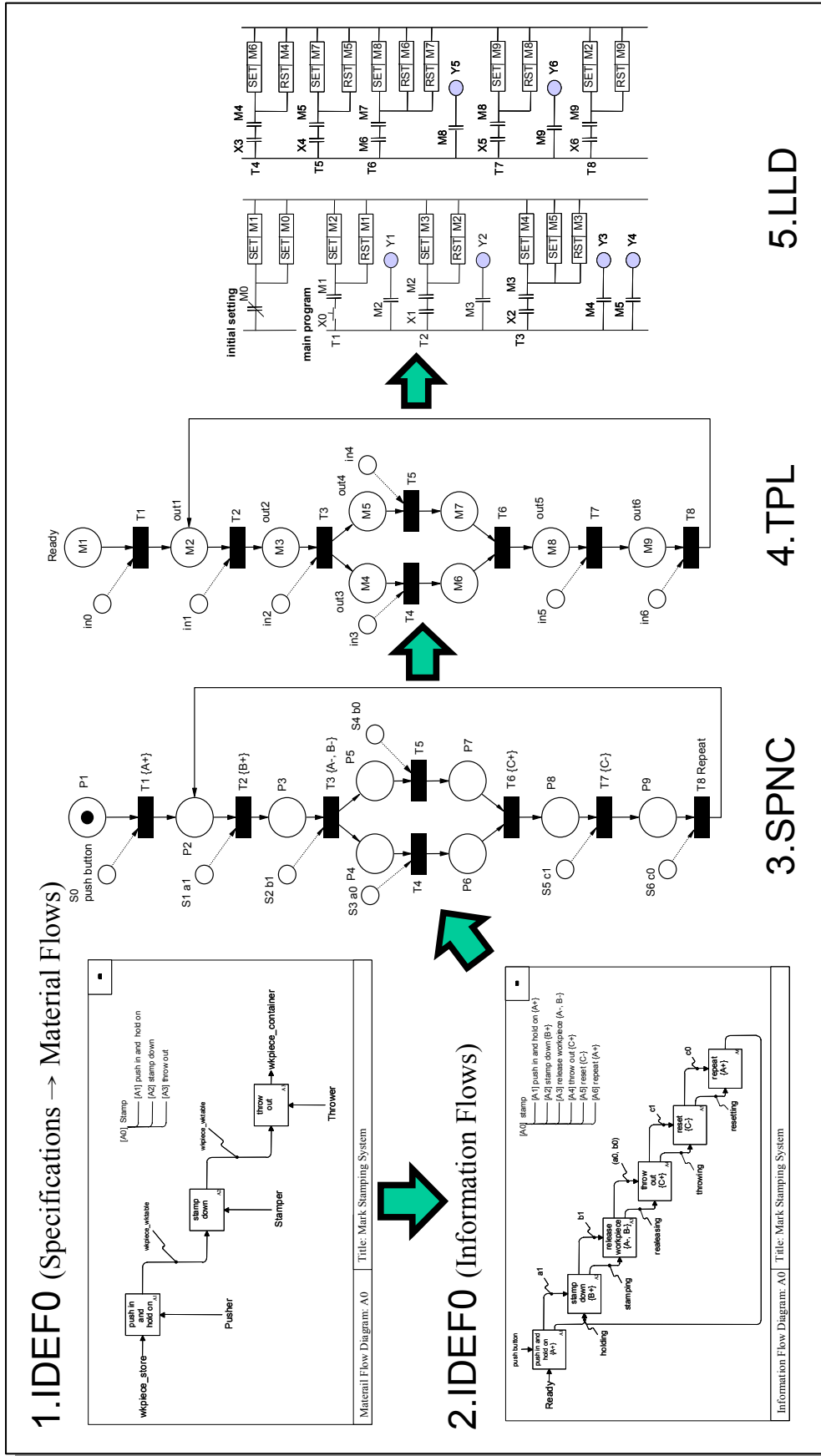


Fig. 3.10. Design of the sequence controller using IDEF0/SPNC/TPL/LLD approach.

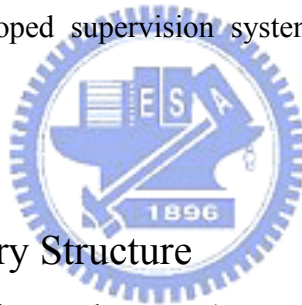
Table 3.1. Notations for the stamping process.

SPNC element	TPL element	LLD element	Description
P1	M1	M1	Ready
P2	M2, out1	M2, Y1	Holding {A}
P3	M3, out2	M3, Y2	Stamping {B}
P4	M4, out3	M4, Y3	Releasing {A}
P5	M5, out4	M5, Y4	Releasing {B}
P6	M6	M6	--
P7	M7	M7	--
P8	M8, out5	M8, Y5	Throwing {C}
P9	M9, out6	M9, Y6	Resetting {C}
T1	T1	--	Push in and Hold on {A+}
T2	T2	--	Stamp down {B+}
T3	T3	--	Release workpiece {A-, B-}
T4	T4	--	--
T5	T5	--	--
T6	T6	--	Throw out {C+}
T7	T7	--	Reset {C-}
T8	T8	--	Repeat {A+}
S0	in0	X0	Push button {ON}
S1	in1	X1	Sensor a1 {ON}
S2	in2	X2	Sensor b1 {ON}
S3	in3	X3	Sensor a0 {ON}
S4	in4	X4	Sensor b0 {ON}
S5	in5	X5	Sensor c1 {ON}
S6	in6	X6	Sensor c0 {ON}

Chapter 4

Remote Supervision for Human-in-the-Loop Systems

In remote-controlled processes, human operations may violate desired safety requirements and result in catastrophic failure. For such human-in-the-loop systems, this chapter proposes a systematic approach to develop supervisory agents that guarantee that remote manual operations meet safety specifications. The PN is applied to model, design, and verify a supervisory controller that prevents human errors. Then, the Java technology is adopted to implement the supervisor as an intelligent agent for on-line supervision of the remote control system. To demonstrate the feasibility and practicability of the proposed approach, the developed supervision system is applied to a rapid thermal process (RTP).



4.1. A Novel Supervisory Structure

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 Fig. 4.1 (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. In this chapter, we propose applying a supervisory design to the present remotely controlled, human-in-the-loop system so as to prevent abnormal operations from being carried out. Fig. 4.1 (b) shows the proposed supervisory control scheme for a remotely located system with the human in the loop.

First, 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.

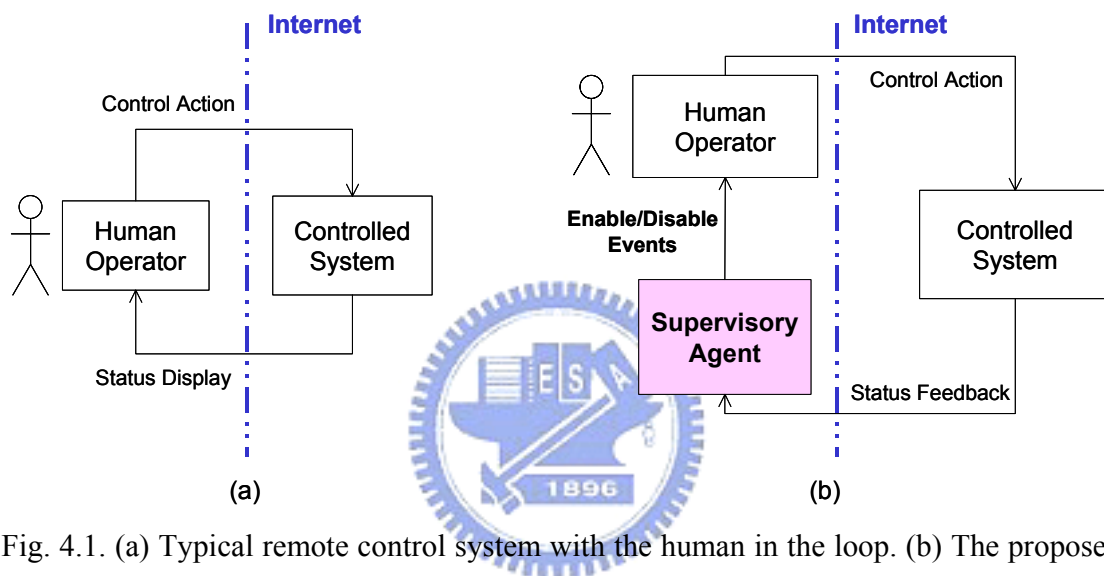


Fig. 4.1. (a) Typical remote control system with the human in the loop. (b) The proposed remote supervisory control scheme.

4.2. Design of the Supervisor Using PN

This section first shows the required control modes and specification types for remote supervisory control. Then, the PN-based procedure for designing the supervisor is described with a simple door-valve system for a RTP.

4.2.1 Control Modes

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

1) *Automatic control 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 control 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.

4.2.2 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 specification 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 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.

4.2.3 PN-Based Design for the Supervisor

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 4.1: 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 :

$$G = G_1 \parallel G_2, \quad (4.1)$$

where,

$$P = P_1 \cup P_2;$$

$$T = T_1 \cup T_2;$$

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

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

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



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:

$$S_G = G \parallel H, \quad (4.2)$$

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. Moreover, the PN plant model is constructed using the task-oriented concept. Each operation is modeled as a task with a start transition, an end transition, a progressive place and a completed place. Note that the start transition is a controllable event as “command” input, while the end transition is an uncontrollable event as “response” output. The supervisor design procedure consists of the following steps:

- Step 1) Construct the PN model of the plant using the task-oriented approach.
- Step 2) Construct the PN model of the specifications using the mutual exclusion concept.
- Step 3) Compose the plant and specification models to yield the supervisor model.
- Step 4) Verify and refine the supervisor model to obtain a live, bounded and reversible model.

4.2.4 Example: A Door-Valve System

Consider a simple example of the interaction for the chamber door-gas valve system in a rapid thermal processor. The general PN model, shown in Fig. 4.2 (a), can be used to describe the open/close tasks for both the door and valve. The initial states of the door and valve are both closed. Assume that one basic safety specification is that the door and valve must not be open at the same time. A PN model for this specification constructed using the mutual exclusion concept is shown in Fig. 4.2 (b). In this model, the `start_open_door` and `start_open_valve` commands are mutually exclusive. Intuitively, performance of the `start_open_valve` command is only allowed if the door is closed and the `start_open_door` event has not yet been fired. If the `start_open_door` command has been fired, the `start_open_valve` command cannot be executed until the `end_close_door` response is given to signal that the door has been closed. The composed PN model of the door-valve system with the safety specification is shown in Fig. 4.2 (c). The supervisory

arcs are shown with dashed lines and the place **ps** showing the supervisor position is drawn thicker than those showing the task positions. 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.

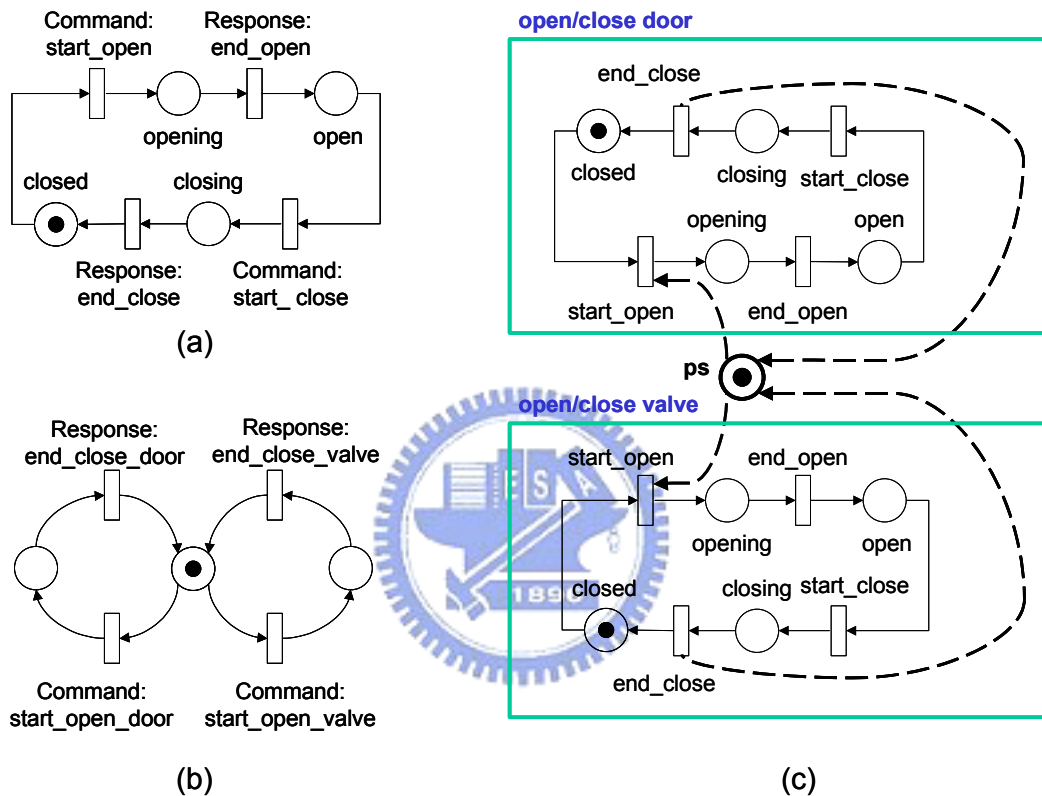


Fig. 4.2. (a) A general model for door and valve components. (b) The mutual exclusion specification model. (c) The composed supervisor for the door-valve system.

4.3. Implementation of the Supervisor Using Java

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.

4.3.1 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.

4.3.2 Client/Server Architecture

Fig. 4.3 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 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.

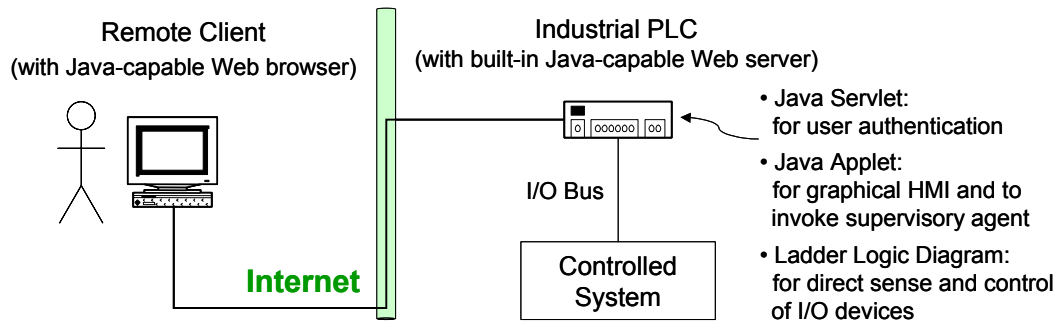


Fig. 4.3. Implementation architecture of the remote supervisory control system.

4.3.3 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.

Fig. 4.4 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 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 Client* can send an action command by pushing an enabled button to control the remote system through the constructed socket.

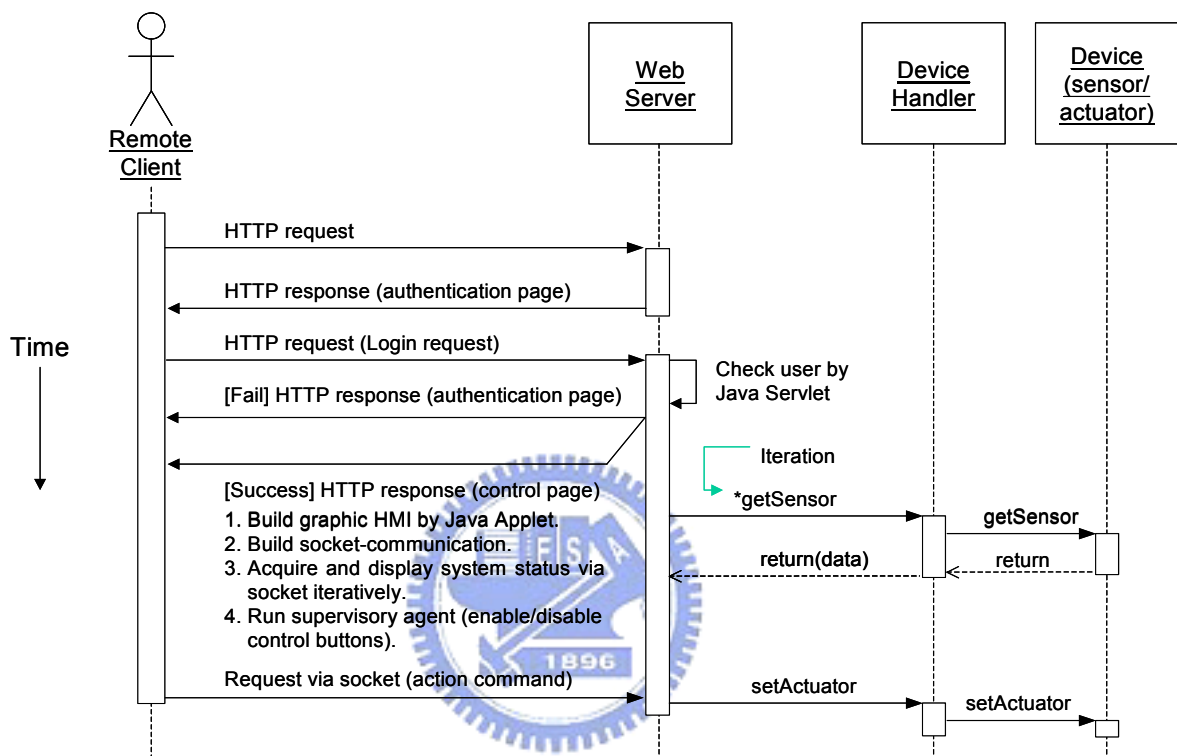


Fig. 4.4. Interactive modeling with sequence diagram for the remote supervisory control system.

4.3.4 Java Implementation

In this thesis, we have employed the Java servlet for authentication and Java applet for graphical HMI. A Java servlet (Hunter and Crawford, 1998) is a compiled code, dynamically loaded to process requests from a 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 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 (Campione and Walrath, 1995) 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 thesis has adopted the Java applet for graphical HMI due to its plentiful availability of application programming interfaces (API). Also, most Web browsers (Navigator or Internet Explorer) provide the JVM to support Java applets. Moreover, as shown in Fig. 4.4, 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 (Hunter and Crawford, 1998). Moreover, Java 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 (Rumbaugh et al, 1991).

4.4. Example: A Rapid Thermal Process

This section demonstrates a practical application of the remote monitoring and supervisory control to a rapid thermal process (RTP) via the Internet.

4.4.1 Description of the RTP System

A rapid thermal processor is a relatively new semiconductor manufacturing device (Fair, 1993). A schematic diagram of the RTP system is shown in Fig. 4.5, 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.

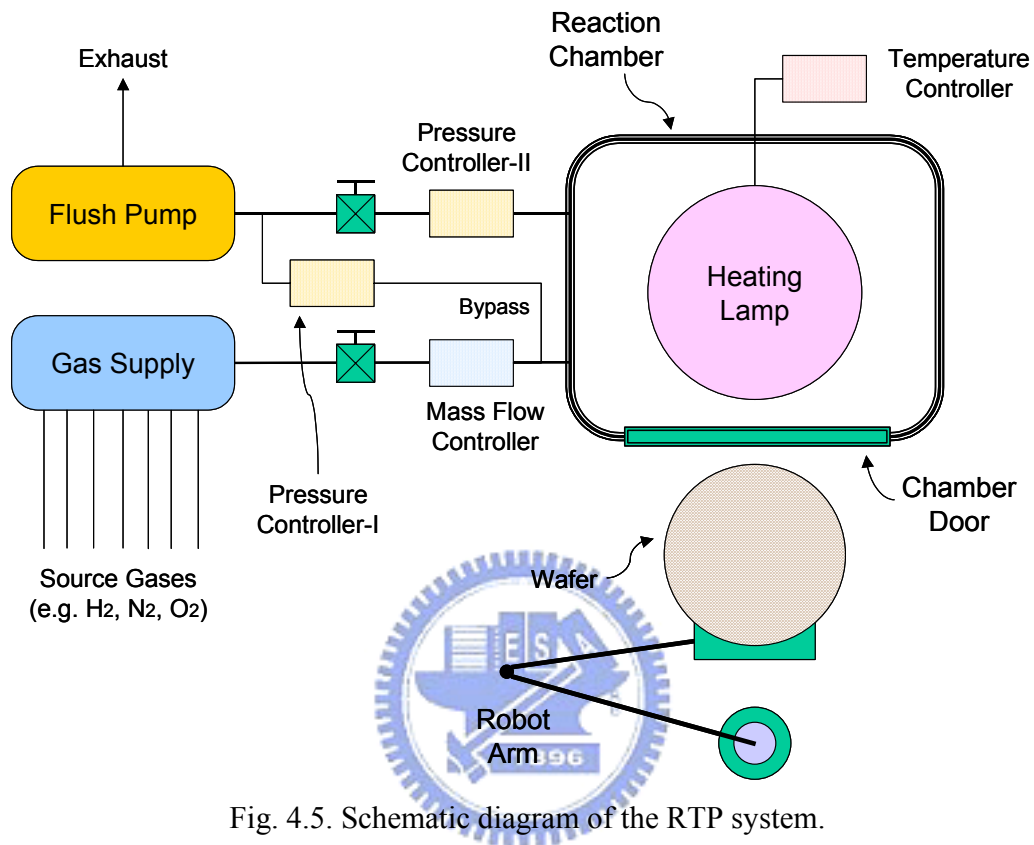


Fig. 4.5. Schematic diagram of the RTP system.

A realistic “recipe” of the hydrogen baking process, i.e. the explicit specification as mentioned in Section 4.2.2, is as follows:

- Step 1) Load the raw wafer.
- Step 2) Close the chamber door.
- Step 3) 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 4) Close the gas valve.
- Step 5) Turn on the heating lamp to bake the wafer with a desired baking temperature and duration of 1000 °C and 4 seconds, respectively.
- Step 6) Turn off the heating lamp.
- Step 7) Turn on the flush pump with a desired pressure of less than 0.05 Torr.

- Step 8) Turn off the flush pump.
- Step 9) Open the chamber door.
- Step 10) 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 4.2.2, must be enforced throughout system operation.

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

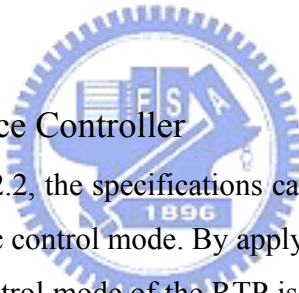
Spec-2: Wafer Loading/unloading is allowed only when the door is open.

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.

4.4.2 Design of the Sequence Controller

As mentioned in Section 4.2.2, the specifications can be satisfied and involved in the sequence controller in 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 Fig. 4.6, which consists of 26 places and 20 transitions, respectively. Corresponding notations are described in Table 4.1. Transitions drawn with dark symbols are events that are controllable by remote clients via the Internet.



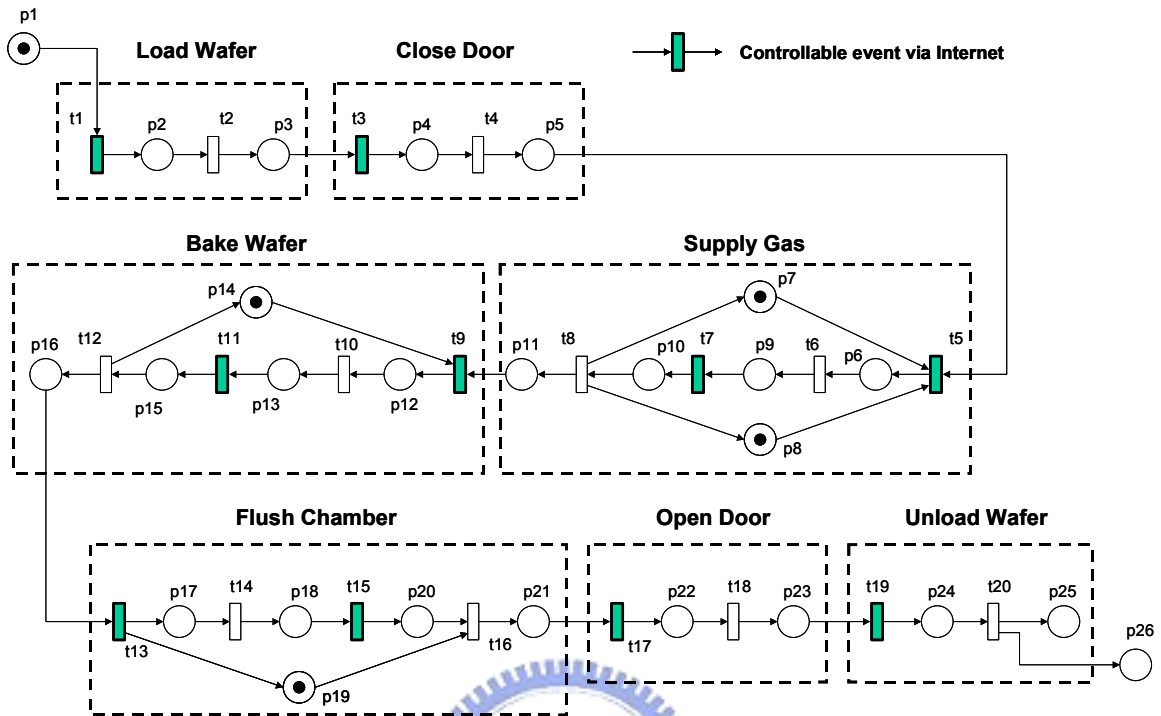


Fig. 4.6. The PN model for automatic control of the RTP system.

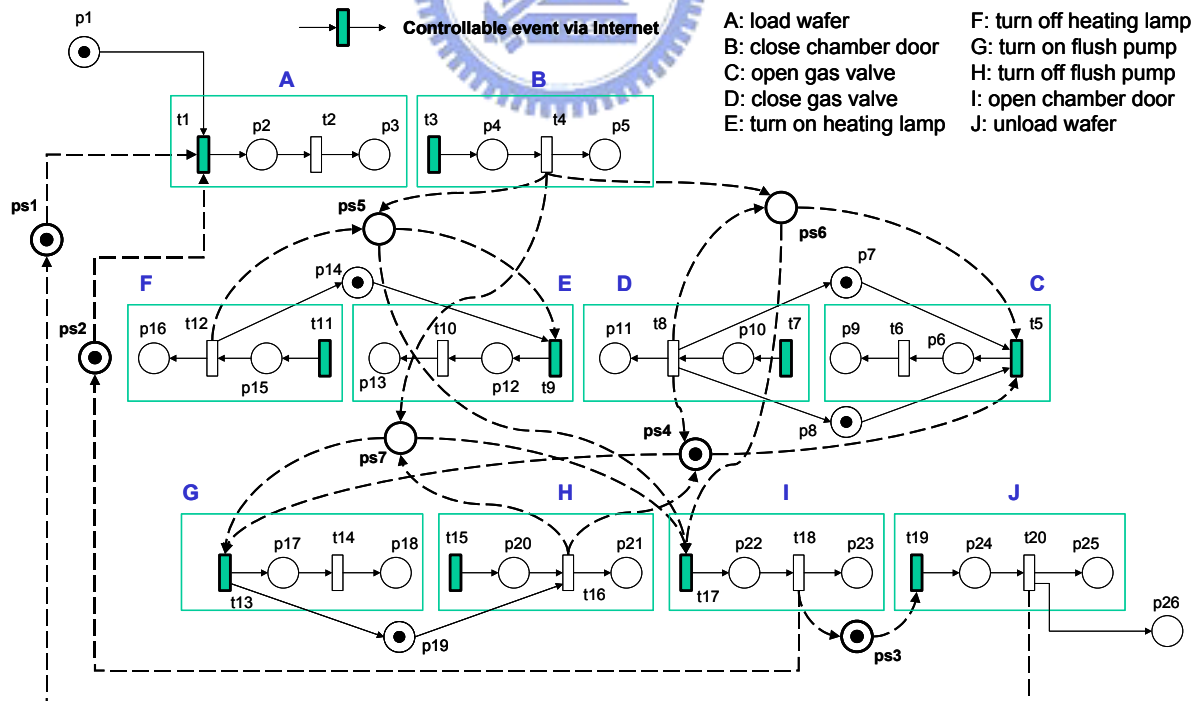


Fig. 4.7. The composed PN model for manual control of the RTP system.

Table 4.1. Notations for the PN of the RTP system in Fig. 4.6.

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		

4.4.3 Design of the Supervisor

For manual control mode, the plant model is formed by unconnecting each pair of transitions for the tasks in Fig. 4.6. 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 Fig. 4.7, 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 4.2. 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).

Table 4.2. Notations for supervisory places of PN in Fig. 4.7.

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

4.4.4 Implementation with Java Technology

The system modeling and design developed in previous stages provide supervisory control models for implementation of the present remote monitoring and control technology. To implement the supervisory control, we use Java due to its object-orientation, portability, safety, and built-in support for networking and concurrency (Hoshi, 1999; Bertolissi and Preece, 1998). 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 Fig. 4.8, 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. Fig. 4.8 also shows that the system is in automatic control mode, and thus only the **Auto-Control** button has been enabled by the supervisory agent. The human operator can only push this button which starts automatic process control by the sequence controller.

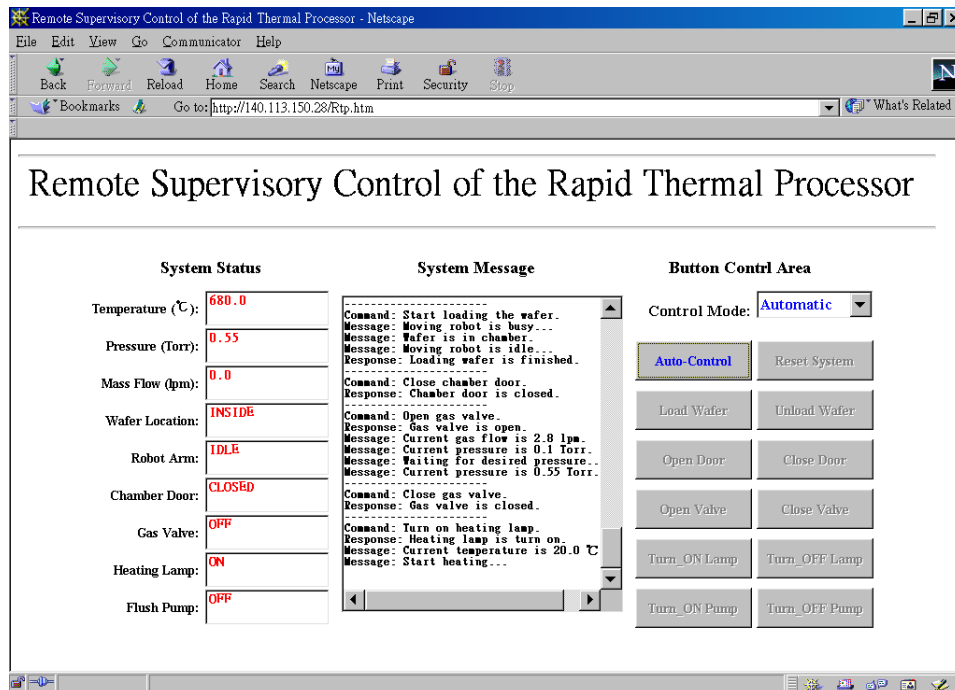


Fig. 4.8. Interactive web page for remote control of the RTP system by a Java applet (only **Auto-Control** button is admissible in the automatic control mode).

Fig. 4.9 shows the web pages for manual control mode after the **Open Valve** button has just been pushed (Step 3 in Section 4.4.1). 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. Fig. 4.10 shows the hardware setup during prototype development.

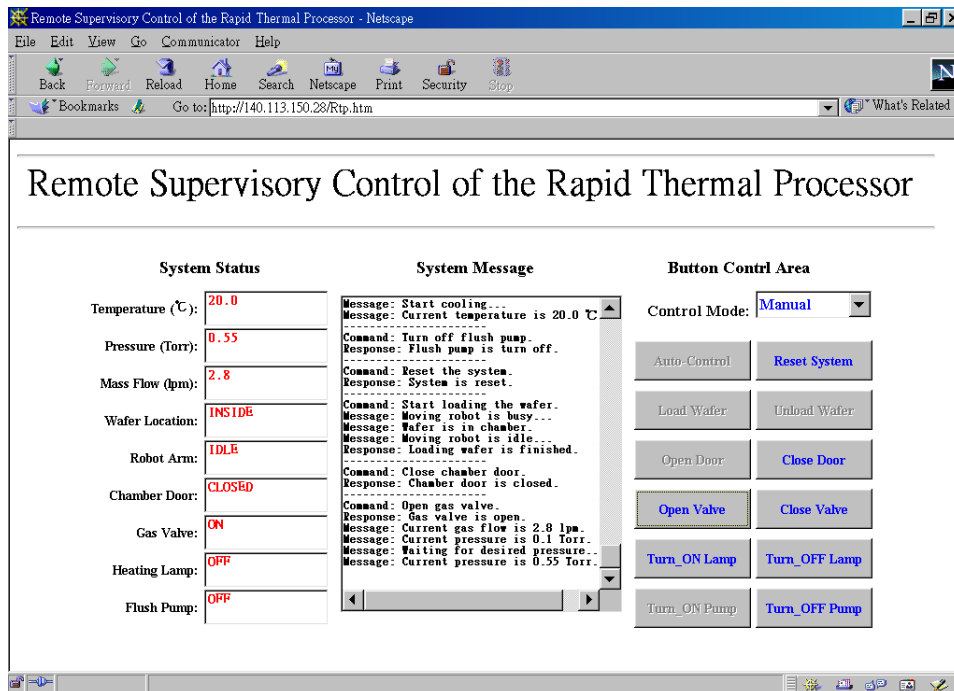


Fig. 4.9. Interactive web page in manual control mode at Step 3 of RTP processing (seven buttons are enabled).

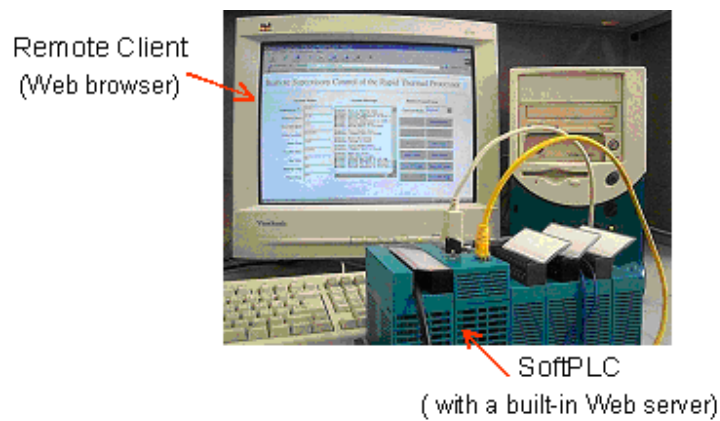


Fig. 4.10. The hardware setup during prototype development.

4.5. Summary

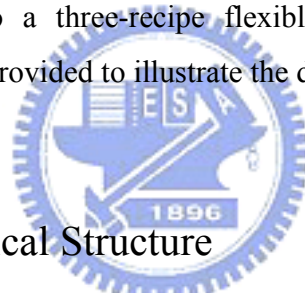
This chapter presents a framework for designing and implementing a PN-based supervisor for Internet-based control systems with the human in the loop. The supervisor is systematically designed by applying the mutual exclusion concept and is then implemented using the Java technology. To demonstrate the practicability of the proposed remote supervisory approach, an application is provided in which an simulated RTP system with an industrial PLC is controlled over the Internet. According to the feedback status of the remotely located system, the developed supervisor provides allowable commands for human operators while disabling operations that violate safety specifications.



Chapter 5

Hierarchical Supervision for Manufacturing Systems

In the previous chapter, a supervisory structure has been proposed to prevent the abnormal human commands from being carried out for remote control systems. However, the supervisor synthesis algorithm has computational complexity that is exponential in the state-space size of the system. In addition, communication delays and packet loss in the Internet are unavoidable. This chapter proposes a hierarchical supervisory scheme resulting in a smaller state-space size in supervisor synthesis. Moreover, fewer packet transmissions are required so that the effects of time delays and packet loss could be moderated. An application to a three-recipe flexible manufacturing system (FMS) controlled over the Internet is provided to illustrate the developed approach.



5.1. Proposed Hierarchical Structure

Hierarchical control is a familiar approach to the design of large-scale DES in order to reduce design complexity (Zhong and Wonham, 1990; Wong and Wonham, 1996; Tittus and Lennartson, 1999; Charbonnier et al., 1999). This chapter applies such hierarchical scheme to design the supervision systems for remote-controlled processes. As shown in Fig. 5.1, 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 manager is only allowed to issue the enabled tasks, and the hierarchical loop is closed in this way.

As compared with the traditional scheduling and planning architecture for manufacturing systems (Gershwin, 1989), the proposed hierarchical scheme specifically designed by applying the virtual control concept is more suitable for the remote supervision. Moreover, the proposed supervisor guarantees that remote human-issued commands lead to normal operations without deadlocks. In addition, as compared with direct remote control of each step (Kress et al., 2001), the proposed approach not only guarantees deadlock-free operation, it also moderates the effects of time delays and packet loss across the Internet since fewer packet transmissions are needed to complete a task.

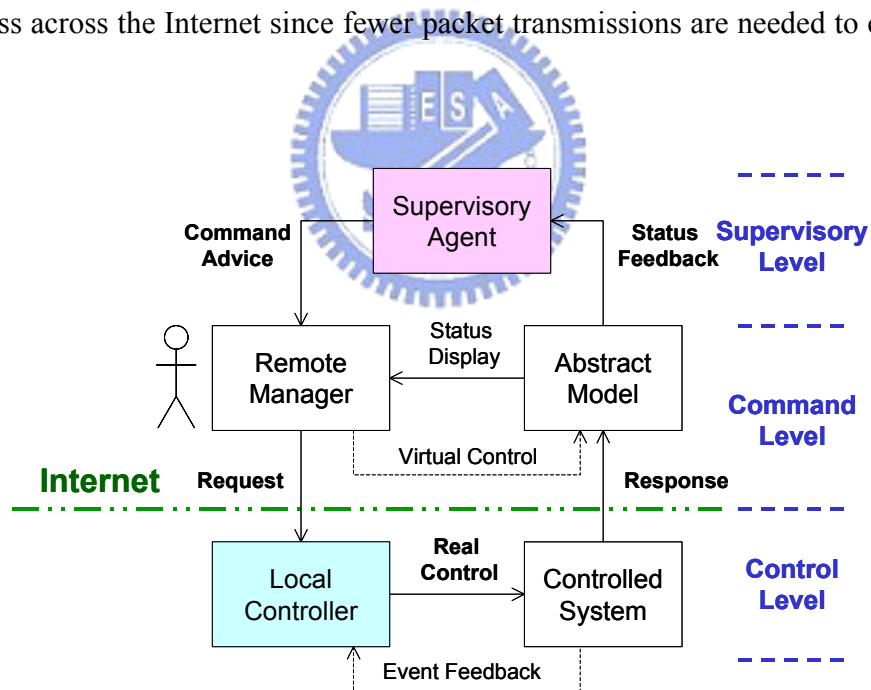


Fig. 5.1. Proposed three-level architecture for hierarchical supervision.

5.2. Design of the Hierarchical Supervision System

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

5.2.1 Multi-Recipe Processes

For multi-recipe systems with parallel or concurrent activities, each recipe describes a number of alternative desired paths through the plant. A recipe specifies the sequence of tasks to be executed and all possible ways the plant can be utilized in order to produce the desired product. Note that our recipe definition here corresponds to the master recipe in the batch control standard, ISA-S88.01 (ANSI/ISA, 1995). The master recipe is that level of recipe that accounts for equipment capabilities and may include process cell-specific information. It is thus natural to view a recipe as a specification on the plant to exhibit a certain task-sequence. However, there can be several independent recipes using the plant simultaneously, and all of these together form a non-deterministic joint specification on the overall system behavior. Since more than one recipe may be required to access the same resource, and each resource can only serve one recipe at a time, deadlock between different recipes may thus occur. The remote control problem then is to design a system that:

- 1) coordinates the resources for different recipes in order to ensure that the specified tasks in all recipes are executed correctly without deadlock occurring, and
- 2) regulates the execution of each task in detailed operations.

5.2.2 Separated Specifications

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 (Fanti et al., 2000). 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 non-deterministic 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, it also makes the system design more flexible, since it avoids the need to redesign the local controller, as only the command-level specification varies.

5.2.3 Design of the Supervisor

In this chapter, we first 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:

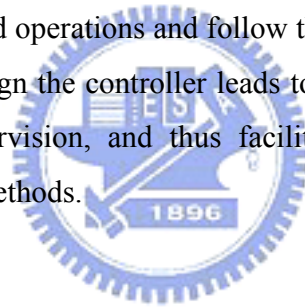
- Step 1) Construct the Petri-net model of the recipe specifications in command level using the task-oriented approach.
- Step 2) Build the Petri-net model of the resource specifications using the mutual exclusion concept.
- Step 3) Compose the recipe and resource models to yield the basic supervisor 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.

5.2.4 Design of the Local Controller

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.



5.3. Example: A Three-Recipe Flexible Manufacturing System

5.3.1 Description of the System

Fig. 5.2 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 FMS is a multi-recipe system with three recipes for three different products described as follows:

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

Recipe 2) *Product x-z*: Load materials x to Machine 1 and z to Machine 2 for processing, and then convey x and z to Machine 3. After processing x-z in Machine 3, unload the product.

Recipe 3) *Product y-z*: Load materials y to Machine 1 and z to Machine 2 for processing, and then convey y and z to Machine 3. After processing y-z in Machine 3, unload the product.

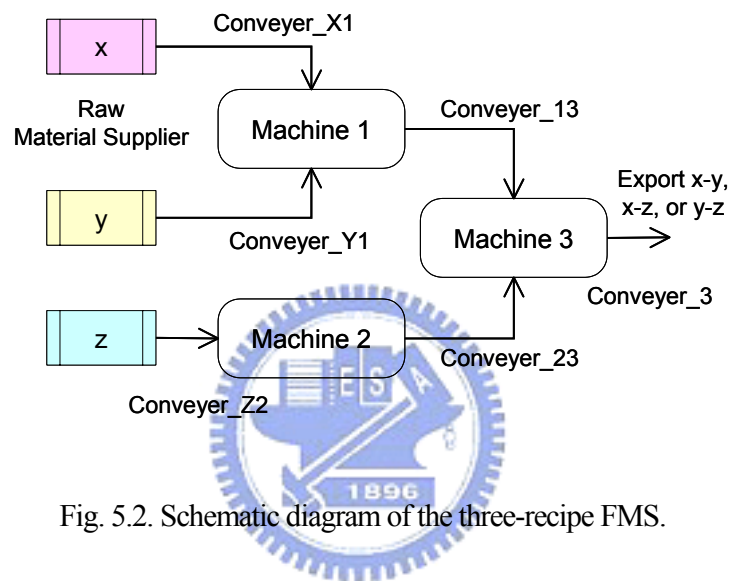


Fig. 5.2. Schematic diagram of the three-recipe FMS.

By applying the task-oriented concept, the PN model for the three recipes is constructed as shown in Fig. 5.3, which consists of 19 places and 22 transitions, respectively. Transitions drawn with dark symbols are events that are controllable by remote managers via the Internet. Corresponding notation is described in Table 5.1.

5.3.2 Design of the Supervisor

The three machines represent resources shared between the different recipes. Since more than one recipe may require access 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 Machine 1 is available.

Spec-2: Raw material loading of z is allowed only when Machine 2 is available.
 Spec-3: Material conveying to Machine 3 is allowed only when Machine 3 is available.
 Spec-4: Liveness, i.e. no deadlock states, must be enforced throughout system operation.

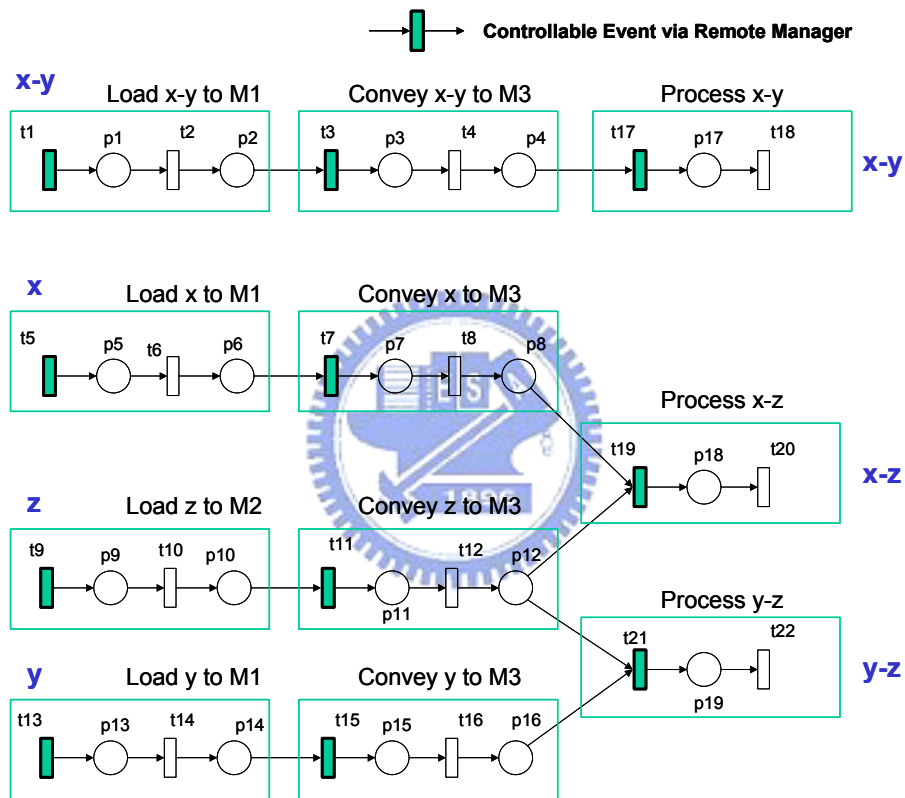


Fig. 5.3. Preliminary PN model of the three-recipe FMS.

Table 5.1. Notations for the PN of the FMS in Fig. 5.3.

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

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. 5.4. 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 5.2.

At this stage, the software package ARP (Maziero, 1990) is used again to verify the behavioral properties of the composed PN models. 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 Machine 2 and Machine 3 are occupied with z for Product $x-z$ or $y-z$, while Machine 1 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. 5.4. 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, while boundedness indicates that the system can be executed with limited resources, and reversibility implies 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.

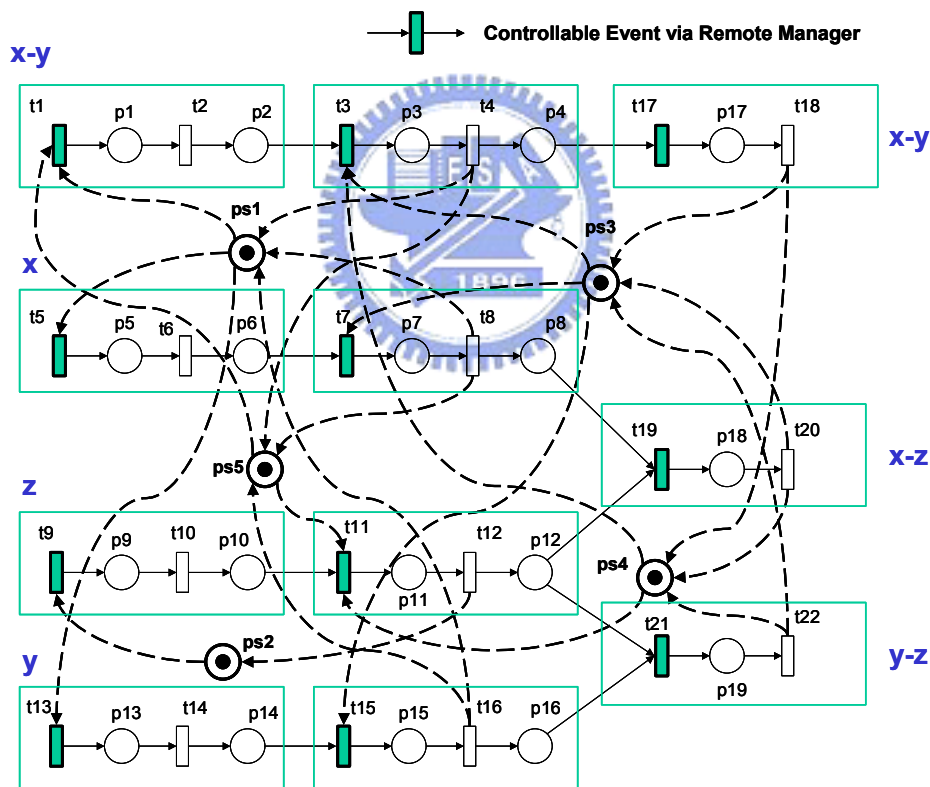


Fig. 5.4. Composed PN model of the three-recipe FMS.

Table 5.2. Notations for supervisory places of the PN in Fig. 5.4.

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 (2-bound)	Spec-4: One token means x-y is not in M1 and z is not in M3. Another means x or y is in M3.

5.3.3 Design of the Local Controller

As mentioned in Section 5.2.4, the detailed operations of each task can also be designed and constructed with PN models. Fig. 5.5 (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.

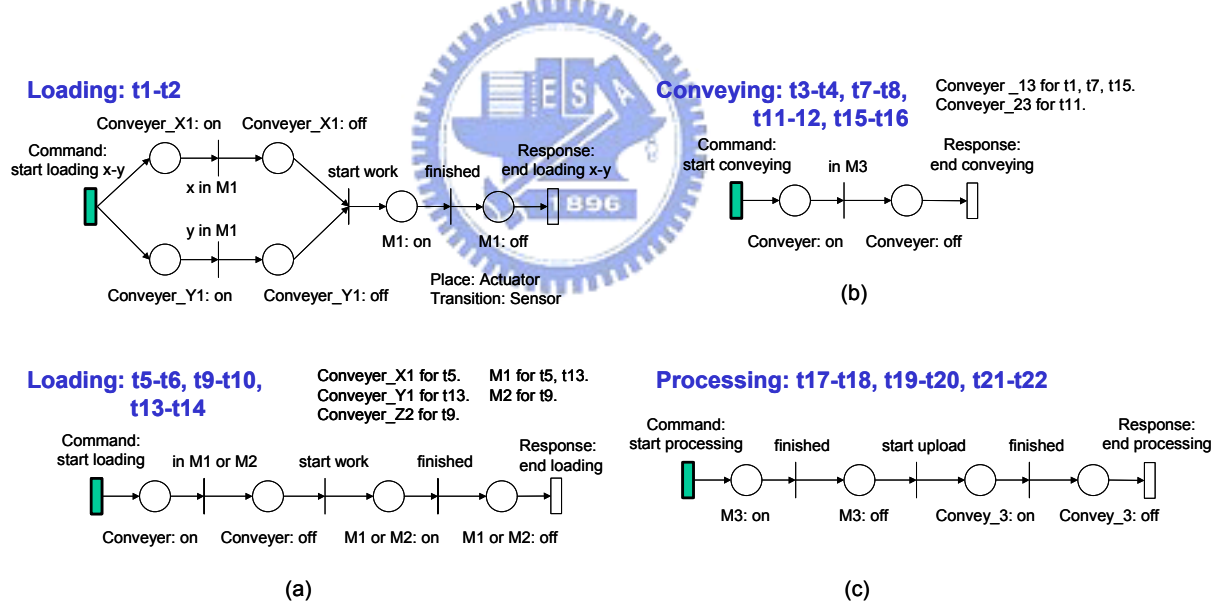


Fig. 5.5. PN models of (a) loading, (b) conveying, and (c) processing tasks for FMS.

5.3.4 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. Fig. 5.6 shows the developed HMI. By pushing the enabled buttons, the remote manager can issue commands to start tasks operated by the local controller. It also shows that Machine 1 is available, and both Machine 2 and 3 are occupied with material z (the pre-state of the mentioned deadlock in Section 5.3.2). In this situation, buttons **Load X to M1** or **Load Y to M1** are 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 buttons **Load X to M1** or **Load Y to M1** to generate 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.

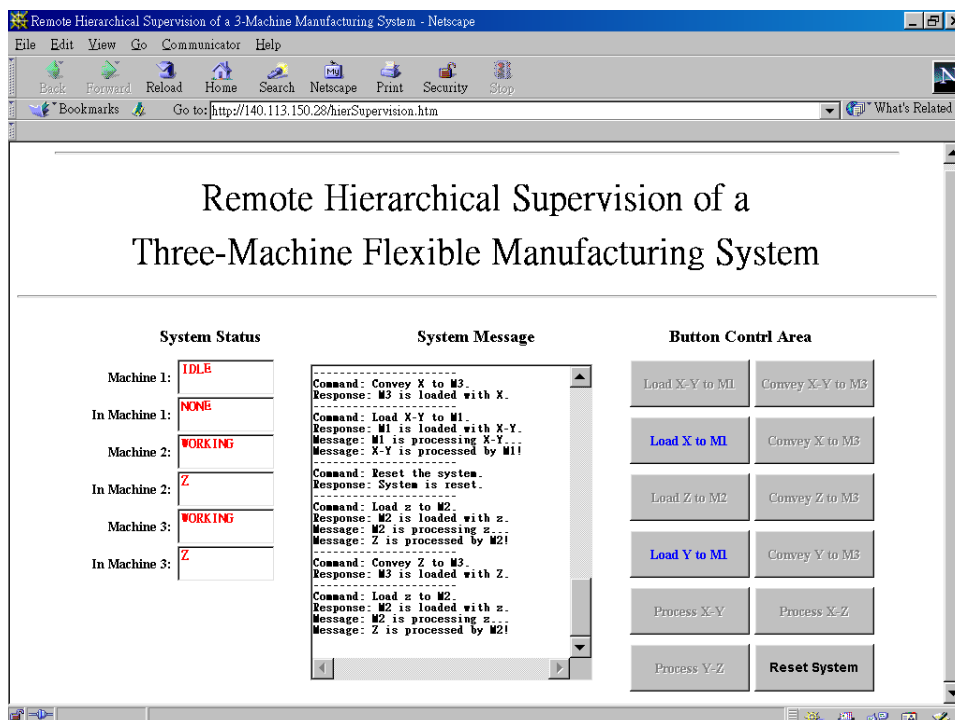


Fig. 5.6. Interactive Web page for remote supervision of the FMS by a Java applet (only three buttons are admissible).

5.4. Discussions

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 5.3, using a conventional nonhierarchical approach 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 supervisor design has a more compact model with a 248-state space.

Moreover, to produce thirty 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.

Table 5.3. Comparison between the nonhierarchical and hierarchical schemes.

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

5.5. Summary

This chapter has presented a unified Petri-net framework to design and implement a three-level hierarchical supervisory system for remote-controlled processes over the Internet. The supervisor in the upper level is systematically synthesized, using PNs, to enforce the command-level specifications of resource constraints and liveness for the processes, and then is implemented with Java 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. An application to a three-recipe FMS with an industrial PLC controlled over the Internet is provided to illustrate the proposed approach. According to the feedback status of the remotely located system, the designed Java-based supervisory agent guarantees that all requested commands from the remote 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 loss across the Internet can be moderated.

Chapter 6

SNMP-Based Management System

For large-scale and distributed systems, a management system is crucial to manage diverse network elements and handle their messages for remote supervision. One approach is to use the simple network management protocol (SNMP). However, in real industrial applications, many basic and major components such as sensors, actuators, and PLC still do not support SNMP function for remote applications. Therefore, this chapter presents a systematic design to embed SNMP agents into PLC for those devices so as to achieve remote monitoring and control through such a standard network protocol. Then, the standard unified modeling language (UML) is adopted for modeling the system, and the PN model is applied to analyze the dynamic behavior of the system. The developed system has been used successfully in a mobile switching center (MSC) of Taiwan Cellular Corporation for the remote supervision, through the Internet, to monitor and control its environmental conditions including the temperature, humidity, power, and security, with a total of 316 sensors and 140 actuators.

6.1. Integration of UML and PN

The UML is a language for specifying, constructing, visualizing, and documenting the elements of a software-intensive system (Booch et al., 1999). It defines the notation and semantics to describe systems using object-oriented and meta-modeling concepts in the spirit of the multi-paradigm modeling (Mosterman et al., 2004). Each model in the UML describes one aspect of a system, and the combination of the various models adequately describes the entire system. However, although UML is convenient for modeling a complex system, UML is not equipped with the necessary techniques for analyzing a system's qualitative and quantitative properties (Jeng and Lu, 2002). One of the major problems in using UML for the formal specification of systems is that the

semantics of UML are imprecise and vague. Particularly, the UML has no execution semantics and the current behavioral specifications in UML are primitive. UML also lacks tools and analysis support for behavioral models (Bernardi et al., 2002; Bordbar et al., 2000).

On the other hand, the PN is a graphical-mathematical tool used to model and analyze various systems, especially for systems with parallel and concurrent activities. PN provides qualitative analysis for system properties such as reachability, liveness, boundedness, and conservativeness. Moreover, by introducing time functions into the PN to form a timed PN, quantitative analysis can then be performed. PN complements the UML in a number of ways. First, it provides a powerful and rich visual formalization for specifying behavior in general, and concurrent behavior in particular. Second, it provides an executable notation, something that UML currently lacks. Statechart is the model that most closely resembles PN in the UML. However, Statechart describes state machines that are, in general, finite state systems whereas PN can be extended to present infinite state systems. Furthermore, PN has, in contrast to UML Statechart, dynamic representation (i.e. the token flow mechanism) and powerful analytical methods. This is why, in this chapter, the PN is adopted to obtain a dynamic and analyzable model for large-scale and long-distance distributed systems. With this approach, both qualitative and quantitative analyses can be applied to achieve reliable remote monitoring and control.

6.1.1 Design Procedures

A remote monitoring system consists of the agent and manager sides. The present approach develops SNMP agents based on the UML modeling with PN analysis. As shown in Fig. 6.1, the use-case diagram and sequence diagram in UML are used to capture the SNMP requirements corresponding to the monitoring and control specifications at the stage of functional and interactive analyses. Then, at the stage of static structural modeling, the class diagram is applied to describe the static relationships of the system. Subsequently, the PN model is constructed according to the above models such that both qualitative and quantitative analyses of the system's dynamic behavior can be performed. Finally, at the architectural design stage, the deployment diagram is

modeled to capture the physical relationships among software and hardware components, and the obtained models are implemented using Java and ladder diagrams on the industrial PLC. The design procedure in Fig. 6.1 is a type of ‘round-trip’ engineering, in which all models may be developed in an iterative and incremental way through a repeated cycle of analysis, design, implementation and testing. Therefore, the proposed approach is quite flexible and it allows making some alterations, such as changing the requirements or fixing a design flaw. A case study of an environmental monitoring system for the mobile switching center is provided in this chapter to illustrate the proposed approach.

6.2. Requirements of SNMP Agents

The SNMP is an application-level protocol that offers network management services in the transmission-control protocol/internet protocol (TCP/IP) suite. It is based on a client/server relationship in which the client issues requests to the server and the server processes requests and responds to the client. The SNMP network management system includes four key components: 1) management station, 2) management agent, 3) management information base (MIB), and 4) management protocol. A management station uses the management protocol to request management agents performing management operations on MIB objects. Essentially, each MIB object is a data variable that represents the manageable attribute. A management station can monitor and control remote elements by retrieving or changing the value of MIB objects of the management agent via the SNMP protocol. The management agent synchronously responds to requests from the management station and may further asynchronously provide important but unsolicited information (e.g. the alarm conditions) to the management station in the monitoring and control center.

Round-trip engineering

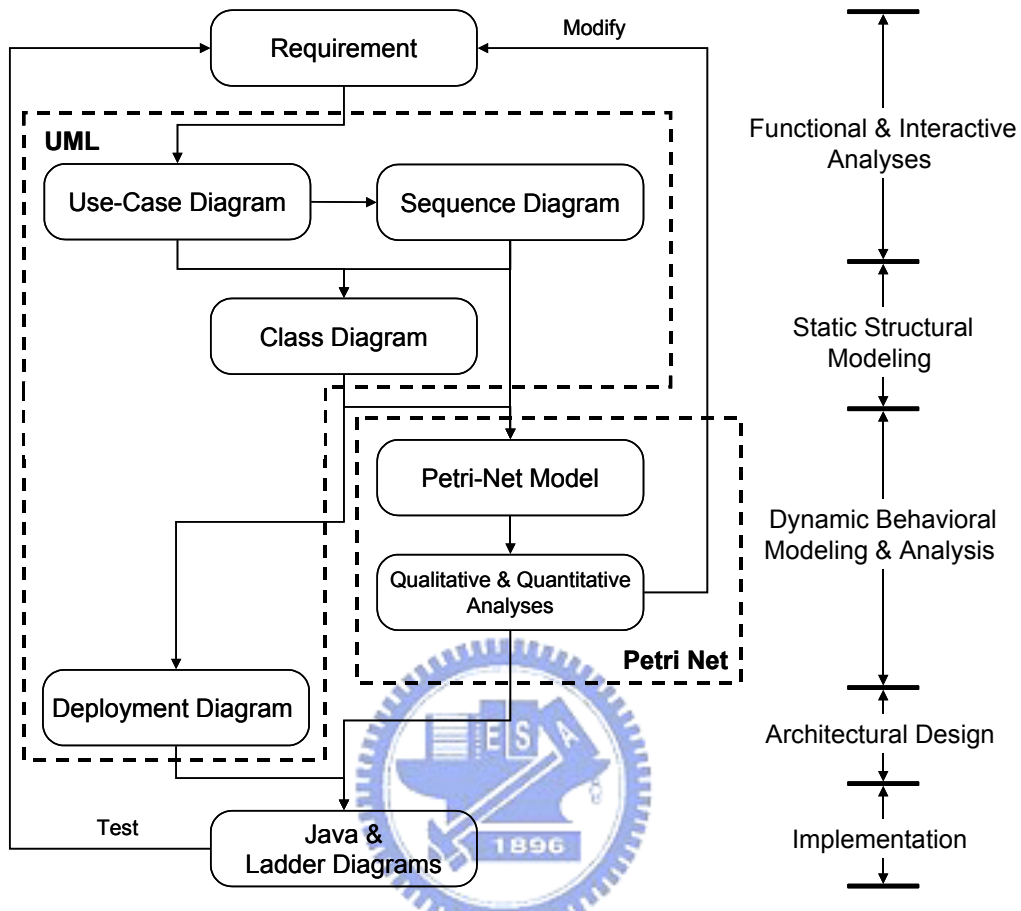


Fig. 6.1. The systematic development procedure for SNMP agents.

In the management station as shown in Fig. 6.2, three basic types of SNMP messages are issued on behalf of a management application:

- *GetRequest*
- *GetNextRequest*
- *SetRequest*

where the first two are variations of the get function. All three messages are transmitted with protocol data units (PDU) and acknowledged by the agent in the form of *GetResponse* message passed to the management application. In addition, an agent may issue a trap message in response to an event that affects the MIB and the underlying managed resources. Since SNMP relies on user datagram protocol (UDP) which is a

connectionless protocol and has high transmission efficiency for small data packets, SNMP is itself connectionless. No ongoing connections are maintained between a management station and agents.

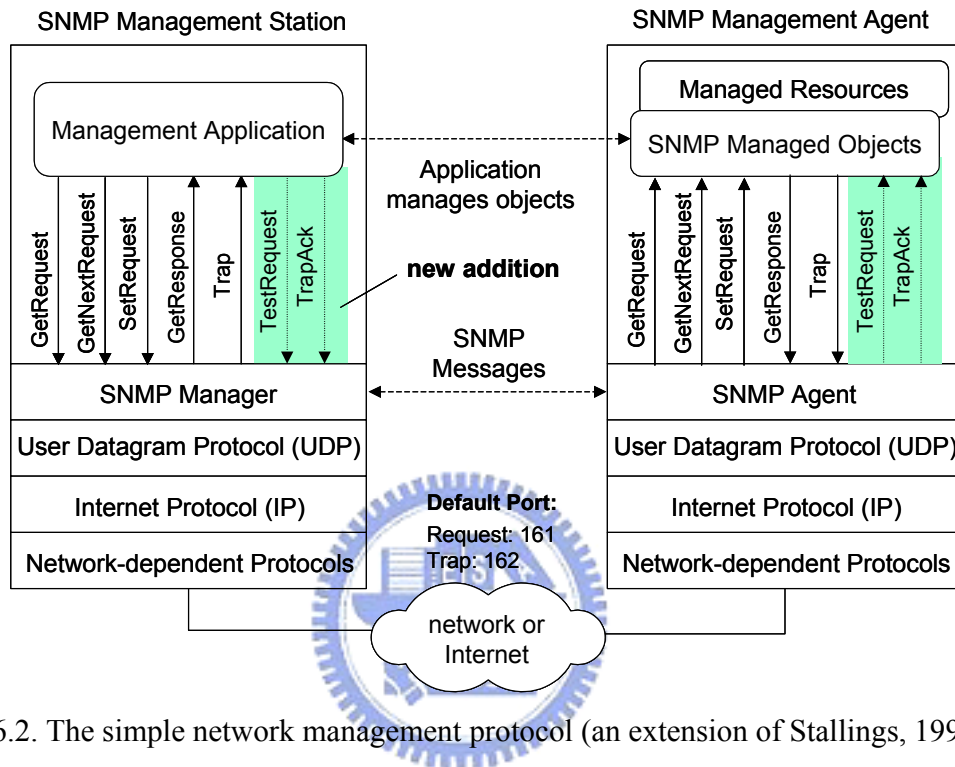


Fig. 6.2. The simple network management protocol (an extension of Stallings, 1993).

Moreover, in the standard SNMP, since traps from the agent are not acknowledged by the manager, there must be a mechanism to ensure that conditions in devices requiring attention are not missed. Therefore, we further design and implement the following two messages based on *SetRequest* to respond to traps:

- *TestRequest*
- *TrapAck*

When an alarm condition occurs, the designed SNMP agent will send the corresponding trap message to the manager periodically. The *TestRequest* message is used to check the alarm conditions in order to avoid false alarms, while the *TrapAck* message is used to confirm alarms. When an alarm is reported to the manager, the manager may use *TestRequest* to reset the alarm. If the physical input for such an alarm is still high, the same alarm trap message will be sent again. On the other hand, after an alarm trap is sent,

the manager may use the *TrapAck* message to confirm the alarm and the SNMP agent will then be disabled to send the same trap message periodically.

Two major advantages are obtained due to the utilization of SNMP for remote monitoring and control as follows.

- 1) De-localization of the monitoring stations: the management stations can be arbitrarily located anywhere through the Internet. Also, integration of a large number of monitoring devices in a given station becomes possible.
- 2) Ease of Access: the remote manager can access the local industrial devices easily via the standard SNMP protocol.

6.3. UML-Based Modeling for SNMP Agents

In the proposed approach, UML modeling and PN analysis are used to develop SNMP agents for remote monitoring and control. Then, the Java language and ladder diagrams are adopted to implement the system on an industrial PLC practically.

6.3.1 Functional Analysis with the Use-Case Diagram

A use-case diagram is used to capture the basic functional requirements of the system. As shown in Fig. 6.3, it consists of three actors and nine use cases. The actors, drawn as stick figures, represent users and other external systems that interact with the described system. The use cases, drawn as ellipses, represent the scenarios of the system. A scenario is a sequence of steps describing interaction between a user and a system. Basically, an *SNMP Manager* can perform the following five use cases:

- *GetRequest*
- *GetNextRequest*
- *SetRequest*
- *TestRequest*
- *TrapAck*

where *GetNextRequest* is an extension of *GetRequest*; *TestRequest* and *TrapAck* are specialized from *SetRequest*. Any one of the above five requests will cause the *SNMP*

Agent to carry out *HandleRequest*, including *GetResponse*, to result in a response to the request. On the other hand, as soon as *Managed Device* lies in the *AlarmCondition*, the *SNMP Agent* will perform *SendTrap* to report the alarms. Then, the *SNMP Manager* can carry out *TestRequest* to check the alarm conditions in order to avoid false alarms, and may perform *TrapAck* to confirm the alarm and then take the necessary control actions.

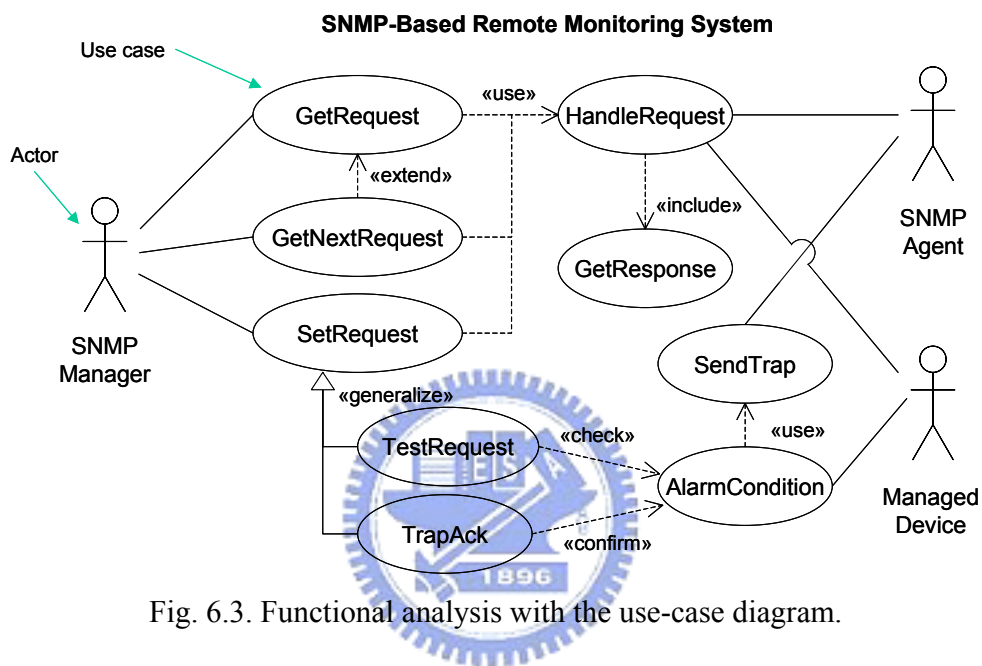


Fig. 6.3. Functional analysis with the use-case diagram.

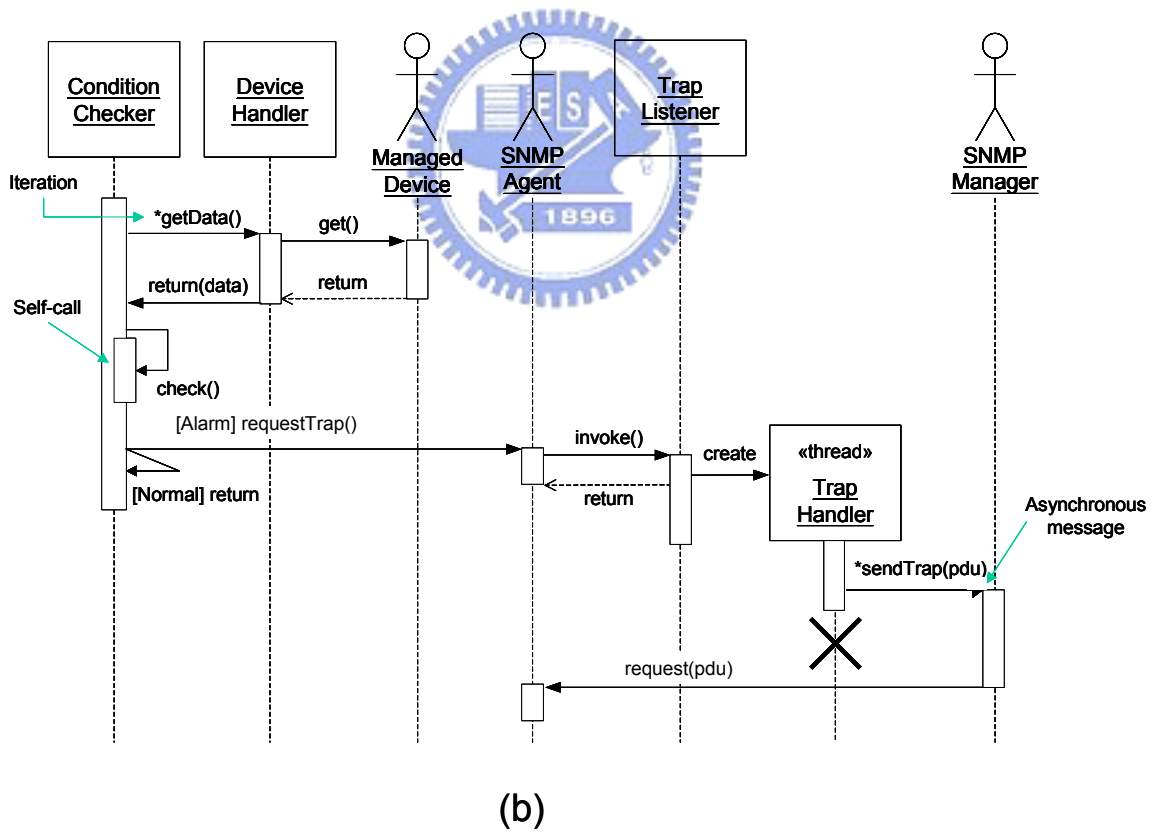
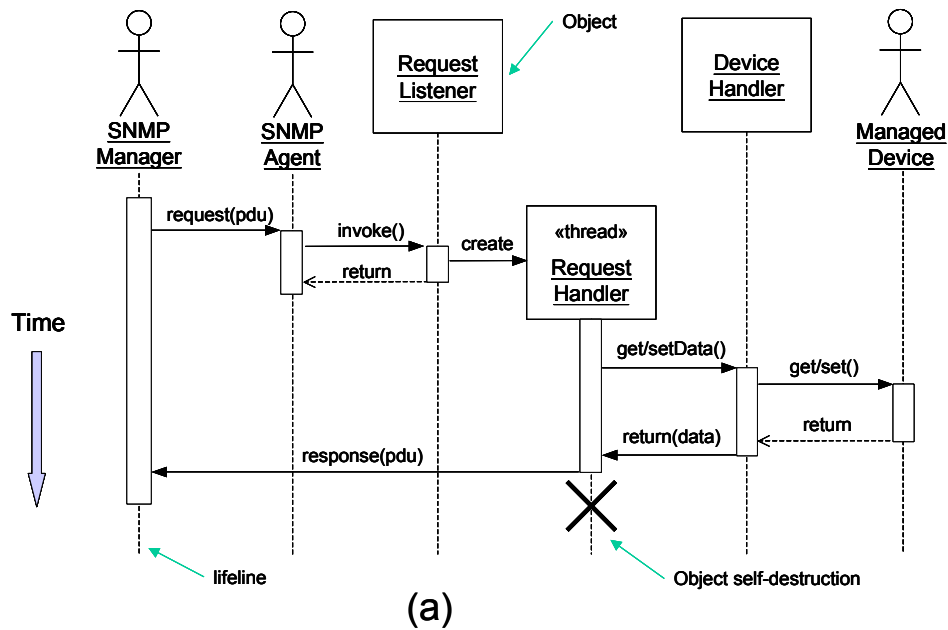


Fig. 6.4. Interaction analysis with the sequence diagrams for (a) the *Request* scenario and (b) the *Trap* scenario.

6.3.2 Interactive Analysis with the Sequence Diagram

A sequence diagram shown in Fig. 6.4 is used to model the object interaction in a system. Whereas the use-case diagram enables modeling of scenarios, the sequence diagram indicates details of the scenario including the objects and classes used to implement the scenario and messages passed between objects. Within a sequence diagram, an object is shown as a box at the top of a vertical dashed line, called the object's lifeline, 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. 6.4 (a) shows the sequence diagram for the *Request* scenario, which includes the five types of requests (*GetRequest*, *GetNextRequest*, *SetRequest*, *TestRequest*, and *TrapAck*) described in the use-case diagram in Fig. 6.3. At the first stage, the *SNMP Manager* may send a request to the *SNMP Agent*. Then, the *SNMP Agent* will invoke the *Request Listener* to create a threaded object, *Request Handler*, to carry out the request. The *Request Handler* then performs the specified actions on the *Managed Device* through the *Device Handler*, and then sends a response to the *SNMP Manager*. After finishing the request, the threaded object *Request Handler* will delete itself so as to release resources for the system.

For the *Trap* scenario as shown in Fig. 6.4 (b), the *Condition Checker* iteratively scans the status of the *Managed Device* through the *Device Handler* and checks its condition (the asterisk indicates the iteration in UML). If the condition is undesirable or faulty, *Condition Checker* will send a *requestTrap* message to the *SNMP Agent*. Then, *SNMP Agent* will invoke the *Trap Listener* to create a *Trap Handler*, a threaded object which carries out the request. The *Trap Handler* sends the trap to *SNMP Manager* asynchronously (the half-arrowhead symbol indicates an asynchronous message in UML), and then deletes itself to release the resources for proceeding use. When *SNMP Manager* receives the trap message, it will send a request of *TestRequest* to check the alarm condition, or perform *TrapAck* to confirm the alarm.

6.4. Example: A Mobile Switching Center

In wireless cellular communication systems, the service area is generally covered by many cells with base stations, and the clusters of cells are connected to mobile switching centers (MSCs). Each MSC receives encoded speech and data packets transmitted from the traffic channels in the base stations and provides call control, processing, and access to the public switched telephone network (Vucetic and Kline, 1998). Since the remote MSC plays an important role in mobile communications, the environmental conditions, emergency management, and safety of such large-scale and long-distance distributed systems are essential considerations. In the present design, an SNMP-based remote monitoring and control system, as shown in Fig. 6.5, is developed to provide real-time data on device status and environmental conditions in the MSC. Also, the embedded SNMP agents detect abnormal conditions in the MSC and report alarms to three de-localized management stations. Furthermore, necessary control actions may be taken through the Internet.

We choose a building complex as our target system. In this system, 24 temperature sensors, 24 humidity sensors, 4 power sensors, 4 current sensors, 4 voltage sensors, and 256 binary sensors for security (e.g. burglar alarms) are connected to two PLCs in the MSC to be monitored. Twelve alarm conditions are considered in the present monitoring system:

- Fire alarm
- Wateriness alarm
- Burglar alarm
- Temperature alarm
- Humidity alarm
- Electric voltage alarm
- Electric current alarm
- Power equipment alarm
- Power supplier alarm
- Dynamo alarm
- Uninterruptible power supply (UPS) alarm

- Air conditioner alarm

Moreover, six control actions can be operated remotely if specific alarm signals are issued:

- Emergency door control (open/close)
- Dynamo control (power on/off)
- UPS control (power on/off)
- Air conditioner control (off/wind/low/middle/high)
- Setting limitations of temperature and humidity
- Enable/disable alarms

Under normal operation, air conditioners are locally controlled to achieve desirable temperature and humidity within the specified ranges. As faults occur and are detected, corresponding control actions are taken by a total of 140 actuators. The actions that can be performed in the present remote monitoring and control system include 1) open emergency door, 2) adjust air conditioner, 3) power on dynamos, and 4) power on UPSs. Moreover, the hardware specifications provide three management stations and two PLC controllers for safety in case of crashes among local agents and remote managers.

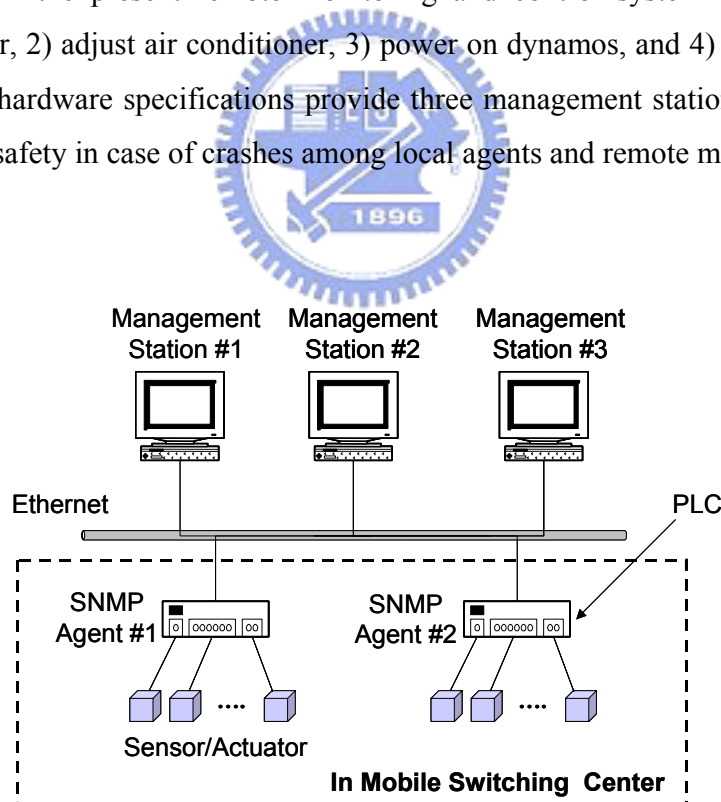


Fig. 6.5. The SNMP-based remote monitoring and control system.

6.4.1 Static Structural Modeling

The class diagram shown in Fig. 6.6 provides the main static structural models of the system. It is developed using information collected in the use-case diagram and sequence diagram discussed in Section III. A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. It also shows the attributes and operations of a class and the constraints on how objects are connected.

Fig. 6.6 is a class diagram of the SNMP-based monitoring and control system. It represents the static structure and object relations of SNMP agents for remote monitoring and control of the MSC. The *SnmpManager* class has five operations corresponding to the five types of requests as depicted in the use-case diagram. The *SnmpAgent* class has the composition relation (represented as a black diamond) with three classes: *RequestListener*, *TrapListener*, and *ConditionChecker*. The composition relation indicates that the composite is explicitly responsible for the creation and destruction of the contained objects. *RequestListener* can create a *RequestHandler*, which has five operations for the five types of requests, in order to process the request and respond to the *SnmpManager*. *TrapListener* may create a *TrapHandler*, which gets the IP addresses of trap managers, sets the hosts, ports of trap managers, and sends the *Trap* to report alarms to trap managers. The *ConditionChecker* uses the *DeviceHandler* to access the managed devices through the *DataTable* which reflects the real I/O status of managed devices and saves system variables, such as *MIB* mapping information and required limits (e.g. limitations as to temperature and humidity).

After real-time status checking, *ConditionChecker* obtains either the *Normal* or *Alarm* condition. As noted in Fig. 6.6, the *Alarm* object has twelve sub-objects, such as *FireAlarm*, *WaterinessAlarm*, etc. As soon as an alarm condition occurs, *SnmpAgent* is requested to create a *TrapHandler* to send a trap to the managers. The *MgdDevice* has a generalized relation with the *Sensor* and *Actuator*. In the present case, the remote controllable actuators are emergency doors, dynamos, UPSs, and air conditioners. In addition, certain system variables such as limitations on temperature and humidity can be set remotely, and all alarms can also be remotely enabled and disabled. The *Sensor* class is 'inherited' by the *BinarySensor* and *AnalogSensor*, the latter of which includes *TemperatureSensor*, *HumiditySensor*, etc. The class diagram can be developed and

modified in an iterative fashion, through a repeated cycle of analysis, design and implementation, and then returning to the first stage of the cycle, as shown previously in Fig. 6.1.

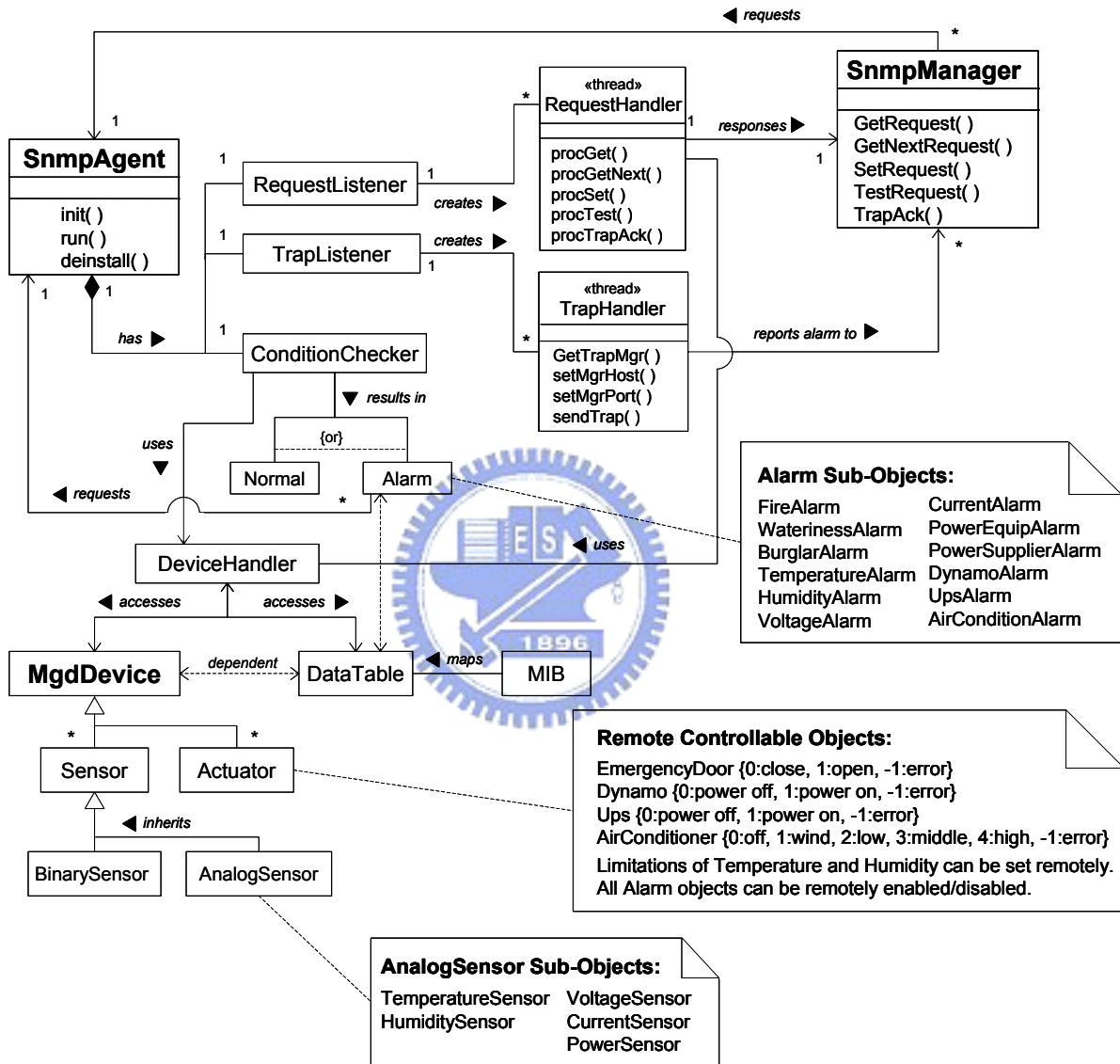


Fig. 6.6. The class diagram of the SNMP-based monitoring and control system.

6.5. PN Modeling and Analysis

In order to obtain a verifiable dynamic model for real applications, we use the PN model replacing the Statechart in UML. This allows us to perform both qualitative and quantitative analyses on the developed remote monitoring and control system.

6.5.1 Dynamic Behavioral Modeling

Based on the sequence diagram and class diagram constructed using UML, information can be extracted to build a PN model. The simplified PN of the remote monitoring and control system for the mobile switching center is shown in Fig. 6.7. It consists of 30 places and 28 transitions. Corresponding notations are described in Table 6.1. For example, the dynamic behavior of the *RequestHandler* in Fig. 6.4 (a) and Fig. 6.6 is modeled as p9-p15 and t6-t13 in Fig. 6.7. The software package ARP (Maziero, 1990) is adopted again to verify the qualitative and quantitative properties of the PN model.

6.5.2 PN Analysis

In our qualitative analysis, validation results via the PN modeling show the present design to be 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 facilities (e.g., limited request buffer size). For quantitative analysis, appropriate parameters such as the time period and the probability of an alarm occurring are assigned for the timed PN modeling. Simulation results show that t1, t12, t13, and t25, drawn with dark symbols in Fig. 6.7, are critical timed transitions of the system. These critical time delays are dependent on the transmission rate between the manager and agent. For example, if the data rate on the line is 512K bps, i.e. 64K characters per second, then the delay is 1/64K second per character. Since the SNMP rides over UDP/IP, of which the maximum packet size is 64K, the delay will be 1 second if there is no significant network congestion. On the other hand, the delay time of t20 can be chosen to avoid sending a great number of traps to managers in a short time

interval for the same alarm condition. In our case, we choose a delay of 30 seconds for t_{20} . That means that if an alarm is reported to the manager but the agent does not receive an acknowledgement within 30 seconds from the manager (i.e. *TestRequest* or *TrapAck*), the designed agent will send the trap again for this alarm condition.

In addition to finding the critical timed transitions, the PN model can also be used to decide time periods, such as t_{14} (time period in which to scan the real I/O status) and t_{16} (time period in which to check the data in *DataTable*), by performing sensitivity analysis based on the p-invariant or static cycle methods (Zuberek, 2001; Srinivasan, 1998).



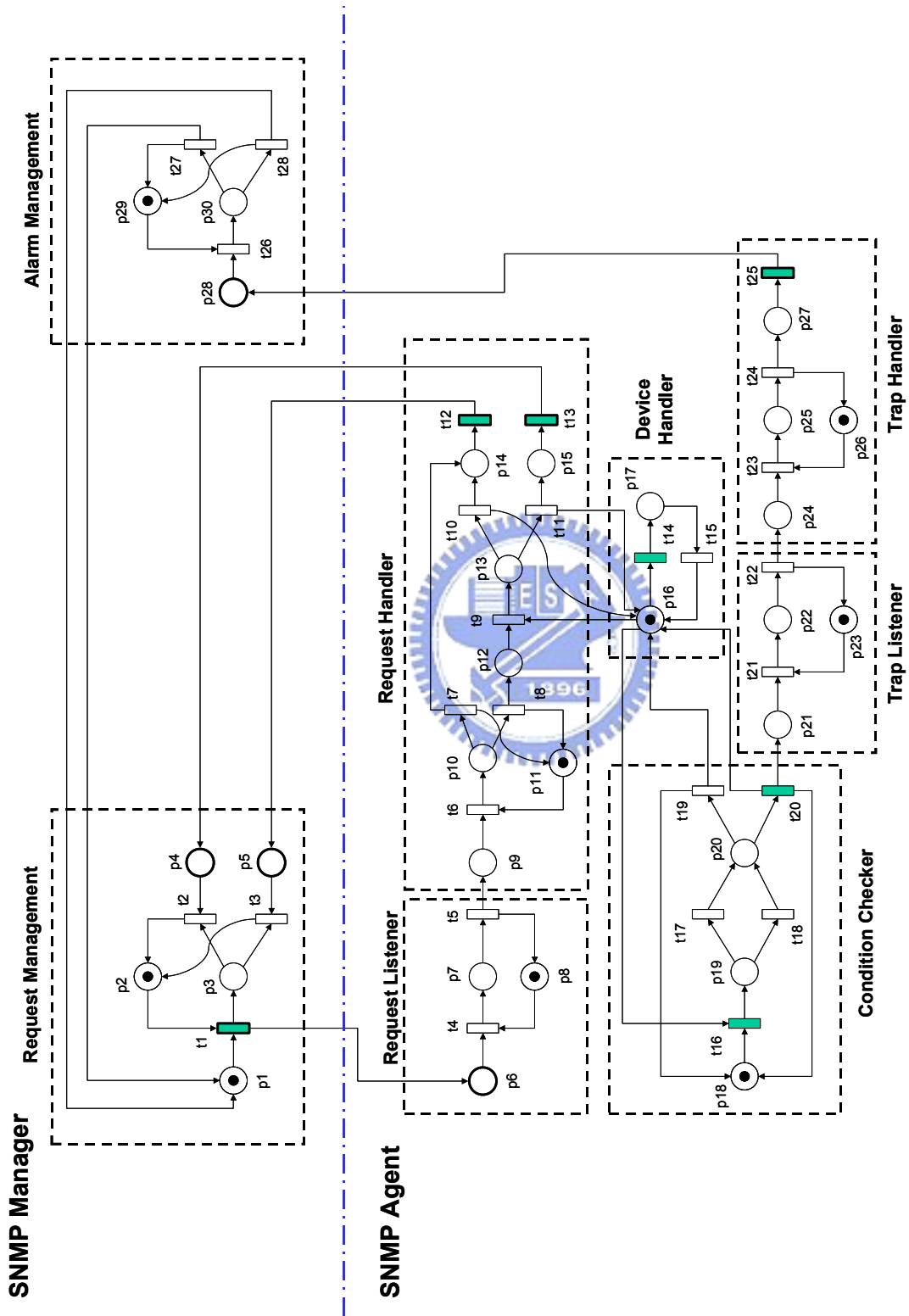


Fig. 6.7. The PNs of the SNMP-based monitoring and control system.

Table 6.1. Notations of the PN for the SNMP-based management system in Fig. 6.7.

Place	Description	Transition	Description
p1	New request	t1	Send request
p2	MIB browser ready	t2	Report result
p3	Waiting for response	t3	Report error
p4	Receiving result message	t4	Invoke request listener
p5	Receiving error message	t5	Finish request handler creation
p6	Request buffer	t6	Decode request PDU
p7	Creating request handler	t7	Error occurs
p8	Request listener available	t8	Finish decoding
p9	Request handler ready	t9	Start processing request
p10	Decoding request PDU	t10	Error occurs
p11	MIB of objects available	t11	End processing request
p12	Decoded commands and variables	t12	Send error message
p13	Processing request	t13	Send result message
p14	Collecting error message	t14	Access devices
p15	Collecting result message	t15	End handling devices
p16	Device handler available	t16	Start checking states
p17	Handling devices (get/set status)	t17	Normal level-condition
p18	Condition checker ready	t18	Abnormal level-condition (Trigger/Hold timer to generate impulse)
p19	Processing level-check	t19	Normal impulse-condition
p20	Processing impulse-check	t20	Abnormal impulse-condition
p21	Trap buffer	t21	Invoke trap listener
p22	Creating trap handler	t22	Finish trap handler creation
p23	Trap listener available	t23	Encode alarm to trap PDU
p24	Trap handler ready	t24	Finish encoding
p25	Encoding trap PDU	t25	Send trap to managers
p26	MIB of traps available	t26	Process trap
p27	Trap PDU ready	t27	Answer <i>TestRequest</i> (check alarm)
p28	Receiving trap message	t28	Answer <i>TrapAck</i> (confirm alarm)
p29	Trap browser ready		
p30	Deciding response for trap		

6.6. Architecture Design and Implementation

A deployment diagram is used to model the physical relationships among software and hardware components in the deployed remote monitoring and control system, as shown in Fig. 6.8. It includes a set of nodes (drawn as cubes) to represent the computational units and relationships among three main machines: (1) the management station, (2) management agent, and (3) managed devices. The management station uses the *SNMP Manager* to communicate with the *SNMP Agent* through an Ethernet connection, while the management agent uses the *Device Handler* to communicate with the managed devices such as sensors and actuators through PLC I/O connections or the industrial network Modbus.

The system modeling and analysis developed in previous stages provide standard models for implementation of the present remote monitoring and control technology. Although UML modeling is not restricted to any particular language in implementation, Java is preferred due to its object-orientation, portability, safety, and built-in support for networking and concurrency. In the implementation of the present design, we need to translate information from multiple UML and PN models into the code and database structure. This translation is not straightforward. However, there is a close correspondence between Java and UML, and a standard mapping is described in (Greenfield, 2001). Also, a mapping between PN and Java is described in (Conway et al., 2002). Moreover, since Java cannot directly control the I/O devices, the ladder diagram implemented on the PLC is applied to make the SNMP agent access the low-level sensors and actuators. The developed SNMP agent is implemented on the Mirle SoftPLC. Fig. 6.9 shows the hardware setup during prototype development.

The developed SNMP-based remote monitoring and control system in this chapter is now operating at an MSC belonging to Taiwan Cellular Corporation. A total of 316 sensors and 140 actuators are handled by two PLCs with 189 rungs in each ladder diagram. Under normal operation, the desirable temperature and humidity of the MSC are locally controlled by air conditioners and only remote monitoring is needed. As any faults occur in the MSC, the SNMP agents will immediately send alarm signals to the three remote management stations, and proper control actions will then be taken to

correct the faults. Thus, environmental conditions in the MSC are supervised by the local SNMP agents and can be further monitored and controlled by the remote manager from great distances through the Internet.

6.7. Discussions

This chapter integrates the PN into UML modeling to achieve design, modeling, analysis, verification, and implementation of remote monitoring and control systems within a systematic framework. The results of this study lead to the following discussion.

- 1) The models developed here for application to SNMP-based remote monitoring and control of mobile switching centers are general models. Since the UML is based on the object-oriented concept, reusable models can be grouped into a library to make the design process more efficient when similar SNMP applications are encountered.
- 2) Basically, if SNMP traps are allowed to go unacknowledged, SNMP agents cannot guarantee that a critical message definitely reaches the management station. In this chapter, *TestRequest* and *TrapAck* are further proposed to respond to the traps and thus, the present SNMP agents ensure that conditions requiring attention in the monitored systems or processes are not missed.

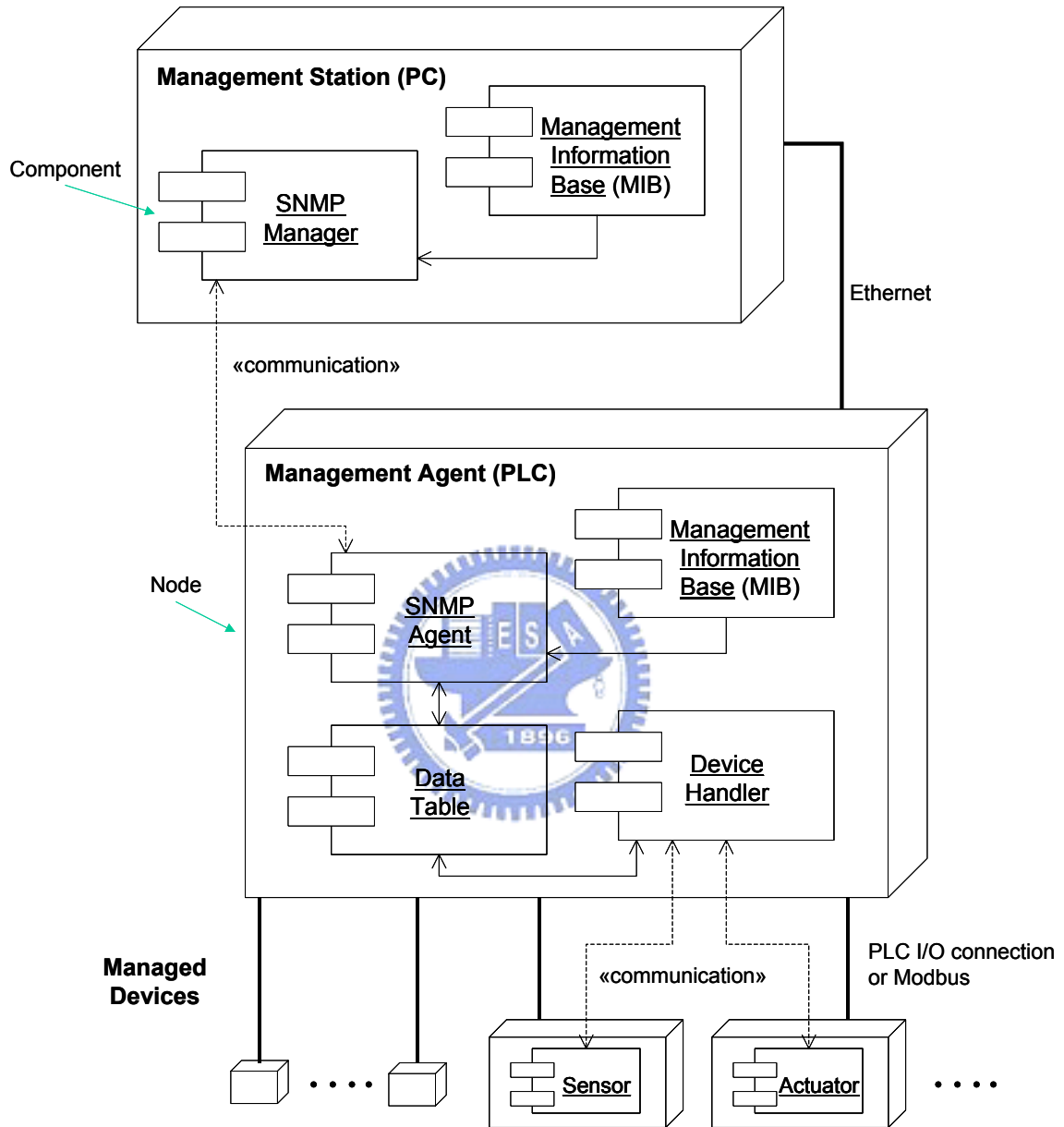


Fig. 6.8. Architectural design with the deployment diagram.

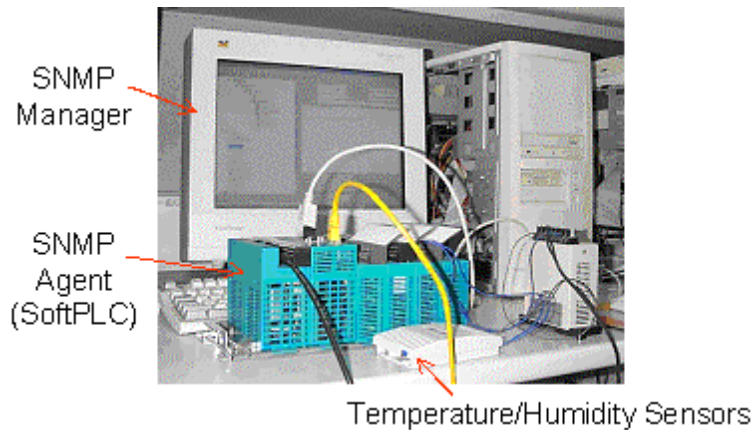


Fig. 6.9. The hardware setup during prototype development.

6.8. Summary

This chapter presents a systematical design and implementation of SNMP agents for device management systems. In the UML-based design of the SNMP agents, the use-case diagram and the sequence diagram are applied to describe the functionalities and interactions, respectively. Then, a class diagram is used to describe static structures, and the PN model is further applied to verify the dynamic behavior of the system. In addition, the deployment diagram is used to model the distribution of physical components in the system. Implementation is then accomplished using the Java language and ladder diagrams on the PLC. For the management of large-scale and distributed systems, the proposed multi-paradigm approach provides systematic design and implementation of SNMP agents to achieve remote monitoring and control by integrating UML modeling and PN analysis.

Chapter 7

CONCLUSION

7.1. Summary of Contributions

For remotely monitored and controlled processes, a series of design and implementation results of the sequence controller, the supervisor, and the device management system are proposed in this thesis. In the current e-automation world, the techniques developed in this thesis are useful for industrial applications. The contributions of this thesis are summarized into five aspects:

1) Rule-based evaluation of the ladder logic diagram (LLD) and Petri net (PN)

To verify the potential of PN in the sequence control applications, this work presents a rule-based comparison to adequately evaluate the LLD and PN. An example of five sequences with increasing complexity for a stamping process is provided to illustrate the proposed approach. The results indicate that the proposed evaluation approach is more reasonable (Lee and Hsu, 2004a).

2) PN-based design for LLD implementation

Since the LLDs are still widely used today in real industry projects, this thesis proposes a PN-based design to the final LLD implementation for sequence control. Starting from the basic sequential specification, the proposed approach combines integration definition language 0 (IDEF0), simplified Petri net controller (SPNC), and token passing logic (TPL), and systematically leads to the LLD for PLC implementation. An application of a stamping process is provided to illustrate the developed approach (Lee and Hsu, accepted).

3) Supervisory control of human behaviors

To prevent abnormal operations of humans, a remote supervisory

scheme is proposed so that undesirable human operations are prohibited. According to the feedback status of a remotely located system, the developed supervisory agent provides allowable commands for operators by disabling those operations that violate safety specifications. The possibility of human errors can be thus either reduced or fully eliminated. An example of rapid thermal processor in semiconductor manufacturing is provided to illustrate the proposed approach (Lee and Hsu, 2003b).

4) Hierarchical supervision of manufacturing systems

To reduce the complexity of supervisory system design, this thesis proposes a hierarchical structure to synthesize subsystems for remote monitored and controlled processes. A three-recipe flexible manufacturing system is also provided to illustrate the developed hierarchical design. The results show that the developed hierarchical design leads to a smaller state-space size. Also, fewer request/response transmissions are consumed resulting in less transmission faults (Lee and Hsu, 2003a).

5) Realization of simple network management protocol (SNMP)-based device management system

To manage diverse network elements, this thesis integrates the PN into the unified modeling language (UML) to achieve modeling, design, analysis, verification, and implementation of SNMP agents within a systematic framework. The developed system has been successfully used in a mobile switching center of Taiwan Cellular Corporation for the remote supervision and management of its various environmental devices (Lee and Hsu, 2004b).

7.2. Future Research

Through the study of applying the PN for remote supervision systems, there are several directions in which this work can be extended in the future as follows:

1) Time-based constraints

The discussed supervisory control framework in this thesis is restricted to purely logical system models (Giua and DiCesare, 1991; Moody and Antsaklis, 1998) For applications with time-based constraints (e.g. communication delays), it is necessary to extend the present model with time-related specifications (Cofer and Garg, 1996; Caramihai et al., 1998).

2) Automatic model transformations

This work provides the design approach by integrating IDEF0/SPNC/TPL/LLD to systematically achieve the sequence controller. Furthermore, the approach by applying the UML with PN is also employed to develop an SNMP-based management system. However, the model transformation between these two approaches is still achieved manually in the present study. Design of computer programs could be the future research to transform the models automatically (Mosterman et al., 2004).

3) Access security

Security is a prime concern for network systems with remote access and only basic user/password and IP-access policies are adopted in this thesis. Several solutions have been proposed for SNMP to improve the access-control policy, such as Secure-SNMP (S-SNMP) and SNMPv3 (Zeltserman, 1999). Improving the security of the present remote systems by applying the new SNMP policies is considered in the future implementation.

4) Multiple-user conditions

The remote control scheme presented in this thesis is focused on the condition of single-user access at a time. Future work should study the conditions of multiple-user access.

5) Error recovery mechanisms

For the remote supervision systems, the missing message and channel

disconnection are unavoidable in Internet. Moreover, process errors or device faults may also occur during the operations. Thus, error recovery mechanisms for the present remote supervision systems can be further investigated (Jeng, 1997; Zhou and Dicesare, 1989).



REFERENCES

- Aicklen, G. H., and Main, P. M. (1995), "Remote control of diverse network elements using SNMP," in *Proc. IEEE Int. Conf. Military Communication*, San Diego, CA, pp. 673-677.
- ANSI/ISA (1995), *S88.01: Batch Control Part 1: Models and Terminology*. Instrument Society of America.
- Balemi, S., Hoffmann, G. J., Gyugyi, P., Wong-Toi, H., and Franklin, G. F. (1993), "Supervisory control of a rapid thermal multiprocessor," *IEEE Trans. Automat. Contr.*, vol. 38, no. 7, pp. 1040-1059.
- Batur, C., Ma, Q., Larson, K., and Kettenbauer, N. (2000), "Remote tuning of a PID position controller via Internet," in *Proc. Amer. Contr. Conf.*, pp. 4403-4406.
- Bernardi, S., Donatelli, S., and Merseguer, J. (2002), "From UML sequence diagrams and Statecharts to analyzable Petri net models," in *Proc. ACM Int. Workshop Soft. Performan.*, Rome, Italy, pp. 35-45.
- Bertolissi, E., and Preece, C. (1998), "Java in real-time applications," *IEEE Trans. Nuclear Science*, vol. 45, no. 4, pp 1965-1972.
- Booch, G., Rumbaugh, J., and Jacobson, I. (1999) *The Unified Modeling Language User Guide*. Reading, MA: Addison-Wesley.
- Bordbar, B., Giacomini, L., and Holding, D. J. (2000), "UML and Petri nets for design and analysis of distributed systems," in *Proc. IEEE Int. Conf. Contr. Appli.*, Anchorage, AK, pp. 610-615.
- Boucher, T. O., Jafari, M. A., and Meredith, G. A. (1990), "Petri net control of an automated manufacturing cell," *Adv. Manuf. Engin.*, vol. 2, no. 3, pp. 151-157.

- Bradshaw, J. M. (1997), "Introduction to Software Agents," *Software Agents*, Bradshaw, J. M. Ed., Cambridge, MA: AAAI Press/MIT Press.
- Campione, M., and Walrath, K. (1998), *The Java Tutorial: Object-Oriented Programming for the Internet*. Second Ed., Reading, MA: Addison-Wesley. [Online]. Available: <http://java.sun.com/docs/books/tutorial/>
- Caramihai, S. I., Dumitrache, I., and Stanescu, A. M. (1998), "Real-time supervision for intelligent manufacturing supported by T-temporal Petri net models," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, San Diego, CA, pp. 582-587.
- Cardoso, P. F., and Monteiro, J. L. (1998), "SNMP and industrial networks," in *Proc. IEEE Int. Conf. Industrial Electronics*, Aachen, Germany, pp. 242-246.
- Charbonnier, F., Alla, H., and David, R. (1999), "The supervised control of discrete-event dynamic systems," *IEEE Trans. Contr. Syst. Technol.*, vol. 7, no. 2, pp. 175-187.
- Cofer, D. D., and Garg, V. K. (1996), "Supervisory control of real-time discrete-event systems using lattice theory," *IEEE Trans. Automat. Contr.*, vol. 41, no. 2, pp. 199-209.
- Conway, C., Li, C. H., and Pengelly, M. (2002), *Pencil: A Petri Net Specification Language for Java*. Mathematics Department, Macquarie University, Sydney. [Online]. Available: <http://www.cs.columbia.edu/~conway/plt/pencil/index.html>
- David, R., and 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.
- Fanti, M. P., Maione, B., and Turchiano, T. (2000), "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, (Special issue on discrete systems and control).

Feldmann, K., Colombo, A. W., Schnur, C., and Stockel, T. (1999a), "Specification, design, and implementation of logic controllers based on colored Petri net models and the standard IEC1131. I. Specification and design," *IEEE Trans. Contr. Syst. Tech.*, vol. 7, no. 6, pp. 657-665.

Feldmann, K., Colombo, A. W., Schnur, C., and Stockel, T. (1999b), "Specification, design, and implementation of logic controllers based on colored Petri net models and the standard IEC1131. II. Design and implementation," *IEEE Trans. Contr. Syst. Tech.*, vol. 7, no. 6, pp. 666-674.

Frey, G. (2000), "Automatic implementation of Petri net based control algorithm on PLC," in *Proc. American Control Conf.*, pp. 2819-2823.

Frey, G. and Litz, L. (2000), "Formal methods in PLC programming," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, Nashville, TN, pp. 2431-2436.

Gershwin, S. B. (1989), "Hierarchical flow control: A framework for scheduling and planning discrete events in manufacturing systems," *Proc. IEEE*, vol. 77, no. 1, pp. 195-208.

Giua, A., and DiCesare, F. (1991), "Supervisory design using Petri nets," in *Proc. IEEE Int. Conf. Decision Contr.*, Brighton, England, pp. 92-97.

Greenfield, J. (2001), *Unified Modeling Language/Enterprise JavaBeans (UML/EJB) Mapping Specification*. Rational Software Corporation Document.

Hoshi, T. (1999), "Current and future Java technology for manufacturing industry," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Tokyo, Japan, pp. 404-409.

- Huang, G. Q., and 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).
- Hunter, J., and Crawford, W. (1998), *Java Servlet Programming*. Sebastopol, CA: O'Reilly & Associates Inc.
- International Electrotechnical Commission (1993), *Programmable Controllers Part 3, Programming Languages, IEC 1131-3*. Geneva: IEC.
- Jeng, M. D. (1997), "Petri nets for modeling automated manufacturing systems with error recovery," *IEEE Trans. Robot. Automat.*, vol. 13, no. 5, pp. 752-760.
- Jeng, M. D. and Lu, W. Z. (2002), "Extension of UML and its conversion to Petri nets for semiconductor manufacturing modeling," in *Proc. IEEE Int. Conf. Robot. Automat.*, Washington, DC, pp. 3175-3180.
- Kress, R. L., Hamel, W. R., Murray, P., and 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).
- Kunes, M., and Sauter, T. (2001), "Fieldbus-Internet connectivity: The SNMP approach," *IEEE Trans. Ind. Electron.*, vol. 48, no. 6, pp. 1248-1256, 2001.
- Lee, J. S. (1999), *A PLC-Based Design for the Controller and the Diagnostic System in Discrete Event Systems*, Master Thesis, Department of Electrical and Control Engineering, NCTU, Taiwan.
- Lee, J. S., and Hsu, P. L. (accepted), "A systematic approach for the sequence controller design in manufacturing systems," *Int. J. Adv. Manuf. Tech.*
- Lee, J. S., and Hsu, P. L. (2003a), "A Petri-net approach to hierarchical supervision for remote-controlled processes," in *Proc. IEEE Int. Conf. Systems, Man and Cybernetic*, Washington, DC, pp. 1880-1885.

- Lee, J. S., and Hsu, P. L. (2003b), "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., and Hsu, P. L. (2004a), "An improved evaluation of ladder logic diagrams and Petri nets for the sequence controller design in manufacturing systems," *Int. J. Adv. Manuf. Tech.*, vol. 24, no. 3-4, pp. 279-287.
- Lee, J. S., and Hsu, P. L. (2004b), "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.
- Liang, G. R., and Hong, H. M. (1994), "Hierarchy transformation method for repetitive manufacturing system specification, design, verification and implementation," *Comput.-Integr. Manuf. Syst.*, vol. 7, no. 3, pp. 191-205.
- Looney, C. G., and Alfize A. R. (1987), "Logic control via Boolean rule matrix transformations," *IEEE Trans. Syst. Man, and Cybern.*, vol. 17, no. 6, pp. 1077-1082.
- 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.
- Miyazawa, I., Tanaka, H., and Sekiguchi, T. (1997), "Verification of the behavior of sequential function chart based on its Petri net model", in *Proc. IEEE Int. Workshop Emerging Technologies and Factory Automation*, pp. 532-537.
- Moalla, M. (1985), "Réseaux de Petri interprétés et Grafcet", *TSI—Technique et Science Informatique*, vol. 14, no.1, pp. 17-30.

- Moody, J. O., and Antsaklis, P. J. (1998), *Supervisory Control of Discrete Event systems Using Petri Nets*. Boston, MA: Kluwer.
- Mosterman, P. J., Sztipanovits, J., and Engell, S. (2004), “Computer-automated multi-paradigm modeling in control systems technology,” *IEEE Trans. Contr. Syst. Technol.*, vol. 12, no. 2, pp. 223-234, (Special section on Computer automated multi-paradigm modeling).
- Murata, T. (1989), “Petri nets: Properties, analysis, and applications,” *Proc. of the IEEE*, vol. 77, no. 4, pp. 541-580.
- Peng, S. S., and Zhou, M. C. (2001), “Conversion between ladder diagrams and PNs in discrete-event control design— A survey,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Tucson, AZ, pp. 2682-2687.
- Pessen, W. (1989), “Ladder-diagram design for programmable controllers,” *Automatica*, vol. 25, no. 3, pp. 407-412.
- 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.1 Rep. RADC-TR-65--377, vol. 1, pages 1-Suppl. 1. 1966.
- Prabhaka, A. (1993), *Integration Definition for Function Modeling (IDEF0)*. National Institute of Standards and Technology, FIPS 183.
- Ramadge, P. J., and 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., and Wonham, W. M. (1989), “The control of discrete event systems,” *Proc. IEEE*, vol. 77, no. 1, pp. 81-98.
- Rasmussen, J., Pejtersen, A. M., and Goodstein, L. P. (1994), *Cognitive Systems Engineering*. New York, NY: John Wiley and Sons.

- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenzen, W. (1991), *Object-Oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice Hall.
- 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.
- Srinivasan, R. S. (1998), "Modeling and performance analysis of cluster tools using Petri nets," *IEEE Trans. Semicond. Manuf.*, vol. 11, no. 3, pp. 394-403, (Special section on Petri nets in semiconductor manufacturing).
- Stallings, W. (1993), *SNMP, SNMP2, and CMIP*. Reading, MA: Addison-Wesley.
- Tilbury, D., and Khargonekar, P. (2001), "Challenges and opportunities in logic control for manufacturing systems," *IEEE Contr. Syst. Maga.*, vol. 21, no. 1, pp. 105-108.
- Tittus, M., and Lennartson, B. (1999), "Hierarchical supervisory control for batch processes," *IEEE Trans. Contr. Syst. Technol.*, vol. 7, no. 5, pp. 542-554.
- Uzam, M., and Jones, A. H. (1998), "Discrete event control system design using automation Petri nets and their ladder diagram implementation," *Int. J. Adv. Manuf. Tech.*, vol. 14, no. 10, pp. 716-728 (Special issue on Petri nets applications in manufacturing system).
- Uzam, M., Jones, A. H., and 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.
- Venkatesh, K., Zhou, M. C., and Caudill, R. (1994a) "Comparing ladder logic diagrams and Petri nets for sequence controller design through a discrete manufacturing system," *IEEE Trans. Indu. Electron.*, vol. 41, no. 6, pp. 611-619, (Special section on Petri nets in manufacturing).

- Venkatesh, K., Zhou, M. C., and Caudill, R. (1994b), "Evaluating the complexity of Petri nets and ladder logic diagrams and for sequence controllers design in flexible automation," in *Proc. IEEE Symp. Emerging Technology and Factory Automation*, pp. 428-435.
- Vucetic, J., and Kline, P. (1998), "Signal monitoring system for wireless network operation and management," in *Proc. SBT/IEEE Int. Symp. Telecommu.*, pp. 296-300.
- Weaver, A., Luo, J., and Zhang, X. (1999), "Monitoring and control using the Internet and Java," in *Proc. IEEE Int. Conf. Industrial Electronics*, San Jose, CA, pp. 1152-1158.
- Wong, K. C., and Wonham, W. M. (1996), "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 6, pp. 241-273.
- Wooldridge, M., and Jenkins, M. R. (1995), "Intelligent agents: theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115-152.
- Yang, S. H., Chen, X., and Alty, J. L. (2002), "Design issues and implementation of Internet-based process control systems," *Contr. Engin. Pract.*, vol. 11, no. 6, pp. 709-720.
- Zeltserman, D. (1999), *A Practical Guide to SNMPv3 and Network Management*. Upper Saddle River, NJ: Prentice-Hall.
- Zhong, H., and Wonham, W. M. (1990), "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 35, no. 10, pp. 1125-1134, Oct.
- Zhou, M. C., and Dicesare, F. (1989), "Adaptive design of Petri net controllers for error recovery in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 5, pp. 963-973.

Zhou, M. C., and 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., and 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).

Zhou, M. C., and Twiss, E. (1995), "A comparison of relay ladder logic programming and Petri net approach for sequential industrial control systems," in *Proc. IEEE Int. Conf. Control Applications*, pp. 748-753.

Zhou, M. C., and Twiss, E. (1998), "Design of industrial automated systems via relay ladder logic programming and Petri nets," *IEEE Trans. Syst., Man, and Cybern.*, Part C, vol. 28, no. 1, pp. 137-150.

Zhou, M. C., and Venkatesh, K. (1998), *Modeling, Simulation and Control of Flexible Manufacturing Systems: A Petri Net Approach*. Singapore: World Scientific.

Zuberek, W. M. (2001), "Timed Petri nets in modeling and analysis of cluster tools," *IEEE Trans. Robot. Automat.*, vol. 17, no. 5, pp. 562-575.

Zurawski, R., and 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).

VITA

Aug 16, 2004

PERSONAL DATA

Name: 李俊賢, Jin-Shyan Lee

Date of Birth: Nov. 7, 1975

E-mail: jslee.ece88g@nctu.edu.tw



EDUCATION

1999/9 – 2004/7	Receive the Ph.D. degree in the Department of Electrical and Control Engineering at National Chiao-Tung University, Taiwan, ROC.
1997/9 - 1999/6	Receive the M.S. degree in the Department of Electrical and Control Engineering from National Chiao-Tung University, Taiwan, ROC.
1995/9 - 1997/6	Receive the B.S. degree in the Department of Mechanical Engineering from National Taiwan University of Science and Technology, Taiwan, ROC.

EXPERIENCE

2004/9	Co-organizer of a special section: “Computer automated multi-paradigm modeling” in <i>IEEE Int. Conf. Computer-Aided Control System Design</i> 2004, Sept 1-4, Taipei, Taiwan.
2003 - Present	Assistant Reviewer of <i>IEEE Transactions on Systems, Man and Cybernetic, Part A: Systems and Humans</i> .
2003/7 - 2004/6	One-year Visiting Researcher in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, USA.
2003/10	Invited Speaker of North Jersey IEEE Control Systems Chapter.
2003/3 - 2003/6	Teaching assistant of the class: “Computer-Controlled Systems.”
2002/2 – 2003/6	Secretary of the Chinese Automatic Control Society (CAC).

2002/4 - 2003/6	Research Assistant of the project: “Human Technology — Intelligent Transportation System (ITS)” sponsored by the Ministry of Education for promoting University Academic Excellence.
2001/8 - 2003/6	Research Assistant of the project: “Design of Real-Time Supervisory Systems for Intelligent Vehicles” sponsored by the National Science Council (NSC).
2001/7 - 2001/8	Engineering internship of the Industrial Control Business in Mirle Automation Corporation, research and develop the Internet-based environmental monitoring systems.
2001/3 - 2001/6	Teaching assistant of the class: “Advanced Digital Control Systems.”
2000/8 - 2001/7	Research Assistant of the project: “The Lane Departure Warning System with Image Processing for Automobile Drivers” sponsored by the NSC.
1999/11 - 2001/12	Research Assistant of the project: “Core Technology Study and Prototype Development of Advanced Vehicle Safety Systems (AVSS)” sponsored by the China Engineering Consultants, Inc.
1998/8 - 2000/7	Research Assistant of the project: “Integration of Advanced Mechatronics Systems and Information Intelligence to Construct an Open-Structured Controller” sponsored by the NSC.
1998/3 - 1998/6	Teaching assistant of the class: “Micro-Computer Lab.”

ATTENDED CONFERENCES

- **International Conferences**

1. 2004 Sept., IEEE Int. Conf. Computer-Aided Control System Design, Taipei, Taiwan.
2. 2004 Aug., SICE Annual Conference, Sapporo, Japan.
3. 2003 Oct., IEEE Intl. Conf. Systems, Man and Cybernetic, Washington, DC, USA.
4. 2003 Sept., Chinese Institute of Engineers-USA Annual Convention, Newark, NJ, USA.
5. 2002 Sept., IEEE Int. Conf. Control Applications, Glasgow, Scotland, UK.
6. 2001 Oct., IEEE Intl. Conf. Systems, Man and Cybernetics, Tucson, AZ, USA.
7. 2000 Sept., IEEE Int. Conf. Control Applications, Anchorage, AK, USA.

- **Domestic Conferences**

1. 2003 Mar., Chinese Automatic Control Conference, Taoyuan, Taiwan.
2. 2002 Mar., Chinese Automatic Control Conference, Tainan, Taiwan.
3. 2001 Mar., Chinese Automatic Control Conference, Taoyuan, Taiwan.
4. 2000 Mar., Chinese Automatic Control Conference, Hsinchu, Taiwan.

ACHIEVEMENTS

2004/8	Winner of the SICE International Scholarship in the 2004 SICE Annual Conference, Sapporo, Japan.
2004/8	Finalist of both the Annual International Award and Young Author's Award in the 2004 SICE Annual Conference, Sapporo, Japan.
2004/2	3 rd Place of IEEE Student Paper Presentation Contest in the graduate category, awarded by North New Jersey IEEE Section, Newark, USA.
2003/9	3 rd Place of Student Paper Contest Scholarship in the Chinese Institute of Engineers (CIE)-USA Annual Convention, Newark, USA.
2003/7 - 2004/6	One-year Visiting Scholarship in Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, USA, sponsored by National Science Council (NSC), Taiwan.
2002/3	Student Paper Award of the 2002 Chinese Automatic Control Conference, Taiwan.
2001/8 - 2002/7	2001 Excellent Ph.D. Student's MOE Scholarship of the Department of Electrical and Control Engineering, National Chiao-Tung University, awarded by the Ministry of Education (MOE).
2001/3	Student Paper Award of the 2001 Chinese Automatic Control Conference, Taiwan.
2000/8 - 2001/7	2000 Excellent Ph.D. Student's MOE Scholarship of the Department of Electrical and Control Engineering, National Chiao-Tung University, awarded by the Ministry of Education (MOE).

PUBLICATION LIST

August 16, 2004

JOURNAL PAPERS

- | |
|---|
| 1. J. S. Lee and P. L. Hsu, "An improved evaluation of ladder logic diagrams and Petri nets for the sequence controller design in manufacturing systems," <i>International Journal of Advanced Manufacturing Technology</i> , vol. 24, no. 3-4, pp. 279-287, August 2004 (SCI). |
| 2. J. S. Lee and P. L. Hsu, "Design and implementation of the SNMP agents for remote monitoring and control via UML and Petri nets," <i>IEEE Transactions on Control System Technology</i> , vol. 12, no. 2, pp. 293-302, March 2004 (SCI). |
| 3. J. S. Lee and P. L. Hsu, "Remote supervisory control of the human-in-the-loop system by using Petri nets and Java," <i>IEEE Transactions on Industrial Electronics</i> , vol. 50, no. 3, pp. 431-439, June 2003 (SCI). |
| 4. J. S. Lee , M. C. Zhou, and P. L. Hsu, "An application of Petri nets to supervisory control for human-computer interactive systems," accepted as a regular paper to appear in <i>IEEE Transactions on Industrial Electronics</i> (SCI). |
| 5. J. S. Lee and P. L. Hsu, "A systematic approach for the sequence controller design in manufacturing systems," accepted as a regular paper to appear in <i>International Journal of Advanced Manufacturing Technology</i> (SCI). |
| 6. J. S. Lee , M. C. Zhou, and P. L. Hsu, "Statechart Modeling and Web-Based Simulation of Hybrid Dynamic Systems for e-Automation," accepted as a regular paper to appear in <i>Journal of Chinese Institute of Industrial Engineers</i> (EI). |
| 7. J. S. Lee and P. L. Hsu, "Implementation of a remote hierarchical supervision system using Petri nets and agent technology," <i>IEEE Transactions on Systems, Man and Cybernetic, Part A: Systems and Humans</i> (revised). |

INTERNATIONAL CONFERENCE PAPERS

- | |
|---|
| 1. J. S. Lee , M. C. Zhou, and P. L. Hsu, "Multi-paradigm modeling approach for hybrid dynamic systems," accepted to present in <i>IEEE Int. Conf. Computer-Aided Control System Design</i> , Taipei, Taiwan, Sept. 2004 (invited paper). |
| 2. J. S. Lee , M. C. Zhou, and P. L. Hsu, "Petri net-based design of modular supervisors for remotely human control systems," <i>SICE Annual Conference</i> , Sapporo, Japan, August 2004, pp. 1271-1276 (Winner of the SICE International Scholarship, also the Finalist in both the Annual International Award and Young Author's Award). |
| 3. J. S. Lee and P. L. Hsu, "A Petri-Net approach to hierarchical supervision for remote-controlled processes," <i>IEEE Intl. Conf. Systems, Man and Cybernetic</i> , Washington, D.C., October 2003, pp. 1880-1885 (invited paper). |

4. J. S. Lee and P. L. Hsu, "An IDEF0/Petri net approach to the system integration in semiconductor manufacturing systems," <i>IEEE Intl. Conf. Systems, Man and Cybernetic</i> , Washington, DC, October 2003, pp. 4910-4915.
5. J. S. Lee and P. L. Hsu, "Development of the supervisory agent for Internet-based control systems with human-in-the-loop," <i>Chinese Institute of Engineers (CIE)-USA/GNYC Annual Convention</i> , Newark, NJ, September 2003 (3rd Place of Student Paper Contest).
6. J. S. Lee and P. L. Hsu, "Design of remote environmental monitoring systems," <i>IEEE Int. Conf. Control Applications</i> , Glasgow, Scotland, September 2002, pp. 856-861.
7. J. S. Lee and P. L. Hsu, "UML-based modeling and multi-threaded simulation for hybrid dynamic systems," <i>IEEE Int. Conf. Control Applications</i> , Glasgow, Scotland, September 2002, pp. 1207-1212.
8. J. S. Lee and P. L. Hsu, "An object-oriented design of the hybrid controller for automated vehicles in an AHS," <i>IEEE Intelligent Vehicles Symposium</i> , Versailles, France, June 2002, pp. 115-120.
9. J. S. Lee and P. L. Hsu, "A new approach to evaluate ladder logic diagrams and Petri nets via the IF-THEN transformation," <i>IEEE Intl. Conf. Systems, Man and Cybernetics</i> , Tucson, AZ, October 2001, pp. 2711-2716.
10. J. S. Lee and P. L. Hsu, "A PLC-based design for the sequence controller in discrete event systems," <i>IEEE Int. Conf. Control Applications</i> , Anchorage, AK, September 2000, pp. 929-934.

DOMESTIC CONFERENCE PAPERS

1. J. S. Lee and P. L. Hsu, "An IDEF0/Petri net approach to the emulator design for semiconductor manufacturing systems," <i>Chinese Automatic Control Conference</i> , Taoyuan, Taiwan, March 2003, pp. 1209-1214.
2. J. S. Lee and P. L. Hsu, "Modeling and simulation for hybrid dynamic systems," <i>Chinese Automatic Control Conference</i> , Tainan, Taiwan, March 2002, pp. 20-25 (Student Paper Award).
3. J. S. Lee and P. L. Hsu, "Design of PLC-based fault diagnostic systems via the Petri net and logic functions," <i>Chinese Automatic Control Conference</i> , Taoyuan, Taiwan, March 2001, pp. 30-35 (Student Paper Award).
4. J. S. Lee and P. L. Hsu, "A PLC-based design for the sequence controller in discrete event systems," <i>Chinese Automatic Control Conference</i> , Hsinchu, Taiwan, March 2000, pp. 12-17.