บทที่ 4

การออกแบบและสร้างซอฟต์แวร์เพทริเน็ต

ความนำ

ในงานวิจัยนี้เป็นการพัฒนาซอฟต์แวร์ประยุกต์บนเครื่องไมโครคอมพิวเตอร์ เพื่อใช้เป็นเครื่องมือในการสร้างแบบจำลองและวิเคราะห์หาคุณสมบัติของระบบควบคุมลำดับโดยใช้ทฤษฎีเพทริเน็ต การพัฒนากระทำบนเครื่องไมโครคอมพิวเตอร์ IBM PC/Compatible ซึ่งใช้ไมโครโปรเซสเซอร์ 80486 DX2-66 และเขียนด้วยภาษาวิชวลเบสิก โดยใช้ไมโครซอฟต์วิชวล-เบสิก เวอร์ชัน 3.0 (Microsoft Visual Basic 3.0) [13,14] เป็นเครื่องมือในการพัฒนา ในขั้นต้นจึงตั้งเป้าหมายของการพัฒนาไว้เพื่อใช้เป็นขอบเขตในการทำงานดังนี้

1. สามารถใช้เขียนแบบจำลองเพทริเน็ตได้ และ เนื่องจากแบบจำลองเพทริเน็ตเป็นภาษารูปภาพ โปรแกรมที่พัฒนาขึ้นมา จึงต้องสามารถใช้วาดรูปองค์ประกอบต่าง ๆ ของแบบจำลองเพทริเน็ตได้ ซึ่งเรียกว่าเป็น กราฟฟิกเอดิเตอร์ (Graphic Editor)

2. สามารถตรวจสอบวากยสัมพันธ์ (Syntax) ของแบบจำลองเพทริเน็ตที่ออกแบบได้

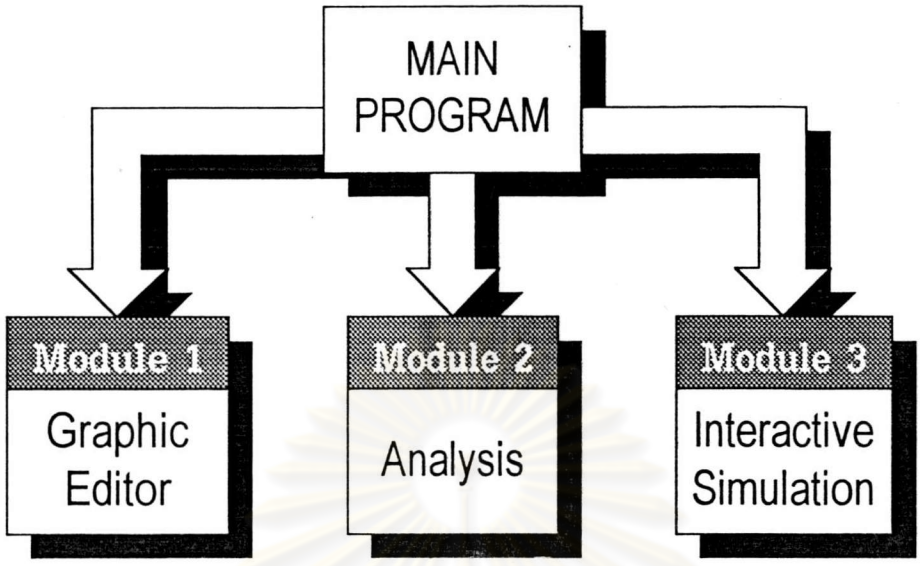
3. สามารถวิเคราะห์แบบจำลองเพทริเน็ตเพื่อหาคุณสมบัติของเพทริเน็ต โดยใช้วิธีรีชเอบิลิตรีทรีได้

4. สามารถสร้างแบบจำลองเพทริเน็ต ในขณะที่ผู้ใช้สามารถตรวจดูพฤติกรรมของระบบที่ออกแบบได้ ซึ่งเรียกว่า การจำลองแบบการทำงานของเพทริเน็ตเชิงโต้ตอบ (Interactive Simulation)

แนวความคิดในการออกแบบ

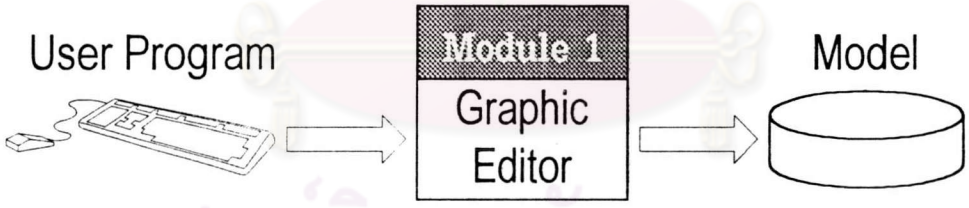
1. โครงสร้างของระบบซอฟต์แวร์เพทริเน็ต

ในการออกแบบซอฟต์แวร์เพทริเน็ตนี้ แบ่งงานออกเป็น 3 ส่วนหลัก ๆ ดังแสดงในรูปที่ 4.1 โดยมีโปรแกรมหลักทำหน้าที่จัดการ และเรียกใช้แต่ละโมดูลอีกต่อหนึ่ง



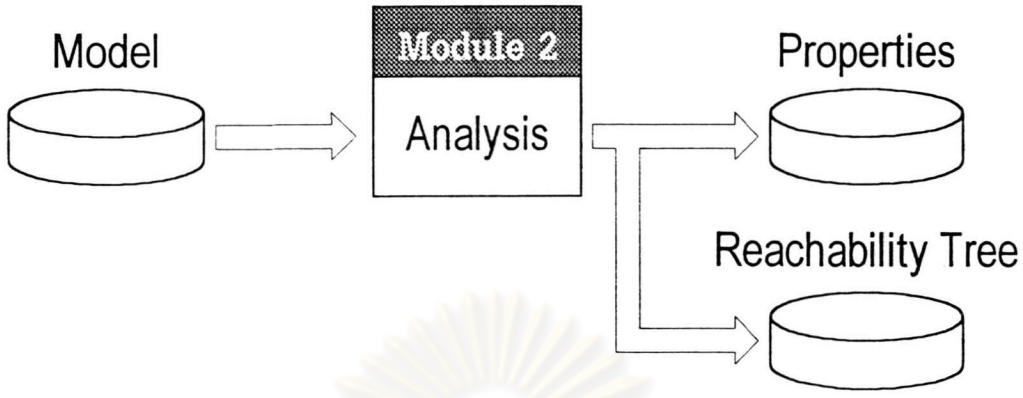
รูปที่ 4.1 โครงสร้างของระบบ

โมดูลแรกทำหน้าที่เป็นกราฟฟิกเอดิเตอร์ ซึ่งเป็นเครื่องช่วยในการวาดแบบจำลองเพทริเน็ต สามารถเก็บโปรแกรมที่เขียนลงในแฟ้มข้อมูลและอ่านขึ้นมาได้ อินพุตของโมดูลนี้เป็นการรับโปรแกรมที่เขียนจากผู้ใช้และให้เอาต์พุตเป็นแฟ้มข้อมูลที่เก็บแบบจำลองเพทริเน็ตที่ผู้ใช้เขียนขึ้น



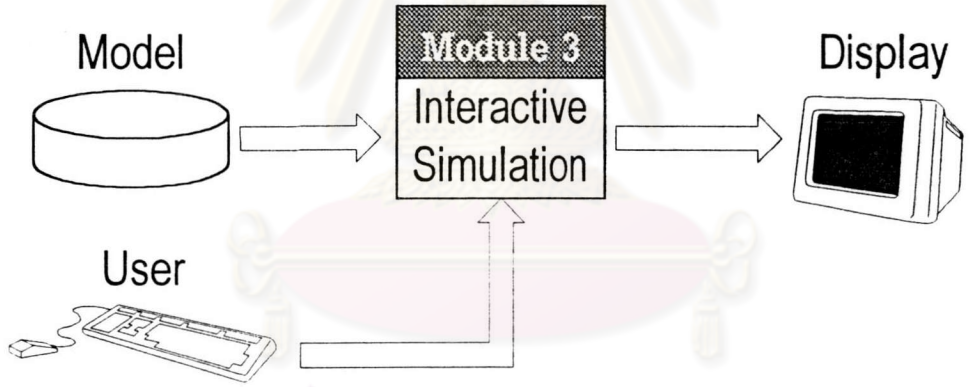
รูปที่ 4.2 โมดูลสำหรับเขียนแบบจำลองเพทริเน็ต

โมดูลที่สองทำหน้าที่วิเคราะห์แบบจำลองเพทริเน็ต โดยใช้วิธีรีชเอบิลิตีทรี ซึ่งให้ผลการวิเคราะห์เป็นรีชเอบิลิตีทรี และ คุณสมบัติของแบบจำลองเพทริเน็ตที่ผู้ใช้ออกแบบ เช่น Safeness Boundness Conservation และ Liveness เป็นต้น



รูปที่ 4.3 โมดูลสำหรับวิเคราะห์แบบจำลองเพทริเน็ต

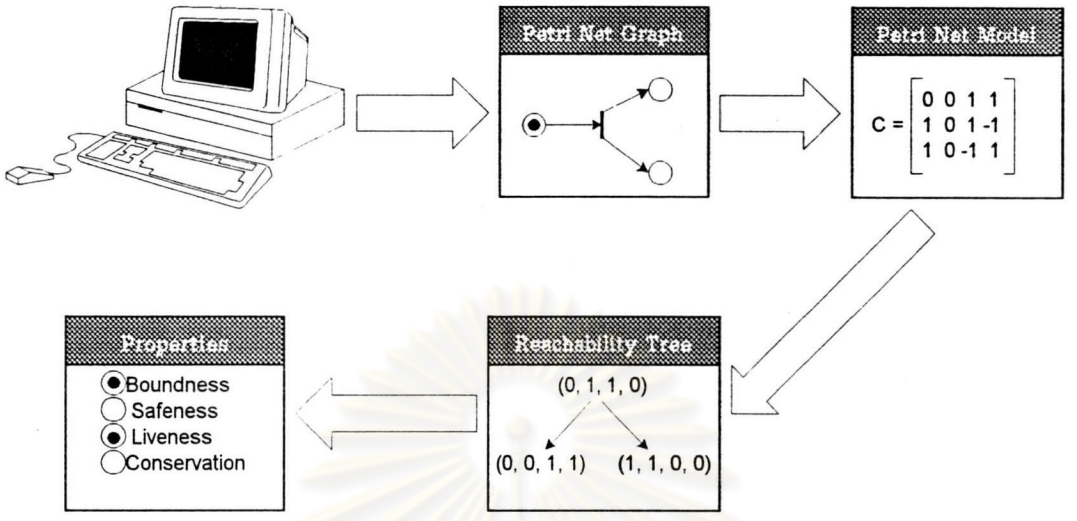
โมดูลที่สามทำหน้าที่จำลองแบบการทำงานของเพทริเน็ตเชิงโต้ตอบ ถูกออกแบบมาเพื่อให้ผู้ใช้เข้าใจพฤติกรรมของระบบในขณะที่ออกแบบแบบจำลองเพทริเน็ต



รูปที่ 4.4 โมดูลการจำลองแบบการทำงานของเพทริเน็ตเชิงโต้ตอบ

2. เส้นทางของข้อมูล (Dataflow)

เริ่มต้นจากการป้อนโปรแกรมโดยผู้ใช้ เมื่อสั่งให้เก็บข้อมูล ก็จะเกิดแฟ้มข้อมูลของแบบจำลองเพทริเน็ตที่เขียนขึ้น และเมื่อผู้ใช้สั่งให้วิเคราะห์แบบจำลองเพทริเน็ต ซอฟต์แวร์จะสร้างรีชเอบิลิตีทรีและแสดงคุณสมบัติของแบบจำลองเพทริเน็ตที่เขียนขึ้น ขณะที่ผู้ใช้สร้างแบบจำลองเพทริเน็ต ผู้ใช้สามารถจำลองแบบการทำงานของเพทริเน็ตเพื่อดูพฤติกรรมของระบบที่ออกแบบได้ เส้นทางของข้อมูลสำหรับซอฟต์แวร์เพทริเน็ตนี้แสดงดังรูปที่ 4.5



รูปที่ 4.5 เส้นทางของข้อมูล

3. ส่วนแสดงผล

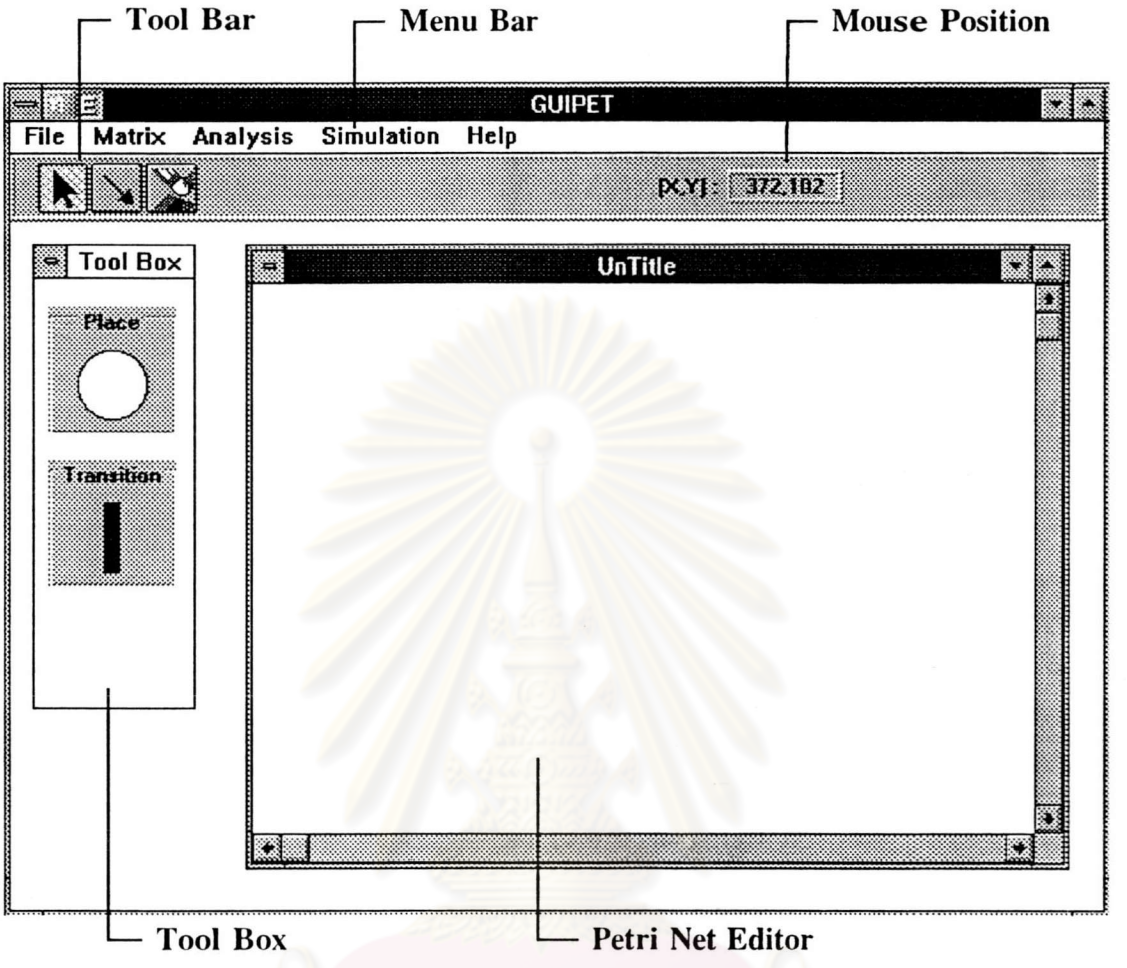
หัวข้อนี้เป็นการวางโครงร่าง รูปแบบ หน้าตา ของซอฟต์แวร์ที่จะพัฒนาขึ้น เพื่อให้การออกแบบส่วนอื่น ๆ ในภายหลังอ้างอิงตามรูปแบบที่วางไว้ ในงานวิจัยนี้พัฒนาซอฟต์แวร์บนระบบปฏิบัติการไมโครซอฟต์วินโดว์ ซึ่งเป็นระบบติดต่อกับผู้ใช้แบบกราฟฟิก เนื่องจากรูปภาพเป็นเรื่องที่สำคัญสำหรับแบบจำลองเพทริเน็ต หากไม่สามารถแสดงรูปภาพได้ก็ยากที่จะทำความเข้าใจกับแบบจำลองเพทริเน็ตที่เขียนขึ้น ดังนั้นจึงให้ความสำคัญในเรื่องการแสดงผลเป็นพิเศษ โดยจะพัฒนาให้สามารถแสดงผลได้ชัดเจน สวยงาม และน่าใช้ หน้าจอจะแบ่งเป็น 4 ส่วนหลัก ๆ ดังแสดงในรูปที่ 4.6 คือ

3.1 ส่วนใช้งานสำหรับแสดงแบบจำลองเพทริเน็ ต (Petri Net Editor) เป็นส่วนที่ให้ผู้ใช้สร้างและแก้ไขแบบจำลองเพทริเน็ต

3.2 ส่วนแสดงรายการคำสั่ง (Menu Bar) เป็นรายการคำสั่งที่ผู้ใช้สามารถเรียกใช้งานได้ เช่น การสร้างและเก็บแฟ้มข้อมูลใหม่ การวิเคราะห์แบบจำลองเพทริเน็ต เป็นต้น

3.3 ส่วนแสดงรายการเครื่องมือ (Tool Bar) เป็นรายการเครื่องมือที่ผู้ใช้สามารถเรียกใช้ได้ เช่น ตัวชี้ (Pointer) ยางลบ (Rubber) และ อาร์ก เป็นต้น

3.4 ส่วนแสดงกล่องเครื่องมือ (Tool Box) เก็บรูปเพลส และ ทรานซิชัน ที่ผู้ใช้สามารถนำไปใช้ได้ โดยใช้วิธีลาก (Drag) จากกล่องเครื่องมือ และปล่อย (Drop) ที่ส่วนใช้งานสำหรับแสดงแบบจำลองเพทริเน็ต



รูปที่ 4.6 ส่วนแสดงผล

<u>โครงสร้างข้อมูลของแบบจำลองเพทริเน็ต</u>

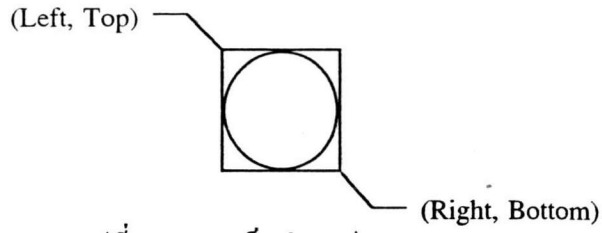
ในการออกแบบโครงสร้างข้อมูลของซอฟต์แวร์เพทริเน็ตนี้ ต้องคำนึงถึงการจัดการข้อมูลของส่วนสร้างและส่วนวิเคราะห์แบบจำลองเพทริเน็ต ในแบบจำลองเพทริเน็ตมีองค์ประกอบสำคัญ 3 ส่วน คือ เพลส ทรานซิชัน และ อาร์ก ซึ่งเป็นตัวบอกความสัมพันธ์ระหว่างเพลสกับทรานซิชัน ดังนั้นชนิดของข้อมูลที่ใช้ในโครงสร้างข้อมูลนี้แบ่งออกเป็น 3 ชนิด คือ

1. <u>ข้อมูลแบบเพลส</u>

Position	Place Name	Token	Used

รูปที่ 4.7 ข้อมูลแบบเพลส

Position : ตำแหน่งของเพลสบนเพทริเน็ตเอดิเตอร์ คือ Left Top Right Bottom



รูปที่ 4.8 การเก็บตำแหน่งของเพลส

Place Name : ชื่อของเพลส

Token : จำนวนโทเค็นที่อยู่ในเพลสนั้น

Used : สถานะของเพลสโดยมี 2 สถานะ คือ

Yes : เพลสถูกแสดงบนเพทริเน็ตเอดิเตอร์

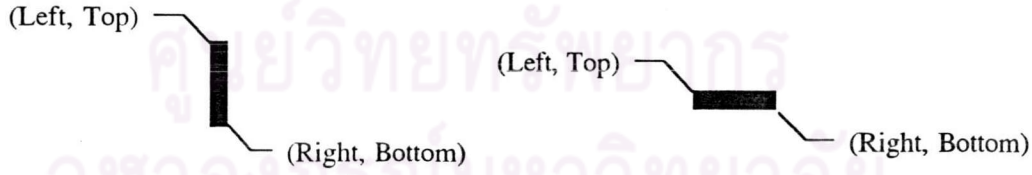
No : เพลสไม่ได้ถูกแสดงบนเพทริเน็ตเอดิเตอร์

2. <u>ข้อมูลแบบทรานซิชัน</u>

Position	Transition Name	Vertical	Used

รูปที่ 4.9 ข้อมูลแบบทรานซิชัน

Position : ตำแหน่งของทรานซิชันบนเพทริเน็ตเอดิเตอร์



(a) ทรานซิชันในแนวตั้ง (b) ทรานซิชันในแนวนอน

รูปที่ 4.10 การเก็บตำแหน่งของทรานซิชัน

Transition Name : ชื่อของทรานซิชัน

Vertical : แสดงผลของทรานซิชันในแนวตั้ง หรือ แนวนอน

Yes : ทรานซิชันในแนวตั้ง

No : ทรานซิชันในแนวนอน

3. ข้อมูลแบบอาร์ก

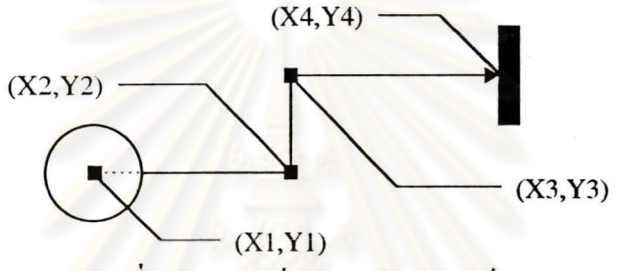
X(1to10)	Y(1to10)	EndId	Atype	PlaceId	TransId	Weight	Arrow(1to3)	Used

รูปที่ 4.11 ข้อมูลแบบอาร์ก

X(1to10) : อาร์เรย์ของตำแหน่งทางเดินของอาร์กในแนวนอน

Y(1to10) : อาร์เรย์ของตำแหน่งทางเดินของอาร์กในแนวตั้ง

EndId : จำนวนตำแหน่งทางเดินของอาร์กทั้งหมด



รูปที่ 4.12 การเก็บตำแหน่งของอาร์ก

Atype : ชนิดของอาร์ก คือ

"O" : เอาต์พุตอาร์ก

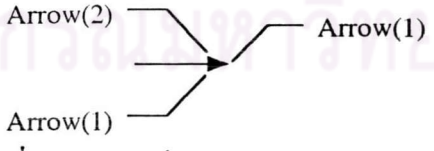
"I" : อินพุตอาร์ก

PlaceId : หมายเลขเพลสที่เป็นอินพุตเพลสหรือเอาต์พุตเพลสของอาร์กนี้

TransId : หมายเลขทรานซิชันที่ต่อกับอาร์กนี้

Weight : จำนวนของอาร์ก

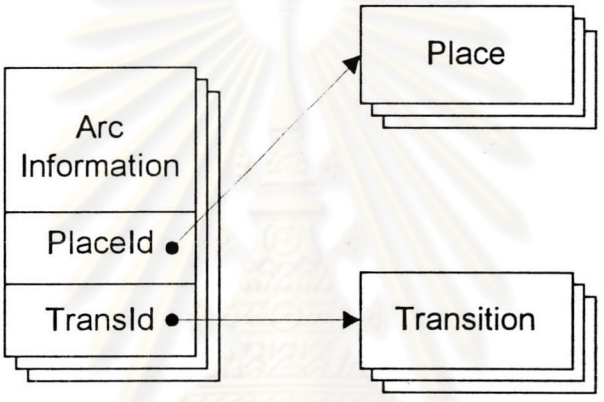
Arrow(1to3) : อาร์เรย์ของตำแหน่งหัวลูกศรของอาร์ก 3 ตำแหน่ง แสดงดังรูป



รูปที่ 4.13 การเก็บตำแหน่งของหัวลูกศร

Used : สถานะของอาร์ก

เนื่องจากแบบจำลองเพทริเน็ตเป็นแบบจำลองทางกราฟฟิก ดังนั้นในการวิเคราะห์แบบจำลองเพทริเน็ตด้วยคอมพิวเตอร์ จึงต้องแปลงข้อมูลทางกราฟฟิกให้อยู่ในรูปที่คอมพิวเตอร์เข้าใจได้ ซึ่งข้อมูลแบบเมตริกซ์เป็นข้อมูลชนิดหนึ่งที่คอมพิวเตอร์สามารถประมวลผลได้ แบบจำลองเพทริเน็ตสามารถแทนให้อยู่ในรูปเมตริกซ์ของอินพุตฟังก์ชันและเอาต์พุตฟังก์ชัน ดังนั้นโครงสร้างข้อมูลของซอฟต์แวร์เพทริเน็ตดังแสดงในรูปที่ 4.14 จึงถูกออกแบบให้ใช้ข้อมูลชนิดอาร์กเป็นตัวเชื่อมความสัมพันธ์ระหว่างข้อมูลชนิดเพลสและทรานซิชัน ซึ่งโครงสร้างข้อมูลแบบนี้สามารถถูกแปลงไปสู่เมตริกซ์ของอินพุตฟังก์ชันและเอาต์พุตฟังก์ชันได้โดยง่าย



รูปที่ 4.14 โครงสร้างข้อมูลของแบบจำลองเพทริเน็ต

<u>กราฟฟิกเอดิเตอร์</u>

กราฟฟิกเอดิเตอร์เป็นส่วนสร้างและแก้ไขแบบจำลองเพทริเน็ต ถูกพัฒนาด้วยภาษาวิชวลเบสิกบนระบบปฏิบัติการไมโครซอฟต์วินโดว์ ซึ่งการเขียนโปรแกรมด้วยภาษาวิชวลเบสิกแตกต่างจากการเขียนโปรแกรมแบบเดิมที่โปรแกรมด้วยภาษาคอมพิวเตอร์ที่ทำงานตามลำดับขั้นตอนที่กำหนดโดยผู้เขียนโปรแกรมเท่านั้น (Procedural Language) โดยภาษาวิชวลเบสิกเป็นภาษาคอมพิวเตอร์ที่ทำงานขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้น (Event-Driven Language) กล่าวคือเมื่อเกิดเหตุการณ์เฉพาะขึ้น ก็จะมีการเรียกวิธีดำเนินการ (Procedure) สำหรับเหตุการณ์นั้นขึ้นมาทำงาน ในการเขียนโปรแกรมด้วยภาษาวิชวลเบสิกประกอบด้วย 3 ขั้นตอนหลัก คือ การออกแบบวัตถุ (Object) ของโปรแกรม การแก้ไขคุณสมบัติของวัตถุแต่ละวัตถุ และการเขียนวิธีดำเนินการสำหรับแต่ละเหตุการณ์ที่จะเกิดขึ้นกับวัตถุเหล่านั้น ดังนั้นในการออกแบบกราฟฟิกเอดิเตอร์จึงออกแบบให้มีวัตถุที่สำคัญ 3 ชนิดคือ เพลส ทรานซิชัน และ อาร์ก เพื่อให้สอดคล้องกับชนิดของข้อมูลที่

กล่าวไว้แล้วในหัวข้อโครงสร้างข้อมูล สำหรับวิธีดำเนินการและคุณสมบัติของแต่ละวัตถุจะอธิบายในการทำงานของกราฟฟิกเอดิเตอร์ซึ่งมีรายละเอียดดังนี้

1. การวาดองค์ประกอบของแบบจำลองเพทริเน็ต

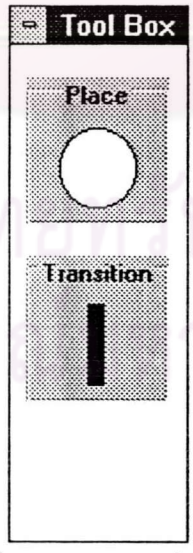
ในการวาดองค์ประกอบของแบบจำลองเพทริเน็ตบนเพทริเน็ตเอดิเตอร์ ผู้ใช้ต้องเลือกเครื่องมือในการวาดจากรายการเครื่องมือดังรูปที่ 4.15 ซึ่งแสดงปุ่มของรายการเครื่องมือในการวาด และ กล่องเครื่องมือดังแสดงดังรูปที่ 4.16 ซึ่งเก็บองค์ประกอบที่ใช้ในการวาดประกอบด้วยเพลส และ ทรานซิชัน

รูปที่ 4.15 รายการเครื่องมือ

Point Button : ใช้ในการเลือกองค์ประกอบของแบบจำลองเพทริเน็ต

Arc Button : ใช้สำหรับลากอาร์กเพื่อเชื่อมความสัมพันธ์ระหว่างเพลสและทรานซิชัน

Rubber Button : ใช้สำหรับลบองค์ประกอบของแบบจำลองเพทริเน็ต



รูปที่ 4.16 กล่องเครื่องมือ

ในการวาดแต่ละองค์ประกอบจะมีวิธีในการวาดแตกต่างกันไป ซึ่งแต่ละวิธีต้องใช้เมาส์ (Mouse) เป็นเครื่องมือสำคัญในการวาด แสดงรายละเอียดในการวาดแต่ละองค์ประกอบได้ดังนี้

1.1 การวาดเพลส

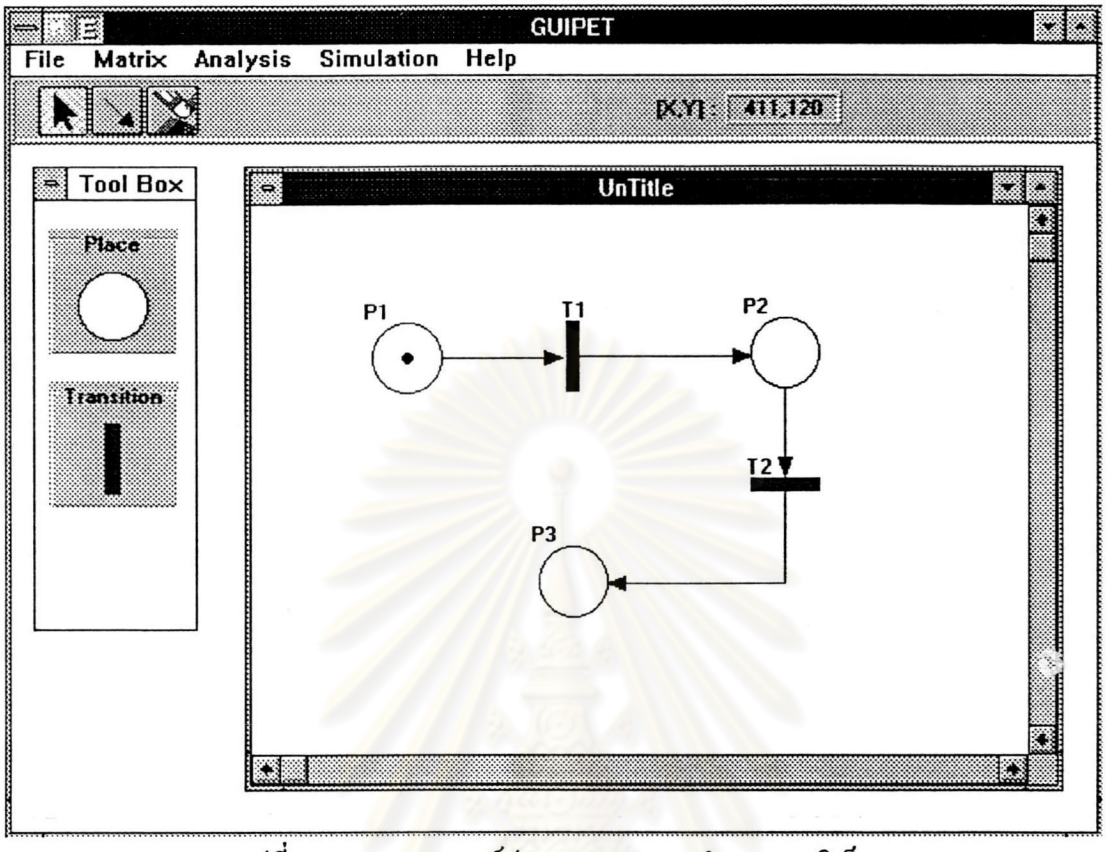
ผู้ใช้สามารถวาดเพลสโดยเลือก "Point Button" จากรายการเครื่องมือ และใช้เมาส์ลาก (Drag) เพลสจากกล่องเครื่องมือไปปล่อย (Drop) ยังเพทริเน็ตเอดิเตอร์ เพลสจะ ปรากฏบนเพทริเน็ตเอดิเตอร์ ซึ่งโปรแกรมในการวาดเพลสจะถูกเขียนในตัวดำเนินการสำหรับเหตุการณ์เมาส์ลากและปล่อย (DragDrop) ของเพทริเน็ตเอดิเตอร์โดยมีการตรวจสอบเงื่อนไขว่าวัตถุที่ลากเป็นเพลส และตำแหน่งของเพลสจะถูกบันทึกลงในอาร์เรย์ของข้อมูลแบบเพลส

1.2 การวาดทรานซิชัน

ผู้ใช้สามารถวาดทรานซิชันโดยเลือก "Point Button" จากรายการเครื่องมือและใช้เมาส์ลากทรานซิชันจากกล่องเครื่องมือไปปล่อยในบริเวณเพทริเน็ตเอดิเตอร์ ทรานซิชันในแนวตั้งจะปรากฏบนเพทริเน็ตเอดิเตอร์ ซึ่งโปรแกรมในการวาดทรานซิชันจะถูกเขียนในตัวดำเนินการสำหรับเหตุการณ์เมาส์ลากและปล่อยของเพทริเน็ตเอดิเตอร์โดยมีการตรวจสอบเงื่อนไขว่าวัตถุที่ลากเป็นทรานซิชันและตำแหน่งของทรานซิชันจะถูกบันทึกลงในอาร์เรย์ของข้อมูล แบบทรานซิชัน

1.3 การวาดอาร์ก

ผู้ใช้สามารถวาดอาร์กโดยเลือก "Arc Button" จากรายการเครื่องมือและใช้เมาส์คลิก (MouseClick) ลงบนเพลสหรือทรานซิชันที่ต้องการสร้างความสัมพันธ์ เคลื่อนเมาส์ไปในทิศที่ต้องการจะเห็นเส้นลากตามไปด้วย เมื่อถึงจุดที่ต้องการก็ใช้เมาส์คลิกลงบนเพลส หรือทรานซิชันที่ต้องการ ขณะที่วาดอาร์กโปรแกรมจะตรวจสอบวากยสัมพันธ์ของแบบจำลองเพทริ-เน็ตที่สร้างขึ้น โดยไม่ยอมให้วาดอาร์กเชื่อมต่อระหว่างเพลสกับเพลส หรือ ทรานซิชันกับทรานซิ-ชัน ถ้าการตรวจสอบวากยสัมพันธ์ถูกต้องอาร์กจะปรากฏบนเพทริเน็ตเอดิเตอร์ ซึ่งโปรแกรมในการวาดอาร์กจะถูกเขียนในตัวดำเนินการสำหรับเหตุการณ์เมาส์คลิกและการเคลื่อนเมาส์ (MouseMove) ของเพทริเน็ตเอดิเตอร์ และตำแหน่งทางเดินของอาร์กทั้งหมด ชนิดของอาร์ก หมายเลขเพลส และ หมายเลขทรานซิชันที่เชื่อมต่อ จะถูกบันทึกลงในอาร์เรย์ของข้อมูลแบบอาร์ก



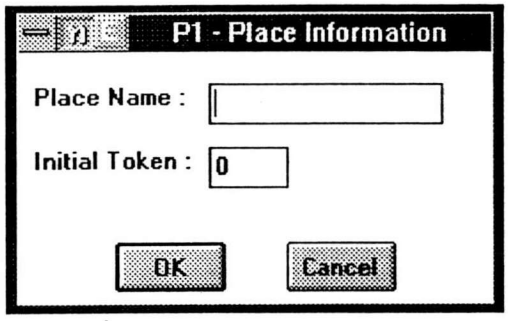
รูปที่ 4.17 การวาดองค์ประกอบของแบบจำลองเพทริเน็ต

2. <u>การแก้ไของค์ประกอบของแบบจำลองเพทริเน็ต</u>

ในการแก้ไของค์ประกอบของแบบจำลองเพทริเน็ต "Point Button" จากรายการเครื่องมือต้องถูกเลือกอยู่เสมอ โดยมีรายละเอียดของแต่ละองค์ประกอบดังนี้

2.1 การแก้ไขเพลส

ผู้ใช้สามารถแก้ไขตำแหน่งของเพลสโดยใช้เมาส์คลิกและลากเพลสแล้วไปปล่อยยังตำแหน่งที่ต้องการในเพทริเน็ตเอดิเตอร์ ซึ่งตำแหน่งใหม่ของเพลสจะถูกบันทึกแทนตำแหน่งเดิมในอาร์เรย์ของข้อมูลแบบเพลส เพลสในตำแหน่งเดิมจะถูกลบและเพลสในตำแหน่งใหม่จะปรากฏในเพทริเน็ตเอดิเตอร์ และผู้ใช้ยังสามารถแก้ไขจำนวนโทเค็นในแต่ละเพลสที่ต้องการได้โดยการดับเบิ้ลคลิก (DoubleClick) บนเพลสที่ต้องการแก้ไข โปรแกรมจะแสดงกรอบข้อความ (Dialog Box) ดังแสดงในรูปที่ 4.18 ซึ่งผู้ใช้สามารถแก้ไขจำนวนของโทเค็นโดยพิมพ์จำนวนโทเค็นใหม่ลงไป และกดปุ่ม "OK" จำนวนโทเค็นใหม่จะถูกบันทึกลงในอาร์เรย์ของข้อมูลแบบเพลส และแสดงจำนวนโทเค็นใหม่ในเพทริเน็ตเอดิเตอร์



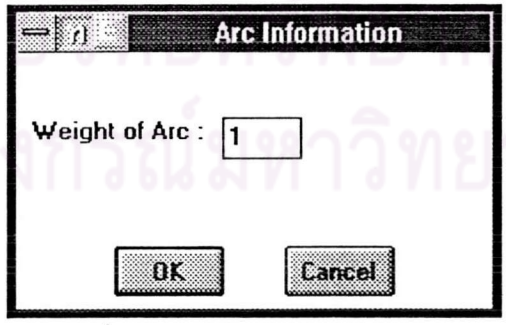
รูปที่ 4.18 กรอบข้อความของเพลส

2.2 การแก้ไขทรานซิชัน

ผู้ใช้สามารถแก้ไขตำแหน่งของทรานซิชันได้โดยวิธีเดียวกับการแก้ไขตำแหน่งของเพลส และยังสามารถแก้ไขการแสดงทรานซิชันจากแนวตั้งเป็นแนวนอนได้โดยคลิกขวาบนทรานซิชันที่ต้องการแก้ไข แต่ถ้าต้องการให้แสดงเป็นแนวตั้งเหมือนเดิม ก็คลิกขวาบนทรานซิชันนั้นอีกครั้งหนึ่ง

2.3 การแก้ไขอาร์ก

ผู้ใช้สามารถแก้ไขจำนวนอาร์กที่ลากเชื่อมต่อระหว่างเพลสกับทรานซิชันได้โดยการดับเบิ้ลคลิกลงบนอาร์กที่ต้องการแก้ไข โปรแกรมจะแสดงกรอบข้อความดังรูปที่ 4.19 ซึ่งผู้ใช้สามารถแก้ไขจำนวนอาร์กที่ลากเชื่อมต่อ โดยพิมพ์จำนวนอาร์กใหม่ลงไป และกดปุ่ม "OK" จำนวนอาร์กใหม่จะถูกบันทึกลงในอาร์เรย์ของข้อมูลแบบอาร์ก และแสดงจำนวนอาร์กใหม่ในเพทริเน็ตเอดิเตอร์



รูปที่ 4.19 กรอบข้อความของอาร์ก

3. การลบองค์ประกอบของแบบจำลองเพทริเน็ต

ในการลบองค์ประกอบของแบบจำลองเพทริเน็ต ผู้ใช้ต้องเลือก "Rubber Button" จากรายการเครื่องมือ และใช้เมาส์คลิกบนเพลส ทรานซิชัน หรือ อาร์กที่ต้องการจะลบ ถ้าลบเพลสหรือทรานซิชัน อาร์กที่เชื่อมต่อระหว่างเพลสกับทรานซิชันนั้นจะถูกลบไปด้วย

4. การจัดการแฟ้มข้อมูล

การจัดการแฟ้มข้อมูลแบ่งเป็น 4 รายการ โดยเลือกจากรายการคำสั่ง ได้แก่ การเก็บแบบจำลองเพทริเน็ตลงแฟ้มข้อมูล ("Save") การเก็บแบบจำลองเพทริเน็ตลงแฟ้มข้อมูลโดยระบุชื่อแฟ้ม ("Save As") การอ่านแฟ้มข้อมูลขึ้นมาใช้งาน ("Open") และการเคลียร์เพทริเน็ตเอดิเตอร์เพื่อเริ่มเขียนแบบจำลองเพทริเน็ตใหม่ ("New") แฟ้มข้อมูลที่ใช้มีนามสกุลเป็น ".PNG" โดยมีรูปแบบแฟ้มข้อมูลแบบข้อความ (Text File) ซึ่งผู้ใช้สามารถแก้ไข้ด้วยโปรแกรมบรรณาธิการ (Text Editor) ได้ โดยแสดงรูปแบบแฟ้มข้อมูลแสดงดังรูปที่ 4.20

```
Place
    P<PlaceId1>  XLoc(<X-Coordinate>)  YLoc(<Y-Coordinate>)  Name(<Place Name>);
    P<PlaceId2>  XLoc(<X-Coordinate>)  YLoc(<Y-Coordinate>)  Name(<Place Name>);
    ..........
    P<PlaceIdn>  XLoc(<X-Coordinate>)  YLoc(<Y-Coordinate>)  Name(<Place Name>);
EndP

Transition
    T<TransId1> XLoc(<X-Coordinate>) YLoc(<Y-Coordinate>) Vertical(<True or False>);
    T<TransId1> XLoc(<X-Coordinate>) YLoc(<Y-Coordinate>) Vertical(<True or False>);
    ..........
    T<TransIdn> XLoc(<X-Coordinate>) YLoc(<Y-Coordinate>) Vertical(<True or False>);
EndT

Arc
    I[<TransId>,<PlaceId>]  Weight(<Weight>)  Path[(X1,Y1),(X2,Y2),...,(Xn,Yn);
    O[<TransId>,PlaceId>]  Weight(<Weight>)  Path[(X1,Y1),(X2,Y2),...,(Xn,Yn);
    ..........
EndA

Initial Marking
    [<Token in P1>,<Token in P2>,...,<Token in Pn>]
```

รูปที่ 4.20 รูปแบบแฟ้มข้อมูล

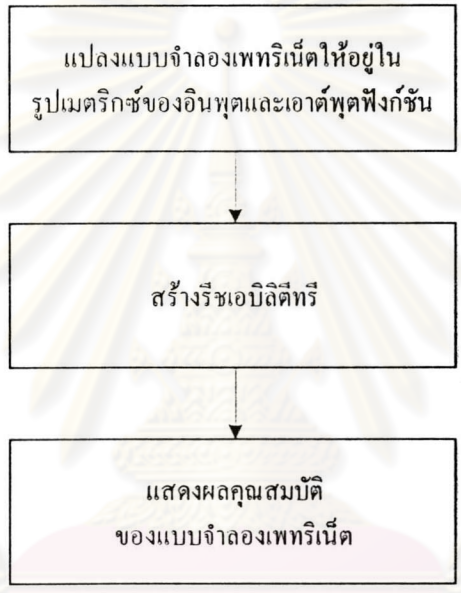
รายละเอียดสำหรับตัวแปรในแฟ้มข้อมูลดังรูปที่ 4.20 แสดงความหมายได้ดังตารางที่ 4.1

ตัวแปร	ความหมาย
Place	กำหนดตำแหน่งเริ่มต้นในการแสดงรายละเอียดของเพลส
P	แทนเพลสและจำนวนเต็มที่ต่อจาก "P" จะแสดงหมายเลขของเพลส
Name	ชื่อของเพลส
EndP	กำหนดตำแหน่งสิ้นสุดในการแสดงรายละเอียดของเพลส
Transition	กำหนดตำแหน่งเริ่มต้นในการแสดงรายละเอียดของทรานซิชัน
T	แทนทรานซิชันและจำนวนเต็มที่ต่อจาก "T" จะแสดงหมายเลขของทรานซิชัน
Vertical	กำหนดการแสดงทรานซิชันในแนวตั้งหรือแนวนอน T : แสดงในแนวตั้ง F : แสดงในแนวนอน
EndT	กำหนดตำแหน่งสิ้นสุดในการแสดงรายละเอียดของทรานซิชัน
Xloc	กำหนดตำแหน่งของวัตถุในแนวแกน X
Yloc	กำหนดตำแหน่งของวัตถุในแนวแกน Y
Arc	กำหนดตำแหน่งเริ่มต้นในการแสดงรายละเอียดของอาร์ก
I	แทนอินพุตอาร์ก
O	แทนเอาต์พุตอาร์ก
Path	กำหนดทางเดินของอาร์ก
Weight	กำหนดจำนวนของอาร์กที่เชื่อมต่อ
EndA	กำหนดตำแหน่งสิ้นสุดในการแสดงรายละเอียดของอาร์ก
Initial Marking	กำหนดการแสดงมาร์กกิงเริ่มต้น

ตารางที่ 4.1 ความหมายของตัวแปรในแฟ้มข้อมูล

ส่วนวิเคราะห์

ส่วนวิเคราะห์เป็นการวิเคราะห์แบบจำลองเพทริเน็ตโดยใช้วิธีรีชเอบิลิตีทรี ซึ่งให้ผลการวิเคราะห์เป็นรีชเอบิลิตีทรีและคุณสมบัติของแบบจำลองเพทริเน็ตที่สร้างขึ้นจากส่วนกราฟฟิกเอดิเตอร์ ก่อนที่จะทำการสร้างรีชเอบิลิตีทรีเราต้องแปลงแบบจำลองเพทริเน็ตให้อยู่ในรูปเมตริกซ์ที่คอมพิวเตอร์สามารถเข้าใจได้ เพื่อความสะดวกในการวิเคราะห์แบบจำลองเพทริเน็ต ดังจะเห็นได้จากโฟล์วชาร์ตแสดงการทำงานในส่วนวิเคราะห์ดังรูปที่ 4.21

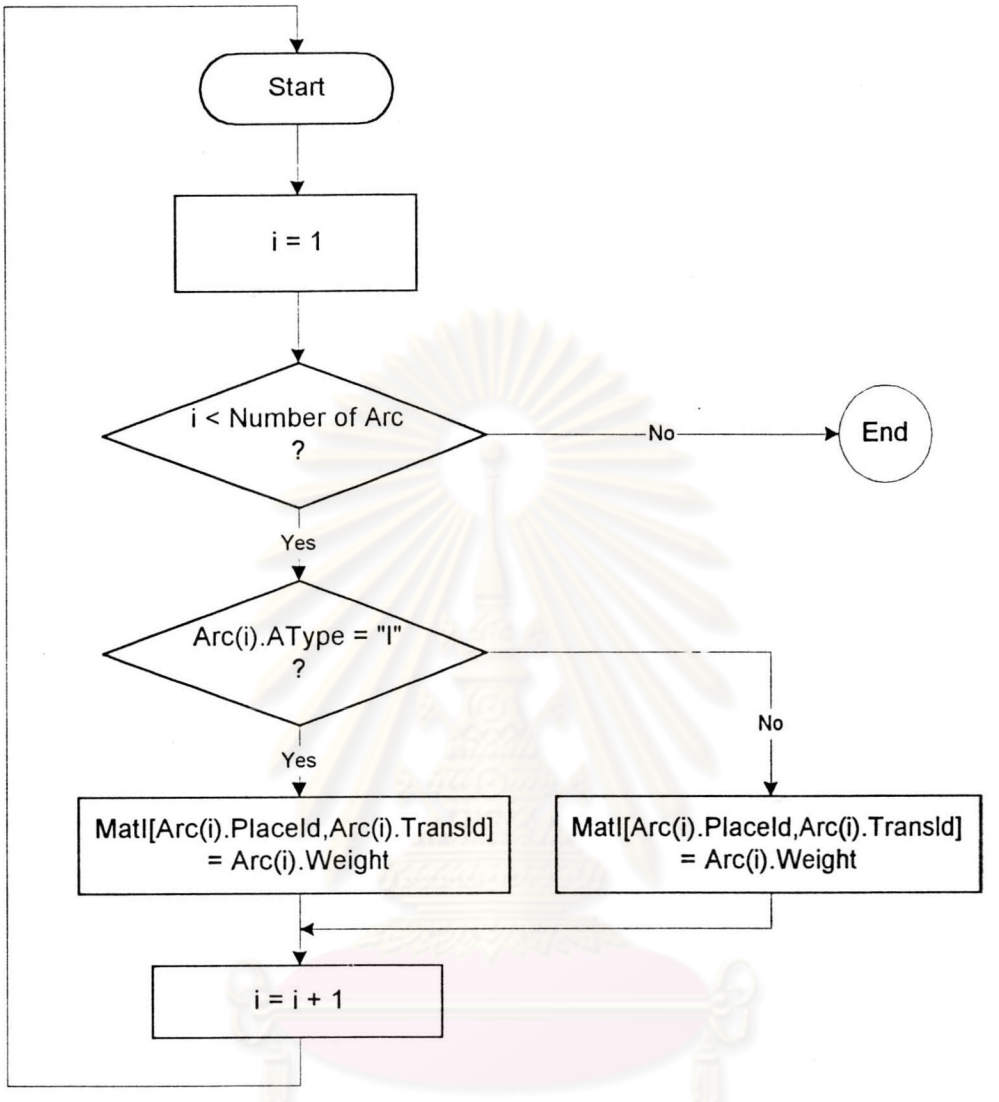


รูปที่ 4.21 โฟล์วชาร์ตแสดงการทำงานในส่วนวิเคราะห์

จากโฟล์วชาร์ตในรูปที่ 4.21 สามารถแสดงรายละเอียดการทำงานในแต่ละขั้นตอนได้ดังต่อไปนี้

1. การแปลงแบบจำลองเพทริเน็ตให้อยู่ในรูปเมตริกซ์

เมตริกซ์ที่ใช้ในการวิเคราะห์แบบจำลองเมตริกซ์มี 2 เมตริกซ์ คือ เมตริกซ์ของอินพุตฟังก์ชันและเอาต์พุตฟังก์ชัน ซึ่งเกิดจากอินพุตอาร์กและเอาต์พุตอาร์กตามลำดับ เนื่องจากข้อมูลแบบอาร์กในแบบจำลองเพทริเน็ตมีการเก็บชนิดของอาร์กที่วาดว่าเป็น อินพุตอาร์ก หรือ เอาต์พุตอาร์ก ดังนั้นจึงเป็นการง่ายที่จะแปลงข้อมูลแบบอาร์กให้อยู่ในรูปเมตริกซ์ที่ใช้ในการวิเคราะห์ได้ ซึ่งแสดงโฟล์วชาร์ตการแปลงแบบจำลองเพทริเน็ตให้อยู่ในรูปเมตริกซ์ได้ดังรูปที่ 4.22



รูปที่ 4.22 โฟล์วชาร์ตการแปลงแบบจำลองเพทริเน็ตให้อยู่ในรูปเมตริกซ์

2. <u>การสร้างรีชเอบิลิตีทรี</u>

ในส่วนนี้เป็นส่วนสร้างรีชเอบิลิตีทรีเพื่อใช้ประโยชน์ในการวิเคราะห์หาคุณสมบัติของแบบจำลองเพทริเน็ตต่อไป ซึ่งรีชเอบิลิตีทรีจะแสดงมาร์กกิงทั้งหมดของการเปลี่ยนแปลงของแบบจำลองเพทริเน็ตจากมาร์กกิงเริ่มต้น โดยมีโครงสร้างข้อมูลของรีชเอบิลิตีทรีแสดงดังรูปที่ 4.23

Level	Parent	Status	Fired	Marking	PosX

รูปที่ 4.23 โครงสร้างข้อมูลของรีชเอบิลิตีทรี

Level : ระดับชั้นของโนดในทรี

Parent : โนดก่อนหน้าที่ชี้มายังโนด

Status : สถานะของโนดประกอบด้วย

Terminal Node : โนดที่ไม่สามารถยิงต่อไปได้

Duplicate Node : โนดที่ซ้ำกับโนดอื่นที่อยู่ในทรี

Interior Node : โนดที่ผ่านการตรวจสอบแล้ว

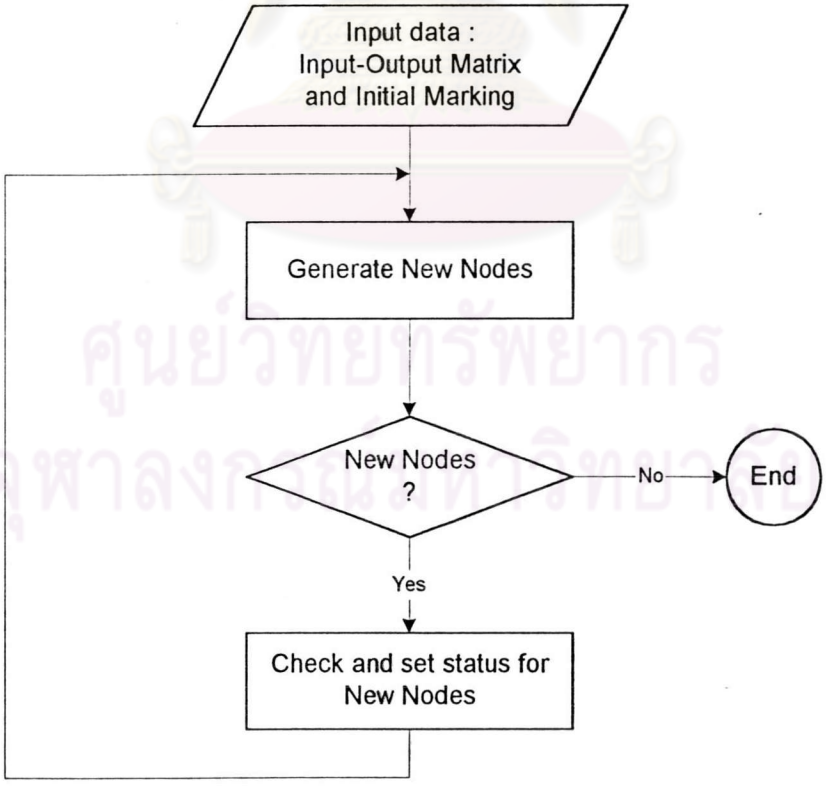
New Node : โนดที่ยังไม่ได้ผ่านการตรวจสอบ

Fired : หมายเลขทรานซิชันที่เป็นตัวยิงมายังโนด

Marking : แทนจำนวนโทเค็นในแต่ละเพลสของโนด

PosX : ใช้สำหรับปรับการแสดงผลของรีชเอบิลิตีทรี

ในการสร้างรีชเอบิลิตีทรีโดยใช้โครงสร้างข้อมูลที่กล่าวไปแล้วข้างต้น มีขั้นตอนและวิธีในการสร้างรีชเอบิลิตีทรีแสดงดังโฟล์วชาร์ตในรูปที่ 4.24



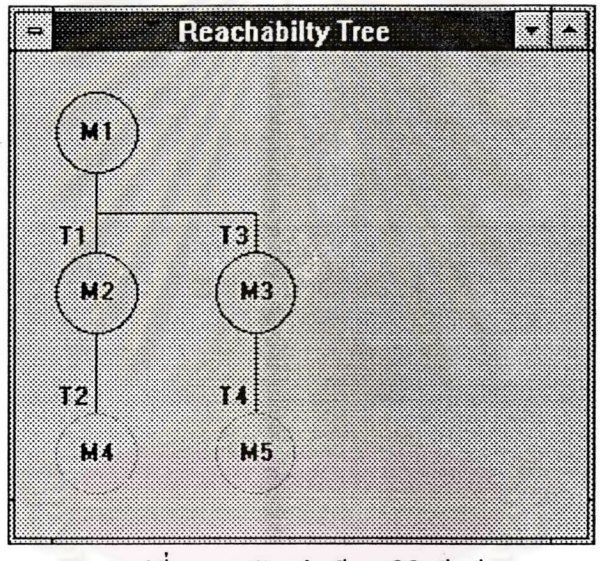
รูปที่ 4.24 โฟล์วชาร์ตการสร้างรีชเอบิลิตีทรี

ข้อมูลที่มาจากส่วนการแปลงแบบจำลองเพทริเน็ตให้อยู่ในรูปเมตริกซ์ ถูกส่งมายังส่วน สร้างรีชเอบิลิตีทรี โดยให้มาร์กกิงเริ่มต้นเป็นโนดเริ่มต้นในการสร้างรีชเอบิลิตีทรี หลังจากสร้างรีชเอบิลิตีทรีเสร็จแล้ว ก็จะนำรีชเอบิลิตีทรีที่ได้มาวิเคราะห์หาคุณสมบัติของแบบจำลองเพทริเน็ต ซึ่งแสดงตัวอย่างรีชเอบิลิตีทรีที่สร้างจากซอฟต์แวร์นี้ได้ดังรูปที่ 4.25 โดยใช้สีที่ล้อมรอบวงกลมในการแทนสถานะของแต่ละโนด คือ

สีดำแทนโนดปกติ

สีเขียวแทนโนดซ้ำซ้อน

สีแดงแทนเทอร์มินัลโนด



รูปที่ 4.25 ตัวอย่างรีชเอบิลิตรีทรี

3. <u>แสดงผลคุณสมบัติของแบบจำลองเพทริเน็ต</u>

ในการตรวจสอบคุณสมบัติของแบบจำลองเพทริเน็ตจะมุ่งเน้นไปยังจำนวนโทเค็นในแต่ละเพลสแต่ละโนดของรีชเอบิลิตีทรี และ สถานะของแต่ละโนดในรีชเอบิลิตีทรี ซึ่งสามารถแบ่งการตรวจสอบตามชนิดของแต่ละคุณสมบัติได้ดังนี้

3.1 Safeness

การตรวจสอบคุณสมบัติ Safeness นี้สามารถดูได้จากจำนวนโทเค็นในทุก ๆ เพลสของทุก ๆ โนดของรีชเอบิลิตีทรีจะต้องมีค่าไม่เกินหนึ่ง

3.2 Boundedness

การตรวจสอบคุณสมบัติ Boundedness นี้สามารถดูได้จากจำนวนโทเค็นในทุกๆ เพลสของทุก ๆ โนดของรีชเอบิลิตีทรีจะต้องมีค่าไม่มากกว่าจำนวนเต็มค่าหนึ่ง

3.3 Conservation

การตรวจสอบคุณสมบัติ Conservation นี้สามารถดูได้จากจำนวนโทเค็นในทุกๆ เพลสของรีชเอบิลิตีทรี จะต้องบวกกันได้เป็นค่าคงที่ค่าหนึ่งในทุก ๆ โนด

3.4 Liveness

การตรวจสอบคุณสมบัติ Liveness ทุก ๆ โนดของรีชเอบิลิตีทรี จะต้องไม่มีสถานะเป็นเทอร์มินัลโนด

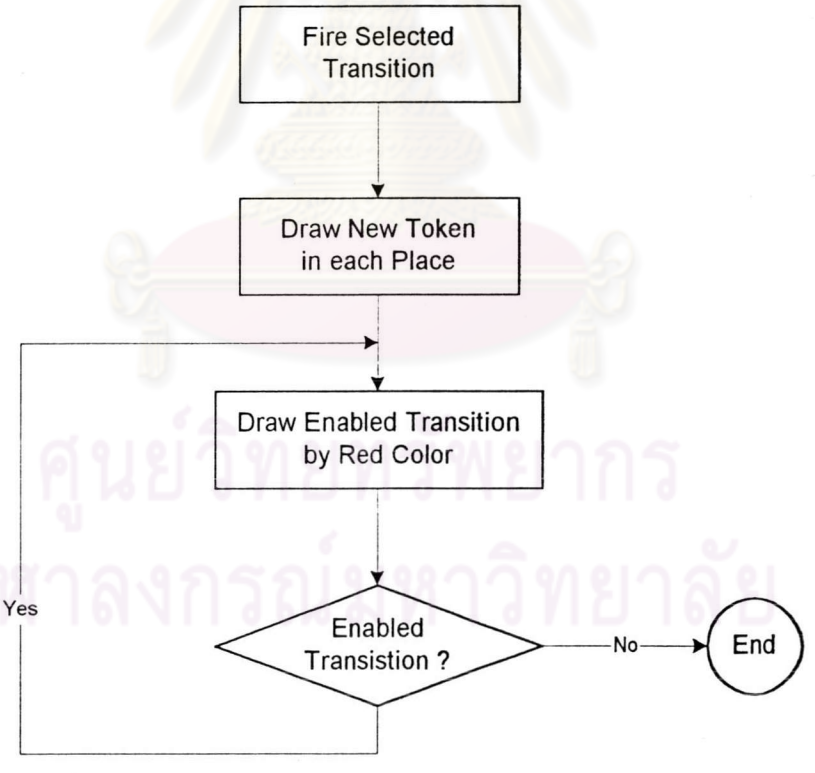
ซึ่งแสดงตัวอย่างผลการวิเคราะห์ดังรูปที่ 4.26 คือ ถ้าหลอดไฟติดที่คุณสมบัติใดแสดงว่าแบบจำลองเพทริเน็ตที่ออกแบบมีคุณสมบัตินั้น

รูปที่ 4.26 ตัวอย่างคุณสมบัติของแบบจำลองเพทริเน็ต

<u>ส่วนจำลองแบบการทำงานของเพทริเน็ตเชิงโต้ตอบ</u>

การจำลองแบบเชิงโต้ตอบเป็นพื้นฐานสำหรับใช้ทดสอบระบบที่ออกแบบ หรือใช้ในการศึกษาระบบ ซึ่งการจำลองแบบเชิงโต้ตอบแสดงให้เห็นถึงปัญหาขณะออกแบบได้ง่ายและมี

ประสิทธิภาพ ในซอฟต์แวร์นี้เป็นการจำลองแบบการทำงานของเพทริเน็ต ซึ่งมีกฎในการจำลองแบบการทำงานเหมือนกับกฎการทำงานของเพทริเน็ตที่กล่าวไว้แล้วในหัวข้อการทำงานของเพทริเน็ตในบทที่ 3 กล่าวคือ การทำงานของเพทริเน็ตถูกควบคุมโดยจำนวนและการกระจายของโทเค็นในแต่ละเพลส เพทริเน็ตทำงานโดยการยิงทรานซิชัน ทรานซิชันยิงโดยการขจัดโทเค็นจากอินพุตเพลสของทรานซิชัน และสร้างโทเค็นใหม่ที่เอาต์พุตเพลสของทรานซิชัน สำหรับซอฟต์แวร์นี้แสดงทรานซิชันที่อีนาเบิลและสามารถยิงได้ด้วยทรานซิชันสีแดง และในการกำหนดให้ทรานซิชันยิงขึ้นอยู่กับผู้ใช้ใช้เมาส์คลิกบนทรานซิชันที่ต้องการให้ยิง แต่ทรานซิชันจะยิงได้ก็ต่อเมื่อทรานซิชันนั้นอีนาเบิล หลังจากทรานซิชันยิงแล้วซอฟต์แวร์จะแสดงผลการกระจายโทเค็นใหม่ที่เพทริเน็ตเอดิเตอร์ เนื่องจากการทำงานของเพทริเน็ตขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้น ดังนั้นโฟล์วชาร์ตการจำลองแบบการทำงานของเพทริเน็ตที่แสดงดังรูปที่ 4.27 เป็นการอธิบายการทำงานเมื่อเกิดเหตุการณ์เมาส์คลิกบนทรานซิชันที่อีนาเบิล



รูปที่ 4.27 โฟล์วชาร์ตการจำลองแบบการทำงานของเพทริเน็ต