Design and analysis of agent-based
FMS control Systems

by

Fan Chi Kit

B.Eng. (Hons), MPhil.

A thesis submitted to the University of Hong Kong for
the degree of Doctor of Philosophy

Department of Industrial and Manufacturing Systems Engineering
The University of Hong Kong

September 2005
Abstract of thesis entitled

**Design and analysis of agent-based FMS control systems**

submitted by

**Fan Chi Kit**

for the degree of Doctor of Philosophy

at The University of Hong Kong

in March 2005

A flexible manufacturing system (FMS) is a complex manufacturing system, which demands a robust control software for its scheduling, planning and control functions. This study considers the applications of multi-agent systems (MAS) in FMS control systems. Its aim was to establish a new hybrid-based approach in the design and analysis of agent-based FMS control systems.

The proposed hybrid approach is based on the UML (Unified Modelling Language) and CPN (Coloured Petri Net) modeling technologies. The UML modeling methodology was adopted in the design and specification of the MAS-based FMS control systems and the individual agent. A mapping mechanism was developed to convert the UML views into CPN models. The CPN models can then be used to represent the dynamic behavior and interactions of agents in the MAS. In the corresponding CPN model, resource allocation by agents is represented by the color tokens.

Three different flexible assembly cell (FAC) configurations were used as examples of manufacturing systems. According to a simple FAC configuration, a generic
agent-based FAC control system was established. Using this generic agent-based FAC control system as the basis, the hybrid UML-CPN modeling approach was used to establish the MAS models for the two FAC configurations. To evaluate system flexibility and the effectiveness of different scheduling approaches, two different pallet allocation rules, the first-come first-served (FCFS) rule and the tabu search-based algorithm, were applied to each of the three FAC configurations. Simulation and analysis of the CPN models were then carried out to evaluate the dynamic behavior and interactions of the agent-based FAC control systems. Token activities of each simulation run were recorded and used for analysis. A data mining algorithm, the Apriori algorithm, was then used to mine the agent activity data to obtain useful information for evaluating the performances of the different agent-based FAC control systems in the case studies. Based on the evaluation results, the UML components of the agent-based system model can be modified in accordance with the dynamic models.

The proposed hybrid UML-CPN models are able to cope with the flexibility requirements of complex flexible manufacturing environments. The examples demonstrated that these hybrids were to support the robust design and flexibility requirements of agent-based control systems corresponding to changes or disturbances in flexible manufacturing systems. The design methodology and analytical framework based on the hybrid UML-CPN models were found to be adaptable and flexible, and were capable of coping with sophisticated planning and scheduling problems in complex automated manufacturing environments under different system configurations.
Declaration

I hereby declare that the Doctor of Philosophy thesis titled “Design and analysis of agent-based FMS control systems” represents my own work. I also declare that the work reported in this thesis has not been previously included in a thesis, dissertation or report submitted to this University or to any other institution for a degree, diploma, or other qualification.

Fan Chi Kit

March, 2005
ACKNOWLEDGEMENTS

I am deeply thanksgiving to peoples who give me supports these years during my postgraduate studies.

Dr. T. N. Wong always gives his generous knowledge and expertise to my research project. I have been indebted to his supervision since my undergraduate’s final year project. His enthusiastic support and systematic guidance lead me towards this research achievements.

I would also like to thank the Department of Industrial and Manufacturing Systems Engineering for the financial support and ancillary facilities.

My thanks is also extended to Mr. W. K. Leung. He has often given his technical guidance to my project and uncountable advices to my personal matters over the years. Although Amy, Devil, Lawerance, May, Michael, Kei, Sai Ming, Tim and Tyson have left the department for years, I would never forget the joyful moments with them.

I dedicate this thesis to my parents and my younger brother for their endless faith to me. Finally, I would like to express my deepest gratitude for the constant support, understanding and love from Lily.
TABLE OF CONTENTS

Declaration .................................................................................................................................................. i
Acknowledgement ....................................................................................................................................... ii
Table of Contents ....................................................................................................................................... iii
List of Figures ............................................................................................................................................. viii
List of Graphs ............................................................................................................................................ xv

Chapter 1. INTRODUCTION

1.1 Research objective .............................................................................................................................. 1-2
1.2 Thesis organization .............................................................................................................................. 1-4

Chapter 2. LITERATURE REVIEW

2.1 Agents and multi agent systems ...................................................................................................... 2-1
2.2 Application of agent-based systems .................................................................................................. 2-2
2.3 Manufacturing applications by using agent ....................................................................................... 2-4
2.4 Agent-based manufacturing control system ..................................................................................... 2-7
    2.4.1 Traditional control architecture and its evolution .................................................................... 2-7
    2.4.2 Holonic Manufacturing system architecture ............................................................................. 2-7
    2.4.3 Agent based control architecture ............................................................................................. 2-9
2.5 Object-oriented modeling of FMS control system application ......................................................... 2-11
    2.5.1 Unified modeling language (UML) .......................................................................................... 2-12
    2.5.2 Extended unified modeling language ....................................................................................... 2-13
2.6 Design and analysis of agent-based FMS control system with UML ................................................. 2-14
2.7 Petri Net model in FMS control system application .......................................................................... 2-15
2.8 Conclusion ......................................................................................................................................... 2-17

Chapter 3. AGENT BASED FLEXIBLE ASSEMBLY SYSTEM CONTROL

3.1 Overview of the proposed agent-based FMS control system ......................................................... 3-2
3.2 FAC configurations ........................................................................................................................... 3-3
    3.2.1 Example 1.................................................................................................................................. 3-3
        3.2.1.1 Product assembly in FAC example ................................................................................. 3-5
        3.2.1.2 Configuration constraints of example ............................................................................. 3-7
### Chapter 4. DESIGN MODEL

4.1 Framework design of agent-based FMS control system 4-2  
4.2 The UML model 4-5  
4.2.1 UML modeling approach 4-5  
4.2.2 Logical view 4-8  
4.2.2.1 Use case diagram 4-9  
4.2.2.2 Class diagram and the agents object relationships 4-10  
4.2.2.2.1 Aggregation 4-12  
4.2.2.2.2 Association 4-14  
4.2.2.2.3 Generalization and specification 4-17  
4.2.2.3 Class diagram 4-19  
4.3 Conclusion 4-24
6.2 Tabu search algorithm for FAC examples  ........................................ 6-4
   6.2.1 Tabu search definition .................................................. 6-5
   6.2.2 Practical application of Tabu search in FAC control systems ........................................ 6-7
   6.2.2.1 Tabu search application in Example 1  ............... 6-7
   6.2.2.2 Tabu search application in Example 2  .................. 6-14
   6.2.2.3 Tabu search application in Example 3 .................... 6-22
   6.2.2.4 Pseudo code for the algorithm for the examples ........ 6-31

6.3 Concluding remarks ................................................................. 6-32

Chapter 7. SYSTEM ANALYSIS AND IMPLEMENTATION

7.1 Introduction to data mining analysis ........................................ 7-1
   7.1.1 Data mining application in this project ...................... 7-3
   7.1.2 Advantages using the Apriori algorithm in Data mining application in this project ................. 7-4
   7.1.3 Interpretation of simulation data .................................. 7-5
   7.1.4 Apriori Algorithm application in this project .............. 7-10
   7.1.5 Association rule and its parameters ......................... 7-11
   7.1.6 Apriori Algorithm implementation ............................. 7-12
   7.1.7 Apriori Algorithm implementation steps .................... 7-13
   7.1.8 Association rules expression in this application ........ 7-19
7.2 System Analysis to the FAC examples .................................... 7-21
   7.2.1 ANOVA analysis of the examples ................................. 7-21
      7.2.1.1 ANOVA tests (Significant tests of the algorithm application to the FAC examples) .............. 7-23
   7.2.2 Comparison of the agent token counts in graphical representations ..................................... 7-26
   7.2.3 Example 1 without algorithm application ................. 7-27
   7.2.4 Example 1 with algorithm application ....................... 7-29
   7.2.5 Example 2 without algorithm application ................. 7-32
   7.2.6 Example 2 with algorithm application ....................... 7-34
   7.2.7 Example 3 without algorithm application ................. 7-36
   7.2.8 Example 3 with algorithm application ....................... 7-38
   7.2.9 Examples evaluation .................................................. 7-41
   7.2.10 CPN property verification ........................................ 7-45
   7.2.11 Analysis methodology conclusion ............................. 7-46
7.3 System implementation ......................................................... 7-46
   7.3.1 Work Flow for the generic FAC control system .......... 7-47
   7.3.2 Tabu search-based algorithm and configuration changes implementation in examples 2 .................. 7-56
7.4 Chapter summary ................................................................. 7-74
Chapter 8. CONCLUSIONS AND FUTURE WORK

8.1 Conclusions .................................................................................. 8-1
8.2 Future work .................................................................................. 8-5

REFERENCES
# LIST OF FIGURES

<p>| Figure 3-1 | Physical configuration of the simple FAC example | 3-4 |
| Figure 3-2 | Top View of the simple FAC example | 3-5 |
| Figure 3-3 | Assembly operation for example 1 | 3-6 |
| Figure 3-4 | Work example for the FMS Control System for FAC example 1 | 3-6 |
| Figure 3-5 | Flow of pallet in generic FAC example | 3-7 |
| Figure 3-6 | Physical configuration of FAC in example 2 | 3-8 |
| Figure 3-7 | Top view of example 2 | 3-9 |
| Figure 3-8 | Physical configuration for the hypothetic example 2 | 3-12 |
| Figure 3-9 | Top view of the FAC example 3 | 3-13 |
| Figure 3-10 | Product 1 | 3-14 |
| Figure 3-11 | Machining specification for product 1 | 3-15 |
| Figure 3-12 | Product 2 | 3-15 |
| Figure 3-13 | Machining specification for product 2 | 3-16 |
| Figure 3-14 | Product 3 | 3-17 |
| Figure 3-15 | Machining specification for product 3 | 3-18 |
| Figure 3-16 | Product 4 | 3-18 |
| Figure 3-17 | Machining specification for product 4 | 3-19 |
| Figure 3-18 | Machining process routes and processing times for the four products in FAC example 3 | 3-19 |
| Figure 3-19 | Possible processing routes for the four products | 3-20 |
| Figure 3-20 | Generic agents structure for FAC | 3-23 |
| Figure 3-21 | Flow chart for the design and analysis of FAC control system | 3-28 |
| Figure 4-1 | Design phases in Rational Unified Process, Kruchton (2003) | 4-3 |
| Figure 4-2 | Conceptual design process for the generic FAC control system | 4-3 |
| Figure 4-3 | Resource requirement for the design and development of the generic FAC control system | 4-4 |
| Figure 4-4 | Why standard methodologies should be used in this project | 4-5 |</p>
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-5</td>
<td>The “4+1” view model</td>
<td>4-7</td>
</tr>
<tr>
<td>4-6</td>
<td>A generic agent based control system of an FAC in use case representation</td>
<td>4-10</td>
</tr>
<tr>
<td>4-7</td>
<td>Agent Relationship Diagram by Static UML Model</td>
<td>4-12</td>
</tr>
<tr>
<td>4-8</td>
<td>Enlarged class relationship diagram snapshot</td>
<td>4-13</td>
</tr>
<tr>
<td>4-9</td>
<td>Example Java code for aggregation</td>
<td>4-14</td>
</tr>
<tr>
<td>4-10</td>
<td>Association relationship between objects</td>
<td>4-15</td>
</tr>
<tr>
<td>4-11</td>
<td>Example Java code for association</td>
<td>4-16</td>
</tr>
<tr>
<td>4-12</td>
<td>Example Java code for generalization</td>
<td>4-18</td>
</tr>
<tr>
<td>4-13</td>
<td>A class in UML static diagram</td>
<td>4-19</td>
</tr>
<tr>
<td>4-14</td>
<td>The class diagram for the scheduling agent</td>
<td>4-21</td>
</tr>
<tr>
<td>4-15</td>
<td>The class diagram for the conveyor control agent</td>
<td>4-22</td>
</tr>
<tr>
<td>4-16</td>
<td>The class diagram for the robot agent</td>
<td>4-22</td>
</tr>
<tr>
<td>4-17</td>
<td>The class diagram for the pallet identity agent</td>
<td>4-23</td>
</tr>
<tr>
<td>5-1</td>
<td>Sequence diagram for use case “Scheduling agent work”</td>
<td>5-4</td>
</tr>
<tr>
<td>5-2</td>
<td>Sequence diagram for use case “Pallet identity agent work”</td>
<td>5-7</td>
</tr>
<tr>
<td>5-3</td>
<td>Sequence diagram for use case “Robot Agent work”</td>
<td>5-10</td>
</tr>
<tr>
<td>5-4</td>
<td>Sequence diagram for use case “Conveyor Control Agent work”</td>
<td>5-12</td>
</tr>
<tr>
<td>5-5</td>
<td>Collaboration diagram for the use case “Conveyor Control Agent Work”</td>
<td>5-14</td>
</tr>
<tr>
<td>5-6</td>
<td>Collaboration diagram for the use case “Pallet Identity agent work”</td>
<td>5-15</td>
</tr>
<tr>
<td>5-7</td>
<td>Collaboration diagram for the use case “Robot agent work”</td>
<td>5-16</td>
</tr>
<tr>
<td>5-8</td>
<td>Collaboration diagram for the use case “Scheduling agent work”</td>
<td>5-17</td>
</tr>
<tr>
<td>5-9</td>
<td>CommonData class in Java</td>
<td>5-19</td>
</tr>
<tr>
<td>5-10</td>
<td>Client class in Java</td>
<td>5-20</td>
</tr>
<tr>
<td>5-11</td>
<td>Form1 class in Java</td>
<td>5-20</td>
</tr>
<tr>
<td>5-12</td>
<td>ServerSocket class in Java</td>
<td>5-21</td>
</tr>
<tr>
<td>5-13</td>
<td>TextEventTest class in Java</td>
<td>5-21</td>
</tr>
</tbody>
</table>
Figure 5-14  tf1 class in Java  
Figure 5-15  tf2 class in Java  
Figure 5-16  tf3 class in Java  
Figure 5-17  tf4 class in Java  
Figure 5-18  ThreadHandler class in Java  
Figure 5-19  Form1 class in Visual Basic  
Figure 5-20  Com1 class in Visual Basic  
Figure 5-21  Com2 class in Visual Basic  
Figure 5-22  RobotProgram class in Visual Basic  
Figure 5-23  wskClient1_DataArrival class in Visual Basic  
Figure 5-24  wskClient2_DataArrival class in Visual Basic  
Figure 5-25  Form1 class in Visual Basic  
Figure 5-26  Barcode class in Visual Basic  
Figure 5-27  DatabaseRet class in Visual Basic  
Figure 5-28  com1 class in Visual Basic  
Figure 5-29  SensorSignPostAgent class in Visual Basic  
Figure 5-30  Com1 class in Visual Basic  
Figure 5-31  WskClient2 class in Visual Basic  
Figure 5-32  WskClient1 class in Visual Basic  
Figure 5-33  WskClient1_DataArrival class in Visual Basic  
Figure 5-34  Form1 class in Visual Basic  
Figure 5-35  wskClient1_DataArrival class in Visual Basic  
Figure 5-36  CAWskClient1 class in Visual Basic  
Figure 5-37  waitServer class in Visual Basic  
Figure 5-38  conveyorCtrl class in Visual Basic  
Figure 5-39  Com1 class in Visual Basic  
Figure 5-40  Com1 class in Visual Basic  
Figure 5-41  The CPN structure  
Figure 5-42  Summary for the design and Analysis methodology in this project  
Figure 5-43  Mapping steps of UML to CPN model  
Figure 5-44  Three use case diagrams
Figure 5-45  An unified use case diagram  
Figure 5-46  Use case transformation  
Figure 5-47  Classes/Objects transformation  
Figure 5-48  Operation transformation  
Figure 5-49  Use case diagram for the use case “Pallet Identity Agent work”  
Figure 5-50  CPN components for the use case diagram  
Figure 5-51  Agents and their objects in the sequence diagram representation  
Figure 5-52  Partial CPN model for the example 1  
Figure 5-53  Agents and their linkages  
Figure 5-54  CPN model for the generic FAC control system template  
Figure 5-55  Places assignment from agents for the generic FAC control system template  
Figure 5-56  Transitions for the proposed FAC control system template  
Figure 5-57  Schematic diagram to the place and transition with respect to operations  
Figure 5-58  Flow chart for the system execution of the FAC control system  
Figure 5-59  Places and transition for the loading and unloading operation  
Figure 5-60  Schematic diagram for the product in and out operation  
Figure 5-61  The scheduling agent interactions in CPN model  
Figure 5-62  Summary for the scheduling agent interactions  
Figure 5-63  CPN models relate to the Pallet identity agent and barcode agent  
Figure 5-64  Flow chart for the pallet identity agent and barcode agent  
Figure 5-65  CPN sub-model for conveyor control agent and robot agent  
Figure 5-66  Agents operation for the conveyor control agent and robot agent  
Figure 5-67  CPN for the generic agent-based control system  
Figure 5-68  Input Arc references for the generic agent-based control system  
Figure 5-69  Output Arc reference for the generic agent-based control system  
Figure 5-70  A simple CPN model in Design/CPN  
Figure 5-71  {if-then-else} example in guard of transition  
Figure 5-72  {for-next} example in CPN representation  
Figure 5-73  Illustrative example showing CPN model
| Figure 5-74 | Figure for the CPN example for the substitute transition page | 5-76 |
| Figure 5-75 | Flow chart for the example CPN model for the substitute transition page | 5-76 |
| Figure 5-76 | Main CPN model for the FAC control system | 5-78 |
| Figure 5-77 | Summary for the mapping between UML model to CPN model | 5-79 |
| Figure 5-78 | Mapping from sequence diagram to CPN model | 5-80 |
| Figure 5-79 | CPN model for the second configuration | 5-81 |
| Figure 5-80 | Comparison of the CPN models of the generic FAC control system to example 3 | 5-83 |
| Figure 5-81 | New CPN components control the number of machines in the FAC example | 5-84 |
| Figure 5-82 | Substitution page for the machine control agent’s transition | 5-85 |
| Figure 5-83 | Modified CPN model for the robot agent to the new example | 5-86 |

| Figure 6-1 | Pallet transportation direction for the FAC configuration 1. | 6-8 |
| Figure 6-2 | Simplified Tabu Search rules to the FAC example 1 | 6-9 |
| Figure 6-3 | Iterations for the Tabu Search | 6-11 |
| Figure 6-4a | The CPN models for FAC example 1 with Tabu Search Application | 6-13 |
| Figure 6-4b | Place and Transition references to figure 6-4a | 6-14 |
| Figure 6-5 | Pallet flow direction for the conveyors in the FAC example 2 | 6-15 |
| Figure 6-6 | Schematic diagram for the conveyors flowing direction of the FAC example 2 | 6-15 |
| Figure 6-7 | Iterations for the Tabu search for example 2 | 6-18 |
| Figure 6-8 | Control statement for example 2 | 6-19 |
| Figure 6-9a | CPN models for configuration 2 in Tabu Search application | 6-19 |
| Figure 6-9b | Place and transition references to figure 6-9a | 6-21 |
| Figure 6-10 | Simplified Tabu search rules to the FAC example 3 | 6-25 |
| Figure 6-11 | Iterations for the Tabu Search | 6-27 |
| Figure 6-12a | CPN models for example 3 in Rule-based application | 6-29 |
| Figure 6-12b | Place and Transition references to figure 6-12a | 6-30 |
Figure 6-13  Pseudo Code for the Tabu Search Algorithm in ML

Figure 7-1  Apriori algorithm application in this project
Figure 7-2  Data file extracts by DesignCPN
Figure 7-3a  Codes for the tokens which show in the exported simulation file
(Part 1)
Figure 7-3b  Codes for the tokens which show in the exported simulation file
(Part 2)
Figure 7-4  Additional code designs for FAC example 3
Figure 7-5  Data extracted from the CPN simulation by Data Mining Algorithm
Figure 7-6  ANOVA for scheduling agent
Figure 7-7  ANOVA for pallet identity agent
Figure 7-8  ANOVA for Barcode agent and Conveyor Control agent
Figure 7-9  ANOVA for Robot Agent
Figure 7-10 Means of token counts for the agents in this project
Figure 7-11 Example association rules for the example
Figure 7-12 Association rules for the two agents interface
Figure 7-13 Example association rules for the example 1 with tabu search
Figure 7-14 Association rules for the two agents interface
Figure 7-15 Example association rules for the example 2 without Tabu search
Figure 7-16 Association rules for the two agents interface
Figure 7-17 Example association rules for the example 2 with Tabu search
Figure 7-18 Association rules for the agent interactions between two agents
Figure 7-19 Summary for the example referring to this chapter’s sections
Figure 7-20 Labels for the figure 7-21
Figure 7-21 Supports and confidences for the examples
Figure 7-22 Association rules for the examples
Figure 7-23 Summary for the supports and confidences to figure 7-22
Figure 7-24 The flow chart for the sequence input procedures
Figure 7-25 Scheduling Agent’s Graphic Interface after Agent’ Graphic Interface
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-26</td>
<td>Pallet Identity and Barcode Reading Agents’ Graphic Interface</td>
<td>7-49</td>
</tr>
<tr>
<td>7-27</td>
<td>Pallet is conveyed passing through a sensor</td>
<td>7-50</td>
</tr>
<tr>
<td>7-28</td>
<td>The example shows the robot processing file sending to the robot controller</td>
<td>7-51</td>
</tr>
<tr>
<td>7-29</td>
<td>Agent Interaction Diagram</td>
<td>7-52</td>
</tr>
<tr>
<td>7-30</td>
<td>Timing Frame to Illustrate Agents’ Works</td>
<td>7-52</td>
</tr>
<tr>
<td>7-31</td>
<td>Flow Chart for the Robot Processing Steps after Order Received by the Scheduling Agent</td>
<td>7-54</td>
</tr>
<tr>
<td>7-32</td>
<td>Conveyor Control Agent’s GUI during Operation</td>
<td>7-55</td>
</tr>
<tr>
<td>7-33</td>
<td>Flow Chart for the FMS Control System after the End of Robot Process</td>
<td>7-55</td>
</tr>
<tr>
<td>7-34</td>
<td>Agent Interaction Diagram</td>
<td>7-56</td>
</tr>
<tr>
<td>7-35</td>
<td>Timing Frame to Illustrate Agents’ Works</td>
<td>7-56</td>
</tr>
<tr>
<td>7-36a</td>
<td>Generic robot agent class model</td>
<td>7-58</td>
</tr>
<tr>
<td>7-36b</td>
<td>Modify robot agent in example 2</td>
<td>7-59</td>
</tr>
<tr>
<td>7-37</td>
<td>Modified class diagram for the robot agent</td>
<td>7-60</td>
</tr>
<tr>
<td>7-38</td>
<td>Sequence diagram for the robot agent in the generic framework</td>
<td>7-61</td>
</tr>
<tr>
<td>7-39</td>
<td>Sequence diagram for the new robot agent in example 2</td>
<td>7-62</td>
</tr>
<tr>
<td>7-40</td>
<td>Collaborative diagram for the robot agent in the generic framework</td>
<td>7-63</td>
</tr>
<tr>
<td>7-41</td>
<td>Collaboration diagram for the new robot agent in example 2</td>
<td>7-64</td>
</tr>
<tr>
<td>7-42</td>
<td>Implementation code for the new robot agent</td>
<td>7-65</td>
</tr>
<tr>
<td>7-43</td>
<td>Modified class for the new robot agent example</td>
<td>7-66</td>
</tr>
<tr>
<td>7-44</td>
<td>The class diagram for the scheduling agent in the generic control system</td>
<td>7-68</td>
</tr>
<tr>
<td>7-45</td>
<td>Class diagram for the new Scheduling agent in example 3</td>
<td>7-69</td>
</tr>
<tr>
<td>7-46</td>
<td>Sequence diagram for the Scheduling agent in example 2</td>
<td>7-71</td>
</tr>
<tr>
<td>7-47</td>
<td>Collaborative diagram for the Scheduling agent in example 2</td>
<td>7-72</td>
</tr>
<tr>
<td>7-48</td>
<td>Tabu class in Java</td>
<td>7-73</td>
</tr>
<tr>
<td>7-49</td>
<td>Form1 class of the scheduling agent in Java</td>
<td>7-74</td>
</tr>
</tbody>
</table>
**LIST OF GRAPHS**

<table>
<thead>
<tr>
<th>Graph 7-1</th>
<th>Graph for supporting plot threshold against no. of rules with varies confidence thresholds</th>
<th>7-18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph 7-2</td>
<td>Graph for Supporting Count Plot against time with vary confidence</td>
<td>7-18</td>
</tr>
<tr>
<td>Graph 7-3</td>
<td>Graph for Supporting Count Plot against time</td>
<td>7-19</td>
</tr>
<tr>
<td>Graph 7-4</td>
<td>Statistical report for the machining centers</td>
<td>7-37</td>
</tr>
<tr>
<td>Graph 7-5</td>
<td>Product A and B throughputs for the FAC model simulation</td>
<td>7-38</td>
</tr>
<tr>
<td>Graph 7-6</td>
<td>Statistical report for the machining centers for the modified FAC example</td>
<td>7-39</td>
</tr>
<tr>
<td>Graph 7-7</td>
<td>Product A and B throughputs comparison</td>
<td>7-40</td>
</tr>
<tr>
<td>Graph 7-8</td>
<td>Numbers of occurrences for agent tokens for the three agents</td>
<td>7-40</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Advanced manufacturing systems, such as flexible manufacturing systems (FMS), are complex manufacturing systems which demand a robust control system for the planning, control and monitoring functions. In particular, the manufacturing control system has to cope with product and process changes. These FMS are mostly established with the conventional centralized and hierarchical control architectures. A central system controller is responsible for the major operation planning, control, and monitoring tasks for all components of the system. However, the control software of a hierarchically controlled system is in general not flexible enough and is difficult to develop and implement. It is difficult to modify the tight working relationship between the system components to cope with changes in the system configuration.

The rapidly increasing complexity of this kind of complex manufacturing systems has prompted considerable efforts in the development of intelligent software for the planning, scheduling, and control of manufacturing systems. Indeed, today’s and future manufacturing systems demand autonomy, distribution, and flexibility. In recent years, researches on computer integrated manufacturing have been focused on the applications and developments of autonomous control (Huang et al., 2000), distributed control (Yeung et al., 1995; Okubo et al., 2000; Hiroki et al., 2000) and intelligent manufacturing (Rao et al., 1997; GONG et al. 1997). For example, Bruccoleri et al. (2003) demonstrated a flexible automation system with agility capability to handle system uncertainties. Other approaches such as conceptual and system integration of the manufacturing system (Mieczystaw et al., 1997; Harish et al., 1997; Gong et al., 1994) have also been proposed.

Recently, researchers have identified the potential to apply distributed artificial intelligence (DAI) in solving various complex manufacturing system problems with multi-agent systems (MAS). The approach is concerned with the distribution and coordination of knowledge and actions in manufacturing environments involving agents.
A typical multi agent-based manufacturing control system is composed of autonomous agents which communicate and collaborate with other components and perform their respective control functions distributedly. MASs are designed to decentralize the control of the manufacturing systems, so as to reduce the complexity and to increase the flexibility and adaptability of the systems (Basran et al., 1997; Nagata et al., 1994; Franklin et al., 1996; Noronha et al., 1991). Such systems can also help to enhance fault tolerance.

To study the application of MAS in FMS control, Fan and Wong (2003) developed an agent-based infrastructure for the control of a flexible assembly cell (FAC). The proposed agent-based FAC control system comprises a collection of agents implemented in a distributed control network. Using the agent-based approach, it is possible to decompose the complex control program into several smaller, autonomous problems. The agent-based system was designed to decentralize the control of FAC to reduce the complexity and to increase the flexibility of the systems. The approach of the agent design was based on the object modeling technique (OMT). According to the proposed control architecture, a standard agent template for the manufacturing cell was also proposed. This agent-based FAC control system architecture (Fan et al, 2003), with the in-built generic agent template and communication protocol, is adopted as the basic framework in this study on the modeling and evaluation of MAS-based FMS control systems.

1.1 Research objective

Formal system design and analysis methodologies can speed up the development cycle for the complex FMS control system. With the help of agents, a complex manufacturing control system can be decomposed into various smaller problems handled by individual agents. This modular functionality structures is easy to modify and other agents are not affected by the modification of a specific agent.

An agent-based FMS control system is actually a complex software system, it is necessary to establish formal modeling methodologies to support its design, development and analysis. In this thesis, the object-oriented methodology UML
(Unified Modeling Language) (Boggs et al, 2002) is used in the development of the agent-based FMS control system. In addition, CPN (Colored Petri Net) models are used to represent the dynamic interactions of agents in agent-based FMS control systems.

The main objective of this thesis is to develop a new hybrid approach for designing and analyzing agent-based FMS control systems. To this aim, UML models are constructed to represent the static and dynamic perspectives, CPN models are then established for the representation and evaluation of dynamic interactions. With respect to the performance evaluation results obtained in the CPN simulation run, the UML models can be modified for fine-tuning of the control system.

Specifically, the sub-objectives of this research are:

- To establish a hybrid UML-CPN modeling approach to represent the static and dynamic behaviors of agent-based FMS control systems.
- To establish a mechanism to map UML models onto CPN models and vice versa.
- To construct CPN models for representing the dynamic agent interactions in the agent-based FMS control system.
- To establish a generic agent-based FAC control system for a simple flexible assembly cell (FAC) as an FMS example. This generic agent-based FAC control system can then be modified to establish the respective MAS-based control systems for different FAC system configurations.
- To evaluate the performances of agent-based FMS control systems with CPN simulation.
- To study the behaviors of the agent-based FMS control system under different scheduling rules. A tabu search-based algorithm for pallet allocation is developed and implemented.
- To establish a data mining tool for the extraction of meaningful information from results obtained in CPN simulation runs.
1.2 Thesis organization

In chapter 2, a literature review is presented. This chapter presents the background introduction on agents and applications of multi-agent systems in manufacturing. Formal specifications for the manufacturing systems are also introduced in this chapter. Chapter 3 provides the overviews of a generic FAC configuration. This framework is used as the generic FAC framework throughout the thesis. Afterwards, the application of the UML is applied into the construction of this framework in chapter 4. The dynamic model methodology using Color Petri Net is then introduced in chapter 5. CPN is used to supplement the UML model to analyze the dynamic system behavior. Chapter 6 illustrates example applications for the generic control system. Chapter 7 presents the dynamic analysis issue for the application examples. This chapter also illustrates the practical application examples for the generic control system. Finally, The final chapter summarizes the proposed approach, its contribution and recommendations for further research.
CHAPTER 2
LITERATURE REVIEW

In general, a flexible manufacturing system (FMS) consists of CNC machines, robots, material transfer system, automated guided vehicle (AGV), automatic storage and retrieval system (AS/RS) and loading/unloading stations. The FMS control system is responsible for scheduling and planning of production operations, control and monitoring functions. This is a complex system which should control and maintain the harmony of the manufacturing devices. FMS control systems are usually established with the conventional centralized or hierarchical control architecture. This kind of traditional control structure for manufacturing system is not capable to cope with nowadays’ flexible manufacturing requirements (Ryu et al., 2003). Any change in the FMS such as change of devices or change of processes will lead to rewriting of the control system (Ramirez-Serrano et al., 2003). Its rigid structure and the old fashion design architecture are not sufficient for intelligent system design (Adamou et al., 1998; Duffie et al., 1996; McDermott et al., 1997). In recent years, agent technology has been widely used in control frameworks for FMS (Choi et al., 2000; Zhao et al., 2000; Aerts et al., 2002).

2.1. Agents and multi agent systems
One of the main applications of software agents is in the development of distributed artificial intelligent agents (DAI) system. DAI system is a collaborative system by grouping intelligent elements to perform the system task. Intelligence, distributed task abilities and fast response to events happening in autonomous systems are the crucial successful factors in DAI applications. Smith et al. (1981) defined a DAI problem solving architecture which is one of the earliest DAI applications in problem solving. The problem solvers in the DAI system make collaborated decision according to the sub-problem results. Agent-based DAI applications constitute from a group of intelligent entities (agents) with communication ability (Yin et al., 2003; Cetnarowicz et al., 2001) and agents perform tasks in an autonomous way.
Intelligent agent design is a new paradigm for software development (Jennings *et al.*, 1998). Without direct human intervention, an agent controls its action or communicates with other agents via computer network. This abstraction of agent communication structure widely applies in the distributed network (Huang *et al.*, 2002). Agents share information through the network and use the abstraction of intelligence to resolve incoming requisition (Tian *et al.*, 2002). The DAI capability of multi-agent structure enhances integrability, configurability, adaptability, extendibility, agility and reliability of computer integrated systems (Shen *et al.*, 2000; Sousa *et al.*, 2003).

A multi-agent system (MAS) solves problem according to the prescribed global objective (D’Ambrosio *et al.*, 1996). The goal directed agent entity is able to support robust FMS applications because agents can make decision according to the dynamic changing environment from their inference engine (Ferber 1999). It can also enhance the performance of complex system by distributing complex tasks to other available agent resources (Kim *et al.*, 1999). An example is the prototype multi-agent system with short-term forecast ability proposed by Hadeli *et al.* (2004). This MAS is able to handle change and disturbance in manufacturing control, it attains optimized solution from the forecasts by using the congestion free system objective.

Evolution of agent applications has been extensively applied to enterprise application with shop floor integration since 1990s. Benefits of agent technology include the agent’s fast response ability and agents can communicate in bi-directional way without geographical limitation (Walsh *et al.*, 2003). The Internet is now the most convenient way for agent communication. In the example illustrated in Arasten *et al.* (1996), the enterprise agent and shop floor agent are integrated in different geographical locations. In another example, agents are used to solve the scheduling problem via internet (Aydin *et al.*, 2004).

**2.2. Application of agent-based systems**
Rapid development of the internet promotes the conversion of business activities towards computerization. Traditional enterprise is a labor-intensive structure and labor cost becomes the major overhead for business processes. Enterprises therefore
attempt to investigate new technology to reduce the overhead. Multi agent application examples in various fields of study are focused on software implementation in computer-related systems. Examples are enterprise application, decision support system and intelligent engineering application etc. For example, virtual enterprise applies software agents to its business operations to replace human intervention (Gou et al., 2003). This virtual enterprise application enhances the competitiveness by sharing information between business partners. The operation agent then performs contract review, resource analysis for the system automatically. The decision support system in a virtual enterprise helps to make complex decisions instantly without human intervention (Carrie 2000). The enterprise application also enhances more business activities to supply chain by using information technology (Walsh et al., 2003).

An agent-based control system is an autonomous computer software system designed for decision making in intelligent control system. Jiang et al. (2003) proposed an example MAS system for decision making by using multi-goals and mult-variant approach. Vagin et al., (2004) proposed an agent based intelligent manufacturing system for rule searching in order to find the optimum decision. These two examples demonstrated that agent-based control systems are suitable for iterative problem solving heuristics. There is an increasing number of agent research in manufacturing environment (Whinston et al., 1997). Jennnings (1998b) proposed a roadmap for agent applications. He also provided some guidance to agent applications in a complex manufacturing environment.

Barad et al. (1997) presents flexibility measurements for manufacturing systems. The reliable, maintainable and extensible design framework of agent structures is suitable for engineering system applications (Goonetillake et al., 2002; Harding et al., 2001; Montreuil et al., 2000). Distributed structure of the intelligent agent framework helps the decomposition of large and complex problem into sub-problems. Each of the sub-problems can then be handled by individual agents locally. Mobile agents have been found to be useful for this kind of applications and various projects have been reported (Dalmeijer et al., 2000). Fox et al. (2000) proposed an agent system with mobility in supply chain applications where the system requires frequent
communication and cooperation between agents. Mobility of agents enhances the system performance by sending the required agent to anywhere in the system. An agent is sent to specific area upon requisition, for example, a contingent agent will be sent to an error occurring area in the agent-based system to handle emergency condition.

2.3. Manufacturing applications by using agent

There have been a number of agent-based systems developed for manufacturing applications. Researchers proposed to establish the theory of MAS with the aim to optimize the production plan and schedules dynamically (Miyashita et al., 1998). In other examples, manufacturing resource conflicts are resolved by negotiation based multi-agent systems (Berker et al., 1996). The multi-agent framework is also applied to material handling system and warehouse system design in shop floor application (Ito et al., 2002). Koo et al. (2002) proposed a multi-agent framework for AGV application in manufacturing systems. These multi-agent control examples are suitable to the dynamic and unpredictable environment in manufacturing systems (Sousa et al., 2003). The multi-agent framework overcomes the deficits of traditional control structure by providing stability control and interaction stability in the dynamic changing environment and significant advantages possessing by multi-agent systems are:

- Multit-agent system resolves problems in a distributed form. Problem decomposition prevents specific controller from overloading by distributing the decomposed sub-problems to other available resources in the system.
- Flexibility of multi-agent control system obtains by the modification of agent interaction to the required changes instead of rewriting the whole control in traditional control structure.
- Multi-agent structure provides encapsulated agent structure to the control system. Additional or removal of agents or agent objects will not affect existing elements in the system which they do not belong to the changing entity.
In agent-based control system applications, the MAS composes three prospective research domains, they are communication, coordination, negotiation and intelligence.

**Communication** –
Multi-agent systems for DAI applications have to distribute tasks through network. The internet protocol design for communication is therefore very important. Soudani et al. (2002) demonstrated the importance of a reliable data exchange protocol in distributive system environment. The real time control system in this example has a specific internet protocol in order to prevent information overflow. Moore (1999) presents another agent protocol for the system application. Heikkila et al. (2001) presents a list of requirements for agent communication protocol in DAI system. They also discussed the essential of standardized communication mechanism between agents. Standardization of agent protocol enhances the efficiency of information exchanges and it can prevent agent systems from redundant information overflow.

**Coordination** -
The stability of a robust and reliable system relies on the disturbance handling ability of agent-based control system. Errors or emergent events may appear anytime during system operation. Agent-based systems equipped with contingency plans are able to improve the overall system performance. A systematic design approach is adopted to manufacturing systems today (Ingemansson et al., 2004). Error detection and recovery ability is one of the critical multi agent research areas (Huang et al., 2002).

**Negotiation and intelligence** –
Fast response to resource allocation is one of the factors to enhance system performance. The speed of decision making from filtered data is determined by data extraction heuristic process (Li et al., 2001). Sorting, filtering and evaluation abilities of MAS are also receiving attention in research application (Tian et al., 2002). These applications filter out unnecessary information before systems decide on suitable plans to react. The filtering process also prevents agents from
information flooding which affects the quality of decision making. The filtered data is ready to be interpreted by intelligent or knowledge-based application. Nwagboso et al. (2004) demonstrated an MAS in intelligent application. Jennings et al. (1998) introduced an agent system with intelligence and fast response ability in manufacturing application. To explore the idea of task distribution by using agents, Usher et al. (2003) illustrated an example on an agent-based manufacturing control system with negotiation ability. Negotiation based bidding mechanism for resource allocation to tardy jobs has shown noticeable improvement in shop floor applications (Ushler, 2003).

An agent-based manufacturing system is capable to apply in medium-to-small quantities manufacturing system (Ounnar et al., 2004). Majority of the agent projects are applied to this size range of manufacturing control system (Villa et al., 2002; Ito et al., 2002; Reis et al., 2001; Shen et al., 2000).

Researchers have proposed and illustrated agent-based control applications to FMS. (Dumitrache et al., 2000; Friedrich et al., 1999; Tso et al., 1998; Siemiatjowski et al., 1997; Parunak 1990). Agent models are built in standard methodology and then implemented into real system. The control systems are able to attain flexibility, reusability and fast response to both internal and external uncertainties in shop floor environment by its distributive control structure by using the standard modeling methodology (Monostori et al., 1999). Negotiation based manufacturing control systems (Maione et al., 2003) and multi agent systems with learning ability (Tacla et al., 2003) are example in resolving complex manufacturing activities.

Some agent-based manufacturing system applications are integrated with other AI techniques, for example, genetic algorithm in FMS rescheduling application (Yang et al., 2003), automated assembly plan generation (Smith et al., 2003) and neural network design and implementation FMS approach (Abdelhameed et al., 2002). As well as agent-based manufacturing application, a successful multi agent based FMS implementation also involves three levels of interaction for the control system. They are the interaction level, intelligent level and network communicating level for the control system. The interaction level improves interaction between agents in the
control system, the intelligent level promotes autonomous system, the network level enhances the communication capability of agents (Gong et al., 1997).

2.4. Agent-based manufacturing control system

2.4.1. Traditional control architecture and its evolution

Traditionally, complex manufacturing systems such as FMS are often equipped with the conventional centralized or hierarchical control structures. Edwards et al. (1998) addressed that the top-down hierarchical structural approach limits the flexibility of shop floor control systems. The rigid structure is insufficient to cope with information exchange between low level system controllers.

To overcome the disadvantages of these traditional manufacturing control structures, it is necessary to introduce new types of manufacturing control structures for FMS control systems. Intelligence of the control system helps to improve the efficiency of FMS to system activities and exceptional event during system operation (Yan et al., 1998). Therefore, modern design of FMS control systems should possess intelligence, respond rapidly to exceptional events such as machine failure, and reconfigure resources to product and process changes (Gunasekaran, 1997; Manfredi et al., 2003). Shewchuk et al. (1998) presented prescribed standards for the flexibility requirement in modern manufacturing control system. Holon manufacturing system (HMS) is one of the earliest control paradigms provides flexibility in autonomous control in manufacturing systems.

2.4.2. Holonic Manufacturing system architecture

The ultimate goal of agent-based FMS researches is to establish an FMS control system which is error prone, flexible to dynamically changing environment and ease to maintain. Holonic Manufacturing system (HMS) is one of the earliest structures attaining these requirements in FMS application (Jarvis et al., 2003). McFarlane et al. (2000) described Holon as “An Autonomous and co-operative building block of a manufacturing system for transforming, transporting, storing physical and information objects”.

2 - 7
A number of research institutions and industrial companies from Europe, North America and Japan are participating in a HMS consortium. The consortium is working on collaborative HMS projects which most of the projects are equipped with internet-based mobile framework (Feltcher et al., 2003). The HMS consortium suggests ways to implement the manufacturing system by using modular units. These procedures apply the integration of the entire range of manufacturing activities from order booking through design, production and marketing and these activities are implemented with flexible manufacturing system in order to establish an autonomous manufacturing system.

An HMS framework provides a high degree of coordination between manufacturing facilities in different geographical locations. The HMS provides flexibility that does not exist in traditional control structures, especially the ease of implementation of new introductory components to the control system (Chirn et al., 2000). It is straightforward to add new holons to the system. A new holon is equipped with proper communication interfaces and it can work with existing holons in the HMS system. This new holons should also be interconnected through the communication network to cooperate with existing holons in order to achieve the system goal.

Researchers have identified the importance of instinct characteristics of “Holon” such as reconfigurability and flexibility to dynamic changing environment in general FMS control system. Emphasis has been placed on the improvement on interaction ability between shop floor devices. Cheung et al. (2000) and Giret et al. (2004) introduce examples of FMS having complex negotiation structure with high communication ability.

It is widely accepted that holons in FMS consists of an information part and the physical processing part. It is therefore a structure combining both software and hardware control technologies for a manufacturing application, whilst in general, agents are software agents which are firstly found in software application (Sousa et al., 1999). For manufacturing applications, the term “Agent” is sometimes mixed with the word “Holon” (Kotak et al., 2003; Arai et al., 2001). In fact, both these two
structures comprise intelligent elements which they communicate and work towards a global system objectives.

2.4.3. Agent based control architecture

Agents and multi-agent systems have been applied in many FMS control system applications, particularly in the areas of scheduling management and shop-floor control. There are two important objectives of agent applications in FMS: deployment of complex activities and distribution of work loading to devices; and exceptional event and error handling in shop floor. Kouiss et al. (1997), Cavalieri et al. (2000) and Vancza et al. (2000) introduced agent-based dynamic scheduling for FMS. Seo et al. (2002) presented a new FMS scheduling methodology for different part types by using an agent-based control system. Another example shows the specification of agent design and agents interaction by using model representation in Choi et al. (2000).

An agent-based FMS control system requires an appropriate task deployment strategy in real time (Gourgand et al., 2003). This strategy responds to events in system operation. The agent-based system should closely monitor the autonomous manufacturing operation; the control system should make sure the operation should run according to the plan. Agents should react to any delay or exceptional events and prepare alternative contingency strategy with other agents in respect to the delay. Because of these reasons, interactions between agents should be well planned. Static representation of agent attributes and dynamic representation of agent activities require detail specification in the design stage (Szczerbicki, 1995). Examples of deployment strategy in agent applications are shown in Lee et al. (2000), Gologlu (2004), Adzakpa et al. (2004). Mehrabi et al. (2002) predicated that agent-based control systems with reconfigurable ability will be the next evolution for FMS control system.

The capabilities of agents to react to system changes and exceptional events are an emerging research area of FMS control systems. Exceptional events will interrupt the resource allocation of a manufacturing control system. It is necessary for the agents of a system to adapt to any changes which maintaining productivity in reality.
In order to prevent system break down, Lawley et al. (2001) presented examples for common deadlock avoidance in shop floor control. They proposed agents with event handling capability to overcome the common deadlock problem. Machine breakdown is another common issue in manufacturing. Inamoto (1999) proposed an agent-based control system to monitor and control the workflow automation. Inappropriate line balancing of a production system will result in bottleneck problems and possibly system halt. Sousa et al. (1999) applied a new contracting methodology in dynamic scheduling of manufacturing order between agents to resolve the bottleneck problem.

Agent-based FMS control systems are mostly multi-agent systems. A multi-agent system is a collection of heterogeneous or homogeneous entities possessing different talents and control process in a dynamic, non-deterministic complex system. Agents in the multi-agent system may have different or same capability in specific functionalities. This capability limits agents to specific control purpose and ensures a task is to be distributed to the right agent. The control strategy and data are decentralized among agents in the MAS system. Agents exchange information in an asynchronous way as required.

A multi-agent system performs the system activities in cooperative and autonomous ways by agent collaboration. Since the advent of networking technology, researchers have explored flexibility of the agent application to multi-agent structure in FMS control systems (Roy et al., 2001; Lee et al., 2003; Anussornnisarn et al., 2002). Reis et al. (2001) proposed a coordination mechanism for multi-agent scheduling in highly dynamic changing environment. Zha et al. (2003) proposed a framework for multi-agent decision making system in a complex FMS. Kornienko et al. (2004) designed a multi-agent system to solve a complex shop floor job assignment problem.

In order to attain the generic agent requirement in manufacturing applications. A formal methodology of generic agent structure is necessary in a large and complex system. Villa et al. (2002) presented detailed specification of the intelligent agent requirements for a complex system. This formal methodology helps to enhance both
the static specification and dynamic interaction between agents in more precious way during design stage. Design and analysis of the agent-based FMS control system to respond to the real time shop floor environment are difficult tasks. The modular design approach of agent systems increases the flexibility of the FMS to cope with the dynamic changing environment (Jionghua et al., 2003). The modular design format is easy to modify but at the same time does not affect other existing components in the control system. The object-oriented modeling methodology is commonly used in this aspect to represent the agent structure in modular representation.

2.5. Object-oriented modeling of FMS control system application

Graphical and visualized design modeling is a familiar form of models for the design of sophisticated systems (Berio et al., 2000). Dynamical information exchange of the target system can be shown clearly in symbolic charts (Laing et al., 2002). The closest and detail representation of most of the modern graphical representation models are now also traceable. These models are easy to modify and maintain.

The ICAM DEFinition methodology (IDEF) is a system development method adopted by the Department of Defense in the USA. IDEF has been adopted as the modeling methodology for the design of manufacturing control systems (Cho et al., 1999; Dorador et al., 2000). The model is able to represent tight relationships among entities in FMS control systems but it misses most of the dynamic information representation between entities (Cho et al., 1999; Bauer et al., 1991; Kim et al., 1993).

To overcome the defects of IDEF in dynamic system representation, the object-oriented modeling (O-O) method has been adopted in the design of complex models such as FMS control systems. The O-O method is an appropriate modeling and implementation methodology to represent systems with flexibility and real time response. The O-O approach has been adopted in the development of agent-based FMS control systems (Fraile et al., 1999; Reveliotis, 1999; Gambin et al., 1999). O-O models are able to promote expandability, reusability and modifiability in the design of FMS control system. It also reduces the design complexity of FMS control software and increases the reusability of existing system functions.
With O-O modeling, elements of an FMS system are represented in abstraction by object relationships, class attributes and behaviors of the physical components or operating functions. O-O models are useful in understanding FMS control system specification through static model and dynamic information exchanges show in activity diagrams. Moreover, it provides a good communication tool in visual formats to both experts or non experts. Moreover, the object-oriented modeling tool provides a simulation testbed for evaluating the performance of the virtual design framework during the development stage (Law et al., 2000). Flexibility is also achieved by modification of the required dynamic models while the other models remain unchanged (Naso et al., 2004).

2.5.1. Unified modeling language (UML)
The Unified Modeling Language (UML) is one of the industry-standard for specifying visualizing, constructing and documenting software systems. One of the goals is to unify various previous O-O modeling languages into a common standard (Law et al., 2001). Based on the UML methodology, using the fundamental ideas of objects, classes, methods, messages, encapsulation, inheritance, polymorphism etc., system designers are able to design and document a complex system with many simple systems in O-O structures. This can help to reduce the effort in designing such complex systems.

UML consists of a set of diagrams in order to represent a complex system in diagrammatic notation. The diagrams are especially useful and suitable for real time distributed control applications (Nett et al., 1992). The UML provides detailed specification and functionality information to the control system. For manufacturing system applications, the UML is able to present three important perspectives for system representation. They are the models of use case, static diagrams and dynamic models(Kim et al., 2003). Static diagrams provide graphical representations to show system components and their relationships. Use case diagrams show the user requirements of the target system and display the events for the static diagrams. Use case and dynamic diagrams show event happening and they are complementary with each other. Dynamic diagrams for the system show system behavior over time to
specific objects in prescribed event. And use case provides graphical view for system activity and detail of use case’s event represents by dynamic diagram. The three UML models provide a complete description for complex systems. These models also help system implementer in the following aspects:

- Identify system requirements and way of information exchange
- Reduce possible system redundancy
- Design and review system in visualize format which is easy to understand. Errors and problems can be identified before system implementation.
- The graphical representation provides a communication tool for system designers and users.

O-O modeling method and advanced O-O representation by UML are commonly used to construct agent frameworks. System designers design agents in the platform by using UML graphical views. O-O entities like objects, classes can directly convert from the design of the visual UML models to executable programming source codes (Wang et al., 2002).

2.5.2. Extended unified modeling language

Although UML has been widely applied in many software oriented systems, it is found that UML is lacking of liveness, safeness verification properties in real time control systems (Lavazza et al., 2001). Agent oriented design is now widely accepted in the development of real time manufacturing systems. Most of the characteristics for agents cannot be represented by existing UML models. Especially the dynamic interaction between state models of UML is not capable to represent agent models completely (Viviane et al., 2004).

The current UML is sometimes insufficient for modeling complex agent-based systems. To describe the functionality of an object, showing the acceptable path in a normal case sequence diagram is enough. However, an agent is definitely more than an object. As the agent responds according to the change of environment, the conventional sequence diagram is insufficient to include all possible alternative paths. In addition, the current UML does not have a standard way to express the interaction between agents when a new agent enters into the agent-based system environment.
In object-orientation, an object can be instantiated whenever it is needed. For agents, expression such as threading, decision making and agent discovery are common properties. These features cannot be expressed completely through common UML views. Thus, researchers are now trying to find new paths to extend UML for agent-based representations. Recently, UML extensions such as AUML and AIP (agent interaction protocol) have been proposed by researchers from the agent community (Odell et al., 2000).

2.6. Design and analysis of agent-based FMS control system with UML

The UML has been used to model agent-based FMS control systems (Tsai et al., 2004; Giachetti et al., 2001; Du et al., 2001). Examples in Cheng et al. (1999), Kadar et al. (1998), Booth (1998), and Suraj et al. (1997) illustrate real time FMS control system applications by using UML design models. Specification of the system requirement of the agent-based system can be represented by the object model in UML. Inheritance and encapsulation of UML objects, and reusability of UML models increases the level of flexibility to represent agent objects. Besides, the UML models of agent objects can be directly converted into the source code of an object oriented programming language source. The programming framework thus shortens the time of programming in system implementation stage.

An FMS control system normally has its scheduling, controlling and planning algorithms. The inherence properties of agent structure enriches the reliability, flexibility of manufacturing planning and scheduling capability in dynamic manufacturing environment (Maturana et al., 1996). Reconfigurable agent structure by using object oriented model provides flexibility to change. Agent in the system is also able to allocate task to other resource dynamically during machine break down or special event happening (Trentesaux et al., 1998).

The importance of using FMS simulation brings attention in the field of analytical research (Horing et al., 1998; Caprihan et al., 1997). Simulation helps to identify the problems in target system. Alternative solutions can also be proposed after the evaluation of simulation. Production planning system (Kenne et al., 2001), Expert system for manufacturing rescheduling process (Kunnathur et al., 2004), Disturbance
analysis of Production system (Ingemansson et al., 2004) are the application examples in manufacturing by simulation.

The UML model is not adequate to represent the complete requirements of agent-based FMS control system. Detailed interactions between objects should also be evaluated before dynamic UML model integration. A new extension of the representation of the agent based control system in FAC by using UML and new analysis tool by using Petri-net for verification is presented in this thesis.

2.7. Petri Net model in FMS control system application

Petri nets were found between 1960’s to 1970’s. It is an efficient tool for the description and analysis of synchronization, communication and resource sharing between concurrent processes and real time system. Systems that are characterized as being concurrent, asynchronous, distributed, non-deterministic and stochastic can be effectively modeled and analyzed by using Petri Net (Bohez et al., 2004). Advanced extensions of Petri Net including Color Petri Net (Arjona et al., 2003), Timed Petri Net (Dotoli et al., 2004), Classical Petri Net (Kurihara et al., 2002) have been used in manufacturing applications. The Petri Net modeling approach also helps in the integration of FMS systems (Elmekkawy et al., 2003; Bohez et al., 2004; Seeluangsawat et al., 2004). Petri nets in the examples enhance the design representation, synthesis and analysis of complex FMS control systems.

The integrated UML and Petri Net modeling approach in this thesis provides an integrative framework supporting requirement description, model specification and design, model analysis and simulation, and model implementation.

A Petri Net model is a suitable design and analysis tool for real time control systems such as FMS control systems. FMS activities are represented by the token movements in dynamic Petri Net simulation environment, token activities are controlled by the probability setting in the simulation menu. Decision making in real control systems is simulated by declaration of conditional statement to control token movements. Moreover, system evaluation can be performed by tracking the token movement record. Petri Net is a symbolic and easy-to-understand design system to
evaluate events in complex systems. The token movement helps system designers to analyze the system model in a more comprehensive manner to represent both the internal and external environmental changes inside an FMS (Ahuja, 1988). The following context shows some points that should be identified by simulation

- The specific goal and activity of the component association
- The components inside the system occur concurrently or independently
- How the components are linked to work cooperatively
- The behavior model of the components inside the FMS model interactively

Qualitative features like boundness, liveness and reversibility are the desired characteristics possessed by FMS control system (Huang et al., 2001). As revealed in research works (Zhang et al., 2000; Kang et al., 2000; Jeng et al., 1998; Tamas, 2000) showing the analysis application in the FMS examples, Petri Net models are well suited to represent the synchronous and asynchronous behavior of individual entity inside the system. A Petri Net model also provides useful information showing the conflicts, precedence relations, non deterministic behavior and deadlock nodes among the FMS system.

The use of an object-oriented model and a Petri net model are complementary with each other in FMS applications (Huang et al., 2001; Chen et al., 1997). A Petri Net model is similar to object-oriented representation in the design of a system model. A Petri net represents both the system’s structural and behavioral properties for analysis. It is not only a visual representation model but also can be formally represented in a mathematical form, which clearly shows the dynamic events by using the Petri net simulation tool. After the numerical and analytical analysis with the Petri Net, interactions of the Petri net model can be used to represent the dynamic UML model of the system. There are literatures showing the applications of Petri Net models in agent interaction and evaluation (Basile et al., 2004; Kattan et al., 2004; Saitou et al., 2002).

Static model, use case model, the framework of the target system can convert into object oriented programming structure. Chen et al. (1997) showed a design example
of object oriented design by using Petri net analysis in FMS control system, where the Petri Net model with entity relationship diagram in object oriented method is used for the design of manufacturing control system. Another conceptual paradigm of Petri net model integrated with object oriented method in manufacturing system is shown in Wang (1996). He illustrated how a complex system can be decomposed and represented by smaller intelligent entities (agents) step by step and how are the flexibility and adaptability of the agents retrieving from the color Petri net analysis result. Research works including Zhang et al. (2000), Kang et al. (2000), Jeng et al. (1998) and Tamas (2000) are typical examples on applying Petri Net based applications to the design and analysis of FMS projects.

2.8. Conclusion
This chapter presents a brief overview of design structure and evolution of the manufacturing control system. Relating to this research, the agent-based applications in manufacturing and FMS in particular are reviewed. The traditional control architecture is found to be lack of the flexibility and rigid cohesive hierarchical structure is a not reusable structure. It is not able to handle both the internal and external environmental changes, especially, the verification of the modeling structure of a manufacturing control system in dynamic analysis. Agent-based control structures in manufacturing control systems have been reviewed. With the corporation of the modular type control structure, the flexibility of agents is found suitable to resolve the problems in the traditional control structure. Precise evaluation of the interactions between agents should be well defined in the design control structure. Proper modeling and analysis tools should be employed to model the control structure. UML and Petri Net are suitable methodologies for applications in the modeling and analysis of manufacturing systems.
CHAPTER 3

AGENT BASED FLEXIBLE ASSEMBLY SYSTEM

CONTROL

With the advent of computers and information technology in recent decades, researchers have proposed to improve the flexibility and the adaptability of FMS control systems with the implementation of AI-based distributed control structures (Buyukozkan 2004). For instance, AI-based control systems (Wang and Xue., 2002; Xue et al., 2001; Harding et al., 2001) are able to react to dynamic process changes to increase the flexibility of manufacturing systems. AI technology can help to promote co-operation and harmony of devices with the FMS control system (Simeu-Abazi et al., 2001; Tan et al., 2000). For example, machine fault tolerances are monitored by neural network control system and tool setups are optimized by using hidden Markov modeling system (Malakooti et al., 2000; Xu et al., 2004). Other examples include the application of problem solving heuristic to handle frequently shifting manufacturing process (Grieco et al., 2001), creation of multiple routing solution transportation problems by genetic algorithm (Zhao et al., 2001), solving scheduling problem by distributive system (Trentesaux et al., 2000) and reactive decision support system application for FMS (O’Kane et al., 2000).

Recently, agents and agent-based systems are applied in the establishment of distributive manufacturing systems. Agent technology is considered to have the potential to satisfy the requirements of integrability, configurability, adaptability, extendibility, agility and reliability in a new manufacturing control system architecture (Wang et al., 2003; Zaremba et al., 2003). It is necessary to formalize
the intelligent entities in a standardized form to facilitate the design and development of this kind of agent-based distributive control architectures (Shen et al., 2000; Lang et al., 2002; Bongaerts et al., 2000). The characteristics of object-oriented models such as inheritance, encapsulation and reusability are suitable for these requirements.

This chapter first presents one simple FAC example and then two hypothetical FAC examples. The simple FAC configuration is first presented to construct as a generic template for FAC control framework. Later on, the agent templates for the two hypothetical FAC control systems are then constructed according to the simple agent template. These three examples are case study frameworks throughout this thesis. In fact, the proposed generic template is also applicable to other FAC control systems. Characteristics of reusability, extendability and modifiability for object oriented models are revealed through application in the example (Bruccoleri et al., 2003). At the end of this chapter, agents for the generic agent-based control system are listed.

3.1. Overview of the proposed agent-based FMS control system
A Flexible manufacturing system is a complex system which comprises autonomous devices such as material transfer systems, robots, machining centers, automatic guided vehicles and loading/unloading systems. These devices integrate and form an autonomous control system for specific manufacturing purpose. For an agent-based FMS control system, agents are assigned to different functionalities and they are responsible for specific manufacturing activities inside the FMS control system.

In the coming paragraphs, three flexible assembly cells (FAC) with different
configurations are introduced. The first FAC is used to propose a generic FAC control system template. The system configuration is a simple FAC. Based on the generic FAC template, necessary control agents of a general FAC are proposed. This example shows the basic requirements on both the control software and hardware with regard to the proposed generic agent-based control system template. To explore the adaptability of the template, two hypothetical FAC examples are then introduced. The examples involve another control systems with a complex material transfer system and additional assembly centers.

3.2. FAC configurations
3.2.1. Example 1
This simple FAC example (figure 3-1 and figure 3-2) is a flexible assembly cell (FAC). The FAC configuration is used as generic template for the FAC control system. This FAC comprises two Adept SCARA-typed assembly robots and a conveyor loop with 4 conveyors forming a material transfer system for pallets transportation. A single Adept MV programmable controller controls the motions of the two robots. These robots can handle assembly operations and material transfer tasks. Stoppers are located at the front of these robots (the area is called robot processing area) and arriving pallets are clamped for robot processing. Functions and capabilities of the two robots are identical and task allocation to either one of the robots is considered by their availability. The robot controller is equipped with two serial communication ports for communication.

There are four conveyors in the FAC. They form a loop connection for pallet transportation. A Programmable Logic Controller (PLC) controls the coordination between sensors, pneumatic cylinders and signal indicators in the pallet
transportation system. It is the central processing unit for motion control of the conveyor system. Ladder diagram is used to prepare the control programs for conveyor motion control. The PLC is also equipped with a serial communication port for communication.

The FAC can house a maximum of 6 pallets at any one time. A tailor-made pallet positional detection system is established to monitor the pallet locations inside the FAC. This system consists of a set of twelve sensors in the conveyor set, thereby providing twelve locations for pallet detection. The sensor feedback is sent to the controlling computer through a digital control interface.

![Figure 3-1 Physical configuration of the simple FAC](image)

Figure 3-1 Physical configuration of the simple FAC
In the current implementation of the prototype control infrastructure for the FAC, a network of four computers is used. That is, there is not a single central control computer. The control software is established as a multi-agent control system with different agents installed in different computers via a TCP/IP network. Within the system, serial communication connections are the basic interfaces for connecting the system hardware devices and computers, while the agents communicate in the network through TCP/IP protocol. Using this approach, it is straightforward to connect the control system, and hence the assembly cell itself, to the internet to establish web-based applications.

3.2.1.1. **Product assembly in FAC example 1**

In this example, all pallets are loaded with the same product type. Therefore, the assembly task is identical for all parts in the FAC. The product is shown in Figure 3-3. The assembly operations in this example are very simple, which can be handled
by either one of the two Adept SCAR robots. In other words, the two assembly workstations in the FAC are identical. The assembly task requires only one robot, the other robot is a duplicate workstation. The operation of this example process is simple, it is:

1. A round rod is picked from hole A
2. Move the rod from hole A to the area above hole B
3. Insert the rod to hole B

Figure 3-3 Assembly operation for example 1

The actual task is to pick up the cylindrical rod from hole A and then insert into hole B (Figure 3-3). The operation of the assembly process is 10 sec.

Figure 3-4 Work Example for the FMS Control System for FAC example 1
**Flow of pallet in example 1**

Figure 3-5 depicts the material flow in the FAC. The material transfer system comprises four conveyors marked A, B, C and D. Conveyor A is also the loading/unloading point. Pallets will be transferred to conveyor B after being loaded at A. Once the robot process finishes, the corresponding loaded pallet will be transferred from conveyor D back to A and then unloaded in this area. A loaded pallet awaiting robot processes will circulate around continuously until it receives the assignment of a process from the robot agent.

![Figure 3-5. Flow of pallet in generic FAC example](image)

**3.2.1.2. Configuration constraints of example 1**

Constraints of the flexible assembly cell should be identified before integration. Only 6 pallets are allowed to co-exist in the FAC at the same time. The working area for each robot restricts the robot movement inside the robot working envelop. The robot payloads are also limited. TCP/IP and serial communication ports are the two main communication media between agents. Error detection of network status has also been built to prevent the loss of data transfer via the TCP/IP or serial communication.

**3.2.2. Example 2**

This example involves a hypothetical FAC. The configuration of this hypothetical
FAC is more complex than the one in example 1. This FAC contains 3 assembly robots (They are named 1, 2 and 3 respectively in figure 3-6), and they are identical in functionality and capability. The conveyor system consists of 6 conveyors which link up the assembly workstations and the AS/RS (figure 3-6 and figure 3-7). For each of the assembly workstation, a stopper is located at the robot processing area to clamp a pallet for processing.

Similar to example 1, the programmable logic controller (PLC) coordinates sensors, pneumatic cylinders, signal indicators activities in the material transfer system and it is also equipped with a serial communication port for computer communication.

The FAC in this example can house a maximum of 10 pallets at any one time. A
tailor-made pallet positional detection module with more sensors (twenty sensors in this example) is also established in order to monitor pallet locations. The sensor feedback is sent to the computer through the digital control interface at the agent computer.

![Figure 3-7 Top view of example 2](image)

This example utilizes four computers for control purpose and the control software is also established as a multi-agent system with different agents installed in the computers within the LAN. However, agent computers are linked to their respective shop floor devices via serial communication interfaces.
3.2.2.1. **Product to be assembled by FAC example 2**
A simple product is also proposed to assemble in this example. The product type is the same as in FAC example 1. The operation detail of the assembly process is described in section 3.2.1.1. This product is also the product assembly in example 1. The three robots installed in this example are also able to assemble this product with the same processing rate. The operation time for each assembly is 10sec.

3.2.2.2. **Flow of Pallets in example 2**
The pallet transportation system in this example is more complicated when it is compared to that of example 1. Instead of a simple loop conveyor system, the material transfer system for this setup has 1 buffer line for pallets as shown in figure 3-7 (named conveyor E). The conveyor loop ABCD moves in clockwise direction to go through the working envelops of robots 1, 2 and 3. Conveyor E acts as the buffer line and it can transfer pallet in a bi-direction manner. Conveyor F is also a bi-directional transfer system where pallets are loaded or unloaded from the automatic storage and retrieval system(AS/RS).

3.2.2.3. **Configuration constraints**
This example permits twelve pallets to co-exist at any one time. Assembly processes are restricted to robot working envelops of the assembly robots. Productivities for the three robots are identical and it is assumed that the same quality of products will be achieved at the same production rate by these robots. Error detection of network status are also built to prevent the loss of data transfer by the TCP/IP communication.

3.2.3. **Example 3**
The configuration of this FAC is different from examples 1 and 2, the FAC is
established to test the modeling flexibility to FAC configuration. Four conveyors are installed in this FAC as shown in figure 3-8. This FAC system is assumed to have infinite raw material supply at the raw material and infinite removal rate for finished pallets.

The FAC configuration in this example is similar to that of example 1 but this FAC comprises more machines. Machines are categorized into three machine types: milling machines, grinding machines and drilling machines. Four machines for each machine type are installed in the cell, therefore, twelve machines are available.

In the cell, the four milling machines are identical. They are able to operate similar machining processes at the same identical processing rates. For the grinding machines, they are not identical with each other. Their machining process orientations are slightly different in configuration. Two grinding machines are vertically oriented and the other two grinding machines are horizontally oriented. For the drilling machines, two of the four machines are also vertically oriented and the other two machines are horizontally oriented. Detail locations for the twelve machines are shown and listed in figure 3-9.

The machining centers are able to produce four different product types assigned by the scheduling agent. The label “loading/unloading in figure 3-9 is the loading and unloading area for this FAC example. Similar to the previous example, the PLC coordinates sensors, pneumatic cylinders, signal indicators activities in the material transfer system and it is also equipped with a serial communication port for computer communication.
In this example, the FAC system can house a maximum of twelve pallets at any one time. This is also the maximum number of manufacturing processes coexisting at the machining centers. Pallet information is also located and monitored automatically by the signals feedback from sensors and barcode system in the FAC, and the information is sent back to the computer through the digital control interface at the agent computer.

This FAC control system is implemented with four computers for control purpose and the multi-agent system with different agents are linked up together within a LAN by the agent computers. As well as the previous examples, these agent computers are linked to their corresponding shop floor devices via serial communication.
Figure 3-9 Top view of the FAC example 3
3.2.3.1. **Products to be produced by FAC example 3**

In this example, the FAC is engaged in the production of four product types. Each of the product types is required to be processed by three machines. In this FAC example, raw materials with identical size are used. A new workplace is mounted onto a pallet before this pallet is loaded into the cell at the loading and unloading area.

*Product 1*

Figure 3-10 shows the first product for this FAC example:

![Figure 3-10 Product 1](image)

To manufacture this product, it involves three machining processes. The raw material is firstly milled into required thickness. Afterwards, the surface at the top of the pallet requires better surface finish according to its product specification, the grinding machine is then responsible for this operation. The machining times for product 1 are listed in figure 3-11.
Step 1. Material removal by milling machine

Step 2. Drill 4 holes

Step 3. Grind the top surface

Process flow and time required:

Milling (60s) -> Drill 4 holes (10s) -> Grinding (120s)

Total machining time (190s)

Figure 3-11 Machining specification for product 1

Product 2

Figure 3-12 shows the second product for this FAC example:
Product two also involves three machining processes. The blank is firstly milled to the required shaped as shown in figure 3-13. After that, the four holes are drilled. The grinding machine is then used for surface finishing of the top face. The machining times for product 2 are also listed in figure 3-13.

<table>
<thead>
<tr>
<th>Step 1. Material removal by milling machine</th>
<th>Step 2. Drill 4 holes</th>
<th>Step 3. Grind the top surface</th>
</tr>
</thead>
</table>

Process flow and time required:

Milling (240s) -> Drill 4 holes (10s) -> Grinding (120s)

Total machining time (370s)

Figure 3-13 Machining specification for product 2
Product 3

Figure 3-14 shows the third product for this FAC example:

![Figure 3-14 Product 3](image)

For product 3, raw material is firstly milled into required shape. The four horizontally oriented holes are then drilled. At the end, the surface finish process is performed by grinding machine. The machine orientations for drilling and grinding in this example are different from products 1 and 2. Vertical oriented drilling and grinding machines are used to produce products 1 and 2 but product 3 requires these machines to perform processes in horizontal direction due to product requirement. Figure 3-15 shows the machining times for product 3.
Step 1. Material removal by milling machine

Step 2. Drill 4 holes

Step 3. Grind the surface of the four holes

Process flow and time required:

Milling (120s) -> Drill 4 holes (10s) -> Grinding (120s)

Total machining time (250s)

Figure 3-15 Machining specification for product 3

Product 4

Figure 3-16 shows the fourth product for this FAC example:

Figure 3-16 Product 4

For product 4, material is firstly removed by milling process according to the required shape. Four holes are then drilled by drilling. Figure 3-17 shows the
orientation for the drilling processes, they are also oriented in horizontal direction.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Material removal by milling machine</td>
<td>Drill 4 holes</td>
<td>Grind the surface of drilled holes</td>
</tr>
</tbody>
</table>

Process flow and time required:

Milling (200s) -> Drill 4 holes (10s) -> Grinding (100s)

Total machining time (310s)

Figure 3-17 Machining specification for product 4

Products summaries:
The four products and their required machining processes are listed, figure 3-18 summarizes the possible routes of machines to the twelve machines according to the product specification for the four products:

<table>
<thead>
<tr>
<th>Product 1</th>
<th>Any M (60s) -&gt; D1 or D2 (10s)-&gt; G1 or G2 (120s) Total: 190s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 2</td>
<td>Any M (240s) -&gt; D1 or D2 (10s) -&gt; G1 or G2 (120s) Total: 370s</td>
</tr>
<tr>
<td>Product 3</td>
<td>Any M (120s) -&gt; D3 or D4 (10s) -&gt; G3 or G4 (120s) Total: 250s</td>
</tr>
<tr>
<td>Product 4</td>
<td>Any M (200s) -&gt; D3 or D4 (10s) -&gt; G3 or G4 (100s) Total: 310s</td>
</tr>
</tbody>
</table>

Figure 3-18 Machining process routes and processing times for the four products in FAC example 3

3.2.3.2. Flow of pallets in example 3

Each of the product types are manufactured by three machines as shown in figure 3-18. Each machine pair in the routings are located in the same conveyor. For each product, the machining process sequence should be:
Milling machine process -> Drilling process -> Grinding process

According to this sequence, each product type has sixteen possible routes in this FAC, figure 3-19 summarizes the possible routings for the four products:

<table>
<thead>
<tr>
<th>Product 1 &amp; Product 2</th>
<th>Route 1</th>
<th>Route 2</th>
<th>Route 3</th>
<th>Route 4</th>
<th>Route 5</th>
<th>Route 6</th>
<th>Route 7</th>
<th>Route 8</th>
<th>Route 9</th>
<th>Route 10</th>
<th>Route 11</th>
<th>Route 12</th>
<th>Route 13</th>
<th>Route 14</th>
<th>Route 15</th>
<th>Route 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 -&gt; D1 -&gt; G1</td>
<td>M2 -&gt; D1 -&gt; G1</td>
<td>M2 -&gt; D1 -&gt; G2</td>
<td>M2 -&gt; D2 -&gt; G1</td>
<td>M2 -&gt; D2 -&gt; G2</td>
<td>M3 -&gt; D1 -&gt; G1</td>
<td>M3 -&gt; D1 -&gt; G2</td>
<td>M3 -&gt; D2 -&gt; G1</td>
<td>M3 -&gt; D2 -&gt; G2</td>
<td>M3 -&gt; D3 -&gt; G1</td>
<td>M3 -&gt; D3 -&gt; G2</td>
<td>M3 -&gt; D4 -&gt; G1</td>
<td>M3 -&gt; D4 -&gt; G2</td>
<td>M4 -&gt; D1 -&gt; G1</td>
<td>M4 -&gt; D1 -&gt; G2</td>
<td>M4 -&gt; D2 -&gt; G1</td>
<td>M4 -&gt; D2 -&gt; G2</td>
</tr>
<tr>
<td>Product 3 &amp; Product 4</td>
<td>Route 1</td>
<td>Route 2</td>
<td>Route 3</td>
<td>Route 4</td>
<td>Route 5</td>
<td>Route 6</td>
<td>Route 7</td>
<td>Route 8</td>
<td>Route 9</td>
<td>Route 10</td>
<td>Route 11</td>
<td>Route 12</td>
<td>Route 13</td>
<td>Route 14</td>
<td>Route 15</td>
<td>Route 16</td>
</tr>
<tr>
<td>M1 -&gt; D3 -&gt; G3</td>
<td>M2 -&gt; D3 -&gt; G3</td>
<td>M2 -&gt; D3 -&gt; G4</td>
<td>M2 -&gt; D4 -&gt; G3</td>
<td>M2 -&gt; D4 -&gt; G4</td>
<td>M3 -&gt; D3 -&gt; G3</td>
<td>M3 -&gt; D3 -&gt; G4</td>
<td>M3 -&gt; D4 -&gt; G3</td>
<td>M3 -&gt; D4 -&gt; G4</td>
<td>M3 -&gt; D3 -&gt; G3</td>
<td>M3 -&gt; D3 -&gt; G4</td>
<td>M3 -&gt; D4 -&gt; G3</td>
<td>M3 -&gt; D4 -&gt; G4</td>
<td>M4 -&gt; D3 -&gt; G3</td>
<td>M4 -&gt; D3 -&gt; G4</td>
<td>M4 -&gt; D4 -&gt; G3</td>
<td>M4 -&gt; D4 -&gt; G4</td>
</tr>
</tbody>
</table>

3.2.3.3. Configuration constraints

This example permits twelve pallets to co-exist at any one time. Machining processes are completed in the machines. Productivities for the machines are identical and same product quality will be achieved for the possible processing routes. Error detection of network status is also built in this FAC control system to prevent the loss of data transfer in TCP/IP communication.
3.3. **Generic agent-based control system**

Section 3.2 illustrates the physical configurations of the FAC examples. This section presents a generic agent framework and its structural design according to FAC example 1 configuration. Types of agent and their specifications in the generic control system are listed. The agent specification helps the system designer to identify unique and specific functionality for each agent in the generic control framework.

3.3.1. **Generic agent-based FAC control systems**

The FAC control software is responsible for the FAC control functions and planning strategy. Accordingly to the hardware configuration for such FAC, it is necessary to identify required agents for the control system in real world applications. Neglecting the specific shop floor control commands in different FAC configurations, generic agent templates for different types of agents with specific functionalities are proposed.

For example, although shop floor conveyor control commands are totally different in the two examples, a conveyor agent template is proposed. The agent template is responsible to embed the commands for conveyor control motions and provides a standard communication programming method to the agents sharing the conveyor information. Evaluation of agent interactions is the concern during the system design and analysis stage. Pallet arrival is assumed to be the system resource input or output for the conveyor control agent. An object-oriented agent based control system will deal with these input and output activities.

Each agent template possesses two distinct objects, they are objects for agent communication level and shop floor control level. The shop floor control object
contains fundamental device control commands for several device control purpose and it also provides room for further device control implementations. Each of the agents communicates via this agent communication object. Essential agent communication interfaces are proposed for agents which are used to exchange shop floor information from their shop floor control objects. These two levels compliment each other without interference.

3.3.2. Description of generic agent-based FAC control systems
The generic agent-based control system connects to the Internet and hence it is possible to send manufacturing information to other agents using electronic media formats, including the world-wide-web (www) and the wireless application protocol (WAP) for mobile communications. In the production control system, the agents work cooperatively towards prescribed scheduling and planning heuristic algorithm performed by scheduling agent. For security reason, the instruction will be verified by the product identification module after an order is received by the external information interface inside the control system. After the scheduling agent receives the verified instruction from the product identification module, the agent checks the availability of the system with other controlling modules in the shop floor. If the whole system is ready, pallets are loaded onto the FAC and each pallet receives the required assembly process or machining process according to the instruction of the scheduling agent. Relationship of the modules or agents depends on their work dependencies during the operation of the cell. Agents with common characteristics share the common and useful information and functionality classes. The proposed generic agent-based structure for FAC example 1 consists of agent units corresponding to typical manufacturing functionalities in assembly systems (Figure 3-9).
3.3.3. **Overview of the agent-based control architecture**

Figure 3-20 shows the generic agent structure for FAC, the agents are designed according to their own specific purposes. The design prevents duplication of functionalities in other agents in the same system. And agents are designed and grouped together according to their functional similarities. For example, the class of conveyor agent is composed of other four conveyor control objects in the generic FAC example. Each of the four conveyors is responsible for a conveyor control instruction in the shop floor.

3.3.4. **Agent listing**

The set of agents proposed for the FAC control system examples are conceptually identified according to general FAC system requirements. It comprises commonly
used intelligent control components for manufacturing system operations. Inside
the control system, an agent interacts with other agents through their communication
interfaces. Addition or elimination of an agent or agent object will not affect the
overall system performance because of the distributive nature of the control system.

3.3.4.1. Classification of agents
The agent based control framework depicted in figure 3-20 is applicable to other
FAC examples. Although other FAC configurations may be different in many
ways, the generic agent architecture is also applicable to these configurations. The
proposed generic agents in the generic agent framework are listed in the following:

**Scheduling agent**
The scheduling agent is responsible for coordinating tasks between the pallet identity
agent, the barcode reading agent and the conveyor control agent. This agent links
up with the central database to retrieve and save the overall system information. It
is embedded with a specific scheduling algorithm to cater for the scheduled
instruction in the FAC.

**Pallet identity agent**
The pallet identity agent is responsible for the provision of the local pallet
information and pallet position information from the sensor signal interpreter agent.
The information is sent to the scheduling agent, the barcode reading agent and the
conveyor agent. The pallet identity agent has its own pallet information database.

**Barcode reading agent**
The barcode reading agent is responsible for the provision of the local barcode
information from the barcode signal polling agent. This information is sent to the
scheduling agent, the pallet identity agent and the conveyor agent. The agent has its
own barcode information database.

**Conveyor control agent**
The conveyor control agent is responsible for the provision of the local solution from the two sub-level conveyor agents. The information is required by the scheduling agent, the pallet identity agent and the barcode reading agent. The agent has its own barcode information database.

**Sensor signal interpreter agent**
The sensor interpreter agent is responsible for the provision of the local sensor information from the sensor position translation agent. The agent co-operates with the pallet identity agent to identify the pallets’ locations. The agent has a common pallet information database sharing the information with the pallet identity agent.

**Barcode signal polling agent**
The barcode signal polling agent is responsible for the provision of the local barcode information from the barcode readers. The information pending from the barcode readers will be read to the agents in default time intervals.

**Sensor signal positional agent**
The sensor signal positional agent is responsible for the provision of the sensor signal interpreter agent. The agent translates the raw signal into the pallets’ positional readable format to the controller.

**Robot control agent**
The robot control agent is responsible for the provision of the local robot manipulation. The agent also assigns the required control program to the robot to meet the order requirements.
**Order handling agent**

The order handling agent provides the interface for the cellular FMS connecting with the external order requesting agent through the internet communication or mobile communication media.

### 3.4. FAC control architecture overview

The FAC control architecture comprises complicated control entities responsible for different production process control functions. The common concern of this kind of control system is the uncertainties and interaction between control devices. Poor interaction or exceptional events will halt the system operation. Therefore, in the development of the agent-based FAC control system, it is necessary to adopt some formal development methodologies in the system design and programming phase.

In this research, the UML is used in the development of the generic agent-based FAC control system. Using the UML approach, a framework for the generic agent-based FAC control architecture is proposed. Corresponding to the control architecture, the structure of individual agents and protocols are established with the object-oriented approach. However, the UML is insufficient in modeling the highly dynamic agent-based FAC control system, the dynamic view of the UML is not able to provide a complete representation of all possible agent interactions in the system.

In the development of the proposed system, Color Petri Nets (CPN) and a data mining technique are used to supplement the UML static and dynamic models to provide dynamic modeling, simulation and analysis for the dynamic interactions of agents.

The design and analysis of the generic agent-based FAC control system is therefore divided into 3 phases (Figure 3-21). The first phase involves the
development of the UML static model. The second phase corresponds to the establishment of the dynamic model. During this phase, the UML dynamic model and the CPN model have to be used together to provide a complete representation of the dynamic modeling and simulation of agent interactions in the system. Finally, detail analysis is carried out in the third phase with the aid of a data mining technique.

3.4.1. UML construction
The UML logical view, physical view and scenarios view are used to address the infrastructure for the control entities in the control system. The scenario view is the first model to be established. It presents the structure for the design system with logical views and it will be used later in validating the other different views. The functional requirements of the system are presented in the logical view and it also displays and identifies the abstraction model for the control system. Functional requirements of the agents are classified into different classes or object according to the classification of the agents. The modular structure of the agent helps to avoid redundant effort in the system design stage.

3.4.2. Mapping from dynamic UML model to CPN model
Process view in the UML model and the CPN model are used to provide a complete representation of the dynamic interactions of agents. In this respect, the process view in the UML model has to be mapped into the CPN representation. The CPN model is then used for constructing the dynamic interactions of agents.
3.4.3. **Analysis of the design control system by a Data mining algorithm**

The model will then be used to undergo simulation process. The data retrieved from simulation will then be evaluated by a data mining algorithm. Association rules between agents, occurrence of specific agent activities and agent behavior are extracted out by using this data mining algorithm. Design models of the agents are modified according to the mining result and then the system model can be updated for the next simulation. Iteration of this design and analysis process is performed until the requirement of the control system is satisfied.
3.4.4. **Finalized simulation result for the implementation of the control system**

The process view of UML will then be constructed according to the modified mapping back from the CPN model to the UML dynamic model. This mapping process ascertains the liveness, boundedness, safeness and conservativeness of the control system in the design stage and it enhances the detail descriptions by UML models. The deployment view is the last model to be defined before implementation.

3.5. **Conclusion**

This chapter presents a simple FAC example (FAC example 1) and two hypothetical FAC examples (FAC examples 2 and 3). The simple FAC example consists of 2 robots. In this example, agents are proposed and designed according to this configuration. Because of the simple configuration of this example, control entities with fundamental functionalities are required. Reusing this control structure, complicated control elements for agents can be added to other FAC configurations. The hypothetical examples provide other configurations to illustrate the possibility of reusing the control system. Specifications of major agents are defined in this chapter and the next chapter presents the construction of agents with the object-oriented modeling methodology. An analytical analysis will be used to verify the performance of agent interactions. A data mining tool is employed in this project to extract any hidden association rule from CPN simulation result. The mined result assists to the modification of the dynamic UML model. Intensity of the association rule reveals the frequency of interaction between agents. The liveness, conservativeness, safeness and reversibility checking of CPN together with the intensity measurement of data mining to the simulation will provide a complete analysis to the interaction between system components.
CHAPTER 4

DESIGN MODEL

The design goal of the proposed agent-based FAC control system structure is not only restricted to the control of the example FACs in this thesis. The ultimate goal is to design a generic autonomous FAC control system. Normally, an FAC control system is a discrete event system which distributes tasks to system devices upon arrival of an order. The control system is responsible for manufacturing system scheduling and planning, and assigns resources to attain the requirements of the receiving order. Agent applications in FMS control systems have been identified in some research cases (Lin et al, 1994, Choi et al, 1997, Kouiss et al, 1997, Choi et al, 2000) as reviewed in Chapter 2. This chapter is on the UML modeling of the generic agent-based FAC control system.

The system development method ICAM DEFinition methodology (IDEF) is commonly used in structural modeling for representing large and complex manufacturing systems (Bravoco et al, 1985) including CIM systems and FMS design frameworks (Dorador et al, 2000). The specification of the IDEF model provides information model, functional model and data model to represent the proposed design system. Disadvantages of using IDEF include the lack of a dynamical modeling methodology to event happening, and the structural design by using IDEF is not coherently related with one another. System designers or users may find it hard to relate several IDEF models. Although the model provides functional and informative views in the design of a target system, it lacks the dynamic view for interactions between functional entities.

The object-oriented approach has become the de facto software engineering standard to support the design and modeling of complex systems. The use of O-O methodologies such as UML are able to provide suitable and widely-used modeling constructs for developing structured, configurable, reusable and readily distributed multi-perspective models of complex systems. As reviewed in Chapter 2 (Section 2.5), researchers have attempted to use UML and the extended UML for developing
agent-based systems. As indicated by researchers, UML is not sufficient in the modeling of agent-based applications as it is not able to completely represent the dynamic agent interactions.

In this thesis, the modeling and simulation of the proposed multi-agent based FAC control system has been established with a hybrid of the UML and Color Petri Net methodologies. This chapter presents the UML modeling of the proposed agent-based FAC system and the dynamic analysis of the proposed system with CPN.

4.1. Framework design of agent-based FAC control system
Agent-based applications are usually implemented in DAI systems in which agents coordinate their work via network. Through negotiation and communication with other agents, and interaction between shop floor devices and agents, the agent-based FAC control system is able to distribute and allocation tasks to agents.

The reusability, reconfigurability and encapsulation structures for agents are required in the implementation of agent-based FAC control system. The O-O modeling approach is a suitable tool for this purpose. In object-oriented modeling techniques, the models are important perspective views to represent components in a target system. They are the static view, dynamical view and functional view. In the development of the proposed system, these three views are used in system modeling and implementation. They also provide useful perspective views for the representation of complicated FAC control systems.

While the concept of object inspires fine-course programming, it also gives development cycle a new direction. The spiral development cycle in Rational Unified Process (RUP), (Boggs and Boggs., 2002), formally broke the traditional development of system into many sub-cycles. Analysis, design, coding and testing are performed in every sub-cycle. Integrations are concurrently made in every iteration so as to ensure the objective is not deviated. Finally, the software product will be released in the last iteration. Figure 4-1 depicts the multiple design cycles using RUP.
In agent-based systems, an agent can be regarded as an entity which coexists with other agents. As new agents and new relationships may be formed from time to time, analysts must ensure that the system has enough extensibility to cope with this kind of changes. In order to handle this requirement, the iterative development cycle in RUP (Rational Unified Process) would be a suitable methodology in reflecting the continuous refinement.

Figure 4-1. Design phases in Rational Unified Process, Kruchton (2003)

Figure 4-2 shows the conceptual flow of the generic FAC control system. The application requirements of the control system are collected and then prepared for the design process for the control system.

Figure 4-2 Conceptual design process for the generic FAC control system

For the conceptual design process, there are many resources involving in the design system towards the final shop floor control system, figure 4-3 shows the resources required to the final control system. System development cycle is normally a project which comprises peoples. For a FAC control system, they are the system engineer, system user and software engineer etc. And they have to co-operate with each other
to set series of FAC control requirements and possible work-flows between FAC devices. And the framework of the control system is the product for this collaborated work.

Figure 4-3 Resource requirement for the design and development of the generic FAC control system

The generic control system is a discrete event system and the design and analysis of the control system is a process-oriented development process. For the system development process for complicated control systems such as the FAC in this example, the development tools and methodology of the control system are vital. Figure 4-4 lists the problems and requirement commonly encountered in the development of complex control systems. The corresponding methodologies for resolving the problems are also listed in the table.
<table>
<thead>
<tr>
<th>Problem and requirement</th>
<th>Reason to criticize the problem/s</th>
<th>Use of standard methodology to resolve the problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigor and formality</td>
<td>Division of labour</td>
<td>System requirement and system flow identification and define set of system activity</td>
</tr>
<tr>
<td>Separation of concerns</td>
<td>Easy system maintenance</td>
<td>Define set of model</td>
</tr>
<tr>
<td>Anticipation of change</td>
<td>Increase flexibility and reuse of existing design components</td>
<td>Define in modular based system component</td>
</tr>
<tr>
<td>Modularity</td>
<td>Standardization of the agent components</td>
<td>4+1 views by UML model</td>
</tr>
<tr>
<td>Abstraction and interconnectivity</td>
<td>Development cycle is easy for the next control system with different configuration</td>
<td>UML model framework</td>
</tr>
</tbody>
</table>

Figure 4-4 Why standard methodologies should be used in this project.

4.2. The UML model

The UML approach is adopted in the development of the agent-based FAC control system. Basically, agents in the system are modelled by specific object relationships, class attributes and behaviours of the physical components or operating functions in the FAC systems. With the help of the dynamic model and the functional model, a proposed framework of the agent-based system is built. The generic FAC models contain objects corresponding to the scheduling agent, robot control agent, pallet control agent and conveyor control agent as identified in Chapter 3.

UML provides nine diagrams to allow users to capture the characteristics and to define the model for different aspects of a system. Different diagrams are used to establish the 4+1 UML views of system architecture. In this section, the diagrams in the use case view, logical view and component view are used to show the UML modeling of the agent-based FAC control system.

4.2.1. UML modeling approach

A complex system can be represented by fundamental UML models such as objects, classes, methods, messages, encapsulation, inheritance, polymorphism (Becker et al.,...
1994). An UML model is especially applicable to the design of real time distributed control application (Smirnov et al., 1998). The modeling application benefits from the fast processes, quick response, and complex interaction by real time synchronous input and output between agents. Fast response to the changing environment, dynamic information exchange and inter-modules connectivity are the critical success factors for the application. Software implemented by UML model is highly dynamic to the external shop floor environment. The object-oriented design and analysis approach using the UML model provides clear pictures for agent objects and then provides way to represent messages sending and receiving between these agent objects in both the dynamic and functional UML models.

The visualization of the UML enables the scope of architecture to be viewed in different aspects in the stage of system design. The 4+1 View of Architecture (Kruchten, 1995) proposes an architectural blueprint interaction between the views in UML modelling: 1. logical view, addresses the functional requirements of the system, is an abstraction of the design model and identifies major design packages, subsystems and classes; 2. implementation view, describes the static modules in the organization in terms of packaging, layering and configuration management; 3. process view, addresses the concurrent aspects of the system at runtime-tasks, threads, or processes as well as their interactions; 4. deployment view, shows mappings of various executables and other runtime components in the underlying platforms or computing nodes; 5. use-case view, drives the discovery and design of the architecture in the inception and elaboration phases which will be used later in validating the different views. From the 4+1 View of Architecture, the overall scope of the application can be displayed logically. With UML, components as well as their relationship can be traced easily, while at the same time, the cohesiveness can also be reflected systematically. The use of the 4+1 View of Architecture is definitely a classical example in extending the potential of UML. Figure 4-5 shows a brief summary of the 4+1 view model.
Using the UML as the design approach, major activities and modules of the active agent system are illustrated. In this project, it starts with the study of use cases which it is a detail descriptions of the identified objects and how they work together. Common functionality of the objects are identified and extrapolated into the same collection. This collection is collaborated into classes or objects. These classes or objects show the static relationships with each other. The objects in the same class are called agent in this project. The agent classification among classes diagram are grouping according to the agent functionality. For example, although there are classes for robot and pallet control agent represents in the class diagram, they are embedded into different packages refereeing to specific agents. An association is a structural relationship that specifies the connection between one or more members of the classes. Specific functional entities of agents are embedded in agent classes. Inheritance characteristics of agents designing in this way enhance expandability but it will not degrade the functional flexibility of agents.

When compared with traditional modeling methodologies, UML modeling technique has the following advantages:

- Complex FAC control system can be decomposed into modules and the rigid structure of traditional manufacturing structure resolved by the modules design of UML model.
• Systematic UML design and modeling methodology reduce redundant codes in the design of the FAC control system. Reusability and inheritance characteristics reduce the coding size of system, encapsulation provides privacy of objects from public access.
• The dynamic model of UML method analysis captures event happen over time and the model illustrates interaction of system activities by method description over objects.
• The UML model of the target system indicates the relation and attribute of the object modules in control system.

It can also decompose the complex system into components revealing the following characteristics:

• Provides details of the proposed system which is hidden during the design stage.
• The proposed complex system is decomposed into units according to specific functionality, and the modular entities are easy to maintain when compared with the traditional control architecture for the complex system.
• The modular entities are categorized according to agent functionality. For example, functional entities related to conveyor control are grouped into conveyor control agent object.
• Having the standardized methodology to the construction of agent object by UML method. Interaction between agents and internal agent object are clearly defined by the dynamic UML design model.
• Because the agents of the proposed system construct in layer format, the reusability of layer prevents duplication of time designing for repeating model.

4.2.2. Logical view
The logical view addresses the functional requirements of the system. It also shows the implemented behavior in the use cases and clear pictures for each component are also presented in this view. Classes and their relationships are clearly shown in the class diagrams and state chart diagram. In this view, the use case diagram illustrates possible events among actors (agents). Class diagrams will be used to represent
agents of the generic agent-based control system. The static relationships shown by the class diagrams are the essential models for the construction of classes of objects. Agents of the control system show their relationships in the class diagram. Objects inside the diagram provide practical information for the connection between the devices or the logical or controlling units inside a generic FAC control system. It also illustrates relationship of dependency between classes.

4.2.2.1. Use case diagram

Use case diagrams display events happening in the design system. At later stage of the UML design process, it captures the system scenario, which drives design and implementation of the overall system. By using the use case diagram, the use case in the diagram provides the external structural view for the control system. The interactions and events happening inside the diagram ensure the design of the control system according to the prescribed planning, and the diagram captures the system functionality in a graphical format.

Actors in the use case diagram represent specific agents. The class diagram shows an agent and its components. The class diagram and the use case diagram are used complementary to show the static view of the FAC control system. Figure 4-6 shows an example of use case diagram in the FAC control system. This use case diagram shows the possible events in the FAC control system. A use case is represented by an oval in the use case diagram, and the actor relating to a specific use case is connecting with an arrow line between these two entities. The use case diagram presents the set of required actors (manufacturing devices or virtual entities/agents) and relevant processes in use cases, therefore, the relations of the inputs to the outputs are clearly shown.

The use case realizes essential and critical functionality and defines the key classes in the system. The use case helps in the process unfold, it gives a clear picture on how the agents correlate to the proposed system events. After the system requirements for agents have been extracted from list of agents, the use case diagram is used to display the interactions between agents and system events. This list provides guidance on how agents interact with other agents in the generic agent
control system. In this project, agents are introduced according to the functionality required in generic FAC control system. An agent is represented by an actor in the use case diagram. For example, the conveyor control agent controls the motions of conveyors and the description of the agent in the list provides general description on how this agent controls the conveyor. It also provides description on the interactions of the conveyor control agent and other agents.

Figure 4-6. A generic agent based control system of an FAC in use case representation

4.2.2.2. Class diagram and the agents object relationships

The class diagram provides internal views for the generic agent based control system. The design of the class diagram captures the internal views of the control
components and system designers understand the class and object functionality. Different components in the control system are separately design in different classes of objects and their relationships are clearly illustrated by the static relationships between classes. The division of the object/class for agents enhances the modular design of the control system. Benefits such as easy system maintenance and distributed design effort are obtained from this modular design of the control system.

System requirement of the target system should be extracted before the construction of the UML model. This process provides information such as objects and classes for the system, and how these objects interact with each other. In Chapter 3, the agent list provides the generic agent-based system with a documentary account of system requirements. The list reveals agents according to functionality classification. Figure 4-7 shows the relationship of the classes or objects in some of the developed agents.
4.2.2.2.1. Aggregation

Aggregations of agent classes are shown in the agent relationship diagram to show their relationships in the generic control system. The aggregation groups dissimilar sets of objects to each other. In general, an aggregation implies the relationship “x object is part of relationship to object y”. Figure 4-8 is the enlarged snapshot of
The pyramid symbol is the aggregation relationship in UML representation. It shows the “is_part_of relationship” between agent objects/classes. In the diagram, the sensor object and the motion controller object are “part of relationship” to the conveyor detection object and conveyor position control agent respectively. Because the sensor and motion controller are dissimilar items and they are the shop-floor-related control entities, these objects are grouped by using the “is_part_of relationships” to the control entities at the high level controller.

This example of aggregation shows that the content of the sensor object can be retrieved by the conveyor control agent but not vice versa. And the retrieved sensor object is privately used by the conveyor control agent object and the content is privately used by the conveyor control agent object. Figure 4-9 lists example coding in Java for aggregating the conveyor agent control objects.
4.2.2.2. Association

A relationship is a link between instances of two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes. Figure 4-10 shows the association relations of the example of figure 4-7. The association relationship shows a link between two objects. The pallet information database object is retrievable by the pallet identity agent object and the sensor signal the interpret agent. These two agents retrieve detail information of the pallet in the FAC control system from the pallet information database. Because the database for the pallets needs to update the information when the pallets move to new positions, the Pallet information database is thus known to these two agents for information updating.
The association example shows the contents between two agents in bidirectional way. Taking the example of the association between pallet identity agent object and pallet information data object, they are linked together by a line which reveals the association relationship in UML representation. Figure 4-11 shows the java framework example to represent the association relationship for these two agents.

Figure 4-10. Association relationship between objects
<<Code for pallet identity agent object>>

```java
public class pallet_id_agent {
    EmpType pallet_info_data;
    public EmpType getEmpType () {
        if UCase(strData1) = "CLOSE" Then
            WskClient1.Close
            TCPIPStatus = "Closing Connection."
            cmdConnect1.Enabled = True
        End If

        if strData1 = "Welcome" & vbCrLf Then
            txtReceive.Text = "strData"
        End If

        if strData1 = "F1" And con1flag = True Then
            send3 = True
            txtReceive.Text = txtReceive.Text + strData & vbCrLf
            BarComm1.Output = "6"
            txtReceive.Text = txtReceive.Text + "6" & vbCrLf
            con1flag = False
        End If

        .......
    }
}
```

<<Code for pallet information database object>>

```java
public class pallet_info_data {
    EmpType pallet_id_agent;
    public EmpType getEmpType () {
        if strData1 = "re" Then
            WskServer.Close
            cmdListen.Enabled = True
        End If

        select case Mid$(strData1, 1, InStr(1, strData1, ":"))
        case "4"
            BarComm1.Output = "4"
        end select

        if strData1 = "3" Then
            BarComm1.Output = "3"
            txtStatus.Text = "Right" & vbCrLf
        End If

        if strData1 = "4" Then
            ...................
        End If
    }
}
```

Figure 4-11 Example Java code for association
4.2.2.2.3. Generalization and specification

UML uses the term “generalization” to specify the classification relationship between a general element and a more specific element. The term generalization is being interpreted to represent the classification hierarchy between objects. For example, the generic FAC control system object in figure 4-7 is a general representation to the other classes like scheduling agent, conveyor control agent and the other agents within the same figure. Conversely, every class is a more specialized functional units than the generic FAC control system object. The more specific element may contain information that is particular to it, as long as it remains completely consistent with the description of the more general element. Generalization applies mainly to classes, packages and use cases.

In other words, specification is the reversed specification of object or class from the top hierarchy to specify the details of the class at the bottom of the hierarchy. Figure 4-11 shows the generic FAC control system class is a general object for the scheduling agent object. Although the scheduling agent object belongs to the object of the generic FAC control system, details must be specified inside the scheduling agent content. In the case of class, the specification relationship expresses the meaning of "the elements of one class are also described by another class by using the type of another class". In the reciprocal case, the generalization relationship signifies 'is a' or 'is a kind of'. Figure 4-12 shows the specification example for the generic FAC control system. The triangle without filling in figure 4-7 shows the specification relationship symbol in UML.
<<Code for pallet identity agent object>>

Public class schedulingagent extends generic_FAC_ctrlsystem {
    skServer.Protocol = sckTCPProtocol
    txtServerIP = WskServer.LocalIP
    WskClient.Protocol = sckTCPProtocol
    WskClient.Protocol = sckTCPProtocol
    BarComm1.CommPort = 1
    blnReply = False
    WskClient.Connect txtClientIP.Text, txtClientPort.Text
    lngTime = 0

    While (Not blnReply) And (lngTime < 10000)
        DoEvents
        lngTime = lngTime + 1
    Wend

    If lngTime >= 10000 Then
        MsgBox "Unable to connect to remote Server at " + WskClient.RemoteHost, vbCritical
        WskClient.Close
        TCPIPStatus = "Disconnected."
    Else
        TCPIPStatus = "Connected."
        cmdConnect.Enabled = False
        WskClient.SendData ("PLC" & vbCrLf)
        logout.Enabled = True
    End If

    ..................
}

Figure 4-12 Example java code for generalization
4.2.2.3. Class diagram

Having the overall picture for the manipulation of the FAC control system available from the use case diagram, details of the agents should be defined to accomplish the control system. In this project, the UML static model is used to design agents in detail form. It describes the types of classes that exist in the system and shows the relationship between them. Each class is represented graphically by rectangular boxes accommodating lists of attributes and operations. Classes are connected together by lines or links that are either of the association type or the generalization type. An association is a structural relationship that specifies the connection between one or more members of the classes. A generalization is a relationship between a general class and a derived class. Inheritance is an example of the generalization. One class is defined from another class by means of inheritance. The operations defined in the class diagram include all the services that can be requested from an object to perform the desired assigned behavior for the agent. Specific functional entities of agents are embedded in agent classes by the operation representation in class or object. Inheritance characteristic of agent object enhances the expandability of agent but it will not degrade the functional flexibility of agents.

In object-oriented programming structure, variable declarations and methods are defined inside an object or class. An object or class normally constitutes three layers. Figure 4-13 shows the three layers in UML format.

<table>
<thead>
<tr>
<th>Name</th>
<th>Attribute</th>
<th>Operation</th>
</tr>
</thead>
</table>

Figure 4-13 A class in UML static diagram

The use case diagram describes the proposed events in the FAC control system. Activities inside the use case will be further expressed in a dynamical UML diagram. Accordingly, detail specifications for the actors of the agent classes are represented in the static UML model. An Agent class is a space for imbedding agent attribute and behavior. The agent attribute is a collection of constants, variable declarations
in a programming environment. Each agent possesses specific functionality in the
generic control system. For each agent, they are exceptional with each other.

The attribute of an agent object contains agent system information. It can be
defined as public or private retrieval states. For public attribute setting, all agent
objects or classes can access this attribute during system operation. But restricted
access right denies system component accessing information by unauthorized agent
classes. The access right for an agent object is also applicable to the method
declaration of agents. A method is defined in the operation layer in object. It is a
sub-routine liked programming structure which it is called by agent object to
manipulate necessary operation for the agent upon prescribed condition. As well as
the attribute of the agent object, other agents can invoke public agent method but
private method can only be called by assigned agents.

Encapsulation of an object is carried out by the accessing right of agent attribute
and method. The right prevents unauthorized access of agents from specific object
information and operation. Attribute and method are embedding inside the agent
object and the accessing right will not affect normal agent operation. Encapsulation
in this way enhances reusability of system resource and functionality which have
been developed in other objects.

For example, the scheduling agent in the FAC control system has three types of
classes, namely, classes for receiving shop-floor production order, agent
communication class and class for shop floor scheduling. If it is necessary to change
the scheduling algorithm or rule, the class of the scheduling agent is the only class
which have to be changed in order to implement the new rule. Figure 4-14 shows
this static UML diagram for the scheduling agent. “ScheduleRule” is the class
responsible to scheduling rule of pallet to process allocation. For this agent object, it
possesses one attribute class and two methods. These methods retrieve system
information by using the attribute class in the same class. For security reason,
attribute class has private retrieval settings and this pallet information is only
accessible to the components in the same class.
Classes tf1, tf2, tf3 and tf4 in figure 4-14 belong to class TextEventTest and all these classes belong to Form1 class. The Form1 class is the graphical user interface class to display the scheduling agent online information. Properties of the user interface from the form in tf1 inherited from its parent class TextEventTest and Form1. The inheritance property is shown by the arrow that points from the parent class to its child class. Figures 4-15 to 4-17 display the classes for the conveyor control, robot and pallet identity agents respectively.
Figure 4-15 The class diagram for the conveyor control agent

Figure 4-16 The class diagram for the robot agent
Figure 4-17 The class diagram for the Pallet identity agent
In UML representation, it is not necessary to represent every agent in individual objects model. For example, figure 4-12 shows the pallet identity agent class in the FAC control system. The sensor signal positional agent is embedded inside the pallet identity agent (SensorSigPostAgent). Signals sent from the shop floor sensors are received by this class, because this class belongs to the class of pallet identity agent. Apart from the sensor signal position agent, the pallet identity agent works closely together with the Barcode signal polling agent (Barcode) to identify the pallet location in the conveyor of the FAC control system.

The use case diagram and the object diagram are collaboratively called logical view in the UML design system. The logical view provides information such as functional requirement and correlation information between agents of the generic FAC control system. It also shows events by use cases and its relations with actors (agents). Details of the events description will be further described in the dynamic UML model. A clear picture for each component of object and class is also presented in this view. Classes and their relationships are clearly shown in the static UML diagrams.

4.3. Conclusion
This chapter describes the physical realization of the use case for the FAC control system and it illustrates how to construct the functional entities for the generic FAC control system. The construction of the class diagrams in the UML design models shows the impact of the constraints to restrict the functional and non-functional requirements of the classes. This formal methodology helps system designers to describe the physical distribution of the system in terms of distributed functionality among functional entities. In this project, the generic FAC control system is implemented in the agent-based control system environment. By using the encapsulation of the UML model, rules or intelligent algorithm can also be embedded as functional objects. Nevertheless, the UML model is found insufficient to represent system dynamics. Extended models or tools should be employed to evaluate the system performance in a dynamical environment.
CHAPTER 5

DYNAMIC MODEL

With the UML modeling structure, the static model with class diagrams is established to represent the general requirements of the agent-based FAC control system. It is then necessary to prepare the dynamic model to describe the operation sequences corresponding to the prescribed classes. The dynamic UML model serves to describe the behavior setting of the class model. However, the operation details between classes and their interactions are not available in the dynamic UML representation. In this regard, Color Petri Net (CPN) models are used to describe the highly dynamic agent interactions. Application of the CPN model is able to enhance the analytical capability of the UML model. This can help to improve the representation and facilitate analysis of the UML dynamic model before system implementation.

5.1. UML Representation of the dynamic behavior for the agent components

The agent class models in chapter 4 can only serve to represent static information in class models, dynamic specifications for agent interactions are not provided in these models. With the static model, a use case presents the generic description of the system and the system functionalities. To represent the dynamic behavior, it is necessary to first of all identify and specify the corresponding events with regard to a use case. In UML, the sequence diagram and collaboration diagram are used to depict these scenarios. A scenario is an instance of a use case which shows a particular series of interactions among objects in a system. For our agent-based system models, these diagrams are able to represent the agent interactions. Nevertheless, they are mainly used to represent the sequential control nature of the system. They are insufficient in the representation of the complex interactions among agents within the distributive control system, especially when uncertainties and exceptions are involved.
5.1.1. Sequence diagrams

A sequence diagram is a UML model which shows the dynamic information. The diagram captures the agent interactions with time sequence in the system. This modeling diagram allows the system designer to specify necessary communication interfaces for agents. It describes activities happening in the use case. The horizontal axis of this diagram shows the life span of the object or class at the top of the diagram, while the vertical axis shows the interaction sequences for these instances. The classes or objects inside the diagram are defined in class diagrams. Instances from various classes may be used in a sequence diagram and they are not restricted to an unique class family. The instance may come from different agent classes linked by the interaction specification of sequence diagrams. Hence, agent interactions are constructed. The sequence diagram is very useful in elaborating and detailing the dynamic design and the sequence and origin of invocation of objects.

A class and its instance are shown in a rectangular box. For instance, the object “client” is an instance of the “Client” class. This relationship is an important characteristic in UML. This encapsulation and class relationship among classes and objects enhance the reusability of the design entities of the system. Figures 5-1 to 5-4 illustrate the events for the four use cases, they are “Scheduling agent work”, “Pallet Identity agent work”, “Robot agent work” and “Conveyor Control Agent work” respectively.

5.1.1.1. Sequence diagram for use case “Scheduling agent work”

Figure 5-1 shows the sequence diagram for the use case “scheduling agent”. Classes such as Form1, CommonData, TextEventTest and Client are used for the creation of instances in this diagram. These classes are defined in chapter 4 (figure 4-5) and they are objects in the class “Scheduling agent”. The class “Form1” provides graphical user interface instances of information display on the computer screen. The instance “test” is created from the “TextEventTest” class which is responsible for thread handling. It is also the scheduling agent’s thread controlling the connection with other agents. The instance “client” is designed to work with instance “commonData” while a new thread is
made for each connection. The instance “commonData” of the class “CommonData” is a database which provides pallet information such as pallet ID, process required for the assembly process etc. The sequence of events carried out in the “scheduling agent work” is:

1. Agents of the system (Actor: Agents connect request) send the connection requisition to the scheduling agent. Once the scheduling agent identifies and accepts the requisition, a thread will be established in order to handle the communication interaction between agents. Connection verification of agents are performed by operations check(), check1(), check2() and check3() in Figure 5-1.

2. The operations mentioned in step 1 will perform verification process with instance “commonData”, which is a database storing the pallet information. Operations such as change(), checkonline() and changeonline() are responsible to these verification processes and they will send update information to the connected agents.

3. “Form1” is the instance responsible for showing the graphic user interface on the computer screen. Updated information is received and shown on the computer screen by this instance.
Figure 5-1. Sequence diagram for use case “Scheduling agent work”
5.1.1.2. Sequence diagram for use case “Pallet identity agent work”

Figure 5-2 shows a sequence diagram for the pallet identity agent. Form1, wskClient1, wskClient2, com1, Barcode, SensorSingPostAgent and DatabaseRet are instances for this agent. These classes have been defined in figure 4-8. SensorSingPostAgent is a sub-class of the pallet identity agent and this agent works collaboratively with the pallet identity agent in this example. The sequence of events carried out by “pallet identity agent” is as follows:

1. The Pallet identity agent works collaboratively with the scheduling agent. They exchange information frequently using the communication channel established by actor “Scheduling agent request”. After the channel has been established, the pallet identity agent communicates with the robot agent using method GetData() with instance wskClientRobot. It is the instance for the robot agent. Furthermore, the pallet identity agent will use method GetData1() to communicate with instance wskClientSch which is the instance of scheduling agent.

2. The incoming login() request sending from the scheduling agent creates the instance “Form1” for graphic user interface for the pallet identity agent. The interface is established after completion of the verification process using “login()” method. It is the thread for this sequence diagram.

3. After communication with scheduling agent is established in step 1, the pallet identity agent will communicate with the shop floor computer using the method “initialize()”. The computer is responsible for receiving electrical signals from proximity sensors attached in the conveyors. These signals are encoded and locations for each pallet in the conveyors are determined.

4. Pallet instructions such as moving, lifting and pausing are sent to the shop floor conveyor controller via instance “Com1” by send() method. The device feedback will acknowledge the receiving instruction back to the device control computer by check() method.

5. The barcode information attached in the pallet is read by the pallet identity agent. For lower level machine control purpose, the barcode reader is handled by the methods RetIDInfo() and receive() for sending and receiving functions respectively.
6. After the updated information from shop floor is received, method “showpallet()” will display the details on the computer screen. The information includes the update pallet locations and pallet barcode information.

7. Communication between pallet identity agent and robot agent is handled by method GetData(). “wskClient1Robot” is an instance of the robot agent. Instructions for the robot operations are sent to the robot agent by using “Send out request” method. This operation is sent after the right pallet has been conveyed to the robot working area. The conveying action is accomplished through the communications between pallet identity agent to scheduling agent, barcode reader and scheduling agent.
Figure 5-2. Sequence diagram for use case “Pallet identity agent work”
5.1.1.3. Sequence diagram for use case “Robot agent work”

Figure 5-3 shows the sequence diagram for “Robot agent”. Classes such as Form1, wskClient1_DataArrival, WskClient2, Com1, Com2 and RobotProgram are the instances in this diagram. The classes were defined in chapter 4 (Figure 4-7). Both the actors “Operator” and “Conveyor control agent” are required to establish connection by using cmdConnect_Click1() and cmdConnect_Click2() methods. After the connection is established, the operations of robot agent work in this diagram will operate according to the prescribed flow of sequences. This diagram provides the flow of robot agent work as follows:

1. It is necessary to activate this sequence diagram by two actors. The actors are “Operator” and “Conveyor control agent”. After the thread of the robot agent work is activated, the thread will start the sequence diagram works.

2. Actor “Operator” is any human operator who works at the FAC control system. The operator can stop the robot operation manually for safety purpose in emergency. There are two robots in the generic FAC control system frameworks, cmdConnect_Click1() and cmdConnect_Click2() are the methods of two GUI buttons responsible for stopping the two assembly robots respectively.

3. Robot agent communicates with pallet identity agent through instance “WskClient1” in this sequence diagram. Updated pallet information is exchanged by GetData() method between the two agents.

4. Robots will start assembly tasks after the pallet is conveyed to the robot working areas. Acknowledgement of the pallet conveyor information is exchanged by instance “WskClient2” between robot agent and pallet identity agent. Robot agent receives updated conveyor status via this instance by GetData1() method. The harmony of robots and conveyors is accompanied using this information exchange method.

5. Shop floor devices control is responsible by instances “Com1” and “Com2”. Robot instructions are sent to the assembly robots via these instance to the robot controller in shop floor using send() methods.
6. Robot operations are programmed in an instance RobotProgram. Different robot programs are managed by this instance and they are requested according to the pallet information sending from “WskClient2”. Robot programs are requested by the robots via Com1 or Com2. Robot program retrieval request is sent by retrivfile() methods either from Com1 and Com2. The program of the robot operation is then sent to the robots directly via receive() methods back to the instances.
Figure 5-3. Sequence diagram for use case "Robot Agent work"
5.1.1.4. Sequence diagram for use case “Conveyor control agent work”

Conveyor control agent work is the sequence diagram for the conveyor control agent. Details of the sequence diagram are shown in figure 5-4. Classes Form1, CAWskClient1, waitServer, conveyorCtrl and caCom1 are instances for the agent and they were defined in figure 4-6. The descriptions of the sequence diagram are as follows:

1. Conveyor control agent and scheduling agent exchange information frequently, therefore, actor “scheduling agent request” is designed to establish connection between these two agents in the sequence diagram.

2. “Form1” contains thread control codes for the conveyor agent for which this thread will be created after the scheduling agent is established. “login()” method is responsible for this communication establishment.

3. “wskClient” will be the next instance to be created after the thread Form1 is created. The instance is responsible for exchanging and handling message between robot agent and this agent.

4. “waitServer” is the next instance to create. It is responsible for handling communication between this agent and pallet identity agent. The robot agent, conveyor control agent and pallet identity agent exchange real time pallet information, robot information, therefore, assembly processes are allocated to pallets using this information.

5. Conveyor control agent sends control commands to shop floor devices using instance “conveyorCtrl”. The agent sends commands to programmable logic controllers in the shop floor. Commands such as pallet lifting to robot working area, conveyor directional control to the conveyors are tasks performed by using this instance.
Figure 5-4. Sequence diagram for use case “Conveyor Control Agent work”
5.1.2. Collaboration diagram

The sequence diagram shows the dynamic specification for the instances according to time sequence, but there is a lack of association information. In this respect, collaboration diagram is helpful in an agent’s dynamic specification. Simply speaking, a sequence diagram shows how components interact with each other in time and sequence order and a collaboration diagram shows the association between these components for the target system.

In this section, the collaboration diagrams are utilized to show association of data flow between instances of system components. Although the collaboration diagrams show only the association between instances, however, the number of an event denotes its order in the event sequence. The number system indicates that the collaboration diagram can be easily converted back to the sequence diagram. Figures 5-5 to 5-8 show the relevant collaboration diagrams corresponding to the sequence diagrams in figures 5-1 to 5-4. Associations between actors and instances are clearly demonstrated in each collaboration diagram and the detailed time sequences of the events are given in the sequence diagrams.
Figure 5-5 Collaboration diagram for the use case “Conveyor Control Agent Work”
Figure 5-6. Collaboration diagram for the use case “Pallet Identity agent work”
Figure 5-7. Collaboration diagram for the use case “Robot agent work”
Figure 5-8 Collaboration diagram for the use case “Scheduling agent work”
5.1.3. Coding frameworks

After the dynamic and static models for the agents of the prototype FAC control system have been established, these models are to be converted into the prototype using UML supported programming language. Although the framework is a preliminary version for the generic FAC control system, the system design methodology is able to support flexibility and re-configurability in the development of control systems for flexible manufacturing systems.

5.1.3.1. Scheduling agent

The scheduling agent is designed and prepared with the JAVA programming language. The prototype of the coding framework includes classes “Client”, “CommonData”, “Form1”, “ServerSocket”, TextEventTest”, “tf1”, “tf2”, “tf3”, “tf4” and “Threadhandler” (Figure 4-3). Corresponding to the sequence diagram (Figure 5-1) and the collaboration diagram (Figure 5-8), the programming framework for the scheduling agent can be generated automatically using existing UML creation tools. Figure 5-9 shows the coding framework for the instance “commonData”. Figures 5-10 to 5-18 depict the codes for the rest of the classes belonging to the “SchedulingAgent” package. Methods and instances of the scheduling agent are converted into the UML supported JAVA programming language. These codes are strictly governed by the UML standard and the related methods and instances are inherited in the right classifications.
package SchedulingAgent;

public class CommonData
{
    public static Integer Initial_Value = 0;
    public static Long realTime = null;
    private Long accounts;
    private Integer onlineNo;

    public CommonData() {}

    public void change() {}

    public void changeonline() {}

    public void checkonline() {}

    public void check() {}

    public void check1() {}

    public void check2() {}

    public void check3() {}

    public void check4() {}
}

Figure 5-9 CommonData class in Java
package SchedulingAgent;

public class Client {
    public static Integer INITIAL_VALUE = 0;
    public Integer SIZE = 4;
    
    public Client() {}
    
    public Integer check(Integer j) {
        return null;
    }
}

Figure 5-10 Client class in Java

package SchedulingAgent;

public class Form1 {
    public TextEventTest theTextEventTest;
    public CommonData theCommonData;
    public ThreadHandler theThreadHandler;
    public Client theClient;
    
    public Form1() {}
    
    public void dispose() {}
    
    private void refreshEvent() {}
    private void stop_click() {}
    
    public void run()
    {
    }
}

Figure 5-11 Form1 class in Java
package SchedulingAgent;

public class ServerSocket
{
    public ServerSocket() {}
}

Figure 5-12 ServerSocket class in Java

package SchedulingAgent;

public class TextEventTest extends Frame
{
    public tf1 theTf1;
    public tf2 theTf2;
    public tf3 theTf3;
    public tf4 theTf4;
    
    public TextEventTest() {}

    public void addText1(String text) {}

    public void addText2(String text) {}

    public void addText3(String text) {}

    public void addText4(String text) {}

    public void run()
    {
    }
}

Figure 5-13 TextEventTest class in Java
package SchedulingAgent;

public class tf1 {
    public tf1() {}
}

Figure 5-14 tf1 class in Java

package SchedulingAgent;

public class tf2 {
    public tf2() {}
}

Figure 5-15 tf2 class in Java

package SchedulingAgent;

public class tf3 {
    public tf3() {}
}

Figure 5-16 tf3 class in Java

package SchedulingAgent;

public class tf4 {
    public tf4() {}
}

Figure 5-17 tf4 class in Java
5.1.3.2. Robot agent

The coding framework of the robot agent is implemented in Visual Basic. A system modeled in UML model can be implemented with any UML supported programming language. This is an important aspect for the design of the generic FAC control system that the model is not restricted to a specific programming language. For the robot agent, the programming framework contains classes “Com1”, “Com2”, “Form1”, “RobotProgram”, “wskClient1_DataArrival” and “wskClient_DataArrival”(Figure 4-7). The sequence diagram and collaboration diagram are presented in Figures 5-3 and 5-7 respectively. The Visual Basic framework of codes for the robot agent are listed in Figures 5-19 to 5-24.
Option Explicit

Private r1flag As Boolean
Private r2flag As Boolean
Private r1check As Boolean
Private r2check As Boolean

Public awskClient1_DataArrival As wskClient1_DataArrival
Public awskClient2_DataArrival As wskClient2_DataArrival
Public aCom1 As Com1
Public aCom2 As Com2

Public Function cmdConnect_Click1() As void
End Function

Public Function cmdConnect_Click2() As void
End Function

Public Function logout() As void
End Function

Public Function login() As void
End Function

Public Function stopRobot1() As void
End Function

Public Function stopRobot2() As void
End Function

Figure 5-19 Form1 class in Visual Basic
Option Explicit

Private text As String
Public aRobotProgram As RobotProgram

Public Function send() As String
End Function

Public Function receive() As String
End Function

Figure 5-20 Com1 class in Visual Basic

Option Explicit

Public aRobotProgram As RobotProgram

Public Function send() As String
End Function

Public Function receive() As String
End Function

Figure 5-21 Com2 class in Visual Basic

Option Explicit

Public Function retrivfile() As String
End Function

Public Function sendfile() As String
End Function

Figure 5-22 RobotProgram class in Visual Basic
5.1.3.3. **Pallet identity agent**

The pallet identity agent contains classes such as “Form1”, “wskClientRobot”, “wskClientSch”, “Com1”, “barcode”, “sensigPos” and “databaseret” (Figure 4-8).

---

```vbnet
Option Explicit
Private strData1 As String
Private temp1 As String
Private pos2 As Integer
Private check1 As String

Public Function GetData() As String
    End Function

Public Function check(robotcmd As Variant) As Boolean
    End Function

Public Function SendData1(data As String) As void
    End Function

Figure 5-23 wskClient1_DataArrival class in Visual Basic
```

```vbnet
Option Explicit
Private strData2 As String
Private temp1 As String
Private pos2 As Integer
Private check2 As String

Public Function GetData1() As String
    End Function

Public Function check() As Boolean
    End Function

Public Function SendData2() As void
    End Function

Figure 5-24 wskClient2_DataArrival class in Visual Basic
```
These classes contain methods representing the dynamic interactions of the pallet identity agent classes. Accordingly, the sequence diagram in figure 5-2 and the collaboration diagram in figure 5-6 show the dynamic interactions of the agent classes. The corresponding agent frameworks in Visual Basic are depicted in figures 5-25 to 5-33.

Option Explicit

Private check As Integer
Private flag As Boolean

Public WskClient1 As wskClient1_DataArrival
Public WskClient2 As WskClient2
Public aCom1 As Com1
Public aCom2 As Com2

Public Function cmdConnectSch() As void
End Function

Public Function cmdConnectRob() As void
End Function

Public Function logout() As void
End Function

Public Function login() As void
End Function

Public Function showpallet() As void
End Function

Public Function showbarcode() As void
End Function
Option Explicit
Private readID As Integer

Public Function RetDInfo() As String
End Function

Figure 5-26 Barcode class in Visual Basic

Option Explicit
Private DatabaseRet As String

Public Function Retdatabase() As String
End Function

Figure 5-27 DatabaseRet class in Visual Basic

Option Explicit
Private Com1 As mCom1Object

Public Function Com1_send() As String
End Function

Public Function Com1_receive() As Boolean
End Function

Public Function setCom1_aRobotProgram() As void
End Function

Public Function getCom1_aRobotProgram() As void
End Function

Figure 5-28 com1 class in Visual Basic
Option Explicit
Private Port1A As Byte
Private Port2A As Byte
Private Port1B As Byte
Private Port2B As Byte
Private Port1C As Byte
Private Port2C As Byte

Public Function vbout() As Byte
End Function

Public Function vbin() As Byte
End Function

Public Function initialize() As Byte
End Function

Figure 5-29 SensorSignPostAgent class in Visual Basic

Option Explicit
Private text As String

Public Function send() As String
End Function

Public Function receive() As Boolean
End Function

Figure 5-30 Com1 class in Visual Basic
Option Explicit
Private mwskClient2_DataArrivalObject As wskClient2_DataArrival

Public Function wskClient2_DataArrival_GetData1() As String
End Function

Public Function wskClient2_DataArrival_check() As String
End Function

Public Function wskClient2_DataArrival_SendData2() As String
End Function

Figure 5-31 WskClient2 class in Visual Basic

Option Explicit
Private mwskClient1_DataArrivalObject As wskClient1_DataArrival

Public Function wskClient1_DataArrival_GetData() As String
End Function

Public Function wskClient1_DataArrival_check() As String
End Function

Public Function wskClient1_DataArrival_SendData1() As String
End Function

Figure 5-32 WskClient1 class in Visual Basic
5.1.3.4. **Conveyor control agent**

The conveyor control agent is responsible for controlling conveyor motions in the FAC. The class diagram for the conveyor control agent is shown in figure 4-6. The dynamic interactions between the agent classes inside this agent have been described in figures 5-4 and 5-5. They are the sequence and collaboration diagrams respectively. Figure 5-34 to 5-40 present the coding frameworks for the conveyor control agent corresponding to the three views of the UML model.
Option Explicit

Public wskClient As CAWskClient1
Public waitServer As waitServer
Public conveyorCtrl As conveyorCtrl
Public com1 As caCom1

Public Function login() As void
End Function

Public Function waitConnect() As void
End Function

Public Function GerData() As void
End Function

Figure 5-34 Form1 class in Visual Basic

Option Explicit
Private strData1 As String
Private temp1 As String
Private pos2 As Integer
Private check1 As String

Public Function GetData1() As String
End Function

Public Function check() As Boolean
End Function

Public Function SendData1() As void
End Function

Figure 5-35 wskClient1_DataArrival class in Visual Basic
Option Explicit
Private mswkClient1_DataArrivalObject As wskClient1_DataArrival

Public Function Client1_DataArrival_GetData() As String
End Function

Public Function Client1_DataArrival_check() As Boolean
End Function

Public Function Client1_DataArrival_SendData1() As void
End Function

Figure 5-36 CAWskClient1 class in Visual Basic

Option Explicit

Public Function waitConnect() As String
End Function

Public Function GetData() As Boolean
End Function

Public Function opname() As void
End Function

Figure 5-37 waitServer class in Visual Basic
Option Explicit
Private start As String
Private stop As String
Private R1Work As String
Private R2Work As String

Public Function sendtoPLC() As String
End Function

Public Function recefromPLC() As Boolean
End Function

Figure 5-38 conveyorCtrl class in Visual Basic

Option Explicit
Private text As String

Public Function send() As String
End Function

Public Function receive() As Boolean
End Function

Figure 5-39 Com1 class in Visual Basic
5.2. Petri Nets

The UML model is not a suitable modeling structure in supporting system changes and it lacks the dynamic analytical capability for the FAC control system (Liu et al., 2002). A descriptive model with simulation capability is therefore required to model the concurrency and dynamic behavior of the agent-based FAC control system. Petri nets, originated in the early work of Carl Adam Petri in 1960’s, have developed into a graphical and mathematical modeling tool for studying concurrent systems (Peterson, 1977). Petri nets have been widely applied to the description of dynamical systems, in the modeling and analysis of concurrent components (Zimmermann et al., 2001; Li et al., 2004).

Petri nets are widely used to model and analyze manufacturing systems. Typical applications include scheduling and planning applications (Reyes et al., 2002), FMS deadlock identification (Elmekkawy et al., 2003), performance evaluation of logistic
application in manufacturing systems (Piera et al., 2004) and the autonomous manufacturing system simulation (Singh et al., 2003).

5.2.1. Color Petri Net

Basic Petri nets are insufficient in modeling complex systems which are hierarchical and temporal in nature. For instance, operations and activities in an FMS are mainly temporal activities, and they mostly involve similar but different attributes. Extended Petri nets such as Color Petri nets (CPN) and Timed Petri nets (TPN) have been developed to overcome these problems. In TPNs, time delays are added to transitions and they are able to model and analyze temporal functions in manufacturing such as scheduling and performance evaluation. For CPNs, tokens are able to carry data values. CPNs are able to model complex systems and data flows.

In our agent-based FAC control systems, CPN models are used to represent the dynamic agent interactions. CPNs were invented by Jesnen (Jensen, 1992). The advantages of CPNs over ordinary Petri nets are as follows:

1. The token in a CPN has state vector and the vector represents multiple resources inside the system. The multiple token flows simulate multiple processes of the FAC control system.
2. Attributes of a CPN are simulated by the firing of color tokens. Different colors represent different data types in attributes. System resources can be distinguished by color sets.
3. The guard in a transition and an expression of an arc provide conditional statement for the CPN model. The prescribed conditional statement of the guard or arc restricts the token movement.

It is the motivation for this project to integrate both the advantages of UML and CPN. The UML and the CPN models are complementarily used in modeling the agent-based FAC control system. Analysis of the CPN model will perform iteratively until complete system behaviors are identified. Result of the analysis will be fed back to the UML
model at the end of the project stage. The framework of UML and CPN models are used to construct an FAC control system template. This template is applicable to different FAC configurations and hence saves time in developing new FAC control systems.

5.2.2. Color Petri Net model definition

A Petri net structure normally comprises four basic elements: Place ($P$), Transition ($T$), Transition input $I(t_j)$ and Transition output $O(t_j)$. In a CPN $C$, a set of tokens is represented by $\mu_{\text{ColSet}}$ in place $P$. The $\text{ColSet}$ represents colors of tokens in the mathematical representation. Moreover, the multiple token notation is denoted by $k$ which is a non-negative number. The following mathematical equation describes a CPN model ($C$) with its graphical representation in the figure.

\[
C = (\mu_{\text{ColSet}}, P, T, I, O),
\]

where
\[
P = \{p_1, p_2, p_3, \ldots, p_n\}, \ n \geq 0
\]
\[
T = \{t_1, t_2, t_3, \ldots, t_m\}, \ m \geq 0
\]
\[
P \cap T = \emptyset
\]

where token $\mu_{\text{ColSet}}$ is defined for $C$.

\[
k(p_i I(t_j))_{\text{ColSet}} \quad k(p_{i+1} O(t_j))_{\text{ColSet}}
\]

![Figure 5-41. The CPN structure](image)

In figure 5-41, places $p_i$ is the input place of transition $t_j$, where $p_i \in I(t_j)$. The place $p_{i+1}$ is the output place of transition $t_j$, $p_{i+1} \in O(t_j)$. Multiple token $k$ for the places $p_i$ and $p_{i+1}$ are used as the firing condition of transition $t_j$. At the same time, the color set of tokens with multiple token criteria for the input or output of transition $t_j$ are denoted as follows:
The input and output of transition $t_j$ are $k(p_i, I(t_j))_{ColSet}$ and $k(p_{i+1}, O(t_j))_{ColSet1}$ respectively. The number of tokens for the input and output are assumed to be the same, this number is $k$ where $k$ is any positive integer. Settings of the token number can be different in quantity for input and output. The color token for the input place here is ColSet and that for the output place is ColSet1. When the number of tokens with color set “ColSet” of the input place $p_i$ is $k$, the transition $t_j$ will remove these tokens. At the same time, the transition will fire $k$ tokens with color set “ColSet1” to the output place $p_{i+1}$.

The number and color setting of the tokens are important characteristics of CPNs. Types of resource normally differentiate by the colors of tokens. And number of tokens represents available resources in the represented system. Both the multiplicity and color representation of tokens enhance the concurrent representation of the system during simulation. There are four important aspects which can be revealed and verified using the token flows of simulation; they are reachability, traceability, safeness and conservativeness:

- **Reachability** is verified by a reachability tree. This tree denotes token flowing direction in a CPN. The hierarchical structure of the reachability tree shows precedent and descendent relationships among token movements. A deadlock in a system is normally revealed by the movement record where a token is found to stop in the deadlock area.

- **Traceability** is a token movement locus which is prescribed by the equation of the prescribed system. Redundant resources allocation is found in the area of repeated token movements.

- **Conservativeness** of tokens is governed by the token movements between input and output. The conservative number of token must keep constant throughout a simulation. A CPN, $C$, with initial marking $\mu$ is called conservative, if and only if, the number of the prescribed input tokens and output tokens are constant during a simulation.
• Safeness is one of the boundedness property in CPNs. This property restricts the maximum allowable token inside a specific place or transition. If the maximum setting is found insufficient to cope with the loading of a system, the system requires additional places and transitions to reduce the workload. This implies more processing centers or devices are to be added in order to resolve this loading problem.

A number of tools are available for modeling and simulating CPNs (Brink, 1996). Most CPN editing tools provide simulation plug-in, typical examples are Design/CPN, CPNtools, PetriTool etc. Simulation is simply run after the CPN model of a system is constructed. The token movement record and the reachability tree are revealed by the plug-in instantaneously. Using the computer editing and simulation tool, modification is done in real time. The performance of the modification is then evaluated. These processes are repeated until satisfactory result is obtained.

5.2.3. **Color Petri Net model for the generic FAC control system**

The UML models for the generic FAC control system has been introduced in chapter 4 and section 5.1. These models describe the static and dynamic information of the FAC system. This information includes the agents, agent components and agent interactions. Although the UML is used to describe the FAC framework, the models do not have analysis capability. Performance evaluation of highly dynamic systems such as the FAC is important, in this regard, the UML is insufficient in the construction of the FAC control system.

Additional design model for the generic FAC should then be used together with the UML model. This model should be able to perform simulation. CPN is therefore chosen in this research to evaluate the performance of the system.

The flow chart in figure 5-42 describes the design and analysis methodology of this research project. Firstly, the UML models have mapped onto CPN models. The CPN simulation will take records of token movements and the data of these records are exported to a text file for analysis. Afterwards, a data mining algorithm is applied to
analyze these records. Information such as association rules and agent interactions are extracted by using this algorithm. Performance of the FAC system is thus revealed by this information. Modification of the CPN model is required recursively until the FAC performance is satisfactory. After that, the UML models will be updated according to the modified CPN model and the UML models are then become the finalized models for system implementation.

5.2.4. Mapping of the UML model to CPN model
This section describes the methodology for UML-CPN mapping. This finalized model of this conversion can enhance the designed system possessing analysis capability. The proposed methodology is also adaptable to other large scale manufacturing system applications.

Chapter 4 introduces the creation of UML models to the proposed generic FAC template, collaboration diagram and object diagram are used to create the agents of the template. These UML models provide static view for the system. They are not able to
represent dynamic information such as message communication protocol, agent interaction. Besides, system simulation can not be represented and performed by this model. Other modeling tool is therefore introduced to overcome the problem.

The mapping of UML to CPN is generalized into several steps. They are:

1. Use case diagram refinement
2. Use case model transformation
3. Class/Object transformation
4. Operation transformation
5. Token assignment
6. Algorithm representation

Figure 5-43 shows a schematic diagram for the proposed model mapping mechanism. In a typical UML model, several use case diagrams are always built for a system. Therefore, an unified use case diagram is firstly built according to these use case diagrams. This is the earliest step which is called use case transformation. The unified use case diagram are then used to transform into CPN components. The transformed CPN model becomes the CPN framework for the control system. This framework is subsequently used to transform more CPN components referring to the class/object transformation. The CPN components representing class/object are then used as reference in the operation transformation. Creation of the CPN components in this transformation is referring to the operations of objects or classes in the operation transformation. The mapping process of the CPN framework is basically completed after these transformations.

The framework constructed by the transformations is not able to perform simulation. Dynamic characteristic for the CPN should be performed by some movement objects. In CPN, token is the fundamental entity to perform this dynamic simulation process. Token assignment is then the next transformation step. After this transformation process, a generic FAC template is then completely mapped to CPN model after the
completion of this process. Additional modeling implementation such as algorithm application is always implemented by computer programming type control statements. The final transformation of the CPN components is proposed to convert such control statement into CPN model.
1. Use case diagram refinement

2. Use case transformation

3. Class/Object transformation

4. Operation transformation

5. Token assignment

6. Algorithm representation

Figure 5-43 Mapping steps of UML to CPN model
Figure 5-43 illustrates the transformations in a schematic diagram. The following paragraphs explain these transformation with detail descriptions with practical application examples.

1. Use case diagram refinement
Several use case diagrams are normally proposed in UML implementation. Each of these diagrams provides partial functional view for the proposed system but these diagrams are difficult to provide an overall system view. Therefore an unified transformation process is required to unify these use case diagrams.

Figure 5-44 shows three example use case diagrams which they are the diagrams in an example system and they are proposed to describe different functionalities for the FAC example 1. Common use cases or actors are labeled by same ID in these diagrams. Duplication of the components will be eliminated in the unification. An unified use case diagram is therefore simplified as shown in figure 5-45. For example, agent “A1” appears in use case diagrams 1 and 3 respectively in figure 5-44 and agents “A2” appears in use case diagram 2 and 3 respectively in the same figure. The relationships of the two agents are then simplified and integrated into a collaborated use case diagram as shown in figure 5-45.
Use Case diagram 1

Use Case diagram 2

Use Case diagram 3

Figure 5-44 Three use case diagrams

Figure 5-45. An unified use case diagram
2. **Use case model transformation**

The unified use case diagram in step 1 provides an overall system view and it is ready to transform into CPN components. A CPN framework is then constructed and more CPN components in subsequent transformations will be created according to this core structure.

For the transformation, an actor or an use case is proposed to transform into a place and a transition connected by an arc (Figure 5-46). In an agent-based control system, each actor is used to represent an agent and it is normally used to interact with use case. A place is the CPN entity that distributes tokens. The decision to fire a token from a place is controlled by the connected transition. In CPN, a control statement of transition defines a criteria to fire from its connected place. If-then-else is the commonly used criteria in the transition’s control statement.

In the description of UML model, an use case is described by a sequence diagram or a collaboration diagram. These two diagrams are normally used to express the activities happened in an use case, therefore, activities of several objects or classes collaboratively worked together. Use case comprises activities of one or several actors and they interact with other actors or actor’s objects. Hence, the use case will be treated as an actor and it is treated as an unify work by one actor. It will also be represented by one place, one transition and they are connected by an arc. And the use case is then subsequently explored by more CPN components in the subsequent transformation processes. Figure 5-46 summarizes the corresponding mapping mechanism between UML and CPN components.
An use case or actor is transformed into a place and a transition with an arc, for example, use case (U2) transforms into a place (P3) and a transition (T3) as shown in figure 5-43. For actor A1, it also requires to transform into place (P1) and transition (T1).

3. Class/Object transformation
In use case model transformation, actors and use cases are transformed into CPN components. These UML components are transformed to a set of CPN components respectively. One set of CPN components contains a place, a transition and an arc. An actor represents an agent in UML model representation. The agent was transformed to CPN components in step 2. In this transformation process, it is not necessary to explore the place and transition to represent the actor. But use case contains one or more agent activities, therefore, the CPN components for these use cases are able to be represented by addition CPN components.
Figure 5-47 Classes/Objects transformation

Figure 5-47 shows how the objects or classes are transformed into CPN components in this transformation process. In figure 5-43, objects/classes O1, O2 and O3 interact with each other to attain the prescribed objective of the use case (U2) in the collaboration diagram, they are transformed into transitions T4, T5 and T6 respectively. The relationships of the objects/classes are represented by the arrows M1, M2 and M3 in UML model side. These relationships are required to transform into places with arcs to connect with transition T4, T5 and T6. After the completion of these connections, T3 is completely transformed to CPN components. For each pair of these objects’ transitions, a place should be used to connect the two transitions. Therefore, an object should be represented by a transition and a place connected by an arc.

4. Operation transformation
In UML model, a class or an object comprises attributes and operations. Operations of classes/objects require to transform to CPN components. Operations are normally used in the collaboration diagram to communicate with other objects/classes. These operation linkages provide dynamic interaction information in CPN representation. Figure 5-43 shows an object O3 which has already transformed into transition T6.
Referring to the object operations, two places (P7 and P8) and two transitions (T7 and T8) are used to express the two operation of object O3. This transformation is summarized in figure 5-48. Each operation is transformed into one place and one transition and they are connected by an arc. A transition is used to control the token flow of its connected place. Each object inside an use case is resource distributor which is defined to distribute resource around the model. Each object tries to distribute its resource in order to interact with other objects. Resource distributions of the agents are governed by control statements. Therefore, the set of CPN components constitute a place with an arc connected to a transition to represent an object’s operation.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>UML model (Original UML model)</th>
<th>CPN model (After transformation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation transformation</td>
<td>Operation in object</td>
<td>One place, one transition and one arc</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Operation 1, Operation 2" /></td>
<td>Places with 2 arcs</td>
</tr>
</tbody>
</table>

Figure 5-48 Operation transformation
5. Token assignment
At the end of these transformation processes, colorsets are required to be defined. Colorsets of a CPN model represent the availability of resources of entities. For example, each FAC control system example has the limitation that it is the maximum number of pallets available concurrently. Therefore, the same number of the pallet colorset tokens should be defined. This setting limits the number of pallets for the proposed system during simulation. In FAC example 1, the maximum number of pallets coexisting in the system is six. Therefore, six tokens for the pallet colorset are required to define and the pallet colorset is named at the place as “OrderIn”.

6. Algorithm representation
After the CPN framework for the generic FAC template is constructed, this framework is able to implement a FAC control system. But this system contains only fundamental modules for shop floor devices control with necessary communication agent interfaces. In general FAC control systems, schedule and planning engines are always applied to enhance their system performance. Therefore, the algorithm application for the agents is also proposed in this framework.

Algorithm application in agents is always implemented by control statements. If-then-else, for-next loop and while loop etc are the most commonly used control statements in algorithm implementation. In the proposed agent-based control system, these control statements can be implemented into the CPN framework. The coming paragraphs will present a prototype framework with algorithm application using the generic control template.

5.2.5. Practical Mapping Application
5.2.5.1. Use case diagram refinement
FAC example 1 is used as the generic FAC template in this project. The template will also be used to implement FAC examples 2 and 3. Figure 4-6 is a collaborated use case diagram for the proposed generic FAC control system template. This collaborated use case diagram has already been integrated from several use case diagrams. Common
actors or use cases were eliminated and the collaborated diagram will be used as the practical example to show UML-CPN transformation.

5.2.5.2. Use case model transformation
The collaborated use case diagram is now ready for transforming into a CPN core framework. Transformation of the use case components to CPN components is to be established with reference to Figure 5-46. The actors and use cases are firstly transformed into CPN components and the result of the transformation becomes the core framework for the generic FAC template. Figure 5-49 shows an extraction of the collaborated use case components which this portion view contains two agents with a use case for the system. This diagram is used to illustrate a simple example in order to illustrate the use case model transformation. The transformation process has been performed effectively. This diagram shows two actors (scheduling agent and robot agent) connect to an use case (Pallet Identity Agent work) and the use case diagram is embedded in the pallet identity agent.

![Figure 5-49. Use case diagram for the use case “Pallet Identity Agent work”](image)

Figure 5-50 shows the CPN models after this transformation. A set of one place, one transition with an arc is required to represent an actor or an use case. The places of the UML components are used to define tokens for distribution. And the tokens are fired from these places according to the criteria setting of their precedence connected transitions via arcs.
For example, the robot agent possesses two tokens in its agent place. The two tokens represent the two robots in FAC example 1. One of the two tokens will be fired with the incoming pallet’s token to the transition next to robot agent. When these two tokens are run out and fired from the robot place, no token will be assigned to the subsequent incoming tokens. Therefore, the shortage of robot resource to incoming pallets is simulated and the pallet tokens queue at the robot station. These token information in simulation will be recorded and exported to a data file for analysis.

As well as the robot agent, the scheduling agent is able to assign tokens and these tokens are distributed to other agents by its scheduling result. When token passes to the transition of the scheduling agent, the control statement of the transition governs the token flowing direction to other agents. Pallet token information such as part type to produce and allocated assembly robot are distributed from the scheduling agent. The tokens for part type and robot will be delivered back to the scheduling agent until these tokens are fired back to place ProductOut. Tokens back to this place simulate the pallets unloading from the FAC when they have received assigned assembly process in example 1.

Use case “Pallet Identity Agent Work” is also required to convert to CPN components. In UML model, the activities prescribed by use case have been further represent by the collaborated diagram in figure 5-6. This diagram shows interactions between the agent objects. The use case is a collaboration model for several agent representations as shown in this figure. Because of this reason, an use case is converted to the set of CPN components as well as actors. Hence, one place, one transition connected by an arc are used to represent this use case. Place is used to distribute tokens and control statement in the transitions are used to control the flow of token around the CPN. Figure 5-50 shows the CPN model for the use case in figure 5-49.
5.2.5.3. Class/Object transformation

In section 5.2.5.2, actors and use case are transformed into CPN components and formed a CPN framework for the generic FAC control system template. The use case in the example shows CPN components of the use case collaboratively represent the other agents. This is shown by the collaboration diagram in figure 5-6. The expressions in this diagram provide more information for the entity interactions inside this use case. These entities include interactions between agents, agent’s objects and agent classes. These components in the collaboration diagram (figure 5-6) work cooperatively and their interactions are mainly comprised by TCP/IP. The objects/classes in the collaboration diagram are summarized in figure 5-51.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Agent’s object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Agent</td>
<td>wskClient2</td>
</tr>
<tr>
<td>Robot Agent</td>
<td>wskClient1</td>
</tr>
<tr>
<td>Pallet Identity Agent</td>
<td>SensorSignPostAgent</td>
</tr>
<tr>
<td>Barcode Agent</td>
<td>Com1</td>
</tr>
</tbody>
</table>

Figure 5-51 Agents and their objects in the sequence diagram representation

Use case “Pallet Identity Agent Work” contains agents interaction of four agents, they are scheduling agent, robot agent, pallet identity agent and barcode agent. With the exception of the pallet identity agent and barcode agent, other agent objects/classes in the figure have been transformed to CPN components as shown in figure 5-50. To explore the details of the use case, the two agents which do not perform CPN...
transformation in the previous steps, thereafter, they are required to transform to CPN components in this transformation step.

For each use case, several agents or their objects associate with each other to perform and complete the task in this use case. In this example, the use case “Pallet Identity Agent work” is a collaborative work performing by several agents. Two of the agents are the stimulators to this use case which they have already transformed into CPN components in the actor transformation process. The remaining two agents (pallet identity agent and barcode agent etc) are required to transform into CPN component in this class/object transformation process. These agents are also represented as actors in the collaborated use case diagram in figure 5-6 but their agent class can possibly be inherited in the collaboration diagram as objects. These objects are created inside the use case diagram to implement specific functionality for a specific use case.

For instance, the use case “Pallet Identity Agent work” is implemented in the pallet identity agent and this agent is required to create objects inherited from other agents. The inherited objects will then become routine calls to communicate to the inherited agent. The object embedded by inheritance can also reduce the complexity of graphical representation for a large scale system. Essential use case can be used to represent specific functionality and their details are also prescribed in a collaboration diagram.

Because of this reason, agent interactions are extracted one by one according to these collaboration diagram. Duplicated CPN components of agents are combined into unique CPN components. On the other hand, their new interactions to other agents or agent objects should be created according to the interaction relationships between agent objects in the collaboration diagram. The newly found agent interaction is implemented by a place with two arcs as shown in figure 5-47. This implementation is referred to object relationship for the CPN transformation. In other circumstances, if a new agent is found for the CPN framework, the implementation of a set of place and a transition connected with an arc should be created for this agent. Figure 5-52 shows the CPN
model which is further transformed by this transformation according to the object/class transformation shown in figure 5-51.

![CPN model](image)

**Figure 5-52 Partial CPN model for the example 1**

### 5.2.5.4. Operation transformation

The CPN component has already been transformed from UML class/object. Interactions between agent classes and objects are now restricted to coarse agent definitions of classes and objects. In UML representation, interactions between these agents are described by their attributes and operations. These attributes contain system resources and these resources are allocated by the operation control statements. The operations of objects or classes constitute agent interactions for the control system. They link up the objects/classes for the system as shown in the UML collaboration diagram.

In UML representation, it may not be necessary to create operation in agent. If the required operation exists in other agents, the operation can be reused by using OO characteristic “implement”. This characteristic reuses the existing agent method to call an agent function but this agent does not physically exist inside the agent caller. For the
transformation of CPN model, the aim of using the CPN model is to construct a simulation structure, in which, the CPN model is constructed according to the interactions between agents. Therefore, operation transformation is necessary to perform which the operation is treated as the agent physical existing in the CPN model.

For the operation transformation, if a new agent is found in the agent interaction by operation call in the collaborated diagram, set of a place and transition connected by an arc is created to the CPN framework to represent the agent for the method call. On the other hand, a place with two arcs are used to create new interaction between the CPN components for the two interacted agents.

Using this transformation, more agent interactions described by operations in collaboration diagram are able to represent by the CPN model. Moreover, the details of the operation transformation are able to express. The operation transformation example among the agents is listed in figure 5-53:

<table>
<thead>
<tr>
<th>Agents</th>
<th>Operation for the agents linkage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Agent, Pallet Identity Agent</td>
<td>Send(), check()</td>
</tr>
<tr>
<td>Pallet Identity Agent, Barcode Agent</td>
<td>RetDInfo(), receive()</td>
</tr>
</tbody>
</table>

Figure 5-53. Agents and their linkages

Figure 5-54 shows a completed CPN model for the generic FAC control system template where the shaded area represents the CPN content which has been described in figure 5-52. These CPN models are transformed and completed by operation transformation. Figure 5-55 summarizes the completed place representations for the generic agent FAC template. $P_{\text{OrderIn}}$ and $P_{\text{ProductOut}}$ are the additional places added to the generic template. They are the assigned places of system entry and exit for the FAC system respectively.
Figure 5-54. CPN model for the generic FAC control system template
<table>
<thead>
<tr>
<th>Actor/Object/Class/Operation</th>
<th>Place</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>$P_{\text{OrderIn}}$</td>
<td>Place for Order Incoming to the example FAC</td>
</tr>
<tr>
<td>Scheduling agent</td>
<td>$P_{\text{OHA}}$</td>
<td>Order Handling Agent</td>
</tr>
<tr>
<td></td>
<td>$P_{\text{SA}}$</td>
<td>Scheduling Agent</td>
</tr>
<tr>
<td>Robot agent</td>
<td>$P_{\text{RA}}$</td>
<td>Robot Agent</td>
</tr>
<tr>
<td>Conveyor Control Agent</td>
<td>$P_{\text{CCA}}$</td>
<td>Conveyor Control Agent</td>
</tr>
<tr>
<td>Barcode Agent</td>
<td>$P_{\text{BA}}$</td>
<td>Barcode Agent</td>
</tr>
<tr>
<td>Pallet Identity Agent</td>
<td>$P_{\text{PIA}}$</td>
<td>Pallet Identity Agent</td>
</tr>
<tr>
<td>/</td>
<td>$P_{\text{ProductOut}}$</td>
<td>Place for Product Out to the example FAC</td>
</tr>
</tbody>
</table>

Figure 5-55. Places assignment from agents for the generic FAC control system template

These CPN places are represented by the set $P$ where $P = \{P_{\text{OrderIn}}, P_{\text{OHA}}, P_{\text{SA}}, P_{\text{RA}}, P_{\text{CCA}}, P_{\text{BA}}, P_{\text{PIA}}, P_{\text{ProductOut}}\}$ in the generic agent-based FAC control system template.

After the descriptions of the assigned places, the transitions of the proposed FAC control system template are the next components to assign. The transitions are named according to the listed operations. For example, transition between scheduling agent and pallet identity agent possesses operations “send()” and “check()”, they are collaboratively named as “Process Order”. Detail descriptions of the operations such as “send()” and “check()” can be separately described by another sub graph for the transition “Process Order”. The sub-graph creation provides flexibility for system developers to express other descriptions for the same system without changing the original CPN model (Figure 5-57).

The completed CPN model in figure 5-54 provides elementary interfaces to all the agents for the proposed FAC system template. Other transitions for the generic FAC control system are then summarized in figure 5-56. The transitions of the CPN model for the generic FAC control system template are then represented by the set $T$ where,

$$T = \{T_{\text{EnterOrder}}, T_{\text{ProcessOrder}}, T_{\text{PalletTransfer}}, T_{\text{PalletIdentityProcess}}, T_{\text{ProductionProcess}}\}$$
<table>
<thead>
<tr>
<th>Use case</th>
<th>Transition in CPN</th>
<th>Mathematical representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling agent work</td>
<td>EnterOrder</td>
<td>$t_{EnterOrder}$</td>
</tr>
<tr>
<td></td>
<td>ProcessOrder</td>
<td>$t_{ProcessOrder}$</td>
</tr>
<tr>
<td></td>
<td>Loading/UnloadingProcess</td>
<td>$t_{Loading/UnloadingProcess}$</td>
</tr>
<tr>
<td>Conveyor control agent work</td>
<td>PalletTransfer</td>
<td>$t_{PalletTransfer}$</td>
</tr>
<tr>
<td>Pallet identity agent work</td>
<td>PalletIdentityProcess</td>
<td>$t_{PalletIdentityProcess}$</td>
</tr>
<tr>
<td>Robot agent work</td>
<td>Production Process</td>
<td>$t_{ProductionProcess}$</td>
</tr>
</tbody>
</table>

Figure 5-56 Transitions for the proposed FAC control system template

Figure 5-57 Schematic diagram to the place and transition with respect to operations
5.2.5.5. Token assignment

After the places and transitions are completely transformed into CPN model, tokens should then be assigned. At the beginning of system execution, pallets are loaded to the FAC. Figure 5-58 outlines the flow of pallets in the FAC. Any finished part will be unloaded from pallet when it is conveyed back to the loading/unloading station. New parts for machining process will be replenished into this pallet and the pallet will then be loaded again into the FAC. A new cycle of the FAC operation to this pallet is thus repeated.

![Flow chart for the system execution of the FAC control system](image)

Figure 5-58 Flow chart for the system execution of the FAC control system
Figure 5-59 shows the six color tokens (A, B, C, D, E and F) assigned in the FAC control system example 1. The maximum number of the pallets coexisting is limited to six in the example constraint, therefore, the six tokens are assigned. The labels “ordent” in the figure are a variable setting that these variables permit any of the six tokens to flow around the labeled CPN components.

Orderin and ProductOut in figure 5-59 are places added to distribute in the system entrance and collect the tokens at the system exit. These places are assigned programming scripts to record and export the product in and product out data for analysis. Figure 5-60 is a schematic diagram for these product in and out operations.
Figure 5-60 Schematic diagram for the product in and out operation

Scheduling agent

The product in and out CPN sub-model can now be grouped with the scheduling agent as shown in figure 5-61. This figure captures the interactions of the scheduling agent and the order handling agent. After the pallet has assigned a pallet token (any one of these tokens: A, B, C, D, E or F), the pallet is ready to move around the FAC. The scheduling agent then works collaboratively with the place OrderHandlingAgent. This place belongs to the Order handling agent in the system. The information of the pallet is now entered into the transition EnterOrder which is one of the transitions for the scheduling agent. Referring to the UML model, it is the operation handled by the scheduling agent object “client” shown in figures 5-1 and 5-8.

A “true” or “false” token from the Order handling agent is used to simulate the agreement of the two agents. The two agents, scheduling agents and order handling agent, communicate with each other to negotiate the order arrangement. The “true” token will fire to the transition “EnterOrder”. By now two tokens have been fired by this transition to the output place “Scheduling agent”. Once the two tokens are in the input place “Scheduling agent” of “ProcessOrder”, the process order will communicate with the Pallet identify agent for the arrangement of pallets to the right robots. Figure 5-61 summaries the above process and figure 5-62 shows the flow chart for the scheduling agent interaction.
Figure 5-61 The scheduling agent interactions in CPN model

Figure 5-62 Summary for the scheduling agent interactions
**Pallet identity agent and barcode agent**

Within the CPN model of the FAC, a pallet is identified by the pallet token. Based on the updated information of the token such as the required assembly process by the specific robot, the pallet identity agent arranges the pallet to the right robot. During the pallet transportation, the pallet identity agent works with the barcode agent. The barcode agent will update the pallet locations periodically and this is done by information retrieval via the barcode of the pallets. Figure 5-63 shows the CPN sub-model related to these two agents and the processes of these two agents are summarized in figure 5-64.

![Figure 5-63 CPN models relate to the Pallet identity agent and barcode agent](image)

![Figure 5-64 Flow chart for the pallet identity agent and barcode agent](image)
Conveyor control agent and robot agent

The conveyor control agent receives the updated pallet location information from the pallet identity agent. The conveyor control agent will then transfer the pallets to the assigned robots. The conveyor control agent will assign tokens to the incoming pallet. There are totally 6 tokens in the place conveyor control agent. These tokens ensure the number of pallets inside the FAC does not exceed its maximum limit. The pallets are then conveyed to the right robots for assembly operations. The robot agent will perform the required assembly process by the transition ProductionProcess. After a pallet has finished its assembly process, the conveyor control agent token will be released back to the agent and the pallet token will be sent back to the place “ProductOut”. Figure 5-65 shows the CPN sub-model for these two agents. The corresponding agent operations are shown in figure 5-66.
The CPN model for the generic agent-based FAC control system is now completely defined. The CPN model is also the framework for the FAC control system. Figure 5-67 shows the CPN model.

Complete CPN model

The CPN model for the generic agent-based FAC control system is now completely defined. The CPN model is also the framework for the FAC control system. Figure 5-67 shows the CPN model.
Figure 5-67 CPN for the generic agent-based control system
A generic agent-based control system template has been proposed. The transformation process for this generic template presents a generalized conversion methodology from UML to CPN models. Using the generic template, different FAC control systems can be developed with shorten development cycle. The basic generic FAC template enhances the productivity of the system design and analysis to both the process and product changes upon the existing agent framework. Time and effort can be saved in designing new FAC control systems. These features are benefited by the UML’s encapsulation characteristic and the transformation of UML-CPN model.

The generic template is proposed according to the requirements of FAC example 1. The control system is an agent-based control system which the control components possess distributed abilities. The general mathematical description of the CPN models for the control system is:

For the generic agent-based control system, the CPN, \( C = (\mu_{ColSet}, P, T, I, O) \), where,

\[
P = \{ P_{\text{OrderIn}} , P_{\text{OHA}} , P_{\text{RA}} , P_{\text{CCA}} , P_{\text{BA}} , P_{\text{SA}} , P_{\text{PLA}} , P_{\text{ProductOut}} \}
\]

\[
T = \{ T_{\text{EnterOrder}} , T_{\text{ProcessOrder}} , T_{\text{PalletTransfer}} , T_{\text{PalletIdentityProcess}} , T_{\text{ProductionProcess}} \}
\]

\[
I = \{ \begin{align*}
I(t_{\text{LoadingUnloadingProcess}}, P_{\text{ProductOut}}) , & \quad I(t_{\text{EnterOrder}}, P_{\text{OrderIn}}) , \quad I(t_{\text{EnterOrder}}, P_{\text{OHA}}) , \\
I(t_{\text{ProcessOrder}}, P_{\text{SA}}) , & \quad I(t_{\text{PalletIdentityProcess}}, P_{\text{PLA}}) , \quad I(t_{\text{PalletTransfer}}, P_{\text{BA}}) , \quad I(t_{\text{PalletTransfer}}, P_{\text{CCA}}) , \\
I(t_{\text{ProductionProcess}}, P_{\text{RA}}) & \end{align*}
\}
\]

\[
O = \{ \begin{align*}
O(t_{\text{LoadingUnloadingProcess}}, P_{\text{OrderIn}}) , & \quad O(t_{\text{EnterOrder}}, P_{\text{SA}}) , \quad O(t_{\text{ProcessOrder}}, P_{\text{PLA}}) , \\
O(t_{\text{PalletIdentityProcess}}, P_{\text{BA}}) , & \quad O(t_{\text{PalletTransfer}}, P_{\text{RA}}) , \quad O(t_{\text{ProductionProcess}}, P_{\text{ProductOut}}) & \end{align*}
\}
\]

\( P \) represents the set of agents in the proposed control system, and \( T \) represents the set of events for the proposed control system. The description of the input arcs and output arcs are listed in figure 5-68 and figure 5-69 respectively.
<table>
<thead>
<tr>
<th>Input Arc</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I\left(t_{\text{LoadingUnloadingProcess}}, P_{\text{ProductOut}}\right)$</td>
<td>Input arc for transition $t_{\text{LoadingUnloadingProcess}}$ from place $P_{\text{ProductOut}}$</td>
</tr>
<tr>
<td>$I\left(t_{\text{EnterOrder}}, P_{\text{OrderIn}}\right)$</td>
<td>Input arc for $t_{\text{EnterOrder}}$ transition from place $P_{\text{OrderIn}}$</td>
</tr>
<tr>
<td>$I\left(t_{\text{ProcessOrder}}, P_{\text{OHA}}\right)$</td>
<td>Input arc for $t_{\text{ProcessOrder}}$ transition from place of Order Handling Agent</td>
</tr>
<tr>
<td>$I\left(t_{\text{ProcessOrder}}, P_{\text{SA}}\right)$</td>
<td>Input arc for $t_{\text{ProcessOrder}}$ transition from place of Scheduling Agent</td>
</tr>
<tr>
<td>$I\left(t_{\text{PalletIdentityProcess}}, P_{\text{PIA}}\right)$</td>
<td>Input arc for $t_{\text{PalletIdentityProcess}}$ transition from place of Pallet Identity Agent</td>
</tr>
<tr>
<td>$I\left(t_{\text{PalletTransfer}}, P_{\text{RA}}\right)$</td>
<td>Input arc for $t_{\text{PalletTransfer}}$ transition from place of Barcode Agent</td>
</tr>
<tr>
<td>$I\left(t_{\text{PalletTransfer}}, P_{\text{CCA}}\right)$</td>
<td>Input arc for $t_{\text{PalletTransfer}}$ transition from place of Conveyor Control Agent</td>
</tr>
<tr>
<td>$I\left(t_{\text{ProductionProcess}}, P_{\text{RA}}\right)$</td>
<td>Input arc for $t_{\text{ProductionProcess}}$ transition from place of Robot Agent</td>
</tr>
</tbody>
</table>

Figure 5-68 Input Arc references for the generic agent-based control system
In a typical FAC control system, intelligence is always applied by the control entities in order to provide guidelines in automatic control system. The intelligence might be rule, intelligent algorithm or bidding mechanism etc. The generic framework template is therefore acted as the core structure which new intelligent application is embedded inside the agent object. The agent object is therefore to transform into CPN model which this new agent characteristic can also be simulated by the CPN model. The next section is going to present such a transformation methodology to the intelligent structures, which they are normally comprising by control statements.

5.2.5.6. Algorithm representation

A core FAC control system template has already been introduced from section 5.2.5.1 to 5.2.5.5. The proposed UML and CPN models contain essential control entities for a general FAC. For example, scheduling agent, robot agent, conveyor control agent etc are the agents which are necessary and essential in a typical FAC control system. The constitution of these agents is already defined in formal specification using UML and CPN. The generic template can be used to implement other FAC control systems.
Different FACs are not simply implemented by the template. The objects defined in the proposed FAC contain commonly used control entities in a FAC, reusing the structure can save effort to define every entity one by one. Having a generic framework structure, basic agent interactions are ready to be used to design a FAC control system. For more complicated control system, different rules or algorithm application etc is the next process to implement into this agent template.

The FAC control template is implemented in a computer control environment. Hence, intelligent algorithm application normally comprises computational control statements. In general, statements such as if-then-else, for-while-loop and for-next loop are the most commonly used control statements in programming languages. Complicated heuristic statements or control algorithms are to be created by these statements.

Data expression and model declaration
Design/CPN (http://www.daimi.au.dk/designCPN/what.html) is a CPN editing tool and it possesses simulation functionality. This tool is developed and implemented by the University of Aarhus, Denmark. Figure 5-71 shows a simple CPN example in the tool. In this example, one place and a transition is connected by two arcs. Symbols P1 and T1 denote place and transition respectively and the token is defined at place P1. “Ord” is the token variable and it contains two sets of color tokens, A and B. The color tokens are not restricted to be denoted by color representation; they can also be distinguished by any letter, number or symbol. In the same figure, some programming liked statements in the rectangular box is called declaration area. In this area, data expressions such as variable, string, integer, constant are declared. Inside this box, the following statement is expressed:

\[
\text{color Ord = with A|B;}
\]

This statement declares the name of color token “Ord” and it contains color tokens “A” and “B”. A symbol “1`A + 2`B” at the place P1 (Figure 5-70) represents the
quantities of the tokens, they are one token of A and two tokens of B. In the graphical representation, there are two arcs and the symbols “orderin” are marked on the arcs. The “orderin” constrains the type of tokens flowing across the arcs. In this example, token A or B is permitted to flow across the arcs. Furthermore, the guard (statement at the top of the transition) of the transition states “if orderin=A then orderin else orderin=nil” and it is another control statement to guide the token flow. This statement and the guard provide flexibilities to control the flow of tokens.

Figure 5-70 A simple CPN model in Design/CPN

The declaration statement is designed according to the programming format and syntax of ML program. The programming language provides wide varieties of control statements and program expressions as well as the programming languages liked C++, Java, VB etc. Most commonly used algorithms can also be applied in this program. The following contents are the most commonly used control statement in the programming language. Three more examples of the FAC control systems will be implemented using the following statements.
Conditional statements in CPN representation

*If – Then – Else*

This control statement will return a prescribed value. It is possible to define multiple values but they are returned according to the checking condition. The following example shows how the if-then-else works

\[
\text{If (Conditional statement) then (Boolean value return)}
\]

\[
e.g. \text{If (pallet arrives) then (pneumatic actuator is activated to lift the pallet up)}
\]

\[
\text{but if (pallet does not arrive) then (pallet is kept conveying)}
\]

Control statements can also be implemented to the CPN models. This conditional statement can be implemented by the guard in transition or the arc statement.

**Example:**

\[
\text{if orderin = A then resource = C else resource = D}
\]

The CPN model example in figure 5-71 has 3 places (P1, P2 and P3) and 1 transition (T1). The two places P1 and P2 contain tokens for firing. P1 contains tokens A and B and P2 contains tokens C and D. The transition T1 will fire one of the tokens (A or B) randomly from the place P1 to the transition T1. The firing process will be terminated until all tokens have been fired. Tokens to be fired to P3 are determined by the transition T1. There is a conditional statement in the guard of the transition T1. If the token A in P1 is moved to T1, token C in place P2 will be moved. The two tokens from place P1 and P2 will then be fired to the place P3 subsequently. On the other hand, if token B is removed from the place P1 to the transition T1, token D will be removed.
from the place P2 instead of token C. Selection of the token to be fired from place P2 is governed by the guard of the transition T1. Finally, the tokens removed from the place P1 and P2 will be fired to place P3. The pairs of tokens are A and C or B and D to the place P3.

*For-next loop*

In CPN, the for-next loop counts the number of tokens passing specific place or transition. The representation of the for-next loop in CPN as follows:

Figure 5-72 shows an example for the for-next loop in CPN representation. There are 2 places (P1 and P2) and two transitions (T1 and T2). The place P1 carries token A at the initial state of CPN simulation. The transition T1 removes token A from P1 and adds this token to transition T1. And the token will be fired to P2 subsequently. Noted that the arc between T1 to P2 has a marking “orderin, n+1”. This marking of the arc can carry two kinds of tokens concurrently. They are the tokens in the variable “orderin” and the counter variable “n”. “n” is declared as zero at the initial state of simulation, the first trial of token firing from the places P1 to P2 via the transition T1 will result n=n+1=0+1=1 for the counter variable n. As the result of this counter in the first trial (n=1) can not satisfy the requirement of n=4 in the guard of the transition T2. The token n=1 will send back to T1 from P2. According to the setting of this guard, the token “A” at the token variable “orderin” will only be fired from the place P2 to transition T2 until the variable of n reaches 4.
Illustrative examples of the control statement application in CPN model

In the following paragraph, an example with two agents is extracted. Activities of the two agents become logical with the control statements. The scheduling agent and pallet identity agent are the two agents in this illustrative example. Figure 5-73 shows the original interactions between these two agents in CPN model:

![CPN Model Diagram](image)

Figure 5-73 Illustrative example showing CPN model

The scheduling agent (SA) and pallet identity agent (PIA) are interacted by the event “Order allocation”. The transition “Order allocation” will fire tokens one by one from the place SA to PIA. These tokens include “A”, “B” and “C” and they represent stamps for different robot process requirements. For example, the token A requires robot A operation and the token B required robot B operation. The last token, C does not require any robot assignment.

Figure 5-74 shows the modified CPN model for the model in figure 5-73. This figure is called the transition page in Design/CPN. Activities happening inside this page are the exploration description for the transition “Order Allocation” in figure 5-73. The incoming token from the place SA in figure 5-73 is the input for the transition “Order Allocation” in this figure. Simple decision making criteria is implemented in this model. The flow chart in figure 5-75 reveals the functionality of this CPN model. At the beginning of the flow chart, the system checks the type of incoming token which is brought from the input arc of the scheduling agent(SA) place. The incoming token determines the sort of robot assignments. For example, the scheduling agent allocates robot A’s process to the incoming token of A and allocates robot B’s process to the token B. After suitable robot assignments have been allocated to these incoming tokens,
token will fire to correct robot’s place. This process simulates the transportation of pallets to the assigned robots in the FAC. With the exception of token C, tokens A and B will then be sent back to transition “Order allocation” at the end of this CPN model.

If orderin = B then rob = b else rob = a

if rob != nil then [orderin, rob]

orderin, rob

orderin, rob

orderin, rob

If orderin = B then rob = b
else rob = a

If token=nil then wait for next resource allocation

If token = nil then wait for next resource allocation

Robot A process

Robot B process

Robot A

Robot B

Token C?

Yes

Yes

No

No

Release robot token roba or robb to the next waiting pallet

Check the type of incoming tokens and assign robot for pallet processing

Any robot assignment

Transfer pallet to the assigned robot
Figure 5-75 Flow chart for the example CPN model for the substitute transition page

This simple example illustrates how conditional statements are presented in the CPN model. After the CPN model is constructed, it can simulate the operations of the control system prior the program implementation stage. Interactions between control elements (agents) and required components can be measured using the result of simulation. The CPN model can be modified after simulation. The modification is performed iteratively until system performance is satisfied. Because the structure of the CPN model is similar to the actual programming environment, the CPN model helps the system designer to communicate with the programmer in the later coding stage. And the CPN model can also be used as the test bed for further system modification and amendment.

5.3. Practical FAC control system example
5.3.1. Example 1
The CPN model for the FAC example 1 is illustrated in the figure 5-54 which it is also the generic FAC control system template for the examples 2 and 3.

5.3.2. Example 2
Figure 5-67 shows a CPN model for example 2. The configuration of this example has been presented in chapter 3. The agents of this FAC control system include scheduling agent, barcode agent, pallet identity agent, conveyor control agent, robot agent and order handling agent. Place OrderIn and ProductOut are the places for system input and output. The model in figure 5-76 is the substitute page for this main model. The substitute page in figure 5-76 is referred to a specific transition (ProcessAgentProcess) example. Markings such as “P” and “I/O” at the places “RobotAgent”, “Barcode Agent”, “ConveyorControlAgent” show that these places have already been defined in the main CPN model. In order to complete the process in this substitute page, new place or transition is permitted to create. Structure of the substitute page is similar to the subroutine of programs. That’s why the substitute page is also called the sub-routine structure in CPN.
Figure 5-76  Main CPN model for the FAC control system
Figure 5-77 shows the mapping summary for the UML and CPN models for the control system for example 2.

<table>
<thead>
<tr>
<th></th>
<th>UML model</th>
<th>CPN model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Resource distributor</strong></td>
<td>Scheduling agent request (Actor)</td>
<td>Scheduling agent (Place)</td>
</tr>
<tr>
<td></td>
<td>Pallet identity agent (Actor)</td>
<td>Pallet Identity agent (Place)</td>
</tr>
<tr>
<td></td>
<td>Barcode agent (Actor)</td>
<td>Barcode Agent (Place)</td>
</tr>
<tr>
<td></td>
<td>Robot Agent connector (Actor)</td>
<td>Robot Agent (Place)</td>
</tr>
<tr>
<td><strong>Activity requirement area</strong></td>
<td>Scheduling agent work (use case)</td>
<td>EnterOrder (Transition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessOrder (Transition)</td>
</tr>
<tr>
<td></td>
<td>Conveyor control agent work (use case)</td>
<td>PalletTransfer (Transition)</td>
</tr>
<tr>
<td></td>
<td>Pallet Identity Agent work (use case)</td>
<td>PalletIdentifyprocess (Transition)</td>
</tr>
<tr>
<td></td>
<td>Robot agent work (use case)</td>
<td>ProductionProcess (Transition)</td>
</tr>
</tbody>
</table>

![Figure 5-77 Summary for the mapping between UML model to CPN Model](image)

Although the CPN model in figure 5-67 is simple, the model is designed to evaluate important interactions between agents. For example, refer to the sequence diagram (figure 5-1) of the scheduling agent, the activities described in this diagram are simplified to transitions “EnterOrder”, “ProcessOrder” and a place “OrderIn”. Figure 5-78 shows the summary for the representations.
The CPN model for the pallet identity agent is simply described by transition “PalletIdentifyProcess”. This agent works closely with the barcode agent through the method “RetIDinfo()” in figure 5-2. This is a method for Pallet identity agent to identify the pallet information by retrieving the barcode scanning result.

The conveyor control agent will send the verified pallets for robot processes. The pallet transfer process is represented by the “PalletTransfer” transition in the CPN model. “PalletTransfer” is a complicated process and it requires cooperation between the conveyor motion controller device and the pallet identity process. This can be revealed in the sequence diagram for the use case “Conveyor control agent work” (Figure 5-4).

The FAC configuration example 2 is different from the generic FAC configuration. The hypothetic configuration provides one more assembly robot. Branches of conveyors are also found in this configuration. The pallet transfer system is more complicated when compared with the simple loop of conveyors in the generic FAC example.

Although the configurations between the two FAC examples are different, the generic control system example can also be used as the framework for design and analysis of the new FAC control system. Figure 5-79 shows the CPN model which is modified from the generic control system framework.

<table>
<thead>
<tr>
<th>CPN models</th>
<th>Sequence diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnterOrder</td>
<td>Method of check(), check1(), check2(), check3() to retrieve information from commonData instance.</td>
</tr>
<tr>
<td>ProcessOrder</td>
<td>checkinteger() method is used to send the information of the pallet to other agents in order to make announcement for the robot processing requirement.</td>
</tr>
</tbody>
</table>

Figure 5-78 Mapping from sequence diagram to CPN model
The modified framework will undergo simulation later and the performance of this configuration can be evaluated and analyzed.
5.3.3. Example 3

The generic FAC template is reused to implement this FAC control system. This agent-based template is found suitable to implement the control system. The scheduling agent is responsible for pallet allocation to the twelve machines. The conveyor agent and pallet identity agent are responsible for pallets transfer to the allocated machines. Pallet identifications and pallet locations are monitored by the pallet identity agent and these pallets is loaded and unloaded in a destination area.

Maximum allowable number of pallets in this FAC is twelve and it equals the number of the machines. Referring to the generic template in figure 5-66, the tokens for place “Orderin” are modified as A to H which they are eight different color token types to represent the maximum allowable pallets. Figure 5-81 shows the modification of the CPN model for this FAC example.

![Figure 5-81 Modified CPN model for the place OrderIn](image)

The machine control agent is inherited from the robot agent which was previously designed to perform production process. Device control via RS232 is also the standard interface for the machines. This newly inherited agent object controls the eight
machines. Figure 5-80 shows changes of the CPN models for the generic FAC to this hypothetic FAC example. A set of components should also be added to the CPN model. These newly added components include the increased number of machines as the tokens in this example. The shaded place and shaded transition in figure 5-79 are the new components for the machine control agent.

Figure 5-80. Comparison of the CPN models of the generic FAC control system to example 3
In figure 5-81, the un-shaded place and transition are the CPN components for the machine control agent of the generic template. Twelve new tokens of MC in the figure are assigned to the place to represent the twelve machines. When a pallet is moved by the robot to a machine, machine control agent will allocate a MC token to this pallet. In some case, if all the MC tokens are occupied, no MC token will allocate to this pallet. A MC token will be released from the machining center which they finish required process and the token is released back to the un-shaded place. The token releasing action simulate the machining resource is released from fully occupy status.

![Diagram](image)

**Figure 5-81. New CPN components control the number of machines in the FAC example**

The substitution page should also be created in the machine control agent place. In figure 5-78, example 3 has twelve machines for machining processes. The figure shows clearly the twelve transitions for the machines which they are named as A, B and C, D, E to L. In this example, twelve machines are proposed for performing manufacturing processes, therefore, twelve new transitions should be created to represent the machines. Figure 5-82 shows the substitution page for the transition of the machine control agent.
In figure 5-82, places C1, C2, C3 and C4 represent the four conveyors which are responsible for conveying pallets to the twelve machines. Each of these conveyors’ place attaches with one place and transition. The four places have been allocated three tokens to represent the three machines located at each conveyor. The incoming pallet token at the robot agent will fire to the shaded place which is the machining control agent’s place. Machining control agent will then fire the incoming token at the substitution page back to the CPN components in main structure as shown in figure 5-81. In this substitution page, token activities to any one of the four conveyors’ place are able to export to the simulation file for later analysis stage.
Except the addition of the CPN components for machine control agent, the existing robot agent is necessary to modify in this example. New place and transition with token should create for this agent. One robot is required in this example, therefore, new CPN components should be created as shown in figure 5-83. As well as the changes in figure 5-82, new place and transition with token are necessary to create to the robot agents. One RA tokens is assigned to the new created place to represent the robot in the loading/unloading area of the FAC example. A RA token will be fired together with a pallet token to the shaded machine control agent’s transition when any one of the pallet tokens(A to L) is fired to the robot agent place. When the “RA” token is released from this place, no token will then be assigned to new incoming pallets. This situation simulates the next arrived pallet will queue at the loading/unloading station until the previous pallet is freed from busy status.

![Diagram](image)

Figure 5-83. Modified CPN model for the robot agent to the new example

Four product types are able to manufacture in this FAC example, so four different types of color tokens should be created at place “ProductIn” of the generic agent control system template as shown in figure 5-67. Each of the color tokens possesses three tokens respectively. “P1” represents token for product 1 and “P2” represents token for
product 2 etc. Two different color token types are required to draw out from the place “ProductIn” to the transition “EnterOrder”. For example,

Token firing example 1 “P1, A”
Token Firing example 2 “P2, H”

The first example represents a pallet token “A” is fired together with one P1 token from place “ProductIN” to transition “EnterOrder” via the connected arc. The pairs of tokens will be fire throughout the CPN model until the token released from the transition “Loading/Unloading Process”. The pairs of tokens simulate an assignment in the FAC control system. This is the assignment of pallet “A” that the pallet requires machining center to perform product 1 machining processes.

The differences between the tokens “P1”, “P2”, “P3” and “P4” are the machining time settings in CPN model. These tokens will be held by one of the conveyors (C1, C2, C3, C4, C5, C6) by the default time settings. The substitution page for the machine control agent transition at this figure simulates the required processes. Pseudo codes of the program in the machine control agent transition is proposed to handle this situation, for example,

If token “P1” then machining center requires 190 seconds to complete the product 1 machining processes
If token “P2” then machining center requires 370 seconds to complete the product 2 machining processes
If token “P3” then machining center requires 250 seconds to complete the product 3 machining processes
If token “P4” then machining center requires 310 seconds to complete the product 4 machining processes

These statements have been coded to the machine control agent transition. If “P1” arrives with any incoming pallet token to the machine control agent substitution page, the product type tokens will be held in machining centers as shown in figure 3-18. Timed-CPN is then used in this example to handle the product type varieties in the example 3.
5.4. Conclusion

The use case of the static UML diagram is able to describe the occurrences of events in the FAC but it lacks the capability to undertake dynamic analysis and analytical capability. Color Petri Net (CPN) is used for the representation of dynamic behavior in the FAC control system. UML and CPN are complementary in the representation of FAC control systems. After the preliminary static framework introduced in Chapter 4, the Color Petri Net (CPN) model is then established to represent the interaction among agents. Mapping between UML and CPN is introduced in this chapter. And the CPN model for the generic FAC control system is presented. The CPN models can cope with the increased complexity of manufacturing control system when compared with traditional PN models. It provides a framework for the construction and analysis of the distributed and concurrent systems. CPN can also provide primitive programming language for data types (Color tokens) and primitive for construction of synchronized real time systems. Through color tokens, resources allocation by agents are formally described and thus suitable for agent-based designed control system. After the preliminary static model of the FAC control system has been reviewed by CPN analysis, the UML model is modified. The modified model is then converted into a programming framework for subsequent implementation of the FAC control system. Hence, much time and effort can be saved by these tools in the development of FAC control systems.
CHAPTER 6

PALLET ALLOCATION USING TABU SEARCH METHOD

Modeling of the FAC control systems using UML and CPN has been presented in the previous chapters. The dynamic representation using CPN models was introduced. The CPN models can adapt to the configuration changes of the FAC. In this chapter, a scheduling heuristic based on Tabu Search is introduced. In autonomous manufacturing systems, such as an FMS, reconfiguration of the material transfer systems is always required to cope with the frequent product and process changes. As illustrated in this chapter, the Tabu search algorithm is applied to the scheduling agent which is responsible for material transfer control in the FAC control framework. The three FAC configuration examples are used to show how Tabu search is implemented and represented in the agent-based FAC frameworks.

6.1. Introduction

Scheduling and the related planning and control activities are complex manufacturing functions and they have to change dynamically according to different setups, unique environments and process planning. The system performance is affected by factors such as precedence constraints, due dates, production levels, lot-size and priority rules etc. Gologlu (2004). Scheduling problems in the flexible job shop can be decomposed into two categories, that is, routing problems and operation scheduling problems (Scrich et al., 2004). These problems are considered as among the most difficult problems in combinatorial optimization.
The kind of approximation algorithms - meta-heuristics have been widely applied to combinatorial optimization problems. Among the meta heuristics, genetic algorithms and other evolution algorithms have been applied to solving optimization problems and they are empirical algorithms to solve problem in a feasible region. However, they mostly demand high effort and time for generating the optimum solutions (Smith et al, 2004; Raman et al, 2004).

Among these meta-heuristics, tabu search is regarded as an effective algorithm with adaptive memory when compared with memory-less type heuristics (Simulated Annealing) or rigid memory type heuristics (Evolutionary Algorithm). The memory-less algorithms of meta heuristics do not solve problem economically and effectively, and the rigid memory type algorithm do not provide feasibility to solve problems in a fast pace. In this regard, the tabu search-based algorithm with adaptive memory is able to guide the search upon a local search domain to overcome these difficulties.

Tabu search is an optimization technique that extends local search algorithms with a memory function. The efficiency of the search is obtained by the memory function to avoid revisiting the local optimum in the feasible solution space. The methodology was first presented by Glover in 1986 (Glover, 1986) and then developed subsequently (Glover, 1989; Glover, 1990). Applications of this algorithm include scheduling problems, routing problems or traveling salesman problems (Laguna et al., 1996; Landrieu et al., 2001).
Tabu search-based algorithms have been used in real time control applications (Kattan et al., 2003) and were found applicable to FAC control systems. In manufacturing, tabu search has been applied to wide areas of manufacturing applications such as manufacturing cell design, job shop scheduling, process plan optimization and manufacturing system (Jason et al., 2001). In complex manufacturing problems such as job shop scheduling, tabu search techniques have been successfully applied and they are found to be effective in the convergence of system optimization. These advantages are shown in examples including the part type selection problem in FMS (Arikan et al., 2003), job shop scheduling (Finke et al., 2002) and assembly flow shop problem (Allahverdi et al., 2004). As revealed in these examples, the advantages and disadvantages of Tabu search are:

Advantages:
- It can minimize the inclusion of duplicate search hence it can reduce the processing time for optimization problems.
- It is a fast moving searching algorithm with effectiveness. The memory function in the algorithm guides the search to avoid revisiting the local optimal.

Disadvantages:
- Similar to other meta-heuristics, tabu search does not guarantee exact optimum solution.

In this chapter, a Tabu search-based technique and rule based application are established and implemented in the scheduling agents of the three examples to solve the pallet dispatching and routing problems. A distinguishing feature of tabu search
is its exploitation of adaptive forms of memory, which allows the search to penetrate complexities that often confound alternative approaches. This feature is useful in the generation of an effective pallet routing schedule for processing in the FAC examples. These application also enhance the event distribution of machine utilization rates of the machining centers. The performance of the FAC control system will be simulated and the three example results will be compared and evaluated by using the CPN models. The revised result will map back to the UML models and the coding frameworks of the examples will then be converted into programming codes. In this regard, the flexibility of the generic FAC control system is shown by the configuration and design changes.

6.2. Tabu search algorithm for FAC examples

The overall approach of Tabu search is to avoid retracing in cycles by forbidding or penalizing moves which take the solution, in the next iteration, to points in the solution space previously visited (hence "tabu") (Landrieue et al, 2001; Geyik et al, 2004). A general procedure of Tabu search is as follows:

1. Start with an initial solution, store it as the current seed and the best solution (tabu list).
2. Generate neighbors of the current seed solution by a neighborhood structure.
3. Select a neighbor which is not tabu or satisfies a given aspiration criterion and move it as seed solution
4. Update the tabu list
5. Store the new seed as the best solution, if the new seed solution is better for an objective function.
6. Repeat steps 2-5 until a termination criterion is satisfied.

6.2.1. Tabu Search definition

In Tabu search, the problem should always be a minimizing or maximizing function, that is, optimize $f(x)$ subject to $x \in X$, where $X$ is the set of feasible solutions. In this example, a solution $X$ represents the traveling distance of the pallets.

In the actual formulation of practical optimization problems, the function $f(x)$ is mostly nonlinear and the system can not be represented by an exact mathematical function and equation. With Tabu search, there are important functions to avoid duplicating a previous search and revisiting a solution in a local search, namely, recency-based memory, frequency-based memory and aspiration criteria.

In this chapter, the recency-based memory is used in the Tabu search-based solution technique. The recency-based memory is used because it is a short term memory function in application. When it is compared with the other methods, this short term memory provides efficient routing solution by using less memory size for optimum computation. It is also applicable to small size routing problems such as the FAC application described in this thesis.

Recency-based memory

Recency-based memory records solution attributes (or they are called elements or components) that have changed recently. Selected attributes in recently visited solutions become tabu-active during the local search. In other words, the solution containing these attributes are classified as tabu. Before the tabu search is
described, the following notations are introduced first:

- **x** the current solution, x represents the total distance travelled of pallets to the assigned robots
- **x_{best}**, the best-known solution of the total distance travel among the tabu search iterations
- **N(x)**, the neighborhood of x, which is another solution for the total distance travel by changing one of the routes for the pallets.
- **N^*(x)**, the “admissible” subset of N(x) which it is not assumed to be the restricted route or they are called “tabu tenure”. (i.e., non-tabu).

In tabu search, the elements in **N^*(x)** are excluded from the feasible solution region of the pallet allocation problem. Solutions are not permitted to revisit the prescribed numbers for each element in the “tabu tenure”.

The general framework of the tabu search heuristic for the pallet allocation problem is outlined in the following:

Step 1: Find an initial solution **x_0** ∈ X, Set **x_{now} = x_{best} = x_0**, where initial solution of the distance travel for pallet is **x_0**. **x_{now}** is the current value of total distance travel for the pallet and **x_{best}** is the best known solution of total distance travel of the pallet. Where they are equal at the initial state of the Tabu Search,

- Assign both pallets 1 and 3 for processing in robot A and robot B respectively
- Assign the remaining tokens of robot A and robot B for processing pallets 2 and 4.
Step 2: Intensification phase

- If termination condition (e.g. simple iteration count, no admissible improving move, no change in $x^{\text{best}}$ in iterations) is satisfied then go to Step 3
- Choose $x^{\text{next}} \in N(x^{\text{now}})$ such that $x^{\text{best}}$ is not tabu or satisfies aspiration criterion
- Move from $x^{\text{now}}$ to $x^{\text{best}}$, i.e. set $x^{\text{now}} = x^{\text{next}}$
- If $x^{\text{now}}$ is better than $x^{\text{best}}$, then set $x^{\text{best}} = x^{\text{now}}$
- Update recency-based memory. Repeat Step 1.

Step 3: Diversification phase

- If termination condition is satisfied, then stop, go back to step 2 when it is not satisfied.

6.2.2. Practical application of algorithm application in FAC control systems

6.2.2.1. Tabu search application in example 1

The configuration of the FAC example 1 has been described in chapter 5. The conveyor system in this example is composed of four conveyors with a single loop and two assembly robots (Figure 3-1). Figure 6-1 depicts the material flow path in the FAC. The default moving direction for each conveyor is shown by arrows in the figure. The four conveyors are marked as A, B, C and D respectively. The loading/unloading area for pallets locates at conveyor A. After a pallet is loaded into the FAC, it is conveyed according to the arrow direction. Every pallet is assigned specific assembly processes, hence, it is conveyed to the designated robot for assembly operation. Once the part loaded in the pallet has been assembled by the robot, the pallet moves back to conveyor A for unloading. When the robots are
fully occupied with pallets, the other pallets are directed to move around the conveyor loop until one of the robots is free.

![Diagram of Pallet Transportation Direction for FAC Configuration 1](image)

**Figure 6-1** Pallet transportation direction for the FAC configuration 1.

The scheduling agent of the FAC control system is responsible for pallet allocation to assembly robots. In chapter 5, the scheduling agent does not possess an allocation strategy in example 1. Pallets are allocated to the robots simply by the first-come-first-serve approach, that is, pallets are assumed to distribute to the robots directly from the conveyor control agent. But in this example with the introduction of the Tabu search-based algorithm, the places for the conveyors are defined. It is because the scheduling agent will search for the best route for each pallet and this route requires detail definition of the conveyors. In the CPN model, therefore, the 4 conveyors are now represented by 4 places.

The scheduling agent applies the Tabu search algorithm and it then attempts to arrange the pallet schedules in a more efficient way. If function \( N(x) \) represents the pallet travel distance, the algorithm will search for a convergent solution for the distance travel. For illustration purpose in this example, only one product type is assigned to the two robots for simplicity. The capabilities and performance of the two robots are also assumed to be identical. Whenever a robot becomes unoccupied,
a pallet will be assigned to the robot for assembly operation. The scheduling agent will also search an optimum path for assigning pallets to the assembly robots. Rules for Tabu search are as follows:

Path selection: (Figure 6-1 shows how the paths are arranged in the FMS control system)

Path 1: Loading -> A -> B -> Assembly process at robot 1
Path 2: Loading -> A -> B -> C -> D-> Assembly process at robot 2
Path 3: A -> B -> C -> D -> A (Cyclically)

Condition:

Option 1: The closest pallet to robot 1 will be selected for processing in this robot
Option 2: The closest pallet to robot 2 will be selected for processing in this robot

Path 1 and Path 2 are the pallet conveying paths to the assembly robots 1 and 2 respectively. The scheduling agent will assign assembly tasks to the pallets according to the following rules:

<table>
<thead>
<tr>
<th>Condition</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1</td>
<td>Assign Path 1</td>
</tr>
<tr>
<td>Option 2</td>
<td>Assign Path 2</td>
</tr>
<tr>
<td>Else</td>
<td>Assign Path 3</td>
</tr>
</tbody>
</table>

Figure 6-2 Simplified Tabu Search rules to the FAC example 1

Referring to the configuration 1 constraints, a maximum of six pallets can coexist in the FAC. At any time, four of the six loaded pallets are assigned with assembly tasks to the two robots. Each robot is assigned two pallets for assembly operations. That is, one of the pallets is waiting when the other one is being processed at the
robot. The remaining two pallets without any task assignment are kept circling around the FAC according to path 3, until they are assigned assembly operations when the robots become available.

To establish the CPN model for system simulation and performance analysis, activities of the FAC system tasks are indicated by token movements. As in example 1, there are 6 colors for tokens representing the six pallets in the FAC. They are named as 1, 2, 3, 4, 5 and 6 respectively. For the robot assembly tasks, two other color tokens RobA and RobB are used to represent the resources available at robots 1 and 2 respectively. Each of the robot is assumed to have two assignments concurrently, so there are two RobA tokens and two RobB tokens in the CPN model. At the start of the simulation, the Tabu search-based algorithm has an initialization phase. Figure 6-3 shows the initialization phase which contains two essential steps.
Iteration 1:

<table>
<thead>
<tr>
<th>Pallet 1</th>
<th>Pallet 2</th>
<th>Pallet 3</th>
<th>Pallet 4</th>
<th>Pallet 5</th>
<th>Pallet 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 2</td>
<td>Path 2</td>
<td>Path 1</td>
<td>Path 1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Where $p_{Robs} \in t_{Conveyor1}^o - a t_{Conveyor1}$ is governed by the condition

The first loaded pallet is assigned a robot assembly token of robot 2, then the second pallet is also assigned the last robot 2 token. This arrangement minimizes the total distance travel of the loaded pallets in the FAC. In the beginning of the FAC control stage, the pallets are loaded one by one and the first two pallets move to the farthest side from the loading/unloading area. Each conveyor permits a maximum of two pallets in operation. In this stage, the first loaded pallet moves to robot 2 for assembly process and the second pallet awaits near robot 2. At the same time, the third loaded pallet moves to robot 1 for assembly process and the fourth loaded pallet awaits near robot 1.

Iteration 2:

<table>
<thead>
<tr>
<th>Pallet 1</th>
<th>Pallet 2</th>
<th>Pallet 3</th>
<th>Pallet 4</th>
<th>Pallet 5</th>
<th>Pallet 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Path 2</td>
<td>-</td>
<td>Path 1</td>
<td>Path 1 / 2</td>
<td>Path 1 / 2</td>
</tr>
</tbody>
</table>

Iteration 2 shows the status after the first and third pallets have finished their robot assembly tasks. Afterwards, the two pallets move to the load/unload station for unloading. Subsequently, the second and fourth pallets move to robots 2 and 1 respectively. The shaded area of the table shows the fifth and sixth loaded pallets that they circle around the conveyors. The labels “Path 1/2” are the possible ways for these two pallets according to the rule in figure 6.2 after a token of robots 1 or 2 is released.

Referring to the CPN model in figure 6-4, if either pallet 1 or pallet 2 has completed its process in robot 1 or robot 2, the assigned token will be released back to the place “Robs”. Figure 6-4a is the overall CPN modeling view for example 1 with Tabu search application and figure 6-4b shows the place and transition markings to figure 6-4a. This token will then be assigned to pallet 5 or pallet 6. In this example, the Tabu search application is simplified to 3 paths and 2 conditions as
shown in figure 6-2. Control statements in figure 6-2 comprise the main heuristic logic for the Tabu search application in this example. These statements are implemented into the CPN model using the approach for converting the control statement to CPN model (described in chapter 5).
Figure 6-4a. The CPN models for FAC example 1 with Tabu Search Application.
<table>
<thead>
<tr>
<th>P1</th>
<th>OrderInf</th>
<th>T1</th>
<th>EnterOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>OrderHandlingAgentInf</td>
<td>T2</td>
<td>ProcessOrder</td>
</tr>
<tr>
<td>P3</td>
<td>SchedulingAgent</td>
<td>T3</td>
<td>PalletIdentify</td>
</tr>
<tr>
<td>P4</td>
<td>PalletIdentityAgent</td>
<td>T4</td>
<td>Pallettransfer</td>
</tr>
<tr>
<td>P5</td>
<td>BarcodeAgents</td>
<td>T5</td>
<td>ProductionProcess</td>
</tr>
<tr>
<td>P6</td>
<td>ConveyorControlAgentf</td>
<td>T6</td>
<td>LoadingUnloadingProcess</td>
</tr>
<tr>
<td>P7</td>
<td>RobotAgentf</td>
<td>T7</td>
<td>Conveyor1</td>
</tr>
<tr>
<td>P8</td>
<td>ProductOutf</td>
<td>T8</td>
<td>Conveyor2</td>
</tr>
<tr>
<td>P9</td>
<td>aCount</td>
<td>T9</td>
<td>Separate</td>
</tr>
<tr>
<td>P10</td>
<td>A</td>
<td>T10</td>
<td>adeduct</td>
</tr>
<tr>
<td>P11</td>
<td>B</td>
<td>T11</td>
<td>R1</td>
</tr>
<tr>
<td>P12</td>
<td>RA</td>
<td>T12</td>
<td>Cdeduct</td>
</tr>
<tr>
<td>P13</td>
<td>acollect</td>
<td>T13</td>
<td>bdeduct</td>
</tr>
<tr>
<td>P14</td>
<td>BRA</td>
<td>T14</td>
<td>R2</td>
</tr>
<tr>
<td>P15</td>
<td>CC</td>
<td>T15</td>
<td>Conveyor4</td>
</tr>
<tr>
<td>P16</td>
<td>bcollect</td>
<td>T16</td>
<td>Conveyor3</td>
</tr>
<tr>
<td>P17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P18</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P19</td>
<td>RB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P20</td>
<td>ccollect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P21</td>
<td>BRB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P22</td>
<td>Robs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P23</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-4b. Place and Transition references to figure 6-4a.

6.2.2.2. Tabu search application in example 2

The configuration of FAC example 2 has been described in chapter 5, a schematic view of the FAC configuration was illustrated in figure 3-5 at chapter 3. In this example, the Tabu search-based algorithm is implemented in the scheduling agent. There are totally three assembly robots and detailed descriptions of the configuration and its constraints were introduced in chapter 3.
In the CPN model in chapter 5, the scheduling agent in example 1 simply allows the pallet tokens to flow to the place of robot. The example ignores the configuration of the conveyors and only one place for the conveyor control agent is used for the representation of the conveyors. Pallets are dispatched to robots according to the first-come-first-serve rule. In example 2 in this chapter, places of conveyors are defined individually. Together with the application of the Tabu search based algorithm, the pallet tokens are distributed by the algorithm to individual conveyor places explicitly.

![Diagram](image)

Figure 6-5 Pallet flow direction for the conveyors in the FAC example 2

![Diagram](image)

Figure 6-6 Schematic diagram for the conveyors flowing direction of the FAC example 2
The above two figures (figure 6-5 and 6-6) depict the pallet flow direction of the conveyors in the FAC. This FAC comprises a loop of conveyors (A, B, C and D) with two branching conveyors E and F. Pallets in this system are loaded and unloaded at one end of conveyor F. Furthermore, conveyor E is a buffer for this FAC material transfer system. It is a place for the pallets to stay temporarily when the other conveyors are fully occupied. The pallets will move back to conveyor F to unload after completing the assembly processes.

The Tabu search-based algorithm of the pallet allocation process is applied to the scheduling agent for this FAC control system. The aim of this application is to minimize the travel distance of the pallets. The function \( f(x) \) represents the pallet traveling distance. The following paragraph shows the path selections and the options for the conveyors.

Path selections
Path 1: Loading at F -> C -> Assembly process at the robot 3 -> D -> A -> B -> unloading at F
Path 2: Loading at F -> C -> D -> A -> Assembly process at the robot 1 -> B -> unloading at F
Path 3: Loading at F -> C -> D -> A -> B -> Assembly process at the robot 2 -> unloading at F
Path 4: Loading at F -> C -> E -> C -> Assembly process at the robot 3 -> D -> A -> B -> unloading at F
Path 5: Loading at F -> C -> E -> C -> D -> A -> Assembly process at the robot 1 -> B -> unloading at F
Path 6: Loading at F -> C -> E -> C -> D -> A -> B -> Assembly process at the robot 2 -> unloading at F

Options:

Option 1: The closest pallet to the robot 1 will assign process to this robot

Option 2: The closest pallet to the robot 2 will assign process to this robot

Option 3: The closest pallet to the robot 3 will assign process to this robot

In summary, there are 6 path selections and 3 options. Among the 6 choices, three of the paths lead to the robots directly without traveling to the buffer conveyor E. The remaining paths are routes to the robots via buffer E. Referring to chapter 3, a maximum of up to ten pallets can coexist in this FAC and the robots perform the identical task in order to simplify the example. In example 3, the robots are represented by individual color tokens and each of the robot can distribute two tokens. At the initial stage of the Tabu search-based algorithm, the earliest loaded pallets are conveyed to the robot at the farthest side. And they move to each of the three robots in groups of two. The reason is that while one pallet is being processed another one has to wait beside the robot.
Iteration 1:

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path3</td>
<td>Path3</td>
<td>Path2</td>
<td>Path2</td>
<td>Path2</td>
<td>Path1</td>
<td>Path1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

P1, P2……P10 represent pallet number 1, 2,…, 10 etc.

Where \( p_{Robs} \in t_{Conveyor} \) \( ^o \ - \ o_{t_{Conveyor}} \) is governed by the condition

The maximum allowable number of pallets to coexist in a conveyor is 2. The first two loaded pallets are assigned a robot assembly token of the robot 3, then the third and forth pallets are assigned a robot 2 token. This arrangement minimizes the total distance travel of the loaded pallets in the FAC. In the beginning of the FAC control stage, the pallets are loaded one by one and the first two pallets is conveyed to the farthest side from the loading/unloading area. Each conveyor permits a maximum of two pallets in operation. To minimize the distance travel of the fourth pallet, the fifth and sixth pallets are assigned robot 1 token and the remaining pallets are assigned to path 4-6.

Iteration 2:

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Path3</td>
<td>-</td>
<td>Path2</td>
<td>-</td>
<td>Path1</td>
<td>Path3</td>
<td>Path2</td>
<td>Path1</td>
<td></td>
</tr>
</tbody>
</table>

The products in the first pallet, third pallet and fifth pallet are unloaded at conveyor F after they have finished the assembly processes. Iteration of the Tabu search-based algorithm will complete if no more order is received by the order-handling agent. Tokens inside the places represent resources available for pallets in the particular conveyor.

Figure 6-7. Iterations for the Tabu search for example 2

A total of six pallets are assigned with the assembly tasks. For this reason, the six robot tokens are used up. Thus, there are four loaded pallets are circling around the FAC without robot assembly assignments. No more pallet is loaded into the cell until one of the loaded pallet is unloaded at conveyor F. Figure 6-8 depicts the control statement for pallet routing options in this example.
If option 1 then path 1
    elseif option 2 then path 2
        elseif option 3 then path 3
            elseif path 1 is full then path 4
                elseif path 2 is full then path 5
                    elseif path 3 is full then path 6
                        endif
                endif
            endif
        endif
    endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endif
endi
Figure 6-9a. CPN models for example 2 in Tabu Search application
<table>
<thead>
<tr>
<th>P1</th>
<th>OrderInf</th>
<th>T1</th>
<th>EnterOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>OrderHandlingAgentf</td>
<td>T2</td>
<td>ProcessOrder</td>
</tr>
<tr>
<td>P3</td>
<td>SchedulingAgentf</td>
<td>T3</td>
<td>PalletIdentify</td>
</tr>
<tr>
<td>P4</td>
<td>PalletIdentityAgentf</td>
<td>T4</td>
<td>Pallettransfer</td>
</tr>
<tr>
<td>P5</td>
<td>BarcodeAgentf</td>
<td>T5</td>
<td>ProductionProcess</td>
</tr>
<tr>
<td>P6</td>
<td>ConveyorControlAgentf</td>
<td>T6</td>
<td>LoadingUnloadingProcess</td>
</tr>
<tr>
<td>P7</td>
<td>RobotAgentf</td>
<td>T7</td>
<td>ConveyBuf</td>
</tr>
<tr>
<td>P8</td>
<td>ProductOutf</td>
<td>T8</td>
<td>R1</td>
</tr>
<tr>
<td>P9</td>
<td>A</td>
<td>T9</td>
<td>adeduct</td>
</tr>
<tr>
<td>P10</td>
<td>Buffer</td>
<td>T10</td>
<td>Conveyor1</td>
</tr>
<tr>
<td>P11</td>
<td>B</td>
<td>T11</td>
<td>Conveyor2</td>
</tr>
<tr>
<td>P12</td>
<td>RA</td>
<td>T12</td>
<td>R2</td>
</tr>
<tr>
<td>P13</td>
<td>BRA</td>
<td>T13</td>
<td>bdeduct</td>
</tr>
<tr>
<td>P14</td>
<td>aCount</td>
<td>T14</td>
<td>Conveyor3</td>
</tr>
<tr>
<td>P15</td>
<td>aCollect</td>
<td>T15</td>
<td>Conveyor4</td>
</tr>
<tr>
<td>P16</td>
<td>C</td>
<td>T16</td>
<td>R3</td>
</tr>
<tr>
<td>P17</td>
<td>D</td>
<td>T17</td>
<td>Separate</td>
</tr>
<tr>
<td>P18</td>
<td>RB</td>
<td>T18</td>
<td>cdeduct</td>
</tr>
<tr>
<td>P19</td>
<td>E</td>
<td>T19</td>
<td>ddeduct</td>
</tr>
<tr>
<td>P20</td>
<td>BRB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P21</td>
<td>RC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P22</td>
<td>BRC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P23</td>
<td>Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P24</td>
<td>CRC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P25</td>
<td>ccollect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P26</td>
<td>dcollect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P27</td>
<td>Robs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P28</td>
<td>RA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-9b. Place and Transition references to figure 6-9a
The CPN model for this example is presented in figure 6-9. Figure 6-9a is the overall CPN modeling view for the example 2 with Tabu search application. Although the model seems complicated but it is designed according to the generic FAC control system. Reusing the control statement introduced in chapter 5, the tabu search-based scheduling rule in figure 6-8 is implemented by the CPN models.

6.2.2.3. Tabu search application in example 3

Figure 3-9 shows the configuration of FAC example 3. Similar to previous examples, the Tabu search algorithm is implemented to the scheduling agent. In this FAC example, there are totally twelve machines. A robot is used to load and unload pallets. This robot is responsible for transportation. Detail descriptions of the configuration and its constraints were introduced in chapter 3.

In FAC example 3, the system is comparatively complicated when compared with examples 1 and 2. This FAC example contains twelve machines. In examples 1 and 2, the FACs are used to produce one product but FAC example 3 has to produce four products. Therefore, the complexity of the control strategy for the scheduling agent in the control system has to be increased.

Referring to figure 3-7, the four product types are manufactured by three machines in the cell. Each of the four products is required to manufacture by three machining processes respectively. The layout for the twelve machining centers and their characteristic in machining orientation are introduced in chapter 3. These machines are layout in four conveyors and each of these conveyors contains 3 machines. Pallets are first loaded to these machines conveyors, they have to be milled by one of
the four milling machines. The remaining two machining processes are drilling and grinding processes. These parts are drilled and then they are transferred to grinding machines for grinding processes. The allocations of machines to the drilling and grinding processes are relied to product requirement. The constraints for the allocation are due to the machining orientation to the products.

The lists of the machining centers to the product types are shown in figure 3-18. This figure summaries the allocation of the machines to the product types. The layouts of the FAC examples are suitable for mixed production of different part types. Common machines such as milling machines can be shared by different part types in this layout.

Referring to figure 3-9, the pallets in this system are loaded and unloaded by a robot. This robot is also responsible for the unloading process to transfer pallets away from the cell. Tabu search-based algorithm is also applied to the scheduling agent. According to the examples 1 and 2, the Tabu-search application is used to allocated pallets to the closest idle robots for assembly process. In the example 3, the FAC allocates pallet to machines for machining processes.

In this example, the conveyor system layout is similar to that of FAC example 1, the conveyors are layout as a loop. The pallets are conveyed around the conveyor loop until the pallets finish their required three processes. Different from example 1, this FAC example can produce four product types. The scheduling agent will search an optimum path for assigning pallets to the machines.
Rules for Tabu search are as follows:

Path selection (referring to figure 3-19)

1. For milling processes of the four product types (Any of the four milling machines)
2. For drilling processes of the four product types (Selection of drilling process depends on process orientation)
3. For grinding processes of the four product types (Selection of grinding process depends on process orientation)

Condition 1: The closest pallet to the nearest milling machine.
Condition 2: The closest pallet to the nearest drilling machine.
Condition 3: The closest pallet to the nearest grinding machine.

Paths 1 to 3 are the pallet conveying paths in ascending order to the types of machines. The scheduling agent will assign machines to the pallets according to the following rules:
A simple rule is applied to FAC example 3, the application is trying to eliminate the randomness effect to the FAC components. The scheduling agent will assign pallets to machines by a rule.

Referring to the configuration of the FAC example 3, a maximum of twelve pallets can coexist in the FAC. At any time, the number of pallets in the cell is equal to the total number of machines in the cell. Each machine is assigned one pallet for machining operations. And each of the pallets is fired with three other tokens, they are the token for milling machine, drilling center (vertical or horizontal oriented) and grinding machine (vertical or horizontal oriented). At any time, three pallets are assigned to manufacture each product types, that is, four product types will have twelve assignments.

To establish the CPN model for system simulation and performance analysis, activities of the FAC system tasks are indicated by token movements. As in
example 3, there are twelve colors for tokens representing the six pallets in the FAC. They are named as 1, 2, 3, 4, 5 to 12 respectively. For the product type tasks, three other color tokens M (Milling), D (Drilling) and G (Grinding) are used to represent the resources available at the machines respectively. At the start of the simulation, the Tabu search-based algorithm has an initialization phase. Figure 6-11 shows the initialization phase which contains two essential steps.
Iteration 1:

<table>
<thead>
<tr>
<th>Pallet 1</th>
<th>Pallet 2</th>
<th>Pallet 3</th>
<th>Pallet 4</th>
<th>…..</th>
<th>Pallet 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 1 (M4)</td>
<td>Path 1 (M3)</td>
<td>Path 1 (M2)</td>
<td>Path 1 (M1)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Where $p_{Robs} \in t_{Conveyor} - t_{Conveyor}$ is governed by the condition

The first four loaded pallets are assigned milling m/c token. This arrangement must be allocated to the pallets according to the machining process priorities. The arrangements of tabu search application minimize the total distance travel of the loaded pallets in the FAC. In the beginning of the FAC control stage, the pallets are loaded one by one and the first pallet moves to the farthest side from the loading/unloading area. Each conveyor permits a maximum of four pallets in operation. In this stage, the first loaded pallet moves to milling machining center (M4) for milling process and the second pallet is conveyed to milling machining center (M3). At the same time, the third loaded pallet moves to M2 and the fourth loaded pallet is conveyed to M1.

Iteration 2:

<table>
<thead>
<tr>
<th>Pallet 1</th>
<th>Pallet 2</th>
<th>Pallet 3</th>
<th>Pallet 4</th>
<th>Pallet 5</th>
<th>Pallet 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 2 (D4)</td>
<td>Path 2 (D3)</td>
<td>Path 2 (D2)</td>
<td>Path 2 (D1)</td>
<td>Path 1 (M4)</td>
<td>Path 1 (M3)</td>
</tr>
</tbody>
</table>

Iteration 2 shows the status after the first two pallets have finished their machining tasks. Afterwards, the two pallets move to the load/unload station for unloading. The shaded area of the table shows the fifth and sixth loaded pallets that they circle around the conveyors. The labels “Path 1/2” are the possible ways for these two pallets according to the rule in figure 6-10 after a token of M4 or M3 is released.

Figure 6-11. Iterations for the Tabu Search

Referring to the CPN model in figure 6-12, if either pallet 1 or pallet 2 has completed its process in M4 or M3, the assigned token will be released back to the place of milling machines. Figure 6-12a is the overall CPN modeling view for example 3 with Tabu search application and figure 6-12b shows the place and transition markings to the figure 6-12a. This token will then be assigned to pallet 5.
or pallet 6. In this example, the Tabu search application is simplified to 3 paths and 2 conditions as shown in figure 6-2. Control statements in figure 6-10 comprise the main heuristic logic for the Tabu search application in this example. These statements are implemented into the CPN model using the approach for converting the control statement to CPN model (described in chapter 5)

To summarize, there are 3 path selections for each of the four product types. Among the 3 choices, the pairs of machines have the least cumulative machining time is chosen to allocate to the incoming pallet. Referring to chapter 3, a maximum of up to twelve pallets can coexist in this FAC and the robots perform the transportational task for loading and unloading tasks to the FAC conveyors.
Figure 6-12a CPN models for example 3 in Rule-based application
<table>
<thead>
<tr>
<th>P1</th>
<th>OrderInf</th>
<th>T1</th>
<th>EnterOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>OrderHandlingAgentf</td>
<td>T2</td>
<td>ProcessOrder</td>
</tr>
<tr>
<td>P3</td>
<td>SchedulingAgentf</td>
<td>T3</td>
<td>PalletIdentify</td>
</tr>
<tr>
<td>P4</td>
<td>PalletIdentityAgentf</td>
<td>T4</td>
<td>Pallettransfer</td>
</tr>
<tr>
<td>P5</td>
<td>BarcodeAgentf</td>
<td>T5</td>
<td>ProductionProcess</td>
</tr>
<tr>
<td>P6</td>
<td>ConveyorControlAgentf</td>
<td>T6</td>
<td>LoadingUnloadingProcess</td>
</tr>
<tr>
<td>P7</td>
<td>RobotAgentf</td>
<td>T7</td>
<td>Product1</td>
</tr>
<tr>
<td>P8</td>
<td>ProductOutf</td>
<td>T8</td>
<td>Product2</td>
</tr>
<tr>
<td>P9</td>
<td>Raw Material Incoming</td>
<td>T9</td>
<td>Product3</td>
</tr>
<tr>
<td>P10</td>
<td>Grinding(Horizontal)</td>
<td>T10</td>
<td>Product4</td>
</tr>
<tr>
<td>P11</td>
<td>Loading</td>
<td>T11</td>
<td>MFG1</td>
</tr>
<tr>
<td>P12</td>
<td>Unloading</td>
<td>T12</td>
<td>MFG2</td>
</tr>
<tr>
<td>P13</td>
<td>Conveyor1</td>
<td>T13</td>
<td>MFG3</td>
</tr>
<tr>
<td>P14</td>
<td>Conveyor2</td>
<td>T14</td>
<td>MFG4</td>
</tr>
<tr>
<td>P15</td>
<td>Conveyor3</td>
<td>T15</td>
<td>UnloadConveyor1</td>
</tr>
<tr>
<td>P16</td>
<td>Conveyor4</td>
<td>T16</td>
<td>UnloadConveyor2</td>
</tr>
<tr>
<td>P17</td>
<td>Milling</td>
<td>T17</td>
<td>UnloadConveyor3</td>
</tr>
<tr>
<td>P18</td>
<td>Drilling(Veritical)</td>
<td>T18</td>
<td>UnloadConveyor4</td>
</tr>
<tr>
<td>P19</td>
<td>Robot1</td>
<td>T19</td>
<td>Unloadingstation</td>
</tr>
<tr>
<td>P20</td>
<td>Drilling(Horizontal)</td>
<td>T20</td>
<td>Backtoloading</td>
</tr>
<tr>
<td>P21</td>
<td>Grinding(Vertical)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P22</td>
<td>ProductOut</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6-12b.  Place and Transition references to figure 6-12a
6.2.2.4. Pseudo code for the algorithm for the examples.

Except the CPN modeling assignment as shown in the figures 6-4, 6-9 and 6-12, a program script will be used to control the flow of the token in the scheduling agent of the CPN model. This programming script is used as the subroutine to control the tokens flow in the CPN model. This subroutine will be called by a method in order to make decision for the token flow. Figure 6-13 is the pseudo code for the Tabu search-based application in examples.

```
Procedure tabusearch
begin
    select a current point, currentNode, at random
    update NodesGenerated
    bestNode <- currentNode
    print
    repeat
        select a new node, newNode, that has the shortest distance in the neighborhood of currentNode that is not on the tabuList, using the twoInterchange method
        update NodesGenerated for all nodes checked in neighborhood of currentNode
        currentNode <- newNode
        if evaluation(currentNode) < evaluation(bestNode)
            bestNode <- currentNode
            print
        until some counter reaches limit
end
```

Figure 6-13 Pseudo Code for the Tabu Search Algorithm in ML
6.3. Concluding remarks

Three CPN models are created in this chapter for the algorithm application for the three FAC examples. The CPN models are modified from the FAC example 1 template as shown in figure 5-68 in chapter 5. Figures 6-4, 6-9 and 6-12 are the new CPN models for the algorithm application. Places (P1-P8), transition (T1-T6) and arc connect between these CPN entities are the components inherited from the generic FAC template. The newly created components are used to implement the algorithm application.

The static model created in chapter 5 for these CPN components are reused by these algorithm examples. The pre-defined CPN and UML components greatly save time and effort to implement different FAC configurations and algorithm application. The new CPN components upon the generic template are created according to the pseudo codes (figure 6-13). The static model and dynamic model created according to the generic FAC example 1 are remained unchanged to the implementation of the CPN models to the FAC configuration changes and algorithm application. Additional components are added to the generic FAC template to fulfill the required changes. Therefore, the capability and reusability of the template using CPN and UML model are obtained.

This chapter discusses the construction of CPN models with the incorporation of the algorithm application for pallet allocation. They are Tabu search-based and rule based application. As in chapter 5, this chapter presents the three new CPN models generated for FAC examples 1, 2 and 3 respectively. Effects on configuration changes of the FAC in generating the CPN models have been presented in chapter 5.
and the effects on design changes of the agents of the FAC are presented in this chapter. These two chapters provide the reference on dynamic CPN model construction with analysis capability. Altogether the models have been generated and these models are designed based on the generic FAC control framework template. The CPN models have to be evaluated and analyzed and the next chapter will present the analysis procedures for the CPN models. The generic UML model for the FAC will be modified according to the analysis result presented in next chapter.
CHAPTER 7

SYSTEM ANALYSIS AND IMPLEMENTATION

In the last two chapters, CPN models have been established for the FAC control system configuration examples. This chapter presents how the simulation data is analyzed. The CPN models are built and simulated with Design/CPN which is a CPN editing tool with simulation capability. The token activities of each example are recorded and each of these simulations exports a file for analysis. Each simulation file contains a bulk of agent activity data. It is necessary to convert such data into useful information. In this regard, a data mining algorithm is a promising tool for this purpose. This information is useful to evaluate the FAC examples’ performances. That is, the different configurations of the FAC control systems can be evaluated and the results of the simulations are analyzed. This analysis cycle contributes to the design and modeling of the complex agent-based FAC control system framework.

7.1. Introduction to data mining analysis

Data mining is a process to extract implicit, previously unknown and potentially useful information from bulks of data. Generally, data mining is an analyzing process and it summarizes hidden data into useful information. This process is also called knowledge discovery in database (KDD). This technique is now widely used in database applications in commercial environments. Classification rule learning, k-means, clustering analysis, association rule mining and neural networks are commonly applied methodologies in data mining application. A KDD process normally consists of several steps:
1. Understanding and formulating the problem
2. Collecting and Preprocessing the data for analysis
3. Extract or discover knowledge hidden in the data by using data mining algorithm
4. Interpreting and evaluating the knowledge extracted
5. Review the process by using the discovered knowledge

Data mining is widely used in business applications nowadays. For instance, the process is applied to quantitative analysis (Tsoumakas et al, 2004) and bulks of discrepancies data are converted to meaningful information through the analysis (Schallehn et al, 2004). Typically, four types of relationships are sought out by data mining, they are classes, clusters, associations and sequential patterns (Berzal et al, 2001):

- **Classes**: According to the data behavior, data with similar attributes are grouped and located in different predetermined groups.
- **Clusters**: The attributes of the data items are the key elements in this classification. Data items are clustered according to logical relationships.
- **Associations**: Association creates rules that describe how often events occur together. Such relationships are typically expressed in support and confidence level.
- **Sequential patterns**: Data is mined to anticipate behavior patterns and trends.

Among the KDDs, one of the most well-studied tasks is the discovery of association rules (Kouris et al, 2005). This is applicable to various data types such as text document, census data, supermarket transaction records etc, they are collaboratively called market basket data (Feng et al, 2001).
7.1.1. Data mining application in this project

The Apriori algorithm (Agrawal et al, 1994) is one of the most successful association rule mining algorithm. For instance, it examines a long list of transactions in order to determine which set of items is the most frequent occurrence one in a market basket application. The result of the application is the frequency of the occurrences of interesting subsets of items and it can be useful for decision making, statistical analysis and machine learning (Agrawal et al, 1999).

In a data mining application, the efficiency of the algorithm is a crucial factor. As described in section 7.1, four typical types of relationships are often sought using data mining applications. The Apriori algorithm is regarded as one of the efficient and well performed algorithms in frequent pattern mining. This algorithm is applicable in the mining of association rules, sequential patterns, preference correlations from database (Sarasere et al, 1995). It also contributes to applications in document searching (Agrawal et al, 2002). The speed of the document searching is greatly enhanced by the association rules and sequential patterns of the algorithm. Other applications of the algorithm include finding correlation and data pattern analysis of basket data in retail market (Brin et al, 1997).

In manufacturing applications, the Apriori algorithm is applied to process control, engineering data analysis and quality information processing (Zakarian et al, 2003). The objective of the application is to retrieve rules or information hidden inside the basket data. In this research project, the CPN models of the examples described in the last two chapters are used for the simulation experiments. In some of the places in these CPN models, the token activities are marked. Each place is given a legend
for identification from other places. Thus, the simulation results by using these
token records are extracted in a file and the meanings of the data can be revealed by
using the Apriori algorithm.

In this project, the algorithm results are helpful in the modification of the agent
interactions and they provide simulation references to the FAC control system
eamples. Therefore, this data mining application is useful for the CPN simulation
result analysis.

7.1.2. Advantages using the Apriori algorithm in Data mining application in this
project

Using the Apriori algorithm, first of all, the interaction frequencies among the agents
in the FAC control system can be extracted. Secondly, the interactions between
agents related to system performance are studied. Thirdly, the token occurrences in
specific CPN components reveal problems such as deadlock, bottleneck of the
control system etc. Thereafter, interactions among agents are also revealed by the
association rules extracted from the simulation data (Figure 7-1). These are critical
factors for non-deterministic systems such as our FAC control system.
7.1.3. **Interpretation of simulation data**

In this project, data files are the output of CPN simulations. The three examples are used to perform simulation using the Design/CPN and each simulation run exports a file. This simulation files are used for data mining analysis.

Figure 7-2 shows an exported simulation output data file. Each of the items is constituted by a four digits number. Each of these items represents a specific token movement in certain places in the CPN model. For each data collection period, fifty token activities of the items are exported to the file and this data capturing process will be terminated on reaching twenty thousand steps. The data captured is exported according to the priority of token movements. The priority of the token movements is recorded in sequence from left to right and downward from row to row in the files.
In the construction of the generic CPN template in example 1, there are some places marked with flags. Each flag contains programming scripts for the export of token record. When a token passes through the marked place, the flag will identify the token, therefore, a specific 4-digits legend is exported to file. This legend is specifically assigned to place according to both the token types and the agent classification. Figure 7-3 shows the table for the 4-digits numbers in the CPN models. In this project, the legend assignments in all the examples are identical to places. It is because the three examples are designed by the generic FAC framework. Results of the examples are thus compared in accordance with the same reference in the CPN models.
Figure 7-3a. Codes for the tokens which show in the exported simulation file (Part 1)
Data Code Examples 1, 2, 3 Example 2,3 Example 3

6000 – 6011 Pallet “x” is at the place of “Barcode Agent” with “y” status provides from the place of “Conveyor Control Agent”, when

<table>
<thead>
<tr>
<th></th>
<th>Examples 1, 2, 3</th>
<th>Example 2,3</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6000</td>
<td>x = G</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6001</td>
<td>x = H</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6002</td>
<td>x = K</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6003</td>
<td>x = L</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6004</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 6005</td>
<td></td>
</tr>
</tbody>
</table>

7000 – 7011 Pallet “x” is at the place of “Conveyor Control Agent”, when

<table>
<thead>
<tr>
<th></th>
<th>Examples 1, 2, 3</th>
<th>Example 2,3</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7000</td>
<td>x = G</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7001</td>
<td>x = H</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7002</td>
<td>x = K</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7003</td>
<td>x = L</td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7004</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>y = CB</td>
<td>Code = 7005</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7-3b. Codes for the tokens which show in the exported simulation file (Part 2)

In figure 7-3, the token codes are classified according to the first two digits. For instance, codes 20xx relate to token activities in the places of robot agent and the token activities for the conveyor control agent place are exported to file with codes 70xx. The codes are classified into three groups. They are categorized for the three examples according to the captioned headers. The right column of the table belongs to example 3 only.

According to the examples in this project, the maximum allowable pallets coexist in the FAC is six for example 1 and that for example 2 is eight. For example 3, the maximum allowable pallets coexist in the FAC is twelve. The assignments of the pallet tokens are then relied on these maximum allowable pallets in the FAC examples.
Although the robots or machines are assumed to have identical assembly processes in examples 1 and 2, the legends are designed to be traceable for multiple assembly processes under same control system. This is especially important in example 3 because has four product types have to be produced in the control system. This is useful in the cases when the generic FAC template is applied to examples with various processing tasks, traceability of the assembly processes can be identified by the tokens of pallets.

Each pallet in the control system is assigned a unique identity, therefore, each pallet activity in the individual agent is recorded in the simulation file. Besides, the activities of the agents are categorized according to the number, such as the number “40xx” represents records for the scheduling agent. The last two digits xx are further categorized to particular pallet records.

The new CPN model to FAC example 3 has been modeled in chapter 5. Referring to figures 7-3a and figure 7-3b, the same token legends are used for these FAC control examples.

<table>
<thead>
<tr>
<th>7051 – 7058</th>
<th>Tokens related to the newly created machine control agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>7081 – 7084</td>
<td>Tokens related to the four transportation robots</td>
</tr>
<tr>
<td>8008</td>
<td>Product 1</td>
</tr>
<tr>
<td>8009</td>
<td>Product 2</td>
</tr>
<tr>
<td>8010</td>
<td>Product 3</td>
</tr>
<tr>
<td>8011</td>
<td>Product 4</td>
</tr>
</tbody>
</table>

Figure 7-4. Additional code designs for FAC examples 3

The codes in figure 7-4 are newly proposed codes and they are programmed to FAC example 3, these codes refer to the transitions created in chapter 6. The
activities for the new machine control agent in this example and the CPN components are then able to simulate and the data is able to export to simulated file.

7.1.4. **Apriori Algorithm application in this project**

Data mining algorithms can be applied to extract information from data or database. The tools intelligently transform the processed data into useful information. Data mining is now recognized as one of the promising areas of research encompassing database, statistics and machining learning. The Apriori algorithm is one of the most commonly used techniques in data mining applications. The Apriori method was proposed by Agrawal and Srikant (Agrawal et al., 1993; Agrawal et al., 1994), its main concept is that a subset of a frequent itemset must be frequent. It is primarily useful for the description of the behavior captured from database, that is, mining of association rules. Association rules describe the association relationships among the attributes in the set of relevant data. Each association rule contains two measures, the support and the confidence. The support indicates the frequencies of the occurring patterns and the confidence denotes the strength of implication in the rules.

Generally, manufacturing systems are interfaced with computer networks for high level control nowadays and this control level dominates the communication between control devices. Complicated interactions of the system controllers are not easy measured and the system activities in this level cannot be simply predetermined. The complexity of the system interaction does not have a formal evaluation standard,
for this reason, the association rules mining algorithm is applied to reveal and study the system activities from the extracted data.

The Apriori algorithm is used to discover the association rules using the simulation data for the FAC examples. Each file records a set of agent activities which are identified by the legends shown in figure 7-3. Any previously unknown and potentially useful information among the data is extracted using the algorithm.

7.1.5. Association rule and its parameters

An association rule expresses an association relationship among a set of objects. Suppose we have an example rule as follows:

\[ A \rightarrow B, \text{ where } A \text{ and } B \text{ are sets of agent activities in figure 7-3} \]

The meaning of this rule expresses the number of transaction records of simulation data which contain the agent “A” tend to contain the other agent “B”. For example, the conveyor control agent works closely with the robot agent in the generic control system. The conveyor control agent informs the arrival of pallet to the robot agent, the robot agent will then perform the required assembly process to the incoming pallet. In this association rule application, the association rule “70xx -> 20xx” means the number of transaction records of simulation data which contain the conveyor control agent tend to contain the other robot agent.

Generally, each association rule is defined by the support of the set of agent activities that are involved in the rule. That is, the support is a measure of the
number of transaction records in the simulation data file that contain the two items A and B, hence, it indicates the frequencies of occurring patterns. The confidence is a measure of the number of times that the right hand side of agent activity \( y \) appears in the transaction records where the left hand side agent activity \( x \) item appears, it therefore denotes the strength of implication in the rules.

The support of the rule, \( s \), in \( A \rightarrow B \) is expressed as:

\[
\text{Support (} A \cup B \text{),}
\]

Regarding the simple rule \( A \rightarrow B \), the confidence \( C \) is therefore the percentage of transactions that contains both \( A \) and \( B \) as a percentage of all transactions that contain \( A \), that is:

\[
\text{Confidence, } c = \frac{\text{Support (} A \cup B \text{)}}{\text{Support (} A \text{)}}
\]

Support \( s \) and confidence \( c \) are commonly expressed as probabilities and hence indicated by percentages.

### 7.1.6. Apriori Algorithm implementation

By using the Apriori algorithm, given a user specified minimum support and minimum confidence, the problem of mining association rules is to find all the association rules whose support and confidence are larger than the respective thresholds. An itemset may contain one or more agent activities that are involved in an association rule. For this project application, each transaction record for the data
The mining process contains fifty token records as mentioned in section 7.2.1. With the data, agent activities and their association rules are mined out by this algorithm.

In data mining applications, many algorithms require multiple passes over the whole set of data. That is, it is required to examine the same data file completely for each candidate generation, and the number of the candidates increases exponentially according to the increment of items in the itemset in each iteration. For example, the occurrence frequency of the candidate of the itemset in the first iteration of the mining algorithm contains one item only but the number of items in the itemset increases in the successive iterations. The increment of the items in itemset and the repeatedly database seeking time increase the time for the association rule mining exponentially.

The application of the Apriori algorithm is able to achieve satisfactory performance by reducing the size of the candidate sets. The minimum support and confidence are set by user input and the association rules are pruned out with these threshold settings. The memory required for the candidate generation is thus greatly reduced and the computational cost is lowered by the reduction of candidates.

**7.1.7. Apriori Algorithm implementation steps**

The Apriori algorithm is outlined in the pseudo code in figure 7-5. The following steps correspond to the procedure of the Apriori algorithm applied in this project.
1. The Apriori algorithm searches the transaction records in the simulation file for each itemset. Any item that has a support count less than the minimum support count required is removed from the pool of candidate itemsets.

2. Initially, items in the simulation file are members of 1-candidate itemsets. The support count of the 1-candidate itemset is determined. Itemsets with support counts less than the minimum required support count are removed and the remaining 1-candidate itemsets become candidates. These 1-candidate itemsets are merged to create 2-candidate itemset candidates that each itemset comprise two agent activities.

3. The support count of each of these 2-candidate itemsets are determined and the itemsets found with a support count greater than or equal to the minimum support count are kept to create 3-candidate itemsets in the next iteration. The itemsets with lower support counts are removed from the candidate list. Steps 1 and 2 are then repeated for the generation of 4, 5 to x-candidate itemsets until the support counts of all the x-candidate itemsets are lower than the minimum required support count.

4. All the itemsets generated from step 3 are the frequent itemsets found by the Apriori algorithm. All these itemsets are then used for the generation of association rules using the agent activity items inside the x-candidate itemsets. Any combination of the items in the itemsets are used to sort out the rules using the minimum confidence threshold setting.

5. The Apriori algorithm recursively generates all the subsets of each frequent x-candidate itemsets and creates association rules based on subsets with a confidence greater than the minimum confidence.
$L_1 = \{\text{large 1-itemsets}\}$;
for ( $k=2$; $L_{k-1} \neq \emptyset$ ; $k++$ )
do begin
  $C_k = \text{apriori\_gen}(L_{k-1}, \text{minsup})$ ; //New candidates
  for all transactions $t \subseteq D$
do begin
    $C_t = \text{subset}(C_k, t)$ ; //Candidates contained in subset $t$
    for all candidates $c \subseteq C_t$ do
      $c.\text{count}++$
    end
  end
  $L_k = \{c \subseteq C_k | c.\text{count} \geq \text{minsup}\}$
end

Figure 7-5. Pseudo code for Apriori algorithm

Let $P = \{2000, 2001, \ldots, 7005\}$ be the list of items presented in figure 7-3 and $D$ represents the set of records of the items. Each record in $D$ contains 50 token activities which are any items in $P$, such that $D \subseteq P$. Moreover, let $A$ be an itemset of agent activities which contains one or more token activities. $D$ records in a transaction $T$ of each trial of simulation is said to contain $A$ if and only if $A \subseteq D \subseteq T$.

An association rule is an implication of the form $A \rightarrow B$ with support and confidence, where $A \subseteq P, B \subseteq P$, and $A \cap B = \emptyset$. ($P$ is probability)

$\{A\} \rightarrow \{B\}$ (Support, Confidence)

Support $s$ of the rule $\{A\} \rightarrow \{B\}$, where $s$ is define percentages of occurrence of the itemsets $A$ and $B$. The percentage is counted by total numbers of records in $D$ in the agent activity simulation file $T$. Let $n_i$ be number of the record counts of the
items A and B coexisting in the simulation file $T$, $n_2$ be the total number of record counts of D in transition file T. Therefore, support $s$ of the rule is

$$s = P(A \cup B) = \frac{n_1}{n_2}.$$

Thereafter, the confidence denotes the strength of implication in the rule $A \rightarrow B$ which contains both $A$ and $B$ as a percentage of all transactions that contain $A$. So, the probability of the confidence for the rule becomes,

$$c = \frac{P(A \cup B)}{P(A)}.$$

Let $n_3$ be the number of record counts of the items $A$ in the simulation file $T$, the confidence is also expressed as follows:

$$c = P(A \rightarrow B) = \frac{P(A \cup B)}{P(A)} = \frac{n_1}{n_2} \cdot \frac{n_3}{n_3} = \frac{n_1}{n_2}.$$

The two terms, the confidence $c$ and support $s$ are the two important parameters describing the frequencies of occurring patterns and the strength of implication to the association rule respectively.

The number of association rules generated by the Apriori algorithm is determined by the threshold settings of support and confidence. The threshold settings of the support and confidence find only the large itemsets which have support above the predetermined minimum support. In addition, the confidence threshold derives all
rules, based on each large itemset which has more than its predetermined confidence.

There is no general guideline to the threshold settings for the Apriori algorithm. Lower support and confidence setting in the Apriori application normally result in many association rules. On the contrary, high support and confidence generate fewer rules. Computational effort and time are exponentially proportional to the number of mining rules.

In this project, the minimal support and confidence are defaulted to be 30%. Graphs 7-1 to 7-3 are the graphs of support and confidence for the generic template of example 1’s simulation data. The same Apriori algorithm setting is also applicable to the three examples. The number of rules and mining time for the examples are kept steady at particular threshold settings. The support and confidence become level off around 15% and 30% respectively. These figures are also applicable to the simulation. 20,000 steps are performed for each simulation and 50 steps for each agent activities are captured periodically. These settings balance the computation time for data mining for each bulk of the simulations and effort for the Apriori application.
Graph 7-1. Graph for supporting plot threshold against no. of rules with varies confidence thresholds

Graph 7-2. Graph for Supporting Count Plot against time with vary confidence
7.1.8. Association rules expression in this application

For example, the following is an example association rule of a result using the Apriori algorithm. That is,

\{4003, 2003\} -> \{7003\} \quad (35\%, 70\%)

\{4003, 2003\} and \{7003\} are two itemsets in this association rule. The itemsets comprise two and one items respectively. If the last two digits of the three items are ignored, referring to the figure 7-3, the legends of the items become:

\{Scheduling agent, Robot agent\} -> \{Conveyor Control agent\}

The itemset \{Scheduling agent, Robot agent\} is the antecedent and the \{Conveyor control agent\} is the consequent. In this rule, given the occurrence of the antecedent, how often does the consequent occur? It is 35\% which is the support of this rule.
And the conditional probability of the consequent, given the antecedent, is the confidence 70%.

The last two digits are all “03” and these items are identified as pallet C referring to figure 7-3. The pallet is originated from the scheduling agent and then proceeds for the assembly process at the robot agent, thereafter, the pallet is then removed by the “Conveyor control agent”. This process is verified because the agent activities are recorded according to the sequence of occurrence. The probability that the transaction records contain the specified agent activities is 35%, and the conditional probability of the pallet being found in the conveyor control agent, given that the pallet has visited both the scheduling agent and robot agent, is 70%.

Although many rules are mined by the data mining algorithm, most of the rules are neglected. Typically, the interactions of the agents which do not have physical connections are neglected. For instance, the scheduling agent works closely with the conveyor control agent but the interaction between these two agents are not pruned. In contrast, the conveyor control agent and the robot agent have to collaborate to direct the pallet to move to the robot for assembly operations, association rules related to these two agents are extracted for evaluation. In addition to the association rule for the two agents, the association rules among the scheduling agent, the conveyor agent and the robot agent are also valuable. The collaboration of these three agents are revealed by the support and confidence of the association rules. In particular, the association rules are compared among the different FAC examples with configuration or process changes.
7.2. System Analysis to the FAC examples

7.2.1. ANOVA analysis of the examples

Figure 7-5 shows the counts for each agent in the examples. The examples are simulated and analyzed by the categories without algorithm application and with algorithm application. Therefore, six sets of simulated samples are obtained to do data analysis. The analysis is performed by occurrence counts of the agents codes in figures 7-3 and 7-4. The counts are the token occurrence frequencies in the places of agents. For example, the number of counts for the scheduling agent is calculated by the occurrence of “20xx” in the simulation file. Six simulation trials are performed for each example, therefore, each agent possesses six counts.
<table>
<thead>
<tr>
<th>Agent</th>
<th>Example 1</th>
<th>Example 1 (Tabu Search)</th>
<th>Example 2</th>
<th>Example 2 (Tabu Search)</th>
<th>Example 3</th>
<th>Example 3 (Tabu Search)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Agent</td>
<td>14012 14000</td>
<td>13890 13880</td>
<td>7026 7030</td>
<td>6965 6975</td>
<td>9098 9076</td>
<td>7500 7534</td>
</tr>
<tr>
<td></td>
<td>14016 14021</td>
<td>13875 13891</td>
<td>7031 7020</td>
<td>6966 6970</td>
<td>9065 9083</td>
<td>7564 7532</td>
</tr>
<tr>
<td></td>
<td>14008 14011</td>
<td>13893 13873</td>
<td>7028 7033</td>
<td>6973 6977</td>
<td>9056 9085</td>
<td>7598 7519</td>
</tr>
<tr>
<td>Pallet Identity Agent</td>
<td>14286 14288</td>
<td>13874 13877</td>
<td>7060 7055</td>
<td>6975 6978</td>
<td>8095 8043</td>
<td>8099 8056</td>
</tr>
<tr>
<td></td>
<td>14290 14291</td>
<td>13880 13880</td>
<td>7061 7077</td>
<td>6977 6970</td>
<td>8033 8045</td>
<td>8055 8066</td>
</tr>
<tr>
<td></td>
<td>14274 14277</td>
<td>13870 13874</td>
<td>7064 7071</td>
<td>6980 6981</td>
<td>8032 8054</td>
<td>8067 8067</td>
</tr>
<tr>
<td>Barcode Agent &amp; Conveyor</td>
<td>13996 13991</td>
<td>27901 27910</td>
<td>7042 7048</td>
<td>14963 14960</td>
<td>10593 10544</td>
<td>13990 13955</td>
</tr>
<tr>
<td>Control Agent</td>
<td>14009 14005</td>
<td>27905 27895</td>
<td>7035 7041</td>
<td>14958 14955</td>
<td>10565 10566</td>
<td>13943 13955</td>
</tr>
<tr>
<td></td>
<td>13981 14010</td>
<td>27913 27912</td>
<td>7038 7045</td>
<td>14962 14967</td>
<td>10587 10588</td>
<td>13565 13999</td>
</tr>
<tr>
<td>Robot Agent</td>
<td>14366 14370</td>
<td>15003 15012</td>
<td>7044 7058</td>
<td>7960 7955</td>
<td>6434 6411</td>
<td>7043 7022</td>
</tr>
<tr>
<td></td>
<td>14368 14380</td>
<td>15008 15004</td>
<td>7057 7053</td>
<td>7962 7958</td>
<td>6487 6432</td>
<td>7055 7098</td>
</tr>
<tr>
<td></td>
<td>14390 14400</td>
<td>15017 15005</td>
<td>7045 7063</td>
<td>7967 7955</td>
<td>6432 6435</td>
<td>7032 7054</td>
</tr>
<tr>
<td>Machine control agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6043 6044</td>
<td>9012 9085</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6066 6075</td>
<td>9055 9044</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6023 6055</td>
<td>9032 9033</td>
</tr>
</tbody>
</table>

Figure 7-5 Data extracted from the CPN simulation by Data Mining Algorithm

The simulation data is collected according to the agent types as shown in figure 7-5. In this project, the examples 2 and 3 are built upon the generic FAC control framework of example 1. Most of the generic components are reused in the examples. Most of the interactions between agents remain unchanged and modifications of the agent components are confined to the agent objects. However, the behaviors of the example systems should be examined, especially for the behavior of agents in the control systems. Although some agents remain unchanged across examples in this project, their behaviors might become different in the distributed control structures.
7.2.1.1. ANOVA tests (Significant tests of the algorithm application to the FAC examples)

In general, the purpose of ANOVA (Analysis of Variance techniques) is to test the data to find out if there are significant differences between the means (for groups or variables) for statistical significance. This is accomplished by analyzing the variance, that is, partitioning the total variance into the component that is due to random error and the components that are due to difference between the means. These latter variance components are then tested for statistical significance, and if significant, reject the null hypothesis of no differences between means, and accept the alternative hypothesis that the means are different from the population.

Using the test, the token counts at the referred agents are examined and effects of the configurations and algorithm application are evaluated. The two-ways ANOVA is used for the analysis, the null hypothesis is as follows:

\[ H_0 : \mu_{\text{Config1}} = \mu_{\text{Config2}} = \mu_{\text{Tabu}} = \mu_{\text{NoTabu}} \] (Null Hypothesis)

In this project, the three FAC control system examples are used for the simulation. The null hypothesis \( H_0 \) is that of no differences among the means of token counts for these examples. The six means correspond to the examples of example 1 without algorithm application, example 1 with algorithm application, example 2 without algorithm application, example 2 with algorithm application, example 3 without algorithm application and example 3 with algorithm application. Detailed descriptions of these examples are introduced in chapters 5 and 6 respectively. The means of the token counts at the referred agents for the hypothesis tests are generated.
from the six examples as shown in figure 7-5. Data in this figure is used to perform the ANOVA test for each of the commonly used agents.

Figures 7-6 to 7-9 show the two-way ANOVA analysis results between the six means. The two independent variables in a two-way ANOVA are called factors. Factors of the configuration and algorithm application are used to compare their effects on the token counts in this test.

### Tests of Between-Subjects Effects

**Dependent Variable: OCCUR**

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrected Model</td>
<td>334259465.6^a</td>
<td>5</td>
<td>66851393.11</td>
<td>248756.372</td>
<td>.000</td>
</tr>
<tr>
<td>Intercept</td>
<td>34236931.52</td>
<td>1</td>
<td>34236931.52</td>
<td>12739587</td>
<td>.000</td>
</tr>
<tr>
<td>CASE</td>
<td>7136531.333</td>
<td>3</td>
<td>2378843.778</td>
<td>8851.695</td>
<td>.000</td>
</tr>
<tr>
<td>Error</td>
<td>8623.333</td>
<td>30</td>
<td>297.744</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3757960560</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>334267527.9</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^a. R Squared = 1.000 (Adjusted R Squared = 1.000)

Figure 7-6 ANOVA for scheduling agent

### Tests of Between-Subjects Effects

**Dependent Variable: OCCUR**

<table>
<thead>
<tr>
<th>Source</th>
<th>Type III Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrected Model</td>
<td>349173524.9^a</td>
<td>5</td>
<td>65634764.96</td>
<td>442272.102</td>
<td>.000</td>
</tr>
<tr>
<td>Intercept</td>
<td>3401261280</td>
<td>1</td>
<td>3401261280</td>
<td>21540603</td>
<td>.000</td>
</tr>
<tr>
<td>CASE</td>
<td>524732.833</td>
<td>3</td>
<td>174910.644</td>
<td>1107732</td>
<td>.000</td>
</tr>
<tr>
<td>Error</td>
<td>4737.000</td>
<td>30</td>
<td>157.900</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>3750439842</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corrected Total</td>
<td>349178561.9</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^a. R Squared = 1.000 (Adjusted R Squared = 1.000)

Figure 7-7 ANOVA for pallet identity agent

7 - 24
The items of primary interest in the tables are the effects listed under the "Source" column and the values under the "Sig." column. Factor terms is listed under the “Source” column, They are used to tested for its ability to account for variation in the dependent variable. The dependent variable is the token occurrence of agents in this research project. For example, occurrences of example 1, 2 and 3 with algorithm application and without algorithm application are compared in the factor of corrected model. The effect of each example is compared to the other 5 examples. For the factor “Intercept”, the occurrences of the examples 1, 2 and 3 are compared
with the occurrences under the categories of with algorithm and without algorithm application. In this factor comparison, the effects of different FAC configuration to the factors of algorithm application are used to compare. If the value of "Sig" is less than the value predetermined by the experimenter, that effect is significant. Under the column of source, there are few items to be examined.

The tables deliver the same conclusion, that is, they reject the null hypothesis for the agent token counts referring to the values in the column “Sig.” Referring to the F-Test of the two-way ANOVA, the items of “Source” column are significant, their applications in the FAC control systems generate different effects on the agents in the examples. Although the examples are created with a generic FAC control system template from example 1, their agent behaviors exhibit differences. Using the ANOVA test, the occurrence counts are found suitable to be used for the test. The FAC control systems of the three examples are inherited from the generic FAC template in the example 1. These systems will exhibit different behaviors with configuration changes and different algorithm applications. The ANOVA test is used to provide proofs of these behavior differences but the differences should need another tools or methodologies to further explain these findings.

7.2.2. Comparison of the agent token counts in graphical representations
As the results of ANOVA reject the null hypothesis of the agents activities, additional scientific tool should be employed to validate the significance of the factors to the examples. Figure 7-10 illustrates the means of token counts for the agent activities. The means of the token counts for the agents are calculated by averaging of the token counts listed in the figure 7-5.
Figure 7-10 Means of token counts for the agents in this project

According to figure 7-10, the agent activities of the agents in the example 1 are always found to be greater when compared with the application examples 2 and 3. On the contrary, the means of token counts for the conveyor control agent and robot agent are found to be higher for example 2 when they are compared with examples 1.

7.2.3. Example 1 without algorithm application

The example is the generic FAC control system template. The FAC control system is built according to FAC example 1 configuration and the scheduling agent in this system does not implement any algorithm application. Some rules mined by the Apriori algorithm application are listed in figure 7-11.
The supports of these rules are around 51% and this is the percentage of the total number of occurrences among the total transaction records for the following agents: Order handling agent, Scheduling agent, Pallet Identity agent, Barcode agent, Robot agent and Conveyor Control agent.

Furthermore, the confidence of these rules are within the range of 70-100%. These rules show the occurrence frequencies of agent activities in the right hand side of the arrow, given the occurrence frequencies of the agent activities in the left hand side of the same arrow. The first rule in figure 7-12 represents that among the occurrences of other agent activities, the occurrence frequency of robot agent activity is found to be 100% (including order handling agent, scheduling agent, pallet identity agent, barcode agent and conveyor control agent).

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>30xx -&gt; 40xx</td>
<td>Interface of the order handling agent and scheduling agent</td>
<td>S~30%</td>
<td>C~40-50%</td>
</tr>
<tr>
<td>40xx -&gt; 50xx</td>
<td>Interface of the scheduling agent and pallet identity agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40xx -&gt; 70xx</td>
<td>Interface of the scheduling agent and conveyor control agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60xx -&gt; 70xx</td>
<td>Interface of the barcode identity agent and conveyor control agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70xx -&gt; 60xx</td>
<td>Interface of the conveyor control agent and barcode identity agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40xx -&gt; 20xx</td>
<td>Interface of the scheduling agent and robot agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70xx -&gt; 20xx</td>
<td>Interface of the conveyor control agent and robot agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20xx -&gt; 70xx</td>
<td>Interface of the robot agent and conveyor control agent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 7-12 Association rules for the two agents interface
Referring to figure 7-12, the table lists the association rules related to two agent activities for example 1. The activities between the two agents are revealed by these rules. All these rules have supports of around 30% which account for one third of the total transaction records. Given the occurrence record of support, there is a probability of 40-50% that the agents are found to occur together with the other agents in the same association rules. These agents in pairs constitute 1/3 of the activities and the working relationships are revealed by the strength of implication of the rules (40-50%) in the control system. From these figures, the agents are found to be working closely with each other.

### 7.2.4. Example 1 with algorithm application

This examples is implemented on the same FAC example 1 configuration. Just like previous section, this example is also implemented by using the generic framework template. In addition, the scheduling agent of this example applies the Tabu search-based algorithm for the pallet allocation process. Details of the CPN model descriptions are explained in chapter 6.

To compare the result of this example to that of example 1 without Tabu search, the same set of association rules are extracted, that is, the rules include the Order handling agent, the Scheduling agent, the Pallet Identity agent, the Barcode agent, the Robot agent and the Conveyor Control agent. These rules are the records of the agent activities in accordance with the specific sequence. Figure 7-13 lists some rules mined by the Apriori algorithm.
Figure 7-13 Example association rules for the example 1 with tabu search

In this example, a search algorithm (Tabu search-based algorithm) is implemented to the scheduling agent. This algorithm is applied to enhance the pallet allocation efficiency. The improvement is evident by the supports and confidences of the mined rules. The reduction of the support is due to efficient pallet allocation and it is reflected in the number of token records for the system output. The two outputs of the example 1 are:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1 without Tabu search</td>
<td>6998</td>
</tr>
<tr>
<td>Example 1 with Tabu Search</td>
<td>13955</td>
</tr>
</tbody>
</table>

A significant improvement is noted from the output of example 1 with Tabu Search. When the output is compared with that of without Tabu search application, the output has an increment of almost 100%. The contribution is due to the pallet allocation effectiveness.
As indicated in figure 7-14, the confidences of the association rules in the example increase by 10% when compared with figure 7-12. However, the supports of these rules are kept constant. The increment of the confidence is brought by the tightening of working relationships between two agents in the association rules.

Although the tabu search-based algorithm is implemented to the scheduling agent, the number of assembly robots is insufficient to cope with the pallet allocation rate. Thus, a reduction of the number of loaded pallets is found in this FAC example. This is significantly evident in the increased number of the token movements in the conveyor control agent.

For the place of the robot agent:
Average token count in the robot agent in Example 1 without Tabu search: 14,379
Average token count in the robot agent in Example 1 with Tabu search: 15,008

For the place of the conveyor control agent:
Average count of Tokens in the conveyor control agent in Example 1 without Tabu search: 13,998
Average count of Tokens in the conveyor control agent in Example 1 with Tabu Search: 17,906

The resources of the FAC are used efficiently and the rate of the production can be
increased by the additional of robot to the FAC.

7.2.5. Example 2 without algorithm application

This example is designed according to the other FAC configuration and this FAC has different conveyors and robots configuration. In this example, the control system reuses the agents in the generic system framework and they are modified to adapt to the new configuration. Construction of the CPN model is presented at chapter 5. The following rules are mined and extracted by the Apriori algorithm:

```
30xx 40xx 50xx 60xx 20xx => 70xx s= 30.74% c= 81.56%
30xx 40xx 50xx 60xx => 20xx 70xx s= 30.74% c= 69.52%
30xx 40xx 50xx => 60xx 20xx 70xx s= 30.74% c= 56.02%
30xx 40xx => 50xx 60xx 20xx 70xx s= 30.74% c= 50%
30xx => 40xx 50xx 60xx 20xx 70xx s= 30.74% c= 49.11%
```

Figure 7-15 Example association rules for the example 2 without Tabu search

Tokens are distributively allocated to agent places in this example. The rules are extracted such that the same rules are extracted in example 1 for ease of comparison. According to figure 7-15, the supports of the association rules related to most of the agents are found to be 30%, and the confidences are reduced when compared with the rules in example 1 without Tabu search. The agents in the rules include Order Handling agent, Scheduling agent, Pallet Identity agent, Barcode agent, Robot agent and Conveyor Control agent. They are found in the rules according to their occurrence sequences in the control system.

Referring to figure 7-15, both the supports and confidences of the association rules in example 2 are reduced. It is due to the distribution of tasks in the control system.
for FAC example no. 2. This can also be verified by the following figures:

Number of rules mined out in example 1 without Tabu search: 800
Number of rules mined out in example 2 without Tabu search: 200

The number of rules is greatly reduced when example 2 is compared with example 1. The Apriori algorithm extracts association rules according to the minimum support and confidence settings. When the tokens fired are equally distributed among the places in the control system, the number of token occurrences are also equally distributed throughout the places. Therefore, the numbers of extracted rules are reduced as some rules do not meet the minimum support setting.

Apart from the rules, one more important parameter is exported to the simulation file, it is a place marking for system output. The output is the number of pallets which have completed the assembly processes. This number provides the system performance reference for the FAC control system and the results for the two examples are:

<table>
<thead>
<tr>
<th>Example</th>
<th>Pallets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1 without Tabu</td>
<td>6998</td>
</tr>
<tr>
<td>search</td>
<td></td>
</tr>
<tr>
<td>Example 2 without Tabu</td>
<td>5224</td>
</tr>
</tbody>
</table>

For example 2, the system output is reduced because the limited 20,000 token steps are shared by the extra places when compared with example 1. Although the number of the output in example 2 is smaller, the supports and confidences of the association rules are kept constant (figure 7-16) when compared with example 1 (Figure 7-12).
### 7.2.6. Example 2 with algorithm application

This example is implemented with the FAC example 2. In this example, the tabu search-based algorithm is applied to the scheduling agent of the control system. The detailed UML model for this example is presented in chapter 6.

Some rules are mined by the Apriori algorithm as follows:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>30xx 40xx 50xx 60xx 20xx -&gt; 70xx</td>
<td>22.74%</td>
<td>61.56%</td>
</tr>
<tr>
<td>30xx 40xx 50xx 60xx -&gt; 20xx 70xx</td>
<td>22.74%</td>
<td>49.52%</td>
</tr>
<tr>
<td>30xx 40xx 50xx -&gt; 60xx 20xx 70xx</td>
<td>22.74%</td>
<td>36.92%</td>
</tr>
<tr>
<td>30xx 40xx -&gt; 50xx 60xx 20xx 70xx</td>
<td>22.74%</td>
<td>30.00%</td>
</tr>
<tr>
<td>30xx -&gt; 40xx 50xx 60xx 20xx 70xx</td>
<td>22.74%</td>
<td>28.11%</td>
</tr>
</tbody>
</table>

Comparing with figure 7-15, the supports and confidences of the same association for the agents listed in figure 7-17 are remarkably reduced. Obviously, the reduction of the rules are due to the tabu search-based algorithm.

The only difference between this example to the example at section 7.2.5 is the algorithm application in the scheduling agent. The application of the Tabu
search-based algorithm improves the performance of the FAC control system. This is revealed by the outputs:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 2 without Tabu search</td>
<td>5224</td>
</tr>
<tr>
<td>Example 2 with Tabu search</td>
<td>14491</td>
</tr>
</tbody>
</table>

The output of this example increases by three times that of example without Tabu search. This is the contribution of the tabu search-based algorithm in the scheduling agent.

Although the tabu search-based algorithm enhances the pallet allocation capability, the number of robots restrict the output of the FAC. The number of tokens in the robot places is halved that of example without Tabu search. Although the workload of the robots is decreased, the number of tokens found in the conveyor is also decreased. Tokens related to the robots are found insufficient for efficient pallet allocation. This factor is revealed as follows:

For the robot agent place:
- Occurrences of Token in the robot agent in Example 2 without Tabu search: 15008
- Occurrences of Token in the robot agent in Example 2 with Tabu search: 7959

For the conveyor control agent place:
- Occurrences of Token in the conveyor control agent in Example 2 without Tabu search: 17,906
- Occurrences of Token in the conveyor control agent in Example 2 with Tabu search: 14,960

The tabu search-based application in the scheduling agent is the main contribution for the output in this example. In example without tabu search, the number of token count in conveyor control agent is doubled when compared to this example. But
this example shows that the counts are halved when compared to application without Tabu search. According to the outputs of these examples, the pallets are effectively conveyed to the robots for assembly process which the output of this example is increased by three times.

The effectiveness of this control system example is also revealed by the reduction of token count of the robot agent and conveyor agent, their token counts are reduced when compare with that of example 1. In this example, the pallets are queuing in the control system and the number of robots is insufficient to cope with the incoming pallets. Thus the token counts of these agents are increased when compared to example 1. Configuration change, besides the tabu search based algorithm application in this project, is another factor affecting the system performance.

**7.2.7. Example 3 without algorithm application**

Flexibility of the generic control system has been illustrated by the examples 1 and 2 using the generic FAC template. In this example, the hypothetic FAC example 3 with twelve machines is analyzed. Graph 7-4 shows some results extracted from the data mining algorithm application. The figure shows the occurrence counts at the twelve machines.
Each of the four simulation trials in graph 7-4 executed 20,000 simulation steps as well as simulation examples 1 and 2. The total number of occurrence counts for the eight machining centers are almost the same in the four simulation trials. For the same machine, the variations of the occurrence counts are due to the random pallet allocation to the machines. Ranges for the occurrence counts for the machines are from 60 to 150 in this graph.

This control system example is able to produce four product types. Graph 7-5 shows the product throughputs among the four simulation trials. The throughputs vary among the trials because they are due to the randomness simulation setting.
Throughput for Product 1, 2, 3 and 4

Graph 7-5 Product A and B throughputs for the FAC model simulations

7.2.8. Example 3 with algorithm application

Tabu search is applied to example 3 control system, the application is trying to eliminate the randomness effect to the FAC components. The scheduling agent will assign pallets to machining centers according to Tabu search.

Simulation is then performed using the new setting and graph 7-6 shows the results of the machine throughputs. The machine throughputs are reduced to range 160-175 which is comparatively narrowed than graph 7-4 (The range of the graph 7-6 is 120-200).

Although the range is narrower, the modified FAC product throughputs are improved according to graph 7-7. Improvement of the product throughputs provides evidence of higher machine utilizations using Tabu search algorithm. The algorithm application to the scheduling agent modification is effectively revealed by the simulation model.
Graph 7-8 shows the occurrences counts of these agents in the comparison between with-algorithm or no algorithm application. The numbers of agent occurrences are averaged from 6 simulation trials. Three agents in the figure are the scheduling agent, the robot agent and the machine control agent. The three agent occurrences of the rule application example are all increased when they are compared with the no rule application example. The figure provides evidence that the machine control agent, the scheduling agent and the robot agent were busy in algorithm application example simulations.
Throughputs for Product 1, 2, 3 and 4

Graph 7-7 Product A and B throughputs comparison

Numbers of occurrences for agents

Graph 7-8. Numbers of occurrences for agent tokens for the three agents.
Agent interaction between two agent (no Tabu Search application)

| 40xx -> 708x | Interface of the scheduling agent and robot agent   |
| 708x -> 705x | Interface of the robot agent and machine control agent |
| 705x -> 708x | Interface of machine control agent and robot agent   |
|              | S~30%                                                |
|              | C~60%                                                |

Agent interaction between two agent (with Tabu Search application)

| 40xx -> 708x | Interface of the scheduling agent and robot agent   |
| 708x -> 705x | Interface of the robot agent and machine control agent |
| 705x -> 708x | Interface of machine control agent and robot agent   |
|              | S~45%                                                |
|              | C~80%                                                |

Figure 7-18. Association rules for the agent interactions between two agents

The frequent agent occurrences counts are then explained by the association rules. Figure 7-18 shows results to the three agents with algorithm and no algorithm application. The figure shows association rules related to any two of the three agents (The scheduling agent, robot agent and machine control agent). For example, the figure shows association rules for interfaces between

- scheduling agent and robot agent,
- robot agent and machine control agent
- machine control agent and robot agent

The supports of the association rules between two agents are around 30% and 45% to rule and without rule application respectively. The supports of rule application has 50% occurrences increment (from 30% to 45%) when they are compared to supports of no rule application. Simulation shows agents effectively to be used by better pallet allocation to machining center with lower utilization rate. Number of the product A and B are also produced by the rule application.

7.2.9. Examples evaluation

Figure 7-24 summaries the differences of the FAC examples for data analysis.
Referring to the summary (figure 7-19), six examples are used to perform the analysis. Preliminary statistical data have been extracted and introduced in the sections. These data are extracted by the Apriori algorithm application in this project. The Apriori algorithm is used to count the number of occurrences for each agent code defined at figure 7-3. In the process of association rule extraction, appearances of several agent codes coexisted are used as reference in association rule extraction. The Apriori algorithm is useful in this project, this application is used to extract both the statistical data and association rule extraction.

Beside the statistical data analysis, the association rules extracted for the examples are used to further explain the effects of FAC changes and algorithm application. In agent based control system, effective interactions between agents can greatly increase system performance.
<table>
<thead>
<tr>
<th>Label</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Order handling agent and scheduling agent 30xx -&gt; 40xx</td>
</tr>
<tr>
<td>2</td>
<td>Scheduling agent and pallet identity agent 40xx -&gt; 50xx</td>
</tr>
<tr>
<td>3</td>
<td>Scheduling agent and conveyor control agent 40xx -&gt; 70xx</td>
</tr>
<tr>
<td>4</td>
<td>Barcode identity agent and conveyor control agent 60xx -&gt; 70xx</td>
</tr>
<tr>
<td>5</td>
<td>Conveyor control agent and barcode identity agent 70xx -&gt; 60xx</td>
</tr>
<tr>
<td>6</td>
<td>Scheduling agent and robot agent 40xx -&gt; 20xx</td>
</tr>
<tr>
<td>7</td>
<td>Conveyor control agent and robot agent 70xx -&gt; 20xx</td>
</tr>
<tr>
<td>8</td>
<td>Robot agent and conveyor control agent 20xx -&gt; 70x</td>
</tr>
</tbody>
</table>

Figure 7-20 Labels for the figure 7-21

Figure 7-20 lists the labels used to identify the supports and confidences referring to the interactions between agents. For example, label 1 in figure 7-21 is the information for interactions for the order handling agent and the scheduling agent. It provides the supports and confidences of Apriori algorithm to the FAC examples.

The increase of the supports and confidences of the FAC examples from algorithm application to without algorithm application provides information, that the cooperation between the two agents are frequently interacted to the efficiency of algorithm application. Effective allocation of the pallets to the FAC will also increase the workloads of the agents. Therefore, the pallet tokens frequently appear in the agent places. The increased rate of the token flow to the CPN components reflects the heavy workload to the referred agent, provided that the outputs of the FAC examples are also increased. In the examples, the algorithm application improved the product outputs by the efficient pallet allocation to production process. Hence, the agents are proved to work effectively with each other in the algorithm application examples.
<table>
<thead>
<tr>
<th></th>
<th>Example 1</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>2</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>3</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>4</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>5</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>6</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>7</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
<tr>
<td>8</td>
<td>S30% C40-50%</td>
<td>S30% C60%</td>
<td>S30% C40-50%</td>
<td>S45% C50-60%</td>
<td>S30% C60%</td>
<td>S45% C80%</td>
</tr>
</tbody>
</table>

Figure 7-21 Supports and confidences for the examples

| 30x 40x 50x 60x 20x => 70x |
| 30x 40x 50x 60x => 20x 70x |
| 30x 40x 50x => 60x 20x 70x |
| 30x 40x => 50x 60x 20x 70x |
| 30x => 40x 50x 60x 20x 70x |

Figure 7-22 Association rules for the examples

The agents in the association rules (Figure 7-22) include Order Handling agent, Scheduling agent, Pallet Identity agent, Barcode agent, Robot agent and Conveyor Control agent. They are found in the rules according to their occurrence sequences in the control system. These rules provide operation information for most of the system agents in the examples. These agents are also inherited by the generic FAC
template and they are implemented to all these examples.

<table>
<thead>
<tr>
<th></th>
<th>Example 1</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tabu Search</td>
<td>Tabu Search</td>
<td>Tabu Search</td>
<td>Tabu Search</td>
<td>Tabu Search</td>
<td>Tabu Search</td>
</tr>
<tr>
<td>Support &amp; Confidence</td>
<td>S51%</td>
<td>S31%</td>
<td>S30%</td>
<td>S22%</td>
<td>S35%</td>
<td>S15%</td>
</tr>
<tr>
<td></td>
<td>C100%</td>
<td>C56%</td>
<td>C81%</td>
<td>C50%</td>
<td>C85%</td>
<td>C50%</td>
</tr>
</tbody>
</table>

Figure 7-23 Summary for the supports and confidences to figure 7-27

Figure 7-23 shows the supports and confidences for the examples. The figures show that the algorithm application decreases the values for both the values. The association for the agents in figure 7-22 is inversely proportional to interactions between two agents in figure 7-21. Although the two agent interactions are busy, the cooperation between several agents are reduced. This is because interactions of the two agents frequently appear in the data mining time frame. When these pairs of data occupied within the time frame, their interactions with other agents are reduced. For the same timeframe of data extracted, the same association rules in figure 7-22 for the examples without algorithm application are comparatively smaller to the algorithm application examples.

7.2.10. **CPN property verification**

The examples were performed CPN simulation and their simulation data are exported for evaluation. The simulation can guarantee the reachabilities of the agents. The reachability test using the CPN simulation tool makes sure the constructed CPN has no deadlock or communication interaction problem before the model is simulated. The reachability tree can also be revealed at any time and it is automatically generated by this editor tool. This feature is important to highly dynamic systems such as FAC. Especially the designed of the FAC examples are based on the
generic FAC template from the example 1. The reachability of CPN model was previously done by reachability tree, the precedences and postcedences of places and transitions are listed in tree-liked structure.

Deadlock or bottleneck of the FAC examples are also verified by the token flows. Tokens are found blocking at the regions of the problems. The exported data plus the Apriori algorithm application that can be used complimentarily with this visual checking to avoid the occurrences of these deadlock and bottleneck problems.

7.2.11. Analysis methodology conclusion

Examples 1 to 3 show the adaptabilities of the analysis methodology to FAC. These examples are constructed by the same generic FAC control system and their system performance analysis can be used for comparison purpose. Analysis results are used to predict the FAC performances during the analysis stage. For example, the effects of the Tabu search-based algorithm are compared between the examples.

Apart from the algorithm application, the adaptability of the generic control system to a large scale FAC example is revealed by example 3. A tabu search application is also used to compare the performance of the same system. From these examples, the analysis methodologies show the advantages to the design and analysis of the complicated FAC control system.

7.3. System implementation

Formal specification of manufacturing systems has been studied for several decades. Bernard et al (1997) presented a study of manufacturing control system using object
oriented modeling. This model is adaptable to design changes and its prevents excessive system modifications by the inheritance of objects and classes. The UML application is now applied in many network-based automated control systems (Kumar et al, 2004).

The UML modeling tool “Rational Rose” is used for the implementation of the UML models in this project. It is an industrial standard modeling tools for UML construction. Editing models such as class diagram, sequence diagram, deployment diagram, collaboration diagram, use case diagram, state transition diagram and component diagram are available. This tool is fully compatible with UML supported programming languages such as Visual studio 6.0 and Visual .Net and hence supporting Visual Java and Visual Basic.

7.3.1. **Work Flow for the generic FAC control system**

The flow chart in figure 7-24(next page) shows the work flow for the scheduling agent input. The scheduling agent is able to receive input order in real time. The continuous order input simulates the manufacturing orders in practical applications. In practice, manufacturing orders are usually generated by enterprise information systems such as ERP & MRP systems. It is possible for the agent-based FAC control system to link up with the ERP system to obtain manufacturing orders as system inputs.
Once the orders have been received by the scheduling agent, the connection status with the other agents are displayed and highlighted in the scheduling agent computer (Figure 7-25). In this GUI, the fields “Client 1”, “Client 2” and “Client 3” refer to the online status of the robot control agent, the pallet identity agent and the conveyor control agent respectively.
Figure 7-25 Scheduling Agent’s Graphic Interface after Agents’ Connection

Figure 7-26 Pallet Identity and Barcode Reading Agents’ Graphic Interface

Order display field

Barcode info.
Pallet entry point
Pallet identity field
Light signal
Barcode info. 1
The FAC control system is initialized by the order input. Pallets are loaded on the conveyor system manually at the pallet entry point. The pallet identity agent interprets the pallet identity number according to the pallet entry priorities. That is, the first pallet will be assigned the number “0”, the second one will be assigned the number “1” and so on. The pallet identity numbers are displayed at the pallet identity fields in the pallet identity agent graphic user interface (Figure 7-26).

When a pallet passes one of the twelve sensors in the assembly cell, the corresponding light signal will be highlighted and the corresponding pallet number is displayed in the field of pallet identity. If a pallet part is identified to be the next palletized part to be operated according to the task assignment sequence, the pallet identity agent will validate the pallet identity with information read by the barcode reading agent. A barcode strip is attached to the side of each pallet for reading. Corresponding to the code recorded in the barcode, the appropriate task information
including the robot program can be retrieved.

After the confirmation of the pallet identity, the scheduling agent will send requests to the robot control agent and the conveyor control agent respectively. Receiving the pallet parking request from the scheduling agent, the conveyor control agent will direct the conveyor controller to position and clamp the pallet at the desired robot processing area. After the pallet is clamped at one of the robot processing areas, the signal “End” is displayed by the conveyor control agent’s GUI. Regarding the robot control agent, it receives the robot processing request from the scheduling agent and hence it will send the corresponding robot program to the Adept robot controller (Figure 7-28). The corresponding robot can then start the required assembly operations.

![Diagram of robot processing](image)

Figure 7-28 The example shows the robot processing file sending to the robot controller
After the robot has completed the assembly operation on the component according to the robot program, the robot controller will send the robot program end signal to the robot control agent. The ending acknowledgement signal will then be sent from the robot control agent to the scheduling agent, pallet identity agent and the conveyor control agent. The pallet will then be released from the robot processing area and put
back onto the conveyor (Figure 7-32). Because the conveyors are stopped by the conveyor control agent to prevent collision, the conveyor will be restarted. The work flow is summarized in the flow chart in Figure 7-33. Figure 7-34 shows the interaction among the various agents, and the corresponding time frame is given in Figure 7-35.
Pallet identity agent detects next order pallet arrives to the robot processing

Pallet identity agent compares the pallet information with the barcode strip from barcode reading agent

Valid information

Send the pallet information to scheduling agent

Scheduling agent sends robot processing request to robot

Send the pallet parking request to the conveyor control agent

Performs pallet parking process in the robot processing area

Processing complete?

NO

YES

Download the robot processing program to the desired robot control from robot control agent

Robot perform their required processing to the pallet

Figure 7-31 Flow Chart for the Robot Processing Steps after Order Received by the Scheduling Agent
Figure 7-32 Conveyor Control Agent’s GUI during Operation

Figure 7-33 Flow Chart for the FMS Control System after the End of Robot Process

Robot has been clamped and fixed
7.3.2. **Tabu search-based algorithm and configuration changes implementation in example 2**

Details of the generic FAC control system has been described in section 7.3.1. This framework is the basis for the system configurations for the FAC control systems in examples. It is necessary to modify the generic framework to suit the requirements
in the other FAC examples. Modifications are mainly focused to the addition of objects to the agents.

Basically, the configuration change for example 1 from the generic FAC involves the addition of an assembly robot, the robot agent should therefore establish one more interface to communicate to the controller of the additional robot. The place “c” is the extra robot in the CPN model (Figure 5-78) and “Com3” in figure 7-36b shows the object for this amendment to the configuration changes. The modified diagram is shown in figure 7-37 for the example 1. This object belongs to the class of robot agent in the UML model.

After the object is created in the class model, the collaboration diagram and sequence diagram are modified to cope with the new object (figure 7-38 to figure 7-41).
Figure 7-36a Generic robot agent class model
Figure 7-36b Modify robot agent in example 2
Figure 7-37 Modified class diagram for the robot agent
Figure 7-38 Sequence diagram for the robot agent in the generic framework
Figure 7-39 Sequence diagram for the new robot agent in example 2
Figure 7-40 Collaborative diagram for the robot agent in the generic framework
Figure 7-41 Collaborative diagram for the new robot agent in example 2
The new object “Com3” is able to send messages to the shop floor robot controller and monitor the status of the robots. The message sending capability is inherited from some newly created methods of the new robot agent. They are shown in the sequence diagram in figure 7-39 and collaborative diagram in figure 7-41. When these two figures compare with their generic robot agent frameworks in figure 7-38 and 7-40 respectively, the new methods are responsible for the message sending functionality. Figure 7-42 shows implementation for these newly created methods from these modifications.

```vbnet
Option Explicit

Public aRobotProgram As RobotProgram

Public Function send() As String
End Function

Public Function receive() As String
End Function
```

Figure 7-42 Implementation code for the new robot agent.

The new object class and methods are now created, they are implemented into the robot agent class for the generation of the programming framework. A new object is inherited from the newly created class Com3 shown in figure 7-43. Method stopRobot3() is created for the emergency stopping of the newly added robot. By using the new object of Com3, the newly added methods in figures 7-39 and 7-41 can be implemented and then used to control the extra robot in this example.
Option Explicit

Private r1flag As Boolean
Private r2flag As Boolean
Private r1check As Boolean
Private r2check As Boolean

Public awskClient1_DataArrival As wskClient1_DataArrival
Public awskClient2_DataArrival As wskClient2_DataArrival
Public aCom1 As Com1
Public aCom2 As Com2
Public aCom3 As Com3

Public Function cmdConnect_Click1() As void
End Function

Public Function cmdConnect_Click2() As void
End Function

Public Function logout() As void
End Function

Public Function login() As void
End Function

Public Function stopRobot1() As void
End Function

Public Function stopRobot2() As void
End Function

Public Function stopRobot3() As void
End Function

Figure 7-43 Modified class for the new robot agent example
Apart from the configuration change from the generic FAC example 1 as shown in example 1, the tabu search-based application is implemented in this example. Modification of the generic UML model is illustrated by using this example 2. The tabu search-based application for the FAC control system is embedded to the scheduling agent class.

The scheduling agent originally equips with interfaces to other agents and they are essential to cooperate with each other for pallet allocation to the assembly process. This agent is the control agent to coordinate the system work. Results of this algorithm application should be implemented to the system and the agents work with each other to implement the effective pallet allocation schedule.

Figures 7-44 and 7-45 show the class diagrams for the scheduling agents of the generic control framework and example 4 respectively. A new object “Tabu” in figure 7-45 replaces the object “ScheduleRule” in figure 7-44 for this tabu-search algorithm application. Originally, the “ScheduleRule” is proposed for the scheduling agent and it is reserved for any scheduling rule or algorithm in the generic application example. In example 1, the scheduling rule of “First-come-first serve” is used. For the Tabu search-based application, the object is replaced.

In the Tabu object in figure 7-45, a method “generatecode” is created to be responsible for the generation of candidates for the search algorithm. The candidates representing best routing is sorted out heuristically with the assistance of the attributes “currentnode”, “bestnode” and “newnode” in this new object.
Figure 7-44 The class diagram for the scheduling agent in the generic control system
Figure 7.45 Class diagram for the new Scheduling agent in example 3.
Figures 7-46 and 7-47 show the sequence and collaboration diagrams respectively for the scheduling agent in example 2. The sequence diagram shows that the method of realtimecheck() is used for retrieving updated pallet information from the commonData object. The “commonData” object provides updated shop floor information, including the pallet locations in the conveyors. And the pallet information is necessary for the generatenode() method with which the data is used for the tabu search-based application. The solution of the tabu search will be retrieved by the method taburesult() and this result is further sent to the conveyor control agent. This agent will convey the pallet to the right assembly robot for processing. The two figures summarize the changes of the scheduling agent in example 2. In the generic control framework, the object inside the rectangulars are missing because the “ScheduleRule” object is reserved for algorithm application.

The Tabu search application is implemented back to the UML model in accordance with the algorithm presented in Chapter 6. The UML model is then updated with the Tabu search applications.
Figure 7-46 Sequence diagram for the Scheduling agent in example 2
Figure 7-47 Collaboration diagram for the Scheduling agent in example 2
The newly modified scheduling agent of example 4 is implemented with the Java programming language. Figure 7-48 shows the codes in Java for the Tabu object. The codes show that the methods and attributes are created in accordance with the class diagram in figure 7-45.

```java
//Source file: D:\Project\java\SchedulingAgent\Tabu.java

package SchedulingAgent;

public class Tabu
{
    public realtimecheck();
    public generatenode();
    public taburesult();

    public Tabu() {
        public currentnode thecurrentnode;
        public bestnode thebestnode;
        public newnode thenewnode;
    }
}
```

Figure 7-48 Tabu class in Java

The newly created Tabu class is further implemented in the scheduling agent framework as shown in figure 7-49. By using the newly created object “tabu”, the object inherits the characteristics which have been defined in figure 7-48. Thereafter, the Tabu pseudo code presented in the figure is implemented to the programming framework in the Tabu object.
7.4. Chapter Summary

The design and analysis processes for the generic agent-based manufacturing control system are illustrated in this chapter. The generic agent-based control system framework is built according to the agent classification. Characteristics of the generic agent-based control system are revealed in the examples. Flexibilities such as reconfiguration changes and tabu-search algorithm application are applied to the
four examples. The structures of these examples are based on the generic control framework. Association rules in this project are used to explore the relationships between agents and this is contributed to the evaluation of agent interactions in the dynamic control system. The data mining algorithm can also provide some reference information which are extracted from repeated data counting process. The UML components are modified in accordance with the dynamic models.
CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this thesis, a generic agent-based manufacturing control system is proposed and designed corresponding to a general flexible assembly cell (FAC) configuration. The agent-based FAC control system is actually a multi-agent system (MAS) which is able to handle the scheduling, control and monitoring functions of the assembly cell. The MAS comprises various agents such as the scheduling agent, robot agent, pallet control agent, in accordance with the different entities and functions of a general FAC. Accordingly, specifications of the different functional agents are defined. The object-oriented modeling methodology is adopted in the design and specification of the agent-based FAC control system and the individual agents. In fact, it is the aim of this project to use the UML as the modeling tool in designing the agent-based system. From the studies in this thesis, the UML technology is found to be a suitable and an useful tool for the initial modeling and documentation of a complex agent-based FAC control system. Physical realization of the UML use case illustrates how to construct the functional entities for the generic agent-based FAC control system. The construction of the class diagrams in the UML design models shows the impact of the constraints to restrict the functional and non-functional requirements of the agent classes.

Reusing this control structure, complicated control elements for agents can be added to other agent-based manufacturing control system configurations. This formal methodology for designing agent-based FMS control systems helps system
designers to describe the configuration of the system in terms of the distributed functionalities among different functional entities.

The use case of the static UML diagram is able to describe event happenings in the FAC control system but it lacks of the dynamic analytical capability. A few researchers have proposed to extend the UML to cater for the dynamic requirements of agent-based systems. In this thesis, we consider it is more appropriate to use the standard UML in the initial modeling and representation of MAS, and then more powerful dynamic modeling tools can be used to represent the dynamic behaviours and interactions of agents in the MAS. In this regards, the Color Petri Net (CPN), an extended form of the Petri net modeling tool, is used for the representation of dynamic behaviors and interactions of the agent-based FAC control system. In fact, a hybrid of the UML and CPN modeling approaches has been proposed for modeling agent-based systems in this thesis. The UML and CPN tools are complementarily integrated the representation of FAC control system. An approach has been developed in this thesis to establish an automatic mapping between the UML and CPN models. After the preliminary static framework of an FMS control system has been defined in UML representations, the CPN model can then be established to represent the interactions among agents. In the corresponding CPN model, resource allocations by agents are represented by the colour tokens. As illustrated by the case studies in this thesis, this hybrid UML-CPN approach is a suitable and powerful approach for modeling agent-based manufacturing control systems. Compare with the use of non-standard UML extensions to incorporate agent-based modeling requirements, our hybrid approach is based on standard UML. Besides, the few currently available extended UML tools are designed for the contemporary agent technology and each of them is restricted to a limited set of particular agent-based
applications. The extended UML tools have to be further developed to cope with future agent advancements. Besides, the UML and its extensions are basically modeling tool, and not able to support dynamic system simulation and performance evaluation. In contrast, the CPN models can accommodate the increased complexities of agent technology and manufacturing control systems. The CPN also provides a framework for the construction and analysis of the distributed and concurrent system behaviors, and it is able to simulate and evaluate the dynamic functioning of multi-agent systems.

The scheduling agent is responsible for allocating loaded pallets to the robots for processing. Initially, pallets are dispatched with the simple first-come first-served (FCFS) rule. A tabu search-based algorithm has then been proposed and implemented in the scheduling agent for pallet allocation. As illustrated in this thesis, it is then straightforward to modify the colour tokens in the corresponding places in the CPN model to incorporate this change of agent functioning capability.

Two different FAC configurations are used as examples of manufacturing systems in this thesis. The hybrid UML-CPN modeling approach is used for the design and representation of each of the three FAC configurations. The two pallet allocation approaches, that is, the FCFS rule and the tabu search-based algorithm, are separately applied in the three FAC configurations. Therefore, altogether three agent-based FAC control system configurations have been established as case studies in the thesis. The UML and CPN models are then established for the four FAC control system configuration examples. Simulation and analysis of the CPN models are then carried out to evaluate the interactions and performances of the four different MAS configurations. Token activities of each example are recorded and each of these
simulation exports a file for analysis. Each simulation file contains a bulk of agent activity data. A data mining algorithm, the Apriori algorithm, is then used to mine the agent activity data to obtain useful information for evaluating the performances of the different agent-based FAC control systems in the case studies.

As depicted in this thesis, a generic agent-based control system framework is firstly built according to the agent classification with respect to a general FAC configuration. This generic system architecture is then used in the modeling of the actual multi-agent systems in the four different FAC configurations in the case studies. It was shown that the generic agent-based control system is able to support different kinds of flexibilities such as system reconfiguration and/or the application of different scheduling rules. The proposed hybrid UML-CPN modeling approach has been proved to be a useful and flexible tool for modelling the static and dynamic behaviours of agent-based systems. Furthermore, simulation and analysis of the CPN models are carried out to evaluate the performances of the different agent-based FAC control system configurations. After the simulation run, association rules are used to explore the relationships between agents and this is contributed to the evaluation of agent interactions in the dynamic control system. The data mining algorithm can also provide some reference information extracted from repeated data counting processes. Based on the evaluation results, the UML components of the agent-based system model can be modified in accordance with the dynamic models.

The proposed methodology for designing and analyzing agent-based manufacturing control systems has been successfully implemented in the illustrative examples in this thesis. The hybrid UML-CPN models are able to cope with the flexibility requirements of the complex flexible manufacturing environments. As
verified by the examples, the combination of the UML and CPN models are able to support the robust design and flexibility requirements of the agent-based control systems corresponding to changes or disturbances in flexible manufacturing systems. This design methodology and analytical framework based on the UML and CPN models are found to be adaptable and flexible, and is able to cope with the complexity of the planning and scheduling problems in the complex automated manufacturing environments of different system configurations.

8.2 Future Work

This scope of the thesis can be further extended in the following research directions:

- For future work, the MAS structure can be further explored to incorporate other decentralized complexities of the manufacturing control systems. This can be achieved by adding other intelligent algorithms to the flexible agent structure proposed in this research project.

- A standardized agent protocol can be established to act as the agent object, it can then serve to provide the standardized communication protocol to all agents in the control system. This is achieved by object encapsulation for the agent objects and the protocol can further enhance the agent structure flexibility.

- The current MAS model can be extended to incorporate resource failures or other uncertainties. This demands the capabilities for dynamic task reallocation in case of this kind of unpredicted events. This may need the incorporation of additional entities such as buffer or duplicate resources in the hybrid UML-CPN models. A strategy for dynamic process planning and scheduling is also required.

- A basic agent negotiation scheme is utilized in this research project. The effect
of negotiation applications can be studied in future research. More specific negotiation strategies can be studied and implemented to the generic agent-based control system.

- The research methodology can be further extended and applied to other agent-based applications such as supply chain management. In particular, the applications and adaptability of the hybrid UML-CPM methodologies for designing and analyzing MAS can be further studied.
REFERENCES


Adzakpa, Kossi P.; Adjallah, Kondo H.; Yalaoui, Farouk. On-line maintenance job scheduling and assignment to resources in distributed systems by heuristic-based optimization. Journal of Intelligent Manufacturing. 15, 2004, 131-140


Agrawal, R.; Aggarwal, C. and Prasad, V. V. V. Depth-first generation of large itemsets for association rules. IBM Tech. Report RC21538, July 1999


Arasten, Amund; Brugali, Davide; Menga, Giuseppe; Mosconi, Luca. Using Agent Technology in Virtual Factories. Proceedings of the 27th International Symposium on Industrial Robotics (ISIR’96) 6-8 October, 1996 Milano.


Berio, Giuseppe; Dileva, Antonio; Giolito, Piercarlo; Vernadat, Francois. Object-oriented process development in the M*-OBJECT methodology. Journal of Intelligent Manufacturing. 11, 2000, 113-125


Berzal, Fernando; Cubero, Juan-Carlos; Marín, Nicolás and Serrano, José-María TBAR: An efficient method for association rule mining in relational databases Data & Knowledge Engineering, Volume 37, Issue 1, April 2001., 47-64


Bongaerts, Luc; Monostori, Laszlo; McFarlane, Duncan; Kadar, Botond. Hierarchy in distributed shop floor control. Computers in Industry, 43, 2000, 123-137


Brin, S; Motwani, R. and Silverstein, C. Beyond market basket: Generalizing association rules to correlations. SIGMODE’ 97, 265-276, 1997


Carrie, Allan S. From integrated enterprises to regional clusters: the changing basis of competition. Computers in Industry, 42, 2000, 289-298

Cavalieri, Sergio; Garetti, Marco; Macchi, Marco; Taisch, Marco. An experimental benchmarking of two multi-agent architectures for production scheduling and control. Computers in Industry, 43, 2000, 139-152


Fletcher, M., Brennan, R. W. and Norrie, D. H. (2003), Modeling and reconfiguring intelligent holonic manufacturing systems with Internet-based mobile agents, Journal of Intelligent Manufacturing, 14, 7-23

Fox, Mark S.; Barbuceanu, Mihai; Teigen, Rune. Agent-Oriented Supply-Chain Management. International journal of Flexible Manufacturing Systems. 12, 2000, 165-188


Friedrich, Holger, Rogalla, Oliver and Dillmann, Rudiger, Integrating skills into multi-agents systems. Journal of Intelligent Manufacturing 1998, 9, 119-127


Glover, F. Tabu search – Part II. ORSA Journal on Computing 2(1), 1989, 4-32


Ito, Teruaki; Mousavi Abadi, S. M. Agent-based material handling and inventory planning in warehouse. Journal of Intelligent Manufacturing. 13, 2002, 201-210


Jensen, K. (Kurt), Color Petri nets: basic concepts, analysis methods, and practical use. Berlin: Springer-Verlag, 1992


Kotak, Dilip; Wu, Shaohong; Fleetwood, Martin; Tamoto, Hiroshi. Agent-based holonic design and operations environment for distributed manufacturing. Computers in Industry, 52, 2003, 95-108


Landrieu, Antoine; Mati, Yazid; Binder, Zdenek. A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. Journal of Intelligent Manufacturing. 12, 2001, 497-508


Liu, Dongsheng; Wang, Jianmin; Chan, C. F. Stephen; Sun, Jiaguang; Zhang, Li. Modeling workflow processes with colored Petri nets. Computers in Industry, 49, 2002, 267-281


Malakooti, Behnam; Raman, Vishnu. An interactive mutli-objective artificial neural network approach for machine setup optimization. Journal of Intelligent Manufacturing. 11, 2000, 41-50


McFarlane, Duncan; Bongaerts, Luc; Monostori, Laszlo; Kadar, Botond. Hierarchy in distributed shop floor control. Computers in Industry, 43, 2000, 123-137


O’Kane, James F. A knowledge-based system for reactive scheduling decision-making in FMS. Journal of Intelligent Manufacturing, 11, 2000, 461-474


Raman, Ramanujam; Marefat, M. Integrated process planning using tool/process capabilities and heuristic search. Journal of Intelligent Manufacturing. 15, 2004, 141-174


Saitou, Kazuhiro; Malpathak, Samir; Qvam, Helge.  Robust design of flexible manufacturing systems using, colored Petri net and genetic algorithm.  Journal of Intelligent Manufacturing. 13, 2002, 339-351


Smirnov, Alexander V. Agent-based Knowledge management for concurrent enterprise configuring. 1998


Tamas Kis A Petri net model for integrated process and job shop production planning Journal of Intelligent Manufacturing Volume 11, Issue 2, Apr 2000, 191-207

Tan, Wei; Khoshnevis, Behrokh. Integration of process planning and scheduling-a review. Journal of Intelligent Manufacturing. 11, 2000, 51-63


Trentesaux, D.; Pesin, P.; Tahon, C. Distributed artificial intelligence for FMS scheduling, control and design support. Journal of Intelligent Manufacturing. 11, 2000, 573-589


Whinston, A. Intelligent agents as a basis for decision support systems.  Decision Support Systems, 20, 1, 1997


Zakarian, A., Mohanty, P. A New Data Mining Algorithm for Manufacturing Process Control, 2003


