THE UNIVERSITY OF CALGARY

# Real Time Control of Manufacturing Cells Utilizing Fuzzy Logic Part Dispatching

by

Andre Joachim Naumann

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

IN MECHANICAL ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

CALGARY, ALBERTA

AUGUST, 1994

ISBN 0-315-99434-7

Canada

Name André Naumann

*Dissertation Abstracts International* is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Engineering - Industrial
SUBJECT TERM

0596
SUBJECT CODE

U·M·I

## Subject Categories

# THE HUMANITIES AND SOCIAL SCIENCES

### COMMUNICATIONS AND THE ARTS
Architecture ..... 0729
Art History ..... 0377
Cinema ..... 0900
Dance ..... 0378
Fine Arts ..... 0357
Information Science ..... 0723
Journalism ..... 0391
Library Science ..... 0399
Mass Communications ..... 0708
Music ..... 0413
Speech Communication ..... 0459
Theater ..... 0465

### EDUCATION
General ..... 0515
Administration ..... 0514
Adult and Continuing ..... 0516
Agricultural ..... 0517
Art ..... 0273
Bilingual and Multicultural ..... 0282
Business ..... 0688
Community College ..... 0275
Curriculum and Instruction ..... 0727
Early Childhood ..... 0518
Elementary ..... 0524
Finance ..... 0277
Guidance and Counseling ..... 0519
Health ..... 0680
Higher ..... 0745
History of ..... 0520
Home Economics ..... 0278
Industrial ..... 0521
Language and Literature ..... 0279
Mathematics ..... 0280
Music ..... 0522
Philosophy of ..... 0998
Physical ..... 0523
Psychology ..... 0525
Reading ..... 0535
Religious ..... 0527
Sciences ..... 0714
Secondary ..... 0533
Social Sciences ..... 0534
Sociology of ..... 0340
Special ..... 0529
Teacher Training ..... 0530
Technology ..... 0710
Tests and Measurements ..... 0288
Vocational ..... 0747

### LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General ..... 0679
- Ancient ..... 0289
- Linguistics ..... 0290
- Modern ..... 0291

Literature
- General ..... 0401
- Classical ..... 0294
- Comparative ..... 0295
- Medieval ..... 0297
- Modern ..... 0298
- African ..... 0316
- American ..... 0591
- Asian ..... 0305
- Canadian (English) ..... 0352
- Canadian (French) ..... 0355
- English ..... 0593
- Germanic ..... 0311
- Latin American ..... 0312
- Middle Eastern ..... 0315
- Romance ..... 0313
- Slavic and East European ..... 0314

### PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy ..... 0422
Religion
- General ..... 0318
- Biblical Studies ..... 0321
- Clergy ..... 0319
- History of ..... 0320
- Philosophy of ..... 0322

Theology ..... 0469

### SOCIAL SCIENCES
American Studies ..... 0323
Anthropology
- Archaeology ..... 0324
- Cultural ..... 0326
- Physical ..... 0327

Business Administration
- General ..... 0310
- Accounting ..... 0272
- Banking ..... 0770
- Management ..... 0454
- Marketing ..... 0338

Canadian Studies ..... 0385
Economics
- General ..... 0501
- Agricultural ..... 0503
- Commerce-Business ..... 0505
- Finance ..... 0508
- History ..... 0509
- Labor ..... 0510
- Theory ..... 0511

Folklore ..... 0358
Geography ..... 0366
Gerontology ..... 0351
History
- General ..... 0578
- Ancient ..... 0579
- Medieval ..... 0581
- Modern ..... 0582
- Black ..... 0328
- African ..... 0331
- Asia, Australia and Oceania 0332
- Canadian ..... 0334
- European ..... 0335
- Latin American ..... 0336
- Middle Eastern ..... 0333
- United States ..... 0337

History of Science ..... 0585
Law ..... 0398
Political Science
- General ..... 0615
- International Law and Relations ..... 0616
- Public Administration ..... 0617

Recreation ..... 0814
Social Work ..... 0452
Sociology
- General ..... 0626
- Criminology and Penology ... 0627
- Demography ..... 0938
- Ethnic and Racial Studies ..... 0631
- Individual and Family Studies ..... 0628
- Industrial and Labor Relations ..... 0629
- Public and Social Welfare .... 0630
- Social Structure and Development ..... 0700
- Theory and Methods ..... 0344

Transportation ..... 0709
Urban and Regional Planning .... 0999
Women's Studies ..... 0453

# THE SCIENCES AND ENGINEERING

### BIOLOGICAL SCIENCES
Agriculture
- General ..... 0473
- Agronomy ..... 0285
- Animal Culture and Nutrition ..... 0475
- Animal Pathology ..... 0476
- Food Science and Technology ..... 0359
- Forestry and Wildlife ..... 0478
- Plant Culture ..... 0479
- Plant Pathology ..... 0480
- Plant Physiology ..... 0817
- Range Management ..... 0777
- Wood Technology ..... 0746

Biology
- General ..... 0306
- Anatomy ..... 0287
- Biostatistics ..... 0308
- Botany ..... 0309
- Cell ..... 0379
- Ecology ..... 0329
- Entomology ..... 0353
- Genetics ..... 0369
- Limnology ..... 0793
- Microbiology ..... 0410
- Molecular ..... 0307
- Neuroscience ..... 0317
- Oceanography ..... 0416
- Physiology ..... 0433
- Radiation ..... 0821
- Veterinary Science ..... 0778
- Zoology ..... 0472

Biophysics
- General ..... 0786
- Medical ..... 0760

### EARTH SCIENCES
Biogeochemistry ..... 0425
Geochemistry ..... 0996
Geodesy ..... 0370
Geology ..... 0372
Geophysics ..... 0373
Hydrology ..... 0388
Mineralogy ..... 0411
Paleobotany ..... 0345
Paleoecology ..... 0426
Paleontology ..... 0418
Paleozoology ..... 0985
Palynology ..... 0427
Physical Geography ..... 0368
Physical Oceanography ..... 0415

### HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences ..... 0768
Health Sciences
- General ..... 0566
- Audiology ..... 0300
- Chemotherapy ..... 0992
- Dentistry ..... 0567
- Education ..... 0350
- Hospital Management ..... 0769
- Human Development ..... 0758
- Immunology ..... 0982
- Medicine and Surgery ..... 0564
- Mental Health ..... 0347
- Nursing ..... 0569
- Nutrition ..... 0570
- Obstetrics and Gynecology .. 0380
- Occupational Health and Therapy ..... 0354
- Ophthalmology ..... 0381
- Pathology ..... 0571
- Pharmacology ..... 0419
- Pharmacy ..... 0572
- Physical Therapy ..... 0382
- Public Health ..... 0573
- Radiology ..... 0574
- Recreation ..... 0575
- Speech Pathology ..... 0460
- Toxicology ..... 0383

Home Economics ..... 0386

### PHYSICAL SCIENCES
#### Pure Sciences
Chemistry
- General ..... 0485
- Agricultural ..... 0749
- Analytical ..... 0486
- Biochemistry ..... 0487
- Inorganic ..... 0488
- Nuclear ..... 0738
- Organic ..... 0490
- Pharmaceutical ..... 0491
- Physical ..... 0494
- Polymer ..... 0495
- Radiation ..... 0754

Mathematics ..... 0405
Physics
- General ..... 0605
- Acoustics ..... 0986
- Astronomy and Astrophysics ..... 0606
- Atmospheric Science ..... 0608
- Atomic ..... 0748
- Electronics and Electricity ..... 0607
- Elementary Particles and High Energy ..... 0798
- Fluid and Plasma ..... 0759
- Molecular ..... 0609
- Nuclear ..... 0610
- Optics ..... 0752
- Radiation ..... 0756
- Solid State ..... 0611

Statistics ..... 0463

#### Applied Sciences
Applied Mechanics ..... 0346
Computer Science ..... 0984
Engineering
- General ..... 0537
- Aerospace ..... 0538
- Agricultural ..... 0539
- Automotive ..... 0540
- Biomedical ..... 0541
- Chemical ..... 0542
- Civil ..... 0543
- Electronics and Electrical ..... 0544
- Heat and Thermodynamics ... 0348
- Hydraulic ..... 0545
- Industrial ..... 0546
- Marine ..... 0547
- Materials Science ..... 0794
- Mechanical ..... 0548
- Metallurgy ..... 0743
- Mining ..... 0551
- Nuclear ..... 0552
- Packaging ..... 0549
- Petroleum ..... 0765
- Sanitary and Municipal ..... 0554
- System Science ..... 0790

Geotechnology ..... 0428
Operations Research ..... 0796
Plastics Technology ..... 0795
Textile Technology ..... 0994

### PSYCHOLOGY
General ..... 0621
Behavioral ..... 0384
Clinical ..... 0622
Developmental ..... 0620
Experimental ..... 0623
Industrial ..... 0624
Personality ..... 0625
Physiological ..... 0989
Psychobiology ..... 0349
Psychometrics ..... 0632
Social ..... 0451

Nom _____

*Dissertation Abstracts International* est organisé en catégories de sujets. Veuillez s.v.p. choisir le sujet qui décrit le mieux votre thèse et inscrivez le code numérique approprié dans l'espace réservé ci-dessous.

SUJET

CODE DE SUJET

U·M·I

## Catégories par sujets

# HUMANITÉS ET SCIENCES SOCIALES

### COMMUNICATIONS ET LES ARTS
Architecture ..... 0729
Beaux-arts ..... 0357
Bibliothéconomie ..... 0399
Cinéma ..... 0900
Communication verbale ..... 0459
Communications ..... 0708
Danse ..... 0378
Histoire de l'art ..... 0377
Journalisme ..... 0391
Musique ..... 0413
Sciences de l'information ..... 0723
Théâtre ..... 0465

### ÉDUCATION
Généralités ..... 515
Administration ..... 0514
Art ..... 0273
Collèges communautaires ..... 0275
Commerce ..... 0688
Économie domestique ..... 0278
Éducation permanente ..... 0516
Éducation préscolaire ..... 0518
Éducation sanitaire ..... 0680
Enseignement agricole ..... 0517
Enseignement bilingue et multiculturel ..... 0282
Enseignement industriel ..... 0521
Enseignement primaire ..... 0524
Enseignement professionnel ..... 0747
Enseignement religieux ..... 0527
Enseignement secondaire ..... 0533
Enseignement spécial ..... 0529
Enseignement supérieur ..... 0745
Évaluation ..... 0288
Finances ..... 0277
Formation des enseignants ..... 0530
Histoire de l'éducation ..... 0520
Langues et littérature ..... 0279
Lecture ..... 0535
Mathématiques ..... 0280
Musique ..... 0522
Orientation et consultation ..... 0519
Philosophie de l'éducation ..... 0998
Physique ..... 0523
Programmes d'études et enseignement ..... 0727
Psychologie ..... 0525
Sciences ..... 0714
Sciences sociales ..... 0534
Sociologie de l'éducation ..... 0340
Technologie ..... 0710

### LANGUE, LITTÉRATURE ET LINGUISTIQUE
Langues
- Généralités ..... 0679
- Anciennes ..... 0289
- Linguistique ..... 0290
- Modernes ..... 0291

Littérature
- Généralités ..... 0401
- Anciennes ..... 0294
- Comparée ..... 0295
- Médiévale ..... 0297
- Moderne ..... 0298
- Africaine ..... 0316
- Américaine ..... 0591
- Anglaise ..... 0593
- Asiatique ..... 0305
- Canadienne (Anglaise) ..... 0352
- Canadienne (Française) ..... 0355
- Germanique ..... 0311
- Latino-américaine ..... 0312
- Moyen-orientale ..... 0315
- Romane ..... 0313
- Slave et est-européenne ..... 0314

### PHILOSOPHIE, RELIGION ET THÉOLOGIE
Philosophie ..... 0422
Religion
- Généralités ..... 0318
- Clergé ..... 0319
- Études bibliques ..... 0321
- Histoire des religions ..... 0320
- Philosophie de la religion ..... 0322

Théologie ..... 0469

### SCIENCES SOCIALES
Anthropologie
- Archéologie ..... 0324
- Culturelle ..... 0326
- Physique ..... 0327

Droit ..... 0398
Économie
- Généralités ..... 0501
- Commerce-Affaires ..... 0505
- Économie agricole ..... 0503
- Économie du travail ..... 0510
- Finances ..... 0508
- Histoire ..... 0509
- Théorie ..... 0511

Études américaines ..... 0323
Études canadiennes ..... 0385
Études féministes ..... 0453
Folklore ..... 0358
Géographie ..... 0366
Gérontologie ..... 0351
Gestion des affaires
- Généralités ..... 0310
- Administration ..... 0454
- Banques ..... 0770
- Comptabilité ..... 0272
- Marketing ..... 0338

Histoire
- Histoire générale ..... 0578
- Ancienne ..... 0579
- Médiévale ..... 0581
- Moderne ..... 0582
- Histoire des noirs ..... 0328
- Africaine ..... 0331
- Canadienne ..... 0334
- États-Unis ..... 0337
- Européenne ..... 0335
- Moyen-orientale ..... 0333
- Latino-américaine ..... 0336
- Asie, Australie et Océanie ..... 0332

Histoire des sciences ..... 0585
Loisirs ..... 0814
Planification urbaine et régionale ..... 0999
Science politique
- Généralités ..... 0615
- Administration publique ..... 0617
- Droit et relations internationales ..... 0616

Sociologie
- Généralités ..... 0626
- Aide et bien-être social ..... 0630
- Criminologie et établissements pénitentiaires ..... 0627
- Démographie ..... 0938
- Études de l'individu et de la famille ..... 0628
- Études des relations interethniques et des relations raciales ..... 0631
- Structure et développement social ..... 0700
- Théorie et méthodes ..... 0344
- Travail et relations industrielles ..... 0629

Transports ..... 0709
Travail social ..... 0452

# SCIENCES ET INGÉNIERIE

### SCIENCES BIOLOGIQUES
Agriculture
- Généralités ..... 0473
- Agronomie ..... 0285
- Alimentation et technologie alimentaire ..... 0359
- Culture ..... 0479
- Élevage et alimentation ..... 0475
- Exploitation des pâturages ..... 0777
- Pathologie animale ..... 0476
- Pathologie végétale ..... 0480
- Physiologie végétale ..... 0817
- Sylviculture et faune ..... 0478
- Technologie du bois ..... 0746

Biologie
- Généralités ..... 0306
- Anatomie ..... 0287
- Biologie (Statistiques) ..... 0308
- Biologie moléculaire ..... 0307
- Botanique ..... 0309
- Cellule ..... 0379
- Écologie ..... 0329
- Entomologie ..... 0353
- Génétique ..... 0369
- Limnologie ..... 0793
- Microbiologie ..... 0410
- Neurologie ..... 0317
- Océanographie ..... 0416
- Physiologie ..... 0433
- Radiation ..... 0821
- Science vétérinaire ..... 0778
- Zoologie ..... 0472

Biophysique
- Généralités ..... 0786
- Medicale ..... 0760

### SCIENCES DE LA TERRE
Biogéochimie ..... 0425
Géochimie ..... 0996
Géodésie ..... 0370
Géographie physique ..... 0368
Géologie ..... 0372
Géophysique ..... 0373
Hydrologie ..... 0388
Minéralogie ..... 0411
Océanographie physique ..... 0415
Paléobotanique ..... 0345
Paléoécologie ..... 0426
Paléontologie ..... 0418
Paléozoologie ..... 0985
Palynologie ..... 0427

### SCIENCES DE LA SANTÉ ET DE L'ENVIRONNEMENT
Économie domestique ..... 0386
Sciences de l'environnement ..... 0768
Sciences de la santé
- Généralités ..... 0566
- Administration des hôpitaux ..... 0769
- Alimentation et nutrition ..... 0570
- Audiologie ..... 0300
- Chimiothérapie ..... 0992
- Dentisterie ..... 0567
- Développement humain ..... 0758
- Enseignement ..... 0350
- Immunologie ..... 0982
- Loisirs ..... 0575
- Médecine du travail et thérapie ..... 0354
- Médecine et chirurgie ..... 0564
- Obstétrique et gynécologie ..... 0380
- Ophtalmologie ..... 0381
- Orthophonie ..... 0460
- Pathologie ..... 0571
- Pharmacie ..... 0572
- Pharmacologie ..... 0419
- Physiothérapie ..... 0382
- Radiologie ..... 0574
- Santé mentale ..... 0347
- Santé publique ..... 0573
- Soins infirmiers ..... 0569
- Toxicologie ..... 0383

### SCIENCES PHYSIQUES
#### Sciences Pures
Chimie
- Généralités ..... 0485
- Biochimie ..... 487
- Chimie agricole ..... 0749
- Chimie analytique ..... 0486
- Chimie minérale ..... 0488
- Chimie nucléaire ..... 0738
- Chimie organique ..... 0490
- Chimie pharmaceutique ..... 0491
- Physique ..... 0494
- PolymÇres ..... 0495
- Radiation ..... 0754

Mathématiques ..... 0405
Physique
- Généralités ..... 0605
- Acoustique ..... 0986
- Astronomie et astrophysique ..... 0606
- Electronique et électricité ..... 0607
- Fluides et plasma ..... 0759
- Météorologie ..... 0608
- Optique ..... 0752
- Particules (Physique nucléaire) ..... 0798
- Physique atomique ..... 0748
- Physique de l'état solide ..... 0611
- Physique moléculaire ..... 0609
- Physique nucléaire ..... 0610
- Radiation ..... 0756

Statistiques ..... 0463

#### Sciences Appliqués Et Technologie
Informatique ..... 0984
Ingénierie
- Généralités ..... 0537
- Agricole ..... 0539
- Automobile ..... 0540
- Biomédicale ..... 0541
- Chaleur et ther modynamique ..... 0348
- Conditionnement (Emballage) ..... 0549
- Génie aérospatial ..... 0538
- Génie chimique ..... 0542
- Génie civil ..... 0543
- Génie électronique et électrique ..... 0544
- Génie industriel ..... 0546
- Génie mécanique ..... 0548
- Génie nucléaire ..... 0552
- Ingénierie des systèmes ..... 0790
- Mécanique navale ..... 0547
- Métallurgie ..... 0743
- Science des matériaux ..... 0794
- Technique du pétrole ..... 0765
- Technique minière ..... 0551
- Techniques sanitaires et municipales ..... 0554
- Technologie hydraulique ..... 0545

Mécanique appliquée ..... 0346
Géotechnologie ..... 0428
Matières plastiques (Technologie) ..... 0795
Recherche opérationnelle ..... 0796
Textiles et tissus (Technologie) ..... 0794

### PSYCHOLOGIE
Généralités ..... 0621
Personnalité ..... 0625
Psychobiologie ..... 0349
Psychologie clinique ..... 0622
Psychologie du comportement ..... 0384
Psychologie du développement ..... 0620
Psychologie expérimentale ..... 0623
Psychologie industrielle ..... 0624
Psychologie physiologique ..... 0989
Psychologie sociale ..... 0451
Psychométrie ..... 0632

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Real Time Control of Manufacturing Cells Utilizing Fuzzy Logic Part Dispatching" submitted by Andre Joachim Naumann in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering.

Dr. P. Gu
Supervisor and Committee Chairman, Department of Mechanical Engineering

Prof. A.A. Torvi
Department of Mechanical Engineering

Dr. P. Wojcik
Alberta Research Council, Adjunct to the Department of Mechanical Engineering

Dr. J. Balakrishnan
Faculty of Management

September 1, 1994
Date

# ABSTRACT

This thesis presents a manufacturing cell control structure and fuzzy logic part dispatching method which represent original contributions to the field of manufacturing cell control. The control structure consists of a supervisor module comprised of primary and secondary control levels, and a scheduler module containing a fuzzy logic part dispatching method. The fuzzy logic part dispatching method considers many aspects of the shop floor and shows better performance in terms of the number of late parts and average buffer maximum loading than five common dispatching rules (EDD, LIFO, FIFO, SPT, Slack/OPNR). The control structure has proven to be very expandable and flexible, deals effectively with breakdowns and disruptions in real time, and also incorporates mechanisms whereby part dispatching can be optimized. The structure is portable and can be used at different control levels as well as in a decentralized control environment. Finally the structure supports the concept of intelligent part and machine entities.

# ACKNOWLEDGEMENTS

I extend a very special thank you to Dr. P. Gu for his professional guidance and support throughout this work. I also wish to acknowledge him for his encouragement and for the expressions of confidence in my abilities that he extended to me throughout the project in order to support me in searching for answers and dealing with problems as they arose.

Thanks are extended to Prof. A.A. Torvi, Dr. P. Wojcik, and Dr. J. Balakrishnan, the examination committee members, for devoting their time and energy toward the thesis.

Finally, the financial support provided by the Alberta Heritage Scholarship Fund, the Natural Sciences and Engineering Research Council of Canada (NSERC) through grant #OGP0105754, Industry Canada through the Intelligent Manufacturing Systems research program grants, and the Department of Mechanical Engineering through teaching assistantships, has been very much appreciated and has made this endeavour possible for me.

**Dedicated to my Parents**

# TABLE OF CONTENTS


| | | Page |
|---|---|---|
| Approval Sheet | | ii |
| Abstract | | iii |
| Acknowledgements | | iv |
| Dedication | | v |
| Table of Contents | | vi |
| List of Tables | | x |
| List of Figures | | xi |
| **Chapter 1** | **Introduction** | 1 |
| 1.1 | Control and Scheduling Structures | 1 |
| 1.2 | Problem Statement | 4 |
| 1.3 | Research Objective | 5 |
| 1.4 | Research Approach | 6 |
| 1.5 | Organization of the Thesis | 8 |
| **Chapter 2** | **Literature Review** | 10 |
| 2.1 | Introduction | 10 |
| 2.2 | Control Structures | 11 |
| 2.3 | Artificial Intelligence Control Techniques | 17 |
| 2.4 | Artificial Intelligence Scheduling Techniques | 22 |

**Chapter 3** **General Overview of the Proposed Control Structure** 28

**Chapter 4** **Scheduler** . . . . . 37

4.1 Introduction . . . . . 37

4.2 Fuzzy Logic Method Sub-Module . . . 39

4.2.1 Background . . . . . 39

4.2.2 Theory . . . . . 40

4.2.3 Application and Example . . . 51

4.3 Dispatching Rule Sub-Module . . . 64

**Chapter 5** **Supervisor** . . . . . 68

5.1 General Functionality . . . . . 68

5.2 Supervisory Tasks . . . . . 72

5.2.1 Part Movement and Part Production . . 72

5.2.2 Material Handling . . . . . 73

5.2.3 Machine Control and Machine Status Monitoring . . . . . 74

5.2.4 Error Recovery . . . . . 75

5.2.5 Virtual Reconfiguration . . . 75

5.2.6 Cell Buffer and Inventory Monitoring . 76

5.3 Secondary Control Level . . . . . 76

5.3.1 Petri Net Description . . . . . 77

5.3.2 Petri Net for Loading a Machine . . 79
5.3.3 Petri Net for Operating a Machine . . 82
5.3.4 Petri Net for Moving Parts . . . 84
5.3.5 Petri Net for Unloading a Machine . . 84

**Chapter 6** **Implementation** . . . . . 88
6.1 Introduction . . . . . 88
6.2 Object Oriented Class Hierarchy . . . 89
6.2.1 System Class . . . . . 90
6.2.2 Cell Class . . . . . 92
6.2.3 Equipment Class . . . . . 94
6.2.4 Tool Class . . . . . 96
6.2.5 PartType Class . . . . . 96
6.2.6 Part Class . . . . . 99
6.2.7 Class Linkages . . . . . 101
6.3 Methods and Message Passing . . . 103
6.3.1 Supervisor Methods . . . . 103
6.3.2 Scheduling Methods . . . . 104
6.3.3 Fuzzy Logic Class Methods . . . 105
6.4 Simulation Model . . . . . 106
6.5 Operation of the Simulation . . . . 108

**Chapter 7** **Case Studies** . . . . . 111

7.1 Introduction . . . . . 111

7.2 Case 1: Fuzzy Logic Base Case . . . 115

7.3 Case 2: Alternate Process Plan . . . 125

7.4 Case 3: Large Cases . . . . . 128

7.5 Case 4: Finite Buffer Limit Study . . . 135

7.6 Case 5: Machine Breakdown and Repair . . 143

7.7 Case 6: Reconfigurable Case . . . 147

7.8 Case 7: Robot Movement . . . . 149

7.9 Case 8: Weight Optimization . . . 151

**Chapter 8** **Conclusions** . . . . . 160

8.1 General Discussion . . . . . 160

8.2 Supervisor Discussion . . . . 161

8.3 Scheduler Discussion . . . . 163

8.4 Original Contribution . . . . 166

8.5 Final Summary . . . . . 168

8.6 Future Research . . . . . 169

**References** . . . . . 171

**Appendix A: Equipment Class Instances** . . . . 178

**Appendix B: Tool Class Instances** . . . . 181

**Appendix C: PartType Class Instances** . . . . 184

# LIST OF TABLES


| | | Page |
|---|---|---|
| Table 4.1: | Centroid Calculation Values . . . . . | 63 |
| Table 7.1: | Base Case Machine Data . . . . . | 119 |
| Table 7.2: | Base Case Part Data . . . . . | 120 |
| Table 7.3: | Base Case Rule Comparison . . . . . | 121 |
| Table 7.4: | Alternate Process Plan Case Machine Data . . . . | 126 |
| Table 7.5: | Alternate Process Plan Case Part Data . . . . | 127 |
| Table 7.6: | Alternate Process Plan Case Rule Comparison . . | 129 |
| Table 7.7: | Large Case 1 Machine Data . . . . . | 129 |
| Table 7.8: | Large Case 1 Part Data . . . . . | 130 |
| Table 7.8: | Large Case 1 Part Data (continued) . . . . . | 131 |
| Table 7.9: | Large Case 2 Machine Data . . . . . | 132 |
| Table 7.10: | Large Case 2 Part Data . . . . . | 133 |
| Table 7.10: | Large Case 2 Part Data (continued) . . . . . | 134 |

# LIST OF FIGURES


| | | Page |
|---|---|---|
| Figure 3.1: | Cell Controller & Interactions . . . . | 30 |
| Figure 3.2: | Supervisor and Scheduler Sub-Modules . . . | 32 |
| Figure 3.3: | Class Hierarchy . . . . . | 36 |
| Figure 4.1a: | Crisp Set of Temperature . . . . . | 42 |
| Figure 4.1b: | Fuzzy Set of Temperature . . . . . | 42 |
| Figure 4.2a: | Correlation Minimum Encoding . . . . | 48 |
| Figure 4.2b: | Correlation Product Encoding . . . . | 48 |
| Figure 4.3: | Fuzzy Logic Rules . . . . . | 53 |
| Figure 4.4: | Fuzzy Logic Membership Functions . . . | 57 |
| Figure 4.5: | Application of Fuzzy Logic Methodology . . . | 60 |
| Figure 5.1: | Primary Control level . . . . . | 70 |
| Figure 5.2: | Petri Net for Loading a Machine . . . . | 80 |
| Figure 5.3 | Petri Net for Operating a Machine . . . . | 83 |
| Figure 5.4: | Petri Net for Moving Parts . . . . . | 85 |
| Figure 5.5: | Petri Net for Unloading a Machine . . . . | 86 |
| Figure 6.1: | An Instance of Class 'System' . . . . | 91 |
| Figure 6.2: | An Instance of Class 'Cell' . . . . . | 93 |
| Figure 6.3: | An Instance of Class 'Equipment' . . . . | 95 |
| Figure 6.4: | An Instance of Class 'Tool' . . . . . | 97 |
| Figure 6.5: | An Instance of Class 'PartType' . . . . | 98 |

Figure 6.6: An Instance of Class 'Part' . . . . . 100

Figure 6.7: The Overall Class Structure . . . . . 102

Figure 7.1: Manufacturing Cell . . . . . 112

Figure 7.2: Screen View at SimTime = 32 Minutes . . . 117

Figure 7.3: Screen View at SimTime = 33 Minutes . . . 118

Figure 7.4: Inventory Levels, Part Types D,H,I, and J . . 123

Figure 7.5: Buffer Level, VertMill2 . . . . . 124

Figure 7.6: Comparison of Number of Late Parts . . . 136

Figure 7.7: Comparison of Maximum Part Lateness . . . 137

Figure 7.8: Comparison of Average Tardiness . . . . 138

Figure 7.9: Comparison of Average Production Times . . . 139

Figure 7.10: Comparison of Average Maximum Machine Buffer Levels . . . . . 140

Figure 7.11: Buffer Levels of Drill1 . . . . . 144

Figure 7.12: Buffer Levels of Drill2 . . . . . 145

Figure 7.13: Inventory Levels (Drill1 Normal and Broken) . . 146

Figure 7.14: Inventory Levels for Reconfigurable Case . . . 148

Figure 7.15: Screen View of Robot Action at SimTime = 66 Minutes . 150

Figure 7.16: Effect of Scheduler and Process Step Weight Modification on Number of Late Parts . . . 153

Figure 7.17: Effect of Buffer Level and Process Step Weight Modification on Number of Late Parts . . . 154

Figure 7.18: Effect of Scheduler and Buffer Level Weight Modification on Number of Late Parts . . . 155

Figure 7.19: Effect of Scheduler and Process Step Weight Modification on Average Production Time . . . 156

Figure 7.20: Effect of Buffer Level and Process Step Weight Modification on Average Production Time . . . 157

Figure 7.21: Effect of Scheduler and Buffer Level Weight Modification on Average Production Time . . . 158

# CHAPTER 1

## 1. INTRODUCTION

## 1.1 CONTROL AND SCHEDULING STRUCTURES

The current trend of manufacturing is towards the production of a greater variety of parts in smaller volumes which is in direct contrast to the high rate, high volume, fixed production facilities of the past. Business trends towards shortened product life cycles, greater product customization and the need for a larger variety of products is driving production facilities away from fixed automation plants. As a result of these trends, the requirement for increased flexibility in manufacturing systems and control structures has been steadily increasing. Many of the facilities operating today utilize flexible manufacturing systems (FMS) which are integrated manufacturing systems consisting of a variety of automated pieces of manufacturing equipment linked together with communication and material handling systems. These present day flexible manufacturing

systems are often controlled through a highly integrated and interconnected computer controlled hierarchical structure. It has recently been suggested that as the current hierarchical control structures grow in size and complexity, they may not be able to effectively provide the necessary flexibility to deal with the degree of changes in product demand and product variety that are presently occurring. As a result, several authors have advocated the use of heterarchical control structures which purport to have increased flexibility and fault tolerance (Hatvany [13], Duffie and Piper [8], Jones and Saleh [17]). Furthermore, the degree of complexity of the control and scheduling problems have prompted a number of developments in the area of intelligent control and scheduling. These developments, which promise to ultimately improve the flexibility and power of manufacturing systems, have involved the application of expert systems (Wu and Wysk [43]), neural networks (Cho and Wysk [3]), fuzzy logic (Custodio et al. [6]) genetic algorithms (Herrmann et al. [14]) and Petri nets (Huang and Chang [15]). There are still, however, many difficulties that need to be overcome before a highly flexible, autonomous and possibly intelligent control system can be realised.

One area of concern is the difficulty in dealing with disruptions in the manufacturing environment. Many of the existing and proposed control structures are very rigid and therefore encounter difficulties when things do not run smoothly. Schedules which are often developed under static shop floor conditions fail when those shop floor conditions change unexpectedly. Since there will always be dynamic changes in the manufacturing environment, systems for real time control and scheduling need to

be developed which can account for and deal with these changes and disruptions. The intent of this particular research is to develop a control package consisting of supervisory and scheduling aspects, which has the ability to deal with disruptions in real time and also which maintains a high degree of flexibility. The ultimate vision is to have a manufacturing system where all the entities such as parts or machines are independent of one another. These entities would possess certain knowledge, some degree of intelligence, and would be able to work and communicate in an autonomous way with the other entities in the manufacturing environment as they carry out their assigned tasks. This would result in very high level of flexibility; however, care must be taken that the overall goals of the system are still met and that they are met efficiently. It is unrealistic to expect that the implementation of independent entities can be done in this one project, but the concept is one of the underlying precepts used for developing the proposed control structure. A second underlying precept was the feeling that supervision and scheduling are often indistinguishable. Therefore, in this work both the scheduling and supervisory functions were, to a certain degree, blended together in a effort to find a more appropriate way to increase flexibility and fault tolerance in the manufacturing environment.

The concept of intelligent supervision and scheduling has also been applied in this project. The idea of an intelligent system implies reasoning and the ability to make decisions even as conditions change. Furthermore, it implies the ability to learn, or to improve and optimize a decision strategy. The goal of this project is to develop a system which contains mechanisms for the system to be improved or optimized. Thus the system

may not follow a rigid structure or plan such as a fixed schedule or control sequence, but rather adjusts its function according to the current situation on the manufacturing floor. In a sense, this may make the system somewhat unpredictable; however, it will also make the system more flexible. It is anticipated that the unpredictability can be resolved using simulation techniques similar to those used when modelling a shop floor system with existing discrete event simulation techniques.

One final intent of the proposed control structure is to provide a scheduling or part dispatching methodology that considers a wider range of constraints and variables than current dispatching rules. Dispatching rules typically have been developed to provide optimum performance for certain parameters given a specific cell or system configuration (Kusiak [27], French [9], Blackstone et al.[2]). Thus one dispatching rule may work very well given one type of shop floor configuration, but may work very poorly under a different configuration. The goal of the proposed structure is to develop a more general, less sensitive approach.

## 1.2 PROBLEM STATEMENT

It has been proposed that as current scheduling and control structures grow in size and complexity, they become too inflexible to deal quickly and effectively with product changes, schedule changes or disruptions in the manufacturing environment. As a result, there is a need for the development of faster responding and more flexible real time

control structures. As well, in order to fully and successfully implement these new control structures, current scheduling methods need to be improved to account for multiple concurrent constraints and objectives.

## 1.3 RESEARCH OBJECTIVE

The overall goal is to develop a highly flexible real time control structure that can be applied at the manufacturing cell level and which is capable of dealing with shop floor disruptions. The intent is to build the structure with a framework able to ultimately support autonomous intelligent entities and which will fit into a distributed manufacturing control structure. In the cell control structure, supervisory aspects will be linked with scheduling considerations in order to provide real time scheduling and supervision which can account for changes in the manufacturing environment. The system will be designed to deal in real time with deviations to existing schedules caused by machine or tool breakages. In order to deal with breakdowns a new part dispatching method will be developed which will have increased flexibility and will consider more aspects of the manufacturing environment than several of the currently available dispatching methods. The control structure will be designed to provide regular ongoing control and part dispatching activities such as: part movement, part production, control of material handling equipment, control of machines, monitoring machine and tool status, error recovery, monitoring cell input and output buffers, and monitoring part inventory levels. The system will also incorporate the concept of virtual configuration and will be designed

to be expandable.

The focus of this project will be at the manufacturing cell level. Specifically, the control of one cell will be addressed where the cell consists of a number of different part processing machines and a material handling system. The cell is considered to be buffer constrained.

## 1.4 RESEARCH APPROACH

The development of the cell control structure will follow a decentralized hierarchical approach whereby the cell controller will operate autonomously and receive minimal supervision from other control levels. The goal is to have cell level distributed decision making and control throughout the shop floor while still ensuring compliance with a common plan or overall schedule. In order to do this, a cell controller consisting of a supervisor and a scheduler is envisioned, where the supervisor and scheduler work together to control a manufacturing cell with minimal interference from a higher level of control. The higher level of control would only be responsible for setting goals and not in controlling the ongoing activities of the cell. Within the cell controller structure, the intention is to develop a framework whereby autonomous part and machine agents could ultimately interact in a heterarchical environment. The supervisor will have a two level structure consisting of a primary and secondary control level. The primary level will address general control issues, whereas the secondary level will address more detailed

control. This layering will provide some modularity to the structure which will make the inclusion of more elements into the system or modification of the system easier. The scheduler will include conventional dispatching rules for comparison purposes but will mainly operate using a new fuzzy logic dispatching method.

The entire control structure will be developed using the object oriented paradigm, Smalltalk 80. The object oriented environment closely resembles the real world and facilitates the development of individual entities which are able to contain attributes, knowledge and reasoning abilities. A simplified discrete event simulation will also be developed in order to test the control structure.

In order to restrict the scope of the project to a manageable level the following assumptions have been made:

1) Each type of part is assumed to have its own process plan that has been developed elsewhere but is available for the structure to use.
2) The system that will be modelled is a manufacturing cell consisting of a number of machines having finite buffer capacities. Two scenarios will be considered: a) material handling is included within the process plan and therefore part movement is not considered, and b) material handling and part movement are considered and controlled by the control structure.
3) It is assumed that a master schedule exits and that due dates and part requirements have been given.

4) The simulation will model production of several different types of parts of given batch sizes. The cell begins empty and the simulation runs until all parts are completed and the cell is empty again. Since the intent of the project is simply to test new concepts, the issues regarding simulation warm up and statistical sampling will, to a large degree, be ignored.
5) Blocking and deadlocking issues will not be addressed except to the extent that an error recovery message will be enabled.
6) Machine and tools will be deterministically broken to show how the control structure resolves a breakdown. The control structure will not contain a breakdown recovery portion but will instead perform scheduling and control in a manner which will ensure that parts are still being produced.
7) The control structure will not be optimized to any degree, but rather mechanisms for optimization will be developed and shown to work.
8) The fuzzy logic dispatching method will be compared to certain dispatching rules with respect to several variables (defined later in this thesis).
9) The focus is on minimizing part production time and the number of late parts while avoiding deadlocking problems.

## 1.5 ORGANIZATION OF THE THESIS

The thesis is organized into the following chapters. Chapter 2 consists of a literature review of a number of relevant control and scheduling methods. Chapter 3

provides a brief overview of the proposed cell control structure. Chapter 4 lays the ground work for fuzzy logic and describes the fuzzy logic part dispatching method in detail. Chapter 5 provides an in depth discussion of the supervisor structure and its different control levels. The chapter also describes the various task requirements of the supervisor and how they are performed. Chapter 6 begins with a brief discussion of the object oriented programming system and continues with a fairly in depth description of how the control structure is implemented and how data is obtained. Chapter 7 presents a number of case studies and comparison studies which are designed to show how the various supervisory and scheduling aspects of the control structure work, and how the structure can be optimized. Chapter 8 provides some additional discussions about the control structure, presents some conclusions, and discusses future research possibilities.

# CHAPTER 2

## 2.0 LITERATURE REVIEW

## 2.1 INTRODUCTION

The thrust of this project was to develop a new control structure which had more autonomy and flexibility than existing structures. In order to provide a base for comparison a number of existing techniques will be discussed which include hierarchical and heterarchical structures, artificial intelligence techniques and intelligent agents. Many of the papers that will be presented include concepts which were used in this project's control structure; those that do not, have been included for completeness and continuity. Ultimately, fuzzy logic was chosen to enable part dispatching, and a hierarchical control structure incorporating Petri nets was developed which supported the concept of intelligent objects (agents). The reasoning behind these choices will be left for discussion in later chapters, since this chapter is presented more as a grounding of available

techniques and concepts. The chapter will begin with an overview of some of the research done on general control systems, followed by discussions on the use of artificial intelligence in control systems and scheduling.

## 2.2 CONTROL STRUCTURES

Although the intent of this project is to move away from a centralized hierarchical control structure and to develop a more autonomous decentralized heterarchical structure, much work has been done on hierarchical structures and it is useful to review some of these efforts since they have provided important insights which can aid in the development of future systems. Several models have been developed to represent this hierarchical control structure, however, the most common model in use today is the Computer Integrated Manufacturing (CIM) reference model which contains the following five levels (from top to bottom): factory, shop floor/system, cell, workstation, and equipment (Joshi and Smith [18], Kals [19], Jones and McLean [16]). This type of structure allows for the development of an ever increasing detailed degree of control from the factory level down to the equipment level while maintaining a uniform plan of action.

Some of the control systems which fit neatly into a hierarchical control structure of this type are those developed using Petri nets (PN). Kasturia et al. [22], modelled a cell controller using coloured Petri nets (CPN) which consisted of five separate blocks entitled: process orders, scheduler, dispatcher, system status and material manager. These

blocks get the order requests from the shop control level, schedule the orders according to the system status and material availability, and then dispatch the final schedule and material to the appropriate workstation for processing. The beauty of modelling a control system as a PN or CPN lies in the ability to mathematically analyze the model to ensure all operations are valid and that the system is deadlock free before actual implementation. After analysis, the CPN described above was used for real time control and scheduling of a cell consisting of a CNC (computer numerically controlled) machining workstation and an assembly workstation.

Huang and Chang [15] also sought to control a manufacturing cell using PNs and developed a coloured timed Petri net (CTPN) for this purpose. Their CTPN is characterized by inhibitor arcs and interrupt arcs where error diagnosis and concurrent processing can be carried out in deterministic time. The CTPN models the CNC machines and robots within a cell, controls the loading, unloading and processing of parts, deals with error recovery, and drives the overall simulation. A cell having a capacity of six jobs and consisting of one CNC lathe, one CNC milling machine, one robot, and pallet storage was successfully controlled by the CTPN model. Ten common dispatching rules were used to dispatch the parts.

Teng and Black [39] used PNs to model a manufacturing cell consisting of three CNC machines, two decouplers, one inspection station and a robot. A decoupler was used to represent equipment placed in a cell to perform functions that workers would

normally perform such as: work-in-process inspection, inventory control, part manipulation and intercell transportation and provides a way of incorporating human activity into the control structure. The control system was developed to represent a pull system such as a Just In Time or Kanban system and it is unique in that it incorporates a number of message places which represent the pull system control. Furthermore, the PN includes defect detection and correction as well as tool failure or machine breakdown detection. Another PN model which deals with exception handling and breakdowns in manufacturing cell control was presented by Hasegawa et al. [12] They discussed a two layered PN structure consisting of a high (supervisory) layer which represents interruptive processes permitting exception handling, and a low (operation) layer which represents normal production processes. A mode token flows on the supervisory layer and is used to enable an operation layer. A process token flows on the operation layer thereby activating all detailed operations. The operation layer is connected to a place on the supervisory layer and is active only when a mode token exists in its supervisory place. The difference between an ordinary layered PN and this layered PN (LPN) lies in the behaviour of the tokens. In the ordinary PN, each PN can be decomposed to the lowest network and all networks are continuous. However the LPN is unique in that the networks among nodes are discrete and two kinds of tokens flow on different layers.

As the above examples indicate, Petri nets can indeed be powerful tools which can be used to successfully model, analyze and control manufacturing systems. Several aspects of these examples have been incorporated into the proposed control structure;

however, in order to move away from centralized hierarchical structures, other forms of control must also be developed.

Three systems which have been developed in an effort to move away from centralized control structures towards more decentralized structures are: the Production Control System (PCS) (Curtis and Tiemersma [5]) developed by the ESPRIT 809 project (European Strategic Program for Research and Development in Information Technology); the Production Activity Control system (PAC) (Bauer et al. [1]) developed by the ESPRIT 477 (COSIMA) project, and a production cell control structure developed at Laboratoire Universitaire de Recherche en Production Automatisee (LURPA) (Gendreau et al. [10]). The PCS, system which deals with shop floor control at the cell, workstation and equipment level, consists of five main modules: scheduling, dispatching, workstation control, station control, and monitoring and diagnosis. Production orders and process plans are received from the system level of the CIM reference model and these are used by the scheduler to create a work plan, which is simply a sequence of jobs to be executed in the cell. The work plans developed by the scheduling module provide the commands which the dispatching module uses along with shop floor status information to release jobs to the different elements within the cell. The workstation control module supplies the workstation with job information, coordinates the tasks of the equipment within the workstation, and monitors the status of the workstation. The station control module provides the same functionality as the workstation control module but its focus is on auxiliary manufacturing functions such as tool, jig and fixture presetting, and material

storage. The monitoring and diagnostic module provides cell status information to other modules and also supplies performance data to the system level. In a normally functioning cell, scheduling is initiated when the ongoing work plan is almost finished; however, in the case of disturbances within the cell or the insertion of a high priority order, a rescheduling decision is taken by the cell supervisor and the scheduling module is used to predict problems which could propagate from the disturbance. Thus the PCS system provides a mechanism of converting system level production orders into action at the equipment level both under normal operating circumstances and during disruptions.

The COSIMA project developed an architecture consisting of a factory coordination (FC) level and the Production Activity Control (PAC) level in an effort to ensure both flexibility and decentralization in shop floor control related activities. The PAC system sits at the cell control level and appears to have many aspects in common with the PCS. The PAC system consists of five modules: a scheduler, a dispatcher, a monitor, movers and producers. The scheduler accepts production requirements from a higher planning system (FC level) and develops a detailed plan which determines the precise use of the different facilities over a specified time frame. The scheduler checks the system capacity, generates a schedule and then releases the schedule to the dispatcher for implementation. The main purpose of the dispatcher is to react to the current state of the production environment and to select the best course of action in order to fulfil the plan developed by the scheduler. The dispatcher is the controlling element of the PAC and works in real time as it receives information, analyses the various alternatives and

broadcasts its decisions to the movers and the producers. The movers and producers are the elements which effect movement and production of the various parts and jobs. The role of the monitor is to supply cell status information to the scheduler and dispatcher so they can carry out their respective planning and control tasks. Overall, the PAC has modularized the cell control structure into a form similar to the PCS.

The third control structure that will be discussed is the LURPA production cell controller. The LURPA controller is made up of four functional components: a scheduler, a driver, a communicator and an information collector. The cell driver manages, in real time, the activities of each piece of equipment (or operating system) in the cell. It also monitors the behaviour of the cell using information gathered by the information collector. The combination of the driver, the information collector and the operating system controllers constitute the control loop of the cell. The role of the cell scheduler is to use the production requirements supplied by a higher control level, to build a schedule which is then implemented by the driver. The schedule is comprised of a series of operative and communication tasks and is developed using management production rules or sub-contractor know-how. The communicator is used to send requests to service cells within the workshop in order to fulfil transportation or manual preparation needs.

All three of these control structures (PCS, PAC, LURPA) have attempted to move away from a strictly centralized control structure. They have done so by providing each cell controller with the ability to control and schedule its own cell independently of other

cells while relying only upon a general schedule or production plan supplied by a higher control level. Each cell control structure is somewhat generic and can be used for a variety of different types of cells and this lends an aspect of portability to the control structure. As will be shown later, the control structure proposed in this thesis mimics a portable decentralized control structure.

## 2.3 ARTIFICIAL INTELLIGENCE CONTROL TECHNIQUES

Artificial intelligence (AI) techniques are playing an increasingly more important role in the control and scheduling of manufacturing systems. Expert systems, for example, are used to improve decision making procedures as well as to expand the flexibility of systems and have been used by a number of authors in their control systems. Maimon [31] developed a manufacturing system which incorporates a process sequencer that is driven by a small expert system. The actual control structure is made up of three hierarchical levels: a scheduler, a process sequencer, and a dynamic resource allocator. Based on the manufacturing status and the production state of the part, the process sequencer infers the next process, the appropriate material handling move and the production program to down load. The process sequencer expert system consists of knowledge base components containing facts, rules, a module which tracks the system status and production state, and an inference engine. As a result of using the expert system a feasible set of actions can be successfully developed both during normal operation and in the event of failures. Thus, the control system contains its own expertise

and can rely less on human operators for assistance.

Domenikos and Tatsiopoulos [7] take the concept of an expert control system one step further. They represent an entire shop floor environment using four linked knowledge based systems (KBS) developed within an object oriented paradigm. Two knowledge based systems contain information and knowledge related to generic shop floor environments and specific production methods. The knowledge and experience of shop floor management is represented in a third KBS and is used to develop general production control requirements which are then used in a fourth KBS to create feasible instructions and control rules for carrying out the management of shop floor production operations. Their goal was to use expert system methodology to provide, in a suitable form, the needed shop floor control knowledge to allow a supervisory control system to perform its control tasks. This is somewhat different than the approach taken by Maimon where the expert system was used in a more direct way to control the activities on a shop floor. The KBOLS (Knowledge-Based On-Line Simulator) architecture developed by Manivannan and Banks [33] also uses expert systems to directly control the shop floor environment. The architecture includes: a knowledge based controller capable of interacting with the shop floor and a manufacturing simulator, a shared black board data structure, and a learning module. The knowledge based controller is especially designed for analyzing and dealing with interruptions due to machine breakdowns and rush orders. The controller consists of two inference engines; one whose primary purpose is to diagnose the fault and the second which is a forward chain inferencer that searches for

a control decision. A black board environment is used to link the various knowledge bases and the entire architecture uses simulation, knowledge, and learning to generate an effective applicable control decision.

The black board environment is also used for intelligent control in a framework called PLATO-Z (Production Logistics And Timing OrganiZer). In PLATO-Z (O'Grady and Lee [35]) four black boards were developed to perform scheduling, dispatching, error handling and monitoring. O'Grady and Lee used a multi-black board/actor model in order to increase control flexibility because they felt that expert systems used on their own were too rigid. PLATO-Z uses knowledge sources (rule bases), heuristic algorithms, and optimizing procedures in this new environment in an effort to provide a control structure which has a certain degree of intelligence and promotes the use of autonomous actors that communicate through message passing.

The concept of autonomous actors leads into the possibility of having independent agents which perform shop floor control. The black board/actor system developed by O'Grady and Lee was implemented using an object oriented language, which appears to be the language of choice when developing independent actor/agent systems. Since the object oriented programming language and autonomous agents seem to go hand in hand, they will be discussed next in the context of shop floor control.

O'Grady and Seshadri [36] presented X-Cell, a cell control system which uses

object oriented programming to map cell control functions onto objects. X-Cell contains three main modules: a scheduler, an operation dispatcher and a monitor. Each of these modules were implemented as a collection of objects, with messages sent between objects within each module and also between objects within different modules. Several advantages of X-Cell include: a relatively straight forward adaption to control other cells, the provision of error handling, and, as a result of the object and message passing structure, an ability to easily transfer the system to a multi-processing environment.

The independence of the various objects within an object oriented environment was even more fully utilized by Maley [32], and LeFrancois and Montreuil [29]. Maley introduces the use of negotiation between intelligent parts and intelligent workstations in order to manage the flow of parts throughout the system. Each part maintains its own data base containing quality control histories, performance measures, due dates, and process plans. The parts communicate with the workstation to determine the workstation's processing abilities, current load, historical quality control characteristics and estimated completion time in order to decide whether to request the workstation for processing or not. Workstations communicate with parts to determine the type of operation to be performed, the material being used, and the part's priority. Multiple parts undergo the negotiating procedure at the same time, but each part is in charge of its own negotiations and continually negotiates with the workstations as the need arises in order to fulfil its production plan.

LeFrancois and Montreuil introduced an object oriented knowledge representation framework which they developed for modelling, analyzing and controlling the operations of a workstation. The framework consists of a new class of objects called agents which have three types of reasoning: meta-reasoning to control the inference/search reasoning activities, a local reasoning level incorporating simple rule base procedures, and an extended reasoning level representing the reasoning activities that cannot be performed using simple procedures and thus require external knowledge agents to assist in the decision making process. This framework allows LeFrancois and Montreuil to provide objects with intelligence, and they have used the framework to assist in developing and validating the schedule and control strategies for a rolling mill workstation.

A different type of framework was developed by Lin and Solberg [30]. Their adaptive control and scheduling framework was based on distributed information processing, distributed decision making, and a heterarchical market-like model. Each functional unit of parts and resources is equipped with an intelligent (software) agent and these agents communicate and negotiate in real time with each other to achieve mutual agreements for task sharing. Parts and machine agents go through a multi-step negotiating procedure before a part is committed to a machine for a certain operation. Using this framework, the part agents make decisions based on the objective functions of the part, while the resource agents make decisions based on the price evaluation system. The framework is implemented in an object oriented environment.

So far, we have seen a number of control structures which range from rigid hierarchical structures to very flexible structures using autonomous agents and negotiating procedures. The current research trend appears to be towards the development of heterarchical structures which have autonomy and self control, but also which have direction regarding global issues. In general, this is also the intent of the current project, but before going into a discussion of the proposed control structure, a review of scheduling techniques is in order since scheduling plays a great role in the ultimate control of a manufacturing system.

## 2.4 ARTIFICIAL INTELLIGENCE SCHEDULING TECHNIQUES

As the heading suggests, this section will focus on some of the various artificial intelligence approaches to scheduling. A significant amount of research has gone into the development of dispatching rules and algebraic techniques for scheduling, but these do not appear to have the flexibility or scope to deal with the large variety of manufacturing systems or to deal effectively with disruptions or changes. More and more researchers are investigating the application of artificial intelligence techniques to scheduling and it seems appropriate to review a few of those techniques here.

Expert systems have been used in many instances not only for control purposes but also to assist in the scheduling of manufacturing systems. MADEMA (MAnufacturing DEcision MAking) is a framework described by Chryssolouris et al. [4]

which supports multi-criteria decision making and scheduling in a shop floor environment. MADEMA uses a small rule base system that performs backward chaining on rule bases in order to select the appropriate optimization criteria from candidates such as: flow time, wait time, operation cost, tardiness, and quality. Thus MADEMA uses an expert system to move away from modern dispatching and scheduling techniques which optimize only on one or two criteria, and instead considers a number of criteria which improves the flexibility and responsiveness to change.

Wu and Wysk [43] presented an expert system, discrete event simulator combination called MPECS (Multi-Pass Expert Control System) which also uses multiple criteria to make scheduling decisions. The expert system looks at certain scheduling rules and principles, current cell conditions, and a number of criteria in order to select several good alternative scheduling rules (control policies). These rules are then fed into a discrete event simulator for assessment and the best scheduling rule is chosen based on a specified performance measure. As a result, the cell is scheduled and controlled in a way that is more representative of the ongoing cell conditions. Furthermore, a learning module has been added which allows MPECS to adapt itself to different systems and conditions by continually modifying its knowledge base. Cho and Wysk [3] developed IWC (Intelligent Workstation Controller) which operates along the same lines as MPECS. However, instead of using an expert system to select alternative scheduling rules, IWC uses a neural network. The neural network takes as input seven factors which represent the current workstation status, and as output, a choice of eight common dispatching rules.

The network was trained with ninety sets of input/output training vectors using a back propagation technique. Once trained, an input vector representing the current workstation status is applied to the network. The network selects the two best alternative dispatching rules, which are then fed into a discrete event simulator for evaluation and selection of the best strategy. The IWC appears to be robust, adaptive and was able to deal with noisy input data. Even more significantly, once trained, the neural net decision making was very fast and therefore may be appropriate for the real time control of manufacturing systems. Unfortunately, as the number of input nodes, output nodes and training sets increase, so will the training time.

Both MPECS and IWC find the one best dispatching rule based on simulation runs which are performed periodically whenever an event occurs. Alternatively, instead of relying on one rule, Grabot and Geneste [11] use a fuzzy logic method to combine a number of dispatching rules in order to obtain a compromise between the satisfaction of several criteria. Within the fuzzy logic method, a dispatching rule is expressed as two production rules covering a low and a high range of the antecedent and consequent (input and output) parts of the rules, where each range is represented by a membership function. A number of dispatching rules are developed in this format and combined using fuzzy logic techniques. As a result, criteria which are more important contribute to a greater degree to the outcome, whereas criteria having lesser importance are still considered instead of being completely ignored. Thus many factors and many rules can be taken into account at one time, thereby increasing the flexibility and robustness of the system.

Grabot and Geneste applied their fuzzy logic procedure to a conventional job shop scheduling problem. Custodio et al. [6], on the other hand, developed a more elaborate control and scheduling system which combined two levels of fuzzy logic control. One level applies fuzzy logic methodology to two aspects: 1) to routing rules in order to choose a resource for the next operation of a part, and 2) to dispatching rules to select the next part for processing from the parts waiting in the input buffer of a resource. Fuzzy logic allows for the combination of all the rules and several criteria in order to generate a decision about part movement. As a result, the movement of parts is more reflective of the ongoing status of the shop floor and of the current decision criteria. The second level applies fuzzy logic in a way that is similar to a more conventional fuzzy logic controller. A fuzzy scheduler was developed which tries to maintain a specific production rate based upon production levels and work in progress. The fuzzy scheduler uses fuzzy logic applied to the error between cumulative production and cumulative demand to set production rates for each part type.

The fuzzy techniques of Grabot and Geneste, and Custodio et al. allow the use of multiple criteria and multiple rules in order to control and schedule systems which are constantly changing. These methods more fully take into account what is happening in the manufacturing environment than the other methods discussed so far and introduce a way of dealing with change and uncertainty.

Before concluding the chapter, two artificial intelligence techniques which have

been used as search techniques to find optimum scheduling solutions will be presented: Petri nets and genetic algorithms. Lee and DiCesare [28] presented a method which uses a Petri net model to help find an optimum schedule. A Petri net model is constructed to represent the manufacturing system, and is used to track all the behaviours of the system with a reachability graph of the net. An optimal schedule is obtained by using a heuristic search technique to guide the generation of the reachability graph and to find an optimal path from the initial marking of the Petri net to the final marking. The schedule that is developed is an ordering of initiations and activities; therefore, the schedule is event driven, rather than time driven. The method they have developed appears to be quite powerful and can handle routing flexibility, shared resources, lot sizes and concurrency as well as being able to avoid potential deadlocks.

Genetic algorithms have also been used by several researchers as search techniques to search for good schedules. Uckun et al. [42] describe a number of ways of arranging the chromosomes to include machines, machine schedules, job orders, or process plans. They use a schedule builder to create legal schedules after factoring in constraints such as machine set-up and down times. The genetic algorithm is used to search through these chromosome arrangements to achieve an optimum schedule with respect to machine utilization and work in progress criteria. In contrast to the search proposed by Uckun et al., Herrmann et al. [14] utilizes existing dispatching rules to devise a schedule. The idea is to assign a separate dispatching rule to each machine and it is the task of the genetic algorithm to search the space of machine rule combinations to ultimately find the

optimum scheduling rule for each machine. This technique provides better local and global results than obtained through the use of only one fixed dispatching rule for all machines.

This then concludes the literature review and provides the basis from which ideas were drawn during the development of the proposed control structure and fuzzy logic dispatching method. Throughout the project, the underlying theme has been to incorporate autonomy and flexibility within a real time control structure. In developing the proposed structure, several new unique ideas have been incorporated into the control structure along with new ideas which are extensions of existing supervisory and scheduling concepts.

# CHAPTER 3

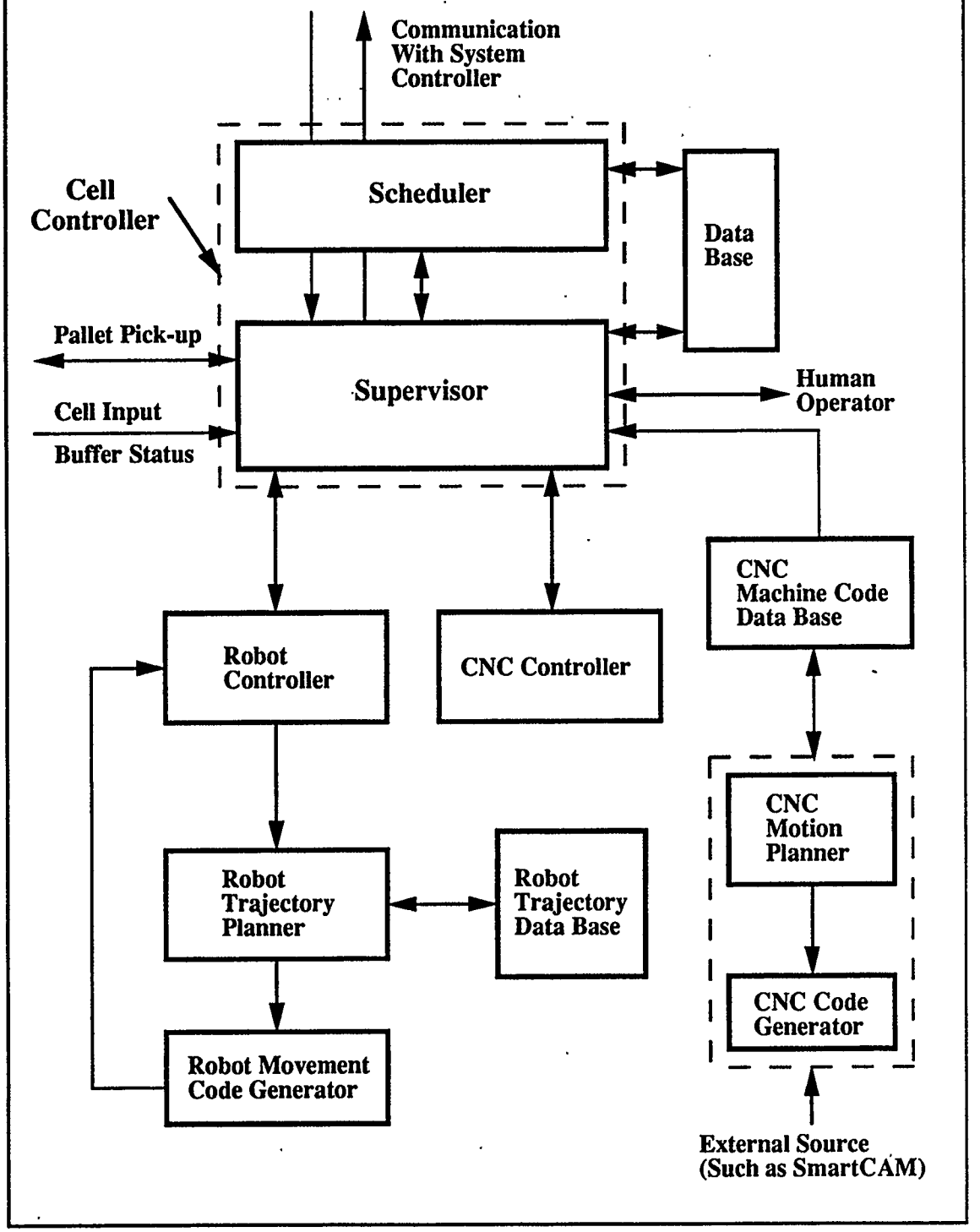
## 3. GENERAL OVERVIEW OF THE CONTROL STRUCTURE

The overall primary goal of this research was to develop a real time control structure which has flexibility in dealing with disruptions, exhibits expandability, and will ultimately support the concept of autonomous agents. Ideally the control structure will also be portable in the sense that the structure could be used for a variety of different system configurations or even over a range of different control levels with only minor modifications. The primary focus of this particular piece of work was to develop a control structure specifically for application and testing at the manufacturing cell control level.

It is often difficult to discern the difference between scheduling and control on a shop floor and this difficulty is very apparent at the cell level. At progressively finer levels of control, such as at the equipment control level, scheduling is less of a concern

since the issues are more often that of providing sequential control, and on controlling how actions can be accomplished rather than when. In direct contrast to this, at the cell level the control structure must take timing into account, not only within the cell but also when interacting with other cells or even the entire shop floor system. For example, a schedule for the shop floor dictating due dates and order sizes will directly translate into certain scheduling requirements for a given cell, and therefore scheduling part flow within a cell to meet these requirements is an important issue. Unfortunately, there is a large overlap between true scheduling, part dispatching and the control of the various elements within the cell such as part processing machines, material handling systems, or the parts themselves. Bearing this overlap in mind, a control structure was developed which was loosely broken into two sections: a scheduling section, and a supervisory section as shown in Figure 3.1. This is a somewhat more compact control structure than those proposed by Curtis and Tiemersma (PCS), Gendreau et al. (LURPA), or Bauer et al. (PAC); however, as will be discussed more thoroughly in chapter 5, the activities performed by the dispatchers, drivers, monitors etc. in the control structures mentioned above are incorporated into the proposed structure. Even though these two sections exist in the proposed structure, the problem of cell control involves both scheduling and supervision operating in concert and therefore throughout this work there may not be a crystal clear distinction made between these two processes. In fact, it is the blending of scheduling and supervision that allows for a unified approach to cell control. With these thoughts in mind, the general overall control structure will now be presented; further details of the structure will be given in sections 4, 5 and 6 on scheduling, supervision, and

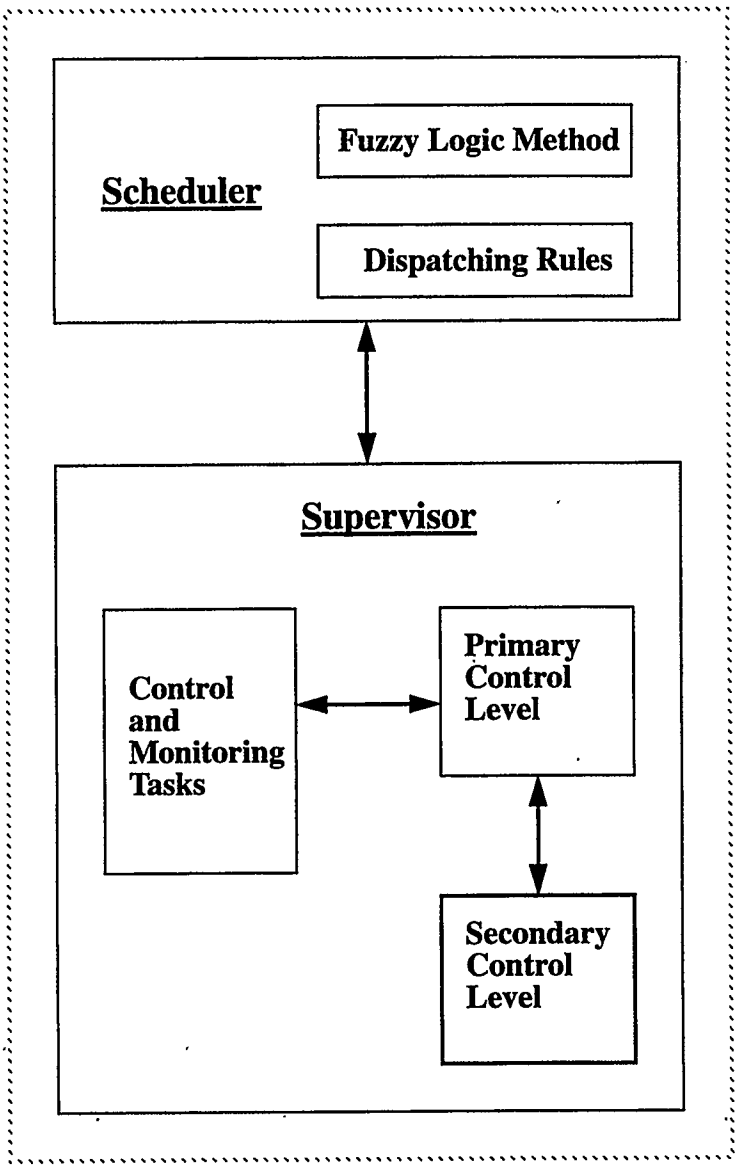
Figure 3.1: Cell Controller & Interactions



implementation respectively.

The main responsibility of the scheduler is to generate priority levels for the parts waiting in the input buffer of a given machine. The level of priority of a part dictates when that part will be selected for processing. The scheduler is made up of two sub-modules (Figure 3.2), one of which contains a series of common dispatching rules, and the other which contains the fuzzy logic part dispatching method. Conceptually, both sub-modules accomplish the same thing, that is identifying which part within an input buffer should be processed next; however, they are fundamentally different. Dispatching rules typically have been developed to provide optimum performance for certain parameters given a specific cell or system configuration (French [9], Kusiak [27], Blackstone et al. [2]). Thus, one dispatching rule may work very well given one type of shop floor configuration, but may work poorly under a different configuration. In fact, a change in the part mix may positively or negatively affect the ability of the dispatching rule to effectively schedule parts even if the system configuration has not changed. The fuzzy logic method attempts to provide a more general, less sensitive approach. The method looks at a wider range of parameters and incorporates a number of different rules as it develops priority levels. More importantly, the parameters under consideration, and the guiding rule structure can be easily changed within the method. Thus if there are certain aspects of a given shop floor which are unique, that uniqueness can easily be incorporated into the fuzzy logic method. So, instead of having one dispatching rule which may not apply for all cases, the fuzzy logic method allows customization of the

Figure 3.2: Supervisor and Scheduler Sub-modules



rule base which gives it the flexibility to schedule under a greater variety of circumstances. Dispatching rules have been incorporated into the proposed scheduler mainly for comparison purposes in order to evaluate the fuzzy logic method.

The supervisor provides the main controlling element of the cell controller and as a result it has many tasks to perform. These include: part movement and part production, control of material handling, controlling the various machines, monitoring machine and tool status, error recovery, implementing virtual cell reconfiguration when necessary, monitoring cell input and output buffers, and monitoring part inventory levels. The general mode of operation of the supervisor is shown in Figure 3.2 which shows the interconnection between the supervisor's control and monitoring tasks, the primary control level, and the secondary control level. As events occur within the shop floor, the supervisor recognizes these events through the monitoring tasks and then initiates the appropriate control actions which are fulfilled through the use of the primary and secondary control level structure. The primary control level is the key link between the cell controller and the elements within the cell, such as the various machines. When a control task is required, the supervisor activates the primary control level, which in turn activates the secondary control level. The secondary control level is responsible for the more detailed control actions, for example: the planning and implementation of the actual robot movement.

The supervisor and scheduler have been developed to work on a cooperative basis

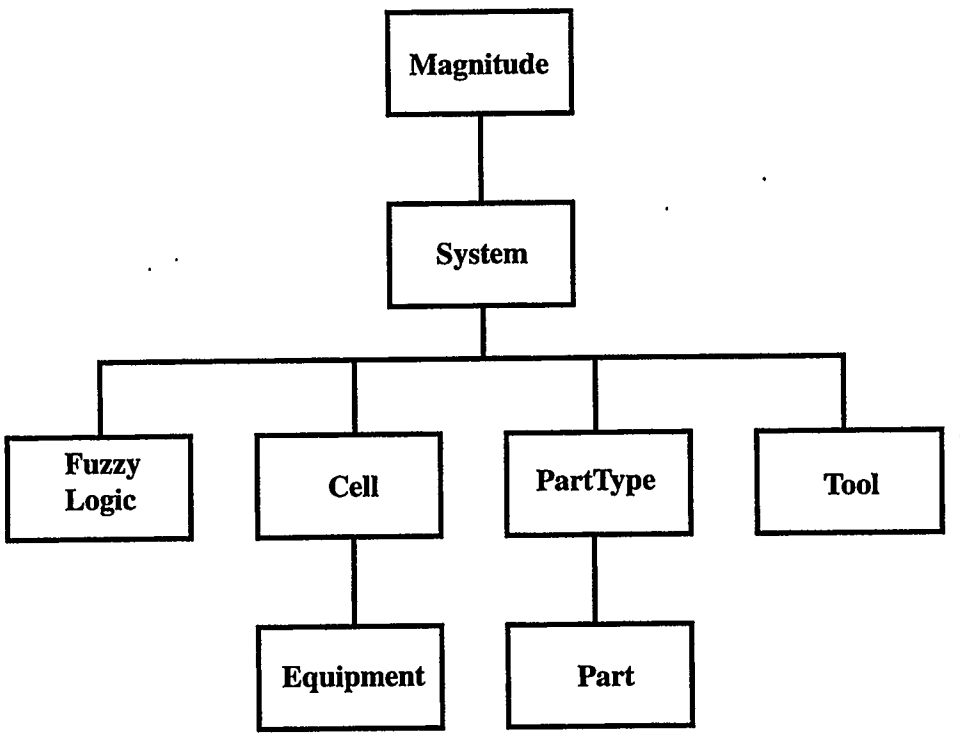
within a modular cell control structure. The cell controller works autonomously to produce parts and to deal with issues within the cell. It is guided by goals set by alternate control systems rather than being controlled directly by those systems. The scheduling and supervisory methods used are generic and do not hinge upon the type of machines in the cell or on what type of cell it is. The structure will work equally well for a machining cell, assembly cell, inspection cell or combination. As will also be discussed later, the structure has been developed to ultimately allow parts and machines to interact and operate more autonomously.

The control structure has been implemented using the object oriented programming paradigm. The reasoning behind this choice is several fold. First, this paradigm most closely resembles the real world in the sense that the real world consists of individual unique objects which have their own attributes, may retain knowledge, and also may have varying degrees of intelligence or reasoning abilities. Second, in order to have the proposed control structure achieve maximum flexibility, the objects within the system should have a certain degree of intelligence and an ability to reason. The object oriented paradigm achieves both these goals because it allows the representation of real world objects and has facilities to assign attributes, knowledge and reasoning abilities to those objects. Although the storage of knowledge and reasoning does not naturally exist in inanimate objects such as machines or parts, these aspects can be artificially implemented using current computer technology. The object oriented environment is also structured to perform communication using message passing between objects. The objects could

then communicate with each other and therefore, in a small way, direct their own destinies according to certain rules.

The overall object oriented class structure of the control system is shown in Figure 3.3 and shows the inheritance of the various classes. Seven new classes; System, Fuzzy Logic, Cell, Equipment, PartType, Part and Tool have been developed and placed under the Smalltalk 80 Magnitude class. Separate objects are created as unique instances of each class and it is these objects which communicate with each other in the system in order to perform the various scheduling and supervising tasks. As more objects such as parts or machines are required, the object oriented programming structure simply allows the creation of new instances of each particular class. These instances then inherit all the appropriate attributes, knowledge and reasoning from the class structure. A more detailed discussion of each of the classes is given in chapter 6 on implementation. In the following two chapters more detail will be provided about the scheduler and supervisor functions.

Figure 3.3: Class Hierarchy



# CHAPTER 4

## 4. SCHEDULER

## 4.1 INTRODUCTION

The main thrust of the current work, and indeed the main area of original contribution, is contained within the scheduler module of the control structure. Here, fuzzy logic has been used in a unique way in order to provide a scheduling methodology which can consider many aspects of the shop floor while developing a schedule in real time. In the context of the current control system, the primary responsibility of the scheduler is to generate priority levels for parts waiting in the input buffer of a given machine. This is done using either the fuzzy logic method scheduling sub-module or the dispatching rule sub-module (see Figure 3.2). Since the dispatching rules have been included in the scheduler module mainly for comparison purposes, the bulk of the discussion will centre around the fuzzy logic method and its application, with only a short

review of the various dispatching rules.

One of the main objectives of this thesis was to develop a control structure which can deal with disruptions and breakdowns within the system. Significant portions of the supervisor structure address this objective directly but one of the keys to implementing a complete control scheme is to ensure a flexible scheduling technique. Various approaches have been applied to job shop scheduling including mathematical programming (French [9], Kusiak [27]), dispatching rules (Panwalkar and Iskander [37], Blackstone et al. [2]), neural networks (Cho and Wysk [3]), genetic algorithms (Herrmann et al. [14]), and expert systems (Chryssolouris et al. [4], Maimon [31], Wu and Wysk [43]); however, many of these techniques ultimately only develop a fixed schedule and are therefore inflexible. Although a number of different methods are used to optimize the final fixed schedule, with the most elaborate case being the use of a genetic algorithm, few of the fixed schedules account for unexpected changes or disruptions on the shop floor. Wu and Wysk, and Cho and Wysk, on the other hand, perform ongoing optimization techniques which develop very short term fixed schedules that are used only for a short finite time and are then updated as required if the system status changes. In this way, their systems account for disruptions in the shop floor. The proposed control structure has also been designed to deal with shop floor disruptions, but does so on a real time basis and incorporates fuzzy logic part dispatching to accomplish that task. The fuzzy logic dispatching method currently incorporates rule bases related to machines, such as machine buffer levels, and to parts, such as part inventory levels, due dates and process

plans. The intent of the method is to incorporate a wider range of scheduling issues than those considered by most current dispatching rule systems. Basically what the fuzzy logic dispatching module does is to assign priority values to each part in a manner which reflects current shop floor conditions, and as the shop floor conditions change, so do the relative priorities of the parts. The details of how this is accomplished using fuzzy logic will be discussed next.

## 4.2 FUZZY LOGIC METHOD SUB-MODULE

### 4.2.1 Background

When conditions in the shop floor change and the current schedule in use no longer reflects an accurate plan, then it is often up to the individuals on the shop floor to make corrections to the schedule to allow part manufacture to continue. A shop floor can be a very complicated place; however, experienced individuals frequently have clear ideas about the best course of action under the given conditions. These individuals may even be able to formalize the reasoning behind their actions into a series of linguistic rules which could ultimately be used to help control and schedule the system. Using the linguistic rules developed by the individuals and applying them in an automatic and formal way is one of the precepts of fuzzy logic reasoning. Alternatively, in some cases it may be possible to model and control the shop floor using mathematical programming. Unfortunately, it is often very difficult to represent a shop floor mathematically and in

those cases where the modelling can be successfully accomplished, the system often cannot be analyzed in a realistic time frame. Fuzzy logic methodology operating on linguistic rules may represent an avenue for real time control which allows the use of individuals' knowledge and avoids some of the difficulties involved with mathematical programming techniques. It was on this premise that the application of fuzzy logic to shop floor control and scheduling was undertaken. Before discussing the actual method of application, a review of fuzzy logic theory is in order.

### 4.2.2 Theory

The concept of fuzzy logic has been around for many years but it was Professor Lotif A. Zadeh who, in 1965, was the first to introduce mathematical tools which now allow imprecise and qualitative information to be expressed in more exact ways (Kaufman & Gupta [23], Zadeh [44]). Essentially, the idea of a fuzzy set was developed and introduced as a generalization of the ordinary notion of a set (Tong [40]). This basic idea has been used in many ways; however, it will be discussed here in a form which seems most applicable to the scheduling problem.

An ordinary set, or a "crisp set", is precise in its meaning, having a definite transition from membership within the set to non-membership. The set has sharp boundaries; either an element belongs to a particular set or it does not. For example, consider the measurement of room temperature in the closed interval 15-40 $^{\circ}$C where the

concern is to describe the linguistic term 'warm'. An ordinary set which defines this can be expressed in terms of a membership function $\mu$ which can take values of either 0 or 1. If $\mu(T) = 0$ then the temperature T is not a member of the set 'warm'; if $\mu(T) = 1$ then the temperature is a member. This is shown graphically in Figure 4.1a where the membership function $\mu$ is represented by a rectangular function. Thus all the temperature values between 22.5 and 32.5 $^{\circ}$C, inclusive, represent 'warm'. A fuzzy set on the other hand allows the transition from non-membership to full membership to occur gradually rather than abruptly. This is also known as the concept of graded membership. To continue with the temperature example, a fuzzy set representing 'warm' might be as shown in Figure 4.1b. The fuzzy set uses all the values between 0 and 1 where 1 represents full enlistment into the term 'warm', 0 represents no enlistment, and values between 0 and 1 represent partial enlistment or partial membership. Therefore the qualitativeness of the measure (in this case temperature) is reflected as a gradual transition to full membership.

The concept of fuzzy sets becomes important in situations where the precise mathematical definition of a process cannot be easily formulated or easily solved. This occurs frequently in very complex industrial processes such as batch reactors, blast furnaces, cements kilns, and steel making (King & Mamdani [25]), and in many other non-linear, time varying systems. While these processes are difficult to control automatically, they are often controlled quite satisfactorily by human operators. The operators' control strategy is often based on intuition and experience, and the challenge

Figure 4.1a: Crisp Set of Temperature

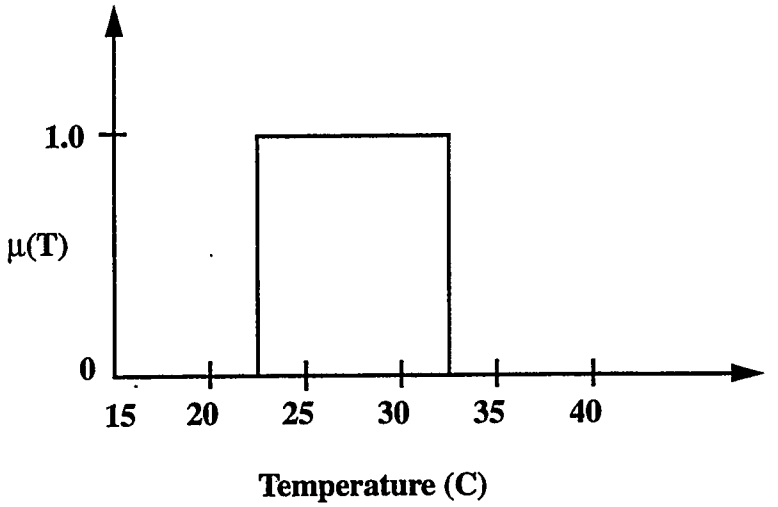
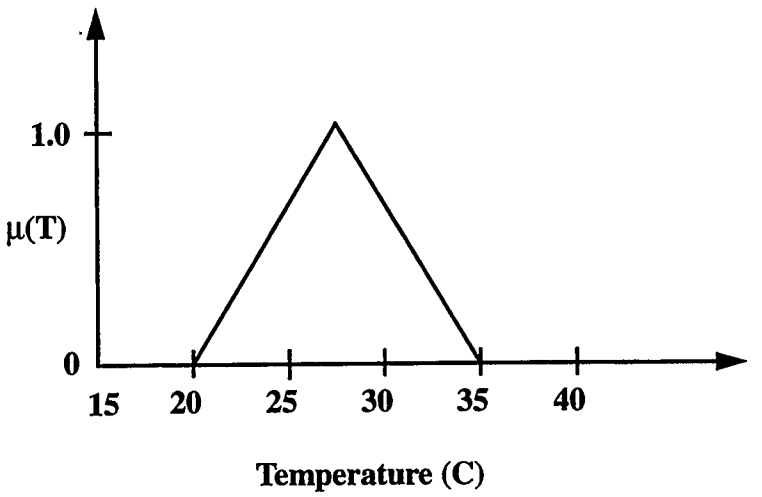



Figure 4.1b: Fuzzy Set of Temperature



is to represent the control strategy in a form that can be automated. While difficult to do, it is often possible to represent the control strategy as a series of linguistic rules or statements. Two aspects which may hinder the development of linguistic rules are: 1) the operator actions are often inconsistent, erratic or prone to error and 2) the operator often responds to a complex pattern of measurements and observations of unmeasurable variables such as colour, consistency, etc. [25]. If these problems can be overcome and linguistic rules can be developed, then the theory of fuzzy sets and algorithms developed by Zadeh can be used to develop an automatic control structure based on the linguistic rules. The scheduling of a shop floor can be considered to be a problem which falls into the realm of fuzzy logic application due to its complexity and due to the difficulty of rescheduling in short time frames in order to adjust for changes on the shop floor.

Fuzzy sets may be combined in a manner similar to conventional sets by means of several simple operations (Zadeh [44]). Three operations will be considered here: intersection, union, and compliment (or negation). First of all a fuzzy set is defined as follows. "A fuzzy subset A of a universe of discourse (support set) X is characterized by a membership function $\mu_A(x)$. This function assigns to each element $x \in X$ a number $\mu_A(x)$ in the interval 0 to 1 which represents the grade of membership of x in A" (Kickert & Lemke [24]). Three basic operators which are commonly used can then be defined as follows [44]:

a) The *union* of two fuzzy sets A and B of the universe of discourse X is denoted

by $A \cup B$ with a membership function defined by:

$$\mu_{A \cup B}(x) = \max [\mu_A(x); \mu_B(x)]. \quad x \in X \tag{4.1}$$

The union corresponds to the connective '*or*' as in 'If A *or* B'.

b) The intersection of two fuzzy sets A and B is denoted by $A \cap B$ with a membership function defined by:

$$\mu_{A \cap B}(x) = \min [\mu_A(x); \mu_B(x)]. \quad x \in X \tag{4.2}$$

The intersection corresponds to the connective '*and*' as in 'If A *and* B'.

c) The complement of a fuzzy set A is denoted by $\neg A$ with a membership function defined by:

$$\mu_{\neg A}(x) = 1 - \mu_A(x). \quad x \in X \tag{4.3}$$

Complementation corresponds to the negation '*not*' as in 'If *not* A'.

For the purpose of this thesis, the notation given in Kosko [26] is used to represent the various rules. Thus the rule 'If A *and* B *then* C', will be abbreviated by (A,B;C)

where A and B are called the antecedents, C is called the consequent and the comma represents the '*and*' connective.

The definition of a fuzzy set allows the representation of linguistic terms such as 'high', 'medium', 'low' or 'not low' etc. as fuzzy subsets of a given universe of discourse such as 'temperature'. The above operators allow the manipulation of the various antecedent fuzzy subsets within the given rule in order to develop specific results. The next step now is to provide a mechanism whereby the results of the operations on the antecedents within a rule can be mapped to the consequent fuzzy set. How this is done is described as follows: "Given an implication statement 'If A *then* C', the implied relation R can be expressed in terms of the cartesian product of the subsets A and C (of universes of discourse X and Y) and is denoted by $R = A \times C$" [25]. The implied relation matrix is defined by:

$$\mu_R(x,y) = \mu_{A\times C}(x,y) = \min [\mu_A(x); \mu_C(y)]. \quad x \in X, y \in Y \tag{4.4}$$

This procedure is termed correlation minimum encoding and it is one of two commonly used encoding schemes which are used to develop the implied relation matrix. The second encoding scheme is called correlation product encoding. In correlation product encoding the implied relation matrix is defined by [26]:

$$\mu_R(x,y) = \mu_A(x) \cdot \mu_C(y) \quad x \in X, y \in Y \tag{4.5}$$

The implied relation matrix represents the mapping between the antecedent fuzzy set(s) and the consequent fuzzy set. In other words, it defines the general relationship between the inputs and the outputs of a given rule. The next question is: what will be the output set given an input set differing somewhat from the one used to develop the relation matrix? In other words, given the rule: 'If A *then* C' and the input set A′, what will be the output set C′? The implied relation R is used to infer the output set C′ by the use of the compositional rule of inference [44]:

$$C' = A' \circ R = A' \circ (A \times C) \tag{4.6}$$

where '$\circ$' denotes the composition operator. The compositional rule of inference is also known as the max-min composition relation whose membership function can be further defined by:

$$\mu_{C'}(y) = \max_{x} \min[\mu_{A'}(x); \mu_{R}(x,y)] \tag{4.7}$$

and which gives the value of C′ as a fuzzy output set. Thus, the max-min composition relation is used in conjunction with either correlation minimum encoding or correlation product encoding in order to generate a fuzzy output set for a certain rule, given a unique input set. It should be noted that the two encoding schemes, correlation minimum

encoding and correlation product encoding, result in markedly different fuzzy output sets as can be seen in Figure 4.2.

The correlation minimum encoding scheme produces clipped output sets as shown in Figure 4.2a. The scaled output sets (Figure 4.2b), developed using correlation product encoding, retain the same shape as the consequent sets whereas the clipped output sets are flat at a given level. In a sense the correlation product encoding preserves more information about the consequent set; information which may be of use when several output sets from several rules are combined. For this reason, correlation product encoding is often the chosen encoding scheme for current fuzzy logic applications [26].

Up to this point it has been shown how, when given a rule in the form 'If A *and/or/not* B *then* C', fuzzy logic can be applied to provide a fuzzy output set. The final important step is the amalgamation of all the output sets derived from the various control rules and the determination of a unique output. Once this is presented, an example will be given to help illustrate all the steps in the process.

Generally a control structure is made up of a series of rules, not just one single rule and therefore a mechanism is required which incorporates all rule outputs. Consider a set of $m$ rules with consequents $C_i$ (i = 1 to $m$) for which a set of outputs $C_i'$ (i = 1 to $m$) have been developed where each output $C_i'$ is a fuzzy set having $k$ elements. All these outputs need to be combined and then 'defuzzified' or converted into an appropriate

Figure 4.2a: Correlation Minimum Encoding

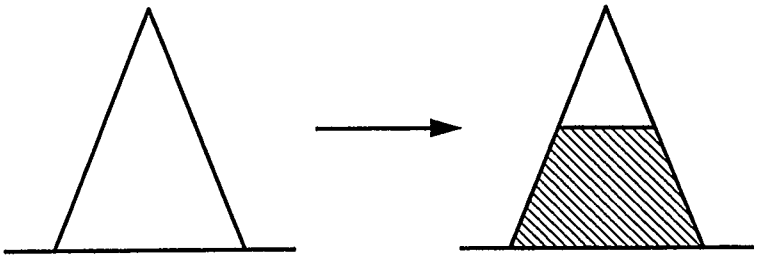
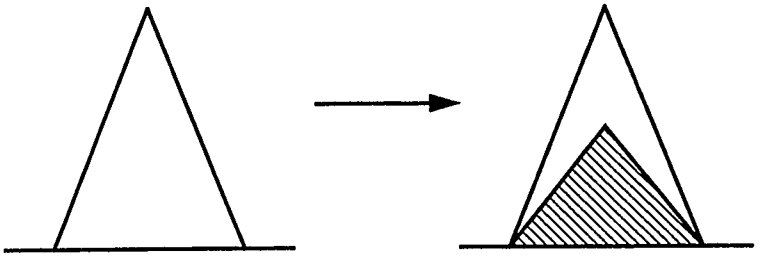



Figure 4.2b: Correlation Product Encoding



singular valued output. In order to combine all the output sets $C_i'$ the total output set C is given by:

$$C = \sum_{i=1}^{m} w_i C_i' \tag{4.8}$$

where $w_i$ is a non-negative weight representing the reliability or strength of the ith rule.

Two defuzzification schemes will be discussed: the maximum-membership defuzzification scheme and the fuzzy centroid defuzzification scheme. The simplest scheme is the maximum-membership scheme which simply chooses that element $y_{max}$ that has maximum membership in the output fuzzy set C:

$$C(y_{max}) = \max_{1 \le j \le k} C(y_j) \tag{4.9}$$

The difficulty with the maximum-membership defuzzification scheme is two-fold. Firstly, the final output set C may not have a unique maximum. There may exist several positions of y containing the maximum or the maximum may be a broad plateau making

a unique choice of $y_{max}$ impossible. This problem affects correlation-minimum encoding more than correlation-product encoding since the minimum encoding scheme clips off the final output set C resulting in larger flat plateaus. Secondly, the maximum-membership scheme ignores much of the information in the final output set C. Thus if C is highly asymmetrical, the maximal value may not be a true representation of all the rules. For example there may be one rule which produces a maximal value near the low range end of the output set C which is only slightly larger than all other values across the entire range of output set C. Using the maximum-membership scheme a defuzzified output taken from the low range end of the output set C would be chosen which ignores the affect of the remaining slightly weaker values. The defuzzified output would therefore be skewed towards the low range end as a result of the one slightly stronger rule while at the same time ignoring the effect of the other slightly weaker rules. As a result, the defuzzified output does not accurately reflect the affect of all the rules but only accounts for the strongest rule.

To help overcome these problems the fuzzy centroid defuzzification scheme is used. In this scheme the centroid of the entire C final output set is determined and this value is used as the singular output. The centroid calculation is determined as follows:

$$\overline{C} = \frac{\sum_{j=1}^{k} y_j \cdot C(y_j)}{\sum_{j=1}^{k} C(y_j)} \tag{4.10}$$

All the elements are now in place for the application of fuzzy logic, but before proceeding with an example, a summary is in order. Given a rule in the form: 'If A *and* B *then* C', the antecedents A and B can be evaluated using the 'minimum' operator to generate a combined antecedent set. Using correlation minimum encoding or correlation product encoding, an implied relation matrix can be developed which then defines the mapping between input and output fuzzy sets for the given rule. The relation matrix is used within the compositional rule of inference in order to provide output values for specified input values of the fuzzy sets. The result of applying the compositional rule of inference is an output fuzzy set. It should be noted that the entire encoding and compositional scheme can be represented graphically (as will be shown in the following example). Finally all the output fuzzy sets for all the rules are summed and an appropriate defuzzification scheme is employed to get a single unique output.

### 4.2.3 Application and Example

Fuzzy logic has been applied in this thesis to the problem of part dispatching. More specifically, the fuzzy logic methodology has been used to calculate a priority level for each part in the input buffer of a given machine. Thus when the machine wishes to process a new part, it can review the priority of each part in the input buffer and select the part with the highest priority as the part to be processed next.

Of primary importance for the success of the dispatching algorithm was the

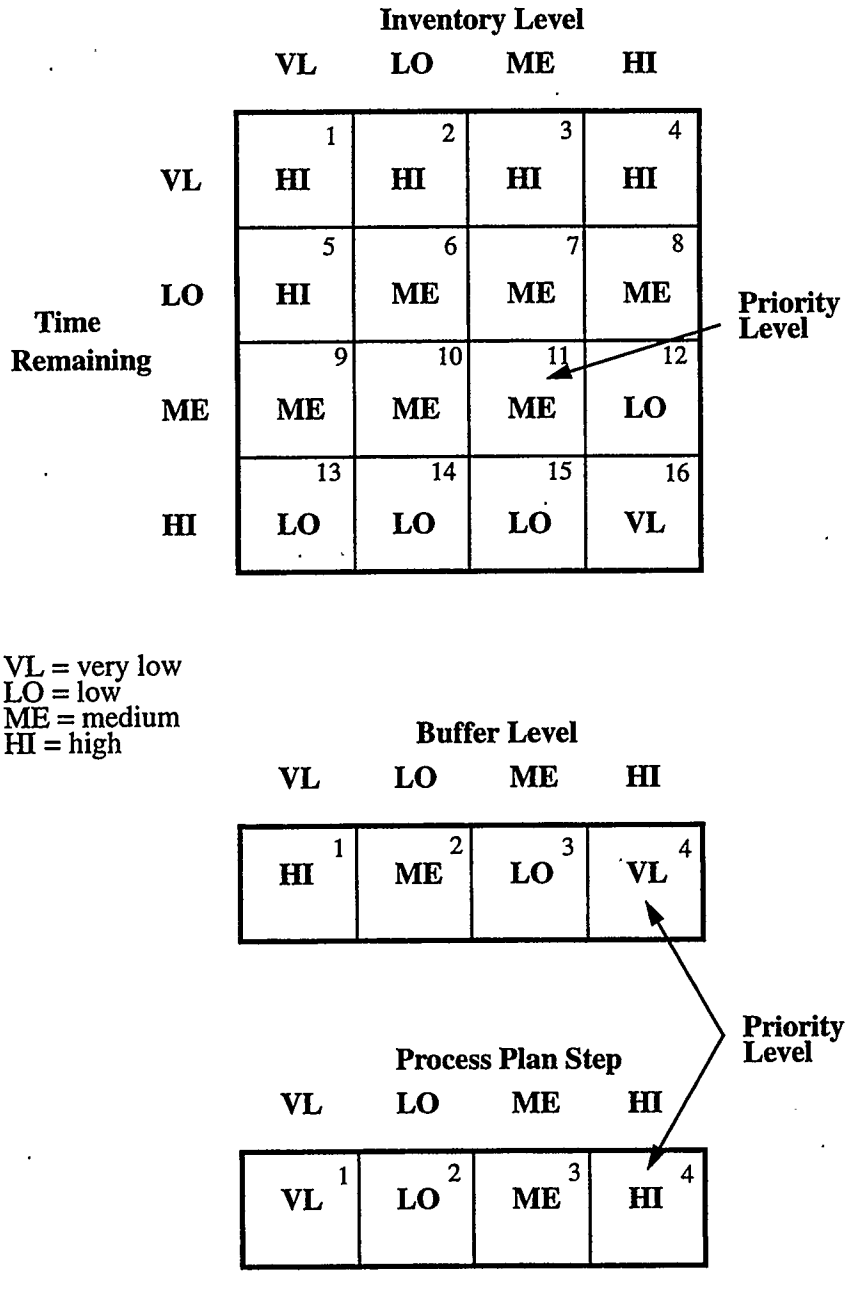
development of the rule structure. The rule base currently in use can be loosely divided into three rule sets: scheduling, machine considerations, and part considerations. These three rule sets were selected mainly because there are certain aspects which are often recognized as scheduling functions such as due dates and inventory levels, whereas other aspects are most often associated with machines such as buffer control and finally some aspects such as the process plan of parts can be associated mainly with parts. Therefore at this stage three separate sets of rules have been implemented to cover all three of these areas. It should be noted that the current rule set represents a starting place for the scheduling algorithm and is expected to grow and be refined as work continues in this area.

The first set of rules deals with the relationship between inventory level and due date; the goal being to minimize part lateness while meeting inventory levels. Sixteen rules have been developed and are represented in matrix form as shown in Figure 4.3. Inventory level is given across the top of the matrix, time remaining is given along the side, and the priority level is given inside each element of the matrix. A rule is read as follows:

Rule 5: If inventory level is very low *and* time remaining is low *then* priority is high.

or in condensed form: (VL,LO;HI)

Figure 4.3: Fuzzy Logic Rules



These sixteen rules develop a scheduling priority level which will be used in conjunction with the other rules to determine a total part priority level. There are no specific guidelines in the literature for defining the control rules of a fuzzy logic system, and therefore the sixteen scheduling rules shown here represent the application of intuition coupled with a certain degree of refinement through testing. They likely do not represent an optimum selection of rules; indeed an optimum set of rules will undoubtably be different for different shop floor configurations and part process plans. Optimization of the system will be discussed in greater detail in later chapters.

Four rules were developed to help control the buffer level of the machines. This rule set is a "look ahead" rule set in that the buffer level of the machine after the current machine in the process plan is considered. Thus if the next machine's buffer is close to full, the priority of the part will be reduced and a part not going to that machine is more likely to be selected.

Four rules were also developed to account for the process plan of the part. Here the degree to which the part has completed its process plan is considered, and is used to keep the part progressing through the shop floor as quickly as possible. This rule seems to help reduce the early onset of machine blocking and deadlock for this particular control structure. The three rule sets (scheduling, buffer and process plan step) were each weighted and combined to provide a final priority level for each part.

A second important aspect when developing a fuzzy logic control scheme is the development of the fuzzy membership functions. Again there are no specific guidelines for the development of membership functions; however, the use of triangular or trapezoidal membership functions seem to be the most common. Triangular functions were used in this application because they would provide a tighter and more unique output function. The triangular functions can be defined by a triplet $\mu_A = (a,b,c)$ [23] where:

$$\mu_a \begin{cases} = 0 & \text{if } x < a, \\ = \dfrac{x - a}{b - a} & \text{if } a \le x \le b, \\ = \dfrac{c - x}{c - b} & \text{if } b \le x \le c, \\ = 0 & \text{if } x > c \end{cases} \qquad (4.11)$$

and where x represents the current antecedent levels of time, inventory level, buffer level, process plan step, or the consequent level of priority, and where a represents the location of the left base of the triangle, b the location of the vertex and c the location of the right base.

The overall range of the membership function is chosen to provide a steep transition between each fuzzy set while still allowing room for each antecedent or consequent fuzzy subset. It has also been suggested, as a rule of thumb, that the fuzzy

sets should overlap by approximately 25% [26]. The designations VL, LO, ME and HI are somewhat arbitrary, but are commonly used as are designations such as NB (negative big), NS (negative small), PB (positive big) etc. The antecedent and consequent membership functions which are used in the current application are shown in Figure 4.4.

Once the rule sets and the fuzzy membership functions have been defined, a solution methodology may be applied. The method used in this work consists of six separate steps:

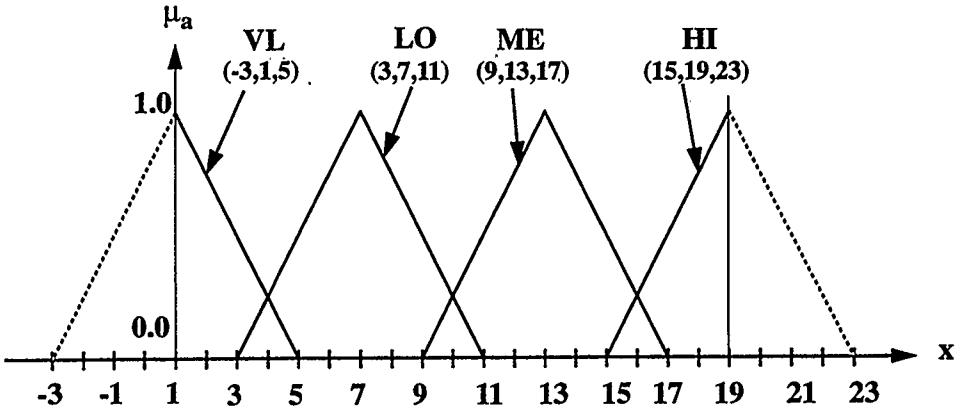
1. determine the value of the variable in question
2. fuzzify the variable
3. for each rule determine each antecedent fuzzy value
4. for each rule determine the fuzzy output set
5. calculate the net effect of all the fuzzy output sets
6. defuzzify the total output set.

By way of example, the methodology will be more clearly described. The example will be based on the following system configuration:

inventory level: - current level at 15 parts

- goal level is 100 parts

Figure 4.4: Fuzzy Logic Membership Functions




**Antecedent Membership Functions**

Triangular fuzzy membership functions are each defined by a triplet $\mu_a = (a,b,c)$ where:

$\mu_a = 0$ if $x < a$,

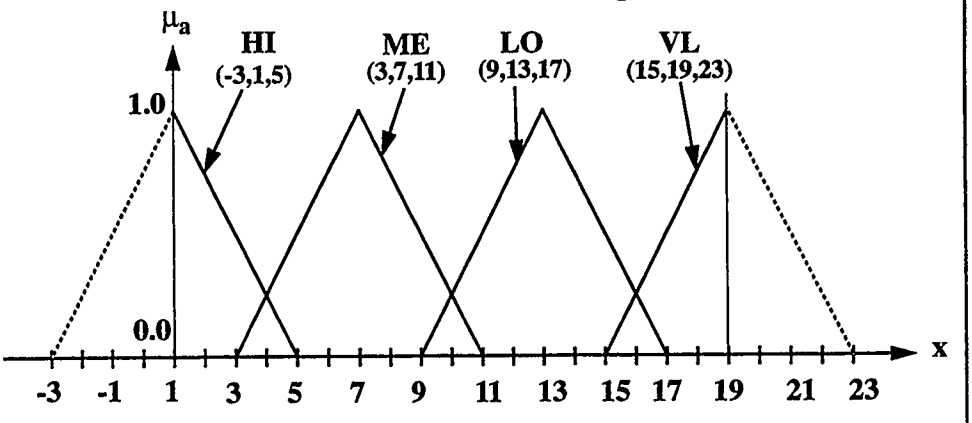
$= \frac{x - a}{b - a}$ if $a \leq x \leq b$

$= \frac{c - x}{c - b}$ if $b \leq x \leq c$

$= 0$ if $x > c$

and x represents the current antecedent level of time, inventory level, buffer level, or process plan step, or the consequent level of priority

**VL = very low**
**LO = low**
**ME = medium**
**HI = high**




**Consequent Membership Functions**

time remaining: - current level at 27 minutes

- due date is 60 minutes

buffer level: - current level at 4 parts

- finite buffer limit is 10 parts

process plan step: - current step is step 1

- process plan size is 4 steps

**Step 1:** determine the value of the variables

inventory level = 15 parts

time remaining = 27 minutes

buffer level = 4 parts

process plan step = step 1

**Step 2:** fuzzify the variables

The variables are fuzzified using the following general formula:

$$\textit{fuzzy level} = \left( \left( \frac{\textit{variable level}}{\textit{goal level}} \right) \cdot (\textit{membership function range}) + 1 \right) \textit{rounded} \qquad (4.12)$$

Based on a fuzzy range of 1 to 19, the fuzzy levels are calculated as follows:

fuzzy level of inventory = (((15/100) · (19-1)) + 1) rounded = 4

fuzzy level of time remaining = (((27/60) · (19-1) + 1) rounded = 9

fuzzy level of buffer level = (((4/10) · (19-1) + 1) rounded = 8

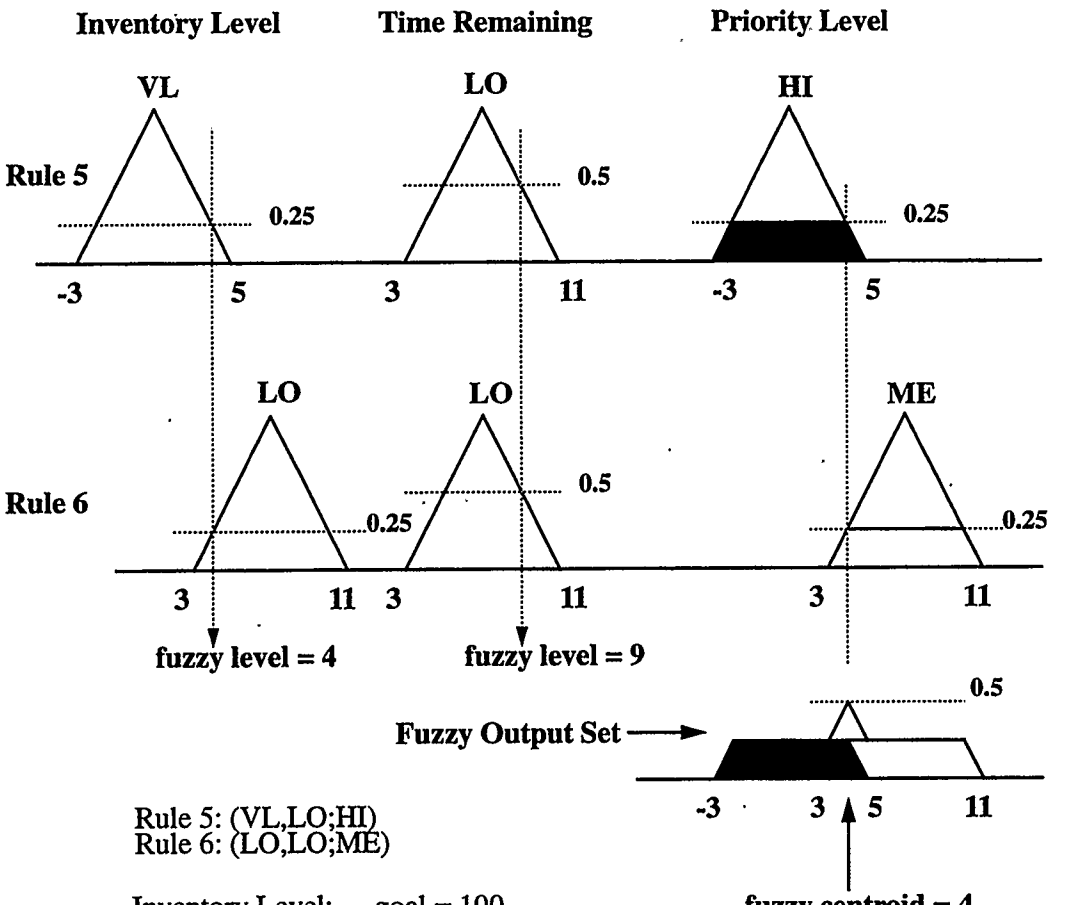
fuzzy level of process plan step = (((1/4) · (19-1) + 1) rounded = 5

**Step 3:** for each rule determine the antecedent fuzzy values

Consider first the scheduling (inventory level/time remaining) rule set. Conceptually, all rules are accessed while running the fuzzy logic algorithm but realistically in this example only two rules are fired. Given a fuzzy inventory level of 4 and referring to Figure 4.4, one can see that a fuzzy inventory level of 4 activates two fuzzy antecedent sets, VL (very low) and LO (low). Furthermore, the fuzzy time remaining level of 9 activates the fuzzy set LO (low). In reality all the other sets are activated to degree zero. Thus, referring to the rule matrix in Figure 4.3 one can see that two rules, rule 5 and rule 6 are fired. They are (VL,LO;HI) and (LO,LO;ME).

The application of steps 3 to 6 are shown graphically in Figure 4.5. Consider first rule 5, which has the two antecedents, VL and LO, and the consequent, HI. The fuzzy inventory level of 4 activates the VL antecedent to degree 0.25 while the time remaining fuzzy level of 9 activates the LO antecedent to degree 0.5. Figure 4.5 shows graphically

Figure 4.5: Application of Fuzzy Logic Methodology




Rule 5: (VL,LO;HI)
Rule 6: (LO,LO;ME)

Inventory Level: goal = 100
current level = 15
fuzzy level = 4

Time Remaining: deadline = 60 minutes
current time remaining = 27 minutes
fuzzy level = 9

Priority Level: fuzzy centroid = 4
priority = 4

how the encoding and compositional inferencing scheme works. In the current scheduling configuration, the max-min composition scheme is used in conjunction with correlation minimum encoding resulting in a clipped output fuzzy set represented graphically as a trapezoid. Correlation minimum encoding was chosen for this application because for the current selection of rules and membership functions, correlation minimum encoding had a tendancy to generate unimodal total output sets for the inventory level/time remaining rule set. The correlation product encoding scheme on the other hand often created multi-modal total output sets which were more difficult to analyse.

In a similar fashion it can be determined that rule 2 of the buffer level rule set is fired and that the fuzzy buffer level of 8 activates the LO antecedent to degree 0.75. Rule 2 of the process plan rule set is also fired and the fuzzy process plan step level of 5 activates the LO antecendent to degree 0.5.

**Step 4:** for each rule determine the fuzzy output set

Since the two antecedents VL and LO of rule 5 of the scheduling rule set are combined using the connective '*and*', the 'minimum' operator is used on the antecedents thereby yielding a result of 0.25. This result is applied to the rule using correlation minimum encoding coupled with the max-min composition scheme to generate a HI priority level output fuzzy set ranging from -3 to 5 having a maximum value of 0.25 (Figure 4.5). Similarly for scheduling rule 6, the inventory level LO antecedent is

activated to degree 0.25 while the time remaining LO antecedent is activated to degree 0.5. The minimum operator is again used giving a result of 0.25 which is applied through the max-min composition scheme, using correlation minimum encoding, to give a ME priority level output scheduling fuzzy set ranging from 3 to 11 with a maximum value of 0.25.

Similarily a ME priority level output buffer level fuzzy set ranging from 3 to 11 with a maximum value of 0.75 and a LO priority level output process plan step fuzzy set ranging from 9 to 17 with a maximum value of 0.5 were developed.

**Step 5:** calculate the net effect of each of the output fuzzy sets

As shown in Figure 4.5 the summation of the output sets ME and HI using unit weights results in a total scheduling output set ranging from -3 to 11 with a maximum of 0.5. Since there exists only one output fuzzy set for each of buffer level and process plan step, these output sets become the respective total output sets.

**Step 6:** defuzzify the output set

In the current system a centroid defuzzification scheme is used to calculate the singular output for each of the total output sets. The calculation for the scheduler total output set is considered first. For the fuzzy output set C the $y_j$ and $C(y_j)$ values for use

in equation 4.10 are as follows:

Table 4.1: Centroid Calculation Values

| $y_j$ | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|
| $C(y_j)$ | 0 | .25 | .25 | .25 | .25 | .25 | .25 | .5 |
| $y_j$ | 5 | 6 | 7 | 8 | 9 | 10 | 11 | |
| $C(y_j)$ | .25 | .25 | .25 | .25 | .25 | .25 | 0 | |

where $y_j$ represents the distance of element j from the origin and $C(y_j)$ represents the value of the output set at the location $y_j$. The above values when substituted into Equation 4.10 over the entire range of k = 15 elements results in a fuzzy centroid of 4.0. It is interesting to note that the fuzzy scheduling output set is unimodal, symmetrical and has a sharp maximum. As more scheduling rule sets are added or different rules are fired, this may not be the case. The fuzzy schedule centroid is then used as the part's scheduling rule set priority value.

The buffer level and process plan step centroids are determined in a simliar manner to be 7.0 and 13.0 respectively. The final part priority level is then calculated as a weighted sum of the three separate rule set priority levels (centroids). The various rule sets are weighted as follows:

scheduler rule set weight = 3

buffer level rule set weight = 1

process plan step weight = 1

The final priority level is then calculated to be :

final priority level = (3·4 + 1·7 + 1·13) / (3 + 1 + 1) = 6.4

The final priority value ranges from 1 to 19 where a 1 represents a high priority to process the part and 19 represents a low priority. Once priority levels have been generated for all the parts in the machine's input buffer, the machine selects the part with the highest priority, the part is loaded, and processing commences.

## 4.3 DISPATCHING RULE SUB-MODULE

In addition to the fuzzy logic method, the scheduler is equipped with several common dispatching rules which can also be chosen by the program operator to generate part priority levels. Currently there are five different dispatching rules to chose from: first in first out (FIFO), last in first out (LIFO), earliest due date (EDD), shortest processing time (SPT), and slack per operation remaining (Slack/OPNR). The main reason for including these five dispatching rules was to provide a basis for comparison with the fuzzy logic dispatching method. These rules are more rigorously defined below

and are used to select the part which will be processed on the machine next. The following symbols are used:

- $i$ - part
- $j$ - machine
- $k$ - operation
- $a_{ij}$ - arrival time of part i at the queue of machine j
- $d_i$ - due date of part i
- $z_j$ - selection of the part for processing on machine j
- $s_j$ - set of all parts in the input buffer of machine j
- $p_{ik}$ - processing time of operation k on part i
- $t$ - current time
- $q_i$ - current operation
- $m_i$ - number of operations
- $r_i$ - summation of remaining processing times

The following mathematical formulations are used for each respective rule (or strategy) and are applied to all the parts in the machine's input buffer (Cho and Wysk [3], Blackstone et al. [2]). The part which best fulfills the particular strategy is the one chosen for processing.

FIFO:

$$z_j \leftarrow \min_{i \in s_j}[a_{ij}] \tag{4.13}$$

LIFO:

$$z_j \leftarrow \max_{i \in s_j}[a_{ij}] \tag{4.14}$$

EDD:

$$z_j \leftarrow \min_{i \in s_j}[d_i] \tag{4.15}$$

SPT:

$$z_j \leftarrow \min_{i \in s_j}[p_{ij}] \tag{4.16}$$

S/OPNR:

$$z_j \leftarrow \min_{i \in s_j}[(d_i - t - r_i) \;/\; r_i] \tag{4.17}$$

where:

$$r_i = \sum_{k=q_i}^{m_i} p_{ik} \tag{4.18}$$

This then concludes the discussion of the scheduler and its sub-modules. The main functional aspect of the scheduler is the fuzzy logic dispatching method which uses fuzzy logic inferencing to assign priority levels to the parts waiting in a particular machine's input buffer. The use of fuzzy logic allows the incorporation of a number of

different criteria and goals and therefore considers many aspects of the cell status before setting a priority level. The rule set and the membership functions increase the flexibility of the system, and as will be shown later, provide mechanisms for learning or optimization. The scheduler works closely with the supervisor in order to facilitate real time part dispatching and cell control. A discussion of the supervisor functions will be presented next in chapter 5. The actual implementation of the scheduler within the Smalltalk environment as well as the procedure used to access and utilize the scheduler will be discussed in the implementation section, Chapter 6.

# CHAPTER 5

## 5. SUPERVISOR

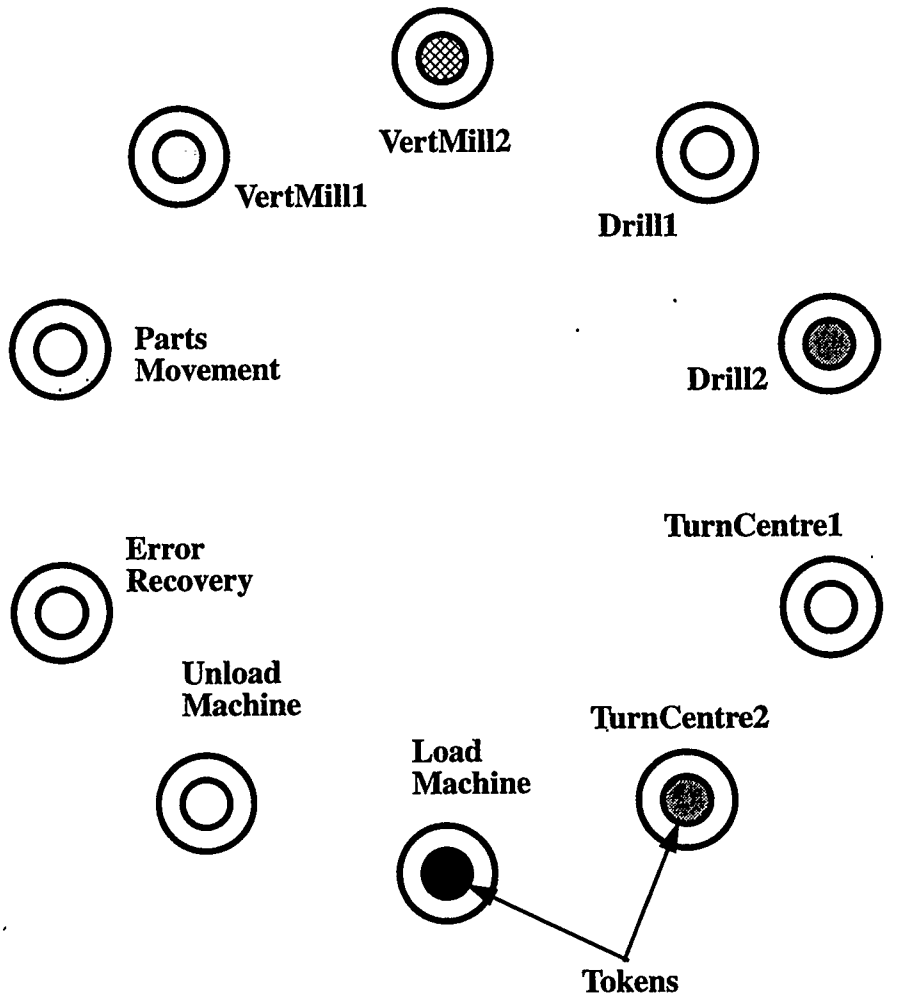
### 5.1 GENERAL FUNCTIONALITY

The ongoing cell control is performed by the supervisor. In the proposed control configuration the supervisor performs its function by monitoring the shop floor for specific types of events, and then reacts to these events by sending messages to the appropriate elements in the shop floor. The supervisor performs a number of tasks which, for the purposes of this system, can be separated into two areas: supervisory tasks and simulation tasks. The supervisory tasks are those tasks which are normally required of a supervisory system in a real shop floor. These tasks include: controlling the machines, monitoring for breakdowns, initiating error recovery routines, directing part movement and part production, controlling the material handling system, initiating the scheduling of parts, calling for part pick-up, monitoring inventory levels, monitoring the input buffer

and finally, virtually reconfiguring the cell when required. The simulation tasks are those tasks which the supervisor performs to allow the functioning of the current simulation system such as: file management, initialization, data collection and printing, and the control of simulation time. Some of the simulation functions such as file management, data collection, data evaluation and the printing of statistic data, would carry over into a real shop floor situation. The supervisory tasks are discussed below while the discussions about the control of the actual simulation and the detail on how each of the supervisory tasks have been implemented are left for the next section on implementation.

Before going into the details of the supervisory tasks, a review of the general structure of the supervisor is in order. The supervisor structure is comprised of two levels; a primary and a secondary control level similar to the two level structure proposed by Hasegawa et al. [12]. The primary control level is represented as a group of control nodes as shown in Figure 5.1. Here ten nodes represent six part processing machines, three different types of material handling operations, and one set of error recovery procedures. A sequence of instructions is attached to each node and these instructions are considered to be the secondary control level which, for the purposes of this project, have been represented using Petri nets. The supervisory control structure was developed in this manner in order to provide a maximum amount of flexibility. Additional machines or processes can easily be added to the primary control level without affecting the secondary control level underneath. Similarly, a piece of equipment can be replaced with another type of equipment or the sequence of operations may be completely changed in

Figure 5.1: Primary Control Level



the secondary level without affecting the primary control level.

The primary control level represents a key element of the supervisor control structure. This level represents the control link between the ongoing monitoring performed by the supervisor and the actual detailed control sequences which drive the various pieces of processing equipment and material handling equipment. As the supervisor reacts to specific events within the shop floor it places tokens in the appropriate nodes of the primary level to activate the operations at the secondary level. Tokens of different attributes are used to designate a certain type of part or a certain type of error. When a token is removed from a node in the primary level, the underlying secondary control level is deactivated.

The secondary control level is the detailed control required to perform a specific action such as loading or operating a machine. The focus of this research was not on this level of control; however, a brief discussion of the secondary control level is included below for completeness. The main discussion of this chapter will be on how the supervisor completes its required tasks.

## 5.2 SUPERVISORY TASKS

### 5.2.1 Part Movement and Part Production

Two of the primary tasks of the supervisor are to control and direct the movement of parts and to facilitate part production throughout the shop floor. The application of these two tasks occurs primarily when a machine completes processing a part. Once a machine has completed processing a part, the supervisor initiates a sequence of steps which will first, move the part to its next destination and second, initiate selection of the next part for processing on the current machine. To determine the next destination of a newly completed part the supervisor accesses the part's process plan. The next step in the part's process plan dictates the next destination for the part and this destination could either be a machine or, if the part has been completely processed, the cell's output buffer. If the next destination is another machine the supervisor checks the machine status and the tool status to ensure everything is functioning properly. If so, the supervisor updates the part's process plan and has the part placed into the input buffer of the new machine. If everything is not working properly, then the supervisor goes through a series of steps which allows the part to be processed by an alternate machine. In the current system configuration each machine has one alternate machine and if a tool is broken or the entire machine is inoperative, then the part is redirected to the alternate machine. Prior to having the part moved, the supervisor also checks to ensure there is room in the next machine's input buffer. If so, the part is moved. If not, the supervisor makes a note

regarding the full buffer to ensure further blocking or deadlocking checks are carried out. As well, if the part cannot be moved then the machine is placed into a 'full but idle' status with the part sitting in the machine until room becomes available.

Once the recently finished part has been removed, the supervisor contacts the scheduler to initiate the part selection process for the current machine. After a part is selected the supervisor has the part loaded into the machine and initiates processing.

### 5.2.2 Material Handling

The supervisor also directs and controls the material handling system whenever any type of part movement is required. Five part movement scenarios are currently included in the simulation: loading a machine, unloading a machine to another machine's input buffer, unloading a machine and placing the part in the cell's output buffer, loading a part from a cell input buffer into a machine input buffer, and moving a part from one machine input buffer into another. As discussed above, the supervisory control is done at the primary control level. The supervisor simply informs the material handling system, in this case a robot, where to move and also which part to move. As will be shown later, the secondary control structure takes care of all the other details. The supervisor is informed when the robot's movement is complete.

### 5.2.3 Machine Control and Machine Status Monitoring

The supervisor is responsible for controlling all the machines. This includes ensuring that the appropriate tools and processing software are loaded and that the correct part is on the machine. The supervisor activates the machines at the correct time and monitors for part completion. The supervisor also monitors each machine for a change in status such as a breakdown or a repair event. In the case of a breakdown the supervisor coordinates removal of the part currently being processed, redistribution of any parts from the machine's input buffer to an alternate machine and sending out an error message indicating that the machine has broken down. The scope of the current project did not include the implementation of a machine repair routine or of a very elaborate part redistribution mechanism. The supervisor ensures that parts which would normally be processed on the broken machine are redirected to an alternate machine. If a machine has been repaired then the supervisor ensures that parts are no longer directed away from the machine but are instead processed in the normal fashion. Tool breakages are dealt with in a similar fashion. If a tool breaks, the parts are redirected to an alternate machine until the tool is repaired or replaced with a new one. A mechanism for alternate tools has not been put into place in the current simulation as it was felt that the redirection of the part was a more rigorous problem since selection of an alternate tool would not affect part movement or part scheduling.

### 5.2.4 Error Recovery

One important task of the supervisor is to initiate error recovery routines if things go awry. Error recovery is an entire study in its own right and was not considered to be within the scope of this research except to the extent of dealing with machine or tool breakdown (which have been discussed) or system deadlocking. In the current stage of development, if deadlock occurs the system simply generates an error message and the simulation stops. However, it should be noted that the control structure goes through all the appropriate steps. For example, when a deadlock error is generated, error recovery is initiated at the primary control level and action is taken; in this case an error message is generated. The control structure, by its design, allows the later addition of a very basic or a very elaborate error recovery module at the secondary control level.

### 5.2.5 Virtual Reconfiguration

The concept of reconfigurability allows for the physical or virtual reconfiguration of a cell or system to allow either improvement of the system or to provide some other specific functions. One such specific function is the arrival of a special rush order which has priority over every other order currently within the system. In this situation the supervisor adjusts the current cell structure into a virtual structure which allows the special order to be processed as quickly as possible. The mechanism whereby the supervisor does this is by consistently ensuring that the special rush order parts have the

highest priority over all other parts within the system irrespective of due dates, process plan issues, buffer loading or inventory levels. Once the special order has completed processing the virtual cell is dismantled and the system operates as normal.

### 5.2.6 Cell Buffer and Inventory Monitoring

The supervisor must, on a continuous basis, monitor the input and output cell buffers as well as inventory levels. As parts arrive and as room becomes available within the system, the supervisor ensures that the parts get placed into the appropriate machine input buffer. Similarly the supervisor monitors the cell output buffer and calls for part pick-up when necessary. The supervisor dictates the output buffer level at which the buffer must be emptied and also the amount removed. As parts get placed into inventory, the supervisor updates inventory levels.

## 5.3 SECONDARY CONTROL LEVEL

The secondary control level has been represented using Petri nets (PN) in order to show some of the types of operations which occur at this control level. The following four Petri nets will be shown but not discussed in detail: loading, operating, and unloading a machine, and moving a part. First, however, a general description of Petri nets will be given.

### 5.3.1 Petri Net Description

Petri nets provide a powerful tool for representing and analyzing asynchronous and concurrent systems. There are many advantages in modelling a system using Petri nets: (1) they describe the modelled system graphically, enabling easy visualization of complex systems, (2) they allow modelling of the system hierarchically, (3) a systematic and complete qualitative analysis of the system is possible using well-developed existing Petri net techniques, (4) well formulated Petri net synthesis schemes exist to aid development of Petri nets and (5) the performance evaluation of a system is possible using timed Petri nets (Kamath and Viswanadham, [20]). Petri nets have been used in many flexible manufacturing system applications including cell control applications (Teng and Black [39], Hasegawa et al. [12], Zhou [45], Ravichandran and Chakravarty [38], Merabet [34], Tzafestas [41]) and are used in this application to represent the detailed control of robot movement and machine operation.

A PN is comprised of a set of places, a set of transitions and a set of directed arcs which connect the places to the transitions. Pictorially, the places are represented by circles and the transitions by bars. Places may contain tokens (shown as dots) which then create a marked PN; the 'marking' of a PN represents the current state of the system being modelled. Generally, places represent conditions and transitions represent events where the presence of a token in a place represents a positive (true) condition while an empty place represents a negative (false) condition. The occurrence of an event is

represented by firing a transition and this results in the movement of a token from the input place (a place from which the arc is directed to the transition) to an output place (a place to which the arc is directed from the transition). This results in a change in the system state of the Petri net. A transition can be fired only if all the input places to that transition are enabled and when it fires, a token is removed from each of the input places and a token is added to each of the output places. A transition may also be inhibited through the use of inhibitor arcs. An inhibitor arc from a place to a transition is drawn with a circle at the end instead of an arrow and causes the transition to be disabled if a token is located in the place connected to the arc.

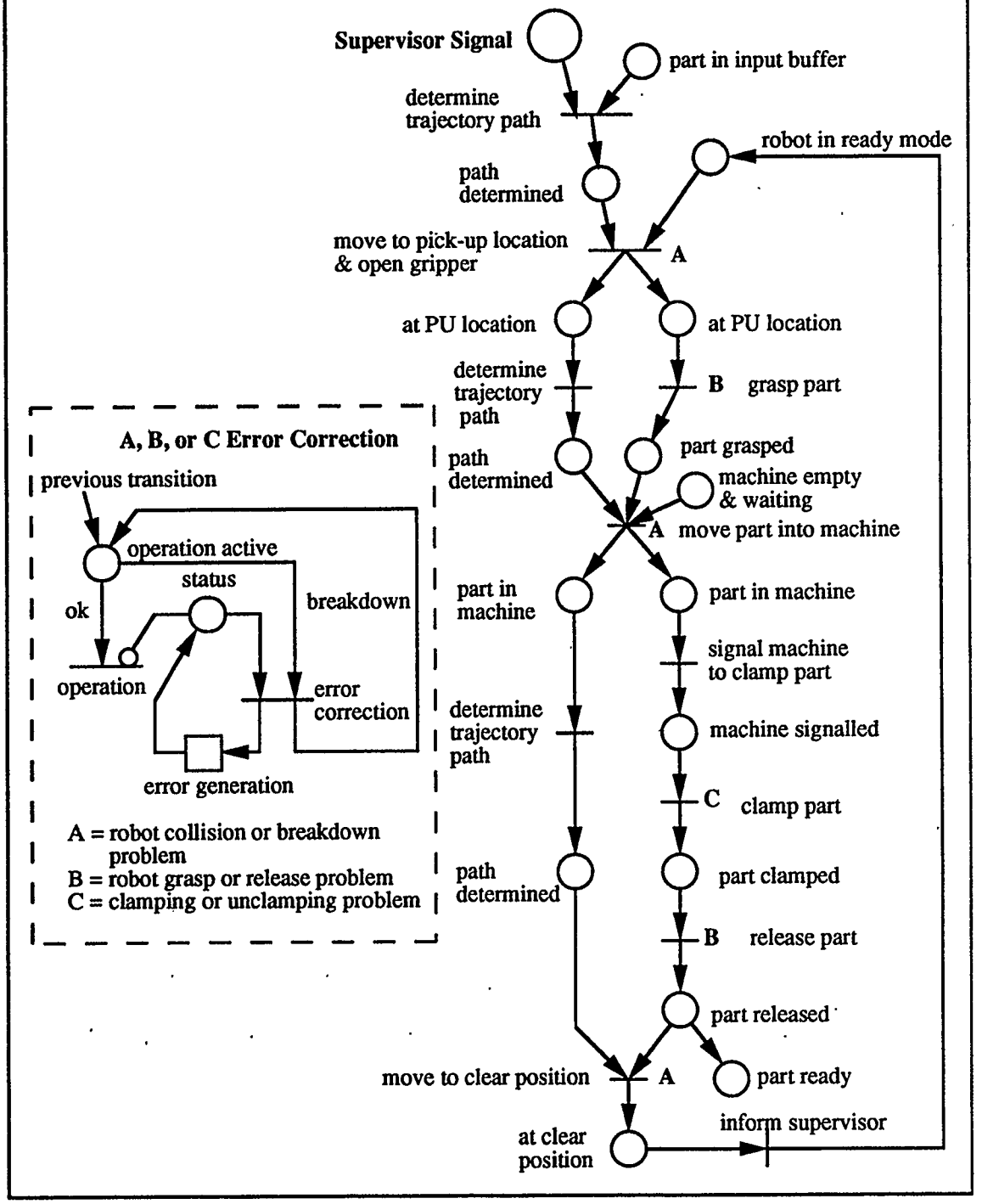
A reachability tree can be developed for Petri nets and it describes if all markings can be reached. If so, the Petri net is considered live. This is an important feature as it indicates that the PN and therefore the system being modelled is free of deadlock and will perform as modelled. Boundedness is another important property of a PN. It defines if a section of the system is bounded; for example a buffer may be bounded in that it may hold only a certain number of parts. All these aspects of PNs can be analyzed mathematically or simulated, allowing systems to be analyzed in great detail. Timed Petri net simulations have been developed which allow real time simulation of the process. Timed PNs have transitions which are fired at appropriately timed intervals, thereby simulating the time required for each event. Coloured PNs have also been developed to help model several identical processes and to reduce the size and complexity of the PN. In a coloured PN, a colour is associated with each token as well as with each place and

transition. The reader is referred to the above references for more detailed descriptions of Petri nets and their applications. Next, each of the four Petri nets developed for this project will be described.

### 5.3.2 Petri Net for Loading a Machine

The PN depicting the sequence of operations for loading a machine is given in Figure 5.2. To initiate the loading action the supervisor places a token (with a certain attribute representing the type of part) in the "Load Machine" node of the primary level. The robot determines the appropriate trajectory path depending upon the location of the part pick-up point in the input buffer and the type of part (represented by the token attribute). The methodology is as follows (refer to Figure 3.1): The part type and location are given to the robot controller which then passes this information on to the trajectory planner. The trajectory planner accesses the data base and if it has the appropriate path on file it simply loads the path to the code generator, which generates the code and passes the completed code to the controller for execution. If the path does not exist then the trajectory planner will plan the trajectory based upon its information of the work cell, part type and part location in conjunction with part grasping information. The work cell lay out and part grasping information are contained within the data base and can be updated manually or via the cell controller when it requests production of a new part. Alternatively the part position can be determined using sensors or a vision system and this information could be given to the robot controller. The new trajectory is then saved and

Figure 5.2: Petri Net for Loading a Machine



passed on to the code generator for processing.

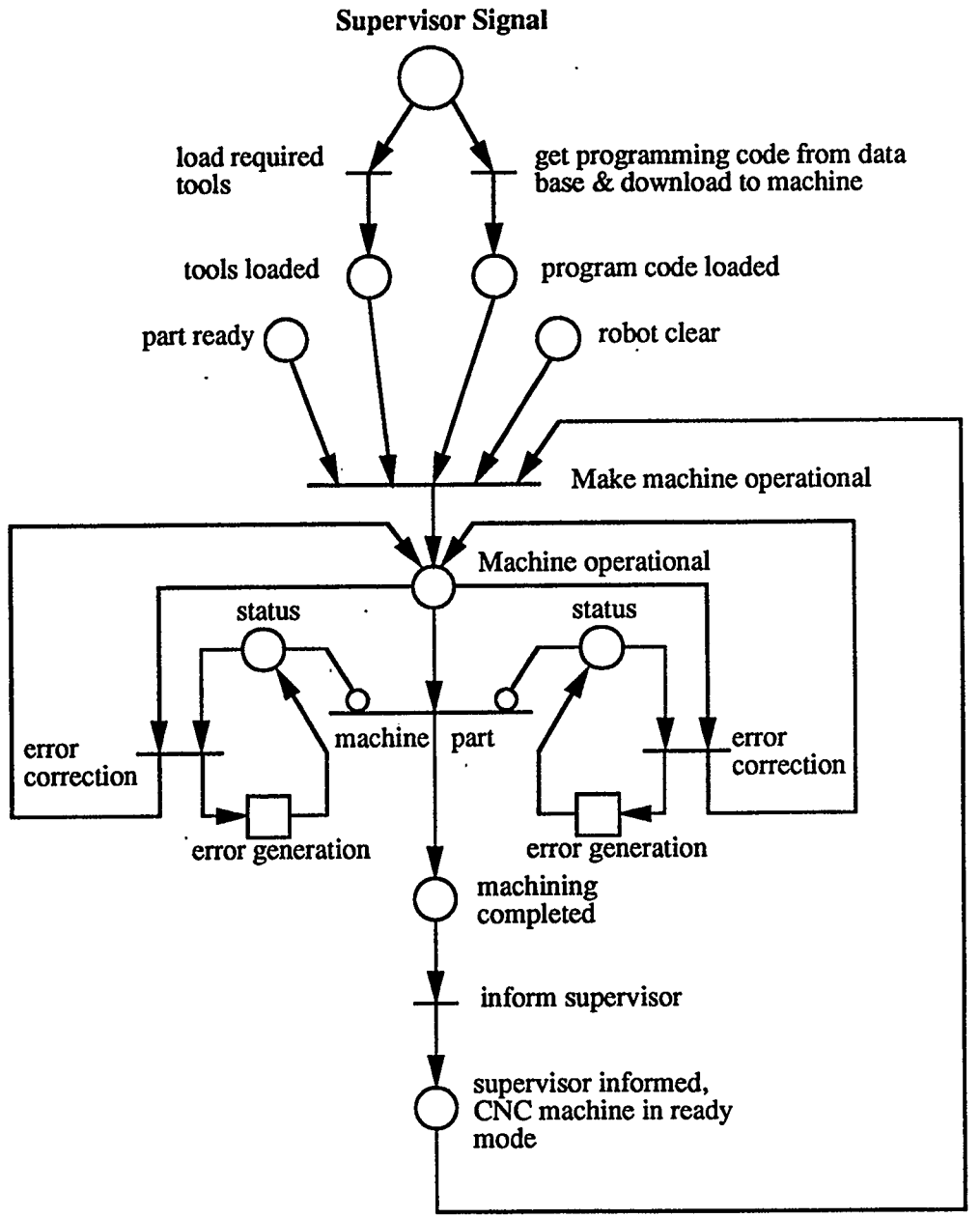
Once the path is determined, if the robot is ready (has completed all other movements) then the robot will move to the pick-up location opening its gripper on the way. At the pick-up location the robot will grasp the object and while doing so, the controller will initiate action to determine the trajectory path to the set down location in the machine. Once the trajectory is determined (similar procedure to that described above), and the part is gripped, and the machine is ready, the robot will move and place the part in the machine. The robot will signal the machine to clamp the part and while the clamping is taking place the robot controller will access the trajectory path to the clear position. As soon as the part is clamped, the robot releases the part and moves to the clear position. The robot then signals the supervisor and places itself in ready mode.

The capital letters A, B and C at the various transitions in Figure 5.2 (and the subsequent figures) indicate error checking (Teng and Black [39]). The error checking Petri net is also shown in Figure 5.2 and operates as follows. First the operation token is fired from the previous transition thereby initiating the task. If an error condition occurs while executing the task to which the error check is attached, a token is placed into the status place thereby disabling the task execution. The tokens in the operation and status places now activate the error correction. When the error has been corrected a token is removed from the status place and another is placed into the operation place thereby enabling task execution.

### 5.3.3 Petri Net for Operating a Machine

The generic PN depicting the sequence of operations for operating a machine is given in Figure 5.3. First the supervisor signals the machine by placing a token in the node of the particular machine to be activated. The attributes of the token determine the type of part to be machined. A sub task element of the supervisor gets the correct program code and tool requirements from the data base, down loads the program to the machine controller and initiates action to load the required tools if they are not already loaded. Once the tools and program are loaded, the robot is clear, the machine is ready, the part is ready and no error conditions exist, the part can be machined. The error conditions for tool and operation work in the same manner as described above. Once the part is machined, the machine controller informs the supervisor and places the machine in ready mode. As noted in Figure 3.1, motion planning for the machine tools and program code generation is in an isolated module. It is anticipated that these activities will have occurred during the design stage at locations away from the shop floor using appropriate software packages such as SmartCAM. The resulting code and tool requirement specifications have then been loaded into the data base to be accessed by the supervisor sub task element.

Figure 5.3: Petri Net for Operating a Machine



### 5.3.4 Petri Net for Moving Parts

The Petri net for moving the parts is given in Figure 5.4. As before, the procedure is initiated by placing a token in the "Parts Movement" node of the primary level. The attributes of the token indicate to the robot controller the type of part and the procedure followed is similar to that given above for loading a machine (except for clamping).

### 5.3.5 Petri Net for Unloading a Machine

The Petri net is given in Figure 5.5. The operation is initiated with a token in the "Unload Machine" node of the primary level. This initiates trajectory planning by the robot controller. Once the trajectory path is determined and the machine and robot are ready, the robot will move in to pick up the part. When the part is properly grasped the robot will signal the controller to unclamp the part, after which the robot will move the part to the set down location. While part grasping and unclamping is taking place, the robot controller determines the trajectory path. Upon completion of the task the robot places itself in the ready mode and informs the supervisor. The error detection is similar to the method mentioned above.

This then completes the description of the supervisor module of the cell controller. The supervisor structure, with its two levels of control, exhibits flexibility due to its modularity and ease with which modifications may be made to either of the two control

Figure 5.4: Petri Net for Moving Parts

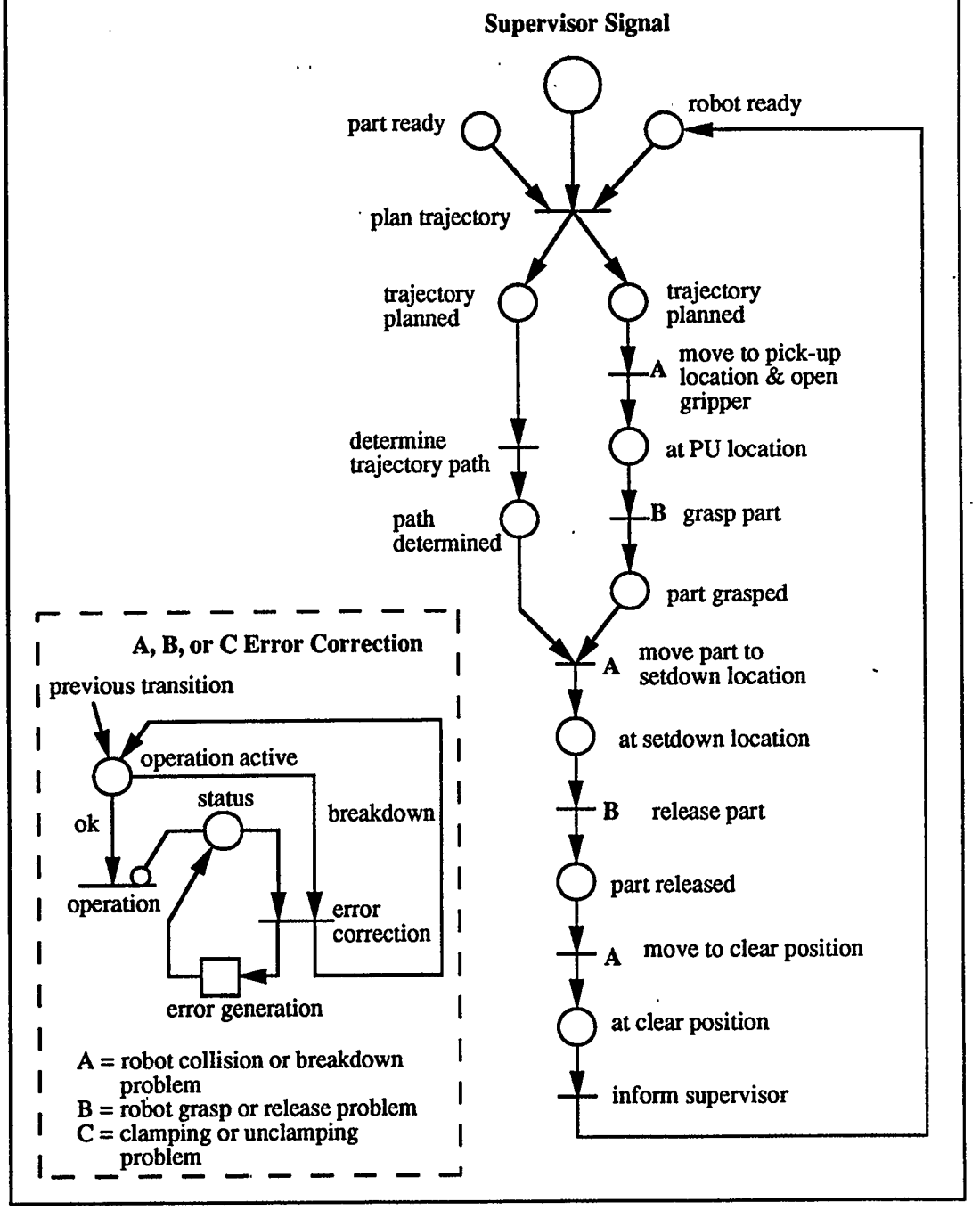
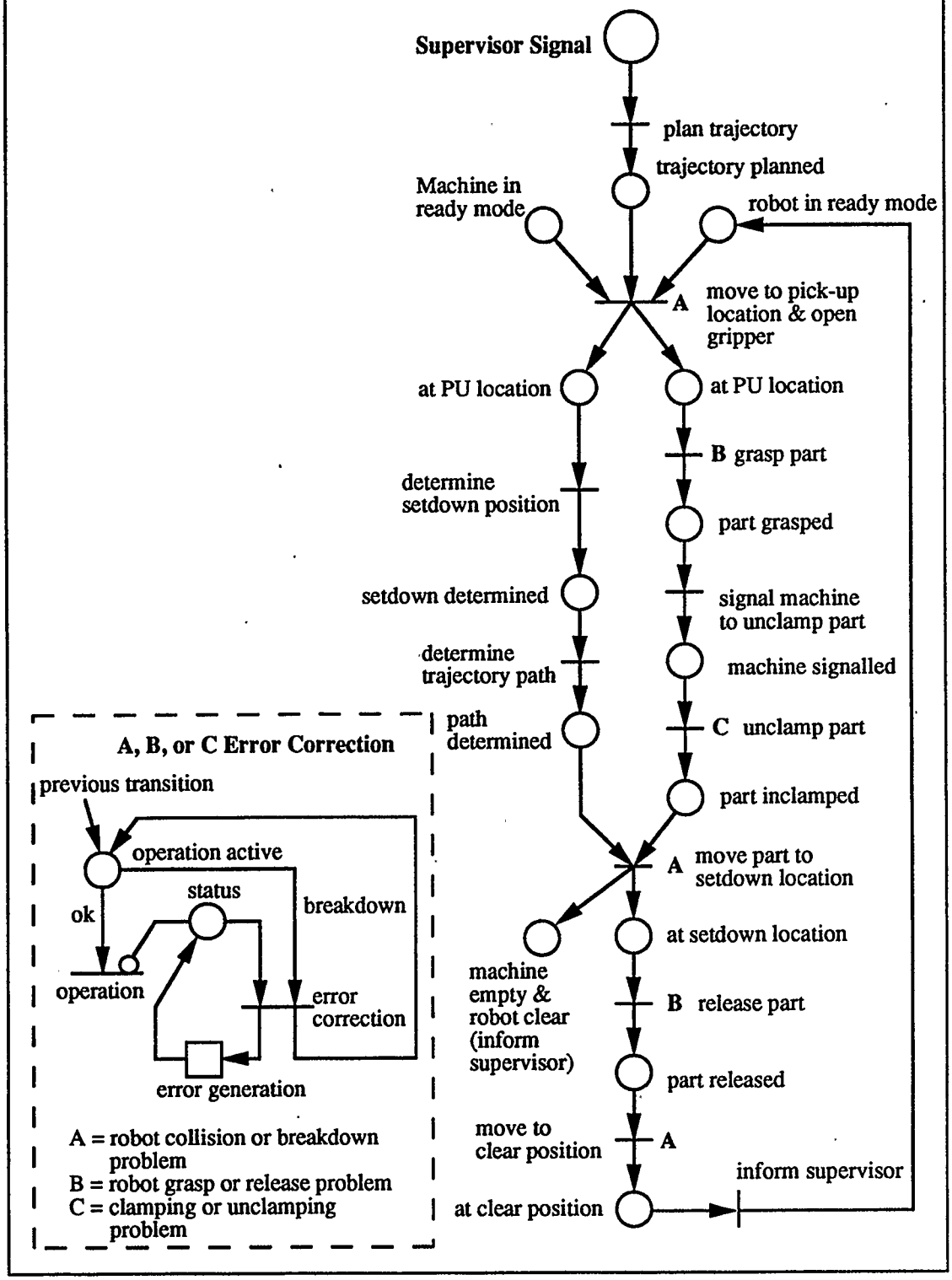


Figure 5.5: Petri Net for Unloading a Machine



levels. The supervisor has been designed to deal with the many required tasks of cell supervision and has also been designed to work interactively with the scheduler module. Utilizing cooperation between the supervisory and scheduling modules allows for coordinated real time control to take place. The actual implementation and mechanics of the supervisor and scheduler will be discussed next. The emphasis will be on the general operation of the controller and the linkages which drive the decision making process rather than an in depth detailed discussion of the actual object oriented programming code. This falls in line with the purpose of this thesis which is to develop and test new concepts rather than very specific detailed applications.

# CHAPTER 6

---

## 6. IMPLEMENTATION

## 6.1 INTRODUCTION

The proposed control structure has been implemented in the object oriented programming environment, Smalltalk 80. The object oriented environment is particularly suited for this type of application since it allows for a relatively straight forward representation of a manufacturing environment as a group of objects having certain attributes, knowledge and reasoning abilities. These objects communicate with each other in order to fulfil their own separate functions.

This chapter will review in detail the overall class hierarchy of the cell control structure. It will discuss where the scheduler and controller methods are imbedded, and it will also describe the methods which exist in the fuzzy logic class structure and how

they are accessed. Finally the chapter will conclude by giving a description of the simulation which is driven by the control structure and will also present an overview of the data gathering facilities.

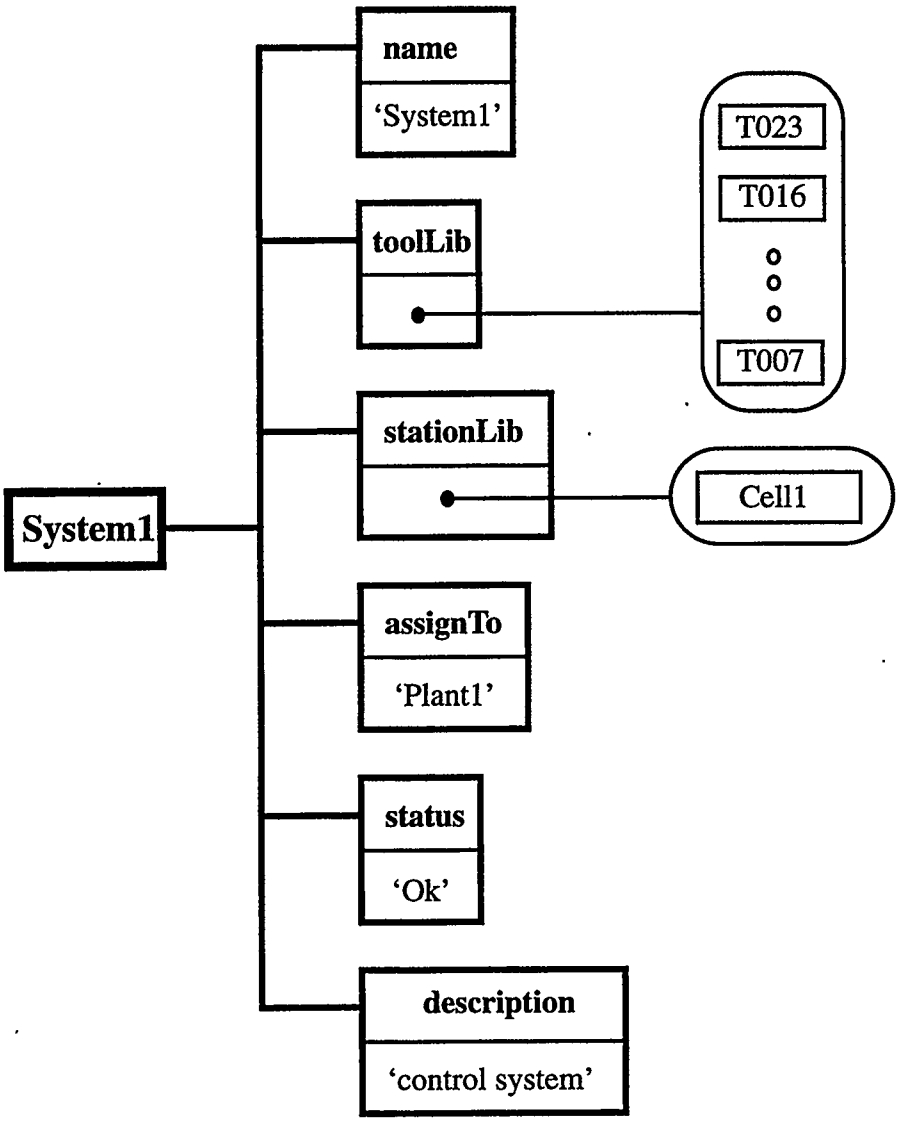
## 6.2 OBJECT ORIENTED CLASS HIERARCHY

As shown in Figure 3.2, the class hierarchy is made up of seven new classes: **System, Fuzzy Logic, Cell, Equipment, PartType, Part,** and **Tool,** all of which have been placed under the Smalltalk 80 **Magnitude** class. The class hierarchy has been developed in this manner in order to use the concept of inheritance as much as possible, thereby allowing attributes and methods to be passed down to classes lower in the hierarchy. The hierarchy has also been developed to represent the physical organization of parts, equipment, and cells within a manufacturing system as realistically as possible and although the main thrust of this work is cell control, the class hierarchy was developed in three levels to include three separate groups of elements: system, cell, and equipment. Including a system class in the hierarchy provides a place where global decisions can be made and passed down to the various cell controllers. Each cell controller, as will be discussed in more detail later, is imbedded mainly at the cell level but interacts heavily with equipment; hence the inclusion of cell and equipment classes. The class hierarchy can be thought of as a mechanism to separate the system, cell, equipment, and part objects, and provide containers in which to place unique knowledge or functional abilities.

### 6.2.1 System Class

The **System** class represents the shop floor of a manufacturing system. Each instance of the **System** class has the following instance variables or attributes: *name*, *toolLib*, *stationLib*, *assignTo*, *status*, and *description*. An example of an instance of **System** and all its instance variables is shown in Figure 6.1. The instance variable *name* is System1, which is simply the name assigned to the current instance of the **System** class. *ToolLib* contains a dictionary made up of **Tool** class instances. *ToolLib* can be thought of as a container for all the various tools available to System1 and could represent the shop floor's tool inventory. The instance variable *stationLib* contains a dictionary which is made up of instances of the **Cell** class. Thus, *stationLib* is a container representing all the manufacturing cells grouped within the particular system. In this case only one cell, Cell1, and one system, System1 have been represented. *AssignTo* is used to designate the structure or element in the class hierarchical level above the current class level to which System1 belongs; in this case Plant1. The instance variable *status* denotes the operating status of the system. The system is either 'Ok' meaning that everything is functioning properly, or 'Broken'. *Description* simply provides a linguistic representation of what the system is or does. All of these instance variables are inherited by all the subclasses of the class hierarchy, and although the actual values contained in the instance variables will differ, conceptually the functional representation of the instance variables will not change. It should also be noted that in some subclasses, not all of the instance variables developed in previous classes are used.

Figure 6.1: An Instance of Class 'System'

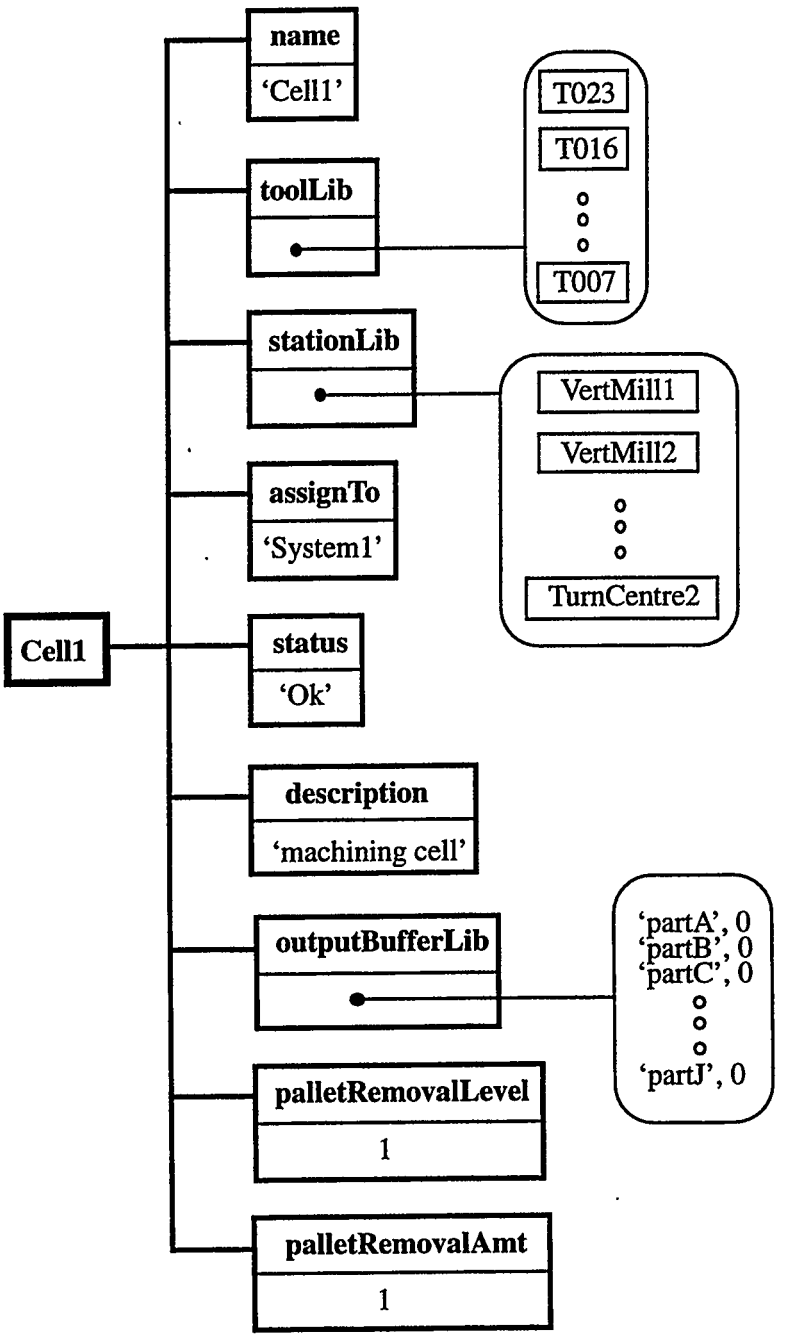


### 6.2.2 Cell Class

The next class in the class hierarchy is the **Cell** class. As shown in Figure 6.2, the six instance variables developed in the **System** class have been inherited by the **Cell** class. The instance variables *name*, *assignTo*, *status*, and *description* are self explanatory. *ToolLib* still consists of a dictionary containing instances of tools, but now these tools represent the tools used by Cell1 and not by the whole shop floor. As before, *stationLib* is a container of instances but in this case the instances are **Equipment** instances (or objects). All of these **Equipment** objects are considered to be the elements making up the **Cell** instance Cell1. Three more instance variables have been added at this level: *outputBufferLib*, *palletRemovalLevel*, and *palletRemovalAmt*. These three instance variables are concerned with the control of completed parts and how they leave the cell. *OutputBufferLib* is a listing which tracks the number of completed parts of each part type that arrives at the cell output buffer. The instance variables *palletRemovalLevel* and *palletRemovalAmt* dictate the cell's output buffer inventory level at which pallets (parts) need to be removed and also how many should be removed at one time.

The **Cell** class also contains a number of methods and class variables. The bulk of the methods deal with the initialization of cells, tool, parts etc. as well as with certain aspects of running the entire simulation. All the methods for the different classes will be discussed later in Section 6.3. The class variables are variables which are accessible to all instances of the class and all instances of the subclasses. Certain class variables at the

Figure 6.2: An Instance of Class 'Cell'

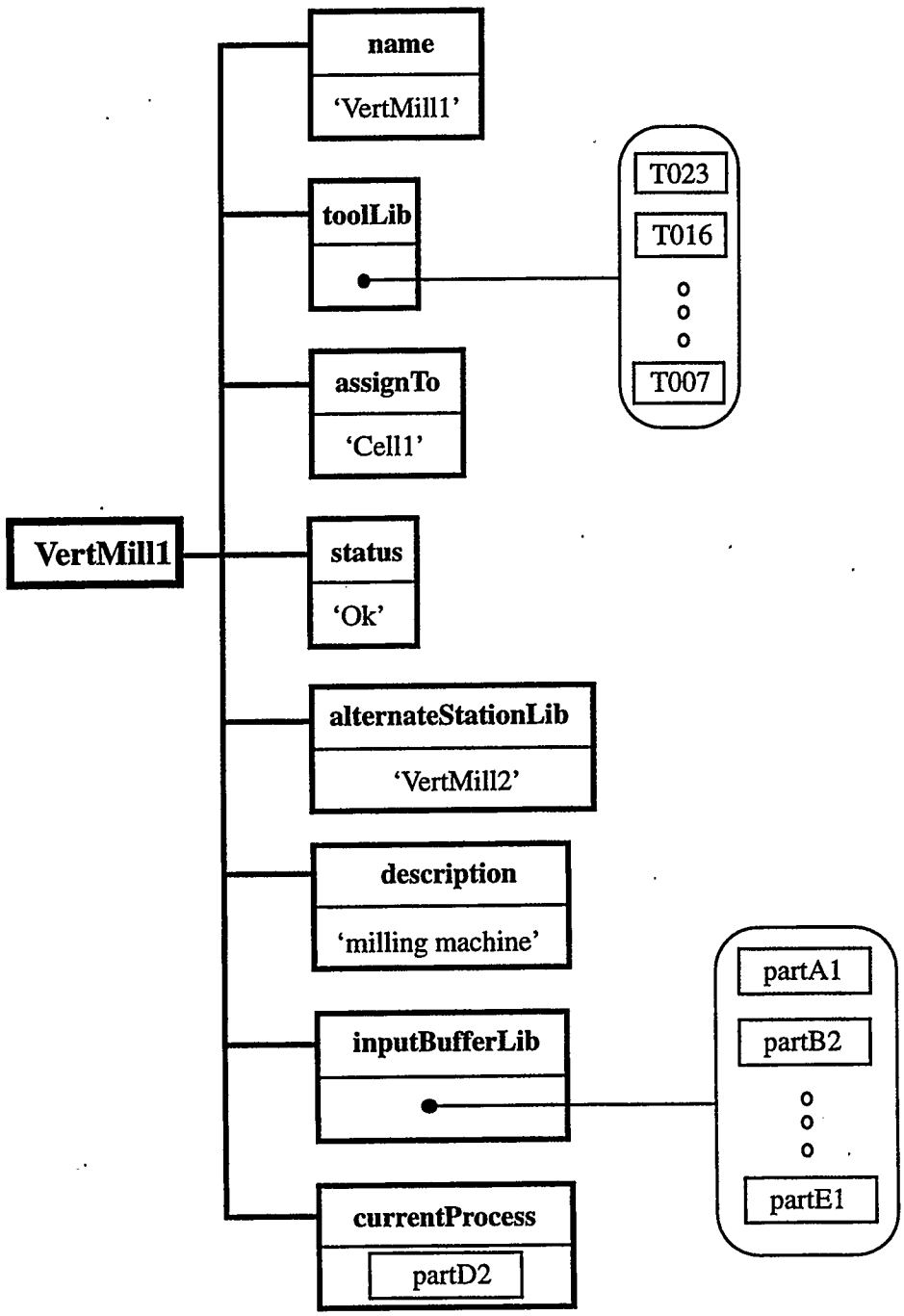


**Cell** level are used to store information required to run the simulation. The remaining class variables store data on equipment, parts and certain data for the fuzzy logic algorithm.

### 6.2.3 Equipment Class

Five instance variables (*name*, *toolLib*, *assignTo*, *status* and *description*) have been inherited by this class from the various super classes. The remaining instance variables from the super classes are not used at the **Equipment** class level at this time and therefore will not be discussed further. Three new instance variables have been added at this level (see Figure 6.3): *alternateStationLib*, *inputBufferLib*, and *currentProcess*. *AlternateStationLib* contains the name of an alternate machine to which parts are redirected if the current machine breaks down. At the current stage of development only one alternate machine is specified; however, in the future it is anticipated that multiple machines could be used which can either fully or partially take over the current machine's function. The instance variable *inputBufferLib* consists of a dictionary which contains instances of the **Part** class. In the real world this would represent all the parts waiting in the input buffer of the machine. *CurrentProcess* contains an instance of the **Part** class which represents the part that is currently being processed on the machine. A listing of the **Equipment** class instances used in this project is given in Appendix A.

Figure 6.3: An Instance of Class 'Equipment'



### 6.2.4 Tool Class

The **Tool** class provides all the instance variables for the tools that are used in the entire system. All the instance variables (shown in Figure 6.4) are inherited from the **System** class. The instance variable *assignTo* in this case designates the particular machine or piece of equipment that will have access to each particular tool. A listing of the **Tool** class instances used in this project is given in Appendix B.

### 6.2.5 PartType Class

In order to maintain some uniformity in the parts being processed by the system, the class **PartType** was developed to define the attributes of several specific types or groups of parts. The simulation as it currently stands is able to represent groups of similar parts, individual parts of each group, or in an extreme case, a group of parts which contain only one part. Thus, situations ranging from a few large groups of parts to a large number of single parts could ultimately be modelled. The **PartType** class uses only one inherited instance, *name*. The rest of the instance variables shown in Figure 6.5 are unique to the class **PartType** and its subclasses. The instance variable *processPlan* contains a two dimensional array which describes the process plan for each type of part. The process plan currently includes the tool required, the machine on which a particular process will be performed, and the time the process will take. *InventoryLevel* keeps track of the number of completed parts of a given part type. *TimeLevel* tracks the time that has

Figure 6.4: An Instance of Class 'Tool'

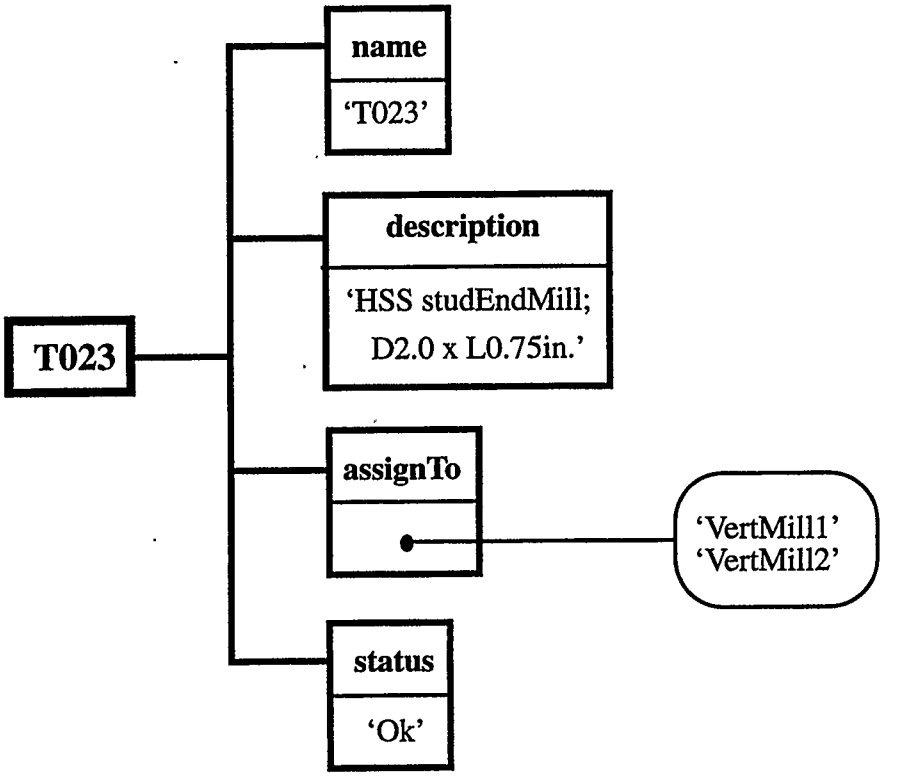
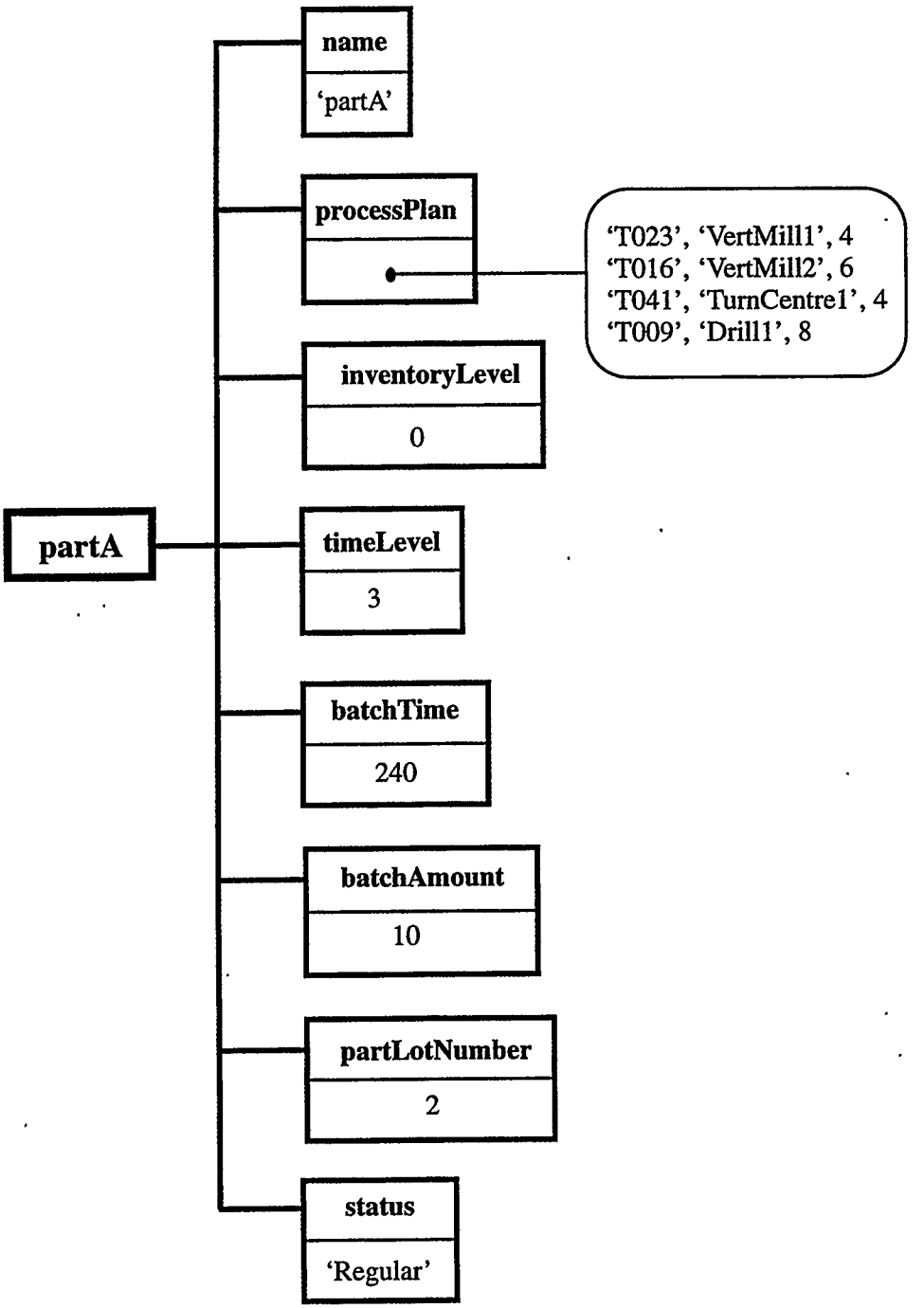


Figure 6.5: An Instance of Class 'PartType'

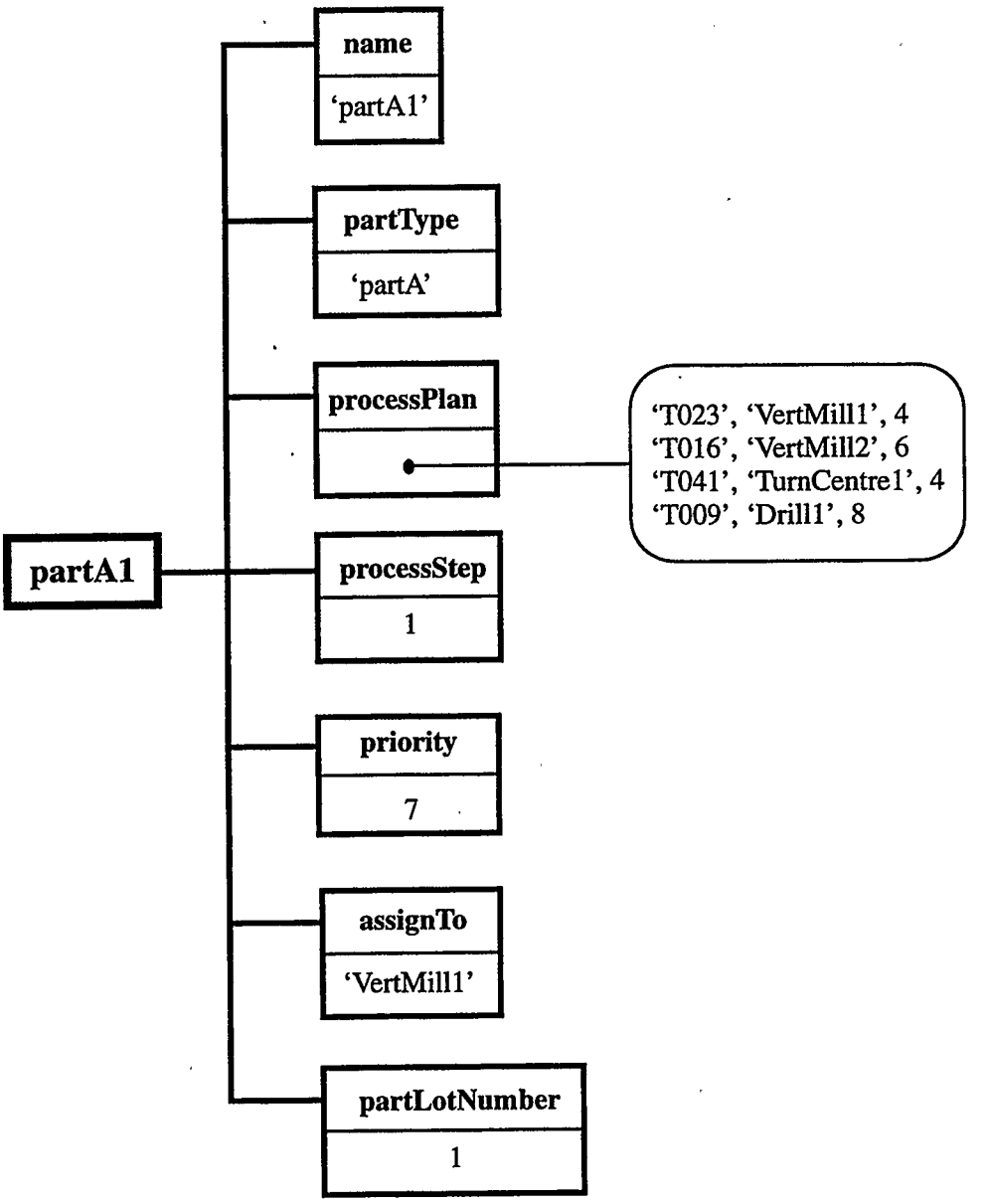


passed since work on a given type of part (batch) has begun and *batchTime* denotes the due date for the batch. *BatchAmount* defines the size of the batch in terms of a number of parts. The instance variable *partLotNumber* tracks which part of the given part type is currently entering the system and is used mainly to generate and identify new parts. *Status* denotes if the part type is regular or special. A special part type is one which is considered to be a rush part type. If the status of a part type is special then the system is virtually reconfigured to allow the part type to have priority over other parts. A mechanism also exists within the **PartType** class which allows the program operator to use an alternate process plan for a specified part. A listing of the **PartType** class instances used in this project is given in Appendix C.

### 6.2.6 Part Class

The **Part** class represents the actual parts which are processed by the system. The instance variables for **Part** are shown in Figure 6.6. The new instance variable *partType* defines the generic part type or group to which that part belongs. As before, *processPlan* denotes the process plan of the part, and this process plan is the same as the process plan for the part type. *ProcessStep* defines the next process that the part needs to undergo. The instance variable *priority* stores the current priority level of the part. *AssignTo* specifies the piece of equipment where the part is currently located. Finally, *partLotNumber* is used to identify a certain part in a given part type batch. All other inherited instance variables are unused at this time.

Figure 6.6: An Instance of Class 'Part'



### 6.2.7 Class Linkages

Figure 6.7 shows a little more clearly how all the classes are linked together to provide a coherent shop floor representation. The bolded path shows the interconnections between all the classes beginning with the **System** class instance System1 and ending with the **Tool** and **PartType** classes. System1 *stationLib* is the container holding all the **Cell** instances for System1; in this case there is only one cell, Cell1. Cell1 is connected to the **Equipment** class through Cell1 *stationLib* which contains all the different **Equipment** class instances for Cell1; one example of which is VertMill1. Each equipment instance has a *toolLib* instance variable which contains all the **Tool** class instances representing all the various tools that each piece of equipment can use; for example tool T023. The *inputBufferLib* instance variable of each piece of equipment contains instances of the **Part** class. These instances represent the actual parts which are awaiting processing on the machine. One of the parts awaiting processing on VertMill1 is partA1. Each of the **Part** instances belongs to a certain **PartType** which is designated by the **Part** instance variable *partType*. Thus, partA1 is of part type partA.

The only linkages which are not shown here are the linkages to the **FuzzyLogic** class. The **FuzzyLogic** class consists only of a number of methods which are linked to the rest of the structure using message passing. The fuzzy logic methods and the other methods in the structure are the real driving force behind scheduling and control and are discussed next.

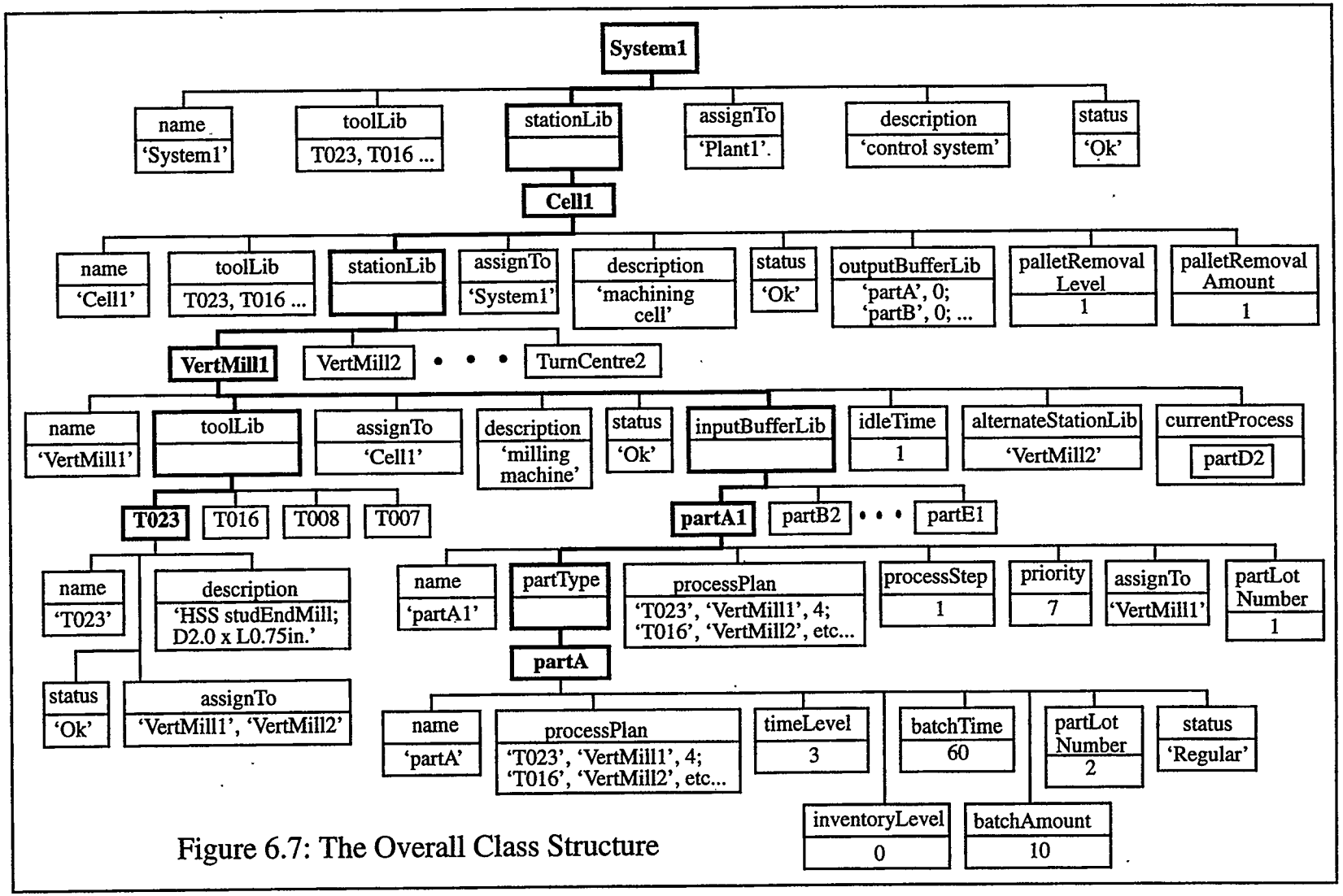



Figure 6.7: The Overall Class Structure

## 6.3 METHODS AND MESSAGE PASSING

The methods in an object oriented environment are the drivers behind any action. In this particular control structure there are methods which do control, scheduling, fuzzy logic analysis, and many other tasks. The methods are accessed when objects (including class instances) send messages to one another. These messages can send information, return information, or simply invoke a specific procedure. It is beyond the scope of this write-up to review all the different messages and methods that have been developed in any detail. Instead only the most important methods will be discussed and then only in a relatively superficial manner in order to provide an overview of how the overall control structure behaves.

### 6.3.1 Supervisor Methods

Most of the supervisor methods are imbedded within the **Cell** class level. This is appropriate since the controller at this stage has been designed to provide control for manufacturing cells. The supervisor methods include: ongoing monitoring of the cell for breakdown or repair events, blocking or deadlocking checking, control of part movement, control of material handling equipment, error recovery, virtual reconfiguration, and data collection. In short, the supervisor methods at the **Cell** class level coordinate the general running of the cell. Several supervisor methods are situated at the **Equipment** class level, but these are normally accessed using messages from the **Cell** class level methods. The

supervisor methods at the **Equipment** class level do the following: update equipment status as parts are moved around, update the blocking or deadlocking events, identify which scheduling method is to be used (chosen by operator), check the equipment buffer status (availability of space in the buffer), and select the highest priority part when part selection is required. There are also a few methods at the **Equipment** level which help in facilitating part scheduling.

### 6.3.2 Scheduling Methods

The bulk of the scheduling methods reside at the **Equipment** class level at this time mainly because the scheduling aspect of the control structure deals with the dispatching of parts from a machine's input buffer through the machine. The supervisor at the **Cell** class level initiates the scheduling process when it is informed that the machine is ready for a part. The scheduling methods include methods for each scheduling rule: FIFO, LIFO, EDD, SPT, Slack/OPNR and Fuzzy (the fuzzy logic part dispatching method), as well as methods which calculate and allocate the final priority level of each part.

The fuzzy logic control rules representing the equipment buffer levels are stored as an **Equipment** class method. Those rules dealing with part aspects such as inventory level, due dates, and process plan step, are stored as **PartType** class methods. Separating the various control rule sets in this manner was done in order to provide the parts and

machines with their own internal knowledge which was independent of other machines and parts and also independent of the main controlling structure. Furthermore, it allows knowledge of a machine's buffer level to reside with the machine, and knowledge of part aspects to reside with the parts. The various methods are also structured in a manner which gives each machine a certain degree of independent control over its own input buffer, and gives each part a certain degree of independent control over process and scheduling concerns. The emphasis in the current scheduling methodology is for the parts to gather information and to develop their own detailed priority levels with respect to their own internal knowledge (rule bases). Therefore methods have been included in the **PartType** class level which provide the necessary message passing and reasoning ability to accomplish this task. Overall, the structure of the methods has been designed to reinforce the concept of independent part and machine entities.

### 6.3.3 Fuzzy Logic Class Methods

As was mentioned earlier, the fuzzy logic class has no instance variables, but consists only of a series of methods. There are two groups of methods: those that perform the fuzzy logic reasoning and calculations, and those that define the fuzzy logic membership functions. The reasoning methods include: reading the rules, accessing the appropriate membership functions, developing the consequent fuzzy sets, determining the output fuzzy set and finally calculating the centroid of the output fuzzy set. The membership function methods simply define all the antecedent and consequent

membership functions.

## 6.4 SIMULATION MODEL

In order to test the control system and to demonstrate its effectiveness, a number of methods were developed to build a simple, but effective simulation model. The model is a very basic discrete event simulation incorporating a next event list for the control of time. All aspects of the simulation are deterministic (in the sense that they were chosen randomly before the simulation began versus generated randomly as the simulation was running) including part due dates, process plans, alternate process plans, breakdowns, repairs etc. The simulation methods have been inserted at the cell level and include: initializing the entire shop floor representation (cells, machines, parts, tools etc.), initializing and controlling the next event list, coordinating part arrivals, initializing machine and tool breakdowns and repairs, data collection and presentation to screen and files, fuzzy logic rule and membership function initialization, reconfigurable initialization, and initializing alternate process plans if requested. A number of instance variables have also been added to the existing instance variables of the various **Cell**, **Equipment**, **PartType**, and **Part** classes strictly for the purposes of data collection. Data is collected for machines for items such as: idle time, utilization and input buffer maximums; and for parts: number of parts completed, batch completion time, part production time, lateness, tardiness, number of late parts, mean lateness, mean tardiness, average time in buffers, and average machining time. The simulation also maintains data files with continuous

data on buffer levels, inventory levels and the part processing sequence for each machine. The purpose of this simulation was to test the control structure to see if the various new concepts worked. It was beyond the scope of this project to develop a fully blown simulation package with different part arrival mechanisms, complete stochastic representations, and statistical analysis elements. The simulation as it is currently built, provides enough fundamentals to do an analysis of the new concepts and to provide sufficient data for analysis purposes. Although the simulation is set up deterministically, in a sense it is also random because many of the elements may be modified randomly by the operator of the simulation at the start of the simulation.

The simulation offers the ability to adjust and test a number of different things. Clearly, machines, parts and tools can be added and modified. Process plans and alternate process plans can be changed. The fuzzy logic rules can be modified, new rules can be added or the membership functions can be changed. The relative rule set weights can also be adjusted either for one run or a series of runs for comparison purposes. Different input buffer constraints can be tested. Repair and breakdown events can be added, deleted, or changed. Rush priority parts can be identified which then allows simulation of a reconfigurable case. Finally, robot control and movement can be simulated.

## 6.5 OPERATION OF THE SIMULATION

The simulation begins by initializing the shop floor. All the machines, part types and tools are defined and the shop floor model is set up in the simulation. Any immediate breakdown events are initialized as are any future breakdown and repair events. A next event list is set up which contains the time when an event is complete and a description of the event. All the fuzzy logic rule bases and fuzzy data requirements are initialized and the actual simulation is set in motion by generating arrivals of parts. One part of each part type is created and placed in the input buffer of the machine which is identified in the first step of the part's process plan. It has been assumed for this project that there will always be parts waiting to be processed by the cell until an amount of parts equal to the batch size have been placed into the cell. Furthermore, whenever a part which is on its first process step is selected for processing from the input buffer of the machine, the arrival of a part of the equivalent part type is generated. Priority levels are generated for all the parts at each machine. The high priority parts are selected, loaded into the machines, and the next event list is updated. The simulation then proceeds in discrete time intervals through the next event list. Events (tasks) are performed, new parts arrive when appropriate, and the next event list is continually updated. Throughout the simulation process, breakdown events and repair events are processed as they occur. Part priority levels are calculated according to the scheduling rule that is currently in effect. If the fuzzy logic method is activated, the scheduler sends messages to the parts in the input buffer of the machine under consideration requesting the parts to generate

detailed priority levels. The parts get information from other machines regarding buffer levels, access their own process plans for process information, and also access inventory levels and due dates for scheduling information. The parts then initiate the fuzzy logic procedure and determine priority levels for the buffer limit, process plan step and the scheduling rule sets. The part then passes the priority level information to the scheduler at which time the scheduler determines a final priority level and assigns this level to the part. Finally, the machine looks through the input buffer and selects the appropriate part. Subsequent loading of the part onto the machine is controlled by the supervisor. The simulation run is completed when all the parts required to fill the batch sizes have been processed by the system. Throughout the simulation, data is gathered and stored in files and at the end of the simulation a number of statistics are generated and stored.

Chapter 6 has provided an overview of how the proposed control structure has been implemented in an object oriented environment. The object oriented environment provides a mechanism whereby parts and machines can be given certain attributes and an ability to direct their own actions. Although the parts do not control their own destinies entirely in the proposed control structure, the structure has been developed to ultimately allow this. The cell controller has also been developed in a somewhat modular form. Thus, if more cell controllers are required by the system, new instances of the **Cell** class need only to be generated. The actual structure of the controller also lends itself to being used at other levels of control. For example, a shop floor could be controlled using the proposed supervisor and a scheduler structure. In this situation the scheduler may

dispatch parts from cell to cell instead of from machine to machine. The supervisor's primary control level could then represent different cells instead of different machines, and the secondary control level could represent the detailed operations within each cell. The message passing between parts and machines could now be between part types or parts, and cells, thereby dictating the priority of a part type or part to enter a cell. Thus the same conceptual control structure could be used at different control levels. Care has also been taken to allow the proposed control structure to perform and to react to problems in real time. This will be shown in the next chapter along with additional examples which are used to demonstrate all the workings of the supervisor and scheduler. A study comparing dispatching rules and the fuzzy logic dispatching method will also be presented as well as a technique for selecting the best combination of rule set weights.

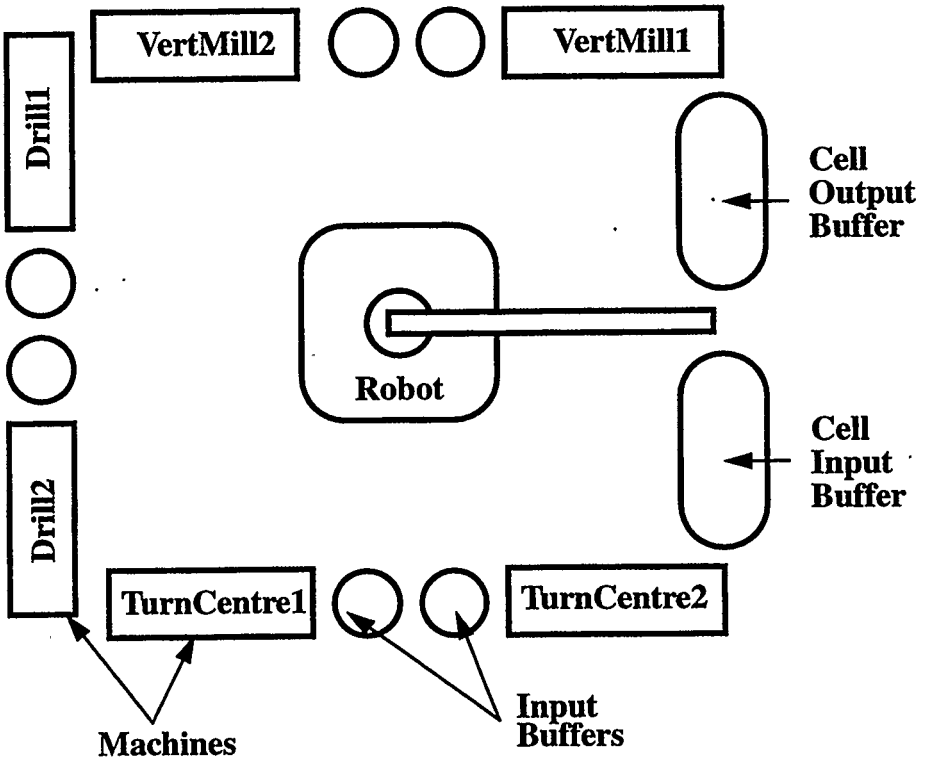
# CHAPTER 7

## 7. CASE STUDIES

## 7.1 INTRODUCTION

In order to show how the various aspects of the control structure work, a number of test cases will now be presented. All the test cases consider the control and scheduling of a manufacturing cell similar to the one shown in Figure 7.1 which consists of six machines serviced by a material handling system (represented here by a robot). Each machine has its own finite capacity input buffer and the cell has one input buffer and one output buffer. The part movement and machine loading times are included in the total processing time in all but one of the case studies. In one case study the full movement and loading times are presented and the robots actions are fully documented. Certain assumptions were made to restrict the scope of the overall project to a manageable level and these assumptions were outlined in chapter 1. The following conditions were also

Figure 7.1: Manufacturing Cell



applied during the case studies:

1. each part type has its own process plan
2. part routings and processing times are deterministic and are given by the part type process plan
3. each operation in the process plan must be done in the sequence given in the process plan
4. each operation must be completed before the next operation can begin
5. each machine has a finite buffer limit
6. due dates and part requirements are available from a master schedule and are fixed
7. due dates are different for each part type while part batch sizes are set at ten parts
8. each machine has one alternate machine
9. machine and tool breakdowns will be considered and parts rerouted to alternate machines
10. blocking and deadlocking will result in error messages only
11. no pre-emption is allowed
12. each machine can perform only one operation at a time
13. each part can be processed by only one machine at a time
14. the part is considered to have completed processing when a tool or machine breakdown occurs

15. the part is removed after each operation is completed even if the next operation is on the same machine
16. 'fuzzy logic optimized' implies a certain degree of optimization or improvement, not global optimization
17. each machine has the appropriate tooling

Eight different case studies will be presented which will emphasize the various aspects of the fuzzy logic dispatching method. The cases will also show how the various functions of the supervisor operate in conjunction with the fuzzy logic dispatching method. Throughout this study, there were two primary objectives: 1) to minimize or eliminate late parts, and 2) to control and minimize buffer levels in order to avoid or reduce deadlocking situations. The rule sets were developed with these two operations in mind; however, data was collected for other possible objectives such as minimizing maximum lateness, mean lateness, or tardiness etc. as discussed in chapter 6. The results of the additional data will also be presented and discussed even though the rule base has not been specifically developed for these additional objectives. Case 1 will show generally how the control structure works and presents some typical types of output. Case 2 represents a run using alternate part process plans. Case 3 presents two large size cases: one with six machines and twenty part types (a total of 200 parts and 800 operations), and one with ten machines and twenty part types (200 parts, 800 operations). Case 4 summarizes the results of a finite buffer limit study and shows the effect that varying the finite buffer limit has on the various dispatching rules and the fuzzy logic

method for a number of objectives. Case 5 presents an example of a machine breakdown and repair. Case 6 presents a reconfigurable case. Case 7 is an example which includes part movement time and also provides a description of the robot's activities. Finally, case 8 presents a methodology for selecting the best rule set weights.

## 7.2 CASE 1: FUZZY LOGIC BASE CASE

The intent of this case study is to show that the fuzzy logic method works and to provide some base line data for succeeding cases. For the remainder of this chapter, case 1 will represent the 'base case' and models the following situation:

- six machines as shown in Figure 7.1
- part movement is assumed to be included in processing times
- material handling is not considered
- ten part types, each having a batch size of ten parts
- each part's process plan consists of four operations
- there are no breakdowns
- finite buffer limits for each machine are set at ten parts (a maximum of ten parts allowed in the machine's input buffer)
- there are no priority parts
- no alternate process plans are used
- the fuzzy logic rule set weights have been optimized to 1:3:1 for buffer,

scheduler, and process step rules respectively

Figure 7.2 shows a typical screen view presented by the simulation; this particular view shows events at a simulation time of 32 minutes. The screen view shows the operation or process which is currently under way in each machine and also indicates which parts are waiting in each machine's input buffer. The screen view is updated whenever a new event occurs. This is shown in Figure 7.3 where the simulation time has been updated to 33 minutes. The event that caused the update was the completion of the processing of partB2 on VertMill2. As soon as partB2 completed processing it was placed in the input buffer of Drill1 which is the next machine in partB2's process plan. The removal of partB2 also triggered the fuzzy logic scheduling method described earlier, which then generated priority levels for all the parts in the input buffer of VertMill2. PartB3 had the highest priority and was therefore loaded into VertMill2 as shown in the figure.

A variety of data is collected during each simulation run and this data is summarized in Table 7.1 (machine data) and Table 7.2 (part data) for an optimized fuzzy logic run. As seen from the tables, part type A only had one part which was late, and all the machines had input buffer maximum levels less than the finite buffer limit of ten parts. A small study was also done which compared the optimized fuzzy logic method with the non-optimized fuzzy logic method and two dispatching rules, LIFO and FIFO. The results are presented in Table 7.3. Here, the maximum buffer size represents the

Figure 7.2: Screen View of Machine Process and Buffer Status at SimTime = 32 Minutes

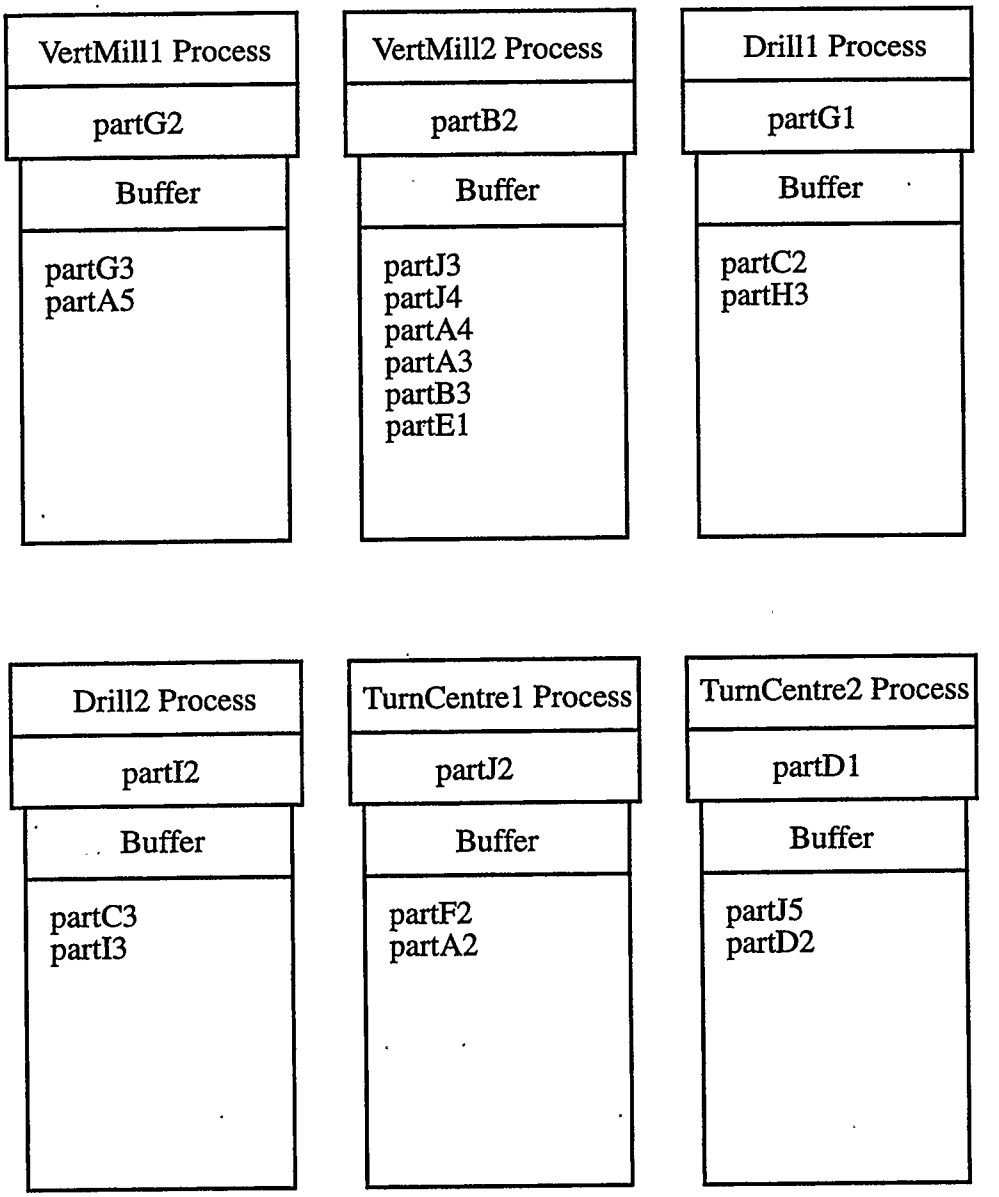
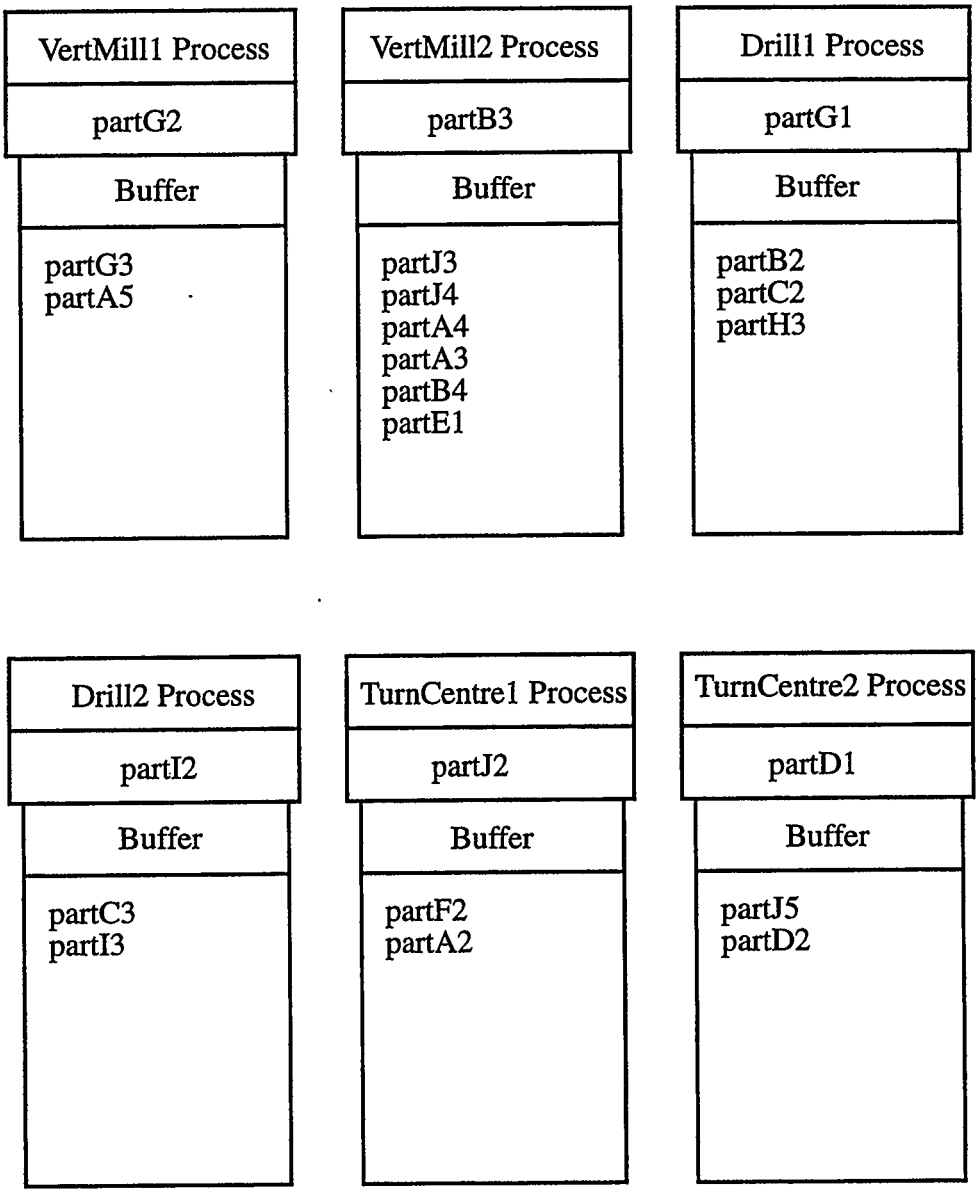


Figure 7.3: Screen View of Machine Process and Buffer Status at SimTime = 33 Minutes



maximum of all the machine input buffer maximums shown in Table 7.1. All the parts of all the part types are considered when calculating mean tardiness given tardy, average production time, and maximum lateness and all the machines are considered when determining maximum and minimum machine utilization. Table 7.3 clearly shows the improved performance of the optimized fuzzy logic method over the other three selection rules in terms of the number of late parts. Upon reviewing a number of test cases, several trends have become apparent. First, the fuzzy logic method generally maintains

Table 7.1: Base Case Machine Data

| Machine | Input Buffer Maximum | Idle Time (minutes) | Machine Utilization (%) |
|---|---|---|---|
| VertMill1 | 7 | 13 | 96.0 |
| VertMill2 | 9 | 73 | 77.4 |
| Drill1 | 9 | 13 | 96.0 |
| Drill2 | 8 | 63 | 80.5 |
| TurnCentre1 | 5 | 23 | 92.9 |
| TurnCentre2 | 4 | 103 | 68.1 |

a lower buffer level in more machines than the other three methods. Second, FIFO and LIFO very often have a number of machines with input buffers filled to capacity whereas the fuzzy logic method will only have one or two machines at the maximum finite buffer limit. Third, the optimized fuzzy logic method tends to have higher buffer maximums (therefore fuller buffers) than the fuzzy logic method, which is likely a result of the overall higher average production time. An explanation for the overall higher average

Table 7.2: Base Case Part Data

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| A | 240 | 255 | 15 | 1 | 15 | 73.9 |
| B | 200 | 200 | 0 | 0 | 0 | 48.1 |
| C | 300 | 245 | 0 | 0 | 0 | 69.8 |
| D | 300 | 270 | 0 | 0 | 0 | 92.9 |
| E | 240 | 236 | 0 | 0 | 0 | 54.7 |
| F | 320 | 286 | 0 | 0 | 0 | 104.7 |
| G | 320 | 319 | 0 | 0 | 0 | 68.2 |
| H | 340 | 323 | 0 | 0 | 0 | 52.0 |
| I | 260 | 238 | 0 | 0 | 0 | 49.3 |
| J | 240 | 159 | 0 | 0 | 0 | 40.8 |

production time for the optimized fuzzy logic method is as follows. The rule set for process plan step tends to help move the parts through the system and when the weight of the process step rule set is reduced relative to the schedule rule set (by increasing the schedule rule set weight) the parts do not move as quickly through the system resulting in a longer production time.

Table 7.3: Base Case Rule Comparison

| | Selection Rule | | | |
|---|---|---|---|---|
| | Fuzzy | Optimized Fuzzy | FIFO | LIFO |
| No. of Tardy Parts | 4 | 1 | 23 | 11 |
| Mean Tardiness Given Tardy (minutes) | 37.5 | 15 | 40.3 | 34.6 |
| Average Production Time (minutes) | 60.2 | 65.4 | 124.4 | 59.5 |
| Maximum Lateness (minutes) | 62 | 15 | 74 | 70 |
| Maximum Buffer Size (units) | 9 | 9 | 10 | 10 |
| Maximum Machine Utilization (%) | 96 | 97.2 | 98.1 | 97.2 |
| Minimum Machine Utilization (%) | 68.1 | 69 | 69.6 | 69 |

Since the base case fuzzy logic method is optimized by increasing the scheduler rule set weight from one to three (in the base case fuzzy logic method all weights are set at one), it follows that the average part production time will be greater for the optimized

fuzzy dispatching case. Further comparisons between optimized fuzzy logic, non-optimized fuzzy logic and five dispatching rules will be presented in case 4.

Whereas Table 7.1 and 7.2 provide a simulation summary, Figure 7.4 and Figure 7.5 provide an ongoing view of four part type inventory levels and one machine's buffer level. Figure 7.4 clearly shows how each part type is given priority according to its due date. Part type J has the earliest due date of 240 minutes (Table 7.2) and therefore is processed in advance of other part types such as part type H which has a later due date of 340 minutes. Although not all ten part types are shown, it can be seen that as a part type approaches its due date, the urgency to place uncompleted parts in inventory increases and therefore the part type's priority increases.. For example, part type D has a due date of 300 minutes but at a time of 245 minutes only six of the required ten parts had been completed. This increased part type D's priority level resulting in the quick completion of the remaining four parts instead of allowing the parts to remain idle in a machine input buffer. With reference to the completion times and due dates shown in Table 7.2, it is apparent that the fuzzy logic scheduling method selects the appropriate part consistently, and in this particular optimized case, only one part is late.

One of the primary goals of the fuzzy logic method was to keep buffer levels reasonable. VertMill2 was one of the machines that had a high buffer maximum (nine parts); however, as one can see from Figure 7.5, this maximum was reached for only a short time frame early on in the simulation run. The machine buffer level fluctuates

Figure 7.4: Inventory Levels

Part Types D, H, I, and J

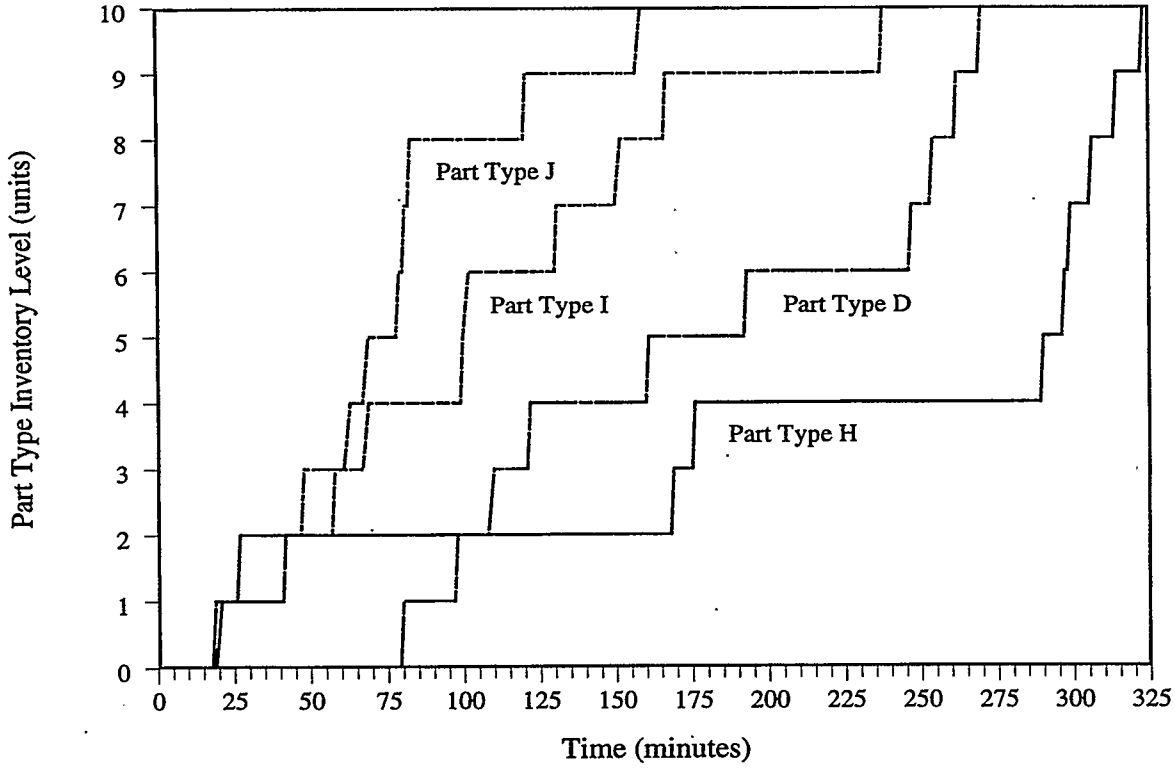
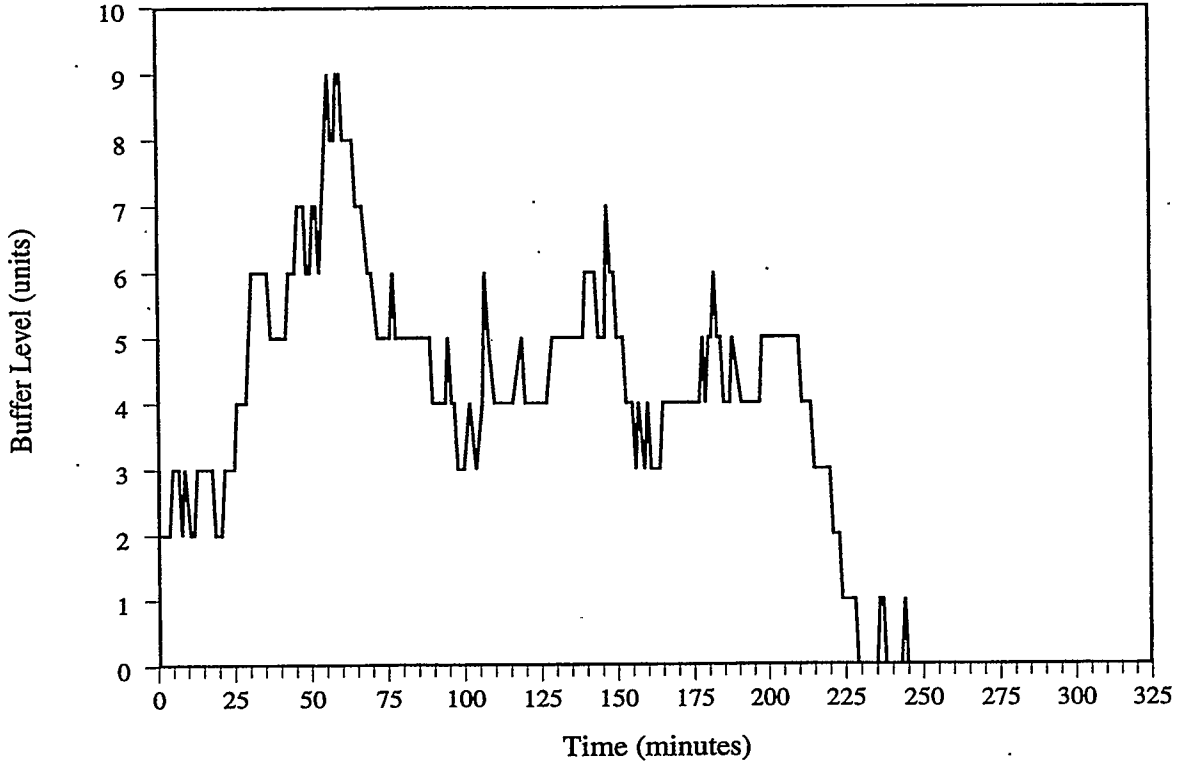


Figure 7.5: Buffer Level

VertMill2



continuously, but on the whole the buffer level of VertMill2 was maintained at an average of about three parts. Furthermore, judging by the buffer levels of other machines and cases (not shown) the buffer level maximums seem to vary both in magnitude and in location depending upon the part mix and part process plans. Further on in the chapter it will be shown that the fuzzy logic method is quite effective in controlling buffer levels and avoiding deadlock, especially with respect to some common dispatching rules.

Overall, case 1 shows generally what is occurring throughout a simulation run and also shows the various methods and types of data capture and presentation. Case 1 presents the base results to which other cases will be compared. It should be noted that although the eight cases presented in this thesis represent only a small fraction of the total test cases, they are representative of the functions that the control system can perform and representative of the overall trends.

## 7.3 CASE 2: ALTERNATE PROCESS PLAN

Each part type has been provided with one alternate process plan in order to increase the flexibility in the manufacture of the part and to provide a method for improving part flow throughout the shop floor. Case 2 uses all the alternate process plans for each part type to generate an optimized fuzzy logic comparison case to the base case. Tables 7.4 and 7.5 present the part type and machine data for the alternate process plan case. These tables show that four parts were late and that all the machines except for

VertMill1 had buffer maximums less than the finite buffer capacity limit of ten parts. Table 7.6 also shows a comparison of the optimized fuzzy logic method with the non-optimized fuzzy logic method and two dispatching rules, LIFO and FIFO. In this particular case, the finite buffer limit of ten parts was too small for the dispatching rule FIFO and therefore the system deadlocked. Other than that, the trends described in the base case held for this case. The optimized fuzzy logic method had the least number of late parts and the smallest maximum lateness value, but as before, it had a longer average production time than the non-optimized fuzzy logic method.

Table 7.4: Alternate Process Plan Case Machine Data

| Machine | Input Buffer Maximum | Idle Time (minutes) | Machine Utilization (%) |
|---|---|---|---|
| VertMill1 | 10 | 6 | 98.2 |
| VertMill2 | 7 | 76 | 76.7 |
| Drill1 | 7 | 66 | 79.8 |
| Drill2 | 8 | 6 | 98.2 |
| TurnCentre1 | 8 | 46 | 85.9 |
| TurnCentre2 | 4 | 106 | 67.5 |

Table 7.5: Alternate Process Plan Case Part Data

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| A | 240 | 218 | 0 | 0 | 0 | 54.0 |
| B | 200 | 220 | 20 | 1 | 20 | 49.4 |
| C | 300 | 230 | 0 | 0 | 0 | 65.1 |
| D | 300 | 265 | 0 | 0 | 0 | 110.0 |
| E | 240 | 254 | 14 | 2 | 8 | 54.9 |
| F | 320 | 282 | 0 | 0 | 0 | 67.3 |
| G | 320 | 306 | 0 | 0 | 0 | 60.5 |
| H | 340 | 326 | 0 | 0 | 0 | 139.8 |
| I | 260 | 261 | 1 | 1 | 1 | 63.8 |
| J | 240 | 188 | 0 | 0 | 0 | 66.5 |

## 7.4 CASE 3: LARGE CASES

Two large cases will be presented here with the intention of showing that the fuzzy logic method is capable of dealing with many machines and many parts. Both cases consist of non-optimized fuzzy logic method simulation runs having a finite buffer limit set at ten parts. The first case, Large Case 1, consists of the same six machines as the base case but now twenty part types of ten parts each are being processed. Table 7.7 shows that although the six machines are able to handle the additional load, the input buffer maximums have increased substantially. Never-the-less, the fuzzy logic method is able to maintain production with the result that only 18 of the 200 parts were late (Table 7.8).

The second case, Large Case 2, consists of ten machines as shown in Table 7.9, and twenty part types. No real effort was made to set up the part type process plans in a way which balanced the loading on the machines and this is made evident by some very low machine utilizations. Again, the intent was simply to show that the fuzzy logic method worked and was capable of dealing with many machines and many parts. Table 7.10 shows some part data and it is clear from the number of tardy parts (45) that significant optimization for this case is required both in terms of optimizing the fuzzy logic method and also in balancing the machine loading to increase machine utilization and reduce input buffer maximums. Furthermore, there may be one or two bottleneck machines causing large maximum lateness values of a number of parts types especially

Table 7.6: Alternate Process Plan Case Rule Comparison

| | Selection Rule | | | |
|---|---|---|---|---|
| | Fuzzy | Optimized Fuzzy | FIFO | LIFO |
| No. of Tardy Parts | 14 | 4 | | 11 |
| Mean Tardiness Given Tardy (minutes) | 5.3 | 7.3 | D E | 5.3 |
| Average Production Time (minutes) | 58.8 | 73.1 | A D | 56.5 |
| Maximum Lateness (minutes) | 37 | 20 | L O | 34 |
| Maximum Buffer Size (units) | 8 | 10 | C K | 10 |
| Maximum Machine Utilization (%) | 97.3 | 98.2 | E D | 97.3 |
| Minimum Machine Utilization (%) | 66.9 | 67.5 | | 66.9 |

Table 7.7: Large Case 1 Machine Data

| Machine | Input Buffer Maximum | Idle Time (minutes) | Machine Utilization (%) |
|---|---|---|---|
| VertMill1 | 10 | 23 | 96.4 |
| VertMill2 | 9 | 143 | 77.8 |
| Drill1 | 10 | 23 | 96.4 |
| Drill2 | 10 | 123 | 80.9 |
| TurnCentre1 | 7 | 43 | 93.3 |
| TurnCentre2 | 6 | 203 | 68.4 |

Table 7.8 Large Case 1 Part Data

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT* (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| A | 480 | 510 | 30 | 4 | 18 | 88.7 |
| B | 400 | 342 | 0 | 0 | 0 | 57.2 |
| C | 600 | 538 | 0 | 0 | 0 | 131.9 |
| D | 600 | 536 | 0 | 0 | 0 | 112.6 |
| E | 480 | 504 | 28 | 5 | 19.2 | 121.5 |
| F | 640 | 590 | 0 | 0 | 0 | 199.4 |
| G | 640 | 643 | 3 | 1 | 3 | 98.3 |
| H | 640 | 421 | 0 | 0 | 0 | 61.4 |
| I | 520 | 573 | 53 | 1 | 53 | 78.7 |
| J | 480 | 305 | 0 | 0 | 0 | 50.6 |

* MTGT - Mean Tardiness Given Tardy

Table 7.8 Large Case 1 Part Data (continued)

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT* (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| K | 480 | 518 | 38 | 1 | 38 | 82.5 |
| L | 400 | 210 | 0 | 0 | 0 | 50.5 |
| M | 600 | 377 | 0 | 0 | 0 | 94.1 |
| N | 600 | 318 | 0 | 0 | 0 | 71.3 |
| O | 480 | 404 | 0 | 0 | 0 | 69.8 |
| P | 640 | 430 | 0 | 0 | 0 | 116.5 |
| Q | 640 | 615 | 0 | 0 | 0 | 110.9 |
| R | 680 | 540 | 0 | 0 | 0 | 99.1 |
| S | 520 | 568 | 48 | 6 | 34.8 | 139.4 |
| T | 480 | 247 | 0 | 0 | 0 | 61.8 |

* MTGT - Mean Tardiness Given Tardy

Table 7.9: Large Case 2 Machine Data

| Machine | Input Buffer Maximum | Idle Time (minutes) | Machine Utilization (%) |
|---|---|---|---|
| VertMill1 | 5 | 112 | 74.7 |
| VertMill2 | 10 | 112 | 74.7 |
| VertMill3 | 6 | 152 | 65.6 |
| HorMill1 | 3 | 292 | 33.9 |
| HorMill2 | 6 | 122 | 72.4 |
| Drill1 | 8 | 52 | 88.2 |
| Drill2 | 10 | 42 | 90.5 |
| Drill3 | 10 | 72 | 83.7 |
| TurnCentre1 | 10 | 2 | 99.5 |
| TurnCentre2 | 5 | 162 | 63.3 |

Table 7.10: Large Case 2 Part Data

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| A | 240 | 246 | 6 | 1 | 6 | 94.5 |
| B | 200 | 179 | 0 | 0 | 0 | 52.5 |
| C | 300 | 428 | 128 | 9 | 67 | 74.6 |
| D | 300 | 296 | 0 | 0 | 0 | 81.4 |
| E | 240 | 225 | 0 | 0 | 0 | 52.2 |
| F | 320 | 442 | 122 | 8 | 78.5 | 78.1 |
| G | 320 | 311 | 0 | 0 | 0 | 116.4 |
| H | 320 | 302 | 0 | 0 | 0 | 58.2 |
| I | 260 | 406 | 146 | 7 | 112.3 | 59.0 |
| J | 240 | 217 | 0 | 0 | 0 | 50.7 |

Table 7.10: Large Case 2 Part Data (continued)

| Part Type | Due Date (min) | Completion Time (min) | Maximum Lateness | Number of Tardy Parts | MTGT (min) | Average Prod'n Time (min) |
|---|---|---|---|---|---|---|
| K | 240 | 307 | 67 | 1 | 67 | 70.2 |
| L | 200 | 262 | 62 | 4 | 52 | 64.4 |
| M | 300 | 294 | 0 | 0 | 0 | 39.41 |
| N | 300 | 292 | 0 | 0 | 0 | 62.7 |
| O | 240 | 228 | 0 | 0 | 0 | 69.1 |
| P | 320 | 358 | 38 | 2 | 22.5 | 85.2 |
| Q | 310 | 178 | 0 | 0 | 0 | 54.6 |
| R | 340 | 325 | 0 | 0 | 0 | 84.9 |
| S | 260 | 385 | 125 | 6 | 102.7 | 68.5 |
| T | 240 | 281 | 41 | 7 | 24.7 | 118.8 |

part type C, part type F, part type I and part type S.

## 7.5 CASE 4: FINITE BUFFER LIMIT STUDY

The first three case studies dealt with situations where the finite buffer limit was set at ten parts. As the finite buffer limit is increased or decreased from this value it is very possible that the part flow through the system will change resulting in either poorer or improved performance measures. In order to test the effect of the finite buffer limit, a study was conducted where the finite buffer limit was varied from a low of three parts to a high of twenty four parts. A number of performance measures were monitored and include: the number of late parts, maximum lateness, average tardiness given tardy, average production time and average machine maximum buffer levels. Also within the study, a comparison was made between the performance of the fuzzy logic method and the performance of five common dispatching rules; namely, FIFO, LIFO, EDD, SPT, and Slack/OPNR. The base case attributes were all used except for the finite buffer limit.

The results are shown in Figures 7.6, 7.7, 7.8, 7.9, and 7.10. Figure 7.6 depicts the affect that changing the finite buffer limit has on the total number of late parts. There are several things to note regarding the output. First, many of the dispatching rules caused the simulation to deadlock. The lowest shown value of finite buffer limit for a given rule represents the minimum buffer size that was required to prevent deadlock. None of the rules including the fuzzy and fuzzy optimized methods prevented deadlock

Figure 7.6: Comparison of Number of Late Parts

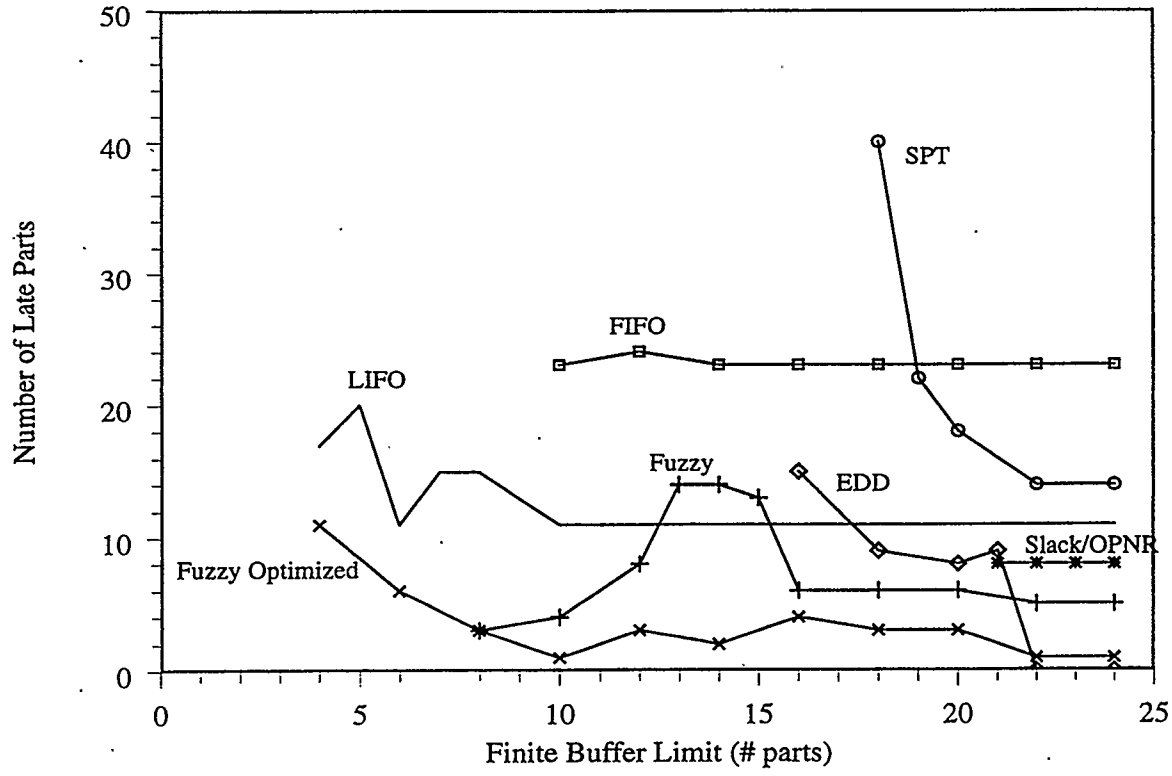


Figure 7.7: Comparison of Maximum Part Lateness

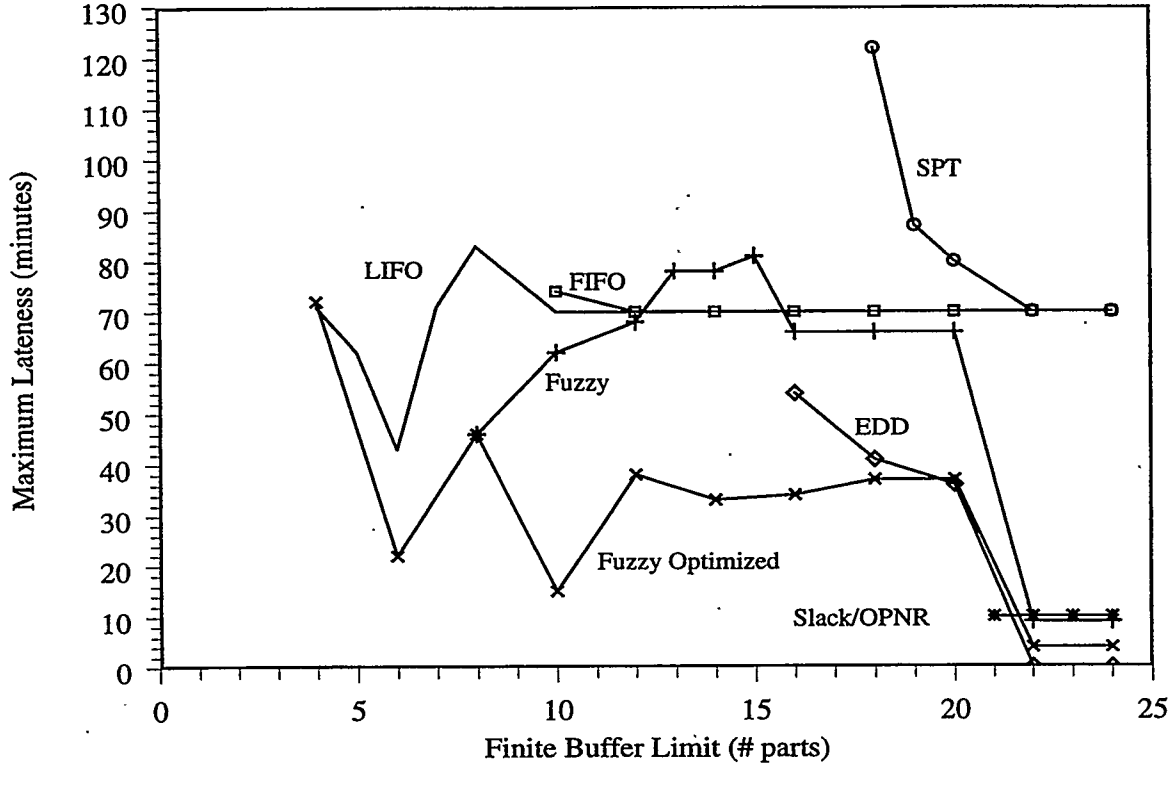


Figure 7.8: Comparison of Average Tardiness

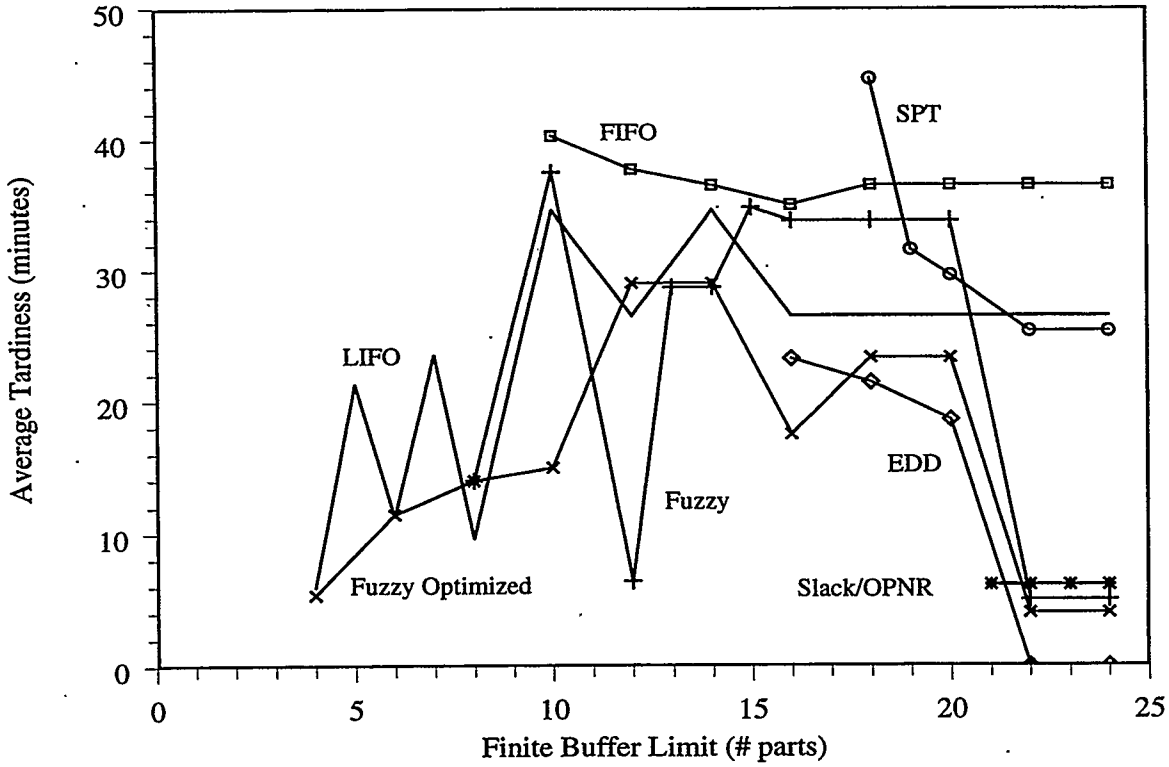


Figure 7.9: Comparison of Average Production Times

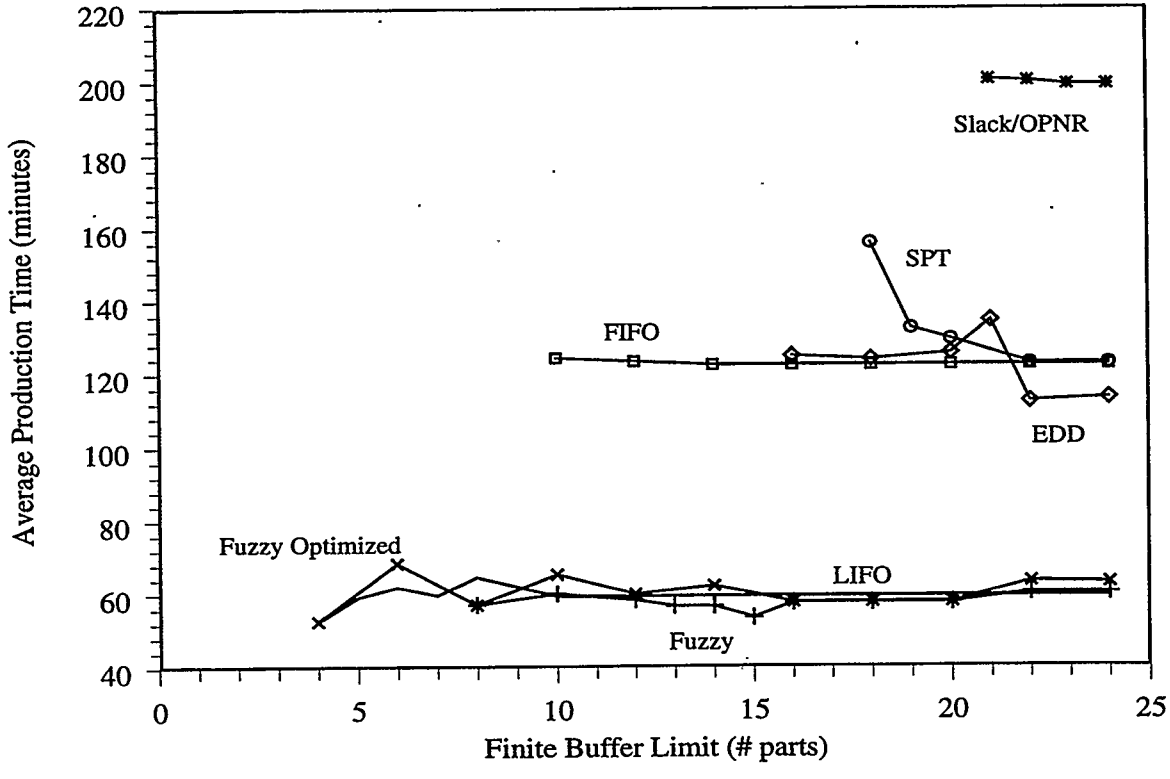
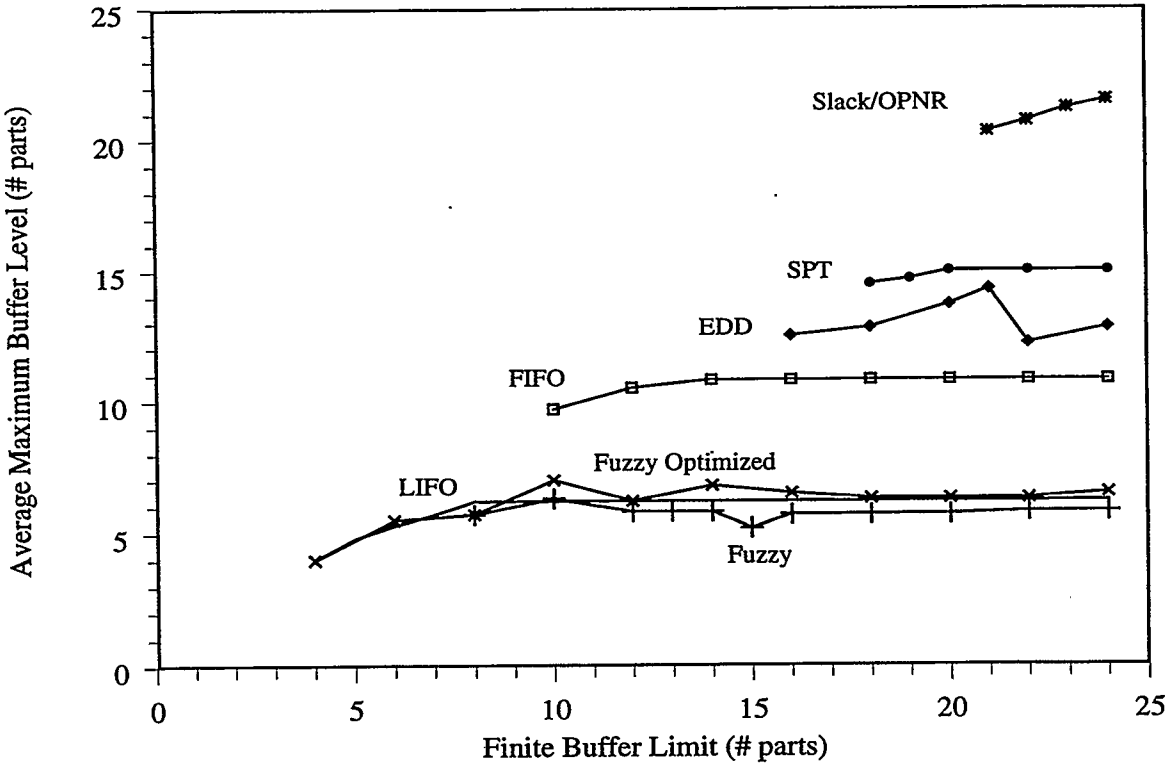


Figure 7.10: Comparison of Average Maximum Machine Buffer Levels



at finite buffer limits less than four parts. For the particular mixture of part types and process plans it can be seen that the optimized fuzzy logic method and LIFO performed equally well with respect to preventing deadlock and that Slack/OPNR performed the poorest. Second, it is clear from Figure 7.6 that fuzzy optimized performed better (fewer late parts) than all other dispatching rules over a large range of finite buffer limits. As expected, EDD performed the best as the finite buffer limit size was increased greatly. It should also be noted that for certain rules (LIFO, FIFO and Slack/OPNR) increasing the finite buffer limit above a certain point does not provide a decrease in the number of late parts. Third, it is also apparent from this case, and from some of the data presented earlier, that the fuzzy logic method requires optimization or tuning to consistently out-perform the other rules.

Minimizing part lateness was one of the primary goals of the current fuzzy rule sets; however, data was also gathered for the performance measures of maximum lateness, average production time, and average tardiness given tardy. Although no effort was made to refine the rule structure to improve the results based on these three performance measures, it will still be useful to see how the fuzzy logic method fares against the other rules. Figure 7.7 shows how maximum part lateness varies for each rule across the range of finite buffer limits. It can be seen that the optimized fuzzy logic method fares better (has lower maximum part lateness) than most of the other rules over a wide range of finite buffer limits. The same cannot be said for average tardiness given tardy. Given the results shown in Figure 7.8, it is not possible to say that the fuzzy optimized method

is necessarily better or worse than the other rules; in fact, in some instances the non-optimized fuzzy method performs better than the optimized fuzzy method. Clearly, for the fuzzy method to give improved results for average tardiness given tardy, changes or additions to the rule sets are required. The situation is somewhat better for the average production time performance measure shown in Figure 7.9. At least here, fuzzy optimized competes favourably with fuzzy and LIFO to provide results consistently and dramatically better than the remaining four rules. When all four performance measures are taken into consideration the optimized fuzzy logic method seems to provide overall better performance than the remaining dispatching rules, in spite of the fact that the rule sets have not been specifically developed to take all the performance measures into account.

One final goal of the current fuzzy rule sets was to minimize the machine buffer levels. Figure 7.10 looks at the average of the maximum buffer levels of each machine for the various rules over a range of finite buffer limits. Thus, a low average will indicate that the peak loadings within each of the machines' buffer was low and provides some indication of the ability of the particular dispatching rule (or method) to control buffer loading. As seen in Figure 7.10 the fuzzy and fuzzy optimized methods compare well with LIFO and show a better ability to control maximum buffer levels than the remaining four rules.

## 7.6 CASE 5: MACHINE BREAKDOWN AND REPAIR

One of the overall goals of the control structure was to deal with disruptions and breakdowns on the manufacturing floor on a real time basis. Case 5 presents a base case situation, without optimization, having breakdown and repair events. The machine Drill1 begins the simulation run broken, is repaired after 30 minutes, breaks down after 60 minutes and finally is repaired again after 130 minutes. During the time periods that Drill1 was broken, the supervisor rerouted the parts to an alternate machine, Drill2, where they awaited scheduling in the normal manner by the fuzzy logic scheduling sub-module. Figure 7.11 shows the buffer level of Drill1 throughout the simulation and one can clearly see the difference between a normal case (no breakdown or repair events) and the broken case. When the breakdown event at 60 minutes occurs, there are two parts left in the input buffer of Drill1. The supervisor coordinates removal of these parts and by 66 minutes the parts have been removed and placed in the buffer of the alternate machine Drill2 causing its buffer to fill completely to the finite buffer limit of ten parts (Figure 7.12). All subsequent parts destined for processing on Drill1 are rerouted to Drill2 until Drill1 is repaired again at a time of 130 minutes. Figure 7.12 shows the effect that the rerouting of parts from Drill1 has on Drill2. The buffer levels of Drill2 are consistently higher during the Drill1 broken case than during a case were Drill1 is functioning normally. Drill2 is often overloaded with a buffer that is completely full, yet the fuzzy logic scheduler is able to maintain production, and as shown in Figure 7.13, continues to try and reduce late parts. Part I has a due date of 260 minutes and Part H has a due date

Figure 7.11: Buffer Levels of Drill1

Normal and Broken Cases

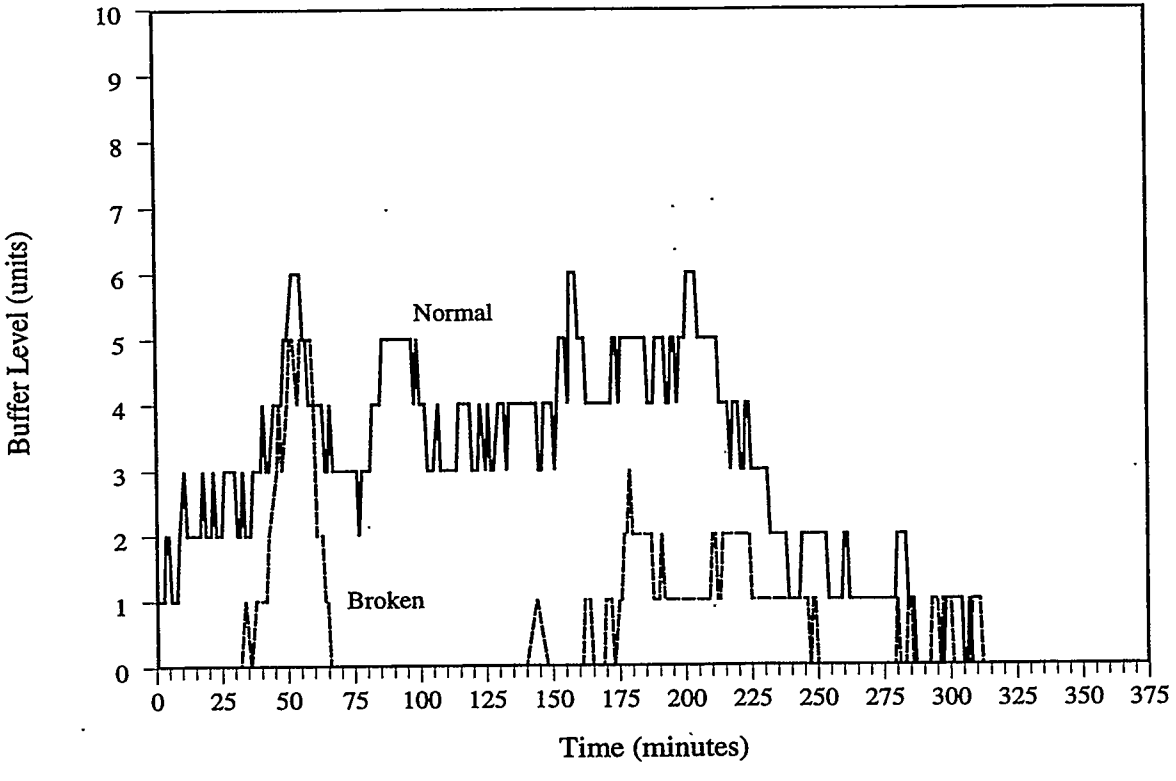


Figure 7.12: Buffer Levels of Drill2

Drill1 Normal and Broken Cases

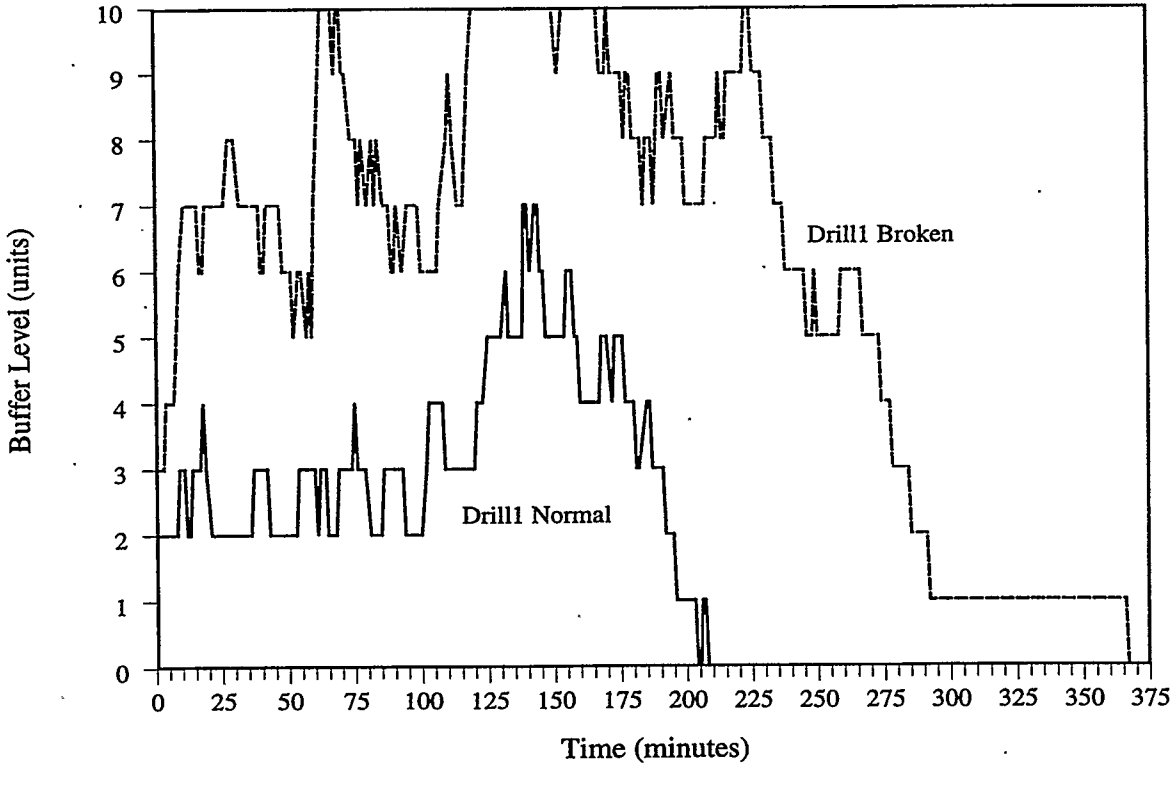
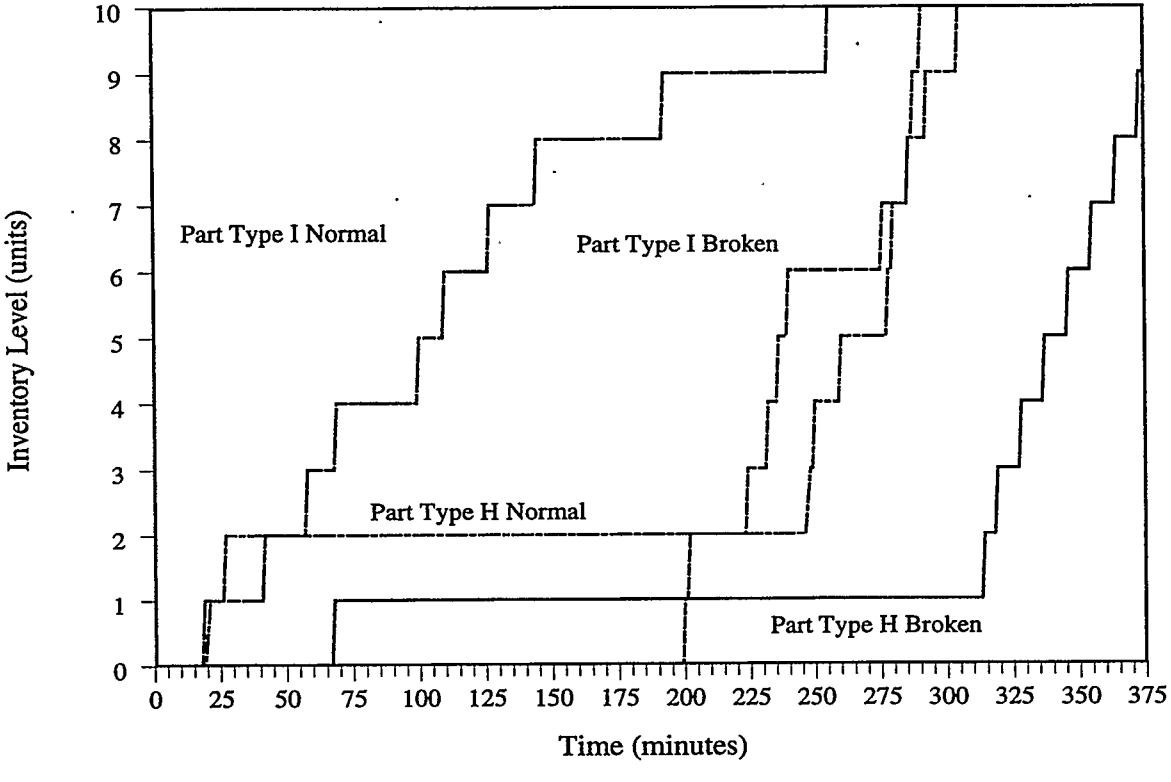


Figure 7.13: Inventory Levels of Part Type H and I
(Drill1 Normal and Broken)



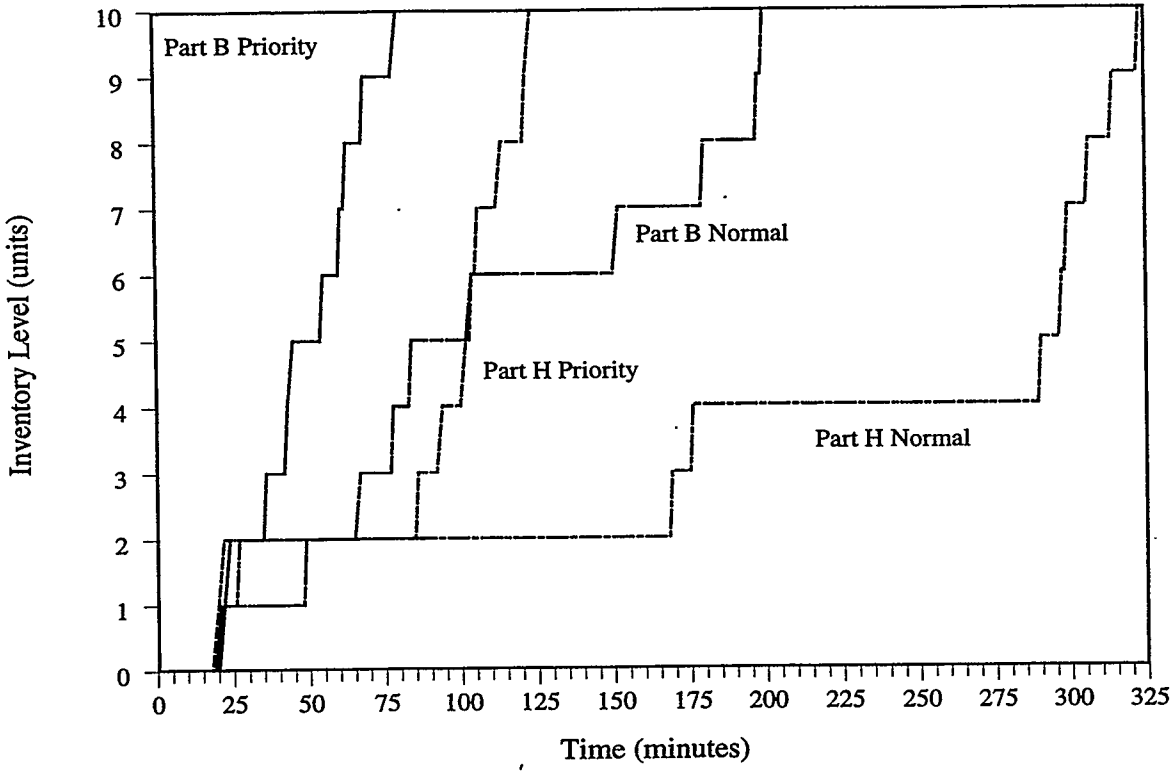
of 340 minutes, both of which were met in the non-broken case. Part I and H each have one operation that needs to be done on Drill1 and therefore a breakdown of Drill1 causes production of these parts to slow down. As a result, Part I has four parts which are late by a maximum of 30 minutes and Part H has five parts which are late by a maximum of 25 minutes. As was mentioned in the introduction, the emphasis of this project was to develop a procedure which could deal with breakdowns in a manner which would keep part production going versus providing an optimum rescheduling/rerouting mechanism. It is clear from the Case 5 example that the current supervisor and fuzzy logic scheduling structure can do that and one future step may be to incorporate an optimizing procedure to help minimize late parts during a breakdown scenario.

## 7.7 CASE 6: RECONFIGURABLE CASE

As discussed in chapter 5, the control structure has been developed to allow the representation of virtual reconfiguration in the situation where special rush orders need to be processed. Case 6 presents a non-optimized fuzzy logic base case where two part types: part type H and part type B, have special rush priority. As a result of the rush priority, the supervisor adjusts the current cell structure into a virtual structure which allows the special parts to be processed as quickly as possible. This is done by consistently ensuring that the special rush orders have the highest priority over all other parts within the system. The results of the reconfiguration can be seen in Figure 7.14. Part type B which would normally complete processing at 200 minutes, was finished in

Figure 7.14: Inventory Levels-Reconfigurable Case

Parts B and H Have Priority



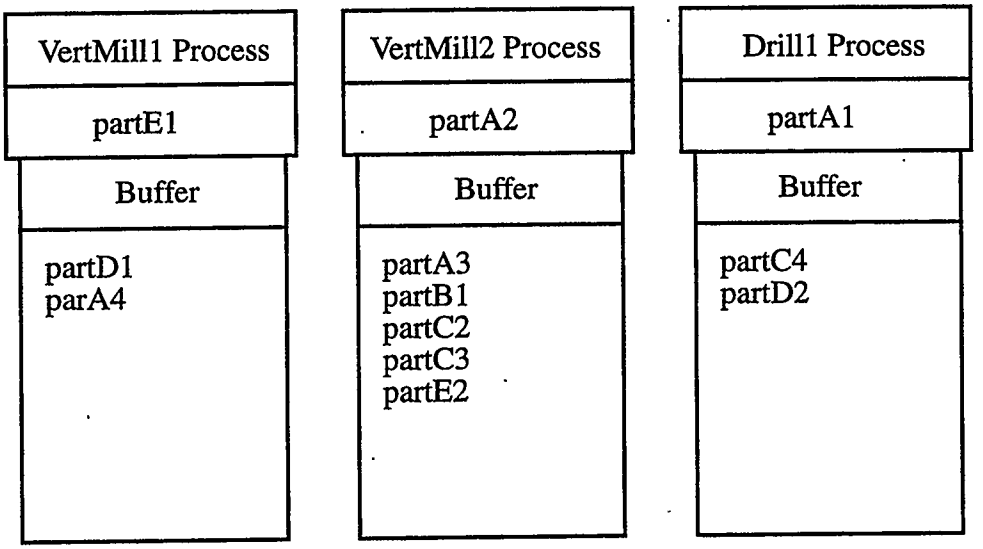
about 85 minutes. Similarly, part type H finished in about 125 minutes, or about 200 minutes earlier than in the non-reconfigurable case. Thus it can be seen that reconfiguring the system in this manner has very positive results and that the control and scheduling structure is capable of operating with the reconfigurability aspect in place.

## 7.8 CASE 7: ROBOT MOVEMENT

All the cases discussed so far have assumed that the part movement times have been incorporated into the processing times and that sufficient material handling resources exist to prevent any parts from waiting. Case 7, on the other hand, considers movement times and also utilizes the services of a robot to move the parts. The test case consists of three machines: VertMill1, VertMill2, and Drill1, and one part handling robot. Five part types (A,B,C,D, and E) are considered, each having a batch size of ten parts. The non-optimized fuzzy logic method is used and no machine or tool breakdowns are considered. Furthermore, each movement of the robot is assumed to take one minute and the robot has only a single manipulator which can handle only one part at a time.

The robot is under the control of the supervisor which activates the robot and indicates the action required. Figure 7.15 shows a screen view of the simulation at a SimTime of 66 minutes. The screen view shows a short chronology of the robot's actions over the previous 16 minutes. Generally speaking, the robot appears to be fast enough that the parts do not often have to wait. One example of a part waiting occurs at a

Figure 7.15: Screen View of Machine Process, Buffer Status and Robot Action at SimTime = 66 Minutes




SimTime Robot Action

| SimTime |
| --- |
| 66 |

50 partE1 loaded into VertMill2
51 partD1 unloaded from Drill1 into the input buffer of Drill1
(machining completed at time 49)
52 partD1 loaded into Drill1
53 partA1 unloaded from VertMill1 into the input buffer of Drill1
(machining completed at time 52)
54 partA3 loaded into VertMill1
55 partA4 loaded into the input buffer of VertMill1
56 partE1 unloaded from VertMill2 into the input buffer of VertMill1
(machining completed at time 55)
57 partA2 loaded into VertMill2
63 partD1 unloaded from Drill1 into the input buffer of VertMill1
(machining completed at time 62)
64 partA1 loaded into Drill1
65 partA3 unloaded from VertMill1 into input buffer of VertMill2
(machining completed at time 64)
66 partE1 loaded into VertMill1

SimTime of 51 minutes. At 51 minutes the robot completes unloading partD1 from Drill1; however, partD1 completed processing at SimTime 49 minutes and therefore has been waiting on machine Drill1 for one minute resulting in extra machine idle time. Only three machines were modelled in this case study because it was found that with the current part and process plan mix, any more machines resulted in excessive machine idle times due to parts waiting for the robot. Therefore more machines would require a second robot, the inclusion of which is beyond the scope of this project. This small example does however show that the control structure is capable of incorporating and controlling a material handling system.

## 7.9 CASE 8: WEIGHT OPTIMIZATION

Throughout this thesis it has been shown that the fuzzy logic dispatching method requires a certain degree of optimization before it begins to outperform existing dispatching rules. The optimization procedure currently in use involves applying a weight to each of the process step, schedule and buffer level rule sets. This weight is then used to develop an overall priority level for a part. Case 8 will discuss how a weight can be chosen in an effort to provide more optimum results. All the examples run in this section conform to the base case with the exception that the weights are modified. Three separate groups of simulations were run. In the first, the buffer level weight was set at one and both the process step and scheduler weights were incrementally varied in unit steps from one to ten resulting in a set of 100 cases. In the second group, the scheduler weight was

set at one while the buffer level and process step weights were varied in unit increments from one to ten. Similarly, in the third group, the process step weights were set at one and the other two varied. This resulted in three groups of data, each consisting of 100 simulation runs. Of primary importance was minimizing the performance measure related to the number of late parts and therefore the number of late parts was plotted on three dimensional graphs for the 300 cases as shown on Figures 7.16, 7.17, and 7.18. These three figures give a clear pictorial representation of how the number of late parts are affected as the relative weightings of the three rule sets are changed. If the weights are expressed in the form process step weights:scheduler weights:buffer level weights, it can be seen from Figure 7.16 that the optimum weight setting to minimize the number of late parts would be 1:3:1, resulting in only one late part. Similarly for Figure 7.17, the optimum is 1:1:2 with three late parts and for Figure 7.18, the optimum is 1:3:1 with one late part. From this type of analysis it is clear that the most optimum combination of weights to be used for the fuzzy logic base case is 1:3:1 if one wishes to minimize the number of late parts, and this is the combination that was used whenever the optimized fuzzy method was referenced. On the other hand, if minimizing the performance measure of average production time is considered, then this performance measure can be plotted for the 300 cases as shown in Figure 7.19, 7.20 and 7.21. Figure 7.19 indicates an average production time optimum of 52 minutes at 2:1:1. Figure 7.20 indicates the same optimum of 52 minutes in a number of places; firstly in the range (6-10):1:(2-4) and secondly at 2:1:1 (hidden behind a crest). The optimum shown in Figure 7.21 is 57 minutes located at 1:2:1, thus the best overall combination with respect to average

Figure 7.16: Effect of Scheduler and Process Step Weight Modification on the Number of Late Parts

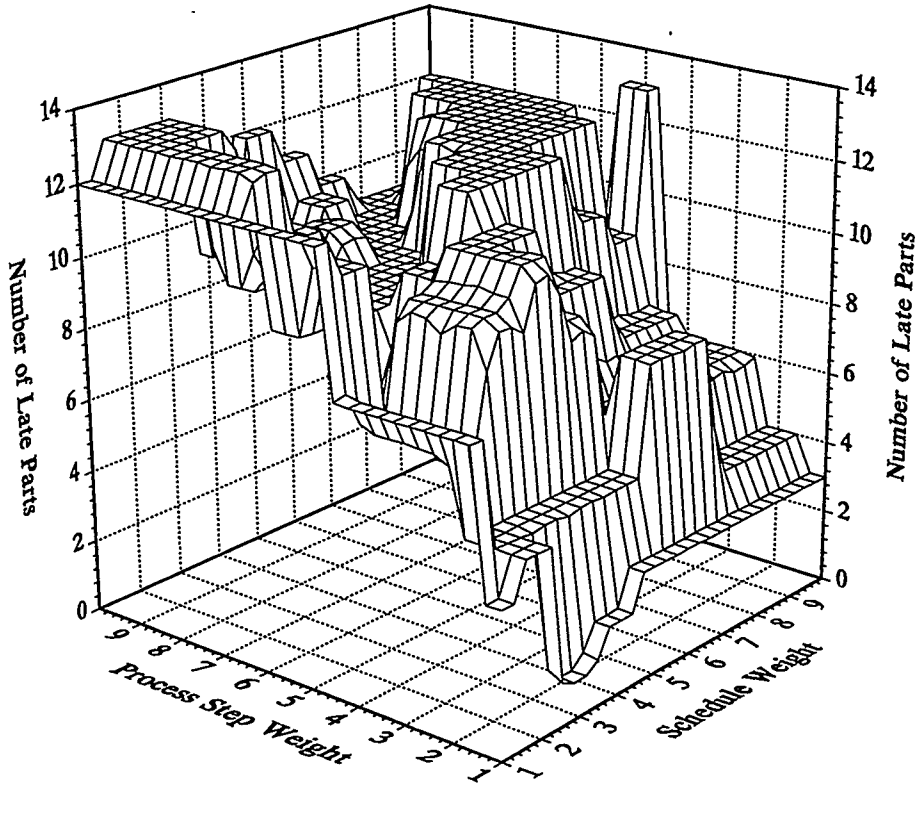


Figure 7.17: Effect of Buffer Level and Process Step Weight Modification on the Number of Late Parts

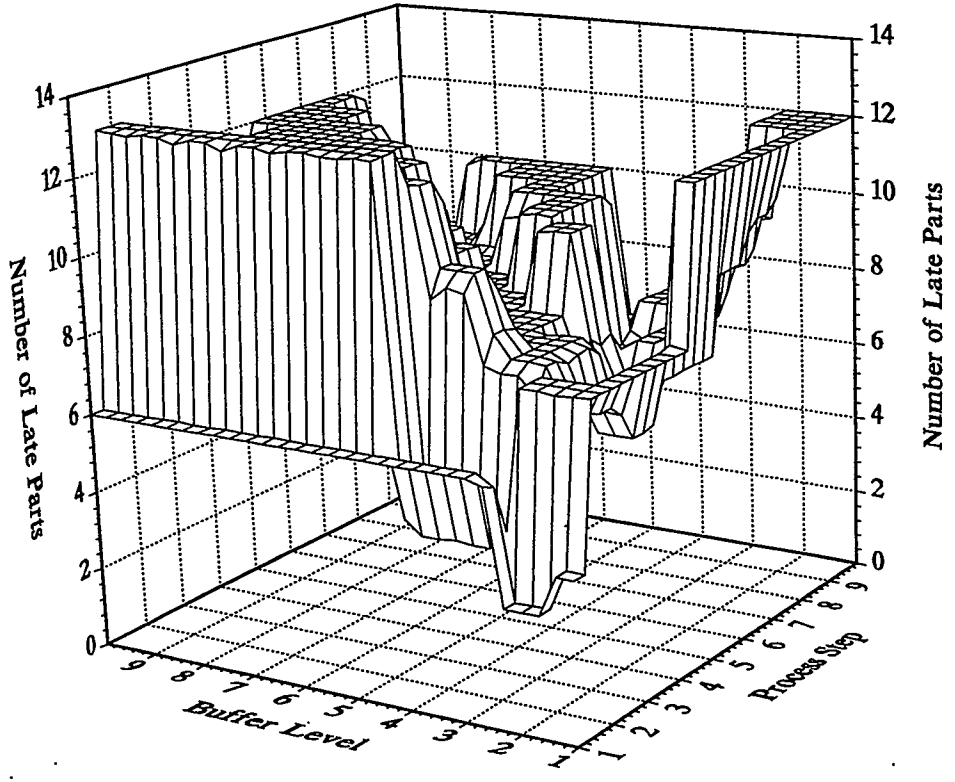


Figure 7.18: Effect of Scheduler and Buffer Level Weight Modification on the Number of Late Parts

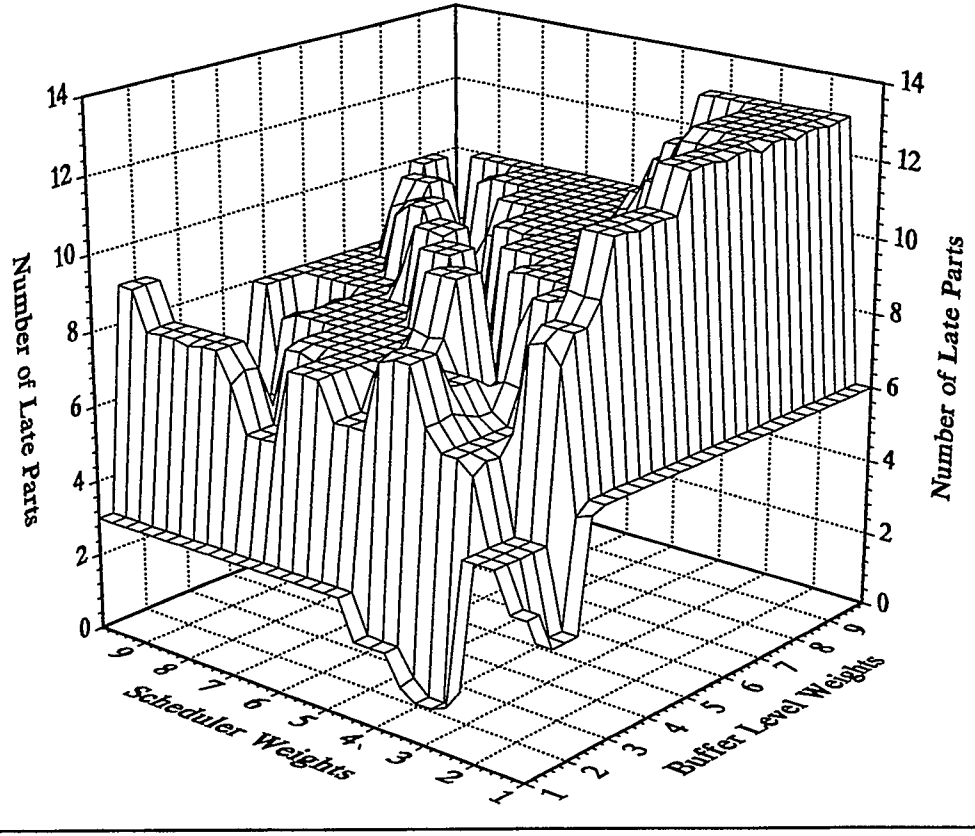


Figure 7.19: Effect of Scheduler and Process Step Weight Modification on Average Production Time

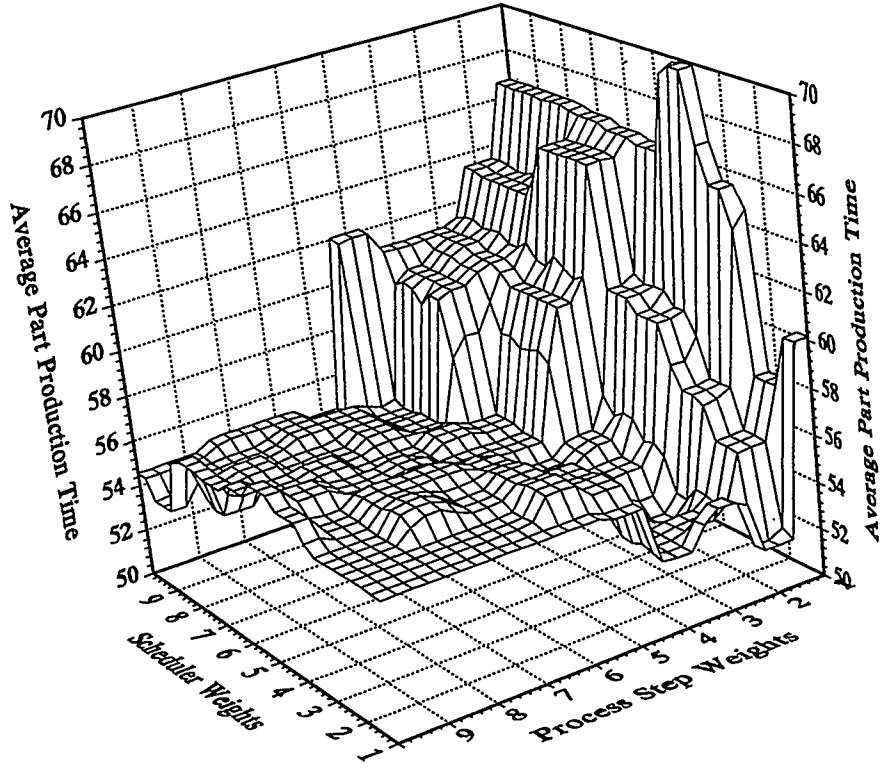


Figure 7.20: Effect of Buffer Level and Process Step Weight Modification on Average Production Time

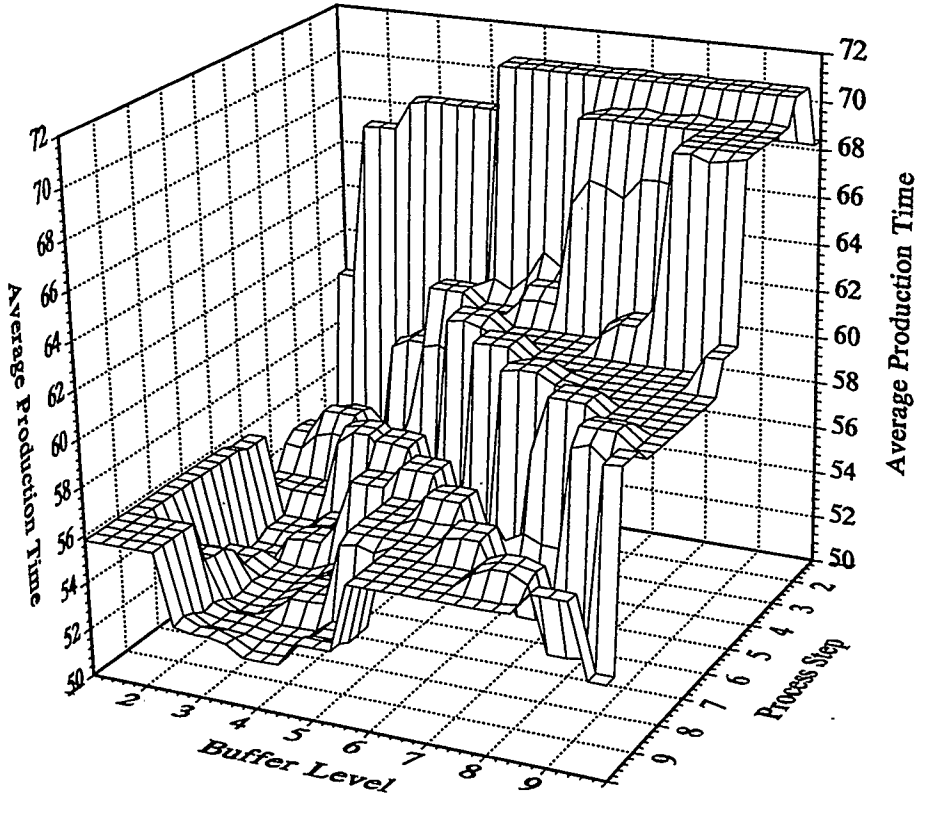
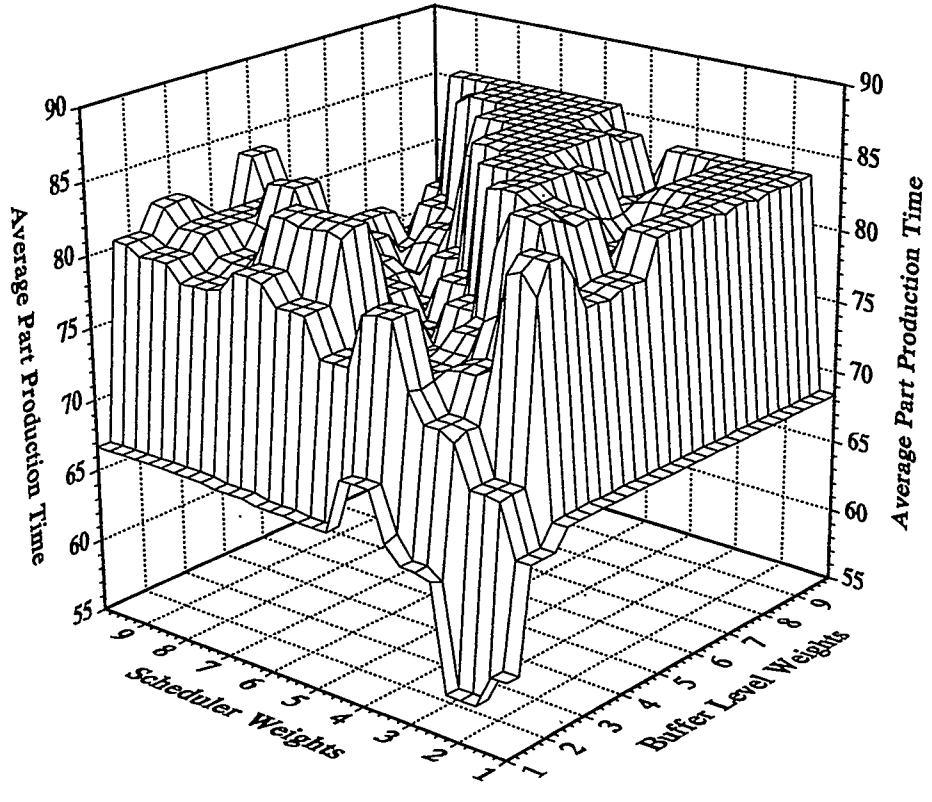


Figure 7.21: Effect of Scheduler and Buffer Level Weight Modification on Average Production Time



production time is 2:1:1 yielding a time of 52 minutes.

Very often in production facilities there are a number of primary performance measures which need to be met. In case 8, if both the performance measures of late parts and average production time were equally important then there would be a conflict because 1:3:1 results in few late parts but a high average production time whereas 2:1:1 results in a low average production time and more late parts. The issue of resolving this type of conflict has not been dealt with in this project. However, case 8 clearly shows that the correct selection of weights can have a significant impact on the outcome of various performance measures. The procedure used here to find the optimum would become prohibitively time consuming if a large search space exists but alternate searching strategies exist which may converge onto a solution in a much quicker fashion.

This concludes chapter 7. Several test cases have been presented which show that the control structure is very effective in controlling and scheduling a simulated manufacturing cell. The cases have shown how the control structure dispatches parts, how it deals with breakdown and repair events, how it reconfigures for special parts, and how robot movement is performed. Examples have also shown that the system can deal with very large cases, and that the optimized fuzzy logic method appears to work better for certain performance measures than a number of dispatching rules over a large finite buffer limit range. Finally, it has been shown how the rule set weights can be modified in order to optimize the system for a specific performance measure.

# CHAPTER 8

## 8. CONCLUSIONS

### 8.1 GENERAL DISCUSSION

The cell control structure, its implementation and a number of test cases have now been presented. The control structure's modular design incorporates an internal supervisory and scheduler structure and provides a great degree of flexibility during implementation and operating. This modular design, combined with the development of the system in an object oriented environment, allows the control structure to be very portable and expandable. Thus additional machines and part objects can be created and easily introduced into the system. The control structure can also work effectively in a manufacturing cell, an inspection cell or an assembly cell; the type of parts or machines are not important and do not affect the control structure. Furthermore, the structure has been implemented at the cell level; however, the concept could be used at other levels,

such as the shop floor level. The procedure for selecting part types which should be processed by a certain cell is not that much different than selecting parts for processing on a specific machine, and therefore the fuzzy logic method could still apply. There will still be scheduling considerations (due dates, inventory goals), buffer considerations (room in the cell), and other unique considerations which could easily be added to the fuzzy logic scheduling rule base. The only differentiation in using the control structure at different levels may be a different fuzzy logic rule base and different elements under the control of the supervisor. For example, at the shop floor level, the nodes of the primary level may represent cells instead of machines and the secondary control level would be the detailed operation of the cell. The control structure could be implemented hierarchically or autonomously. If implemented autonomously the parts and machines would be considered to be independent entities and the message passing that is currently done through the scheduler module could be done directly from one entity to another. All in all, the combination of the two layer supervisory structure and the fuzzy logic dispatching method has provided a very flexible and portable control environment.

## 8.2 SUPERVISOR DISCUSSION

As described in chapter 5, the supervisor is made up of primary and secondary control levels. The division of the control structure into these two levels has simplified implementation of the control system and also provided the flexibility to easily make changes. Elements at the primary control level can easily be added or removed without

affecting the existing elements or the detailed operational instructions in the secondary layer. Additionally, elements of the secondary level can be modified without affecting other portions of the secondary control level or the primary level. This type of two level modular structure reduces the complexity of the computer code and therefore makes development or refinement of the code easier.

The sample cases given in chapter 7 show the ability of the supervisor to perform all the required tasks; namely, controlling the machines, monitoring for breakdowns, initiating error recovery routines (breakdown, deadlock messages), directing part movement and part production, controlling the material handling system (robot), initiating the scheduling of parts, calling for part pick-up, monitoring inventory levels, monitoring the cell input buffer, and virtually reconfiguring the cell. The supervisor structure is set up in a manner that can deal with changes and disruptions. Alternate machines exist to allow the rerouting of parts in the event of machine or tool breakdowns, and alternate process plans can be used to redistribute part flow. The supervisor works closely with the scheduler module in order to dispatch parts, and it is clear from the examples given that the supervisor structure can react in real time using a variety of dispatching strategies.

## 8.3 SCHEDULER DISCUSSION

The heart of the control structure is the fuzzy logic dispatching method. The fuzzy logic method provides the mechanism whereby a number of different constraints and conditions within the shop floor can be considered at once. Current dispatching rules typically focus on only a few aspects of the shop floor whereas the fuzzy logic method is restricted only by the rule set, which can be expanded or customized to suit each particular job shop. Thus, scheduling aspects as well as control considerations can be included in the rule sets and this ultimately provides a great deal of power and flexibility when it comes to dispatching parts. Currently the rule base includes scheduling aspects such as due dates and inventory levels and as well as control aspects such as buffer level and part process plans. The extent and degree of the rule set is limited only by an ability to generate realistic rules and to develop methods of defuzzifying the appropriate variables.

The fuzzy dispatching method has been shown to be a very powerful and useful method. The examples given in chapter 7 show quite clearly that the optimized fuzzy logic method often performs significantly better than the five dispatching rules considered (FIFO, LIFO, EDD, SPT, and Slack/OPNR) especially in terms of minimizing the number of late parts and in preventing deadlocking. Furthermore, even though specific rules have not been developed for the performance measures of maximum lateness and average production time, the fuzzy logic method still shows equivalent or improved performance

relative to the dispatching rules.

One of the keys to having a successful fuzzy logic method is optimization. In this project, weights were assigned to each rule set and these weights were adjusted until an "optimum" was reached with respect to a certain performance measure (mainly number of late parts). The optimization procedure that was described in chapter 7 consisted mainly of conducting a number of trials and selecting the best result. As more rule sets are implemented, this strategy would become too time consuming and ineffective. Therefore alternative search strategies can be used such as neural networks or genetic algorithms to either find an optimum prior to implementing the system on the shop floor, or to search for an optimum on an ongoing basis while the control system is actually in operation. It is clear, however, from the results obtained thus far, that optimization is very important since there is a large difference between the performance of the optimized and non-optimized fuzzy logic methods.

Two other important aspects of having a successful fuzzy logic method are the rule bases and the membership functions. As mentioned in chapter 4, the rule bases were developed mainly through intuition and experimentation. Unfortunately no hard and fast guidelines exist in the current literature which can aid in the generation of rule sets. Two possible sources of rules are to consult with the people on the shop floor or to implement various versions of existing dispatching rules. In addition to selecting appropriate rules, it is also important to select the degree of activation of the rules such as low, very low,

medium, etc. Again there are no straight forward answers; however, expert systems, genetic algorithms, or neural networks can be used to help adjust the rules and the rule activations until an optimum or at least an improved rule structure has been developed. Similarly, the development of the membership functions is more of an art than a science, although some guidelines are given in Kosko [26]. As well, Karr and Gentry [21] have advocated the use of genetic algorithms and/or heuristics to help tune and modify the membership functions to make them more effective for the problem at hand.

Although the selection of weights, rules and membership functions is not always clear, a number of methods exist which can refine the selection of all these elements and therefore it becomes less critical to make perfect selections immediately. Furthermore, the ability to change the weights, rules or membership functions, opens up a large variety of optimization techniques which makes the control and scheduling structure very flexible. As conditions change on the shop floor, so to must the control structure change, and to have a large variety of adjustment options at hand is very powerful. The manner in which fuzzy logic is implemented enables changes to occur in real time (adaptively) or to be implemented more reactively by an operator.

## 8.4 ORIGINAL CONTRIBUTION

The main area of original contribution lies in the unique utilization of fuzzy logic for part dispatching. Fuzzy logic has been used for scheduling by other authors in several ways: controlling the flow of parts among resources (Custodio et al. [6]), production rate computation in order to maintain a set production level [6], and combining several dispatching rules (Grabot and Geneste [11]). As described in chapter 2, Custodio et al. uses four decision factors or routing rules to choose a resource for the next operation of a part and uses three decision functions (dispatching rules) to select the next part for processing. Fuzzy logic is then used on these routing and dispatching rules to do the final selections of parts and machines. The proposed fuzzy logic dispatching method is similar to the method presented by Custodio et al. in that it uses a number of rules or criteria for part dispatching; however, unlike Custodio et al. the proposed method includes scheduling considerations.

The proposed fuzzy logic method also attempts to move away from using common dispatching rules such as those implemented by Grabot and Geneste, and provides a method where any applicable rules related to parts, machines, scheduling or other critical elements, can be incorporated into the decision making process. Thus, the method becomes more all encompassing than the one proposed by Custodio et al. or Grabot and Geneste. Custodio et al. also uses fuzzy logic as a control method to select production rates which maintain specified production levels. In the early development of the fuzzy

logic dispatching scheme described in this thesis, a scheduler to meet a specified demand rate was also created and it operated in a fashion that was similar to the one described by Custodio. The scheduler was required to maintain a specified target inventory level and as the inventory level approached or exceeded the target, the fuzzy scheduler adjusted the production rate accordingly. The development and application of the rate based fuzzy scheduler has not been discussed here since it was felt that the scheduling (dispatching) of a due date, batch order driven shop floor was a more important and difficult problem.

There is also a certain degree of original contribution within the actual control structure that has been proposed. Several different control structures were discussed in chapter 2, but none of them combine a two level supervisor with a scheduler in the manner proposed here. The two level supervisor provides an increased degree of flexibility when it comes to adding or modifying the structure. The proposed control structure also describes how the supervisor and scheduler uniquely interact to dispatch parts, deal with broken machines and tools, and implement the concept of reconfigurability.

To summarize, the proposed fuzzy logic dispatching method provides a more all encompassing approach to part dispatching. It includes a full range of issues beyond common dispatching rules and it provides a scheduling methodology for a due date, batch order driven problem. The method improves greatly on the goal attainment of various performance measures as compared to several common dispatching rules. Furthermore,

the dispatching method shows great flexibility when combined with the proposed control structure and incorporates many options for optimization.

## 8.5 FINAL SUMMARY

The main objective of this research was to develop a highly flexible real time control structure that can be applied at the manufacturing cell level and which is capable of dealing with shop floor disruptions. The scheduling portion of the control structure is an integral part of meeting the above objective but it was also intended to consider a wider range of constraints and performance measures. One further requirement for the scheduling method was the inclusion of mechanisms for improvement or optimization. The intent was also to develop a framework which supported the concept of autonomous agents, decentralized control and provided a high degree of expandability and portability.

The above goals have all been met. The control structure deals effectively with machine and tool breakages and has the flexibility to deal with a wide range of numbers of parts and machines. The two level supervisor structure allows for the easy addition and modification of elements and the object oriented environment promotes the easy creation of those new shop floor elements. The dispatching of parts occurs in real time utilizing priority levels established from a number of different criteria, thereby considering a broad perspective of shop floor conditions. Optimization mechanisms exist such as rule weight modification, rule addition or modification, or membership function adjustments.

The fuzzy logic dispatching method has been shown to be more effective in minimizing the number of late parts than several dispatching rules and is certainly competitive with respect to minimizing average maximum buffer levels. As far as preventing deadlocking, the optimized fuzzy logic method was equivalent to LIFO but out-performed all the other rules that were investigated.

The control structure is a self contained modular unit needing only certain specific due date/inventory level goals to operate. Thus it would be effective in a decentralized control environment. Parts and machines have been developed as independent entities to a certain degree and the control structure could be converted easily to an environment where parts and machines could operate autonomously. The control structure is expandable and portable. As described in the general discussion of this chapter, the structure could easily represent different types of cells or different control levels. Finally, the control structure incorporates a unique combination of elements which provide a high degree of flexibility, autonomy, portability, and an ability to deal with disturbances.

## 8.6 FUTURE RESEARCH

There are many directions that future research can take with respect to this project. The most important one is to investigate methods of optimizing the rule set weights, the membership functions, or the rules themselves. It is suggested that neural networks, expert systems, genetic algorithms or a combination of all three be reviewed and

considered as potential optimization procedures.

Another important aspect is to develop a rule structure which can attain the goals of a number of additional performance measures such as maximum lateness, average lateness, earliness, mean tardiness given tardy, and average production time. At the same time, methods need to be developed which allow the system to achieve the best overall results given that some of the performance measures can often be conflicting.

Finally, the current simulation needs to be made more user friendly and interactive. Also consideration should be given to linking the control structure to a commercial simulation package which can provide a much better base for the development of test cases and the capture of statistical data.

# REFERENCES


1. Bauer A., Bowden R., Browne J., Duggan J., Lyons G., Shop Floor Control Systems From Design to Implementation, Chapman & Hall, Cornwall, Great Britain, 1991.

2. Blackstone Jr. J.H., Phillips D.T., Hogg G.L., "A State-of-the-art Survey of Dispatching Rules for Manufacturing Job Shop Operations", International Journal of Production Research, Vol 20, No. 1, 1982, pp. 27-45.

3. Cho H., Wysk R.A., "A Robust Adaptive Scheduler for an Intelligent Workstation Controller", International Journal of Production Research, Vol. 31, No. 4, 1993, pp. 771-789.

4. Chryssolouris G., Wright K., Pierce J., Cobb W., "Manufacturing Systems Operation: Dispatch Rules Versus Intelligent Control", Robotics & Computer-Integrated Manufacturing, Vol. 4, No. 3/4, 1988, pp. 531-544.

5. Curtis W., Tiemersma Ir.J.J., "A Real Time Control Network for Small Batch Part Manufacturing", Computer Integrated Manufacturing, Proc. of the Seventh CIM-Europe Annual Conference, May 29-31, 1991, pp. 115-125.

6. Custodio L.M.M., Sentierio J.J.S., Bispo C.F.G., "Production Planning and Scheduling Using a Fuzzy Decision System", IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, April 1994, pp. 160-167.

7. Domenikos H.G., Tatsiopoulos I.P., "Intelligent Design of Shop-floor Management Within a Supervisory Control System", Computer Integrated Manufacturing, Proc. of the Seventh CIM-Europe Annual Conference, May 29-31, 1991, pp. 127-138.

8. Duffie N.A., Piper R.S., "Non-Hierarchical Control of A Flexible Manufacturing Cell", Robotics & Computer-Integrated Manufacturing, Vol. 3, No. 2, 1987, pp. 175-179.

9. French S., Sequencing and Scheduling, An Introduction to the Mathematics of the Job-Shop, Ellis Horwood Ltd., Chichester, England, 1987.

10. Gendreau D., Lesage J.J., Timon G., "An Integration of Production Management Rules and Fabrication Know-how for Real Time Cell Production Control", Robotics & Computer-Integrated Manufacturing, Vol. 10, No. 1/2, 1993, pp. 115-122.

11. Grabot B., Geneste L., "Dispatching Rules In Scheduling: A Fuzzy Approach", International Journal of Production Research, Vol. 32, No. 4., 1994, pp. 903-915.

12. Hasegawa M., Masayuki T., Temmyo T., Matsuka H., "Modelling of Exception Handling in Manufacturing Cell Control and Its Application to PLC Programming", Proceedings 1990 IEEE Int. Conf. on Robotics and Automation, Cincinnati, Ohio, Vol. 1, May 13-18, 1990, pp. 514-519.

13. Hatvany J., "Intelligence and Cooperation in Heterarchical Manufacturing Systems", Robotics & Computer-Integrated Manufacturing, Vol. 2, No. 2, 1985, pp. 101-104.

14. Herrmann J.W., Lee C., Hinchman J., "Global Job Shop Sceduling with a Genetic Algorithm", Research Report 93-26, Department of Industrial & Systems Engineering, University of Florida, Gainsville, Fl., September 1993.

15. Huang H., Chang P., "Specification, Modelling and Control of a Flexible Manufacturing Cell", International Journal of Production Research, Vol. 30, No. 11, 1992, pp. 2515-2543.

16. Jones A.T., McLean C.R., "A Proposed Hierarchical Control Model for Automated Manufacturing Systems", Journal of Manufacturing Systems, Vol 5., No. 1, 1986, pp. 15-25.

17. Jones A., Saleh A., "A Decentralized Control Architecture for Computer Integrated Manufacturing", Proceeding of the IEEE International Symposium on Intelligent Control, 1989, pp. 44-49.

18. Joshi S.B., Smith J.S., "Intelligent Control of Manufacturing Systems". In A. Kusiak (Ed.) Intelligent Design and Manufacturing, John Wiley & Sons Inc., 1992, pp. 491-520.

19. Kals H.J.J., "Advanced Manufacturing in Small Batch Part Production", Computer Applications in Production and Engineering: Proceedings of the Fourth International IFIP TC5 Conference on Computer Applications in Production and Engineering - Integration Aspects, CAPE '91, Bordeaux, France, September 10-12, 1991, pp. 85-94.

20. Kamath M., Viswanadham N., "Applications of Petri Net Based Models in the Modelling and Analysis of Flexible Manufacturing Systems", Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, 1986, pp. 312-317.

21. Karr C.L., Gentry E.J., "Fuzzy Control of pH Using Genetic Algorithms", IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, February 1993, pp. 46-53.

22. Kasturia E., DiCesare F., Desrochers A., "Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets", IEEE Conference on Robotics and Automation, 1988, pp. 1114-1119.

23. Kaufmann A, Gupta M.M., Fuzzy Mathematical Models in Engineering and Management Science, Elsevier Science Publishing Company Inc., New York, N.Y., 1991.

24. Kickert W.J.M., Van Nauta Lemke H.R., "Application of a Fuzzy Controller in a Warm Water Plant", Automatica, Vol. 12, 1976, pp. 301-308.

25. King P.J., Mamdani E.H., "The Application of Fuzzy Control Systems to Industrial Processes", Automatica, Vol. 13, 1977, pp. 235-242.

26. Kosko B., Neural Networks and Fuzzy Systems, A Dynamical Systems Approach To Machine Intelligence, Prentice Hall, Englewood Cliffs, N.J., 1992.

27. Kusiak A., Intelligent Manufacturing Systems, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1990.

28. Lee D.Y., DiCesare F., "Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search", IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, April 1994, pp. 123-132.

29. LeFrancois P., Montreuil B., "An Object-Oriented Knowledge Representation For Intelligent Control of Manufacturing Workstations", IIE Transactions, Vol. 26, No. 1, January 1994, pp. 11-26.

30. Lin G.Y., Solberg J.J., "Integrated Shop Floor Control Using Autonomous Agents", IIE Transactions, Vol. 24, No. 3, July 1992, pp. 57-71.

31. Maimon O.Z., "Real-time Operational Control of Flexible Manufacturing Systems", Journal of Manufacturing Systems, Vol. 6, No. 2, 1987, pp. 125-136.

32. Maley J.G., "Managing the Flow of Intelligent Parts", Robotics & Computer-Integrated Manufacturing, Vol. 4, No. 3/4, 1988, pp. 525-530.

33. Manivannan S., Banks J.,"Design of a Knowledge-based On-line Simulation System to Control a Manufacturing Shop Floor", IIE Transactions, Vol. 24, No. 3, July 1992, pp. 72-83.

34. Merabet A.A., "Synchronization of Operations in a Flexible Manufacturing Cell: The Petri Net Approach", Journal of Manufacturing Systems, Vol. 5, No. 3, 1986, pp. 161-169.

35. O'Grady P.J., Lee K.H., "An Intelligent Cell Control System for Automated Manufacturing". In A. Kusiak (Ed.), Knowledge-Based Systems in Manufacturing, Taylor & Francis Inc., Philadelphia, 1989, pp. 151-172.

36. O'Grady P.J., Seshadri R., "Operation of X-Cell - An Intelligent Cell Control System", Computer Integrated Manufacturing Systems, Vol. 5, No. 1, February 1992, pp. 21-30.

37. Panwalkar S.S, Iskander W., "A Survey of Dispatching Rules", Operations Research, Vol. 25, No. 1, January-February 1977, pp. 45-61.

38. Ravichandran R., Chakravarty A.K., "Decision Support in Flexible Manufacturing Systems Using Timed Petri Nets", Journal of Manufacturing Systems, Vol. 5, No. 2, 1986, pp. 89-100.

39. Teng S., Black J.T., "Cellular Manufacturing Systems Modelling: The Petri Net Approach", Journal of Manufacturing Systems, Vol. 9, No. 1, 1990, pp. 45-53.

40. Tong R.M., "A Control Engineering Review of Fuzzy Systems", Automatica, Vol. 13, 1977, pp. 559-569.

41. Tzafestas S., "Petri-net and Knowledge-based Methodologies in Manufacturing Systems Modelling Simulation and Control", Proceedings of the 5th CIM Europe Conference, May 17-19, 1989, pp. 39-50.

42. Uckun S., Bagchi S., Kawamura K., Miyabe Y., "Managing Genetic Search in Job Shop Scheduling", IEEE Expert, October 1993, pp. 15-24.

43. Wu S.D., Wysk R.A., "Multi-pass Expert Control System - A Control/Scheduling Structure for Flexible Manufacturing Cells", Journal of Manufacturing Systems, Vol. 7, No. 2, 1988, pp. 107-120.

44. Zadeh L.A., "Knowledge Representation in Fuzzy Logic", IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1, March 1989, pp. 89-99.

45. Zhou M.C., "Combination of Petri Nets and Intelligent Decision Makers for Manufacturing Systems Control", Proceedings of the 1991 IEEE International Symposium on Intelligent Control", Arlington, Virginia, August 13-15, 1991, pp. 146-151.

# APPENDIX A: EQUIPMENT CLASS INSTANCES

---

VertMill1

name: 'VertMill1'

toolLib: T023, T016, T008, T007, T025

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'VertMill2'

description: 'milling machine'

inputBufferLib: empty

currentProcess: empty

VertMill2

name: 'VertMill2'

toolLib: T023, T016, T008, T007, T025

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'VertMill1'

description: 'milling machine'

inputBufferLib: empty

currentProcess: empty

Drill1

name: 'Drill1'

toolLib: T008, T007, T010, T009, T030

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'Drill2'

description: 'drill'

inputBufferLib: empty

currentProcess: empty

Drill2

name: 'Drill2'

toolLib: T008, T007, T010, T009, T030

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'Drill1'

description: 'drill'

inputBufferLib: empty

currentProcess: empty

TurnCentre1

name: 'TurnCentre1'

toolLib: T008, T007, T010, T040, T041

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'TurnCentre2'

description: 'CNC turning centre'

inputBufferLib: empty

currentProcess: empty

TurnCentre2

name: 'TurnCentre2'

toolLib: T008, T007, T010, T040, T041

assignTo: 'Cell1'

status: 'Ok'

alternateStationLib: 'TurnCentre1'

description: 'CNC turning centre'

inputBufferLib: empty

currentProcess: empty

# APPENDIX B: TOOL CLASS INSTANCES

---

T007

name: 'T007'
description: 'HSS drill; D0.5 x L2.25in.'
assignTo: 'VertMill1', 'VertMill2', 'Drill1', 'Drill2',
'TurnCentre1', 'TurnCentre2'
status: 'Ok'

T008

name: 'T008'
description: 'HSS drill; D1.5 x L4.875in.'
assignTo: 'VertMill1', 'VertMill2', 'Drill1', 'Drill2',
'TurnCentre1', 'TurnCentre2'
status: 'Ok'

T009

name: 'T009'
description: 'HSS drill; D0.375 x L2.25in.'
assignTo: 'Drill1', 'Drill2'
status: 'Ok'

T010

name: 'T010'
description: 'HSS drill; D0.75 x L2.5in.'
assignTo: 'Drill1', 'Drill2', 'TurnCentre1', 'TurnCentre2'
status: 'Ok'

T016

name: 'T016'
description: 'HSS fourFluteEndMill; D1.5 x L3.0in.'
assignTo: 'VertMill1', 'VertMill2'
status: 'Ok'

T023

name: 'T023'
description: 'HSS studEndMill; D2.0 x L0.75in.'
assignTo: 'VertMill1', 'VertMill2'
status: 'Ok'

T025

name: 'T025'
description: 'HSS twoFluteEndMill; D0.5 x L1.0in.'
assignTo: 'VertMill1', 'VertMill2'
status: 'Ok'

T030

name: 'T030'
description: 'tap; D0.375-UNC x L2.25in.'
assignTo: 'Drill1', 'Drill2'
status: 'Ok'

T040

name: 'T040'
description: 'TC narrow gauge bit.'
assignTo: 'TurnCentre1', 'TurnCentre2'
status: 'Ok'

T041

name: 'T041'

description: 'TC roughing bit.'

assignTo: 'TurnCentre1', 'TurnCentre2'

status: 'Ok'

# APPENDIX C: PARTTYPE CLASS INSTANCES

---

partA

name: 'partA'

processPlan: 'T023', 'VertMill1', 4
'T016', 'VertMill2', 6
'T041', 'TurnCentre1', 4
'T009', 'Drill1', 8

inventoryLevel: 0

timeLevel: 0

batchTime: 240

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partB

name: 'partB'

processPlan: 'T023', 'VertMill2', 4
'T008', 'Drill1', 6
'T041', 'TurnCentre2', 4
'T007', 'Drill2', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 200

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partC

name: 'partC'

processPlan: 'T007', 'Drill2', 3
'T041', 'TurnCentre1', 4
'T030', 'Drill1', 5
'T023', 'VertMill2', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 300

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partD

name: 'partC'

processPlan: 'T041', 'TurnCentre2', 7
'T008', 'Drill2', 4
'T023', 'VertMill1', 8
'T016', 'VertMill2', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 300

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partE

name: 'partE'

processPlan: 'T023', 'VertMill2', 4
'T016', 'VertMill1', 6
'T007', 'Drill2', 4
'T040', 'TurnCentre1', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 240

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partF

name: 'partF'

processPlan: 'T007', 'TurnCentre1', 5
'T023', 'VertMill2', 3
'T008', 'Drill1', 4
'T041', 'TurnCentre1', 6

inventoryLevel: 0

timeLevel: 0

batchTime: 320

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partG

name: 'partG'

processPlan: 'T023', 'VertMill1', 5
'T040', 'TurnCentre1', 4
'T007', 'Drill1', 3
'T041', 'TurnCentre2', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 320

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partH

name: 'partH'

processPlan: 'T010', 'Drill1', 3
'T041', 'TurnCentre2', 5
'T030', 'Drill2', 4
'T016', 'VertMill1', 1

inventoryLevel: 0

timeLevel: 0

batchTime: 340

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partI

name: 'partI'

processPlan: 'T009', 'Drill2', 4
'T008', 'Drill2', 5
'T016', 'VertMill1', 4
'T007', 'Drill1', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 260

batchAmount: 10

partLotNumber: 1

status: 'Regular'

partJ

name: 'partJ'

processPlan: 'T007', 'TurnCentre2', 4
'T016', 'VertMill2', 4
'T041', 'TurnCentre1', 5
'T025', 'VertMill1', 2

inventoryLevel: 0

timeLevel: 0

batchTime: 240

batchAmount: 10

partLotNumber: 1

status: 'Regular'