

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

ABSTRACT

A PETRI-NET BASED METHODOLOGY FOR MODELING, SIMULATION, AND CONTROL OF FLEXIBLE MANUFACTURING SYSTEMS

**by
Venkatesh Kurapati**

Global competition has made it necessary for manufacturers to introduce such advanced technologies as flexible and agile manufacturing, intelligent automation, and computer-integrated manufacturing. However, the application extent of these technologies varies from industry to industry and has met various degrees of success. One critical barrier leading to successful implementation of advanced manufacturing systems is the ever-increasing complexity in their modeling, analysis, simulation, and control. The purpose of this work is to introduce a set of Petri net-based tools and methods to address a variety of problems associated with the design and implementation of flexible manufacturing systems (FMSs). More specifically, this work proposes Petri nets as an integrated tool for modeling, simulation, and control of flexible manufacturing systems (FMSs). The contributions of this work are multifold. First, it demonstrates a new application of PNs for simulation by evaluating the performance of pull and push diagrams in manufacturing systems. Second, it introduces a class of PNs, Augmented-timed Petri nets (ATPNs) in order to increase the power of PNs to simulate and control flexible systems with breakdowns. Third, it proposes a new class of PNs called Real-time Petri nets (RTPNs) for discrete event control of FMSs. The detailed comparison between RTPNs and traditional discrete event methods such as ladder logic diagrams is presented to answer the basic question 'Why is a PN better tool than ladder logic diagram?' and to justify the PN method.

Also, a conversion procedure that automatically generates PN models from a given class of logic control specifications is presented. Finally, a methodology that uses PNs for the development of object-oriented control software is proposed. The present work extends the PN state-of-the-art in two ways. First, it offers a wide scope for engineers and managers who are responsible for the design and the implementation of modern manufacturing systems to evaluate Petri nets for applications in their work. Second, it further develops Petri net-based methods for discrete event control of manufacturing systems.

**A PETRI-NET BASED METHODOLOGY FOR MODELING, SIMULATION,
AND CONTROL OF FLEXIBLE MANUFACTURING SYSTEMS**

**by
Venkatesh Kurapati**

**A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy**

Department of Mechanical and Industrial Engineering

January 1995

Copyright 1995 by Venkatesh Kurapati

ALL RIGHTS RESERVED

APPROVAL PAGE

A PETRI-NET BASED METHODOLOGY FOR MODELING, SIMULATION, AND CONTROL OF FLEXIBLE MANUFACTURING SYSTEMS

Venkatesh Kurapati

Dr. MengChu Zhou, Dissertation Adviser / Date
Assistant Professor of Manufacturing Engineering,
Electrical and Computer Engineering, NJIT

Dr. Reggie J. Caudill, Committee Chairman, Dissertation Co-Adviser / Date
Professor of Mechanical and Industrial Engineering, NJIT

Dr. Anthony Robbi / Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Zhiming Ji / Date
Assistant Professor of Mechanical and Industrial Engineering, NJIT

Dr. Ernest Geskin / Date
Professor of Mechanical and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Venkatesh Kurapati

Degree: Doctor of Philosophy in Mechanical Engineering

Date: January 1995

Undergraduate and Graduate Education:

- Doctor of Philosophy in Mechanical Engineering
New Jersey Institute of Technology, Newark, NJ, 1995
- Master of Science in Manufacturing Systems Engineering
Florida Atlantic University, Boca Raton, FL, 1993
- Master of Technology in Mechanical Engineering
Indian Institute of Technology, Madras, India, 1991
- Bachelor of Technology in Mechanical Engineering
S.V. University, Tirupati, India, 1988

Major: Manufacturing Engineering

Presentations and Publications:

Kurapati, Venkatesh, Kaighobadi Mehdi, MengChu Zhou, and Reggie Caudill, "Augmented Timed Petri Nets for Modeling of Robotic Systems with Breakdowns," *Journal of Manufacturing Systems*, Vol. 13, No. 4, 1994, pp. 289-301.

Kurapati, Venkatesh, MengChu Zhou, and Reggie Caudill, "Comparing Ladder Logic Diagrams and Petri Nets for the Design of Sequence Controllers Through a Discrete Manufacturing System," (in press) *IEEE Transactions on Industrial Electronics*, Vo. 41, No. 6, 1994.

Kurapati, Venkatesh, MengChu Zhou, Kaighobadi Mehdi, and Reggie Caudill, "A Petri Net Approach to Investigating the Performance of Push and Pull Paradigms in Flexible Factory Automated Systems using Petri Nets," (in press) *International Journal of Production Research*.

- Zhou, MengChu, and Kurapati Venkatesh, "Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach," World Scientific Publishing, London, UK, to appear in 1995.
- Kurapati, Venkatesh, MengChu Zhou, and Reggie Caudill, "Evaluating the Design Complexity of Ladder Logic Diagrams and Petri Nets for Design of Sequence Controllers in Flexible Automation," *Proc. of Seiken/IEEE Symposium on Emerging Technologies in Flexible Automation*, Kyoto, Japan, November 6-10, 1994, pp. 428-435.
- Kurapati, Venkatesh, MengChu Zhou, and Reggie Caudill, "Automatic Generation of Petri Net Models from Logic Control Specifications," *Proc. of the 4th Int. Conf. on Computer Integrated Manufacturing and Automation Technology*, Troy, NY, October 10-12, 1994, pp. 242-247.
- Kurapati, Venkatesh, MengChu Zhou, Reggie Caudill, and E.B. Fernandez, "A Control Software Design Method for CIM Systems," *Proc. of the Computer Integrated Manufacturing in Process Industries*, New Brunswick, NJ, April 15-17, 1994, pp. 565-579.
- Kurapati, Venkatesh, MengChu Zhou, and Reggie Caudill, "Comparison of Petri Nets and Ladder Diagrams for Sequence Control of Discrete Manufacturing Systems," *Proc. of IEEE Regional Conf. in Control Systems*, New Jersey Institute of Technology, Newark, NJ, August, 1993, pp. 73-76.

**This dissertation is dedicated to
my father, Sankara Narayana and mother, Sree Lakshmi**

ACKNOWLEDGMENT

The author expresses his deep sense of gratitude to his adviser, Dr. MengChu Zhou, for his invaluable guidance, friendship, and moral support throughout this research. Dr. Zhou's untiring help is sincerely appreciated.

Special thanks to his adviser Dr. Reggie Caudill for his encouragement, mentorship, and suggestions. The author is grateful to Professors Anthony Robbi, Zhiming Ji, and Ernest Geskin for their suggestions and constructive comments. Their service as committee members is appreciated.

The research assistantship provided by Department of Mechanical and Industrial Engineering during Spring 1993, the financial support by Center for Manufacturing Systems from Summer 1993 to Fall 1994, and the tuition fee by Manufacturing Systems Engineering Program during Fall 1994 is sincerely acknowledged. The author expresses his gratefulness to Dr. Rong Chen for his constant support, encouragement, and guidance.

The author acknowledges The Robotics Center, Florida Atlantic University, Boca Raton for providing the equipment needed to illustrate the use of Real-time Petri nets and to perform the bench-mark study between Petri nets and ladder logic diagrams.

The author also appreciates Rutgers Computer Services and Machine Vision and Microsystems Laboratory at NJIT for the computer facilities. The author also wants to extend his special thanks to Hamwantee Singh, the reference librarian at NJIT for her suggestions and help in finding the literature related to PNs.

The author is thankful to his sisters, brother-in-law, Kamela Mohammed, and Amita Bhatnagar for their constant support and cooperation throughout this research.

Finally, the author wishes to express his sincere gratitude to his parents and the God for giving the knowledge, the determination, and the strength to complete this dissertation and paving the way to lead to the intellectual life.

TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION.....	1
1.1. Background and Motivation.....	1
1.2. Goals and Objectives.....	3
1.3. Organization.....	6
2. FLEXIBLE MANUFACTURING SYSTEMS: AN OVERVIEW.....	8
2.1. Introduction.....	8
2.2. Definitions of FMS.....	9
2.3. Impetus for Change.....	12
2.4. Installation, Implementation, and Integration of FMSs.....	15
2.5. Applications of FMSs.....	19
2.6. Problems in Installation and Implementation of FMSs.....	21
2.6.1. Managerial Problems.....	21
2.6.2. Technical Problems.....	23
2.7. Summary.....	34
3. PETRI NETS AS AN INTEGRATED TOOL AND METHODOLOGY IN FMSs.....	36
3.1. Concepts and Terminology of Petri Nets.....	36
3.2. Applications of PNs in FMSs.....	41
3.2.1. Simulation and Performance Evaluation.....	42
3.2.2. Breakdown Modeling.....	44
3.2.3. Discrete Event Control.....	45
3.2.4. Control Software Development.....	47

TABLE OF CONTENTS
(Continued)

Chapter	Page
4. PERFORMANCE EVALUATION OF PUSH AND PULL PARADIGMS IN FLEXIBLE AUTOMATION.....	49
4.1. Introduction.....	49
4.2. Application Illustration.....	52
4.2.1. System Configuration and Assumptions.....	53
4.2.2. PNM Formulation and Analysis.....	57
4.3. Procedure for PN Modeling and Analysis and Simulation results.....	66
4.3.1. FMS with the Pull Paradigm.....	68
4.3.2. FAS with the Pull Paradigm.....	69
4.3.3. FMS with the Push paradigm.....	70
4.3.4. FAS with the Push Paradigm.....	71
4.3.5. Summary of Results.....	72
4.4. Summary.....	74
5. AUGMENTED TIMED PETRI NETS FOR MODELING BREAKDOWN HANDLING.....	76
5.1. Introduction.....	76
5.2. Augmented-timed Petri Nets (ATPNs).....	78
5.3. Application Illustration: A Flexible Assembly System.....	84
5.3.1. ATPN Modeling of the System.....	87
5.3.2. Simulation and Analysis of the ATPN Model.....	91
5.4. Summary.....	94

TABLE OF CONTENTS
(Continued)

Chapter	Page
6. REAL-TIME PETRI NETS FOR CONTROL AND SIMULATION.....	96
6.1. Introduction.....	96
6.2. Real-time PNs.....	96
6.3. Real-time PNs and Other PN Extensions for Control.....	101
6.4. Example: An Automatic Assembly System.....	105
6.5. A case study: An Electro-Pneumatic System.....	108
6.6. Software Description to Execute Real-time Petri Nets.....	111
7. COMPARISON OF REAL-TIME PETRI NETS AND LADDER LOGIC DIAGRAMS.....	114
7.1. Introduction.....	114
7.2. Comparison Criteria for Control Logic Design by PNs and LLDs.....	116
7.3. Comparison Through an Electro-pneumatic System.....	120
7.3.1. Sequence Controller Design for a Given Sequence.....	120
7.3.2. Control for Other Sequences.....	122
7.3.3. Discussions.....	126
7.4. Analytical Formulas to Evaluate the Complexity of PNs and LLDs.....	130.
7.4.1. Logical AND, Logical OR, and Sequential Modeling.....	131
7.4.2. Timed logical AND, Timed Logical OR, and Timed Sequential Modeling.....	134
7.4.3. Other formulas for Estimating Basic Elements in PN or LLD.....	137
7.5. Methodology to Use the Analytical Formulas.....	141
7.6. Illustration of the Methodology Through Examples.....	145
7.6.1. An Automatic Assembly System.....	145

TABLE OF CONTENTS
(Continued)

Chapter	Page
7.6.2. An Electro-pneumatic System Without Sustained Signals.....	146
7.6.3. An Electro-pneumatic System with Sustained Signals.....	149
7.7. Summary.....	154
8. CONVERSION OF LOGIC SPECIFICATIONS INTO PETRI NET MODELS.....	156
8.1. Introduction.....	156
8.2. Illustration of the Formulation of PN Model.....	157
8.2.1. Single Path Sequences with no Repetitive Actions.....	157
8.2.2. Single Path Sequences with Repetitive actions.....	164
8.2.3. Multi-path Sequences with Simultaneous Parallel Paths.....	166
8.2.4. Multi-path Sequences with Alternative Parallel Paths.....	167
8.2.5. Multi-path Sequences with Option of Bypassing Nodes.....	168
8.2.6. Multi-path Sequences with Option of Repeating Nodes.....	169
8.3. Merging of Common places in a PN Model.....	169
8.4. Summary.....	173
9. AN OBJECT-ORIENTED DESIGN METHODOLOGY FOR DEVELOPMENT OF FMS CONTROL SOFTWARE.....	174
9.1. Introduction.....	174
9.2. Methodology for FMS Control Software Development.....	179
9.2.1. Methodology.....	179
9.2.2. Fundamentals of OOD.....	182
9.2.3. Object Modeling Technique Diagram as a Static Modeling Tool.....	184
9.2.4. Petri Nets as a Dynamic Modeling Tool.....	186

TABLE OF CONTENTS
(Continued)

Chapter	Page
9.3. Illustration of the Methodology with an FMS.....	188
9.3.1. OMT Diagram and PNM of the FMS.....	190
9.3.2. Complete Structure of Objects with Their Static and Dynamic Relations.....	198
9.3.3. Reusability, Extensibility, and Modifiability of the Design.....	200
9.4. Summary.....	202
10. CONCLUSIONS.....	205
10.1. Contribution.....	205
10.1.1. Simulation and Performance Evaluation.....	206
10.1.2. Breakdown Modeling.....	207
10.1.1. Discrete-Event Control.....	208
10.2. Limitations	211
10.3. Further Research.....	214
APPENDIX A.....	217
SOFTWARE PACKAGE TO EXECUTE TPN AND ATPN.....	217
APPENDIX B.....	227
SOFTWARE PACKAGE TO EXECUTE RTPN	227
REFERENCES.....	246

LIST OF TABLES

Table	Page
2.1. Comparison of how machine and work-parts spend their time in the shop of a conventional system and in an FMS using optimistic-pessimistic format (Salomon and Biegel 1984).....	14
2.2. Classification of Manufacturing Systems and FMS.....	16
2.3. Cross-references to research related to problems in FMS.....	33
3.1. Major applications of PNs in FMSs.....	43
4.1. Conventions of Petri net modeling.....	53
4.2. Conveyance time matrix in the system for production of PR1 and PR2 (time units).....	54
4.3. Processing times and the sequence of parts in the system (time units)..	55
4.4. Variation of moving lot size with respect to number of AGVs and assigned tasks).....	56
4.5. Explanation of typical places and transitions in the PNMs shown in Figs. 4.4 and 4.5).....	63
4.6. FFAS functioning as FMS with the pull paradigm.....	68
4.7. System functioning as FAS with the pull paradigm (the same legend as Table 4.6's).....	69
4.8. System functioning as FMS with the push paradigm (the same legend as Table 4.6's).....	70
4.9. System functioning as FAS with the push paradigm (the same legend as Table 4.6's).....	71
4.10. Solution sets for FMS and FAS (the same legend as Table 4.6's).....	73
5.1. MTBFs of Robots in the FAS (minutes).....	86
5.2. Time and breakdown sequence of robots for various robot breakdown rates	86
5.3. Interpretation of places and transitions used in Fig. 5.5.....	89
5.4. Performance of the assembly system with and without breakdown of robots.....	91

**LIST OF TABLES
(Continued)**

Table	Page
5.5. Optimum number of assembly fixtures required for various robot breakdown rates.....	93
6.1. Various methods of Petri net based sequence control.....	101
6.2. The input mapping table where X_i is input channel number.....	106
6.3. The output mapping table where Y_i is number sent to the digital output interface	107
6.4. Attributes of places and transitions in the RTPN.....	107
6.5. The input mapping table where X_i is input channel number.....	109
6.6. The output mapping table.....	110
6.7. Attributes of transitions modeling actions.....	110
7.1. Representations by Petri nets and ladder logic diagrams.....	117
7.2. Comparison of the basic elements in LLD and PNs.....	126
7.3. Required basic elements to control the system in Fig. 6.3.....	146
7.4. Required basic elements to control the system in Example 2.....	147
7.5. Required basic elements to model the Sequence 1 using PN.....	149
7.6. Required basic elements to model the Sequence 1 using LLD.....	152
8.1. Required basic elements to model the Sequence 1 using PN.....	162
8.2. Comparison of the basic elements in LLD and PNs obtained by two methods.....	163
9.1. Research related to OOD, PNs, and Ada.....	178
9.2. Interpretation of typical places and transitions in PNMs shown in Figs. 9.9 and 9.10.....	192

LIST OF FIGURES

Figure	Page
3.1. (a) A simple assembly cell, (b) Petri net model.....	39
3.2. Timed Petri net model of the assembly cell (a). Before firing of transition 1 (before assembly started) (b). After firing of transition 1 (after assembly started) (c). During firing of transition 2 (during assembly) (d). After firing of transition 2 (after assembly finished).....	40
4.1. Layout of the manufacturing system investigated.....	54
4.2. PNM for production of parts A and B under pull paradigm.....	58
4.3. PNM for production of parts A and B under push paradigm.....	62
4.4. PNM of the FMS under pull paradigm.....	64
4.4. PNM of the FMS under push paradigm.....	65
4.6. Procedure for PN modeling and analysis.....	66
4.7. Performance of the system with different configurations.....	74
5.1. An Example of a Timed Petri net Model.....	79
5.2. Example of an ATPN model.....	83
5.3. Layout of the flexible assembly system.....	84
5.4. ATPN model for breakdown handling of Robot 3.....	87
5.5. ATPN model of flexible assembly system with breakdown handling.....	90
5.6. Effect of MTBF on production volume.....	92
5.7. Effect of MTBF on average robot utilization.....	92
5.8. ATPN model for real-time control.....	95
6.1. Procedure for formulating a RTPN based controller.....	99
6.2. Controlling a system using a RTPN based controller.....	100
6.3. Schematic of an automatic assembly system.....	106
6.4. RTPN model for the system.....	108

**LIST OF FIGURES
(Continued)**

Figure	Page
6.5. Schematic of an electro-pneumatic system considered.....	109
6.6. RTPN for sequence 1: ST, A+,B+,{C+,A-}, {B-,C-}.....	111
7.1. (a) LLD and (b) PN for Sequence 1: ST, A+, B+, {C+, A-}, {B-, C-}.....	121
7.2. (a) LLD and (b) PN for Sequence 2.....	123
7.3. (a) LLD and (b) PN for Sequence 3.....	125
7.4. LLD for Sequence 4.....	127
7.5. PN for Sequence 4.....	128
7.6. PNs and LLDs modeling logical AND.....	131
7.7. PNs and LLDs modeling logical OR.....	132
7.8. PNs and LLDs for sequential modeling.....	133
7.9. PNs and LLDs for timed logical AND.....	134
7.10. PNs and LLDs for timed logical OR.....	135
7.11. PNs and LLDs for timed sequential modeling.....	136
7.12. Typical cylinder-actuating circuit (Pessen 1989b) (a). requiring sustained solenoid signals (b). not requiring sustained solenoid signals.....	138
7.13. PNs and LLDs modeling emergency stop.....	139
7.14. PNs and LLDs modeling counter.....	140
7.15. LLD modeling a relay.....	141
7.16. Method to estimate basic elements.....	142
7.17. LLD model for the system shown in Fig. 6.3.....	146
7.18. (a) PN and (b) LLD for sequence:{A+, B+},{A-, D+},{B-, D-, C+},C	148
8.1. PN model for the Sequence 1.....	158

**LIST OF FIGURES
(Continued)**

Figure	Page
8.2. PN model for the Sequence 2.....	159
8.3. PN model for the Sequence 3.....	160
8.4. PN model for the Sequence 4.....	161
8.5. PN model corresponding to the sequence in Example 1.....	164
8.6. PN model corresponding to the sequence in Example 2.....	165
8.7 (a) Control flow consisting multipath sequence with simultaneous parallel paths, (b) Simplified PN modeling multipath sequence with simultaneous parallel paths.....	166
8.8. (a) Control flow consisting multipath sequence with alternative parallel paths, (b) PN modeling the multipath sequence with alternative parallel paths.....	167
8.9. (a) Control flow consisting multipath sequence with option of bypassing nodes, (b) PN modeling the multipath sequence with option of bypassing nodes.....	168
8.10. (a) Control flow consisting multipath sequence with option of repeating nodes, (b) PN modeling the multipath sequence with option of repeating nodes.....	169
8.11. PN model after merging of common places in Fig. 8.5.....	170
8.12. PN model after eliminating conflicts in Fig. 8.11.....	171
8.13. PN model (a) after merging the common places in Fig. 8.6, (b) after eliminating conflicts.....	172
9.1. Proposed systematic methodology for FMS control software development.....	181
9.2. Principle of object-oriented design of FMS control software.....	182
9.3. A class definition in OMT.....	184
9.4. OMT diagram for Generalization.....	184
9.5. OMT diagram for Aggregation.....	185

**LIST OF FIGURES
(Continued)**

Figure	Page
9.6. OMT diagram for Association.....	186
9.7. The configuration of FMS	189
9.8. OMT diagram of the FMS.....	191
9.9. PNM of machine 1 (MC1).....	191
9.10. PNM of the FMS under push paradigm.....	193
9.11. PNM for the FMS control system.....	194
9.12. PNM of exception handling by main controller.....	196
9.13. Class definitions of important object classes in FMSs.....	198
9.14. The expanded configuration of FMS.....	201
9.15. OMT diagram for the expanded FMS.....	202
9.16. PNM of the FMS under pull paradigm.....	203

CHAPTER 1

INTRODUCTION

In this chapter, the background, motivation, and contributions of this work are stated. The organization of this dissertation is outlined.

1.1. Background and Motivation

Global competition has made it necessary for manufacturers to introduce such advanced technologies as flexible and agile manufacturing, intelligent automation, and computer-integrated manufacturing. However, the application extent of these technologies varies from industry to industry and has met various degrees of success. One critical barrier leading to the successful implementation of flexible manufacturing and related advanced systems is the ever-increasing complexity in their modeling, analysis, simulation, and control. Further more, the current literature on advanced manufacturing systems indicate a need for developing modeling tools that are useful for developing integrated manufacturing control software (Chaar *et al.* 1993a,b).

Integrated control software is aimed not only to control the system but also to simulate and analyze the performance of the system (Chaar *et al.* 1993a,b). The present research is mainly motivated by the fact that there is a growing need to use advanced and formal methodologies for modeling, simulation, and control of flexible manufacturing systems (FMSs). Research and development over the last three decades have provided new theory and formal tools based on Petri nets (PNs) and related concepts for the design of supervisory controllers. Furthermore, object-oriented software engineering methods combined with PNs have great potential to develop integrated control software. However, in order to promote the industrial applications of PNs for modeling, simulation, and control of FMSs, there is a need to enrich the theory and applications of PNs, and to justify the Petri net-based control methods over traditional ladder logic diagrams-based control.

The purpose of this work is to introduce a set of Petri net-based tools and methods to address a variety of problems associated with design and implementation of FMSs.

The motivation for the present work is described below:

- Even though PNs have been successfully applied to various problems related to FMSs (Murata 1989, Silva and Valette 1990, DiCesare and Desrochers 1991, Cecil *et al.* 1992, Zhou and DiCesare 1993), there are still some areas where the power of PNs has not been exploited. For example, the application of PNs to study the performance of push and pull paradigms (Sarkar 1989) is not reported in the available literature. Such studies should help to select between push and pull paradigms and to widen the application of PNs to develop integrated control software.
- Even though there are several types of PNs available for discrete control (Crockett *et al.* 1987, Murata *et al.* 1986, Stefano and Mirabella 1991, Valette *et al.* 1983), there is a need for developing new classes of PNs that are simple, easy to understand, and handy to model factory floor operations. Furthermore, there is a need to enhance the power of timed PNs to realistically model the breakdown situations in FMSs. There are several types of PNs with a varying degree of complexity are available for supervisory control of FMSs. However, new class of PNs closer to ordinary PNs is to be developed because they are easy to understand.
- Even though there are several studies on the application of PNs for supervisory control of FMSs (Valette *et al.* 1983, Murata *et al.* 1986, Crockete *et al.* 1987, Boucher *et al.* 1989, Stefano and Mirabella 1991, Jafari 1992, Zhou *et al.* 1992a,b, Zhou and DiCesare 1993), there is a need to demonstrate the advantages of the application of PNs compared with traditional techniques such as ladder logic diagrams. Comparison studies showing their relative strengths help establish PNs as a standard tool to design sequence controllers.

- Even though PNs have been applied to design supervisory controllers for a long time (Chocron and Cerny 1980, Murata *et al.* 1986, Zhou *et al.* 1992a,b, Ferrarini 1993, Jafari 1992, Zhou and DiCesare 1993) there is a need for developing systematic methodologies that aid development of integrated control software. Such methodologies should span all control levels. The lowest level of control is at machine or cell level, where each component in the machine or cell is synchronized with other. At this level, there is a need to develop methodologies that design PN models from a given logic control specification. At the highest level of control the function of control software is not only to synchronize the activities of several cells but also to analyze the system performance. There is also a need for a methodology for development of control software using an advanced software engineering technique, object-oriented design based upon PNs. The use of PNs in object-oriented design is of immense help because PNs are very useful for modeling dynamic relations in object-oriented design.

1.2. Goals and Objectives

The present research advances the state-of-the-art in the areas discussed in the previous section. The primary goal of this work is to develop a PN-based approach to modeling, simulation, and control of FMSs. The particular sub-goals and the corresponding objectives are given next. In the following, first a sub-goal is stated and then the objectives needed to achieve that goal are presented.

1. *To analyze the state of research in FMSs and problems faced by the firms implementing FMS and to explore the benefits ensued from successful implementation of FMS.*

The objective is to review the relevant research on:

- The definition of the term "Flexible Manufacturing System,"

- The impetus for change from the traditional manufacturing systems to the FMS, and
- The major concepts, analytical models, and application and implementation problems in FMSs.

2. *To show PNs as a powerful tool in investigating difficult problems in the production management arena by investigating a complex problem often faced in the management of FMSs, i.e., comparing the performance of a push system with that of a pull system.*

The objectives are:

- To present a PN approach to address a typical operations management problem stated earlier,
- To formulate PN models (PNMs) considering important parameters in a manufacturing system example such as processing times at work cells, number of AGVs, routings of AGVs and their travel times among work cells, production and moving lot sizes, machine setup times, and machine loading/unloading times, and
- To analyze PNMs to compare the performance of a manufacturing system operating under either a *push* or a *pull* paradigm.

3. *To formulate graphically models that clearly capture the details of breakdown handling in flexible automated systems.*

The objectives are:

- To introduce a new class of PNs called Augmented-timed Petri nets (ATPNs) aimed to model conveniently breakdown handling in manufacturing systems,
- To illustrate a methodology to formulate ATPN models for breakdown handling, and

- To model, simulate, and analyze a flexible assembly system using ATPNs for estimating the optimum number of assembly fixtures for various robot breakdown rates.

4. *To compare the traditional control technique of ladder logic diagrams (LLDs) with PNs when they are used to design discrete event controllers for manufacturing systems.*

The objectives are:

- To introduce the *Real-time PN (RTPN)* which closely resembles an ordinary PN as an integrated tool to develop discrete event controllers,
- To identify the criteria to compare LLDs with RTPNs for design of sequence control,
- To compare LLDs and RTPNs in designing sequence controllers that respond to specification changes,
- To formulate mathematical formulas to calculate the number of basic elements to model certain building blocks of logic models using both PNs and LLDs, and
- To present a methodology that synthesizes analytical formulas for estimating the total number of basic elements required to design sequence controllers using PNs and LLDs.
- To present a conversion procedure to design PN models from logic control specifications.

5. *To formulate an object-oriented design (OOD) methodology for developing reusable, extendible, and modifiable control software for FMSs using object modeling technique (OMT) diagrams and PNs.*

The objectives are:

- To present an OOD methodology combining the concepts of OMT diagrams, and PNs by discussing the rationale for and advantages of selecting OOD, PNs and Ada respectively,
- To emphasize the use of PNs as a dynamic framework for OOD by discussing the advantages of PNs over earlier used dynamic models,
- To demonstrate how PNs can support the concepts of reusability and extensibility by adopting the bottom-up approach of FMS modeling, and
- To illustrate the methodology by developing object modeling technique (OMT) diagram and PN model of an FMS, and
- To demonstrate the benefits of the methodology to support reusability, modifiability, and extendibility of the software system when the configuration and specifications of the FMS are subject to change.

1.3. Organization

The next chapter presents an overview of FMSs and indicates the need for integrated modeling tools. Chapter 3 contains the discussion of the fundamentals of PNs and its applications as an integrated tool and methodology in FMS. Even though PNs have been applied for a long time to FMS, there is a need to enrich the theory and application of PNs in order to support their industrial applications in modeling, simulation, and control of FMSs. Chapter 4 shows the modeling power of PNs for performance evaluation of both push and pull paradigms in FMSs. During this performance evaluation it is assumed that breakdowns do not occur.

However, breakdown modeling is essential for the design and operation of FMSs. Hence, Augmented-timed Petri nets (ATPNs) that model breakdowns are introduced in Chapter 5. Real-time Petri nets (RTPNs) intended for real-time control of FMSs are introduced in Chapter 6. The procedure to use RTPNs for control of FMSs is illustrated

through an automatic assembly system and an electro-pneumatic system. In order to justify the PN-based control over traditional LLD-based control, comparison of RTPNs and LLDs is presented in Chapter 7. The design complexity of RTPNs and LLDs is also quantitatively evaluated through the design of different sequence controllers. Chapter 7 concludes that PNs offer an efficient approach to design sequence controllers. However, the first step to develop RTPN-based controller is the design of PN model for a given logic control specification. Chapter 8 is devoted to develop a conversion procedure to design PN models from a class of given logic control specification. After the power of PNs for simulation is shown in Chapters 4 and 5, and the efficacy of PNs for control is emphasized in Chapters 6-8, Chapter 9 proposes a methodology to design control software that integrates the applications of PNs for both simulation and control. It presents an object-oriented methodology for developing manufacturing control software highlighting PNs as a dynamic modeling tool. Finally, Chapter 10 contains a discussion on the contributions and limitations of this research along with suggestions for further research.

CHAPTER 2

FLEXIBLE MANUFACTURING SYSTEMS: AN OVERVIEW

2.1. Introduction

Increasing global competition has made many business leaders and policy makers turn their attention to such critical issues as productivity and quality. Businesses seek new approaches to production processes and manufacturing techniques and explore new boundaries of technology. One of the frequently prescribed remedies for the problem of decreased productivity and declining quality is the automation of factories. More specifically, technologies such as computer integrated manufacturing (CIM), robotics, and Flexible manufacturing systems (FMSs) are the focal points of much research and exploration. Such views as shown in the following “official” statements are representative of the new attitude toward advanced technologies:

“FMS can help our economic recovery... Flexible manufacturing systems can bring tremendous economic advantages to batch manufacturers. Beyond the attraction of increased efficiency, companies must automate if they are to compete in foreign and domestic markets with companies in Japan, Germany and other foreign countries, which are automating their manufacturing operations vigorously ” (Goldhar 1984).

Other similar views are held by business leaders on factory automation in general and FMS in particular (Goldhar 1984). Despite rapid world-wide growth of FMS installation (Darrow 1987), many manufacturers still shy away from these advanced technologies. What has kept the manufacturers from getting serious about flexible manufacturing systems is probably the improper performance evaluation criteria used to evaluate managers (Buffa 1984). While there is little disagreement about the necessity of rapid movement toward the automation of manufacturing processes and the establishment of such systems as FMS (Buffa 1984), there is also a great deal of confusion about the

fundamental approaches necessary to do so. The same is true about FMS technological complexities and the barriers to its application, implementation, and particularly, integration of FMS with rest of the operations.

The purpose of this chapter is to present an overview and a survey of research in FMS.

The objective is to review the relevant research on:

- The definition of the term "flexible manufacturing system"
- The impetus for change from the traditional manufacturing systems to the FMS
- The major concepts, analytical models, and application and implementation problems in the FMS

This chapter is organized as follows. In the next section, various definitions of FMSs are discussed. Section 3 elaborates the impetus for change from conventional manufacturing systems to FMSs. Section 4 deals with the issues related to installation, implementation, and integration of FMSs. Some of the important real-life installations of FMSs and their applications are detailed in section 5. In section 6, more emphasis is given to installation and implementation of FMSs are detailed in greater depth. Finally, in the last section, some general conclusions are drawn from the reviews and analyses and a cross-reference framework.

2.2. Definitions of FMS

Despite all of the interest in flexible manufacturing systems (FMSs), there is no uniformly agreed upon definition of the term FMS. The main distinguishing feature of FMS from traditional manufacturing systems is "flexibility" (Gupta and Buzacott 1989) which does not have the precise definition. One of the most referred to definition of FMS is by Ranky (Ranky 1983), who defines an FMS as a system dealing with high level distributed data processing and automated material flow using computer controlled machines, assembly cells, industrial robots, inspection machines and so on, together with computer integrated

material handling and storage systems. In fact, the scope and variety of flexible manufacturing is commonly disputed and are the focus of many research efforts. However, the components and characteristics of an FMS, as described by different authors and researchers, are generally as follows (Davis *et al.* 1989):

- Potentially independent NC machine tools,
- An automated material handling system, and
- An overall method of control that coordinates the functions of both the machine tools and materials handling system so as to achieve flexibility.

The specific manufacturing situations that would be suitable for the adoption of FMSs were identified as early as 1973. The following are the production situations that are encompassed by FMS (Darrow 1987):

- A variety of high precision parts are machined (typically job shop)
- A relatively large number of direct numerical control (DNC) machines are required.
- Some form of automated material handling system (MHS) is used to move the work pieces into, within, and out of the FMS.
- On-line computer control is used to manage the entire FMS under conditions of varying parts production mixes and priorities.

It can be concluded from the above that an FMS involves a number of machine centers and material handling systems integrated by a hierarchy of computer control. Furthermore, FMS is capable of randomized routing of parts instead of running parts in straight line through work stations. The term CMS (Computerized Manufacturing System) and Variable Mission Manufacturing (VMS) have also been synonymously used with FMS. In a flexible manufacturing system numerically controlled (NC) machines are controlled by computers; parts are handled by robots; and finished products are carried to specific destination via automatically guided vehicles (AGVs). Tool magazines and automatic tool changing systems are utilized, and as engineering or design changes occur in

the product, they are incorporated into the computer programs or data base. According to Klahorst (1981) FMS is a group of machines and related equipment brought together to completely process a group of or family of parts and includes the following primary and secondary components for a complete FMS:

Primary components:

- Machine tools
- A material handling system
- A supervisory computer control network

Secondary Components:

- Numerical control (NC) process technology
- Spindle tooling
- Work-holding fixtures
- Operations management

Klahorst (1981) indicates that the precise semantics of these components depends on the type of application problems to which the intended FMS will be applied. For instance, the primary components of FMS machining modules may include head changer, machining center, maxi machining center, vertical turning lathe, NC milling, NC turning, and head indexer, while the primary components for typical material handling FMS module would include roller conveyer, in/out shuttle, through shuttle, guided vehicle, shuttle car, and tow line. What seems to be the differentiating point in these definitions and elaborations is where one places the emphasis on flexibility. Some place it on flexibility in changing machine configurations which permits change of product designs without much delay. Others place the emphasis on the flexibility in handling of materials. The former group sees the FMS's potential in adaptability to demand for different product designs and the benefits of low inventory due to made-to-order production, while the latter finds the advantages in maximizing machine utilization (Mullins 1984). The nature of *flexibility* in terms of affecting the definition of an FMS is well treated in (Yilmaz and Davis 1987). Of

course, these two points of emphasis are not mutually exclusive or in conflict. Rather they can be supportive of each other. In defining FMS what is important is "understanding of what FMS represents conceptually, and what it means to a company in terms of manufacturing strategy" (Hughes and Hegland 1983).

2.3. Impetus for Change

The average annual growth rate of the U.S. market for FMSs was projected to be 27% over the years 1989-1992. The growth is due to the increased desire for change and increased interest in the automated factory. According to Robert J. Maller of Deere & Co. two issues will preoccupy the minds of manufacturing managers in the coming decades: the concern for quality and concern for cost reduction. Manufacturers will be able to pursue two goals which have traditionally been considered as conflicting and irreconcilable; low volume and low cost production in response to rapid market changes (Yilmaz and Davis 1987). However, it should be noted that this interest in the automation of factories is not new (Buffa 1984). Accordingly, there was a period of interest in automated factories during the 1950s. Then this phase of interest declined and rose again in early 1960s. Another decline followed and by 1973 there was little interest shown in the era. Now, again in the 1990s the headlines of the press represent the renewal of interest in automated factory concept. But this time the dream of the factory of the future seems to be closer to reality than ever, thanks to new technological advancements in computers, robotics, fixtures and other components of advanced manufacturing technologies. The main reason behind this new surge of attention directed to FMS and other forms of the automated factory is increased competition, particularly international. The main incentive is reduced cost in production and adaptability to an ever changing environment. Automated systems such as FMSs, have the potential to improve the position of firms on both counts.

Other reasons that account for the renewal of interest in advanced technologies include “truncation of product life cycle, and increasing complexity of products” (Goldhar 1984). For example, those companies that have installed FMS have reported the following results (Klahorst 1981):

- Benefits related to cost reduction programs (55%)
- Benefits related to market response improvement (30%)
- Benefits related to flexibility in production (15%)

Salomon and Biegel (1984) compare FMS with conventional manufacturing technology under various states of risk and show that FMS provide substantial productivity improvement. Table 2.1 summarizes these findings. The entries in the first column of this table signify the productivity improvement. The subsequent columns compare how machine and work-parts spend their time in the shop of a conventional system and in an FMS using optimistic-pessimistic format. As revealed from the table, FMS outperforms the conventional system in terms of productivity improvement.

The case of unwanted dedicated machinery explains why FMS is a necessity rather than a luxury for some manufacturers in the face of world-wide competition. R&D comes up with the operating principle for a different kind of widget. Design sets to the drafting tables and graphics terminals and arrives at part prints. Sales takes a look and projects requirements of 10,000 every year. And manufacturing, unable to economically produce the new widget on presently installed machinery, tools up to meet the exiting demand by installing the latest in high volume, hard automation machine tools. The widget in our story flops, however, as introductory products often do, and manufacturing faces the question “What do we do with all that dedicated machinery.

Table 2.1 Comparison of how machine and work-parts spend their time in the shop of a conventional system and in an FMS using optimistic-pessimistic format (Salomon and Biegel 1984)

Parameter	Conventional System Performance	FMS Performance		
		Pessimistic	Most Likely	Optimistic
Percentage of machine time the machine spends without parts	50	35	20	5
Percentage of machine time that there is a part on the machine	50	65	80	95
Percentage of time that the part is not being worked on while on the machine	70	35	21	7
Percentage of time that the part is being worked on while on the machine	30	65	79	93
Percentage of manufacturing lead time that the part spends either moving or waiting	95	92.5	90	85
Percentage of manufacturing lead time that the part spends on the machine	5	7.5	10	15

Apart from these kinds of internal problems, external factors such as changes in demand pattern and consumer tastes, changing regulatory environment and labor force, and changes in competitive policies of other firms, all contribute to the renewed efforts of manufacturers to willingly embrace new options. There seems, however, to be some over-correction of past errors. For example, some companies spend huge amounts of capital on equipment without proper justification or knowledge of how to take advantage of their potentials (Buffa 1984). Based on extensive search and analysis of empirical studies, Yilmaz and Davis (1987) presented some propositions regarding issues of flexibility, productivity, and quality. From their investigation, major findings support the premise that FMS investment leads to reduced labor cost, increased output, decreased manufacturing cost, increased flexibility, and reduced production lead time.

2.4. Installation, Implementation, and Integration of FMSs

It is estimated that 75% of machined parts produced in the U.S. are in lots of 50 units or less (Gilbert and Winter 1986). The need for small lot production has justified installation of the FMS in a number of manufacturing companies (Harvey 1984, Attaran 1992, Jari 1991, Ram and Yash 1991, Kakati and Dhar 1991). But the installation, implementation and integration of FMS create unique issues and problems. However, as a study reported in 1988 shows (Darrow 1987), only 64 FMSs have been installed in the U.S. and 253 world-wide, with the majority being in metal-cutting operations. This number, though small, has helped bring to the surface many issues regarding the installation and operation of FMSs. Subsequently more researchers and practitioners addressed these issues and made the installations of FMS easier and lucrative. Due to the concentrated research efforts in the area of FMS, the number of FMS installations have been sharply increased. A study reported in (Dimitrov 1990) shows that approximately 750 FMSs are installed in 26 countries. Another case study shows how traditional machine tools can be integrated into FMS (Kwok 1988).

Appropriateness of FMS to a given production environment is extremely important and has to be established before investment commitments are made. To help determine their fitness, the type of FMS has to be considered. Classification of manufacturing systems (with respect to their type, degree of flexibility, and volume-handling capability) helps to determine when and where an FMS is most beneficial. Flexible Manufacturing Systems can be broadly classified into dedicated FMS, sequential FMS, and manufacturing cells. Table 2.2 shows various classes FMS and the range of production volume for each classification. Klahorst (1981) provides a clear analysis of the installation/integration process of an FMS and provides some insightful guidelines drawn from the experiences of the Kearney and Trecker Company (a major producer of FMS equipment).

Table 2.2 Classification of Manufacturing Systems and FMS

Type of Manufacturing System	Level of Flexibility	No. of Parts in Product Family	Average Lot Size
Transfer Lines	Low	1-2	7,000 & Up
Dedicated FMS	Medium	3-10	1,000-10,000
Sequential or Random FMS	Medium	4-50	50-2,000
Manufacturing Cell	Medium	30-500	20-500
Stand-alone NC Machine	High	200 & Up	1-50

Some of the questions raised in this analysis with respect to the installation of FMSs are: who should do it, when should it be done, and what are the responsibilities of final users. For instance, Klahorst (1981) argues that since 50% of an FMS project value is related to machines, industrial engineers are the primary people who should be involved in the process of FMS design and installation from the start. The circumstances under which installation of FMS should be installed are obviously significant factors. According to Klahorst (1981), FMS should be installed:

- when part size and mass exceed “jib crane” standards.
- when production volume is in excess of two parts per hour.
- when processing needs more than two machine types to complete a work piece.
- when more than five machines are required.
- when phased implementation is planned so that material handling provisions can be considered in the initial phases and bad habits can be avoided from the start.

The conclusion is that the more of the above conditions exist, the more incentive there is for transforming a conventional system into an FMS.

Blumenthal and Dray (1985) caution manufacturers that factory automation such as FMS is still in its infancy and need to be focused in research. Given the fact that there are numerous uncertainties present in FMS installation, the need for an effective way to uncover the potential problems in such decisions is obvious. A very useful approach to

discover the potential problems with a system's operation is to simulate the system and pinpoint such problems. Simulation, in the case of an FMS, will a) substantially trim installation costs, b) ensure that the design of the system is accurate, and c) help spell the FMS design. Some of the major simulation studies and models related to FMS are explored in subsequent sections.

The cooperation between users and vendors has been frequently emphasized both by researchers and users/vendors of FMS as a major factor of success in implementation of FMS. For example, Hughes and Hegland (1983) observe that an FMS:

"requires a level of cooperation and exchange of sensitive business planning information between vendor and user heretofore unheard of in typical capital manufacturing equipment acquisitions." They see the relationship between the vendor and the user as a permanent one and call it "long-term partnership in productivity."

They argue that one of the most important factors contributing to the success of an FMS installation and implementation is the degree of commitment the potential company must make to be successful with FMS. Contrary to common belief, installation of an FMS is not limited to large corporations with vast financial resources. Mullins (1984) reports cases where small firms have installed FMSs that have been successful. But regardless of size, the magnitude of commitment needed is enormous.

Introducing an FMS into an organization has significant strategic implications, such as replacing "economies of scale" with "economies of scope" (Goldhar 1984). The fact that such technologies have strategic implications does not mean that their implementation has to be wholesale installation and that the FMS has to be a full-fledged system with extensive risk involved. FMS can be installed incrementally and such a move can be made through employment of stand-alone machines or the utilization of manufacturing cells. Whether FMS introduction is wholesale or incremental, "understanding the characteristics of different manufacturing systems will help recognize the potential problems associated

with the installation and implementation of them.” Black (1983) provides a useful and brief explanation of such characteristics and elaborates on different aspects of each manufacturing system that could help decide the implementation approach. Black's work is a ground work for identifying potential issues related to FMS installation. Regardless of size and scope, the question of whether an FMS is useful has significant strategic implications. Jukka and Iouri (1990) provides some guidelines for economics and success of FMSs. From a list of case studies of FMSs, they analyze costs and relative benefits of several hundred FMSs in the world. Primrose and Leonard (1991) provide a helpful overview of flexible manufacturing transfer and present a framework for investment consideration in FMS. They emphasize the significance of financial evaluation in adopting FMS and not relying only on “hopes” for future benefits. They provide guidelines as how to identify benefits related to FMS. In the same venue, Krinsky *et al* (1991) provide an analytical model for the evaluation of FMS investment, using the von Neuman-Morgenstern theory of utility together with the mean-variance approach of portfolio analysis. They identify a measure that takes into consideration both the capital cost of the new technology (FMS) and the monetary value of its output.

Analytical model building of FMS is a significant area which has to be cleared before a successful installation, integration, and implementation of the FMS is achieved. Currently, there are numerous studies related to various theoretical aspects of FMS. Stecke (1983) tackles the analytical issues related to production planning problems of FMS. The problems of grouping and loading for FMS production planning are examined in detail and formulated as mixed integer programs. Similarly, Buzacott and Yao (1986) review the basic features of FMSs and develop models for determining the production capacity of such systems. These models show the desirability of a balanced work load, the benefit of diversity in job routing if there is adequate control of the release of jobs, and the superiority of common storage for the system over local storage at machines. There are numerous simulation and analytical models dealing with various aspects of FMSs. Among them are:

perturbation analysis, queuing networks, artificial intelligence, and more recently Petri nets. However, the exploration of all these models is beyond the scope of this chapter. Interested readers are referred to some of the following chapters connected with these models: simulation (Schroer and Tseng 1985, Rolston 1985); perturbation analysis (Suri and Dille 1985); queuing networks (Suri and Diehl 1985); artificial intelligence (Dhar 1991); and Petri nets (Silva and Valette 1990, Venkatesh 1990, Raju and Chetty 1993, Zhou and DiCesare 1993). This thesis is dedicated to exploration of PN-based approaches to modeling, simulation, and control of FMSs.

2.5. Applications of FMSs

There are numerous reports and case histories about the installation of FMSs. However, accurate statistics about the application of these systems are difficult to obtain and determining the extent of FMS installations throughout the machine-tool industry is not a clear-cut task and entails many confounding issues both statistically and methodologically (Ranky 1983).

Despite these impediments, a number of trade and professional journals report case studies of FMS installation and experiences gained through them. Some of the better known firms where either partial or complete FMSs have been installed and are operational include:

- General Electric,
- Ingersall Milling Machine Company,
- GM's Pontiac Division and Saturn plant and locomotive plant in Erie, Pennsylvania,
- Chrysler's Toronto plant,
- Cadillac's Livonia engine plant,
- Ford's Sterling Heights transmission and chassis-axle plant,
- GM's Buick City,

- Hughes aircraft plant in El Segundo, California,
- Pratt and Whitney's plant in Columbus GA, and
- Allen Bradley.

What could be generalized from the reports on FMS is that in almost all cases there have been reported improvements in quality, reductions in labor and inventory costs, and increased responsiveness to the changes in the market place. An interesting report by Kaku (1992) indicates that a number of installed FMSs are “under-utilized” in the sense that the flexibility inherent in the systems installed are being used. The author, in his visit to eight establishments with installed and running FMS, only a few were utilizing the full “flexibility” of their system and the rest used the FMS as a dedicated transfer line.

Cooperative and close relationship with the suppliers has already been discussed as a main factor in successful FMS implementation. Several case studies point to the signs that suppliers of FMS are more welcome now when they approach potential clients than a few years ago. In a round-table discussion reported by Dallas (1984), all participants agreed that FMS can succeed if:

- it is functioning in the right economic context,
- the company's organizational structure has been redesigned to accommodate the special requirements of FMS,
- there is closer cooperation between vendors and users of the technology, and
- the management understands that the rules of the game have changed.

The last requirement is particularly important since it entails the mind-set reorientation of managers with respect to performance evaluation, capital rationing criteria, and human resources management. The more difficult aspects of making FMS work are the issues associated with management and organization of the systems (Klahorst 1981). A more detailed discussion of FMS implementation issues follows in the next section.

2.6. Problems in Installation and Implementation of FMSs

An FMS poses several problems in its implementation, particularly in its integration. These problems can be classified into two sets of inter-related problems: technical and managerial problems. Technical problems arise due to the complexity of the technology and technical and analytical decisions to be made in introducing advanced manufacturing systems. The managerial resistance to change is the primary reason for most of managerial problems and thus add to the complexity of implementing FMSs. Furthermore, managerial problems include “infrastructure” complexity created by such advanced technologies. Even though there are many earlier reports on problems related to FMSs (Harvey 1984, Attaran 1992, Anthony 1991, Ismael 1991), a few explore both the technical and managerial problems. The proposed approach of classifying problems connected with FMS into technical and managerial areas helps to comprehend the prominence of the combined role played by both management and technology in FMSs, and to aid researchers and practitioners in FMSs, in focussing on their specific interest. In this section, first the managerial and then the technical problems are discussed.

2.6.1. Managerial Problems

A number of studies have been devoted to the management of FMSs. National Institute of Standards Technology broadly addresses three main aspects of problems in FMS (Goldhar 1984). These are:

- How the control architecture can be simplified
- Why FMSs are difficult to configure
- What can be done to ensure a consistently high level of quality in the products

Harvey (1984) believes that the major barriers to the efficient factory of the year 2000 are financial, human, institutional, and technological. Accordingly, executives must be open to new ways to be able to financially justify investment in the factory of the future. Human and institutional barriers need to be discussed together. Among these barriers are

communication and education. The key to technological barriers, according to Harvey (1984), is integrating the individual parts of the factory, which can be accomplished through Computer Integrated Manufacturing (hence the relationship between FMS and CIM). Harvey (1984) provides a four-step procedure as a guideline to building a supportive organizational climate for the factory of the future:

1. Educate senior management,
2. Set goals and develop strategies,
3. Establish a corporate-wide culture, and
4. Develop an unified communication structure

Fear of change is another problem associated with implementation of FMS. Kiesler (1983) deals with this issue under the broad subject of the impact of new technology on the work place. In a broader sense, French (1984) discusses problems and issues concerning the CIM. Several of these issues are applicable to FMS as well. According to French (1984) the following are the major problems faced by manufacturers that may lead to failure in CIM (or FMS) implementation.

- Inadequate measurement system
- Partially obsolete facilities
- Inadequate data base
- User hostility
- Shortage of technical skill
- Incompatibility between systems
- Management generation gap
- Changes in management philosophy
- Facilities with mixed processing
- Dynamic volume and mix
- Outdated organization
- Varieties of process options

- Loss of superior/subordinate support

Similarly, Scalpone (1984) considers the lack of education in both management and technical staff to be one of the major factors contributing to the failure of advanced technologies. In summary, the main managerial barriers of successful implementation of an FMS seem to be concentrated in few major areas. These areas include:

- Lack of top management commitment and support
- Inadequate training and education of the personnel involved
- Improper evaluation of the situation/environment which presumably justified installation of the FMS
- Lack of long-term and committed relationship with the vendors of both raw material and the FMS equipment
- Lack of total commitment to the installation and implementation of the FMS
- Existence of misconceptions about FMSs (such as FMS being good only for large companies and only applicable to large scale production).

2.6.2. Technical Problems

The smooth and economical functioning of any FMS depends upon the effectiveness of strategies for designing, controlling and monitoring of FMS. This section briefly describes the typical and important technical problems encountered in the successful implementation of FMS. In the following section, selected past and current research concerned with each of these problems are summarized and some suggested solutions are reviewed. Since problems related to FMSs crosses the boundaries of manufacturing engineering, industrial engineering, computer engineering, computer science, electrical engineering, and operations management, it is difficult to cover each and every problem in great detail. The approach adopted in this section is based on presenting a synopsis of some selected chapters and major results. However, numerous references are provided for the readers who desire to pursue more detailed coverage of specific problems.

Even though some researchers have broadly divided problems of FMS into planning, designing, and controlling (Suri and Dille 1985, Stecke 1989), and yet others (Imman 1991) generally discuss issues relating to FMS implementation, much specific and detailed scrutiny of such problems is still needed. Understanding of the problems associated with the successful implementation of FMS requires a closer look at each and every problem more specifically and in greater depth. In order to address the specific problems in FMS, the following three step approach is adopted:

1. Considering all of the functional subsystems constituting FMS and understanding the prominence of each subsystem.
2. Addressing problems related to each subsystem.
3. Analyzing the impact of each subsystem on the system as a whole.

This section details some of the most common and important functional subsystems present in an FMS:

- CNC/DNC machine tool technology
- Tool management
- Automated material handling
- Communication network to integrate elements present in the FMS (real-time control issues are also included here)

The problems related to CNC/DNC machine tools require deeper understanding of the machine tool technology itself and falls beyond the scope of the present work. Therefore, problems specific to CNC/DNC machine tools are not discussed here.

Tool management

The various definitions of tool management available in the literature may be due to diversified aspects by which the tool management is viewed. Some researchers define it in terms of four distinct sets of activities: 1) stock control and administration, 2) functional control, 3) handling and transportation, and 4) programming (Gray and Stecke 1988). It is also defined as a strategy which aims at resolving problems related to various tool

activities, including: a) acquisition, b) storage, c) database development, d) selection and allocation, e) inspection, f) presetting, g) delivery, h) loading, i) monitoring, j) replacement, k) requirement planning, and inventory control of tools (Reddy *et al.* 1990a). Despite the various orientations and approaches, there are some common goals in tool management. These goals are discussed in the following paragraphs.

Tooling is one element in FMS that is most prone to change due to external factors and may often cause discrepancy in FMS functioning. In other words, the probability of change in the tooling due to change in one or all of the external requirements (see Section 3), is much greater than the probability of changing any of the other internal elements. The changes in tooling include changes either in the number of each tool type and/or number of different tool types, and/or tool position in FMS (tool crib, magazines, etc.). The extent of individual or combined effect of external factors on tooling changes with respect to its number of tools of each type and the variety of tools can be enormous due to the huge tooling (several number of tools of several tool types) in most of the FMSs. Further, the influence of any external factor on tooling with respect to its position can become more complicated by tool flow from tool crib to inspection, to presetting stations, to tool magazine, and finally to the machine spindle. The constant wear and tear on tools adds to the complexity of the situation mainly when FMS is operating for a long period of time. Hence, tools are more prone to replacements during the long time functioning of FMS. Due to the factors and importance of tool management explained above, tooling is the most dynamic and critical facility in FMS and requires keen attention.

Tool management is a very complicated task and is often stressed by FMS users and researchers (Gray and Stecke 1988, Reddy *et al.* 1990a,b). Despite such complexity, there are successful working FMSs whose performance have been considerably augmented with efficient tool management (Gaymon 1986, Tomek 1986). Tool management also attains paramount significance considering its economic impact, since tooling accounts for 25-30% of the fixed costs of production in automated machining environment (Tomek

1986, Cumings 1986). Several firms have recently developed integrated tool management systems with tremendous encouraging results (Tomek 1986). In fact, the centralized tool management has introduced the fifth generation of FMS environment, indicating that tool management is one of the important subsystem in the FMS which influences the whole structure and operation of the FMS (Heywood 1988).

Owing to the complexities in tool management, Maccarini *et al* (1987), have suggested different tool room layouts comparing their performances and have identified the parameters which describe better tool room behavior while the production process is being developed. However, the recent literature reveals that there are still many tool management problems to be solved (Reddy *et al.* 1990b, Zavanella *et al.* 1990), . This is mainly due to the lack of comprehensive understanding of tool management which is necessary before attempting any tool related activities such as development of algorithms for tool optimization, development of control software, design of a tool delivery system, and framing of a new storage or tool flow strategy. Modeling of the tool management can be of critical help in this respect and aids in understanding of complex asynchronous concurrent interactions/tasks in tool management (Venkatesh and Chetty 1992). A more recent survey on tool management in FMSs is made by Veeramani *et al.* (1992). They have examined tool management related research efforts in academia and outlined the characteristics of comprehensive tool management systems that are being used in industry.

Automated material handling

Recently, automated guided vehicle systems (AGVSs) have received increased attention by the designers and engineers of automated manufacturing systems (Gould 1990). An AGVS may consists of multiple automated guided vehicles (AGVs) and computers to control them. AGVs are unmanned vehicles that carry workpieces among the workstations following guide paths. They are usually controlled either by on-board computers or by a central computer. The communication between an AGV and its controller is generally established through dedicated wiring embedded in the floor, although some recent AGVSs

utilize wire-less communication. AGVSs are widely used in FMSs as they provide flexibility in routing parts among elements present in the system. These systems are highly complex and expensive due to the dynamic environment in which FMS functions. Furthermore, if the AGVS is not efficient, the whole system performance may be impaired by the possible starvation of machines in the system.

A variety of analytical methods have been proposed by researchers for the design and control of AGVSs. Egbelu and Tachoco (1986), Mahadevan and Narendran (1990), Maxwell and Muckstadt (1982), and Wysk *et al.* (1987) have proposed alternate procedures for estimating the number of AGVs required. Hodgson *et al.* (1987) devised an analytical control strategy for scheduling AGVs which was extracted from Markov decision process optimal control policies. King *et al.* (1989) evaluated heuristic control strategies for a system under varying arrival patterns. Bozer and Srinivasan (1989) suggested a tandem configuration for reducing software and control complexity of an AGVS. Malmborg (1990) developed an analytical modeling strategies that can be applied for both design of AGVS and performance evaluation of FMSs. The application of these analytical methods is limited due to their restrictive assumptions and hence they can not effectively be used for controlling the FMS under investigation. This is because the above mentioned analytical approaches can not predict the dynamic behavior of the AGVS with respect to time. Often simulation studies are preferred over analytical approaches as they provide more insight into the problem and they are not hindered by too many assumptions. A number of simulation studies have been conducted to evaluate the effect of different parameters such as number of AGVs, number of pallets, buffer sizes, dispatching rules, bi-directional flows etc. (Egbelu and Tachoco 1984, 1986, Newton 1985, Ozden 1988, Vosniakos and Mamalis 1990). Vosniakos and Mamalis (1990) have also addressed design issues of AGVSs for FMSs such as zone blocking, loading/unloading, and traffic control. Schroer and Tseng (1985) have demonstrated the use of GPSS in modeling AGV based FMSs. The main limitation of the available simulation methods are that they are time

consuming and need large computer memory. Hence, they are limited in their practical usefulness for on-line monitoring of AGVS.

Owing to the limitation of the available analytical and simulation methods, there is a need for a single and versatile tool for addressing both design and operational control issues of AGVSs. Petri Nets (PNs), a powerful modeling tool in the context of FMSs have been successfully used for such analyses. Alanche *et al.* (1984) modeled AGV movements with Petri nets (PNs) and qualitatively investigated the deadlocks and possible vehicle collisions. Davis *et al.* (1989) used PNs to formalize rules for allocating and dislocating the zones in an AGVS. Archetti and Sciomachen (1989) have analyzed an AGVS quantitatively using PNs. Occena and Yokota (1991) modeled an AGV system in a Just-In-Time environment and introduced a new dispatching rule to have better inventory and transport control, compared to the traditional dispatching rules available. From their study they concluded that traditional dispatching rules such as shortest processing time, first-in-first-served, etc. do not perform well in the FMS environment. Hence, they concluded there is no specific rule that performs better in an FMS that is operating under Just-In-Time environment. This is mainly because in such systems, processing on machines starts only when there is a demand and processed parts are dispatched only when the successive machine is ready to take the part from its preceding machines.

Although there is extensive literature on hardware issues, research concerning the design and control issues of AGVSs for FMSs is limited. In order to fully exploit the increased flexibility and adaptability by AGVSs, an in-depth study of the design and control issues is essential. Also, for the realistic analysis of AGV-based FMSs, both the machine scheduling and AGV scheduling have to be simultaneously investigated (Sabuncuoglu and Hommuertzheim 1986).

Control system development among elements present in the FMS

A fundamental building block of FMS is data communications. Communication development in FMSs is probably the single largest and most troublesome problem area encountered by users and suppliers of factory automation systems. A study by General Motors revealed that upto 50% of the cost for new automation projects was directly attributed to communication and control devices (Balph and Vittera 1985). This is because, the complete flexibility and efficiency of any FMS can be realized only when a systematic and reliable control system is developed to coordinate and monitor different activities taking place at the different levels in the system hierarchy (Jones and McLean 1986). Even with adequate automation equipment, an FMS may not live up to its performance potential due to the lack of appropriate integration and control of the FMS operations (Maimon 1987). Furthermore, this area of factory automation has become a test bed for establishing various network architectures, protocols, access methods, communication media, and band widths for many computer vendors. Hence, communication and control system development among various elements in FMS is very important. However, development of such systems in FMS is a complicated task because:

- An FMS is a complex distributed processing system where each element in an FMS has a data base and there are several such elements in FMS that have to communicate for the manufacturing of a finished product.
- Communication and compatibility among the equipment of different vendors is critical. These equipment include computers, local area networks (LANs), cell controllers, programmable logical controllers (PLCs), robots, machine tools, and similar digital control devices.

To alleviate the complexity of communication development, many efforts are underway in Europe and the USA to develop a common set of vendor-independent communication protocols which would be usable by all types and brands of factory automation equipment. Towards this aim, Manufacturing Automation Protocol (MAP) has

recently been developed. The objective of MAP is to make communication possible between mainframe systems, cell controllers, workstation terminals, programmable logic controllers, material handling systems, robots and other types of factory equipment. MAP has received some criticism by researchers and users because of its complexity. For example, MAP specification is suitable at plant level communications, but many times becomes too heavy at cell level.

Typical static data sets in an FMS include configurations of machines, AGVs, and robots and their characteristics and process plans. Common dynamic data sets include status of machines, AGVs, and robots that change with respect to real time. The control algorithms act as an interface between the static and dynamic data sets at different levels of hierarchy in the FMS. Maimon (1987) extensively discussed the tasks of the operational control in such an interface. He also presented a generic hierarchical control system and illustrated the development procedure with an example of flexible manufacturing cell to carry out the FMS integration from the production requirements to the actual operation of the machines.

However, the last task of the operational control system, namely, continuous monitoring and handling of breakdowns is not detailed. For example, there is no mention of about how tool wear is monitored in real time. Tool wear monitoring is essential to achieve uninterrupted machining and to schedule tool replacements when the limit on tool life is reached. The complexity of tool wear monitoring and its significance in the context of FMS functioning is highlighted by Masory (1991) and Venkatesh *et al.* (1994). Handling of machine and/or robot breakdowns in manufacturing systems is also essential to minimize the system down time and thereby not impairing the system performance. In case of breakdowns, there are many concurrent actions to be taken to bring back the system to normal functioning.

Furthermore, some design parameters such as the number of assembly fixtures required to achieve maximum production rate may change with and without consideration of breakdowns in the system as addressed later in this work. In addition, some important control and monitoring issues have to be addressed in detail during breakdown handling. However, there are a few reported studies that consider breakdown handling in the analysis of a manufacturing system.

For example, the importance of breakdown handling is highlighted and breakdown of work stations is considered in modeling of flexible manufacturing system by Barad and Sipper (1988) and Sheng and Black (1990). However, neither of these papers present the detailed modeling and analysis of the system with breakdowns.

Jafari (1992) presented an architecture for a shop-floor controller using Petri nets. However, the proposed control system architecture is not applicable to assembly systems. Many authors (Jones and McLean 1986, Maimon 1987, Jafari 1992, Ghosh 1989) have developed hierarchical control structures for FMS control as it supports information collapsing and data abstraction at different levels in the factory. The major objectives of the communication network in FMS are resource and information sharing. The design of computer networks for FMS is treated by Ghosh (1989).

In the same paper, Ghosh developed a methodology for performance evaluation of a network. Ghosh illustrated the design and performance evaluation methodology for three different configurations of networks, namely, direct numerical control (DNC), computer numerical control (CNC), and multi-level control hierarchy with local area network. Zhou *et al.* (1994), presented an overview of various models for performance evaluation of communication networks in manufacturing. They also presented a Petri net method for modeling and performance of token bus local area networks.

There are a few studies that propose heterarchical control structure for automated manufacturing systems. Duffie *et al.* (1988) presented a heterarchical control structure after describing the limitations of hierarchical control structure. They illustrated the heterarchical control scheme with an experimental manufacturing system.

Even though the proposed scheme in Duffie *et al.* (1988) is simple and easy to understand, it can not be easily expanded when the FMS grows in complexity (several machines, robots, and AGVs). Jones and McLean (1986) proposed a hierarchical control model for automated manufacturing systems, which serves as a research test bed to aid in the identification, design and testing of standards for the automated factories of the future. They also emphasized that a standard factory model must address all of the necessary functional, control, data flow, and interface issues.

This chapter has shown that the research on FMS is vast and growing. In order to help the readers focus on their area of interest, the research reviewed in the present chapter has been classified and the results are shown in Table 2.3. Although the references cited in Table 2.3 overlap in various areas of FMS (such as managerial , tool management, etc.), the *primary* focus of the papers reviewed has been used as the classification criterion.

The existing research in the area of FMS indicates the need for developing integrated tools to address not only all the technical problems discussed above but also help solve all the managerial problems discussed earlier. Such integrated tools would be of immense help to integrate the efforts of both technical and managerial personnel which is very essential for the full realization of FMS benefits.

Tools that are useful to specify, model, design, evaluate, control, and monitor FMSs are urgently required. Such tools serve as a common medium among several personnel involved in the above activities. Hence, using such tools the integration between the people and between the tasks of the system can be easily achieved. The integration between people in FMSs is of paramount importance for the success of such systems.

Table 2.3 Cross-references to research related to problems in FMS

Managerial Problems	Related Research References:	Technical Problems	Related Research References:
Installation, implementation, and integration issues	Goldhar (1984), Darrow (1987), Klahorst (1981), Mullins (1984), Hughes (1983), Gilbert (1986), Harvey (1984), Attaran (1992), Jari (1992), Ram (1991), Dimitrov (1990), Kwok (1988), Bluementhal (1985), Black (1983), Jukka (1990), Primrose (1991), Krinsky (1991), Stecke (1983), Buzacott (1986), Kaku (1992), Kiesler (1983), French (1984), and Scalpone (1984).	Tool management	Gaymon (1984, 1987), Reddy (1990a,b,c), Gray (1988), Tomek (1986), Cumings (1986), Heywood (1988), Maccarini (1987), Zavanella (1990), and Venkatesh (1992), Veeramani (1992).
		Automated material handling	Davis (1989), Schroer (1985), Gould (1990), Egbelu (1984, 1986), Mahadevan (1990), Maxwell (1982), Wysk (1987), Hodgson (1987), King (1989), Bozer (1989), Malmborg (1990), Newton (1985), Ozden (1988), Vosniakos (1990), Alanche (1984), Archetti (1989), Occena (1991), and Sabuncuoglu (1986)
		Control system development among elements present in the FMS	Balsh (1985), Jones (1986), Maimon (1987), Masory (1991), Barad (1988), Sheng (1990), Venkatesh (1992, 1994), Jafari (1992), Ghosh (1989), Duffie (1988), and Zhou (1994)

2.7. Summary

In this chapter the literature on definitions of FMS, reasons for change from conventional systems to FMS, installation and implementation issues of FMS, applications issues, and finally problems of FMS were briefly reviewed. The FMS-related problems were categorized into two major areas -managerial and technical. Both managerial and technical problems are discussed along with the earlier research. There is a vast source of materials on the subject, although the interest and the technology of the field is relatively recent. There are a number of predictions and forecasts in regard to the future of FMS. For example, Hughes and Hegland (1983) reported the result of a Delphi study with regard to the future of FMS. This report predicted major advances in various areas of FMSs such as:

- The level of application of automated fixturing and holding devices on numerically controlled machines,
- Reliable and practical sensing strategies for implementing adaptive control in all current metal-cutting operations.
- Extension of flexible production systems to machine tool industry,
- In-process adaptive control of surface roughness in machining,
- Non-contact high-speed on-line inspection systems with closed loop feedback to the machine control system, and
- Increased application of diagnostic components in FMSs

Many of these predictions are close to reality or have already been realized. Whenever an FMS is installed the experiences gained through the installation, implementation, and integration of the system are shared by the industry making the growth rate self-accelerating. However, despite these predictions and growth, the myopic views of some managers toward capital expansion in such areas as FMS have caused delay in widespread utilization of these systems (Hays and Wheelwright 1984). On the other hand, there are some positive trends.

The use of flexible machining cells will be of primary concern throughout most of the manufacturing industries. Integrated modeling tools that address both technical and managerial problems are essential for the success of manufacturing systems. This chapter has made a solemn attempt to comprehensively highlight the issues and problems related to FMSs starting from their planning to implementation. There is an increasing need for integrated modeling tools for the development of control systems in FMSs. PNs are claimed to be ideal integrated modeling tool in the area of FMSs (Cecil, *et al.* 1992, Proth 1992, Zhou and DiCesare 1993). The next chapter introduces the concepts of PNs and presents a survey of PN applications in FMSs.

CHAPTER 3

PETRI NETS AS AN INTEGRATED TOOL AND METHODOLOGY IN FMSs

Petri nets(PNs) are a powerful graphical and mathematical modeling tool to solve many problems related to asynchronous concurrent systems (Murata 1989). They have gained popularity as a versatile tool for addressing design issues related to FMSs (Silva and Valette 1990). In this chapter the fundamentals of PNs and its applications in FMSs are discussed.

3.1. Concepts and Terminology of Petri Nets

A PN is defined as a bipartite graph containing *places* (pictured by circles) and *transitions* (pictured by bars). Places and transitions are connected by *directed arcs* (pictured by arcs with arrows). Places contain *tokens* (pictured by black dots). The distribution of tokens in the places of a PN is called its *marking*. Sometimes, *weights* (pictured as labels on the arcs) may also be used to facilitate the modeling. If there is no weight on an arc, unit weight is assumed. Places can model different entities comprising the system, such as robots, machines, and AGVs; and different intermediate states of the system, e.g., *Robot_i_is_ready_to_load_part_j*. Transitions can model activities involved in the system, e.g., *Robot_i_loads_part_j*. Directed arcs model information, material, and control flow in the system. A token in a place represents the true value of the conditions modeled by the place.

All places that have arcs leading into (out of) a transition are said to be that transition's input (output) places. A transition is said to be *enabled* if each of its input places has at least a number of tokens equal to the weight of the arc leading from that place to the transition. An enabled transition *fires* by removing a number of tokens equal to the weight of the corresponding input arc from each of its input places and depositing a number of tokens equal to the weight of the corresponding output arc in each of its output places.

Hence, firing a transition changes the token distribution over some places, which in turn changes the marking of the PN. Each marking of PN models a unique state of the system. Hence, as each transition in the PN fires, each unique state of the system can be easily determined by observing its PN. For example, if there are 40 tokens in a place modeling the *buffer_between_machines_i_and_j*, the work-in-process-inventory between these two machines is said to be 40. The fundamentals of PNs can be seen in (Murata 1989, DiCesare and Desrochers 1991, Silva and Valette 1990, Zhou and DiCesare 1993). To maintain consistency the following PN definitions are given.

Formally, a PN Z is a five tuple, $Z = (P, T, I, O, m)$ where

1. P is a finite set of places;
2. T is a finite set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;
3. $I: P \times T \rightarrow N$, is an input function that defines the set of directed arcs from P to T where $N = \{0, 1, 2, \dots\}$
4. $O: P \times T \rightarrow N$, is an output function that defines the set of directed arcs from T to P ;
5. $m: P \rightarrow N$, is a marking whose i th component represents the number of tokens in the i th place. An initial marking is denoted by m_0 .

The execution rules of a PN include enabling and firing rules:

1. A transition $t \in T$ is *enabled* if and only if $m(p) \geq I(p, t)$, $\forall p \in P$
2. Enabled in a marking m , t *fires* and yields a new marking m' following the rule:

$$m'(p) = m(p) + O(p, t) - I(p, t), \forall p \in P.$$

The marking m' is said to be *reachable* from m . Given Z and its initial marking m_0 , the reachability set is the set of all markings reachable from m_0 through various sequences of transition firings. Important PN properties related to stability, repetitiveness, and absence of deadlocks can be defined and their implications for the system behavior are reported in (Narahari and Viswanadham 1985, Zhou and DiCesare 1993).

Timed Petri Nets

Association of time with transitions in the PN described above results in Timed PN. Formally, a Timed PN (TPN) is a net Z in which each transition is associated with either a deterministic or random firing delay time. Note that the random time delays may follow standard probabilistic distributions. There are two events for a transition firing, i.e., *start_firing* and *end_firing*. Between them, the firing is active. The removal of tokens from a transition's input place(s) occurs at *start_firing*. Their deposition to a transition's output place(s) occurs at *end_firing*. While a transition's firing is active, the time to end firing, called the *remaining firing time* decreases, from firing duration to zero at which its firing is completed. *Instantaneous description (ID)* (Venkatesh *et al.* 1990) defining a state of a TPN is a four tuple $ID = (m, F, Q, A)$ where:

1. m is a marking;
2. F is a *binary selector function*, $F: T \rightarrow \{0,1\}$. If $F(t) = 1$, t is enabled, otherwise disabled;
3. $Q: T \rightarrow R^+$, is *remaining firing time function*. If $Q(t) = q$, there is q amount of time to complete firing t . Q is a cumulatively decreasing time function;
4. $A: T \rightarrow R^+$, is *active time function*. If $Q(t) = q'$, t is said to be active for q' amount of time. A is a cumulatively increasing time function.

ID is useful for the quantitative and behavioral analysis of the system. To illustrate the PN concepts, an assembly cell shown in Fig. 3.1 (a) is modeled. It contains 2 part feeders (PF1 and PF2) and a robot (R). Feeders supply parts for assembly and robot does assembly operations. PF2 feeds a part to the empty assembly area automatically. The system operates as follows: 1) to start a cycle, robot (R) and parts must be available, 2) R transfers a part from PF1 to assembly area and starts assembly, timing duration is one time unit, and 3) R assembles the parts, and transfers the finished product to the output buffer, the time is two units. Figure 3.1 (b) shows the PN model for this assembly cell.

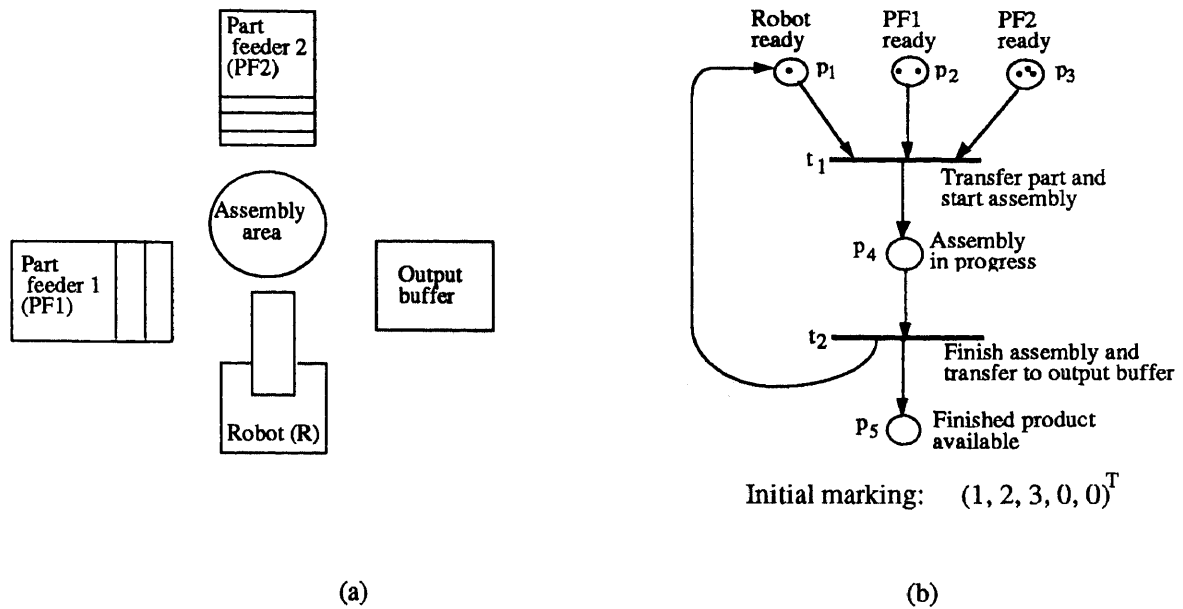


Figure 3.1 (a) A simple assembly cell (b) Petri net model

Figure 3.2 shows various TPN models in a chronological order. At time zero (see Fig. 3.2(a)), initial marking models the initial condition; F-function indicates that t_1 is enabled; Q-function models that there is one time unit necessary to finish t_1 's firing; A-function means that there is no transition active.

After one time unit (i.e. after assembly starts), the change in marking is shown in Fig. 3.2(b) indicating that R is not ready because it is doing assembly, PF1 and PF2 contains one and two parts respectively, assembly is in progress, and there is no finished product. F-function shows that t_2 is enabled, Q-function shows that t_2 needs two time units to complete its firing, and A-function shows that t_1 is active for one time unit. Similar explanations can be given for PN models in Figs. 3.2(c) and 2(d). The software package developed in (Venkatesh *et al.* 1992) for the simulation of PN models is used in this work. This is developed in Ada and is presented in Appendix A.

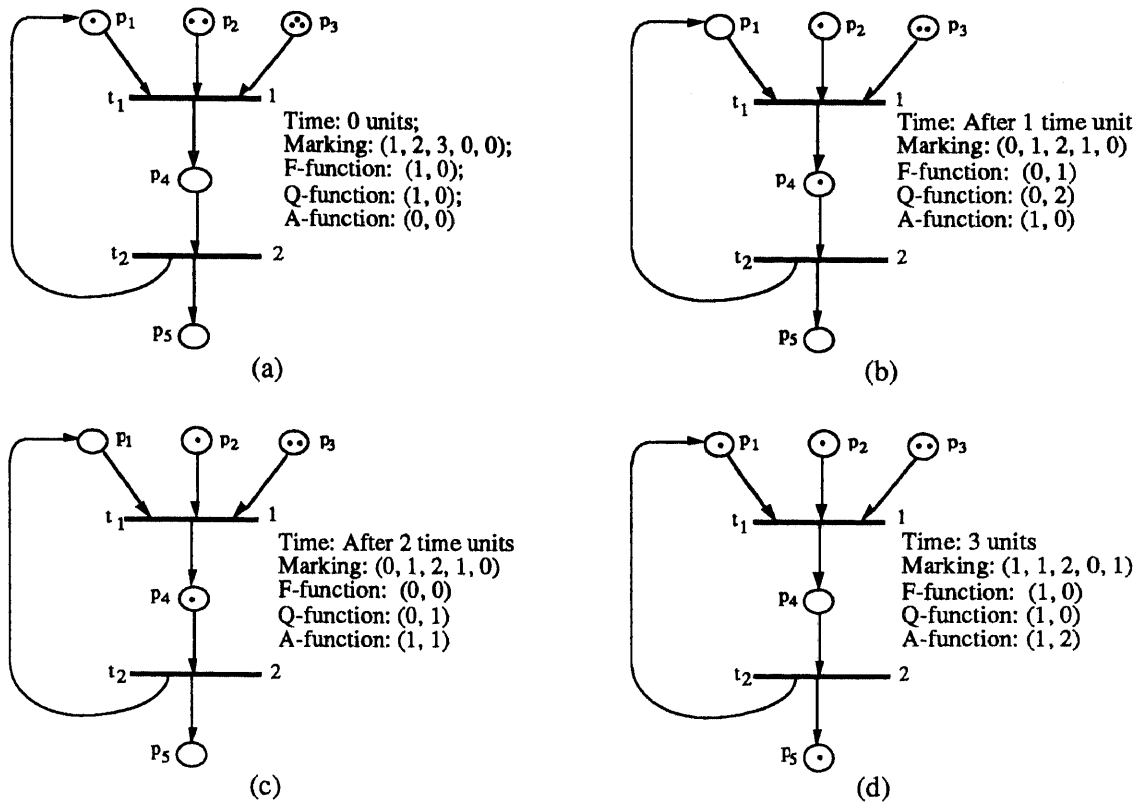


Figure 3.2 Timed Petri net model of the assembly cell
 (a). Before firing of transition 1 (before assembly started)
 (b). After firing of transition 1 (after assembly started)
 (c). During firing of transition 2 (during assembly)
 (d). After firing of transition 2 (after assembly finished)

3.2. Applications of Petri Nets in FMSs

Tools such as queuing networks, Integrated computer aided manufacturing DEFinition (IDEF), perturbation analysis, mathematical programming, and simulation packages like SIMAN, SLAM, and XCELL can offer solutions for the problems related to manufacturing system design. This work instead explores PNs to address the same issues. More recently, Kochikar and Narendran (1994) classified the main approaches for modeling and analysis of FMSs. They evaluated the available models against certain criteria such as: modeling and decision power, ability to model at different levels, model verifiability, ability to represent system evaluation, efficacy on computational considerations, quality of results, interactivity, and ease of understandability and use.

Based on these criteria, Kochikar and Narendran (1994) concluded that PNs and their extensions score highly among the available models. The advantages of PNs over other tools include: hierarchical graphical modeling, direct generation of control code from the models, validation of control code with either analytical or simulation methods, qualitative property analysis, performance analysis, and real-time control and monitoring (Stecke 1989, Vernon and Zahorjan 1987, Zhou, *et al.* 1992, DiCesare and Desrochers 1991, and Kaighobadi and Venkatesh 1994). The disadvantage of PNs is that the models become bigger when complex systems are modeled. However, in order to eliminate this disadvantage "colored PNs" (Murata 1989) can be used. Another disadvantage of PNs is the absence of a standard class of PNs that can address all the issues related to modeling, simulation, and control of FMSs.

Many surveys dealing with the implementation of FMS projects report that the integration between people working on the projects is as much as essential as integrating different activities of production (Huang and Sakurai 1990). System designers, manufacturing managers and industrial, software, and production engineers are the typical personnel involved in designing and implementing the FMS projects. Since these personnel have different backgrounds, powerful, versatile, and easier modeling tools are needed to achieve successful integration among their efforts and objectives. That PNs offer such a tool to achieve this purpose is evident from its diversified applications in FMSs.

Typical applications of PNs in manufacturing include performance evaluation (Al-Jaar and Desrochers 1990); modeling, validation, and analysis (Narahari and Viswanadham 1985, Venkatesh 1990); specification and implementation (Courvoisier *et al.* 1989); discrete event control design (Zhou *et al.* 1992a,b, and Zhou 1993); and simulation and scheduling (Valavanis 1990, Venkatesh 1992). They were used to address tool management issues (Reddy *et al.* 1992), model Automated Storage/Retrieval Systems (Knapp and Wang 1992), design of Automated Guided Vehicle Systems (Raju and Chetty

1993), and to determine the optimal number of kanbans in Just-In-Time manufacturing systems (Jothishankar and Wang 1992). Table 3.1 summarizes the typical applications of PNs in FMSs. Moreover, the enthusiasm of various researchers in FMSs has shown that large scale industrial application of PNs is close (Berchi and Frosi 1988). Realizing the potentials of PNs, many researchers are actively exploring the new applications of PNs in FMSs. In the following paragraphs, the applications of PNs and the further scope of PNs in each specific application area are summarized. It is noted that all the applications discussed below are important to demonstrate the use of PNs as a tool for modeling, simulation, and control.

3.2.1. Simulation and Performance Evaluation

Mascolo *et al.* (1991) analyzed the limitations of the available models and used PNs to model various types of kanban systems. They have modeled single kanban systems and dual kanban systems using PNs and concluded that PNs are suitable for performance evaluation of manufacturing systems operating with JIT principles. However, they have not presented the quantitative performance results of the manufacturing systems that were modeled by PNs. Sheng and Black (1990) modeled a cellular manufacturing system operating with JIT principles using PNs.

Jothishankar and Wang (1992) determined the optimal number of kanbans in JIT manufacturing using stochastic PNs. Kochikar and Narendran (1992) proposed modified colored PNs to enhance the power of ordinary PNs and built compact PN models of FMSs. Their proposed modifications render the PN to be more dynamic to reflect the currently selected system parameters. Venkatesh *et al.* (1992) applied timed PNs for scheduling of robots in an FMSs operating with JIT principles. Raju and Chetty (1993) used another extension of PNs, called Priority nets to model FMSs that utilize AGVs for material handling.

Table 3.1 Major applications of PNs in FMSs

Applications of PNs in FMSs	Typical personnel involved	Typical references
Specification of system requirements, model development and validation	Industrial engineers and system designers	Courvoisier <i>et al.</i> (1989), Venkatesh (1990)
System simulation	Manufacturing, industrial and system engineers	Valvanis (1990); Righini (1993); Venkatesh <i>et al.</i> (1990)
System design and performance evaluation	Industrial engineers and system designers/analysts	Raju and Chetty (1993), Chan and Wang (1993), Venkatesh <i>et al.</i> (1994a)
Scheduling of machines, AGVs, and robots	Production engineers and Operation managers	Venkatesh <i>et al.</i> (1992), and Raju and Chetty (1993)
Formal specification tool for software design and development	Software, industrial, and manufacturing engineers	Guha <i>et al.</i> (1987), Booch (1990), Venkatesh <i>et al.</i> (1994c)
Database and communication development	Computer hardware and software engineers	Murata (1989)
Real-time control, monitoring, and diagnostics	Control and manufacturing engineers	Zhou and DiCesare (1989), 1993, Srinivasan and Jafari (1991), Venkatesh <i>et al.</i> (1994b)
Management of facilities in a system	Operation and system managers	Reddy <i>et al.</i> (1992)

By assigning attributes to the places which can be dynamically changed depending upon the tasks of AGVs, they have provided a dynamic scheduling algorithm to schedule AGVs in FMSs. Using Priority nets, they have also analyzed the performance of FMS corresponding to various scheduling rules and product mixes.

Reddy, *et al.* (1993) used timed PNs for modeling tool management systems in FMSs. Using timed PNs, in a centralized tool crib management environment, they have compared the performance of a FMS with and without tool sharing among machining centers. Chan and Wang (1993) combined the concepts of high level PNs and stochastic

PNs and constructed PN models for the performance evaluation of an FMS. Righini (1993) introduced 'modular PNs' for simulation of FMSs and presented an algorithm that allows automatic composition of subnets into a larger model. Wang and Hafeez (1994) used generalized stochastic PNs to investigate the different policies followed in the traffic management of automated guided vehicles (AGVs) in FMSs. Using stochastic PNs they have compared the performance of tandem and conventional AGV systems in FMS. Hsieh and Shih (1994) also used modularized floor-path nets to model AGV systems in FMSs and presented rules for combining PN models to preserve the desirable properties of PNs. Kochikar and Narendran (1994) proposed a new class of PNs, called Conditional Predicate/Transition nets by modifying high level PNs and illustrated their use for the simulation of FMSs. Conditional Predicate/Transition nets allow dynamic switching of arcs to facilitate the dynamic selection of operating policies.

However, none of the above papers showed the suitability of PNs to investigate the design and performance issues of FMSs functioning under either *push* or *pull* paradigms. Also, earlier works of PNs in FMSs addressing push and pull paradigms (Mascolo *et al.* 1991, Jothishankar and Wang 1992, Venkatesh *et al.* 1992, Yim and Linn 1993) have not explicitly presented (i) the differences between PN modeling of push and pull paradigms, and (ii) modeling of production and moving lot sizes. Chapter 4 investigates the application of PNs for performance evaluation of push and pull paradigms in FMSs.

3.2.2. Breakdown Modeling

Although PNs are proved as a tool to solve a variety of problems relating to manufacturing systems, their full application to address design and analysis issues of FAS with breakdowns remains to be explored. PN modeling of breakdowns was considered (Barad and Sipper 1988, Sheng and Black 1990). Barad and Sipper (1988) used timed PNs for modeling breakdown of a machine considering issues such as the rerouting flexibility of

assembly fixture and repair and maintenance of the breakdown machine. The focus of their paper was to illustrate the PN model considering a machine breakdown while illustrating the flexibility of modeling a flexible manufacturing system (FMS) with PNs. However, they have not presented the performance evaluation of a manufacturing system considering breakdowns of machines. Sheng and Black (1990) modeled a PN for machine breakdown while demonstrating the application of PN in a cellular manufacturing system. They have considered the failure of a tool as a breakdown of machine and modeled how such situations can be easily modeled with PNs. However, they have not presented the quantitative analysis indicating the effect of the failure rate of tools on the production rate and buffer sizes of the system. In addition to the above authors, Huang and Chang (1992) also modeled the breakdowns in an FMS using an inhibitory arc concept.

The performance of a transfer line was analyzed by considering the breakdown of machines using stochastic PNs (Al-Jaar and Desrochers 1990). Stochastic PNs are also used for the performance analysis of a flexible assembly system considering various robot failure rates (Zhou and Leu 1991). None presented the detailed breakdown handling procedures and optimization issues for various machine/robot breakdown rates.

Furthermore, all earlier researchers considered only breakdowns that arrive before starting of an activity. However, in the real life situations breakdowns may arrive when an activity is in progress. This restricts their accuracy for analysis of a realistic assembly system. Chapter 5 investigates the application of PNs for breakdown modeling.

3.2.3. Discrete-Event Control

Recently, Petri nets (PNs) have been increasingly applied to design discrete-event control systems for manufacturing systems. Zhou and DiCesare (1989) proposed different error recovery schemes to ensure uninterrupted production in automated manufacturing systems. Jafari (1992) proposed an architecture for controlling FMSs using colored PNs. Bruno

and Marchetto (1986) proposed a new class of PNs for the rapid prototyping of process control systems. Boucher *et al.* (1989) illustrated the method of controlling a simple manufacturing cell using PNs. Hitachi developed a commercial product based on an enhanced PN that interacts with the physical system to control (Murata *et al.* 1986). PNs have been also applied to model and implement local area networks in FMSs (Venkatesh and Ilyas 1995). The reasons for the increase in the use of PNs for FMS control are discussed in (Boucher *et al.* 1989, Zhou *et al.* 1992).

Realizing the potential of PNs for control, in France, GRAFCET - a PN like representation tool is proposed as a standard specification of sequence controllers (David and Alla 1992). Its international standard version is called sequential function charts (Falcione 1993). More details and advantages of PNs for controlling manufacturing systems can be found in (Boucher *et al.* 1989, and Zhou and DiCesare 1993). More recent studies of PNs for discrete event control can be seen in (Ferrarini 1992, Venkatesh *et al.* 1993, 1994, Zhou *et al.* 1992a, 1992b, Zhou and DiCesare 1993). However, none of the earlier studies have in detail compared ladder logic diagrams (LLDs) and PNs for the design of sequence controllers. Boucher, *et al.* (1989) used LLDs and PNs to control the same manufacturing system and reported the graphical representation by PNs makes the controller more tractable than that of LLDs. However, they have not formally quantified the comparison between PN and LLDs to design sequence controllers. The sequence controller in this work means a class of discrete event controllers without choices in executing the operations/activities. The detailed comparison of LLDs and PNs is very important in realizing their advantages and disadvantages and, particularly, in establishing PNs as an emerging design technique for effective sequence control of industrial automated systems. Chapters 6 and 7 introduces Real-time PNs (RTPNs) and compares PNs and LLDs, respectively.

One critical task in the development of PN based controller is to design PN models given the sequence control specifications. Also, one of the factors for the comprehensive comparison of PNs and LLDs in discrete event control is the availability of standard procedures for designing controllers. There exists systematic design procedures for designing LLDs (Pessen 1989). Even though there exist several methods to formulate a PN model using bottom-up, top-down, hybrid (Zhou et al. 1992a, and Zhou and DiCesare 1993), and incremental approaches (Ferrarini 1992), they are difficult to directly apply to generate automatically the PN models from certain logic control specification. Hence, for the large scale application of PNs in industry, there is a need for systematic design procedures for developing PN models. Chapter 8 presents a procedure to formulate a PN model from a given class of logic control specification.

3.2.4. Control Software Development

Excellent reviews are presented on recent methods for developing manufacturing control software and concluded that current methodologies are not sufficient to support planning, scheduling, and monitoring activities involved in manufacturing (Chaar *et al.* 1993a,b). Realizing the inherent complexity in controlling a flexible manufacturing system, object-oriented programming is proposed for control software development (Hsu 1992). Venkatesh and Fernandez (1992) combined the concepts of PNs, OOD, and Ada to develop control software for FMSs. Haidar, et al. (1994) combined PNs and OOD approaches to develop control software for FMSs. They use statecharts of FMSs to develop PNs from which they develop a hierarchy of controllers. In all the earlier studies that used OOD for control software development the benefits of OOD such as reusability and extendibility of the software system were not sufficiently demonstrated (Glassey and Adiga 1990, Hsu 1992). Also, the earlier researchers (Chaar et al. 1993b, Glassey and Adiga 1990, Hsu 1992) have not paid much attention to describing the dynamic behavior of the objects

present in the software system and to analyze the system performance. Furthermore, earlier OOD studies lack systematic OOD methodologies which help select the data structures and operations of objects and models the dynamic behavior of objects. Even though PNs and OOD are separately used for developing control software, there is a need for systematic methodology that emphasize the role of PNs as a dynamic modeling tool in OOD.

Hence, keeping in mind the limitations of PNs and their further scope in the above applications, the following chapters not only introduce new classes of PNs but also develop methodologies that support the modeling, simulation, and control of FMSs using PNs.

CHAPTER 4

PERFORMANCE EVALUATION OF PUSH AND PULL PARADIGMS IN FLEXIBLE AUTOMATION

4.1 Introduction

Even though PNs have been successfully applied to various problems related to FMSs (Murata 1989, Silva and Valette 1990, DiCesare and Desrochers 1991, Cecil *et al.* 1992, Zhou and DiCesare 1993), there are still some areas where the power of PNs has not been exploited. For example, the application of PNs to study the performance of push and pull paradigms is not reported in the available literature. Such studies not only help to select between push and pull paradigms but also aid to widen the application of PNs for modeling and simulation of flexible manufacturing systems.

This chapter shows PNs as a powerful tool to investigate the problem often encountered in manufacturing systems management, namely comparing the performance of a factory automated system operating under *push* and *pull* paradigms. The difficulty in solving this type of problem is compounded by many parameters such as processing times at work cells, number of automated guided vehicles and their routings, lot sizes, and setup times. The PN method to solve such a problem is illustrated by considering a manufacturing system. Its PN models are formulated and then analyzed to compare the performance of system with *push* and *pull* paradigms. The results show that for the particular system and operational parameters, the *push* paradigm outperforms the *pull* one.

Manufacturing systems consist of machines, robots, and automated guided vehicles (AGVs) that are controlled by computers. Numerous asynchronous concurrent actions involved in these systems make their analysis difficult. This chapter focuses on the modeling and analysis issues related to manufacturing systems that are operated according to either *push* or *pull* paradigms.

The primary goals of a manufacturing system are to minimize the work in process inventory (WIP), and maximize the system utilization and output rate (Stecke and Solberg 1981, Suri 1984, Chang et al 1985, and Maione *et al* 1986). In the manufacturing arena, it is well known that low WIP can be achieved by implementing just-in-time (JIT) which is based on a *pull* production paradigm. However, implementation of JIT may lead to lower system utilization (Monden 1981). In contrast to the *pull* paradigm *push* production paradigm results in maximum system utilization and output rate, at the expense of higher WIP (Monden 1989, Sarkar and Fitzsimmons 1989).

Even though it is popular to divide production control systems into push and pull systems, there are no generally accepted definitions for these systems (Spearman, et al. 1990). However, several authors distinguished push and pull paradigms (Kimura and Terada 1981, Schonberger 1983, Spearman, et al. 1990). For example, Kimura and Terada (1981) discussed them as mechanisms of production orders. According to them, in push system production and inventory control is based on the forecast value. In pull system, a certain amount of inventory is held at each stage and its replenishment is ordered by the succeeding process at the rate it has been consumed. Also, there are several ideas to implement push and pull systems (Schonberger 1983, Spearman, *et al.* 1990, Lee 1987). Schonberger (1983) described the implementation of a pull paradigm using kanban techniques, and stated that push was simply a schedule based system which could be implemented using a master production schedule that was exploded by a computer into detailed schedules.

Spearman, *et al.* (1990) stated that push systems scheduled the start of jobs whereas pull system authorized the production. They reported that pull systems were not applicable to production environments which were controlled by job orders. Motivated by this, they described a pull-based production system called CONWIP and presented its advantages over push and some pull systems. Lee (1987) presented a parametric appraisal of a JIT system and concluded that, unlike the traditional push method, raising the pull

demand in a JIT system did not ensure a high process utilization level. The details of differences between the *push* and *pull* ones were also discussed in detail by Sarkar and Fitzsimmons (1989). In this chapter, we regard push and pull paradigms as operational paradigms. In a push system, a preceding machine produces parts without waiting for a request from the succeeding machine. On the other hand, in a pull system a preceding machine produces parts only after it receives a request from the succeeding machine. The difference in modeling such systems is described in detail in subsequent sections.

To minimize WIP and maximize the system utilization and output rate of manufacturing systems simultaneously, it is often difficult to select between *push* and *pull* paradigms. This is because for certain system configurations and operational parameters, *push* may perform better than *pull* and vice versa. To make the best decision, detailed design and performance analysis of a system has to be performed.

Another reason that adds to the difficulty of the present problem is the inadequacy of research on JIT in the area of flexible manufacturing systems (FMSs). In other words, even though there is much reported research in both the areas of JIT (in conventional manufacturing systems) and FMSs individually, there are only a few research studies on implementing JIT in FMSs (Venkatesh *et al.* 1992a). Huang (1984) emphasized the uncertain consequences of the system output, the sensitivity in performance of a JIT production system to lot size, the impact of setup time reduction on the efficiency, and the effect of variability of processing times on the JIT implementation and its subsequent performance. Sarkar and Fitzsimmons (1989) investigated the effects of variations in processing times on the performance of conventional *push* and *pull* systems. Yim and Linn (1993) analyzed an FMS with Petri nets and concluded that there was no significant difference in output rate between push and pull based AGV-dispatching rules for a busy FMS. However, none of the earlier chapters have presented any methodology for studying the effects of variations of lot sizes and processing times on the performance of *push* and *pull* systems. This chapter presents PNs as a suitable tool to answer these questions.

The unique features of this work compared to earlier works on PNs in manufacturing systems are presented in subsequent sections. The primary goal of this chapter is to show PNs as a powerful tool in investigating difficult problems in production management arena. It investigates a complex problem often faced in the management of manufacturing systems, i.e., comparing their performance functioning under *push* and *pull* paradigms. The specific objectives of this work are:

1. To present a PN approach to address a typical operations management problem stated earlier,
2. To formulate PN models (PNMs) considering important parameters in a manufacturing system example such as processing times at work cells, number of AGVs, routings of AGVs and their travel times among work cells, production and moving lot sizes, machine setup times, and machine loading/unloading times, and
3. To present the detailed analysis of PNMs to design and compare the performance of manufacturing systems operating under either *push* or *pull* paradigms.

4.2. Application Illustration

For the performance evaluation of the system, timed PNs are used in this chapter. Instantaneous description (ID) of timed PNs described in Chapter 3 is used here for the quantitative analysis of the system. A PN approach to analyzing a system consists of two parts: modeling with PN and analysis of the PNM by either analytical methods (Narahari and Viswandaham 1985) or simulation (Dubois1989 and Valvanis 1990). The simulation method is applied in this chapter with the software package presented in Appendix A. The detailed conventions of PN modeling useful for the design, performance evaluation, and monitoring of manufacturing systems are shown in Table 4.1.

Table 4.1 Conventions of Petri net modeling

Implementation issue	PN modeling
Setup time	Firing duration of the transition modeling setup activity
Conveyance time	Firing duration of the transition modeling conveyance activity
Moving lot size or production lot size	Weight of the directed arc modeling the function of moving or production kanban
Routing of an AGV	Firing sequence of transitions
Number of AGVs for a transportation task, work stations and pallets	Initial marking in the corresponding places modeling AGVs, workstations and pallets
Values of inprocess inventories	Number of tokens deposited in the places modeling input and output buffers of workstations
Utilization times of workstations, robots, AGVs	Active time durations of transitions modeling the activities of work stations, robots, AGVs
Production volume	Number of tokens deposited in the place modeling 'counter for production volume'
Dynamic system state useful for design, performance analysis, and monitoring and control of the system	Marking of the PNM giving the token distributions, F, Q, and A functions giving the status of transitions with respect to time

4.2.1. System Configuration and Assumptions

A manufacturing system example as shown in Fig. 4.1 contains four workcells (WCs) and an assembly shop (AS). Each WC consists of a machining cell (MC), an assembly cell (AC) and a robot (R). AGV(s) are present in the system to transport parts and subassemblies among work cells and the assembly shop (AS). The track layout of AGVs is also shown in Fig. 4.1. The conveyance times among work centers and assembly shop to produce products 1 and 2 (PR1 and PR2) are shown in Table 4.2. There is a need for producing two product varieties, PR1 and PR2, both in the quantity of seventeen each (the demand of seventeen products is chosen arbitrarily). PR1 requires parts A, B, C, and D in the quantities of 2, 1, 3, and 2, respectively.

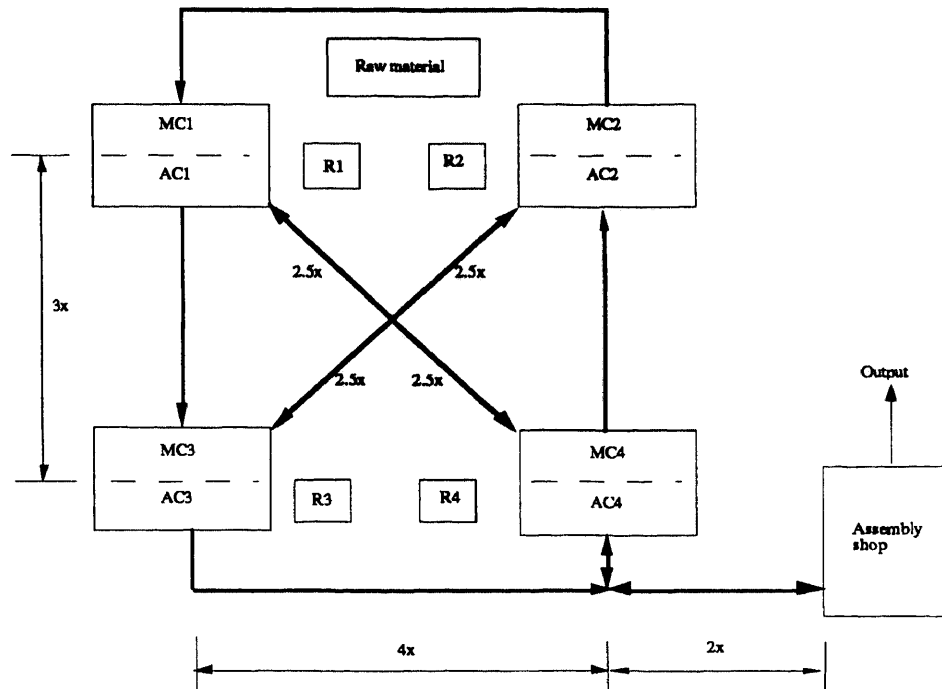


Figure 4.1 Layout of the manufacturing system investigated

Table 4.2 Conveyance time matrix in the system for production of PR1 and PR2 (time units)

	WC1	WC2	WC3	WC4	AS
WC1	–	4	3	–	–
WC2	–	–	5	5	–
WC3	5	–	–	4	6
WC4	5	3	5	–	–
AS	7	–	2	–	–

These parts are machined by MC1, MC2, MC3, and MC4. PR2 requires subassemblies SA, SB, SC, and SD in the quantities of 2, 1, 3, and 2, assembled by AC1, AC2, AC3, and AC4 respectively. The machining/assembly sequence of above parts and subassemblies is same as shown in Table 4.3 along with the corresponding times. The values in parentheses in Table 4.3 show the assembly times for PR2. Since PR1 requires only machining operations, the manufacturing system becomes a flexible manufacturing system (FMS) when it is producing PR1.

Similarly, the system is actually a flexible assembly system (FAS) when it is producing PR2. This identification schema allows for generally comparing the performance of FMS and FAS when both of them function under similar system configuration but with different processing times.

Table 4.3 Processing times and the sequence of parts in the system (time units)

Part	Sequence	MC1 (AC1)	MC2 (AC2)	MC3 (AC3)	MC4 (AC4)
A (SA)	MC1--MC3 (AC1--AC3)	35 (8)	—	42 (6)	—
B (SB)	MC1--MC3 (AC1--AC3)	46 (10)	—	38 (9)	—
C (SC)	MC2--MC4 (AC2--AC4)	—	26 (7)	—	34 (11)
D (SD)	MC2--MC4 (AC2--AC4)	—	30 (9)	—	40 (12)

The assumptions made about this system are as follows:

1. The outer path of the AGV track is unidirectional and the inner one is bi-directional.
2. The setup time for any MC or AC and loading/unloading a MC or AC is one time unit and sequence independent.
3. If there are more than one AGV in the system, the delays due to traffic are negligible.
4. There are no breakdowns in the system during its operation. However, detailed breakdown handling is possible with 'Augmented-timed PNs' (Venkatesh, *et al.* 1994a) as described in the next chapter.

Statement of the problem

The manufacturing system considered performs in four different system configurations, namely, FMS and FAS with the *pull* paradigm and FMS and FAS with the *push* one. In each configuration there are several parameters to be studied. The parameters whose impact on the system performance has to be investigated in this study are production lot sizes (PLSs), number of unique transportation tasks (N) which decides the moving lot size (MLS), and the number of AGVs assigned for each unique different transportation task (n). For example, PLS of Part A is defined as the number of A parts that are processed on MC1

before changing the setup of MC1 to produce another different part (here Part B). MLS is defined as the number of parts that an AGV carries corresponding to each unique transportation task. The routing of an AGV and the variation of MLS with respect to N are shown in Table 4.4. The goals are minimization of work-in-process-inventory and maximization of system utilization and output rate. To achieve these goals, PNMs of system for the four different configurations have to be formulated and the influence of the parameters on the goals has to be investigated. Also, the optimum values of N, n, and PLSs have to be determined for each system configuration. Finally, the performance of system under *push* and *pull* paradigms has to be compared.

Table 4.4 Variation of moving lot size with respect to number of AGVs and assigned tasks

Number of unique different transportation tasks (N)	Task of AGV and its routing	Moving lot size (MLS)
1	AGV1: Starts from work cell 1 (WC1), delivers parts from WC1 and WC2 to WC3 and WC4, delivers parts to assembly shop (AS) and returns to WC1 Routing: WC1-WC2-WC3-WC4-WC3-AS-WC1	$2 + 1 + 3 + 2 = 8$
2	AGV1: Starts from WC1, delivers parts from WC1 and WC2 to WC3 and WC4 and returns to WC1 Routing: WC1-WC2-WC3-WC4-WC1	$2 + 1 + 3 + 2 = 8$
	AGV2: Starts from WC4, delivers parts from WC4 and WC3 to AS and returns to WC4 Routing: WC4-WC3-AS-WC4	$2 + 1 + 3 + 2 = 8$
3	AGV1: Starts from WC1 and delivers parts to WC3 and returns to WC1 Routing: WC1-WC3-WC1	$2 + 1 = 3$
	AGV2: Starts from WC2, delivers parts to WC4 and returns to WC2 Routing: WC2-WC4-WC2	$3 + 2 = 5$
	AGV3: Starts from WC4, delivers parts from WC4 and WC3 to AS shop and returns to WC4 Routing: WC4-WC3-AS-WC4	$3 + 2 + 2 + 1 = 8$

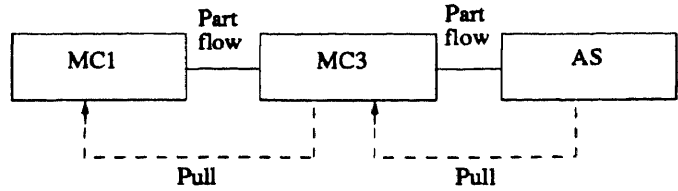
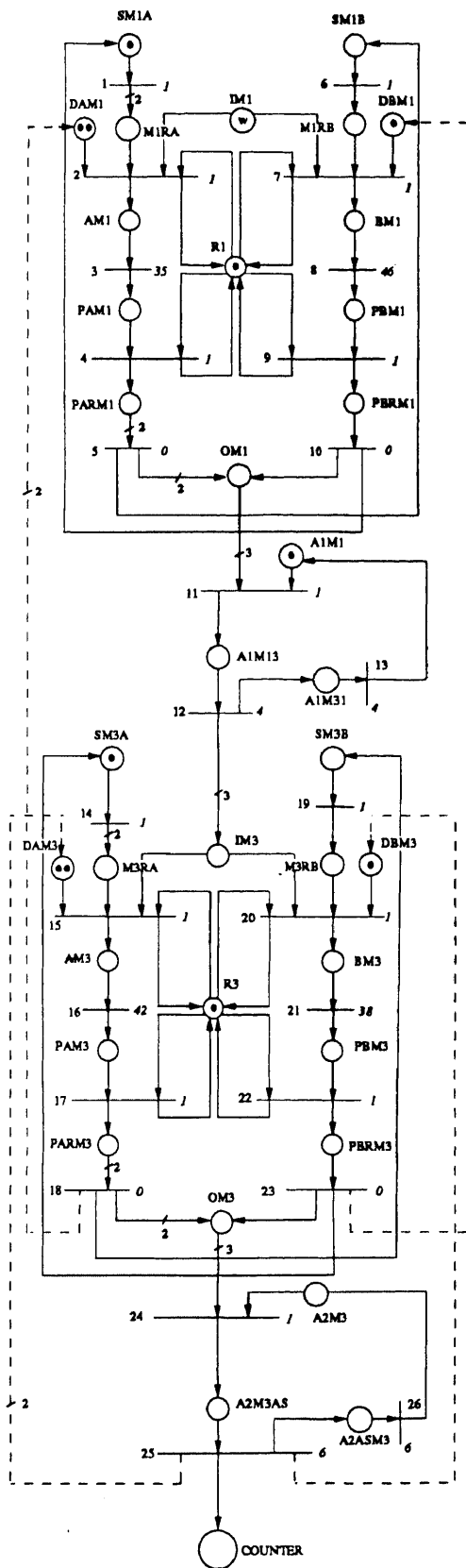
4.2.2. PNM Formulation and Analysis

PNM for *pull paradigm*

To illustrate the concepts of PN modeling Fig. 4.2 shows the PNM for MC1 and MC3 operating with the *pull* paradigm to produce parts A and B corresponding to PR1. In Fig. 4.2, dotted arcs model the *pull* paradigm. The modeling of production of parts C and D by MC2 and MC4 is similar to production of parts A and B by MC1 and MC3 except that the PLSs for C and D are different. However, the variation of PLSs can be easily modeled by changing certain weights in the PNM as illustrated in Fig. 4.2 and discussed in next paragraphs. Hence, the discussion on modeling production of parts C and D is omitted here. Also, to simplify the modeling process, in Fig. 4.2, it is assumed that there is AGV1 which transfers parts from MC1 to MC3 and AGV2 which transfers parts from MC3 to assembly shop. Since time duration of operations in the system are deterministic, transitions are associated with deterministic firing durations (from Tables 4.3 and 4.4).

Processing of Part A

It is assumed that initially Part A has to be loaded on to MC1. This requires MC1 to be setup to process A modeled by depositing a token in place *setup_for_MC1_to_process_part_A*. Setup time is one unit of time and associated with t_1 modeling the activity *machine_setup*. After one time unit the completion of setup is modeled by firing t_1 and depositing the number of tokens equal to the production lot size (PLS) of part A in place *MC1_ready_to_process_part_A*. PLS is modeled as a weight on the output arc from t_1 to place *MC1_ready_to_process_part_A*. Since MC1 is functioning with the *pull* paradigm, MC1 cannot start processing until there is a demand for Part A from MC3. Hence, MC1 has to wait till MC3 requests MC1 to process Part A. The request for Part A at MC1 arrives when MC3 completes processing of Part A. The completion of processing Part A at MC3 is modeled by t_{18} , *number_of_parts_as_specified_in_final_assembly_are_ready_in_MC2's_output_buffer*.



PLACES:

- SMiA: Setup_for_MCi_to_process_part_A (i = 1,3)
- MiRA: MCi_ready_to_process_part_A
- DAMi: Demand_for_part_A_at_MCi
- IMi: Input_buffer_of_MCi_with_parts_ready_to_feed_MCi
- Ri: Ri_ready
- AMi: Part_A_loaded_on_MCi's_table
- PAMi: Processed_part_A_on_MCi's_table
- PARMi: Processed_part_A_ready_at_MCi
- OMi: Output_buffer_of_MCi_ready_with_parts_A_and_B
- AjMi: AGVj_ready_at_the_output_buffer_of_MCi (j = 1,2)
- A1M13: AGV1_traveling_from_MC1_to_MC3
- A1M31: AGV1_traveling_from_MC3_to_MC1
- SMiB: Setup_for_MCi_to_process_part_B
- MiRB: MCi_ready_to_process_part_B
- DBMi: Demand_for_part_B_at_MCi
- BMi: Part_B_loaded_on_MCi's_table
- PBMi: Processed_part_B_on_MCi's_table
- PBRMi: Processed_part_B_ready_at_MCi
- A2M3AS: AGV2_traveling_from_MC3_to_AS
- A2ASM3: AGV2_traveling_from_As_to_MC3
- Counter: Counter_for_noting_production_volume

TRANSITIONS:

- 1,6,14,19: Machine_setup
- 2,7,15,20: Loading_by_Robot
- 3,8,16,21: Completion_of_part_processing
- 4,9,17,22: Unloading_by_Robot
- 5,10,18,23: Number_of_parts_as_specified_in_the_final assembly_are_ready_in_MCi's_output_buffer
- 11: AGV1_starts_from_MC1_to_MC3
- 12: AGV1_reaches_MC3
- 13: AGV1_returns_to_MC1
- 24: AGV2_starts_from_MC3_to_AS
- 25: AGV2_reaches_AS
- 26: AGV2_returns_to_MC3

NOTE:

1. w = Number of tokens modeling the availability of sufficient raw material
2. Transition times are shown for FMS case
3. Initial marking is shown
4. $\overset{k}{\underset{r}{|}} \quad \begin{matrix} k \\ r \end{matrix}$ k is transition number
r is time units
5. PLS for A and B: 2 and 1
6. Dotted arcs model the concept of the pull paradigm

Figure 4.2 PNM for production of parts A and B under pull paradigm

Hence, the request for MC1 to process Part A from MC3 is modeled by an output arc from t_{18} to place *demand_for_part_A_on_MC1*. However, before the system begins production at MC1, i.e. before the processing of Part A by MC3 (before firing of t_{18}), demand for Part A should be present. This is modeled by depositing a certain number of tokens in place *demand_for_part_A_on_MC1*. The number of tokens in this place is controlled by production lot size (PLS) of Part A. Since, the PLS for Part A is 2, the initial marking of place *demand_for_part_A_on_MC1* is also two. Once the production starts, to continuously model the *pull* from MC3 to MC1, the output arc from t_{18} to place *demand_for_part_A_on_MC1* is drawn with weight two (PLS for Part A).

Whenever MC3 completes processing it fires t_{18} and sends a request for MC1 to produce two parts of A. The raw material before MC1 is modeled by depositing sufficient number of tokens in place *input_buffer_of_MC1_with_parts_ready_to_feed_MC1* as its initial marking. Now processing of Part A on MC1 can start since there is demand for Part A, MC1 is ready to process Part A, and raw material is available. But, since robots are used to load the machining cells, MC1 starts processing as soon as Robot 1 (R1) loads the raw material. This is modeled by t_2 , *loading_by_robot*. After R1 completes loading, Part A is loaded onto the MC1's table. This is modeled by firing t_2 and depositing a token in place *part_A_loaded_on_MC1's_table*. The completion of processing of Part A on MC1 is modeled by firing t_3 , *completion_of_part_processing* and putting a token in place *processed_part_A_on_MC1's_table*. The unloading operation of the processed Part A by R1 is modeled by firing t_4 , *unloading_by_robot* and depositing a token in place *processed_part_A_is_ready_at_MC1*. MC1 continues to produce another Part A since there is still one token left in place *demand_for_part_A_on_MC1*. When MC1 finishes processing of second Part A, there would be two tokens present in place *processed_part_A_is_ready_at_MC1*. Hence, there are two processed parts of A as specified in the final assembly (for the production of PR1, two parts of A are required according to the bill of materials of PR1) are available at the output buffer of MC1. This is

modeled by firing t_5 , with an input arc from *processed_part_A_ready_at_MC1* with weight two and depositing two tokens in place *output_buffer_of_MC1_ready_with_parts_A_and_B*. The state changes in the system can be simulated and visualized by token movements in the PNM.

Processing of Part B

Upon the completion of processing two parts of A, MC1 starts to process Part B. This requires changing the setup of MC1 which is modeled by an output arc from the t_5 to place *setup_for_MC1_to_process_part_A*. The modeling methodology for processing Part B is similar to that of Part A. Once Part B is processed, MC1 has to produce again Part A and the change-over for the setup is modeled by an output arc from t_{10} to place *setup_for_MC1_to_process_part_A*. After the completion of processing of Part B, the output buffer of MC1 contains two parts of A and one part of B.

This is modeled by firing t_{10} and depositing a token in place *output_buffer_of_MC1_ready_with_parts_A_and_B*. Now, these parts have to be transferred to MC3 for further processing. AGV1 is used to unload the parts from the output buffer of MC1 and transport to MC3. The presence of AGV1 at MC1, is modeled by depositing a token in place *AGV1_ready_at_MC1*.

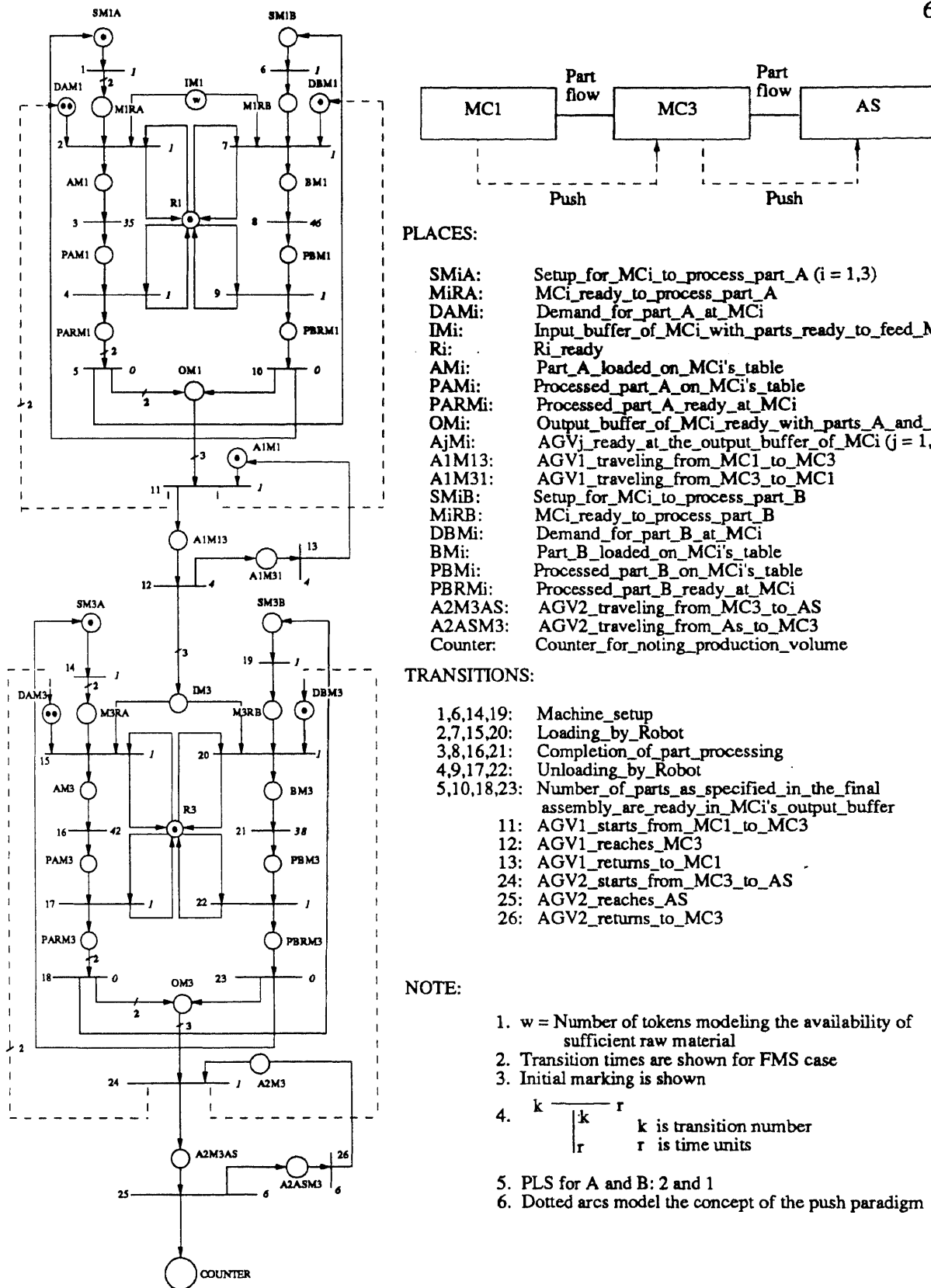
Since, AGV1 cannot move till the output buffer of MC1 contains two parts of A and one part of B, the input arc for t_{11} modeling *AGV1_unloads_and_starts_at_MC1* is labeled with weight three (summation of the PLSs of parts A and B). As soon as place *output_buffer_of_MC1_ready_with_parts_A_and_B* contains three tokens, AGV1 starts conveying parts from MC1 to MC3. This is modeled by firing t_{11} and depositing a token in place *AGV1_travelling_from_MC1_MC3*. After the conveyance time between MC1 and MC3, AGV1 reaches MC3 and unloads the parts at the input buffer of MC3. This is modeled by firing t_{12} and putting three tokens in the place *input_buffer_of_MC3_with_parts_ready_to_feed_MC3*.

The modeling methodology for processing of parts A and B by MC3 is similar to MC1 except that the demand for parts at MC3 is generated by the assembly shop and AGV3 is used to transfer parts between MC3 and the assembly shop (AS). Observe that PNM's model the functions of production and moving kanbans in JIT. For example, in the PNM shown in Fig. 4.4, only after MC3 finishes the processing of Part A (t_{18}), it gives a signal (moving kanban function modeled by the output arc from transition to the place DAM1) to MC1 to start processing. Then t_2 gives a signal (production kanban function) to MC1 to start production.

PNM for *push* paradigm

Fig. 4.3 shows the PNM's for MC1 and MC3 operating with push paradigm for parts A and B. In Fig. 4.3, dotted arcs model the *push* paradigm. The modeling methodology for *push* paradigm is similar to that of the *pull* one except the way tokens are deposited in places *demand_for_part_A_on_MC1* and *demand_for_part_B_on_MC1*. The same PNM used for the *pull* paradigm (shown in Fig. 4.2) is slightly modified to analyze the system operating with the *push* paradigm and is shown in Fig. 4.3. Notice that in Fig. 4.3 the dotted arcs represent the only change compared to Fig. 4.2. That is, output arcs from t_{18} and t_{23} in Fig. 4.2 are removed and attached at t_{11} in Fig. 4.3. Also, output arcs from t_{25} in Fig. 4.2 are removed and attached at t_{24} . In the *push* paradigm MC1 produces parts without waiting for a request from downstream machine. Hence, as soon as AGV1 leaves MC1, MC1 starts processing. This is modeled by each output arc from t_{11} to place *demand_for_part_A_on_MC1* and *demand_for_part_B_on_MC1*.

Similarly, MC3 produces parts without waiting for a request from AS. Hence, as soon as AGV2 leaves MC3, MC3 starts processing which is modeled by two output arcs from t_{24} as shown in Fig. 4.3. The PNM for whole manufacturing system is obtained by duplicating the similar modeling methodology to MC2 and MC4.



PLACES:

- SMiA: Setup_for_MCi_to_process_part_A (i = 1,3)
- MiRA: MCi_ready_to_process_part_A
- DAMi: Demand_for_part_A_at_MCi
- IMi: Input_buffer_of_MCi_with_parts_ready_to_feed_MCi
- Ri: Ri_ready
- AMi: Part_A_loaded_on_MCi's_table
- PAMi: Processed_part_A_on_MCi's_table
- PARMi: Processed_part_A_ready_at_MCi
- OMi: Output_buffer_of_MCi_ready_with_parts_A_and_B
- AjMi: AGVj_ready_at_the_output_buffer_of_MCi (j = 1,2)
- A1M13: AGV1_traveling_from_MC1_to_MC3
- A1M31: AGV1_traveling_from_MC3_to_MC1
- SMiB: Setup_for_MCi_to_process_part_B
- MiRB: MCi_ready_to_process_part_B
- DBMi: Demand_for_part_B_at_MCi
- BMi: Part_B_loaded_on_MCi's_table
- PBMi: Processed_part_B_on_MCi's_table
- PBRMi: Processed_part_B_ready_at_MCi
- A2M3AS: AGV2_traveling_from_MC3_to_AS
- A2ASM3: AGV2_traveling_from_As_to_MC3
- Counter: Counter_for_noting_production_volume

TRANSITIONS:

- 1,6,14,19: Machine_setup
- 2,7,15,20: Loading_by_Robot
- 3,8,16,21: Completion_of_part_processing
- 4,9,17,22: Unloading_by_Robot
- 5,10,18,23: Number_of_parts_as_specified_in_the_final
assembly_are_ready_in_MCi's_output_buffer
- 11: AGV1_starts_from_MC1_to_MC3
- 12: AGV1_reaches_MC3
- 13: AGV1_returns_to_MC1
- 24: AGV2_starts_from_MC3_to_AS
- 25: AGV2_reaches_AS
- 26: AGV2_returns_to_MC3

NOTE:

- 1. w = Number of tokens modeling the availability of sufficient raw material
- 2. Transition times are shown for FMS case
- 3. Initial marking is shown
- 4.
$$\begin{array}{c} k \quad r \\ | \quad | \\ k \quad r \\ | \quad | \\ k \quad r \end{array}$$
 k is transition number
r is time units
- 5. PLS for A and B: 2 and 1
- 6. Dotted arcs model the concept of the push paradigm

Figure 4.3 PNM for production of parts A and B under push paradigm

The transportation of material among workcells is unique depending on the value of N and hence as N changes the routing of AGV changes as specified in Table 4.4, which in turn slightly changes the PN modeling of material transfer among workcells in the system. The number of AGVs for each unique transportation task, n is modeled by varying the initial marking of places, *AGV_ready_at_the_output_buffer_of_MC*.

To avoid redundancy, only the PNM of the system functioning as FMS under the *pull* paradigm with the values of both N and n equal to one is shown in Fig. 4.4. The interpretations of places and transitions are shown in Table 4.5. Figure 4.5 shows the PNM corresponding to the system functioning with the *push* paradigm. Initial marking is shown in all PNMs. The PNMs used for FMS can also be used when System functions as FAS. In such a PNM, the only difference is that the machining times associated with transitions are replaced by assembly times.

Table 4.5 Explanation of typical places and transitions in the PNMs shown in Figs. 4.4 and 4.5

Place	Explanation
DAM1	Demand for Part A on MC _i
IM1	Input buffer of MC1 with parts ready to feed MC1
SM1A	Setup for MC1 to process Part A
M1RA	MC1 ready to process Part A
AM1	Part A loaded on MC1's table
PAM1	Part A is being unloaded by R1
PARM1	Part A is ready at the MC1
OM1	Output buffer of MC1 ready with parts A and B
AGM1	AGV at the output buffer of MC1
AM12	AGV traveling from MC1 to MC2
AM1	AGV at the input buffer of MC1
AMO3	AGV at the output buffer of MC3
A1ASM1	AGV1 traveling from AS to MC1
Transition	
1,6,14,19,28,33,39,44	Signal for machine setup
2,7,15,20,29,34,40,45	Robot finishes the loading operation
3,8,16,21,30,35,41,46	Completion of part processing
4,9,17,22,31,36,42,47	Robot finishes the unloading operation
5,10,18,23,32,37,43,48	Number of parts as specified in final assembly are ready in MC's output buffer
11,13,25,27,38,49	AGV starts from MC
12,24,26,50,51,52	AGV reaches its destination

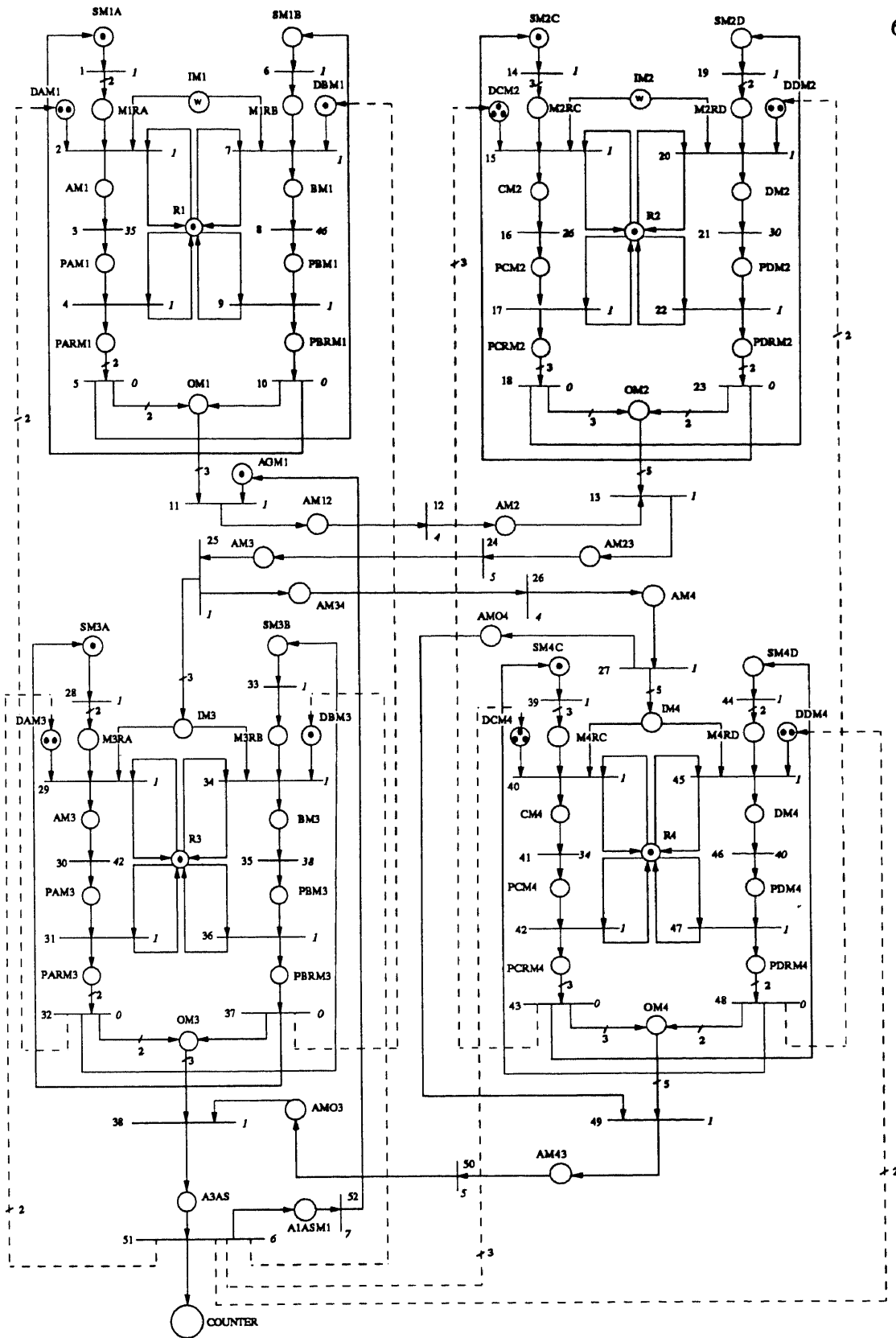


Figure 4.4 PNM of the FMS under pull paradigm

- NOTE: 1. Transition times are shown for FMS case 3. $\begin{matrix} k & r \\ | & \\ k & r \\ | & \\ r & \end{matrix}$ k is transition number 4. PLS for A,B,C & D : 2,1,3 & 2
 2. Initial marking is shown 5. N = 1, n = 1

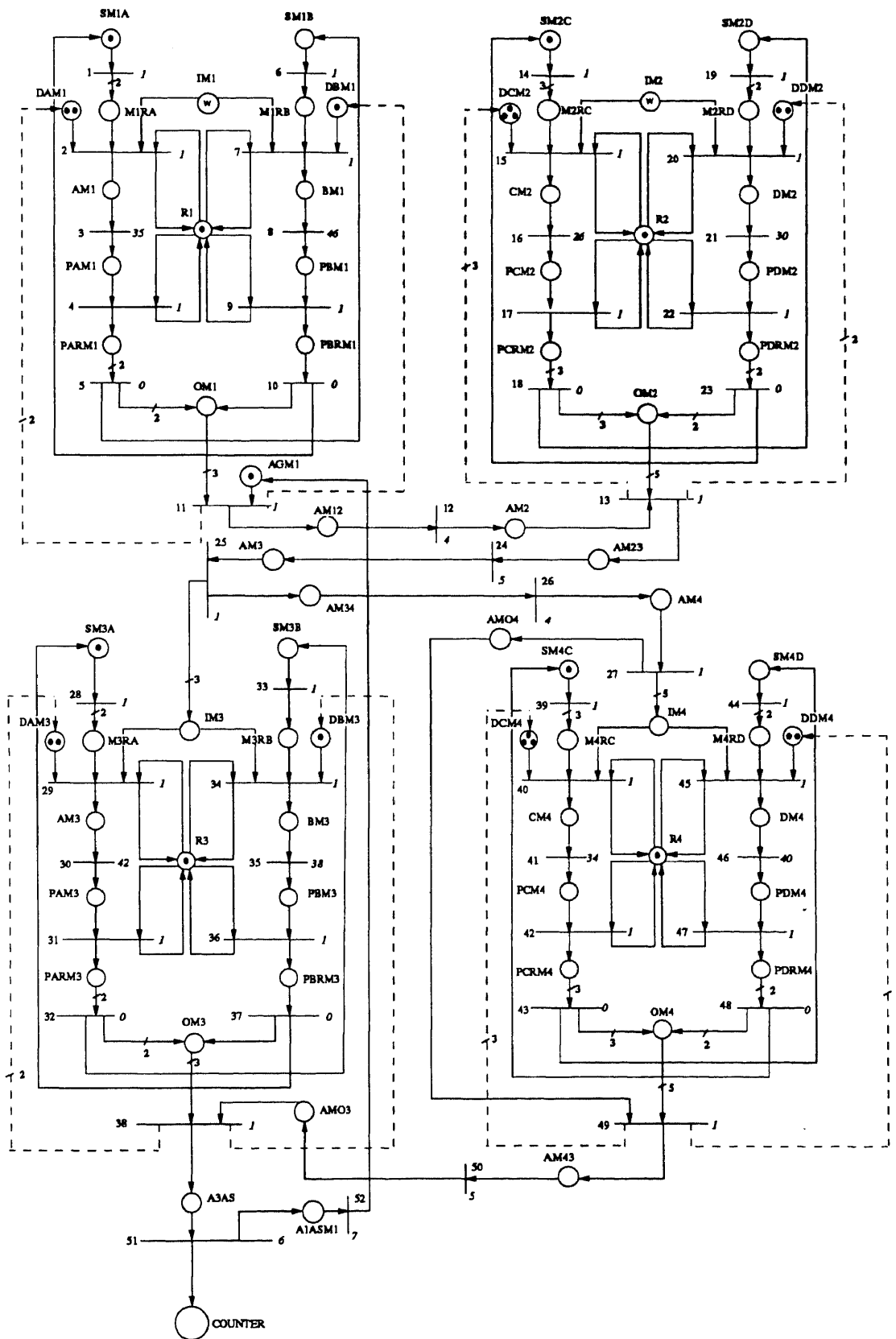


Figure 4.5 PNM of the FMS under push paradigm

NOTE: 1. Transition times are shown for FMS case 3.
 2. Initial marking is shown

$$k \begin{array}{|c} r \\ \hline k \\ \hline r \end{array}$$
 k is transition number 4. PLS for A,B,C & D : 2,1,3 & 2
 r is time units 5. N = 1, n = 1

4.3. Procedure for PN Modeling and Analysis and Simulation results

Figure 4.6 presents the procedure for PN modeling and analysis that is adopted to compare the performance of push and pull paradigms.

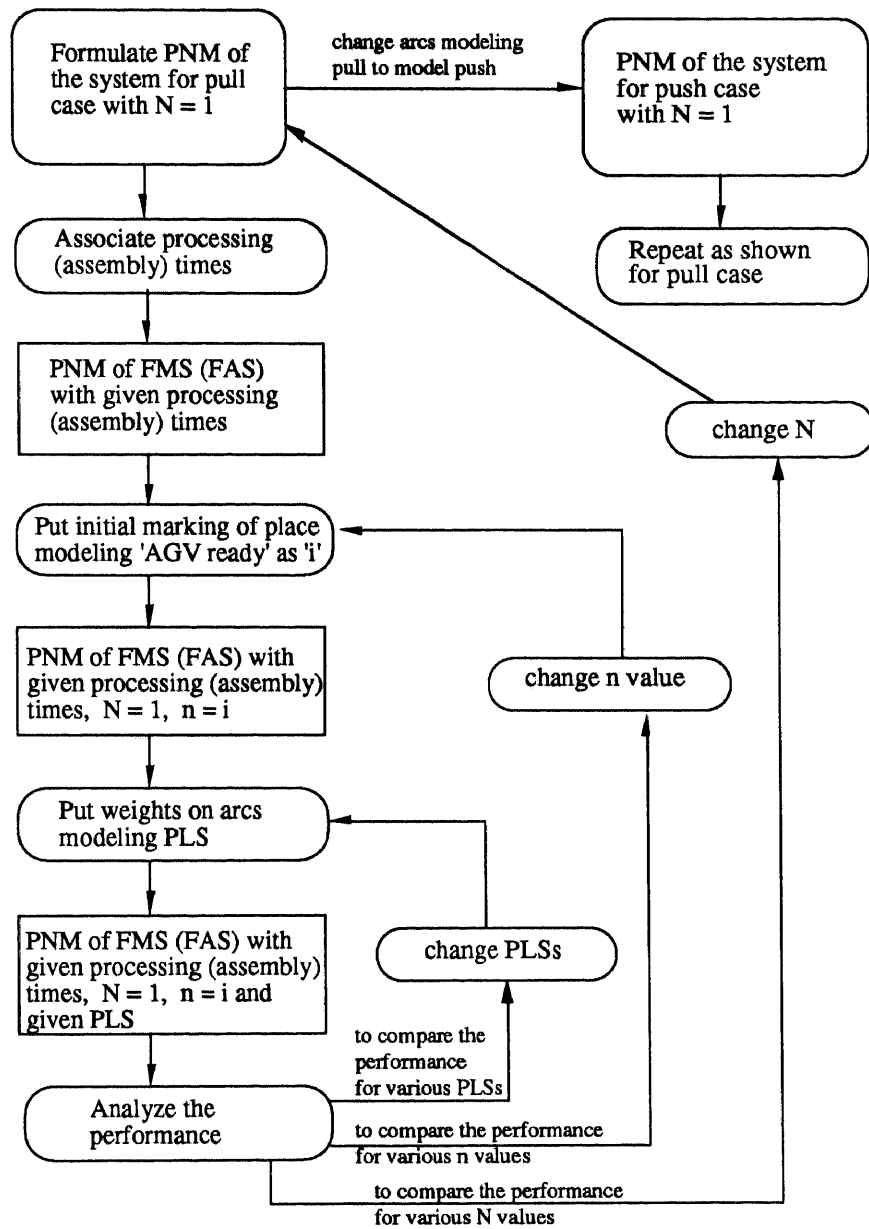


Figure 4.6 Procedure for PN modeling and analysis

From this figure it can be inferred that a PNM formulated for $N = 1$ can be slightly modified to study the performance of push and pull performance for both FMS and FAS cases corresponding to various values of N , n , and PLSs. This shows the reusability of the PNM obtained for a given value of N . Hence, PNs offer a powerful solution.

Simulation results

The software package developed in Venkatesh, *et al.* (1992) is used for the quantitative analysis of PNM. For each configuration of the system there exist twelve different combinations of the parameters, PLSs, N , and n . This is because there are three different values of N , two different values of n , and two different combinations of PLSs. The explanation for the various values of N is already discussed earlier (refer to Table 4.4). The results corresponding to two values of n only are presented because during simulation it is found that when n takes a value more than two, there is no improvement in system performance. The details of two combinations of PLSs are explained below.

There are two different combinations of PLSs for parts A, B, C, and D namely, 2, 1, 3, 2 and 1, 1, 1, and 1. In the first combination parts (subassemblies) A (SA), B (SB), C (SC), and D (SD) are produced in the lot sizes equal to their exact requirement as in the bill of materials of PR1 (PR2). In the second case, these are produced in unit lot sizes. In other words, the first combination corresponds to the loading sequence 1A, 1A, and 1B on MC1 and 1C, 1C, 1C, and 1D, 1D on MC2. The second combination corresponds to 1A and 1B on MC1 and 1C and 1D on MC2. It is clear that in the former case, the setup time required to produce one finished product is less compared to the latter. However, the work in process inventory in the former may be more compared to the latter. Further, PLSs affect the utilization of machines, robots, and AGVs.

The combination of the N , n , and PLSs, which results in the lowest minimum work-in-process inventory, the highest feasible utilization, and the highest output rate, is regarded as the optimal solution set for the system. In the following paragraphs, the results corresponding to four different configurations of the system are discussed.

4.3.1. FMS with the *pull* Paradigm

Table 4.6 shows the system performance when the system functions as an FMS with the *pull* paradigm. Consider PLSs of 2,1,3, and 2. In this case $N = 1$ and $n = 1$ is the

Table 4.6 System functioning as FMS with the pull paradigm

N	n	π Time Units	OR ($34/\pi$)	AMU (%)	AVU (%)	ARU (%)	BS1	BS2
PLS for A, B, C, and D: 2, 1, 3, and 2								
1	1	9898	0.00343	54.07	11.83	1.55	3	5
	2	9898	0.00343	54.07	5.92	1.55	3	5
2	1	9898	0.00343	54.07	5.95	1.55	3	5
	2	9898	0.00343	54.07	2.98	1.55	3	5
3	1	9708	0.00350	55.13	3.80	1.58	3	5
	2	9708	0.00350	55.13	1.90	1.58	3	5
PLS for A,B,C and D: 1, 1, 1, and 1								
1	1	5446	0.00624	51.15	21.50	1.43	3	5
	2	5446	0.00624	51.15	10.75	1.43	3	5
2	1	5446	0.00624	51.15	10.80	1.43	3	5
	2	5446	0.00624	51.15	5.40	1.43	3	5
3	1	5175	0.00657	53.89	11.12	1.49	3	5
	2	5175	0.00657	53.89	5.56	1.49	3	5

Legend:

N	Number of unique different transportation tasks
n	Number of AGVs assigned for each unique transportation task
π	Time required for production of 17 products of each PR1 and PR2
OR	Output rate (parts/time unit)
AMU	Average machine cell utilization
AVU	Average AGV utilization
ARU	Average robot utilization
BS1,BS2	Maximum input buffer sizes at MC3 and MC4 respectively

solution set. This is because with an increase in either N or n or both, output rate, average machine cell utilization (AMU), and average robot utilization (ARU) are slightly increased, but average vehicle utilization (AVU) is significantly decreased. Now, consider the PLSs of 1,1,1, and 1. In this case, due to the same reasons stated above, $N = 1$ and $n = 1$ is the solution set. After observing the system performance corresponding to the above two

solution sets, it can be stated that the solution set corresponding to PLSs 1, 1, 1, and 1 results in the solution set. This is because with a decrease in the PLSs from 2, 1, 3, and 2 to 1, 1, 1, and 1, output rate is improved by 82%; AVU is increased by 82%; AMU and ARU are not much affected. Hence, when the system functions with the *pull* paradigm, PLSs of 1,1,1, and 1 with $N = 1$ and $n = 1$ is the optimal solution set.

4.3.2. FAS with the *pull* Paradigm

Table 4.7 shows the system performance when the system functions as an FAS with the *pull* paradigm.

Table 4.7 System functioning as FAS with the pull paradigm (the same legend as Table 4.6's)

N	n	π Time Units	OR ($34/\pi$)	AAU (%)	AVU (%)	ARU (%)	BS1	BS2
PLS for A, B, C, and D: 2,1,3, and 2								
1	1	3717	0.00915	36.95	31.61	4.15	3	5
	2	3495	0.00973	39.40	16.81	4.40	3	5
2	1	3495	0.00973	39.29	16.88	4.38	3	5
	2	3495	0.00973	39.29	8.44	4.38	3	5
3	1	3305	0.01030	41.52	11.18	4.66	3	5
	2	3305	0.01030	41.52	5.59	4.66	3	5
PLS for A,B,C and D: 1, 1, 1, and 1								
1	1	2409	0.01410	28.75	48.61	3.19	3	5
	2	1854	0.01830	37.35	31.74	4.22	3	5
2	1	1854	0.01830	37.35	31.88	4.15	3	5
	2	1854	0.01830	37.35	15.94	4.15	3	5
3	1	1657	0.02050	41.80	34.72	4.65	3	5
	2	1657	0.02050	41.80	17.36	4.65	3	5

Here again, when PLSs are 2, 1, 3 and 2, $N = 1$ and $n = 1$ is a solution set due to the same reasons listed above. But in the case of PLSs - 1, 1, 1, and 1, when $N = 1$, an increase in n from 1 to 2 resulted in the following: output rate is increased by 30%; AAU and ARU are increased by 30% and 32%, respectively; and AVU is decreased by 34.71%.

With a further increase in either N or n or both, output rate, ARU, and average assembly cell utilization (AAU) are not much affected. Therefore, $N = 1$ and $n = 2$ is a solution set for PLS values of 1,1,1, and 1. Now, it can be concluded that with a decrease in PLSs from 2, 1, 3, and 2 to 1, 1, 1, and 1, output rate is improved by 100%; AAU, AVU, and ARU were slightly increased. Hence, for this case, PLSs of 1,1,1, and 1 with $N = 1$ and $n = 2$ is the optimal solution set.

4.3.3. FMS with the *push* paradigm

Table 4.8 shows the system performance of the FMS functioning with the *push* paradigm. In this case, the solution set when PLSs are 2,1,3, and 2 is $N = 1$ and $n = 2$. This is because an increase in n from 1 to 2 (without changing N), improves the output rate by 19%; AMU by 19%; AVU by 20%; and ARU by 21%. Beyond this, with an increase in either N or n or both, output rate is not affected; AVU is reduced significantly; ARU, AMU increased slightly and buffer sizes increased significantly.

Table 4.8 System functioning as FMS with the push paradigm (the same legend as Table 4.6's)

N	n	π Time Units	OR ($34/\pi$)	AMU (%)	AVU (%)	ARU (%)	BS1	BS2
PLS for A, B, C, and D: 2, 1, 3, and 2								
1	1	8566	0.00397	62.63	13.67	1.80	3	5
	2	7234	0.00470	74.56	16.37	2.18	3	5
2	1	7234	0.00470	85.11	9.65	2.49	36	60
	2	7234	0.00470	85.51	4.84	2.50	36	60
3	1	7229	0.00470	88.46	6.33	3.52	66	60
	2	7229	0.00470	88.46	3.16	2.73	66	60
PLS for A,B,C and D: 1, 1, 1, and 1								
1	1	4410	0.00771	63.46	26.55	1.75	3	5
	2	3485	0.00975	80.51	33.97	2.24	3	5
2	1	3337	0.01020	84.20	17.89	2.34	3	5
	2	3337	0.01020	84.20	8.95	2.34	3	5
3	1	3325	0.01020	91.73	12.46	2.60	3	32
	2	3325	0.01020	92.290	6.23	2.61	3	32

In the case of PLSs - 1,1,1, and 1, when $N = 1$, with an increase in n from 1 to 2, output rate is increased by 26%; AMU by 27%; AVU by 28%; and ARU by 28%. Again due to the same reasons stated above, $N = 1$ and $n = 2$ is the solution set. Based on the system performance corresponding to the above two solution sets, it can be concluded that with a decrease in PLSs from 2, 1, 3, and 2 to 1, 1, 1, and 1 output rate and AVU are improved by 107% and 107% respectively. Also, there are slight increases in AMU and ARU but buffer sizes do not change. Hence, when the system functions as an FMS with the *push* paradigm, PLSs of 1,1,1, and 1 with $N = 1$ and $n = 2$ is the optimal solution set.

4.3.4. FAS with the *push* paradigm

Table 4.9 shows the system performance of FAS functioning with the *push* paradigm. In this case, the solution set when PLSs are 2,1,3, and 2 is $N = 1$ and $n = 2$.

Table 4.9 System functioning as FAS with the push paradigm (the same legend as Table 4.6's)

N	n	π Time Units	OR ($34/\pi$)	AAU (%)	AVU (%)	ARU (%)	BS1	BS2
PLS for A, B, C, and D: 2, 1, 3, and 2								
1	1	3717	0.00915	37.00	31.50	4.66	3	5
	2	2385	0.01420	58.49	24.82	6.59	3	5
2	1	2385	0.01420	69.18	31.11	7.93	51	82
	2	2385	0.01420	69.46	15.60	7.96	51	82
3	1	2380	0.01420	75.63	21.94	11.65	122	82
	2	2380	0.01420	75.63	21.94	11.65	122	82
PLS for A,B,C and D: 1,1,1, and 1								
1	1	2409	0.01410	28.75	93.82	3.19	3	5
	2	1262	0.02690	55.74	48.61	6.22	3	5
2	1	1077	0.03150	72.45	62.91	8.01	20	17
	2	1077	0.03150	72.45	31.45	8.01	20	17
3	1	1065	0.03190	74.70	40.53	8.41	19	27
	2	1065	0.03190	74.70	20.27	8.41	19	27

This is because with an increase in n from 1 to 2 (without changing N), it results an increase in the output rate by 55%; an increase in AAU and ARU by 58% and 41% respectively; and a decrease in AVU by 21%. Beyond this, with an increase in either N or n or both, output rate is not affected. AVU shows decreasing trend; AAU and ARU are slightly increased; and buffer sizes are significantly increased. In the case of PLSs - 1,1,1, and 1, also when $N = 1$, with an increase in n from 1 to 2, output rate, AAU and ARU are increased by 91%, 94%, and 95% respectively; but AVU decreased by 48%. Again, due to the same reasons stated above, $N = 1$ and $n = 2$ is the solution set. Based on the system performance corresponding to the above two solution sets, it can be concluded that with a decrease in PLSs from 2, 1, 3, and 2 to 1, 1, 1, and 1, output rate and AVU are improved by 89% and 96% respectively; AAU and ARU are very slightly decreased and buffer sizes remains same. Hence, when the system functions as an FAS with the *push* paradigm, PLSs of 1,1,1, and 1; $N = 1$ and $n = 2$ is the optimal solution set.

4.3.5. Summary of Results

From Tables 4.6 and 4.7 when FMS and FAS function with the *pull* paradigm, the buffer sizes are minimum and constant irrespective of the values of PLS, N , and n . However, with the *push* paradigm from Tables 4.8 and 4.9, they increase with either PLS and/or N . For a given PLS combination with constant N , n does not affect the buffer sizes. This observation confirms to the general notion that higher lot sizes result in higher buffer sizes. When this system functions under the *pull* paradigm, the buffer sizes are same in both FMS and FAS. However, under the *push* paradigm when N is greater than one, the buffer sizes in case of FAS are larger compared to the FMS case because assembly times in FAS are less than processing times in FMS. Hence as the processing times vary, buffer sizes also vary in case of the *push* paradigm for higher values of N . The optimal solution sets for FMS and FAS functioning with *push* and *pull* paradigms are shown in Table 4.10.

Table 4.10 Solution sets for FMS and FAS (the same legend as Table 4.6's)
 $N = 1$ and $PLS = 1, 1, 1$, and 1 , Demand: 17 products,
 OR (production volume/time units) = $(17/\pi)$

Operation Paradigm	π	OR ($34/\pi$)	n	AMU or AAU	AVU	ARU	BS1	BS2
FMS case								
PUSH	3485	0.00975	2	80.51	33.97	2.24	3	5
PULL	5446	0.00624	1	51.15	21.50	1.42	3	5
FAS case								
PUSH	1262	0.02690	2	55.74	48.61	6.22	3	5
PULL	1854	0.01830	2	37.35	31.74	4.22	3	5

It is concluded that for both FMS and FAS, the optimal solution value of $N = 1$ is same for both *push* and *pull* paradigms. In the case of FAS, $n = 2$ is the best solution for both *push* and *pull* paradigms. But, in the case of FMS, the solution values of n are different for *push* and *pull* paradigms. In other words, the *push* paradigm in FMS requires two AGVs, but the *pull* paradigm requires only one AGV. Figure 4.7 shows the performance of FMS and FAS for solution sets listed in Table 4.10. In both cases, the *push* paradigm yields better performance.

This is because, in case of FMS (FAS), *push* paradigm results in an increase in output rate, AMU (AAU), AVU, and ARU by 56 (47)%; 57 (49)%; 58 (53)% and 58 (47)% respectively and buffer sizes remain same as shown in Table 4.10. Hence, for the particular system considered, the *push* paradigm helps achieve the goals of implementing JIT in flexible factory systems. Therefore, before adopting either *pull* or *push* as operational strategy for a particular system, it is important to evaluate thoroughly the system performance for both cases with an aim not only to reduce inventories but also to increase system utilization.

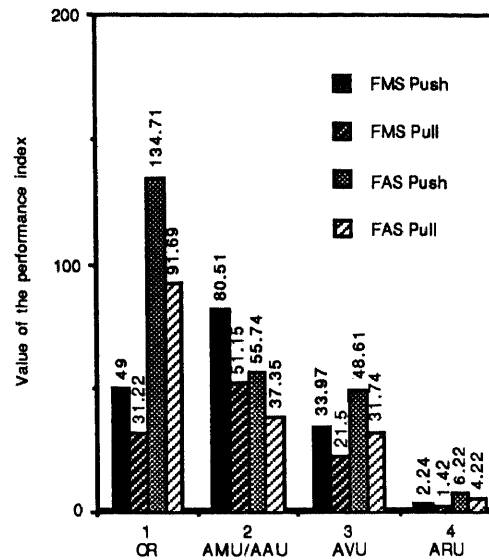


Figure 4.7 Performance of the system with different configurations

Legend: OR' OR * 5,000 (finished product/time unit)
 AMU Average machine cell utilization (%)
 AAU Average assembly cell utilization (%)
 ARU Average robot utilization

4.4. Summary

The performance of a particular FMS was evaluated using timed Petri nets by changing different operational parameters in the system operating under *push* and *pull* paradigms. The performance criteria are the buffer sizes, output rate, utilization of machines, AGVs, and robots. The configuration that results in the minimum buffer sizes, maximum system utilization and output rate is considered as the optimal solution. The manufacturing system considered is investigated as both FMS and FAS, by changing the production lot sizes, the number of unique transportation tasks (which decides moving lot sizes), and the number of AGVs for each transportation task. To achieve this, PNMs of the system are formulated and analyzed quantitatively through a PN driven simulation package. The analysis shows that in both cases of FMS and FAS, the *push* paradigm performed better than the *pull* one

for this system. This is because unlike the general notion that only *pull* paradigm results in minimum WIP, *push* may result not only in minimum WIP but also maximum system utilization and output rate for certain configurations and operation parameters. The results may change if some of the system parameters such as processing and/or assembly times change. From the simulation results it can be observed that the utilizations of system elements are generally low. This may be due to the parameter settings of machining and conveyance times. In other words, by changing these values, the system utilization can be increased. The same PNMs can be used by associating the new time values with the corresponding transitions modeling such activities. It is concluded that 1) before adopting either *push* or *pull* paradigm, the system should be evaluated with goals to reduce inventories and increase system utilization and output rate, and 2) PNs can be a very useful tool to perform such evaluation.

From the simulation results, many inferences are useful for the operations managers, system designers, manufacturing, industrial and software engineers. For example, the results show that average robot utilization is very low in all the cases studied in this chapter. This is because there is a dedicated robot for each work cell. To increase robot utilization, the possibility of sharing a common robot between two work cells should be studied further. However, scheduling of robot movements should be done to prevent impairing the performance of other system elements. This issue is another interesting topic for research in manufacturing systems (Venkatesh *et al* 1992). PNs can be used also to generate the supervisory control code of the systems as illustrated for manufacturing systems (Zhou *et al.* 1992, Srinivasan and Jafari 1991). The further enhancement of this work is to implement the discrete controller based on the control logic embedded in the PNMs as shown in the Chapter 9. Hybrid push-pull paradigms can be studied by changing the dotted output arc (in Fig. 4.4) from t_{32} to t_{29} . In this chapter, during the simulation of the system breakdowns are assumed not to occur. A class of PNs that are useful for detailed breakdown modeling and simulation are proposed next.

CHAPTER 5

AUGMENTED-TIMED PETRI NETS FOR MODELING BREAKDOWN HANDLING

5.1 Introduction

Flexible manufacturing and assembly systems consist of machines, robots, and automated guided vehicles, aiming to meet the dynamically changing needs of the market. Numerous asynchronous concurrent actions involved in these systems make their analysis difficult. Breakdowns of the system components further complicate the investigation of the issues related to their design, performance optimization, and control. Furthermore, detailed breakdown handling helps implement the real-time shop-floor controller as reported by Zhou and DiCesare (1989). Even though there are several types of PNs available for discrete control (Crockett *et al.* 1987, Murata *et al.* 1986, Stefano and Mirabella 1991, Valette *et al.* 1983) there is a need for extending PNs to model realistically the operations in factory floors. More specifically, there is a need to enhance the power of timed PNs to realistically model the breakdown situations in FMSs.

This chapter proposes a new class of modeling tools called Augmented-timed Petri nets (ATPNs) for modeling and analyzing manufacturing systems with breakdowns. These models aid designers in better understanding of concurrency, synchronization, and sequential relations involved in breakdown handling and in system simulation for performance analysis.

A flexible assembly system consisting of three robots with various breakdown rates is used to illustrate modeling, simulation and analysis with ATPNs. ATPN models for breakdown handling are presented and analyzed for estimating the system performance and designing the optimum number of assembly fixtures. The ATPN models can also be used for the real-time control of the system. In case of a breakdown many design and control issues have to be addressed. Considering the importance of breakdowns in production control, Gershwin and Berman (1981) presented the analysis of transfer lines consisting of

two unreliable machines with random failures. Groenevelt *et al.* (1992) investigated issues related to the estimation of economic lot size and safety stock levels for an unreliable manufacturing system with a constant failure rate. Glassey and Hong (1993) presented a model for the analysis of behavior of an unreliable n-stage transfer line with finite size buffers.

Most of the above researchers studied conventional manufacturing systems by estimating the economic safety stocks to handle breakdown situations. However, to implement Just-In-Time manufacturing and to increase the level of automation, there is a need to handle breakdowns by reducing the safety stock levels and implementing efficient breakdown handling procedures. Due to breakdowns, the optimal system operational parameters that are designed for the system without considering breakdowns have to be changed. Furthermore, these parameters may differ as the component breakdown rates vary. The issues to be specifically addressed when considering a system with breakdowns are detailed breakdown handling to ensure uninterrupted production, and estimation of the new optimum design parameters for different breakdown rates. Estimation of the new optimum design parameters for various breakdown rates aids system designers and production managers in achieving optimal system performance and is the focus of this chapter.

Although PNs are proved as a tool to solve a variety of problems relating to manufacturing systems, their full application to address design and analysis issues of FMS with breakdowns remains to be explored. PN modeling of breakdowns was considered in (Barad and Sipper 1988, Sheng and Black 1990). The former presented a PN model considering a machine breakdown while illustrating the flexibility of modeling a flexible manufacturing system (FMS) with PNs. The latter modeled a PN for machine breakdown while demonstrating the application of PN in a cellular manufacturing system. Performance of a transfer line was analyzed by considering the breakdown of machines using stochastic PNs (Al-Jaar and Desrochers 1990). Stochastic PNs is also used for the

performance analysis of a flexible assembly system considering various robot failure rates (Zhou and Leu 1991). None of the above presented the detailed breakdown handling procedures and performance optimization issues for various machine/robot breakdown rates. Furthermore, they considered only breakdowns that arrive before starting of an activity. However, in the real life situations breakdowns may arrive when an activity is in progress. In their models the transition time delays either follow either exponential distribution or are instantaneous. This restricts their accuracy for analysis of a realistic assembly system.

The goal of this chapter is to formulate graphical models that clearly capture the details of breakdown handling in FMSs to address issues related to their design, performance, and control. The objectives of this chapter are to:

1. introduce a new class of PNs called Augmented Timed Petri nets (ATPNs) aimed to model conveniently breakdown handling in manufacturing systems,
2. illustrate a methodology to formulate ATPN models for breakdown handling, and
3. model, simulate, and analyze an FAS using ATPNs for estimating the optimum number of assembly fixtures for various robot breakdown rates.

5.2. Augmented Timed Petri Nets

Before introducing Augmented-timed PNs, consider a timed PN modeling a machining operation as shown in Fig. 5.1. In Fig. 5.1 (a) before machining is started, machine, part, and tool availability are modeled by depositing tokens in places *machine_ready*, *part_ready*, and *tool_ready*. This triggers machining by absorbing the input tokens and firing the transition modeling the activity *machining*. During machining, machine and tool are busy and there is no processed part available, as modeled in Fig. 5.1 (b).

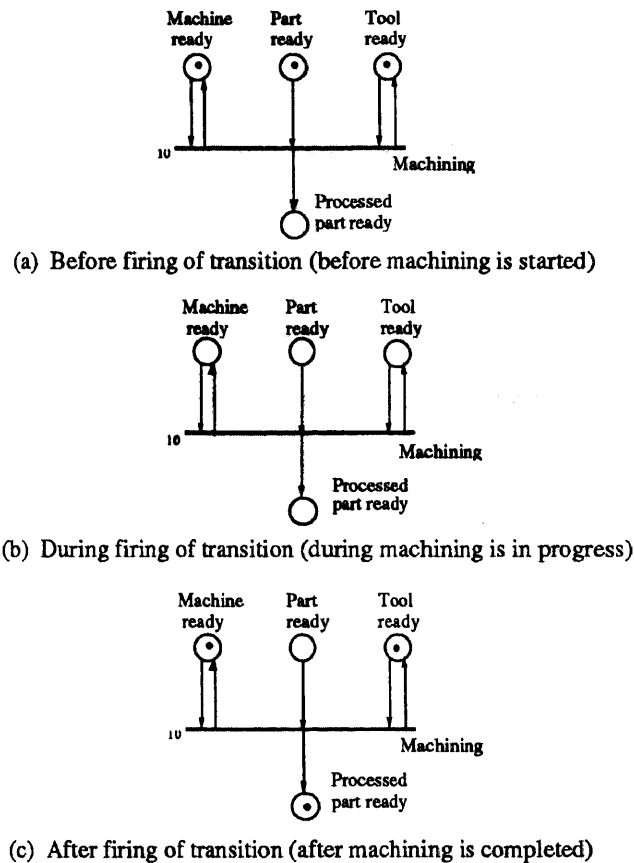


Figure 5.1 An example of a Timed Petri net model

Once the machining is completed, the part is processed and machine and tool are ready to process another part. This is modeled in Fig. 5.1 (c) by depositing each token in the places *processed_part_ready*, *machine_ready*, and *tool_ready*. Timed PN described above cannot easily model breakdowns in manufacturing systems. There exists an extension of PNs with inhibitor arcs which can model the breakdowns. However, they can only model breakdowns that arrive before the start of an activity. In real-life situations breakdowns may come at any time. Unlike the previous classes of PNs, ATPNs are proposed to model breakdowns which may occur before an activity starts and/or during an activity. Breakdowns may not only result due to the power or interface failure but also due to the subcomponent failure in a component. For example, a machine breakdown may result due to a fault in cutting fluid lubrication or tool handling system and a robot may

breakdown because of an error in its gripper. To model breakdowns, the following new constructs are added to TPNs, leading to ATPNs:

1. *Deactivation place*: This is used to model the message that is sent from cell controller to stop the operation of the breakdown component and start the standby component. Deactivation places are pictured by two concentric circles.
2. *Deactivation transitions*: Two kinds of deactivation transitions are introduced. The former models the activity - change-over from the breakdown component to standby one and vice versa. The latter models an activity that is being executed by the component and immediately stopped at the time of breakdown. These transitions are pictured as shaded ones.
3. *Input and output deactivation arcs*: These are used to model the control and information flow among component and cell controllers. The input arc models control flow from cell controller to the breakdowns component's controller (to stop its operation). The output arc models control flow from cell controller to the standby component's controller (to start the operations of breakdown component).
4. *Secondary arc*: This is used to model the conditions that exist before and after change-over from one component to another. It is pictured by a dotted arc.

Formally, an ATPN, Z'' is an eleven tuple, $Z'' = (P, T, I, O, m, D, P^d, T^d, I^d, O^d, I^s)$ where the last five tuples are the new tuples proposed in this chapter as an addition to TPN. They are defined as follows:

1. $P^d \subset P$ is a finite set of deactivation places. $p^d \in P^d$ connects the two transitions defined in T^d which is described next.
2. T^d is a finite set of deactivation transition pairs. Each pair consists of two transitions: deactivating, t_1' (generating the deactivation command), and deactivated, t_1'' (the transition that gets deactivated). Denote

$$T^d = \{(t_1', t_1''), (t_2', t_2''), \dots, (t_k', t_k'')\},$$

$$T^{d'} = \{t_1', t_2', \dots, t_k'\}, \text{ and}$$

$T^{d''} : \{t_1'', t_2'', \dots, t_k''\}$, where

$T^{d'}, T^{d''} \subset T^d, t_i' \neq t_i'', 1 \leq i \leq k, \text{ and } k=|P^d|,$

3. $I^d: P^d \times T^{d''} \rightarrow \{0,1\}$, input function defining a set of deactivation arcs from P^d to $T^{d''}$,
4. $O^d: P^d \times T^{d'} \rightarrow \{0,1\}$, output function defining a set of deactivation arcs from $T^{d'}$ to P^d ,
5. $I^S: P \times T \rightarrow \{0,1\}$, secondary input function defining a set of secondary arcs from P to T.

When a system is working normally without breakdowns, the initial marking of $p^d \in P^d$ is always zero. The firing rule of t_i' is same as a normal transition while that of t_i'' is different. The firing rule of t_i'' is same as a normal transition till the p_i^d contains a token. As soon as p_i^d contains a token, the firing of t_i'' is stopped (the activity modeled by t_i'' is stopped). When t_i'' is stopped because of p_i^d , it does not deposit the tokens in its output places in P. In other words, the tokens that t_i'' has taken from its input places for its normal firing are absorbed by t_i'' . When a place is connected to a transition by I^S , the enabling rule of the transition is same. However firing rule is different from normal firing rule in updating the new marking. That is, if place p_i is connected to t_j by I^S , firing t_j does not take any token from p_i . In other words, during execution of t_j , p_i contains a token. Summarizing the above concepts, the firing rules of ATPN are given as follows:

1. A transition $t \in T - T^{d''}$ is *enabled* if and only if $m(p) \geq I(p, t)$ and $m(p) \geq I^S(p, t), \forall p \in P$.

A transition $t \in T^{d''}$ is *enabled* if and only if $m(p) \geq I(p, t)$ and $m(p) \geq I^S(p, t), \forall p \in P$, and $\forall p^d \in P^d$.

2. Transition $t \in T^{d''}$ immediately terminates its firing, if $\exists p^d \in P^d, I^d(p^d, t) = 1$ and $m(p^d) = 1$.
3. Enabled in a marking m , t *fires* and results in a new marking m' following the rule:
 $m'(p) = m(p) + O(p, t) - I(p, t) \forall p \in P$ and if during the firing process $\exists p^d \in P^d \ni$

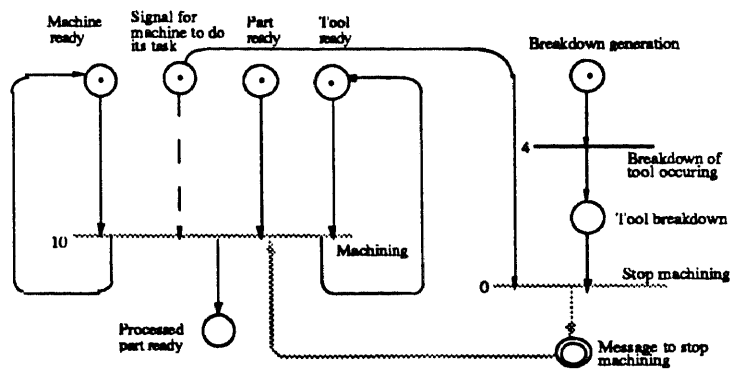
$I^d(p^d, t) = 1$ and p^d is not marked.

If $t \in T^d$, $m'(p) = m(p) - I(p, t) + O(p^d, t) - I^d(p^d, t) \forall p \in P$, and

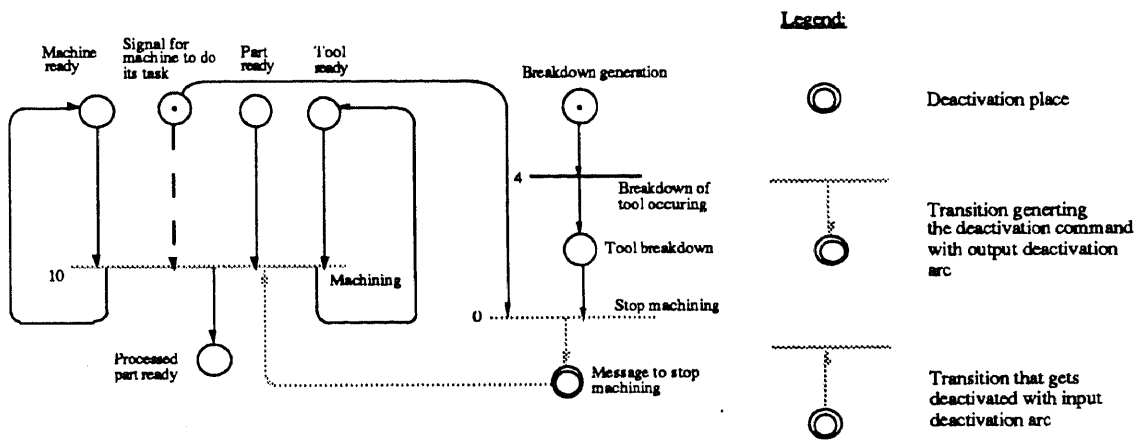
if during the firing process $\exists p^d \in P^d \ni I^d(p^d, t) = 1$ and p^d is marked.

The instantaneous description (ID) of ATPN is the same as that of TPN. Figs. 5.2 shows an example of ATPN. Fig. 5.2 (a) models machining before breakdown. The breakdown generation is modeled by the place *breakdown_generation*. Assume that the breakdown occurs four minutes after machining is started. This is modeled by associating 4 minutes to transition *breakdown_of_tool_occurring*. Fig. 5.2 (b) models the system when machining is in progress. Since the arc connecting place *signal_for_machine_to_do_its_task* and transition *machining* is secondary arc, there is token present in this place during machining. Four minutes after the machining started, the tool breakdown arrives and hence machining should be stopped. This is modeled in Fig. 5.2 (c) by removing a token from place *breakdown_generation* and depositing in the place *tool_breakdown*. The actions involved to stop machining is represented transition *stop_machining* which removes tokens from places *signal_for_machine_to_do_its_task* and *tool_breakdown* and deposits a token in *message_to_stop_machining*. Once this place contains a token, it sends a message to machine controller to immediately stop machining.

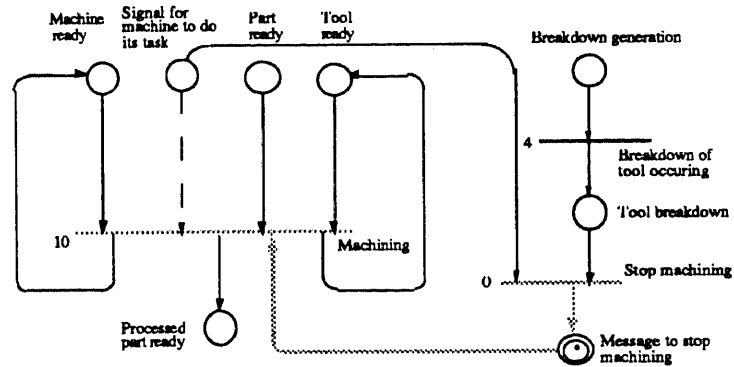
In the above example, breakdown time (four minutes) is assumed. However, breakdowns may occur at random in shop-floor. To consider the random occurrence of breakdowns in the analysis, the breakdowns times should be generated by assuming proper probability distributions. Also, strategies to minimize the down time during the period of breakdowns should be modeled during the analysis. These issues and other related breakdown handling issues are elaborately addressed and modeled in the PNM that are presented in the subsequent sections. For the analysis of ATPN model, simply referred to as Petri net model (PNM), a software package was developed using the principles of ATPNs and is presented in Appendix A. Note that RTPN is a special case of ATPN.



(a) Before machining is started



(b) Before breakdown (machining is in progress)



(c) After breakdown (machining is stopped)

Figure 5.2 Example of an ATPN model

The package presented in Appendix A gives the following information with respect to real time:

- Marking of each place in the PNM
- Active time of each transition
- Remaining time of each transition
- Transitions which are enabled
- Conflicts between transitions (a conflict results when a single resource is required to serve more than one customer simultaneously).

5.3. Application Illustration: A Flexible Assembly System

An FAS is investigated to show the application of ATPNs. The system can be used for assembling a variety of products as shown in Fig. 5.3.

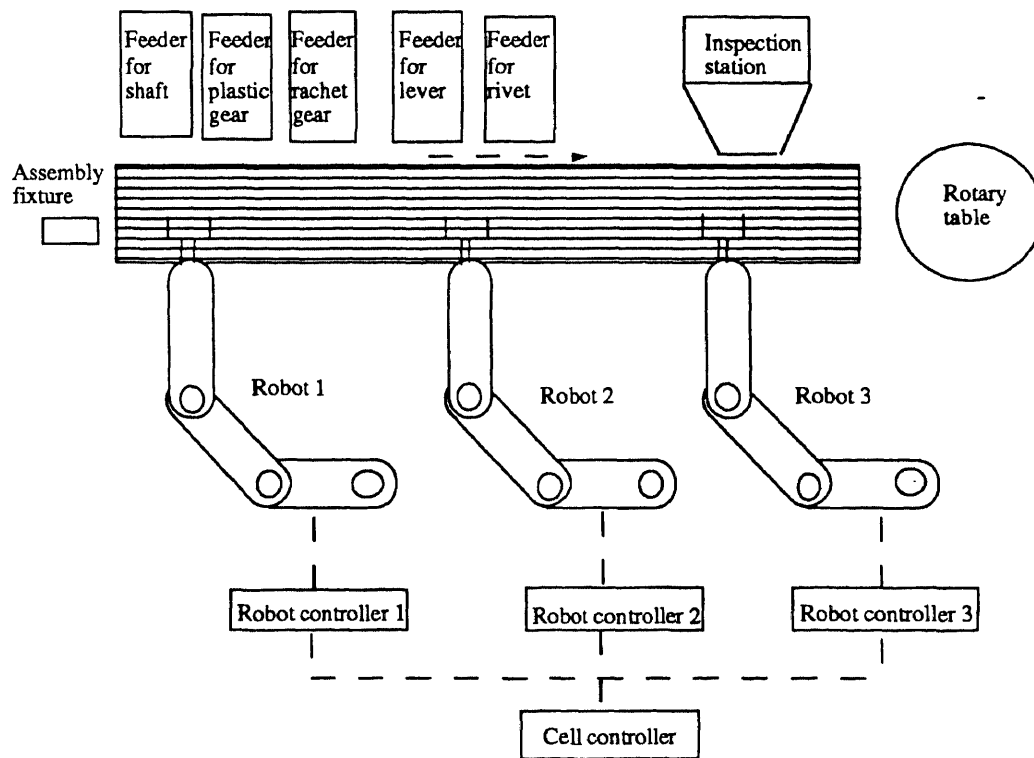


Figure 5.3 Layout of the flexible assembly system

In order to focus the objective, only one product, plastic ratchet assembly is considered. The system consists of 3 robots and an inspection station to do assembly and inspection and is controlled by a cell controller. Each robot is controlled by its own controller. The functions of the cell controller are to give signals to robots to do their tasks and also signals to stop their functioning at the time of breakdown and change-over.

The system described here is similar to industrial systems as Westinghouse robot assembly systems (Nof 1985). The assembly operation is split into tasks performed by three robots as follows:

Robot 1 (R1): Picks up and places a shaft in the assembly fixture; picks up and presses a plastic gear on the shaft; and picks up and presses a ratchet gear on the shaft.

Robot 2 (R2): Picks up and transfers the subassembly from position 1 to position 2. Picks up and keeps a lever in assembly fixture, and inserts a rivet in lever and rivets lever to the gear.

Robot 3 (R3): Transfers assembly to rotary table and inspects position and operations of lever. After R3's operation, assembly fixture returns to R1 to start the assembly of a new product. Assembly time for R1, R2, and R3 are thirteen, eighteen, and ten minutes respectively.

Various robot breakdown rates are considered to analyze the FAS. The system is evaluated for four values of mean time between failure (MTBF). The ranges of MTBFs are chosen randomly. The values of MTBFs are obtained by assuming that they follow uniform distribution within the given range.

These ranges and the values of MTBFs for R1, R2, and R3 for different cases are shown in Table 5.1. Table 5.2 shows the exact time and breakdown sequence of robots for four different cases with different breakdown rates.

Table 5.1 MTBFs of Robots in the FAS (minutes)

Breakdown number	Robot 1 range: 1000 to 10000	Robot 2 range: 4000 to 8000	Robot 3 range: 6000 to 14000
1	2,900	4,860	9,080
2	3,350	4,900	11,300
3	6,860	6,020	12,580
4	8,390	6,380	13,780

Table 5.2 Time and breakdown sequence of robots for various robot breakdown rates

Case number	Time (minutes) and breakdown sequence of robots		
1	R1 at 2,900	R2 at 4,860	R3 at 9,080
2	R1 at 3,350	R2 at 4,900	R3 at 11,300
3	R2 at 6,020	R1 at 6,860	R3 at 12,580
4	R2 at 6,380	R1 at 8,390	R3 at 13,780

In each case, given the MTBF, the actual breakdown time is assumed to follow a truncated normal distribution to avoid negative values for time. Based on our experience, data with coefficient of variation below 20% are usually reliable and hence, we have assumed the standard deviation in normal distribution to be 10% of the mean.

The following assumptions are made in this model of the FAS to focus on the objectives of the chapter. All the parts required for assembling the product are always available in automatic part feeders and there always exist demand for the finished product. There is a standby robot for each robot in the system that will come on-line when the corresponding robot fails. With this planning approach, unexpected manual intervention and subsequent productivity loss can be precluded. Moreover, it helps for the smooth control of system during breakdown situations. Maintaining a standby robot for each robot may not be economically feasible. However, to demonstrate the application of ATPNs in the case of breakdown handling a standby robot is assumed for each robot. Change-over time is the time required to transfer the programs concerned with the breakdown robot to the standby robot (to carry subsequent assembly operations), and to remove the unfinished part from the assembly area of the breakdown robot. It is same for all robots and is

assumed 20 minutes. The repair time for all robots is assumed to be the same and equal to 100 minutes. These time durations can be changed to random variables depending on the system under consideration.

5.3.1. ATPN Modeling of the System

Breakdown handling involves many concurrent actions like passing information to a standby robot or an operator, scheduling the unfinished part, repairing the breakdown robot, etc. Fig. 5.4 shows an ATPN model (PNM) for breakdown handling of R3.

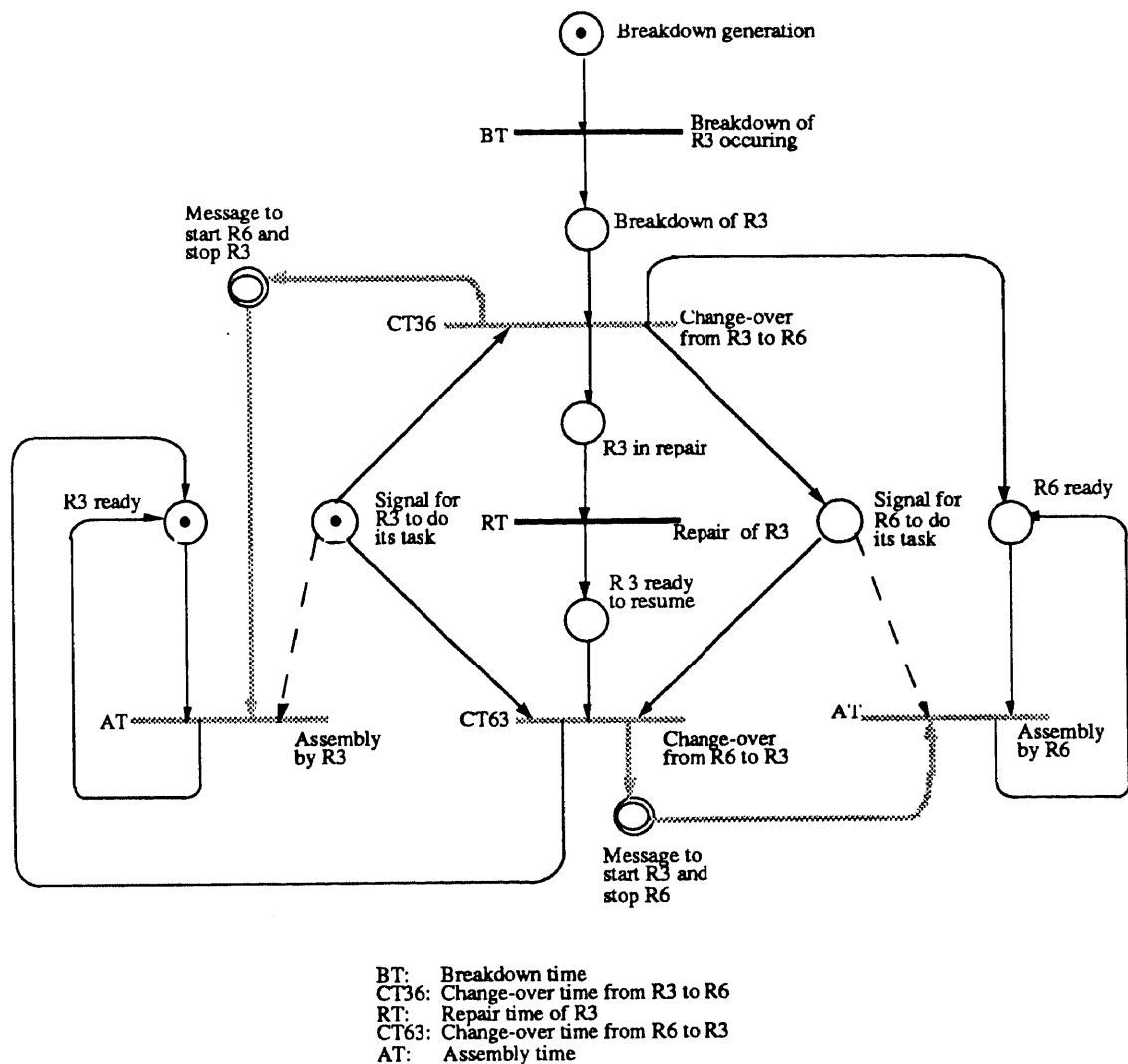


Figure 5.4 ATPN model for breakdown handling of Robot 3

The modeling methodology of ATPN is explained and the activities modeled and controlled by the PNM are chronologically listed here. The initial state of the FAS is modeled by the initial marking shown in Fig. 5.4. The time durations for the activities in the system are modeled by associating times to the transitions modeling corresponding activities. These are shown on the left hand side of each transition. At the start of the system, R3 is ready to do its task (modeled by depositing a token in place *R3_ready*) and the cell controller gives the signal to R3 to do its task (modeled by depositing a token in place *signal_for_R3_to_do_its_task*). R3 functions normally till a breakdown occurs. Breakdown occurs after a random time called breakdown time (BT, obtained from Table 5.2) which is associated to the transition *breakdown_of_R3_occurring*. When it occurs, a token will be deposited in place *breakdown_of_R3*. Then, after a change-over time from R3 to R6 (standby for R3) the following concurrent operations are executed by the cell controller:

1. Sending a signal to the R3's controller to stop the operation and R6's to start. This is modeled by depositing a token in place *message_tostart_R6_and_stop_R3* and removing the token from the place *signal_for_R3_to_do_its_task*. As soon as place *message_to_stop_R3* gets token it stops the functioning of R3 by deactivating the transition *assembly_by_R3*.
2. Activating the working of R6 by depositing a token in place *R6_ready*. The change-over between R3 and R6 is modeled by removing a token from place *signal_for_R3_to_do_its_task* and depositing it in place *signal_for_R6_to_do_its_task*.
3. Sending a message to the controller at the higher level to repair R3. This is modeled by depositing a token in place *R3_in_repair*.

During the repair of R3, R6 performs assembly operations. After certain time, R3 is repaired, modeled by firing the transition *repair_of_R3*. This resumes R3's operation, modeled by place *R3_ready_to_resume*. Now, the cell controller has to send signals for

change-over from R6 to R3. Its functions for the change-over from R6 to R3 are similar to previous change-over and hence the modeling is also similar. The various breakdown rates can be modeled by associating various values of time duration to transition *breakdown_of_R3_occurring*.

The above ATPN model for breakdown handling is exactly the same when a skilled operator is used to replace the functions of a standby robot. Furthermore, the same ATPN modeling approach can be extended for the breakdown handling of other robots, machines, AGVs, and cell controllers in a manufacturing system or other production interruption.

ATPN model for designing the optimum number of assembly fixtures

ATPN models can be used to address various design issues. As an example, this chapter determines the optimal number of assembly fixtures for various robot breakdown rates. To this end, FAS has to be modeled using ATPNs considering the breakdowns of all robots. Then, the obtained model has to be quantitatively analyzed to determine the effect of various MTBFs on the system performance. The ATPN model of FAS is shown in Fig. 5.5 and is obtained by duplicating the model shown in Fig. 5.4 for R 1, and R 2. The interpretation of places and transitions used in the PNM are listed in Table 5.3.

Table 5.3 Interpretation of places and transitions used in Fig. 5.5

Place	Interpretation
AF	Assembly fixture(AF) before robot 1 ready, where n is number of AFs
BGR _i	Breakdown generation for robot i ($i = 1$ to 3)
R _i B	Robot i breakdown
R _i	Robot i ready
MSR _{ji}	Message to start robot j and stop robot i ($j = 4$ to 6)
SR _i	Signal for robot i to do its task
R _i IR	Robot i in repair
R _i R	Robot i ready to resume
AF _k	Assembly fixture ready after robot k 's operation ($k = 1$ to 2)
Counter	Counter for noting production output
Transition	
BR _i A	Breakdown of robot i occurred
CR _{ij}	Change-over from robot i to robot j
RR _i	Repair of robot i
ASR _i	Assembly by robot l ($l = 1$ to 6)

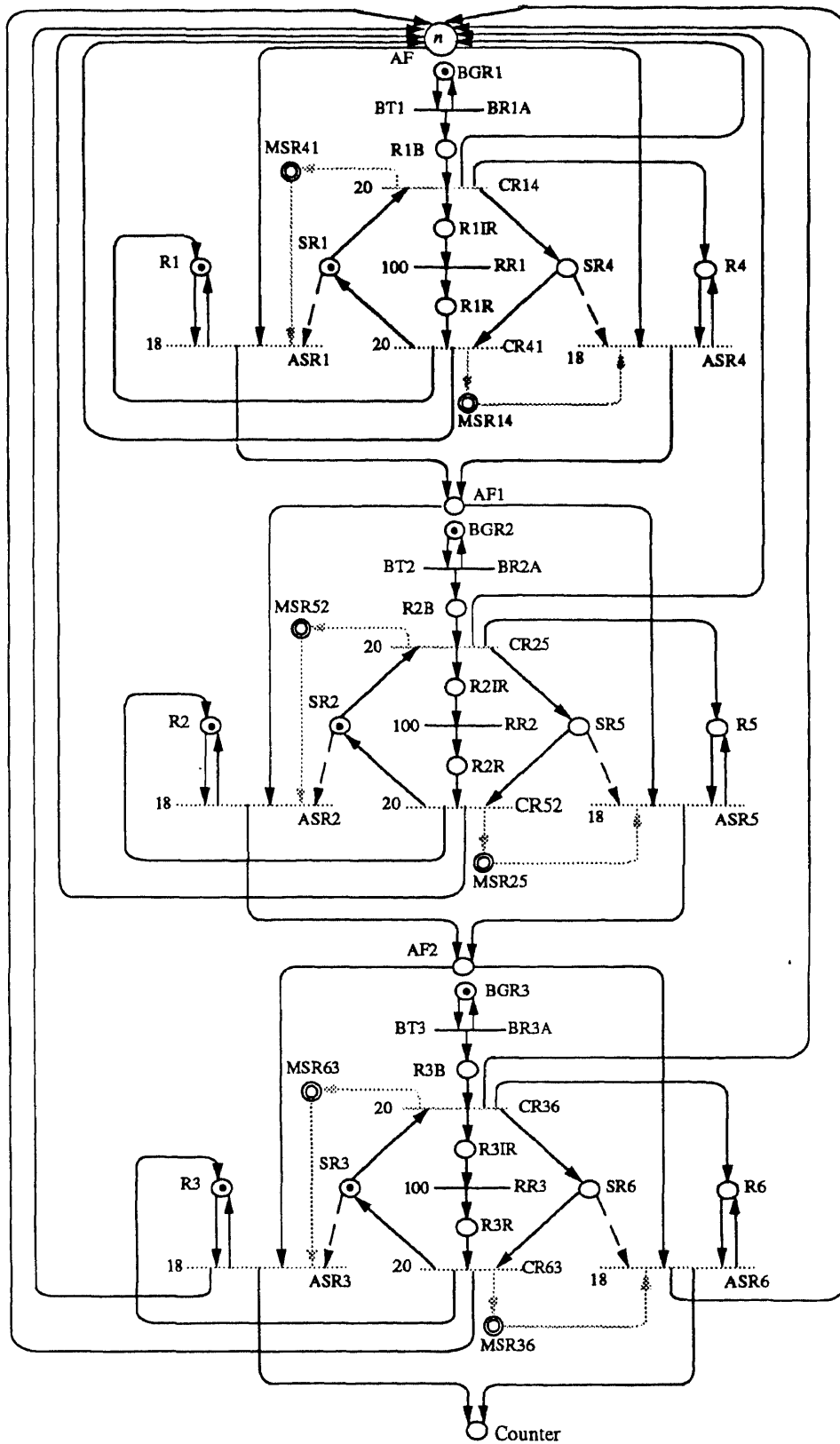


Figure 5.5 ATPN model of the assembly system with breakdown handling
 n = Number of assembly fixtures

In case of a breakdown the assembly fixture is removed from the FAS, the unfinished part is removed from the assembly fixture, and the assembly process starts again from R1. This is modeled by output arcs from six transitions that are modeling change-overs to place *AF (assembly_fixture_ready_before_R1)*. Once, a finished product is transferred by R3 to rotary table, the assembly fixture is to be send back to R1. This is modeled by output arcs from two transitions, one from R3 and another from R6 to place *AF (assembly_fixture_ready_before_R1)*.

5.3.2. Simulation and Analysis of the ATPN Model

The system is simulated for 20,000 time units. The execution of ATPN for one set of parameters takes approximately two minutes CPU time on a VAX system. Table 5.4 shows the performance of the system with and without breakdown of robots for different values of MTBFs and the number of assembly fixtures. During simulation, number of assembly fixtures is increased till the increase in production output is less than 1%. Fig. 5.6 and 5.7 shows the effect of MTBF on production volume and utilization.

Table 5.4 Performance of the assembly system with and without breakdown of robots

No. of Assembly Fixtures	Production output				
	No Robot Breakdown	Average Robot Utilization (%) Breakdown of Robots			
		Case 1	Case 2	Case 3	Case 4
1	1333 33.33	1302 16.33	1308 16.37	13180 16.50	1318 16.50
2	2499 62.49	1562 19.57	1571 19.67	1645 20.60	1664 20.83
3	2499 62.50	1873 23.45	1887 23.62	2353 29.44	2077 26.83
4	2499 62.50	2135 23.45	2147 26.87	2478 31.08	2417 30.25
5	2499 62.50	2425 30.36	2439 30.53	2484 31.08	2479 31.03
6	2499 62.50	2469 30.92	2473 30.10	2484 31.08	2483 31.08
7	2499 62.50	2476 31.13	2476 31.10		
8	2499 62.50	2477 31.03			

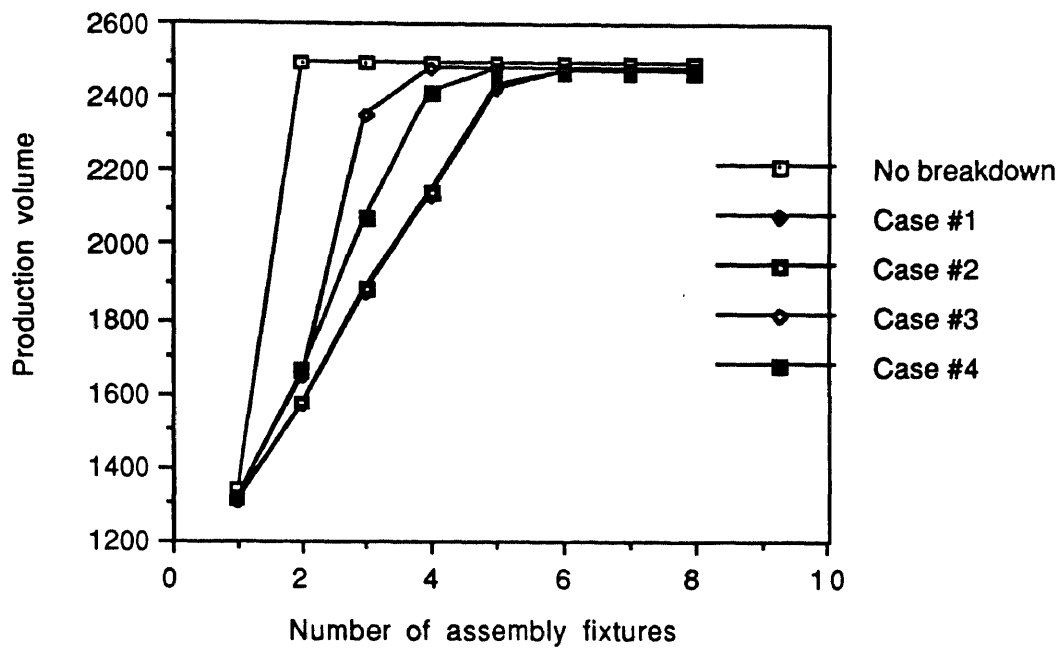


Figure 5.6 Effect of MTBF on production volume

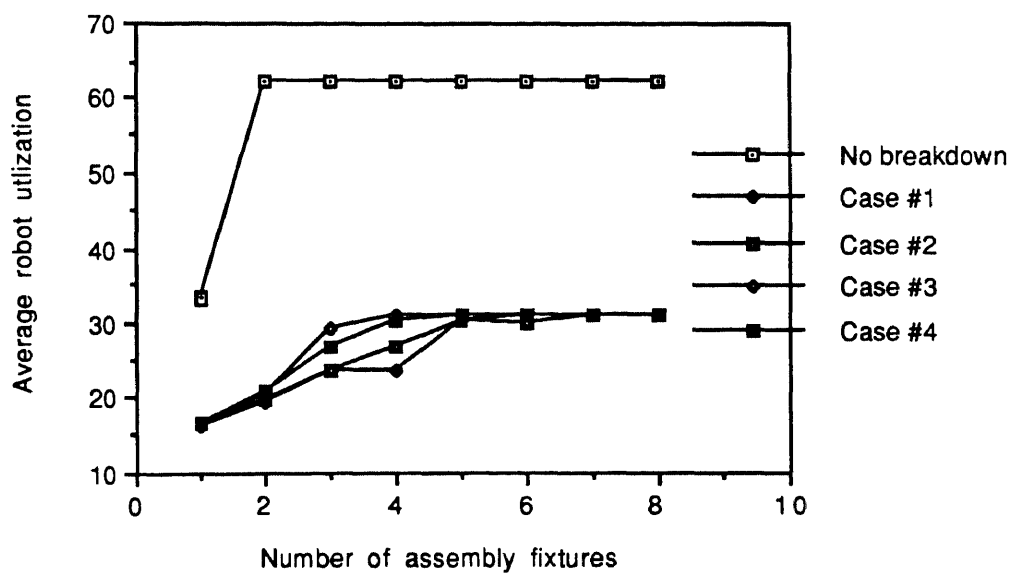


Figure 5.7 Effect of MTBF on average robot utilization

From the simulation results the following conclusions are drawn. When there is no breakdown in the system, there is a significant increase in production output when the number of assembly fixtures is changed from one to two. However, after it increases to more than two, there is no increase either in production output or average robot utilization due to the deterministic nature of the FAS. Hence, the optimum number of assembly fixtures required without any breakdown in the FAS is two. Figs. 5.6 and 5.7 also indicate the sensitivity of the production output and average robot utilization for different cases when there are robot breakdowns.

In the presence of breakdowns, to achieve the maximum production rate more assembly fixtures are required than that without breakdowns. The number of assembly fixtures does not increase linearly with the increasing value of MTBF. This is the conclusion drawn by observing the values of optimum assembly fixtures for various cases of MTBFs as shown in Table 5.

Table 5.5 Optimum number of assembly fixtures required for various robot breakdown rates

Case number	Time and breakdown sequence of robots			n_{opt}
1	R1 at 2,900	R2 at 4,860	R3 at 9,080	6
2	R1 at 3,350	R2 at 4,900	R3 at 11,300	6
3	R2 at 6,020	R1 at 6,860	R3 at 12,580	4
4	R2 at 6,380	R1 at 8,390	R3 at 13,780	5

n_{opt} : Optimum number of assembly fixtures

From the results it can be inferred that the system performance depends not only on the MTBF but also on the exact breakdown sequence of robots. Even though the value of MTBF increases from Case #1 to case #4, the number of assembly fixtures decreased from Case #1 as well as Case #2 to Case #3 and increased from Case #3 to Case #4.

5.4. Summary

A new class of modeling tools called Augmented Timed Petri nets (ATPNs) are introduced to model breakdown handling in manufacturing systems. ATPNs can model their operations in detail considering the breakdowns of various components. The methodology for formulating the ATPN models is illustrated by considering a flexible assembly system. Also, the application of ATPNs for optimization and design is shown by investigating the optimum number of assembly fixtures for the system under various robot breakdown rates. The methodology proposed in this chapter can be extended to deal with breakdowns of several machines, AGVs, and cell controllers and other production interruptions. ATPNs provide rapid, flexible, and realistic modeling.

ATPN models can be extended for the real time control. In such cases, the transition modeling the occurrence of breakdown is not associated with random breakdown times. Instead, the sensors/limit switches that recognize the breakdown in the system are modeled as input places for this transition. When there is a breakdown in the system, these places get tokens and thus immediately fire the transition, modeling the occurrence of breakdown. Fig. 5.8 shows an ATPN model intended for real-time control.

When the system contains several components, the size of ATPN models may grow. In such cases, *colored PNs* can be combined with the principles of ATPNs to formulate concise graphical models. This research can be extended to study some important issues such as robot scheduling during breakdowns when only a single robot exists as a standby to all three robots, system performance when several product varieties are produced simultaneously in the system with random routing of parts, and cost consideration for standby robots and breakdown handling. The effect of random distributions of repair and change-over times on the system performance can also be investigated by associating various time values to the transitions modeling repair and change-over activities.

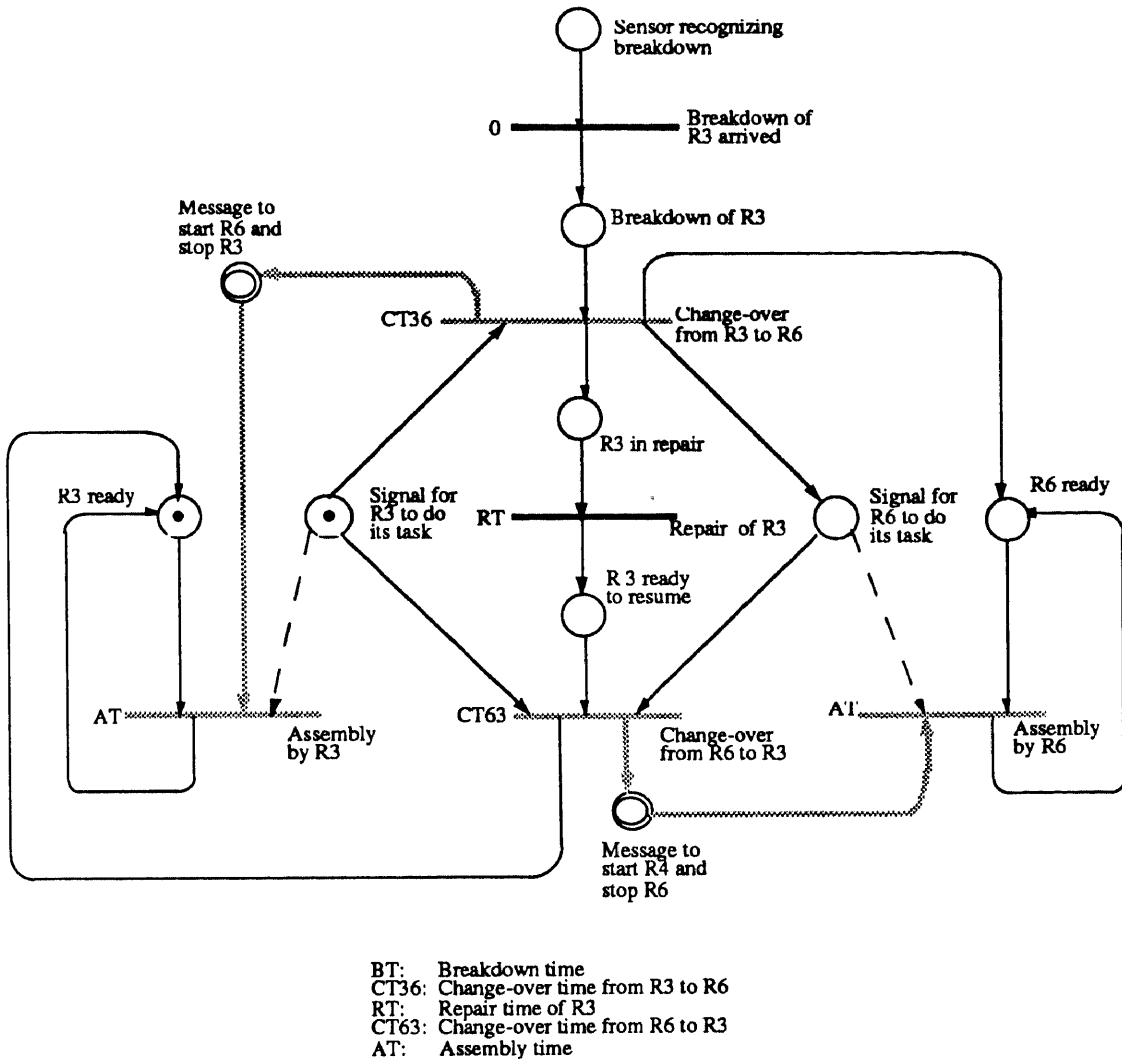


Figure 5.8 ATPN model for real-time control

CHAPTER 6

REAL-TIME PETRI NETS FOR CONTROL AND SIMULATION

6.1. Introduction

Several types of Petri nets (PNs) are available for supervisory control of FMSs. PNs have been augmented and implemented in a variety of ways to achieve real-time control (Chocron and Cerny 1980, Valette *et al.* 1983, Murata *et al.* 1986, Crockett *et al.* 1987, Boucher *et al.* 1989, Stefano and Mirabella 1991). The classification of various PN-based control schemes is described later in this chapter. New class of PNs for discrete-event control that closely resemble ordinary PNs is of the paramount importance for the development of control software because ordinary PNs are relatively simple and easier to understand.

Due to the close resemblance of ordinary PNs and ladder logic diagrams (LLDs), ordinary PNs are useful for the design recovery of ladder logic diagrams. In other words, LLDs can be converted into ordinary PNs. However, the design recovery of LLDs using earlier classes of PNs (Chocron and Cerny 1980, Valette *et al.* 1983, Murata *et al.* 1986, Crockett *et al.* 1987, Boucher *et al.* 1989, Stefano and Mirabella 1991) is relatively difficult as they do not closely resemble LLDs. The importance of design recovery of LLDs is emphasized in (Falcione and Krogh 1993). Motivated by these facts and based on the research in PN control literature, a new class of PNs called Real-time PNs (RTPNs) is proposed for sequence controller design.

6.2. Real-time Petri Nets

RTPNs can be obtained by associating timing, and I/O sensory information to the untimed PNs. It is an eight tuple and defined as: $RTPN=(P, T, I, O, m, D, X, Y)$ where:

1. P is a finite set of places;
2. T is a finite set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$;

3. $I: P \times T \rightarrow N$, is an input function that defines the set of directed arcs from P to T where $N = \{0, 1, 2, \dots\}$;
4. $O: P \times T \rightarrow N$, is an output function that defines the set of directed arcs from T to P ;
5. $m: P \rightarrow N$, is a marking whose i^{th} component represents the number of tokens in the i^{th} place. An initial marking is denoted by m_0 ;
6. $D: T \rightarrow R^+$, is a firing time function where R^+ is the set of non-negative real numbers;
7. $X: P \rightarrow \{-, 0, 1, 2, \dots, K\}$ and $X(p_i) \neq X(p_j)$, $i \neq j$, is an input signal function, where K is the maximum number of input signal channels, and "-" is the dummy attribute indicating no assigned channel to the place.
8. $Y: T \rightarrow L$, is an output signal function, where L is a set of integers.

In a RTPN, the first five tuples represent the untimed PN and the last three are extensions added to it and explained below:

1. Timing vector (D) is intended to associate time delays to transitions modeling the activities in the system,
2. Input signal vector (X) reads the state of the input signals from digital input interface. X associates attributes to every place. $X_i = X(p_i)$ and is an attribute associated with place p_i , representing the input channel number associated with p_i . For example, if p_i models a limit switch, the RTPN reads the status of that switch from the digital input interface through the channel number represented by X_i . The initial marking, $m_0(p_i)$ is considered as the first attribute of p_i and X_i is the second one. The contents of any input channel X_i are either 0 or 1.
3. Output signal vector (Y) is intended to send output signals through digital output interface. Y associates attributes to every transition. $Y_i = Y(t_i)$ and is the attribute associated to transition t_i which represents the number that is to be sent to the digital output interface. For example, t_i may model the activity "send signal to actuate

solenoid A," or "execute a procedure to control a robot." Each solenoid is activated by writing a specific number on to the digital output interface. During execution of the program, when a transition fires, RTPN writes the decimal number corresponding to the output channel to digital output interface. The contents of any output channel are either 0 or 1. The usage of this vector is later detailed in the example system.

There are two events for a transition firing, *start_firing* and *end_firing*. Between these the firing is in progress. The removal of tokens from a transition's input place(s) occurs at *start_firing*. The deposition of tokens to a transition's output places(s) occurs at the *end_firing*.

While transition firing is in progress, the time to end firing, called the *remaining firing time*, decreases from firing duration to zero at which its firing is completed. The execution rules of a RTPN include enabling and firing rules:

1. A transition $t \in T$ is *enabled* iff $\forall p \in P$ and $I(p,t) \neq 0$, $m(p) \geq I(p, t)$ and $X(p)$ has content 1.
2. Enabled in a marking m , t *fires* and results in a new marking m' following the rule:

$$m'(p) = m(p) + O(p,t) - I(p,t), \forall p \in P.$$

In the first rule, the first condition, $m(p) \geq I(p, t)$, ensures that the marking of each input place of a transition should be either equal to or greater than the input arc weight, and the second condition, ' $X(p)$ has content 1' ensures that for each input place of a transition, the second attribute of that place is 1 indicating that the input signal modeled by that place is true. The second rule, after firing of a transition, updates the marking of its output places and is the same as in ordinary PNs. The design procedure for formulating a RTPN based controller is shown in Fig. 6.1 and is briefed in the following five steps:

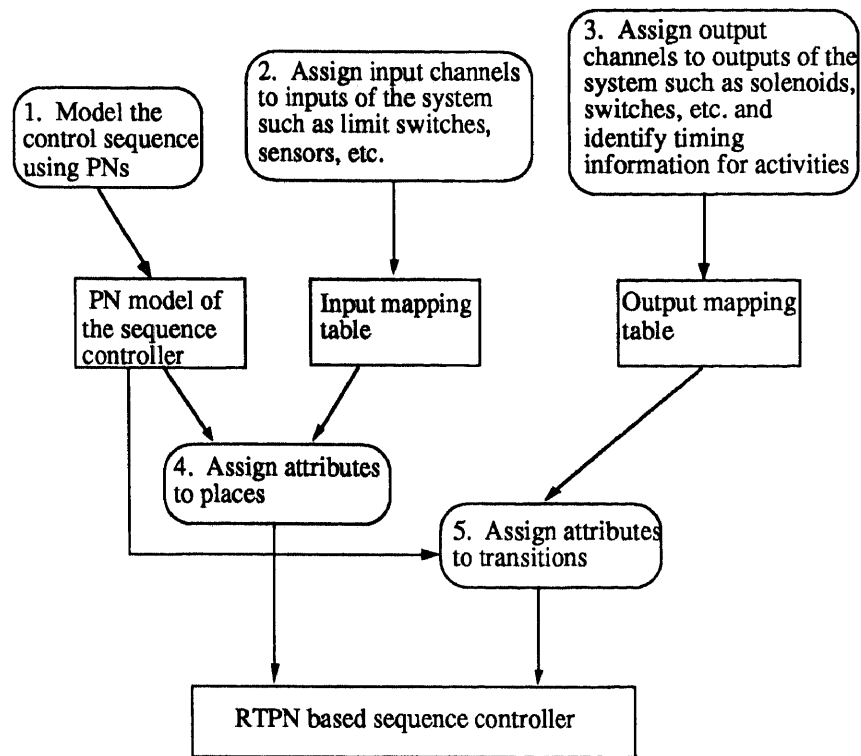


Figure 6.1 Procedure for formulating a RTPN based controller

1. Model the control sequence using PNs to obtain the PN model of the sequence controller.
2. Assign input channels to inputs of the system such as limit switches, sensors, etc. to formulate an input mapping table.
3. Assign output channels to outputs of the system such as solenoids, switches etc. Also, identify timing information for activities to obtain an output mapping table.
4. Using the input mapping table assign input channel number to each place (limit one channel per place) in PN based controller. The initial state of the system decides the initial marking of RTPN. In the PN model some places do not represent the inputs of the system as they represent the intermediate states of system or logical places to model counters in the sequence. Hence, no channel is assigned to these places, represented by "-".

5. Using the output mapping table and the action(s) that are modeled by a transition, assign a number to each transition in a PN based controller. The operations and the time delays given in the sequence to be controlled decide firing time function of RTPN. In the PN model some transitions may represent concurrent actions. Hence, care should be taken to assign the numbers for such transitions.

By following the above procedure a RTPN based controller can be formulated for a given sequence. The *instantaneous description (ID)* defined earlier in the case of timed PNs can also be extended for RTPNs. Observe that RTPNs can also be used for simulation and performance evaluation by associating times with transitions and dummy attributes for places modeling input signal information, and transitions modeling output signal information.

There are several ways to use PNs to perform the sequence control. One is based on "token game" (Zhou *et al.* 1992a) and the other converts the net into either Programmable Logic Controllers (Valette *et al.* 1983) or control code directly (Zhou and DiCesare 1993). The first scheme, as illustrated in Fig. 6.2, is used here.

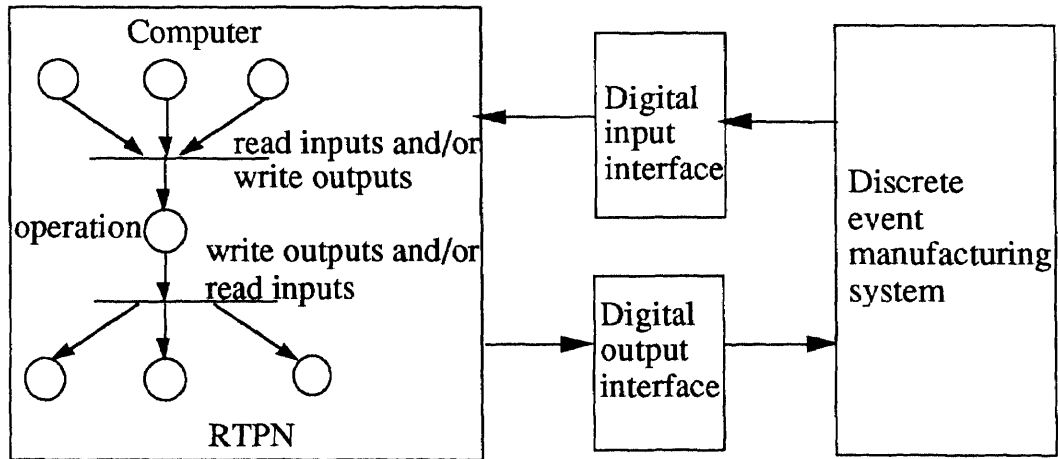


Figure 6.2 Controlling a system using a RTPN based controller

6.3. Real-time PNs and Other PN Extensions for Control

There are several types of PNs for control with various implementation schemes as shown in Table 6.1. The implementation schemes differ in terms of the extensions added to ordinary PNs in order to model timers, counters, input/output signal information, process status functions, method of execution of PN model, the hardware on which the PN is executed, and the level of sophistication.

Table 6.1 Various methods of Petri net based sequence control

Implementation scheme of Petri net controller	Hardware used	Reference
Petri net → High-level language description → Compiler → Control sequence tables and decision programs → control	Intel 8085 Assembler	Chocron and Cerny (1980)
Petri net → High-level language description → Interpreter → Control tables → control	A PLC based on ZILOG 80A microprocessor	Valette, <i>et al.</i> (1983)
Control Petri net → Interpreter → control	16 bit microcomputer	Murata, <i>et al.</i> (1986)
Petri net → Execution algorithm → control	VAX 11/780 and IBM-PC/XT	Crockett, <i>et al.</i> (1987)
Petri net → Software → control Note: Details of implementation are not given	IBM-PC/XT	Boucher, <i>et al.</i> (1989)
Petri net → Petri net description language → Token player → control	IBM-PC/XT	Zhou, <i>et al.</i> (1992a)
Control Petri net → Petri net simulator → control	8086 CPU and DSP (TMS32020)	Stefano and Mirabella (1991)
Real-time Petri net → Software system based on token player → control	IBM-PC/AT	Venkatesh, <i>et al.</i> (1994)

Even though RTPNs and earlier classes of PNs for control share similar principles, some of the differences between them are listed below:

1. Earlier studies use a variety of places to model timers and counters (Murata *et al.* 1986). This might make the model difficult to understand. In RTPNs neither new places nor transitions are introduced for modeling them. In order to find the process I/O and process status, Murata, *et al.* (1986) uses new functions, in the definition of their PNs which are called as C-nets. Also, Murata, *et al.* (1986) define a new place called 'act box' to define process I/O functions. However, in RTPNs such new functions are not included in order to keep the definition of RTPN simple and easy to understand. In order to find the process I/O and process status, attributes are included in RTPN. C-nets are more suitable at the higher level of cell control. For example, C-nets have a new place called 'receive box' to detect the request signal to start a job and to generate a numbered token corresponding to each job variety. In other words, C-nets can distinguish the part-varieties that enter into the system. RTPNs do not have that level of sophistication. C-nets can resolve conflicts by using a new place called 'conflict box' and the 'process status function'. RTPNs do not allow conflicts in the PN model. C-nets introduce a new transition 'count gate' to count the repetitive cycles. In RTPNs, ordinary transitions and places with multiple weights on arcs model such repetitive cycles. C-nets use a new place called 'timer box' to model timing. In RTPNs, timing is modeled by assigning it as a second attribute to a transition.
2. In (Stefano and Mirabella 1991) new sets of places are introduced for modeling I/O signals which increases the number of places in PN model. In RTPNs, I/O signals are modeled as attributes for places and transitions respectively. Places are assigned with input signals because they model the limit switches. Transitions are assigned with output signals because they actuate solenoids or motors. Hence, due to the use of attributes RTPNs have fewer nodes and links compared to PNs in

- (Murata *et al.* 1986, Stefano and Mirabella 1991), thereby reducing the net size and hence the graphical complexity.
3. In earlier works (Chocron and Cerny 1980, Boucher *et al.* 1989, Stefano and Mirabella 1991), the resetting of timers, counters, and emergency stop are not explicitly modeled. Furthermore, often they use additional functions to model and implement timers and counters (Stefano and Mirabella 1991). Using RTPNs all these features can be clearly modeled. The automatic resetting of timers and counters is also embedded in the execution of RTPNs.
 4. In Crockett, *et al.* (1987), actions (output signals) are associated with places and events (input signals) are with transitions. Macro places are introduced to model sub-PN, and switching places are introduced to resolve conflicts. In RTPNs, sub-PNs are modeled by ordinary places and transitions. Also, RTPNs do not allow conflicts. Crockett, *et al.* (1987) have not mentioned how their PNs can be used for both control and simulation without changing the original PN that is aimed for control only.
 5. Valette, *et al.* (1983) and Chocron and Cerny (1980) used PN description languages which are input to the translator or compiler. The translator or compiler generates the control tables that drive the actuators. In Valette, *et al.* (1983), the description language consists of instructions. These instructions specify 1) declarations of places and transitions and arcs among them, 2) VAR allowing variable declarations including input and output addresses corresponding to sensors and actuators and timer values, and 3) specification of boolean conditions attached to transitions and the operations that have to be executed. RTPNs can be implemented by a simple implementation scheme. For example, RTPNs eliminate the usage of description languages used in (Chocron and Cerny 1980, Valette *et al.* 1983) since the RTPN model can be directly used for control with the help of a token player. By adopting this implementation scheme, the need to translate the PN

model to higher level net description language and the generation of control tables is avoided. The actual implementation of the token player in RTPNs is transparent to the users and hence their only task to control a system is simplified to model the control logic. Zhou, et al. (1992a) also used simple PN descriptive language as input to the PN controller. Descriptive languages are efficient in terms of the memory requirements. For a given PN model, RTPNs need more memory space than other classes of PN that use descriptive languages because in case of RTPNs, the PN model is stored in the form of input and output matrix and thereby consumes more memory space.

6. Even though Boucher, *et al.* (1989) reported PN based control by associating output signals to transitions, no explicit comments are made how the input signal information is mapped into their PN model. They also used ordinary PNs without the addition of new places and transitions. The details on the method of PN execution are not given in Boucher, *et al.* (1989).
7. Zhou, *et al.* (1992a) used ordinary PNs for control. They also assign both input and output signal information to transitions. Even though, they use attributes for each place and transition, their attributes do not explicitly mention about how the input and output signal information is mapped on to places and transitions. Also, explicit modeling of timing information is not given in their PN implementation. In RTPNs such mapping is made clear because of the attributes to places and transitions, and time can be modeled as the second attribute of a transition. Hence, like any timed PNs, the deterministic delays given in a control sequence can be clearly modeled in RTPNs.
8. Sheng and Black (1989) also used ordinary PNs and expert systems to control a manufacturing cell. They modeled the cell using PNs and generated the cell control rules from the firing sequence of the PN using the concept of forward chaining approach in expert systems. However, the details of the mapping of the

input/output signal information and modeling of timers and counters are not reported. The differences between RTPNs and others are summarized below.

RTPNs are suitable at the lowest level of system control since they model the system more realistically by naturally mapping the limit switches, start, and stop buttons as places with attributes and actuators as transitions with attributes. RTPNs are easy to understand and require a simple procedure to implement. Since RTPNs and ordinary PNs have the same graphical representation, RTPNs are useful for the design recovery of ladder logic diagrams. This is because ladder logic diagrams can be converted into ordinary PNs which in turn can be converted into RTPNs by assigning proper attributes to places and transitions. However, using earlier classes of PNs the design recovery is difficult. This is because they (Chocron and Cerny 1980, Valette *et al.* 1983, Murata *et al.* 1986, Crockett *et al.* 1987, Stefano and Mirabella 1991) have special symbols for places and transitions to represent various functions in discrete-event control. The disadvantages of RTPNs include: 1) restriction of not allowing conflicts in the PN model, 2) lack of facility to distinguish product varieties, and 3) requirement of more memory space to store the structure of PN model because of the usage of input and output matrix.

6.4. Example: An Automatic Assembly System

In order to illustrate the control by RTPNs a simple assembly system shown in Fig. 6.3 is considered. It consists of a pallet storage station, assembly station, an inspection station, an AGV, and a robot. Two workpieces are assembled by the robot at the assembly station. After the assembly is completed, the robot unloads the finished product on a work table. The AGV transports empty pallets from pallet storage station to the assembly station. A Robot is also used to unload the finished product from work table and place it on the pallet which is on the AGV. Finally, the AGV transports the finished product loaded on pallet to inspection station for inspection. The availability of Workpieces 1 and 2 is sensed by two sensors A and B (not shown in figure), respectively.

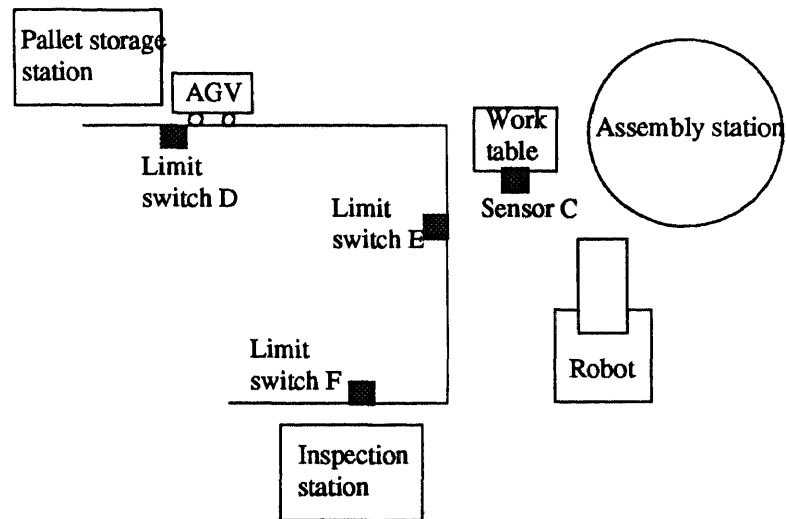


Figure 6.3 Schematic of an automatic assembly system

Limit switches D, E, and F sensing the presence of AGV are at pallet storage station, assembly station, and inspection station respectively. Sensor C recognizes the loading of finished product on work table by robot. Basically there are three operations to be executed in this system as follows:

1. Robot's assembly and transfer of the finished product on to work table,
2. AGV's travel from pallet storage station to assembly station, and
3. AGV's travel from assembly station to inspection station.

Assume the time duration for operation 'i' is ' τ_i ' time units and is executed when output relay 'i' is high. There is a digital I/O interface to control this system with details as shown in Tables 6.2 and 6.3.

Table 6.2 The input mapping table where X_i is input channel number.

Limit switch/Sensor	A	B	C	D	E	F
X_i	0	1	2	3	4	5

Table 6.3 The output mapping table where Y_i is number sent to the digital output interface.

Output relay	Output Channel number
1	0
2	1
3	2

Assume this system is controlled by PN with places A, B, C, D, E, and F modeling the corresponding sensors, and transition t_i modeling operation i . The attributes of places and transitions for this RTPN are listed in Table 6.4. The first and second attributes of a place are initial marking and assigned input channel number, respectively. For a transition the first attribute is time duration and the second attribute is the number to be sent to the I/O interface to activate the corresponding output relay.

Table 6.4 Attributes of places and transitions in the RTPN

Place	1st attribute	2nd attribute	Transition	Output relay	Output channel to be activated (x)	1st attribute	2nd attribute (2x)
A	1	0	t_1	1	0	τ_1	1
B	1	1	t_2	2	1	τ_2	2
C	0	2	t_3	3	2	τ_3	4
D	1	3					
E	0	4					
F	0	5					

Following the procedure illustrated in Fig. 6.1, the RTPN model obtained is shown in Fig. 6.4. Observe that time durations for these transitions need not be known to execute RTPN (or to control the system). However, they are needed when the same RTPN model is used for performance evaluation.

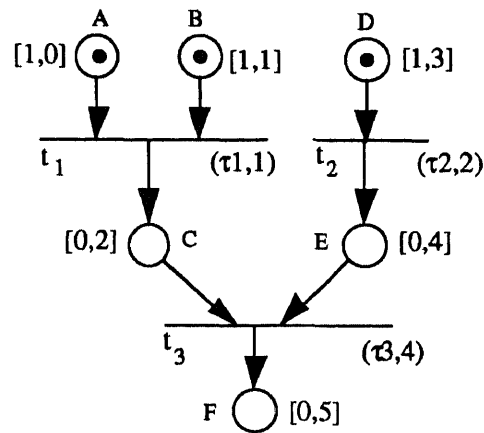


Figure 6.4 RTPN model for the system

6.5. A case study: An Electro-Pneumatic System

This section demonstrates the use of RTPNs for the control of concurrent tasks, through a practical electro-pneumatic system located in The Robotics Center, Florida Atlantic University. The system considered in this chapter is shown in Fig. 6.5. It consists of four pneumatic pistons (A, B, C, and D) which are operated by spring-loaded five ports and two-way solenoid valves. Each piston has two normally open limit switches. For example, when the end of piston A contacts limit switch a0 (a1), a0 (a1) is closed indicating that the piston A is at the end of its return stroke (forward stroke). The time that a piston takes for completion of either forward or backward stroke is one second. In manufacturing, typical functions of these pistons can be to load/unload the part from the machine table, to extend/retract a cutting tool spindle, etc.

Three push buttons are provided to start the system (switch SW1), to stop the system normally (switch SW2) and to stop the system immediately in emergency (switch SW3, ES). The system has 11 inputs corresponding to 8 limit switches (two for each piston) and 3 push buttons. The system has 6 outputs corresponding to 4 solenoid valves and two lights that indicate the status of the system. In this work, PNs are implemented through IBM PC and digital I/O interface as shown in Fig. 6.5.

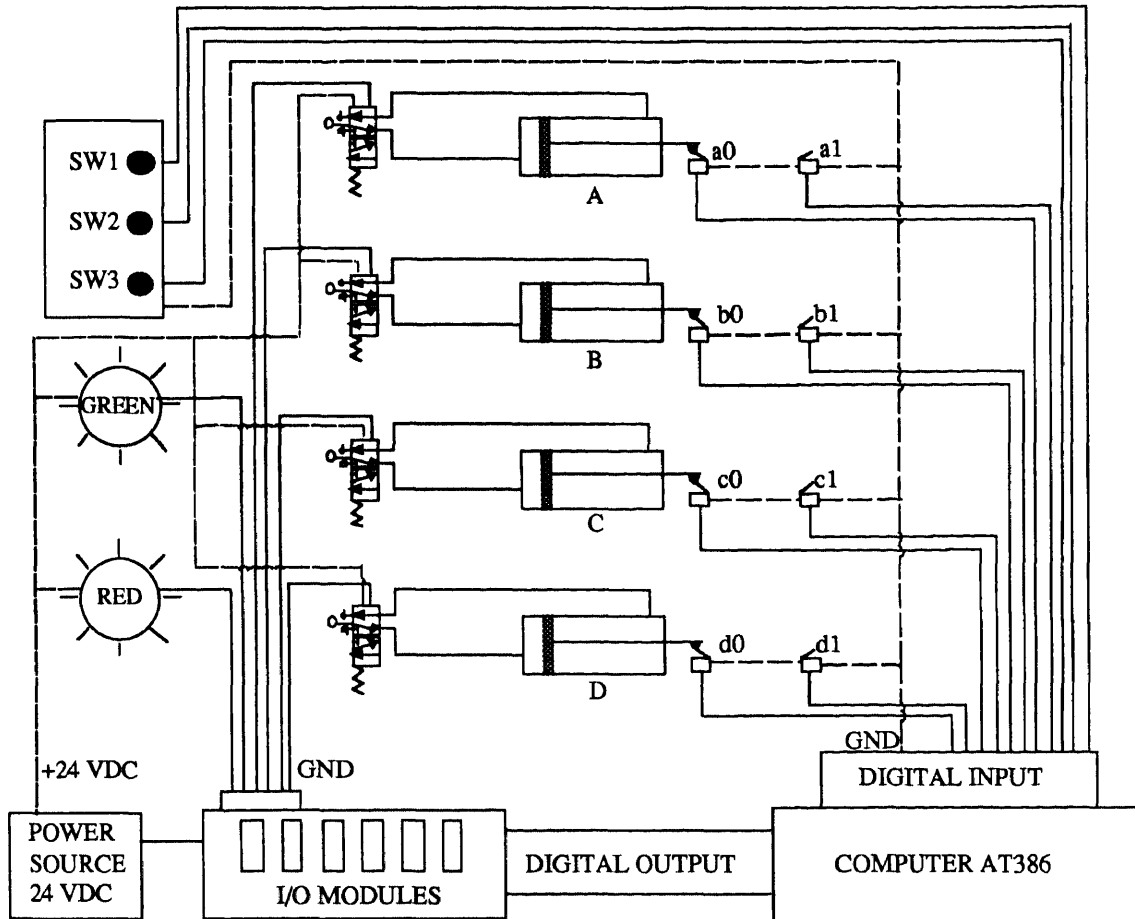


Figure 6.5 Schematic of an electro-pneumatic system considered

The procedure given in Fig. 6.1 is followed to design the control logic by PNs. Tables 6.5 and 6.6 show the I/O mappings of the system to PN respectively. For more hardware details on this system and its implementation with PNs refer to (Venkatesh *et al.* 1993).

Table 6.5 The input mapping table where X_i is input channel number

Switch	a1	b1	c1	a0	b0	c0	SW1	ES
X_i	0	1	2	4	5	6	8	10

Table 6.6 The output mapping table

Solenoid or Light	Output Channel number	Y_i	
		NA	ND
A	0	1	-1
B	1	2	-2
C	2	4	-4
D	3	8	-8
Light 1	4	32	-32
Light 2	5	64	-64

- Note: 1. Y_i is number sent to the digital output interface.
 2. NA (ND) means the number that is to be written on digital output interface to activate (deactivate) the solenoid/light

Sequence controller design

Consider that the system has to be controlled to execute the following sequence:

'START, A+, B+, {C+, A-}, {B-, C-}', where A+ represents that the piston has to do forward stroke and A- return one. {C+, A-} represents two concurrent actions taking place simultaneously: Piston C to do forward stroke and Piston A to do return one. Fig. 6.6 shows the RTPN corresponding to this sequence. Note that in the RTPN, a place has attributes $[n_1, n_2]$ where n_1 is the first attribute representing initial number of tokens and n_2 is the second one mapping input channel number. Similarly, a transition has attributes (n_1', n_2') where n_1' is the firing duration and, n_2' is the number to be written on the digital output interface. When concurrent actions such as {C+, A-} are to be modeled, care should be taken to associate the second attribute to transitions as shown in Table 6.7.

Table 6.7 Attributes of transitions modeling actions

Action	Output channel to be activated	Output channel to be deactivated	Transition modeling action	Second attribute of transition
do A+	0	-	t_2	$2^0 = 1$
do B+	1	-	t_4	$2^1 = 2$
do {C+, A-}	2	0	t_6	$2^2 - 2^0 = 3$
do {B-, C-}	-	1, 2	t_8	$-2^1 - 2^2 = -6$

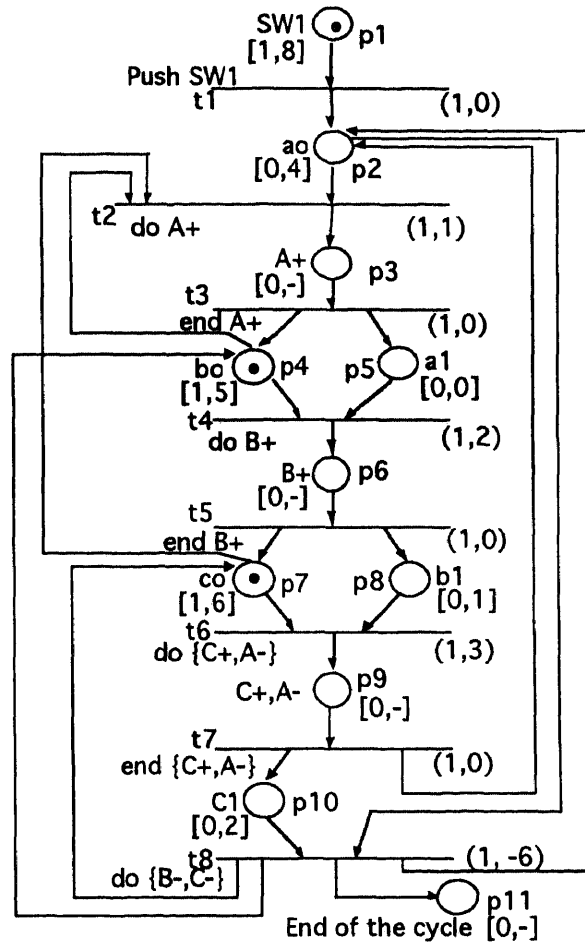


Figure 6.6 RTPN for sequence 1: ST, A+, B+, {C+, A-}, {B-, C-}

6.6. Software Description to Execute Real-time Petri Nets

The software package developed to execute ATPNs is written in C and presented in Appendix B. The sample input and output for an RTPN is also presented in this Appendix. The current software package runs on IBM compatible PCs by interacting with the digital input/output interface and the system under control. The biggest PN model that can be controlled using this package contains one hundred places and one hundred transitions. However, larger nets can be controlled when this software package is installed in a workstation with more memory. The software package has five major modules with their functions described below.

Read_Petri_net

This module reads an input file that specifies the structure of the PNM; transition timing durations and output signal vector; initial marking and the input signal vector. The structure of the PNM means the connectivity between places and transitions including the weights on the connecting arcs. It is given as an input and output matrix as shown in the sample input presented in Appendix B.

Enabled_transitions

This module generates an F-function as an output function. It scans the whole PNM at each instant of time and finds the transitions that are ready to fire. Thus, the F-function indicates the transitions that are enabled to fire with respect to real-time. The F-function is the input for modules "Conflict", "New_marking", and "Main".

Conflict

This module determines the transitions that are in conflict and stops the program execution until the conflict is resolved. Once the conflict is resolved, the program execution is resumed. A conflict in PNM results when an element is shared by two other elements of the system (e.g. a single robot serving two machines that demand service at the same time). In such cases, the module detects the conflicts and interactively resolves it based on the criteria entered by the user.

Minimum_time

This module scans the whole PNM and detects the transition that has minimum time to fire. As there can be more than one transition with minimum time, the outputs from this module are both the number of transitions with minimum time and their identity. This module is important for simulation and performance evaluation but not for real-time control.

New_marking

This module contains two submodules: 1) "Read_marking" checks for the second attributes of all places that are inputs for enabled transitions. 2) "Update_marking" fires the transitions and changes the current marking. Read_marking uses F-function as input. If a

transition t_i is enabled it checks for the second attribute of all input places of t_i . If the second attribute of an input place is high it sets the variable "flag" corresponding to that place as TRUE. After flags corresponding to all the input places of t_i are set to TRUE, it removes the tokens from these places and sends a signal to "update_marking" to fire t_i . After receiving the signal corresponding to the transition to be fired from "read_marking", "update_marking" sends the second attribute of the transition to be fired to the digital output interface and deposits tokens in all the output places of the transition fired.

Main

This module coordinates the functioning of above modules and generates a status report of the system elements. The report is stored in an output file which is updated whenever a transition is fired in the PNM. Whenever an event occurs in the system, the output file is appended by the time at which the event occurred, marking of the PNM, F-function, Q-function, and A-function. Recall that marking, F, Q, and A functions are described earlier in Chapter 3 while defining the 'instantaneous description' of timed PN.

By including dummy value for the second attribute of each place and associating timing values to each transition, this package can also be used for performance evaluation. The functional objective of this package and the package included in the Appendix A is the same except that latter can do only simulation but not real-time control.

CHAPTER 7

COMPARISON OF REAL-TIME PETRI NETS AND LADDER LOGIC DIAGRAMS

7.1 Introduction

According to a given logic specification, a sequence controller in a manufacturing system synchronizes and coordinates the operations of these units. The sequence controller in this chapter means a class of discrete event controllers without choices in executing the operations/activities. Design methods for sequence controllers play a prominent role in advancing industrial automation. A comparison of Real-time Petri nets (RTPNs) and ladder logic diagrams (LLDs) is presented in this chapter.

Traditionally, LLDs are used to capture the sequence of operations executed by the system's control software. They specify the I/O procedures of the Programmable Logic Controller (PLC) that drive and perform the repetitive operations of the system. These diagrams grow so complex that locating the cause when a problem is detected becomes extremely difficult (Chaar *et al.* 1991). Furthermore, their usage is limited to control the system; they can not analyze and evaluate the qualitative and performance characteristics. They often need significant changes as the specification changes resulting in difficult maintenance problems.

The increasing complexity and varying needs of modern discrete manufacturing systems have challenged the use of LLDs for programmable logic controllers. The methodologies based on research results in computer science have recently received growing attention by academic researchers and industrial engineers in order to design flexible, reusable, and maintainable control software. Particularly, Petri nets (PNs) are emerging as a very important tool to provide an integrated solution for modeling, analysis, simulation, and control of industrial automated systems. However, in order to establish PNs as alternative to LLDs there is a need for benchmark studies to formally compare them.

It is observed that none of the earlier studies on PN based control has compared LLDs and PNs for the design of sequence controllers in detail and formally. Boucher, *et al.* (1989) used LLDs and PNs to control a same manufacturing system and reported the graphical representation by PNs makes the controller more tractable than that of LLDs. However, they have not formally quantified the comparison between PN and LLDs to design sequence controllers. Some of the problems associated to compare PNs and LLDs are: (1) unlike in case of LLDs, there exist several classes of PNs with various implementation schemes for discrete control as shown in Table 6.1, and (2) identification of the criteria with respect to which the comparison should be performed.

The contribution of this chapter is two fold. First, certain criteria are identified to compare LLDs and PNs in designing sequence controllers subject to the changing control requirements. The comparison is performed through a practical industrial automated system. Secondly, some analytical formulas and a methodology are developed to estimate the number of basic elements used in the PN and LLD designs prior to their constructions. The results will be useful for researchers and engineers to design control systems for complex industrial automated systems.

The goal of this chapter is to compare LLDs and PNs when they are used to design discrete event controllers for manufacturing systems. The objectives of this chapter are:

1. To identify the criteria to compare LLDs and RTPNs for design of sequence control,
2. To compare LLDs and RTPNs in designing sequence controllers that respond to specification changes,
3. To formulate mathematical formulas to calculate the number of basic elements to model certain building blocks of logic models using PNs and LLDs, and
4. To present a methodology that synthesizes these analytical formulas for estimating the total number of basic elements required to design sequence controllers using PNs and LLDs.

The comparison criteria between LLDs and RTPNs are identified in Section 3. The comparison between LLDs and RTPNs is performed through a practical system as a case study in Section 4. Section 5 presents analytical formulas to estimate the basic elements required to implement certain building blocks of control logic. A methodology that uses these formulas to estimate the number of basic elements required to implement a given control logic specification using PNs and LLDs is presented in Section 6. In Section 7, this methodology is illustrated through two examples and the case study considered in Section 4. Finally summary of this chapter is presented in Section 8.

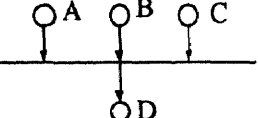
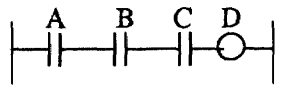
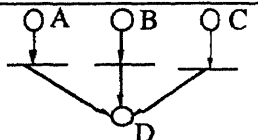
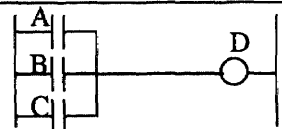
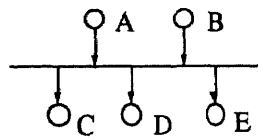
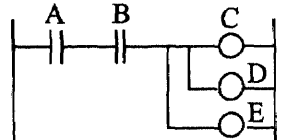
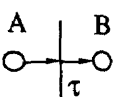
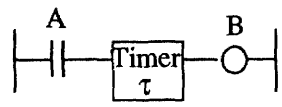
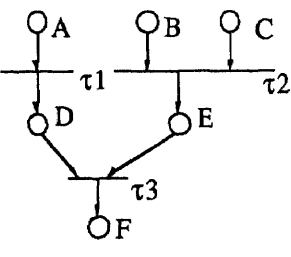
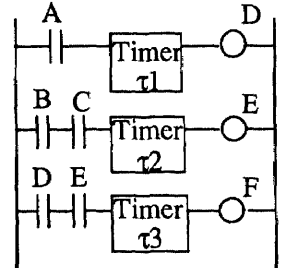
7.2. Comparison Criteria for Control Logic Design by PNs and LLDs

The application of LLDs for sequence control is widely known as they are used by several industries (Pessen 1989, Michel 1990). Graphically, RTPNs are same as PNs except that in the former, places are given attributes to model input sensory information and transitions are associated with attributes to model output and timing information. Hence, the comparison presented in this chapter corresponds to RTPNs which are referred to PNs hereon for short. In order to start an operation in a system some conditions have to be fulfilled which are called as pre-conditions. Upon its completion, an operation results in some conditions which are called as post-conditions. A transition in a PN models an operation. The input and output places of a transition model the pre- and post- conditions respectively.

In LLDs as shown in the previous section, an operation is modeled by activating an output coil corresponding to a relay or a solenoid. In addition to an output coil a relay consists of number of contacts, some normally open (N.O) and some normally closed (N.C). In order to energize the output coil of a relay, the corresponding contacts of that relay should be switched, i.e., an N.O. contact closes, while an N.C. one opens. The LLD also uses various input and output elements. A typical output element is a solenoid which is usually used to actuate pneumatic or hydraulic solenoid valves. Push-button and limit switches constitute input elements.

The contacts of relay along with the input elements constitute pre-conditions. The solenoids and output coils are referred to as post-conditions. The logic and other basic building blocks used in sequence control are modeled by PNs and LLDs as shown in Table 7.1. The explanations for these symbols is given below.

Table 7.1 Representations by Petri nets and ladder logic diagrams

Logic constructs	Petri nets	Ladder logic diagrams
Condition or the status of a system element	○ Place	No explicit representation
An activity	—— Transition	No explicit representation
Flow of information or material.	→ Directed arc	No explicit representation
Objects such as machines, robots, pallets, etc.	○ Token(s) in place(s)	No explicit representation
<u>Logical AND</u> IF A = 1 and B = 1 and C = 1 THEN D = 1		
<u>Logical OR</u> IF A = 1 or B = 1 or C = 1 THEN D = 1		
<u>Concurrency</u> IF A = 1 and B = 1 THEN C = 1 and D = 1 and E = 1		
<u>Time delay</u> IF A = 1 THEN delay "τ time units" B = 1		
<u>Synchronization</u> IF A = 1 THEN delay "τ1 time units"; D = 1 IF B = 1 and C = 1 THEN delay "τ2 time units"; E = 1 IF D = 1 and E = 1 THEN delay "τ3 time units"; F = 1		

The first four rows show the basic PN elements to model conditions, status, activity, information and material flow, and resources. Note that LLDs do not have the corresponding explicit representations. Logical AND and logical OR can be easily modeled by both PN and LLD with similar complexity. Other important concepts, e.g., concurrency, time delay, and synchronization are also illustrated in Table 7.1. The systematic methods to formulate PN models can be seen in (Zhou *et al.* 1992b, Zhou and DiCesare 1993) and the methods of developing LLDs can be seen in (Pessen 1989, Michel 1990). Two of the important factors for comparison of PN and LLD for discrete event control are identified as design complexity and response time.

Design Complexity

Design complexity is defined as the complexity associated in designing the control logic for a given specification. Since it is influenced by many factors, e.g., the experience of designers, size of control program, and number of dynamic steps necessary for coding or changing control program, it is very hard to formally quantify. However, it can be characterized by two factors namely graphical complexity and adaptability for change in specification.

Graphical complexity: It is mainly determined by the number of nodes and links for a given graphical control logic design. Graphical complexity influences the understandability of control logic by people who do not have knowledge of either PN or LLD. Hence, it is an important factor in designing the logic at the initial stages and subsequently debugging the errors during its implementation. The graphical complexity in terms of the net size is a major issue in manufacturing systems (Zhou, *et al.* 1992a) and it was reported that the simpler the graphical representation of control logic, the easier to track the controller (Boucher, *et al.* 1989). Graphical complexity may also influence response time as described later. For example, in the case of LLD, the response time depends on the size of the LLD because as the number of rungs increases scan time also increases. Hence, a shorter LLD results in a faster controller (Pessen 1989).

Adaptability to change in specifications: This factor is gaining much importance in the context of agile manufacturing in which control sequences need to be changed often to meet the dynamically changing requirements of the market. The control software should be easily adaptable to changes in specifications in order to improve the software productivity and thus keep minimal development time. One of two designs is said to be more adaptable if it needs fewer changes compared to another in order to fulfill a specification change.

Response Time

Response time is termed as scan time in LLD literature and execution time in PN. Response time can be defined as the minimum time that a control model takes to respond to an external event. Its importance to control real-time systems is clear since it decides how fast the control system responds to an event in the system/process under control.

An important factor that influences graphical complexity and adaptability is the physical appearance (size) of the model, whereas the response time is influenced by not only physical appearance but also the method of implementation. Methods of implementation constitute the software and hardware used to control the system using either PNs or LLDs. Graphical complexity and adaptability cannot be quantified whereas response time can be measured accurately, given a logic design and implementation. However, since there are several ways to implement PNs as shown in Table 6.1 and LLDs (Michel 1990, Pessen 1989) both in terms of hardware and software, it is very difficult to make a fair comparison of LLDs and PNs solely on the basis of response time criterion.

Hence, we propose some common measures that give an idea about the graphical complexity, adaptability, and response time. One such measure is the number of nodes and links used in a control logic model. For PNs nodes are places and transitions and links are arcs; whereas in LLDs, nodes are normally opened/closed switches, timers, counters, relays, and push buttons, and links are connections. If more nodes and links are used in a design, it is graphically more complex and thus may need more response time. In the similar manner, a control logic is more adaptable if it needs fewer changes in the number of

nodes and links compared to another logic to meet a change in specification. Hence, this study uses the number of nodes and links in an LLD and a comparable PN as a measure to compare their design complexity and response time. For the sake of convenience, nodes and links are called as basic elements.

7.3. Comparison Through An Electro-Pneumatic System

One effective way to perform the comparison between LLDs and PNs is through an actual industrial automated system. The system shown in Fig. 6.5 is used for this comparative study. The same input/output mapping tables used in Chapter 6 are used in this chapter also.

7.3.1. Sequence Controller Design

Sequence 1: START, A+, B+, {C+, A-}, {B-, C-}

Consider that the system has to be controlled to execute the above sequence where A+ represents that the piston has to do forward stroke and A- return one. {C+, A-} represents two concurrent actions taking place simultaneously: Piston C a forward stroke and Piston A a return one. Fig. 7.1 (a) shows the LLD and Fig. 7.1 (b) shows the PN corresponding to this sequence. Note that in the PN, a place has attributes $[n_1, n_2]$ where n_1 is the first attribute representing initial number of tokens and n_2 is the second one mapping input channel number. Similarly, a transition has attributes (n_1', n_2') where n_1' is the firing duration and, n_2' is the number to be written on the digital output interface.

As discussed earlier, basic elements in a LLD or PN are nodes and links. In LLD, nodes are push buttons, normally opened switches, normally closed switches, output relay coils, timers, and counters and links are connections which connect these nodes. In PN, nodes are places and transitions and links are arcs connecting them. The LLD shown in Fig. 7.1 (a) has 58 basic elements (24 nodes and 34 links), whereas the PN shown in Fig. 7.1 (b) has 50 basic elements (21 nodes and 29 links).

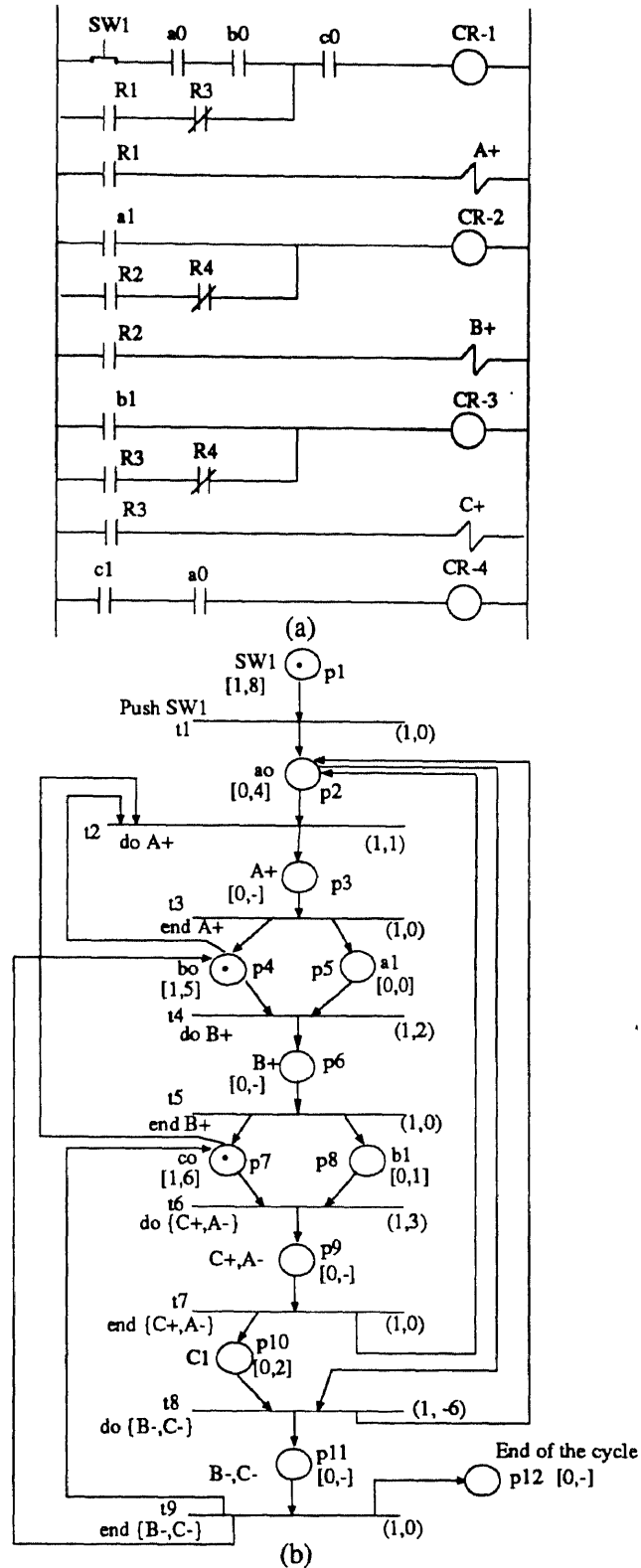


Figure 7.1 (a) LLD and (b) PN for Sequence 1: ST, A+,B+,{C+,A-}, {B-,C-}

Even though PN uses fewer basic elements, it looks more complex due to the fact that all loops have to be closed to represent repetitive processes. This complicates the graphical appearance of PN compared to LLD. However, when the specification is changed the complexity of LLD grows faster than PN and this is illustrated below.

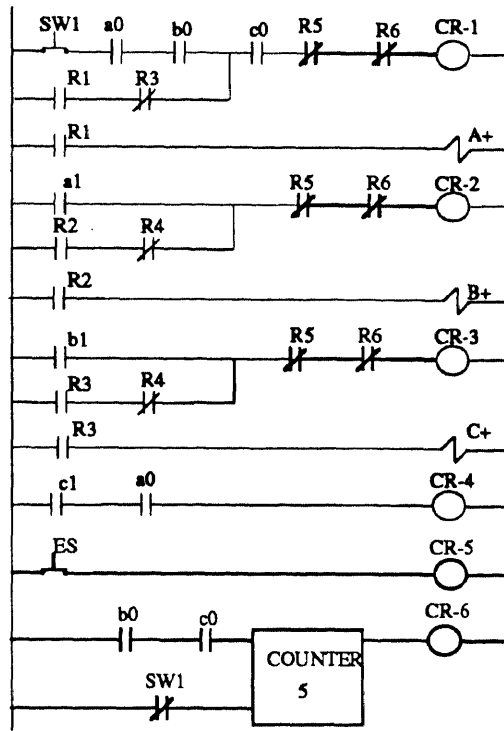
7.3.2. Control for other sequences

In order to compare the LLDs and PNs various sequences with increasing complexity are considered. These sequences will involve emergency stop, counters for counting the number of repetitive operations, and timers for providing delays between certain operations. In order to highlight the changes in a model (PN or LLD) from one sequence to next sequence, given a model for a sequence, the additional basic elements needed to model the next sequence are shown by bold lines.

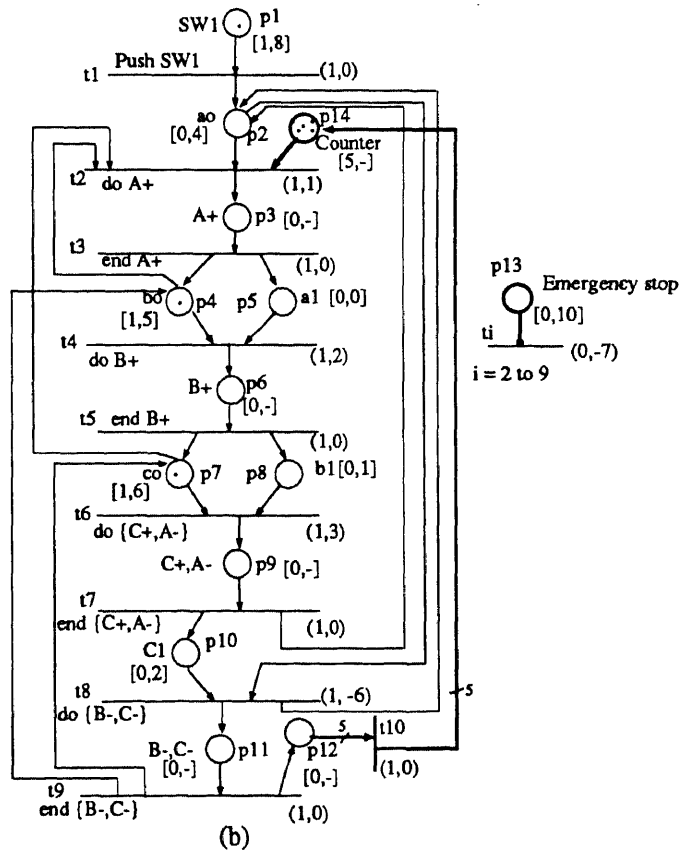
Sequence 2: START, 5 [A+, B+, {C+, A-},{B+, C-}] (with emergency stop and counter)

Now, consider that the specification is changed such that the new control sequence is indicated as above. In this sequence, there is a need to provide emergency stop and a counter.

Both the LLD and the PN are implemented such that when emergency stop switch, ES is pressed, the whole system including the active elements are immediately stopped. In other words, when switch ES is pressed all the solenoids are immediately deactivated. However, there exists several ways of modeling an emergency stop. Fig. 7.2 (a) shows the LLD and Fig. 7.2 (b) the PN corresponding to this sequence. In order to incorporate emergency stop, PN uses a place p_{13} with an inhibitory arc as an input place for t_2 - t_9 whereas LLD uses an output coil CR-5. When ES is pressed, CR-5 would become active and deactivates the coils CR-1, CR-2, and CR-3 which drive pistons A, B, and C respectively.



(a)



(b)

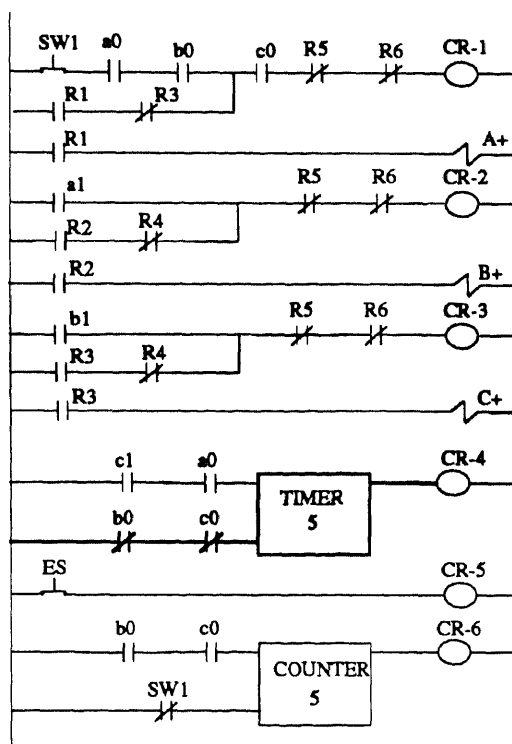
Figure 7.2 (a) LLD and (b) PN for Sequence 2

In the PN, when the ES is pressed, the output transition corresponding to the place modeling ES writes an integer on the digital output interface which deactivates all the solenoids. The LLD shown in Fig. 7.2 (a) has 87 basic elements (37 nodes and 50 links), whereas the PN shown in Fig. 7.2 (b) has 64 basic elements (24 nodes and 40 links). Notice that there is no significant change in the physical appearance of LLD or PN compared to Sequence 1.

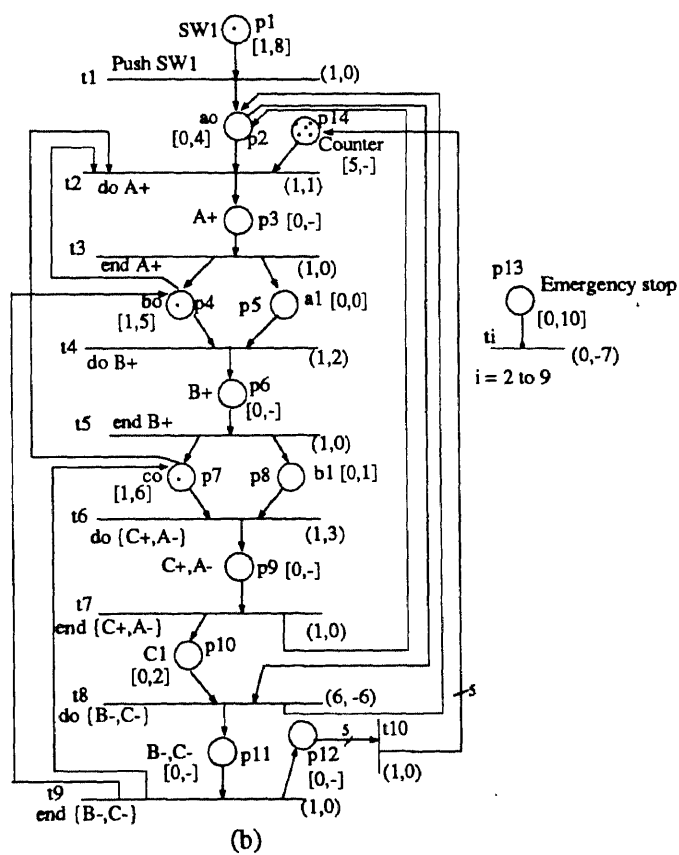
Observe that LLD needs more additional basic elements compared to the PN. This is because the PN needs only one place p_{14} with an arc as an input to t_2 to implement the counter. The counter resetting is modeled by t_{10} . In contrast to this, LLD additionally needs two normally opened switches b_0 , c_0 as inputs for counter, four normally closed switches (SW1 as a reset signal for counter, and switch R6 as an input for CR-1, CR-2, and CR-3), a counter, and 16 more links to implement this sequence.

Sequence 3: START, 5 [A+, B+, {C+, A-}, 5 sec, {B+, C-}] (with emergency stop, counter, and timer)

The sequence is changed such that there is a need to incorporate a timer in the control logic to provide 5 seconds delay in between {C+, A-} and {B+, C-}. Fig. 7.3 (a) shows the LLD and Fig. 7.3 (b) the PN accordingly. The LLD shown in Fig. 3 (a) has 94 basic elements (40 nodes and 54 links), whereas the PN shown in Fig. 7.3 (b) is same as the one shown in Fig. 7.2 (b) with 64 basic elements. It is observed that LLD needs more additional basic elements compared to the PN. This is because the same PN used in the earlier sequence is used without changing the physical appearance. In the PN shown in Fig. 7.2 (b), only the first attribute of t_8 is changed to obtain the PN shown in Fig. 7.3 (b) to incorporate time delay of 5 seconds in the sequence. On the other hand, LLD needs additionally two normally closed switches, a timer, and four links to implement this sequence.



(a)



(b)

Figure 7.3 (a) LLD and (b) PN for Sequence 3

Sequence 4: START, 3[A+, B+, {C+, A-}, 5 sec, {B+, C-}], 10 sec, 2[A+, B+, {C+, A-}, 5 sec, {B+, C-}] (with emergency stop, counters, and timers)

This new sequence represents a complex one in which Sequence 3 is divided into two segments (one with three cycles and another with two cycles) with 10 seconds time delay between them. The LLD shown in Fig. 7.4 has 130 basic elements (54 nodes and 76 links), whereas the PN in Fig. 7.5 has 69 basic elements (26 nodes and 43 links). In this case also note that LLD needs more additional basic elements compared to PN. This is because the PN needs only one additional transition t_{11} with an input arc from a new place p_{15} and an output arc to p_{14} to reset the counter.

7.3.4. Discussions

As mentioned in Section 2 the number of basic elements is a common measure that gives an idea about graphical complexity, adaptability, and response time. It is observed that as the specification changes, PN requires fewer changes compared to LLD.

Table 7.2 summarizes how the number of basic elements increases in LLD and PN as the complexity of sequence control specifications increases.

Table 7.2 Comparison of the basic elements in LLD and PNs

Sequence #	Basic elements in LLD			Basic elements in PN		
	Nodes	Links	Total	Nodes	Links	Total
1	24	34	58	21	29	50
2	37	50	87	24	40	64
3	40	54	94	24	40	64
4	54	76	130	26	43	69

It can be inferred that the PN shown in Fig. 7.1 (b) used to execute the first sequence is slightly modified to get the PN shown in Fig. 7.5 corresponding to the last sequence. However, the LLD shown in Fig. 7.1 (a) is significantly modified to get the LLD shown in Fig. 7.4.

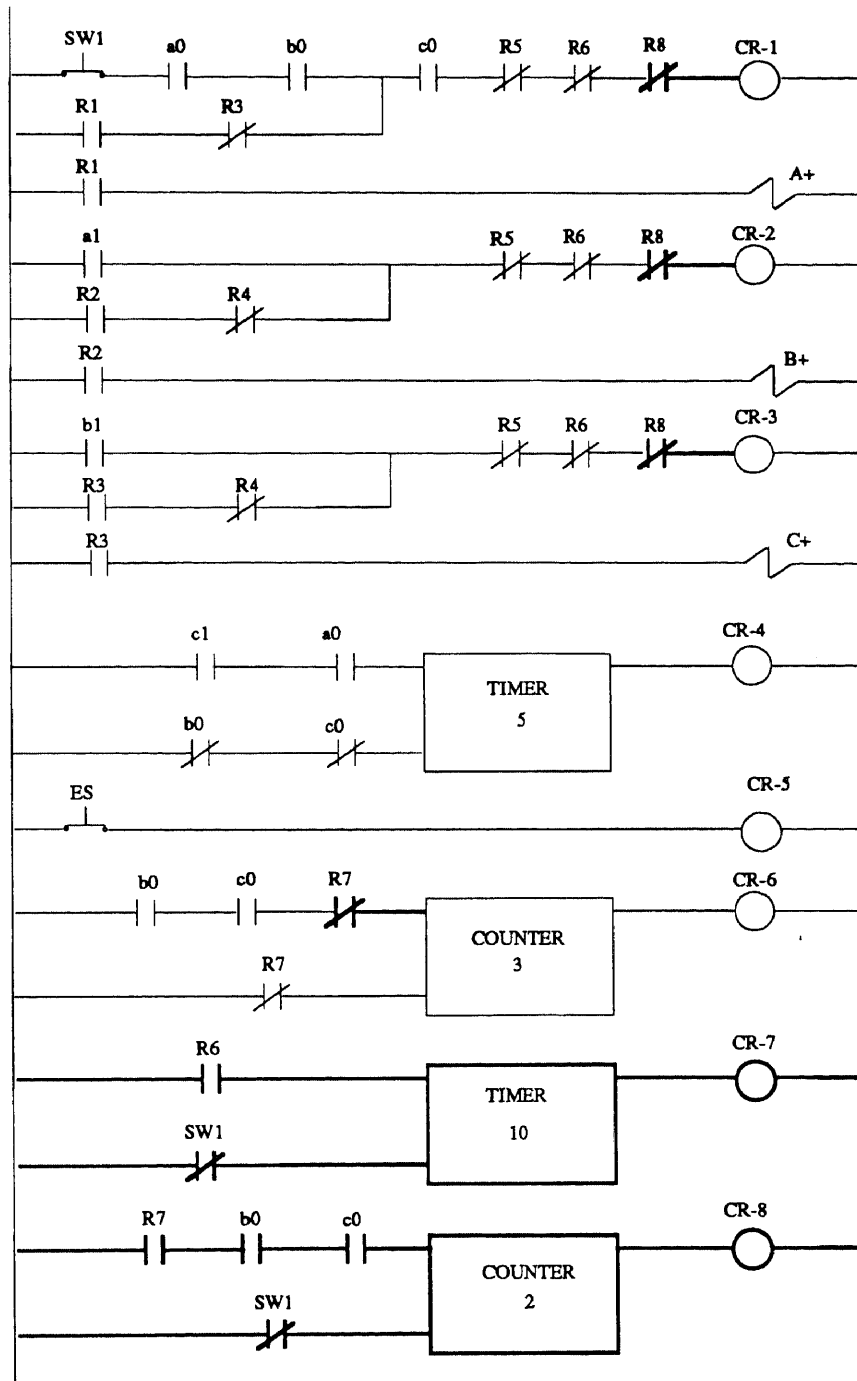


Figure 7.4 LLD for Sequence 4

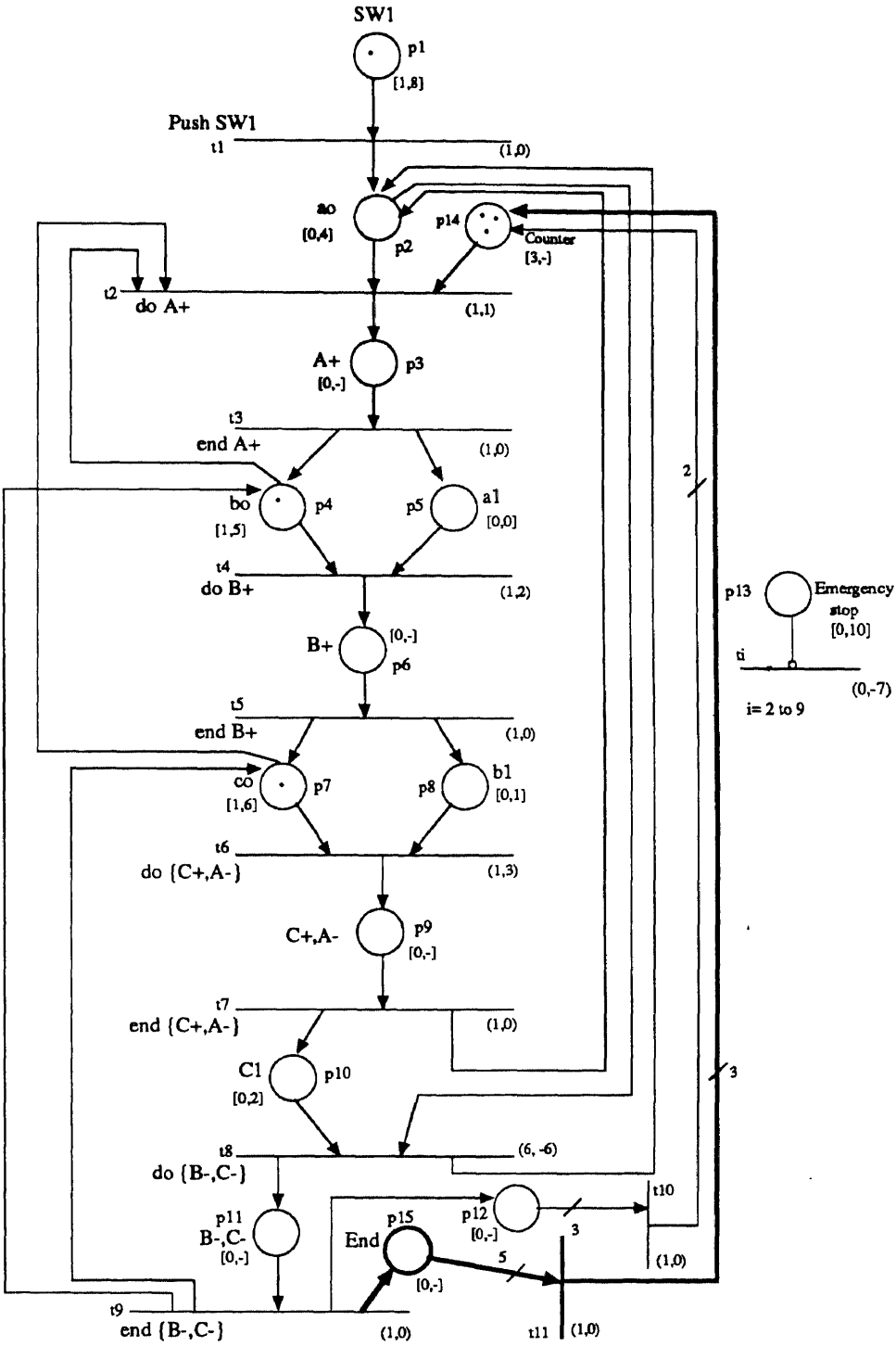


Figure 7.5 PN for Sequence 4

The modifications in terms of basic elements can be quantified using Table 7.2. Also, observe that the physical appearance of PN is preserved (with slight modifications) starting from the first sequence to the last one. This is not true in the case of LLDs as in the previous figures. Furthermore, this case study reveals that PNs and LLDs do not differ much when the control sequence is relatively simpler as seen in the first sequence.

In fact, PN model may appear more complex than LLD at the first sight as shown for the first sequence. However, when this sequence is gradually modified to result in a complex one, PNs are more easily modifiable and hence maintainable than LLDs.

Ease in modifiability and maintainability yields several advantages such as improvement in readability, understandability, and reliability as concluded in (Murata, *et al.* 1986). In LLDs, nodes appear multiple times which may lead to difficulty in understanding the logic and cause errors in developing the logic. LLDs needs more basic elements to model timers and counters compared to PNs. In addition to these findings, the following points are experienced during the design and implementation of sequence controllers using LLDs and PNs:

1. Using PNs the control logic can be qualitatively analyzed to check properties such as absence of deadlocks and presence of reinitilizability in the system. Using LLDs qualitative analysis is not possible until it is simulated or implemented.
2. During implementation of control Sequences 3 and 4 it is found that debugging of the control logic with LLDs is difficult compared to PN. This is because PNs help to dynamically track the system with the help of the states of places and transitions (Venkatesh and Ilyas 1995).
3. Using PNs, the initial state of the system can be directly represented by its initial marking.
4. The reason for the compleity of LLD and the difficulty in debugging is that, in an LLD, a node appears more than once in the diagram. Due to this, tracking of control from one node to other becomes very difficult.

7.4. Analytical Formulas to Evaluate the Complexity of PNs and LLDs

In general, the fewer the basic elements in a controller, the better the model used to implement the controller (Boucher *et al.* 1989, Pessen 1989a,b). The number of basic elements decides the length of the control model. A shorter model uses less number of basic elements and is usually easier to understand, check, diagnose, and maintain. It also takes less time to enter the controller/computer (Pessen 1989, Venkatesh *et al.* 1994). Moreover, when low cost controllers with short memory are used it is possible that they may run out of memory if the control model is too large (Pessen 1989). Counting of basic elements in a PN or LLD becomes cumbersome when their control specification becomes complex. Also, before physically modeling the given control logic with either LLD or PN, there is a need for a method that can help control engineers select between PN and LLD by estimating the basic elements used in these two models.

Motivated by these reasons, this section presents some analytical formulas to estimate the basic elements used in a PN or LLD. Most of the control specifications can be modeled by using the logic constructs such as logical AND (NAND), logical OR (NOR), sequential model, and timed sequential model. It is obvious that AND/OR basically requires the same number of basic elements as NAND/NOR. Hence, the analytical formulas presented for AND/OR can also be applied to NAND/NOR cases. The formulation of these analytical formulas consists of their derivation and verification. This is done by physically modeling each logic construct using PN and LLD as described below.

For the sake of convenience, the number of basic elements used in PN and LLD are represented by α and β , respectively. α is the summation of α_n and α_l , where α_n and α_l represent the number of nodes and links used in PN respectively. Similarly, β is the summation of β_n and β_l . We also define a function Δ which is defined as $\Delta = \alpha - \beta$ which helps to decide whether PN is better or LLD is better in terms of design complexity.

For example, if Δ is negative, PN is preferred as it uses a smaller number of basic elements. On the other hand LLD is preferred when Δ is positive. In all the formulas presented in this chapter, m stands for number of pre-conditions, and n for number of post-conditions.

7.4.1. Logical AND, Logical OR, and Sequential modeling

Logical AND

Figure 7.6 shows the PNs and LLDs for deriving the formulas to obtain α and β for logical AND. They are given as follows:

For PN: $\alpha = 2(m + n) + 1$, where
 $\alpha_n = m + n + 1$, and
 $\alpha_l = m + n$

For LLD: $\beta = 2m + 3n$ where $\beta_n = m + n$, and $\beta_l = m + 2n$

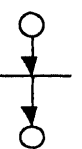
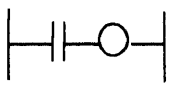
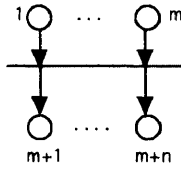
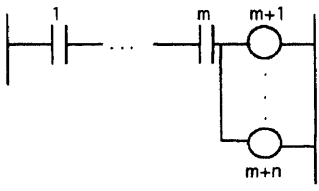
Petri nets	Ladder logic diagrams
 <p>Nodes = 3 Links = 2</p>	 <p>Nodes = 2 Links = 3</p>
 <p>Nodes = $m + n + 1$ Links = $m + n$</p>	 <p>Nodes = $m + n$ Links = $m + 3n$</p>

Figure 7.6 PNs and LLDs modeling logical AND

Here, $\Delta = 1 - n$. This indicates when $n = 1$, both PN and LLD yield the same number of basic elements and if $n > 1$, PN is preferred.

Logical OR

Figure 7.7 shows the PNs and LLDs for deriving the formulas to obtain α and β for logical OR.

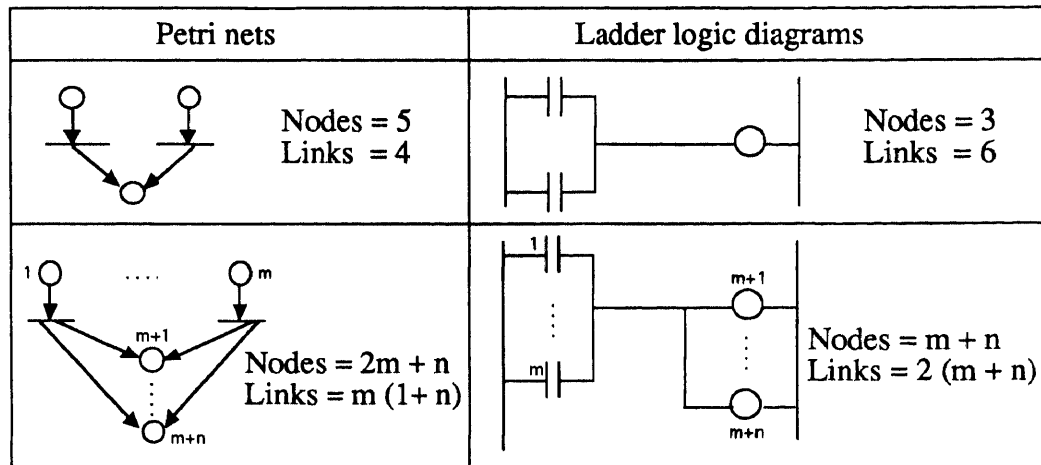


Figure 7.7 PNs and LLDs modeling logical OR

They are given as follows:

For PN: $\alpha = 3m + n + mn$, where

$$\alpha_n = 2m + n, \text{ and}$$

$$\alpha_l = m(1 + n)$$

For LLD: $\beta = 3(m + n)$, where $\beta_n = m + n$, and $\beta_l = 2(m + n)$

$\Delta = -n(2-m)$. LLD is better than PN for $m > 2$.

Sequential modeling

A condition is called a sequential condition when it acts as a pre-condition for its output operation (say operation $i+1$ and post conditions for its input operation (say operation i). The assumption here is that all operations except the first and last operations in the sequence have only one pre-condition and post-condition.

For sequences that violate this assumption, number of basic elements can be found by decomposing the sequence with logical AND where some portions of the sequences have $n = 1$ and others have $n > 1$. Figure 7.8 shows the PNs and LLDs to derive the formulas to obtain α and β for sequential models.

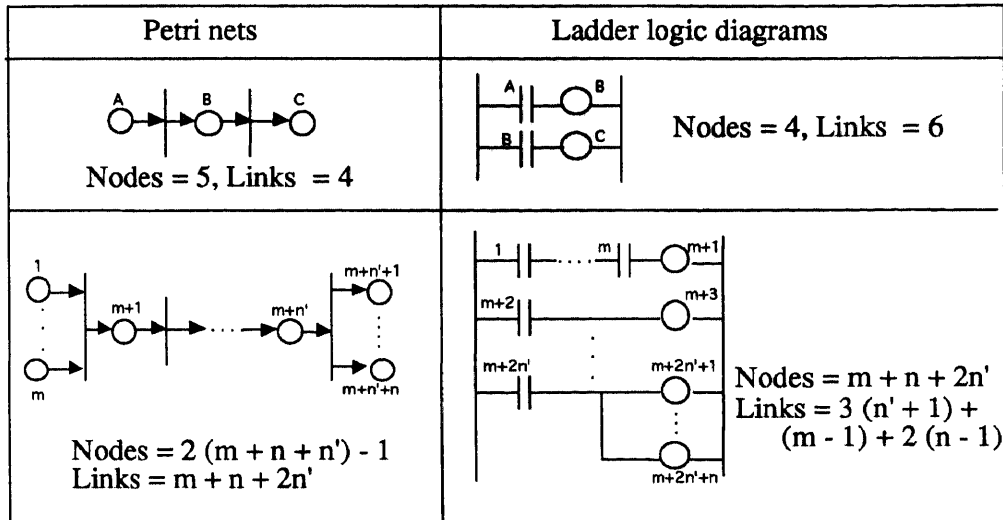


Figure 7.8 PNs and LLDs for sequential modeling

For PN: $\alpha = 3(m + n) + 4n' - 1$, where

$$\alpha_n = 2(m + n + n') - 1, \text{ and}$$

$$\alpha_l = m + n + 2n'.$$

Here, n' represents the number of sequential conditions.

For LLD: $\beta = 2m + 3n + 5n'$, where

$$\beta_n = m + n + 2n', \text{ and}$$

$$\beta_l = 3(n' + 1) + (m - 1) + 2(n - 1)$$

$$\Delta = f(m, n, n') = m - n' - 1.$$

7.4.2. Timed logical AND, Timed logical OR, and Timed sequential model Timed logical AND

In this model delays are associated with operations of the logical AND considered before.

Figure 7.9 shows the PNs and LLDs to derive the analytical formulas to obtain α and β for timed logical AND. They are given as follows:

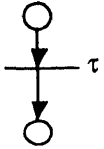
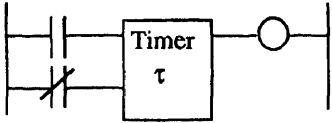
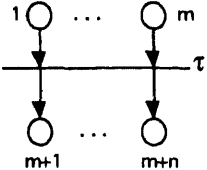
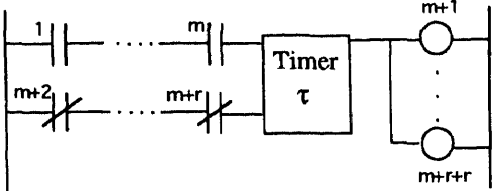
Petri nets	Ladder logic diagrams
 <p>Nodes = 3 Links = 2</p>	 <p>Nodes = 4 Links = 6</p>
 <p>Nodes = $m + n + 1$ Links = $m + n$</p>	 <p>Nodes = $m + n + r + 1$ Links = $m + 2n + r + 2$</p>

Figure 7.9 PNs and LLDs for timed logical AND

For PN: $\alpha = 2(m + n) + 1$, where

$$\alpha_n = m + n + 1, \text{ and}$$

$$\alpha_l = m + n$$

For LLD: $\beta = 2m + 3n + 2r + 3$, where

$$\beta_n = m + n + r + 1, \text{ and}$$

$$\beta_l = m + 2n + r + 2$$

$\Delta = -n - 2r - 2$. This indicates that PN is always preferred to model timed logical AND.

Timed Logical OR

In this model it is assumed that for each timer there is only one reset signal. Figure 7.10 shows the PNs and LLDs to derive the analytical formulas to obtain α and β for timed logical OR.

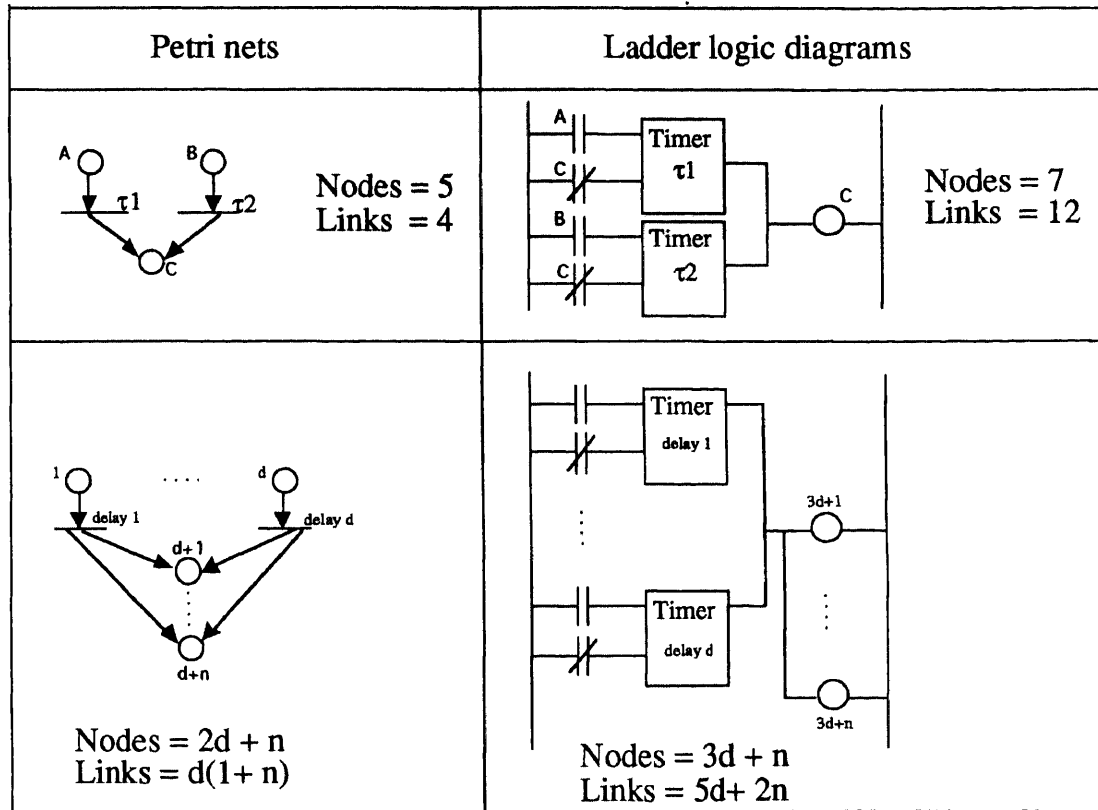


Figure 7.10 PNs and LLDs for timed logical OR

α and β are given as follows:

For PN: $\alpha = 3d + n + dn$, where

$$\alpha_n = 2d + n, \text{ and}$$

$$\alpha_l = d(1 + n).$$

In this chapter d represents the number of delays in the sequence.

For LLD: $\beta = 8d + 3n$, where

$$\beta_n = 3d + n, \text{ and}$$

$$\beta_l = 5d + 2n$$

$$\Delta = f(d,n) = -5d - 2n + dn.$$

This indicates that for d and n values which satisfy $5d + 2n > dn$, PN is preferred. For d and n values that do not satisfy the above condition, LLD is preferred. For example, for the pairs, $d = 3, n = 4$; and $d = 4, n = 3$, PN is preferred.

On the other hand for $d = 7, n = 8$, LLD is preferred. Similar analysis can be performed for any given d and n values using Δ function.

Timed Sequential Modeling

In this model it is assumed that for each timer there is only one reset signal. Figure 7.11 shows the PNs and LLDs to derive the analytical formulas to obtain α and β for this case.

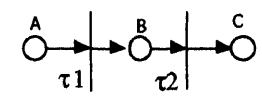
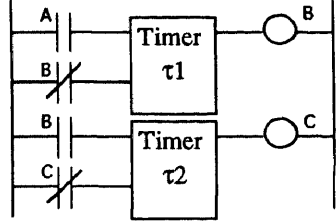
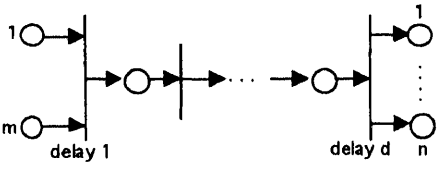
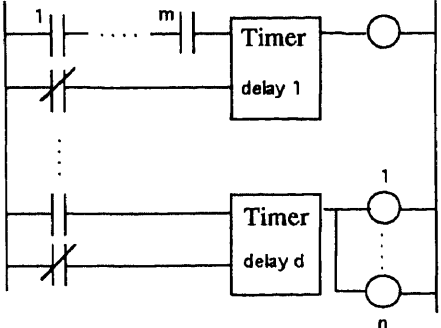
Petri nets	Ladder logic diagrams
 <p>$\tau_i = \text{time delay } i$ Nodes = 5, Links = 4</p>	 <p>Nodes = 8 Links = 12 $x_i = \text{Enabling condition for timer } i$</p>
 <p>Nodes = $2d + m + n - 1$ Links = $2d + m + n - 2$</p>	 <p>Nodes = $4d + m + n - 2$ Links = $6d + m + 2n - 3$</p>

Figure 7.11 PNs and LLDs for timed sequential modeling

α and β are given as follows:

For PN: $\alpha = 4d + 2(m + n) - 3$, where

$$\alpha_n = 2d + m + n - 1, \text{ and}$$

$$\alpha_l = 2d + m + n - 2.$$

For LLD: $\beta = 10d + 2m + 3n - 5$, where

$$\beta_n = 4d + m + n - 2 \text{ and}$$

$$\beta_l = 6d + m + 2n - 3$$

$$\Delta = f(d,m,n) = -6d - n + 2$$

Δ indicates that PN is always better than LLD.

7.4.3. Other Formulas for Estimating Basic Elements in PN and LLD

The models presented till now are common for both PNs and LLDs. However, there are certain models that are specific to PN or LLD. For example, the implementation of timer and counter is similar in LLDs. In PNs, timer is implemented by associating delays to certain transitions and hence the implementation of timer and counter are not the same. Hence, there is a need for separate formulas to estimate the number of basic elements to model a counter in case of a PN.

Also, to control certain systems, relays have to be used in LLDs. This is explained here. A discrete event system can be controlled by controlling two types of elements: those that require sustained actuating signals, and those that need only a momentary or pulsed actuating signal (Pessen 1989b). The first type of elements can be exemplified by an on-off solenoid valve with return spring and shown in Fig. 7.12 (a). As long as the solenoid is actuated, the valve remains open (or closed, as the case may be). As soon as the solenoid is released, the return spring returns the valve to its original position. Hence, the solenoid needs a sustained actuating signal to keep the valve open. This requires the use of relays in LLDs. The second type of elements can be exemplified by a solenoid valve with two opposing solenoids but without a return spring and is shown in Fig. 7.12 (b).

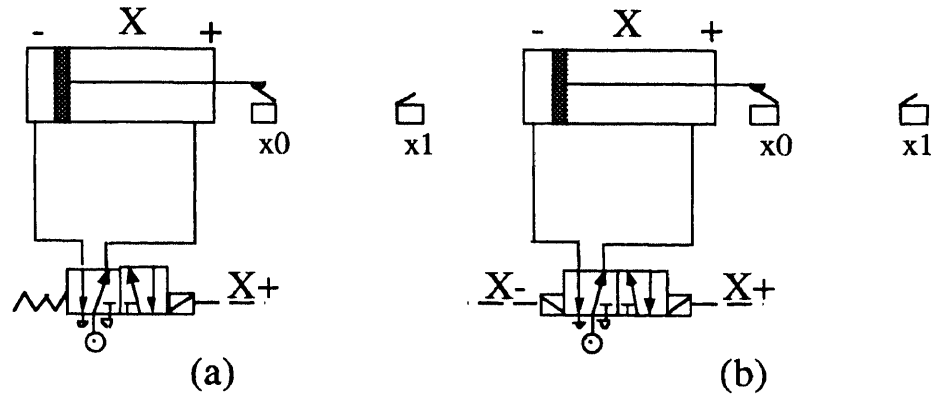


Figure 7.12 Typical cylinder-actuating circuit (Pessen 1989b)
 (a). requiring sustained solenoid signals
 (b). not requiring sustained solenoid signals

Here, a momentary solenoid signal is sufficient to shift the valve into its other position, and the valve will remain there until the opposing solenoid is actuated. It is strictly not allowed to actuate both solenoids at the same time. Actuating both solenoids simultaneously results in an undefined state and termed as 'interlock' as the two solenoids fight each other causing heat up and burn out. Using a single PN model, both types of systems in Fig. 7.12 can be controlled. This can be accomplished by changing the corresponding attributes for transitions as reported in PN based control using Real-time PNs. However, when using ladder logic diagrams, two separate design procedures have to be followed to design two separate diagrams as shown in (Pessen 1989b). The formulas presented earlier are relatively simple compared to the ones described below.

Emergency stop modeling

Emergency stop can be implemented in two different ways. In the first, when the push button corresponding to emergency stop is pushed, all the operations that are in progress and those that are ready to start are immediately stopped. In other words, the pistons executing the operations are stopped at the position where they are. In the second implementation, when the push button is pushed, the system is brought to initial condition. That is, all the pistons that are executing operations are retracted. The second method of

implementing emergency stop is very common and hence it is implemented and studied in this work. Denote the number of basic elements needed to model emergency stop (ES) modeling by PN and LLD as α_{ES} and β_{ES} respectively. In a PN, emergency stop is modeled by a place and inhibitory arcs from this place to transitions that model execution of actions. Fig. 13 shows the implementation of ES using PN and LLD.

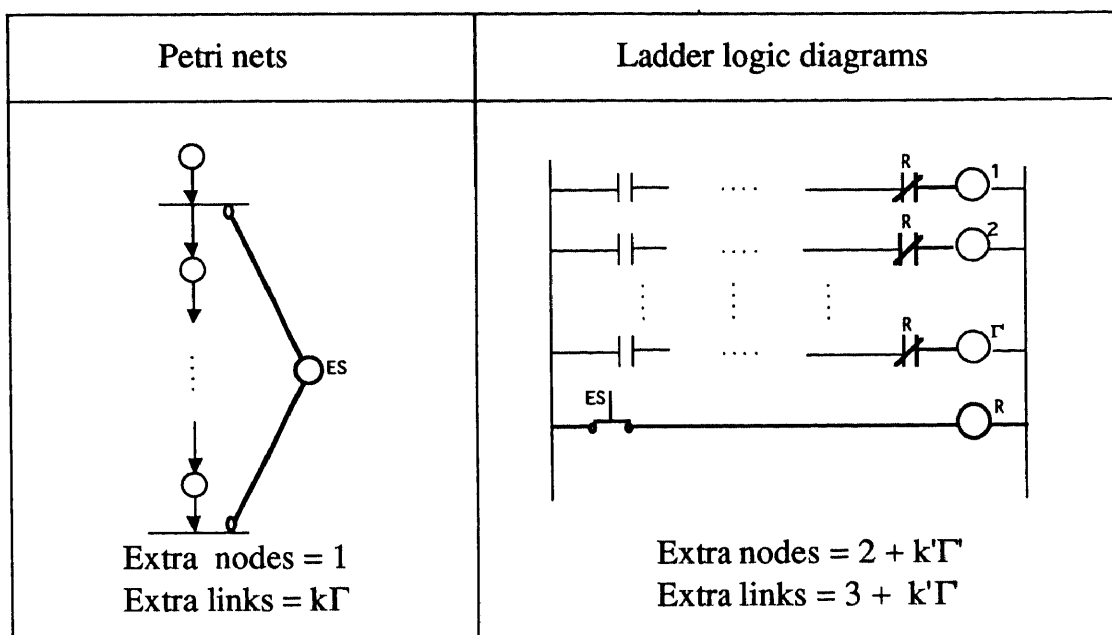


Figure 7.13 PNs and LLDs modeling emergency stop

Given any sequence, the additional basic elements needed are shown with bold lines. α_{ES} is given as $\alpha_{ES} = 1 + k\Gamma$ where $k = 1$ if each action is modeled by one transition, and $k = 2$ if each action is modeled by two transitions (one to represent start and second to represent end) and Γ stands for number of actions in a given specification. β_{ES} is given as $\beta_{ES} = 5 + 2k'\Gamma$ where $k' = 1$ if each action is executed by sustained actuating signal, and $k' = 2$ if each action is executed by a pulsed actuating signal and Γ stands for number of pistons in a given specification.

Basic elements needed to model a counter

Irrespective of the given sequence, the implementation of a counter is unique as shown in Fig. 7.14.

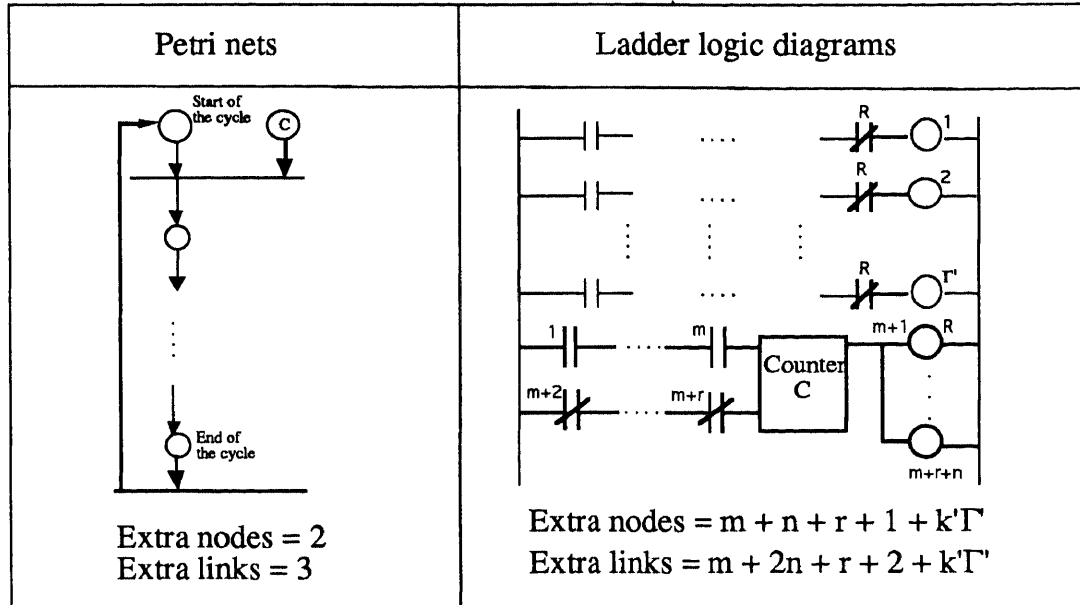


Figure 7.14 PNs and LLDs modeling counter

In Fig. 7.14, given any sequence, the additional basic elements to implement a counter are shown with bold lines. As it can be observed from this PN model three new arcs and two new nodes are needed to incorporate a counter in a sequence. Hence, the number of basic elements needed to model a counter using PNs is five. In LLDs, even though the implementation of timer and counter are similar, the output of counter may have to be used as a precondition to execute certain actions. This is usually done by adding the output of counter as a precondition for certain actions with normally closed contact. This results to include an extra node and a link for each action. Hence, The number of basic elements to implement counter in LLD is given as $\beta = (2m + 3n + 2r + 2k'\Gamma' + 3)$ where $r = \text{number of reset signals}$, $k' = 1$, if each action is executed by sustained actuating signal

and $k' = 2$, if each action is executed by a pulsed actuating signal and Γ stands for number of pistons in a given specification.

Basic elements needed to model a relay

When LLDs are used to control a system with sustained actuating signals, relays are needed to implement the sustained actuating signals. Fig. 7.15 shows the implementation of a relay. It can be observed that the number of basic elements needed to model a relay as $2m + 2r + 6$ where $r = \text{number of reset signals}$.

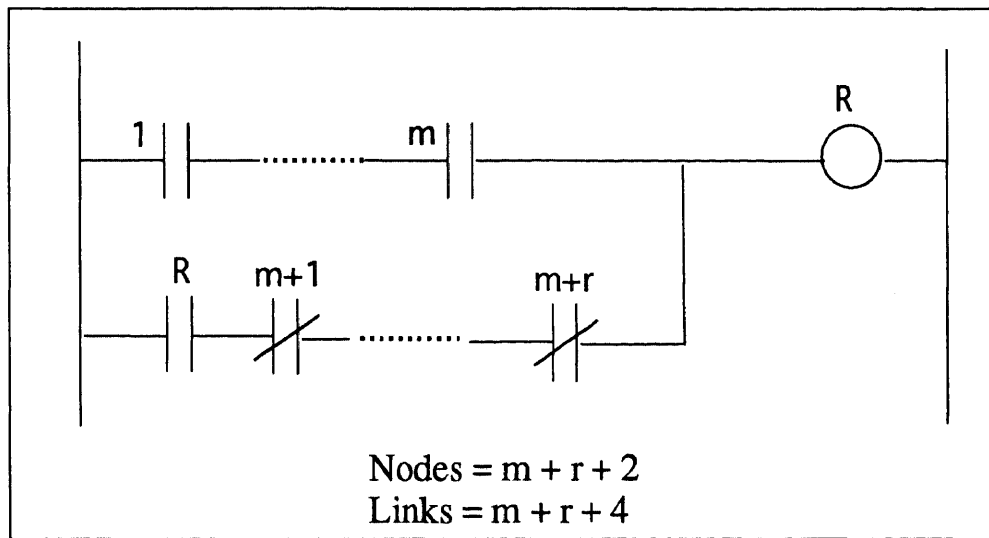


Figure 7.15 LLD modeling a relay

7.5. Methodology to use the Analytical Formulas

Figure 7.16 illustrates the method to obtain the total number of basic elements to model a given control logic using the analytical formulas.

Most of the control logics can be represented using the basic building blocks of logic constructs described in the earlier section. Hence, in order to find the basic elements to model a given control logic using the models developed, first it has to be decomposed into the logic constructs or segments. Then for each segment, the corresponding analytical formula is applied to find the number of basic elements.

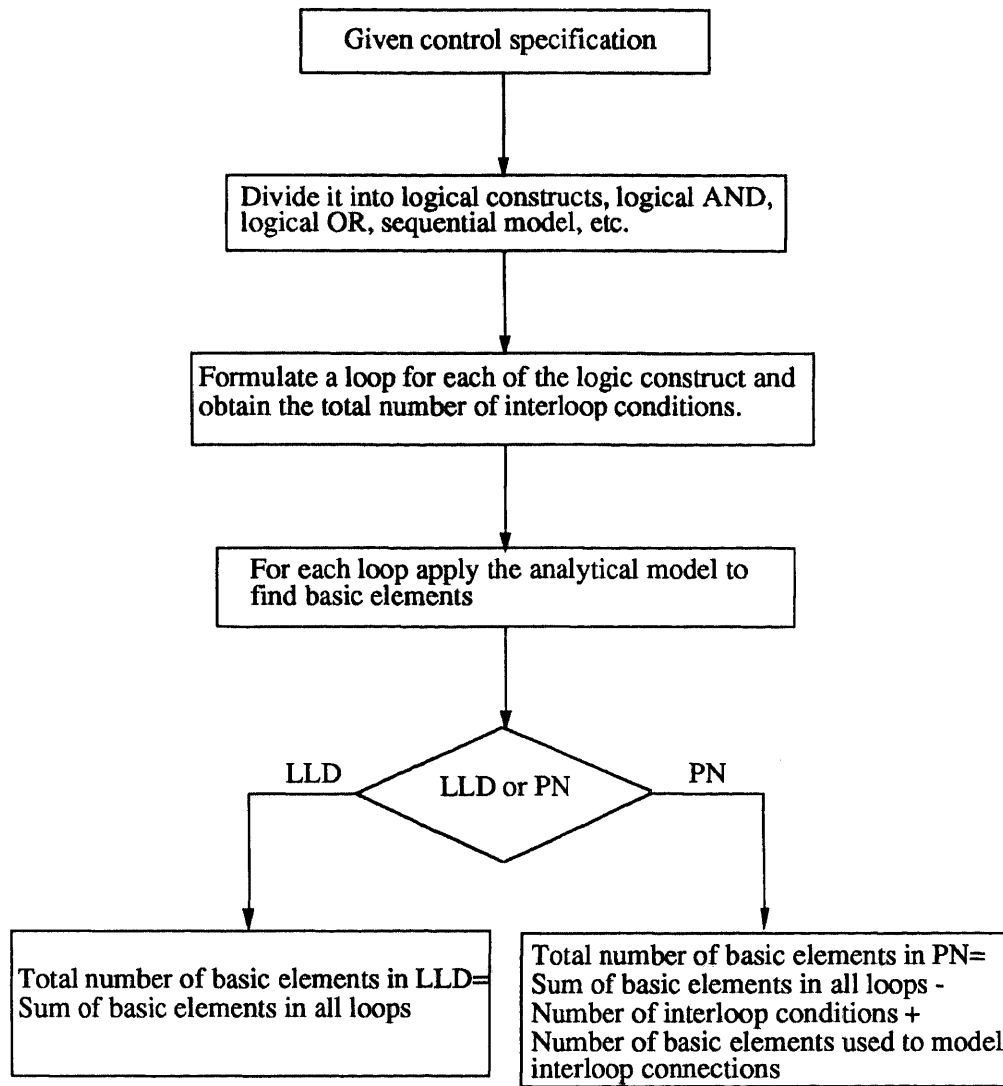


Figure 7.16 Method to estimate basic elements

The basic elements corresponding to all such segments are then added to obtain the total number of basic elements (β) needed to model the given control logic using a LLD.

$$\beta = \sum_{i=1}^k \beta_i$$

where β = Total number of basic elements to model the control logic using PN and β_i = Number of basic elements in segment i , and k = Total number of segments.

The final step (as shown in Fig. 7.16) is slightly different in the case of PN due to its physical model. This is because when modeling a control logic, intersegment conditions and connections exist in a PN model. They are explained below. For some cases in the given control logic, the output condition for segment i may be one of the input conditions for segment j . These conditions are called as intersegment conditions. In LLD, such conditions are separately modeled in both segments i and j . Even though such conditions appear in more than one segment, the place modeling such condition physically appears only once in the PN. Therefore, in case of PN to account for the repetitive count of these intersegment conditions, the total number of such conditions has to be subtracted from the summation of basic elements in all segments.

For some control specifications it may possible that an output condition of segment i causes an action which produces an input condition(s) for segment j . That means intersegment connections exist between segments i and j . Basically, intersegment connections model the power/control flow from one segment to another. In LLD these connections need not be explicitly modeled since the vertical line at the left hand side of LLD always models the existence of electric power. Depending upon the logic of each rung, the current flows from this power line and energizes the output of rung connected to the right hand side line in LLD. However, in case of PN the power flow (called as control flow in PN terminology) across segments is modeled by the movement of tokens in PN. Since, transition firings cause the flow of tokens through arcs, the intersegment connections have to be explicitly modeled. The total number of basic elements to model intersegment connections is estimated as shown below.

Let θ_{ij} = The number of basic elements used to model intersegment connections between segments i and j

$$\theta_{ij} = \theta_{ijn} + \theta_{ijl}$$

θ_{ijn} = Nodes (typically transitions) used to model intersegment connections between segments i and j

θ_{ijl} = Links used to model intersegment connections between segments i and j , then
 $\theta_{ijn} = q$, where q represents the total number of output conditions in segment i producing inputs for segment j .

Sometimes, it may happen that an action corresponding to the output condition in segment i produces several inputs for segment j . Considering this, q_{ijl} is given as follows:

$$\theta_{ijl} = q + \sum_{r=1}^q w_r$$

where w_r represents the number of inputs for segment j produced by output condition r in the segment i . The first term in θ_{ijl} corresponds to the output arcs from the places modeling output conditions in segment i and the second term corresponds to the input arcs to the places modeling input conditions in segment j . Hence, θ_{ij} is given as:

$$\theta_{ij} = 2q + \sum_{r=1}^q w_r$$

The total number of basic elements modeling intersegment connections in the whole control logic is then given as follows:

$$\theta = \sum_{i=j}^k \theta_{ij}$$

Considering intersegment connections, in case of PN, the total number of basic elements used to model the total number of intersegment conditions have to be added for the summation of basic elements in all segments. Hence, considering both intersegment conditions and connections, the total number of basic elements needed to model the control logic using PN is termed as α and given as follows:

$$\alpha = \sum_{i=1}^k \alpha_i - \phi + \theta$$

where, α_i = Number of basic elements in segment i ,

k = Total number of segments,

ϕ = Total number of intersegment conditions,

θ = Total number of basic elements needed to model intersegment connections.

Once, α and β are known, we can select PN or LLD based on $\Delta = \alpha - \beta$. If Δ is positive LLD is preferred, otherwise, PN is preferred from the design complexity view point.

7.6. Illustration of the Methodology Through Examples

In order to make the application of the methodology more clear this section presents three examples. In the first example, the PN has intersegment conditions without intersegment connections. In the second example, the PN has intersegment connections without intersegment conditions. Also, to control this system using LLDs sustained signals are not required. Finally, the electro-pneumatic system earlier considered in Section 3 is dealt in Example 3. The PN models of this system have both intersegment connections and intersegment conditions. Also, to control this system using LLDs sustained signals are needed. The systems considered in these examples are very common in flexible automation.

7.6.1. An automatic assembly system

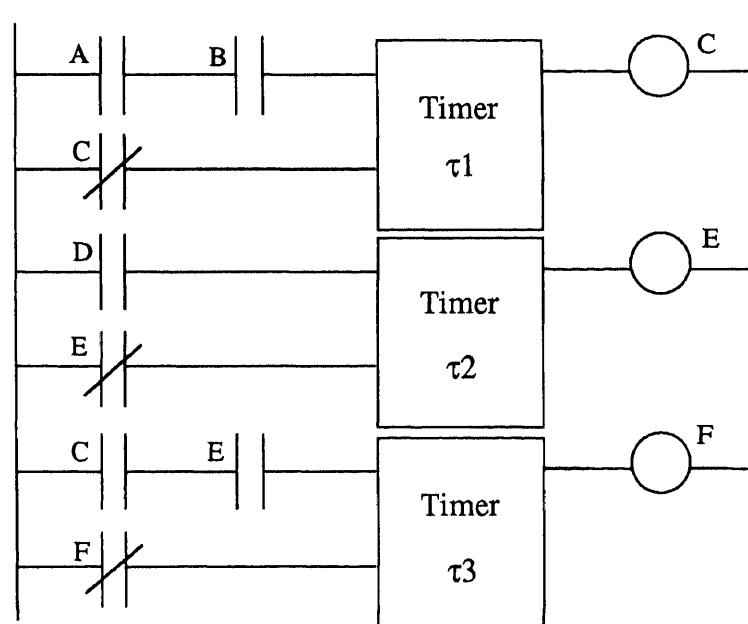
Consider the automatic assembly system shown in Fig. 6.3. The functions of the elements present in the system have to be synchronized by a control logic. With an objective to use smaller control program, we need to find which one between PN and LLD is better to model this control logic. The methodology described in the above section is used to estimate the number of basic elements needed for PN and LLD. Table 7.3 shows the results obtained.

Here, $\phi = 2$, since C and E are intersegment conditions, and $\theta = 0$ since there are no intersegment connections. Hence, $\alpha = 7 + 5 + 7 - 2 = 17$; $\beta = 12 + 10 + 12 = 34$, and $\Delta = \alpha - \beta = -17$. This indicates for this system, PN gives 50 % shorter model than LLD. This can also be validated by manually counting the basic elements after formulating the physical models of PN and LLD.

Table 7.3 Required basic elements to control the system in Fig. 6.3

Segment	Description of logic construct	Models applied	α_i	β_i
1	A.B, $\tau_1 \rightarrow C$	Timed logical AND $m = 2, n = 1, r = 1$	$2m+2n+1 = 7$	$2m+3n+2r+3 = 12$
2	D, $\tau_2 \rightarrow E$	Timed logical AND $m = 1, n = 1, r = 1$	$2m+2n+1 = 5$	$2m+3n+2r+3 = 10$
3	C.E, $\tau_3 \rightarrow F$	Timed logical AND $m = 2, n = 1, r = 1$	$2m+2n+1 = 7$	$2m+3n+2r+3 = 12$

The PN and LLD to control the system is shown in Figs. 6.4 and Fig. 7.22 respectively. It can be easily observed that PN and LLD uses 17 and 34 total basic elements to model the control logic.

**Figure 7.17** LLD model for the system shown in Fig. 6.3

7.6.2. An Electro-Pneumatic System Without Sustained Signals

Now, consider another system where intersegment connections exist in the control specification. Low cost automated systems with hydraulic/pneumatic pistons actuated by solenoids exhibit such type of intersegment connections. This is illustrated in this example.

The system consists of four pneumatic pistons (A, B, C, and D) which are to be sequenced by double activated five ports and two-way solenoid valves. Each piston has two normally open limit switches. Also, consider that all pistons are of the type shown in Fig. 7.12 (b), which require momentary actuating signals. When the end of piston X contacts limit switch x1 (x0), x1 (x0) is closed indicating that the piston X is at the end of its forward stroke (return stroke). A push button, START is provided to start the system. It is given that pistons A, B, C, and D have to be sequenced according to the following sequence: START, {A+, B+}, {A-, D+}, {B-, D-, C+}, C-.

The methodology earlier described is followed to estimate the number of basic elements needed for PN and LLD. Table 7.4 shows the results obtained.

Table 7.4 Required basic elements to control the system in Example 2.

Segment	Description of logic construct	Models applied	α_i	β_i
1	START.c- --> {A+,B+}	Logical AND m = 2, n = 2	$2m+2n+1 = 9$	$2m+3n = 10$
2	a+.b+ --> {A-,D+}	Logical AND m = 2, n = 2	$2m+2n+1 = 9$	$2m+3n = 10$
3	a-.d+ --> {B-,D-,C+}	Logical AND m = 2, n = 3	$2m+2n+1 = 11$	$2m+3n = 13$
4	b-.d-.c+ --> {C-}	Logical AND m = 3, n = 1	$2m+2n+1 = 9$	$2m+3n = 9$

Here, $\phi = 0$, since there are no intersegment conditions. θ is calculated as follows:

Upon observation it can be seen that intersegment connections exist between Segments 1 and 2; 2 and 3; 3 and 4; and 4 and 1. For example, A+ and B+ in Segment 1 results in a+ and b+ in Segment 2. Therefore, for Segments 1 and 2, $q = 2$. Assuming A+ is action 1 and B+ as 2, $\theta_{12} = 2q + w_1 + w_2 = 2.2 + 1 + 1 = 6$. Similarly, for Segments 2 and 3, $\theta_{23} = 6$. For Segments 3 and 4 assuming B-, D-, C+ as first, second, and third actions, $\theta_{34} = 2q + w_1 + w_2 + w_3 = 2.3 + 1 + 1 + 1 = 9$. For Segments 4 and 1, $\theta_{41} = 2q + w_1 = 2.1 + 1 = 3$. Hence, $\theta = \theta_{12} + \theta_{23} + \theta_{34} + \theta_{41} = 6 + 6 + 9 + 3 = 24$.

Hence, α , β , and Δ are given as follows:

$\alpha = (9 + 9 + 11 + 9) + 24 = 62$, $\beta = 10 + 10 + 13 + 9 = 42$, and $\Delta = \alpha - \beta = 20$. This indicates for the example system considered, LLD gives 32.25 % shorter program than PN. This can also be validated by manually counting the basic elements after formulating the physical models of PN and LLD. The PN and LLD for to control the system is given Fig. 7.18 (a) and (b) respectively.

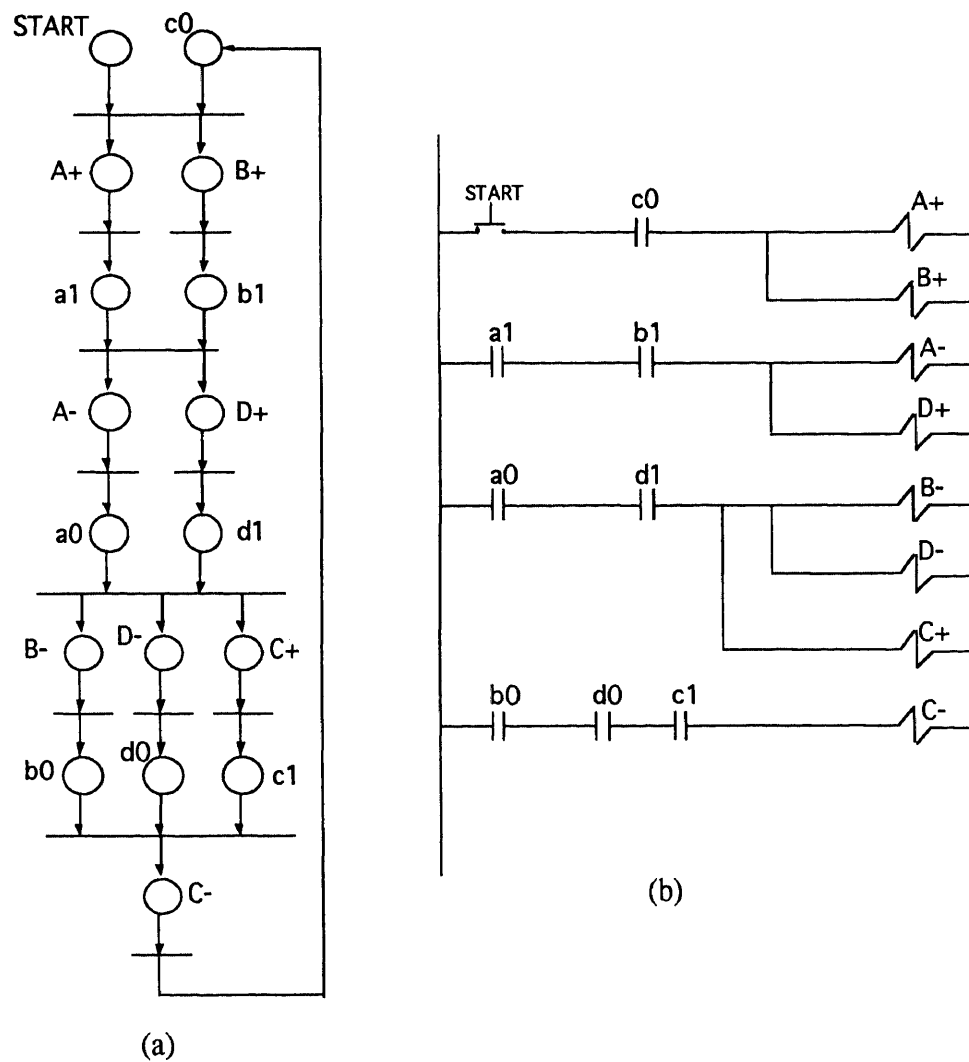


Figure 7.18 (a) PN and (b) LLD for sequence: {A+, B+}, {A-, D+}, {B-, D-, C+}, C

After counting the basic elements, it can be easily observed that PN and LLD uses 62 and 42 total basic elements to model the control logic. This indicates that when the system contains only pneumatic pistons that do not require sustained signals for solenoids, LLD would yield a shorter program than PN.

However, if time delays are added in the above sequence, PN structure will not be changed as time can be modeled as an attribute to a transition. On the other hand the length of LLD will be increased to incorporate the timers.

7.6.3. An Electro-Pneumatic System with Sustained Signals

The electro-pneumatic system considered in Section 3 is considered here.

The number of basic elements in PNs and LLDs is estimated instead of manual counting in Section 3.

Estimation of the number of basic elements in PNs

Table 7.5 presents the calculations to estimate the number of basic elements in the PN modeling Sequence 1.

Table 7.5 Required basic elements to model the Sequence 1 using PN.

(a) Basic elements in each segment without considering θ and ϕ

Segment	Description of logic construct	Model to be applied	α_i
1	ST- --> a0, b0, c0	Logical AND m = 1, n = 3	$2m+2n+1 = 9$
2	a0 --> A+	Logical AND m = 1, n = 1	$2m+2n+1 = 5$
3	a1.b0 --> B+	Logical AND m = 2, n = 1	$2m+2n+1 = 7$
4	b1.c0 --> {C+, A-}	Logical AND m = 2, n = 1	$2m+2n+1 = 7$
5	c1.a0 --> {B-, C-}	Logical AND m = 2, n = 1	$2m+2n+1 = 7$
6	{B-, C}- --> End	Logical AND m = 1, n = 1	$2m+2n+1 = 5$

(b) Basic elements representing intersegment conditions

Segments j and k	Intersegment condition	ϕ_{jk}
1 and 2	a0	1
1 and 3	b0	1
1 and 4	c0	1
1 and 5	a0	1
5 and 6	{B-, C-}	1

(c) Basic elements representing intersegment connections

Segments j and k	Intersegment connection	Values of variables	θ_{jk}
2 and 3	A+ --> a1	$q = 1, w_1 = 1$	$2q + w_1 = 3$
3 and 4	B+ --> b1	$q = 1, w_1 = 1$	$2q + w_1 = 3$
4 and 5	C+, A- --> c1, a0	$q = 1, w_1 = 2$	$2q + w_1 + w_2 = 4$
5 and 3	B- --> b0	$q = 1, w_1 = 1$	$2q + w_1 = 3$
5 and 4	C- --> c0	$q = 1, w_1 = 1$	$2q + w_1 = 3$

In calculation, observe that x_0 is a precondition to execute action $X+$ where x and X stands for corresponding limit switch and piston identity. For simplicity, denote the number of basic elements in PN corresponding to Sequence i as α_{s_i} . Now, following the formulas presented earlier, α_{s_i} can be estimated as follows:

$$\alpha_{s_1} = (9+5+7+7+7+5) - (1+1+1+1+1) + (3+3+4+3+3) = 40 - 5 + 16 = 51.$$

Observe that a_0 is presented as a precondition in both Segments 2 and 5. However, in PN each limit switch appears physically once (with several input and output arcs). Hence, to compensate this the previous value of α_{s_1} has to be subtracted by one. Then, the final value of is 50 which exactly matches with the value obtained in Section 3.

Sequence 2 is obtained by adding a counter and emergency stop to Sequence 1. Hence, α_{s_2} can be obtained by adding the basic elements needed to model a counter (α_C) and emergency stop (α_{ES}) to α_{s_1} . Then, $\alpha_{s_2} = \alpha_{s_1} + \alpha_C + \alpha_{ES}$. By following the models presented earlier, $\alpha_C = 5$. The number of basic elements to model ES can be calculated using the earlier developed formula. Since there are four actions (recall that

{C+,A-} and {B-,C-} are treated as two concurrent actions) present in the sequence and each action is modeled by two transitions (one to start and second to end), $\alpha_{ES} = 1 + k\Gamma = 1 + 2.4 = 3$. Therefore, $\alpha_{s2} = 58+5+9 = 64$ as obtained earlier in Section 4.

Sequence 3 is obtained by adding delay before {B-,C-} of Sequence 2. However, in PNs this delay is implemented by changing the second attribute of transition modeling 'do {C+,A-}'. Hence, there is no need for extra basic elements. In other words, $\alpha_{s3} = \alpha_{s2} = 64$ as given in Section 4. Sequence 4 is obtained by adding a counter and timer to Sequence 3. Again, we know that $\alpha_C = 5$, and there is no need for extra basic elements to model timer. Hence, $\alpha_{s4} = \alpha_{s3} + \alpha_C = 64 + 5 = 69$ as obtained in Section 4.

Estimation of the number of basic elements in LLDs

When sustained actuating signals are used, the following simple three step procedure is developed to estimate β .

1. Assign a relay for each segment in the given sequence except the segment containing 'Start' command,
2. Identify set and reset signals for each sustained signal implemented by relay. Note that, setting (resetting) of a relay executes forward (return) stroke of a piston,
- 3: Treat set signals for relay as preconditions, and apply the formulas to calculate the number of basic elements.

For the case study considered in Section 4, the results obtained after each step by following the above procedure are as follows:

After Step 1: Relay 1 (R1), R2, R3, and R4 are assigned to A+,B+,{C+,A-}, and {B,C, respectively.

After Step 2: Set signals for the sustained signal by R1 are ST, a0, b0, c0, R1. Since A- is in Segment 3, R3 is a reset signal for R1. Since R2 has to wait till R1 completes all of its actions, set signals corresponding to R2 are a1 and R2 and reset signal is R4 because B- is in Segment 4. Similarly set signals for

R3 are b1 and R3 and reset signal is R4 because C- is in Segment 4. Set signals for R4 are c1 and a0.

After Step 3: Table 7.6 shows the calculations to estimate β .

Table 7.6 Required basic elements to model the Sequence 1 using LLD.

Segment	Description of logic construct	Model to be applied	β
1(a)	ST.a0.b0.c0.R3 --> R1	Relay $m = 4, r = 1$	$2m+2r+6 = 16$
1(b)	R1 --> A+	Logical AND $m = 1, n = 1$	$2m+3n = 5$
2(a)	a1.R4 --> R2	Relay $m = 1, r = 1$	$2m+2r+6 = 10$
2(b)	R2 --> B+	Logical AND $m = 1, n = 1$	$2m+3n = 5$
3(a)	b1.R4 --> R3	Relay $m = 1, r = 1$	$2m+2r+6 = 10$
3(b)	R3 --> C+	Logical AND $m = 1, n = 1$	$2m+3n = 5$
4	c1.a0 --> R4	Logical AND $m = 2, n = 1$	$2m+3n = 7$

From the table, $\beta_{S1} = 16+5+10+5+10+5+7 = 58$. In Sequence 2 counter and emergency stop are added to Sequence 1. Denote β_C and β_{ES} the number of basic elements needed to model a counter and emergency stop, respectively. $\beta_{S2} = \beta_{S1} + \beta_C + \beta_{ES}$. β_C and β_{ES} are calculated by using the derived formulas as follows:

m is 2 because b0,c0 resulting from executing the last action in the given sequence are to be taken as set signals for the counter. n is 1 because there is only one output from the counter. r is 1 because start switch ST can be taken as a reset signal for the counter. k' is 1 because in the system under study each action that drives the piston forward is executed by sustained actuating signal and Γ' is 3 because three pistons are present. Hence, $\beta_C = 2m + 3n + 2r + 2k'\Gamma' + 3 = (2.2 + 3.1 + 2.1 + 2.1.3 + 3) = 18$. $\beta_{ES} = 5 + 2k'\Gamma' = 5 + (2)(1)(3) = 11$. Hence, $\beta_{S2} = 58 + 18 + 11 = 87$ which is exactly the same as obtained in Section 4. Sequence 3 is obtained by including a delay after {C+, A-} in

Sequence 2. Hence, the timer is to be set with c1 and a0 as input signals and b0 and c0 as reset signals. There is only one output from timer. Hence, $m = 2$, $n = 1$, and $r = 2$. The number of basic elements needed to implement the timer, $\beta_T = 2m + 3n + 2r + 3 = 2 \cdot 2 + 3 \cdot 1 + 2 \cdot 2 + 3 = 14$. Segment 4 in Table 7 already contains the elements and links corresponding to c1, a0, and R4. Hence, to compensate these, $\beta_{S3} = \beta_{S2} + \beta_T - \beta_4$ where $\beta_4 = 7$ denotes the number of basic elements in Segment 4. Therefore, $\beta_{S3} = 87 + 14 - 7 = 94$ as obtained in Section 4.

Since Sequence 4 is obtained by adding a counter and delay to Sequence 3, it can be implemented by adding a counter and timer to the LLD of Sequence 3. Even though the number and identity of set and reset signals for the new counter and timer are different from those used in the earlier sequences, as an approximation we can still use the number of basic elements that were calculated earlier here. In other words, $\beta_{S4} = \beta_C + \beta_T$. Since, β_C and β_T are 18 and 14 respectively, $\beta_{S4} = 94 + 18 + 14 = 126$ which is very close to 130 as obtained in Section 4.

Even though we could accurately estimate the number of basic elements in the last case by accurately identifying the set and reset signals for timer and counter, it may take significant effort to analyze the logic. However, the motivation behind proposing the formulas presented here is to not to exactly calculate the number of basic elements to match that result from manually counting an LLD or PN, but to give an idea of how many basic elements are needed to model the control logic. Using the approach shown here these formulas can be incrementally applied even for complex sequences consisting of timers and counters. In other words, they are first applied to a simple sequence to estimate the number of basic elements. Then the number of basic elements needed for counters and timers is estimated and added to the previous value. Applying these formulas allows designers to select between systems that use sustained actuating signals and momentary/pulse actuating signals as well as PNs and LLDs. For example, for the system considered in Example 2, consider that the pistons are controlled by solenoids that require sustained actuating signals.

Then by applying the formulas as shown earlier, the number of basic elements in LLD can be found as 76. Recall that when momentary/pulse actuating signals are used, the basic elements in LLD are earlier calculated as 42. Also, recall that PN required 62 basic elements. Hence, by using the formulas and following the proposed methodology to estimate the basic elements, control engineers can select between PN and LLD even before they start to build the control model in detail.

7.7. Summary

Development of flexible, reusable, and maintainable control software is important to implement advanced industrial automated systems. Traditional methods of using ladder logic diagrams (LLDs) to design sequence controllers are being challenged by the needs in flexible and agile manufacturing systems. On the other hand, Petri nets (PNs) are an emerging tool that needs to be established for the control of discrete manufacturing systems. A class of PNs called real-time PNs that resemble ordinary PNs were introduced to design sequence controllers. This chapter identified design complexity and response time as the criteria to compare LLDs and PNs. Design complexity is defined and characterized by two factors namely graphical complexity and adaptability to meet changes in control specifications. By designing and implementing the control of an industrial automated system subject to changing control requirements, LLDs and PNs are compared in terms of a common measure namely, the numbers of basic elements, which signify both design complexity and response time.

Motivated by the fact that any sequence controller can be designed by synthesizing the building blocks of logic models, this chapter proposed analytical formulas to estimate the number of basic elements to model the most commonly used building blocks of logic modeling by both PN and LLD. Furthermore, a methodology that uses the developed analytical formulas to estimate the total number of basic elements to model a control logic even before physically modeling it using PNs and LLDs is presented. The concepts

developed in this chapter are demonstrated by considering several examples of sequence controllers. The examples considered here demonstrate the potential for practical application of the research results.

The methodology presented provides an accurate quantitative comparison of PN and LLD in terms of basic elements. By precluding the need for physically building the controllers by either PN or LLD, this methodology serves as an effective aid for a control engineer to select between PN and LLD even before starting to write the control program. The methodology developed is simple and straightforward to apply. However, in case of complex control specifications, decomposing the control specification in terms of the logic constructs may be difficult. This problem can be solved using the traditional methods such as Karnaugh maps, Huffman method, etc. Other factors which have impact on the selection of LLDs and PNs should also be explored in the future work. For example, similar to graphical complexity, irrespective of the implementation scheme an effort to quantify the response time complexity should be made.

CHAPTER 8

CONVERSION OF LOGIC CONTROL SPECIFICATIONS INTO PETRI NET MODELS

8.1 Introduction

In the last chapter, ladder logic diagrams (LLDs) are shown inefficient to develop control software and difficult to debug and maintain. Motivated by the disadvantages of LLDs, Petri nets (PNs) were demonstrated as an effective tool for logic controller design by several researchers and industrial practitioners (Valette *et al.* 1983, Murata *et al.* 1986, Ferrarini 1992, Zhou and DiCesare 1993, Venkatesh *et al.* 1993). One critical task in this development is to design Petri net models given the sequence control specifications. Development of such PN model is the first step to develop Real-time Petri net (RTPN) based controller. Also, one of the factors for the comprehensive comparison of PNs and ladder logic diagrams in discrete event control is the availability of standard procedures for designing controllers.

There exists systematic design procedures for designing ladder logic diagrams (Pessen 1989). However, for the large scale application of PNs in industry, there is a need for systematic design procedures for developing PN models.

The conversion procedure to formulate PN models from sequence control specifications is very straight forward and simple. This procedure is explained below and illustrated in this chapter.

1. Assign a place for each action in the given sequence,
2. Assign an input transition for each place to execute the corresponding action, and assign time delay, if any, as a second attribute to the transition,
3. Connect all the places according to the precedence/concurrency relations given in the specification via transitions,
4. For each place, assign an attribute corresponding to the limit switch that is triggered due to the completion of action modeled by that place. In other

words, following the notation given in Chapters 6 and 7, if a place p , models action $X+$, and the completion of this action is sensed by limit switch $x1$, then assign the attribute to p corresponding to $x1$,

5. For each transition, assign an attribute corresponding to the action that is to be executed after firing that transition, and
6. Model emergency stop and counters, if any.

8.2. Illustration of the Formulation of PN Model

A typical logic control specification usually consists of sequence of discrete events with/without delays in between delays (David 1992, Pessen 1989). In order to illustrate the above procedure to formulate PN model various types of sequences are considered. A logic specification with single-path sequence may or may not contain concurrent actions, but, do not contain choice actions. Specifications with multi-path sequences may or may not contain choice actions. Again, in certain logic specification, a sequence may or may not consist of repetitive actions. Hence, sequences in logic specifications can be broadly classified as four types. They are: 1) single path sequences with no repetitive actions, 2) single path sequences with repetitive actions, 3) multi path sequences with no repetitive actions, and 4) multi path sequences with repetitive actions.

The conversion procedure developed in this chapter can be applied to all four kinds of sequences. In the subsequent sections, the conversion procedure is applied to design PN models corresponding to the sequences reported in (Pessen 1989). These sequences represent the classical discrete event sequence control problems in manufacturing systems.

8.2.1. Single Path Sequences with no Repetitive Actions

The sequences considered in Chapter 7 can be used to illustrate the conversion procedure. Following this procedure, the PN model corresponding to the Sequence 1: Start, A+, B+, {C+, A-}, {B-, C-} is given in Fig. 8.1.

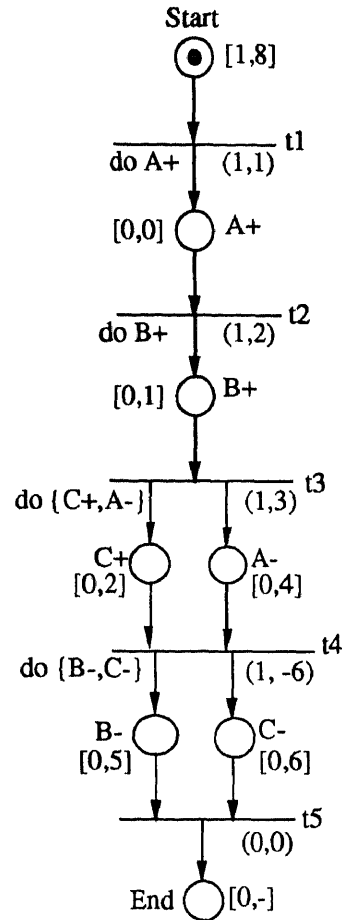


Figure 8.1 PN model for the Sequence 1

Observe that the PN model shown in Fig. 8.1 is far simpler and smaller than the corresponding one shown in Fig. 7.1. This is because, in Fig. 7.1, limit switches are separately modeled by places due to the following reasons: 1) to closely emulate the system physically, and 2) to follow the similar notion of ladder logic diagrams which clearly models the limit switches with normally open switches. In other words, the PN model shown in Fig. 7.1 clearly models the status of limit switches and actions separately. However, in the PN model generated by following the conversion procedure, places modeling limit switches are eliminated by associating the input signal information with places modeling corresponding actions.

This drastically reduces the net size and yields a compact PN model. Even though the places in this PN model do not explicitly distinguish between the actions and limit switches, it is simpler compared to the PN model presented in Chapter 7. Following the conversion procedure, PN models corresponding to the Sequences 2, 3, and 4 that are described in Chapter 7 are given in Figs. 8.2, 8.3, and 8.4, respectively. Observe that the inhibitory arc from the place modeling emergency stop is given only to transitions (t1, t2, t3, and t4) that model actions.

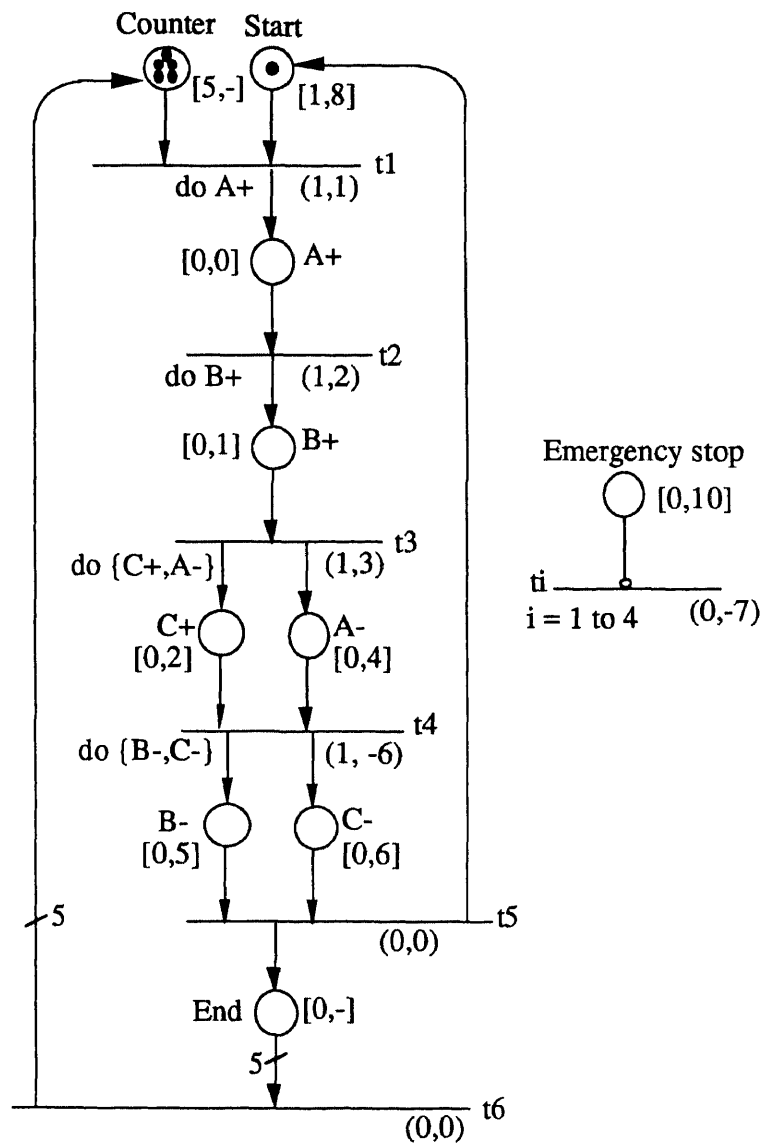


Figure 8.2 PN model for the Sequence 2

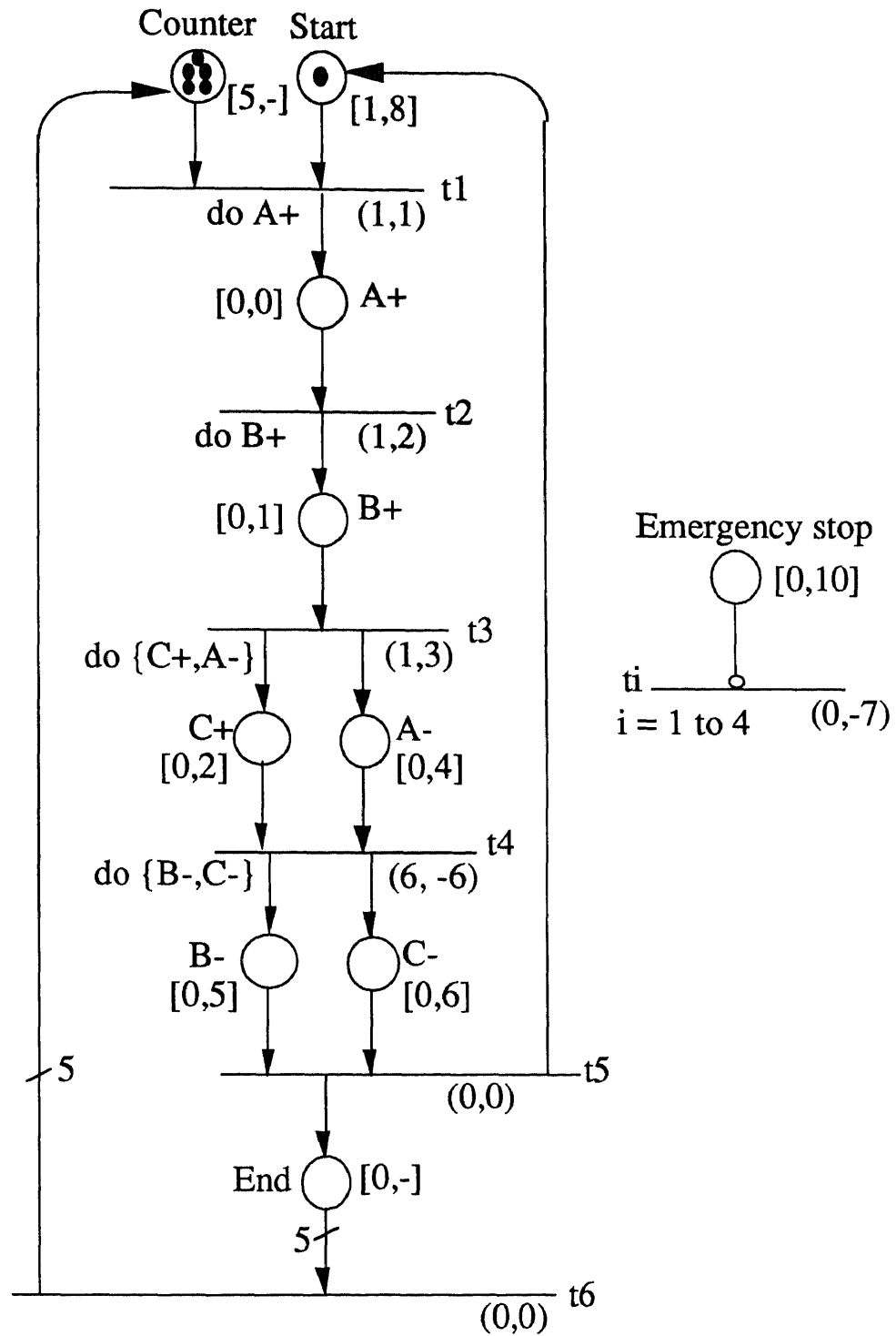


Figure 8.3 PN model for the Sequence 3

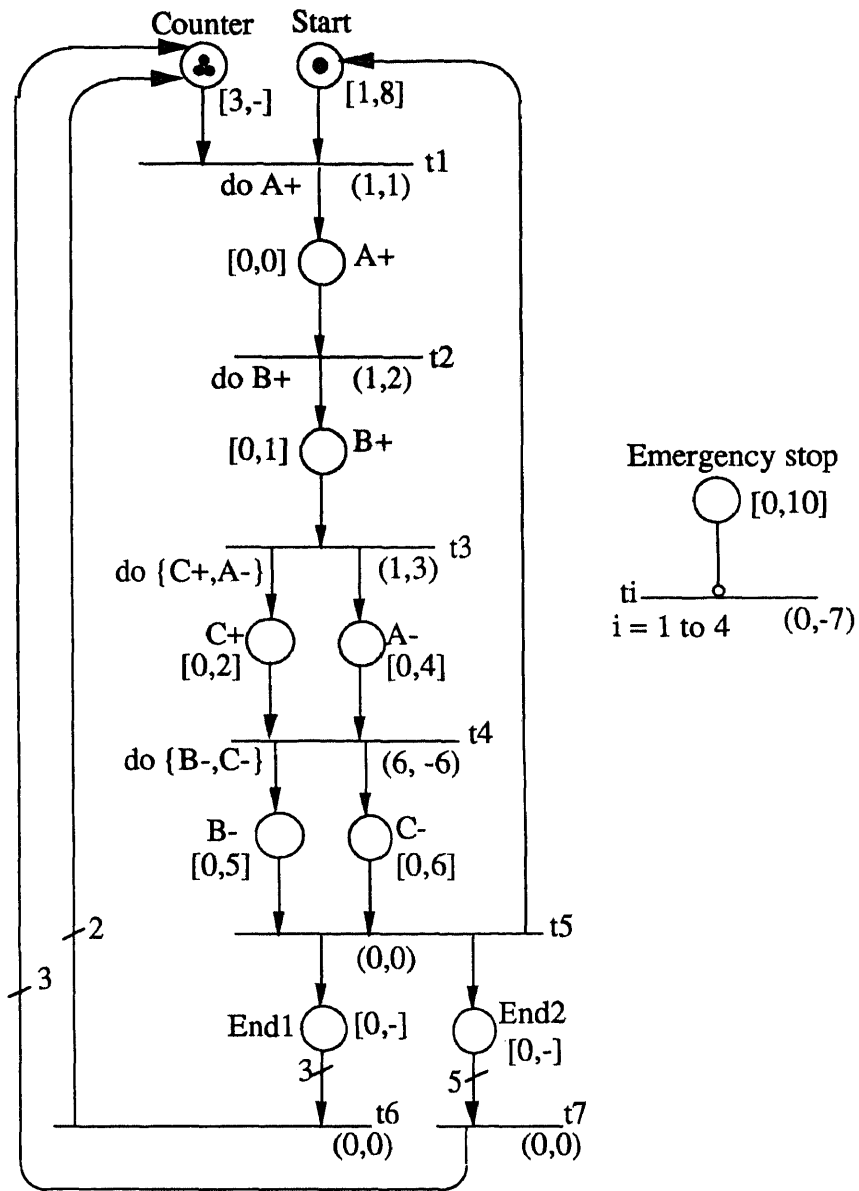


Figure 8.4 PN model for the Sequence 4

Estimation of basic elements in PN models

In order to compare the design complexity of the PN models obtained using the conversion procedure and the PN models presented in Chapter 7, the methodology presented in Chapter 7 is used to estimate the number of basic elements. Table 8.1 presents the calculations to estimate the number of basic elements in the PN modeling Sequence 1.

Table 8.1 Required basic elements to model the Sequence 1 using PN.(a) Basic elements in each segment without considering θ and ϕ

Segment	Description of logic construct	Model to be applied	α_i
1	START \rightarrow A+	Logical AND $m = 1, n = 1$	$2m+2n+1 = 5$
2	A+ \rightarrow B+	Logical AND $m = 1, n = 1$	$2m+2n+1 = 5$
3	B+ \rightarrow C+, A-	Logical AND $m = 1, n = 2$	$2m+2n+1 = 7$
4	C+, A- \rightarrow B-, C-	Logical AND $m = 2, n = 2$	$2m+2n+1 = 9$
5	{B-, C}- \rightarrow End	Logical AND $m = 2, n = 1$	$2m+2n+1 = 7$

(b) Basic elements representing intersegment connections

Segments j and k	Intersegment connection	θ_{jk}
1 and 2	A+	1
2 and 3	B+	1
3 and 4	C+, A-	2
4 and 5	B-, C-	2

Observe that there are no intersegment conditions in this PN model. This is because the places modeling limit switches are eliminated in this model. Had they been such places, completion of A+ results in marking a place modeling the limit switch, a1 which result intersegment conditions. For simplicity, denote the number of basic elements in PN corresponding to Sequence i as α_{si} . Now, following the formulas presented earlier, α_{s1} can be estimated as $\alpha_{s1} = (5+5+7+9+7) - (0) + (1+1+2+2) = 27 - 0 + 6 = 33$. This value exactly matches with the manual count of the basic elements in the PN shown in Fig. 8.1. In contrast to this, the PN model corresponding to Sequence 1 shown in Fig. 7.1 (b) has 50 basic elements.

Sequence 2 is obtained by adding a counter and emergency stop to Sequence 1. Hence, α_{s2} can be obtained by adding the basic elements needed to model a counter (α_C) and emergency stop (α_{ES}) to α_{s1} . Then, $\alpha_{s2} = \alpha_{s1} + \alpha_C + \alpha_{ES}$. By following the models presented earlier, $\alpha_C = 5$. The number of basic elements to model ES can be

calculated using the earlier developed formula. Since there are four actions (recall that {C+,A-} and {B-,C-} are treated as two concurrent actions) present in the sequence and each action is modeled by one transition, $\alpha_{ES} = 1 + k\Gamma = 1 + 1.4 = 5$. Therefore, $\alpha_{S2} = 33+5+5 = 43$ which exactly matches with the number obtained by manually counting basic elements in Fig. 8.2.

Sequence 3 is obtained by adding delay before {B-,C-} of Sequence 2. However, in PNs this delay is implemented by changing the second attribute of transition modeling 'do {C+,A-}'. Hence, there is no need for extra basic elements. In other words, $\alpha_{S3} = \alpha_{S2} = 43$. Sequence 4 is obtained by adding a counter and timer to Sequence 3. Again, we know that $\alpha_C = 5$, and there is no need for extra basic elements to model timer. Hence, $\alpha_{S4} = \alpha_{S3} + \alpha_C = 43 + 5 = 48$ which is the same as the number of basic elements obtained by manually counting the elements in Fig. 8.4.

Table 8.2. compares the number of basic elements in the PN models presented in this chapter with the corresponding models in Chapter 7.

Table 8.2 Comparison of the basic elements in LLD and PNs obtained by two methods

Sequence #	β	α in Chapter 7	α in this chapter
1	58	50	33
2	87	64	43
3	94	64	43
4	130	69	48

From Table 8.2, it can be said that the PN models obtained by following the conversion procedure yields simpler PN models. Even though the PN model obtained for Sequence 1 using the conversion procedure is much shorter than the PN model presented in Chapter 7, this difference will not change when timers and counters are added to the specification.

Also adding the emergency stop to the PN model obtained using the conversion procedure requires less basic elements compared to the PN model shown in Chapter 7. This is because, using the conversion procedure, each action is modeled by one transition, whereas in Chapter 7 each action is modeled as two transitions.

8.2.2. Single Path Sequences with Repetitive Actions

To illustrate the formulation of PN models corresponding to single path sequence with repetitive actions, two examples are considered. The former has a sequence without concurrent actions and the latter has a sequence with concurrent actions.

Example 1

Consider that four cylinders A, B, C, and D are to be controlled according to the sequence: START, A+, B+, C+, C-, A-, D+, A+, D-, B-, A-. The PN model obtained by following the conversion procedure is shown in Fig. 8.5. For simplicity, the place and transition attributes are not shown in the PN models.

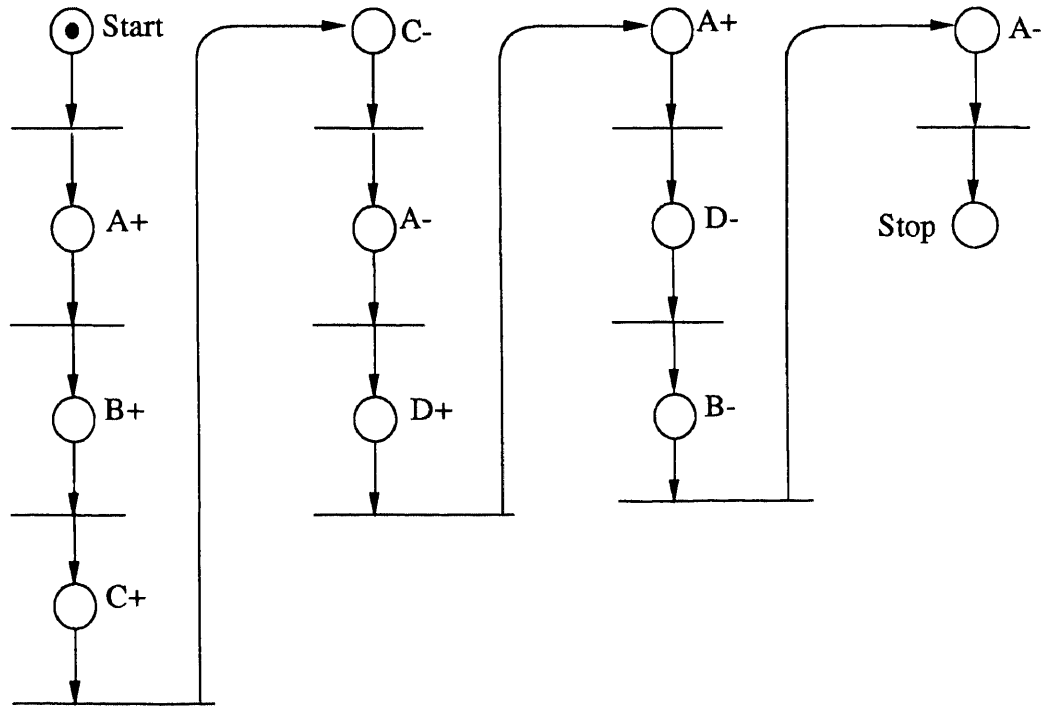


Figure 8.5 PN model corresponding to the sequence in Example 1

Example 2

Now, consider that four cylinders A, B, C, and D according to the sequence: START, A+, B+, C+, {C-, A-}, {D+, A+}, {D-, B-}, A-. Following the conversion procedure the PN modeling the above sequence is shown in Fig. 8.6. Merging of the common places in the PN models shown in Figs. 8.5 and 8.6 is considered later.

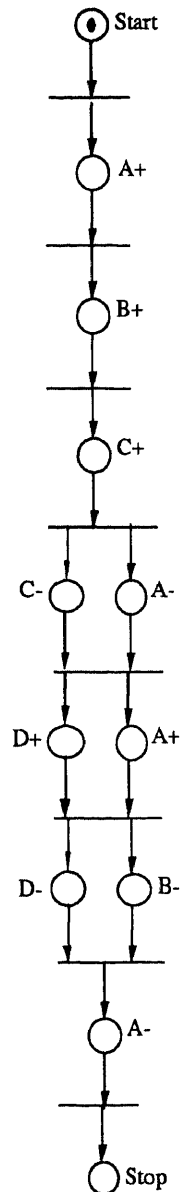


Figure 8.6 PN model corresponding to the sequence in Example 2

8.2.3. Multi-path Sequence with Simultaneous Parallel Paths

Figure 8.7 (a) shows the graphical representation of the control specification that involves the control flow consisting of multi-path sequences with simultaneous parallel paths. In this figure, the block dots represent nodes and $i, j,$ and k represent positive integers. Each node may be a typical controller such as computer, programmable logic controller, or a flip-flop. The control flow consists multi-paths sequences because after the node i completes its actions, it transfers the control simultaneously to paths A and B. As shown in each path there exists several nodes. The control is transferred to node $i + 1$ after all the controllers in paths A and B complete their actions. The PN modeling the control flow in Fig. 8.7 (a) is shown in Fig. 8.7 (b).

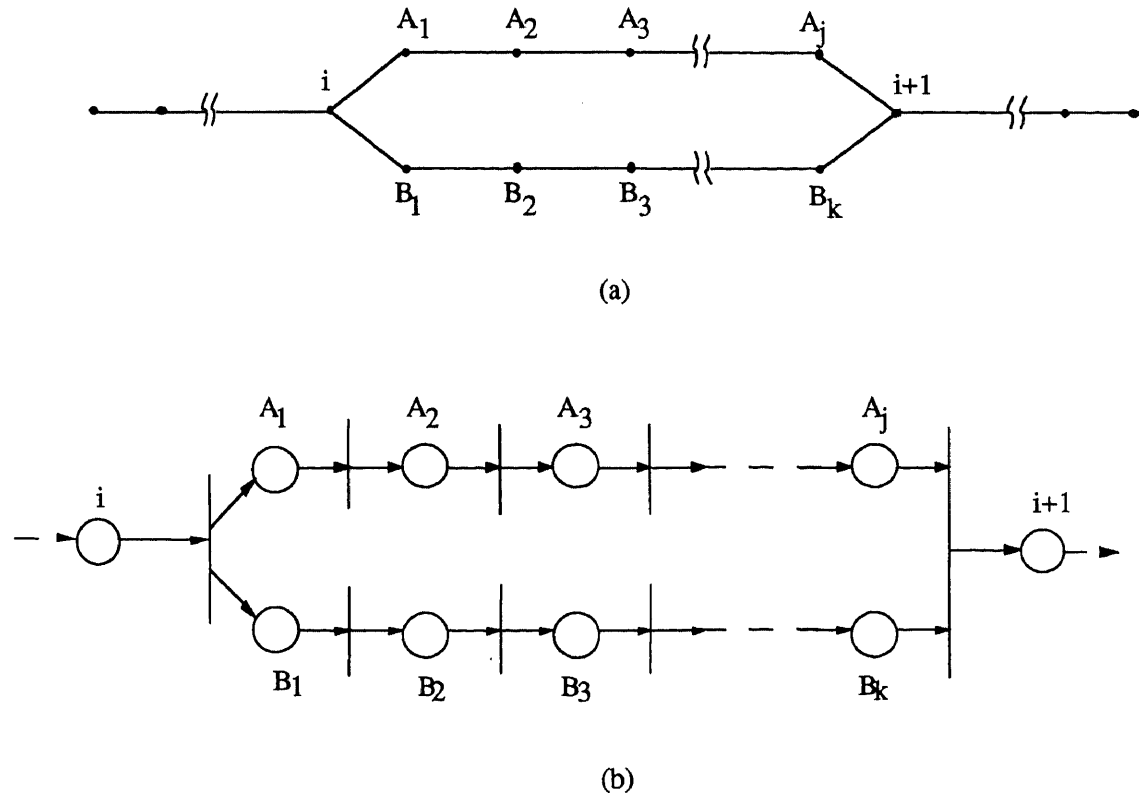


Figure 8.7

- (a). Control flow consisting multipath sequence with simultaneous parallel paths
 (b). Simplified PN modeling multipath sequence with simultaneous parallel paths

8.2.4. Multi-path sequence with Alternative Parallel Paths

Figure 8.8 (a) shows the graphical representation of the control specification that involves the control flow consisting of multi-path sequences with alternative parallel paths. In this figure, there exist two paths after node i . After node i completes its assigned actions, it transfers the control to either path A or path B depending upon the input signal X_p set by the environment. If $X_p = 1$ (0), control is transferred to path A (B). This X_p signal can be set manually to select the desired path or automatically depending on environment conditions. In other words, the place X_p or X_p being marked with a token depends upon the environment or other conditions instead of the sequence controller.

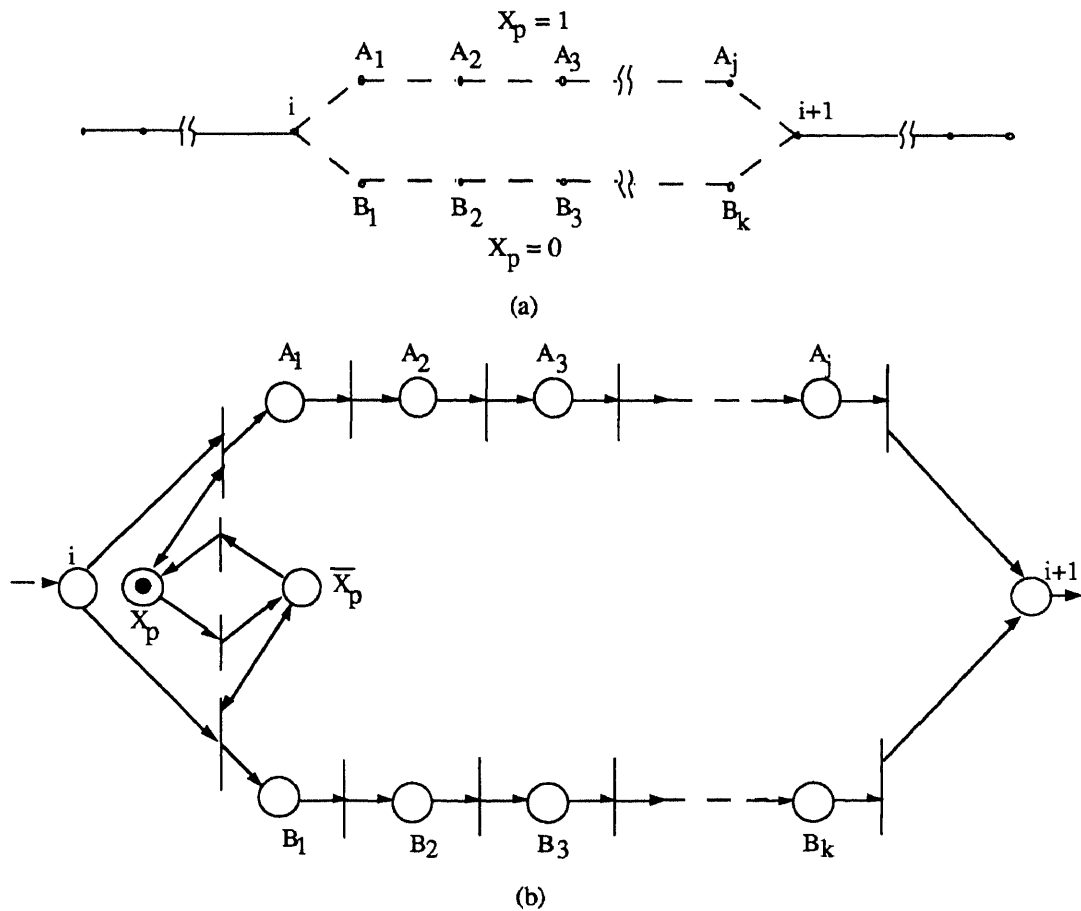


Figure 8.8

- (a). Control flow consisting multipath sequence with alternative parallel paths
 (b). PN modeling the multipath sequence with alternative parallel paths

Once either path A or B has been completed (OR function), the system continues with step $i + 1$. The PN modeling the control flow in Fig. 8.8 (a) is shown in Fig. 8.8 (b). An arc with an arrow at both ends represents a self-loop.

8.2.5. Multi-path Sequences with Option of Bypassing Nodes

Figure 8.9 (a) shows the graphical representation of the control specification that involves the above form of control flow. After the completion of node i , if X_p is 1, the system goes through nodes A_1 to A_j , and then transfers control to node $i + 1$.

If X_p is 0, the nodes A_1 to A_j are bypassed and the control jumps directly from node i to node $i + 1$. The PN modeling the control flow in Fig. 8.9 (a) is shown in Fig. 8.9 (b).

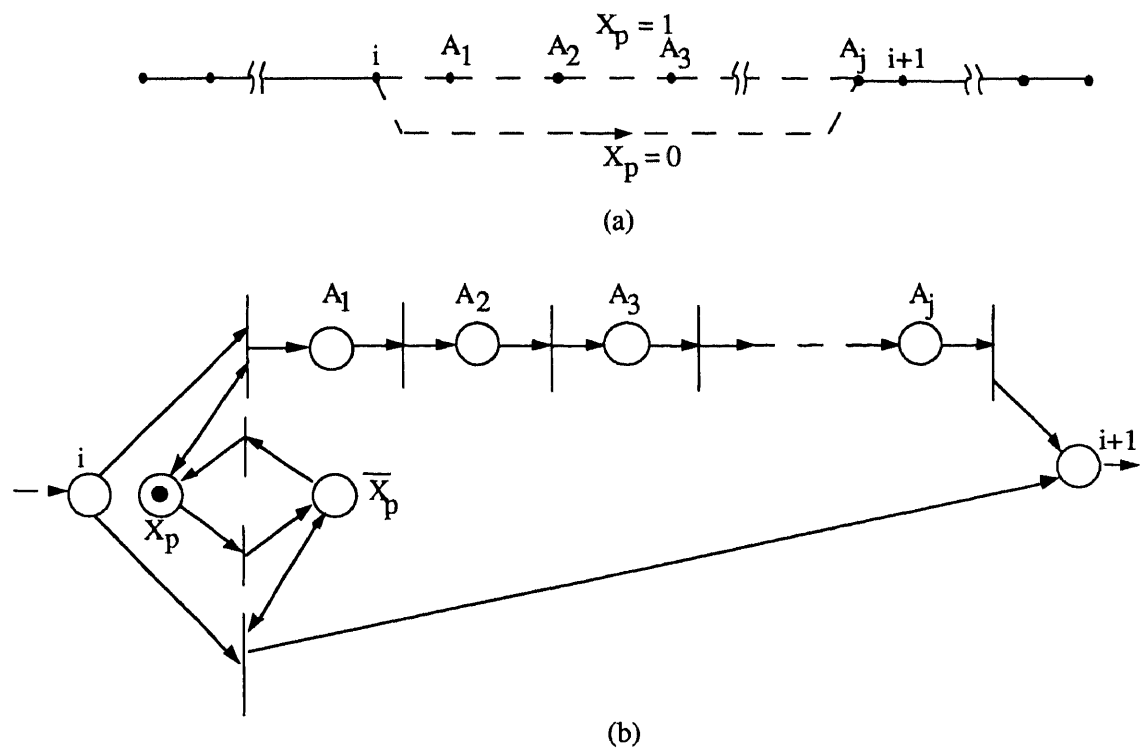


Figure 8.9

- (a). Control flow consisting multipath sequence with option of bypassing nodes
 (b). PN modeling the multipath sequence with option of bypassing nodes

8.2.6. Multi-path Sequences with Option of Repeating Nodes

Figure 8.10 (a) shows the graphical representation of the control specification that involves the above form of control flow. After the completion of node i , the system continues from node A_1 to A_j to $i + 1$ when X_p is 1. If, X_p is 0, the control is transferred back to A_1 . Thus the operations assigned to nodes A_1 to A_j are repeatedly executed until X_p is 1. The PN model corresponding to the control flow in Fig. 8.10 (a) is shown in Fig. 8.10 (b).

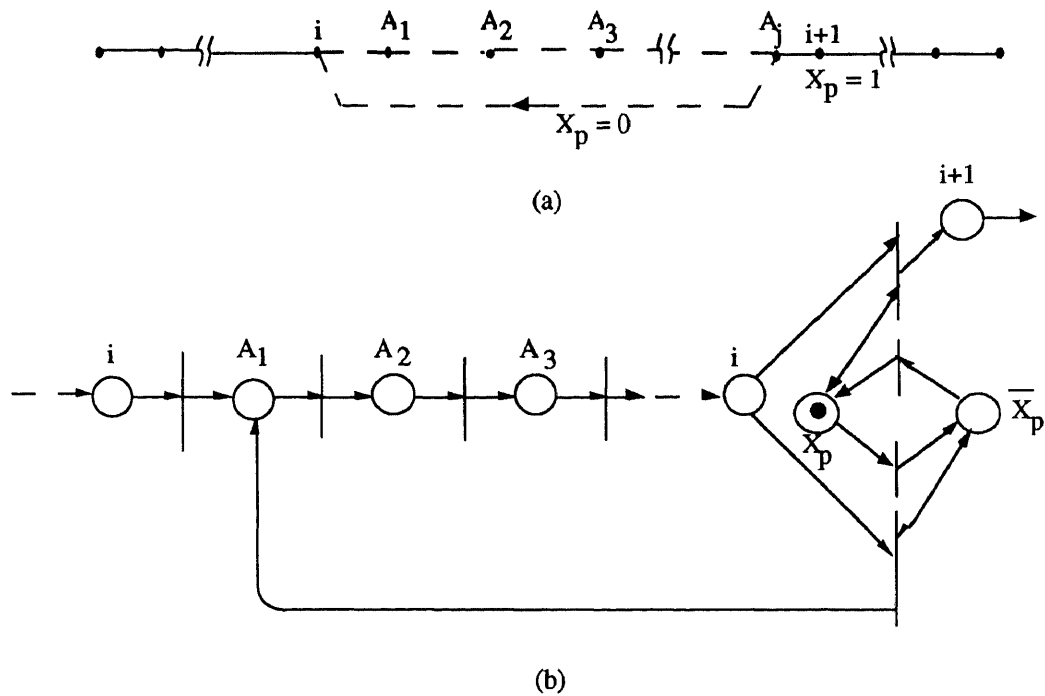


Figure 8.10

- (a). Control flow consisting multipath sequence with option of repeating nodes
 (b). PN modeling the multipath sequence with option of repeating nodes

8.3. Merging of Common Places in a PN Model

In a PN modeling a sequence that consists of repetitive actions, places modeling repetitive actions may appear more than once in a PN model. Even though merging of such common places is theoretically and physically possible, the resulting PN model after merging may become complex and hence occupies more space as illustrated later.

Hence, merging of such places modeling repetitive actions is not recommended.

This is illustrated in the following examples.

Example 1

Consider the PN shown in Fig. 8.5 that modeled a sequence with repetitive actions. Since, A+ and A- appears two times in the given sequence, the PN model in Fig. 8.5 has two places for representing A+, and two places for modeling A-. Such common places representing repetitive actions are merged as shown in the PN model shown in Fig. 8.11.

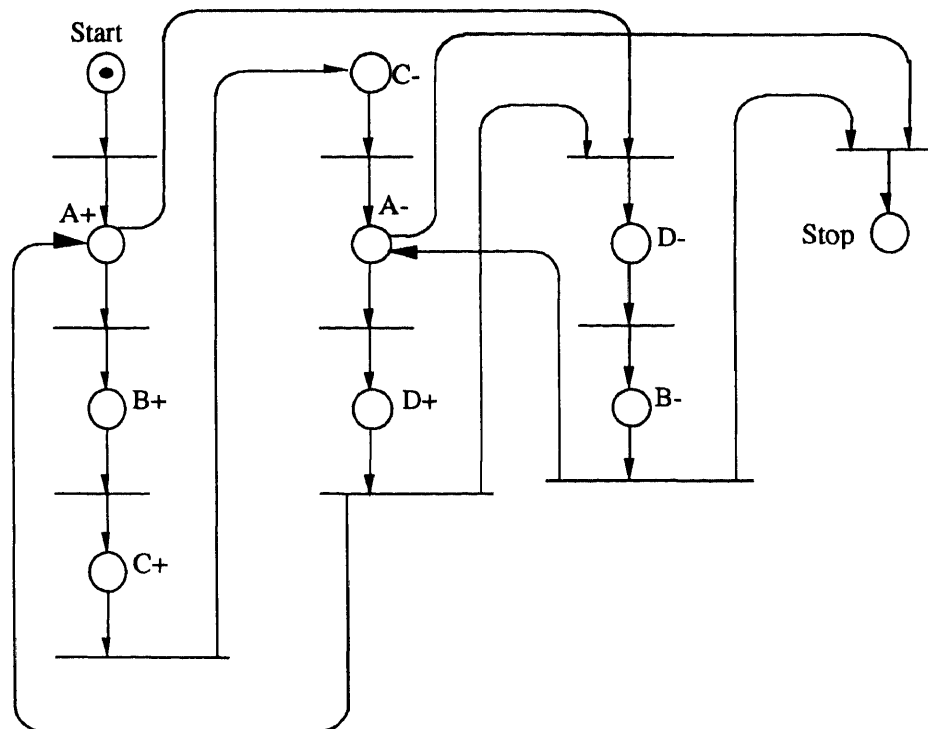


Figure 8.11 PN model after merging of common places in Fig. 8.5

Observe that the PN model shown in Fig. 8.11 has conflicts. In this PN model when the place, A+ is marked, conflict results to execute either B+ or D-. Similarly, when A- is marked, conflict results to execute either D+ or Stop. Hence, in the PN model shown in Fig. 8.12, conflicts are eliminated by introducing two new places P_{rc1} , P_{rc2} , P_{rc3} , and P_{rc4} .

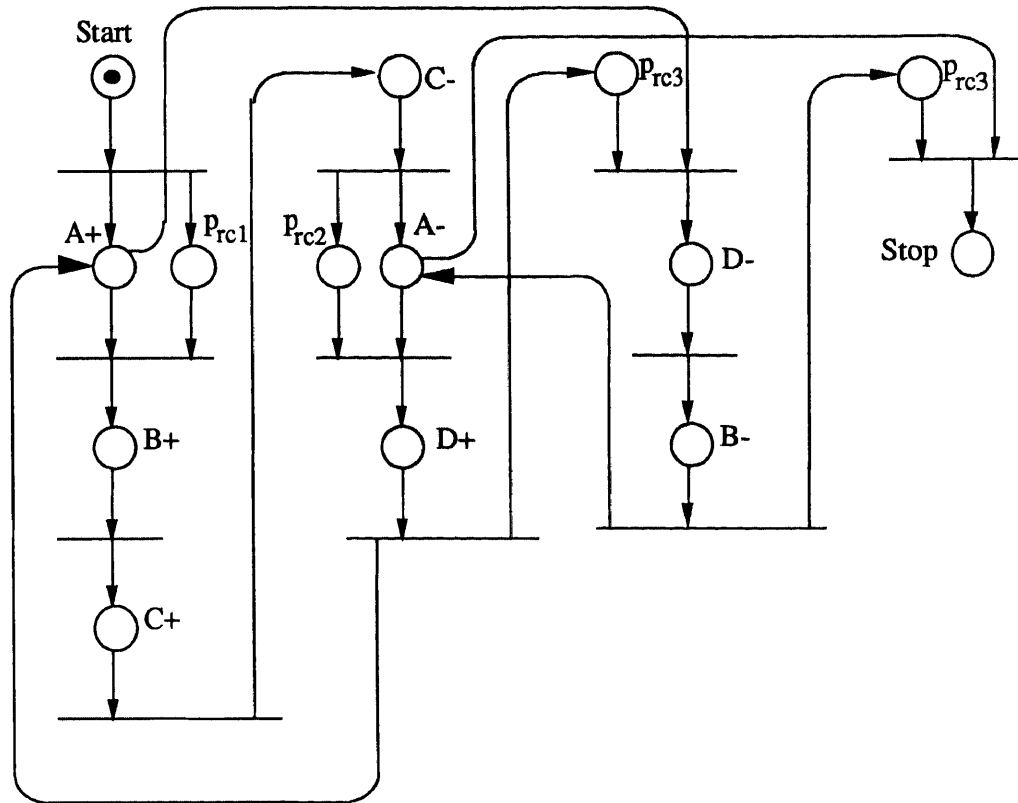


Figure 8.12 PN model after eliminating conflicts in Fig. 8.11

For even the simple sequence considered here, it can be easily seen that the PN shown in Fig. 8.12 is more complex than the one shown in Fig. 8.5. This complexity grows enormously when the given sequence is large contains many repetitive actions. Hence, merging of such common places is not recommended.

Example 2

Now, consider the PN shown in Fig. 8.6 that also modeled a sequence with repetitive actions. Since, A+ and A- appears two times in the given sequence, the PN model in Fig. 8.6 has two places for representing A+, and two places for modeling A-. Hence, such places are merged as shown in Fig. 8.13 (a). The conflicts resulted due to this merging are eliminated in the PN model shown in Fig. 8.13 (b).

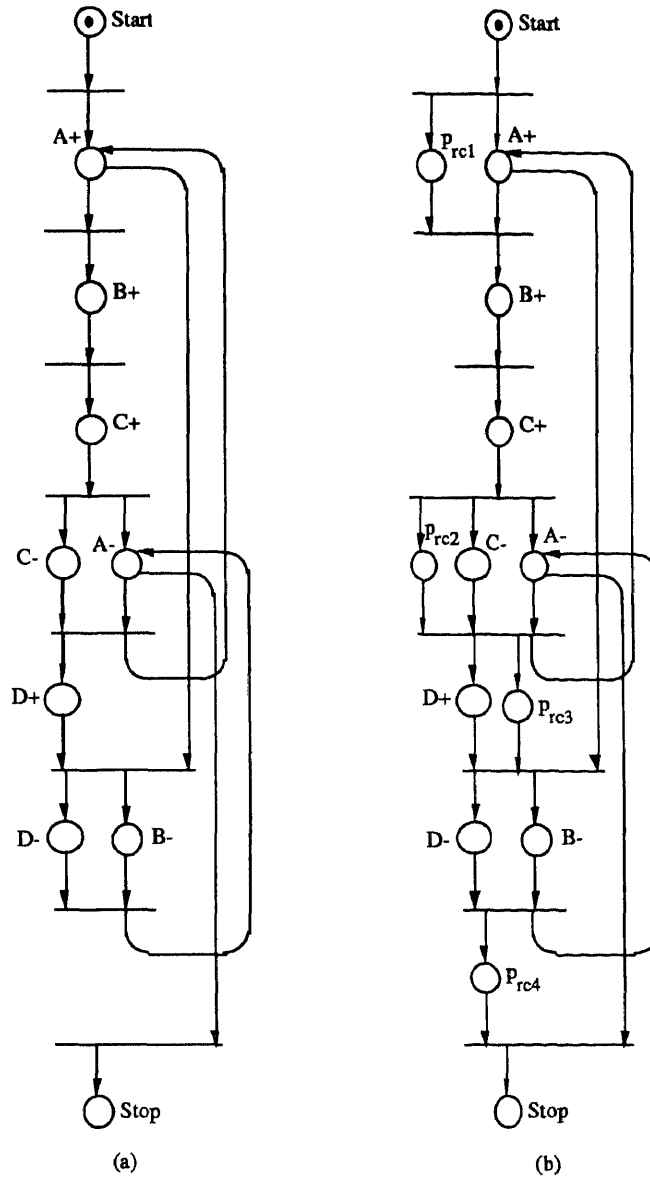


Figure 8.13 PN model (a) after merging the common places in Fig. 8.6, (b) after eliminating conflicts

Again, it can be easily seen that the PN model shown in 8.13 is more complex than the one shown in Fig. 8.6. Hence, while following the conversion procedure presented in this chapter to develop PN, merging is not recommended in the case of PN that models repetitive actions.

8.4. Summary

This chapter has presented a simple conversion procedure to formulate PN models from a given logic control specification. The PN model developed using the conversion procedure is simple and easy to understand. The conversion procedure is applied to design PN models corresponding to the classical control sequences that exist in discrete event control of manufacturing systems. This procedure is illustrated for several types of logic control specifications, single and multi path sequences, with and without repetitive actions. The number of basic elements in the PN model obtained by following the conversion procedure is compared with that obtained in Chapter 7. The comparison clearly indicates that the PN models presented in this chapter have less design complexity compared to those in Chapter 7. It is also concluded that in case of formulating PN models corresponding to sequences with repetitive actions, merging of common places increases the complexity of the PN due to the need to introduce additional places and arcs to keep the decision-freeness. Hence, merging of places in such situations is not recommended.

Using the same PN model, both types of systems namely, systems that require sustained actuating signals and the systems that require only a momentary or pulsed actuating signals can be controlled. This can be accomplished by changing the corresponding attributes for transitions as reported in PN based control using Real-time PNs. However, when using ladder logic diagrams, two separate design procedures have to be followed to design two separate diagrams as shown in (Pessen 1989).

CHAPTER 9

AN OBJECT-ORIENTED DESIGN METHODOLOGY FOR DEVELOPMENT OF FMS CONTROL SOFTWARE

9.1 Introduction

In order to develop integrated control software, there is a need for integrated tools and a systematic design methodology to utilize those tools. The integrated tools should aid in not only real-time control but also simulation. The systematic design methodology should aim to develop reusable, modifiable, and extendible control software in order to meet the changing control specifications. Earlier chapters have shown the application of Petri nets (PNs) for both real-time control and simulation. This chapter makes an attempt to introduce PNs into object-oriented design (OOD) of integrated control software development.

A flexible manufacturing system (FMS) consists of machines, robots, automated guided vehicles, programmable logic controllers, and computers, all of which can be viewed as objects in OOD. The system control software has to be designed to meet the real time constraints of a production line and the dynamically changing needs of the market. Because of the complexity, systematic software development is very important to realize the full benefits of agile and flexible manufacturing. This chapter highlights the difficulties in developing such software and proposes a systematic OOD methodology for its development using object modeling technique (OMT) diagrams and Petri nets (PNs). OOD is used to design reusable and easily maintainable software. OMT diagrams are used to represent explicitly different kinds of static relations such as generalization, aggregation, and association among the objects in FMS. PNs are used as a tool to model the dynamic behavior of the objects and FMS. By adopting the bottom-up approach of PN modeling, this chapter shows that PNs can support important characteristics of OOD, namely, reusability, extendibility, and modifiability. They are also used for the FMS performance analysis. Furthermore, with PN models, a method to identify the data structures and

operations of objects is presented. The proposed methodology is illustrated through an example of FMS. The issues related to the reusability, extendibility, and modifiability of the resulting control software are also discussed by changing the initial specifications of the FMS and thus its OMT diagram and PN model.

Background

Flexibility in the functioning of FMSs is mainly imparted by the use of computers and robots which are programmable according to the shop floor needs. Systematic control software development for FMSs is very important to realize agile and flexible production (May 1986, Marinov and Todorov 1988). Control software that not only supports simulation but also helps in FMS implementation is of paramount importance to exploit the full benefits of FMSs (May 1986). The advantages and the importance of integrated software packages that can be used for both simulation and control are discussed in (Bruno and Marchetto 1986, May 1986, Glassey and Adiga 1990, Venkatesh *et al.* 1991, Johnson, *et al.* 1992).

Traditionally the function of FMS control software was to coordinate and control different elements in a manufacturing system. However, recently there are several efforts to integrate the control software and simulation software in order to expedite the development of control software and system design (Naylor and Voltz 1987, Chaar, *et al.* 1991, 1993a,b, Venkatesh, *et al.* 1991, Johnson, *et al.* 1992). Changes in the production facilities may also require redesign and rewriting of large amount of software. The difficulty in developing the FMS control software is further compounded by the inherent complexity of FMS execution since it is a complex asynchronous concurrent system aiming to produce several product types with different production processes. Hence, the traditional definition of control software has been extended to handle simulation, planning, monitoring, rapid prototyping, and scheduling (Jain 1986, Naylor and Voltz 1987, Chaar, *et al.* 1991, 1993a,b, Venkatesh, *et al.* 1991).

A design methodology for FMS control software should at least deal with the issues related to 1) modeling, simulation, and analysis, and 2) real-time control implementation of FMS. The first one typically involves building models of an FMS and estimating measures of system performance such as throughput rate, utilization of robots, machines, and queue lengths at each machine. The aim of this activity is to suggest the optimum configuration of FMS for the given specifications. The optimum configuration may include the layout of the FMS, the routings of automated guided vehicles (AGVs) among machines, the machine scheduling policies, and other tasks. The second one is to implement effective real-time control of FMS which typically involves coordinating, on-line scheduling and monitoring of system resources. The functional objective of control software is to maintain high system utilization and throughput as well as to satisfy the real-time production deadlines. In addition to this in FMSs, it is very common to find similar machines and robots grouped in cells which function in a similar way. The only difference between the functioning of these cells from discrete control point of view is the routing of parts among machines in cells. Hence, the control software should be written for a cell such that it can be used to control another cell with minimum or no change.

Based on the above discussion, it can be concluded that the control software should be reusable, modifiable, and extendible to: 1) adapt to changes in the system configuration and specifications, and 2) to deal with a complex shop-floor system which often consists of numerous similar components. A systematic design methodology is obviously needed to develop such FMS control software (Naylor and Voltz 1987, Glassey and Adiga 1990, Chaar et al. 1993b, Venkatesh and Fernandez 1993).

Literature review and motivation

Realizing the importance and complexity of FMS software many researchers are investigating different issues related to it. Naylor and Voltz (1987) proposed an approach for designing integrated manufacturing system control software. They use three software concepts, the use of software components extended to include hardware, a common

distributed language environment, and generic software. Smith and Joshi (1992) described reusable software concepts for developing scaleable FMS control architecture, automatic generation of control code, and object-oriented design of equipment controllers. Hsu (1992) described how object-oriented programming is used for FMS control software development along with its advantages.

Stuznebecker (1991) proposed extensions to C++ language with an aim to formulate it as a base language for developing object-oriented environment for distributed manufacturing software. Char *et al.* (1991, 1993a,b) presented excellent reviews on recent methods for developing manufacturing control software and concluded that current methodologies are not sufficient to support planning, scheduling, and monitoring activities involved in manufacturing.

Another research area related to FMS control software is the application of Ada as an implementation language. An approach is proposed for rapid prototyping of control software using Ada and PNs (Sahraoui and Ould-Kaddour 1992). Ada is used for building reusable software components and PNs for specifying task communication and synchronization. Ould-Kaddour and Courvoisier (Ould-Kaddour and Courvoisier 1989) presented a method for real-time software prototyping using PNs for specification, Ada for specification description, and Modula-2 for implementation. Ada was selected as a suitable language for software development of fully automated manufacturing systems (Voltz, *et al.* 1984, Venkatesh *et al.* 1991).

The research on FMS control software has been concentrated on combining OOD concepts, PNs, and Ada as shown in Table 9.1. The benefits of OOD such as reusability, extendibility, and modifiability of the software system were not sufficiently demonstrated with a particular example of FMS in previous works (Glassey and Adiga 1990, Sturzenbecker 1991, Graham, *et al.* 1991, Hsu 1992). They did not discuss in detail the relations among several objects present in an FMS, which are essential for OOD.

Table 9.1 Research related to OOD, PNs, and Ada

Research related to:	Selected references
FMS and OOD	Garg (1985), Glassey and Adiga (1990), Graham, <i>et al.</i> (1991), Sturzenebecker (1991), Hsu (1992), Smith and Joshi (1992)
FMS and PN	Bruno and Marchetto (1986), Cecil, <i>et al.</i> (1992), Jafari (1992), Proth (1992), Venkatesh and Ilyas (1995), Zhou and DiCesare (1989), Zhou, <i>et al.</i> (1991), Zhou, <i>et al.</i> (1992)
FMS and Ada	Voltz, <i>et al.</i> (1984), Venkatesh, <i>et al.</i> (1991)
Ada and PNs	Murata, <i>et al.</i> (1989)
FMS, OOD and PN	Bruno and Marchetto (1986), Ould-Kaddour and Courvoisier (1989), Booch (1991), Sahraoui and Ould-Kaddour (1992)
FMS, OOD and Ada	Chaar, <i>et al.</i> (1993b), Naylor and Voltz (1987)
FMS, OOD, PN, and Ada	Venkatesh and Fernandez (1993)

Furthermore, the earlier research (Glassey and Adiga 1990, Sturzenbecker 1991, Graham, *et al.* 1991, Hsu 1992, Char, *et al.* 1993b) that used OOD for design of FMS software has not paid much attention to describing the dynamic behavior of the objects present in the software system and to analyze quantitatively the system performance. Realizing the importance of these issues, Venkatesh and Fernandez (1993) first proposed an approach for object oriented simulation and control of FMSs using timed PNs and Ada. Subsequently, Fernandez and Han (1993) illustrated the concepts of OOD design by developing Object Modeling Technique (OMT) diagram of a simple assembly cell and show how OOD can adapt to changing conditions in the manufacturing cells. However, there is a need for a systematic methodology that integrates the OOD concepts, OMT diagrams, PNs, and Ada to address problems related to FMS control software. The goal of this chapter is to formulate an OOD methodology for developing reusable, extendible, and modifiable control software for FMSs using OMT diagrams and PNs. The objectives of this work are:

1. To present a OOD methodology combining the concepts of OMT diagrams and PNs by discussing the rationale and advantages of selecting OOD and PNs respectively,
2. To emphasize the use of PNs as a dynamic tool in OOD by discussing the advantages of PNs over earlier used dynamic models,
3. To demonstrate how PNs can support the concepts of reusability and extendibility by adopting the bottom-up approach of FMS modeling,
4. To illustrate the methodology by developing object modeling technique (OMT) diagram and PN model of an FMS example, and
5. To demonstrate the benefits of the methodology to support reusability, modifiability, and extendibility of the software system when the configuration and specifications of the FMS are subject to change.

9.2. Methodology for FMS Control Software Development

9.2.1. Methodology

It is assumed that a reader has basic knowledge on OOD (Booch 1991, Rumbaugh, et al. 1991). While the advantages of OOD and PNs are shown in Fig. 9.1 and will be discussed later, the rationale why they are combined is discussed here. The motivation for combining OOD and PNs is that in a broader sense they are complementary to each other to achieve the goal of system development at incremental stages.

OOD is an appropriate design methodology to support system design (Booch 1991, Rumbaugh, et al. 1991). PNs are a versatile hierarchical modeling tools that support all the stages of system development including its specification, planning, design, evaluation, monitoring, control, and implementation (Zhou and DiCesare 1989, Zhou, *et al.* 1991, Proth 1992).

Earlier techniques of OOD that do not use PNs (Glassey and Adiga 1990, Booch 1991, Rumbaugh, *et al.* 1991, Smith and Joshi 1992) have some disadvantages since they cannot explicitly represent detailed dynamic interactions that involve concurrency and synchronization among objects present in the system. Also, they are difficult to consider timing information in the design for qualitative analysis and performance evaluation of an FMS. PNs can be used to augment the applicability of OOD by eliminating these disadvantages because they are primarily a tool that models and analyzes complex asynchronous concurrent interactions in distributed systems. They can also help select data structures and operations for objects. On the other hand, PNs have some disadvantages since they cannot clearly represent the static relationships among objects, which is essential for reusable software development (Garg 1985, Boucher, *et al.* 1989, Rumbaugh, *et al.* 1991, Venkatesh and Fernandez 1993). However, these relationships can be modeled by using object modeling technique (OMT) diagrams in OOD (Rumbaugh, *et al.* 1991).

Hence, by using OMT to represent the static relations and PNs to represent dynamic relations, a powerful OOD methodology for the development of FMS control software can be developed. The methodology proposed is illustrated in Fig. 9.1. In this approach, OOD is used as a software design tool and PNs are selected mainly as a dynamic modeling and performance evaluation tools. The methodology is explained in the following steps:

1. Apply OOD concepts to find objects in an FMS and to show the explicit relations among them by developing an object modeling technique (OMT) diagram (Rumbaugh, *et al.* 1991). The objective of an OMT diagram is to show the static relations among objects in the FMS.
2. Use PNs for modeling the FMS in order to show the dynamic relations among objects. The objective of a PN model (PNM) is multifold.
3. Use the formulated PNM in the above step to analyze, simulate, and evaluate the FMS performance. The performance criterion can be production rate, system utilization, work-in-process inventory, etc.

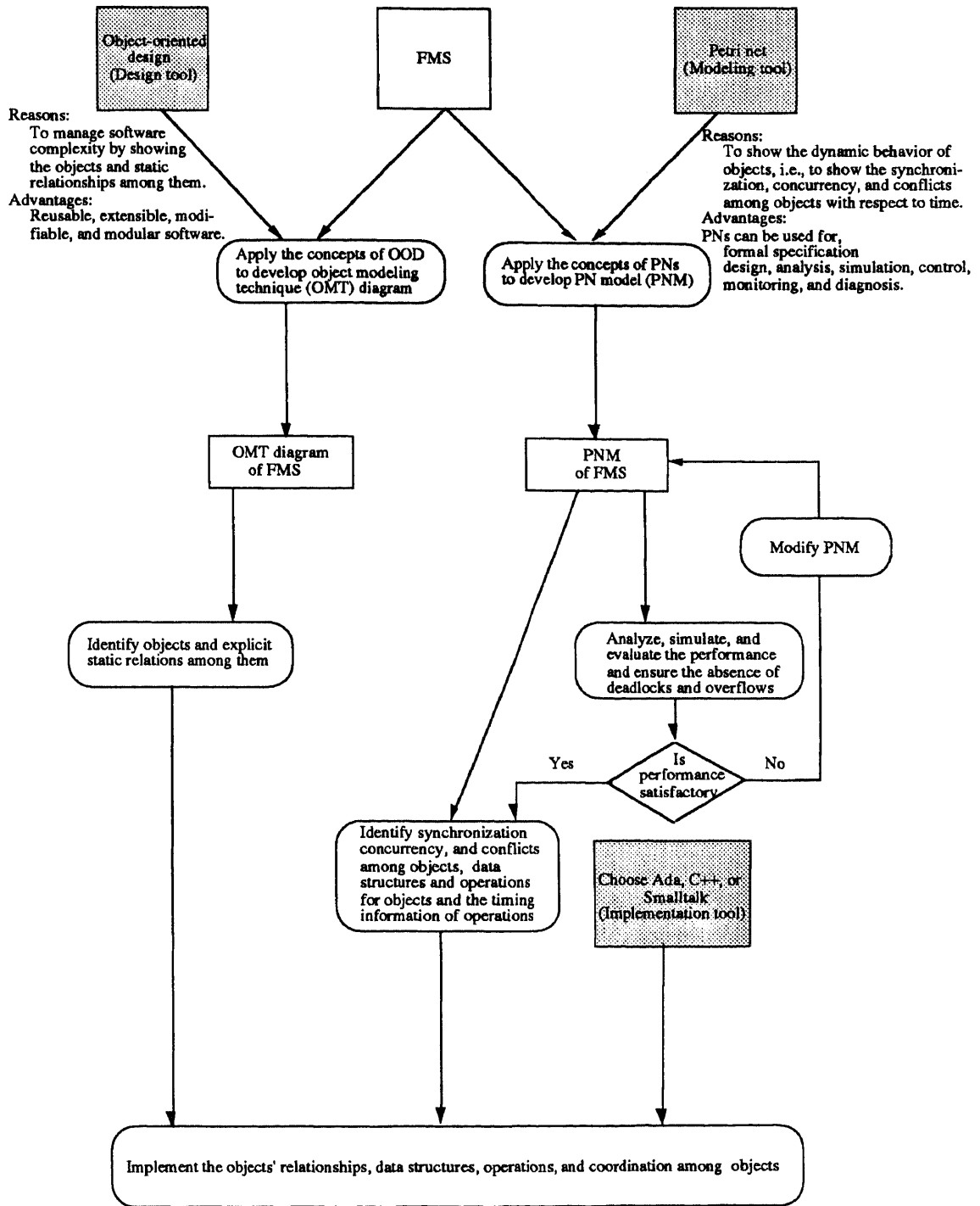


Figure 9.1 Proposed systematic methodology for FMS control software development

At this stage, if its performance is found unsatisfactory, it should be improved by changing some parameters, the system configuration, and/or operational policies. After the analysis of the final PNM is completed, the design parameters of specific interest in controlling FMS can be decided. Furthermore, with the aid of PNM, data structures and operations for objects present in FMS can be systematically identified.

4. Combine OMT and PNM to design the complete structure of objects with their static and dynamic relations. Select an appropriate language such as Ada, C++, or Smalltalk to implement the objects and control of the FMS.

9.2.2. Fundamentals of OOD

Each unit in an FMS interacts with several other objects to complete a set of production tasks. Further, objects can be added or removed from the FMS thereby demanding the software to be extensible, modifiable, and reusable. This makes OOD particularly suitable for designing FMS control software. Fig. 9.2 shows how the concepts of OOD can be used for developing FMS control software.

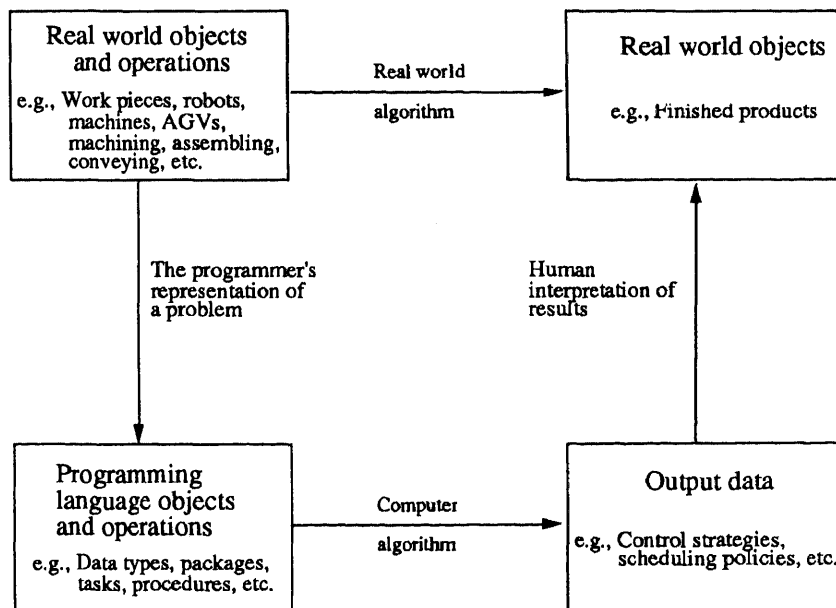


Figure 9.2 Principle of object-oriented design of FMS control software

Using OOD techniques, a software system can be developed by mapping the real-world entities into objects and transformed to predict the behavior of the model. This approach to software development results in an implementation that is understandable and therefore useful in managing the FMS complexity. From Fig. 9.2, it can be observed that the closer the solution space maps to the concepts of the problem space, the easier is to achieve all the goals of the software system such as modifiability, extendibility, and reusability. An interested reader is referred to (Booch 1991, Rumbaugh, *et al.* 1991) for further discussion on the development of OOD and its advantages. The following paragraphs present a brief discussion on the OOD concepts used in this chapter.

The fundamental building block of OOD is an *object* that contains both a data structure and a collection of related procedures. Procedures are also called as operations or methods. Objects interact with each other by sending messages or by calls to their interfaces. Objects with the identical data structure (attributes) and behavior (operations) can be grouped into a class. Each object is an instance of its class. An important feature of OOD is *inheritance*. Inheritance is a mechanism whereby one class of objects can be defined as a special case of another class, automatically including the data structure and behavior of that class. The special cases of a class are known as subclasses. For example, *machine_controller* (subclass) is a special class of *controller* (superclass). *Polymorphism*, another important feature of OOD is a property in which the same operation may behave differently on different classes. For example, an operation called *process_part* behaves differently on two different classes, *milling_machine*, and *drilling_machine*. There are several approaches to develop OOD systems (Booch 1991, Rumbaugh, *et al.* 1991).

The complete OOD methodology consists in developing three orthogonal models (Rumbaugh, *et al.* 1991):

1. Object model: It divides the application into object classes and shows the static data. The relationships (interfaces) among the classes are described by the class structure. The behavior of the objects are defined by operations associated with the object class.

2. Dynamic model: It shows the way the system behaves with internal and external events by capturing the time-dependent behavior of the system.
3. Functional model: It shows how to process the data flow in the system during each event or action.

Since the functional model deals with the lower level implementation of a software system, it is not considered here. This chapter uses object modeling technique (OMT) diagram (Rumbaugh, *et al.* 1991) as an object model and PNs as a dynamic model. There are few other approaches for object modeling (Monarchi and Puhr 1992, Booch 1994). Even though the notations used in these methods are different, they are conceptually similar in their way to represent the object structure.

9.2.3. Object Modeling Technique Diagram as a Static Modeling Tool

Figure 9.3 shows the definition of a class with three separate areas: *class name*, *class attributes*, and *operations*. In the following figures some of the details are omitted if they are not of particular interest in the chapter. The OMT diagram [29] allows three basic types of relations among classes, i.e., *generalization*, *aggregation*, and *association* which are explained below:

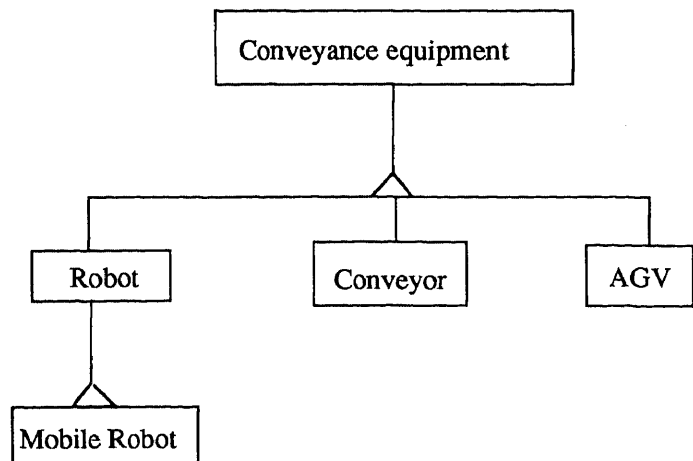
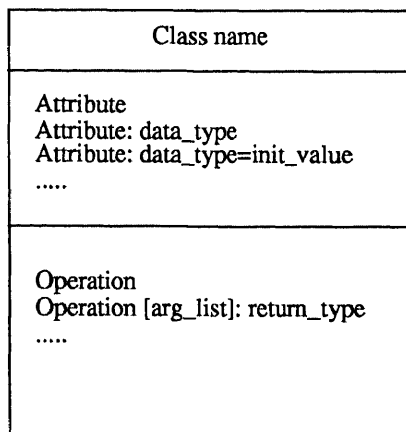


Figure 9.3 A class definition in OMT **Figure 9.4** OMT diagram for Generalization

1. Generalization implies the definition of a superclass that collects the common characteristics of several subclasses. It describes an "is-a" association between a subclass and its superclass. The symbol for generalization is a triangle. For example, in Fig. 9.4, the superclass *conveyance equipment* is a generalization of the classes *robot*, *conveyor*, and *AGV*; *robot* in turn is a generalization of *mobile robot*.
2. Aggregation implies the description of a class in terms of its constituent parts. The concept of aggregation defines an "is-a-part-of" relationship between a subclass and its superclass. For example, Fig. 9.5 shows that an *assembly system* consists of assembly cells, which in turn consists of *robots* and part feeders. Aggregation is denoted by a diamond. The black dot indicates multiplicity, e.g., an *assembly system* is composed of several assembly cells, an *assembly cell* is composed of several robots, and part feeders.
3. Association describes how objects belonging to different classes are related to each other. The OMT symbol for an association is a line connecting two classes labeled with the association name. Associations can be binary, ternary, or even higher order, and can have any number of attributes. For example, in Fig. 9.6, *sensor* monitors *machine*, and *machine* machines *part*.

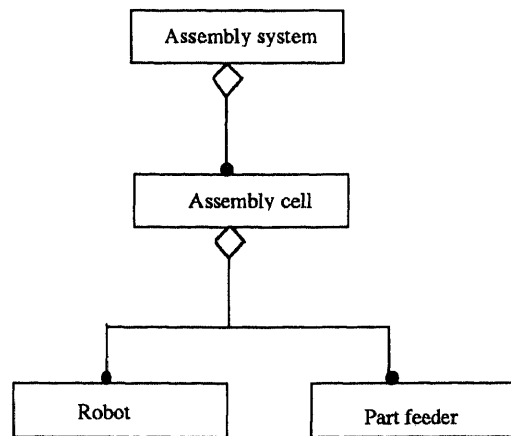


Figure 9.5 OMT diagram for Aggregation

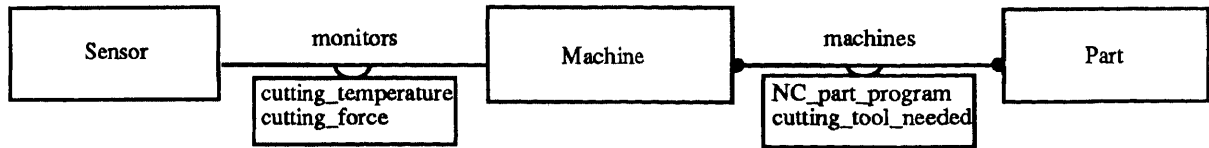


Figure 9.6 OMT diagram for Association

Associations can have attributes, e.g., cutting temperature, cutting force in Fig. 9.6. Since a machine can process several parts, and a part can be processed by several machines, block dots are present at the both ends of link connecting *machine* and *part*.

For more details on developing OMT refer to (Rumbaugh, *et al.* 1991). After the OMT diagram is developed, the next step is to formulate the dynamic model of FMS. State diagrams were used to develop a dynamic model (Rumbaugh, *et al.* 1991). However, they are not convenient to deal with timing aspects and the dynamic behavior of objects in FMS which must be considered (Rumbaugh, *et al.* 1991, Fernandez and Han 1993). Furthermore, they cannot explicitly represent important features such as concurrency and synchronization in FMS which makes it difficult to visualize the functioning of FMS. The state diagrams become complex in case of FMS (Crockett, *et al.* 1987, Zhou and DiCesare 1993). Hence, PNs are chosen to develop a dynamic model of FMS as discussed next.

9.2.4. Petri nets as a Dynamic Modeling Tool

In this chapter PNs are used as a dynamic model in object-oriented design (OOD). In contrast, other researchers use two different kinds of diagrams for representing the dynamic behavior of objects. Rumbaugh, *et al.* (1991) uses state diagrams and event trace diagrams; Booch (1994) uses state transition diagrams and interaction diagrams. State/state transition diagrams are used to represent how objects respond to the internal and external events in the system. Event trace/interaction diagrams are used to study the synchronization aspects and to trace the execution of events in the system.

However, in order to develop systematic control software for FMSs, a more formal dynamic modeling tool is needed. This is because the coordination of the individual units in FMSs is important. Hence, a dynamic modeling tool should model in detail the concurrency and synchronization in the system with respect to time. Furthermore, such tool should help to analyze the system behavior to check for aspects such as deadlocks. Since it is very common in FMSs to share certain resources (e.g. a robot is shared by more than one machine to load/unload), a dynamic modeling tool should represent these aspects to analyze the conflicts during the system execution. In addition to all these requirements, a dynamic modeling tool should support the system designer for system performance evaluation and assist control engineer to control and monitor the FMS. PNs have all these capabilities and hence are suitable as dynamic modeling tool irrespective of the various methods used for object model (Rumbaugh *et al.* 1991, Monarchi and Pühr 1992, Booch 1994). Also, unlike previous works which use two different kinds of diagrams for representing system states and tracing events (Rumbaugh *et al.* 1991 and Booch 1994), PNs can be used as a single tool to represent both the system states and to trace the events in the system when time durations of activities are associated with transitions.

Compared with previous techniques for a dynamic model in OOD, PNs have the following advantages:

1. PNs can explicitly and realistically represent concurrent operations, synchronization activities, and conflicts.
2. They can be easily associated with timing information for performance analysis of the system. Both analytical and simulation methods are available depending on the system complexity and accuracy needed.
3. They allow to check the system behavioral properties such as deadlock and capacity overflow.
4. By using PNs, a more compact model can be obtained and thus avoiding the painstaking enumeration of all the states at the design stage.

5. The developed PN models can also be extended for real-time control and monitoring.
6. The attributes and operations of potential objects can be selected from Petri net models (PNMs) of the FMS. This advantage of PNs reaches far significance when the quote from (Rogers 1991) is recalled: "the potential of OOD is impeded due to the lack of an established methodology for object identification."

After the OMT diagram and PN model (PNM) of FMS are developed, the software system can be implemented by selecting a proper computer language. Some researchers use Ada (Bruno and Marchetto 1986, Naylor and Voltz 1987, Venkatesh, *et al.* 1991, Sahraoui and Old-Kaddour 1992) and others prefer C++ for implementing OOD (Sturzenbecker 1991, Smith and Joshi 1992). Detailed comparison between Ada and C++ for OOD implementation of FMS control software falls beyond the scope of this work and can be an interesting topic for further research.

9.3. Illustration of the Methodology with an FMS

To illustrate the proposed methodology, the FMS discussed in the Chapter 4 is considered. The FMS is assumed to function under push paradigm. In this section the system description is given first. Next, the OMT diagram of the FMS is developed. Then, the PNM is formulated and finalized after the quantitative analysis of the system. OMT diagram and the final PNM are combined to identify objects and the static and dynamic relations among them.

System description

The FMS considered is shown in Fig. 9.7 and consists of four machines. In this system each machine is served by a robot and processes either raw material or intermediate parts. Once a part is processed, the machine setup is needed to process another part type. The activities of the robots are to load tools and parts to and unload used tools and processed

parts from the machines. Input and output buffers are provided at machines to store raw material, the work-in-process inventory, and processed parts. Sensors monitor the functioning of machines and robots. For example, if the temperature during machining exceeds its maximum value, the sensor monitoring the corresponding machine will send a signal to the cell controller to stop machining. AGVs are used to convey parts and subassemblies among the machines and to the assembly shop (AS).

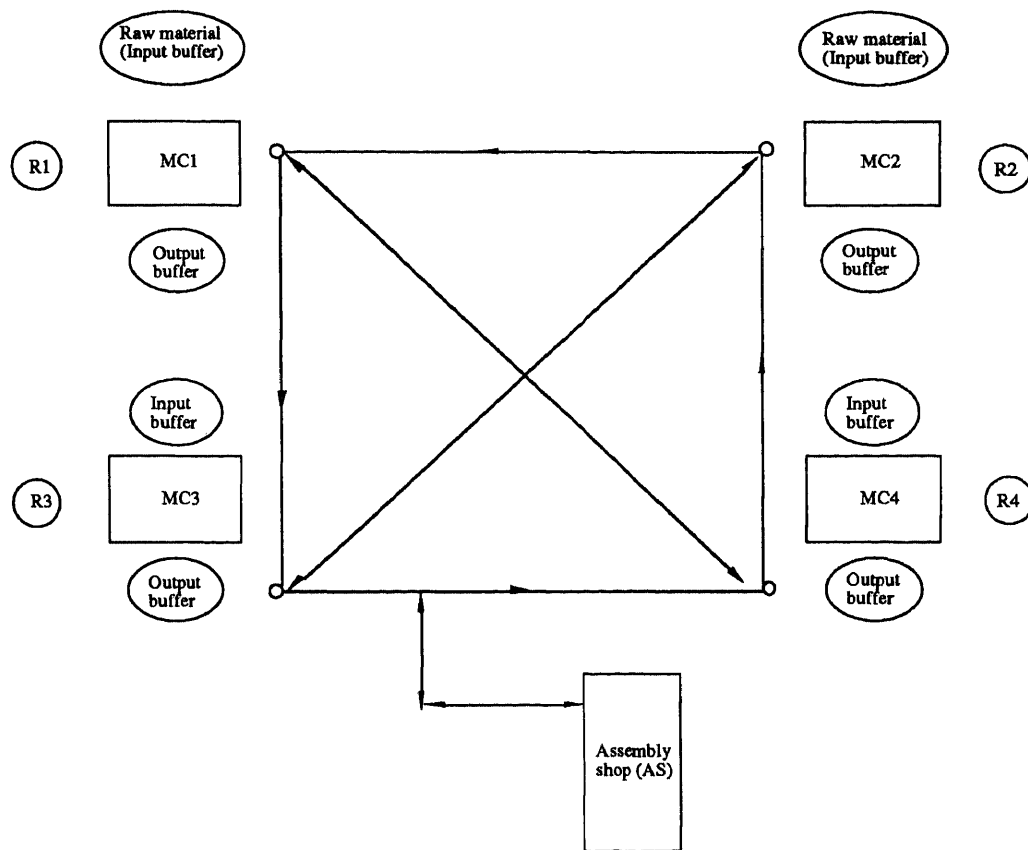


Figure 9.7 The configuration of FMS

The AGV track layout is shown in Fig. 9.7 and can be unidirectional (with an arrow at one end) and bi-directional (arrows at both ends) where the corresponding travel time units are marked. The system is used to produce finished product, PR1 that has been considered in Chapter 4. A machine and a robot constitute one flexible manufacturing cell called "cell" for short.

There is a cell controller responsible for controlling the operations in each cell. There may be one or more PLCs to coordinate the sequencing of different elements present in each cell. Each PLC may control more than one machine, robot, and sensor. It receives signals from sensors and accordingly controls the functions of robots and machines. There is a main controller for the FMS to control the cell controllers and schedule production tasks among cells. During the production if any malfunction/exception occurs at the PLC level (due to the factors such as breakdown of tool and excessive machining temperature) an exception is raised by the PLC and passed to the cell controller. Then the cell controller handles that exception and passes the resume signal to PLC to continue the cell operations. Similarly, if malfunction/exception occurs at the cell controller level (due to the factors, e.g., entering of a new product variety), it raises an exception and sends to the main controller. Then the main controller handles that exception and passes the resume signal to cell controller. There is also a business host which is on the top of the main controller to deal with higher level issues, e.g., production control and corporate policies.

9.3.1. OMT diagram and PNM of the FMS

OMT diagram of the FMS

Fig. 9.8 shows the OMT diagram corresponding to the FMS under investigation. Even though the OMT diagram appears complex, the design shown is believed to be realistic and captures the relevant real world properties of the objects and their functions in this FMS.

For example, this diagram models that the FMS *produces* many finished products, *interacts* with a business host, and *consists of* a main controller, several flexible manufacturing cells, an automated warehouse, and a material handling system. This OMT diagram can be easily extended or modified when the configuration of FMS changes, as shown later.

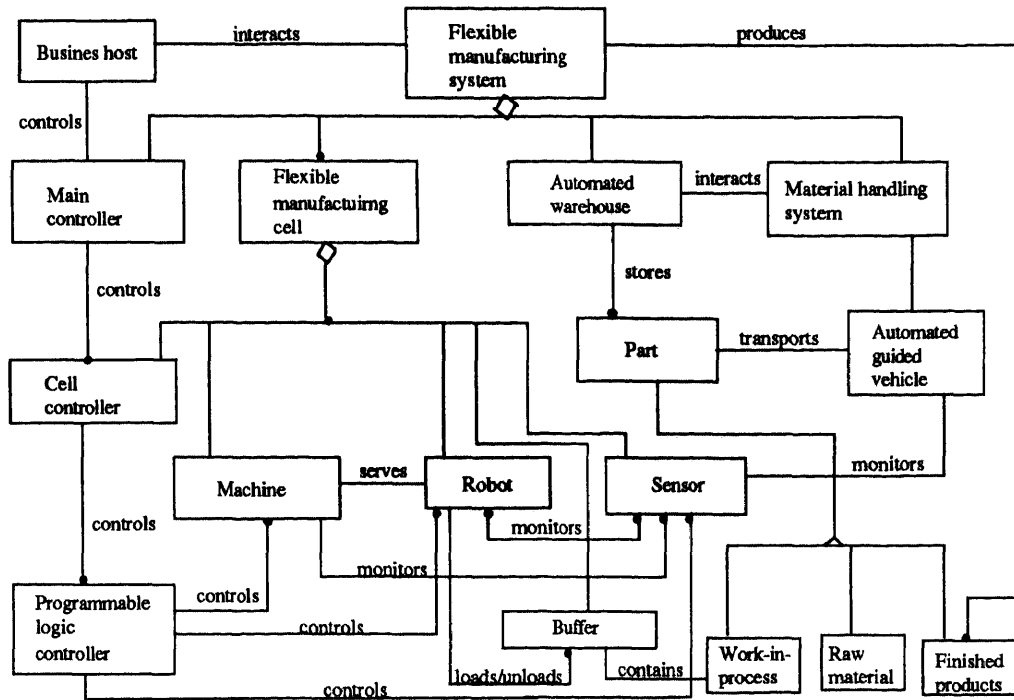


Figure 9.8 OMT diagram of the FMS

PNM of the FMS

To formulate the PNM of FMS, first the PNM corresponding to a machine 1 (MC1) is formulated as shown in Fig. 9.9. The interpretation of its places is listed in Table 9.2.

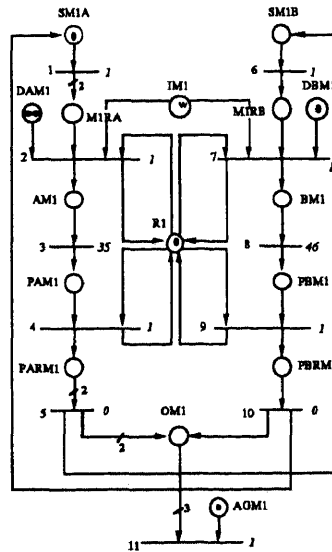


Figure 9.9 PNM of machine 1 (MC1)

Table 9.2 Interpretation of typical places and transitions in PNMs shown in Figs. 9.9 and 9.10

Place	Interpretation
DAM _i (i = 1 and 3)	Demand for Part A on M _i
IM1	Input buffer of MC1 with parts ready to feed MC1
SM1A	MC1 being setup to process Part A
M1RA	MC1 ready to process Part A
R1	R1 ready to load/unload
AM1	Part A loaded on MC1's table and MC1 processing part
PAM1	MC1 finished processing Part A and Part A is being unloaded by R1
PARM1	Part A is ready at the MC1
OM1	Output buffer of MC1 ready with parts A and B
AGM1	AGV at the output buffer of MC1
AM _{ij} (i,j = 1, 2, 3, and 4)	AGV traveling from 'MC _i ' to 'MC _j '
AM _i (i=1)	AGV at the input buffer of 'MC _i '
AMO _i (i = 1, 2, 3, and 4)	AGV at the output buffer of 'MC _i '
A1ASM1	AGV traveling from AS to MC1
Transition	
1,6,14,19,28,33,39,44	Signal for machine setup
2,7,15,20,29,34,40,45	Robot finishes the loading operation
3,8,16,21,30,35,41,46	Completion of part processing
4,9,17,22,31,36,42,47	Robot finishes the unloading operation
5,10,18,23,32,36,43,48	Number of parts as specified in final assembly ready in M's output buffer
11,13,25,27,38,49	AGV starts from MC
12,24,26,50,51,52	AGV reaches its destination

For simplicity, objects like business host, automated warehouse, and sensors are not considered in our models. Here, a bottom-up approach is adopted to construct the PNM of FMS. As the functioning of all machines is almost similar, the same PN modeling methodology can be duplicated to formulate the PNMs of other machine cells. After formulating the PNMs of all cells, they are connected to each other according to the process sequence of parts (given in Chapter 4) as shown in the Fig. 9.10. Observe that transferring of parts in between cells is also modeled in Fig. 9.4. Since, the functioning of all cells is similar, the same PN modeling methodology can be duplicated to formulate the PNMs of other cells in FMS. Figure 9.11 shows the PNM of FMS control system. In the PNMs of Figs. 9.10 and 9.11, some places are pictured as concentric circles. The reasons and motivation for this are discussed later.

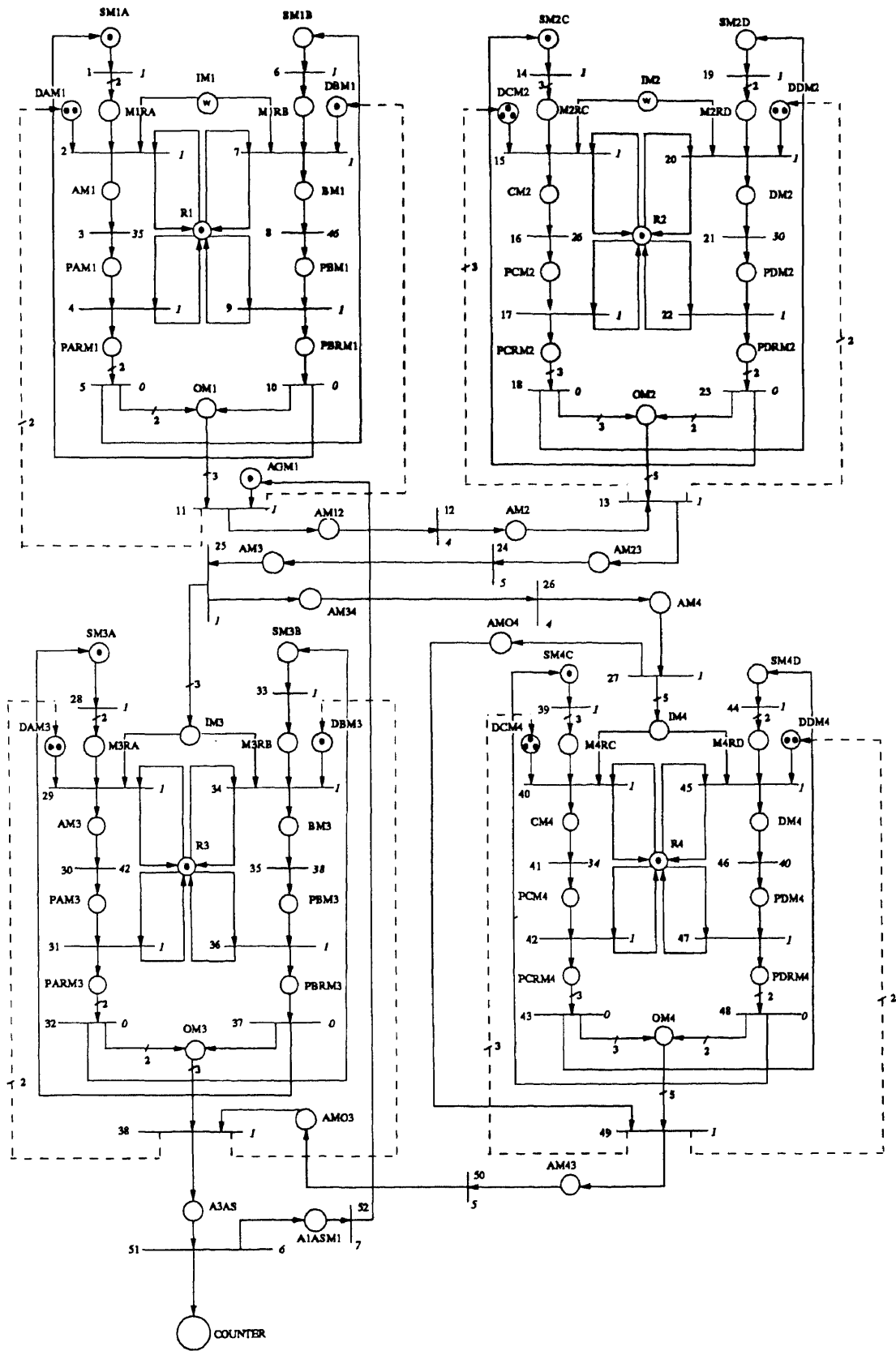
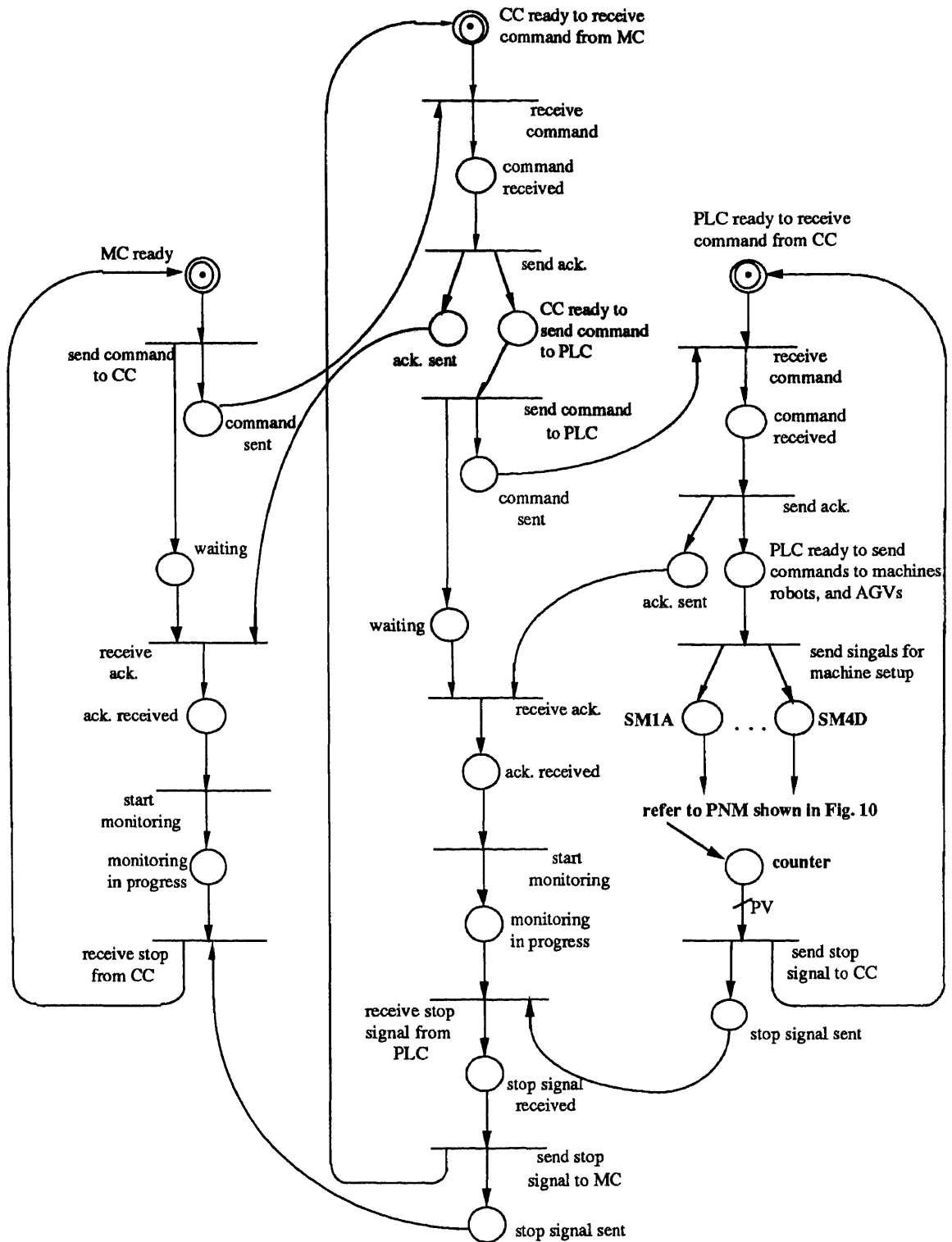


Figure 9.10 PNM of the FMS under push paradigm

NOTE: 1. Transition times are shown for FMS case 3. $\begin{matrix} k & r \\ | & \\ k & k \text{ is transition number} \\ | & \\ r & r \text{ is time units} \end{matrix}$ 4. PLS for A,B,C & D : 2,1,3 & 2
 2. Initial marking is shown 5. N = 1, n = 1



Note: MC --> Main controller; CC --> Cell controller; PLC --> Programmable logic controller; PV --> Production volume

Figure 9.11 PNM for the FMS control system

In PNM's shown in Figs. 9.9 and 9.10, places and transitions represent different states and operations related to each object respectively. It can be observed that these PNM's clearly model the dynamic interactions among the objects in the FMS. While modeling MC3, the PN model of MC1 shown in Fig. 9.9 is reused. In case of modeling MC2 and MC4, the PN model of MC1 is not only reused but also extended to model the machining of a different part variety. In the actual implementation of control software, the PNM corresponding to MC1 (shown in Fig. 9.9) can be treated as a software module and can be reused and extended to produce software modules corresponding to MC2, MC3, and MC4. These modules are then combined to generate the software module corresponding to the FMS. Similarly, the control software corresponding to this FMS can be reused and extended to produce software modules of other systems.

From this example, it can be said that the PN modeling of similar components in FMS supports the concepts of reusability and extendibility which are two essential characteristics of the software generated by OOD methodology (Meyer 1988, Rumbaugh *et al.* 1991). Reusability and extendibility are defined and illustrated in the next section.

Figure 9.11 shows the PNM for the FMS control system for normal production cases (without breakdowns). It models the hierarchical control of the FMS. For simplicity sake, during modeling it is assumed that there is a main controller (M), a cell controller (CC), and a PLC to control the FMS considered. In this PNM, the places and transitions represent different states and operations related to each object respectively. It can be observed that this PNM clearly models the dynamic interactions among the objects in the control system. It can be easily extended to consider more cell controllers and PLCs.

The execution principles of the PNM fulfill the requirements of system operations. For example, in Fig. 9.10, consider "transition 2 (t_2)" that models the activity "Robot finishes the loading operation". In other words, t_2 models an activity: Robot 1 (R1) loads Part A on MC1. The constraints that have to be fulfilled to fire t_2 (to enable the activity modeled by t_2) are modeled by the input places corresponding to t_2 . For example, marked

places: "M1RA" models the condition that "MC1 should be ready to process Part A", "DAM1" there should be demand for Part A, "IM1" there should be raw material needed to produce Part A, finally "R1" should be ready to load the tool on MC1. The time that R1 takes to finish this operation is assumed as 1 time unit and associated with t_2 (shown at the right hand side of t_2 in Fig. 9.10).

From the preceding discussion, objects involved in firing t_2 can be easily recognized as R1, M1, and Part A. A similar discussion can be given for other places and transitions in this PNM, i.e., there is an one-to-one correspondence between the actions in the FMS and transitions, and thus the execution of the PNM precisely specifies the operations involved in FMS. In Figs. 9.10 and 9.11 modeling of exception handling is not shown. However, exceptions do occur sometimes during production. Hence, Fig. 9.12 shows the PNM for exception handling by the main controller (M).

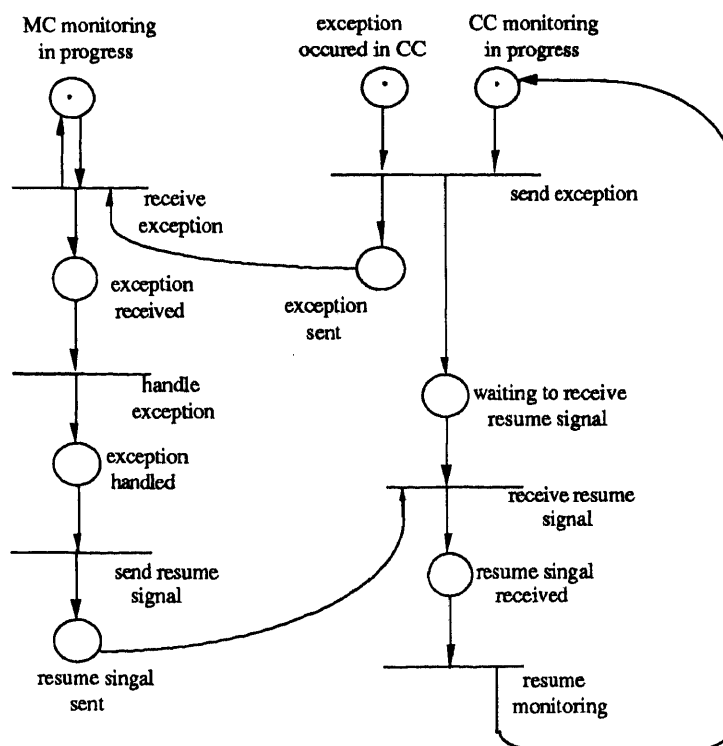


Figure 9.12 PNM of exception handling by main controller

In this PNM, an exception is raised by a cell controller (CC) and handled by the main controller. This net clearly shows the conditions and activities involved in this kind of exception handling. Similarly, another PNM similar to this can be developed to model the exception handling by CC when an exception is raised by a PLC.

Formulating the final PNM by analyzing the performance of FMS

As discussed in the last chapter, the final PNM can be formulated by analyzing it to evaluate the performance of FMS. Estimating the number of AGVs and selecting their routings are important since they affect both the performance and control of FMS. By analyzing the FMS performance and deciding the number of AGVs and their routings, appropriate data structures and operations for AGV objects can be selected in the control software. Another design issue that affects both performance and control of FMS is production lot size (PLS) and moving lot size (MLS). By determining the optimum PLS, signals for changing the setup of machines can be appropriately given during the control of FMS. In the FMS considered, there are two different combinations of PLSs for parts A, B, C, and D namely, 2, 1, 3, 2 and 1, 1, 1, and 1.

In the first combination parts A, B, C, and D are produced in the lot sizes equal to their exact requirement as in the bill of materials of product 1. In the second case, these parts are produced in unit lot sizes. For example, with respect to MC1 and MC2 and based on the process sequences of parts in FMS (shown in Table 4.3), the first combination corresponds to the loading sequence 1A, 1A, and 1B on MC1 and 1C, 1C, 1C, and 1D, 1D on MC2. The second combination corresponds to 1A and 1B on MC1 and 1C and 1D on MC2. It is clear that in the former case, the setup time required to produce one finished product is less compared to the latter. However, the work in process inventory in the former may be more compared to the latter. Further, PLSs may affect the utilization of machines, robots, and AGVs. The quantitative analysis of PNM allows to quantify the influence of PLSs on system performance. For more details on the results of performance evaluation, refer to the Chapter 4.

9.3.2. Complete Structure of Objects with Their Static and Dynamic Relations

After finalizing the PNM, the OMT diagram is combined with it to design the complete structure of object classes with their static and dynamic relations as shown in Fig. 9.13.

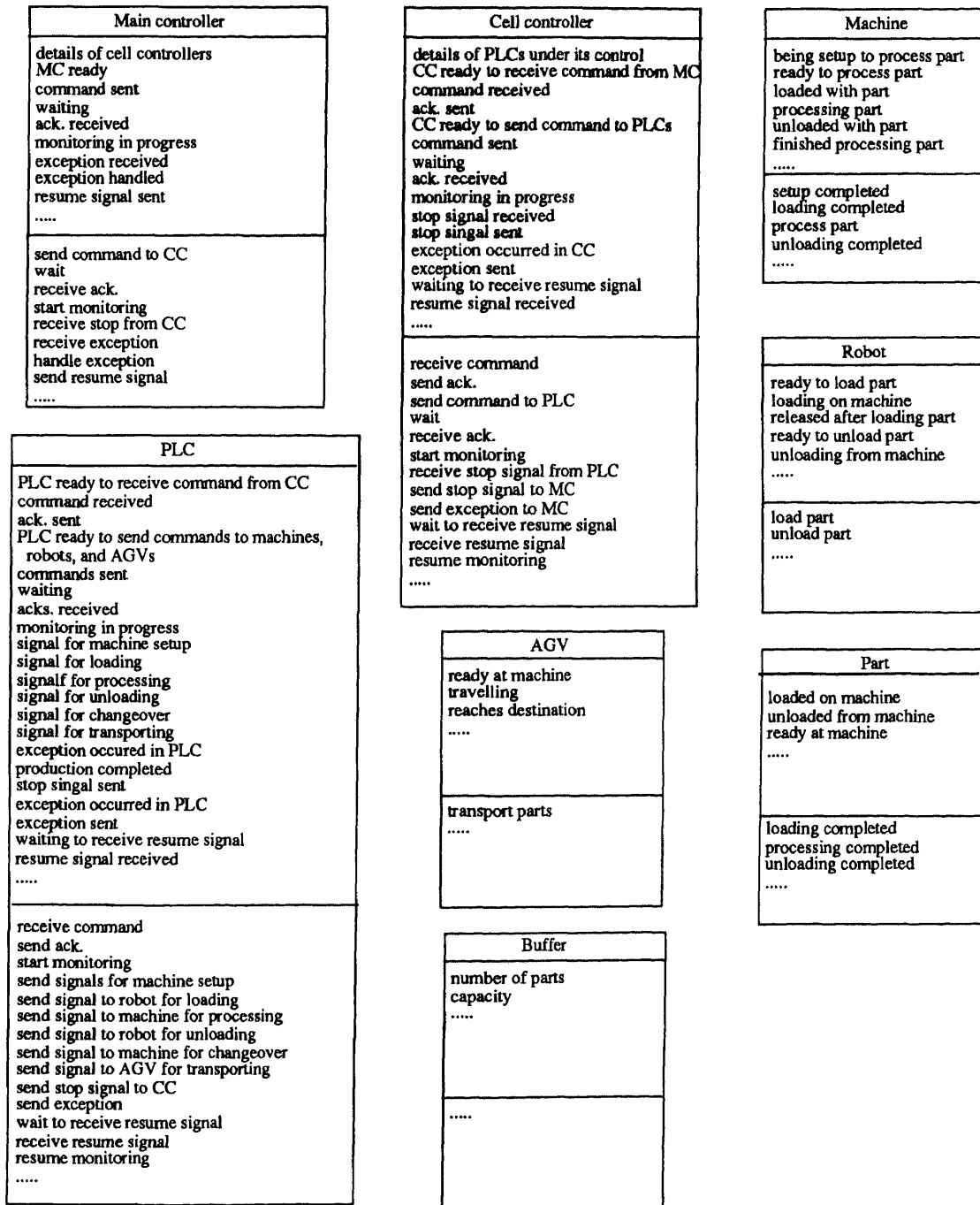


Figure 9.13 Class definitions of important object classes in FMSs

A simple and systematic methodology for selecting objects, their attributes and operations from the PNM is described in this section. As mentioned earlier, PNs can aid to identify the potential objects and their corresponding data structures and operations. Places in PNM aid to identify objects and their data structures, and transitions aid to find operations for objects.

Generally objects are identified when developing the OMT diagram for an FMS. The attributes and operations of the objects can be formulated with aid of its Petri net model. For the FMS example, MC, CC, PLC, machine, AGV, robot, part, and buffer are selected as potential object classes from the OMT diagram. For convenience, those places corresponding to the objects are shown as concentric circles in Figs. 9.10 and 9.11.

The places that represent the intermediate states of objects such as *command_sent* (for MC in Fig. 9.11) and *loaded_with_part* (PAM1 for Machine 1 in Fig. 9.10); and the places that model conditions of processes such as *waiting* and *monitoring_in_progress* (for MC in Fig. 9.11), and *AGV_travelling_from_machine_1_to_machine_2* (AM12 for AGV in Fig. 9.10) aid in selecting the data structures for the objects.

In Figs. 9.10 and 9.11, this type of places are shown as normal circles and aid in selecting the data structures (also called as data attributes) as shown in Fig. 9.13. The values of these attributes are represented by the presence of the tokens in the corresponding places modeling them. Normally, the value of these attributes are of boolean type.

In the PNM transitions represent the activities corresponding to objects such as *send_command_to_CC* and *receive_acknowledgment* (in Fig. 9.10), *AGV_starts_from_machine* (in Fig. 9.11). Transitions corresponding to each object are selected as its operations as shown in Fig. 9.13. It shows the class definitions of important object classes and their data structures and operations. These classes are essential for development of object-oriented control software. A software engineer can understand the logic structure of the whole system by looking at these classes. Hence, these object classes are the fundamental building blocks of an object-oriented software system.

9.3.3. Reusability, Extendibility, and Modifiability of the Design

Reusability and extendibility are two of the important characteristics of the software generated by using OOD (Meyer 1988, Rumbaugh *et al.* 1991) which makes easily adaptable and maintainable software. Reusability is the ability to reuse a module or component developed for a given design, in a new design (Meyer 1988, Rumbaugh *et al.* 1991). To illustrate the reusability, extendibility, and modifiability of the design, consider now that the designer decides modifying FMS as follows:

1. Include a flexible assembly cell (FAC) consisting of two new machines, MC5 and MC6; two new robots, R5 and R6 for pick and place operations, and two part feeders,
2. Include a mobile robot with the additional tracks in FAC to transport material among machines and robots, and
3. Change the operation management strategy of the FMS from push to pull during the production of finished product considered earlier. In other words, machines process parts only when there is a demand.

The new configuration of the FMS is shown in Fig. 9.14. The OMT diagram corresponding to the new FMS is shown in Fig. 9.15. It omits the OMT diagram portion that has not changed in Fig. 9.8. It is noticed that the earlier OMT diagram is reused by extending it to include the elements newly added to the FMS. The third new modification stated above changes the dynamic relations among objects in FMS and may affect the system performance. To accommodate this modification, we redefine the PNM as shown in Fig. 9.16. Observe that the same PNM (Fig. 9.10) used earlier to study the push paradigm is reused to study the pull paradigm with slightly modification. This modification is shown as the dotted arcs modeling the pull paradigm. The system performance can be evaluated again as discussed in Chapter 4. The control software developed earlier for the FMS need not be much changed due to the modularity in OOD. Since the PNMs modeling the FMS corresponding to both push and pull paradigms are basically same, the data

structures of objects need not be changed to include the new specifications of FMS. Hence, this section has briefly illustrated how the proposed design methodology supports reusability, extendibility, and modifiability concepts in developing control software.

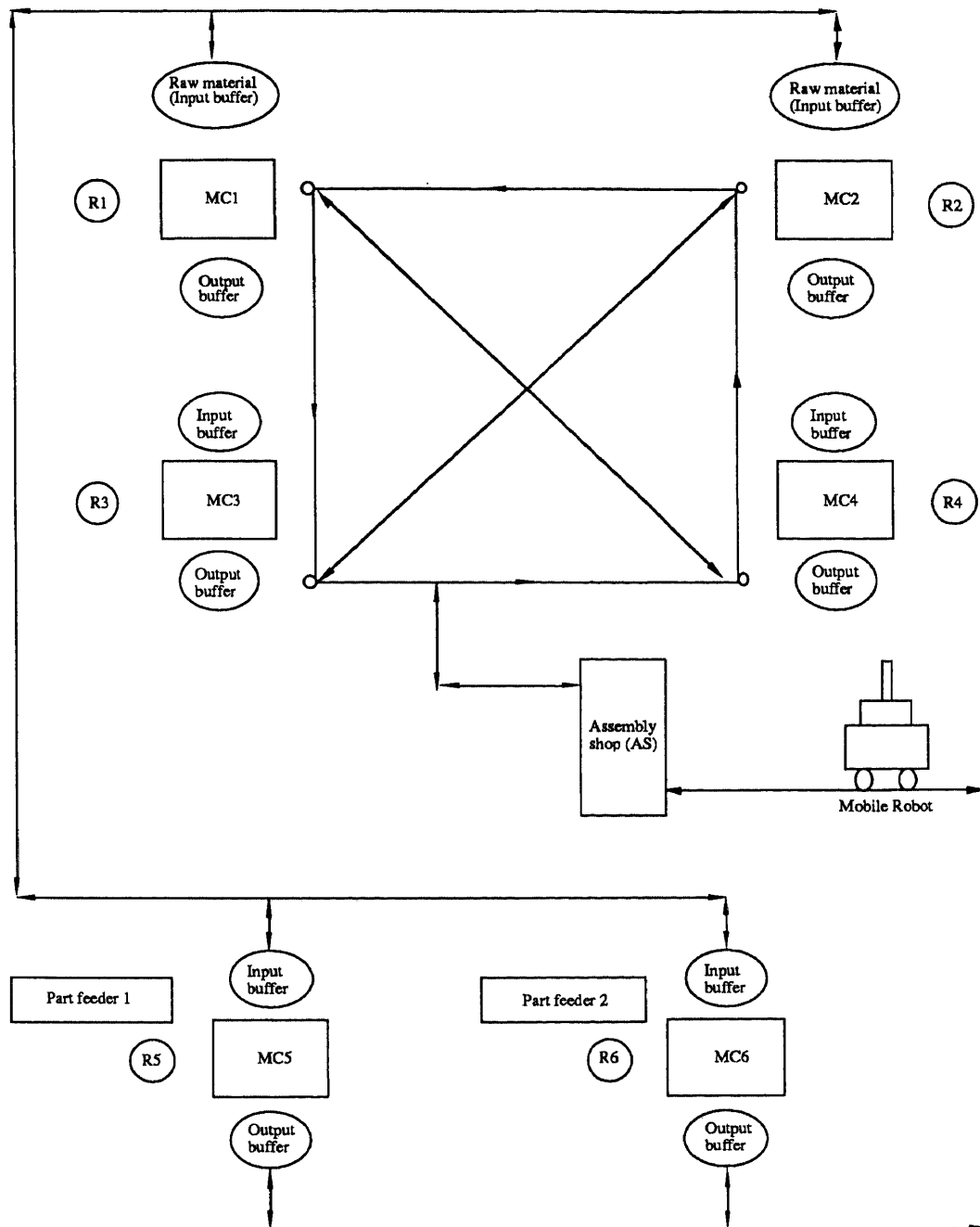


Figure 9.14 The expanded configuration of FMS

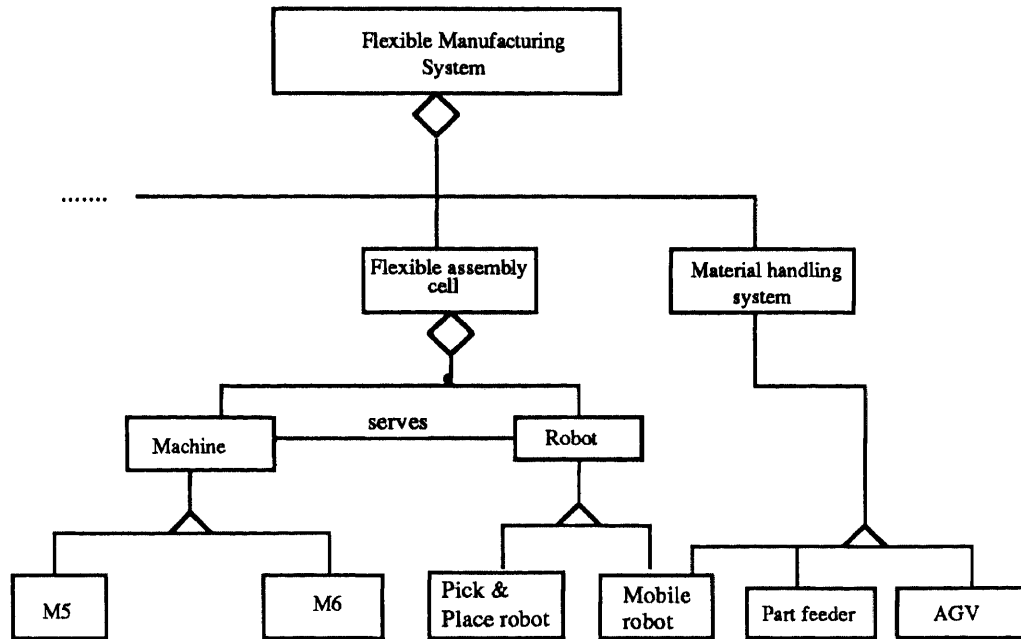


Figure 9.15 OMT diagram for the expanded FMS

9.4. Summary

Development of integrated FMS control software that can be used for planning, scheduling, monitoring, simulation, and control is difficult and hence attracting the growing attention of researchers and practitioners. To ease the task of developing FMS control software, a systematic design methodology is proposed by combining the OOD concepts, OMT diagrams, and PNs. An FMS example is used to illustrate the methodology. OMT diagram for the FMS is developed to find the objects and the static relationships among them. PNM is formulated to study the performance of the system. The PN based method to help identify the data structures and operations of FMS objects is also illustrated. The reusability, extendibility, and modifiability of control software system using this methodology are also illustrated by augmenting the original OMT and PNM to satisfy the new specifications of the FMS. The traditional methods for discrete event dynamic systems in OOD include state/state transition diagrams and event trace/interaction diagrams. They have been proved ineffective for large FMS projects.

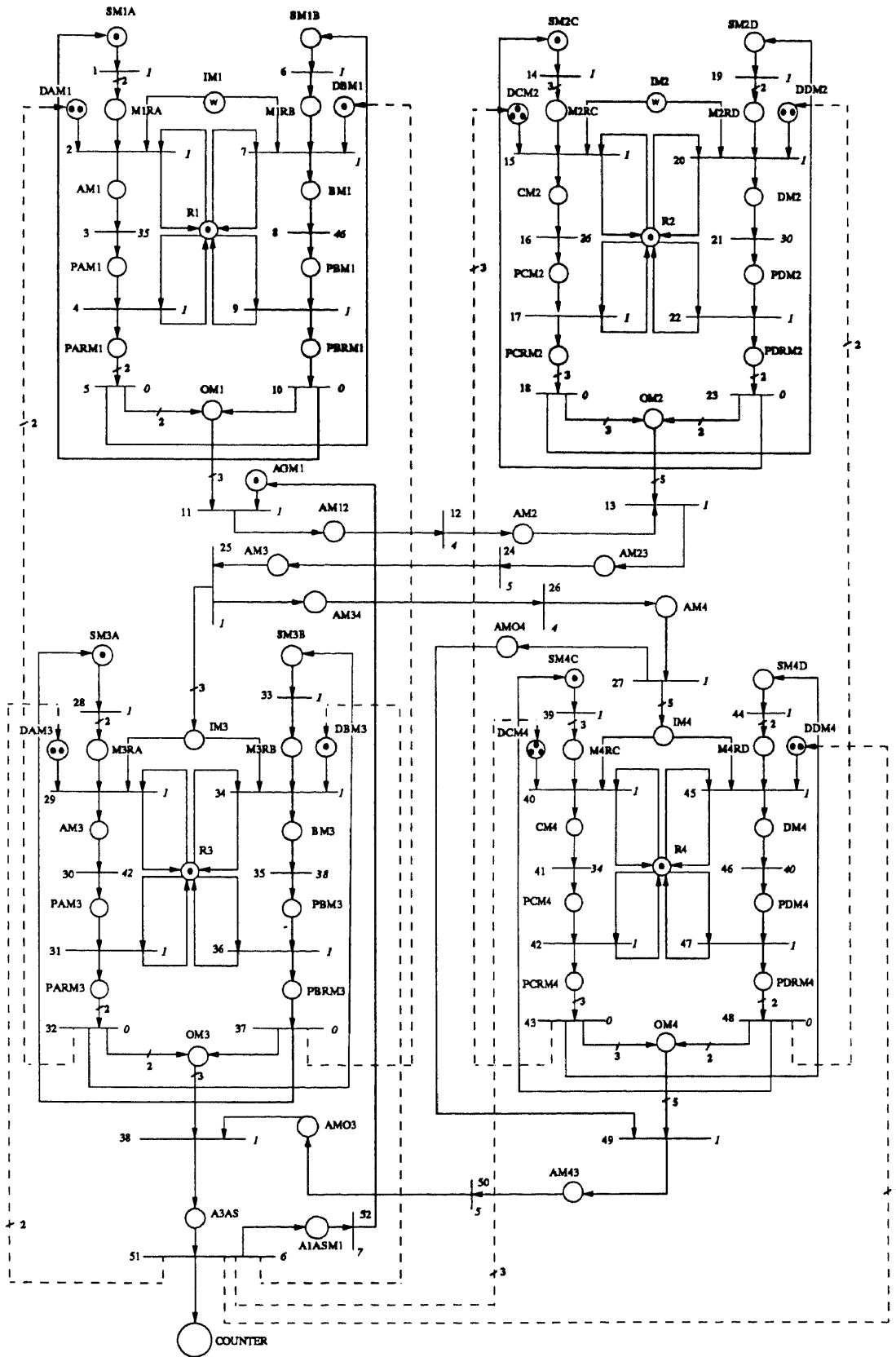


Figure 9.16 PNM of the FMS under pull paradigm

NOTE: 1. Transition times are shown for FMS case 3, $\begin{matrix} k & r \\ | & \\ k & r \end{matrix}$ k is transition number 4. PLS for A,B,C & D : 2,1,3 & 2
 2. Initial marking is shown $\begin{matrix} | & \\ r & r \end{matrix}$ r is time units 5. $N = 1, n = 1$

This chapter emphasized PNs as the dynamic model in OOD and results in the following advantages: 1) By adopting the bottom-up approach of PN modeling, PNs can support two important characteristics of software generated by OOD namely reusability and extendibility; 2) PN models offer a systematic method to identify the data structures and operations of objects in the software system; 3) PNs can be used as an integrated tool to both control the system and analyze the system performance.

The significance of this chapter is two fold. From the OOD point of view, this chapter has embedded PNs as a dynamic modeling tool in an OOD approach to achieve the objectives never before possible, e.g., explicit description of concurrency and synchronization among objects and performance analysis of FMS. From the practice point of view, this chapter has offered an effective systematic methodology to design modifiable, extendible, and reusable control software and helped further establish OOD, OMT diagrams, and PNs for industrial applications.

In the future, OMT diagrams and PNs needs to be extended to deal with such issues as real-time monitoring and fault tolerance, and communication among various objects in FMS. The class definitions of objects presented in this work need to be extended in order to address problems related to material requirement planning, computer aided process planning, and computer aided design. The future studies include standardization of PN techniques in OOD and implement a laboratory system considering the real time control and breakdowns of system components. A bench mark study using traditional methods and proposed method should be useful. The results presented in this chapter represent a good beginning towards the above objective.

CHAPTER 10

CONCLUSIONS

10.1 Contribution

Petri nets are being increasingly studied world-wide by several researchers and industrial practitioners to address a variety of issues related to FMSs starting from their specification, modeling, designing, performance evaluation, and scheduling to real-time control and monitoring. Even though PNs and related technologies have been successfully applied in European and Japanese industries, their acceptance in American industry is rather sluggish. However, by demonstrating the efficacy of PN-aided approaches to solving the related problems in FMS, industrial applications of PNs can be promoted. Motivated by these facts, the present work explores the application of PNs and proposes PNs as a tool and methodology for modeling, simulation, and control in FMSs.

First, this research work makes a survey on FMSs and explores related problems in their implementation. Then, convinced by the suitability of PNs to solve a variety of problems in FMSs, it makes a survey on the application of PNs as an integrated tool and methodology in FMS. Simulation and performance evaluation, breakdown modeling, and discrete event control are found to be three potential areas for exploring both the theory and application of Petri nets. By studying various problems in the above areas with Petri nets, this research makes an attempt to establish PNs as an integrated tool for modeling, simulation, and control of FMSs.

The present work contributes not only to the existing theory of PNs but also to the application spectrum of PNs. In theory, this work proposes extensions to the available classes of PNs and introduces new classes of PNs for modeling breakdown situations and designing discrete event controllers. In practice, it demonstrates the power of PNs, 1) to evaluate and compare the performance of push and pull paradigms in FMSs, 2) to design

reusable, modifiable, and extendible sequence controllers, and 3) to develop object-oriented control software. This research work also gives an insight look at the application of PNs for discrete-event control and answers certain basic questions such as "Why a PN model is better suitable than LLD". Through a practical system, a bench-mark study to compare ladder logic diagrams and PNs for discrete-event control is presented. From the results of this study, the advantages and disadvantages of both LLDs and PNs are explored.

The present work widens the knowledge of PN literature in two ways. First, it will allow engineers and managers who are responsible for the design and the implementation of modern manufacturing systems to evaluate Petri nets for applications in their work. Second, it will provide sufficient breadth and depth to allow development of Petri net-based methods for discrete event control of manufacturing systems. Thus this research will foster further research and applications of Petri nets in aiding the successful implementation of advanced manufacturing systems.

In the following paragraphs the specific contributions of this research in simulation and performance evaluation, breakdown modeling, and discrete-event control are detailed.

10.1.1. Simulation and Performance Evaluation

Earlier research works on PNs have not demonstrated the suitability of PNs for investigating the design and performance analysis issues of FMSs functioning under either *push* or *pull* paradigms. Also, earlier reported works of PNs in FMSs addressing push and pull paradigms have not explicitly presented (i) the differences between PN modeling of push and pull paradigms, and (ii) modeling of production and moving lot sizes. This work investigates the application of PNs for performance evaluation of push and pull paradigms in FMSs using timed PNs by changing different operational parameters in the system operating under *push* and *pull* paradigms.

The performance criteria are the buffer sizes, output rate, utilization of machines, AGVs, and robots. The configuration that results in the minimum buffer sizes, maximum system utilization and output rate is considered as the optimal solution.

The manufacturing system considered is investigated as both FMS and flexible assembly system, by changing the production lot sizes, the number of unique transportation tasks (which decides moving lot sizes), and the number of AGVs for each transportation task. To achieve this, PN models of the system are formulated and analyzed quantitatively through a PN-driven simulation package developed by the author. The analysis shows that in both cases of FMS and flexible assembly system (FAS), the *push* paradigm performed better than the *pull* one for this system. This is because unlike the general notion that only *pull* paradigm results in minimum WIP, *push* may result not only in minimum WIP but also maximum system utilization and output rate for certain configurations and operation parameters. The results may change if some of the system parameters such as processing and/or assembly times change. The same PNMs can be used by associating the new time values with the corresponding transitions modeling such activities. It is concluded that 1) before adopting either *push* or *pull* paradigm, the system should be evaluated with goals to reduce inventories and increase system utilization and output rate, and 2) PNs can be a very useful tool to perform such evaluation.

From the simulation results, many inferences are useful for the operations managers, system designers, manufacturing, industrial and software engineers. For example, the results show that average robot utilization is very low in all the cases studied in this thesis. This is because there is a dedicated robot for each work cell.

10.1.2. Breakdown Modeling

Earlier research studies have not presented the detailed breakdown handling procedures and performance optimization issues for various machine/robot breakdown rates. Furthermore, they considered only breakdowns that arrive before starting of an activity.

However, in the real life situations breakdowns may arrive when an activity is in progress. This restricts their accuracy for analysis of a realistic assembly system.

Thus thesis introduces a new class of tools called Augmented Timed Petri nets (ATPNs) to model breakdown handling in manufacturing systems. ATPNs can model their operations in detail considering the breakdowns of various components. The methodology for formulating the ATPN models is illustrated by considering a flexible assembly system. Also, the application of ATPNs for optimization and design is shown by investigating the optimum number of assembly fixtures for the system under various robot breakdown rates. The methodology proposed in this thesis can be extended to deal with breakdowns of several machines, AGVs, and cell controllers and other production interruptions. ATPNs provide rapid, flexible, and realistic modeling.

10.1.3. Discrete-Event Control

Development of flexible, reusable, and maintainable control software is important to implement advanced industrial automated systems. Traditional methods of using ladder logic diagrams (LLDs) to design sequence controllers are being challenged by the needs in flexible and agile manufacturing systems. On the other hand, PNs are an emerging tool that needs to be established for the industrial discrete-event control of manufacturing systems. A new class of PNs that closely resemble ordinary PNs is of the paramount importance for the development of control software because ordinary PNs are simple and relatively easier to understand.

This work proposes a class of PNs called Real-time PNs (RTPNs) for sequence controller design. A simple and straight-forward procedure is demonstrated to implement them. The detailed differences between earlier classes of PNs that are aimed for control and RTPNs are discussed.

In order to compare RTPNs and LLDs, this thesis identifies design complexity and response time as the criteria. Design complexity is defined and characterized by two factors namely graphical complexity and adaptability to meet changes in control

specifications. By designing and implementing the control of an industrial automated system subject to changing control requirements, LLDs and PNs are compared in terms of a common measure, namely, the number of basic elements, which signify both design complexity and response time.

Motivated by the fact that any sequence controller can be designed by synthesizing the building blocks of logic models, this work also proposed analytical formulas to estimate the number of basic elements to model the most commonly used building blocks of logic modeling by both PN and LLD. Furthermore, a methodology that uses the developed analytical formulas to estimate the total number of basic elements to model a control logic even before physically modeling it using PNs and LLDs is presented. The concepts developed in this chapter are demonstrated by considering several examples of sequence controllers. The examples considered here demonstrate the potential for practical application of the research results.

The methodology presented provides an accurate quantitative comparison of PN and LLD in terms of basic elements. By precluding the need for physically building the controllers by either PN or LLD, this methodology serves as an effective aid for a control engineer to select between PN and LLD even before starting to write the control program. The methodology developed is simple and straightforward to apply. One critical task in the development of PN based controller is to design PN models for given the sequence control specifications. Also, one of the factors for the comprehensive comparison of PNs and LLDs is the availability of standard procedures for designing controllers. There exists systematic design procedures for designing ladder logic diagrams. However, for the large scale application of PNs in industry, there is a need for systematic design procedures for developing PN models.

This thesis presents a simple conversion procedure to formulate PN models from a given logic control specification. The PN model developed using the conversion procedure is simple and easy to understand. The conversion procedure is applied to

design PN models corresponding to the classical control sequences that exist in discrete event control of manufacturing systems. This procedure is illustrated for several types of logic control specifications, single and multi path sequences, with and without repetitive actions. The number of basic elements in the PN model obtained by following the conversion procedure is compared with that obtained previously in this thesis. The comparison clearly indicates that the new PN models have less design complexity.

It is also concluded that while formulating PN models corresponding to sequences with repetitive actions, merging of common places may increase the complexity of the PN. Hence, merging of places in such situations is not recommended. Using the same PN model, both types of systems namely, systems that require sustained actuating signals and the systems that require only a momentary or pulsed actuating signals can be controlled. This can be accomplished by changing the corresponding attributes for transitions as reported in PN based control using Real-time PNs. However, when using ladder logic diagrams, two separate design procedures have to be followed to design two separate diagrams.

Development of integrated FMS control software that can be used for planning, scheduling, monitoring, simulation, and control is difficult and hence attracting the growing attention of researchers and practitioners. The FMS control software should be reusable, modifiable, and extendible to: 1) adapt to changes in the system configuration and specifications, and 2) to deal with a complex shop-floor system which often consists of numerous similar components. A systematic design methodology is obviously needed to develop such FMS control software.

To ease the task of developing FMS control software, this thesis proposes a systematic design methodology by combining the OOD concepts, object modeling technique diagrams (OMT) diagrams and PNs. Its necessity and value through the use of OOD as a design methodology, OMT diagrams as a static modeling tool, and PNs as a dynamic modeling and performance tool is demonstrated. An FMS example is used to

illustrate the methodology. OMT diagram for the FMS is developed to find the objects and the static relationships among them. PNM is formulated to study the performance of the system. The PN based method to help identify the data structures and operations of FMS objects is also illustrated. The reusability, extendibility, and modifiability of control software system design using this methodology are also illustrated by augmenting the original OMT and PNM to satisfy the new specifications of the FMS.

This thesis emphasizes PNs as a dynamic model in OOD and illustrated the following advantages: 1) by adopting the bottom-up approach of PN modeling, PNs can support two important characteristics of software generated by OOD namely reusability and extendibility; 2) PN models offer a systematic method to identify the data structures and operations of objects in the software system; and 3) PNs can be used as an integrated tool to both control the system and analyze the system performance.

10.2. Limitations

The concepts and approaches presented in this work are implemented only partially . The application of RTPNs and its comparison with LLDs is practically implemented using the facilities at The Robotics Center, Florida Atlantic University, Boca Raton. The other concepts in this work are not practically implemented. However, integration and implementation of all the concepts proposed in this work through a practical system is an interesting task. In order to focus the objectives of this research, several assumptions are assumed while modeling, simulating, and controlling the systems considered in the examples. However, those assumptions may not be always true and may not allow to draw general conclusions on the investigations presented. The specific limitations of this research are presented below:

1. During the performance evaluation of FMS and flexible assembly system, a dedicated robot is assumed. However, this not only requires huge investment cost but also results in under-utilization of robots. Also, the reported values of

utilizations of system elements are generally low. Hence, an attempt to increase these utilizations by changing the system parameters should be made.

2. While comparing the performance of push and pull paradigms, machine setup times are assumed to be sequence independent. However, this may not be true when a machine needs to produce several part varieties.
3. The delays due to the traffic of AGVs are assumed to be negligible in this thesis. However, in an FMS employing several AGVs, system performance would be affected due to the AGV waiting times at control zones.
4. In this work, comparison between push and pull paradigms is done by considering a particular system. Hence, it is not possible to draw general conclusions depending upon the results provided to select between push and pull.
5. Issues related to tool handling are not modeled in this work because tools are assumed to be always available at machine. However, detailed modeling of tool handling procedures is essential for realistic system modeling.
6. While studying the performance of push and pull paradigms, breakdown of machines, robots, AGVs, and tools is not assumed.
7. In this work, each workstation is served by a dedicated robot. This is not true in practice as it requires huge investment and result in under-utilization of robots.
8. While modeling the system with ATPNs, each robot is assumed to have a standby robot. This increases the investment cost and result in under-utilization of robots.
9. During modeling a system with ATPNs, the values of repair time and change-over time for all robots are assumed to be the same. This may not be true because repair time depends upon the type of robot and change-over time on the function of robot.

10. When the system contains several components, the size of ATPN models may grow.
11. In modeling of the assembly system, the assembly sequence is assumed to be sequential. However, in a more complicated flexible assembly system part flow among assembly stations may be random.
12. Control of a system through digital input/output interface is only one of the several implementation methods. Different implementation methods of RTPN based control are not illustrated. The uniqueness of the output attributes of transitions should be maintained carefully. For e.g. the second attribute for transition, say 8 can be obtained by many ways, e.g. $(2^4 - 2^3)$ or (2^3) . In the first case, channel 4 is activated and channel 3 is deactivated. In the second case, channel 3 is activated. So, to avoid this kind of confusion, two bits may need to be checked. The first bit should check whether the state of a channel should be changed or not. The second attribute does the actual activation or deactivation. It is like checking a parity.
13. Theoretical analysis of ATPNs to check the properties of liveness, boundness, and reversibility are not studied.
14. RTPNs are particularly more suitable at the lowest control level of FMSs as they do not have constructs to model intercommunication among workstations present in a system. For example, they can not recognize the different part varieties entering the cell.
15. Even though, RTPNs and ATPNs can be combined for the real-time control of a system with breakdowns, such procedure is not clearly illustrated.
16. The software classes developed for the FMS are not implemented.

17. During the development of OOD based control software, class definitions of objects presented in this work did not address problems related to material requirement planning, computer aided process planning, and computer aided design.

10.3. Further Research

In order to eliminate the limitations of this work, the following guidelines are given for further research:

1. During the performance evaluation of FMS and flexible assembly system, in order to avoid the huge investment costs and increase the utilization of robots, the possibility of sharing a common robot between two work cells should be studied further. However, scheduling of robot movements should be done to prevent from impairing the performance of other system elements.
2. Sequence dependent setup times have to be considered while comparing the performance of push and pull paradigms.
3. By modeling a tandem FMS, delays that arise due to traffic congestion can be avoided. Furthermore, grouping of machines in such systems and comparing the performance of push and pull paradigms will be an interesting topic for further research.
4. Analyzing the performance of push and pull paradigms considering several systems will provide suggestions for selecting between push and pull. Also, hybrid push-pull paradigms should be studied.
5. The simulation studies presented in this work can be extended by modeling tool handling procedures such as tool delivery, tool loading, and tool sharing among machines. Such studies will provide realistic estimates of system performance under more dynamic conditions that exist at the shop-floor.
6. Breakdown of machines, robots, AGVs, and tools has to be considered while comparing the performance of push and pull paradigms.

7. This research can be extended to study some important issues such as robot scheduling during breakdowns when only a single robot exists as a standby to all three robots, system performance when several product varieties are produced simultaneously in the system with random routing of parts, and cost consideration for standby robots and breakdown handling.
8. While modeling a system with ATPNs, the effect of random distributions of repair and change-over times on the system performance can be investigated by associating various time values to the transitions modeling repair and change-over activities.
9. When modeling a complex system, *colored PNs* can be combined with the principles of ATPNs to formulate concise graphical models.
10. Modeling of a system with random part flow among workstations may provide new insights into the design of optimum number of assembly fixtures.
11. Theoretical analysis of ATPNs to check liveness, boundness, and reversibility should be done.
12. Illustrating the use of RTPNs for various implementation methods will expose the advantages and disadvantages of RTPNs.
13. RTPNs can be extended by adding more attributes to places and transitions in order to control complex hierarchical manufacturing systems that use advanced communication protocols and several computers for control.
14. The method of integration of RTPNs and ATPNs for the real-time control of a system with breakdowns can be demonstrated using a practical system in order to realize the benefits of ATPNs and RTPNs. In such cases, the transition modeling the occurrence of breakdown is not associated with random breakdown times. Instead, the sensors/limit switches that recognize the breakdown in the system are modeled as input places for this transition. When there is a breakdown in the system, these places get tokens and thus

immediately fire the transition, modeling the occurrence of breakdown. Again, while modeling the emergency stop, instead of using inhibitory arcs in RTPNs, the concept of *deactivating transitions* from ATPNs can be applied.

15. Micro-controllers based on RTPNs can be developed in order to develop adaptable discrete-event controllers that can be easily reused, modified, and extended. This will foster the industrial application of PNs.
16. Other factors which have impact on the comparison and selection of LLDs and PNs should also be explored in the future work. For example, similar to graphical complexity, irrespective of the implementation scheme an effort to compare the response time in terms of response time complexity should be made. Furthermore, comparison of the response time of PN and LLD on the same hardware using different implementation schemes would be useful to commercially develop and market PN based controllers.
17. In the future, OMT diagrams and PNs needs to be extended to deal with such issues as real-time monitoring and fault tolerance, and communication among various objects in FMS.
18. The software classes developed for the FMS should be implemented using such languages as Ada, Smalltalk, and C++ to select the best candidate for the development of OOD based-control software.
19. During the development of OOD based control software, class definitions of objects presented in this work need to be extended in order to address problems related to material requirement planning, computer aided process planning, and computer aided design.
20. The future studies should consider standardization of PN techniques in OOD and implement a laboratory system considering the real time control and breakdowns of system components.

APPENDIX A

SOFTWARE PACKAGE TO EXECUTE TPN AND ATPN

This Appendix contains the source code for executing Timed Petri net and Augmented-timed Petri net models.

Source code

```
-- PROGRAM FOR EXECUTING TIMED-PETRI NET MODEL AND
AUGMENTED-TIMED PETRI NET MODEL.
```

```
-- HARDWARE REQUIRED: Worstation/terminal
```

```
-- SOFTWARE REQUIRED: ADA Compiler in VMS operating system.
```

```
with TEXT_IO;
```

```
procedure TAIWAN is
```

```
use TEXT_IO;
```

```
package INT_IO is new INTEGER_IO(INTEGER);
```

```
use INT_IO;
```

```
Inpt,OTPT: FILE_TYPE;
```

```
type MATRIX is array (1..99,1..99) of INTEGER; --
```

```
type ROW_MATRIX is array (1..99) of INTEGER; --
```

```
type MAT is array (1..100) of INTEGER;
```

```
type SIMP is array (1..2) of INTEGER;
```

```
type SIM is array (1..10) of SIMP;
```

```
INF,OTF: MATRIX;
```

```
DT : SIM;
```

```
F,FF,R,M,D,DD,ATM: ROW_MATRIX; --D(i) transition duration
```

```
--R(i) remaining time of a transition
```

```
--ATM(i) active time of a transition
```

```
A,AX,RA: MAT; --A(i) transition to be disabled
```

```
NT,NDT,NP,T,X,Q,MIN,TMIN,COUNT,CT,H,PT,BX,BY : INTEGER; -- x No. of
transitions to be disabled
```

```
-- COUNT time the system to be simulated
```

```
procedure ENABLED_TRANSITIONS (INM: in MATRIX; Z: out ROW_MATRIX) is
begin
```

```
for i in 1..NT loop
```

```
for j in 1..NP loop
```

```
if (M(J) -abs(INM(I,J))) < 0 then
```

```
GOTO SPOT;
```

```
end if;
```

```

    end loop;

    if DD(i) > 0 then
        z(i):=0;
    else
        z(i):=1;
    end if;
    <<SPOT>> null;

end loop;

end;

procedure CONFLICT (DX: in ROW_MATRIX; Q1:out INTEGER) is
    KJ, KK: INTEGER;
    C: array(1..NT) of INTEGER;

begin
    KK:=0;
    KJ:=0;
    Q1:=0;
    for I in 1..NT loop
        if DX(I) > 0 then
            KK:= KK + 1;
            C(KK):=I;
        end if;
    end loop;

    for I in 1..KK-1 loop

        for J in I+1..KK loop
            for K in 1..NP loop
                if INF(C(I),K)>=1 and INF(C(J),K)>=1 then
                    NEW_LINE;
                    PUT("TIME AT CONFLICT:");
                    PUT(T,7);
                    NEW_LINE;
                    PUT(OTPT,"TIME AT CONFLICT: ");
                    PUT(OTPT,T,7);
                    PUT(OTPT, " ");
                    PUT("CONFLICT :");
                    PUT(C(I),2);
                    PUT(OTPT,"CONFLICT: ");
                    PUT(OTPT,C(I),2);
                    PUT( " ");
                    PUT(OTPT, " ");
                    PUT(C(J),2);
                    PUT(OTPT, C(J),2);
                    PUT( " ");
                    PUT(OTPT, " ");
                    NEW_LINE(OTPT,1);
                end if;
            end loop;
        end loop;
    end loop;
end;

```



```

        KJ:=KJ+1;
        Q1:=KJ;
        GOTO PLACE;
    end if;
end loop;
<<PLACE>> null;
end loop;

end loop;

end;
```

procedure MINIMUM (DY: in out ROW_MATRIX; mini:in out INTEGER;
tmini: in out integer;rr:in out MAT;hh:in out INTEGER) is

```

TX,TY: array (1..NT) of INTEGER;
L : INTEGER;
tmn,mn:integer;
begin
```

```

    L:=0;
    hh:=0;
    for i in 1..100 loop
        rr(i):=0;
    end loop;
```

```

    for i in 1..NT loop

        if DY(i) > 0 then
            L:= L + 1;
            TX(L):=DY(i);
            TY(L):=I;
        end if;
```

```

    end loop;
```

```

    tmn:= TX(1);
    for i in 1..L loop
```

```

        if TX(i) <= tmn then
            tmn:=TX(i);
            mn:=TY(i);
        end if;
```

```

    end loop;
    tmini:=tmn;
    mini:=mn;
```

```

    for j in 1..L loop
```

```

        if TX(j)=tmn then
            hh:=hh+1;
```

```

        rr(hh):=TY(j);
    end if;

end loop;

end;

procedure NEW_MARKING (MM: in out ROW_MATRIX) is

    k:integer;
    b: array (1..k) of integer;

begin

    k:=0;

    for i in 1..NT loop
        if F(i) > 0 then
            k:=K + 1;
            b(k):=i;
        end if;
    end loop;

    for i in 1..k loop
        for j in 1..NP loop
            if INF(b(i),j) > 0 then
                MM(j):=MM(j) - INF(b(i),j);
            end if;
        end loop;
    end loop;

end;

begin

    OPEN(Inpt,IN_FILE,"indat");
    create(OTPT,OUT_FILE,"OUTDAT");
    GET(inpt,NT);
    GET(inpt,NP);
    PUT("NUMBER OF TRANSITIONS: ");
    PUT(NT,2);
    PUT(OTPT,"NO: OF TRAN: ");
    PUT(OTPT,NT,2);
    NEW_LINE; NEW_LINE(OTPT,1);
    PUT("NUMBER OF PLACES: ");
    PUT(NP,2);
    PUT(OTPT,"NO: OF PLACES: ");
    PUT(OTPT,NP,2);

```

```

NEW_LINE;
NEW_LINE(OTPT,1);

for i in 1..99 loop
  for j in 1..99 loop
    inf(i,j):=0;
    otf(i,j):=0;
  end loop;
end loop;

BX := 0;BY := 0;

for i in 1..10 loop
  DT(i)(1) := 0; --(transition generating deactivation pulse)
  DT(i)(2) := 0; --(transition to be deactivated)
end loop;

for i in 1..NT loop
  for j in 1..NP loop
    GET(Inpt,INF(i,j));

  end loop;
end loop;

for i in 1..NT loop
  for j in 1..NP loop
    GET(Inpt,OTF(i,j));
  end loop;
end loop;

PUT("TRANSITION DURATIONS: ");
PUT(OTPT,"TRANSITION DURATIONS : ");

for i in 1..NT loop
  GET(Inpt,D(i));
  PUT(D(i),7);
  PUT(",");
  put(otpt,D(i),7);
  put(otpt,",");
  if (i mod 10 = 0) then
    NEW_LINE;
    NEW_LINE(OTPT,1);
  end if;
end loop;

NEW_LINE; NEW_LINE(OTPT,1);
PUT("INITIAL MARKING: ");
PUT(OTPT,"INITIAL MARKING: ");

for i in 1..NP loop
  GET(Inpt,M(i));
  PUT(M(i),3);PUT(",");
  PUT(OTPT,M(I),3);PUT(OTPT,",");

```

```
    if (i mod 10 = 0) then
      NEW_LINE;
      NEW_LINE(OTPT,1);
    end if;
end loop;
```

```
GET(Inpt,NDT);
for i in 1..NDT loop
  GET(Inpt,DT(i)(1));
  GET(Inpt,DT(i)(2));
end loop;
```

```
NEW_LINE;
PUT("HOW LONG THE SYSTEM HAS TO BE SIMULATED: ");
NEW_LINE;
PUT("GIVE TIME UNITS : ");
GET(COUNT);
PUT(OTPT, "TIME OF SIMULATION: ");
PUT(OTPT, COUNT,7);
NEW_LINE(OTPT,1);
NEW_LINE;
  PUT("COMING UP TO HERE:");
```

```
PUT("GIVE BX&BY ");
GET(BX);
GET(BY);
put(otpt,"BX");
PUT(OTPT,BX,5);
NEW_LINE(OTPT,1);
PUT(OTPT,"BY");
PUT(OTPT,BY,5);
NEW_LINE(OTPT,1);
```

```
for i in 1..99 loop
  DD(i):=0;
  R(i):=0;
  F(i):=0;
  FF(i):=0;
  ATM(i):=0;
end loop;
```

```
T:=0;
CT:=0;
min:=0;
tmin:=0;
H:=0;
PT:=0;
```

```
for i in 1..10 loop
  RA(i):=0;
```

```

end loop;

<<dot>> for i in 1..nt loop
    f(i):=0;
    ff(i):=0;
end loop;

ENABLED_TRANSITIONS(INF,F);

for i in 1..NT loop
    if F(i) > 0 then
        FF(i):= D(i) + T;
    end if;
end loop;

for i in 1..NT loop
    DD(i):=DD(i) + FF(i);
end loop;

--    NEW_LINE;
--    PUT("TIME: ");
--    PUT(T,7);
--    NEW_LINE;
--    NEW_LINE;
--    PUT("MARKING: ");
--    for i in 1..NP loop
--        PUT(M(I),3);
--        PUT(",");
--    end loop;

--    NEW_LINE;

CONFLICT(DD,Q);

if Q > 0 then
    PUT("GIVE NUMBER OF TRANSITIONS TO BE DISABLED : ");
    GET(X);
    NEW_LINE;
    PUT("GIVE TRANSITONS TO BE DISABLED:");
    PUT(OTPT,"TRANSITIONS DISABLED ARE:");

    for i in 1..X loop
        GET(A(i));
        PUT(OTPT,A(I),2);
        PUT(OTPT, ",");
    end loop;
    PUT(OTPT, " ");

    for i in 1..X loop
        DD(A(i)):=0;
        F(A(i)):=0;

```

```

    end loop;
end if;

if T > BX then
    BX := BX+BY;
--    PUT("F-VECTOR : ");

    PUT(OTPT, "TIME: ");
    PUT(OTPT, T, 7);
    NEW_LINE(OTPT,1);

    PUT(OTPT, "MARKING: ");

    for i in 1..NP loop
        PUT(OTPT,M(I),3);
        PUT(OTPT,",");
        if (i mod 10 = 0) then
            NEW_LINE;
            NEW_LINE(OTPT,1);
        end if;
    end loop;
    NEW_LINE(OTPT,1);

    PUT(OTPT,"F-VECTOR : ");

    for i in 1..NT loop
--        PUT(F(i),2);
--        PUT(OTPT,F(i),2);
--        PUT(",");
--        PUT(OTPT,",");
        if (i mod 10 = 0) then

            NEW_LINE(OTPT,1);
        end if;
    end loop;
    NEW_LINE(OTPT,1);
--    PUT("R-FUNCTION: ");
    PUT(OTPT,"R-FUNCTION: ");
    for i in 1..NT loop
--        PUT(R(i),2);
--        PUT(OTPT,R(I),2);
--        PUT(",");
--        PUT(OTPT,",");
        if (i mod 10 = 0) then

            NEW_LINE(OTPT,1);
        end if;
    end loop;
    NEW_LINE(OTPT,1);
    PUT(OTPT,"ACTIVE TIME VECTOR: ");
    for i in 1..NT loop

```

```

    PUT(OTPT,ATM(i),5);
    PUT(OTPT,"");
    if (i mod 10 = 0) then
        NEW_LINE(OTPT,1);
    end if;
end loop;
NEW_LINE(OTPT,1);

end if;

for i in 1..nt loop
    if F(i) = 0 and R(i) = 0 then
        PUT(" ");
    else
        GOTO CONT;
    end if;
end loop;
-- NEW_LINE;
PUT("*****DEAD STATE*****");
PUT(OTPT,"*****DEAD STATE*****");
GOTO POINT;

<< CONT >>  CT := CT + 1;

    if T > COUNT then
        goto POINT;
    end if;

MINIMUM(DD,min,tmin,ra,h);
PT:=t;
t:=tmin;

for i in 1..h loop
    DD(ra(i)):=0;
    ATM(ra(i)):=ATM(ra(i))+tmin-PT;
end loop;
for i in 1..NT loop
    R(i):=0;
end loop;

for i in 1..NT loop
    if DD(i) > 0 then
        R(i):= DD(i) - T;
        ATM(i):=ATM(i)+tmin-PT;
    end if;
end loop;

NEW_MARKING(M);

for j in 1..h loop
    for k in 1..NDT loop
        if ra(j) = DT(k)(1) then
            DD(DT(k)(2)) := 0;
            R(DT(k)(2)) := 0;

```

```
        end if;
    end loop;
end loop;

for j in 1..h loop
    for i in 1..NP loop
        M(i):=M(i) + OTF(ra(j),i);
    end loop;
end loop;

GOTO DOT;

<<POINT>> NEW_LINE;

PUT("EXECUTION TERIMINATED: ");
PUT(OTPT,"EXECUTION TERIMINATED");

CLOSE(Inpt);
CLOSE(OTPT);

end;
```


APPENDIX B

SOFTWARE PACKAGE TO EXECUTE RTPN

This Appendix contains the source code for executing a Real-time Petri net (RTPN). In order to illustrate its usage, an example of RTPN is presented. The sample input to run this PN and the output of this program are also given.

Source code

```
/*  
*****  
THIS PROGRAM IS AN INTEGRATED SOFTWARE SOLUTION FOR BOTH  
SIMULATION AND CONTROL OF AN INDUSTRIAL AUTOMATED SYSTEM.  
*****  
*/
```

HARDWARE REQUIRED: IBM compatible PC and digital input/output interface

SOFTWARE REQUIRED: Borland C

The program contains following modules with their functions explained below:

NAME OF THE MODULE	PURPOSE OF THE MODULE
1. Read_petri_net	Stores the structure of the Petri Net
2. Enabled_transitions	Gives an output vector containing the transitions that can be fired
3. Conflict	Resolves the conflict between two enabled transitions sharing common place
4. Minimum	Gives the number of transition(s), and identity of transition(s) that has (have) minimum firing duration
5. New_marking	It reads the status of inputs and removes tokens from corresponding places when the transitions is fired
6. Main	It is the main program that coordinates the functions of all the above modules and send output signals for relays

```
*****/
```

```

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <math.h>
#include <time.h>

FILE *input;
FILE *output;
FILE *fp;

typedef int matrix[99][99];
typedef int row_matrix[99];
typedef int mat[100];
typedef int simp[2];
simp sim[10];
simp dt[10];
row_matrix f,ff,r,m,d,dd,atm;
matrix inf,otf;
int i,j,k;
mat a,ax,ra;
int nt,ndt,np,t,x,q,min,tmin,count,ct,h,pt,bx,by;
char input_file[20], output_file[20];
int limit_bit_num[99];
int output_value[99];
int input_channel[16],output_channel[16],output_array[16],input_array[16];
long num,num1;
long end_num;
int emergency_stop;

extern int digital_output(long num);
extern int digital_input(long *adl);
extern int binary_decimal(long *num,int ary1[16]);
extern int decimal_binary(long num,int ary1[16]);

int read_petri_net();
int enabled_transitions(matrix inm, row_matrix z);
int conflict (row_matrix dx, int q1);
int minimum(row_matrix dy, int mini, int tmini, mat rr, int hh);
int new_marking (row_matrix mm);

/*int read_petri_net(int inf[99][99],int otf[99][99], int dt[10], int d[99],
int m[99])
*/
int read_petri_net()
{
    printf("\nPlease enter the input file : ");
    scanf("%s", &input_file);
    printf("\nPlease enter the output file : ");

```

```

scanf("%s", &output_file);

input=fopen(input_file, "r");
output=fopen(output_file, "a");
fscanf(input,"%d", &nt);
printf("Number of transitions are: %d\n", nt);
fprintf(output, "Number of transitions are: %d\n", nt);

fscanf(input,"%d", &np);
printf("Number of places are: %d\n", np);
fprintf(output, "Number of places are: %d\n", np);

for (i=1; i<=99; i++)
    for (j=1; j<=99; j++)
        inf[i][j]=0;
        of[i][j]=0;

bx=by=0;

for (i=1; i<=10; i++)
{
    dt[i][1]=0;
    dt[i][2]=0;
}

for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        fscanf(input,"%d", &inf[i][j]);

printf("This is input matrix: \n");

/*for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        {
            printf("%d", inf[i][j]);
            fprintf(output,"%d", inf[i][j]);
            if (j==np) {printf("\n");
            fprintf(output, "\n");}
        }
*/

fprintf(output, "This is input matrix: \n");
for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        {
            fprintf(output, "%d", inf[i][j]);
            if ((j%10) == 0)
                fprintf(output, " ");
            if (j==np)
                fprintf(output, "\n");
        }
}

```

```

for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        fscanf(input,"%d", &otf[i][j]);

/*printf("This is output matrix: \n");
fprintf(output,"This is output matrix: \n");
for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        {
            printf("%d", otf[i][j]);
            fprintf(output,"%d", otf[i][j]);

            if (j==np) {printf("\n");fprintf(output,"\n");}
        }
*/

fprintf(output,"This is output matrix: \n");
for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
        {
            fprintf(output,"%d", otf[i][j]);
            if ((j%10) == 0)
                fprintf(output," ");
            if (j==np)
                fprintf(output,"\n");
        }

for (i=1; i<=nt; i++)
    fscanf(input, "%d", &d[i]);
for (i=0; i < nt; i++)
    fscanf(input, "%d", &output_channel[i]);

printf("Transition durations are: \n");
fprintf(output,"Transition durations are: \n");
for (i=1; i<=nt; i++)
    {
        printf(" %d", d[i]);
        fprintf(output," %d", d[i]);
        if (i%10==0) fprintf(output, "\n");
    }
printf("Output Bit Vector is: \n");
fprintf(output,"Output Bit Vector is: \n");
for (i=1; i<=nt; i++)
    {
        printf(" %d", output_value[i]);
        fprintf(output," %d", output_value[i]);
        if (i%10==0) fprintf(output, "\n");
    }

fprintf(output, "\n");

for (j=1; j<=np; j++)
    fscanf(input, "%d", &m[j]);

```

```

for (k=1; k<=np; k++)
    fscanf(input, "%d", &limit_bit_num[k]);

printf("\nInitial marking is: \n");
fprintf(output, "\nInitial marking is: \n");

for (j=1; j<=np; j++)
{
    printf(" %d", m[j]);
    fprintf(output, " %d", m[j]);
    if (i%10==0) fprintf(output, "\n");
}

printf("\nInput Bit Vector is: \n");
fprintf(output, "\nInput Bit Vector is: \n");

for (j=1; j<=np; j++)
{
    printf(" %d", limit_bit_num[j]);
    fprintf(output, " %d", limit_bit_num[j]);
    if (i%10==0) fprintf(output, "\n");
}

fprintf(output, "\n");
fscanf(input, "%d", &ndt);
for (i=1; i<=ndt; i++)
{
    fscanf(input, "%d", &dt[i][1]);
    fscanf(input, "%d", &dt[i][2]);
}

printf("\nDeactivating transition pairs are like this:\n");
printf("\nNumber of deactivating transition pairs are: %d\n", ndt);
fprintf(output, "\nDeactivating transition pairs are like this:\n");
fprintf(output, "\nNumber of deactivating transition pairs are: %d\n", ndt);

/*for (i=1; i<=ndt; i++)
{
    printf("%d", dt[i][1]);
    printf("%d", dt[i][2]);
    fprintf(output, "%d", dt[i][1]);
    fprintf(output, "%d", dt[i][2]);
}
*/

for (i=1; i<=ndt; i++)
{
    fprintf(output, "%d", dt[i][1]);
    fprintf(output, "%d", dt[i][2]);
}

```

```

        return 1;
    }

int enabled_transitions(matrix inm, row_matrix z)
{
    int i,j,k;

    for (i=1; i<=nt; i++)
    {
        for (j=1; j<=np; j++)
        {
            /*printf("\n m[%d]=%d inm[%d][%d]=%d",
                j,m[j], i,j,inm[i][j]);
            printf(" m[%d]-inm[%d][%d] = %d\n",
                j,i,j, m[j]-inm[i][j]);*/

            if ( (m[j]-inm[i][j]) < 0) goto SPOT;

        }

        if (dd[i] > 0)
            z[i] = 0;
        else
            z[i] = 1;

        SPOT:printf(" ");

        /*printf("\n transition %d is checked\n", i);
        getch();*/

    }

    /*printf("Z vector = ");
    for (i=1; i<=nt; i++)
        printf("\t %d ", z[i]);
    */

    /* This is to check passing parameters for enabled_transitions
    for (i=1; i<=nt; i++)
    {
        for (j=1; j<=np; j++)
        {

            printf("%d", inm[i][j]);
            if(j==np) printf("\n");
            getch();

        }
    }
    */
}

```

Checking of passing parameters finished here.
 call enabled_transitions(inf) in the main program
 and declare int enabled_transitions(matrix inm) in subprogram */

```

    return 1;
}

int conflict (row_matrix dx, int q1)
{
    int i,j,k,kj,kk;
    int c[99];
    int ci,cj;

    kk=0;
    kj=0;
    q1=0;

    /*printf("IN CONFLICT to check for dx[i] :\n");
    for (i=1; i<=nt; i++)
    {
        printf("dx[%d] =%d\n", i,dx[i]);
    }
    printf("\n");
    */

    for (i=1; i<=nt; i++)
    {
        if ( dx[i] > 0)
        {
            kk=kk+1; /*printf("\nkk = %d\n", kk);*/
            c[kk]=i; /*printf("c[%d] = %d\n", kk, c[kk]);*/
        }
    }

    for (i=1; i<=kk-1; i++)
    {
        for (j=i+1; j<=kk; j++)
        {
            for (k=1; k<=np; k++)
            {
                if ( inf[c[i]][k] >=1 && inf[c[j]][k] >=1 )
                {
                    /*printf("inf[%d][%d]: %d inf[%d][%d]: %d\n",
                    c[i],k,inf[c[i]][k],c[j],k,inf[c[j]][k]);*/
                    printf("TIME AT CONFLICT: %d \n",t);
                    fprintf(output,"\nTIME AT CONFLICT: %d \n",t);

                    printf("\nCONFLICT BETWEEN TRANSITIONS");
                    printf(" %d %d\n", c[i], c[j]);
                }
            }
        }
    }
}

```

```

        fprintf(output,"CONFLICT BETWEEN TRANSITIONS");
        fprintf(output," %d %d\n", c[i], c[j]);
        kj=kj+1;
        q1=kj;
        /*printf("q1=%d", q1);*/
        goto PLACE;
    }
    }
    PLACE: printf(" ");
}
}

q=q1;
/*printf("#of pairs of transitions with conflict are: %d\n", q1);*/
return 1;
}

```

```

int minimum(row_matrix dy, int mini, int tmini, mat rr, int hh)
{
int tx[99], ty[99];
int l, tmn, mn, i,j;

l=0;
hh=0;
for (i=1; i<=99; i++)
{
    rr[i]=0;
    tx[i]=0;
    ty[i]=0;
}

/*for (i=1; i<=nt; i++)
printf(" dd[%d] = %d", i, dd[i]);*/

for (i=1; i<=nt; i++)
{
    if (dy[i] > 0)
    {
        /*printf("\ndy[%d] = %d", i, dy[i]);*/
        l=l+1; /*printf(" l = %d", l);*/
        tx[l]=dy[i]; /*printf(" tx[%d] = %d", l,dy[i]);*/
        ty[l]=i; /*printf(" ty[%d] = %d\n", l,i);*/
    }
}

tmn=tx[1];
for(i=1; i<=l; i++)
{
    if (tx[i] <= tmn)
    {
        tmn=tx[i]; /*printf("tmn = %d", tx[i]);*/
    }
}

```



```

        mn=ty[i];/*printf("\tmn = %d\n", ty[i]);*/
    }
}

mini=mn;/*printf("mini = %d", mini);*/
tmini=tmn;/*printf("\t tmini = %d", tmini);*/

for (j=1; j<=l; j++)
{
    if (tx[j]==tmn)
    {
        hh=hh+1;/*printf("\thh = %d", hh);*/
        rr[hh]=ty[j];/*printf("\trr[%d] = %d\n", hh, rr[hh]);*/
    }
}

min=mini;
tmin=tmini;
for(i=1;i<=hh;i++)
    ra[i]=rr[i];
h=hh;

return 1;
}

int new_marking (row_matrix mm)
{
//int emergency_stop;
int i,j,k;
int b[99];
int flag[99];

    k=0;
    //emergency_stop=0;

    for (i=1; i<=nt; i++)
        b[i]=0;

    for (i=1; i<=nt; i++)
    {
        if (f[i] > 0)
        {
            k=k+1;/*printf("k = %d\n", k);*/
            b[k]=i;/*printf("b[%d] = %d\n", k,b[k]);getch();*/
        }
    }

    for (i=1; i<=k; i++)
    {
        for (j=1; j<=np; j++)
        {
            if (inf[b[i]][j] > 0)
            {

```

```

        if(limit_bit_num[j] <=11){
            do{
                digital_input(&num1);
                decimal_binary(num1,input_array);

            }while(input_array[limit_bit_num[j]] != 1);

        }
        flag[j] = 1;
    }
}
//printf("Check here for emergency stop\n");
if(limit_bit_num[np] <=11){

    digital_input(&num1);
    decimal_binary(num1,input_array);

    if (input_array[limit_bit_num[np]] == 1)
        emergency_stop = 1;
}
//printf("checking done\n");

if (emergency_stop != 1)
{

for (i=1; i<=k; i++)
    for (j=1; j<=np; j++)
        if (flag[j]==1)
            mm[j]=mm[j]-inf[b[i]][j];
}
else
{

    printf("Emergency Stop is pressed\n");
    printf("Execution Aborted due to emergency stop\n");
    end_num=32;
    digital_output(end_num);
    getch();
    exit(0);

}

return 1;
}

main()
{
    int i,j,k,sleep_time;
    clock_t end,start;
    int trans;

```

```

num = 0;
num1=0;
sleep_time=0;
trans=0;

digital_output(num);

for(i=0; i<=15; i++){
    output_array[i] = 0;
    input_array[i] = 0;
}
/*fp = fopen("try.in", "r");
for (i=1; i<=10; i++)
{
    printf("This is for sim[%d]\n", i);
    for (j=1; j<=2; j++)
    {
        fscanf(fp, "%d", &sim[i][j]);
        printf("simp[%d]: %d", j, sim[i][j]);
        getch();
        printf("\n");
    }
}

for (i=1; i<=10; i++)
{
    printf("sim[%d]:",i);
    for (j=1; j<=2; j++)
    {
        printf("%d  ", sim[i][j]);
    }
    printf("\n");
}

getch();

for (i=1; i<=10; i++)
{
    for (j=1; j<=2; j++)
    {
        dt[i][j]=sim[i][j];
        printf("%d  ", dt[i][j]);
    }
}

getch();

*/

bx=0;
by=0;
count=0;

for (i=1; i<=99; i++)

```

```

{
    dd[i]=0;
    r[i]=0;
    f[i]=0;
    ff[i]=0;
    atm[i]=0;
    limit_bit_num[i]=0;
    output_value[i]=0;
}

t=0;
ct=0;
min=0;
tmin=0;
h=0;
pt=0;
q=0;
emergency_stop=0;

for (i=1; i<=10; i++)
    ra[i]=0;

for (i=1; i<=99; i++)
{
    for (j=1; j<=99; j++)
    {
        inf[i][j]=0;
        off[i][j]=0;
        f[i]=0;
    }
}

clrscr();
read_petri_net();

printf("\nHOW LONG THE SYSTEM HAS TO BE SIMULATED: \n");
printf("\nPlease give time units : ");
scanf("%d", &count);
fprintf(output, "\nTIME OF SIMULATION: %d \n", count);
printf("\nGive BX and BY : ");
scanf("%d", &bx);
scanf("%d", &by);
fprintf(output, "bx = %d", bx);
fprintf(output, "  by = %d", by);
clrscr();
gotoxy(1,7);
printf("Transition in execution :");
gotoxy(1,10);

```

```

printf("Time after delay : ");

DOT:
for (i=1; i<=nt; i++)
{
    f[i]=0;
    ff[i]=0;
}

enabled_transitions(inf,f);
/*printf("\n before conflict F vector = ");
for (i=1; i<=nt; i++)
    printf("\t %d ", f[i]);
getch();*/

for (i=1; i<=nt; i++)
{
    if (f[i] > 0)
    {
        ff[i]=d[i] + t;
        /*printf("\n ff[%d] = %d", i, ff[i]);*/
    }
}

for (i=1; i<=nt; i++)
{
    dd[i] = dd[i] + ff[i];
    /*printf("\n dd[%d] = %d", i, dd[i]);*/
}

/*printf("\nIN MAIN befoe conflict\n"); getch();
for (i=1; i<=nt; i++)
    printf(" dd[%d] = %d", i, dd[i]);*/

/* for (i=1; i<=nt; i++)
    for (j=1; j<=np; j++)
    {
        printf("%d", inf[i][j]);
        fprintf(output, "%d", inf[i][j]);
        if (j==np) {printf("\n");
        fprintf(output, "\n");}
    }
*/

conflict(dd,q);

/*printf("q=%d\n", q);*/

if (q > 0)

```

```

{
    printf("\nGive the number of transitions to be disabled:\n");
    scanf("%d",&x);
    printf("Give transitions to be disabled:\n");
    fprintf(output, "Transitions disabled are: ");

    for (i=1; i<=x; i++)
    {
        scanf("%d", &a[i]);
        fprintf(output, " %d ",a[i]);
    }
    fprintf(output, "\n");

    for (i=1; i<=x; i++)
    {
        dd[a[i]]=0; /*printf("dd[%d] = %d\n",a[i],dd[a[i]]);*/
        f[a[i]]=0; /*printf("f[%d] = %d\n",a[i],f[a[i]]);*/
    }
}

/*printf("\n After conflict F vector = ");
for (i=1; i<=nt; i++)
    printf("\t %d ", f[i]);

getch();
*/

if (t > bx)
{
    fprintf(output, "t = %d    bx = %d\n",t,bx);
    bx=bx+by;
    fprintf(output, "TIME: %d\n", t);
    fprintf(output, "MARKING: ");
    for (i=1; i<=np; i++)
    {
        fprintf(output, " %d", m[i]);
        if (i%10==0) fprintf(output, "\n");
    }
    fprintf(output, "\n");

    fprintf(output, "F- VECTOR: ");
    for (i=1; i<=nt; i++)
    {
        fprintf(output, " %d", f[i]);
        if (i%10==0) fprintf(output, "\n");
    }
    fprintf(output, "\n");

    fprintf(output, "R- VECTOR: ");
    for (i=1; i<=nt; i++)
    {

```

```

        fprintf(output, " %d", r[i]);
        if (i%10==0) fprintf(output, "\n");
    }
    fprintf(output, "\n");

    fprintf(output, "ACTIVE TIME VECTOR: ");
    for (i=1; i<=nt; i++)
    {
        fprintf(output, " %d", atm[i]);
        if (i%10==0) fprintf(output, "\n");
    }
    fprintf(output, "\n");

}

/*printf("\n Before CONT F vector = ");

for (i=1; i<=nt; i++)
{
    getch();
    printf("\t %d   r[%d]: %d\n ", f[i],i,r[i]);
}
getch();*/

for (i=1; i<=nt; i++)
{
    if ( (f[i]==0) && (r[i]==0) )
        printf(" ");
    else
        goto CONT;
}

printf("*****Program terminated*****\n");
fprintf(output, "*****DEAD STATE*****\n");

goto POINT;

CONT: ct = ct + 1;

if (t > count) goto POINT;

minimum(dd,min,tmin,ra,h);

/*printf("min = %d\n\t", min);
printf(" tmin = %d\n\t", tmin);
printf("h = %d\n\t", h);
for(i=1; i<=h; i++)
    printf("ra[%d] = %d\n", i, ra[i]);
*/

pt=t; /*printf("\n pt= %d\n\t", pt);*/
t=tmin; /*printf("t= %d\n\t", t);*/

```

```

for (i=1; i<=h; i++)
{
    dd[ra[i]]=0;
    /*printf("\n dd[%d] = %d\n", ra[i],dd[ra[i]]);*/
    atm[ra[i]]=atm[ra[i]]+tmin-pt;
    /*printf("atm[%d] = %d\n",ra[i],atm[ra[i]]);*/
}

for (i=1; i<=nt; i++)
{
    r[i]=0;
    /*printf("\n r[%d] = %d\n", i, r[i]);*/
}

for (i=1; i<=nt; i++)
{
    if (dd[i] > 0)
    {
        r[i]=dd[i]-t; /*printf("\n r[%d] = %d\n", i, r[i]);*/
        atm[i]=atm[i]+tmin-pt; /*printf("atm[%d] = %d\n",
                                i,atm[i]);*/
    }
}
/*printf("\nbefore new marking F vector = ");
for (i=1; i<=nt; i++)
    printf("\t %d ", f[i]);
printf("\n");

printf("\nbefore removing tokens MARKING m is = ");
for (i=1; i<=np; i++)
    printf("\t %d ", m[i]);
printf("\n");
*/

new_marking(m);

/*printf("After removing tokens MARKING m is = ");
for (j=1; j<=np; j++)
{
    m[j]=mm[j];
    printf("\t %d", m[j]);
}
*/
/*printf("\nBefore depositing tokens MARKING m is = ");
for (j=1; j<=np; j++)
{
    printf("\t %d", m[j]);
}
*/
for (j=1; j<=h; j++)
{
    //printf(" Time : %d\n", t);

```



```

        sleep_time = d[ra[j]];
        gotoxy(30,7);
        clrcol();
        printf("%d", ra[j]);
        trans=ra[j];
        num = num + output_channel[ra[j]-1];
        digital_output(num);
    }
    if (emergency_stop==1)
    {
        printf("Transition being executed during emergency stop is %d\n",trans);
        getch();
    }

    start = clock();
    delay(sleep_time);
    end = clock();
    gotoxy(30,10);
    clrcol();
    printf("%3.1f", (end - start) / CLK_TCK);
    gotoxy(1,15);
    for (j=1; j<=h; j++)
    {
        for (i=1; i<=np; i++)
        {
            m[i] = m[i] + otf[ra[j]][i];
        }
    }

    /*printf("\nAfter depositing tokens MARKING m is = ");
    for (j=1; j<=np; j++)
    {
        printf("\t%d", m[j]);
    }
    */

    goto DOT;
POINT:
printf("\nEnd of sequence\n");
fprintf(output, "EXECUTION TERMINATED:\n ");
getch();

fclose(output);

}

```

Input file to the software package for the RTPN in Fig. 7.2. (b)

Number of places: 13
Number of transitions: 8

Petri Net connectivity representation:

Input place ID: -->

Transition ID:

```

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 1 0 0 0 0 0 0 1 1
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 1 0 0 0 1

```

Output place ID: -->

Transition ID:

```

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 1 0 1 0 0 1 0 0 0 0 1 0 0

```

Transition duration vector D (contains 1st attribute of transition):

1 1 1 1 1 1 1 1

Output signal vector (contains 2nd attribute of transition):

0 1 0 2 0 3 0 -6

Initial marking (contains 1st attribute of place):

1 0 0 1 0 0 1 0 0 0 0 1 3 0

Input signal vector (contains 2nd attribute of place):

8 4 - 5 0 - 6 1 - 2 - - - 10

Output file generated by the software package for the RTPN in Fig. 7 (b)

```

TIME: 1
MARKING: 0 1 0 1 0 0 1 0 0 0 0 1 1 0
F-VECTOR: 0 1 0 0 0 0 0 0 (transition 2 is enabled)
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 0 0 0 0 0 0 0
                    (transition 1 was active for 1 unit of time)

TIME: 2
MARKING: 0 0 1 0 0 0 0 0 0 0 0 0 0 0
F-VECTOR: 0 0 1 0 0 0 0 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 0 0 0 0 0 0

TIME: 3
MARKING: 0 0 0 1 1 0 0 0 0 0 0 0 0 0
F-VECTOR: 0 0 0 1 0 0 0 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 0 0 0 0 0

TIME: 4
MARKING: 0 0 0 0 0 1 0 0 0 0 0 0 0 0
F-VECTOR: 0 0 0 0 1 0 0 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 1 0 0 0 0

TIME: 5
MARKING: 0 0 0 0 0 0 1 1 0 0 0 0 0 0
F-VECTOR: 0 0 0 0 0 1 0 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 1 1 0 0 0

TIME: 6
MARKING: 0 0 0 0 0 0 0 0 1 0 0 0 0 0
F-VECTOR: 0 0 0 0 0 0 1 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 1 1 1 0 0

TIME: 7
MARKING: 0 1 0 0 0 0 0 0 0 1 0 0 0 0
F-VECTOR: 0 0 0 0 0 0 0 1
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 1 1 1 1 0

TIME: 8
MARKING: 0 1 0 1 0 0 1 0 0 0 1 1 0 0
F-VECTOR: 0 0 0 0 0 0 0 0
R-VECTOR: 0 0 0 0 0 0 0 0
ACTIVE TIME VECTOR: 1 1 1 1 1 1 1 1
EXECUTION TERMINATED:

```

REFERENCES

- Al-Jaar, R.Y., and Desrochers, A., "Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri nets," *IEEE Trans. on Robotics & Automation*, Vol. 6, No. 6, 1990, pp. 621-639.
- Alanche, D., Benzakour, D.F., Gillid, P., Rodriguel, P. and Valette, R., "PSI: A Petri Net based simulator for Flexible manufacturing systems," *Advances in Petri Nets*, Lecture notes in computer science, 188, Springer Verlag, 1984, pp. 1-14.
- Anthony, I.R., "Flexible Manufacturing Systems: Issues and Implementation," *Industrial Management*, Vol. 33, No. 4, July/Aug 1991, pp. 7-11.
- Archetti, F. and Sciomachen, A., "Development, analysis and simulation of Petri Net models: An application to AGV systems," *OR models in FMSs*, 1987.
- Attaran, M. "Flexible Manufacturing Systems: Implementing an Automated Factory", *Information Systems Management*, Vol. 9, No. 2, Spring 1992.
- Balgh, T. and Vittera, J., "Architectural considerations in the development of an IEEE 802.4 token bus chip set," *IEEE 4th Annual International Conference on Computers and Communications*, Phoenix, AZ, March 1985, pp. 439-443.
- Barad, M., and Sipper, D., "Flexibility in Manufacturing Systems: Definitions and Petri net Modeling," *International Journal of Production Research*, Vol. 26, No. 2, 1988, pp. 237-248.
- Berchi, R., and G. Froisi, "Design of the Material Handling System for a Manufacturing Line," *AIRO Workshop on Coordination Management by means of Petri Nets*, April, Modena, Italy, 1988.
- Black, J.T., "Cellular Manufacturing Systems Reduce Setup Time, Make Small Lot Production Economical," *Industrial Engineering*, Vol. 15, No. 11, November, 1983, pp. 39.
- Blumenthal, Marjory and Dray, Jim, "The Automated Factory: Vision and Reality," *Technology Review*, Vol. 88, No. 1, January 1985, pp. 28-37.
- Booch, B., *Object-oriented Analysis and Design with Applications*, The Benjamin/Cummings Publ. Co., 1994.
- Boucher, T.O., Jafari, M.A., and Meredith, G.A., "Petri Net Control of an Automated Manufacturing Cell," *Computers in Industrial Eng.*, Vol. 17 No. 1-4, 1989, pp. 459-463, .
- Bozer, Y.A. and Srinivasan, M.M., " Tandem configurations for automated guided vehicle systems offer simplicity and flexibility," *Industrial Engineering*, Vol. 21, No. 1, 1989, pp. 23-27.
- Bruno, G., and Marchetto, G., "Process translatable Petri nets for the Rapid Prototyping of Process Control Systems," *IEEE Trans. on Software Eng.*, Vol. 12, No. 2, 1986, pp. 346-356.

- Buffa, Elwood, S., *Meeting the Competitive Challenge: Manufacturing Strategies for U.S. Companies*, Richard Irwin, Illinois, 1984, pp. 83.
- Buzacott, J.A. and Yao, D.D., "Flexible Manufacturing Systems: A Review of Analytical Models," *Management Science*, Vol. 32, No. 7, July 1986, pp. 890-905.
- Cecil, J.A., K. Srihari, and E.R. Emerson, "A Review of Petri-net Applications in Manufacturing," *Intl. J. Adv. Mfg. Tech.*, Vol. 7, No. 3, 1992, pp. 168-177.
- Chaar, J.K., D. Teichroew, and R.A. Voltz, "Developing Manufacturing Control Software: a Survey and Critique," *Intl. J. of Flexible Mfg. Sys.*, Vol. 5, No. 2, 1993, pp. 53-88.
- Chaar, J.K., D. Teichroew, and R.A. Voltz, "Real-time Software Methodologies: Are They Suitable for Developing Manufacturing Control Software?," *Intl. J. of Flexible Mfg. Sys.*, Vol. 5, No. 2, 1993, pp. 95-128.
- Chaar, J.K., Voltz, R.A., and Davidson, E.S., "An Integrated Approach to Developing Manufacturing Control Software", *Proceedings of the 1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991.
- Chan, C.C., and Wang, H.P., "Design and Development of a Stochastic High-level Petri net System for FMS Performance Evaluation," *Int. J. of Prod. Res.*, Vol. 31, No. 10, 1993, pp. 2415-2440.
- Chang, Y.C, Sullivan, R.S., Bagchi, U, and J.R, Wilson, J.R., "Experimental Investigation of Real-time Scheduling in Flexible Manufacturing System," *Annals of Operations Research*, Vol. 3, 1985, pp. 355-377.
- Chocron, D., and Cerny, E., "A Petri net Based Industrial Sequencer," *Proc. of IEEE Int. Conf. and Exhibition on Industrial Control and Instrumentation*, 18-22, March, 1980.
- Courvoisier, R. Valette, A.Sahraoui and M. Combacau., "Specification and Implementation Techniques for Multilevel Control and Monitoring of FMS," *Computer Applications in Production and Engineering*, F. Kumara and A. Rolstadas (editors), Elsevier Science Publishers, IFIP, 1989, pp. 509-516.
- Crockett, D., Desrochers, A.A., DiCesare, F., and Ward, T., "Implementation of a Petri Net Controller for a Machining Workstation," *Proceedings of the IEEE Conf. on Robotics and Automation*, 1987, pp. 1861-1867.
- Cumings, S., "Developing Integrated Tooling Systems: A Case Study at Garrett Turbine Engine Company," *Proceedings of Fall Industrial Engineering Conference*, Boston, MA, 1986.
- Dallas, D.B., "The Impact of FMS," *Production*, Vol. 9, No. 10, October 1984, pp. 33-38.
- Darrow, W. P., "International Comparison of Flexible Manufacturing System Technology," *Interfaces*, Vol. 17, November-December 1987:88.
- David, R., and Alla, H., *Petri Nets and Grafset*, Prentice Hall, New York, 1992.

- Davis, W.J., Jackson, R.H.F. and Jones, A.T., "Real time optimization in the automated manufacturing research facility", *Progress in material handling and logistics*, J.A. White and I.W. Dence (editors), Springer Verlag, 1989.
- Dhar, U.R., Overview of Models and DSS in Planning and Scheduling of FMS, *International Journal of Production Economics*, Vol. 25, No. 1-3, Dec. 1991, pp. 121-127.
- DiCesare F., and Desrochers, A.A., "Modeling, Control, and Performance Analysis of Automated Manufacturing Systems using Petri Nets," *Control and Dynamic Systems*, Vol. 47, C.T. Leondes (editor), Academic Press, 121-172, 1991.
- Dimitrov, P., "The Impact of Flexible Manufacturing Systems (FMS) on Inventories," *Engineering Costs and Production Economics*, Vol. 19, Nos.1-3, May 1990, pp. 165-174.
- Duffie, N.A., Chitturi, R. and Mou, Jong-I, "Fault-tolerant Heterarchical Control of Heterogeneous Manufacturing System Entities," *Journal of Manufacturing Systems*, Vol. 7, No. 4., 1988, pp. 315-328.
- Egbelu, P.J. and Tanchoco, J.M.A., "Characterization of automatic guided vehicle dispatching rules," *International Journal of Production Research*, Vol. 22, No. 2, 1984, pp. 359-374.
- Egbelu, P.J. and Tanchoco, J.M.A., "Potentials for bidirectional guide path for automated guided vehicle based systems," *International Journal of Production Research*, Vol. 24, No. 6, 1986, pp. 1075-1097.
- Falcione, A., and Krogh, B.H., "Design Recovery for Relay Ladder Logic," *IEEE Control Systems Magazine*, April, 1993, pp. 90-98, .
- Fernandez, E.B., and C.P. Han, "Object-oriented Design of Flexible Manufacturing Systems," *Proc. of 6th Annual Conf. on Recent Advances in Robotics*, Gainesville, FL, 1993.
- Ferrarini, L., "An Incremental Approach to Logic Controller Design with Petri Nets," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 22, No. 3, 1992, pp. 461-473.
- French, R.L., "Management Looking At CIM Must Deal Effectively With These Issues And Realities," *Industrial Engineering*, Vol. 16 August 1984, pp. 70+.
- Garg, K., "An Approach to Performance Specification of Communication Protocols Using Timed Petri nets," *IEEE Trans. on Software Eng.*, Vol. 11, No. 10, 1985, pp. 1216-1255.
- Gaymon, D.J., "Computers in the Tool Crib," *Manufacturing Engineering*, Vol. 103, No. 9, September, 1986, pp. 41-44.
- Gaymon, D.J., "Meeting Production Needs with Tool Management," *Manufacturing Engineering*, September 1987, Vol. 104, No. 9, pp. 41-47.

- Gershwin, B.S., and Berman, O., "Analysis of Transfer Lines Consisting of Two Unreliable Machines With Random Processing Times and Finite Storage Buffers," *AIEE Transactions*, Vol. 13, No. 1, 1981, pp. 2-11.
- Ghosh, B.K., "Design and Performance Analysis Models of Computer Networks in CIM Systems," *Computers in Industry*, Vol. 12, 1989, pp. 141-152.
- Gilbert, J.P. and Winter, P.J., "Flexible Manufacturing Systems: Technology and Advantages," *Production and Inventory Management*, Vol. 27, No. 4, Fourth Quarter 1986, pp. 53.
- Glasse, C.R., and Hong, Y., "Analysis of Behavior of an Unreliable n-stage Transfer Line With (n-1) Inter-stage Storage Buffers," *International Journal of Production Research*, Vol. 31, No. 3, 1993, pp. 519-530.
- Glasse, C.R., and S. Adiga, "Berkeley Library of Objects for Control and Simulation of Manufacturing (BLOCS/M)," *Applications of Object-Oriented Programming*, (editors) L.J. Pinson, and R.S. Wiener, Addison-Wesley Publishing Company, 1990, pp. 1-26.
- Goldhar, Joel D., "What Flexible Automation Means to Your Business", *Modern Material Handling*, Vol. 39, September 7, 1984, pp. 63-65.
- Graham, J.H., S.M. Alexander, and W.Y. Lee, "Object-oriented Software for Diagnosis of Manufacturing Systems," *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991, pp. 1966-1971.
- Gray, Ann E. and Stecke E. Kathryn., "Tool Management in Automated Manufacturing: Operational Issues and Decision Problems," *working chapter series, CMOM 88-03*, William Simon Graduate School of Business Administration, Univ. of Rochester, November 1988.
- Groenevelt, H., Pintelon, L., and Seidmann, A., "Production Batching With Machine Breakdowns and Safety Stocks," *Operations Research*, Vol. 40, No. 5, 1992, pp. 959-971.
- Guha, R.D, Lange and J. Basiouni, "Software Specification and Design Using Petri nets," *Proceedings of 4th International Workshop on Software Specification and Design*, 1987, pp. 225-230.
- Gupta, D. and Buzacott, J.A., "A Framework for Understanding Flexibility of Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 2, No. 2, 1989, pp. 89.
- Haidar, B., Fernandez, E.B., and Horton, T.B., "An Object-oriented Methodology for the Design of Control Software for Flexible Manufacturing Systems," *Proc. of 2nd Workshop on Parallel and Distributed Real-time Systems*, Cancus, Mexico, April 1994.
- Harry, B. and Malcolm, H., and Koos, K., "FMS Implementation Management: Promise and Performance," *International Journal of Operations and Production Management*, Vol. 10, No. 1, 1990, pp. 5-20.
- Harvey, R.E., "Factory 2000," *Iron Age*, Vol. 227, June 1984, pp. 72-76.

- Hays, Robert H. and Wheelright, S.C., *Restoring Our Competitive Edge: Competing Through Manufacturing*, John Wiley, New York, 1984, pp. 192.
- Heywood, P., "Four Generations of FMS," *American Machinist*, Vol. 32, No. 3, March 1988, pp. 62-63.
- Hodgson, T.J., King, R.E., Manteih, S.K. and Schultz, S.R., "Developing control rules for an AGVS using Markov decision processes," *Material Flow*, Vol. 4, 1987, pp. 85-96.
- Hsieh, S., and Shih, Y., "The Development of an AGVS Model by Union of the Modularised Floor-path nets," *Int. J. Adv. Mfg. Tech.*, Vol. 9, 1994, pp. 20-34.
- Hsu, C.L., "Flexible Manufacturing System Controller Software Development by Object-oriented Programming," *Proc. of the Second Int. Conf. on Automation Tech.*, Taipei, Taiwan, 1992, pp. 53-59.
- Huang, H., and Chang, P., "Specification, Modeling, and Control of a Flexible Manufacturing cell," *Int. J. of Prod. Res.*, Vol. 30, No. 11, 1992, pp. 2515-2543.
- Huang, P.Y., "Analysis of the Necessary Conditions for Implementing JIT Production," *Zero Inventory Philosophy and Practice, Seminar Proceedings*, St. Louis, MO, 1984, pp. 24-29.
- Huang, P.Y., and M. Sakurai, "Factory Automation: the Japanese Experience," *IEEE Transactions on Engineering Management*, Vol. 37, No. 2, May, 1990, pp. 103-108.
- Hughes, Tom and Hegland, Don, "Flexible Manufacturing: The Way to the Winner's Circle," *Production Engineering*, Vol. 30, No. 9, September 1983, pp. 55.
- Inman, A.R., "Flexible Manufacturing Systems: Issues and Implementation," *Industrial Management*, Vol. 33, No. 4, July/August 1991, 7-11.
- Ismael, D. Jr., "Back to Basics: Just What is Involved in Implementing a Flexible Manufacturing System?," *Industrial Engineering*, Vol. 23, No. 4, April 1991, pp. 43-44.
- Jafari, M.A., "An Architecture for a Shop-Floor Controller Using Colored Petri Nets," *Journal of Manufacturing Systems*, Vol. 4, No. 4, 1992, pp. 159-181.
- Jain, S., "Basis for Development of a Generic FMS Simulator," *Proc. of the second ORSA/TIMS Conf. on FMS: Operations research models and applications*, Eds. K.E. Stecke, and R. Suri, 1986, pp. 393-403.
- Jari, M., "The Success of FM investments: Case studies from Small Industrial Economies," *International Journal of Technology Management*, Vol. 6, No. 3, 1991, pp. 277-291.
- Johnson, M.E., L. Thompson, and R. Fontaine, "An Integrated Simulation and Shop-floor Control System," *Manufacturing Review*, Vol. 5, No. 3, 1992, pp. 158-165.

- Jones, A.T. and McLean, C.R., "A Proposed Hierarchical Control Model for Automated Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 5, No. 1, 1986, pp. 15-25.
- Jothishankar, M.C., and Wang, H.P., "Determination of Optimal Number of Kanbans Using Stochastic Petri nets," *Journal of Manufacturing Systems*, Vol. 11, No. 6, 1992, pp. 449-461.
- Jukka, R. and Iouri, T., "Economics and Success Factors of Flexible Manufacturing Systems: The Conventional Explanation Revisited," *International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 2, 1990, pp. 169-190.
- Kaighobadi, M., and K. Venkatesh, "Investigating the Performance of Push and Pull Systems in Flexible Automation Environment Using Petri nets," *Proc. of 1992 Decision Sciences Institute's Annual Meeting*, San Francisco, CA, 1992, pp. 1253-1255.
- Kaighobadi, M., and Venkatesh, K., "Flexible Manufacturing Systems: an Overview," to appear in *Int. J. of Operations and Production Management.*, Vol. 14, No. 4, 1994, pp. 26-49.
- Kakati, M. and Dhar, U.R., "Investment Justification in Flexible Manufacturing Systems", *Engineering Costs and Production Economics*, Vol. 21, No. 3, 1991, pp. 203-209.
- Kaku, B.K., "Fitting Flexible Manufacturing Systems to the task: An Analysis of Current Practices," *Working chapter series MS/S 92-003*, College of Business and Management, University of Maryland, 1992.
- Kiesler, Sara, "New Technology in the Workplace/Robotics: Cause and Effect," *Public Relations Journal*, Vol. 39, No. 12, December 1983, 12-16.
- Kimura, O., and Terada, H., "Design and Analysis of Pull System, a Method of Multi-stage Production Control," *Int. J. of Prod. Res.*, Vol. 19, No. 3, 1981, pp. 241-253.
- King, R.E., Hodgson, T.J. and Monteith, S.K., "Evaluation of heuristic control strategies for AGVs under varying demand and arrival patterns," *Progress in material handling and logistics*, Springer Verlag, 1989.
- Klahorst, Thomas,H., "Flexible Manufacturing Systems: Combining Elements to Lower Costs, Add flexibility," *Industrial Engineering*, Vol. 32, No. 11, November 1981, 112-117.
- Knapp, G.M., and Wang, H.P., "Modeling of Automated Storage/Retrieval Systems Using Petri nets," *Journal of Manufacturing Systems*, Vol. 11, No. 2, 1992, pp. 20-29.
- Kochikar, V.P., and Narendran, T.T., "On Using Abstract Models for Analysis of Flexible Manufacturing Systems," *Int. J. of Prod. Res.*, Vol. 32, No. 10, 1994, pp. 2303-2322.
- Kochikar, V.P., and Narendran, T.T., "Modeling Automated Manufacturing Systems Using a Modification of Coloured Petri nets," *Robotics and Computer Integrated Manufacturing*, Vol. 9, No. 3, 1992, pp. 181-189.

- Krinsky, I., Melnez, A., Mitenbarg, G.H. and Myers, B.L., "Flexible Manufacturing System Evaluation: An Alternative Approach," *International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 2, 1991, pp. 237-253.
- Kwok, S.C., "A Case Report on Integrating FMS and Traditional Machine Tools," *Flexible Manufacturing Systems*, Society of Manufacturing Engineers, Dearborn, MI, 1988.
- Lee, L.C., "Parametric Appraisal of the JIT System," *Int. J. of Prod. Res.*, Vol. 25, No. 10, 1987, pp. 1415-1429.
- Les, Gould., Smart Handling Doubles FMS Productivity, *Modern Materials Handling*, Vol. 4, No. 1, January 1990, pp. 64-66.
- Maccarini, G., Giardini, C., Zavanella, L. and Bugini, A., "Different kind of tool room models for an FMS: a simulation approach and analysis," *International AMSE Conference.*, Vol. 4, 1987, pp. 87-89, Karlsruhe.
- Mahadevan, B. and Narendran, T.T., "Design of an automated guided vehicle based material handling system," *International Journal of Production Research*, Vol. 28, No. 9, 1990, pp. 1611-1622.
- Maimon, O.Z., "Real-time Operational Control of Flexible Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 6, No. 2, 1987, pp. 125-136.
- Maione, B., Semeraro, Q., and B. Turchiano, 1986, "Closed Analytical Formula for Evaluating FMS Performance Measure," *Int. J. of Prod. Res.*, Vol. 24, No. 3, 1986, pp. 583-592.
- Malmberg, C.J., "A model for the design of zone control automated guided vehicle systems," *International Journal of Production Research*, Vol. 28, No. 10, 1990, pp. 1741-1758.
- Marinov, D.D., and N. Todorov, "Software Development Approach in FMS," *Computers in Industry*, Vol. 10, No. 3, 1988, pp. 171-175.
- Mascolo, M.D., Frein, Y., Dallery, Y., and R. David, "An Unified Modeling of Kanban systems Using Petri nets," *Int. J. of Flexible Manufacturing Systems*, Vol. 3, 1991, pp. 275-307.
- Masory, O., "Monitoring of Tool Wear Using Artificial Neural Networks," *International Journal of Material Processing Technology*, Vol. 63, 1990.
- Maxwell, W.L. and Muckstadt, J.A., "Design of automated guided vehicle systems," *IIE Transactions*, Vol. 14, 1982, pp. 114- 124.
- May, B., "FMS Control Software Basics," *Proc. of Flexible Mfg. Systems*, Chicago, Illinois, 1986.
- Meyer, B., *Object-oriented Software Construction*, Prentice-Hall, 1988.
- Michel, G., *Programmable Logic Controllers: Architectures and Application*, John Wiley and Sons, England, 1990.

- Monarchi, D.E., and G.I. Puhr, "A Research Topology for Object-oriented Analysis and Design," *Communications of the ACM*, Vol 35, No. 9, 1992, pp. 35-47.
- Monden, Y, "How Toyota Shortened Supply Lot Production Time, Waiting Time, and Conveyance Time," *Industrial Engineering*, Vol. 13, No. 9, 1981, pp. 22-30.
- Mullins, Peter J., "Feeding Flexible Manufacturing Systems," *Automotive Industry*, Vol. 164, November 1984, pp. 63-64.
- Murata, T., B. Shenker, and S. Shatz, "Detection of Ada Static Deadlocks Using Petri net Invariants," *IEEE Trans. on Software Eng.*, Vol. 15, No. 3, 314-326, 1989.
- Murata, T., Komoda, N., Matsumoto, K., and Haruna, K., "A Petri Net-based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation," *IEEE Trans. on Industrial Electronics*, Vol. 33, No. 1, 1986, pp. 1-8.
- Murata, T., "Petri Nets: Properties, Applications and Analysis," *Proc. of IEEE* , Vol. 77, No. 4, 1989, pp. 541-580.
- Narahari, Y., and Viswanadham, N., 1985, "A Petri net Approach to the Modelling and Analysis of FMSs," *Annals of Operations Res.*, Vol. 30, 1985, pp. 449-472.
- Narumol, U., and Daganzo, C.F., "Impact of Parallel Processing on Job Sequences in Flexible Assembly Systems," *International Journal of Production Research*, Vol. 27, No. 1, 1986, pp. 73-89.
- Naylor, A.W., and R.A. Voltz, "Design of Integrated Manufacturing System Control Software," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 17, No. 6, 881-897, 1987.
- Newton, D., "Simulation model helps determine how many automatic guided vehicles are needed," *Industrial Engineering*, Vol. 17, No. 1, 1985, pp. 68-79.
- Nof, S.Y., *Handbook of industrial robots*, John Wiley and Sons, Inc., New York, NY, 1985.
- Occena, G. and Yokota, T., Modeling of an Automated Guided Vehicle System in a Just-in-time Environment, *International Journal of Production Research*, Vol. 29, No. 3, 1991, pp. 495-511.
- Ould-Kaddour, N., and Courvoisier, M., "A Multi-tasking Environment Based on Petri nets With Objects and Modula-2," *Proc. of the 15th Annual Conf. of IEEE Indu. Electronics Society*, Philadelphia, PA, 1989, pp. 799-804.
- Ozden, M., " A simulation study of multiple load carrying automatic guided vehicles in a Flexible manufacturing system," *International Journal of Production Research*, Vol. 26, No. 7, 1988, pp. 1353-1366.
- Pessen, D.W., *Industrial Automation: Circuit Design and Components*, New York: Wiley, 1989.

- Pessen, D.W., "Ladder-Diagram Design for Programmable Controllers," *Automatica*, Vol. 25, No. 3, 1989, pp. 407-412.
- Primrose, P.L. and Leonard, R. "Selecting Technology for Investment in Flexible Manufacturing," *International Journal of Flexible Manufacturing Systems*, Vol. 4, No. 1, 1991, pp. 51-77.
- Proth, J.M., "Discrete Manufacturing Systems: From Specification to Evaluation," Invited lecture, *Proc. of the Second Int. Conf. on Automation Technology*, Taipei, Taiwan, 1992, pp. 7-8.
- Raju, K., and Chetty, O.V.K., "Priority nets for Scheduling Flexible Manufacturing Systems," *Journal of manufacturing systems*, Vol. 12, No. 4, 1993, pp. 326-340.
- Raju, K.R., and Chetty, O.V.K., "Design and Evaluation of Automated Guided Vehicle Systems for Flexible Manufacturing Systems: An Extended Timed-Petri net-based Approach," *International Journal of Production Research*, Vol. 31, No. 5, 1993, pp. 1069-1096.
- Ram, S.S. and Yash, G.P., "Strategic Cost Measurement for Flexible Manufacturing Systems", *Long Range Planning*, Vol. 24, No. 5, 1991, pp. 34-40.
- Ranky, P., *The Design and Operation of FMS: Flexible Manufacturing Systems*, IFS Publications Ltd., UK, 1986.
- Reddy, C. E., Chetty, O.V.K., and Chaudhuri, D., "A Petri net Based Approach for Analyzing Tool Management Issues in FMS," *Int. J. of Prod. Res.*, Vol. 30, No. 6, 1992, pp. 1427-1446.
- Reddy, C. E., Chetty, O.V.K., and Chaudhuri, D., "Design of a Tool Delivery System Using Expert Simulation in FMS," *Proceedings of the 6th Convention of Computer Engineers*, Trichy, India, September, 1990.
- Reddy, C. E., Chetty, O.V.K., and Chaudhuri, D., "Expert Tool in Flexible Manufacturing Systems," *Proceedings of the International Conference on Automation, Robotics and Computer Vision*, Singapore, September, 1990.
- Reddy, C. E., Chetty, O.V.K., and Chaudhuri, D., "Objective SIMTOOL in FMS," *Proceedings of the 5th International Conference on CAD/CAM Robotics and Factories of the Future*, Norfolk, USA, December 2-5, 1990
- Righini, G., "Modular Petri nets for Simulation of Flexible Production Systems," *Int. J. of Prod. Res.*, Vol. 31, No. 10, 1993, pp. 2463-2477.
- Rogers, R.V., "Understanding Implications of Object-oriented Simulation and Modeling," *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, Charlottesville, VA, 1991, pp. 285-289.
- Rolston, L.J., "Modeling FMSs with MAP/1," *Annals of Operations Research*, Vol. 3, 1985, pp. 189.
- Ross, D.T., "Structured analysis SA: A Language for Communicating Ideas," *IEEE Trans. on Software Eng.*, Vol. 3, No. 1, 1977, pp. 16-34.

- Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, 1991.
- Sabuncuoglu, I. and Hommuertzheim, L., "An investigation of machine and AGV scheduling rules in an FMS," *Operation research models and applications*, K.E. Stecke and R. Suri (editors), 1989, pp. 261-266.
- Sahraoui, A.E.K., and N. Ould-Kaddour, "Control Software Prototyping," *Computers and Industry*, Vol. 20, No. 3, 1992, pp. 327-334.
- Salomon, D.P. and Beigel, J.E., "Assessing Economic Attractiveness of FMS Applications in Small-Batch Manufacturing," *Industrial Engineering*, June 1984, pp. 88-96.
- Sarkar, B.R., and Fitzsimmons, J.A., "The Performance of Push and Pull Systems: A Simulation and Comparative Study," *Int. J. of Prod. Res.*, Vol. 27, No. 10, 1989, 1715-1732.
- Sarkar, B.R., "Simulating a Just in Time Production System," *Computers and Indu. Eng.*, Vol. 16, No. 1, 1989, pp. 127-130.
- Scalpone, R.W., "Education Process Is Vital to Realization of CIM Benefits, Handling of Pitfalls," *Industrial Engineering*, Vol. 16, No. 10, October 1984, pp. 110-116.
- Schonberger, J.R., "Applications of Single-card and Dual-card Kanban," *Interfaces*, Vol. 13, No.4, 1983, pp. 56-67.
- Schroer B.J. and Tseng, F.T., "Modeling complex manufacturing systems using discrete event simulation," *Computers and Industrial Engineering*, Vol. 14, 1985, pp. 455-464.
- Sheng, H.T., and Black, J.T., "Cellular Manufacturing System Modeling: The Petri net approach," *Journal of Manufacturing Systems*, Vol. 9, No. 1, 1990, pp. 41-54.
- Sheng, H.T., and Black, J.T., "Cellular Manufacturing System Modeling: the Petri net Approach," *International Journal of Manufacturing Systems*, vol. 9, No. 1, 1990, pp. 41-54.
- Silva, M., and R. Valette, "Petri nets and Flexible Manufacturing," *Advances in Petri Nets, Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, 1990, pp. 374-417.
- Silva, M., and Valette, R., "Petri Nets in Flexible Manufacturing," *Advances in Petri Nets*, G. Rozenberg (editors). Springer-Verlag, 1990, pp. 375-417.
- Smith, J., and S.B. Joshi, "Reusable Software Concepts Applied to the Development of FMS Control Software," *Intl. J. of Comp. Integrated Mfg.*, Vol. 5, No. 3, 1992, pp. 182-196.
- Spearman, L.M., Woodruff, D.L., and Hopp, W.J., "CONWIP: A Pull Alternative to Kanban," *Int. J. of Prod. Res.*, Vol. 28, No. 5, 1990, pp. 879-894.

- Srinivasan, V.S., and Jafari, M.A., "Monitoring and Fault Detection in Shop Floor Using Timed Petri nets," *Proc. of IEEE Conf. on Systems, Man, and Cybernetics*, Charlottesville, VA, 1991, pp. 355-360.
- Stecke, K. E., "Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems," *Management Science*, Vol. 29, No. 3, March 1983, pp. 273-288.
- Stecke, K., "FMS Design and Operating Problems and Solutions," *Proc. of the 2nd Intelligent Factory Automation Symposium, ISCIE*, 1989, pp. 17-32.
- Stecke, K.E, and J. Solberg, J.J, "Loading and Control Policies for Flexible Manufacturing System," *International Journal of Production Research*, Vol. 19, No. 5, 1981, pp. 481-490.
- Stefano, Di. A., and Mirabella, O., "A Fast Sequence Control Device Based on Enhanced Petri nets," *Microprocessors and Microsystems*, Vol. 15, No. 4, 1991, pp. 179-186.
- Sturzenbecker, M.C., "Building an Object-oriented Environment for Distributed Manufacturing Software," *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991, pp. 1972-1978.
- Suri, R., and G.W. Diehl, "MANUPLAN: A Precursor to Simulation for Complex Manufacturing Systems," *Proc. of Winter Simulation Conference*, 1985.
- Suri, R., and J.W. Dille, "A Technique for On-line Sensitivity Analysis of Flexible Manufacturing Systems," *Annals of Operations Research*, Vol. 3, 1985, pp. 381.
- Tomek, Pavel., "Tooling Strategies Related to FMS Management," *The FMS Magazine*, Vol. 5, No. 4, April 1986, pp.102-107.
- Valavanis, K.P., "On the Hierarchical Modelling, Analysis and Simulation of FMSs with Extended Petri nets," *IEEE Transactions on Systems, Man and Cybernetics*, 20(1), 1990, pp. 94-110.
- Valette, R., Courvoisier, M., Bigou, JM., and Albuquerque, J., "A Petri net Based Programmable Logic Controller," *Computer Applications in Production and Engineering*, E.A. Warman (editor), North-Holland Publishing Company, IFIP, 1983, pp. 103-115.
- Valvanis, K.P., "On the Hierarchical Modeling, Analysis and Simulation of FMSs with Extended Petri nets," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 20, No. 1, 1990, pp. 94-110.
- Veeramani D., Upton, D.M. and Barash, M.M., "Cutting-Tool Management in Computer-Integrated Manufacturing," *International Journal of Flexible Manufacturing Systems*, Vol. 5, NO. 2, 1992, pp.238-265.
- Venkatesh, K., and Chetty, O.V.K., "Petri Nets as an Efficient Modeling Tool for Modeling Tool Management in FMSs," *Proceedings of the 5th Annual Research Conference on Recent Advances in Robotics*, Florida Atlantic University, June 1992, pp. 583-594.

- Venkatesh, K., and E.B. Fernandez, "Object-oriented Simulation and Control of Flexible Manufacturing Systems Using Timed Petri nets and Ada", *Technical Report # TR-CSE-92-93*, Florida Atlantic University, Boca Raton, FL, 1993.
- Venkatesh, K., and Kaighobadi, M., "Modeling, Simulation, and Analysis of Flexible Assembly System Using Stochastic Petri Nets," Presented at Production and Operations Management Society annual meeting, Orlando, FL, October 18-21, 1992.
- Venkatesh, K., and M. Ilyas, "Modeling, Controlling, and Simulation of Local Area Networks for Flexible Manufacturing Systems Using Petri nets," *Computers and Indu. Eng.*, Vol. 25, No. (1-4), 1993, pp. 155-158.
- Venkatesh, K., Chetty, O.V.K., and Raju, K. R., "Simulating Flexible Automated Forming and Assembly Systems," *Journal of Material Processing and Technology*, Vol. 24, 1990, pp. 453-462.
- Venkatesh, K., Ilyas, M., "Real-time Petri Nets for Modeling, Controlling, and Simulation of Local Area Networks in Flexible Manufacturing Systems," (in press) *Computers and Industrial Engineering*, Vol. 28, No. 1, 1995.
- Venkatesh, K., Masory, O, and Jie Wu, "Simulation and Scheduling of Robots in an Flexible Factory Automated System Operating With JIT Principles Using Timed Petri nets," *Proc. of International Conference on Automation and Technology*, Taipei, Taiwan, R.O.C., July 4-6, 1992, pp. 73-80.
- Venkatesh, K., Mehdi, K., Zhou, M.C., and Caudill, R., "Augmented Timed Petri nets for Modeling of Robotic Systems with Breakdowns," *Journal of Manufacturing Systems*, Vol. 13, No. 4, 1994a, pp. 289-301.
- Venkatesh, K., O.V.K. Chetty, and V. Radhakrishnan, "Software Development for Future Unmanned Industries," *Proc. of Int. Conf. on Design Automation and Comp. Integrated Mfg.*, Coimbatore, India, 1991, pp. 80-91.
- Venkatesh, K., *Petri Nets - An Expeditious Tool for Modelling, Simulation and Analysis of Flexible Multi Robot Assembly Systems*, M. Tech. Project Report, Manufacturing Engineering Section, Indian Institute of Technology, Madras, India, 1990.
- Venkatesh, K., Zhou, M.C., and Caudill, R., "Comparing Ladder Logic Diagrams and Petri Nets for Sequence Controller Design through a Discrete Manufacturing System," (in press) *IEEE Trans. on Industrial Electronics*, 1994.
- Venkatesh, K., Zhou, M.C., Caudill, R., "A Control Software Design Methodology for CIM Systems," *Proc. of Rutgers Conf. on CIM in the Process Industries*, East Brunswick, NJ, April 25-26, 1994, pp. 565-579.
- Venkatesh, K., Zhou, M.C., Caudill, R., "Evaluating the Complexity of Petri Nets and Ladder Logic Diagrams to Design Sequence Controllers in Flexible Automation," *Proc. of Seiken/IEEE Symp. on Emerging Technologies & Factory Automation*, Tokyo, Japan, Nov. 6-10, 1994, pp. 428-435.

- Vernon, M., Zahorjan, J., and E.D. Lazowska, "A Comparison of Performance Petri nets and Queuing Network Models," *Proc. of the Int. Workshop on Modeling Techniques and Performance Evaluation*, Paris, France, 1987.
- Voltz, R., T.N. Mudge, and D. Gal, "Using Ada as a Programming Language for Robot-based Manufacturing Cells," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 14, No. 6, 1984, pp. 863-878.
- Vosniakos, G.C. and Mamalis, A.G., "Automated guided vehicle system design for FMS applications," *International Journal of Machine tools and Manufacture*, Vol. 30, 1990, pp. 85-97.
- Wang, H.P.B., and Hafeez, S.A., "Performance Evaluation of Tandem and Conventional AGV Systems Using Generalized Stochastic Petri nets," *Int. J. of Prod. Res.*, Vol. 32, No. 4, 1994, pp. 917-932.
- Wegner, P., "Capital Intensive Software Technology, Part 2: Programming in the large," *IEEE Software*, Vol. 1, No. 3, 1984, pp. 24-32.
- Wysk, R.A., Egbelu, P.J., Zhou, C. and Ghosh, B.K., 1987, "Use of Spreadsheet Analysis for Evaluating AGV Systems," *Material Flow*, Vol. 4, 1987, pp. 53-64.
- Yilmaz, O.S. and Davis, R.P., "Flexible Manufacturing Systems: Characteristics and Assessment," *Engineering Management*, Vol. 4, 1987, pp. 209-212.
- Yim, D., and Linn, R.J., "Push and Pull Rules for Dispatching Automated Guided Vehicles in a Flexible Manufacturing System," *Int. J. of Prod. Res.*, Vol. 31, No. 1, 1993, pp. 43-57.
- Zavanella, L., Maccarini, G.C. and Bugini, A., "FMS tool supply in a Stochastic Environment: Strategies and Related Reliabilities," *Intl. J. Mach. Tools Manufacturing*, Vol. 30, No. 3, 1990, pp. 389-402.
- Zhou, M. C., and DiCesare, F., "Adaptive Design of Petri net Controllers for Error Recovery in Automated Manufacturing Systems," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 19, No. 5, 1989, pp. 963-973.
- Zhou, M. C., and DiCesare, F., *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publishers, Boston, MA, 1993.
- Zhou, M. C., DiCesare, F., and Desrochers, "A Hybrid Methodology for Synthesis of Petri Net Models for Manufacturing Systems," *IEEE Trans. Robotics and Automation*, Vol. 8, No. 3, 1992b, pp. 350-361.
- Zhou, M. C., F. DiCesare, and D. Rudolph., "Design and Implementation of a Petri net Based Supervisor for a Flexible Manufacturing System," *Automatica*, Vol. 28, No. 6, 1992a, pp. 1999-2008.
- Zhou, M.C., and Leu, M.C., "Modeling and Performance Analysis of a Flexible PCB Assembly System Using Petri nets," *Transactions of ASME, Journal of Electronic Packaging*, Vol. 113, 1991, pp. 410-416.