# Petri Nets Simulation Tool

โดย

นายก่อกิจ วีระอาชากุล

นายบุญชัย เอี่ยมเมตตา


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

# Petri Nets Simulation Tool


โดย

นายก่อกิจ วีระอาชากุล 37014011 4D

นายบุญชัย เอี่ยมเมตตา 37014216 4D


อาจารย์ที่ปรึกษา

อาจารย์ อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง Petri Nets Simulation Tool

ผู้จัดทำ 1. นายก่อกิจ วีระอาชากุล 37014011 4D

2. นายบุญชัย เอี่ยมเมตตา 37014216 4D

อาจารย์ อภิเนตร อูนากูล

(.....)

อาจารย์ที่ปรึกษา

# Petri Nets Simulation Tool

ผู้จัดทำ
1. นายก่อกิจ วีระอาชากุล
2. นายบุญชัย เอี่ยมเมตตา

อาจารย์ที่ปรึกษา
อาจารย์อภิเนตร อูนากูล
ปีการศึกษา 2540

บทคัดย่อ

ในการที่จะศึกษาระบบหนึ่งๆ เพื่อที่จะอธิบายและศึกษากระบวนการทำงานของระบบ เกี่ยวกับขั้นตอนการทำงาน, การแจกจ่ายทรัพยากร หรือการเกิดเดทล็อก การอธิบายด้วยโมเดลเป็นวิธีการที่จะช่วยให้เราเห็นภาพ อธิบายได้ง่ายและชัดเจนยิ่งขึ้น ซึ่งโมเดลที่จะใช้จำลองระบบนั้นได้มีการศึกษาและวิจัยออกแบบขึ้นมาหลายชนิด เช่น State Machine, Finite Automation เป็นต้น

Petri Nets เป็นแบบจำลองทางด้านกราฟฟิกที่ได้รับการออกแบบและพัฒนาให้สามารถใช้ในการอธิบายและศึกษาข้อมูลจากการทำงานของระบบในหลายๆ รูปแบบได้ดี

โครงงานนี้เป็นการสร้างทูลหรือแอพพลิเคชั่นที่จะนำมาใช้ช่วยในการศึกษาเกี่ยวกับโมเดลของ Petri Nets โดยสามารถใช้ออกแบบ และแสดงการทำงานจากโมเดลที่ออกแบบ เพื่อให้เห็นการทำงานในแต่ละขั้นตอน รวมทั้งสามารถวิเคราะห์ผลที่ได้จากการทำงานด้วย ซึ่งจะช่วยให้เราสามารถทำการวิเคราะห์และศึกษาการทำงานของระบบได้ง่ายขึ้น ดังนั้นแอพพลิเคชั่นที่สร้างขึ้นมานี้จึงนับว่าเป็นเรื่องที่น่าสนใจ และเป็นประโยชน์สำหรับการศึกษาและวิเคราะห์เกี่ยวกับระบบ

# Petri Nets Simulation Tool

1. Kokit Werarchakul

2. Boonchai Iammetta

Advisor

Apinetr Unakul

Academic Year 1997

Abstract

To describe and study a working system, working steps, a resource distribution or a deadlock incident, the modeling is a good way to help us to understand them easily and clearly. Therefore, the great number of models are designed, for example, State Machine and Finite Automation, etc.

Petri Nets is one of the graphic model which is designed and developed to describe and study several working process information.

The objectives of this project is to build up a sufficient tool or a sufficient application for studying Petri Nets model. The tool or application is not only designed to do modeling to clearly reflect all working processes step by step but and also to analyse the output of the working processes, therefore, it makes us to understand and analyse the working system easily, clearly, accurately and so forth. With all advantages mentioned above, the project is very interesting and it is worth to be researched.

# สารบัญ


บทคัดย่อภาษาไทย ..... I

บทคัดย่อภาษาอังกฤษ ..... II

สารบัญ ..... III

สารบัญภาพ ..... V

สารบัญตาราง ..... VIII

บทที่ 1 บทนำ ..... 1

ความสำคัญและที่มาของการทำโครงงาน ..... 1

บทที่ 2 ทฤษฎีหรือหลักการ ..... 3

2.1. ทฤษฎีของ PETRI NETS ..... 3

*2.1.1. รู้จักกับ Petri Nets ..... 3*

*2.1.2. กฎเกี่ยวกับการส่งผ่าน Tokens ของ Transition หรือ Firing ..... 4*

*2.1.3. คำจำกัดความ ..... 5*

*2.1.4. แนะนำตัวอย่างโมเดลที่ใช้งาน ..... 6*

*2.1.5. วิเคราะห์พฤติกรรมของระบบจาก model ของ Petri nets ..... 10*

*2.1.6. Analysis Methods ..... 13*

*2.1.7. Characterizations of Liveness, Safeness, And Reachability ..... 20*

2.2. ความรู้เบื้องต้นเกี่ยวกับเดลไฟ ..... 25

*2.2.1. ส่วนประกอบของเดลไฟ ..... 25*

*2.2.2. ขั้นตอนในการสร้างคอมโปเนนต์ ..... 29*

*2.2.3. การเขียนคอมโปเนนท์ ..... 32*

บทที่ 3 ขั้นตอนการวางแผนพัฒนาและออกแบบ ..... 40

3.1. ขั้นตอนการวางแผนและพัฒนา ..... 40

3.2. การออกแบบ ..... 42

*3.2.1. Class Diagram ..... 42*

*3.2.2. Class Specification ..... 44*

*3.2.3. Form Diagram ..... 56*

*3.2.4. Queuing Theory ..... 57*

3.3. เกี่ยวกับโปรแกรมและการใช้งาน ..... 60
*3.3.1. ส่วนประกอบของโปรแกรม ..... 60*
*3.3.2 รูปแบบไฟล์ที่ใช้ simulate ..... 69*
*3.3.3. การใช้งานของแอพพลิเคชั่น ..... 71*
บทที่ 4 ประเมินผลการทำงาน ..... 75
4.1. ข้อกำหนดการประเมิน ..... 75
4.2. ผลที่ได้จากการประเมินมีดังนี้ ..... 76
บทที่ 5 บทวิจารณ์และสรุป ..... 79
5.1. สรุปผลงาน ..... 79
5.2. แนวทางการพัฒนาต่อ ..... 79
5.3. แนวทางการประยุกต์ใช้ ..... 80
5.4. สรุปภาพรวมของโครงการ ความสำเร็จ ..... 80
ภาคผนวก ก. EXAMPLES OF INDUSTRIAL USE OF CP-NETS AND DESIGN/CPN ..... 81
ภาคผนวก ข. คู่มือการติดตั้งโปรแกรม ..... 84
กิตติกรรมประกาศ ..... 90
หนังสืออ้างอิง ..... 91

# สารบัญภาพ


รูป 1-1 Petri Nets ที่แสดงการทำ parallel activities 1

รูป 1-2 Petri Nets ที่แสดง language $L= a_n b_n c_n$ โดย $n >= 0$ 1 1

รูป 2-1 petri net แสดง state diagram ของ vending machine 6

รูป 2-2 Petri net แสดง parallel activities 6

รูป 2-3 Petri net แสดง dataflow computation 7

รูป 2-4 simplified model ของ Communication protocol 7

รูป 2-5 Petri net แสดง readers-writers sysytem 8

รูป2-6 Context-sensitive language 8

รูป 2-7 Petri net แสดง producers-consumers system 9

รูป 2-8 finite-capacity net 10

รูป 2-9 reachability graph 10

รูปที่ 2-10 เป็นnon- Reversibility 11

รูป 2-11 เป็น Reversibility 11

รูป 2-12 Transition $t_0, t_1, t_2, t_3$ เป็น ($L_0$-live), $L_1$-live, $L_2$-live และ L_live 14

รูป2-13 The coverability tree of the net 14

รูป 2-14 The coverability graph of the net 15

รูป 2-15 A live Petri Net 15

รูป2-16 A nonlive Petri Net 16

รูป2-17 The coverability tree for both Petri net (a),(b) 16

รูป 2-18 The coverability graph for the two Petri net (a),(b) 16

รูป 2-19 FSP 18

รูป 2-20 FST 18

รูป 2-21 FPP 18

รูป 2-22 FPT 19

รูป 2-23 ESP 19

รูป 2-24 EST 19

รูป 2-25 สัญลักษณ์สำหรับอินพุท Place,transition และเอาท์พุท place,transition 20

รูป 2-26 Key structures characterizing subclass or Petri nets and their Venn diagram 21

รูป2-27 Explanation as to why a Live and Safe Petri Net Cannot Have Source or Sink Places and Transitions 22

รูป 2-28 แสดงสปีดบาร์ 25

รูป 2-29 แสดงคอมโปเนนต์แพลเลต 25
รูป 2-30 แสดงฟอร์ม 26
รูป 2-31 แสดงหน้าต่างโค้ดเอดิเตอร์ 27
รูป 2-32 แสดงออบเจ็กต์อินสเปคเตอร์(Property) 27
รูป 2-33 แสดงออบเจ็กต์อินสเปคเตอร์ (Event) 28
รูป 2-34 ฟอร์ม New Items 29
รูป 2-35 ฟอร์ม Install Component 30
รูป 2-36 ฟอร์ม Install Package 31
รูป 3.1 Gantt Chart อธิบายขั้นตอนการวางแผนการทำงาน 41
รูป 3-2 class diagram ของฟอร์มคอนเทนเนอร์ (Form2) 43
รูป 3-3 class diagram แสดงฟอร์มต่างๆ ที่มีอยู่ 43
รูป 3-4 แสดง Form Diagram 56
รูป 3-5 แสดงโมเดลที่ใช้อธิบายการแสดงการทำงาน(simulation) 57
รูป 3-6 flow chart แสดงการ ซิมมูเลชั่น 59
รูป 3-7 ฟอร์มหลัก 60
รูป 3-8 ฟอร์มที่ใช้ออกแบบโมเดล 61
รูป 3-9 ฟอร์มที่ใช้แสดงผลของ Queue 62
รูป 3-10 ฟอร์มที่ใช้แสดงผลของ Calendar 62
รูป 3-11 เป็นแถบที่ใช้แสดงสถานะขณะที่ทำการ run อยู่โดยมีรายละเอียดดังนี้ 63
รูป 3-12 ฟอร์มของ Place 64
รูป 3-13 ฟอร์มของ Transition 64
รูป 3-14 ฟอร์มของArc 65
รูป 3-15 ฟอร์มของText 66
รูป 3-16 ฟอร์มของ Log file 66
รูป 3-17 ฟอร์มที่แสดงผลของ net ที่สร้างขึ้นมา 67
รูป 3-18 Confirm Box 67
รูป 3-19 option form 68
รูป 3-20 Edit Text ของไฟล์ .sim 72
รูป 3-21 Log File Form 74
รูปภาคผนวก ข-1 กด "start" และ เลือก "run" 84
รูปภาคผนวก ข-2 พิมพ์ "a:\ksetup.exe" ใน Edit 84
รูปภาคผนวก ข-3 ฟอร์ม welcome 85
รูปภาคผนวก ข-4 ฟอร์ม setup type 86
รูปภาคผนวก ข-5 ฟอร์ม Select components to install 87

*รูปภาคผนวก ข-6 ฟอร์ม Progress* 88
*รูปภาคผนวก ข-7 ฟอร์ม Finish* 88
*รูปภาคผนวก ข-8 เรียกแอปพลิเคชั่นได้จากเดสท็อปหรือโฟลเดอร์* 89

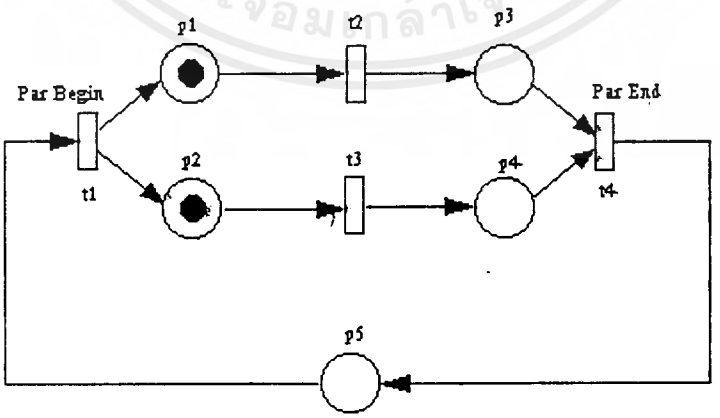
# บทที่ 1
# บทนำ

**ความสำคัญและที่มาของการทำโครงงาน**

ในการที่จะศึกษาระบบหนึ่ง ๆ เพื่อที่จะอธิบายและศึกษากระบวนการทำงานของระบบ เช่น การแจกจ่าย(distributed) resource ต่างๆ, การเกิด deadlock เป็นต้น การอธิบายด้วยโมเดลเป็นสิ่งที่จะช่วยให้เราเห็นภาพและอธิบายได้ง่ายและชัดเจนยิ่งขึ้น ซึ่งโมเดลที่จะใช้จำลองระบบนั้นได้มีการศึกษาและวิจัยออกแบบขึ้นมาหลายชนิด เช่น state machine, Finite automation หรือ Petri Nets เป็นต้น โดยในโครงงานนี้ เราได้สนใจที่จะศึกษาโมเดลที่มีชื่อว่า Petri Nets โดยสาเหตุที่เราเลือก Petri Nets นั้นเพราะความสามารถของโมเดลที่ได้รับการออกแบบมาให้สามารถอธิบายระบบต่าง ๆ ได้หลาย ๆ แบบ หรือการมีความยืดหยุ่นในตัวโมเดลเอง ดังจะขอยกตัวอย่างเปรียบเทียบระหว่างโมเดลแบบ State Machine ที่เป็นโมเดลพื้นฐานที่เรารู้จักกันโดยทั่วไป กับ โมเดลแบบ Petri Nets อย่างคร่าว ๆ ดังนี้

ในรูปแบบโมเดลของ State Machine จะประกอบไปด้วย state (ใช้สัญลักษณ์ วงกลม) แทนเงื่อนไขต่าง ๆ ของระบบ, arc (เป็นเส้นทางที่จะนำไปสู่อีก state หลังจากทำเหตุการณ์ หรือ event ที่ระบุไว้ใน arc) ส่วนในรูปแบบโมเดลของ Petri Nets จะประกอบไปด้วย place(ใช้สัญลักษณ์ วงกลม) แทนแต่ละเงื่อนไขของระบบ, transition (ใช้สัญลักษณ์ สี่เหลี่ยม) แทนแต่ละเหตุการณ์ (event) ของระบบ arc (เป็นเส้นทางที่จะนำไปสู่เหตุการณ์ หรือ event) และ token (ใช้สัญลักษณ์ จุดสีดำใน place)เพื่อระบุความพร้อมของเงื่อนไข หรือ place โดยในแต่ละองค์ประกอบก็สามารถตั้งค่า พร็อพเพอร์ตี้สำหรับตัวมันได้ด้วย ด้วยรูปแบบที่ออกแบบมานี้ทำให้ Petri Nets มีความสามารถเหนือกว่าโมเดลชนิดอื่นเช่น

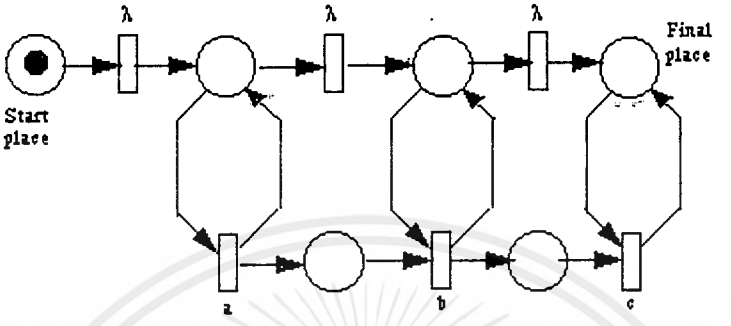
ระบบที่โมเดลอื่นไม่สามารถจำลองขึ้นมาได้ เช่น Parallel Activity หรือ Concurrency ที่มีการทำงานพร้อม ๆ กันไป แต่ Petri Nets สามารถใช้จำลองและอธิบายระบบนี้ได้โดยมี token เป็นตัวบอกการเกิดเงื่อนไขสำหรับเหตุการณ์ ดังรูป




*รูป 1-1 Petri Nets ที่แสดงการทำ parallel activities*

ความยืดหยุ่นของโมเดลที่ถูกออกแบบมา เช่นใน formal language

สำหรับ ระบบที่มี language L= $a_2b_2c_2$ เมื่อใช้ state machine สามารถออกแบบได้ดังรูป แต่ถ้าต้องการเปลี่ยนมาใช้อธิบาย language ใหม่ ที่คล้ายกัน คือ abc ก็ต้องออกแบบใหม่โดยเมื่อค่าเลขที่ยกกำลังมีค่าสูงขึ้นโมเดลจะมีขนาดใหญ่ขึ้นด้วย สำหรับ Petri Nets สามารถออกแบบได้สำหรับทุก language ที่ L= $a_nb_nc_n$ โดย n >= 0 โดยเพียงแต่กำหนด ค่า property ให้กับแต่ละ องค์ประกอบดังรูป




*รูป 1-2 Petri Nets ที่แสดง language L= $a_nb_nc_n$ โดย n >= 0*

เมื่อต้องการให้ค่า n เป็น 2 ก็กำหนดให้ค่า capacity ของ P1 และ P2 เป็น 2

เมื่อต้องการให้ค่า n เป็น 3 ก็กำหนดให้ค่า capacity ของ P1 และ P2 เป็น 3

จะเห็นได้ว่าเป็นวิธีการที่ง่ายกว่าการที่เราจะต้องออกแบบใหม่ทั้งโมเดล

เหล่านี้เป็นเพียงตัวอย่างของประโยชน์สำหรับการใช้โมเดลของ Petri Net โดยยังมีประโยชน์อีกมากมายเหลือเกินที่จะยกมากล่าวในส่วนนี้ และจะได้แสดงให้ดูในส่วนของทฤษฎีต่อไป

นอกจากการใช้งานในด้านการจำลองระบบขึ้นมาแล้วยังสามารถใช้โมเดลที่ออกแบบนี้มาวิเคราะห์การทำงานโดยอาศัยหลักการทางทฤษฎีเพื่ออธิบายพฤติกรรมของระบบเช่น Reachability, Bounded และพิจารณาการเกิด Deadlock ในระบบได้เป็นต้น

ในโครงงานนี้มีความประสงค์ที่จะได้ศึกษาเกี่ยวกับ Petri Nets และสร้าง Tool ที่จะสามารถใช้งานเกี่ยวกับ Petri Nets โดยจะแบ่งหัวข้อการใช้งาน เป็น 4 หัวข้อหลักเกี่ยวกับการใช้งานได้แก่

1) ความสามารถในการออกแบบ
2) การแสดงการทำงานหรือซิมมูเลชั่น
3) การวิเคราะห์โมเดลที่ออกแบบ
4) และการบันทึกผลที่ได้จากการแสดงการทำงาน

ในการพัฒนานี้เรามีโปรแกรม Visual Simnet ซึ่งเป็นโปรแกรมที่ทำงานใน DOS ที่เราสามารถใช้สังเกตและศึกษาเทคนิคต่าง ๆ ในการทำงานได้

# บทที่ 2
# ทฤษฎีหรือหลักการ

## 2.1. ทฤษฎีของ Petri Nets

ทฤษฎีด้าน Petri Nets ที่ยกมากล่าวนี้ได้มาจากหลายทางทั้งการศึกษาเทคนิคการซิมมูเลชั่นจากโปรแกรม simnet, เวปไซต์ต่างๆที่มีการศึกษา และบทความของ MURATA ที่เราได้นำเนื้อหาบางส่วนของบทความนี้มาแปลเป็นทฤษฎีในส่วนนี้ด้วย

### 2.1.1. รู้จักกับ Petri Nets

จากที่ได้อธิบายไปแล้วว่า Petri Nets เป็นแบบจำลองทางด้านกราฟฟิกสำหรับระบบต่างๆ โดยการสร้างและใช้แบบจำลองแทน elements ต่างๆ ในระบบที่เราสนใจ ดังนั้นแบบจำลองนี้จะประกอบด้วยส่วนต่างๆ และการใช้งานดังนี้

1. Places ใช้สัญลักษณ์เป็นวงกลม

places สามารถแบ่งเป็น

1.1 input places แทนเงื่อนไขเริ่มต้น (Pre Condition) สำหรับเหตุการณ์หนึ่งๆ

1.2 output places แทนเงื่อนไขที่ได้(Post Condition) จากการทำเหตุการณ์หนึ่งๆ

ในแต่ละ Place จะมีพร็อพเพอร์ตี้ อยู่ได้แก่

- Capacity ระบุจำนวนสูงสุดของ Token ที่ Place สามารถเก็บไว้ได้
- Marking ระบุจำนวน Token ที่ Place มีอยู่

2. Transitions ใช้สัญลักษณ์เป็นสี่เหลี่ยมด้านขนานหรือจัตุรัสแทนเหตุการณ์ของระบบ

ในแต่ละ Transition จะมีพร็อพเพอร์ตี้อยู่ได้แก่

- Priority กำหนดความสำคัญของ Transition นั้นโดยค่า 0 จะมีค่าสูงสุด
- Probability กำหนดความน่าจะเป็นของการเลือก Transition นั้นโดยระบบ

3. Token ใช้สัญสัาษณ์เป็นจุดสีดำซึ่งจะปรากฏเป็นจุดสีดำเราจะแทนเงื่อนไขที่จะพิจารณาด้วยสัญลักษณ์ของ Token ดังนั้น token จะอยู่ใน Place ซึ่งจะหมายถึงมีเงื่อนไขตามจำนวนของ tokens เข้ามาในเหตุการณ์ (Transition)

4. Arcs ใช้สัญลักษณ์เป็นเส้นที่เชื่อมต่อระหว่าง Transitions และ Places ซึ่งจะบอกทิศทางของเงื่อนไขกับเหตุการณ์

Arcs แบ่งเป็น 2 ประเภทตามทิศทาง

4.1 PreArcs มีทิศทางจาก Place ไป transition

มี Arc Weight w(p,t) ระบุจำนวน Token ที่ต้องการสำหรับการผ่านในแต่ละครั้ง

4.2 PostArcs มีทิศทางจาก Transition ไป Place

มี Arc Weight w(t,p) ระบุจำนวน Token ที่จะเกิดขึ้นสำหรับการผ่าน Arc นี้ในแต่ละครั้ง

**2.1.2. กฎเกี่ยวกับการส่งผ่าน Tokens ของ Transition หรือ Firing**

หมายความว่าการเคลื่อนที่และการเปลี่ยนแปลงของ Tokens ที่อยู่ใน Places แรกจะไปสู่อีก Places อื่น โดยผ่าน Transition เทียบกับระบบได้ว่าการมีเงื่อนไขที่พร้อม (จำนวน token ใน places แรก) สำหรับการเกิดเหตุการณ์ (transition) และผลที่ได้จากการเกิดเหตุการณ์จะเป็นอย่างไร (จำนวน token ที่เกิดขึ้นใน place อื่น) นั้นมีเงื่อนไขอย่างไร

1. transition t จะบอกได้ว่า enable (พร้อมที่จะทำ firing) เมื่อทุก input place p ของ t มีจำนวน tokens อย่างน้อย w(p,t) เท่ากับ weight ของ arcs จาก p ไป t (Pre Arcs)
2. transition ที่ enable จะทำ firing หรือไม่ก็ได้
3. การ fire สำหรับ enable transition t จะย้าย token จำนวน w(p,t) สำหรับแต่ละ input place p ของ t และจะสร้าง tokens จำนวน w(t,p) ให้กับแต่ละ output place p ของ t ซึ่ง w(t,p) คือ weight ของแต่ละ arc จาก t ไป p

ทฤษฎีข้างต้นเป็นทฤษฎีพื้นฐานของ Petri Nets เท่านั้นที่เรียกว่า Place-Transition Petri Nets ทฤษฎีต่อมาเป็นเรื่องที่มีเวลาเข้ามาเกี่ยวข้องด้วย โดยจะเป็นพร็อพเพอร์ตี้ตัวหนึ่งของ Transition ค่า time นี้จะบอกว่าเมื่อ transition นั้น enable นั้น ขึ้นมาแล้ว จะใช้เวลาเท่าใดในการ fire ออกไป ซึ่งจะมีผลต่อลำดับการ fire ของ transition และจะอธิบายต่อไปในส่วนของ Queuing Theory เรียกระดับของ Petri Nets ในขั้นนี้ว่า Time Petri Nets ต่อมามีการขยายความสามารถของ Petri Nets ที่จะสามารถแยกจัดการกับ Token ที่แตกต่างกันโดยกำหนดให้แต่ละ Token มีพร็อพเพอร์ตี้ color และ Arc มีพร็อพเพอร์ตี้ที่จะเลือก(ในกรณีของ PreArc) หรือสร้าง (ในกรณีของ PostArc) Tokens ให้ได้ตาม Color ตามที่ระบุได้ ทฤษฎีนี้เรียกว่า Colored Petri Nets : CPN

### 2.1.3. คำจำกัดความ

Petri Net อธิบายด้วย PN =(P,T,F,W,$M_0$) ซึ่งแต่ละส่วนมีความหมายดังนี้

1. P=$\{p_1,p_2,\ldots,p_m\}$ เป็นเซตที่มีอยู่ของ places
2. T=$\{t_1,t_2,\ldots,t_m\}$ เป็นเซตที่อยู่ของ transitions
3. F: (PXT) U (TXP) เป็นเซตของ arcs (ความสัมพันธ์ระหว่าง transition กับ place)
4. W:F? {1,2,3,...} เป็นเซตของ weight ของ arcs แต่ละตัว
5. $M_0$= {1,2,3,...} เป็นจำนวน token ของ place แต่ละตัวในเซต P เมื่อเริ่มต้น (initial marking) โครงสร้างของ Petri nets แบบ N=(P,T,F,W) ที่ไม่มี initial marking จะใช้ N

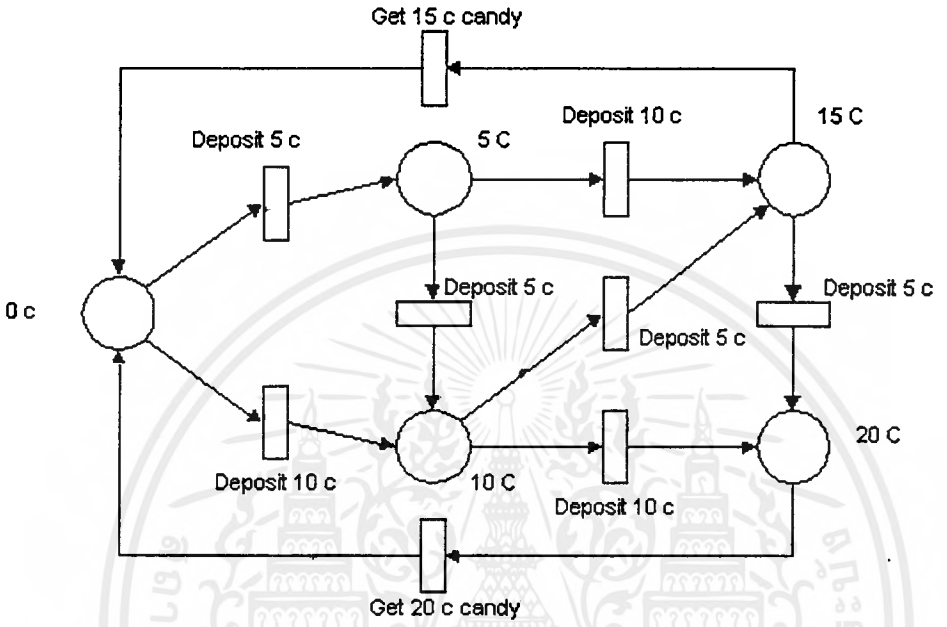
Petri nets ที่มี initial marking ด้วยจะใช้ (N,$M_0$)

- source transition : ไม่มี inputplaces transitions นั้นจะสามารถ generate input condition ขึ้นมาเองให้เกิดการ fire ได้
- Sink transition : ไม่มี input places transitions นั้นจะสามารถ generate output condition ขึ้นมาเองให้เกิดการ fire ได้
- Self-loop : Place p ที่เป็นทั้ง input และ output ของ transition
- Pure net : ไม่มี Self -loop
- Ordinary net: arc weight ของ Arcs ทุกตัวเป็น 1 ซึ่ง Arcs ที่ไม่เขียน weight กำกับจะหมายความว่ามี weight เป็น1
- Infinite capacity net :แต่ละ place มี tokens ไม่จำกัดจำนวน
- Finite capacity met : แต่ละ place มี tokens สูงสุดจำกัดจำนวน
- Extended Petri nets : Petri nets ที่มี inhibitor arcs

### 2.1.4. แนะนำตัวอย่างโมเดลที่ใช้งาน

**1. Finite-State Machines**

เป็นเพียงส่วนหนึ่งของ petri net ดังนั้น petri net จึงสามารถอธิบาย model ของ state machines ได้เช่น ตัวอย่างของ vending machine แสดงขั้นตอนการวางเงินเพื่อซื้อลูกกวาดราคา 15 หรือ 20 โดยมีการวางเงินทีละ 5 หรือ 10 เท่านั้น

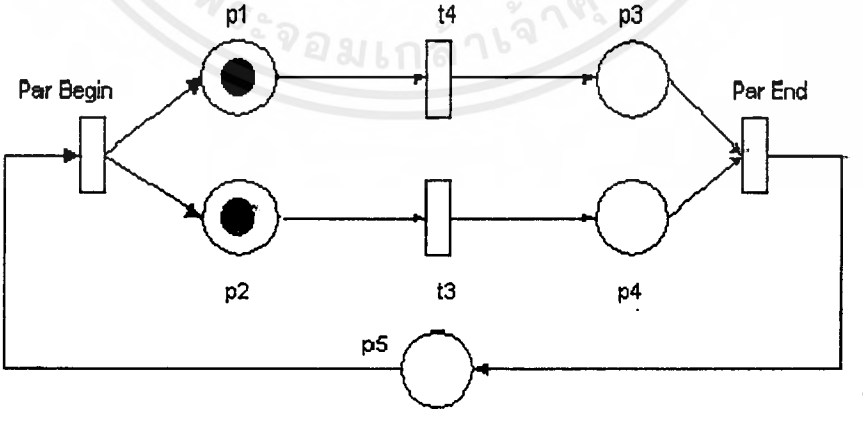



*รูปที่ 2-1 petri net แสดง state diagram ของ vending machine*

**2. Parallel Activities Or Concurrency**

เป็นการทำงานที่ทำพร้อมกันหรืองานที่ในลักษณะขนานกันไป

ตัวอย่างเช่นดังรูป

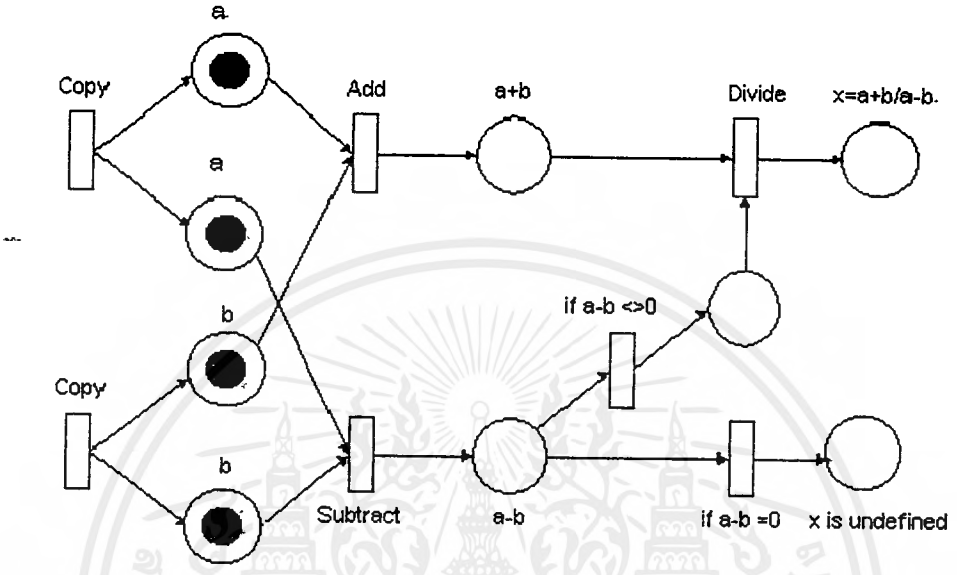



*รูป 2-2 Petri net แสดง parallel activities*

จากรูป t1 จะ generate token ออกมา 2 อันอยู่ที่ p1 และ p2 ซึ่ง t2 และ t3 สามารถเป็น concurrent กัน ซึ่งก็คือ transition ทำการ fire โดยไม่ขึ้นตรงต่อกัน t2 อาจจะ fire ก่อนหรือหลังหรือพร้อมกันกับ t3 ก็ได้

### 3. Dataflow Computation

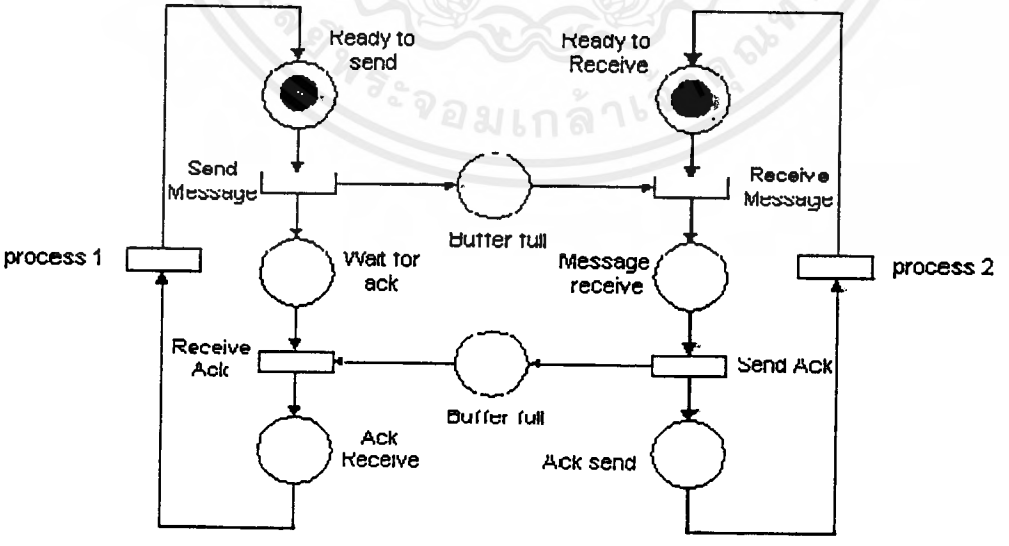
รูป Dataflow Computation แสดงการคำนวณสำหรับ x =(a+b) / (a-b) ซึ่ง Petri nets ไม่เพียงแต่แสดงการไหลของการควบคุม แต่ยังแสดงการไหลของข้อมูล โดย token จะระบุค่าของข้อมูลปัจจุบันที่เป็นการคำนวณอยู่




*รูป 2-3 Petri net แสดง dataflow computation*

### 4. Communication Protocols

รูปแสดงการจัดการรูปแบบของการสื่อสาร

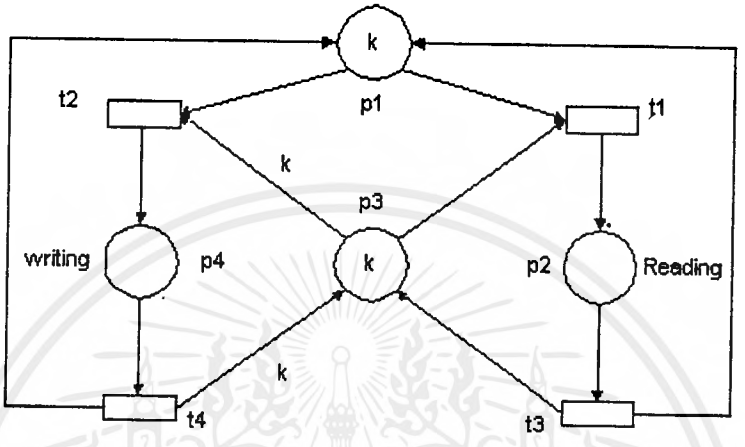



*รูป 2-4 simplified model ของ Communication protocol*

**5. Synchronization Protocols**

แสดงการจัดลำดับการควบคุม รวมถึงการกีดกันและกัน หรือที่เรียกว่า mutual exclusion เช่น ตัวอย่างที่ใช้ในปัญหา reader-writer problem มีกติกาว่ามี resource สำหรับการอ่านและเขียนเพิ่มได้อีก และเมื่อทำการอ่านอยู่จะอ่านเพิ่มได้แต่ไม่สามารถเขียนได้

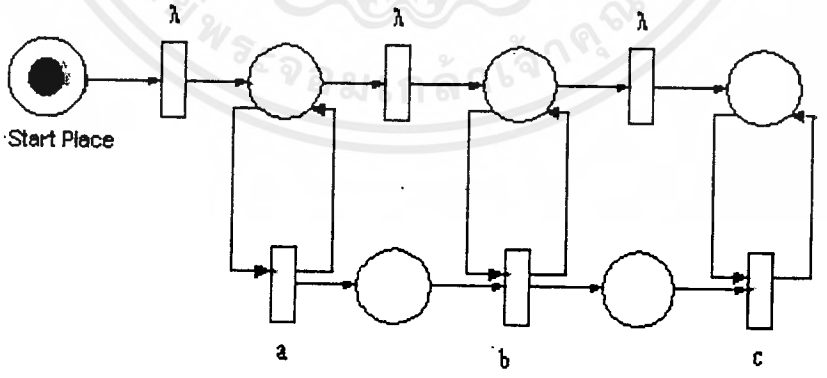
เราสามารถสร้างแบบจำลองเพื่อดูการทำงานได้ดังรูป




*รูป 2-5 Petri net แสดง readers-writers sysytem*

**6. Formal Language**

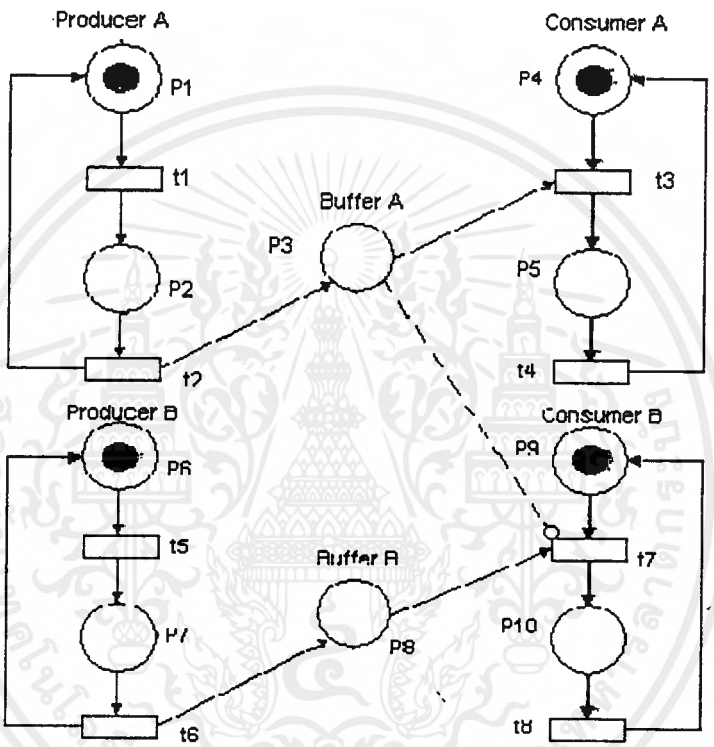
รูปเป็นตัวอย่างของ context-sensitive language (MO) ={$a^n b^n c^n$} เรียก labeled petri nets ลำดับของการ fire จะ generate string ออกมาตามลำดับ fire เมื่อผ่าน transition ใดก็จะ generate string ที่มี label ของ transition นั้น เช่น a,b,c เป็นต้น




*รูป2-6 Context-sensitive language*

**7. Producers - consumers System with Priority**

คล้ายกับตัวอย่างข้างต้นแต่เป็นรูปแบบที่จะมีการเช็คความสำคัญด้วย ดังรูป ซึ่ง กลุ่ม A มีผลต่อการกำหนดการทำงานในกลุ่ม B ตามเงื่อนไขว่า ในส่วนของ A ผู้บริโภคจะได้รับเมื่อ buffer A มี items(tokens) แต่ในกลุ่ม B ผู้บริโภคจะได้รับเมื่อ buffer A ว่าง และ buffer B มี items(tokens) ซึ่งระบบลักษณะนี้จะต้องมีการใช้ inhibitor arc (เมื่อมี property เป็น = มี weight เป็น 0) จากรูป t 7 จะไม่สามารถทำการ fire ได้เมื่อมี tokens ใน buffer A หรือ P3 ดังนั้นถ้า buffer A มี token เท่ากับ 0 และ Buffer B กับ Consumers B มี token อยู่ t7 จึงสามารถ enabled ได้




*รูป 2-7 Petri net แสดง producers-consumers system*

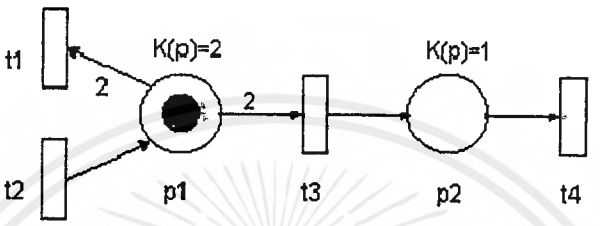
### 2.1.5. วิเคราะห์พฤติกรรมของระบบจาก model ของ Petri nets

หลังจากมี model ของ ระบบแล้วเราสามารถบอกอะไรเกี่ยวกับคุณสมบัติของระบบได้บ้างนั้นเป็นเรื่องที่สำคัญที่เราจะศึกษา petri nets เราจะบอกพฤติกรรมของระบบได้แก่

**A. Reachability**

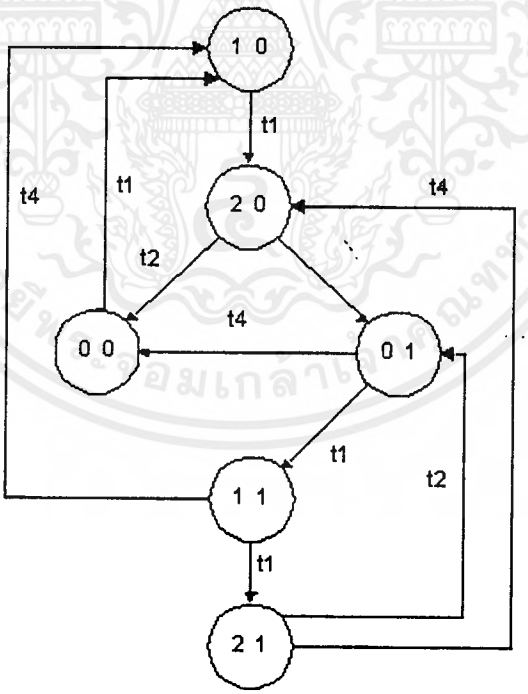
เพื่อศึกษาพฤติกรรมการเคลื่อนที่ของระบบ การ firing จะทำให้มีการเปลี่ยนตำแหน่งของ tokens เกิดขึ้น (marking เปลี่ยน) ผลที่เกิดจากการ firing จะทำให้เกิดลำดับของ marking โดย Marking $M_n$ จะ reachable จาก $M_0$ ก็ต่อเมื่อการ firing ได้ ทำให้ เกิดการเปลี่ยนรูปจาก $M_0$ เป็น $M_n$

ดังเช่นตัวอย่าง




*รูป 2-8 finite-capacity net*

เราสามารถเขียน reachability graph ได้ดังนี้




*รูป 2-9 reachability graph*

**B. Boundedness**

ระบบจะเป็น k-bounded เมื่อk คือ ค่าสูงสุดของจำนวน token ใน place ของทุกการ marking ที่อยู่ใน $R(M_0)$ ระบบที่จะเรียกได้ว่า safe เมื่อเป็น 1- bounded

**C. Liveness**

หลักการของ liveness จะเกี่ยวกับการมี deadlock ในระบบ โดยจะบอกว่า petri net นั้น live ถ้าไม่มีกรณีใดที่ mark จะหยุดการ reach จาก $M_0$ (สรุปง่ายก็คือ ไม่สามารถทำการ firing ต่อไปได้อีก) Petri net ที่ live นี้จะไม่มี deadlock

ระดับของ liveness นั้นจะพิจารณาจาก transition โดยจะพิจารณาว่า transition นั้นเป็น

LO- live (dead) ถ้า transition ทำให้ไม่มีการ fire อีกในทุกๆลำดับการ fire

L1-live (ยังเกิดการ fire ได้) ถ้า transition นั้นยังสามารถ fire อย่างน้อย 1 ครั้งในลำดับของการ fire

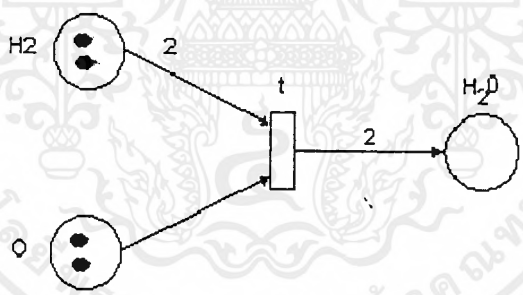
L2- live สำหรับจำนวนเต็มบวก k, ถ้า transition สามารถ fire อย่างน้อย k ครั้งในลำดับการ fire

L3-live ถ้า transition t สามารถปรากฏต่อได้อย่างไม่จำกัดในลำดับของการ firing

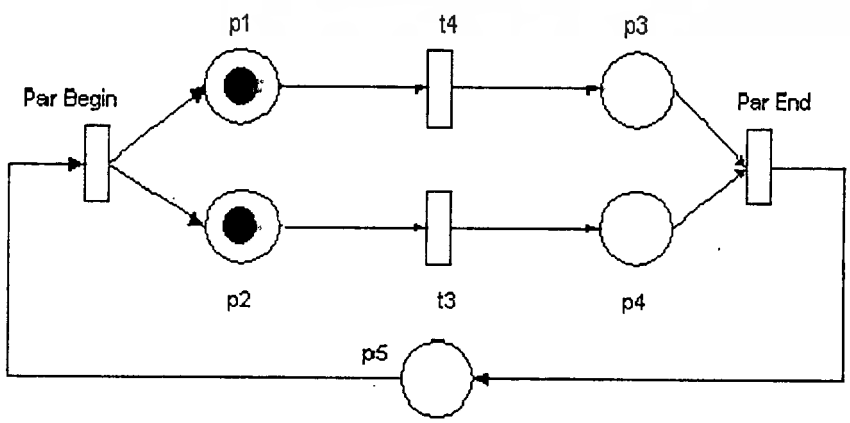
L4-live หรือ live ถ้าเป็น L1-live สำหรับทุก marking M ใน $R(M_0)$

**D. Reversibility and Home State**

บอกว่าเมื่อระบบทำการ firing ไปแล้วระบบสามารถย้อนกลับมามี marking ได้เหมือนเดิมอีกหรือไม่ เช่น รูปที่ 1 แสดงการเกิด $H_2O$เป็น




*รูปที่ 2-10 เป็นnon- Reversibility*




*รูป 2-11 เป็น Reversibility*

**E. Coverability**

บอกว่า Marking M จะ coverable ถ้ามี Marking M' (Marking อื่นๆ)ใน $R(M_0)$ ที่ $M'(p) >= M(p)$ สำหรับแต่ละ p ใน net ให้ M เป็น Minimum Marking ที่จำเป็นเพื่อที่จะ enable transition t โดย

t จะ dead (L0-live) เมื่อ M ไม่ coverable

t จะเป็น L1-live เมื่อ M เป็น coverable

**F. Persistence**

Petri nets$(N, M_0)$ จะกล่าวว่าเป็น persistence ถ้าสำหรับทุกๆ สอง transition ที่ enabled อยู่การ firing ของ transition หนึ่งจะต้องไม่ทำให้อีก transition ที่ enable นั้น disable transition ใน persistence net การ enabled ในแต่ละครั้งจะยังคง enabled จนกระทั่งทำการ fire ทุก marked graphs(แต่ละ place มี 1 PreArc และ 1 PostArc) เป็น persistence แต่ไม่ทุก persistence net ที่เป็น marked graphs.

**G. Synchronic distance**

Synchronic Distance ระหว่าง 2 transition t1 และ t2 ใน petri net คือ

$$d12 = \max_{\sigma}|\sigma(t1) - \sigma(t2)|$$

โดย $\sigma$ คือ ลำดับของการ firing

$\sigma(t_i)$ คือจำนวนครั้งของ transition $t_i$ ที่ fire ใน $\sigma$ สรุปคือจำนวนครั้งสูงสุดที่เกิดขึ้นได้จากการ firing ของ transition t1 ไป t2

**H. Fairness**

แบ่งเป็น 2 concept คือ bounded-fairness กับ unconditional fairness

- transition t1 กับ t2 จะกล่าวว่าเป็นความสัมพันธ์แบบ bounded-fair ถ้าจำนวนครั้งที่มากที่สุดที่ transition หนึ่งสามารถ firing ขณะที่อีก transition ที่ไม่ได้ firing นั้น bounded. Petri net$(N, M_0)$จะเป็น Bounded-fair เมื่อทุกคู่ของ transition ใน net เป็น Bounded-fair
- firing sequence $\sigma$ จะเป็น unconditionally fair ก็ต่อเมื่อทุก transition ใน net ปรากฏใน $\sigma$ Petri net $(N,M_0)$ จะเป็น unconditionally fair net ถ้าทุกการ firing sequence $\sigma$ จาก M ใน $R(M_0)$ เป็น unconditionally fair

**2.1.6. Analysis Methods**

การ Analysis Petri net แบ่งออกได้เป็น 3 group:

1. The Coverability Tree Method (reachability)
2. The Matrix - equation approach
3. Reduction or decomposition techniques

**A. The Coverability Tree**

มีขั้นตอนในการสร้างโดยใช้ Algorithm ดังนี้

Step 1) Label the initial marking $M_0$ as thee root and tag it "new"

Step 2) While "new" markings exist, do the following:

Step 2.1) Select a new marking M.

Step 2.2) If M is identical to a marking on the path from the root to M then tag M " old" and go to another new marking.

Step 2.3) If no transitions are enabled at M, tag M "dead end"

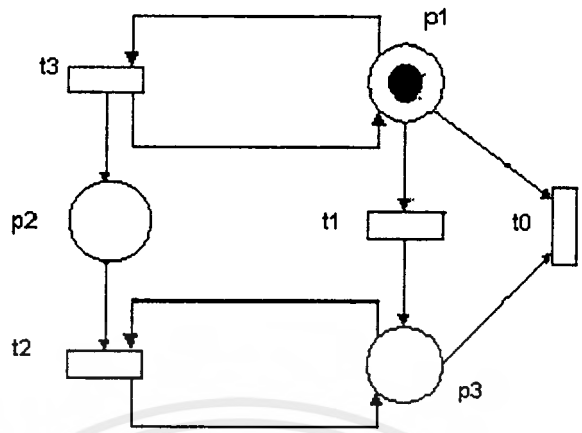
Step 2.4) While there exist enabled transitions at M, do the following for each enabled transition t at M:

Step 2.4.1) Obtain the marking M' that results from firing t at M.

Step 2.4.2) On the path from the root to M if there exists marking M'' such that $M'(p) >= M''(p)$ for each place p and $M' \neq M''$, i.e. M'' is coverable,then replace M'(p) by $\omega$ for each p such that $M'(p) > M''(p)$.

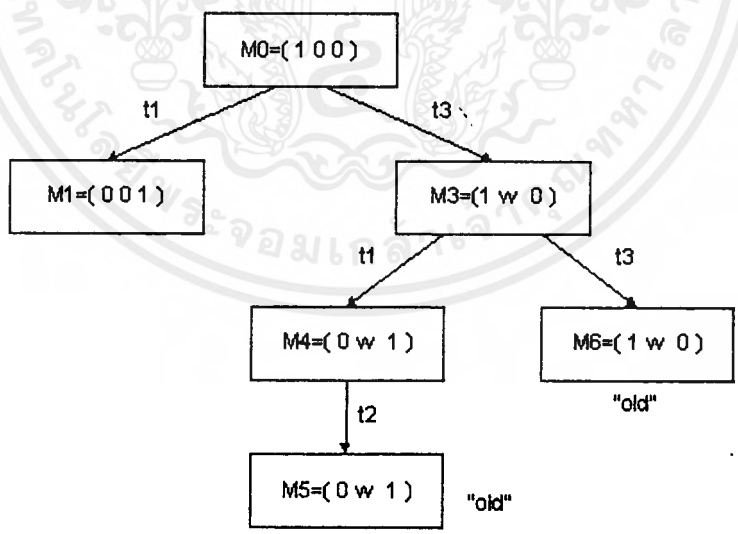
Step 2.4.3) Introduce M' as a node, draw an arc with label t from M to M', and tag M' "new."

Example:

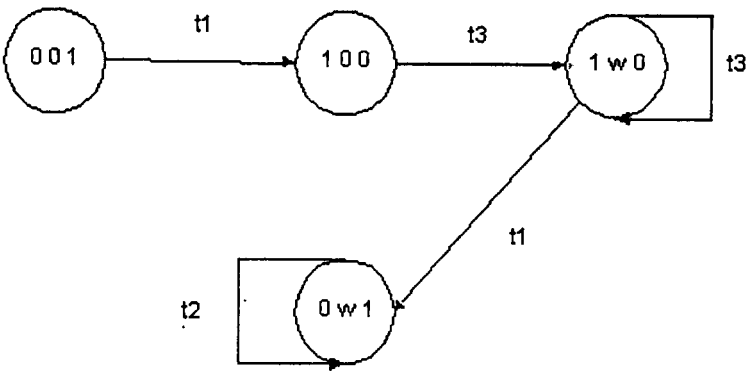



*รูป 2-12 Transition $t_0, t_1, t_2, t_3$ เป็น ($L_0$-live), $L_1$-live, $L_2$-live และ L_live*

จากรูป initial marking $M_0$ =(1 0 0),โดย transition t1,t3 enabled, ถ้า firing t1 จะได้ $M_1$=(0 0 1),แล้วก็ dead-end คือไม่มี transition ไหนที่จะ enabled ได้ที่ $M_1$ แต่ถ้า firing t3 จะได้ $M'_3$=(1 1 0),ถ้าเทียบกับ $M_0$แล้วจะได้ว่า $M_3$=(1 ω?0) และ t1,t3 จะ enabled อีกครั้ง ถ้าเรา firing t1 จาก $M_3$ไป $M_4$ =(0 ω1) แล้ว t2 จะสามารถ fired ได้ แต่เมื่อ fire แล้วได้ $M_5 = M_4$ ซึ่งได้ค่าเดิม แต่ถ้าเราfiring t3 จาก $M_3$ ไป $M_6 = M_3$ เราสามารถใช้ coverability tree แสดงได้ดังรูป




*รูป2-13 The coverability tree of the net*

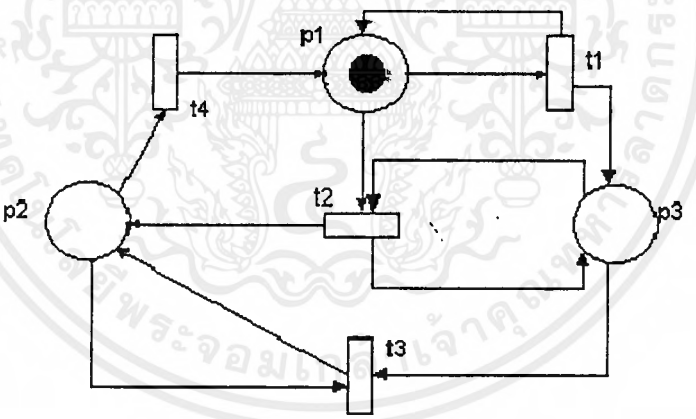



*รูป 2-14 The coverability graph of the net*

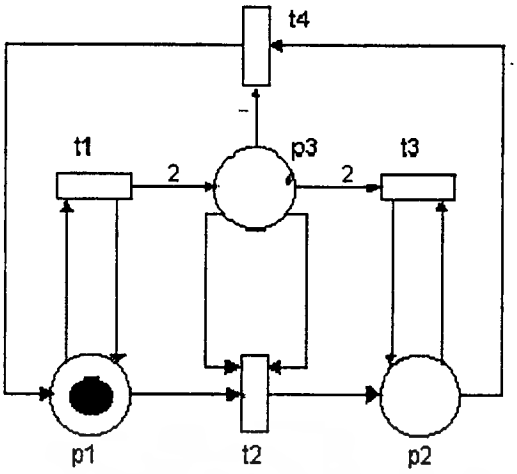
คุณสมบัติที่สามารถศึกษาโดยใช้ coverability tree สำหรับ Petri net(N, $M_0$)ได้มีดังนี้

1) net (N, $M_0$)จะbounded และ R($M_0$) เป็น finite ถ้า ω ไม่ปรากฏใน nodeใดๆใน Tree
2) net (N, $M_0$)จะsave ถ้า ใน node ใดๆปรากฏเฉพาะ 0 ,1 เท่านั้นใน Tree
3) A transition t จะ dead ถ้าไม่ปรากฏมี arc ต่อออกมาจาก label ในTree
4) ถ้า M เป็น reachable จาก $M_0$ เมื่อ M <= M'

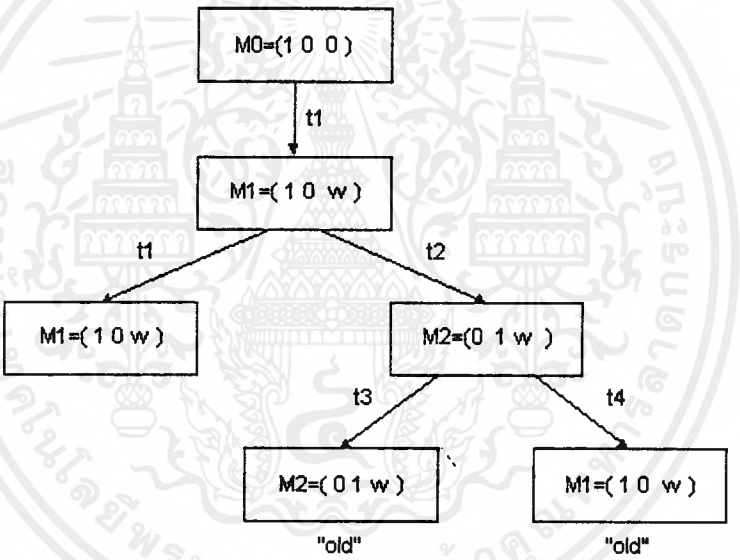
Example : ดังรูป



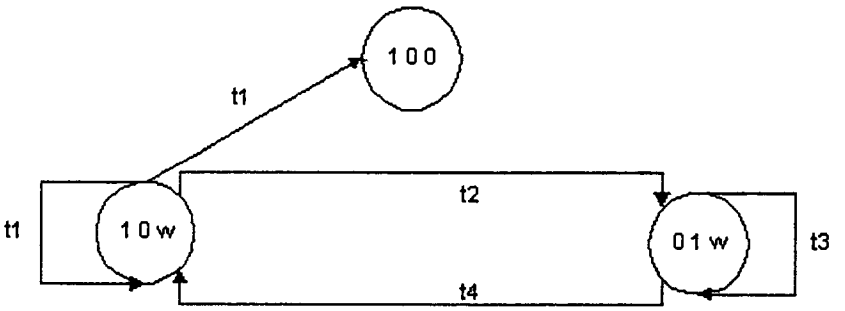

*รูป 2-15 A live Petri Net*




*รูป2-16 A nonlive Petri Net*




*รูป2-17 The coverability tree for both Petri net (a),(b)*




*รูป 2-18 The coverability graph for the two Petri net (a),(b)*

**B. Incidence Matrix and State Equation**

The dynamic behavior ของระบบต่าง สามารถที่จะอธิบายได้ด้วย differential equations หรือ algebric equations.

**- incidence matrix:**

สำหรับ Petri Net N ที่มี n transitions และ m places,

incidence matrix $A = [a_{ij}]_{n*m}$ โดย $a_{ij} = a^+_{ij} - a^-_{ij}$

$a^+_{ij} = w(i, j)$ คือ weight ของ arc จาก transition i ไปที่ output place j

$a^-_{ij} = w(j,i)$ คือ weight ของ arc จาก input place j ไป transition I

จาก transition rule จะเห็นว่า

$a^-_{ij}$ = remove token from place j

$a^+_{ij}$ = add token to place j

$a_{ij}$ = change token in place j

Transition I จะ enable ที่ marking M ถ้า

$a^-_{ij} <= M(j)$ ; j= 1,2,...m

**- State Equation :**

เราจะเขียน marking $M_k$ โดย m*1 column vector, the ith entry of $M_k$ แสดงถึงจำนวน token ใน place j หลังจากที่มีการ firing ครั้งที่ k แล้ว.The kth firing or control vector $u_k$ คือ n*1 column vector ของ n-1 0's และ non zero entry 1 ใน ith แสดงว่า transition i fires ที่การ firing ครั้งที่ k เช่น rowที่i ของ incidence metrix A แสดงการเปลี่ยน marking เนื่องจากผลที่เกิดจากการ firing transition i เราสามารถเขียน state equation ได้ดังนี้

$$M_k = M_{k-1} + A^T u_k \quad ; k=1,2,....$$

Necessary Reachability Condition: สมมติว่า $M_d$ เป็น marking ปลายทางที่ reachable จาก $M_0$ โดยผ่านการ firing เป็นลำดับ $\{ u_1, u_2,..., u_d\}$ เราจะเขียน state equation สำหรับ I = 1,2,...,d ได้ดังนี้

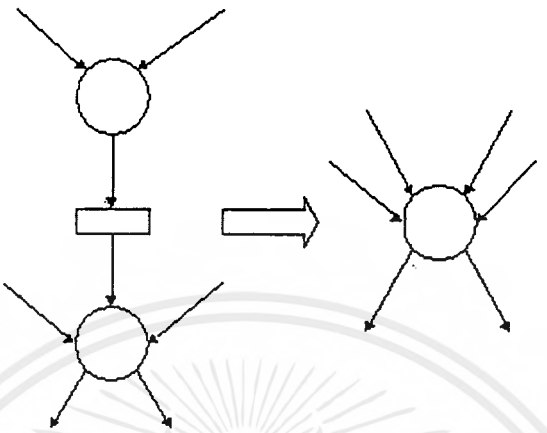
$$M_d = M_0 + A^T \sum^d_{k=1} u_k$$

เรากำหนดให้ $A^T x = \Delta(M)$ ; $\Delta(M)= M_d - M_0$ และ $x = \sum^d_{k=1} u_k$ โดย x คือ n*1 column vector ของ nonnegation integer เรียกว่า firing count vector, The ith entry ของ x แสดงจำนวนครั้งนี้ transition I ต้อง fire จาก $M_0$ เป็น $M_d$

**C. Simple Reduction Rules for Analysis**

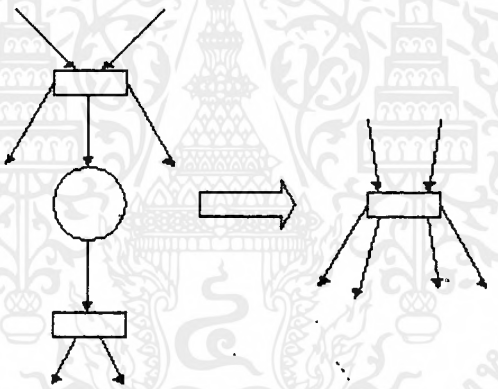
เป็นการลดรูป system เพื่อให้ดูง่ายขึ้นและง่ายต่อการ analysis ด้วยโดยมีรูปแบบการลดรูปดังนี้.

1) Fusion of Series Places (FSP)



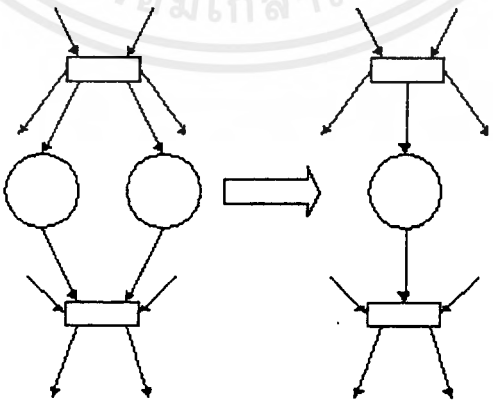
*รูป 2-19 FSP*

2) Fusion of Series Transitions (FST)



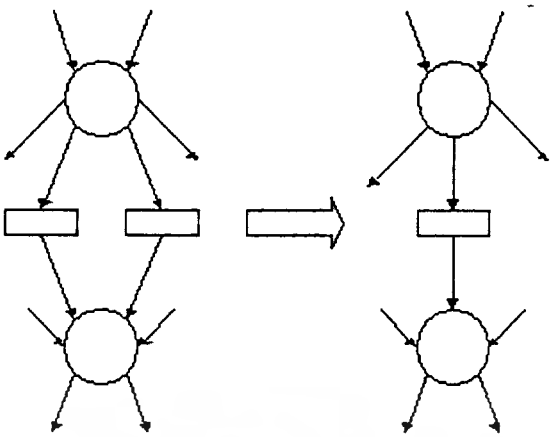
*รูป 2-20 FST*

3) Fusion of Parallel Places (FPP)



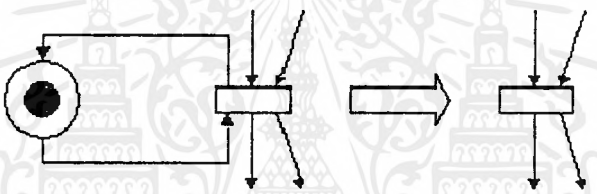
*รูป 2-21 FPP*

4) Fusion of Parallel Transitions(FPT)



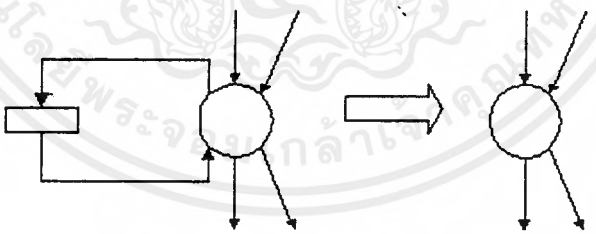
*รูป 2-22 FPT*

5) Elimination of Self-loop Places(ESP)



*รูป 2-23 ESP*

6) Elimination of Self-loop Transitions(EST)



*รูป 2-24 EST*

### 2.1.7. Characterizations of Liveness, Safeness, And Reachability

ในส่วนนี้เราจะพูดถึง subclass และ liveness,safeness,reachability ภายใน subclass ของ Petri nets.

**A. Subclass of Petri Nets**

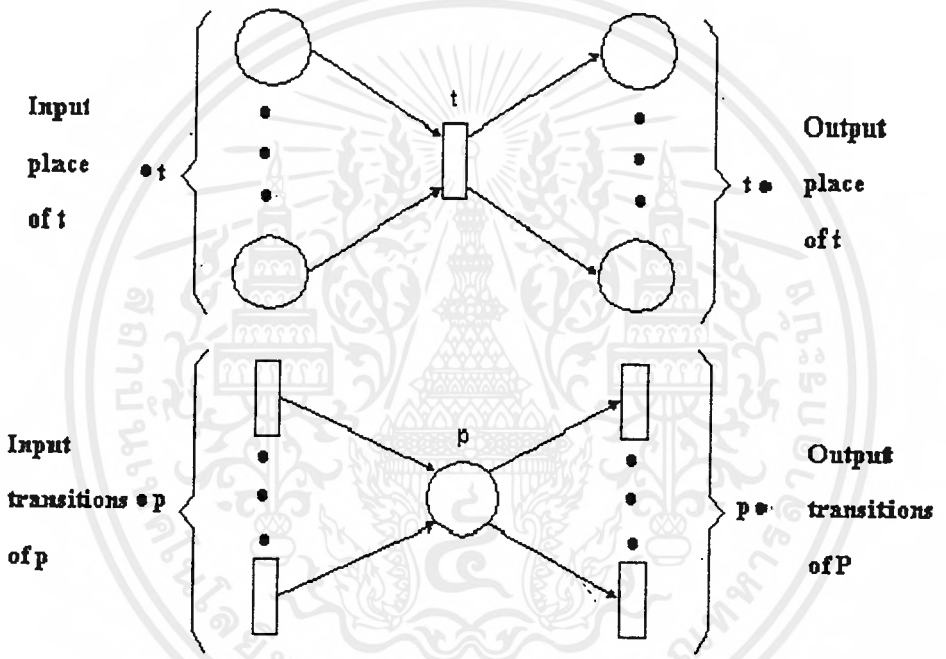
เราได้กำหนด symbols สำหรับ pre-set และ post-set (โดย F เป็นเซ็ตของทุก arc)

•t = { p| (p,t) ∈F } = เซ็ตของอินพุท places ของ transition t

t• = { p| (t,p) ∈ F } = เซ็ตของเอาท์พุท places ของ transition t

•p ={ t | (t,p) ∈ F } = เซ็ตของอินพุท transitions ของ place p

p• = { t | (p,t) ∈ F } = เซ็ตของเอาท์พุท transitions ของ place p




*รูป 2-25 สัญลักษณ์สำหรับอินพุท Place,transition และเอาท์พุท place,transition*

1) A state machine(SM) เป็น ordinary Petri net เมื่อในแต่ละ transition t มี 1 input place และ 1 output place,i.e,

$$|{}^\bullet t| = |t^\bullet| = 1 \quad \text{for all } t \in T$$

2) A marked graph(MG) เป็น ordinary Petri net เมื่อในแต่ละ place p มี 1 input transition และ 1 output transition

$$|{}^\bullet p| = |p^\bullet| = 1 \text{ for all } p \in P$$

3) A free-choice net(FC) เป็น ordinary Petri net เมื่อทุก arc จาก place is a unique outgoing arc หรือ a unique incoming arc to a transition, i.e.,

for all $p \in P$ , $|p^\bullet| <= 1$ or ${}^\bullet(p^\bullet) = \{p\}$; equivalently,

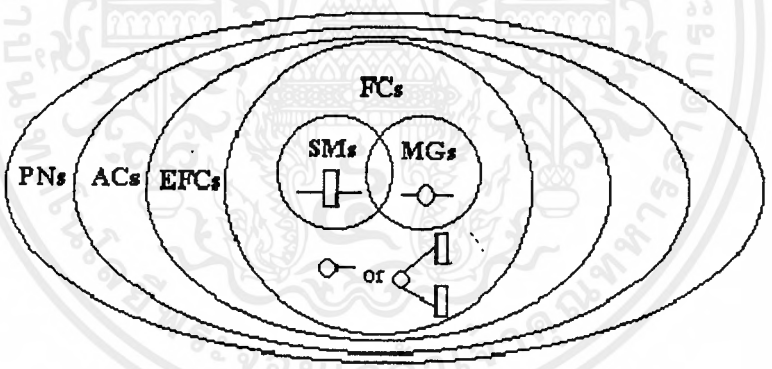
for all $p_1, p_2 \in P$ , $p_1{}^\bullet \cap p_2{}^\bullet \neq \varnothing \Rightarrow |p_1{}^\bullet| = |p_2{}^\bullet| = 1$

4) An extended free-choice net(EFC) เป็น ordinary Petri net เมื่อ

$$p_1{}^\bullet \cap p_2{}^\bullet \neq \varnothing \Rightarrow p_1{}^\bullet = p_2{}^\bullet \quad \text{for all } p_1, p_2 \in P$$

5) An asymmetric choice net (AC) (simple net) เป็น ordinary Petri net เมื่อ

$$p_1{}^\bullet \cap p_2{}^\bullet \neq \varnothing \Rightarrow p_1{}^\bullet \subseteq p_2{}^\bullet \text{ or } p_1{}^\bullet \supseteq p_2{}^\bullet \quad \text{for all } p_1, p_2 \in P$$

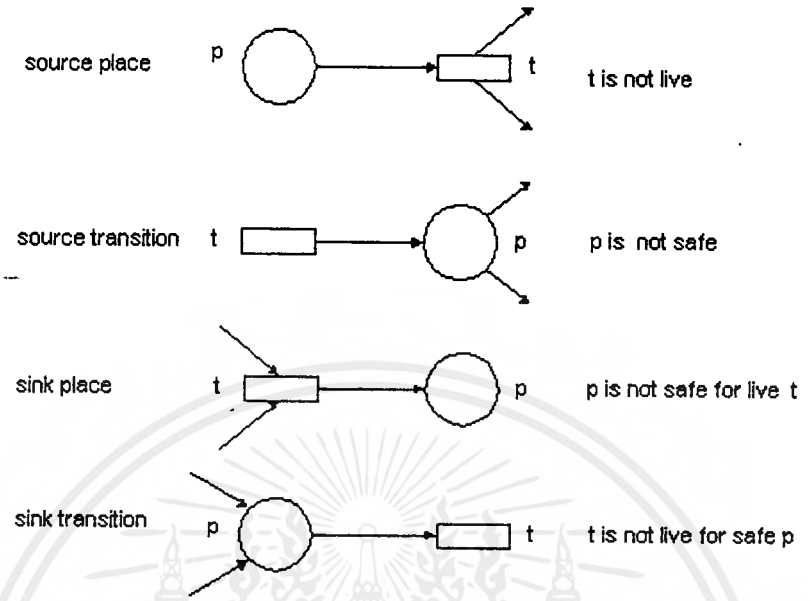



*รูป 2-26 Key structures characterizing subclass or Petri nets and their Venn diagram*

**B. Liveness and Safeness Criteria**

1) Exitstence of Live-Safe Markings:

เราจะพิจารณา condition ของ LS marking $M_0$ สำหรับ Petri net PN ดังตาราง




*รูป2-27 Explanation as to why a Live and Safe Petri Net Cannot Have Source or Sink Places and Transitions*

เรามีทฤษฎีบทดังนี้

Theorem3 : If a Petri net $(N, M_0)$ เป็น live และ safe แล้วจะไม่มี source หรือ sink places และ source หรือ sink transitions, นั่นคือ for all $x \in P \cup T, {}^\bullet x \neq \varnothing \neq x^\bullet$

2) Liveness and Safeness in SM and MG:

เมื่อ transition firing ใน state machine move 1 token จาก place ไป place ทำให้เราสามารถพิสูจน์ ทฤษฎีบทดังนี้

Theorem 4: A Connected state machine(N, $M_0$) เป็น live ถ้า strongly connected และ $M_0$ มีอย่างน้อย 1 token

Theorem 5: A strongly connected state machine (N, $M_0$) เป็น safe ถ้า $M_0$ มี token> 1และ live connected state machine (N, $M_0$) จะ safe ถ้า $M_0$ มี token 1 ตัว

☐ สำหรับ marked graphs แต่ละ place มี 1 arc ที่เข้าและออก โดยมี weight ค่าหนึ่ง if a node not lie on the directed circuit in question ,none of the arcs incident to that node will belong to the directed circuit. Thus ,we have the following token invariance property,

Theorem 6: สำหรับ marked graph, จำนวน token ใน directed circuit จะคงที่ภายใต้การ firing ใดๆนั้นคือ M( C)= $M_0$( C) สำหรับทุก directed circuit C และM ใดๆใน R($M_0$) โดย M( C)จะแสดงจำนวน token ทั้งหมดใน C.

☐ จากทฤษฎีบทที่ 6 ถ้า node ไม่เคย enable เลยในทุกการ firing เราสามารถหา token-free directed circuit ฉะนั้นจึงได้ทฤษฎีบทดังนี้

Theorem 7: marked graph(G, $M_0$) เป็น live ถ้า $M_0$ places มีอย่างน้อย 1 token ในแต่ละ directed circuit ใน G.

Theorem 8:จำนวน token ที่มากที่สุดที่ arc สามารถมีได้ใน marked graph(G, $M_0$) จะเท่ากับ จำนวน token ที่น้อยที่สุดของ $M_0$ที่อยู่บน directed circuit ใน arc ตัวนี้

☐ ถ้าพิจารณาทุก directed circuits $C_1,C_2,\ldots C_m$ โดยจะมี token หลายๆตัวที่สามารถเข้ามาใน incoming arc ของ initial node x of e = (x,y) และจะ fire node x หลายครั้ง โดยที่ไม่ firing ที่ node y เลยจะได้ว่า ถ้า Min{ $M_0(C_1)$, $M_0(C_2)$,..., $M_0(C_m)$}=1 แล้ว M(e) <= 1 สำหรับทุก M ใน R( $M_0$) ดังนั้นเราจึงได้ทฤษฎีบทดังนี้

Theorem 9: A live marked graph (G, $M_0$) จะ safe ถ้าทุก arc(place) เป็นส่วนหนึ่งของ directed circuit C with $M_0(C)=1$

Theorem 10: There exists a live และ safe marking ใน directed graph G ถ้า G เป็น strongly connected

☐ subset ของ arcs E' ใน directed graph G=(V,E) จะเป็น feedback arc set (FAS) ถ้า G'=(V,E-E')เป็น acyclic นั่นคือ ไม่มี directed circuits,และ FAS จะminimal ถ้า no proper subset ของ FAS เป็น FAS และ minimum ถ้าไม่มี FAS ใดเก็บ smaller number ของ arcs, subset ของ marked arc ใน live marked graph จะเป็น FAS ถ้าในแต่ละ arc ใน FAS ของ directed graph ถูก marked เราจะมี live marked graph ได้ดังนี้

Theorem 11: A strongly-connected live marked graph Gจะ safe ถ้าทุก marking M ใน R($M_0$) โดยที่ set ของ marked arcs เป็น minimal FAS.

**C. Reachability Criteria**

Theorem 16: ใน acyclic Petri net, $M_d$ จะ reachable จาก $M_0$ ถ้า exists a nonnegative integer solution x satisfying

☐ set ของ places ในทุก directed circuit ที่เป็น trap(siphon)จะเรียกว่า trap-circuit net หรือ TC net (siphon-circuit net หรือ SC net ) TC net และ SC net ไม่จำเป็นต้องเป็น free-choice หรือ asymmetric-choice

Theorem 17: ใน trap-circuit net, $M_d$ จะ reachable จาก $M_0$ ถ้า exists a nonnegative integer solution x ($N_x$, $M_{0x}$) ไม่มี token-free siphons

☐ set ของ places ในทุก directed circuit ที่เก็บ trap(siphon)จะเรียกว่า TCC net (SCC net) สำหรับ TC ทั่วไป (generalized SC net) จะมีเงื่อนไขสำหรับการ reachability และ stated

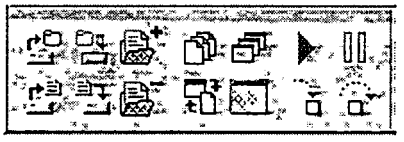
Theorem 18: ใน TCC net, $M_d$ จะ reachable จาก $M_0$ ถ้า exists a nonnegative integer solution x โดยทุก siphon ใน ($N_x$, $M_{0x}$) มี marked trap

**2.2. ความรู้เบื้องต้นเกี่ยวกับเดลไฟ**

**2.2.1. ส่วนประกอบของเดลไฟ**

แบ่งออกเป็นส่วนใหญ่ๆดังนี้

1) ส่วนของสปีดบาร์ ซึ่งจะมีปุ่มต่างๆแทนรายการของเมนู

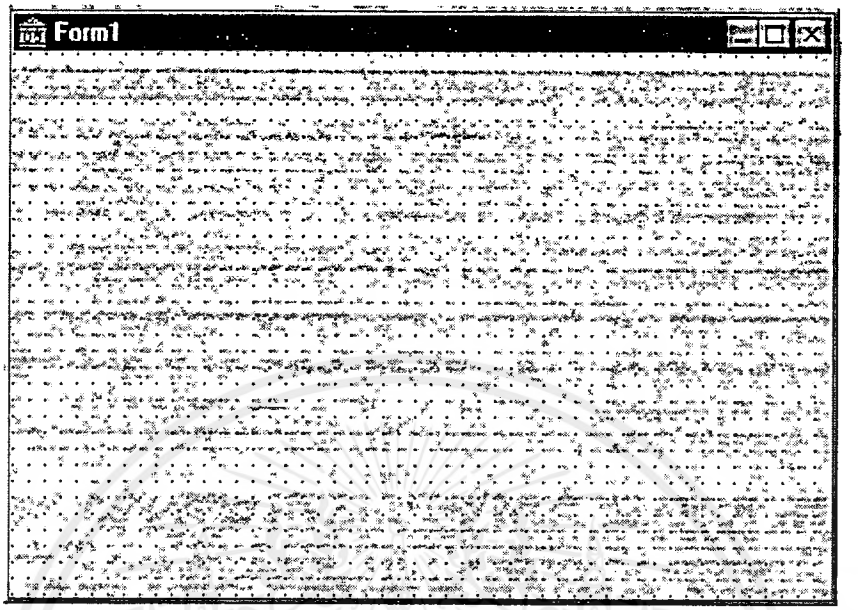


*รูป 2-28 แสดงสปีดบาร์*

2) คอมโปเนนต์แพลเลต (Comonent Palette) จะประกอบด้วยคอมโปเนนต์ต่างๆ( อาจเรียกว่า ออปเจ็กต์ หรือ คอนโทรล) ใช้สำหรับสร้างแอพพลิเคชัน คอมโปเนนต์จะแบ่งออกเป็น 8 กลุ่ม( ตามปกติ ) คือ Standard, Additionals, Data Access, Data Control, Dialogs, System, Vbx และ Samples โดยแต่ละกลุ่มจะมีที่คั่นหน้าดังรูป เมื่อคลิกที่คั่นหน้าใดก็จะมีคอมโปเนนต์ในกลุ่มนั้นๆแสดงให้เห็น

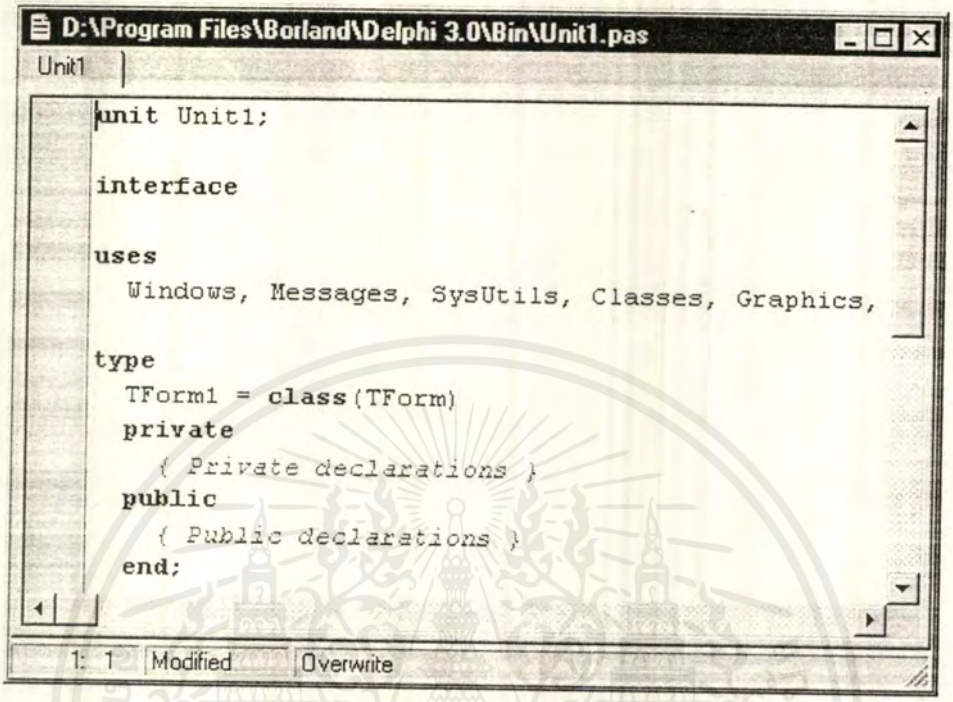

*รูป 2-29 แสดงคอมโปเนนต์แพลเลต*

3) ฟอร์ม เป็นส่วนที่ใช้สำหรับกำหนดคอมโปเนนต์ต่างๆ ฟอร์มเป็นวินโดวส์อันหนึ่งซึ่งมีคุณสมบัติเหมือนวินโดวส์ทั่วๆไป คือ ประกอบด้วย คอนโทรลเมนู, ปุ่มมินิไมซ์, ปุ่มแมกซิไมซ์, ไตเติลบาร์ และขอบหน้าต่างที่สามารถปรับขนาดได้ดังรูป




*รูป 2-30 แสดงฟอร์ม*

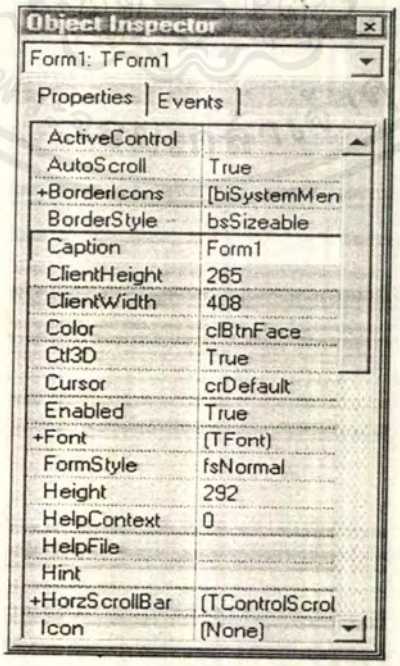
4) โค้ดเอดิเตอร์ เป็นส่วนที่ใช้สำหรับเขียนโค้ดของโปรแกรม หน้าต่างโค้ดเอดิเตอร์อยู่ใต้หน้าต่างฟอร์ม เมื่อคลิกที่ที่คั่นหน้า Unit 1 หน้าต่างของโค้ดจะปรากฏขึ้นมาทับหน้าต่างฟอร์ม หรือเลื่อนหน้าต่างออกไปที่ส่วนอื่นของหน้าจอ หรือคลิกที่ปุ่มมินิไมซ์ของหน้าต่างฟอร์ม จะเห็นหน้าต่างของโค้ดดังรูป




*รูป 2-31 แสดงหน้าต่างโค้ดเอดิเตอร์*

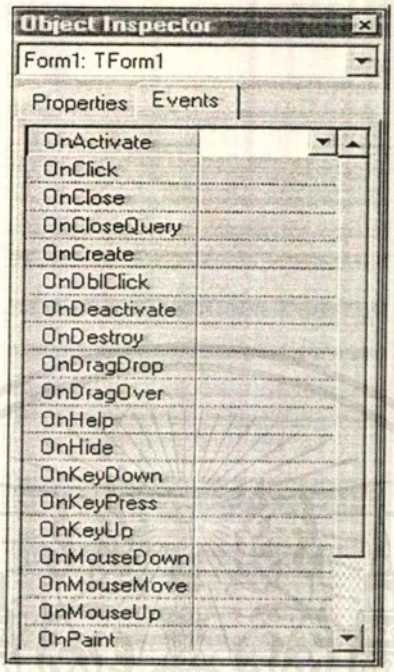
5) ออบเจ็กต์อินสเปคเตอร์ แบ่งออกเป็นสองส่วนคือ

☐ Properties เป็นส่วนที่กำหนดคุณสมบัติของคอมโปเนนต์ต่างๆ




*รูป 2-32 แสดงออบเจ็กต์อินสเปคเตอร์(Property)*

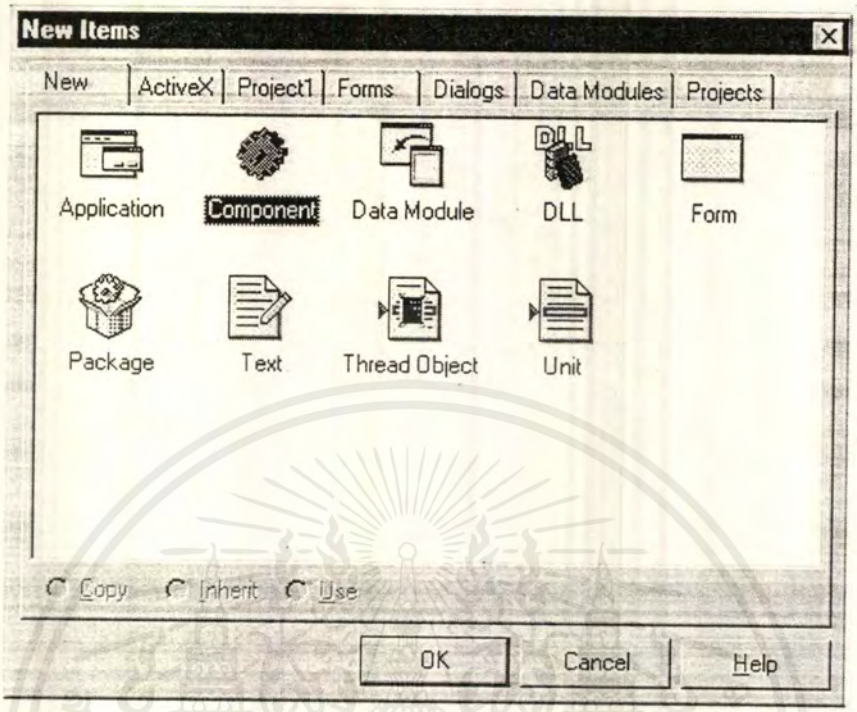
- Event เป็นส่วนที่จะกำหนดการกระทำของคอมโปเนนต์นั้นๆ ซึ่งการกระทำนั้นอาจเกิดจากตัวผู้ใช้ หรือ ตัวโปรแกรมเอง




*รูป 2-33 แสดงออบเจ็กต์อินสเปคเตอร์ (Event)*

### 2.2.2. ขั้นตอนในการสร้างคอมโปเนนต์

1 ) เลื่อนเมาส์ไปที่เมนูไฟล์แล้วเลือกเมนูไอเท็ม นิว (New) ก็จะมีฟอร์มเกิดขึ้นดังรูป




*รูป 2-34 ฟอร์ม New Items*

2) ทำการเลือกไอคอนของคอมโปเนนต์แล้วกดปุ่มโอเคหรือดับเบิลคลิก ก็จะปรากฏฟอร์มขึ้นมาให้ทำการกำหนดค่าต่างๆดังนี้

Ancester Type : เป็นการกำหนดว่าคอมโปเนนต์ที่เราจะสร้างขึ้นมานั้นจะทำการสืบทอดคุณสมบัติ มาจาก คลาสอะไร

Class Name : เป็นการกำหนดคอมโปเนนต์ที่เราจะสร้างนั้นให้มีชื่อคลาสใหม่ว่าอะไร

Palette Page : เป็นการกำหนดว่าคอมโปเนนต์ใหม่นี้จะไปอยู่ในแพลเลตไหน

Unit file name : เป็นการกำหนดว่าจะให้บันทึกคอมโปเนนต์ใหม่ในชื่อไฟล์อะไรและจะอยู่ที่ไหน

Search Path : ในส่วนนี้เราไม่จำเป็นต้องแก้ไข

3) เมื่อกำหนดค่าต่างๆแล้วก็ทำการเลือกว่าจะอินสตอลเลยหรือจะเขียนโค้ดก่อนถ้า เราทำการเลือก อินสตอล : เมื่อกดปุ่ม install จะมีฟอร์มอินสตอลขึ้นมา เราอาจจะเลือก into new package ถ้าเราต้องการที่จะเก็บไว้ในแพ็กเก็จใหม่ โดย

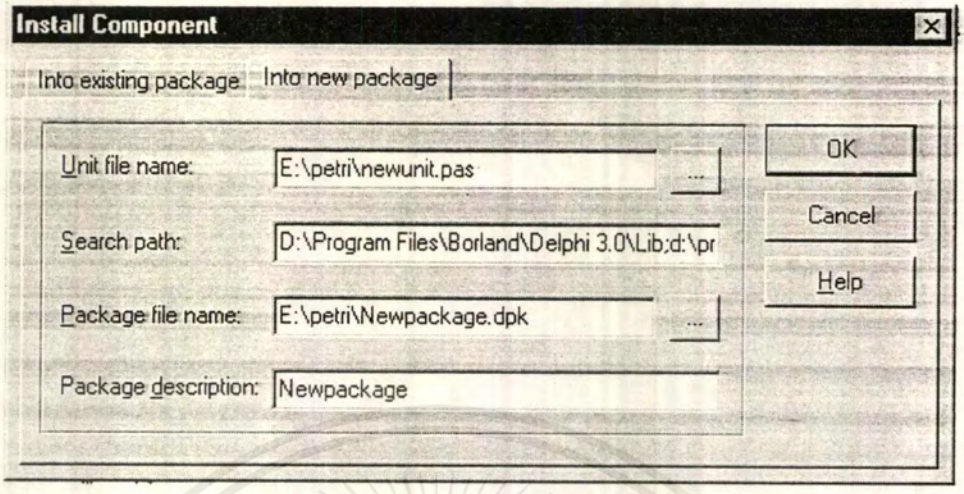
File name : เป็นการกำหนดชื่อและที่อยู่ของแพ็กเก็จ ซึ่งจะมีนามสกุลเป็น .dpk

Descripttion : เป็นชื่อของแพ็กเก็จที่จะแสดงในรายการของแพ็กเก็จ

เขียนโค้ด : เมื่อกดปุ่ม View Unit แล้วจะปรากฏฟอร์มของโค้ดเอดิเตอร์ขึ้นมาให้เราทำการแก้ไขเพิ่มเติมได้ โดยเราจะทำการเขียนโค้ดให้คอมโปเนนต์ออกมาเป็นอย่างไรก็ได้

หลังจากที่ทำการแก้ไขโค้ดเสร็จแล้วถ้าต้องการจะอินสตอลก็ทำดังนี้

เลือกเมนู Component แล้วเลือกเมนูไอเท็ม Install Component แล้วจะมีฟอร์มขึ้นมาให้กำหนดดังรูป




*รูป 2-35 ฟอร์ม Install Component*

ถ้าเราต้องการที่จะเก็บไว้ในแพ็กเกจเก็จใหม่ก็ให้เลือกที่ into new package โดย

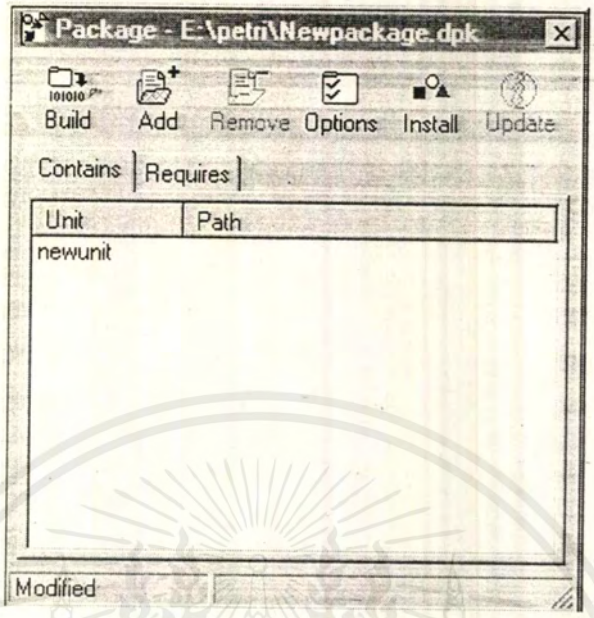
Unit File name : เป็นการระบุชื่อไฟล์ของคอมโปเนนต์ที่จะอินสตอล

Search Path : ในส่วนนี้ไม่ต้องกำหนดเพราะโปรแกรมจะกำหนดมาให้แล้ว

Package File name : เป็นการกำหนดชื่อและที่อยู่ของแพ็กเกจเก็จ ซึ่งจะมีนามสกุลเป็น .dpk

Package Description : เป็นชื่อของแพ็กเกจเก็จที่จะแสดงในรายการของแพ็กเกจเก็จ

เมื่อกดปุ่มโอเคแล้วก็จะมีฟอร์มขึ้นมายืนยันว่าต้องการจะอินสตอลหรือไม่ ถ้ากดปุ่ม Yes ก็จะมีฟอร์มของแพ็กเกจขึ้นมาดังรูป




*รูป 2-36 ฟอร์ม Install Package*

### 2.2.3. การเขียนคอมโปเนนท์

ก่อนอื่นเราจะต้องกำหนดก่อนว่าเราจะสร้าง component ตัวใหม่นี้ให้เป็น class อะไร และ derive มาจาก class ไหนในที่นี้จะยกตัวอย่างของ component line โดยจะกำหนดให้เป็น class Tline ซึ่ง derive มาจาก TGraphicControl เราจะต้องทำดังนี้

```
Type
    Tline = Class(TGraphicControl)
      Private
      { การประกาศใน private }
      Protected
      { การประกาศใน protected }
      Public
      { การประกาศใน public }
      Published
      { การประกาศใน published }
    end;
```

หลังจากนั้นจึงทำตามขั้นตอนต่างๆดังนี้

1. การกำหนด property และ event

คอมโปรเนนท์ที่สร้างขึ้นมาใหม่นั้น เราต้องการให้มีคุณสมบัติอะไรบ้าง เราจะกำหนดได้โดยประกาศไว้ในส่วนของ **published** โดยจะเขียนอยู่หลัง **property** โดยจะแบ่งออกเป็น 2 ส่วนใหญ่ๆคือ

1.1 การกำหนดให้เป็น property ใน object inspector จะแบ่งออกเป็น 2 ส่วน คือ

- property ที่จัดเตรียมไว้ให้แล้ว ในที่นี้เราเพียงแต่กำหนดขึ้นมาได้เลยเช่น

```
published
    property Color;
```

จากที่เรากำหนดเมื่อเรานำ component ตัวนี้ไป install แล้วจะมี property color ขึ้นใน object inspector โดยจะมี Tcolor ให้เราเลือกใช้ได้เลย

- property ที่เรากำหนดขึ้นมาใหม่เลย เช่น ถ้าเราต้องการกำหนดคุณสมบัติของ arc ให้สามารถเปลี่ยนชนิดของหัวลูกศรระหว่าง " สามเหลี่ยม" กับ " วงกลม " แล้วเราจะต้องกำหนด Type ขึ้นมาให้ดังนี้

```
Type
    TLinehead = (Triangle,Circle);
```

หลังจากนั้นเราต้องไปกำหนดให้มีการแสดง property ที่ใช้เลือกหัวลูกศรให้ปรากฏบน object inspector จะทำดังนี้

```
Published
    property Head: TLineHead read FHead write SetLineHead  default Triangle;
```

ซึ่งเมื่อเราทำการ install component นี้แล้วจะเห็นว่ามี property Head โดยสามารถเลือก Head เป็น ' Triangle'หรือ' Circle' ได้ จากตัวอย่างเราสังเกตุเห็นว่าจะมี ตัวหนา **read ,write,default**

**read** Fhead หมายความว่า เราจะทำการอ่านค่าของ Head จาก Fhead ไปใช้

**write** SetLinehead หมายความว่า เราจะทำการรับค่าจาก object inspector โดยใช้ procedure SetLineHead

**default** Triangle หมายความว่า เราได้กำหนดค่าเริ่มต้นของ Head ให้เป็น Triangle

หลังจากนั้นเราจะต้องกำหนดตัวแปรที่จะคอยรับค่าจาก object insprector เมื่อเรามีการเปลี่ยนลักษณะของหัวลูกศร เราจะต้องประกาศตัวแปรนั้นในส่วนของ **Private** และจะต้องขึ้นต้นตัวแปรด้วยตัว" F " และในการรับส่งค่าของproperty จะต้องมี procedure ที่ใช้ในการเปลี่ยนแปลงค่าด้วย โดยถ้าต้องการรับค่าจะต้องกำหนด procedure ให้ขึ้นต้นด้วย 'Set' และถ้าต้องการจะแสดงค่าที่เปลี่ยนไปให้ procedure ขึ้นต้นด้วย ' Get ' โดยจะประกาศ procedure ไว้ในส่วนของ **Private** เหมือนกัน ดังนี้

```
private
    Fhead : TLineHead;
    procedure SetLineHead(Value : TLineHead);
  procedure GetLineHead(Value : TLineHead);   { ในที่นี้ เราไม่ได้ใช้ procedure นี้ใน
   component ของ arc }
end;
```

เราจะยกตัวอย่างของ procedure SetLineHead ดังนี้

```
procedure TLine.SetLineHead(Value: TLineHead);
begin
    if Value <> FHead then  begin  { จะทำการเช็คดูว่าค่าที่รับมาตรงกับ Fhead ? }
      { .... }                     { แล้วแต่ว่าเราต้องการให้ทำอะไรบ้าง }
      FHead := Value;
    end;
end;
```

1.2 การกำหนดให้เป็น event ใน object inspector

เราจะกำหนดคล้ายกับ property โดยเราจะกำหนดไว้หลัง **property** แต่ว่า event ที่จะกำหนดนั้นจะต้องมีอยู่ในตัว delphi เช่น mousedown , enddrag , keydown เป็นต้น การกำหนดทำได้ดังนี้

```
Published
    property OnMouseDown;
    property OnEnddrag ;
```

จะได้ว่า component ตัวนี้จะมี event ให้ใช้เพียง 2 ตัวคือ OnMouseDown กับ OnEnddrag

2. การกำหนดให้มีการสร้าง component ตัวนี้ขึ้นมาเมื่อมีการเรียกใช้ จะต้องทำการประกาศการ create ตัวเองขึ้นในส่วนของ **Public** ดังนี้

```
public
    constructor Create(AOwner: TComponent); override;
```

เราจะยกตัวอย่างของ constructor ดังนี้

```
constructor TLine.Create(AOwner: TComponent);
begin
 inherited Create(AOwner);
 { .... }                    { ส่วนนี้เราอาจจะมีการกำหนดค่าเริ่มต้นต่างๆก็ได้ }
end;
```

3. การแสดงรูปลักษณ์ของ component เราจะต้องเขียนอยู่ใน procedure paint โดยจะต้องประกาศ procedure ไว้ในส่วนของ **protected** ดังนี้

```
protected
    procedure Paint; override;
```

สมมติว่าเราต้องการสร้าง component ที่เป็นเส้นตรงที่ลากจากมุมบนซ้ายไปยังมุมล่างขวาทุกๆ ขนาดของ component เราจะต้องเขียนใน procedure paint ดังนี้

```
procedure TLine.Paint;
b egin
      with Canvas do begin
          Moveto(0,0);
          Lineto(width-1 , height-1 );
      end;
end;
```

จาก procedure ที่เขียนเมื่อ install แล้วเวลาที่เรานำ component ตัวนี้มาใช้งาน ไม่ว่าเราจะเปลี่ยนขนาดของ component เป็นเท่าไรก็จะปรากฏเส้นทแยงมุมจากบนซ้ายไปล่างขวาเสมอ

4. การบันทึกลงไปใน Tab Control หลังจากทำการ install แล้วจะต้องประกาศ procedure register แต่จะไม่ประกาศไว้ในส่วนของ class ที่เราสร้างเราจะประกาศนอก class คือเราจะประกาศไว้หลังจาก end ของ class Tline และ ใน procedure register จะต้องระบุด้วยว่าเราจะให้ component ที่เราสร้างขึ้นมาใหม่นั้นไปอยู่ใน Tab Control อะไร เช่น ถ้าต้องการให้อยู่ใน Tab Control ใหม่ชื่อ ' Petri ' เราจะทำดังนี้

```
procedure Register;
begin
   RegisterComponents(' Petri ', [TLine] );
end;
```

' Petri ' เป็นชื่อของTab Control ที่เราจะให้ component ตัวนี้ไปอยู่

[ Tline ] เป็นชื่อ Class ที่เราสร้างขึ้นมาใหม่

### 2.2.3. การเขียนคอมโปเนนท์

ก่อนอื่นเราจะต้องกำหนดก่อนว่าเราจะสร้าง component ตัวใหม่นี้ให้เป็น class อะไร และ derive มาจาก class ไหนในที่นี้จะยกตัวอย่างของ component line โดยจะกำหนดให้เป็น class Tline ซึ่ง derive มาจาก TGraphicControl เราจะต้องทำดังนี้

```
Type
    Tline = Class(TGraphicControl)
      Private
      { การประกาศใน private }
      Protected
      { การประกาศใน protected }
      Public
      { การประกาศใน public }
      Published
      { การประกาศใน published }
    end;
```

หลังจากนั้นจึงทำตามขั้นตอนต่างๆดังนี้

1. การกำหนด property และ event

คอมโปรเนนท์ที่สร้างขึ้นมาใหม่นั้น เราต้องการให้มีคุณสมบัติอะไรบ้าง เราจะกำหนดได้โดยประกาศไว้ในส่วนของ **published** โดยจะเขียนอยู่หลัง **property** โดยจะแบ่งออกเป็น 2 ส่วนใหญ่ๆคือ

1.1 การกำหนดให้เป็น property ใน object inspector จะแบ่งออกเป็น 2 ส่วน คือ

- property ที่จัดเตรียมไว้ให้แล้ว ในที่นี้เราเพียงแต่กำหนดขึ้นมาได้เลยเช่น

```
published
    property Color;
```

จากที่เรากำหนดเมื่อเรานำ component ตัวนี้ไป install แล้วจะมี property color ขึ้นใน object inspector โดยจะมี Tcolor ให้เราเลือกใช้ได้เลย

- property ที่เรากำหนดขึ้นมาใหม่เลย เช่น ถ้าเราต้องการกำหนดคุณสมบัติของ arc ให้สามารถเปลี่ยนชนิดของหัวลูกศรระหว่าง " สามเหลี่ยม" กับ " วงกลม " แล้วเราจะต้องกำหนด Type ขึ้นมาให้ดังนี้

```
Type
        TLinehead = (Triangle,Circle);
```

หลังจากนั้นเราต้องไปกำหนดให้มีการแสดง property ที่ใช้เลือกหัวลูกศรให้ปรากฏบน object inspector จะทำดังนี้

```
Published
        property Head: TLineHead read FHead write SetLineHead  default Triangle;
```

ซึ่งเมื่อเราทำการ install component นี้แล้วจะเห็นว่ามี property Head โดยสามารถเลือก Head เป็น ' Triangle'หรือ' Circle' ได้ จากตัวอย่างเราสังเกตุเห็นว่าจะมี ตัวหนา **read ,write,default**

**read** Fhead หมายความว่า เราจะทำการอ่านค่าของ Head จาก Fhead ไปใช้

**write** SetLinehead หมายความว่า เราจะทำการรับค่าจาก object inspector โดยใช้ procedure SetLineHead

**default** Triangle หมายความว่า เราได้กำหนดค่าเริ่มต้นของ Head ให้เป็น Triangle

หลังจากนั้นเราจะต้องกำหนดตัวแปรที่จะคอยรับค่าจาก object insprector เมื่อเรามีการเปลี่ยนลักษณะของหัวลูกศร เราจะต้องประกาศตัวแปรนั้นในส่วนของ **Private** และจะต้องขึ้นต้นตัวแปรด้วยตัว" F " และในการรับส่งค่าของproperty จะต้องมี procedure ที่ใช้ในการเปลี่ยนแปลงค่าด้วย โดยถ้าต้องการรับค่าจะต้องกำหนด procedure ให้ขึ้นต้นด้วย 'Set' และถ้าต้องการจะแสดงค่าที่เปลี่ยนไปให้ procedure ขึ้นต้นด้วย ' Get ' โดยจะประกาศ procedure ไว้ในส่วนของ **Private** เหมือนกัน ดังนี้

```
private
     Fhead : TLineHead;
      procedure SetLineHead(Value : TLineHead);
   procedure GetLineHead(Value : TLineHead);   { ในที่นี้ เราไม่ได้ใช้ procedure นี้ใน
    component ของ arc }
end;
```

เราจะยกตัวอย่างของ procedure SetLineHead ดังนี้

```
procedure TLine.SetLineHead(Value: TLineHead);
begin
        if Value <> FHead then   begin  { จะทำการเช็คดูว่าค่าที่รับมาตรงกับ Fhead ? }
           { .... }                      { แล้วแต่ว่าเราต้องการให้ทำอะไรบ้าง }
           FHead := Value;
        end;
end;
```

1.2 การกำหนดให้เป็น event ใน object inspector

เราจะกำหนดคล้ายกับ property โดยเราจะกำหนดไว้หลัง **property** แต่ว่า event ที่จะกำหนดนั้นจะต้องมีอยู่ในตัว delphi เช่น mousedown , enddrag , keydown เป็นต้น การกำหนดทำได้ดังนี้

```
Published
    property OnMouseDown;
    property OnEnddrag ;
```

จะได้ว่า component ตัวนี้จะมี event ให้ใช้เพียง 2 ตัวคือ OnMouseDown กับ OnEnddrag

2. การกำหนดให้มีการสร้าง component ตัวนี้ขึ้นมาเมื่อมีการเรียกใช้ จะต้องทำการประกาศการ create ตัวเองขึ้นในส่วนของ **Public** ดังนี้

```
public
    constructor Create(AOwner: TComponent); override;
```

เราจะยกตัวอย่างของ constructor ดังนี้

```
constructor TLine.Create(AOwner: TComponent);
begin
 inherited Create(AOwner);
 { .... }                    { ส่วนนี้เราอาจจะมีการกำหนดค่าเริ่มต้นต่างๆก็ได้ }
end;
```

3. การแสดงรูปลักษณ์ของ component เราจะต้องเขียนอยู่ใน procedure paint โดยจะต้องประกาศ procedure ไว้ในส่วนของ **protected** ดังนี้

```
protected
    procedure Paint; override;
```

สมมติว่าเราต้องการสร้าง component ที่เป็นเส้นตรงที่ลากจากมุมบนซ้ายไปยังมุมล่างขวาทุกๆ ขนาดของ component เราจะต้องเขียนใน procedure paint ดังนี้

```
procedure TLine.Paint;
b egin
    with Canvas do begin
        Moveto(0,0);
        Lineto(width-1 , height-1 );
    end;
end;
```

จาก procedure ที่เขียนเมื่อ install แล้วเวลาที่เรานำ component ตัวนี้มาใช้งาน ไม่ว่าเราจะเปลี่ยนขนาดของ component เป็นเท่าไรก็จะปรากฏเส้นทแยงมุมจากบนซ้ายไปล่างขวาเสมอ

4. การบันทึกลงไปใน Tab Control หลังจากทำการ install แล้วจะต้องประกาศ procedure register แต่จะไม่ประกาศไว้ในส่วนของ class ที่เราสร้างเราจะประกาศนอก class คือเราจะประกาศไว้หลังจาก end ของ class Tline และ ใน procedure register จะต้องระบุด้วยว่าเราจะให้ component ที่เราสร้างขึ้นมาใหม่นั้นไปอยู่ใน Tab Control อะไร เช่น ถ้าต้องการให้อยู่ใน Tab Control ใหม่ชื่อ ' Petri ' เราจะทำดังนี้

```
procedure Register;
begin
   RegisterComponents(' Petri ', [TLine] );
end;
```

' Petri ' เป็นชื่อของTab Control ที่เราจะให้ component ตัวนี้ไปอยู่

[ Tline ] เป็นชื่อ Class ที่เราสร้างขึ้นมาใหม่

# บทที่ 3
# ขั้นตอนการวางแผนพัฒนาและออกแบบ

**3.1. ขั้นตอนการวางแผนและพัฒนา**

อธิบายตาม Gantt Chart ในหน้าถัดไปมีขั้นตอนดังนี้

การทำงานนั้นแบ่งออกเป็น 2 ช่วงตามภาคเรียน โดยในภาคเรียนที่ 1 จะประกอบด้วยการทำงานได้แก่

T1 ซึ่งก็คือการศึกษาในทฤษฎีและการเขียนโปรแกรมด้วยเดลไฟล์

T2 คือการศึกษาในทฤษฎีพื้นฐานของ Petri Nets ให้สามารถเข้าใจในรูปแบบทั่วไปเช่น กฎเกี่ยวกับการ firing, การพิจารณาค่าพรอพเพอตี้ส์ของ place, transition และ arc ว่าจะมีผลต่อการทำงานของโมเดลอย่างไร

T3 คือการกำหนดสเปคของแอพพลิเคชั่นที่ต้องการทำ

T4 คือการออกแบบแอพพลิเคชั่นที่ต้องการ ให้ได้ตามสเปคที่ได้กำหนด (จะอธิบายต่อไปในส่วนของการออกแบบ)

T5 คือการเขียนโปรแกรม ในการเขียนโปรแกรมในภาคเรียนนี้กำหนดไว้ว่าจะให้สามารถเขียน ในส่วนของการออกแบบทั้งหมด ซึ่งก็ได้แก่ ฟอร์มต่าง ๆ, การสร้างคอมโพเนนต์ place, transition, arc และ text frame และการสร้างออบเจ็กต์ขณะรันแอพพลิเคชั่นด้วย, การเซพไฟล์ที่ออกแบบ และการเปิดไฟล์ที่ออกแบบแล้วเป็นต้น

T6 คือการรวบรวมข้อมูลที่มีอยู่ทั้งหมดเพื่อมาทำรายงาน

T7 คือการเตรียมการ present ในภาคเรียนที่ 1

ในภาคเรียนที่ 2 มีการทำงานต่อจากภาคเรียนที่ 1 ดังนี้

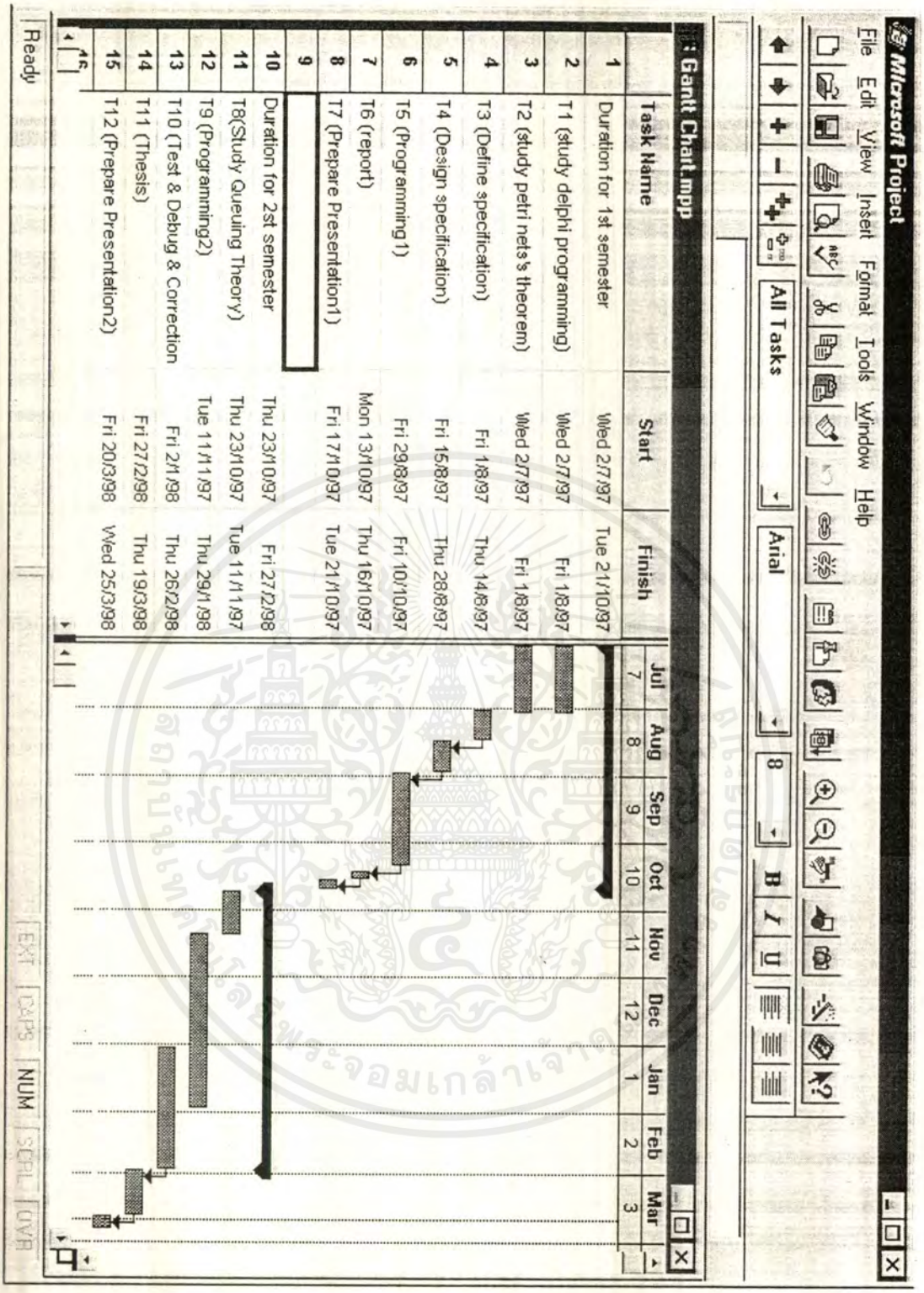
T8 คือการศึกษาทฤษฎีของ Queuing เพื่อนำมาใช้เป็นแนวทางในการซิมมูเลท

T9 คือการเขียนโปรแกรมต่อจากภาคเรียนที่แล้วที่ภาคเรียนที่แล้วทำไม่สำเร็จตามเป้าหมายคือการเขียนคอมโพเนนต์ arc, และการซิมมูเลทโมเดลที่ออกแบบโดยอาศัยทฤษฎีของ Queuing ที่ได้ศึกษามาแล้ว

T10 คือการเทสต์ ดีบัก และ แก้ไขแอพพลิเคชั่นให้ถูกต้องสมบูรณ์

T11 คือการทำรายงานฉบับย่อและวิทยานิพนธ์

T12 คือการเตรียมการ present project




*รูป 3.1 Gantt Chart อธิบายขั้นตอนการวางแผนการทำงาน*

## 3.2. การออกแบบ

### 3.2.1. Class Diagram

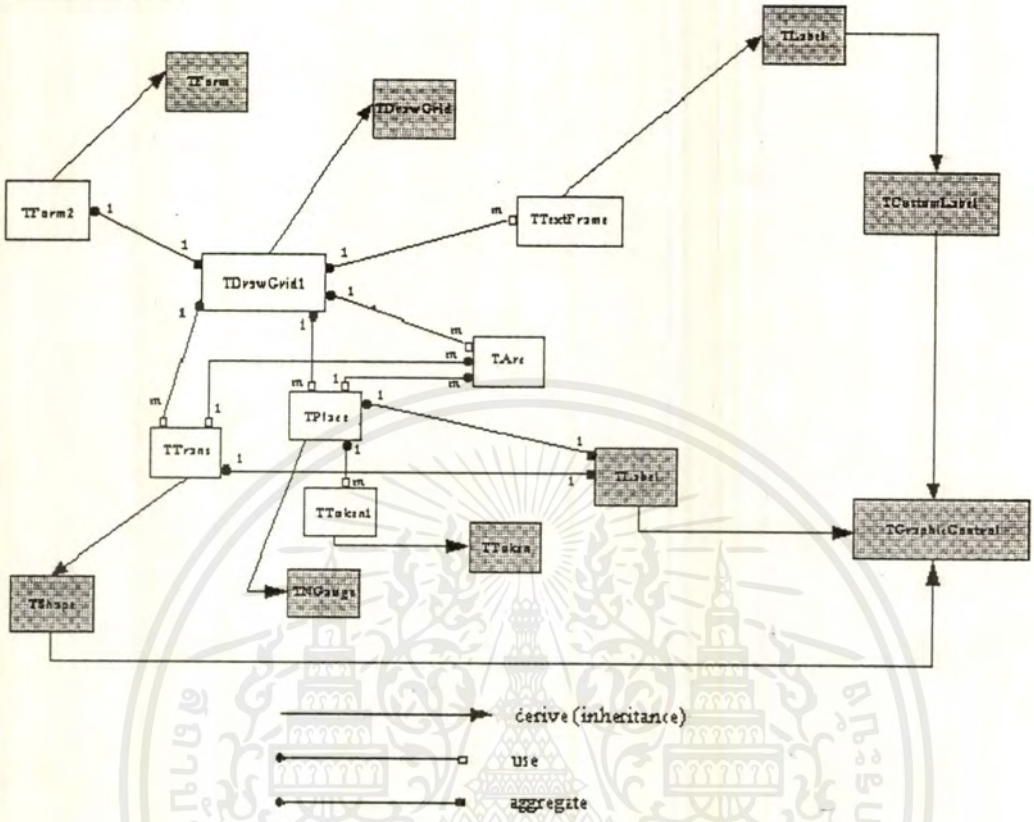
Class diagram จะบอกถึง

1. การ derive มาจากต้นกำเนิดของแต่ละ object
2. ความสัมพันธ์ของแต่ละ object ที่มีอยู่ในโปรแกรม โดยจะระบุว่ามีความสัมพันธ์แบบ 1:1 หรือ 1:m

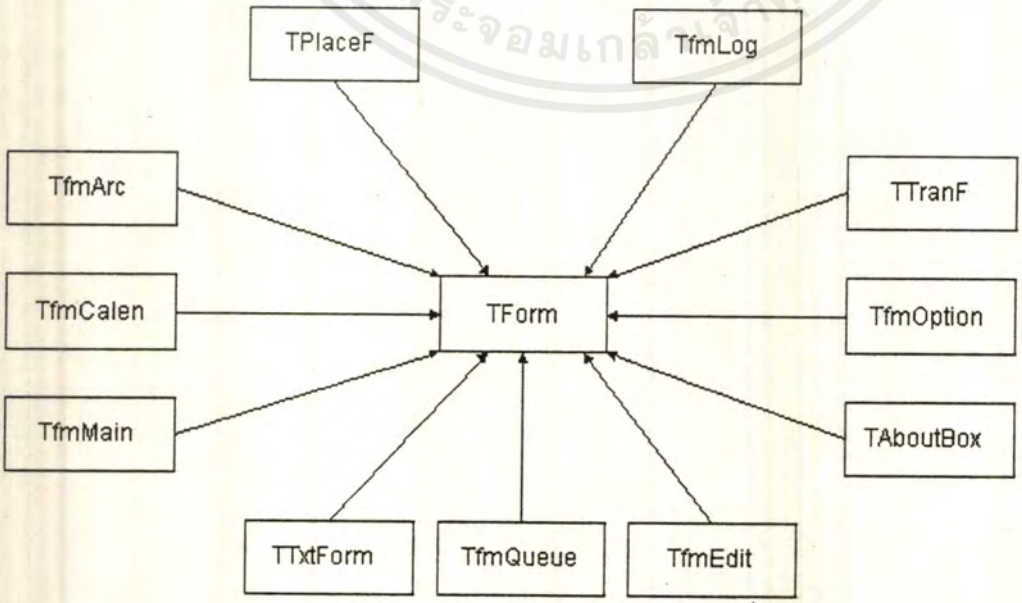
การเขียน class diagram มีวิธีและข้อกำหนดดังนี้

1. เขียนแต่ละ class ที่มีอยู่ให้อยู่ในกรอบสี่เหลี่ยม โดย class ที่สร้างขึ้นมาเองจะอยู่ในกรอบสี่เหลี่ยมสีขาว ส่วน class ที่มีอยู่เดิมแล้วจะอยู่ในกรอบสี่เหลี่ยมที่มีสีทึบ
2. แสดงความสัมพันธ์ของแต่ละ entity โดยพิจารณาดังนี้
   2.1. เราจะใช้เครื่องหมายลูกศรเป็นตัวบอกทิศทางการ derive เช่นถ้า class1 มีลูกศรชี้ไปที่ class2 แสดงว่า class1 derive มาจาก class2
   2.2. ระหว่าง class ใดๆ 2 class จะมีการแสดงความสัมพันธ์กัน โดยแบ่งได้ เป็น 2 แบบ
      2.2.1. ในระหว่าง class 2 class ที่มีเครื่องหมาย ' วงกลมสีดำ' กับ 'สี่เหลี่ยมสีขาว' จะแสดงให้เห็นว่า class ที่มีสี่เหลี่ยมสีขาวจะเป็นส่วนหนึ่งของ class ที่มีวงกลมสีดำ หรือไม่ก็ได้ตามความสัมพันธ์ที่กำกับไว้เช่น class TdrawGrid1 จะประกอบด้วย class Tplace1 หรือไม่ก็ได้ โดยใน DrawGrid1 จะประกอบด้วย Place ตั้งแต่ 0 – m ตัว เป็นต้น
      2.2.2 ในระหว่าง class 2 class ที่มีเครื่องหมาย ' วงกลมสีดำ' กับ 'สี่เหลี่ยมสีดำ' จะแสดงให้เห็นว่า class ที่มีสี่เหลี่ยมสีดำจะต้องเป็นส่วนหนึ่งของ class ที่มีวงกลมสีดำ ตามความสัมพันธ์ที่กำกับไว้เช่น ทุก class Tplace1 จะต้องมี class Tlabel1 เป็นส่วนประกอบ ซึ่งหมายความว่า ทุก Place 1 ตัวจะต้องมี label 1 ตัว( เราใช้ label แสดงชื่อของ Place )

**Class Diagram**




*รูป 3-2 class diagram ของฟอร์มคอนเทนเนอร์ (Form2)*




*รูป 3-3 class diagram แสดงฟอร์มต่างๆ ที่มีอยู่*

### 3.2.2. Class Specification

บอกรายละเอียดเกี่ยวกับคลาสต่างๆ ในโปรเจ็กต์ได้แก่

Class Name : ชื่อของคลาส

Class Role : บทบาทของคลาส

Variable : ตัวแปรที่สำคัญของคลาส

Method : เมทธอดที่สำคัญต่างๆ ทั้งที่เป็นโพรซีเดอร์และฟังก์ชั่นโดยจะแบ่งเป็น เมทธอดธรรมดา และที่เป็นอีเวนต์

**1. Class TfmMain** : เป็นคลาสของฟอร์มหลัก

Class Role : บทบาทของคลาส เป็นคลาสที่เป็นตัวทำงานหลักของแอปพลิเคชั่น การติดต่อกับฟอร์มอื่นปกติจะทำผ่านคลาสนี้ การสั่งการต่างๆ เช่นการสั่งให้มีการซิมมูเลท, การเซฟ, โหลดโมเดลที่ออกแบบไว้แล้ว โดยสามารถเปิดโมเดลได้พร้อมกันทีละหลายๆ โมเดลด้วย

Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ MainMenu1: TMainMenu;
- ☐ StatusBar1: TStatusBar;
- ☐ ToolBar1: TToolBar;
- ☐ Timer1: Ttimer;
- ☐ sbZoomIn: TSpeedButton;
- ☐ sbZoomOut: TSpeedButton;
- ☐ QueueButton: TSpeedButton;
- ☐ EditButton: TSpeedButton;
- ☐ PlaceButton: TSpeedButton;
- ☐ TransButton: TSpeedButton;
- ☐ TokenButton: TSpeedButton;
- ☐ LineButton: TSpeedButton;
- ☐ TxtFrameButton: TSpeedButton;
- ☐ CalendarButton: TSpeedButton;

Method : ประกอบด้วย methods ต่างๆดังนี้

- ☐ FormCreate : จะถูกเรียกใช้อย่างอัตโนมัติเมื่อฟอร์ม fmMain ถูกสร้างขึ้นใช้ในการตั้งค่าเริ่มต้นต่างๆของแอปพลิเคชั่น
- ☐ ResetButtonClick : จะถูกเรียกใช้เมื่อมีการกดปุ่ม reset โดยจะทำการกำหนดค่าบางค่าใหม่
- ☐ EventButtonClick : จะถูกเรียกใช้เมื่อมีการกดปุ่ม Event โดยจะทำการเรียก procedure ที่ทำการ run ทีละ event
- ☐ StepButtonClick : จะถูกเรียกใช้เมื่อมีการกดปุ่ม Step โดยจะทำการเรียก procedure ที่ทำการ run ทีละ step
- ☐ EditText1Click : จะถูกเรียกใช้เมื่อมีการเลือกเมนูไอเท็มของการ Edit โดยจะแสดงฟอร์ม Edit และข้อความที่อยู่ในไฟล์ที่เรากำลังทำงานอยู่
- ☐ RunButtonClick : จะถูกเรียกใช้เมื่อมีการกดปุ่ม Run โดยจะทำการกำหนดว่าได้มีการrun แล้ว
- ☐ Timer1Timer : เป็น timer ·Õè¨Ðà¡Ô´¢Öé¹เพื่อที่จะให้ทำการรันแบบต่อเนื่อง

**2. Class Tform2** : เป็นคลาสของฟอร์ม Container

Class Role : บทบาทของคลาสเป็นคลาสที่ใช้เป็น Container เป็นฟอร์มที่ใช้สำหรับการออกแบบโมเดล 1 ไฟล์ของโมเดลจะอ้างด้วย 1 คลาสนี้

Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ftrans : TTrans1;
- fplace : TPlace1;
- fTxtFrame : TTxtFrame1;
- farc : TArc;
- fToken : TToken;
- FShape1,FShape2,FShape3,FShape4,
- FShape5,FShape6,FShape7,FShape8 : TShape1;
- DragR : Trect;
- GridSize : Integer;
- queue,Service : string[12];
- pm1,pm2,pm3,pm4 : string;
- prior,prop : string[5];

{variable for simultation}

- First_Event : Boolean;
- end_sim : Boolean;
- IsFirePtoT: Boolean;
- IsFireStep: Boolean;
- EventCount,StepCount : integer;
- TimeCount : integer;
- TimeStep : integer;
- BeginEvent: Boolean;
- BeginStep : Boolean;
- TimeEachStep : integer;
- RunStep : Boolean;
- RunSlow : Boolean;

Method : ประกอบด้วย Method ต่างๆดังนี้

- FormClose : จะถูกเรียกใช้เมื่อฟอร์ม containner ถูกปิด
- FormCreate :จะถูกเรียกใช้เมื่อฟอร์ม containner ถูกสร้างขึ้นมา
- Reset_Value :จะถูกเรียกใช้เมื่อมีการกดปุ่ม reset เพื่อที่จะกำหนดค่าเริ่มต้นใหม่กับการ run
- DrawGrid1MouseMove :จะถูกเรียกใช้เมื่อมีการเคลื่อนที่ของ mouse ผ่าน drawgrid
- DrawGrid1MouseDown :จะถูกเรียกใช้เมื่อมีการกด mouse ลงบน drawgrid
- DrawGrid1DragOver :จะถูกเรียกใช้เมื่อมีการ drag component ลงบน drawgrid
- open( filename :string) : ใช้ในการแปลงจาก Petri net file ไปเป็นรูปแสดงบน containner
- DrawPlace(Cellx,Celly:integer) : ใช้สร้าง object ของ place ขึ้นมาพร้อมทั้งกำหนดค่าเริ่มต้น
- DrawTrans(Cellx,Celly:integer) : ใช้สร้าง object ของ transition ขึ้นมาพร้อมทั้งกำหนดค่าเริ่มต้น
- DrawText(Cellx,Celly:integer) :ใช้สร้าง object ของ Text ขึ้นมาพร้อมทั้งกำหนดค่าเริ่มต้นต่างๆ
- DrawArc :ใช้สร้าง object ของ arc ขึ้นพร้อมทั้งกำหนดค่าเริ่มต้นต่างๆ
- create8shape : สร้าง shape สี่เหลี่ยม 8 ตัวที่ใช้แสดงเป็นกรอบของ component เมื่อ click

- DrawGrid1KeyDown : ทำการ check ว่ามีการกด key อะไรบ้างแล้วก็ทำตาม procedure ที่กำหนดไว้
- FindShape(x,y:integer;var Shp:Tcontrol):boolean : เป็นการ checkว่า ณ ตำแหน่งที่เรา Click mouse นั้นมี shape อยู่หรือเปล่าโดยจะส่งค่า True ถ้ามี และส่งค่า false ถ้าไม่มี โดยจะต้องส่งพิกัดของ mouse ที่ click ลงบน containner และจะ return shp กลับมาด้วยถ้ามีการ click โดน shape(place หรือ transition)
- ChkClickLine(X1,Y1,X2,Y2,ClkX,Clky:integer):boolean : เป็นการ check ว่ามีการ click โดน arc หรือเปล่า โดยจะต้องพิกัดหัวท้ายของ arc พร้อมทั้งพิกัดของ mouse ที่ click ลงบน containner แล้วจะส่งค่า true ถ้า click โดนและจะส่งค่า false ถ้า click ไม่โดน
- AddNewArc : เป็นการ add objectของ Tarc ตัวล่าสุดให้กับ place และ Transition ที่ arc ตัวนี้เชื่อมอยู่
- MoveArcWithShape(Sender:TObject): เป็นการย้าย arc ตาม shape(place หรือ transition) เมื่อมีการ drag place หรือ transition ไปช่องอื่น โดยจะต้องส่ง shape ที่ย้ายนั้นเข้ามาด้วย
- ChkParallel(Arc1,Arc2:TArc):Boolean: จะเป็นตัว check ว่า arc1 กับ arc2 ขนานกันหรือเปล่าคือ check ว่ามี place กับ transition ตัวเดียวกัน หรือเปล่า
- Fire1Event เป็นการสั่งให้ Active MDI Form ของ Main Form ทำการซิมมูเลท 1 Event
- Fire1Step เป็นการสั่งให้ Active MDI Form ของ Main Form ทำการซิมมูเลท 1 Step
- CheckToken(PlaceC: TPlace1; number: integer; ColorC: TColor): Boolean เป็นการเช็คว่า Place ที่ระบุนี้มี Token ตามจำนวนและสีที่ต้องการหรือไม่
- Reset_State เมื่อมีการกดรีเซ็ตและทำการสร้าง object ใหม่ขึ้นในฟอร์มก็จะเริ่มตั้งค่าเริ่มต้นต่างๆ เพื่อการเริ่มการซิมมูเลทชั่น
- RandomTransIndex เป็นการจัดลำดับการเลือก Transtion ที่จะให้เข้ามาทำงานใหม่
- ShellSort(Var L: PList; max:integer) เป็นการเรียงลำดับความสำคัญ(Priority) ของแต่ละ Transition ที่อยู่ในลิส L โดยอาศัยเทคนิคที่เรียกว่า shell sort
- SaveLog_List เป็นการบันทึกผลการทำงานในแต่ละ Event ลงไปใน Log List
- SaveMarking_List เป็นการบันทึกค่า Marking ในแต่ละ Event ลงใน Marking List
- SetColor(No_Color: integer):Tcolor เป็นการแปลงค่าของเลขสีให้เป็นค่าของสี
- Color2Int(ColorF: TColor): integer เป็นการแปลงค่าของสีให้เป็นค่าของเลขสี
- IntException0(S1: String): String เป็นฟังก์ชั่นที่เช็คว่า String S1 ที่ส่งเข้ามานี้สามารถแปลงเป็นค่าอินทีเจอร์ได้หรือไม่ ถ้าไม่ได้ก็จะรีเทอร์นค่า '0' ออกมา
- IntException1(S1: String): Stringเป็นฟังก์ชั่นที่เช็คว่า String S1 ที่ส่งเข้ามานี้สามารถแปลงเป็นค่าอินทีเจอร์ได้หรือไม่ ถ้าไม่ได้ก็จะรีเทอร์นค่า '1' ออกมา

**3. Class Ttoken** : เป็นคลาสของ object token

Class Role : บทบาทของคลาส เป็นคลาสของ Token ที่จะสามารถถูกสร้างอ้างเป็นออปเจกต์ที่เข้ามาใน Place

Class Variable : ประกอบด้วยตัวแปรคือ

- ☐ Color : TColor;

**4. Class Tarc** : เป็นคลาสของ object arc

Class Role : บทบาทของคลาส เป็นคลาสของ Arc ที่เชื่อมต่อระหว่าง Place กับ Transition

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ Memoline : integer;
- ☐ Src,Des : TControl;
- ☐ fx,fy,sx,sy : integer;
- ☐ PtrArc : TList;
- ☐ FWeightType : string[12];
- ☐ arcType : char;
- ☐ FColor : string[4];
- ☐ ParallelArc : TArc;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ ArcMouseDown : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดลงบน Arc
- ☐ ArcMouseMove : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มาเคลื่อนที่ผ่าน Arc
- ☐ ArcDragOver: ถูกเรียกใช้อย่างอัตโนมัติเมื่อมีการdrag componet ลงบน Arc

**5. Class Ttrans1** : เป็นคลาสของ object transiton

Class Role : บทบาทของคลาส เป็นคลาสที่แทนอีเวนต์ของระบบ สามารถแสดงตัวว่า ณ ขณะนั้นกำลัง enable อยู่ได้

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ FName : string;
- ☐ Memoline : integer;
- ☐ Fx,Fy : string[4];
- ☐ FSrvValue : string[12];
- ☐ FLabel : Tlabel;
- ☐ TRPtr : TList;
- ☐ ParallelArc : Boolean;
- ☐ TimeEnable : integer;
- ☐ FirstEnable: Boolean;
- ☐ Enable_idle : Boolean;
- ☐ ServSum:integer;
- ☐ disableTime,IdleTime: real;

Class Methods : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ TransMouseMove : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มาเคลื่อนที่ผ่าน transition
- ☐ TransMouseDown : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดลงบน transition
- ☐ TransEndDrag : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมีการปล่อย transition หลังจากทำการ drag
- ☐ TransOnClick : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดแล้วปล่อย(click) บน transition

**6. Class Tplace1** : เป็นคลาสของ object place ของการสร้าง place ลงบน containner

Class Role : บทบาทของคลาส เป็นคลาสที่แทนใช้จำลองแทนเงื่อนไขของระบบ สามารถเก็บ Token เพื่อระบุความพร้อมของเงื่อนไข

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ FName : string;
- ☐ Memoline : integer;
- ☐ Fx,Fy : string[4];
- ☐ FLabel : TLabel;
- ☐ PLPtr : TList;
- ☐ NewAssign: Boolean;
- ☐ tempMarking : integer;
- ☐ Marking : TList;
- ☐ M0 : TList;
- ☐ ArrvSum:integer;
- ☐ ArrvPTime,ArrvDistance,ThroughtPSum:real;
- ☐ PassDistance,PassProb,WaitingTime,Queuelength :real;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ PlaceMouseMove : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มาเคลื่อนที่ผ่านplace
- ☐ PlaceMouseDown : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดลงบน palce
- ☐ PlaceEndDrag: ถูกเรียกใช้อย่างอัตโนมัติเมื่อมีการปล่อย place หลังจากทำการ drag
- ☐ PlaceOnClick : ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดแล้วปล่อย(click) บน place

**7. Class TTxtFrame1** : เป็นคลาสของ object text

Class Role : บทบาทของคลาส เป็นคลาสที่ใช้แสดงข้อความบนฟอร์มที่ออกแบบ

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ FName : string;
- ☐ MemoLine : Integer;
- ☐ FLines : Integer;
- ☐ FOldLines : Integer;
- ☐ FSize : Integer;
- ☐ FrameTypNo : Integer;
- ☐ FWidth : Integer;
- ☐ FHeight : Integer;
- ☐ Fx,Fy : string[4];

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ TxtFrameMouseDown: ถูกเรียกใช้อย่างอัตโนมัติเมื่อมี mouse มากดลงบน TxtFrame
- ☐ TxtFrameEndDrag: ถูกเรียกใช้อย่างอัตโนมัติเมื่อมีการปล่อย TxtFrame หลังจากทำการ drag
- ☐ TxtFrameDragOver:จะถูกเรียกใช้อย่างอัตโนมัติเมื่อมีการ drag component ลงบน TxtFrame

**8. Class TPlaceF** : เป็นคลาสของ place form

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์มที่แสดง และแก้ไขคุณสมบัติของ Place ผ่านฟอร์มนี้

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ PlName: TEdit;
- ☐ PlCap: TEdit;
- ☐ PlQueue: TComboBox;
- ☐ PlMark: TLabel;
- ☐ PlFree: TLabel;
- ☐ PlResrv: TLabel;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ PlNameChange
- ☐ PlCapChange
- ☐ PlQueueChange
- ☐ ChangePos
- ☐ FormShow
- ☐ PlNameKeyDown

**9. Class TTxtForm**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์มที่แสดง และแก้ไขคุณสมบัติของ TextFrame ผ่านฟอร์มนี้

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ Memo1: TMemo;
- ☐ cbFrameType: TComboBox;
- ☐ bbOK: TBitBtn;
- ☐ bbCancel: TBitBtn;
- ☐ seChrSize: TSpinEdit;
- ☐ seWidth: TSpinEdit;
- ☐ seHeight: TSpinEdit;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ bbOKClick : ตกลงแล้วเปลี่ยนตามการแก้ต่างๆ ที่ทำลงไปในฟอร์มนี้
- ☐ bbCancelClick : ยกเลิกการแก้ไขที่ทำลงไปในฟอร์มนี้
- ☐ seChrSizeChange : แก้ขนาดของตัวอักษรของ Text Frame
- ☐ cbFrameTypeChange : แก้ชนิดของเฟรมของ Text Frame
- ☐ seWidthChange : แก้ขนาดของความกว้างเฟรมของ Text Frame
- ☐ seHeightChange : แก้ขนาดของความกว้างเฟรมของ Text Frame
- ☐ Memo1KeyUp : แก้ข้อความของ Text Frame

**10. Class TTranF**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์มที่แสดง และแก้ไขคุณสมบัติของ Transition ผ่านฟอร์มนี้

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ TrName: TEdit;
- ☐ TrServ: TComboBox;
- ☐ TrPrm1: TEdit;
- ☐ TrPrm2: TEdit;
- ☐ TrPrm4: TEdit;
- ☐ TrPrior: TEdit;
- ☐ TrProp: TEdit;
- ☐ TrPrm3: TEdit;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ TrNameChange
- ☐ TrServChange
- ☐ TrPrm1Change
- ☐ TrPrm2Change
- ☐ TrPrm3Change
- ☐ TrPrm4Change
- ☐ TrPriorChange
- ☐ TrPropChange
- ☐ ChangePos
- ☐ FormShow
- ☐ TrNameKeyDown

**11. Class TfmArc**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์มที่แสดงและแก้ไข Arc ผ่านฟอร์มนี้

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ laSource: Tlabel;
- ☐ laDesc: Tlabel;
- ☐ laArcType: TLabel;
- ☐ laWeight: Tlabel;
- ☐ lapara1: Tlabel;
- ☐ lapara2: Tlabel;
- ☐ lapara3: Tlabel;
- ☐ lapara4: Tlabel;
- ☐ lacolor: Tlabel;
- ☐ edPrm1: TEdit;
- ☐ edPrm2: TEdit;
- ☐ edPrm3: TEdit;
- ☐ edPrm4: TEdit;
- ☐ edColor: TEdit;
- ☐ cbArcType: TComboBox;
- ☐ cbWeight: TComboBox;
- ☐ edSrc: TLabel;
- ☐ edDesc: TLabel;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ cbArcTypeChange
- ☐ cbWeightChange
- ☐ edPrm1Change
- ☐ edPrm2Change
- ☐ edPrm3Change
- ☐ edPrm4Change
- ☐ edColorChange
- ☐ ChangePos
- ☐ FormShow

**12. Class Tshape1**

Class Role : บทบาทของคลาส เป็นคลาสของ shape ที่ใช้แสดงกรอบสี่เหลี่ยมเมื่อมีการกดที่ element เพื่อเป็นการระบุว่า object ใดกำลังถูกชี้หรือเลือกอยู่

**13. Class TfmCalen**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์มที่แสดง event ที่กำลัง enable อยู่

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ laEvent: TLabel;
- ☐ laTime: TLabel;
- ☐ Bevel1: TBevel;
- ☐ Bevel2: TBevel;
- ☐ Label1: TLabel;
- ☐ Label2: TLabel;
- ☐ Label3: TLabel;
- ☐ Label4: TLabel;
- ☐ Label5: TLabel;
- ☐ Label6: TLabel;
- ☐ Label7: TLabel;
- ☐ Label8: TLabel;
- ☐ Label9: TLabel;
- ☐ Label10: TLabel;
- ☐ Label11: TLabel;
- ☐ Label12: TLabel;
- ☐ Label13: TLabel;
- ☐ Label14: TLabel;
- ☐ Label15: TLabel;
- ☐ Label16: TLabel;
- ☐ Label17: TLabel;
- ☐ Label18: TLabel;
- ☐ Label19: TLabel;
- ☐ Label20: TLabel;
- ☐ Label21: TLabel;
- ☐ Label22: TLabel;
- ☐ Label23: TLabel;
- ☐ Label24: TLabel;
- ☐ Label25: TLabel;
- ☐ Label26: TLabel;
- ☐ Label27: TLabel;
- ☐ Label28: TLabel;
- ☐ Label29: TLabel;
- ☐ Label30: TLabel;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ ShowEvent : แสดงอีเวนต์หรือ transition ต่างๆ ที่ enable อยู่ด้วย label
- ☐ ClearLabel : ทำให้ label ที่แสดงอีเวนต์มองไม่เห็นเพื่อ เป็นการรีเซ็ต

**14. Class TfmEdit**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์ม Edit สำหรับการแสดงและแก้ไขข้อความต่างๆ เช่น แสดงรูปแบบเท็กซ์ไฟล์ที่บันทึกไว้ของโมเดล

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ Memo1: TMemo;
- ☐ Panel1: TPanel;
- ☐ sbPrint: TSpeedButton;
- ☐ sbSave: TSpeedButton;
- ☐ sbFind: TSpeedButton;
- ☐ sbExit: TSpeedButton;
- ☐ Editfind: TFindDialog;
- ☐ laLine: TLabel;
- ☐ laCol: TLabel;
- ☐ Bevel1: TBevel;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ sbFindClick : ทำการแสดง Find dialog เพื่อทำการค้นหาข้อความที่ต้องการ
- ☐ FormResize :ถูกใช้เมื่อมีการเปลี่ยนแปลงขนาดของฟอร์ม Edit
- ☐ Memo1MouseDown :ถูกใช้เมื่อมี mouse กดลงบนฟอร์ม Edit โดยจะแสดง row,col
- ☐ FindLine_Col : ทำการค้นหาตำแหน่งของ Cursor ว่าอยู่ที่บรรทัดที่เท่าไรและ Column เท่าไร { for Reachability }
- ☐ FindSize(var SizePl,SizeTr,SizeAr : integer): เป็นการหาว่าในฟอร์มที่กำลังแสดงผลอยู่มีจำนวนของ place,transition,arc เท่าไร
- ☐ FindNetType:string : การหาชนิดของ model ว่าเป็นชนิดไหนโดยจะ return ผลที่ได้กลับมา
- ☐ Output(Chr_Y_N:char):string : เป็นการแปลงผลจากค่าที่ได้จากการทำ FindNetType ให้แสดงออกมาในรูปของ yes , no โดยจะใช้ใน procedure ของ displayNetType
- ☐ FormShow : จะถูกเรียกใช้เมื่อฟอร์ม Edit ถูกกำหนดให้มีการแสดงฟอร์มออกมาโดยจะทำการ check ดูว่าจะแสดงผลของอะไรเช่น edit petri net file , reachability , place statistic or transition statistic
- ☐ Showtitle(line:integer;str:string) : เป็นการแสดงหัวข้อในการแสดงผลของการวิเคราะห์ reachability โดยจะต้องส่งบรรทัดเริ่มต้นที่จะแสดงหัวข้อ และ ข้อความของหัวข้อด้วย
- ☐ ChkPre_Post_Arcless(Sender:TControl;var Preless,Postless:boolean)
- ☐ sbSaveClick : ทำการบันทึก text ที่แสดงใน memo ลงไปในไฟล์
- ☐ DisplayTstTran : เป็นการแสดงผลของ transition statistic โดยจะมีการเรียกใช้ฟังก์ชัน CatStringToShow เพื่อเอาข้อความมาแสดงในแต่ละบรรทัด
- ☐ DisplayTstPlace : เป็นการแสดงผลของ place statistic โดยจะมีการเรียกใช้ฟังก์ชัน CatStringToShow เพื่อเอาข้อความมาแสดงในแต่ละบรรทัด

**15. Class TfmLog**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์ม Log file ที่จะแสดงสิ่งที่ได้บันทึกผลที่ได้จากการซิมมูเลท

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ Bevel1: TBevel;
- ☐ bbClear: TBitBtn;
- ☐ edFileName: TEdit;
- ☐ cbMark: TCheckBox;
- ☐ bbClose: TBitBtn;
- ☐ RadioGroup1: TRadioGroup;
- ☐ bbEdit: TBitBtn;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ bbClearClick : ลบ log ที่บันทึกไว้แล้วออกให้หมด
- ☐ bbEditClick : แสดงข้อมูลของ log file ที่บันทึกไว้โดย Edit Form
- ☐ bbCloseClick : ปิด log form นี้

**16. Class TAboutBox** : เป็นคลาสของฟอร์ม About

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์ม About

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- ☐ Image1: TImage;
- ☐ Copyright: TLabel;
- ☐ ProgramIcon: TImage;
- ☐ Comments: TLabel;
- ☐ FreeRes: TLabel;
- ☐ Version: TLabel;
- ☐ PhysMem: TLabel;
- ☐ ProductName: TLabel;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- ☐ FormCreate เมื่อฟอร์มถูกสร้างขึ้นมาก็จะแสดงสถานะของการใช้เมมโมรี่
- ☐ Image1Click : เมื่อมีการ click รูปก็จะทำการปิดฟอร์ม about
- ☐ ProgramIconClick :เมื่อมีการ click ที่ icon ก็จะทำการปิดฟอร์ม about

**17. Class TfmOption** : เป็นคลาสของฟอร์ม Option

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์ม Option ที่จะแสดงและสามารถรับการแก้ไข Option ต่างๆ ของ แอปพลิเคชั่น

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

- Bevel1: TBevel;
- gbMisc: TGroupBox;
- CheckBox2: TCheckBox;
- CheckBox3: TCheckBox;
- CheckBox4: TCheckBox;
- CheckBox6: TCheckBox;
- CheckBox7: TCheckBox;
- CheckBox8: TCheckBox;
- CheckBox9: TCheckBox;
- CheckBox10: TCheckBox;
- CheckBox11: TCheckBox;
- CheckBox12: TCheckBox;
- CheckBox13: TCheckBox;
- CheckBox14: TCheckBox;
- CheckBox15: TcheckBox;
- gbZoom: TgroupBox;
- GroupBox2: TgroupBox;
- GroupBox4: TgroupBox;
- RadioButton1: TradioButton;
- RadioButton2: TradioButton;
- cbUseGrid: TcheckBox;
- cbGridSize: TcomboBox;
- delayspeed: Tedit;
- ComboBox3: TcomboBox;
- ComboBox4: TcomboBox;
- bbOK: TBitBtn;
- bbCancel: TBitBtn;
- Memo1: TMemo;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- bbOKClick :ถูกเรียกใช้เมื่อมีการกดปุ่ม OK บนฟอร์ม Option และจะทำการบันทึก Optionไว้แล้วเปลี่ยนแปลงฟอร์มตาม Option ที่กำหนดแล้วจึงปิดฟอร์ม Option
- bbCancelClick : ปิดฟอร์มโดยไม่เกิดอะไรขึ้น
- FormCreate : จะถูกเรียกใช้เมื่อฟอร์ม Option ถูกสร้างขึ้นมาจะทำการกำหนดค่าเริ่มต้นต่างๆ
- NotFoundIni : เป็นการกำหนดค่าของ Option ให้ถ้าหาไฟล์ .ini ไม่พบ
- CheckIniFile : เป็นการเช็คและแก้ไข syntax ของ initial file(Ksimnet.ini) ก่อนที่จะแสดง Option ต่างๆ ใน Option Form
- Initial :เป็นการแปลงจากไฟล์ .ini มาแสดงผลที่ฟอร์มของ Option
- SaveToMemo1 :ทำการบันทึก Option ต่างๆที่กำหนดลงบนไฟล์
- cbUseGridClick : เป็นการกด Check box เพื่อเลือกว่าจะให้แสดง Grid ใน MDI Child ฟอร์มด้วยหรือไม่
- cbGridSizeChange : เมื่อมีเลือกขนาดของ Grid ก็จะทำการเปลี่ยนขนาดของ Grid ใน MDI Child
- currentPathChange : เมื่อมีการเปลี่ยน Path ในการเปิดและเซฟไฟล์

**18. Class TfmQueue**

Class Role : บทบาทของคลาส เป็นคลาสของฟอร์ม Queue ที่แสดงคิวของ Token ที่มีอยู่ใน Place

Class Variable : ประกอบด้วยตัวแปรต่างๆดังนี้

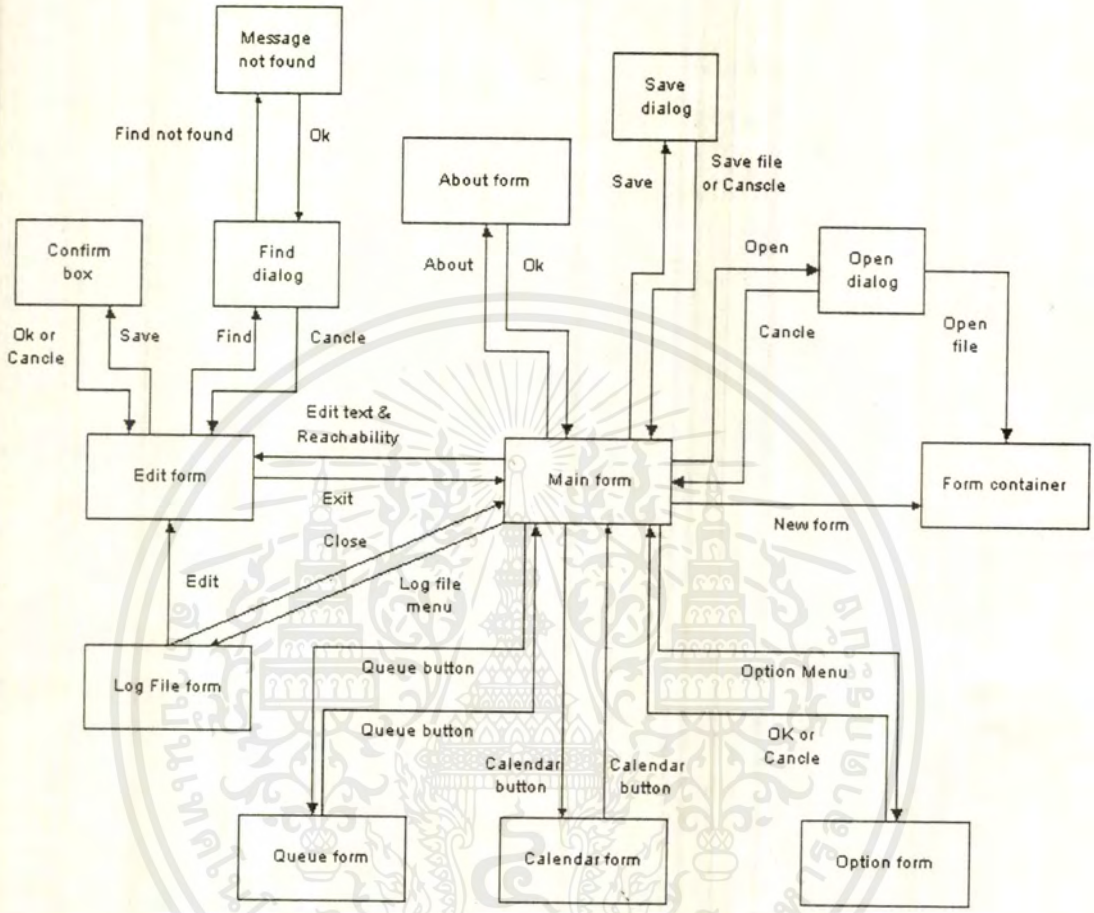
- ComboBox1: TComboBox;
- Label15: TLabel;
- Label14: TLabel;
- Label13: TLabel;
- Label12: TLabel;
- Label11: TLabel;
- Label10: TLabel;
- Label9: TLabel;
- Label8: TLabel;
- Label7: TLabel;
- Label6: TLabel;
- Label5: TLabel;
- Label4: TLabel;
- Label3: TLabel;
- Label2: TLabel;
- Label1: TLabel;
- ComboBox2: TComboBox;

Class Method : ประกอบด้วย Methods ต่างๆดังนี้

- FormShow : เมื่อฟอร์ม Queue มีการแสดงขึ้นก็จะนำชื่อของ Place ต่างๆ ที่มีอยู่มาลิสไว้ใน ComboBox
- ComboBox1Change : เมื่อมีการเปลี่ยนค่าใน ComboBox ใหม่ ก็จะเปลี่ยน Place ที่จะให้แสดงผลใน Queue
- ClearLabel : เป็นการปิดการแสดงผลจำนวน Token ที่อยู่ใน Queue ทั้งหมด
- FormClose : จะถูกเรียกใช้เมื่อมีการปิดฟอร์มของ Queue
- Color2Int(ColorF: TColor): integer : เป็นการแปลงชนิดของค่าสีเป็นค่าของตัวเลข integer

### 3.2.3. Form Diagram

แสดงฟอร์มที่มีอยู่ของแอพพลิเคชั่น การติดต่อกันระหว่างฟอร์มและชื่อฟังก์ชั่นที่ของแต่ละฟอร์มที่จะมาทำงานกับฟอร์มอื่นเช่น main form สามารถเปิด form container หรือ form2

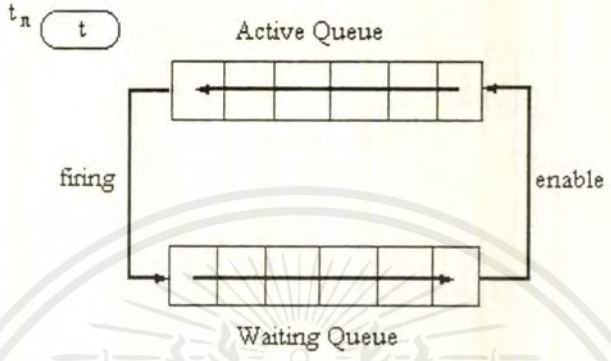



*รูป 3-4 แสดง Form Diagram*

### 3.2.4. Queuing Theory

หลักการที่นำมาใช้ในการอธิบายเกี่ยวกับวิธีการ simulate การจัดการกับ event (Transition) ที่มีอยู่ใน model จะมีขั้นตอนการวิเคราะห์อย่างไรนั้น จะขออธิบายดังต่อไปนี้

โมเดลที่จะช่วยในการอธิบายวิธีการ simulate ดังรูป




*รูป3-5 แสดงโมเดลที่ใช้อธิบายการแสดงการทำงาน(simulation)*

**time $t_n$** จะบอก time ของ step นั้น (time step) โดยเริ่มต้น $t_0$ จะมีค่าเท่ากับ 0

Queue ที่จะเก็บ transition ระบุถึง status ของ event ได้แก่

1. Waiting Queue เก็บ transition ที่มี status เป็น off และรอการ enabled การจัดลำดับ queue ใน Waiting Queue นี้จะถูกทำการ random แล้วนำมาเรียงลำดับตามค่า Property และ Probability ของ Transition แต่ละตัว
2. Active Queue เก็บ transition ที่มี status เป็น on และรอการ firing การจัดลำดับ queue ใน Active Queue นี้จะเรียงตามค่าของ Time ที่จะ fire ออกไป (ค่านี้จะกล่าวอีกที) และลำดับการเข้ามาใน Queue นี้

state ของ event (Transition) แบ่งออกได้เป็น 2 ช่วง คือ

1. ช่วง enable เป็นช่วงที่ transition จะเปลี่ยน status จาก off เป็น on หรือ มีการส่ง token จาก place ให้เข้ามาสู่ transition
2. ช่วง firing เป็นช่วงที่ transition จะเปลี่ยน status จาก on เป็น off หรือ มีการส่ง token จาก transition ไปสู่ place

การ simulate จะเริ่มจากช่วง enable ก่อน และจะมีการเปลี่ยนช่วงเมื่อมีเงื่อนไขในการเปลี่ยน

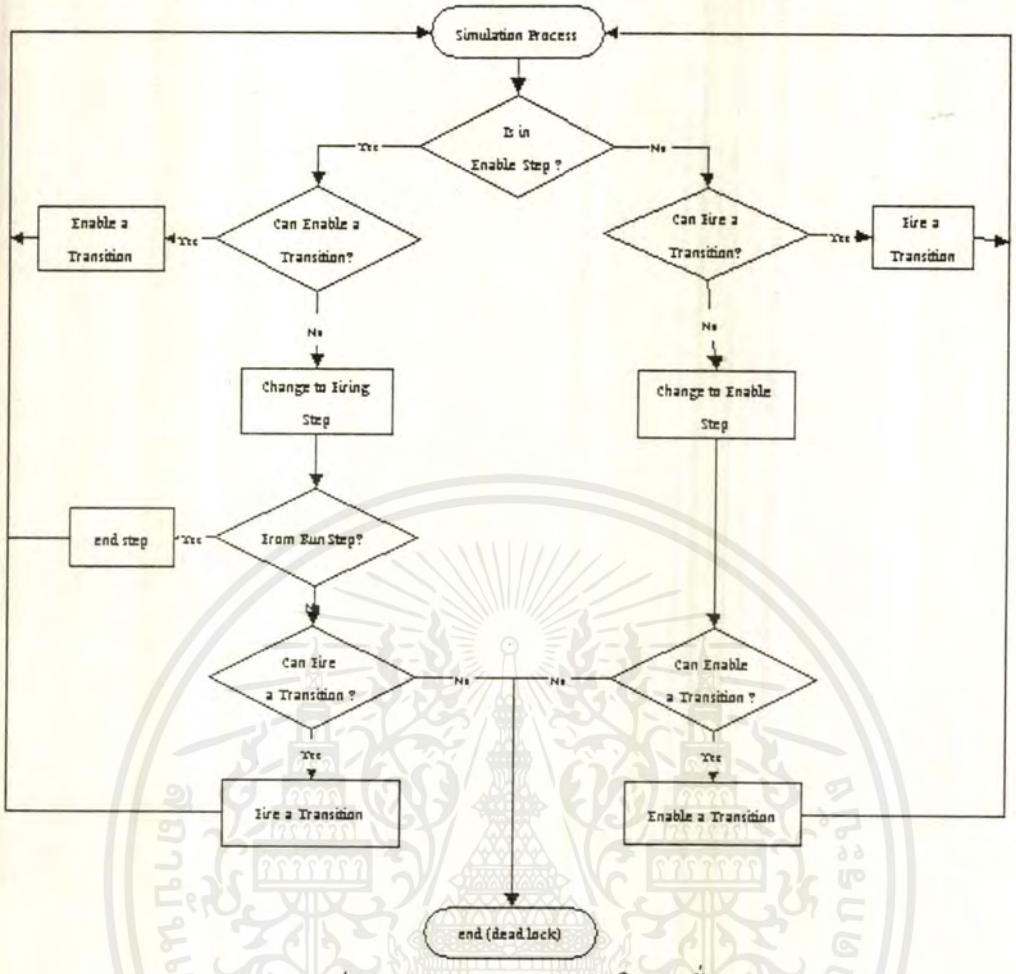
ช่วง enable ของ transition ที่อยู่ใน Waiting Queue มีขั้นตอนดังนี้

1. เรียงลำดับของ Transition ใน Queue ตามการ random และ ค่า priority & probability ของ transition แต่ละตัว
2. เช็คการ enable ของ transition ตามกฎการ firing ถ้า transition นั้นสามารถ enable ได้ก็เปลี่ยน status ของ transition นั้นให้เป็น on แล้วใส่ไว้ใน Active Queue
3. ทำตามข้อ 2 สำหรับ transition ทุกตัวใน Waiting Queue หยุดเมื่อไม่มี transition ตัวใดสามารถ enable ได้อีกหรือหมด transition ใน queue แล้วเปลี่ยนให้เข้าสู่ช่วง firing

ช่วง firing ของ transition ที่อยู่ใน Active Queue มีขั้นตอนดังนี้

1. เรียงลำดับของ transition ใน queue ตามค่าของ parameter1 ซึ่งเป็นค่าที่บอก เวลาที่ใช้ไปสำหรับการ fire ของ transition ตัวนั้น ค่า time step ของ transition จะเท่ากับ ค่า time step ของ ระบบ รวมกับค่า parameter 1 ของ transition นั้น
2. ค่า time step ของ ระบบ จะเท่ากับ ค่า time step ของ transition ตัวหน้าใน Active Queue
3. ทำตามข้อ 2 สำหรับ transition ทุกตัวที่อยู่ใน Active Queue หยุดเมื่อค่า parameter 1 ของ transition ตัวต่อไปไม่เท่าเดิมหรือ มีค่ามากกว่าตัว ปัจจุบัน หรือหมด transition ใน queue แล้วเปลี่ยนให้เข้าสู่ช่วง enable

การ simulation สามารถอธบายด้วย flow chart ของ simulation Process ได้ดังรูป




*รูป3-6 flow chart แสดงการ ซิมมูเลชั่น*

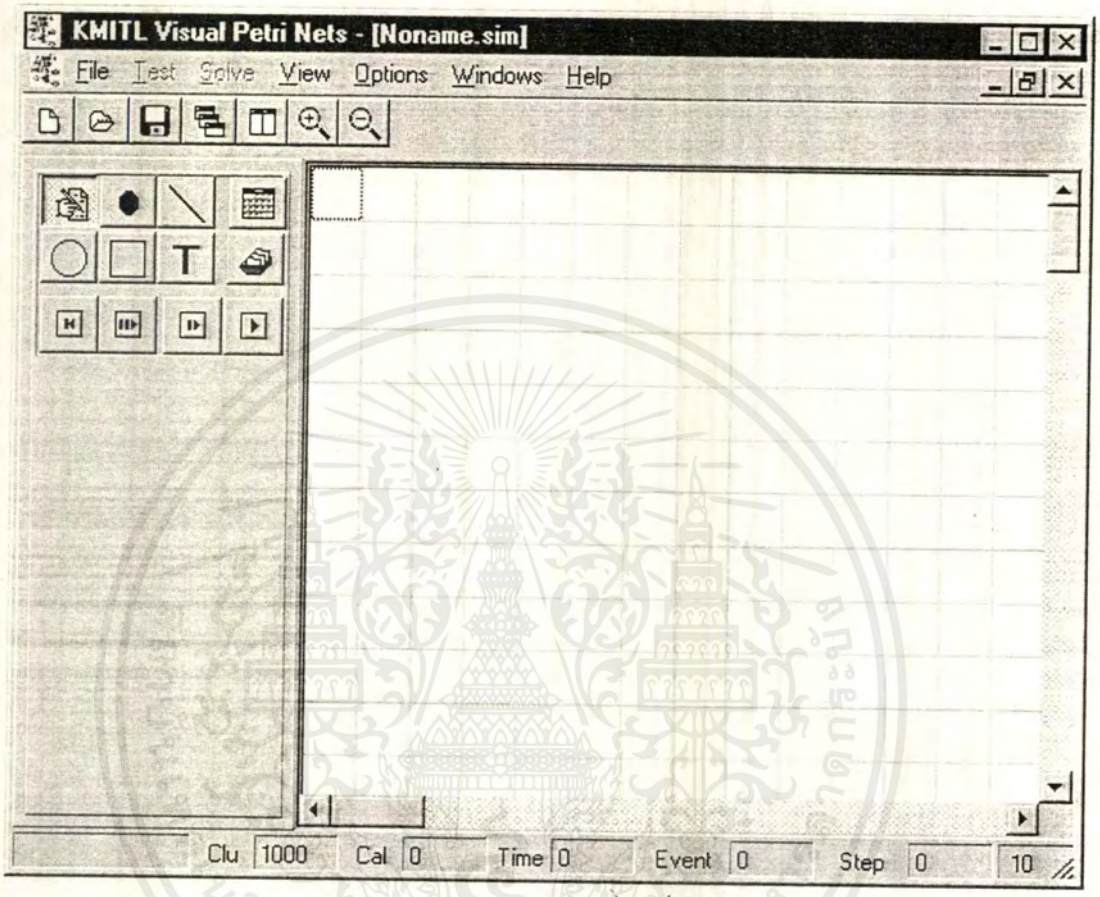
เริ่มต้นจะเช็คว่าอยู่ในช่วง enable หรือไม่ ถ้าใช่ก็มาทำการเช็คในฝั่ง enable โดยเริ่มจากการเช็คว่ามี transition ใด ๆ ใน Waiting Queue ที่สามารถ enable ได้หรือไม่ ถ้าได้ ก็จะ enable transition ตัวนั้นและให้เข้ามาอยู่ใน Active Queue ถ้าไม่ได้ก็เปลี่ยนค่าตัวแปรที่ระบุช่วงให้เป็นช่วง firing แล้วเช็คต่อไปว่าเป็นการรันจากการ run step หรือไม่ ถ้าใช่ก็จบ step แล้ว เข้าสู่รอบของการเช็คใหม่ต่อไป ถ้าไม่ใช่การ run step ก็จะมาเช็คต่อไปว่ามี transition ใด ใน Active Queue ที่สามารถ fire ได้หรือไม่ ถ้ามีก็แสดงว่าเกิด deadlock และจบการทำงาน แต่ถ้ามีก็ทำงาน fire transition ตัวนั้น แล้วให้ออกจาก Active Queue และเข้าสู่รอบการทำงานต่อไป

ในฝั่งของการ firing เริ่มต้นก็จะเช็คว่ามี transition ใดใน Active Queue ที่ fire ได้หรือไม่ ถ้าได้ก็จะ fire transition ตัวนั้นแล้วเอาออกจาก Active Queue ให้อยู่ใน Waiting queue แต่ถ้าไม่ได้ก็เปลี่ยนให้เป็นช่วง enable แล้วเช็คว่ามี transition ใดใน Waiting Queue ที่สามารถ enable ได้หรือไม่ถ้าไม่ก็แสดงว่าเกิด deadlock และจบการทำงาน แต่ถ้ามีก็ทำการ enable transition ตัวนั้นแล้วย้าย transition ตัวนั้นมาใส่ใน Active Queue แล้วเข้าสู่รอบการทำงานต่อไป

## 3.3. เกี่ยวกับโปรแกรมและการใช้งาน

### 3.3.1. ส่วนประกอบของโปรแกรม

1) ฟอร์มหลัก




*รูป 3-7 ฟอร์มหลัก*

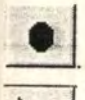
ในฟอร์มหลักได้แบ่งออกเป็นส่วนต่างๆ ดังนี้

☐ ส่วนของการสร้างไดอะแกรม มีด้วยกัน 6 ปุ่ม ดังนี้

 ปุ่มที่ใช้เมื่อต้องการที่จะแก้ไขคุณสมบัติต่างๆของรูป หรือ จัดการเคลื่อนย้ายรูป

 ปุ่มที่ใช้สร้าง Place ในฟอร์ม

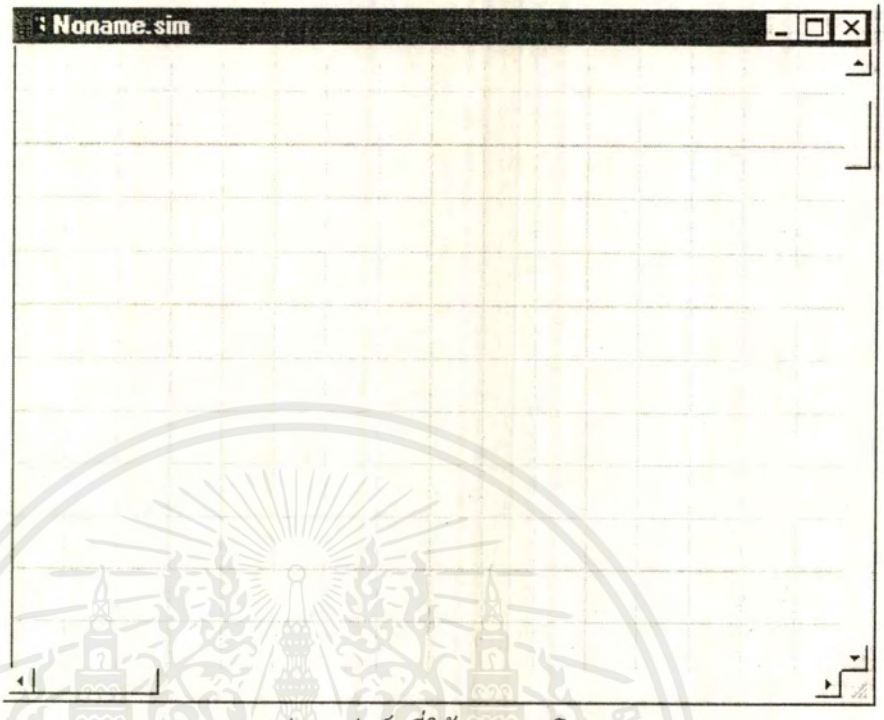
 ปุ่มที่ใช้สร้าง Transition ในฟอร์ม

 ปุ่มที่ใช้สร้าง Token โดยสามารถกำหนดจำนวน Token ได้โดยการ Click ตามจำนวนที่ต้องการ

 ปุ่มที่ใช้สร้าง Arc เชื่อมระหว่าง Place กับ Transition

 ปุ่มที่ใช้สร้างข้อความในฟอร์ม

☐ ฟอร์มที่ใช้ออกแบบโมเดล




*รูป 3-8 ฟอร์มที่ใช้ออกแบบโมเดล*

เป็นฟอร์มที่ใช้วาง diagram ของ Petri Nets โดยการเลือกปุ่มที่จะสร้าง diagram แล้ว คลิ๊กลงไปในแต่ละช่องของตาราง เราสามารถย่อหรือขยายขนาดของของฟอร์มได้และย่อหรือขนาดของกริดได้ด้วยและ มีลักษณะการทำงานแบบ MDI Child Form ของ ฟอร์มหลักคือฟอร์มหลักสามารถเปิด ฟอร์มนี้ขึ้นมาได้หลาย ๆ ฟอร์มพร้อมกัน

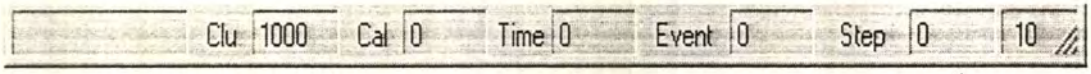
☐ ส่วนในการแสดงผล

ปุ่มที่ใช้เพื่อดูว่าใน Place ที่เราสนใจนั้น มีจำนวน Token อยู่เท่าไร ณเวลาขณะนั้น


*รูป 3-9 ฟอร์มที่ใช้แสดงผลของ Queue*

ปุ่มที่ใช้แสดงว่าจะมีเหตุการณ์ใดกำลังจะเกิดขึ้น ณ เวลาที่เท่าไร


*รูป 3-10 ฟอร์มที่ใช้แสดงผลของ Calendar*




*รูป 3-11 เป็นแถบที่ใช้แสดงสถานะขณะที่ทำการ run อยู่โดยมีรายละเอียดดังนี้*

1. Clu แสดงจำนวน Cluster ที่เหลือสำหรับ Place
2. Cal แสดงจำนวนเหตุการณ์ที่พร้อมที่จะเกิดขึ้น ณ ขณะนั้น
3. Time แสดงเวลาที่ทำอยู่ของเหตุการณ์ ณ ขณะนั้น
4. Event แสดงจำนวนเหตุการณ์ที่ทำไปแล้ว
5. Step แสดงจำนวน Step ที่ทำไปแล้ว
6. ช่องสุดท้ายจะแสดงพิกัดของ mouse ที่ชี้โดยอยู่ในรูปของจำนวนช่อง (x,y)

 ปุ่มเรียก design form ใหม่ขึ้นมา

 ปุ่มให้ทำการเปิดไฟล์ที่ออกแบบไว้แล้ว format file .sim

 ปุ่มให้ทำการเซฟไฟล์ที่ออกแบบไว้ โดยจะเซฟใน format file .sim

 ปุ่มที่ใช้เรียง design form ให้เรียงกันในรูปแบบ cascade

 ปุ่มที่ใช้เรียง design form ให้เรียงกันในรูปแบบ tile verticle

 ปุ่มที่ ใช้ขยายฟอร์มโดยสามารถขยายได้มากที่สุด 2 ครั้ง

 ปุ่มที่ใช้ลดขนาดของฟอร์มโดยสามารถลดขนาดได้มากที่สุด 2 ครั้ง

☐ ส่วนที่ใช้ในการ run

 ปุ่มที่ช้ในการเริ่มต้น run ใหม่

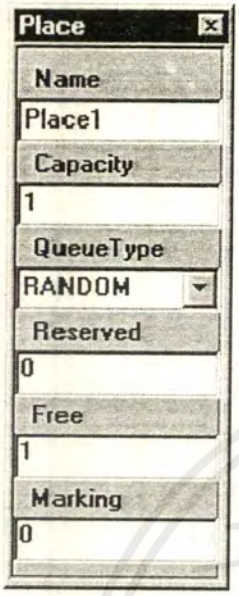
 ปุ่มที่ใช้ในการ run ทีละเหตุการณ์

 ปุ่มที่ใช้ในการ run ทีละ Step ( ซึ่งใน 1 step อาจจะมีหลายเหตุการณ์ได้)

 ปุ่มที่ใช้ในการ run โดยจะแสดงการ firing ในแบบ real time

2) ฟอร์มแสดงคุณสมบัติของ Place , Transition , Arc

☐ Place มีแบบฟอร์มดังนี้




Name : ชื่อของ Place

Capacity : จำนวน Token สูงสุดที่สามารถเก็บได้

Queue Type : ชนิดของการจัดเก็บ Token แบ่งเป็น 3 แบบ

- ☐ LIFO : Last In First Out
- ☐ FIFO : First In First Out
- ☐ RANDOM

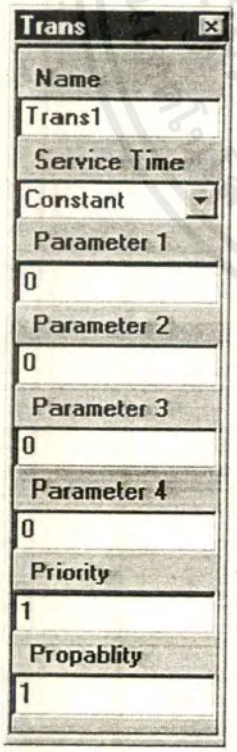
Reserved : จำนวน Token ที่กำลังจะเข้ามาที่ Place

Free : จำนวน Token ที่ยังสามารถเข้ามาได้จะไม่เกิน Capacity

Marking : จำนวน Token ที่มีอยู่ในขณะนั้น

*รูป 3-12 ฟอร์มของ Place*

☐ Transition มีแบบฟอร์มดังนี้




Name : ชื่อของ Transition

Service Time : เป็นฟังก์ชั่นที่ใช้ในการ simulate

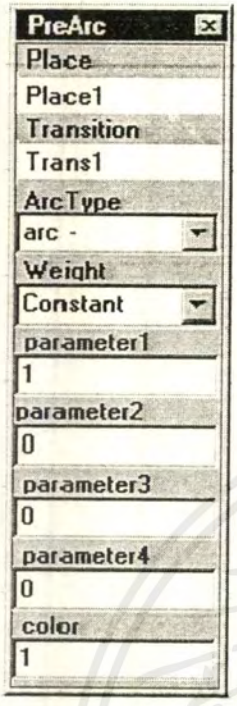
Parameter1 – Parameter4 : เป็นค่าที่ส่งให้ Service Time โดยแต่ละ Service Time ใช้ Parameter ไม่เหมือนกัน

Priority : ใช้กำหนดลำดับความสำคัญของเหตุการณ์หรือ Transition

Probability : ใช้กำหนดความน่าจะเป็นที่จะเกิดเหตุการณ์หรือ Transition

*รูป 3-13 ฟอร์มของ Transition*

☐ Arc มีแบบฟอร์มดังนี้




Transition : บอกชื่อของ Transition ที่ต่ออยู่กับ Arc

Place : บอกชื่อของ Place ที่ต่ออยู่กับ Arc

Arc Type : ใช้บอกเงื่อนไขของการ Firing ระหว่าง Place กับ Transition แบ่งออกเป็น 2 ชนิดคือ PreArc และ PostArc

1. PreArc สามารถแบ่งออกได้อีก 3 ประเภท

   ☐ Test <= , Test = , Test >=

   เช็ค marking ของ input place กับ arc weight ว่าเป็นไปตามเงื่อนไขหรือไม่ ถ้าใช่ Transition จะ Enable

   ☐ Arc -

   ถ้า marking ของ input place มากกว่า weight transition ก็จะ enable และจะลบ marking ของ input place เท่าขนาดของ arc weight

   ☐ Move *

   ถ้า input place มี token มากกว่า arc weight จะลบ Token ใน input place ให้เหลือ 0 แล้วทำการ firing ได้

2. PostArc แบ่งเป็น 3 ประเภท

   ☐ Test <= , Test = , Test >=

   เช็ค free space ของ output place ถ้าเป็นไปตามเงื่อนไขที่ test แล้ว Transition จะ fire ได้

   ☐ Arc -

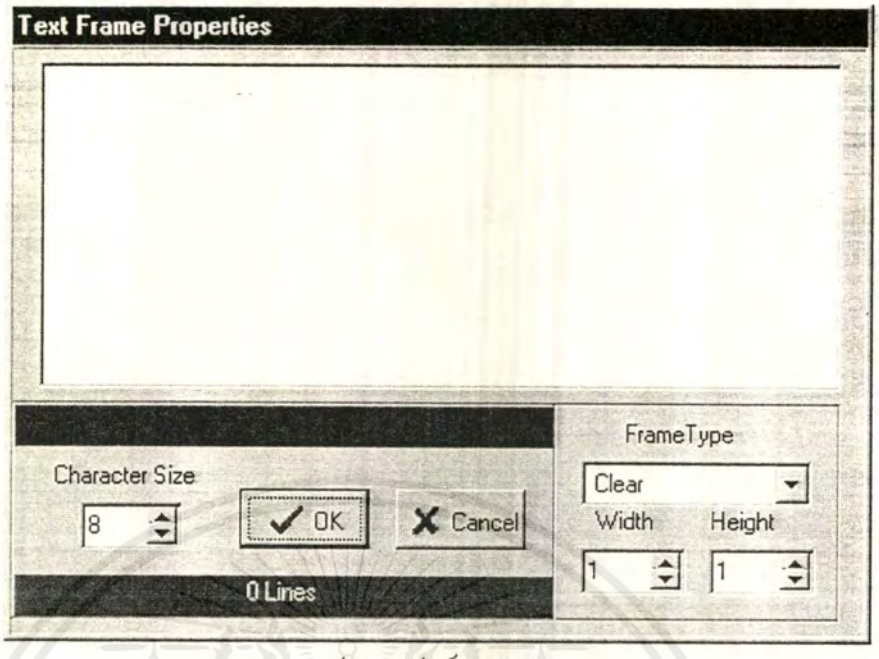
   ถ้า free space ของ output place มากกว่า arc weight จะเพิ่มจำนวน Marking ของ output place ตามค่าใน Parameter 1

   ☐ Move *

   ถ้า weight มากกว่าหรือเท่ากับ free space ของ output place Marking จะเท่ากับ free space ถ้า weight น้อยกว่า free space จะเพิ่ม Marking ตามจำนวนของ weight

*รูป 3-14 ฟอร์มของArc*

3) ฟอร์มที่ใช้สร้างข้อความขึ้นมาบนฟอร์ม




*รูป 3-15 ฟอร์มของText*

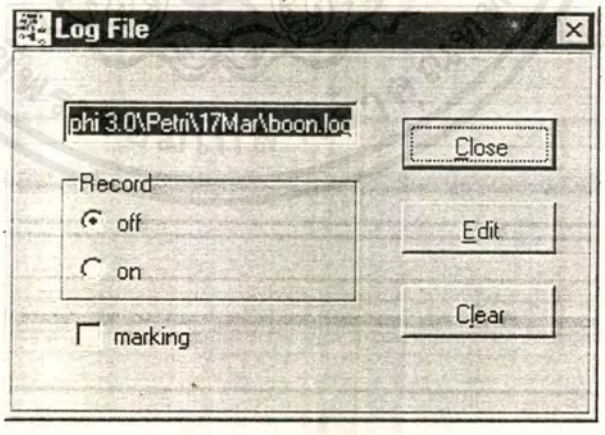
Character Size : กำหนดขนาดของตัวอักษร

Frame Type : กำหนดชนิดของ Border ของข้อความ

Width : กำหนดความกว้างของ Frame Text โดยหน่วยจะเป็นจำนวนช่อง

Height : กำหนดความสูงของ Frame Text โดยหน่วยจะเป็นจำนวนช่อง

4) Log Form : ฟอร์มที่ใช้สร้าง logfile ซึ่งจะเป็น file ที่แสดงการทำงานในแต่ละ event ว่าอยู่ใน step ไหน เวลาที่ใช้ทำ event เป็นเท่าไร รวมถึงยังแสดงด้วยว่าในแต่ละ event นั้นมี Token อยู่ในแต่ละ Place เท่าไร

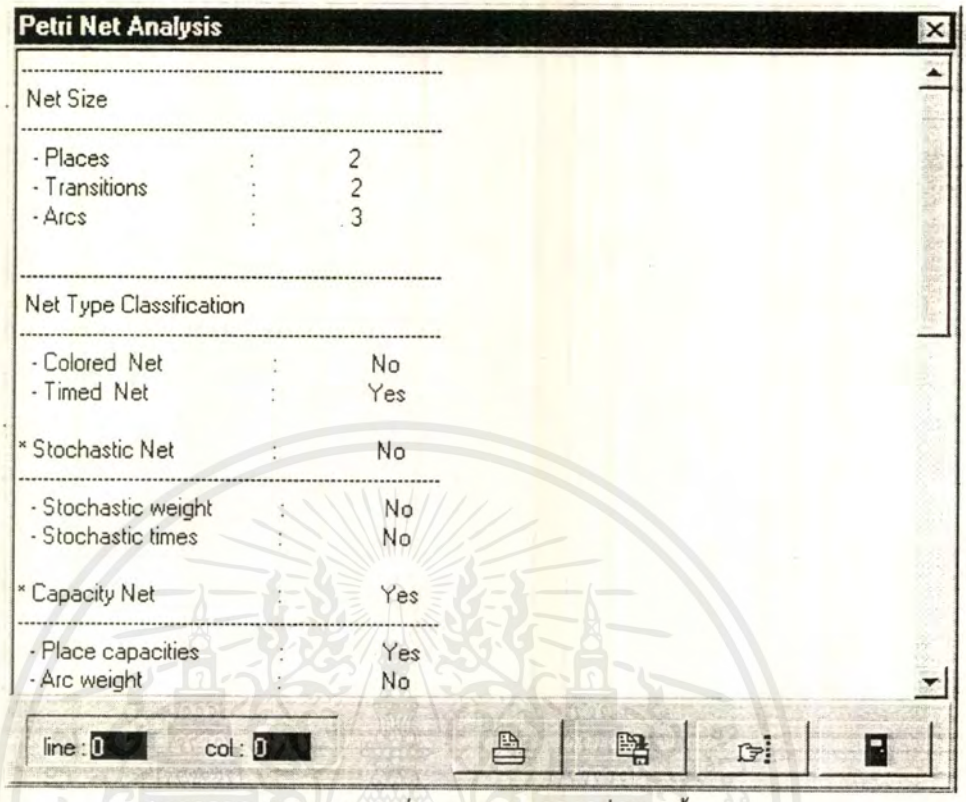



*รูป 3-16 ฟอร์มของ Log file*

☐ on/off จะเป็นตัวกำหนดว่าจะให้มีการแสดงผลลงไปใน log file หรือเปล่า

☐ Edit จะแสดง file ของ log file

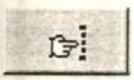
5) Reachability form :




*รูป 3-17 ฟอร์มที่แสดงผลของ net ที่สร้างขึ้นมา*

ปุ่มที่ใช้ทำการ print ผลที่แสดง

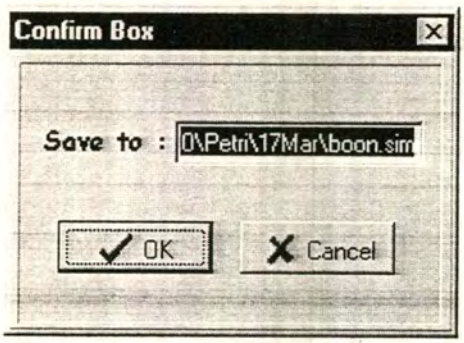
ปุ่มที่ใช้ทำการ save เป็น file ออกมา เมื่อกดแล้วจะมี confirm box ขึ้นมายืนยันว่าจะ save ลงใน file ไหน



ปุ่มที่ใช้หาข้อความที่ต้องการภายใน file ที่แสดงผลอยู่

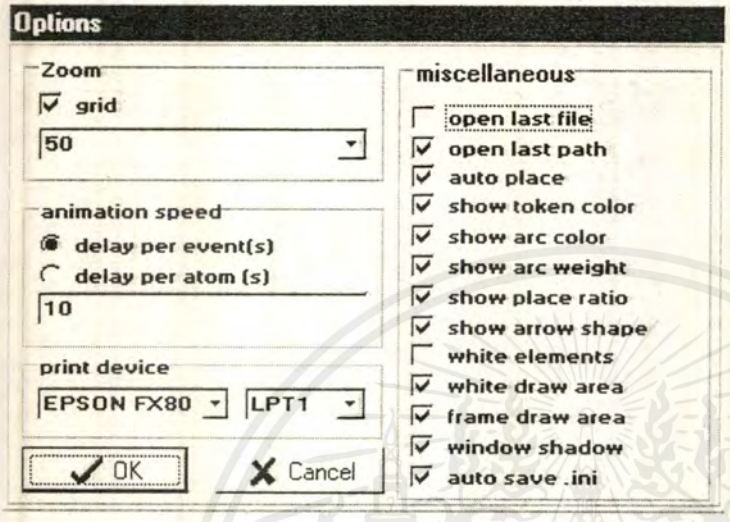
ปุ่มที่ใช้เพื่อออกจาก Form Edit

☐ Confirm Box : ทำการยืนยันว่าจะ save ลง file ไหนโดยสามารถเปลี่ยนชื่อ file ได้




*รูป 3-18 Confirm Box*

6) Option form :




*รูป 3-19 option form*

- ☐ Zoom : เป็นการกำหนดขนาดของ grid
- ☐ Delay / event(s) : เป็นการกำหนดความเร็วในการrun ของเหตุการณ์
- ☐ Delay/ atom(s) : เป็นการกำหนดระยะเวลาระหว่าง atom ในการ run
- ☐ open last file : ให้มีการเก็บ file สุดท้ายไว้เมื่อ open ครั้งต่อไปจะทำการเปิด file นี้
- ☐ open last path : ให้มีการเก็บ path สุดท้ายที่มีการใช้งานไว้ เพื่อการ open ครั้งต่อไป
- ☐ show token color : ให้มีการแสดงสีของ Token เมื่อมี Token สีอื่นที่ไม่ใช่สีดำ
- ☐ show arc color : ให้มีการแสดงสีของ Arc เมื่อมีการกำหนดสีให้ Arc นอกจากสีดำ
- ☐ show arc weight : ให้มีการแสดง weight ของ Arc ขึ้นบนฟอร์ม
- ☐ show place ratio : ให้มีการแสดงอัตราส่วนระหว่างจำนวน token กับ capacity บน Place
- ☐ show arrow shape : ให้มีการแสดงหัวลูกศรของ Arc
- ☐ white elements : ให้ element (place, transition) มีสีขาว
- ☐ white draw area : ให้ draw area มีสีขาว
- ☐ frame draw area : ให้ draw area มีลักษณะเป็น frame
- ☐ window shadow : ให้มีการแสดงเงาเมื่อมีฟอร์มย่อยหรือเมนูบาร์เกิดขึ้นบนฟอร์ม
- ☐ auto save.ini : กำหนดให้มีการ save option โดยอัตโนมัติ

### 3.3.2 รูปแบบไฟล์ที่ใช้ simulate

รูปแบบของภาษาที่ไฟล์ .sim ที่จะเซพเพื่อบันทึกค่าพร็อพเพอตี้ของแต่ละออปเจ็กในโมเดลที่ออกแบบและแอพพลิเคชั่นจะสามารถเปิดไฟล์นี้ขึ้นมาได้มีดังนี้

นิยามของ place

p < placename> < capacity > < waiting discipline > [ x < x_pos > y < y_pos > ]

- placename : ชื่อของ place
- capacity : ความจุสูงสุดของ Place ที่จะสามารถเก็บ Token ได้
- waiting discipline : ชนิดของคิวที่ Token จะเข้ามารอมีดังนี้
  1) FIFO : แบบที่เข้ามาก่อนออกก่อน
  2) LIFO : แบบที่เข้ามาทีหลังออกก่อน
  3) RANDOM : แบบที่ใครจะออกก่อนก็ได้
- x_pos : พิกัดของ Place โดยจะระบุเป็นจำนวนช่องทางแนวแกน x
- y_pos : พิกัดของ Place โดยจะระบุเป็นจำนวนช่องทางแนวแกน y

นิยามของ Token ที่จะเก็บอยู่ใน place

m < color > < number > [< color > < number > ... ]

- color : การระบุสี
- number : จำนวนของ Token ต่อ cluster

นิยามของ Transition

t < name > <distribution > [ < priority > [ < probability >] ] [ x < x_pos> y <y_pos>]

- name : ชื่อของ Transition
- distribution : เป็นservice time ( ฟังก์ชันที่ใช้ในการfiring Token )
- priority : เป็นลำดับความสำคัญของ Transition
- probability : ความน่าจะเป็นของการเกิด enable เมื่อมีเหตุการณ์หลายเหตุการณ์ที่จะเกิดพร้อมกัน
- x_pos : พิกัดของ transition โดยจะระบุเป็นจำนวนช่องทางแนวแกน x
- y_pos : พิกัดของ transition โดยจะระบุเป็นจำนวนช่องทางแนวแกน y

นิยามของ pre-arc ( จาก place ไป transition )

v < placename > <distribution > <color > [ < arctype >]

- placename : ชื่อของ place ที่ arc ต่ออยู่
- distribution : เป็นจำนวนของ Token ที่จะผ่านโดยจะขึ้นอยู่กับฟังก์ชันที่เลือก
- color : การกำหนดสีของ Token ที่จะผ่าน arc ตัวนี้

- arctype : เป็นเงื่อนไขที่จะใช้ในการ firing token โดยจะมีดังนี้
  1) < : จำนวนของ token ที่จะผ่าน < = เนื้อที่ที่ว่าง
  2) = : จำนวนของ token ที่จะผ่าน = เนื้อที่ที่ว่าง
  3) > : จำนวนของ token ที่จะผ่าน >= เนื้อที่ที่ว่าง
  4) - : จำนวนของ token ที่จะผ่าน < = เนื้อที่ที่ว่าง
  5) * : ไม่มีเงื่อนไขในการส่งผ่าน token

นิยามของ textframe .

c < lines> [ <charsize > [ < frametype > [ < f_width> < f_hight > ]]] [ x< x_pos> y< y_pos >]

message

message

...

message

- lines : จำนวนบรรทัดของข้อความ
- charsize : ขนาดของตัวอักษร
- frametype : ชนิดขอบของข้อความ
  0 = None
  1 = Dash
  2 = DashDot
  3 = DashDotDot
  4 = Dot
  5 = InSideFrame
  6 = Solid
- f_width : ความกว้างของ frame ข้อความ
- f_hight : ความสูงของ frame ข้อความ
- x_pos : พิกัดของข้อความตามแนวแกน x
- y_pos : พิกัดของข้อความตามแนวแกน y

### 3.3.3. การใช้งานของแอพพลิเคชั่น

แบ่งหัวข้อหลักได้ดังนี้เป็น

**ก. การออกแบบโมเดล** ทำได้ 2 วิธีคือ

1. การโหลดโมเดลที่มีอยู่แล้วมาใช้งานซึ่งโมเดลที่อยู่แล้วจะอยู่ในรูปแบบของเท็กซ์ไฟล์ที่มีส่วนขยายเป็น .sim

วิธีการโหลดโมเดลก็มีขั้นตอนง่าย ๆ ดังนี้

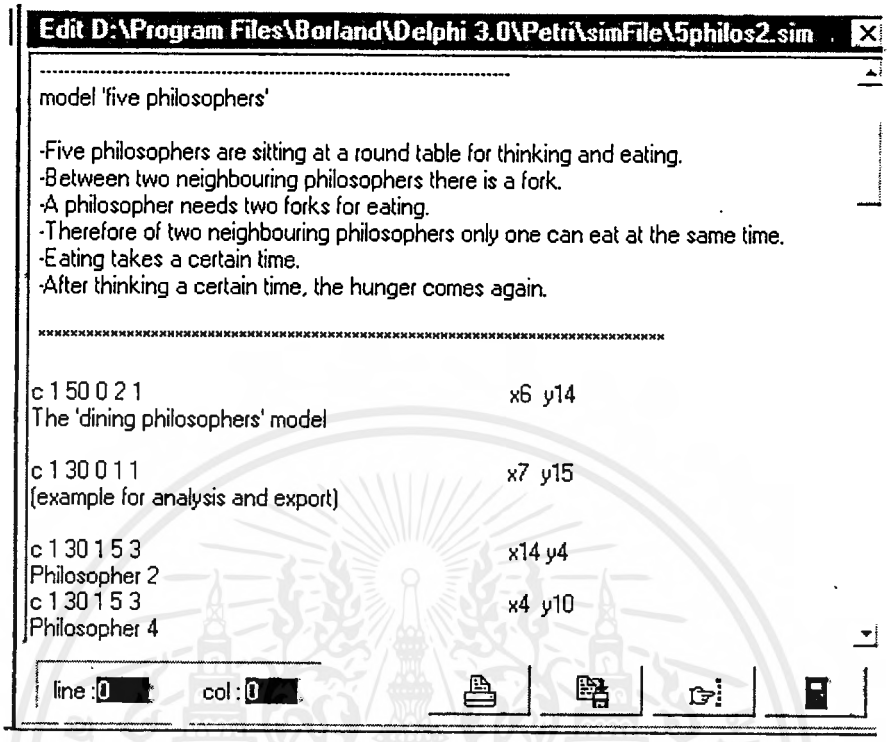
หลังจากเรียกแอพพลิเคชั่นแล้วให้คลิ๊กที่ปุ่ม open หรือ จากเมนู file, open ก็จะปรากฏ dialog ของ open form ขึ้นมาให้เลือกไฟล์ . sim ที่จะเปิด

2. การสร้างโมเดลใหม่และเราก็สามารถที่จะจัดเก็บหรือเซฟไฟล์ที่สร้างนี้เพื่อการโหลดต่อไปได้ด้วย

วิธีการสร้างโมเดลใหม่มีขั้นตอนดังนี้

- คลิ๊กที่ปุ่ม new หรือ จากเมนู file, new ก็จะปรากฏฟอร์มใหม่ที่มีลักษณะเป็นตารางเพื่อใช้ในการออกแบบ
- นึกถึงโมเดลที่ต้องการจะออกแบบโดยใช้ ปุ่มทั้ง 6 ที่เกี่ยวกับการออกแบบ เช่นเมื่อต้องการใช้ diagram ของ transition ก็ลากเมาส์ไปคลิ๊กที่ปุ่ม transition เป็นการเปลี่ยนโหมดการออกแบบให้เป็นการสร้าง transition แล้ว ลากเมาส์ไปคลิ๊กบนตำแหน่งที่ต้องการจะออกแบบในฟอร์มที่ต้องการจะออกแบบนี้
- หลังจากวางไดอะแกรมบนฟอร์มที่ออกแบบ แล้วเราสามารถจัดการกับพร๊อพเพอร์ตี้ของมันได้ด้วยการเมาส์ไปคลิ๊กที่ปุ่ม edit เพื่อเปลี่ยนโหมดให้เป็นการแก้ไข แล้วคลิ๊กขวาบนตำแหน่งของ object ที่ต้องการจะแก้ไขพร๊อพเพอร์ตี้ ก็จะปรากฏฟอร์มที่ใช้จัดการกับพร๊อพเพอร์ตี้ขึ้นมาให้สามารถทำการแก้ไขได้

โมเดลที่ออกแบบมานี้สามารถดูได้จากเมนู File, Edit Text




*รูป 3-20 Edit Text ของไฟล์ .sim*

**ข. การแสดงผลการทำงานของโมเดล (simulation)**

ประกอบด้วย 4 ปุ่มที่ใช้ควบคุมการทำงานของโมเดล ได้แก่

ปุ่มรีเซ็ต ให้เริ่มการทำงานใหม่ตั้งแต่ต้น

ปุ่มรันอีเวนต์ ให้ทำการแสดงการทำงานทีละอีเวนต์

ปุ่มรันเสต็ป ให้ทำการแสดงการทำงานทีละเสต็ป

ปุ่มรันต่อเนื่องกันไปทีละเสต็ป

ปุ่มแสดงคิวของ place

ปุ่มแสดง calendar หรือเหตุการณ์ที่เกิดขึ้น

**ค.** การวิเคราะห์โมเดลที่ออกแบบ โดยดูได้จาก Reachability Form ใน menu Solve, Reachability ข้อมูลที่วิเคราะห์ได้นั้นได้แก่

Net Size

- Places : 15
- Transitions : 10
- Arcs : 40

Net Type Classification

- Colored Net : No
- Timed Net : No
* Stochastic Net : No
- Stochastic weight : No
- Stochastic times : No
* Capacity Net : Yes
- Place capacities : Yes
- Arc weight : No
* Priority Net : No
- Transition priorities : No
- Transition probabilities : No
* Enhanced Arcs : No
- Other test arcs : No
- Move arcs : No

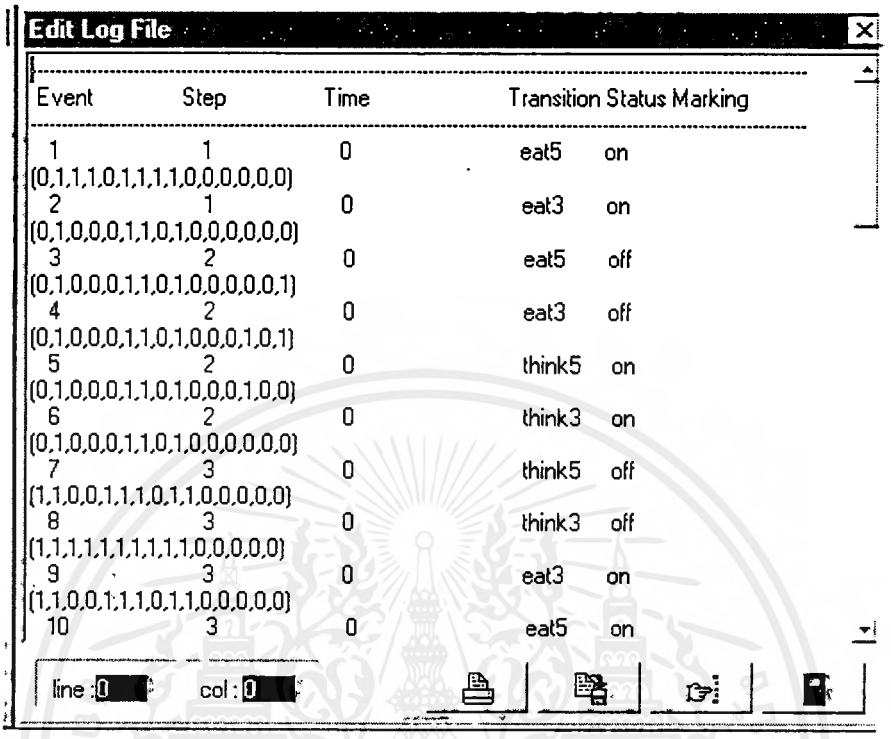
* Enhanced Structures

- Pre-arcless places : No
- Post-arcless places : No
- Pre-arcless transitions : No
- Post-arcless transitions : No
- Parallel arcs : No

Net Analysis

* Dead Structures : No
- Arcless places : No
- Arcless transitions : No
- 0-weight standard arcs : No
- 0-weight move arcs : No

**ง. การเก็บข้อมูลที่ได้แสดงการทำงานที่ผ่านมา ใน** Log Form ใน menu Test, Log File จะปรากฏ log form ที่จัดการเกี่ยวกับ log file ได้โดยการ คลิ๊กที่ปุ่ม edit จะแสดง log file ขึ้นมา




*รูป 3-21 Log File Form*

# บทที่ 4
# ประเมินผลการทำงาน

**4.1. ข้อกำหนดการประเมิน**

การประเมินผลการทำงานนี้จะใช้วิธิการนับจาก Source Code ที่ได้ ( Code Size Estimation ) และการนับขนาดของ Code นี้จะใช้วิธีนับจาก Lines of code โดยมีข้อกำหนดเงื่อนไขสำหรับการนับดังนี้

1. การนับจะนับเฉพาะ Module ที่เกิดจากการเขียนเองเท่านั้น ไม่นับ Module ที่ได้จากการ install จาก component หรือ package ที่มีอยู่แล้วของเดลไฟล์ และ/หรือ การดาวน์โหลดมา
2. lines of code ไม่นับ ชื่อ unit, ส่วนของ uses, การกำหนด type, declare private, declare public, declare var, implementation, ชื่อโพรซีเยอร์, ชื่อฟังก์ชัน เป็นต้น นับเฉพาะในส่วนที่เป็น code ของคำสั่งการทำงาน เท่านั้น
3. การประเมินนี้ถ้าเป็นส่วนที่เขียนเสร็จแล้ว จะถือว่านับได้เลย และหากเป็นส่วนที่ยังไม่ได้เขียน จะใช้การประเมินโดยการพิจารณาเอาจากโพรซีเยอร์หรือ ฟังก์ชั่นที่ต้องมีแล้วกะประมาณไว้
4. Code Size ที่ได้นี้ยอมรับได้ว่าอาจจะไม่ถูกต้องแน่นอน เพราะได้จากการประเมินเท่านั้น เพื่อเป็นส่วนหนึ่งของการพิจารณาหาข้อสรุปความก้าวหน้าของการทำงาน

**4.2. ผลที่ได้จากการประเมินมีดังนี้**

| Module(Unit Name) | จำนวน Objects | จำนวน Methods | ชื่อ Method ที่สำคัญ ( จำนวนบรรทัด ) |
|---|---|---|---|
| Main | 27 | 42 | UpDown1Click(18)<br>FormResize(18)<br>Open1Click(32)<br>Save1Click(14)<br>ChangeTextValue(35)<br>SbZoomInClick(12)<br>SbZoomOutClick(12) |
| tst2 | 3 | 84 | FormCreate(25)<br>DrawGrid1MouseDown(54)<br>PlaceOnClick(22)<br>TransOnClick(23)<br>ArcMouseDown(36)<br>Calline(77)<br>Open(190)<br>DrawGrid1Keydown(67)<br>Fire1Event(49)<br>FirePtoT(54)<br>FireTtoP(46) |
| TrForm | 17 | 10 | TrNameChange(3) |
| Pform | 16 | 5 | PlnameChange(3) |
| Arcform | 19 | 9 | CbArcTypechange(4) |
| TstAbout | 11 | 1 | FormCreate(3) |
| TextForm | 18 | 8 | bbokclick(24)<br>Formshow(15) |
| OptionForm | 32 | 13 | checkinFile(54)<br>Optioninitial(54) |

*ตาราง 4-1 ผลที่ได้จากการประเมินผลงาน*

| Module(Unit Name) | จำนวน Objects | จำนวน Methods | ชื่อ Method ที่สำคัญ ( จำนวนบรรทัด ) |
|---|---|---|---|
| EditForm | 12 | 19 | EditfindFind(16)<br>FindNetTyp(74) |
| LogForm | 6 | 3 | bbCloseClick(1)<br>bbEditClick(5)<br>bbClearClick(3) |
| calendar | 34 | 4 | Showevent(52) |
| queue | 30 | 4 | Compobox1change (24) |
| ConfirmFm | 6 | 0 | - |
| Line | 0 | 12 | Paint(151) |
| Trans | 0 | 10 | Create(11) |
| Place | 0 | 20 | Paint(18)<br>PaintAsPie(28) |
| TxtFrame | 0 | 11 | Draw3Dtext(39) |

*ตาราง 4-1 ผลที่ได้จากการประเมินผลงาน(ต่อ)*

สรุปการประเมินผลการทำงาน source code ประกอบด้วย

1) มี 17 Module

2) มี 24 Class

3) มี 246 Method

4) จำนวนบรรทัดรวม = 4347 บรรทัด

จากผลที่ได้เราจะประมาณได้ว่า

- ☐ 1 Module จะมีประมาณ 15 Method
- ☐ 1 Method จะมีประมาณ 17.6 บรรทัด

# บทที่ 5
# บทวิจารณ์และสรุป

## 5.1. สรุปผลงาน

โดยสรุปแล้วความสามารถของแอพพลิเคชันนี้ที่สามารถทำได้มีดังนี้

1. การออกแบบโมเดลของ Petri Nets โดยรูปของคอมโพเนนต์ที่สามารถสร้างขึ้นในโมเดลนี้ได้แก่ place, transition, token, arc และ text และมี ฟอร์มที่จะจัดการเกี่ยวกับ พรอพเพอร์ตี้ของแต่ละคอมโพเนนต์ ฟอร์มที่ใช้ออกแบบมีการทำงานแบบ MDI Form หรือ ฟอร์มชนิด Multiple-Document Interface ทำให้สามารถเปิดไฟล์โมเดล (.sim) ขึ้นมาทำงานได้พร้อม ๆ กันหลาย ๆ โมเดล รวมทั้งยังสามารถเซฟและเปิดไฟล์ที่ออกแบบไว้แล้วได้ด้วย และการปรับขนาดของโมเดลหรือการซูม เพื่อดูขนาดได้ 3 ระดับ คือ 25/50/75 pixel
2. การแสดงการทำงานของโมเดล (simulation) สามารถทำการ Reset, รันทีละ Event, รันทีละ Step, และรันแบบต่อเนื่อง ในการแสดงผลที่สามารถทำได้ตามทฤษฎีในระดับของ basic time Petri Nets และ Colored Petri Nets, มี status bar แสดงสถานะเกี่ยวกับการ simulate ต่าง ๆ ของ โมเดลได้แก่ Clu (Cluster), Cal(Calendar), Time, Event, Step และตำแหน่งของเคอร์เซอร์ใน Drawgrid และมีฟอร์มที่ใช้ในการแสดงผล ได้แก่ Calendar และ Queue
3. การวิเคราะห์โมเดลโดยเก็บไว้ใน Reachability Form
4. การสร้าง log file เพื่อบันทึกผลที่ได้จากการ simulate ว่าในแต่ละ event อยู่ใน step ไหน ทำใน time ที่เท่าไร ชื่อของ event หรือ transition ที่ทำและสามารถเลือกว่าจะให้แสดง marking ของ แต่ละ place ด้วยหรือไม่ก็ได้ และ log file นี้สามารถทำการเซฟลงไฟล์ .log ได้

## 5.2. แนวทางการพัฒนาต่อ

เนื่องจาก Petri Nets ยังมีส่วนที่ให้ศึกษาต่อไปอีกมากมายนักดังนั้น แอพพลิเคชั่นนี้ สามารถได้รับการศึกษาและพัฒนาไปอีกได้ดังนี้

1. แอพพลิเคชั่นที่ได้สามารถทำงานได้ในระดับ basic time Petri Nets ที่สามารถทำงานได้ถึงระดับของการนำเวลาเข้ามาเกี่ยวข้องและการวิเคราะห์เกี่ยวกับ serviec time ทำได้เพียงค่าที่เป็น constant แต่ยังขาดในส่วนของการวิเคราะห์เกี่ยวกับ service time และประเภทอื่น ๆ อีก
2. การนำ Scope มาช่วยในการแสดงผลทำงานเพื่อแสดงค่าต่าง ๆ ของ Place เช่น arrival distance, passage distance, loss distance, waiting time และ queuelength และของ transition เช่น service distance, service time, idle time และ activity เป็นต้น โดยให้อยู่ในรูปแบบของกราฟที่คล้ายกับการการใช้ออสซิลโลสโคป (ossilloscope)

3. การจัดเก็บ (Save) รูปของโมเดลให้อยู่ในรูปแบบของไฟล์รูปภาพเช่น .WMF, .BMP, .JPG หรือ .TIF เป็นต้น เพื่อประโยชน์ของการนำโมเดลนี้ไปใช้กับงานอื่นต่อไป

### 5.3. แนวทางการประยุกต์ใช้

สิ่งที่สามารถพัฒนาต่อไปสำหรับการใช้ Petri Nets นี้ได้

1. การนำไปใช้เป็นแอพพลิเคชั่นที่จะช่วยผู้บริหารในการตัดใจเกี่ยวกับการทำงานของระบบ เช่น ตัวอย่างของอัตราการทำงานของแต่ละหน่วยในระบบที่มีประสิทธิภาพในการทำงานที่ต่างกัน ในกรณีที่ผู้บริหารไม่มีมาตราการในการวัดหน่วยทำงานได้ว่าที่ไหนหรือใครทำงานเก่งได้อย่างชัดเจน ทำให้ผู้บริหารไม่มีอะไรที่จะมาอ้างอิงเพื่อประกอบการตัดสินใจที่จะแก้ไขหรือปรับปรุง การนำ Petri Nets มาช่วยก็อาจจะทำโดยแทนแต่ละหน่วยทำงานด้วย Place แล้วอาศัยข้อมูลทางสถิติ ที่วิเคราะห์ได้นี้ ทำให้ผู้บริหารสามารถทำการเปรียบเทียบการทำงานของแต่ละหน่วยทำงานได้ด้วยข้อมูลที่มีการวิเคราะห์มาแล้วจริง
2. นอกจากนี้ยังสามารถนำความรู้ด้าน Petri Nets นี้มาใช้ในงานต่าง ๆ อีกมากมาย ดังตัวอย่างที่มีการศึกษาและประยุกต์ใช้ดังที่จะแสดงไว้ในภาคผนวก ก

### 5.4. สรุปภาพรวมของโครงการ ความสำเร็จ

จากการสร้างแอพพลิเคชั่นนี้ทำให้เราได้ประสบการณ์, ความรู้และความสามารถที่มีประโยชน์ดังนี้

1. ความรู้ทางด้าน Petri Nets เราได้เรียนรู้ทฤษฎีของ Petri Nets การจำลองระบบด้วยโมเดลของ Petri Nets ทฤษฎีการทำงาน การซิมมูเลชั่นของโมเดลเพื่อศึกษาพฤติกรรมของระบบ รวมถึงแนวความคิดในการนำPetri Nets ไปประยุกต์ใช้ในการทำงานด้วย
2. ความรู้ทางด้านเคลไฟ ทักษะการเขียนโปรแกรมแบบวิชวล การเขียนโปรแกรมเชิงวัตถุ ( Object Oriented Programming )
3. รู้จักการทำงานร่วมกันเป็นทีมขั้นตอนการพัฒนาซอฟแวร์อย่างมีระบบแบบแผนตลอดระยะเวลา 1 ปี

แอพพลิเคชั่นที่เราได้ทำขึ้นมาในขณะนี้เป็นเพียงส่วนหนึ่งของการพัฒนาเกี่ยวกับ Petri Nets และยังสามารถได้รับการพัฒนาต่อไปได้อีกตามที่ได้กล่าวไว้ในแนวทางการพัฒนาต่อและแนวทางการประยุกต์ใช้

# ภาคผนวก ก.

## Examples of Industrial Use of CP-nets and Design/CPN

The following papers document large-scale practical use of CP-nets and the Design/CPN tool. Most of the projects have been carried out in an industrial environment.

1.J. Berger, L. Lamontagne: A Colored Petri Net Model for a Naval Command and Control System. In: M. Ajmone-Marsan (ed.): Application and Theory of Petri Nets 1993. Proceedings of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science Vol. 691, Springer-Verlag 1993, 532-541.

2.C. Capellmann, H. Dibold: Petri Net Based Specifications of Services in an Intelligent Network. Experiences Gained from a Test Case Application. In: M. Ajmone-Marsan (ed.): Application and Theory of Petri Nets 1993. Proceedings of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science Vol. 691, Springer-Verlag 1993, 542-551.

3.L. Cherkasova, V. Kotov, T. Rokicki: On Net Modelling of Industrial Size Concurrent Systems. In: M. Ajmone-Marsan (ed.): Application and Theory of Petri Nets 1993. Proceedings of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science Vol. 691, Springer-Verlag 1993, 552-561.

4.L. Cherkasova, V. Kotov, T. Rokicki: On Scalable Net Modeling of OLTP. In PNPM93: Petri Nets and Performance Models. Proceedings of the 5th International Workshop, Toulouse, France 1993, IEEE Computer Society Press, 270-279.

5.S. Christensen, J.B. J?rgensen: Analysing Bang & Olufsen's BeoLink Audio/Video System Using Coloured Petri Nets. Submitted to: G. Balbo and P. Azema (eds.): Application and Theory of Petri Nets 1997. Proceedings of the 18th International Petri Net Conference, Toulouse 1997, Lecture Notes in Computer Science, Springer-Verlag 1997.

6.S. Christensen, L.O. Jepsen: Modelling and Simulation of a Network Management System Using Hierarchical Coloured Petri Nets. In: E. Mosekilde (ed.): Modelling and Simulation 1991. Proceedings

of the 1991 European Simulation Multiconference, Copenhagen, 1991, Society for Computer Simulation 1991, 47-52.

7.H. Clausen, P.R. Jensen: Validation and Performance Analysis of Network Algorithms by Coloured Petri Nets. In PNPM93: Petri Nets and Performance Models. Proceedings of the 5th International Workshop, Toulouse, France 1993, IEEE Computer Society Press, 280-289.

8.H. Clausen, P.R. Jensen: Analysis of Usage Parameter Control Algorithms for ATM Networks. In: S. Tohm? and A. Casaca (eds.): Broadband Communications, II (C-24), Elsevier Science Publishers 1994, 297-310.

9.D.J. Floreani and J. Billington: Designing and Verifying a Communications Gateway Using Colored Petri Nets and Design/CPN. In J. Billington and W. Reisig (eds.): Application and Theory of Petri Nets 1996. Proceedings of the 17th International Petri Net Conference, Osaka 1996, Lecture Notes in Computer Science Vol. 1091, Springer-Verlag 1996, 153-171.

10.H.J. Genrich, R.M. Shapiro: Formal Verification of an Arbiter Cascade. In: K. Jensen (ed.): Application and Theory of Petri Nets 1992. Proceedings of the 13th International Petri Net Conference, Sheffield 1992, Lecture Notes in Computer Science Vol. 616, Springer-Verlag 1992, 205-223.

11.P. Huber, V.O. Pinci: A Formal Executable Specification of the ISDN Basic Rate Interface. Proceedings of the 12th International Conference on Application and Theory of Petri Nets, Aarhus 1991, 1-21.

12.J.B. J?rgensen, L.M. Kristensen: Computer Aided Verification of Lamport's Fast Mutual Exclusion Algorithm Using Coloured Petri Nets and Occurrence Graphs with Symmetries. Computer Science Department, Aarhus University, Denmark. Submitted to IEEE Transactions on Parallel and Distributed Systems, 1996.

13.J.B. J?rgensen and K.H. Mortensen: Modelling and Analysis of Distributed Program Execution in BETA Using Coloured Petri Nets. In J. Billington and W. Reisig (eds.): Application and Theory of Petri Nets 1996.Proceedings of the 17th International Petri Net Conference, Osaka 1996, Lecture Notes in Computer Science Vol. 1091, Springer-Verlag 1996, 249-268.

14.W.W. McLendon, R.F. Vidale: Analysis of an Ada System Using Coloured Petri Nets and Occurrence Graphs. In: K. Jensen (ed.): Application and Theory of Petri Nets 1992. Proceedings of the 13th International Petri Net Conference, Sheffield 1992, Lecture Notes in Computer Science Vol. 616, Springer-Verlag 1992, 384-388.

15.K.H. Mortensen, V.O. Pinci: Modelling the Work Flow of a Nuclear Waste Management Program. In: R. Valette (ed.): Application and Theory of Petri Nets 1994. Proceedings of the 15th International Petri Net Conference, Zaragoza 1994, Lecture Notes in Computer Science Vol. 815, Springer-Verlag 1994, 376-395.

16.V.O. Pinci, R.M. Shapiro: An Integrated Software Development Methodology Based on Hierarchical Colored Petri Nets. In: G. Rozenberg (ed.): Advances in Petri Nets 1991, Lecture Notes in Computer Science Vol. 524, Springer-Verlag 1991, 227-252. Also in K. Jensen, G. Rozenberg (eds.): High-level Petri Nets. Theory and Application. Springer-Verlag, 1991, 649-667.

17.J.L. Rasmussen and M. Singh: Designing a Security System by Means of Coloured Petri Nets. In J. Billington and W. Reisig (eds.): Application and Theory of Petri Nets 1996. Proceedings of the 17th International Petri Net Conference, Osaka 1996, Lecture Notes in Computer Science Vol. 1091, Springer-Verlag 1996, 400-419.

18.G. Scheschonk, M. Timpe: Simulation and Analysis of a Document Storage System. In: R. Valette (ed.): Application and Theory of Petri Nets 1994. Proceedings of the 15th International Petri Net Conference, Zaragoza 1994, Lecture Notes in Computer Science vol. 815, Springer-Verlag 1992, 454-470.

19.R.M. Shapiro: Validation of a VLSI Chip Using Hierarchical Coloured Petri Nets. Journal of Microelectronics and Reliability, Special Issue on Petri Nets, Pergamon Press, 1991. Also in K. Jensen, G. Rozenberg (eds.): High-level Petri Nets. Theory and Application. Springer-Verlag, 1991, 667-687.

<Picture>

---

Last modified: Fri Dec 5 13:24:28 1997 -- CP-nets Webmaster

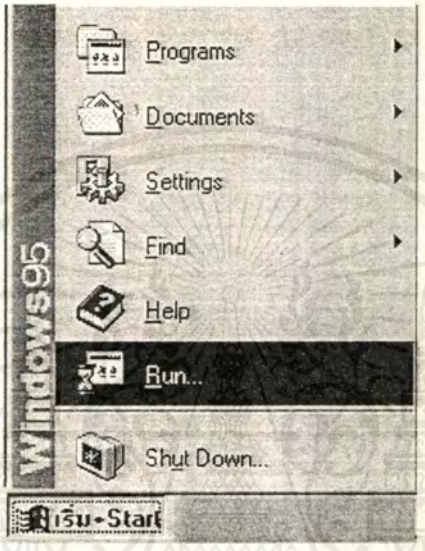
ที่มา http://www.daimi.aau.dk/CPnets/intro/example_indu.html

# ภาคผนวก ข.

# คู่มือการติดตั้งโปรแกรม

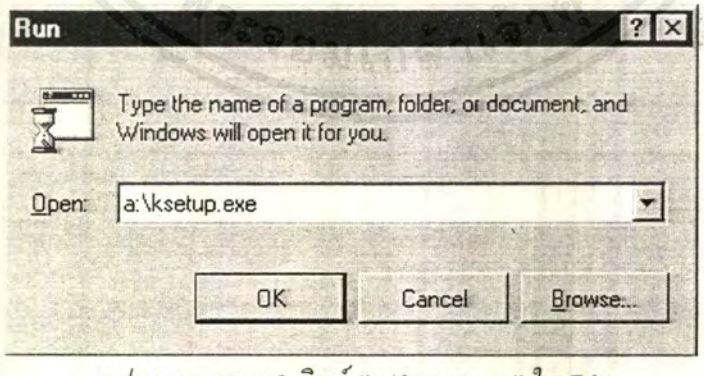
การติดตั้งโปรแกรม ksimnet ทำได้โดยการใส่แผ่นดิสค์ลงในไดรฟ์ A: หลังจากนั้นก็เรียกไฟล์ที่มีชื่อ ksetup.exe ซึ่งผู้อ่านสามารถไฟล์นี้ได้จากตามขั้นตอนดังต่อไปนี้

1. กด "start"
2. เลือก "run"




*รูปภาคผนวก ข-1 กด "start" และ เลือก "run"*

3. ใส่แผ่น setup ลงในไดรฟ์ A:
4. พิมพ์ข้อความใน Edit




*รูปภาคผนวก ข-2 พิมพ์ "a:\ksetup.exe" ใน Edit*

5. เลือกปุ่ม OK

หลังจากที่โปรแกรมเซตอัพถูกอัพโหลดแล้วมันก็จะทำการติดตั้ง โปรแกรมลงในฮาร์ดดิสค์และแสดงไดอะล็อกบ๊อกซ์ต่างๆ ขึ้นเพื่อให้ผู้อ่านได้กรอกหรือตอบคำถามตามลำดับ หลังจากนั้นโปรแกรมเซตอัพก็จะทำการติดตั้งโปรแกรม ksimnet ตามเงื่อนไขที่ผู้อ่านกำหนด

ขั้นตอนต่างๆ ในการติดตั้ง จะแสดงไดอะล็อกบ็อกซ์ให้ผู้อ่านได้ทราบดังนี้

1. โปรแกรมเซตอัพจะแสดงฟอร์ม welcome เพื่อบอกให้ผู้อ่านให้ทราบว่าขณะนี้กำลังอยู่ในระหว่างการติดตั้ง


*รูปภาคผนวก ข-3 ฟอร์ม welcome*

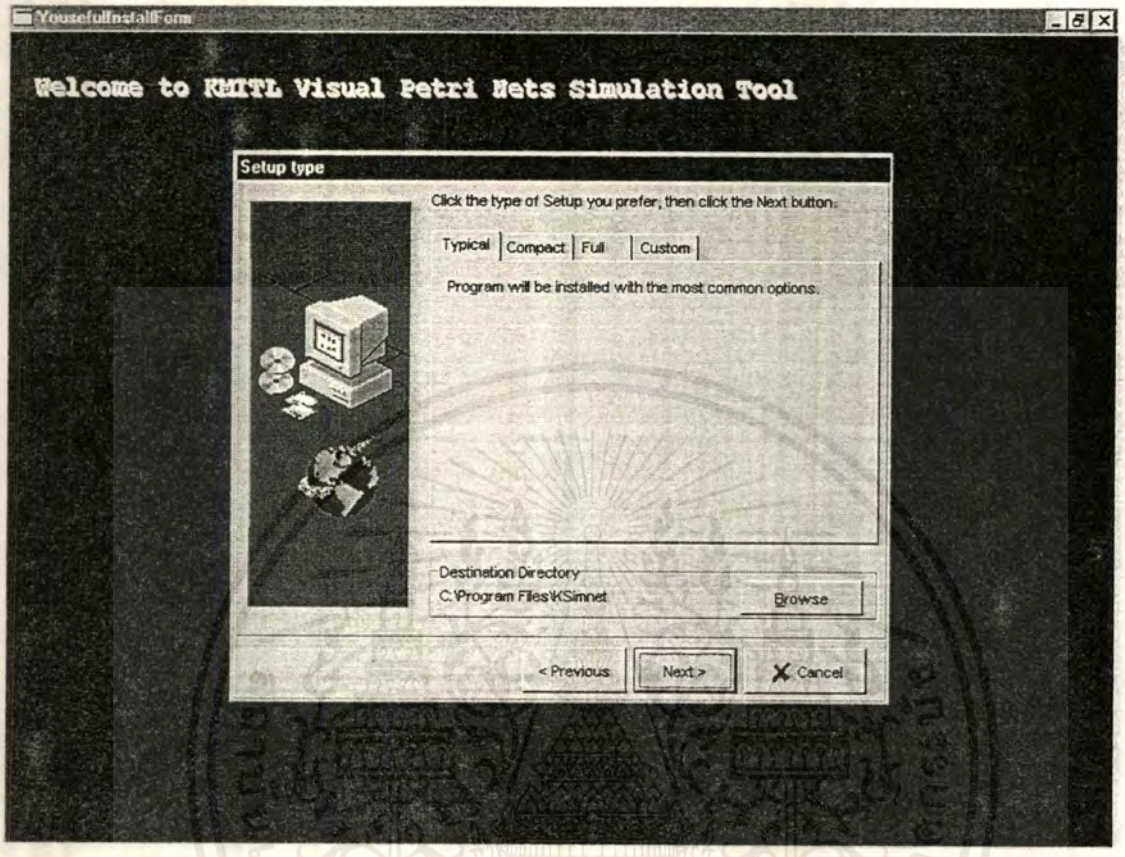
เลือกปุ่ม Next เพื่อทำการติดตั้งต่อไป

เลือกปุ่ม Cancel เพื่อยกเลิกการติดตั้ง

2. ต่อมาจะปรากฏฟอร์ม setup type เพื่อให้ผู้อ่านเลือกวิธีการติดตั้ง ก่อนอื่นจะขอกล่าวถึงไฟล์ทั้งหมดของโปรเจ็กต์ ซึ่งแบ่งเป็นกลุ่ม ดังนี้

   - ksimnet Application ประกอบด้วย ksimnet.exe ที่เป็นแอปพลิเคชั่นที่ใช้รัน และ ksimnet.ini เป็นไฟล์ที่กำหนดค่าออปชั่นตั้งต้นให้กำแอปพลิเคชั่น
   - Sim Files ประกอบด้วยไฟล์โมเดลที่ได้ออกแบบไว้แล้วที่เป็นไฟล์ .sim
   - Source Code ประกอบด้วยไฟล์ซอร์สโค้ดที่เขียนจากเดลไฟล์ เวอร์ชั่น 3 ผู้อ่านสามารถนำซอร์สโค้ดนี้ไปแก้ไขในเดลไฟล์ 3 ได้ และเนื่องจากโปรเจ็กต์นี้มีการสร้างคอมโพเนนท์ขึ้นมาใหม่ 4 ตัว เราสามารถนำคอมโพเนนท์ทั้ง 4 นี้ไปอินสตอลลงคอมโพเนนท์พาเลทของเดลไฟล์เพื่อใช้ในการสร้างแอปพลิเคชั่นอื่นได้ด้วย

กลุ่มของไฟล์คอมโพเนนท์ที่อยู่ในซอร์สโค้ดนี้ได้แก่ petri.dpk, line.pas, line.dcu, place.pas, place.dcu, trans.pas, trans.dcu, txtframe.pas, txtframe.dcu, petri.dcp, petri.drf, petri.res และ petri. Str



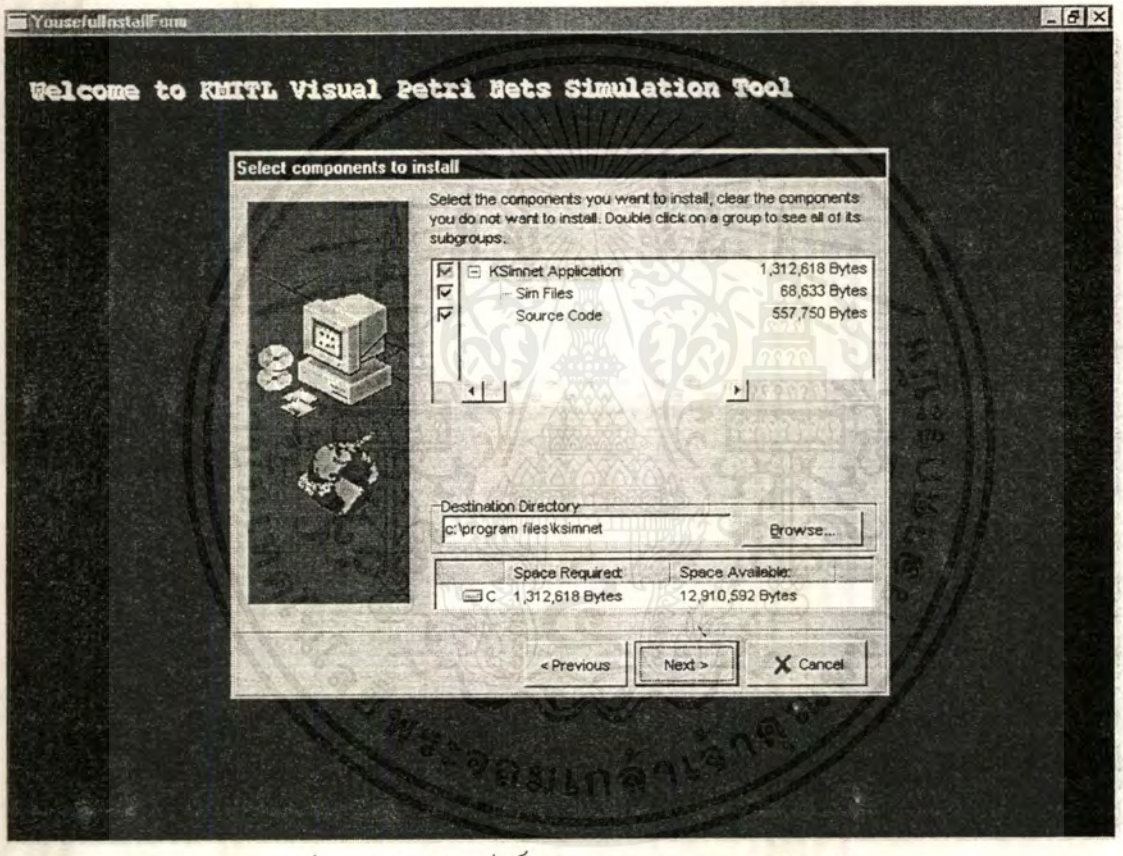

*รูปภาคผนวก ข-4 ฟอร์ม setup type*

การอินสตอลแต่ละประเภทสำหรับโปรเจ็กต์นี้ประกอบด้วยกลุ่มของไฟล์ได้แก่

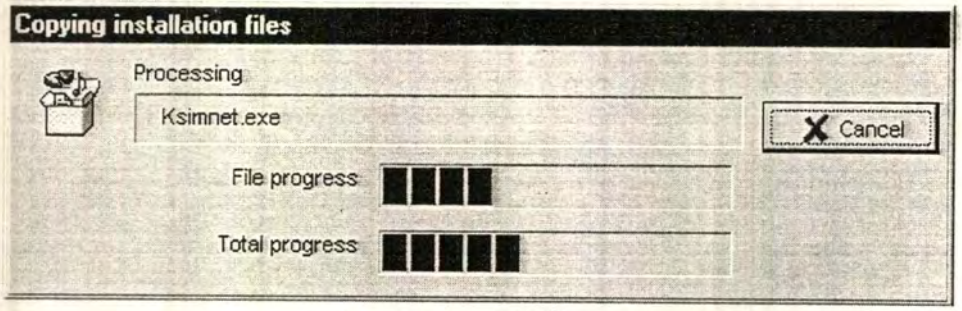
1) Compact เลือกลงให้น้อยที่สุด จะทำการอินสตอลเฉพาะกลุ่ม ksimnet Application
2) Typical เลือกลงเฉพาะที่จำเป็น ทำการอินสตอลเฉพาะ ksimnet Application และ Sim Files
3) Full ทำการอินสตอลทุกกลุ่มโดยอัตโนมัติ ทำการอินสตอลทั้ง ksimnet Application, Sim Files และ Source Code
4) Custom ให้ผู้อ่านทำการเลือกกลุ่มของไฟล์ที่จะติดตั้งเอง

และสามารถเลือกไดเรกทอรี่ที่จะลง (Destination Directory) โดยการคลิ๊กที่ปุ่ม Browse

1. ถ้าหากผู้อ่านเลือกที่จะกำหนดกลุ่มของไฟล์ที่จะติดตั้งเอง โดยคลิ๊กที่ Tab Custom โปรแกรมเซตอัพก็จะแสดงฟอร์ม Select components to install เพื่อให้ผู้อ่านเลือกกลุ่มของไฟล์ที่จะติดตั้ง ซึ่งวิธีการเลือกก็ทำได้โดยการคลิ๊กที่ checkbox ถ้าหากปรากฏเครื่องหมายถูก หน้ากลุ่มใดก็แสดงว่าต้องการติดตั้งไฟล์ แต่ถ้าหากไม่ปรากฏเครื่องหมายถูก ก็แสดงว่าไม่ติดตั้งไฟล์กลุ่มนั้น โดยผู้อ่านสามารถดูเนื้อที่บนฮาร์ดดิสก์ที่ต้องการสำหรับทำการกำหนด ไดเรกทอรี่ที่จะติดตั้ง ได้ด้วย หลังจากเลือกเสร็จแล้วคลิ๊กปุ่ม Next เพื่อให้ลงมือทำการติดตั้ง ในระหว่างการติดตั้งจะมี progress bar แสดงความก้าวหน้าในการติดตั้ง

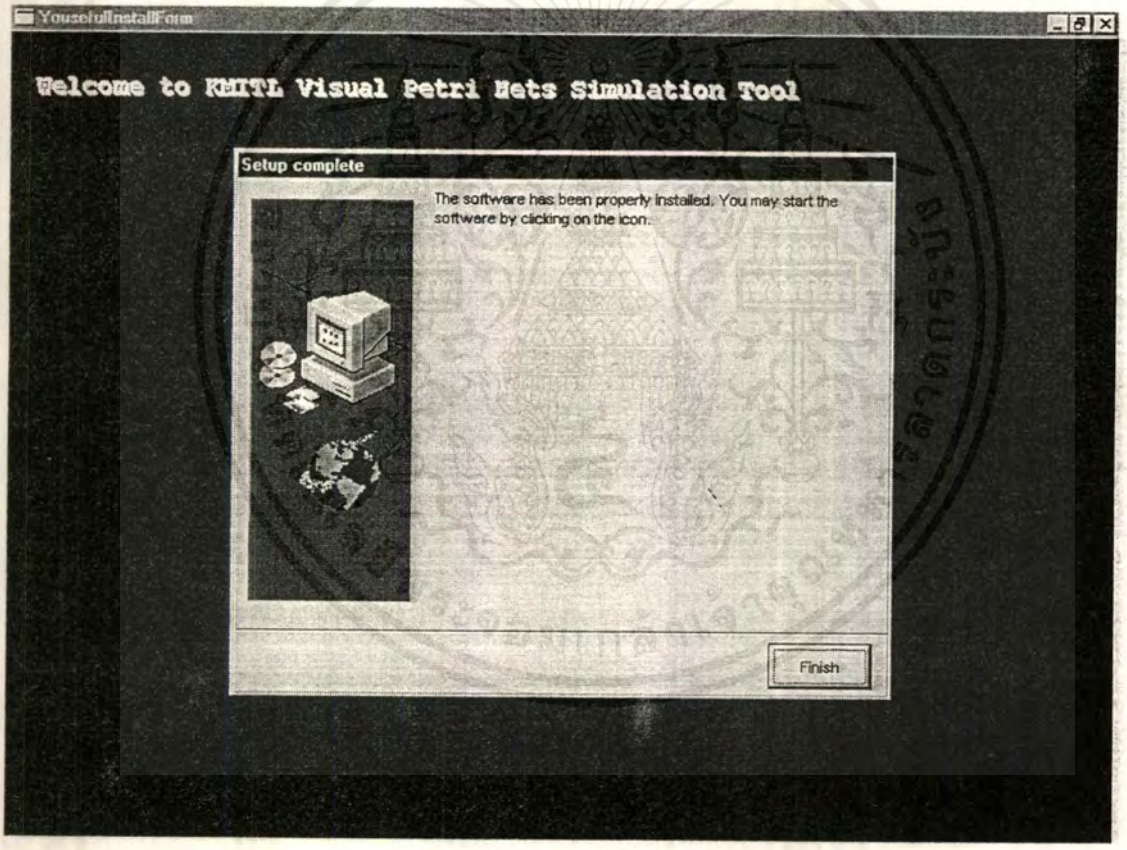



*รูปภาคผนวก ข-5 ฟอร์ม Select components to install*



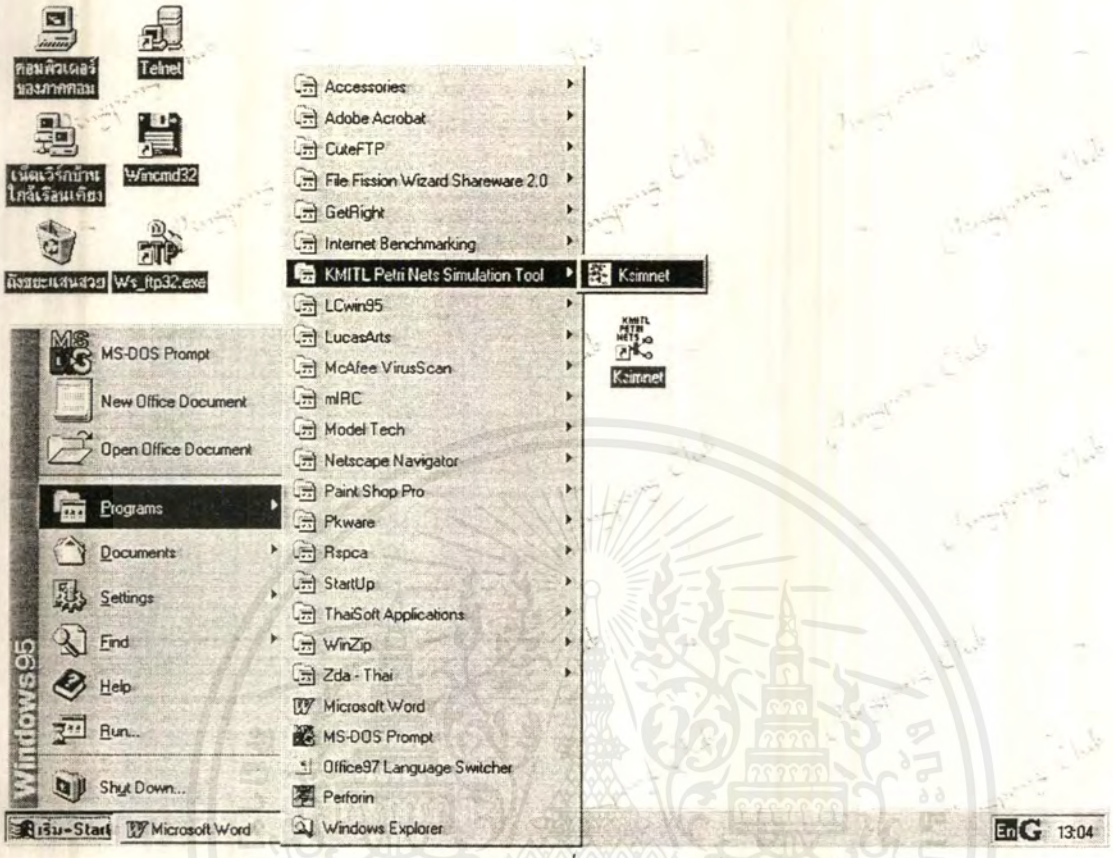

*รูปภาคผนวก ข-6 ฟอร์ม Progress*

2. หลังจากติดตั้งเสร็จโปรแกรมจะแสดงฟอร์ม setup complete คลิ๊กที่ปุ่ม Finish ก็เสร็จจากการติดตั้ง




*รูปภาคผนวก ข-7 ฟอร์ม Finish*

เมื่อติดตั้งเสร็จสามารถเรียกไฟล์แอปพลิเคชั่นได้จากเดสท็อปหรือจากโฟลเดอร์ของวินโดวส์




*รูปภาคผนวก ข-8 เรียกแอปพลิเคชั่นได้จากเดสท็อปหรือโฟลเดอร์*

# กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณอาจารย์ อภิเนตร อูนากูล ซึ่งเป็นอาจารย์ที่ปรึกษาที่ให้คำปรึกษา แนะนำและช่วยแก้ไขปัญหาที่เกิดขึ้น ตรวจแก้ไขสิ่งบกพร่อง จนทำให้ได้โครงงานที่ดีเช่นนี้

ขอบคุณเพื่อนๆน้องๆที่เป็นกำลังใจ ให้คำแนะนำและช่วยแก้ไขปัญหาที่เกิดในการทำงาน

ขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และท่านอาจารย์ทุกท่านที่ให้การศึกษาและวิทยาการความรู้ในด้านต่างๆ

ท้ายนี้ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และญาติพี่น้อง ที่ให้การสนับสนุนปัจจัยต่างๆในการเรียน คำสอน พร้อมทั้งกำลังใจตลอดมา

# หนังสืออ้างอิง

[1] Neil Rubenking ,"DELPHI PROGRAMMING PROBLEM SOLVER",IDG Books Worldwide , Inc.,1996.

[2] Ray Konopka edited by Jeff Duntemann ,"Developing Custom Delphi Components",The Coriolis Group ,Inc.,1996.

[3] TADAO MURATA, FELLOW, IEEE "Petri Nets: Properties, Analysis And Applications" Invited Paper page 541- 574.

[4] จารุวรรณ ระวิภักตร์ ,"รู้ลึก รู้จริง! Style Borland Delphi",บริษัท เฟริสท์ แปซิฟิก มีเดีย (ไทยแลนด์) จำกัด,296 ห พ.ศ. 2540.

การศึกษารายละเอียด ข้อมูลและคำแนะนำต่าง ๆ นอกจากที่ได้ในตำราเรียน ทางคณะผู้ดำเนินการ โครงงานนี้ได้ค้นคว้าเอาข้อมูลต่างๆที่เกี่ยวกับการศึกษาในโครงงาน มาจากอินเตอร์เนต ที่เวบไซต์ที่ให้ข้อมูลที่เกี่ยวข้องดังนี้

เวบไซต์ที่ให้ข้อมูลเกี่ยวกับ delphi

http://www.soton.ac.uk/~ajd/delphi/freeware.html

http://iconz.co.nz/commercial/Borland/Delphi/resource.htm

http://www.iscinc.com/nydug.html

http://sunsite.icm.edu.pl/archive/delphi/

http://www.cdrom.com/pub/delphi_www/

http://www.borland.com/TechInfo/delphi/index.html

http://www.users.dircon.co.uk/~zeus/

http://www.delphi32.com/

http://www.delphiexchange.com/

http://www.intermid.com/sherlock/compmall.html

เวบไซต์ที่ให้ข้อมูลเกี่ยวกับ Petri Nets

http://www.daimi.aau.dk/PetriNets/

http://www.informatik.hu-berlin.de/PNT/pnt-public.html