การพัฒนาเครื่องมือซอฟต์แวร์แปลงแผนภาพกระแสข้อมูล เป็นผังโครงสร้างของโปรแกรม

นายทวีเกียรติ เอี่ยมงามทรัพย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2542

ISBN 974-333-528-5

DEVELOPMENT OF A SOFTWARE TOOL FOR TRANSFORMING DATA FLOW DIAGRAMS TO STRUCTURE CHARTS

MR. TAWEEKIAT EIAMNGAMSUB

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 1999

ISBN 974-333-528-5

หัวข้อวิทยานิพนธ์	การพัฒนาเครื่องมือซอฟต์แวร์แปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม
โดย	นายทวีเกียรติ เอี่ยมงามทรัพย์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.ธัชชัย สุมิตร)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ วันพร ปั้นเก่า)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการ

(อาจารย์จารุมาตร ปิ่นทอง)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ เมธี ศรีสังวาล)

ทวีเกียรติ เอี่ยมงามทรัพย์ : การพัฒนาเครื่องมือซอฟต์แวร์แปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม (DEVELOPMENT OF A SOFTWARE TOOL FOR TRANSFORMING DATA FLOW DIAGRAMS TO STRUCTURE CHARTS) อ.ที่ปรึกษา : อ.ดร. ธาราทิพย์ สุวรรณศาสตร์ , 71 หน้า, ISBN 974-333-528-5

จุดมุ่งหมายของการทำวิจัยคือ เพื่อพัฒนาเครื่องมือซอฟต์แวร์ช่วยแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรม โดยใช้วิธีวิเคราะห์แบบแปลงและวิธีวิเคราะห์แบบทรานแซกชัน ซึ่งจะช่วยประหยัดเวลาและค่าใช้จ่ายในการพัฒนาโปรแกรมได้ นอกจากนี้ เครื่องมือซอฟต์แวร์ที่พัฒนายังสามารถวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของผังโครงสร้างของโปรแกรมได้ในกรณีที่ผู้ใช้ระบุคำอธิบายการทำงานของมอดูลในรูปแบบภาษาในการออกแบบโปรแกรม ซึ่งค่าความสัมพันธ์ต่อกันระหว่างมอดูลนี้เป็นเครื่องมือวัดชนิดหนึ่งที่บอกถึงคุณภาพในการออกแบบผังโครงสร้างของโปรแกรม

จากผลจากวิจัยที่ได้ เครื่องมือซอฟต์แวร์สามารถแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรม และสามารถวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของแต่ละมอดูลในผังโครงสร้างของโปรแกรมและค่าเฉลี่ยของความสัมพันธ์ต่อกันระหว่างมอดูลของผังโครงสร้างของโปรแกรมได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต ทวีเกียรติ เอี่ยมงามทรัพย์

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา ธาราทิพย์ สุวรรณศาสตร์

ปีการศึกษา 2542

TAWEEKIAT EIAMNGAMSUB: DEVELOPMENT OF A SOFTWARE TOOL FOR TRANSFORMING DATA FLOW DIAGRAMS TO STRUCTURE CHARTS. THESIS ADVISOR: TARATIP SUWANNASART, Ph.D., 71 pp., ISBN 974-333-528-5

The objective of this research is to develop a software tool to construct structure charts from data flow diagrams using transform analysis and transaction analysis method. By using this tool, software developer will save time and expense in development process. The software tool can measure module coupling level of structure charts based on every module specification that is written in program design language. The module coupling is a measuring tool that indicates the quality of a structure chart.

As the result from this research, the software tool can transform data flow diagrams to structure charts. The tool also results in a good measure module coupling level from module specification. The tool can measure module coupling for each module in a structure chart and average coupling of the structure chart.

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต ทวีศักดิ์ เอี่ยมงามทับพ์

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา ธนเทพ สุนทรเจริญ

ปีการศึกษา 2542

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ได้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือของคณาจารย์หลายท่านโดยเฉพาะอย่างยิ่ง ขอขอบพระคุณ ดร.ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้ข้อชี้แนะและคำแนะนำต่างๆ ในการวิจัยเป็นอย่างดี และขอขอบพระคุณอาจารย์ท่านอื่นๆ ในห้องปฏิบัติการวิศวกรรมซอฟต์แวร์ที่ให้คำแนะนำและข้อคิดเห็นต่างๆ ของการวิจัยมาด้วยดีมาตลอด และท่านอื่นๆ ที่ไม่ได้กล่าวข้างต้น ซึ่งผู้วิจัยขอขอบพระคุณเป็นอย่างยิ่ง ขอขอบคุณห้องปฏิบัติการวิศวกรรมซอฟต์แวร์ที่ได้เอื้อเฟื้อสถานที่และทรัพยากรต่างๆ

ผู้วิจัยขอขอบคุณบริษัทไมโครเอ็กซ์จำกัดที่ได้ให้ทุนการศึกษาแก่ผู้วิจัยและให้ความสนับสนุนในการศึกษาของผู้วิจัยมาโดยตลอด นอกจากนี้ ผู้วิจัยขอขอบคุณเพื่อนนิสิตในห้องปฏิบัติการวิศวกรรมซอฟต์แวร์และเพื่อนๆ นิสิตที่จุฬาลงกรณ์มหาวิทยาลัยที่ให้คำแนะนำและเป็นกำลังใจที่ดีเสมอมา

ท้ายนี้ ผู้วิจัยใคร่ขอกราบขอบพระคุณ บิดา-มารดา และขอบคุณทุกคนในครอบครัวที่ให้กำลังใจและสนับสนุนผู้วิจัยเป็นอย่างดียิ่งมาตลอดจนสำเร็จการศึกษา

นาย ทวีเกียรติ เอี่ยมงามทรัพย์

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การวิเคราะห์และออกแบบระบบงานเชิงโครงสร้าง (Structured System Analysis and Design) [5] เป็นวิธีการหนึ่งที่ได้รับความนิยมในการนำมาใช้เพื่อวิเคราะห์และออกแบบโปรแกรมระบบงานสำหรับทางธุรกิจ เนื่องจากเป็นวิธีการที่เข้าใจง่าย มีกระบวนการและขั้นตอนที่ชัดเจน และสามารถวัดถึงคุณภาพของการออกแบบโปรแกรมได้ เครื่องมือที่สำคัญอย่างหนึ่งในกระบวนการออกแบบเชิงโครงสร้าง คือ ผังโครงสร้างของโปรแกรม (Structure Chart) ซึ่งสามารถทำได้โดยแปลงจากแผนภาพกระแสข้อมูล (Data Flow Diagram) โดยใช้วิธีวิเคราะห์แบบแปลง (Transform Analysis) และวิธีวิเคราะห์แบบทรานแซกชัน (Transaction Analysis) [2]

ผังโครงสร้างของโปรแกรม เป็นแผนภาพเชิงลำดับชั้นที่แสดงถึงสถาปัตยกรรมของโปรแกรม ว่าประกอบด้วยมอดูล (Module) อะไรบ้างและแต่ละมอดูลสัมพันธ์กันอย่างไร จึงเป็นเครื่องมือที่จะช่วยนักออกแบบและนักวิเคราะห์ระบบสามารถมองเห็นภาพรวมของโปรแกรม ว่ามีความซับซ้อนมากน้อยเพียงใด จำเป็นที่จะต้องปรับเปลี่ยนการออกแบบตัวระบบงานหรือไม่ อย่างไรก็ตาม ในระบบงานที่มีขนาดใหญ่และมีความซับซ้อนนั้น การสร้างผังโครงสร้างของโปรแกรมโดยแปลงจากแผนภาพกระแสข้อมูลจะทำได้ยากยิ่งขึ้น โดยเฉพาะสำหรับนักออกแบบที่ยังไม่คุ้นเคยกับวิธีการดังกล่าว ทำให้ต้องใช้เวลาในการออกแบบระบบโดยรวมเพิ่มมากขึ้นด้วย

1.2 วัตถุประสงค์ของการวิจัย

เพื่อสร้างเครื่องมือซอฟต์แวร์ในการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม

1.3 ขอบเขตของการวิจัย

1.3.1 ข้อมูลนำเข้าที่ใช้แสดงแผนภาพกระแสข้อมูล จะเป็นแฟ้มข้อมูลที่ได้จากเครื่องมือซอฟต์แวร์ที่สามารถวาดแผนภาพกระแสข้อมูลได้

1.3.2 วิธีที่ใช้ในการแปลง คือ วิธีวิเคราะห์แบบแปลง และวิธีวิเคราะห์แบบทรานแซกชัน ซึ่งสามารถทำงานได้ทั้งแบบอัตโนมัติ หรือให้ผู้ใช้กำหนดส่วนของศูนย์กลางการแปลงหรือศูนย์กลางทรานแซกชัน

1.3.3 ผลลัพธ์ของการแปลงจะได้ผังโครงสร้างของโปรแกรมในรูปแบบต่างๆให้ผู้ใช้เลือก

1.3.4 สามารถบอกถึงความสัมพันธ์ต่อกันระหว่างมอดูลได้ โดยใช้วิธีการของออฟฟัต (Offut) ในกรณีที่มีการบันทึกวิธีการทำงานของกระบวนการโดยใช้ภาษาในการออกแบบโปรแกรม

1.3.5 ภาษาที่ใช้ในการพัฒนาเครื่องมือซอฟต์แวร์ คือ ภาษาซีพลัสพลัสหรือภาษาระดับสูงบนเครื่องไมโครคอมพิวเตอร์ที่ใช้หน่วยประมวลผลกลางแบบเพนเทียม หน่วยความจำขั้นต่ำ 16 MB ภายใต้โปรแกรมไมโครซอฟต์วินโดวรุ่น 95 ขึ้นไป

1.4 ขั้นตอนการดำเนินการวิจัย

1.4.1 ศึกษาวิธีการแปลงแผนภาพกระแสข้อมูล เป็นผังโครงสร้างของโปรแกรม

1.4.2 ศึกษาวิธีตรวจสอบความสัมพันธ์ต่อกันระหว่างมอดูลตามวิธีการของออฟฟัต

1.4.3 ออกแบบเครื่องมือซอฟต์แวร์ในการแปลงแผนภาพกระแสข้อมูล เป็นผังโครงสร้างของโปรแกรม

1.4.4 พัฒนาเครื่องมือซอฟต์แวร์ในการแปลงแผนภาพกระแสข้อมูล เป็นผังโครงสร้างของโปรแกรม

1.4.5 ทดสอบและประเมินผลการทำงานของเครื่องมือซอฟต์แวร์

1.4.6 จัดทำคู่มือการใช้เครื่องมือซอฟต์แวร์ที่พัฒนาขึ้น

1.4.7 สรุปผลการวิจัย เสนอแนะ และจัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถมีเครื่องมือซอฟต์แวร์ช่วยในการแปลงแผนภาพกระแสข้อมูล เป็นผังโครงสร้างของโปรแกรมโดยอัตโนมัติ

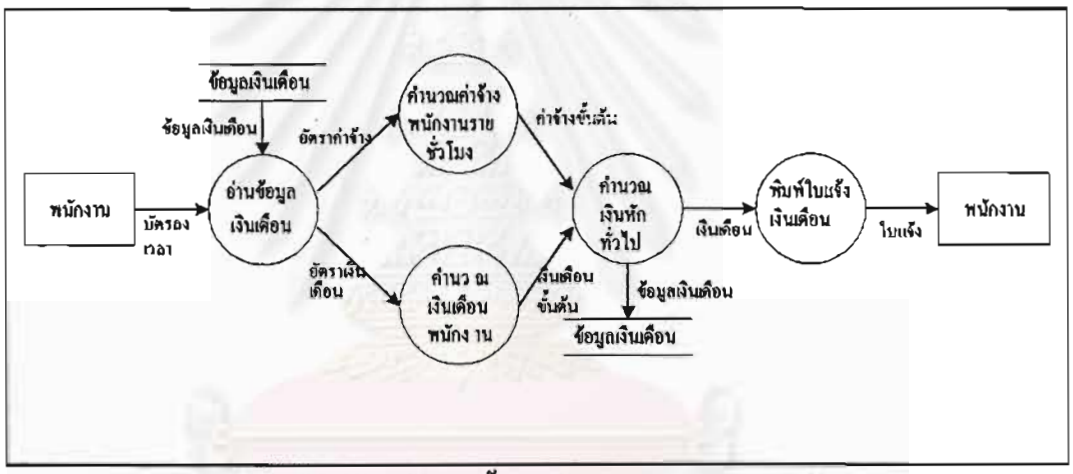
1.5.2 ช่วยประหยัดเวลาและค่าใช้จ่ายของการพัฒนาโปรแกรม

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 แผนภาพกระแสข้อมูล (Data Flow Diagram) [1]

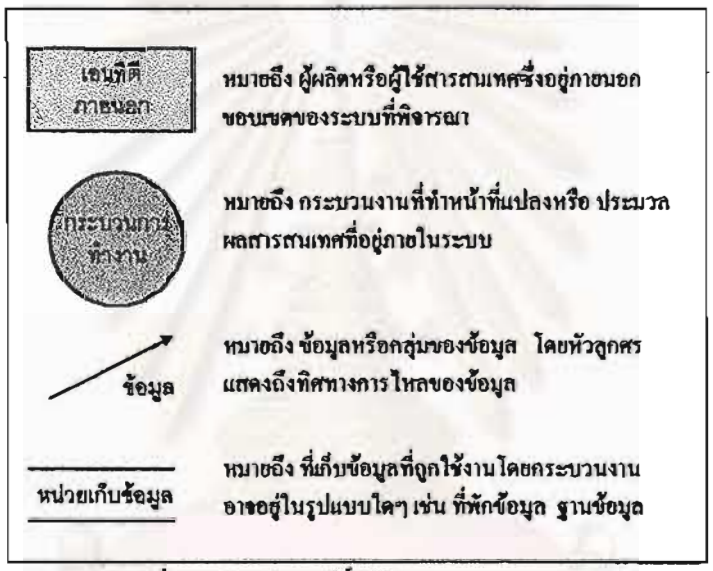
เมื่อสารสนเทศเข้าสู่ระบบงานซอฟต์แวร์ สารสนเทศนั้นจะถูกประมวลผลตามลำดับของการประมวลผลตามขั้นตอนการทำงานซอฟต์แวร์ แผนภาพกระแสข้อมูลเป็นเทคนิคทางรูปภาพที่ใช้แสดงถึงการไหลของสารสนเทศและขั้นตอนการประมวลผลสารสนเทศที่ถูกนำเข้าสู่ระบบไปเป็นผลลัพธ์ของระบบ ตัวอย่างของรูปแบบพื้นฐานของแผนภาพกระแสข้อมูลสามารถแสดงดังในรูปที่ 2.1 ซึ่งเป็นแผนภาพกระแสข้อมูลของระบบคำนวณเงินเดือน



รูป 2.1 ตัวอย่างรูปแบบพื้นฐานของแผนภาพกระแสข้อมูล

แผนภาพกระแสข้อมูลอาจถูกใช้เพื่อแสดงถึงระบบงาน (System) หรือซอฟต์แวร์ที่ระดับใดๆ ของแนวคิดแบบนามธรรม (Abstraction) ได้ ในทางปฏิบัตินั้น แผนภาพกระแสข้อมูลถูกแบ่งออกเป็นระดับย่อยลงไปเพื่อแสดงรายละเอียดของการไหลของสารสนเทศและหน้าที่การทำงานให้เพิ่มมากขึ้น โดยที่ระดับที่ใหญ่ที่สุดหรือระดับศูนย์ของแผนภาพกระแสข้อมูลจะเรียกว่า ตัวแบบมูลฐานของระบบ (Fundamental System Model) หรือ ตัวแบบบริบท (Context Model) ซึ่งจะแสดงระบบทั้งหมดด้วยวงกลมหนึ่งวง ที่มีข้อมูลนำเข้าและข้อมูลผลลัพธ์ซึ่งแสดงด้วยเส้นลูกศรชี้เข้าและชี้ออกตามลำดับ โดยที่รายละเอียดเพิ่มเติมของระบบและข้อมูลที่ไหลในระดับศูนย์นั้นสามารถแสดงโดยแบ่งออกเป็นระดับที่หนึ่ง โดยที่แต่ละวงกลมในระดับที่หนึ่งจะเรียกว่า กระบวนการทำงาน (Process) ซึ่งเป็นหน้าที่ย่อยของระบบทั้งหมดที่อยู่ในตัวแบบบริบท

สัญลักษณ์พื้นฐานที่ใช้ในการสร้างแผนภาพกระแสข้อมูลสามารถแสดงได้ดังรูปที่ 2.2 รูปสี่เหลี่ยมจะใช้แสดงถึงเอนทิตีภายนอก (External Entity) ซึ่งเป็นองค์ประกอบของระบบ เช่น ฮาร์ดแวร์ บุคคล องค์กร ซอฟต์แวร์อื่นที่เกี่ยวข้อง เป็นต้น หรือระบบงานอื่นที่สร้างสารสนเทศเข้าสู่ซอฟต์แวร์ หรือระบบงานอื่นที่รับสารสนเทศจากซอฟต์แวร์ รูปวงกลมใช้แสดงถึงกระบวนการทำงานซึ่งกระทำกับข้อมูลเพื่อแก้ไขข้อมูลในรูปแบบใดรูปแบบหนึ่ง เส้นลูกศรแสดงถึงชิ้นข้อมูลหนึ่งข้อมูลหรือมากกว่านั้น โดยเส้นลูกศรทั้งหมดในแผนภาพจะต้องกำกับชื่อไว้ด้วย สำหรับเส้นคู่จะแสดงถึงหน่วยเก็บข้อมูล (Data Store) ซึ่งใช้จัดเก็บสารสนเทศที่ซอฟต์แวร์ใช้งาน ความง่ายของสัญลักษณ์ของแผนภาพกระแสข้อมูลนี้เป็นเหตุผลหนึ่งที่ช่วยให้เทคนิคการวิเคราะห์แบบโครงสร้าง (Structured Analysis Techniques) เป็นที่นิยมใช้กันอย่างแพร่หลาย



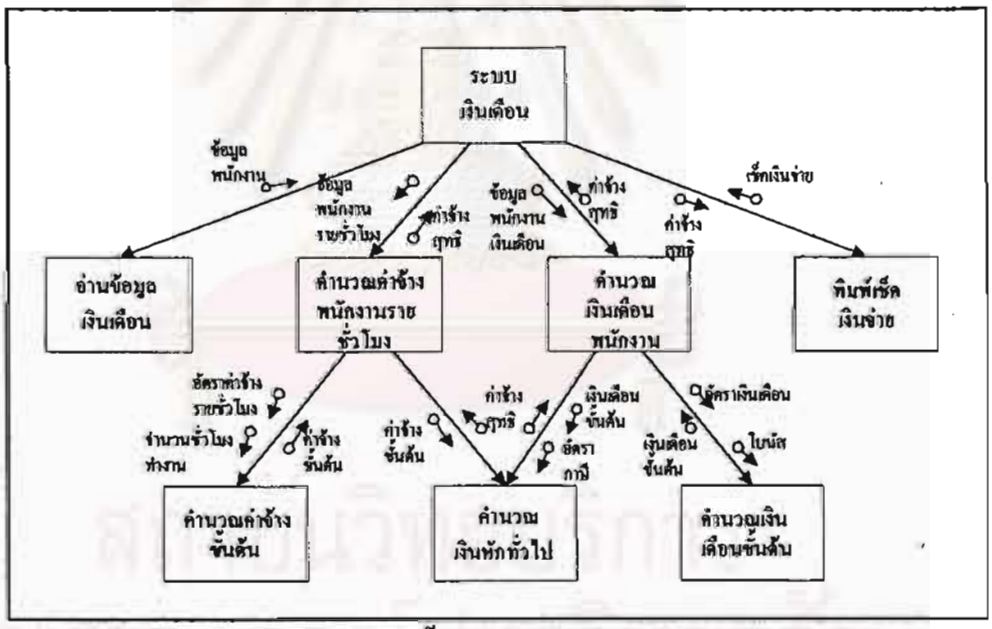
รูปที่ 2.2 สัญลักษณ์ที่ใช้ในแผนภาพกระแสข้อมูล

สำหรับแผนภาพกระแสข้อมูลของระบบคำนวณเงินเดือนในรูปที่ 2.1 จะประกอบด้วยเอนทิตีภายนอกเดียว คือ พนักงาน โดยมีกระบวนการทำงาน 5 กระบวนการ เริ่มจากกระบวนการทำงาน "อ่านข้อมูลเงินเดือนพนักงาน" ซึ่งจะรับข้อมูลนำเข้าคือบัตรลงเวลาทำงานของพนักงานและอ่านรายละเอียดข้อมูลของพนักงานและอัตราค่าจ้างจากแฟ้มข้อมูลเงินเดือน ถ้าข้อมูลพนักงานที่กำลังประมวลผลเป็นพนักงานที่จ้างแบบรายชั่วโมงจะเรียกกระบวนการทำงาน "คำนวณค่าจ้างพนักงานรายชั่วโมง" ขึ้นมาทำงาน แต่ถ้าเป็นข้อมูลพนักงานประจำที่รับเงินเดือนจะเรียกกระบวนการทำงาน "คำนวณเงินเดือนพนักงาน" ขึ้นมาทำงาน จากนั้นจะมีการคิดค่าเงินหักจากค่าจ้างขั้นต้นหรือเงินเดือนขั้นต้น เช่น ค่าภาษี ด้วยกระบวนการทำงาน "คำนวณเงินหักทั่วไป" เมื่อได้ค่าเงินเดือนแล้วจะบันทึกข้อมูลเงินเดือนเพิ่มเข้าไปยังแฟ้มข้อมูลเงินเดือน แล้วเรียกกระบวนการทำงาน "พิมพ์ใบแจ้งเงินเดือน" เพื่อพิมพ์ใบแจ้งเงินเดือนให้กับพนักงาน

จุดสำคัญอย่างหนึ่งเกี่ยวกับแผนภาพกระแสข้อมูล คือ แผนภาพกระแสข้อมูลไม่ได้แสดงถึงลำดับของกระบวนการที่แน่นอน แม้ว่าขั้นตอนหรือลำดับอาจแสดงโดยอุปนัยอยู่ในแผนภาพ โดยปกติแล้ว ลำดับที่แน่นอนนั้นจะระบุได้ก็ต่อเมื่อเข้าสู่ระยะการออกแบบซอฟต์แวร์แล้วเท่านั้น

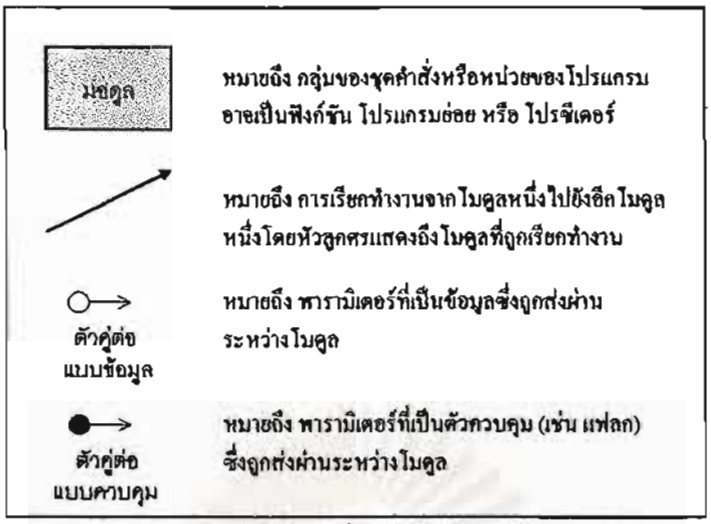
2.2 ผังโครงสร้างของโปรแกรม (Structure Chart) [11]

ผังโครงสร้างของโปรแกรมใช้เพื่อแสดงถึงโครงสร้างของชุดคำสั่งที่อยู่ในโปรแกรม ซึ่งอยู่ในลักษณะของโครงสร้างแบบลำดับชั้น (Hierarchical Structure) โดยที่แต่ละหน่วยของชุดคำสั่งในผังโครงสร้างนี้จะเรียกว่า มอดูล ซึ่งหมายถึงแต่ละหน่วยของโปรแกรม เช่น ฟังก์ชัน (Function) หรือ ซับรูทีน (Sub-routine) หรือ กระบวนคำสั่ง (Procedure) เป็นต้น ผังโครงสร้างของโปรแกรมจะแสดงให้เห็นถึงโครงสร้างของโปรแกรมว่าประกอบด้วยมอดูลอะไรบ้างและแสดงถึงการทำงานของโปรแกรมว่าแต่ละมอดูลมีการเรียกทำงานกันอย่างไร ตัวอย่างของรูปแบบพื้นฐานของผังโครงสร้างของโปรแกรมแสดงดังรูปที่ 2.3



รูป 2.3 ตัวอย่างรูปแบบพื้นฐานของผังโครงสร้างของโปรแกรม

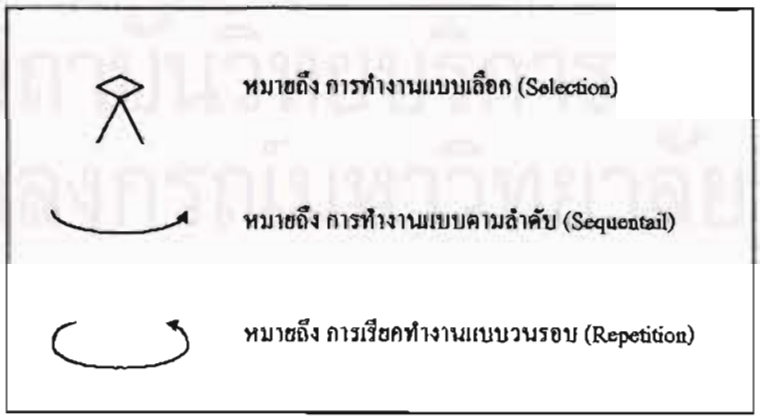
สัญลักษณ์พื้นฐานที่ใช้ในการแสดงถึงผังโครงสร้างของโปรแกรมแสดงได้ดังรูปที่ 2.4 ซึ่งประกอบด้วยกล่องสี่เหลี่ยมที่ใช้แสดงถึงมอดูล ซึ่งภายในกล่องสี่เหลี่ยมจะเป็นชื่อของมอดูลที่บอกให้รู้ว่ามอดูลนี้ถูกออกแบบมาเพื่อใช้ทำหน้าที่อะไร เส้นลูกศรที่เชื่อมโยงระหว่างมอดูลใช้แสดงถึงการเรียกทำงานระหว่างกันของมอดูล โดยมอดูลที่อยู่ข้างบนของเส้นลูกศรจะเป็นมอดูลตั้งต้นที่ทำหน้าที่เรียกมอดูลที่อยู่ข้างล่างขึ้นมาทำงาน



รูป 2.4 สัญลักษณ์ที่ใช้ในผังโครงสร้างของโปรแกรม

สำหรับเส้นลูกศรเล็กๆ ที่อยู่ข้างเส้นลูกศรที่เชื่อมโยงระหว่างมอดูลนั้น เรียกว่า ตัวคู่ต่อ (Couple) ซึ่งใช้แทนการส่งพารามิเตอร์ (Parameter) ถึงกันในการเรียกทำงานระหว่างมอดูล โดยที่หัวลูกศรของตัวคู่ต่อ จะแสดงให้เห็นถึงทิศทางการไหลของพารามิเตอร์ ตัวคู่ต่อจะมีสองประเภท คือ ตัวคู่ต่อแบบข้อมูล (Data Couple) เป็นพารามิเตอร์ที่ใช้เป็นข้อมูลซึ่งแสดงด้วยเส้นลูกศรเล็กๆ ที่มีหัวเป็นวงกลมว่าง กับตัวคู่ต่อแบบควบคุม (Control Couple) เป็นพารามิเตอร์ที่ใช้เป็นตัวควบคุมการทำงาน ซึ่งแสดงด้วยเส้นลูกศรเล็กๆ ที่มีหัวเป็นวงกลมทึบ

สัญลักษณ์เพิ่มเติมที่นิยมใช้เพื่อแสดงถึงรูปแบบของการเรียกทำงานระหว่างมอดูลแสดงดังในรูปที่ 2.5 ได้แก่ การเรียกทำงานโดยลำดับ (Sequential) ซึ่งแสดงด้วยเส้นโค้ง การเรียกทำงานแบบเลือก (Selection) ซึ่งแสดงด้วยรูปเพชร (Diamond) ทางด้านล่างของมอดูลที่มีคำสั่งในการเลือก และการเรียกทำงานแบบวนรอบ (Repetition) ซึ่งแสดงด้วยเส้นโค้งหมุนวน



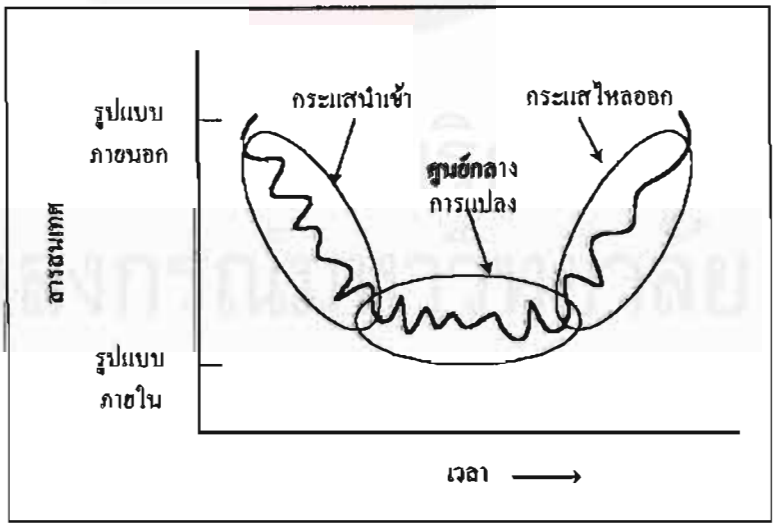
รูปที่ 2.5 สัญลักษณ์เพิ่มเติมที่ใช้ในผังโครงสร้างของโปรแกรม

ผังโครงสร้างของโปรแกรมจะทำให้เห็นถึงภาพรวมของวัตถุประสงค์และโครงสร้างของโปรแกรม อย่างไรก็ตาม แม้ว่าจากผังโครงสร้างของโปรแกรมสามารถที่จะบอกได้ว่าแต่ละมอดูลทำหน้าที่อะไรโดยพิจารณาจากชื่อมอดูลและพารามิเตอร์ที่ใช้งาน แต่จะไม่สามารถบอกได้ว่ามอดูลทำงานอย่างไร ซึ่งจำเป็นที่จะต้องมีข้อกำหนดประกอบ (Supporting Specification) มาใช้ร่วมกัน นอกจากนี้ ลำดับของมอดูลจากด้านซ้ายไปขวานั้นไม่ได้แสดงถึงลำดับการทำงานที่เกิดขึ้นจริงของโปรแกรมเสมอไป แม้ว่าโดยทั่วไปแล้วมักจะนิยมเขียนลำดับจากซ้ายไปขวาที่สอดคล้องกับลำดับการทำงานของโปรแกรมเพื่อความเข้าใจในการอ่านผังโครงสร้างของโปรแกรม

2.3 การแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม (Transform Data Flow Diagrams to Structure Charts) [1]

2.3.1 คุณลักษณะของแผนภาพกระแสข้อมูล

จากตัวแบบบริบทของระบบงานในแผนภาพกระแสข้อมูลในระดับศูนย์นั้น สารสนเทศจะไหลเข้าและออกจากซอฟต์แวร์ในลักษณะที่เป็นรูปแบบภายนอก เช่น ข้อมูลชื่อลูกค้าที่พิมพ์เข้ามาจากแป้นพิมพ์ เสียงสัญญาณจากสายโทรศัพท์ เป็นต้น ซึ่งข้อมูลที่เป็นรูปแบบภายนอกนี้จะถูกแปลงเป็นรูปแบบภายในสำหรับกระบวนการ เช่น รหัสลูกค้า ข้อมูลหมายเลขโทรศัพท์ เป็นต้น โดยสามารถแสดงประวัติของข้อมูลตามเวลาได้ดังรูปที่ 2.6 สารสนเทศจะเข้าสู่ระบบตามเส้นทางที่แปลงข้อมูลภายนอกมายังข้อมูลที่เป็นรูปแบบภายใน เรียกเส้นทางนี้ว่า กระแสนำเข้า (Incoming Flow) จากนั้น ข้อมูลจะไหลเข้ามายังแก่นกลางของซอฟต์แวร์ซึ่งเป็นส่วนของการ

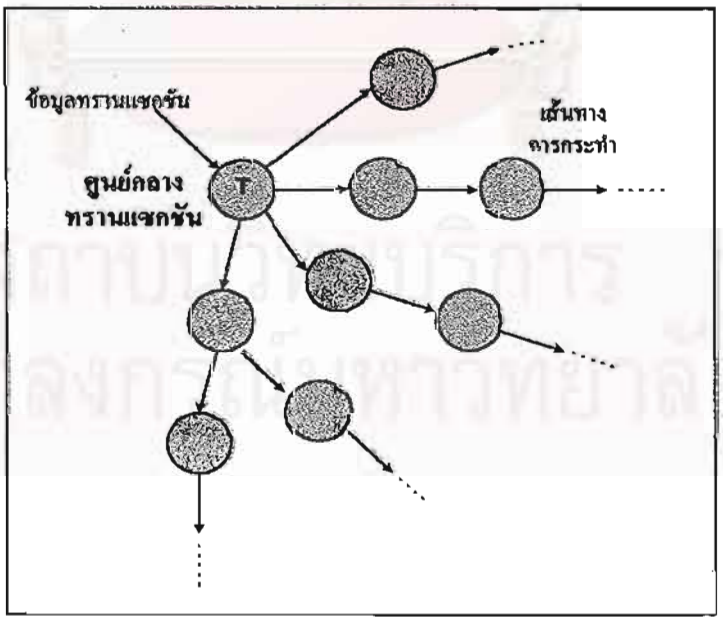


รูปที่ 2.6 การไหลของสารสนเทศในซอฟต์แวร์ [1]

ประมวลผลหรือแปลงข้อมูลนำเข้าให้เป็นข้อมูลผลลัพธ์ เรียกบริเวณแก่นกลางนี้ว่า ศูนย์กลางการแปลง (Transform Center) และในที่สุด ข้อมูลผลลัพธ์จะไหลออกมาผ่านเส้นทางที่จะส่งข้อมูลออกจากซอฟต์แวร์ เรียกเส้นทางนี้ว่า กระแสไหลออก (Outgoing Flow) การไหลของข้อมูลทั้งหมดนี้จะเกิดขึ้นทีละลำดับตามเส้นทางที่เป็นเส้นตรงหรือกลุ่มของเส้นทางจำนวนไม่มากนัก เมื่อส่วนของแผนภาพกระแสข้อมูลมีคุณลักษณะตามที่กล่าวมานี้ จะถือว่าแผนภาพมีคุณลักษณะแบบกระแสการแปลง (Transform Flow)

โดยทั่วไปแล้ว มูลฐานของระบบงาน (System Context) จะแสดงโดยอุปนัยถึงลักษณะแบบกระแสการแปลง ดังนั้น จึงสามารถจัดคุณลักษณะของกระแสข้อมูลทั้งหมดให้อยู่ในประเภทนี้ได้ อย่างไรก็ตาม บ่อยครั้งที่กระแสข้อมูลจะมีคุณลักษณะที่ขึ้นกับชิ้นข้อมูลเพียงตัวเดียว ซึ่งเรียกว่า ทรานแซกชัน (Transaction) ซึ่งเป็นตัวกระตุ้นให้เกิดการไหลของข้อมูลไปยังเส้นทางหนึ่งในหลายๆ เส้นทาง เมื่อแผนภาพกระแสข้อมูลมีลักษณะเช่นนี้ปรากฏขึ้น ดังแสดงตัวอย่างในรูปที่ 2.7 จะถือว่าแผนภาพมีคุณลักษณะแบบกระแส ทรานแซกชัน (Transaction Flow)

กระแสทรานแซกชันจะมีลักษณะคือข้อมูลไหลเข้ามาตามเส้นทางนำเข้า ซึ่งบางครั้งจะเรียกว่า เส้นทางรับเข้า (Reception Path) ที่ทำหน้าที่แปลงสารสนเทศจากภายนอกให้อยู่ในรูปของทรานแซกชัน ทรานแซกชันจะถูกประเมินค่าตามค่าของตัวมันและทำให้เกิดการไหลไปตามเส้นทางหนึ่งจากหลายเส้นทางที่เรียกว่า เส้นทางการกระทำ (Action Paths) โดยจุดรวมที่เป็นตำแหน่งเริ่มต้นของเส้นทางการกระทำต่างๆ นั้น จะเรียกว่า ศูนย์กลางทรานแซกชัน (Transaction Center)



รูปที่ 2.7 ลักษณะการไหลแบบทรานแซกชัน [1]

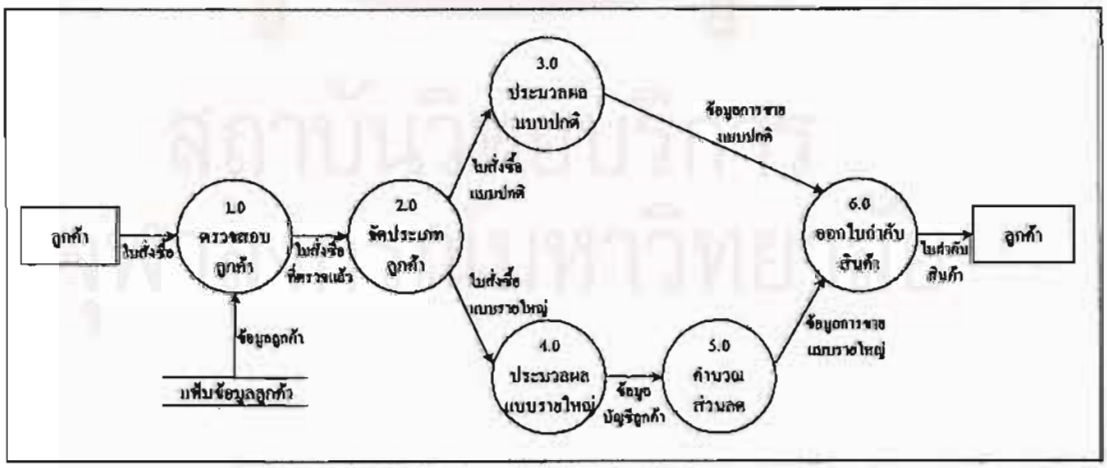
สำหรับแผนภาพกระแสข้อมูลที่มีขนาดใหญ่ๆ นั้น ทั้งคุณลักษณะแบบกระแสการแปลงและกระแสทรานแซกชันอาจจะเกิดขึ้นร่วมกันได้ เช่น แผนภาพที่มีรูปแบบเป็นกระแสทรานแซกชันนั้น การไหลของสารสนเทศที่เกิดขึ้นในเส้นทางการกระทำอาจจะมีคุณลักษณะแบบกระแสการแปลงได้

2.3.2 การวิเคราะห์แบบแปลง (Transform Analysis) [1]

การวิเคราะห์แบบแปลง เป็นกลุ่มของขั้นตอนการออกแบบซึ่งใช้กับแผนภาพกระแสข้อมูลที่มีคุณลักษณะของกระแสการแปลง (Transform Flow) เพื่อจับคู่มายังผังโครงสร้างของโปรแกรมที่ได้กำหนดต้นแบบไว้ล่วงหน้าแล้ว โดยมีขั้นตอนต่างๆ ดังนี้

1) ตรวจทานตัวแบบบริบทของระบบงาน ซึ่งตัวแบบบริบทของระบบงานนั้นจะแสดงโดยนัยอยู่ในแผนภาพกระแสข้อมูลในระดับศูนย์และสารสนเทศสนับสนุนอื่นๆ ซึ่งในเอกสารข้อกำหนดของระบบงาน (System Specification) และเอกสารข้อกำหนดความต้องการซอฟต์แวร์ (Software Requirement Specification) นั้นจะอธิบายถึงการไหลของสารสนเทศและโครงสร้างของสารสนเทศที่ทำงานติดต่อกับซอฟต์แวร์

2) ตรวจทานและปรับแต่งรายละเอียดของแผนภาพกระแสข้อมูล ในขั้นตอนนี้จะทำการขยายรายละเอียดของแผนภาพกระแสข้อมูลที่ได้จากเอกสารข้อกำหนดความต้องการซอฟต์แวร์ให้มากขึ้น โดยอย่างน้อยควรลงถึงระดับที่ 2 ถึง 3 เพื่อให้แต่ละกระบวนการมีระดับของความเกาะตัว (Cohesion) ค่อนข้างมาก เพียงพอที่จะใช้ในการสร้างผังโครงสร้างของโปรแกรมแบบโครงร่าง (First Cut Structure Chart) ได้ ซึ่งในที่นี้จะใช้ตัวอย่างของแผนภาพกระแสข้อมูลตามรูปที่ 2.8



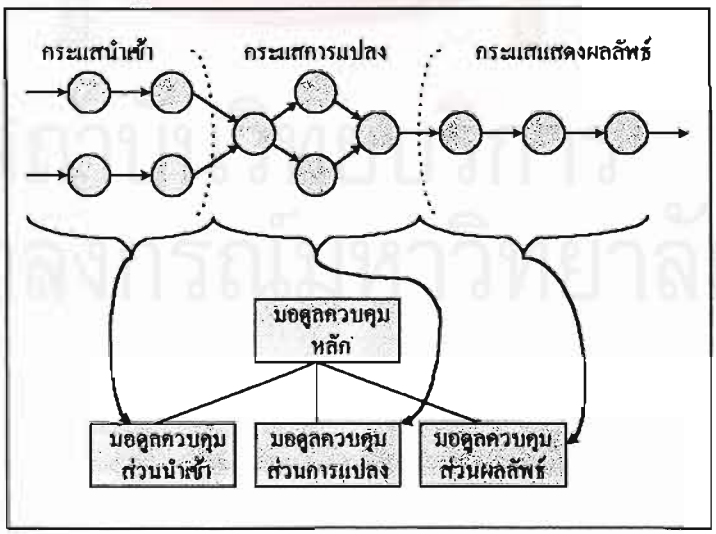
รูปที่ 2.8 ตัวอย่างแผนภาพกระแสข้อมูลแบบกระแสการแปลง

3) พิจารณาว่าแผนภาพมีคุณลักษณะแบบกระแสการแปลงหรือแบบกระแสทรานแซกชั่น ซึ่งโดยปกติแล้วการไหลของสารสนเทศในระบบมักจะมีลักษณะแบบการแปลง แต่ถ้าหากระบบงานที่พิจารณามีการไหลที่เป็นคุณลักษณะแบบกระแสทรานแซกชัน จะต้องใช้วิธีการแปลงแผนภาพกระแสข้อมูลด้วยวิธีที่ต่างไปจากนี้ คือ วิธีวิเคราะห์แบบทรานแซกชัน ดังนั้น ในขั้นตอนนี้ผู้ออกแบบจะต้องพิจารณาลักษณะโดยรวมของแผนภาพกระแสข้อมูลว่าเป็นแบบใด รวมถึงสามารถระบุส่วนย่อยต่างๆ ที่มีลักษณะกระแสการแปลงหรือกระแส ทรานแซกชันได้

4) ระบุศูนย์กลางการแปลงโดยกำหนดขอบเขตของกระแสนำเข้าและกระแสไหลออก ซึ่งขอบเขตของกระแสนำเข้าและกระแสไหลออกนี้จะมีลักษณะเปิดกว้าง กล่าวคือ นักออกแบบแต่ละคนอาจเลือกตำแหน่งขอบเขตที่ต่างกันได้สำหรับแผนภาพกระแสข้อมูลเดียวกัน อย่างไรก็ตาม พบว่าตำแหน่งที่เลือกต่างกันนี้ จะมีผลกระทบเพียงเล็กน้อยกับโครงสร้างของโปรแกรมที่เป็นผลลัพธ์สุดท้าย

5) ทำการสร้างมอดูลปัจจัยระดับแรก (First-Level Factoring) ผังโครงสร้างของโปรแกรมจะแสดงถึงการกระจายการควบคุมการทำงานของโปรแกรมในลักษณะจากบนลงล่าง ซึ่งการสร้างปัจจัย (Factor) นี้เป็นการสร้างระดับต่างๆ ในโครงสร้าง โดยที่มอดูลในระดับบนเป็นส่วนตัดสินใจการทำงานของโปรแกรม มอดูลระดับล่างทำหน้าที่รับข้อมูลจากภายนอก ประมวลผล และแสดงผลลัพธ์แก่ภายนอก ขณะที่มอดูลระดับกลางจะทำหน้าที่ควบคุมการทำงานบางส่วนหรือทำงานประมวลผลบางอย่าง

สำหรับกระแสแบบการแปลงนี้ แผนภาพกระแสข้อมูลจะถูกจับคู่เป็นผังโครงสร้างของโปรแกรมแบบโครงร่างซึ่งประกอบด้วยมอดูลควบคุม 3 ส่วน คือ การนำเข้าข้อมูล การแปลงข้อมูล และการแสดงผลลัพธ์ข้อมูล ดังแสดงในรูปที่ 2.9 สำหรับแผนภาพที่มีความ



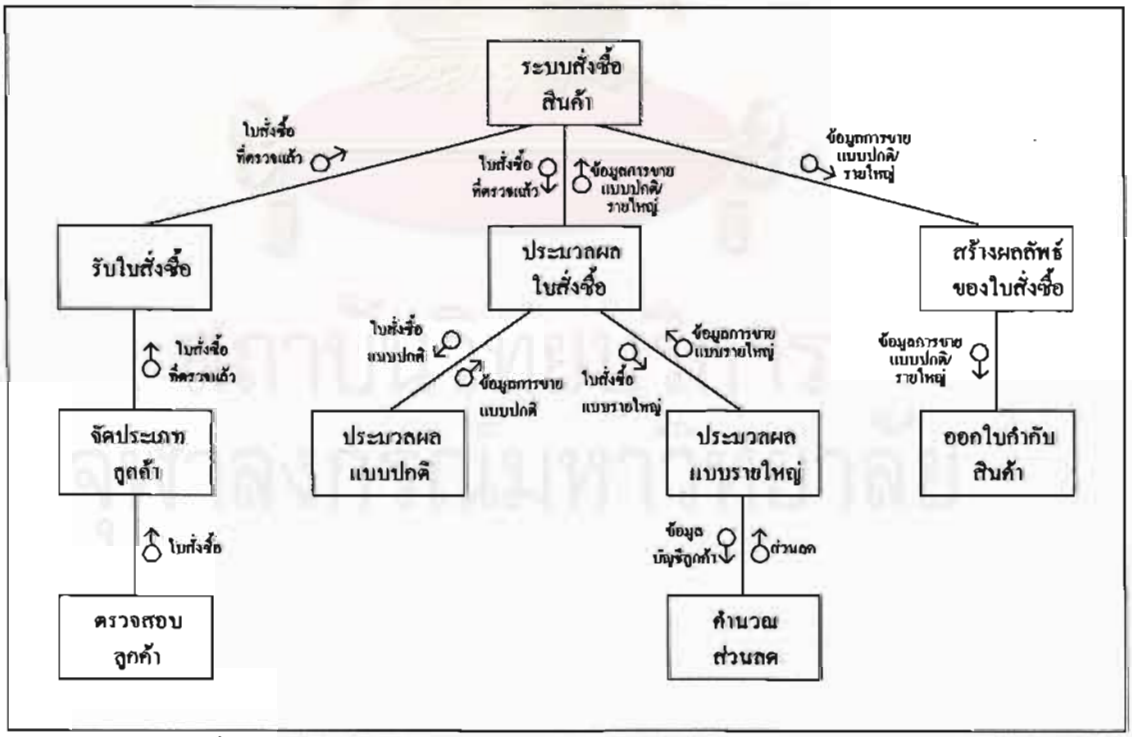
รูปที่ 2.9 การสร้างมอดูลปัจจัยระดับแรกตามวิธีวิเคราะห์แบบแปลง [1]

ซับซ้อนและมีขนาดใหญ่นั้น ระบบอาจจะประกอบด้วยมอดูลควบคุมของส่วนใดๆ มากกว่า 2 มอดูลได้ อย่างไรก็ตามจำนวนมอดูลในระดับที่หนึ่งนั้นควรให้มีจำนวนน้อยที่สุดเท่าที่เป็นไปได้ เพื่อให้ระบบมีความเกาะตัวและความสัมพันธ์ต่อกันระหว่างมอดูลในระดับที่ดี

6) ทำการสร้างมอดูลปัจจัยระดับที่สอง (Second-Level Factoring) เป็นการจับคู่แต่ละกระบวนการที่อยู่ในแผนภาพกระแสข้อมูลมายังผังโครงสร้างของโปรแกรม โดยเริ่มต้นจากศูนย์กลางการแปลงแล้วเคลื่อนออกไปยังเส้นทางนำเข้าและแสดงผลลัพธ์ข้อมูล โดยเป็นการจับคู่นำไปต่อเป็นระดับย่อยลงไปอย่างเป็นลำดับจากมอดูลควบคุมที่สัมพันธ์กัน ตัวอย่างของผลลัพธ์ที่ได้ตามขั้นตอนนี้แสดงได้ดังรูปที่ 2.10

โดยทั่วไปแล้ว การจับคู่จะเป็นลักษณะแบบหนึ่งต่อหนึ่ง แต่ในบางครั้งสามารถรวมกระบวนการสองหรือสามกระบวนการจับคู่เป็นมอดูลเดียวได้ หรืออาจแตกกระบวนการเดียวออกเป็นสองหรือสามมอดูลได้เช่นกัน โดยพิจารณาตามแต่คุณภาพของการออกแบบ

7) ปรับแต่งรายละเอียดของผังโครงสร้างของโปรแกรมที่ได้เพื่อเพิ่มคุณภาพของซอฟต์แวร์ ผังโครงสร้างของโปรแกรมแบบโครงร่างที่ได้จากขั้นตอนที่ (6) นี้สามารถนำมาปรับแต่งได้โดยใช้แนวคิดเกี่ยวกับความเป็นอิสระของมอดูล โดยการยุบรวมหรือแตกมอดูลออกเพื่อให้ได้มอดูลที่มีความเกาะตัวสูงและมีความสัมพันธ์ต่อกันระหว่างมอดูลในระดับต่ำ เพื่อให้ได้ผังโครงสร้างของโปรแกรมที่นำไปสร้างจริงได้ง่าย สามารถทดสอบและบำรุงรักษาได้ง่าย



รูปที่ 2.10 ตัวอย่างผังโครงสร้างของโปรแกรมตามวิธีวิเคราะห์แบบแปลง

2.3.3 การวิเคราะห์แบบทรานแซกชัน (Transaction Analysis) [1]

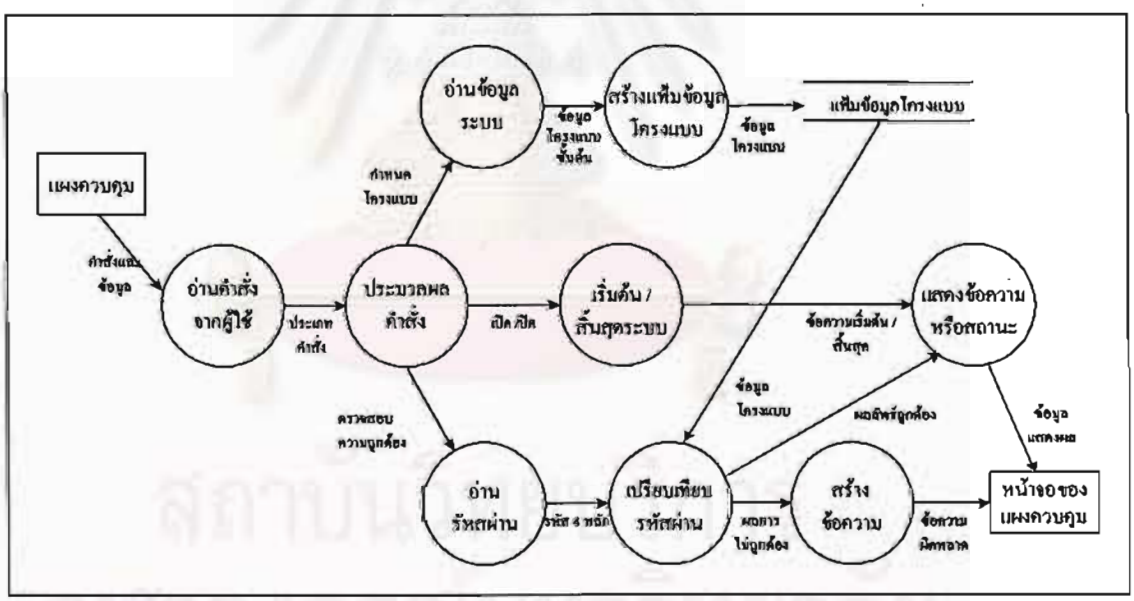
ในหลายๆ ระบบงาน ตัวข้อมูลเดี่ยวจะถูกใช้เพื่อเลือกเส้นทางการไหลของสารสนเทศหนึ่งเส้นทางหรือมากกว่านั้น ซึ่งจะเรียกตัวข้อมูลนี้ว่าทรานแซกชัน ซึ่งการแปลงแผนภาพกระแสข้อมูลที่มีกระแสทรานแซกชัน จะมีขั้นตอนต่างๆ ดังนี้

1) ตรวจทานตัวแบบบริบทของระบบงาน

2) ตรวจทานและปรับแต่งรายละเอียดของแผนภาพกระแสข้อมูล

3) พิจารณาว่าแผนภาพมีคุณลักษณะแบบกระแสการแปลงหรือแบบกระแสทรานแซกชัน ซึ่งขั้นตอนที่ 1 2 และ 3 นี้จะเหมือนกับขั้นตอนของวิธีวิเคราะห์แบบแปลง ซึ่งในที่นี้ จะใช้ตัวอย่างแผนภาพกระแสข้อมูลดังรูปที่ 2.11 ซึ่งเป็นแผนภาพกระแสข้อมูลแบบทรานแซกชัน

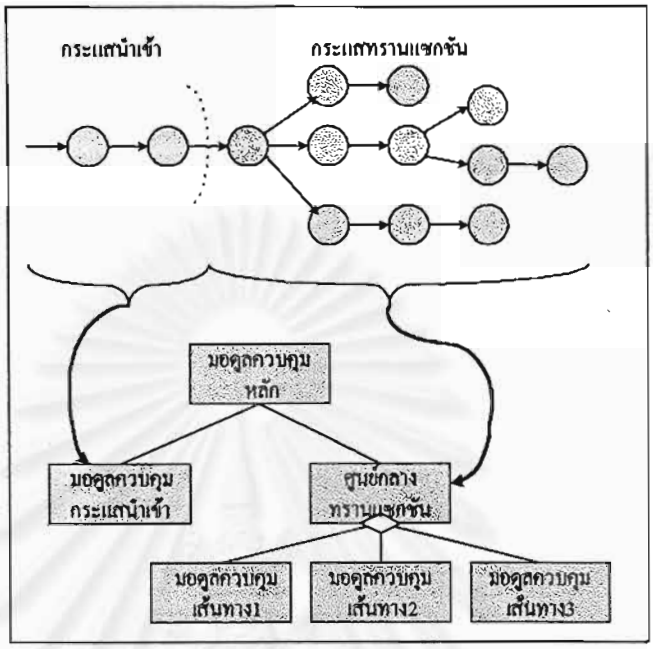
4) ระบุถึงศูนย์กลางทรานแซกชันและพิจารณาคุณลักษณะของแต่ละเส้นทางการกระทำ ศูนย์กลางทรานแซกชันจะเป็นจุดเริ่มต้นของเส้นทางการกระทำต่างๆ ที่ไหลออกจากตัวมัน จากนั้น ให้แบ่งส่วนสำหรับเส้นทางนำเข้าและเส้นทางการกระทำต่างๆ แล้วพิจารณาแต่ละเส้นทางการกระทำว่ามีคุณลักษณะแบบใด เพื่อทำการแบ่งส่วนย่อยๆ ตามคุณลักษณะของเส้นทางนั้นๆ ต่อไป



รูปที่ 2.11 ตัวอย่างแผนภาพกระแสข้อมูลแบบกระแสทรานแซกชัน [6]

5) จับคู่แผนภาพกระแสข้อมูลมาเป็นผังโครงสร้างของโปรแกรม ซึ่งกระแสทรานแซกชันจะถูกแปลงเป็นผังโครงสร้างของโปรแกรมซึ่งประกอบด้วยกิ่งนำเข้า (Incoming Branch) และ กิ่งจ่ายงาน (Dispatch Branch) ซึ่งวิธีการจับคู่สำหรับส่วนนำเข้าจะใช้วิธีการเดียวกับวิธีวิเคราะห์แบบแปลง กล่าวคือ เริ่มต้นจากศูนย์กลางทรานแซกชันให้ทำการจับคู่กระบวนการที่อยู่ในเส้นทางนำเข้ามาเป็นมอดูลในกิ่งนำเข้า สำหรับโครงสร้างของกิ่งจ่ายงานนั้น จะประกอบด้วย

มอดูลตัวเลือกจ่ายงาน (Dispatcher Module) ซึ่งควบคุมการทำงานของเส้นทางการกระทำต่างๆ โดยที่แต่ละเส้นทางการกระทำจะถูกจับคู่เป็นผังโครงสร้างตามคุณลักษณะการไหลของมัน กระบวนการนี้แสดงได้ดังตัวอย่างในรูปที่ 2.12



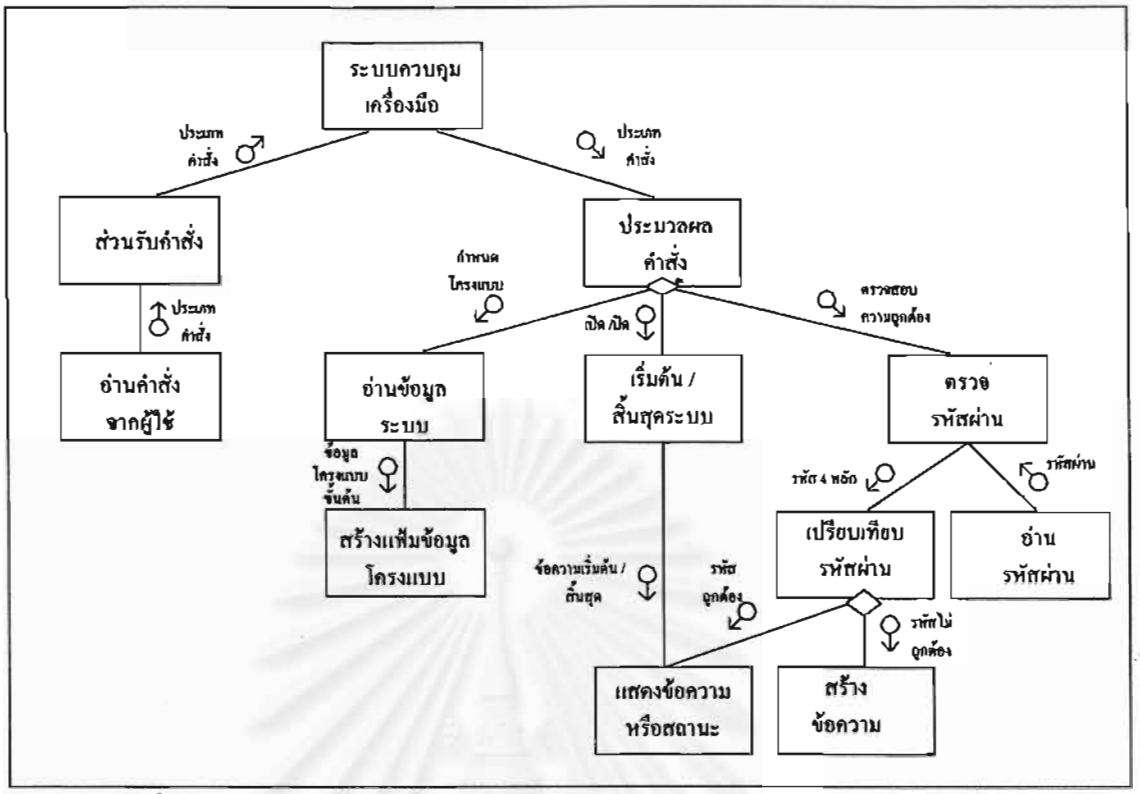
รูป 2.12 การสร้างมอดูลปัจจัยระดับแรกตามวิธีวิเคราะห์แบบทรานแซกชัน [1]

6) สร้างมอดูลปัจจัยและปรับแต่งรายละเอียดของโครงสร้างของส่วนทรานแซกชันและแต่ละเส้นทางการกระทำ ทำการจับคู่กระบวนการในแต่ละเส้นทางมาเป็นผังโครงสร้างโดยใช้วิธีวิเคราะห์แบบแปลงหรือแบบทรานแซกชันตามแต่คุณลักษณะการไหลของแต่ละเส้นทางนั้น ผลลัพธ์ที่ได้จากขั้นตอนนี้ คือ ผังโครงสร้างของโปรแกรมที่เป็นโครงร่าง ดังแสดงตัวอย่างในรูปที่ 2.13

7) ปรับแต่งรายละเอียดของผังโครงสร้างของโปรแกรมที่ได้เพื่อเพิ่มคุณภาพของซอฟต์แวร์ โดยใช้วิธีเดียวกับขั้นตอนในวิธีวิเคราะห์แบบแปลง

2.4 ภาษาในการออกแบบโปรแกรม (Program Design Languages) [1]

ภาษาในการออกแบบโปรแกรมหรือบางครั้งเรียกว่า ภาษาอังกฤษแบบโครงสร้าง (Structured English) หรือ รหัสเทียม (Pseudo code) เป็นการผสมผสานระหว่างคำศัพท์ที่ใช้ในภาษาพูด (เช่น ภาษาอังกฤษ) กับหลักภาษาที่ใช้ในการเขียนโปรแกรมแบบโครงสร้าง (Structured Programming Language) เพื่อใช้ในการออกแบบโปรแกรมซอฟต์แวร์ซึ่งมีลักษณะที่เข้าใจได้ง่าย ตัวอย่างของภาษาในการออกแบบโปรแกรมแสดงดังในรูปที่ 2.14 ภาษาในการออกแบบโปรแกรม



รูปที่ 2.13 ตัวอย่างผังโครงสร้างของโปรแกรมตามวิธีวิเคราะห์แบบทรานแซกชัน

มีลักษณะภาษาที่คล้ายกับภาษาในการเขียนโปรแกรมระดับสูง เช่น ภาษายุคที่สี่ (4 Generation Language) แต่ยังคงมีความยืดหยุ่นสูงกว่า กล่าวคือ ไม่มีข้อกำหนดในเรื่องของไวยากรณ์ของภาษา (Syntax) ที่เคร่งครัดมากอย่างภาษาในยุคที่สี่ เนื่องจากภาษาในการออกแบบโปรแกรมไม่ได้ถูกนำไปแปลภาษา (Compile) เพื่อสร้างรหัสกระทำการ (Executable code) เนื่องจากจุดประสงค์ของภาษาในการออกแบบนั้นใช้เพื่อเป็นการถ่ายทอดความคิดและความเข้าใจในการทำงานของโปรแกรมของผู้ออกแบบซอฟต์แวร์ (Software Designer) ให้แก่ผู้เขียนโปรแกรม (Programmer) โดยไม่ยึดติดในภาษาการเขียนโปรแกรมภาษาใดภาษาหนึ่งโดยเฉพาะ

โดยทั่วไป ภาษาในการออกแบบโปรแกรมจะมีคุณลักษณะดังนี้

1) มีการกำหนดรูปแบบการใช้ภาษาที่ชัดเจนของคำหลัก (Keywords) ในการกำหนดส่วนของโปรแกรม การนิยามตัวแปรหรือข้อมูล

2) มีความยืดหยุ่นในการนำภาษาที่ใช้พูดมาใช้อธิบายวิธีการประมวลผลของโปรแกรม

3) สามารถนิยามข้อมูลได้กว้างขวาง ทั้งโครงสร้างข้อมูลแบบง่าย เช่น ตัวแปรเชิงเดี่ยว (Scalar) แถวลำดับ (Array) และแบบซับซ้อน เช่น รายการโยง (Linked List)

4) สามารถนิยามมอดูลของโปรแกรมหรือโปรแกรมย่อย (Subprogram) และการเรียกใช้โปรแกรมย่อยได้

```
while (not exit code)
        while (not has card input)
                print message "Welcome – Please enter your card:"
        endwhile
        Account_number = Read_card
        call Get_Account_Detail(PIN, Account_balance, Cash_available)
        if PIN is validate then
                print operation choices message
                case choice is
                        "Cash Only":      call Dispense_cash
                        "Print Balance":  call Print_balance
                        "Statement":      call Order_statement
                        "Check book":     call Order_checkbook
                endcase
                update account information
                eject card
                print message "Please take your card or press Continue"
        else
                retain card
        endif
endwhile
```

รูปที่ 2.14 ตัวอย่างภาษาในการออกแบบโปรแกรม

ภาษาในการออกแบบโปรแกรมขั้นพื้นฐาน ควรจะประกอบด้วยคำสั่งในการทำงานดังนี้

1) การนิยามโปรแกรมย่อย เช่น MODULE และ PROCEDURE เป็นต้น
2) การนิยามข้อมูล เช่น TYPE เป็นต้น
3) คำสั่งที่ใช้กับโครงสร้างแบบบล็อก (Block Structuring) เช่น BEGIN-END เป็นต้น
4) การกำหนดเงื่อนไข เช่น IF-THEN-ELSE และ CASE-SELECT เป็นต้น
5) การทำงานแบบวนรอบ เช่น DO-WHILE FOR-DO และ REPEAT-UNTIL เป็นต้น
6) คำสั่งในการใช้อุปกรณ์นำเข้าและแสดงผลลัพธ์ เช่น READ และ WRITE เป็นต้น

2.5 ความสัมพันธ์ต่อกันระหว่างมอดูล (Module Coupling) [1],[2]

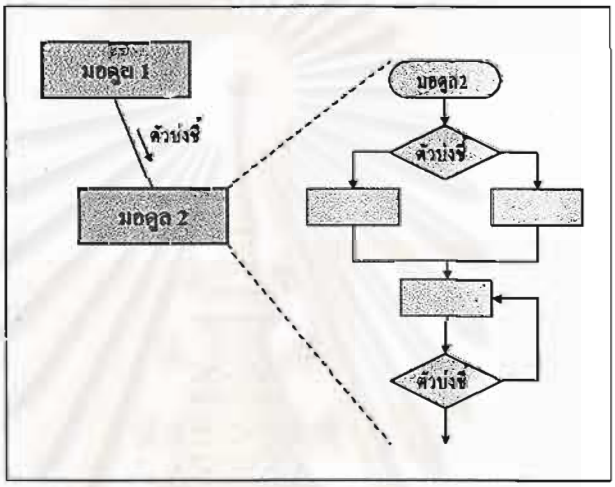
ความสัมพันธ์ต่อกันระหว่างมอดูลเป็นตัววัดถึงการติดต่อระหว่างกันของมอดูลในโครงสร้างของโปรแกรม ซึ่งบอกถึงความซับซ้อนในการติดต่อกันของมอดูลว่ามีการส่งผ่านข้อมูลอะไรถึงกันและลักษณะการใช้งานข้อมูล โดยความสัมพันธ์ต่อกันระหว่างมอดูลสามารถแบ่งออกเป็น 7 ประเภท ดังนี้

1) แบบไม่สัมพันธ์กันโดยตรง (No Direct Coupling) เกิดขึ้นเมื่อมอดูลสองมอดูลไม่มีการเรียกทำงานระหว่างกัน เพียงแต่เป็นมอดูลที่อยู่ในผังโครงสร้างเดียวกัน ซึ่งรูปแบบนี้เป็นความสัมพันธ์ต่อกันในระดับต่ำที่สุด

2) แบบข้อมูล (Data Coupling) เป็นความสัมพันธ์ที่สองมอดูลมีการส่งพารามิเตอร์ที่เป็นข้อมูลชนิดตัวแปรแบบง่าย (Simple data) ระหว่างกัน

3) แบบประทับตรา (Stamp Coupling) เป็นความสัมพันธ์ที่สองมอดูลมีการส่งพารามิเตอร์ที่เป็นข้อมูลชนิดข้อมูลโครงสร้าง (Data Structure) ระหว่างกัน

4) แบบควบคุม (Control Coupling) เป็นความสัมพันธ์ที่สองมอดูลมีการส่งพารามิเตอร์ถึงกันเพื่อใช้ในการควบคุมการทำงาน ซึ่งโดยทั่วไปแล้วจะเป็นพารามิเตอร์แบบตัวบ่งชี้ (Flag) ที่ใช้พิจารณาค่าเพื่อตัดสินใจ ตัวอย่างของความสัมพันธ์แบบนี้แสดงดังในรูปที่ 2.15



รูปที่ 2.15 ตัวอย่างความสัมพันธ์ต่อกันแบบควบคุม [1]

5) แบบภายนอก (External Coupling) เป็นความสัมพันธ์ที่สองมอดูลมีการใช้สภาพแวดล้อมภายนอกโปรแกรมร่วมกัน เช่น อุปกรณ์รับเข้าและแสดงข้อมูล (I/O Device) หรือโพรโทคอลสื่อสาร (Communication Protocol) เป็นต้น

6) แบบร่วมกัน (Common Coupling) เป็นความสัมพันธ์ที่สองมอดูลมีการใช้งานพื้นที่ข้อมูลแบบครอบคลุม (Global Data Area) เช่น ตัวแปรร่วมกัน หรือ แฟ้มข้อมูล เป็นต้น

7) แบบเนื้อหา (Content Coupling) เป็นความสัมพันธ์ที่มอดูลหนึ่งไปใช้ข้อมูลที่อยู่ในขอบเขตของอีกมอดูลหนึ่ง หรือการที่มอดูลหนึ่งกระโดดเข้าไปทำงานที่ตำแหน่งตรงกลางของอีกมอดูล ซึ่งความสัมพันธ์แบบนี้เป็นความสัมพันธ์ต่อกันในระดับที่สูงที่สุดและควรจะหลีกเลี่ยงไม่ให้เกิดขึ้น

ออฟฟัต [2] ได้เสนอประเภทของความสัมพันธ์ต่อกันระหว่างมอดูลเพิ่มเติมจากที่กล่าวมาข้างต้นเพื่อให้สอดคล้องกับภาษาการเขียนโปรแกรมที่มีคุณสมบัติเพิ่มขึ้นในปัจจุบัน โดยพิจารณาจากรูปแบบการใช้งานพารามิเตอร์ของมอดูลซึ่งจัดแบ่งได้เป็น 3 รูปแบบ คือ

1) การใช้คำนวณค่า (Computation-uses หรือ C-uses) เกิดขึ้นเมื่อพารามิเตอร์ถูกใช้ในคำสั่งกำหนดค่า (Assignment statement) หรือคำสั่งแสดงผลลัพธ์ (Output statement)

2) การใช้เป็นเพรดิเคต (Predicate-uses หรือ P-uses) เกิดขึ้นเมื่อพารามิเตอร์ถูกใช้ในคำสั่งเพรดิเคต (Predicate statement) ซึ่งหมายถึงคำสั่งที่เป็นการทดสอบเงื่อนไข

3) การใช้งานทางอ้อม (Indirect-uses หรือ I-uses) เกิดขึ้นเมื่อพารามิเตอร์ที่เป็น C-use มีผลต่อคำสั่งเพรดิเคตในมอดูล

จากรูปแบบการใช้งานพารามิเตอร์ดังกล่าว ออฟฟัตได้จัดประเภทความสัมพันธ์ต่อกันระหว่างมอดูลออกเป็น 12 ประเภท ดังนี้

1) แบบอิสระต่อกัน (Independent Coupling) เกิดขึ้นเมื่อมอดูล A ไม่มีการเรียกมอดูล B และมอดูล B ไม่มีการเรียกมอดูล A รวมถึงทั้งสองมอดูลไม่มีการใช้ตัวแปรใดๆ ร่วมกันและไม่มีการใช้ข้อมูลผ่านสื่อภายนอกร่วมกัน เช่น แฟ้มข้อมูล เป็นต้น ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็นศูนย์

2) แบบเรียกกัน (Call Coupling) เกิดขึ้นเมื่อมอดูล A เรียกมอดูล B หรือมอดูล B เรียกมอดูล A โดยไม่มีการส่งพารามิเตอร์ให้กัน รวมถึงไม่มีการใช้ตัวแปรหรือข้อมูลร่วมกันทั้งแบบภายในและภายนอก ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 1

3) แบบข้อมูลตัวแปรเชิงเดี่ยว (Scalar Data Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรเชิงเดี่ยว (Scalar Variable) ให้กับมอดูล B ซึ่งถูกใช้งานแบบคำนวณค่า ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 2

4) แบบข้อมูลตัวแปรเชิงกลุ่ม (Stamp Data Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรโครงสร้าง (Record หรือ Data Structure) ให้กับมอดูล B ซึ่งถูกใช้งานแบบคำนวณค่า ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 3

5) แบบควบคุมตัวแปรเชิงเดี่ยว (Scalar Control Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรเชิงเดี่ยวให้กับมอดูล B ซึ่งถูกใช้งานแบบเพรดิเคต ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 4

6) แบบควบคุมตัวแปรเชิงกลุ่ม (Stamp Control Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรแบบระเบียนหรือแบบข้อมูลโครงสร้างให้กับมอดูล B ซึ่งถูกใช้งานแบบเพรดิเคต ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 5

7) แบบข้อมูลกึ่งควบคุมตัวแปรเชิงเดี่ยว (Scalar Data/Control Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรเชิงเดี่ยวให้กับมอดูล B ซึ่งถูกใช้งานแบบทางอ้อม โดยไม่มีการใช้งานแบบเพรดิเคต ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 6

8) แบบข้อมูลกึ่งควบคุมตัวแปรเชิงกลุ่ม (Stamp Data/Control Coupling) เกิดขึ้นเมื่อมอดูล A ส่งตัวแปรโครงสร้างให้กับมอดูล B ซึ่งถูกใช้งานแบบทางอ้อม โดยไม่มีการใช้งานแบบเพรดิเคต ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 7

9) แบบภายนอก (External Coupling) เกิดขึ้นเมื่อมอดูล A และ B มีการติดต่อกันผ่านสื่อภายนอก เช่น แฟ้มข้อมูล เป็นต้น ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 8

10) แบบไม่ใช้เฉพาะที่ (Non-Local Coupling) เกิดขึ้นเมื่อมอดูล A และ B มีการอ้างอิงถึงตัวแปรร่วมกันซึ่งไม่ใช่ตัวแปรแบบเฉพาะที่ แต่เป็นตัวแปรที่ใช้งานได้ในหลายมอดูลหรือบางส่วนของระบบงาน ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 9

11) แบบครอบคลุม (Global Coupling) เกิดขึ้นเมื่อมอดูล A และ B มีการอ้างอิงถึงตัวแปรร่วมกัน ซึ่งเป็นตัวแปรแบบครอบคลุม คือเป็นตัวแปรที่ใช้งานได้ในทั้งระบบงาน ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 10

12) แบบส่งผ่าน (Tramp Coupling) เกิดขึ้นเมื่อมอดูล A ส่งพารามิเตอร์ให้กับมอดูล B ซึ่งมอดูล B ไม่ได้มีการใช้งานหรือแก้ไขค่าของพารามิเตอร์ แต่เพียงส่งผ่านพารามิเตอร์ต่อไปยังมอดูลอื่น ความสัมพันธ์แบบนี้จะมีค่าระดับความสัมพันธ์เป็น 11

วิธีการวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของออฟฟัตประกอบด้วย 3 ขั้นตอนหลักดังนี้

1) สร้างแฟ้มข้อมูลสรุป ขั้นตอนนี้เริ่มต้นจากการกราดตรวจ (Scan) คำอธิบายการทำงานของมอดูล (Process Description) ทีละสองมอดูล เพื่อสร้างรายการการนิยามตัวแปร รูปแบบการใช้งานตัวแปร ตำแหน่งที่เรียกใช้มอดูล พารามิเตอร์ที่ใช้ติดต่อกันและรูปแบบการใช้งาน จากนั้นจะทำการวิเคราะห์ข้อมูลต่างๆ ที่ได้มาเพื่อตัดสินว่ามอดูลมีการใช้งานพารามิเตอร์เป็นแบบใด (C-use P-use หรือ I-use)

2) วัดระดับของความสัมพันธ์ต่อกันระหว่างมอดูล โดยการตัดสินว่ามอดูล A และ B ใดๆ มีความสัมพันธ์ต่อกันระหว่างมอดูลเป็นประเภทใดตามข้อมูลที่รวบรวมมาจากขั้นตอนที่หนึ่ง โดยเริ่มต้นพิจารณาจากระดับที่สูงที่สุด (แบบส่งผ่าน) ก่อนไล่ลงสู่ระดับต่ำที่สุด (แบบอิสระต่อกัน) ตามลำดับ และจะหยุดพิจารณาเมื่อสามารถตัดสินเลือกประเภทของความสัมพันธ์ได้ แล้วทำการนับจำนวนของความสัมพันธ์ตามประเภทที่ได้เลือกไว้นั้น

3) คำนวณค่าวัดความสัมพันธ์ต่อกันระหว่างมอดูลของ A กับ B โดยใช้สูตร

$$M_{(A,B)} = i + \frac{n}{n+1} - \frac{1}{2} = i + \frac{n-1}{2(n+1)}$$

เมื่อ $M_{(A,B)}$ คือ ค่าวัดความสัมพันธ์ต่อกันระหว่างมอดูลของ A กับ B

i คือ ค่าระดับของความสัมพันธ์ระหว่างมอดูล (มีค่าระหว่าง 0 ถึง 11)

n คือ จำนวนของความสัมพันธ์ตามระดับที่เลือก

2.6 ตัวแปลภาษา (Compiler) [3]

การทำงานของตัวแปลภาษา จะประกอบด้วย 2 ส่วนหลักๆ คือ ส่วนวิเคราะห์ศัพท์ (Lexical Analyzer) และส่วนวิเคราะห์ภาษา (Parser) โดยส่วนวิเคราะห์ศัพท์จะทำหน้าที่ในการอ่านรหัสต้นฉบับ (Source Code) ของโปรแกรมเข้ามาและทำการแปลงให้เป็นลำดับของหน่วยของคำดั้งเดิมที่เรียกว่า โทเค็น (Token) ตัวอย่างของโทเค็น เช่น คำหลัก (Keywords) ตัวระบุ (Identifiers) ค่าคงที่ (Constants) และตัวดำเนินการ (Operators) เป็นต้น นอกจากนี้ ส่วนวิเคราะห์ศัพท์จะทำหน้าที่อย่างอื่นด้วย เช่น การกำจัดช่องว่างระหว่างคำที่ซ้ำๆ กัน การลงเหมายเหตุ สำหรับส่วนวิเคราะห์ภาษาจะทำหน้าที่ในการวิเคราะห์ไวยากรณ์เพื่อตรวจสอบว่ารหัสต้นฉบับที่นำเข้านั้นเขียนขึ้นตามกฎไวยากรณ์ที่กำหนดหรือไม่ โดยข้อมูลนำเข้าของส่วนวิเคราะห์ภาษาจะเป็นลำดับรายการของโทเค็นที่ได้จากส่วนวิเคราะห์ศัพท์

ในการกำหนดไวยากรณ์ของภาษาในการเขียนโปรแกรมนั้น รูปแบบหนึ่งที่นิยมใช้กัน คือ ไวยากรณ์ไม่พึ่งบริบท (Context-Free Grammar) หรือบางครั้งเรียกว่า บีเอ็นเอฟ (Backus-Naur Form – BNF) ซึ่งข้อดีของวิธีการนี้ คือ สามารถกำหนดกฎของไวยากรณ์ที่ชัดเจนโดยที่ยังคงมีความเข้าได้ง่ายและยืดหยุ่น ตัวอย่างของบีเอ็นเอฟแสดงดังในรูปที่ 2.16 โดยองค์ประกอบในการกำหนดกฎไวยากรณ์จะประกอบด้วย 4 คุณลักษณะ คือ

1) เทอร์มินัล (Terminal) คือ คำเฉพาะที่ใช้ในภาษา ซึ่งได้แก่คำหลักและสัญลักษณ์พิเศษ ตามตัวอย่างในรูปที่ 2.16 จะได้แก่ begin end และ ;

```
statement      ::=   begin statement-list end
statement-list ::=   statement |
                     statement ; statement-list
```

รูปที่ 2.16 ตัวอย่างของบีเอ็นเอฟ

2) นอนเทอร์มินัล (Non-Terminal) คือ คำที่ใช้แสดงถึงกลุ่มของไวยากรณ์ ตามตัวอย่างในรูปที่ 2.16 จะได้แก่ statement และ statement-list

3) สัญลักษณ์เริ่มต้น (Start Symbol) คือ คำที่ใช้แสดงจุดเริ่มต้นของภาษา โดยจะเลือกนอนเทอร์มินัลขึ้นมา 1 คำใช้เป็นสัญลักษณ์เริ่มต้นนี้ ตามตัวอย่างในรูปที่ 2.16 จะได้แก่ statement

4) โปรดักชั่น (Production) เป็นการขยายความกฎของไวยากรณ์โดยการเขียนรูปแบบต่างๆ ที่เป็นไปได้ของไวยากรณ์ออกมา ตามตัวอย่างในรูปที่ 2.16 จะสามารถเขียนออกมาเป็นโปรดักชั่นได้ทั้งหมด 3 โปรดักชั่น ได้แก่

โปรดักชั่นที่ 1: statement ::= begin statement-list end

โปรดักชั่นที่ 2: statement-list ::= statement

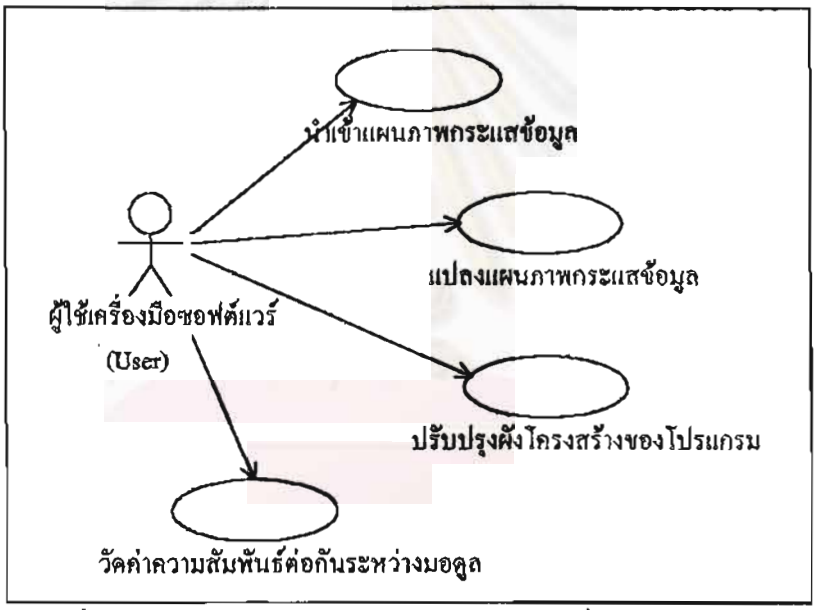
โปรดักชั่นที่ 3: statement-list ::= statement; statement-list

บทที่ 3

การออกแบบเครื่องมือซอฟต์แวร์

3.1 การออกแบบการทำงานของเครื่องมือซอฟต์แวร์

สำหรับงานวิจัยนี้เป็นการพัฒนาเครื่องมือซอฟต์แวร์เพื่อทำการแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรม โดยใช้วิธีการออกแบบและพัฒนาด้วยวิธีการเชิงวัตถุ (Object-Oriented Design and Development) สำหรับการออกแบบการทำงานของเครื่องมือซอฟต์แวร์นี้สามารถแสดงได้ด้วยแผนภาพยูสเคส (Use Case Diagram) ดังแสดงรูปที่ 3.1 ซึ่งประกอบด้วย 4 ยูสเคสดังนี้



รูปที่ 3.1 แผนภาพยูสเคสแสดงการทำงานของเครื่องมือซอฟต์แวร์

1) ยูสเคสนำเข้าแผนภาพกระแสข้อมูล เป็นการกำหนดแผนภาพกระแสข้อมูลที่ต้องการทำการแปลง โดยผู้ใช้สามารถนำแฟ้มข้อมูลของแผนภาพกระแสข้อมูลที่สร้างด้วยโปรแกรม Power Designer 6.1 เข้ามา หรือผู้ใช้อาจวาดแผนภาพกระแสข้อมูลขึ้นใหม่โดยใช้เครื่องมือซอฟต์แวร์ที่พัฒนานี้ อนึ่ง ในงานวิจัยนี้ ผู้วิจัยเลือกใช้แฟ้มข้อมูลจากโปรแกรม Power Designer 6.1 เนื่องจากโปรแกรมดังกล่าวจัดเก็บแผนภาพกระแสข้อมูลเป็นแฟ้มข้อมูลแบบข้อความ (Text File) และมีคำอธิบายรูปแบบการจัดเก็บบอกไว้ ทำให้สามารถนำแฟ้มข้อมูลเข้ามาทำงานได้อย่างสะดวก

2) ยูสเคสแปลงแผนภาพกระแสข้อมูล เป็นการแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรม โดยผู้ใช้สามารถเลือกให้เครื่องมือซอฟต์แวร์ทำการแปลงแบบอัตโนมัติ หรือผู้ใช้อาจกำหนดเงื่อนไขในการแปลงก่อนที่จะสั่งให้เครื่องมือซอฟต์แวร์ทำการแปลงได้ เช่น กำหนดส่วนของศูนย์กลางการแปลง หรือศูนย์กลางทรานแซกชัน เป็นต้น ซึ่งจะเรียกว่าการแปลงแบบกึ่งอัตโนมัติ เครื่องมือซอฟต์แวร์จะแสดงรูปแบบต่างๆ ของแผนภาพกระแสข้อมูลที่ได้จากการแปลงมาให้ผู้ใช้ทำการเลือก จากนั้น ผู้ใช้ทำการจัดเก็บผังโครงสร้างของโปรแกรมที่ต้องการลงเป็นแฟ้มข้อมูล

3) ยูสเคสปรับปรุงผังโครงสร้างของโปรแกรม เป็นการปรับปรุงคุณภาพของผังโครงสร้างของโปรแกรมที่ได้จากการแปลงโดยผู้ใช้สามารถปรับปรุงผ่านเครื่องมือซอฟต์แวร์ เช่น การจัดตำแหน่งของมอดูล การยุบ หรือการรวมมอดูล เป็นต้น นอกจากนี้ ผู้ใช้สามารถเขียนคำอธิบายการทำงานของมอดูลในรูปแบบภาษาในการออกแบบโปรแกรมให้กับแต่ละมอดูลได้

4) ยูสเคสวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล เป็นการตรวจสอบคุณภาพของผังโครงสร้างของโปรแกรมจากคำอธิบายการทำงานของมอดูลที่ผู้ใช้กำหนดไว้

3.2 การออกแบบมอดูลของเครื่องมือซอฟต์แวร์

3.2.1 การออกแบบคลาส

งานวิจัยนี้ใช้รูปแบบการพัฒนาเครื่องมือซอฟต์แวร์ด้วยวิธีการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming – OOP) ซึ่งผู้วิจัยได้ออกแบบวัตถุต่างๆ ดังแสดงในแผนภาพคลาสของวัตถุ (Class Diagram) ในรูปที่ 3.2 โดยประกอบด้วยคลาสต่างๆ 14 คลาส ได้แก่

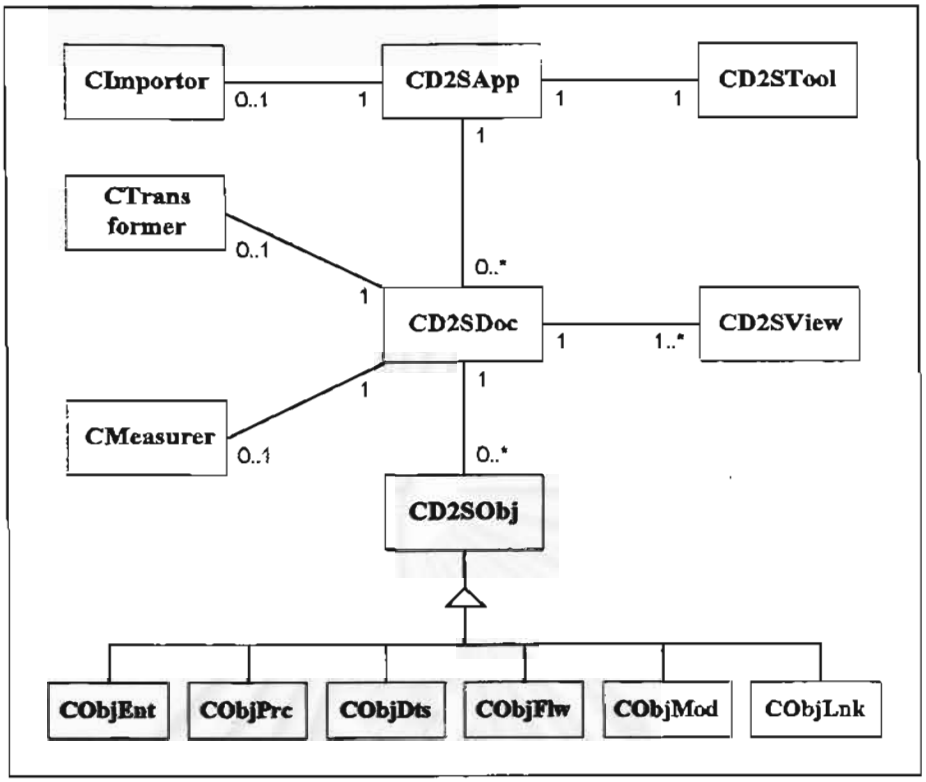
1) คลาส CD2SApp เป็นคลาสที่แสดงถึงตัวเครื่องมือซอฟต์แวร์

2) คลาส CD2SDoc เป็นคลาสที่ใช้เก็บข้อมูลของแผนภาพที่วาด (ทั้งแผนภาพกระแสข้อมูลและผังโครงสร้างของโปรแกรม) โดยในคลาส CD2SDoc นั้น จะบรรจุโครงสร้างข้อมูลแบบรายการ (List Data Structure) ของคลาส CD2SObj ซึ่งใช้แสดงถึงองค์ประกอบต่างๆ ของแผนภาพ

3) คลาส CD2SView เป็นคลาสที่ทำหน้าที่ในการแสดงผลข้อมูลให้กับผู้ใช้

4) คลาส CD2STool เป็นคลาสที่ทำหน้าที่ติดต่อกับแถบเครื่องมือในการวาดแผนภาพเพื่อรับคำสั่งจากผู้ใช้

5) คลาส CD2SObj เป็นคลาสที่แสดงถึงองค์ประกอบของแผนภาพ ซึ่งคลาสนี้จะเก็บคุณลักษณะ (Attributes) และวิธีการทำงาน (Methods) ที่ใช้ร่วมกันสำหรับองค์ประกอบต่างๆ ของแผนภาพ และเป็นคลาสแม่ (Super Class) ให้กับคลาส CObjEnt, CObjPrc, CObjDts, CObjFlw, CObjMod และ CobjLnk



รูปที่ 3.2 แผนภาพคลาสของวัตถุของการออกแบบเครื่องมือ

6) คลาส CObjEnt เป็นคลาสที่แสดงถึงองค์ประกอบแบบเอนทิตีภายนอก

7) คลาส CObjPrc เป็นคลาสที่แสดงถึงองค์ประกอบแบบกระบวนการทำงาน

8) คลาส CObjDts เป็นคลาสที่แสดงถึงองค์ประกอบแบบหน่วยจัดเก็บข้อมูล

9) คลาส CObjFlw เป็นคลาสที่แสดงถึงองค์ประกอบแบบการไหลของข้อมูล

10) คลาส CObjMod เป็นคลาสที่แสดงถึงองค์ประกอบแบบมอดูล

11) คลาส CObjLnk เป็นคลาสที่แสดงถึงองค์ประกอบแบบการเชื่อมโยงมอดูล

12) คลาส CImportor เป็นคลาสที่ทำหน้าที่นำเข้าแฟ้มข้อมูลของแผนภาพกระแสข้อมูลที่ได้จากโปรแกรม Power Designer 6.1

13) คลาส CTransformer เป็นคลาสที่ทำหน้าที่ประมวลผลในการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม

14) คลาส CMeasurer เป็นคลาสที่ทำหน้าที่วัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของรูปผังโครงสร้างของโปรแกรม

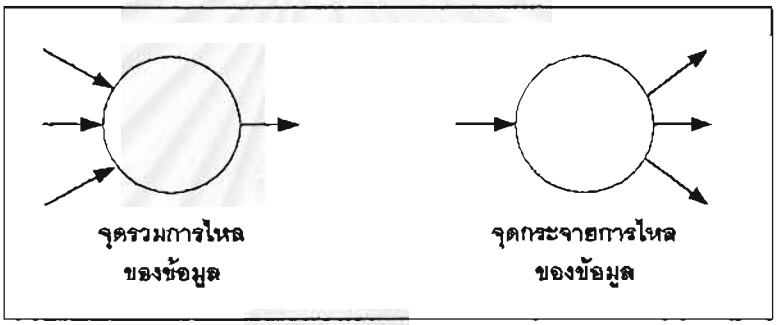
สำหรับคุณลักษณะ (Attributes) และวิธีทำงาน (Methods) ของคลาสต่างๆ นั้นแสดงรายละเอียดไว้ในภาพผนวก ก.

3.2.2 วิธีการพิจารณาคุณลักษณะของแผนภาพกระแสข้อมูล

วิธีการในการพิจารณาคุณลักษณะของแผนภาพกระแสข้อมูลประกอบด้วยขั้นตอนต่างๆ ดังนี้

1) นับจำนวนของกระบวนการทำงานที่มีคุณลักษณะเป็นจุดรวมการไหลของข้อมูล (Central-Point) ซึ่งจุดรวมการไหลของข้อมูล หมายถึงกระบวนการทำงานที่มีจำนวนเส้นการไหลของข้อมูลเข้ามายังกระบวนการทำงานมากกว่า 1 เส้น ดังภาพตัวอย่างที่แสดงในรูปที่ 3.3

2) นับจำนวนของกระบวนการทำงานที่มีคุณลักษณะเป็นจุดกระจายการไหลของข้อมูล (Distributed-Point) ซึ่งจุดกระจายการไหลของข้อมูลหมายถึงกระบวนการทำงานที่มีจำนวนเส้นการไหลของข้อมูลออกจากกระบวนการทำงานมากกว่า 1 เส้น ดังภาพตัวอย่างที่แสดงในรูปที่ 3.3



รูปที่ 3.3 ตัวอย่างของจุดรวมและจุดกระจายการไหลของข้อมูล

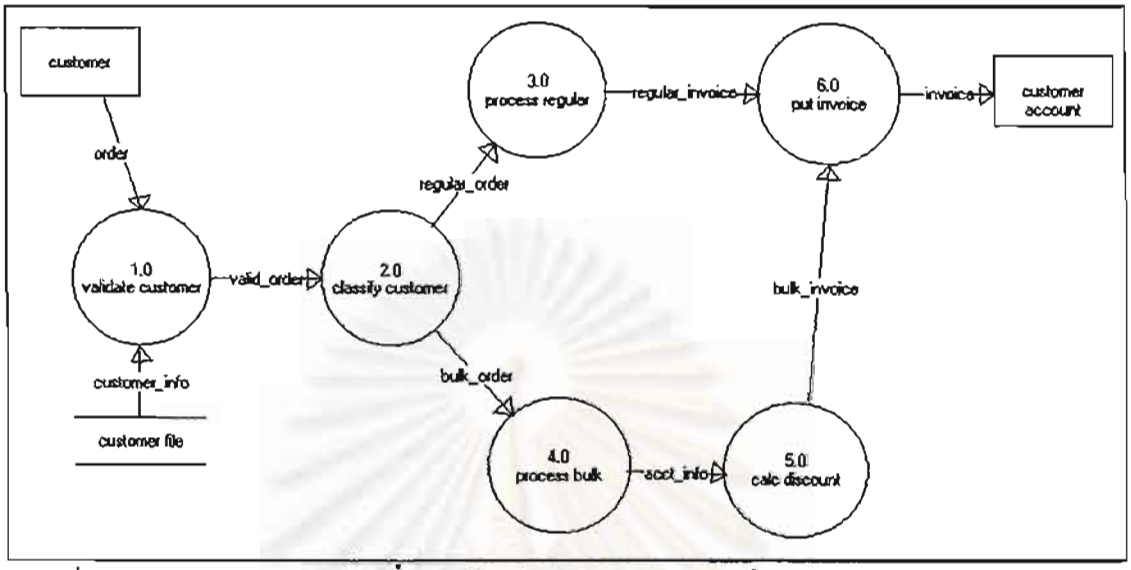
3) ในกรณีที่แผนภาพกระแสข้อมูลไม่มีจุดรวมการไหลของข้อมูล และมีจำนวนจุดกระจายการไหลของข้อมูลอย่างน้อย 1 จุด จะถือว่าแผนภาพกระแสข้อมูลมีคุณลักษณะการไหลของข้อมูลแบบทรานแซกชัน ซึ่งจะใช้วิธีการวิเคราะห์แบบทรานแซกชันในการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม ส่วนกรณีอื่นๆ จะถือว่าแผนภาพกระแสข้อมูลมีคุณลักษณะการไหลของข้อมูลแบบการแปลง ซึ่งจะใช้วิธีการวิเคราะห์แบบการแปลงในการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม

3.2.3 วิธีการแปลงแผนภาพกระแสข้อมูลตามวิธีการวิเคราะห์แบบแปลง

วิธีการในการแผนภาพกระแสข้อมูลตามวิธีการวิเคราะห์แบบแปลงประกอบด้วยขั้นตอนต่างๆ ดังนี้

1) หาเส้นทาง (Path) ต่างๆ ที่อยู่ในแผนภาพกระแสข้อมูล คำว่า "เส้นทาง" ในที่นี้ หมายถึงเส้นทางการไหลของข้อมูลซึ่งประกอบด้วยกระบวนการทำงานต่างๆ ในรูปที่ 3.4 เป็น

ตัวอย่างแผนภาพกระแสข้อมูลที่จะใช้ประกอบการอธิบายวิธีการของวิเคราะห์แบบแปลง ซึ่งจากแผนภาพดังกล่าวจะสามารถหาเส้นทางต่างๆ ได้ดังในตารางที่ 3.1



รูปที่ 3.4 ตัวอย่างแผนภาพกระแสข้อมูล

ตารางที่ 3.1 ตัวอย่างเส้นทางของแผนภาพกระแสข้อมูล

เส้นทางที่	กระบวนการทำงานบนเส้นทาง
1	1.0 – 2.0 – 4.0 – 5.0 – 6.0
2	1.0 – 2.0 – 3.0 – 6.0

2) สำหรับแต่ละเส้นทาง จะทำการหาคู่ของจุดแบ่งที่แบ่งเส้นทางออกเป็น 3 ส่วน ซึ่งจะใช้คู่ของจุดแบ่งนี้ในการทดสอบหาศูนย์กลางการแปลง ซึ่งตารางที่ 3.2 แสดงคู่ของจุดแบ่งต่างๆ ที่ได้ทั้งหมดสำหรับเส้นทางในตารางที่ 3.1

3) ทำการทดสอบหาศูนย์กลางการแปลงตามขั้นตอนต่อไปนี้

(1) เลือกคู่ของจุดแบ่งหนึ่งคู่จากแต่ละเส้นทางออกมาเป็นชุดทดสอบ โดยเริ่มจากการเลือกคู่ของจุดแบ่งลำดับที่ 1 จากเส้นทางที่ 1 และคู่ของจุดแบ่งลำดับที่ 1 จากเส้นทางที่ 2 มาใช้เป็นชุดทดสอบแรก

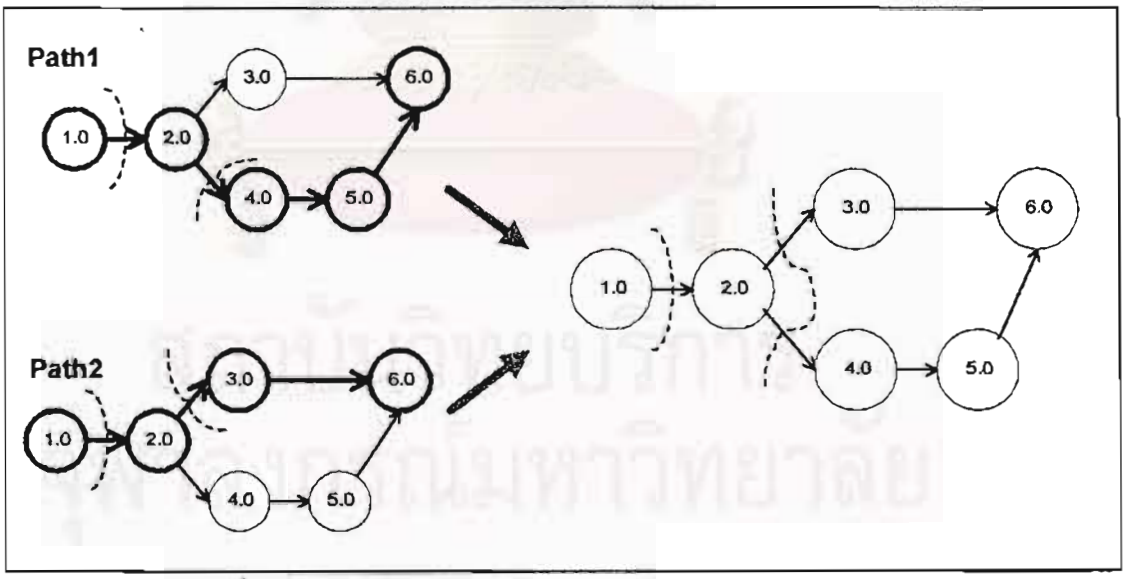
(2) สร้างศูนย์กลางการแปลงจากชุดทดสอบ โดยในรูปที่ 3.5 จะแสดงตัวอย่างรูปของศูนย์กลางการแปลงที่ได้จากชุดทดสอบแรก

(3) ทดสอบว่าศูนย์กลางการแปลงที่ได้มีความถูกต้องหรือไม่ โดยการตรวจสอบว่ามีกระบวนการทำงานใดหรือไม่ที่สามารถอยู่ในส่วนของการแปลงมากกว่าหนึ่งส่วน ถ้ามี จะถือว่าศูนย์กลางการแปลงดังกล่าวไม่ถูกต้อง ซึ่งจากรูปที่ 3.5 จะพบว่าศูนย์กลางการแปลงที่

ได้มีความถูกต้อง โดยในส่วนของการนำเข้าข้อมูลจะมีกระบวนการทำงานหมายเลข 1.0 ในส่วนของศูนย์กลางการแปลงจะมีกระบวนการทำงานหมายเลข 2.0 และในส่วนแสดงผลลัพธ์จะประกอบด้วยกระบวนการทำงานหมายเลข 3.0 4.0 5.0 และ 6.0

ตารางที่ 3.2 ตัวอย่างคู่ของจุดแบ่งของเส้นทางต่างๆ

เส้นทางที่	ลำดับที่ของคู่ของจุดแบ่ง	คู่ของจุดแบ่ง
1	1	(1.0-2.0) กับ (2.0-4.0)
1	2	(1.0-2.0) กับ (4.0-5.0)
1	3	(1.0-2.0) กับ (5.0-6.0)
1	4	(2.0-4.0) กับ (4.0-5.0)
1	5	(2.0-4.0) กับ (5.0-6.0)
1	6	(4.0-5.0) กับ (5.0-6.0)
2	1	(1.0-2.0-) กับ (2.0-3.0)
2	2	(1.0-2.0-) กับ (3.0-6.0)
2	3	(2.0-3.0-) กับ (3.0-6.0)



รูปที่ 3.5 ตัวอย่างศูนย์กลางการแปลงจากชุดทดสอบแรก

(4) ถ้าชุดทดสอบมีความถูกต้องจะจัดเก็บชุดทดสอบนี้เป็นคำตอบ ถ้าไม่ถูกต้องจะไม่จัดเก็บเป็นคำตอบ

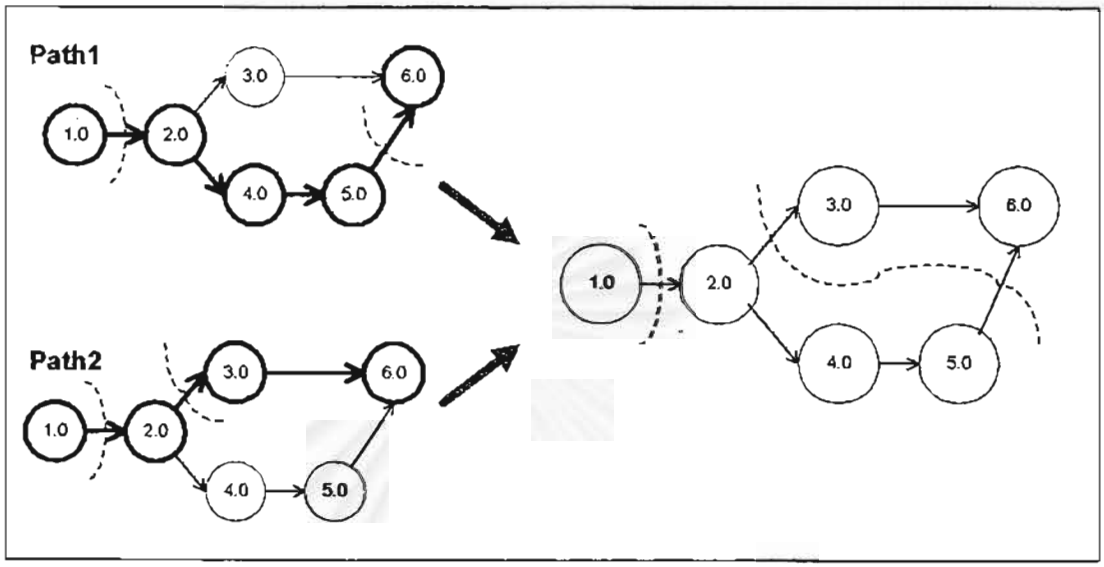
(5) เลือกคู่ของจุดแบ่งคู่อื่นจากแต่ละเส้นทาง แล้วทำการทดสอบความถูกต้องดังวิธีการในข้อ (1) – ข้อ (4) จนครบทุกคู่ของจุดแบ่ง ตารางที่ 3.3 แสดงตัวอย่างของชุดทดสอบทั้งหมดสำหรับแผนภาพกระแสข้อมูลนี้ รูปที่ 3.6 แสดงอีกตัวอย่างหนึ่งของศูนย์กลางการแปลงที่ถูกต้องโดยใช้ชุดทดสอบที่ 7 ส่วนรูปที่ 3.7 แสดงตัวอย่างศูนย์กลางการแปลงที่ไม่ถูกต้องซึ่งเกิดจากชุดทดสอบที่ 9 ซึ่งในรูปที่ 3.7 จะเห็นว่ากระบวนการทำงานหมายเลข 2.0 จะอยู่ในส่วนของศูนย์กลางการแปลงเมื่อพิจารณาจากส่วนแบ่งของเส้นทางที่ 1 แต่ในขณะเดียวกันกระบวนการทำงานหมายเลข 2.0 จะอยู่ในส่วนการนำเข้า ข้อมูลเมื่อพิจารณาจากส่วนแบ่งของเส้นทางที่ 2 ซึ่งลักษณะนี้จะถือว่าชุดทดสอบไม่ถูกต้อง

ตารางที่ 3.3 ตัวอย่างชุดทดสอบต่างๆ ที่ได้

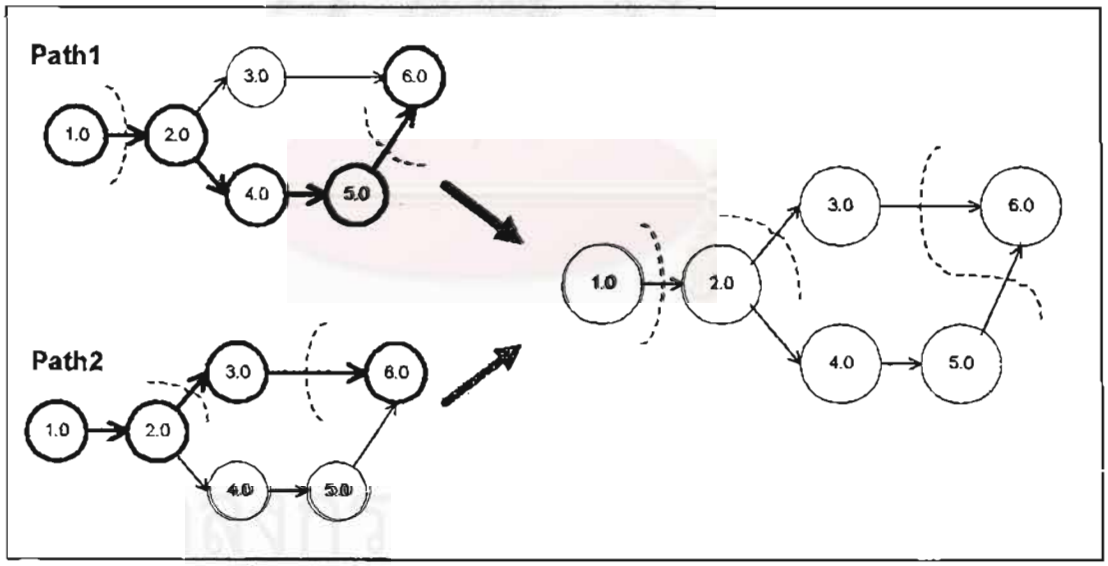
ชุดทดสอบที่	คู่ของจุดแบ่งจากเส้นทางที่ 1	คู่ของจุดแบ่งจากเส้นทางที่ 2	ส่วนนำเข้าข้อมูลของชุดทดสอบ	ส่วนแสดงผลลัพธ์ของชุดทดสอบ
1	1	1	(1.0-2.0)	(2.0-4.0), (2.0-3.0)
2	1	2	(1.0-2.0)	(2.0-4.0), (3.0-6.0)
3	1	3	(1.0-2.0), (2.0-3.0)	(2.0-4.0), (3.0-6.0)
4	2	1	(1.0-2.0)	(4.0-5.0), (2.0-3.0)
5	2	2	(1.0-2.0)	(4.0-5.0), (3.0-6.0)
6	2	3	(1.0-2.0), (2.0-3.0)	(4.0-5.0), (3.0-6.0)
7	3	1	(1.0-2.0)	(5.0-6.0), (2.0-3.0)
8	3	2	(1.0-2.0)	(5.0-6.0), (3.0-6.0)
9	3	3	(1.0-2.0), (2.0-3.0)	(5.0-6.0), (3.0-6.0)
10	4	1	(2.0-4.0), (1.0-2.0)	(4.0-5.0), (2.0-3.0)
11	4	2	(2.0-4.0), (1.0-2.0)	(4.0-5.0), (3.0-6.0)
12	4	3	(2.0-4.0), (2.0-3.0)	(4.0-5.0), (3.0-6.0)
13	5	1	(2.0-4.0), (1.0-2.0)	(5.0-6.0), (2.0-3.0)
14	5	2	(2.0-4.0), (1.0-2.0)	(5.0-6.0), (3.0-6.0)
15	5	3	(2.0-4.0), (2.0-3.0)	(5.0-6.0), (3.0-6.0)
16	6	1	(4.0-5.0), (1.0-2.0)	(5.0-6.0), (2.0-3.0)
17	6	2	(4.0-5.0), (1.0-2.0)	(5.0-6.0), (3.0-6.0)
18	6	3	(4.0-5.0), (2.0-3.0)	(5.0-6.0), (3.0-6.0)

4) สำหรับแต่ละคำตอบที่จัดเก็บไว้ จะนำมาสร้างผังโครงสร้างของโปรแกรม โดยการสร้างวัตถุของคลาส CObjMod จำนวน 4 มอดูลเพื่อใช้แสดงถึงมอดูลควบคุม

ของผังโครงสร้าง ได้แก่ มอดูลราก มอดูลควบคุมการนำเข้าข้อมูล มอดูลควบคุมการแปลงข้อมูล และมอดูลควบคุมการแสดงผลลัพธ์ จากนั้น ทำการสร้างวัตถุ CObjMod และ CObjLnk เพื่อแสดงถึงมอดูลและการเชื่อมโยงระหว่างมอดูลตามศูนย์กลางการแปลงที่ได้



รูปที่ 3.6 ตัวอย่างศูนย์กลางการแปลงจากชุดทดสอบที่ถูกต้อง



รูปที่ 3.7 ตัวอย่างศูนย์กลางการแปลงจากชุดทดสอบที่ไม่ถูกต้อง

3.2.4 วิธีการแปลงแผนภาพกระแสข้อมูลตามวิธีการวิเคราะห์แบบทรานแซกชัน

วิธีการในการแผนภาพกระแสข้อมูลตามวิธีการวิเคราะห์แบบทรานแซกชัน ประกอบด้วยขั้นตอนต่างๆ ดังนี้

1) สร้างวัตถุของคลาส CObjMod 3 มอดูลเพื่อใช้แสดงถึงมอดูลควบคุมของผังโครงสร้าง ได้แก่ มอดูลราก มอดูลควบคุมการนำเข้าข้อมูล และมอดูลศูนย์กลาง ทรานแซกชัน

2) อ่านข้อมูลกระบวนการทำงานที่นำข้อมูลเข้ามายังศูนย์กลางทรานแซกชัน เพื่อสร้างผังโครงสร้างสำหรับส่วนนำเข้าข้อมูลดังกล่าวต่อทางด้านล่างของมอดูลควบคุมการนำเข้าข้อมูล

3) สำหรับแต่ละการไหลของข้อมูลที่ออกจากศูนย์กลางทรานแซกชันซึ่งแสดงถึงเส้นทางการกระทำของแผนภาพกระแสข้อมูลจะพิจารณาว่าส่วนของการไหลดังกล่าวมีคุณลักษณะการไหลแบบการแปลงหรือแบบทรานแซกชัน เพื่อทำการแปลงด้วยวิธีการวิเคราะห์แบบแปลงหรือการวิเคราะห์แบบทรานแซกชันตามลำดับ ซึ่งจากขั้นตอนนี้ จะได้ส่วนของผังโครงสร้างสำหรับแต่ละส่วนของการไหลข้อมูลที่ออกจากศูนย์กลางทรานแซกชัน

4) นำส่วนของผังโครงสร้างทั้งหมดที่ได้จากข้อ 3) มาต่อเข้ากับผังโครงสร้างที่ได้จากข้อ 2) โดยให้อยู่ภายใต้มอดูลศูนย์กลางทรานแซกชัน

3.2.5 วิธีการวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล

ในงานวิจัยนี้ ผู้วิจัยได้ทำการออกแบบภาษาในการออกแบบโปรแกรมโดยนำภาษาในการออกแบบโปรแกรมที่เสนอโดยโรเจอร์ เพรสแมน [1] มาปรับปรุงและเพิ่มเติมคำสั่งใหม่ๆ เข้าไป โดยเขียนอยู่รูปแบบบีเอ็นเอฟดังแสดงในภาคผนวกที่ ข. ซึ่งจากรูปแบบบีเอ็นเอฟจะช่วยให้เครื่องมือซอฟต์แวร์สามารถตรวจสอบได้ว่าคำอธิบายการทำงานของมอดูลที่ผู้ใช้สร้างขึ้นเขียนถูกต้องตามกฎไวยากรณ์ของภาษาในการออกแบบโปรแกรมที่ได้กำหนดไว้หรือไม่ สำหรับขั้นตอนในการวัดค่าความสัมพันธ์ระหว่าง มอดูลจะประกอบด้วยขั้นตอนต่างๆ ดังนี้

1) อ่านบีเอ็นเอฟเข้ามาทำการแปลงให้เป็นโทเค็น ซึ่งจะได้ผลลัพธ์ออกมา 2 ชนิด คือ

(1) รายการโทเค็นของคำหลัก

(2) รายการโปรดักชั่นของบีเอ็นเอฟ ซึ่งประกอบด้วยชื่อของนอนเทอร์มินอล และรายการโทเค็นที่เป็นคำอธิบายของโปรดักชั่น

2) อ่านคำอธิบายการทำงานของมอดูล (ซึ่งอยู่ในรูปแบบของภาษาในการออกแบบโปรแกรม) เข้ามาทีละมอดูลเพื่อทำการแปลงให้เป็นโทเค็น ซึ่งจะได้โทเค็นเพิ่มเติมออกมา 2 กลุ่ม คือ ตัวระบุ และ ค่าคงที่

3) ทำการวิเคราะห์ไวยากรณ์ของคำอธิบายการทำงานของมอดูลโดยการเปรียบเทียบกับบีเอ็นเอฟ เริ่มจากการอ่านโทเค็นของคำอธิบายการทำงานของมอดูลเข้ามาทีละโทเค็นเพื่อเปรียบเทียบกับรายการโทเค็นของโปรดักชั่นแรกของบีเอ็นเอฟ ถ้าผลการเปรียบเทียบไม่ตรงกันจะข้ามไปเปรียบเทียบกับโปรดักชั่นถัดไป

4) ในกรณีที่อ่านโทเค็นของคำอธิบายการทำงานของมอดูลเข้ามาจนครบหนึ่งประโยคคำสั่ง จะทำการวิเคราะห์คำสั่งเพื่อเก็บข้อมูลไว้ใช้ในการวัดค่าความสัมพันธ์ระหว่างมอดูล โดยการตรวจสอบถึงประเภทและลักษณะการใช้งานของพารามิเตอร์และตัวแปรตามวิธีการวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของออฟฟัต เช่น เป็นตัวแปรแบบครอบคลุม ใช้คำนวณค่า ใช้เป็นพรดิเคต หรือใช้แบบทางอ้อม เป็นต้น ซึ่งผู้วิจัยได้ออกแบบกฎที่จะใช้ในการตรวจสอบประเภทและลักษณะการใช้งานของพารามิเตอร์สำหรับบีเอ็นเอฟที่กำหนดดังแสดงในรูปที่ 3.8 โดยได้อธิบายความหมายของคำพิเศษ (หมายถึงคำที่อยู่ระหว่างเครื่องหมาย "<<" กับ ">>") จะมีความหมายดังแสดงในตารางที่ 3.4

```
module <<module create>> ( <<parameter create>> )
define type <<struct create>> as
global <type identifier> <<global create>> ;
shared <type identifier> <<shared create>> ;
if <<predicate>> then <<calling>> endif
elseif <<predicate>> then <<check calling>> endif
while <<predicate>> do <<check calling>> endwhile
repeat <<check calling>> until <<predicate>> ;
call <<called>> ( <<parameter pass>> )
read <variable list> from <<external>> ;
write <variable list> into <<external>> ;
message <<compute>> ;
print <<compute>> ;
<type identifier> <<variable create>> ;
<<assigned>> = call <<called>> ( <<param pass>> )
<<assigned>> = <<compute>> ;
```

รูปที่ 3.8 กฎในการตรวจสอบเพื่อวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล

5) อ่านโทเค็นของคำอธิบายการทำงานของมอดูลถัดไปเข้ามา และดำเนินการตามขั้นตอนที่ 3) – 4) จนจบคำอธิบายการทำงานของมอดูล

6) ทำการวัดค่าระดับความสัมพันธ์ระหว่างมอดูลจากข้อมูลที่ได้จากขั้นตอนที่ 4) จนครบทุกมอดูล

7) คำนวณหาค่าความสัมพันธ์ระหว่างมอดูลเฉลี่ยของผังโครงสร้างของโปรแกรม

ตารางที่ 3.4 ความหมายของคำพิเศษในกฎการวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล

คำพิเศษ	ความหมายเมื่อพบตัวระบุในตำแหน่งนี้
<<module create>>	เป็นตัวระบุมอดูล
<<parameter create>>	เป็นตัวระบุพารามิเตอร์

ตารางที่ 3.4 ความหมายของคำพิเศษในกฎการวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล (ต่อ)

คำพิเศษ	ความหมายเมื่อพบตัวระบุในตำแหน่งนี้
<<struct create>>	เป็นตัวระบุชนิดข้อมูลแบบโครงสร้าง
<<global create>>	เป็นตัวระบุตัวแปรที่เป็นแบบครอบคลุม
<<shared create>>	เป็นตัวระบุตัวแปรที่เป็นแบบใช้ร่วมกัน
<<predicate>>	ตัวระบุถูกใช้งานเป็นเงื่อนไขแบบเพรดิเคต ถ้าพบว่ามีเงื่อนไขแบบเพรดิเคตเกิดขึ้นจริงจากคำพิเศษ <<check calling>>
<<check calling>>	ถ้าพบตัวระบุมอดูล ถือว่ามีเงื่อนไขแบบเพรดิเคตเกิดขึ้นจริง
<<called>>	ถ้าพบตัวระบุมอดูล ถือว่ามีการเรียกทำงานระหว่างมอดูลเกิดขึ้น
<<parameter pass>>	ตัวระบุถูกใช้เป็นพารามิเตอร์ในการเรียกทำงานระหว่างมอดูล
<<external>>	เป็นตัวระบุสื่อภายนอก
<<compute>>	ตัวระบุถูกใช้งานแบบคำนวณค่า
<<assigned>>	ตัวระบุถูกใช้งานแบบคำนวณค่าและเป็นตัวที่ถูกแก้ไขค่า
<<variable create>>	เป็นตัวระบุตัวแปร

3.3 การออกแบบส่วนจัดเก็บข้อมูล

ในงานวิจัยนี้ ผู้วิจัยได้ทำการออกแบบการจัดเก็บข้อมูลโดยการใช้แฟ้มข้อมูล ซึ่งประกอบด้วยแฟ้มข้อมูลที่ใช้จัดเก็บแผนภาพกระแสข้อมูล และแฟ้มข้อมูลที่ใช้จัดเก็บผังโครงสร้างของโปรแกรม โดยมีรายละเอียดโครงสร้างของแฟ้มข้อมูลดังนี้

3.3.1 แฟ้มข้อมูลของแผนภาพกระแสข้อมูล

แฟ้มข้อมูลนี้จะประกอบด้วย 6 ส่วนดังแสดงตัวอย่างในรูปที่ 3.9 ซึ่งเป็นแฟ้มข้อมูลของแผนภาพกระแสข้อมูลในรูปที่ 3.4 โดยแต่ละส่วนจะใช้เครื่องหมาย "|" ในการคั่นข้อมูล และแต่ละส่วนจะมีความหมายดังนี้

1) ส่วนหัวของแฟ้มข้อมูล จะประกอบด้วย 2 บรรทัด คือ ชื่อโปรแกรม และชนิดของแผนภาพที่จัดเก็บ โดยข้อความ 2 บรรทัดดังกล่าวจะเป็นดังนี้

*** DFD to SC Application ***

*** Data Flow Diagram File ***

2) ส่วนขององค์ประกอบที่เป็นเอนทิตีภายนอก โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Entity]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของเอนทิตีภายนอก โดยที่

หนึ่งบรรทัดจะใช้อธิบายถึงเอนทิตีหนึ่งเอนทิตี สำหรับรายละเอียดของเอนทิตีภายนอกที่จัดเก็บประกอบด้วย

(1) หมายเลขอ้างอิงของเอนทิตีภายนอก

(2) ชื่อของเอนทิตีภายนอก

(3) คำอธิบายของเอนทิตีภายนอก

(4) หมายเลขภายในของเอนทิตีภายนอก เพื่อใช้อ้างอิงสำหรับโปรแกรม

(5) แฟลกสถานะว่าเอนทิตีนี้เป็นตัวสำเนาของเอนทิตีภายนอกหรือไม่ โดยค่า 0 คือไม่เป็นตัวสำเนา หรือค่า 1 คือเป็นตัวสำเนา

```
*** DFD to SC ***
*** DFD File ***
[Entity]
1|customer|customer that order goods|7|0|-154|266|-68|215|
2|customer account|customer account for payment|8|0|550|245|636|193|
[Process]
1.0|validate customer|validate customer data|1|-118|132|-18|32|
2.0|classify customer|classify customer type (oridary or VIP customer)|2|62|132|162|32|
3.0|process regular|calculate customer order|3|169|270|269|170|
4.0|process bulk|calculate VIP order and find account number|4|182|-5|285|-105|
5.0|calc discount|calculate order discount|5|352|-9|458|-108|
6.0|put invoice|print order invoice|6|378|269|482|165|
[DataStore]
D1|customer file|customer data file|9|0|-117|-20|-22|-54|
[Flow]
customer|validate customer|order|7|1|1|0|-68|82|-111|265|50|-50|50|-50|
customer file|validate customer|customer_info|9|1|2|0|-70|47|-66|-37|79|-21|50|-50|
validate customer|classify customer|valid_order|1|2|0|0|112|82|-68|82|50|-50|50|-50|
classify customer|process regular|regular_order|2|3|0|0|219|220|112|82|50|-50|50|-50|
classify customer|process bulk|bulk_order|2|4|0|0|236|-59|112|82|50|-50|54|-54|
process bulk|calc discount|acct_info|4|5|0|0|400|-62|236|-59|54|-54|59|-59|
calc discount|put invoice|bulk_invoice|5|6|0|0|430|217|411|-68|59|-59|52|-52|
process regular|put invoice|regular_invoice|3|6|0|0|430|217|219|220|50|-50|52|-52|
put invoice|customer_account|invoice|6|8|0|1|605|217|430|217|52|-52|55|-55|
[Specification]
validate customer::BEGIN
module validate_customer ()
begin
        customer_id = prompt "Customer ID:";
        read customer_data from customer_file;
        if (not found) then
          message "Invalid customer ID";
          exit;
        endif;

        line = 1;
        input order_line;
        while (order_line <> NULL)
              order[line] = order_line;
              line = line+1;
              intput order_line;
        endwhile
end
validate customer::END
```

รูปที่ 3.9 ตัวอย่างแฟ้มข้อมูลของแผนภาพกระแสข้อมูล

(6) ตำแหน่งของเอนทิตีภายนอกในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง

3) ส่วนขององค์ประกอบที่เป็นกระบวนการทำงาน โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Process]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของกระบวนการทำงาน โดยที่หนึ่งบรรทัดจะใช้อธิบายถึงกระบวนการทำงานหนึ่งกระบวนการ สำหรับรายละเอียดของกระบวนการทำงานที่จัดเก็บประกอบด้วย

(1) หมายเลขอ้างอิงของกระบวนการทำงาน

(2) ชื่อของกระบวนการทำงาน

(3) คำอธิบายของกระบวนการทำงาน

(4) หมายเลขภายในของกระบวนการทำงาน เพื่อใช้อ้างอิงสำหรับเครื่องมือซอฟต์แวร์

(5) ตำแหน่งของกระบวนการทำงานในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง

4) ส่วนขององค์ประกอบที่เป็นหน่วยเก็บข้อมูล โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[DataStore]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของหน่วยเก็บข้อมูล โดยที่หนึ่งบรรทัดจะใช้อธิบายถึงหน่วยเก็บข้อมูลหนึ่งหน่วย สำหรับรายละเอียดของหน่วยเก็บข้อมูลที่จัดเก็บประกอบด้วย

(1) หมายเลขอ้างอิงของหน่วยเก็บข้อมูล

(2) ชื่อของหน่วยเก็บข้อมูล

(3) คำอธิบายของหน่วยเก็บข้อมูล

(4) หมายเลขภายในของหน่วยเก็บข้อมูล เพื่อใช้อ้างอิงสำหรับโปรแกรม

(5) แฟลกสถานะว่าหน่วยนี้เป็นตัวสำเนาของหน่วยเก็บข้อมูลหรือไม่ โดยค่า 0 คือไม่เป็นตัวสำเนา หรือค่า 1 คือเป็นตัวสำเนา

(6) ตำแหน่งของหน่วยเก็บข้อมูลในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง

5) ส่วนขององค์ประกอบที่เป็นการไหลของข้อมูล โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Flow]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของการไหลของข้อมูล โดยที่หนึ่งบรรทัดจะใช้อธิบายถึงการไหลของข้อมูลหนึ่งคู่ระหว่างองค์ประกอบ สำหรับรายละเอียดของการไหลของข้อมูลที่จัดเก็บประกอบด้วย

(1) ชื่อขององค์ประกอบที่เป็นตัวส่งข้อมูล

(2) ชื่อขององค์ประกอบที่เป็นตัวรับข้อมูล

(3) หมายเลขภายในขององค์ประกอบที่เป็นตัวส่งข้อมูล

(4) หมายเลขภายในขององค์ประกอบที่เป็นตัวรับข้อมูล

(5) ชนิดขององค์ประกอบที่เป็นตัวส่งข้อมูล ว่าเป็นกระบวนการทำงาน เอนทิตีภายนอก หรือหน่วยจัดเก็บข้อมูล ซึ่งจะมีค่าเป็น 0 1 หรือ 2 ตามลำดับ

(6) ชนิดขององค์ประกอบที่เป็นตัวรับข้อมูล ว่าเป็นกระบวนการทำงาน เอนทิตีภายนอก หรือหน่วยจัดเก็บข้อมูล ซึ่งจะมีค่าเป็น 0 1 หรือ 2 ตามลำดับ

(7) ตำแหน่งของเส้นการไหลในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง

(8) ตำแหน่งเปรียบเทียบระหว่างจุดมุมซ้ายบนขององค์ประกอบที่เป็นตัวส่งข้อมูลกับจุดมุมซ้ายบนของเส้นการไหลของข้อมูล

(9) ตำแหน่งเปรียบเทียบระหว่างจุดมุมซ้ายบนขององค์ประกอบที่เป็นตัวรับข้อมูลกับจุดมุมขวาล่างของเส้นการไหลของข้อมูล

6) ส่วนของข้อกำหนดการทำงานของกระบวนการทำงาน โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Specification]" บรรทัดที่สองจะเป็นชื่อของกระบวนการทำงานและคำว่า ":BEGIN" ตามด้วยชุดของบรรทัดที่เป็นข้อกำหนดการทำงาน และปิดท้ายด้วยบรรทัดที่เป็นชื่อของกระบวนการทำงานและคำว่า ":END" สำหรับในงานวิจัยนี้จะใช้ภาษาในการออกแบบโปรแกรมในการเขียนข้อกำหนดในการทำงานของกระบวนการทำงาน

3.3.2 แฟ้มข้อมูลของผังโครงสร้างของโปรแกรม

ในรูปที่ 3.10 จะแสดงตัวอย่างของผังโครงสร้างของโปรแกรม ซึ่งแฟ้มข้อมูลสำหรับผังโครงสร้างของโปรแกรมที่ออกแบบไว้จะประกอบด้วย 4 ส่วน ดังแสดงตัวอย่างในรูปที่ 3.11 โดยแต่ละส่วนจะใช้เครื่องหมาย "|" ในการคั่นข้อมูล และแต่ละส่วนจะมีความหมายดังนี้

1) ส่วนหัวของแฟ้มข้อมูล จะประกอบด้วย 2 บรรทัด คือ ชื่อโปรแกรมและชนิดของแผนภาพที่จัดเก็บ โดยข้อความ 2 บรรทัดดังกล่าวจะเป็นดังนี้

*** DFD to SC Application ***

*** Structure Chart File ***

2) ส่วนขององค์ประกอบที่เป็นมอดูล โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Module]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของมอดูล โดยที่หนึ่งบรรทัดจะใช้อธิบายถึงมอดูลหนึ่งมอดูล สำหรับรายละเอียดของมอดูลที่จัดเก็บประกอบด้วย

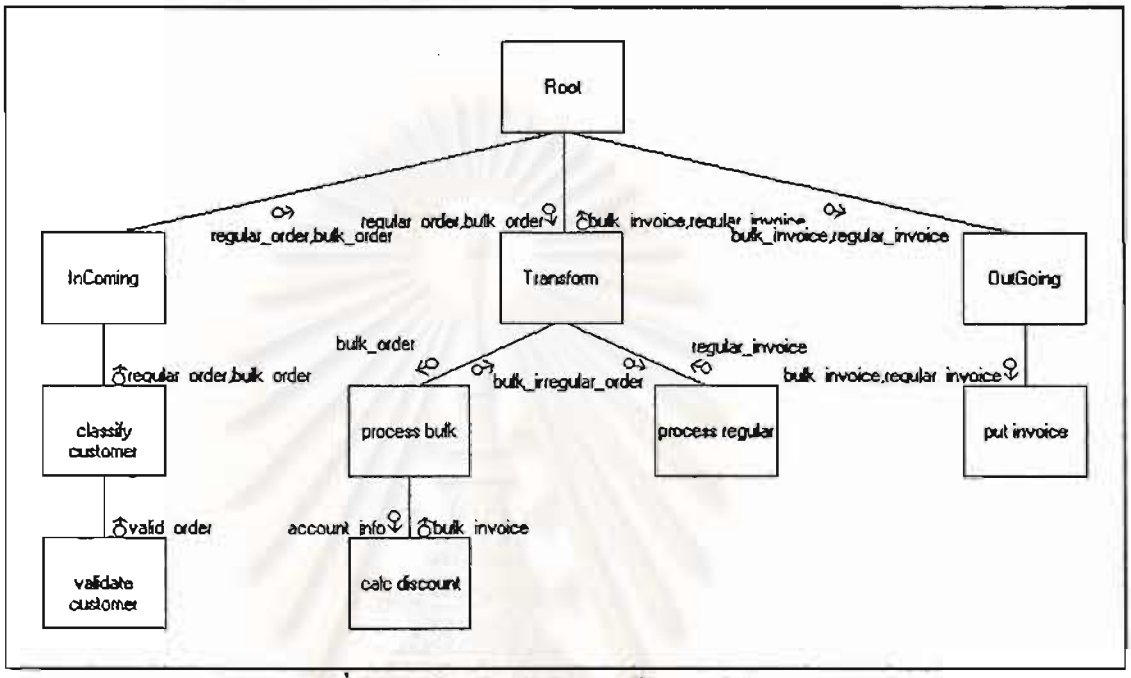
(1) หมายเลขอ้างอิงของมอดูล

(2) ชื่อของมอดูล

(3) คำอธิบายของมอดูล

(4) หมายเลขภายในของมอดูล เพื่อใช้อ้างอิงสำหรับเครื่องมือซอฟต์แวร์

(5) ตำแหน่งของมอดูลในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง



รูปที่ 3.10 ตัวอย่างผังโครงสร้างของโปรแกรม

3) ส่วนขององค์ประกอบที่เป็นการเรียกทำงานระหว่างมอดูล โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Link]" และตามด้วยบรรทัดที่ใช้อธิบายถึงรายละเอียดของการทำงาน โดยที่หนึ่งบรรทัดจะใช้อธิบายถึงการเรียกทำงานระหว่างมอดูลหนึ่งคู่ สำหรับรายละเอียดของการเรียกทำงานที่จัดเก็บประกอบด้วย

(1) ชื่อของมอดูลที่เป็นตัวเรียกทำงาน

(2) ชื่อของมอดูลที่ถูกเรียกทำงาน

(3) หมายเลขภายในของมอดูลที่เป็นตัวเรียกทำงาน

(4) หมายเลขภายในของมอดูลที่ถูกเรียกทำงาน

(5) รายการของตัวคู่ต่อแบบข้อมูลในทิศทางไหลลง

(6) รายการของตัวคู่ต่อแบบควบคุมในทิศทางไหลลง

(7) รายการของตัวคู่ต่อแบบข้อมูลในทิศทางไหลขึ้น

(8) รายการของตัวคู่ต่อแบบควบคุมในทิศทางไหลขึ้น

(9) ตำแหน่งของเส้นเชื่อมโยงระหว่างมอดูลในแผนภาพ ซึ่งใช้รูปแบบพิกัด (X,Y) ของจุดมุมซ้ายบนและมุมขวาล่าง

(10) ตำแหน่งเปรียบเทียบระหว่างจุดมุมซ้ายบนขององค์ประกอบที่เป็นตัวส่งข้อมูลกับจุดมุมซ้ายบนของเส้นเชื่อมโยงมอดูล

(11) ตำแหน่งเปรียบเทียบระหว่างจุดมุมซ้ายบนขององค์ประกอบที่เป็นตัวรับข้อมูลกับจุดมุมขวาล่างของเส้นเชื่อมโยงมอดูล

```
*** DFD to SC ***
*** SC File ***
[Module]
 |Root|root control module|44|500|-100|580|-160|
 |InComing|in-coming control module|45|200|-200|280|-260|
 |Transform|transform center module|46|500|-200|580|-260|
 |OutGoing|out-going control module|47|800|-200|880|-260|
2.0|classify customer|classify customer type (oridary or VIP customer)|48|200|-300|280|-360|
1.0|validate customer|validate customer data|49|200|-400|280|-460|
5.0|process bulk|calculate VIP order and find account number|50|400|-300|480|-360|
6.0|calc discount|calculate order discount|51|400|-400|480|-460|
3.0|process regular|calculate customer order|52|600|-300|680|-360|
4.0|put invoice|print order invoice|53|800|-300|880|-360|
[Link]
Root|InComing|44|45| | |regular_order,bulk_order| |540|-159|240|-201|40|59|40|1|
Root|Transform|44|46|regular_order,bulk_order| |bulk_invoice,regular_invoice| |540|-159|540|-
201|40|59|40|1|
Root|OutGoing|44|47|bulk_invoice,regular_invoice| | | |540|-159|840|-201|40|59|40|1|
InComing|classify customer|45|48| | |regular_order,bulk_order| |240|-259|240|-301|40|59|40|1|
Transform|process bulk|46|50|bulk_order| |bulk_invoice| |540|-259|440|-301|40|59|40|1|
Transform|process regular|46|52|regular_order| |regular_invoice| |540|-259|640|-301|40|59|40|1|
OutGoing|put invoice|47|53|bulk_invoice,regular_invoice| | | |840|-259|840|-301|40|59|40|1|
classify customer|validate customer|48|49| | |valid_order| |240|-359|240|-401|40|59|40|1|
process bulk|calc discount|50|51|account_info| |bulk_invoice| |440|-359|440|-401|40|59|40|1|
[Specification]
validate customer::BEGIN
module validate_customer ()
begin
        customer_id = prompt "Customer ID:";
        read customer_data from customer_file;
        if (not found) then
          message "Invalid customer ID";
          exit;
        endif;

        line = 1;
        input order_line;
        while (order_line <> NULL)
              order[line] = order_line;
              line = line+1;
              intput order_line;
        endwhile
end
validate customer::END
```

รูปที่ 3.11 ตัวอย่างแฟ้มข้อมูลของผังโครงสร้างของโปรแกรม

4) ส่วนของข้อกำหนดการทำงานของมอดูล โดยบรรทัดแรกในส่วนนี้จะเป็นข้อความว่า "[Specification]" บรรทัดที่สองจะเป็นชื่อของมอดูลและคำว่า ":BEGIN" ตามด้วยชุดของบรรทัดที่เป็นข้อกำหนดการทำงาน และปิดท้ายด้วยบรรทัดที่เป็นชื่อของมอดูลและคำว่า ":END" สำหรับในงานวิจัยนี้จะใช้ภาษาในการออกแบบโปรแกรมในการเขียนข้อกำหนดในการทำงานของมอดูล

บทที่ 4

การพัฒนาเครื่องมือซอฟต์แวร์

4.1 สภาพแวดล้อมในการพัฒนาเครื่องมือซอฟต์แวร์

งานวิจัยนี้ทำการพัฒนาเครื่องมือซอฟต์แวร์โดยใช้เครื่องมือในการพัฒนาโปรแกรมไมโครซอฟต์วิชวลซีพลัสพลัส เวอร์ชัน 6.0 (Microsoft Visual C++ 6.0) สำหรับเครื่องคอมพิวเตอร์ที่ใช้พัฒนาเครื่องมือในงานวิจัยนี้เป็นเครื่องไมโครคอมพิวเตอร์ที่ใช้หน่วยประมวลผลแบบเพนเทียมความเร็ว 133 MHz และมีหน่วยความจำหลัก 32 MB บนระบบปฏิบัติการไมโครซอฟต์วินโดว์ (Microsoft Windows 95)

4.2 การพัฒนาหน้าจอโปรแกรม

เครื่องมือซอฟต์แวร์ที่พัฒนาขึ้นในงานวิจัยนี้ใช้รูปแบบการทำงานแบบการติดต่อหลายเอกสาร (Multi-Document Interface – MDI) เพื่อความสะดวกในการใช้งาน เนื่องจากผู้ใช้สามารถเปิดแผนภาพขึ้นมาทำงานได้หลายแผนภาพพร้อมๆ กัน โดยมีการจัดเตรียมระบบเมนูการทำงาน (Menu) และแถบเครื่องมือ (Tool Bar) ให้กับผู้ใช้ด้วย สำหรับหน้าจอโปรแกรมของเครื่องมือซอฟต์แวร์นี้สามารถแบ่งออกเป็น 3 กลุ่มตามลำดับขั้นตอนการใช้งานดังนี้

4.2.1 ขั้นตอนการวาดแผนภาพกระแสข้อมูล

เมื่อผู้ใช้เรียกเครื่องมือซอฟต์แวร์ขึ้นมาทำงานจะปรากฏหน้าจอโปรแกรมดังแสดงในรูปที่ 4.1 โดยเครื่องมือซอฟต์แวร์จะเริ่มต้นการทำงานโดยอยู่ในขั้นตอนการวาดแผนภาพกระแสข้อมูล ผู้ใช้สามารถวาดแผนภาพกระแสข้อมูลโดยเริ่มต้นจากการเลือกองค์ประกอบที่ต้องการวาดจากปุ่มคำสั่งที่อยู่ในแถบเครื่องมือการวาดซึ่งประกอบด้วยปุ่มต่างๆ ดังนี้

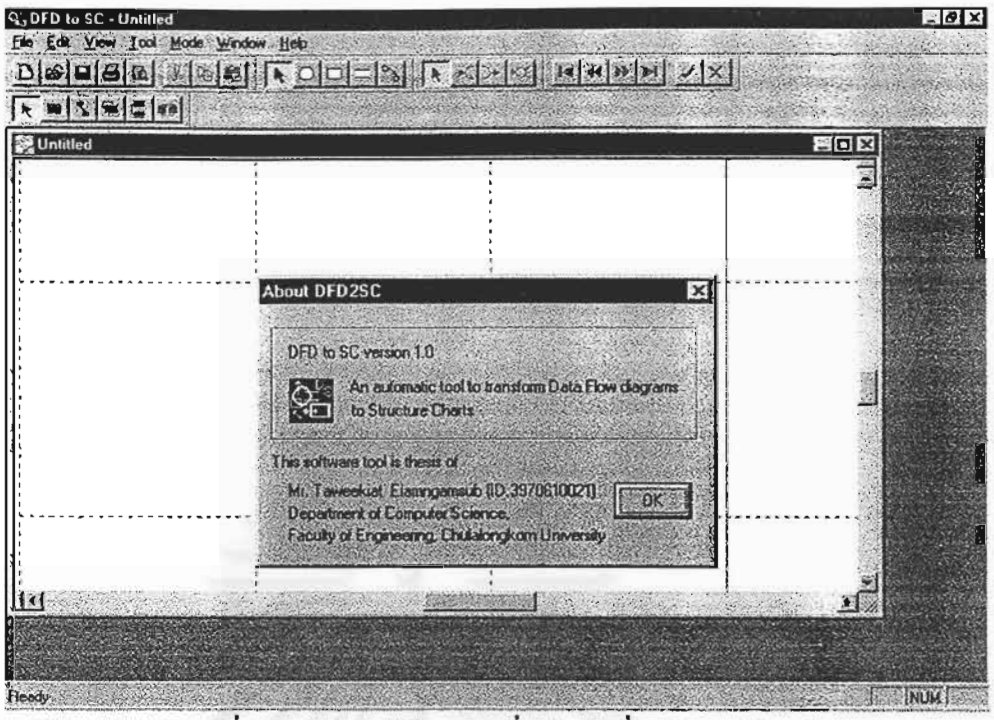
 เลือกเครื่องมือวาดเอนทิตีภายนอก

 เลือกเครื่องมือวาดกระบวนการทำงาน

 เลือกเครื่องมือวาดหน่วยจัดเก็บข้อมูล

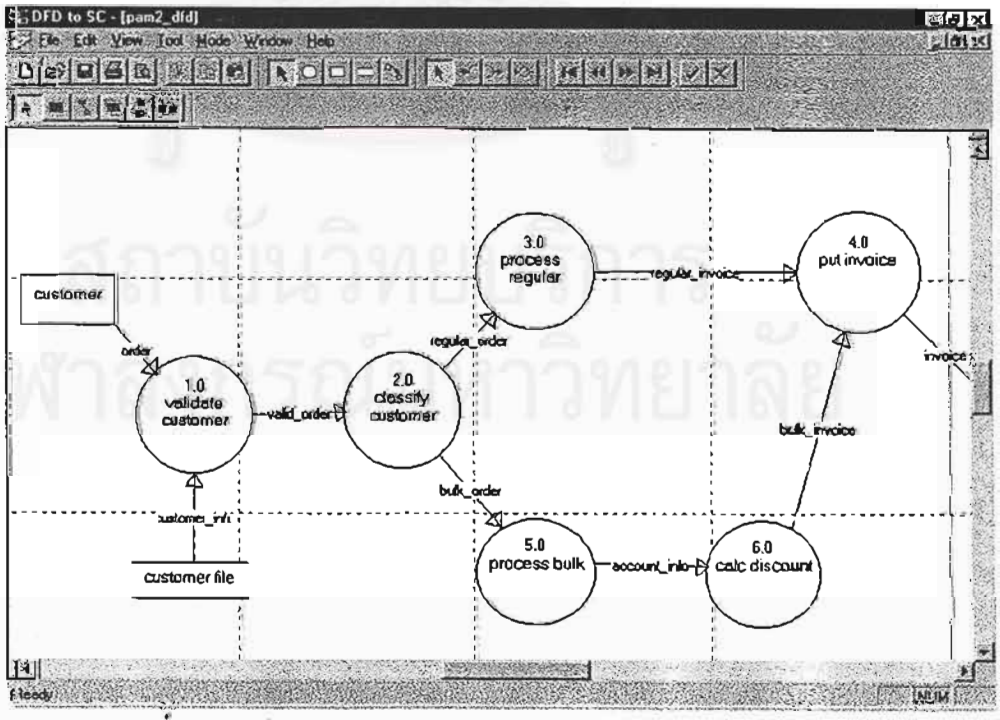
 เลือกเครื่องมือวาดการไหลของข้อมูล

 ยกเลิกการเลือกเครื่องมือการวาด

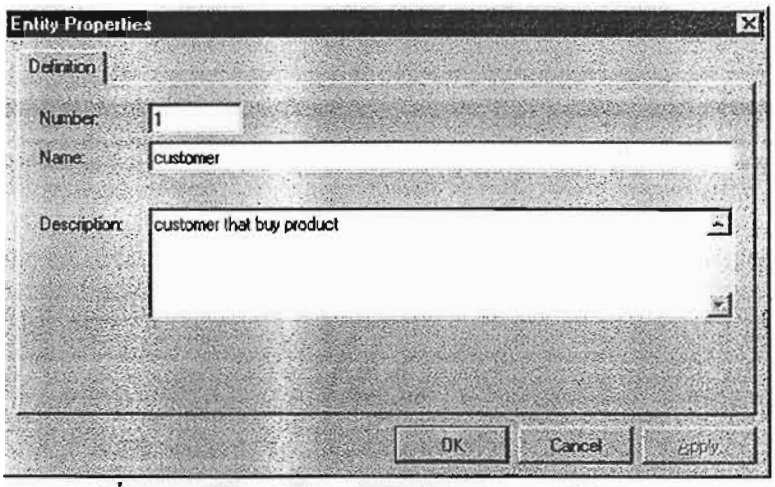


รูปที่ 4.1 หน้าจอโปรแกรมเมื่อเข้าสู่เครื่องมือซอฟต์แวร์

ในรูปที่ 4.2 แสดงตัวอย่างของแผนภาพกระแสข้อมูลที่ผู้ใช้วาดโดยใช้เครื่องมือซอฟต์แวร์ที่พัฒนา เมื่อผู้ใช้ทำการวาดองค์ประกอบแล้วสามารถกำหนดรายละเอียดขององค์ประกอบโดยการดับเบิ้ลคลิกเมาส์ที่ตัวองค์ประกอบ ซึ่งจะปรากฏหน้าต่างบันทึกรายละเอียดขององค์ประกอบดังแสดงในรูปที่ 4.3 - รูปที่ 4.6

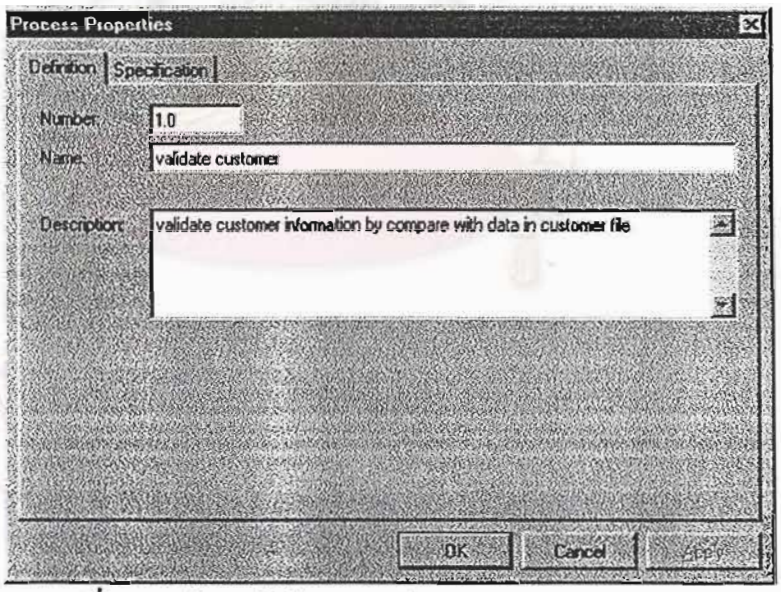


รูปที่ 4.2 หน้าจอโปรแกรมสำหรับการวาดแผนภาพกระแสข้อมูล

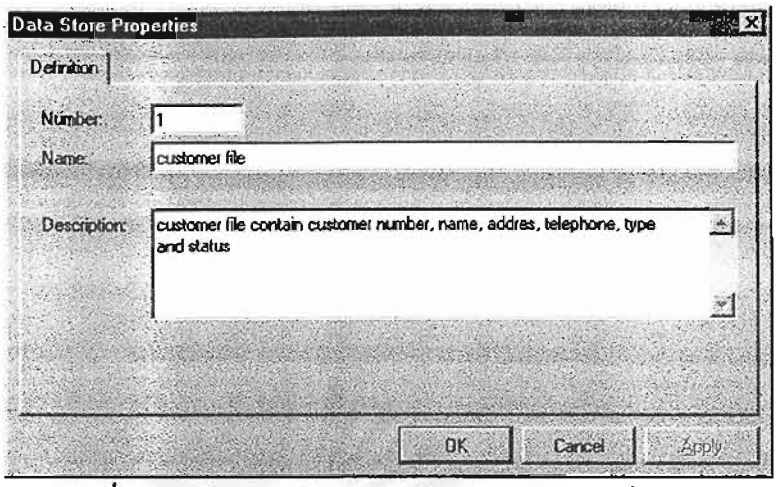


รูปที่ 4.3 หน้าจอบันทึกรายละเอียดของเอนทิตีภายนอก

หลังจากผู้ใช้วาดรูปแผนภาพกระแสข้อมูลเสร็จแล้วสามารถจัดเก็บแผนภาพลงยังแฟ้มข้อมูลเพื่อนำมาปรับปรุงแก้ไขในภายหลังได้ นอกจากนี้ ในกรณีที่ผู้ใช้มีแฟ้มข้อมูลแผนภาพกระแสของข้อมูลที่สร้างด้วยโปรแกรม Sybase's Power Designer 6.1 อยู่แล้ว สามารถนำแฟ้มข้อมูลดังกล่าวเข้ามาได้โดยใช้เมนูคำสั่ง "File"->"Import" เช่นกัน



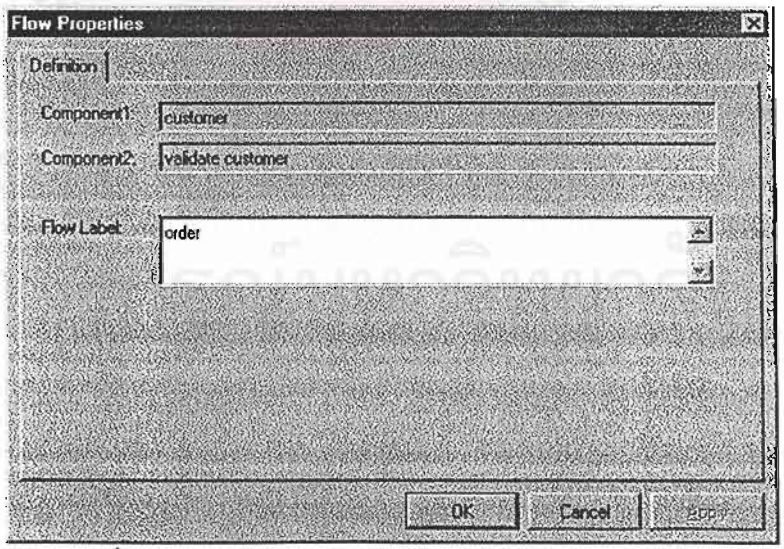
รูปที่ 4.4 หน้าจอบันทึกรายละเอียดของกระบวนการทำงาน



รูปที่ 4.5 หน้าจอบันทึกรายละเอียดของหน่วยจัดเก็บข้อมูล

4.2.2 ขั้นตอนการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม

เมื่อผู้ใช้วาดแผนภาพกระแสข้อมูลครบถ้วนแล้วสามารถสั่งให้เครื่องมือซอฟต์แวร์ทำการแปลงแผนภาพกระแสข้อมูลที่วาดเป็นผังโครงสร้างของโปรแกรมโดยใช้คำสั่ง "Mode"-> "Auto-Transform" ซึ่งจะปรากฏหน้าจอแสดงผลลัพธ์การแปลงดังในรูปที่ 4.7 อนึ่ง ก่อนที่เครื่องมือซอฟต์แวร์จะทำการแปลงแผนภาพกระแสข้อมูล เครื่องมือซอฟต์แวร์จะทำการตรวจสอบความถูกต้องของแผนภาพด้วย เช่น เอนทิตีจะต้องไม่เชื่อมโยงกับเอนทิตีหรือหน่วยจัดเก็บข้อมูล ทุกกระบวนการทำงานจะต้องมีทั้งการไหลเข้าและการไหลออกของข้อมูล เป็นต้น



รูปที่ 4.6 หน้าจอบันทึกรายละเอียดของการไหลของข้อมูล

สำหรับแผนภาพกระแสข้อมูลหนึ่งรูป สามารถที่จะแปลงเป็นผังโครงสร้างของโปรแกรมได้มากกว่าหนึ่งผังได้ โดยในหน้าจอนี้ เครื่องมือซอฟต์แวร์จะแสดงผังโครงสร้างของโปรแกรมทีละผัง ซึ่งผู้ใช้สามารถทีละเลื่อนดูผังโครงสร้างผังอื่นได้โดยการใช้ปุ่มคำสั่งที่อยู่ในแถบเครื่องมือแสดงผล ซึ่งประกอบด้วยปุ่มต่างๆ ดังนี้

 ให้แสดงผังโครงสร้างของโปรแกรมแรก

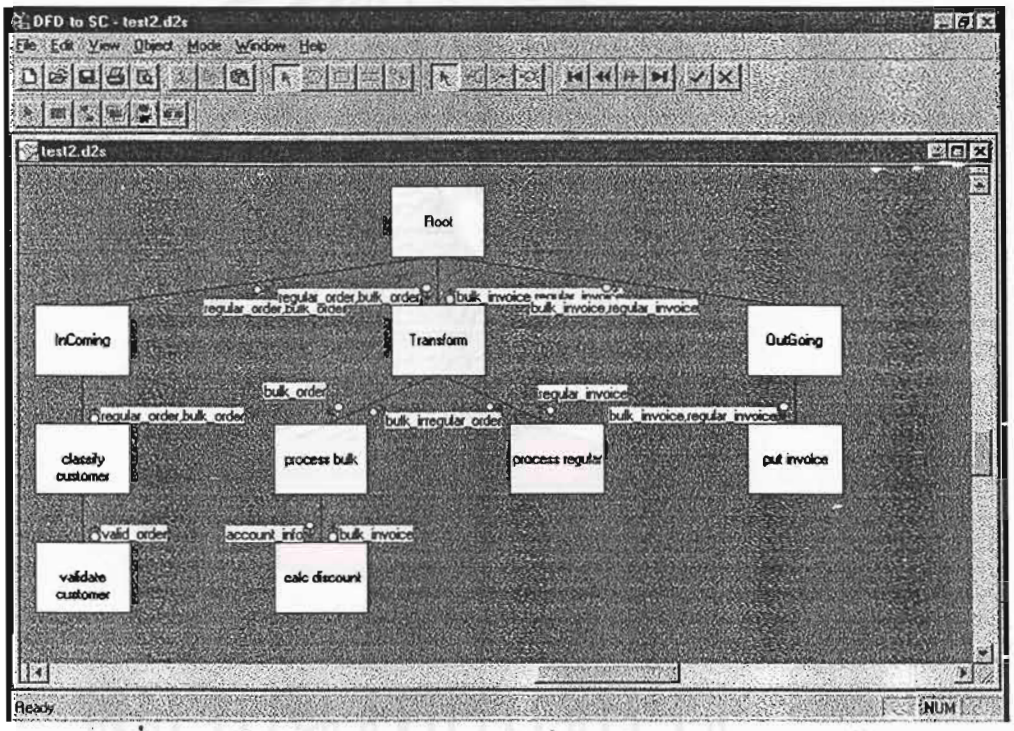
 ให้แสดงผังโครงสร้างของโปรแกรมก่อนหน้านี้

 ให้แสดงผังโครงสร้างของโปรแกรมถัดไป

 ให้แสดงผังโครงสร้างของโปรแกรมสุดท้าย

 เลือกและจัดเก็บผังโครงสร้างของโปรแกรมปัจจุบันลงแฟ้มข้อมูล

 ยกเลิกการแปลงและกลับไปยังหน้าจอการวาดแผนภาพกระแสข้อมูล



รูปที่ 4.7 หน้าจอโปรแกรมแสดงผลลัพธ์การแปลงแผนภาพกระแสข้อมูล

นอกจากการให้เครื่องมือซอฟต์แวร์ทำการแปลงแผนภาพกระแสข้อมูลโดยอัตโนมัติแล้ว ผู้ใช้สามารถที่จะกำหนดส่วนของการแปลง (ส่วนนำเข้าข้อมูล ส่วนแสดงผลลัพธ์ และ/หรือส่วนศูนย์กลางทรานแซกชั่น) ให้กับเครื่องมือซอฟต์แวร์ได้ด้วย ซึ่งคุณสมบัตินี้จะใช้ในกรณีที่ผู้ใช้ทราบส่วนของการแปลงที่แน่นอนเพื่อจำกัดจำนวนของผังโครงสร้างของโปรแกรมที่เป็นผลลัพธ์ให้น้อยลง ผู้ใช้สามารถเลือกระบุส่วนของการ

แปลงได้โดยการเลือกเมนูคำสั่ง "Mode"->"Semi-Transform" ซึ่งจะปรากฏหน้าจอดังรูปที่ 4.8

ในการกำหนดส่วนของการแปลง จะเริ่มต้นจากการเลือกรูปแบบการแปลงจากปุ่มคำสั่งที่อยู่ในแถบเครื่องมือกำหนดส่วนการแปลง ซึ่งประกอบด้วยปุ่มคำสั่งต่างๆ ดังนี้

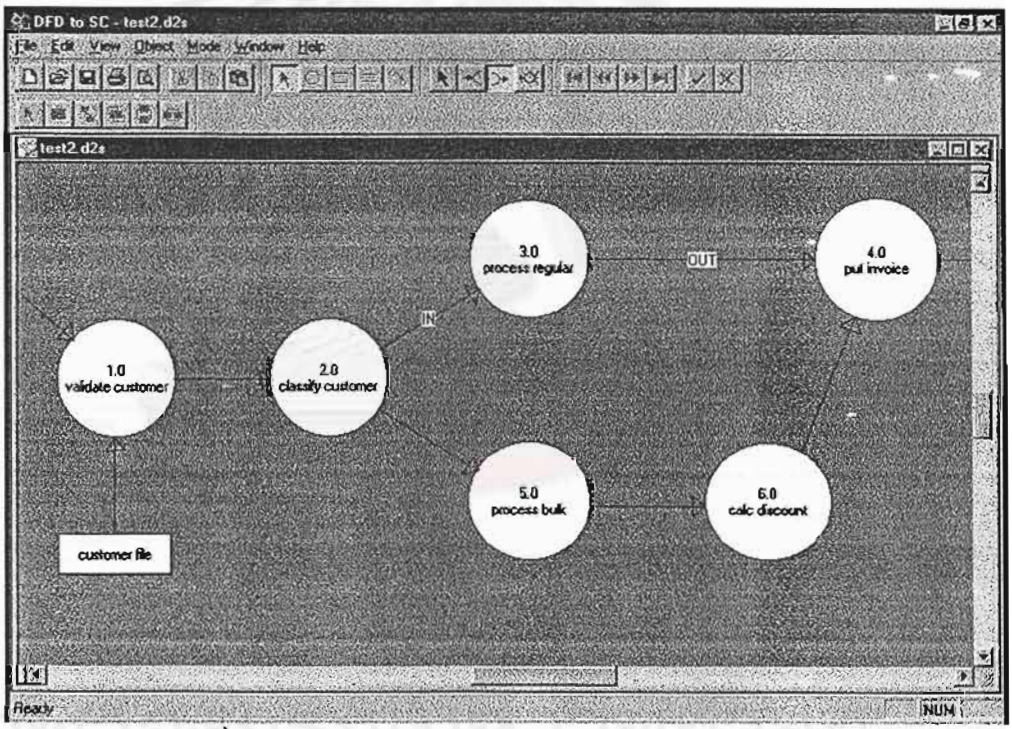
เลือกเครื่องมือการแปลงส่วนนำเข้าข้อมูล

เลือกเครื่องมือการแปลงส่วนแสดงผลลัพธ์

เลือกเครื่องการแปลงส่วนศูนย์กลางทรานแซกชั่น

ยกเลิกการเลือกเครื่องมือการแปลง

หลังจากที่ผู้ใช้กำหนดส่วนการแปลงตามที่ต้องการแล้ว ให้เลือกเมนูคำสั่ง "Mode"->"Auto-Transform" เพื่อให้เครื่องมือซอฟต์แวร์ทำการแปลงแผนภาพกระแสข้อมูลตามส่วนการแปลงที่กำหนด ซึ่งจะปรากฏหน้าจอแสดงผลลัพธ์การแปลงเช่นเดียวกับการแปลงแบบอัตโนมัติ



รูปที่ 4.8 หน้าจอโปรแกรมการกำหนดส่วนของการแปลง

4.2.3 ขั้นตอนการแก้ไขผังโครงสร้างของโปรแกรม

เมื่อผู้ใช้เรียกแฟ้มข้อมูลของผังโครงสร้างของโปรแกรมที่ได้จากการแปลงขึ้นมาทำงาน จะปรากฏหน้าจอโปรแกรมดังแสดงในรูปที่ 4.9 ซึ่งเป็นหน้าจอสำหรับการวาดผังโครงสร้างของโปรแกรม ผู้ใช้สามารถวาดผังโครงสร้างของโปรแกรมโดยเริ่มต้นจากการ

เลือกองค์ประกอบที่ต้องการวาดจากปุ่มคำสั่งที่อยู่ในแถบเครื่องมือการวาดซึ่งประกอบด้วยปุ่มต่างๆ ดังนี้

 เลือกเครื่องมือวาดมอดูล

 เลือกเครื่องมือวาดการเชื่อมโยงมอดูล

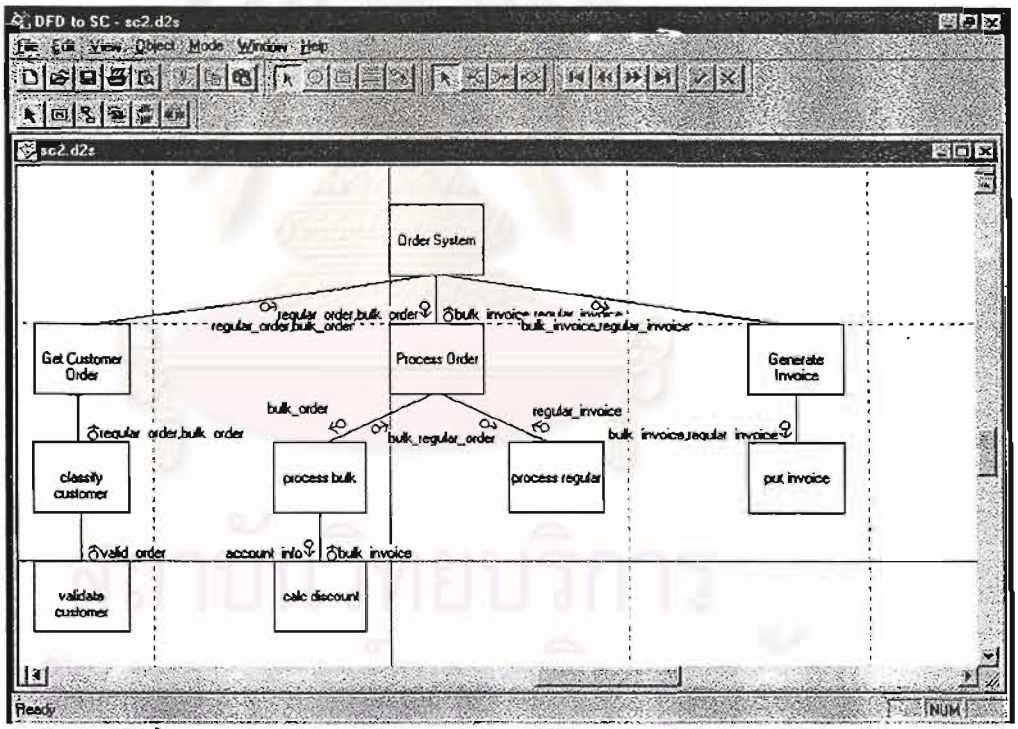
 ยกเลิกการเลือกเครื่องมือการวาด

 เลือกเครื่องมือการขุบรวมมอดูล

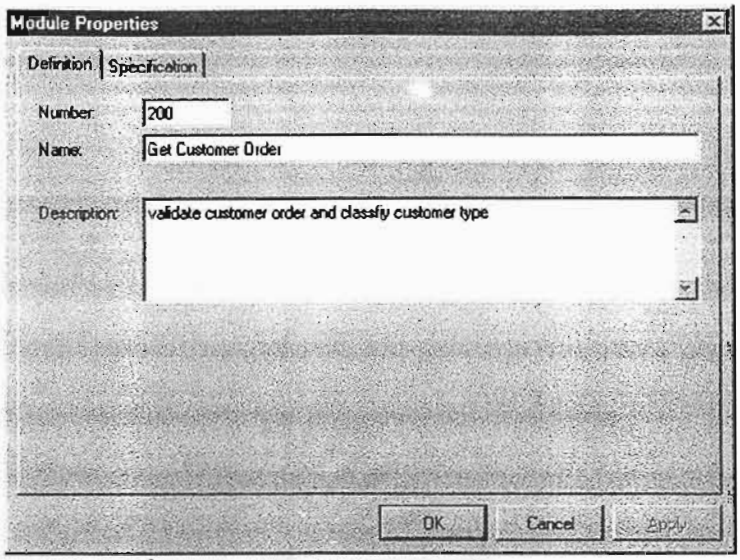
 เลือกเครื่องมือการแตกมอดูลในแนวตั้ง

 เลือกเครื่องมือการแตกมอดูลในแนวนอน

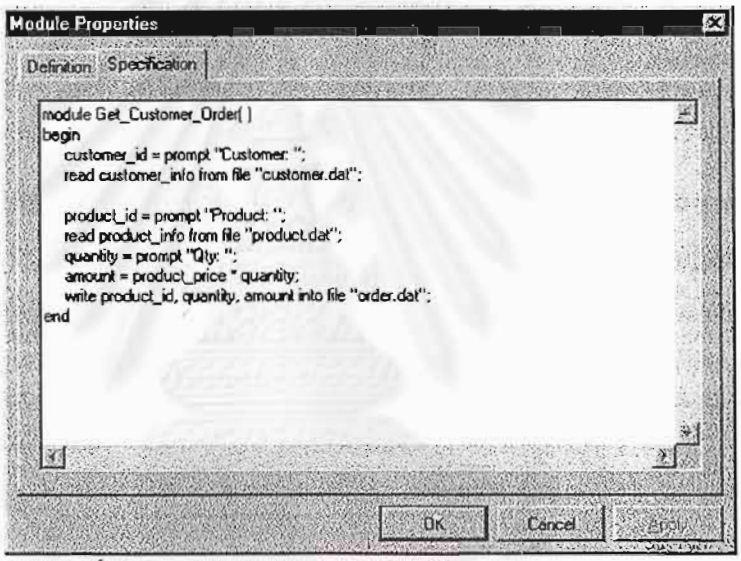
เมื่อผู้ใช้ทำการวาดองค์ประกอบของผังโครงสร้างของโปรแกรมแล้ว สามารถกำหนดรายละเอียดขององค์ประกอบโดยการดับเบิ้ลคลิกเมาส์ที่ตัวองค์ประกอบ ซึ่งจะปรากฏหน้าต่างบันทึกรายละเอียดขององค์ประกอบดังแสดงในรูปที่ 4.10- รูปที่ 4.12



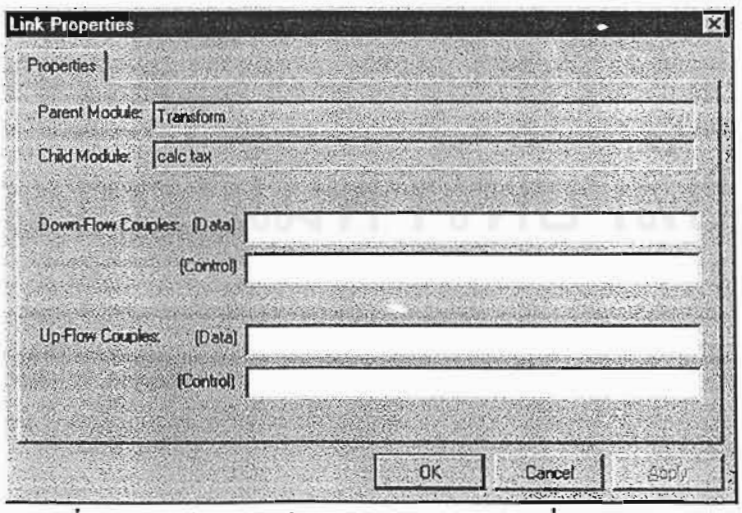
รูปที่ 4.9 หน้าจอโปรแกรมสำหรับการวาดผังโครงสร้างของโปรแกรม



รูปที่ 4.10 หน้าจอบันทึกรายละเอียดของมอดูล



รูปที่ 4.11 หน้าจอบันทึกคำอธิบายการทำงานของมอดูล



รูปที่ 4.12 หน้าจอบันทึกรายละเอียดของการเชื่อมโยงมอดูล

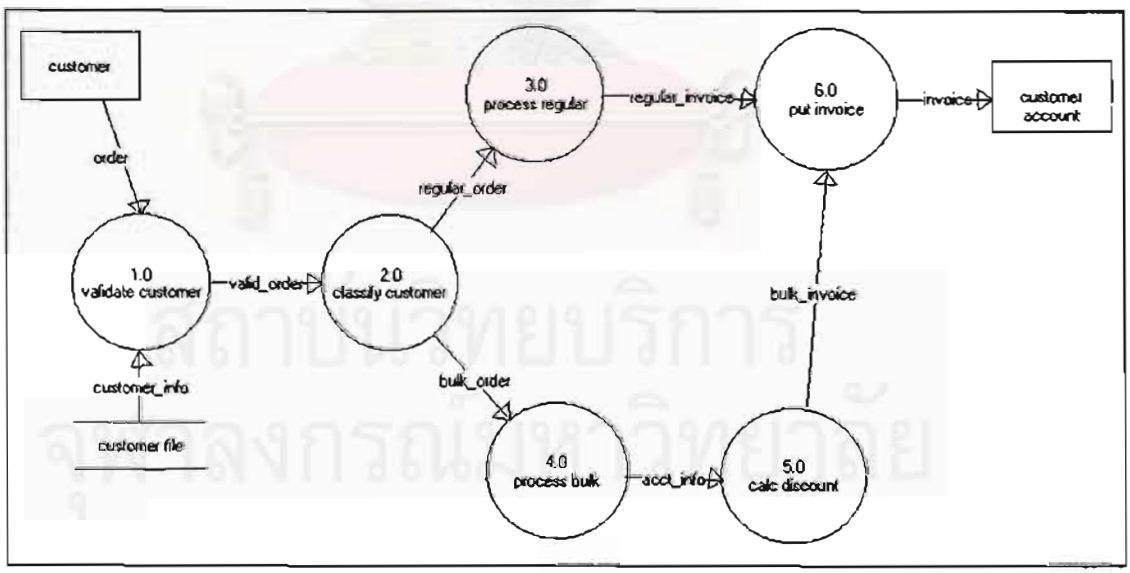
บทที่ 5

การทดสอบการทำงานของเครื่องมือซอฟต์แวร์

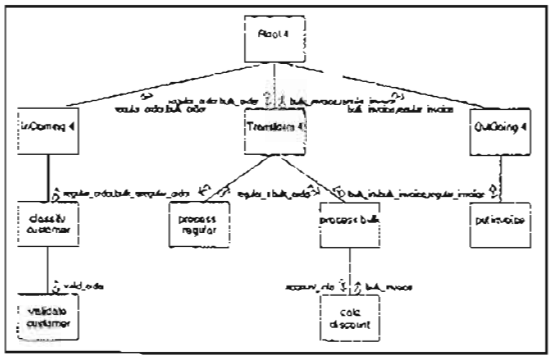
ผู้วิจัยได้นำเครื่องมือซอฟต์แวร์ที่พัฒนาเสร็จแล้วมาทำการทดสอบแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรมด้วยวิธีวิเคราะห์แบบการแปลงและวิธีวิเคราะห์แบบทรานแซกชัน โดยใช้ตัวอย่างของแผนภาพกระแสข้อมูลจากหนังสือที่อ้างอิง[1],[4],[5],[6],[7],[8] ซึ่งผลการทดสอบพบว่าเครื่องมือซอฟต์แวร์ที่พัฒนาสามารถทำการแปลงแผนภาพกระแสข้อมูลได้ รวมถึงสามารถวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลได้ในกรณีที่เขียนคำอธิบายการทำงานของมอดูลในรูปแบบของภาษาในการออกแบบโปรแกรมไว้

5.1 การแปลงแผนภาพกระแสข้อมูล

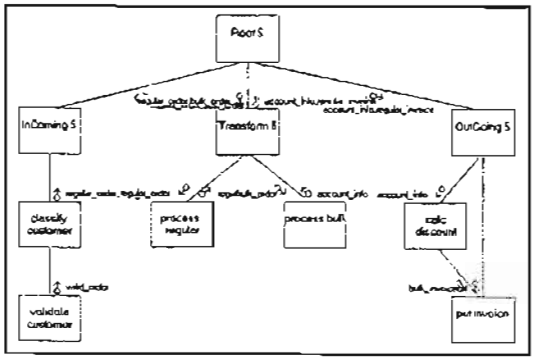
ตัวอย่างแผนภาพกระแสข้อมูลในรูปที่ 5.1 เป็นแผนภาพกระแสข้อมูลสำหรับโปรแกรมสั่งซื้อสินค้า เมื่อทำการแปลงแบบอัตโนมัติด้วยเครื่องมือที่พัฒนาจะได้ผังโครงสร้างของโปรแกรมให้เลือก 8 ผังโครงสร้างดังแสดงในรูปที่ 5.2 ซึ่งเป็นการแปลงด้วยวิธีการวิเคราะห์แบบแปลง



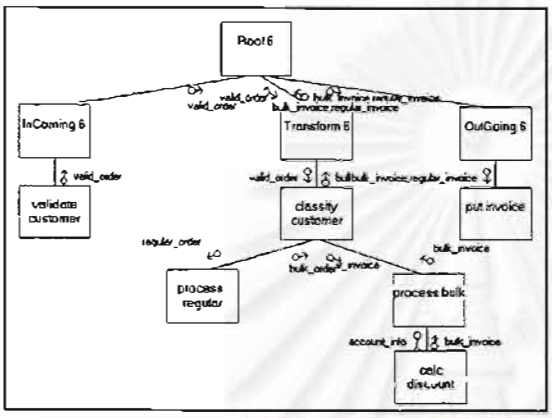
รูปที่ 5.1 แผนภาพกระแสข้อมูลสำหรับโปรแกรมสั่งซื้อสินค้า



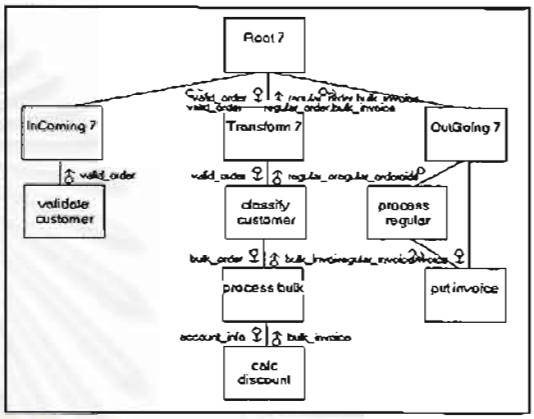
(ก)



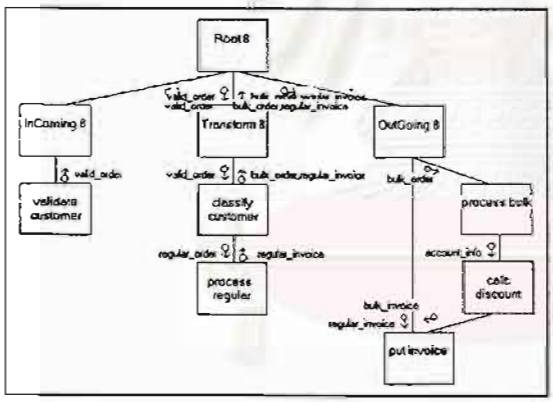
(ข)



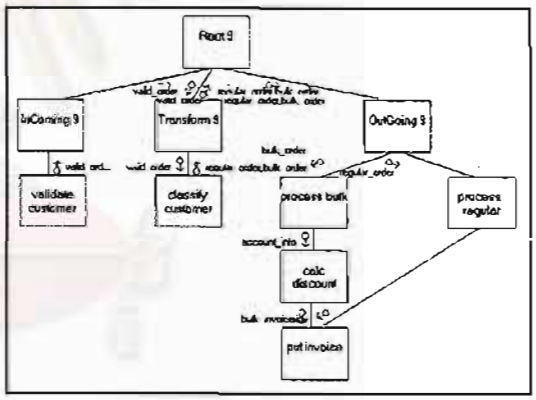
(ค)



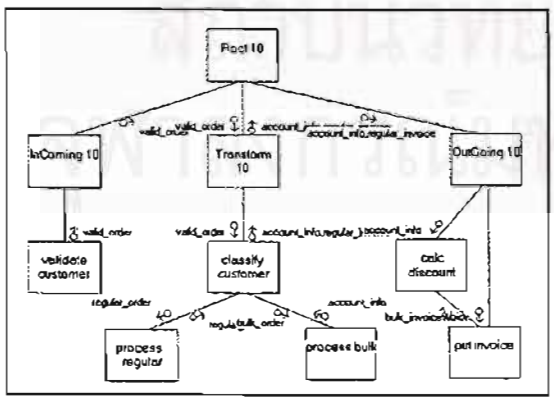
(ง)



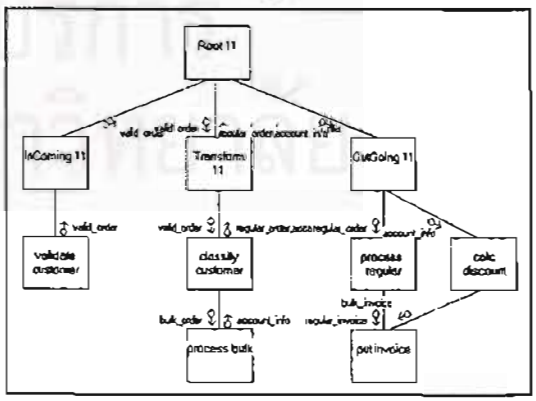
(จ)



(ฉ)



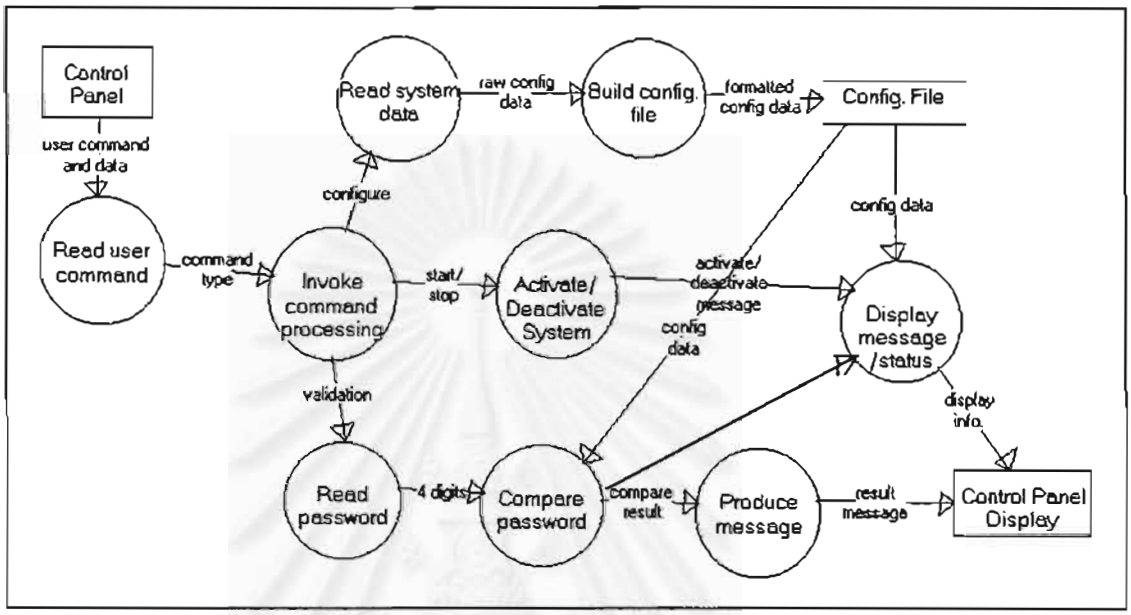
(ช)



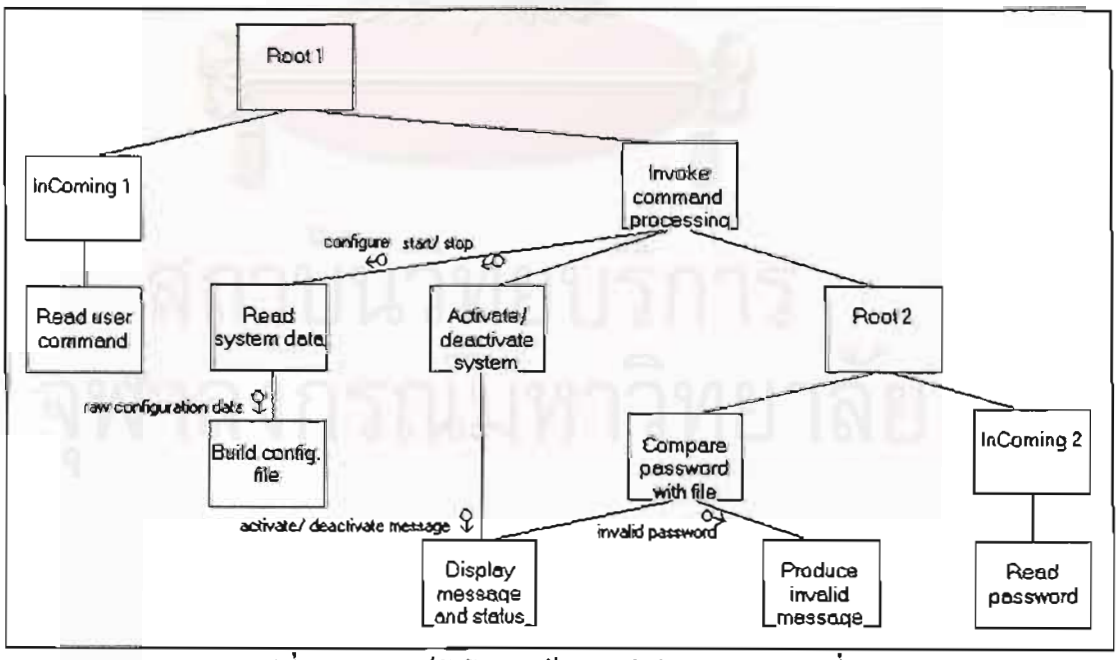
(ซ)

รูปที่ 5.2 ผลลัพธ์ผังโครงสร้างของโปรแกรมจากรูปที่ 5.1

สำหรับแผนภาพกระแสข้อมูลในรูปที่ 5.3 เป็นตัวอย่างของแผนภาพกระแสข้อมูลของโปรแกรมควบคุมเครื่องมืออิเล็คทรอนิคส์ประจำบ้าน เมื่อทำการแปลงแบบอัตโนมัติจะได้ผังโครงสร้างของโปรแกรมที่เป็นคำตอบซึ่งเป็นการแปลงด้วยวิธีวิเคราะห์แบบทรานแซกชันดังแสดงในรูปที่ 5.4



รูปที่ 5.3 แผนภาพกระแสข้อมูลของโปรแกรมควบคุมเครื่องมืออิเล็คทรอนิคส์ประจำบ้าน



รูปที่ 5.4 ผลลัพธ์ผังโครงสร้างของโปรแกรมจากรูปที่ 5.3

5.2 ผลการทดลองวัดค่าสัมพันธ์ต่อกันระหว่างมอดูล

ผู้วิจัยได้ทดลองกำหนดคำอธิบายการทำงานของมอดูลขึ้นเพื่อให้เครื่องมือซอฟต์แวร์ที่พัฒนาทำการวัดค่าความสัมพันธ์ระหว่างมอดูลแต่ละประเภทดังแสดงเป็นตัวอย่างด้านล่าง โดยการระบุคำอธิบายการทำงานของมอดูลผ่านเครื่องมือซอฟต์แวร์ที่พัฒนาในหน้าต่างบันทึกคำอธิบายการทำงานของมอดูล ซึ่งเครื่องมือซอฟต์แวร์สามารถหาค่าความสัมพันธ์ต่อกันระหว่างมอดูลได้ตามกฎในการตรวจสอบประเภทและลักษณะการใช้งานพารามิเตอร์ที่ออกแบบไว้

1) ความสัมพันธ์ระหว่างมอดูลแบบอิสระต่อกัน

```
module x0 ( )
begin
    integer i;
    i = 10 * 300;
end
```

```
module y0 ( )
begin
    integer k, n;
    k = 0;
    if (k > 10) then
        n = n + (k * 10);
    endif;
end
```

มอดูล X และมอดูล Y ไม่มีการเรียกทำงานระหว่างกัน และไม่มีการใช้ตัวแปรแบบครอบคลุมหรือแฟ้มข้อมูลภายนอกร่วมกัน ดังนั้นมอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบอิสระต่อกัน

2) ความสัมพันธ์ระหว่างมอดูลแบบเรียกกัน

```
module x1 ( )
begin
    integer i;
    i = 10 * 300;
    call y1 ( );
end
```

```
module y1 (integer i)
begin
    integer k, n;
    k = 0;
    if (k > 10) then
        n = n + (k * 10);
    endif;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานแต่ไม่มีการส่งค่าพารามิเตอร์ไปให้ และมอดูลทั้งสองไม่มีการใช้ตัวแปรแบบครอบคลุมหรือแฟ้มข้อมูลภายนอกร่วมกัน ดังนั้นมอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบเรียกกัน

3) ความสัมพันธ์ระหว่างมอดูลแบบข้อมูลตัวแปรเชิงเดี่ยว

```
module x2 ( )
begin
    integer i;
    i = 10 * 300;
    call y2 ( i );
    exit;
end
```

```
module y2 (integer i)
begin
    integer k, n;
    k = (i * 10) / 2;
    if (k > 10) then
        n = n + (k * 10);
    endif;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ i ไปให้ ซึ่งในมอดูล Y จะนำค่าพารามิเตอร์ i ไปใช้คำนวณค่าให้ตัวแปร k ดังนั้นมอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบข้อมูลตัวแปรเชิงเดี่ยว

4) ความสัมพันธ์ระหว่างมอดูลแบบข้อมูลตัวแปรเชิงกลุ่ม

```
module x3 ( )
begin
    define type student as
        char name, integer birth_year;
    student  std;
    integer i;
    std.name = "Good Man";
    std.birth_year = 2515;
    i = 10 * 300;
    call y3 ( std, i );
    exit;
end
```

```
module y3 (student new_std, integer i)
begin
    integer  k, n, eng_year;
    k = ( i * 10 ) / 2;
    if (k > 10) then
        n = n +  (k * 10);
    endif;
    eng_year = new_std.birth_year – 543;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ std และ i ไปให้ โดยที่พารามิเตอร์ std มีชนิดข้อมูลแบบโครงสร้าง ในมอดูล Y จะนำค่าพารามิเตอร์ std ไปใช้คำนวณค่าให้ตัวแปร eng_year ซึ่งเป็นความสัมพันธ์แบบข้อมูลตัวแปรเชิงกลุ่ม และนำค่าพารามิเตอร์ i ไปใช้คำนวณค่าให้ตัวแปร k ซึ่งเป็นความสัมพันธ์แบบข้อมูลตัวแปรเชิงเดี่ยว แต่เนื่องจากค่าระดับความสัมพันธ์แบบข้อมูลตัวแปรเชิงกลุ่มมีค่าสูงกว่า ดังนั้นจึงถือว่ามอดูล X และ Y มีความสัมพันธ์ระหว่างมอดูลแบบข้อมูลตัวแปรเชิงกลุ่ม

5) ความสัมพันธ์ระหว่างมอดูลแบบควบคุมตัวแปรเชิงเดี่ยว

```
module x4 ( )
begin
    integer  i;
    i = 10 * 300;
    call y4 ( i );
    exit;
end
```

```
module y4 (integer i)
begin
    integer  k;
    if (i > 10) then
        call z1( );
    else
        k = i / 2;
        callz2 ( );
    endif;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ i ไปให้ ซึ่งมอดูล Y จะนำค่าพารามิเตอร์ i ไปใช้เป็นเงื่อนไขในคำสั่ง if ซึ่งเป็นความสัมพันธ์แบบควบคุมตัวแปรเชิงเดี่ยว และนำค่าพารามิเตอร์ i ไปใช้คำนวณค่าให้ตัวแปร k ซึ่งเป็นความสัมพันธ์แบบข้อมูลตัวแปรเชิงเดี่ยว แต่เนื่องจากค่าระดับความสัมพันธ์แบบควบคุมตัวแปรเชิงเดี่ยวมีค่าสูงกว่า ดังนั้นจึงถือว่ามอดูล X และ Y มีความสัมพันธ์ระหว่างมอดูลแบบควบคุมตัวแปรเชิงเดี่ยว

6) ความสัมพันธ์ระหว่างมอดูลแบบควบคุมตัวแปรเชิงกลุ่ม

```
module x5 ( )
begin
    define type student as
        char name, integer birth_year;
    student  std;
    integer i;
    std.name = "Good Man";
    std.birth_year = 2515;
    i = 10 * 300;
    call y5 ( std, i );
    exit;
end
```

```
module y5 (student new_std, integer i)
begin
    integer k, eng_year;
    k = (i * 10) / 2;
    if (new_std.birth_year < 2510) then
        call z1( );
    endif;
    eng_year = new_std.birth_year – 543;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ std และ i ไปให้ ในมอดูล Y จะนำค่าพารามิเตอร์ std ซึ่งเป็นชนิดข้อมูลโครงสร้างไปใช้เป็นเงื่อนไขในคำสั่ง if ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบควบคุมตัวแปรเชิงเดี่ยว

7) ความสัมพันธ์ระหว่างมอดูลแบบข้อมูลกึ่งควบคุมตัวแปรเชิงเดี่ยว

```
module x6 ( )
begin
    integer  i;
    i = 10 * 300;
    call y6 ( i );
    exit;
end
```

```
module y6 (integer i)
begin
    integer  j,k;
    k = (i * 2) +  5;
    if (k > 10) then
        call z1( );
    else
        j = i / 2;
        callz2 ( );
    endif;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ i ไปให้ มอดูล Y จะนำค่าพารามิเตอร์ i ไปใช้คำนวณค่าให้ตัวแปร k จากนั้นมีการนำตัวแปร k ไปใช้เป็นเงื่อนไขในคำสั่ง if ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบข้อมูลกึ่งควบคุมตัวแปรเชิงเดี่ยว

8) ความสัมพันธ์ระหว่างมอดูลแบบข้อมูลกึ่งควบคุมตัวแปรเชิงกลุ่ม

```
module x7 ( )
begin
    define type student as
        char name, integer birth_year;
    student  std;
    integer i;
    std.name = "Good Man";
    std.birth_year = 2515;
    i = 10 * 300;
    call y7 ( std, i );
    exit;
end
```

```
module y7 (student new_std, integer i)
begin
    integer k, eng_year;
    k = (i * 2) + 5;
    eng_year = new_std.birth_year – 543;
    if (eng_year < 1967) then
        call z1( );
    endif;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ std และi ไปให้ มอดูล Y จะนำค่าพารามิเตอร์ std ไปใช้คำนวณค่าให้ตัวแปร eng_year จากนั้นมีการนำตัวแปร eng_year ไปใช้เป็นเงื่อนไขในคำสั่ง if ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบข้อมูลกึ่งควบคุมตัวแปรเชิงกลุ่ม

9) ความสัมพันธ์ระหว่างมอดูลแบบภายนอก

```
module x8 ( )
begin
    define type student as
        char name, integer birth_year;
    student  std;

    std.name = prompt "What name: ";
    std.birth_year = prompt "What year: ";
    write std into file "student.txt";
    call y8 ( );
    exit;
end
```

```
module y8 ( )
begin
    student std;
    read std from file "student.txt";
    print std.name, std.birth_year;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยไม่มีการส่งค่าพารามิเตอร์ไปให้ แต่ทั้งสองมอดูลมีการใช้แฟ้มข้อมูลภายนอก "student.txt" ร่วมกัน ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบภายนอก

10) ความสัมพันธ์ระหว่างมอดูลแบบไม่ใช้เฉพาะที่

```
module x9 ( )
begin
    shared integer i;
    i = 10 * 300;
    call y9 ( );
    exit;
end
```

```
module y9 ( )
begin
    shared integer i;
    integer  k, n;
    k = ( i * 100 ) / 2;
    n = 0;
    while (k > 10) do
        n = n +  (k * 10);
    endwhile;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยไม่มีการส่งค่าพารามิเตอร์ไปให้ แต่ทั้งสองมอดูลมีการใช้ตัวแปรไม่ใช้เฉพาะที่ i ร่วมกัน ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบไม่ใช้เฉพาะที่

11) ความสัมพันธ์ระหว่างมอดูลแบบครอบคลุม

```
module x10 ( )
begin
    global integer i;
    i = 10 * 300;
    call y10 ( );
    exit;
end
```

```
module y10 ( )
begin
    global integer i;
    integer  k, n;
    k = ( i * 100 ) / 2;
    n = 0;
    while (k > 10) do
        n = n +  (k * 10);
    endwhile;
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยไม่มีการส่งค่าพารามิเตอร์ไปให้ แต่ทั้งสองมอดูลมีการใช้ตัวแปรแบบครอบคลุม i ร่วมกัน ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบครอบคลุม

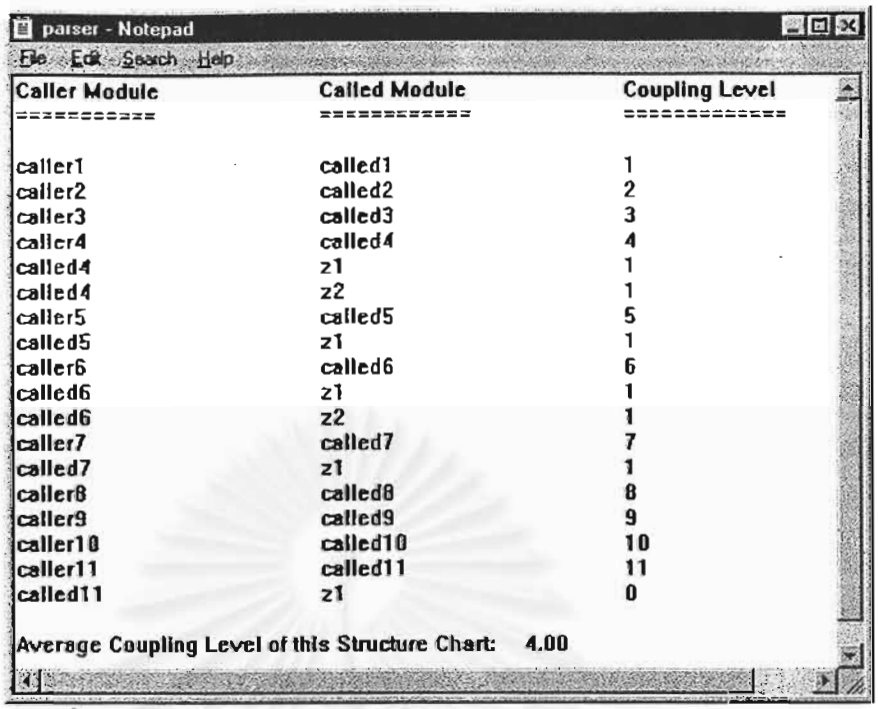
12) ความสัมพันธ์ระหว่างมอดูลแบบส่งผ่าน

```
module x11 ( )
begin
    integer i;
    i = 10 * 300;
    call y11 ( i );
    exit;
end
```

```
module y11 (integer i)
begin
    integer  k, n;
    k = 0;
    while (k > 10) do
        k = k + 1;
        n = n +  (k * 10);
    endwhile;
    call z1( )
    exit;
end
```

มอดูล X เรียกมอดูล Y ขึ้นมาทำงานโดยส่งค่าพารามิเตอร์ i ไปให้ ในมอดูล Y ไม่มีการใช้งานพารามิเตอร์ i แต่จะส่งพารามิเตอร์ i ไปยังมอดูล Z1 ดังนั้น มอดูล X และ Y จึงมีความสัมพันธ์ระหว่างมอดูลแบบส่งผ่าน

ผู้วิจัยได้ทำการสร้างผังโครงสร้างของโปรแกรมซึ่งประกอบด้วยมอดูล X0 ถึง X11 Y0 ถึง Y11 Z1 และ Z1 โดยระบุข้อกำหนดการทำงานของมอดูลต่างๆ ตามตัวอย่างข้างต้น ซึ่งเมื่อสั่งให้โปรแกรมทำการวัดค่าระดับความสัมพันธ์ต่อกันระหว่างมอดูลของผังโครงสร้างของโปรแกรมจะปรากฏหน้าต่างแสดงผลลัพธ์จากการวัดค่าระดับความสัมพันธ์ดังในรูปที่ 5.5 ซึ่งค่าระดับความสัมพันธ์นี้ได้มีการกำหนดค่าไว้แล้วดังกล่าวมาแล้วในหน้าที่ 17 นอกจากนี้ ในบรรทัดสุดท้ายของผลลัพธ์จะแสดงค่าเฉลี่ยของระดับความสัมพันธ์ต่อกันระหว่างมอดูลของผังโครงสร้างของโปรแกรมให้ทราบด้วย


parser - Notepad

File Edit Search Help

Caller Module	Called Module	Coupling Level
caller1	called1	1
caller2	called2	2
caller3	called3	3
caller4	called4	4
called4	z1	1
called4	z2	1
caller5	called5	5
called5	z1	1
caller6	called6	6
called6	z1	1
called6	z2	1
caller7	called7	7
called7	z1	1
caller8	called8	8
caller9	called9	9
caller10	called10	10
caller11	called11	11
called11	z1	0

Average Coupling Level of this Structure Chart: 4.00

รูปที่ 5.5 หน้าต่างแสดงผลลัพธ์การวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูล

5.3 ข้อจำกัดของเครื่องมือซอฟต์แวร์

5.3.1 เครื่องมือซอฟต์แวร์นี้สามารถรองรับจำนวนของกระบวนการทำงานได้โดยไม่มีข้อจำกัดจำนวน เนื่องจากใช้การจัดเก็บข้อมูลภายในแบบรายการ (List) ซึ่งสามารถเก็บข้อมูลได้ไม่จำกัด หากหน่วยความจำของเครื่องคอมพิวเตอร์ยังสามารถจัดสรรมาให้ได้

5.3.2 เครื่องมือซอฟต์แวร์นี้ไม่สามารถทำการแปลงแผนภาพกระแสข้อมูลที่มีลักษณะการไหลแบบวนรอบ (Cyclic Flow) เนื่องจากตามวิธีการวิเคราะห์แบบการแปลงนั้น แผนภาพกระแสข้อมูลจะถูกแบ่งออกเป็น 3 ส่วนซึ่งเป็นการทำงานแบบวิ่งไปข้างหน้า ซึ่งในกรณีที่ผู้ใช้นำแผนภาพกระแสข้อมูลที่มีลักษณะการไหลแบบวนรอบมาทำการแปลง เครื่องมือซอฟต์แวร์จะแจ้งข้อความผิดพลาดให้ทราบว่าไม่สามารถทำการแปลงแผนภาพกระแสข้อมูลที่มีลักษณะดังกล่าวได้

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

จากการวิจัยในครั้งนี้ทำให้ได้เครื่องมือซอฟต์แวร์ในการแปลงแผนภาพกระแสข้อมูลให้เป็นผังโครงสร้างของโปรแกรมด้วยวิธีวิเคราะห์แบบการแปลงและวิธีวิเคราะห์แบบทรานแซกชันได้ โดยการใช้งานเครื่องมือซอฟต์แวร์จะเริ่มจากการนำเข้าแฟ้มข้อมูลของแผนภาพกระแสข้อมูลจากโปรแกรม Power Designer 6.1 หรือผู้ใช้อาจวาดแผนภาพกระแสข้อมูลใหม่โดยใช้เครื่องมือซอฟต์แวร์นี้ได้เช่นกัน จากนั้นผู้ใช้สามารถสั่งให้เครื่องมือซอฟต์แวร์ทำการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรมซึ่งทำงานได้ทั้งแบบอัตโนมัติและแบบกึ่งอัตโนมัติได้ สำหรับในการแปลงแบบกึ่งอัตโนมัตินั้น ผู้ใช้สามารถระบุเงื่อนไขในการแปลงโดยระบุส่วนของศูนย์กลางการแปลงหรือศูนย์กลางทรานแซกชันที่ต้องการ จากนั้นเครื่องมือซอฟต์แวร์จะแสดงผังโครงสร้างของโปรแกรมในรูปแบบต่างๆ ที่ได้มาให้ผู้ใช้ทำการเลือกเพื่อสั่งจัดเก็บ ผู้ใช้สามารถนำผังโครงสร้างของโปรแกรมที่จัดเก็บไว้มาทำการแก้ไขได้ เช่น การยุบรวมหรือแตกมอดูล เป็นต้น ซึ่งเครื่องมือซอฟต์แวร์ที่พัฒนาขึ้นนี้จะช่วยประหยัดเวลาและค่าใช้จ่ายของการพัฒนาโปรแกรมได้ อย่างไรก็ตาม ข้อจำกัดของเครื่องมือซอฟต์แวร์ที่พัฒนา คือ ไม่สามารถแปลงแผนภาพกระแสข้อมูลที่มีลักษณะการไหลแบบวนรอบได้

นอกจากนี้ เครื่องมือซอฟต์แวร์ที่พัฒนายังสามารถวัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของผังโครงสร้างของโปรแกรมได้ในกรณีที่ผู้ใช้เขียนระบุข้อกำหนดการทำงานของมอดูลต่างๆ ตามรูปแบบของภาษาในการออกแบบโปรแกรมที่ได้ออกแบบไว้ โดยใช้วิธีการวัดค่าระดับความสัมพันธ์ต่อกันระหว่างมอดูลของออฟฟัต นอกจากนี้ ผู้ใช้สามารถเพิ่มเติม ลบ หรือแก้ไขกฎไวยากรณ์ของภาษาในการออกแบบโปรแกรมได้ในรูปแบบของคำอธิบายแบบบีเอ็นเอฟ

6.2 ปัญหาและอุปสรรค

6.2.1 ใช้เวลาในการสืบหารูปแบบการจัดเก็บแฟ้มข้อมูลของแผนภาพกระแสข้อมูลที่ได้จากโปรแกรมซอฟต์แวร์อื่นเพื่อนำมาใช้เป็นแฟ้มข้อมูลนำเข้าของเครื่องมือซอฟต์แวร์ที่พัฒนา เนื่องจากโปรแกรมซอฟต์แวร์อื่นจะเก็บแฟ้มข้อมูลในรูปแบบเลขฐานสอง (Binary Format) และผู้ผลิตโปรแกรมซอฟต์แวร์ไม่เผยแพร่รูปแบบการจัดเก็บแฟ้มข้อมูลให้บุคคลภายนอกรู้

6.2.2 ใช้เวลาในการศึกษาวิธีการเขียนโปรแกรมสำหรับวิชวลซีพลัสพลัส โดยเฉพาะในเรื่องวิธีการติดต่อกับผู้ใช้ในการวาดแผนภาพกระแสข้อมูลและผังโครงสร้างของโปรแกรม

6.3 ข้อเสนอแนะเพื่อการวิจัยในอนาคต

6.3.1 เครื่องมือซอฟต์แวร์ที่พัฒนายังขาดคุณลักษณะบางประการทางด้านกราฟิก เช่น การกำหนดความหนาและสีของเส้น การกำหนดฟอนต์ เป็นต้น ซึ่งถ้าได้มีการพัฒนาในส่วนนี้เพิ่มเติมจะช่วยให้การนำไปใช้งานมีความสะดวกและเหมาะสมมากยิ่งขึ้น

6.3.2 เครื่องมือซอฟต์แวร์ที่พัฒนานี้สามารถตรวจสอบความถูกต้องตามกฎไวยากรณ์ของภาษาในการออกแบบโปรแกรมได้ หากสามารถพัฒนาไปสู่การสร้างรหัสต้นฉบับสำหรับภาษาในการเขียนโปรแกรม จะช่วยให้กระบวนการพัฒนาโปรแกรมเป็นไปอย่างรวดเร็วมากยิ่งขึ้น

6.3.3 เครื่องมือซอฟต์แวร์ที่พัฒนาไม่สามารถแปลงแผนภาพกระแสข้อมูลที่มีลักษณะการไหลแบบวนรอบ ซึ่งถ้าได้มีการพัฒนาในส่วนนี้เพิ่มเติมจะช่วยให้การนำไปใช้งานได้อย่างกว้างขวางยิ่งขึ้น

รายการอ้างอิง

1) Roger S. Pressman. Software Engineering : A Practitioner's Approach. 3rd Edition. NY: McGraw-Hill, 1992.

2) A. Jefferson Offutt. A Software Metric System for Module Coupling. URL: http://www.cs.cleson.edu.

3) Alfred V.Aho, Ravi Sethi, and Jeffery D. Ullman. Compilters: Principles, Techniques, and Tools. MA: Addison-Wesley Publishing, 1986.

4) Victor Weinberg. Structured Analysis. NY: Yourdon Press, 1980.

5) James Martin, Carma McClure. Structured Techniques: The Basis for CASE. NJ: Prentice Hall, 1988.

6) Meilir Page-Jones. The Practical Guide to Structured System Design. 2nd Edition. NJ: Prentice-Hall, 1988.

7) Sylvia Goldsmith. A Practical Guide to Real-Time Systems Development. NJ: Prentice Hall, 1993.

8) Ian Sommerville. Software Engineering. 5th Edition. MA: Addison-Wesley Publishing, 1996.

9) Minyi Guo. Automatic Transformation from Data Flow Diagram to Structure Chart. ACM SIGSOFT 1997.

ภาคผนวก

ภาคผนวก ก.

การออกแบบคลาสของวัตถุ

ชื่อคลาส: CD2SApp คลาสแม่ : CWinApp

คำอธิบาย: เป็นคลาสที่แสดงถึงตัวโปรแกรม

ตารางที่ ก-1 รายการคุณลักษณะของคลาส CD2SApp

ชื่อคุณลักษณะ	คำอธิบาย
m_pMainWnd	เลขที่อ้างอิงของวินโดว์ที่โปรแกรมสร้าง

ตารางที่ ก-2 รายการวิธีทำงานของคลาส CD2SApp

ชื่อวิธีทำงาน	คำอธิบาย
InitInstance()	สร้างหน้าจอวินโดว์และแสดงวินโดว์แก่ผู้ใช้

ชื่อคลาส: CD2SDoc คลาสแม่ : COleDocument

คำอธิบาย: เป็นคลาสที่ใช้เก็บข้อมูลของแผนภาพที่วาด (ทั้งแผนภาพกระแสข้อมูลและผังโครงสร้างของโปรแกรม) โดยในคลาส CD2SDoc นั้น จะบรรจุโครงสร้างข้อมูลแบบรายการ (List Data Structure) ของคลาส CD2SObj ซึ่งใช้แสดงถึงองค์ประกอบต่างๆ ของแผนภาพ

ตารางที่ ก-3 รายการคุณลักษณะของคลาส CD2SDoc

ชื่อคุณลักษณะ	คำอธิบาย
m_objects	ที่เก็บข้อมูลแบบรายการขององค์ประกอบของแผนภาพหรือผังโครงสร้าง (เช่น เอนทิตี้ภายนอก กระบวนการทำงาน มอดูล) ที่ผู้ใช้วาด
m_object2	ที่เก็บข้อมูลแบบรายการขององค์ประกอบของผังโครงสร้าง (มอดูล และเส้นเชื่อมโยง) ที่เป็นผลลัพธ์จากการแปลง
m_curView	โหมดวิวการทำงานปัจจุบัน ได้แก่ โหมดวาดแผนภาพ โหมดการแปลงแบบกึ่งอัตโนมัติ โหมดการแปลงแบบอัตโนมัติ และโหมดการวาดผังโครงสร้าง
m_runnumber	เลขที่อ้างอิงสุดท้ายที่กำหนดให้กับองค์ประกอบของแผนภาพไปแล้ว (นำไปใช้เพื่อเป็นเลขอ้างอิงขององค์ประกอบ)
m_pFrameWnd	เลขที่อ้างอิงของวินโดว์ที่เอกสารเชื่อมโยงอยู่

ตารางที่ ก-4 รายการวิธีทำงานของคลาส CD2SDoc

ชื่อวิธีทำงาน	คำอธิบาย
OpenFile()	เปิดแฟ้มข้อมูลแผนภาพกระแสข้อมูลหรือผังโครงสร้างของโปรแกรมขึ้นมาทำงาน
SaveFile()	จัดเก็บแผนภาพกระแสข้อมูลหรือผังโครงสร้างของโปรแกรมลงยังแฟ้มข้อมูล
Draw()	วาดองค์ประกอบที่เก็บไว้ไปแสดงผลยังวิวแสดงผล
Add()	เพิ่มองค์ประกอบใหม่เข้าไปยังที่เก็บข้อมูลองค์ประกอบ
Remove()	ลบองค์ประกอบที่ระบุออกจากที่เก็บข้อมูลองค์ประกอบ
ObjectAt()	หาองค์ประกอบในที่เก็บข้อมูลองค์ประกอบที่อยู่ ณ ตำแหน่งที่ระบุ
GetRunNumber()	อ่าน m_runnumber มาเพิ่มค่าอีกหนึ่ง
GenDocument()	สร้างเอกสารของแผนภาพกระแสข้อมูล หรือผังโครงสร้างของโปรแกรม
OnDrawDfd()	เปลี่ยนโหมดการทำงานปัจจุบันเป็นโหมดการวาดแผนภาพกระแสข้อมูล
OnSemiTransform()	เปลี่ยนโหมดการทำงานปัจจุบันเป็นโหมดการแปลงแบบกึ่งอัตโนมัติ
OnDoTranform()	ทำการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม และจะเปลี่ยนโหมดการทำงานปัจจุบันเป็นโหมดการแปลงแบบอัตโนมัติ
OnDrawSc()	เปลี่ยนโหมดการทำงานปัจจุบันเป็นโหมดการวาดผังโครงสร้าง

ชื่อคลาส: CD2SView คลาสแม่ : CScrollView

คำอธิบาย: เป็นคลาสที่ทำหน้าที่ในการแสดงผลข้อมูลให้กับผู้ใช้

ตารางที่ ก-5 รายการคุณลักษณะของคลาส CD2SView

ชื่อคุณลักษณะ	คำอธิบาย
m_selection	เก็บรายการองค์ประกอบที่ถูกเลือกจากผู้ใช้
m_bGrid	แฟลกสถานะการแสดงเส้นกริด (T/F)

ตารางที่ ก-6 รายการวิธีทำงานของคลาส CD2SView

ชื่อวิธีทำงาน	คำอธิบาย
GetDocument()	ขอเลขที่อ้างอิงของเอกสารที่วิวแสดงผลเชื่อมโยงอยู่
OnDraw()	ทำการวาดหน้าจอเพื่อแสดงผล โดยจะเรียก CD2SDoc::Draw() ทำงาน
Select()	เก็บตัวชี้ขององค์ประกอบที่ผู้ใช้เลือกเข้าไปยัง m_selection และวาดกรอบการเลือกที่ตัวองค์ประกอบ
Deselect()	ลบตัวชี้ขององค์ประกอบที่ระบุออกจาก m_selection และลบกรอบการเลือกเดิมออก
CloneSelection()	สร้างสำเนาขององค์ประกอบที่ผู้ใช้เลือกออกมาเป็นองค์ประกอบอันใหม่

ชื่อคลาส: CD2STool คลาสแม่ : -

คำอธิบาย: เป็นคลาสที่ทำหน้าที่ติดต่อกับแถบเครื่องมือในการวาดแผนภาพและแถบเครื่องมือการแปลงเพื่อรับคำสั่งจากผู้ใช้

ตารางที่ ก-7 รายการคุณลักษณะของคลาส CD2STool

ชื่อคุณลักษณะ	คำอธิบาย
m_nDrawShape	ประเภทขององค์ประกอบที่ผู้ใช้เลือกปัจจุบัน (ได้แก่ เอนทิตี้ภายนอก กระบวนการทำงาน หน่วยเก็บข้อมูล เส้นกระแสข้อมูล มอดูล หรือเส้นเชื่อมโยงมอดูล)
m_nTranShape	ประเภทขององค์ประกอบกำหนดการแปลงที่ผู้ใช้เลือกปัจจุบัน (ได้แก่ ส่วนนำเข้าข้อมูล ส่วนแสดงผลข้อมูล หรือศูนย์กลางทรานแซกชั่น)

ตารางที่ ก-8 รายการวิธีทำงานของคลาส CD2STool

ชื่อวิธีทำงาน	คำอธิบาย
OnLButtonDown()	เลือกองค์ประกอบ หรือกำหนดตำแหน่งที่จะสร้างองค์ประกอบใหม่
OnLButtonUp()	สิ้นสุดสถานะการสร้างองค์ประกอบ หรือการเคลื่อนย้ายองค์ประกอบ
OnLButtonDblClk()	เรียกหน้าต่างคุณลักษณะขององค์ประกอบที่เลือก
OnRButtonDown()	เรียกเมนูคุณลักษณะขององค์ประกอบที่เลือก
OnMouseMove()	เคลื่อนย้ายตำแหน่งขององค์ประกอบที่เลือก

ชื่อคลาส: CD2SObj คลาสแม่ : CObject

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบของแผนภาพ ซึ่งคลาสนี้จะเก็บคุณลักษณะ (Attributes) และวิธีการทำงาน (Methods) ที่ใช้ร่วมกันสำหรับองค์ประกอบต่างๆ ของแผนภาพ และเป็นคลาสแม่ให้กับคลาส CObjEnt, CObjPrc, CObjDts, CObjFlw, CObjMod และ CobjLnk

ตารางที่ ก-9 รายการคุณลักษณะของคลาส CD2SObj

ชื่อคุณลักษณะ	คำอธิบาย
m_id	เลขที่อ้างอิงขององค์ประกอบ
m_nShape	ประเภทขององค์ประกอบ (ได้แก่ เอนทิตี้ภายนอก กระบวนการทำงาน หน่วยเก็บข้อมูล เส้นกระแสข้อมูล มอดูล หรือเส้นเชื่อมโยงมอดูล)
m_pDocument	เลขที่อ้างอิงของเอกสารที่องค์ประกอบเชื่อมโยงอยู่
m_position	ตำแหน่งขององค์ประกอบบนหน้าจอ (เก็บตำแหน่งโคออดิเนตแบบเอ็กซ์-วายของมุมซ้ายบนและมุมขวาล่าง)

ชื่อคลาส: CObjEnt คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบเอนทิตีภายนอก

ตารางที่ ก-10 รายการคุณลักษณะของคลาส CD2SEnt

ชื่อคุณลักษณะ	คำอธิบาย
m_number	หมายเลขของเอนทิตีที่ผู้ใช้กำหนด
m_name	ชื่อของเอนทิตี
m_description	คำอธิบายเอนทิตี
m_clone	แฟลกสถานะว่าเป็นองค์ประกอบสำเนา (T/F)

ตารางที่ ก-11 รายการวิธีทำงานของคลาส CD2SEnt

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดเอนทิตี
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของเอนทิตี

ชื่อคลาส: CObjPrc คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบกระบวนการทำงาน

ตารางที่ ก-12 รายการคุณลักษณะของคลาส CD2SPrc

ชื่อคุณลักษณะ	คำอธิบาย
m_number	หมายเลขของกระบวนการทำงานที่ผู้ใช้กำหนด
m_name	ชื่อของกระบวนการทำงาน
m_description	คำอธิบายกระบวนการทำงาน
m_spec	ข้อกำหนดหรือคำอธิบายการทำงานของกระบวนการทำงาน
m_nTrans	ประเภทขององค์ประกอบกำหนดการแปลงเพื่อแสดงว่ากระบวนการทำงานถูกระบุเป็นศูนย์กลางทรานแซกชั่น

ตารางที่ ก-13 รายการวิธีทำงานของคลาส CD2SPrc

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดกระบวนการทำงาน
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของกระบวนการทำงาน

ชื่อคลาส: CObjDts คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบหน่วยจัดเก็บข้อมูล

ตารางที่ ก-14 รายการคุณลักษณะของคลาส CD2SDts

ชื่อคุณลักษณะ	คำอธิบาย
m_number	หมายเลขของหน่วยจัดเก็บข้อมูลที่ผู้ใช้กำหนด
m_name	ชื่อของหน่วยจัดเก็บข้อมูล
m_description	คำอธิบายหน่วยจัดเก็บข้อมูล
m_clone	แฟลกสถานะว่าเป็นองค์ประกอบสำเนา (T/F)

ตารางที่ ก-15 รายการวิธีทำงานของคลาส CD2SDts

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดหน่วยจัดเก็บข้อมูล
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของหน่วยจัดเก็บข้อมูล

ชื่อคลาส: CObjFlw คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบกระแสข้อมูล

ตารางที่ ก-16 รายการคุณลักษณะของคลาส CD2SFlw

ชื่อคุณลักษณะ	คำอธิบาย
m_comp1_id	เลขที่อ้างอิงขององค์ประกอบต้นทางของกระแสข้อมูล
m_comp2_id	เลขที่อ้างอิงขององค์ประกอบปลายทางของกระแสข้อมูล
m_label	ชื่อข้อมูลบนกระแสข้อมูล
m_comp1_xy	ตำแหน่งเปรียบเทียบของมุมซ้ายบนขององค์ประกอบต้นทางกับมุมซ้ายบนของกระแสข้อมูล
m_comp2_xy	ตำแหน่งเปรียบเทียบของมุมซ้ายบนขององค์ประกอบปลายทางกับมุมขวาล่างของกระแสข้อมูล
m_nTrans	ประเภทขององค์ประกอบกำหนดการแปลงเพื่อแสดงว่าเส้นกระแสข้อมูลถูกระบุเป็นส่วนนำเข้าข้อมูลหรือส่วนแสดงผลลัพธ์

ตารางที่ ก-17 รายการวิธีทำงานของคลาส CD2SFlw

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดเส้นกระแสข้อมูล
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของเส้นกระแสข้อมูล

ชื่อคลาส: CObjMod คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบมอดูล

ตารางที่ ก-18 รายการคุณลักษณะของคลาส CD2SMod

ชื่อคุณลักษณะ	คำอธิบาย
m_number	หมายเลขของมอดูลที่ผู้ใช้กำหนด
m_name	ชื่อของมอดูล
m_description	คำอธิบายมอดูล
m_spec	ข้อกำหนดหรือคำอธิบายการทำงานของมอดูล
m_center	แฟลกสถานะว่ามอดูลเป็นศูนย์กลางทรานแซกชั่น (T/F)

ตารางที่ ก-19 รายการวิธีทำงานของคลาส CD2SMod

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดมอดูล
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของมอดูล

ชื่อคลาส: CObjLnk คลาสแม่ : CD2SObj

คำอธิบาย: เป็นคลาสที่แสดงถึงองค์ประกอบแบบเส้นเชื่อมโยงมอดูล

ตารางที่ ก-20 รายการคุณลักษณะของคลาส CD2SLnk

ชื่อคุณลักษณะ	คำอธิบาย
m_comp1_id	เลขที่อ้างอิงของมอดูลต้นทางของเส้นเชื่อมโยงมอดูล
m_comp2_id	เลขที่อ้างอิงของมอดูลปลายทางของเส้นเชื่อมโยงมอดูล
m_up_data	ชื่อตัวคู่ต่อแบบข้อมูลที่วิ่งขึ้นด้านบนของเส้นเชื่อมโยงมอดูล
m_up_control	ชื่อตัวคู่ต่อแบบข้อมูลที่วิ่งขึ้นด้านบนของเส้นเชื่อมโยงมอดูล
m_down_data	ชื่อตัวคู่ต่อแบบควบคุมที่วิ่งลงด้านล่างของเส้นเชื่อมโยงมอดูล
m_down_control	ชื่อตัวคู่ต่อแบบควบคุมที่วิ่งลงด้านล่างของเส้นเชื่อมโยงมอดูล
m_comp1_xy	ตำแหน่งเปรียบเทียบของมุมซ้ายบนของมอดูลต้นทางกับมุมซ้ายบนของเส้นเชื่อมโยงมอดูล
m_comp2_xy	ตำแหน่งเปรียบเทียบของมุมซ้ายบนของมอดูลปลายทางกับมุมขวาล่างของเส้นเชื่อมโยงมอดูล

ตารางที่ ก-21 รายการวิธีทำงานของคลาส CD2SLnk

ชื่อวิธีทำงาน	คำอธิบาย
Draw()	วาดเส้นเชื่อมโยงมอดูล
OnEditProperties()	แสดงหน้าต่างกำหนดคุณลักษณะของเส้นเชื่อมโยงมอดูล

ชื่อคลาส: CImportor คลาสแม่ : -

คำอธิบาย: เป็นคลาสที่ทำหน้าที่นำเข้าแฟ้มข้อมูลของแผนภาพกระแสข้อมูลที่ได้จากโปรแกรม Power Designer 6.1

ตารางที่ ก-22 รายการคุณลักษณะของคลาส CImportor

ชื่อคุณลักษณะ	คำอธิบาย
prcList	รายการข้อมูลที่เป็นกระบวนการทำงาน
entList	รายการข้อมูลที่เป็นเอนทิตีภายนอก
dtsList	รายการข้อมูลที่เป็นหน่วยจัดเก็บข้อมูล
infoList	รายการข้อมูลที่เป็นตัวข้อมูล
flowList	รายการข้อมูลที่เป็นเส้นกระแสข้อมูล
m_runnumber	เลขจที่อ้างอิงสุดท้ายที่กำหนดให้กับองค์ประกอบของแผนภาพ

ตารางที่ ก-23 รายการวิธีทำงานของคลาส CImportor

ชื่อวิธีทำงาน	คำอธิบาย
ImportDfd()	อ่านแฟ้มข้อมูลของ Power Designer เข้ามา และสร้างรายการข้อมูลที่อยู่ในแฟ้มข้อมูล
ValidateComp()	สร้างองค์ประกอบของแผนภาพกระแสข้อมูลจากรายการข้อมูลที่ได้จากแฟ้มข้อมูล
SaveAsD2S()	จัดเก็บองค์ประกอบลงแฟ้มข้อมูลชั่วคราวในรูปแบบของโปรแกรม DFD2SC

ชื่อคลาส: CTransformer คลาสแม่ : -

คำอธิบาย: เป็นคลาสที่ทำหน้าที่ประมวลผลในการแปลงแผนภาพกระแสข้อมูลเป็นผังโครงสร้างของโปรแกรม

ตารางที่ ก-24 รายการคุณลักษณะของคลาส CTransformer

ชื่อคุณลักษณะ	คำอธิบาย
m_pDocument	เลขที่อ้างอิงของเอกสารที่ตัวแปลงเชื่อมโยงอยู่
LstPath	รายการของเส้นทางสำหรับการแปลงด้วยวิธีวิเคราะห์แบบแปลง
LstAnswer	รายการของชุดคำตอบที่ได้จากการแปลงด้วยวิธีวิเคราะห์แบบแปลง
lstModule	รายการขององค์ประกอบแบบมอดูลที่สร้างขึ้นจากผลลัพธ์ของการแปลง
lstSc	รายการของผังโครงสร้างของโปรแกรมที่เป็นผลลัพธ์จากการแปลง
m_total	จำนวนของผังโครงสร้างที่เป็นผลลัพธ์จากการแปลง
m_current	หมายเลขของผังโครงสร้างที่ถูกแสดงผลอยู่ ณ ปัจจุบัน

ตารางที่ ก-25 รายการวิธีทำงานของคลาส CTransformer

ชื่อวิธีทำงาน	คำอธิบาย
CheckValidDfd()	ตรวจสอบว่าแผนภาพกระแสข้อมูลวาดถูกต้องตามหลักเกณฑ์หรือไม่
IsTransac()	ทดสอบว่าแผนภาพกระแสข้อมูลมีคุณลักษณะกระแสทรานแซกชั่นหรือไม่ ถ้าใช่ จะเรียก TransacAnalysis() ขึ้นมาทำงาน
TransacAnalysis()	ทำการแปลงด้วยวิธีวิเคราะห์แบบทรานแซกชั่น
TransformAnalysis()	ทำการแปลงด้วยวิธีวิเคราะห์แบบแปลง
GenPath()	สร้างเส้นทางการไหลของข้อมูลสำหรับวิธีวิเคราะห์แบบแปลง
TraceForward()	ค้นหาเส้นทางแบบวิ่งไปข้างหน้า
TraceBackward()	ค้นหาเส้นทางแบบวิ่งย้อนกลับ
ConstructSC()	นำผลการแปลงมาสร้างเป็นผังโครงสร้างของโปรแกรม
FindCouple()	หาตัวข้อมูลในแผนภาพกระแสข้อมูลเพื่อนำมาแสดงเป็นตัวคู่ต่อในผังโครงสร้างของโปรแกรม
DrawSC()	แสดงผังโครงสร้างของโปรแกรมผลลัพธ์ตามค่าของ m_current

ชื่อคลาส: CMeasurer คลาสแม่ : -

คำอธิบาย: เป็นคลาสที่ทำหน้าที่วัดค่าความสัมพันธ์ต่อกันระหว่างมอดูลของรูปผังโครงสร้างของโปรแกรม

ตารางที่ ก-26 รายการคุณลักษณะของคลาส Cmeasurer

ชื่อคุณลักษณะ	คำอธิบาย
m_code	เลขที่สุดท้ายที่ใช้กำหนดเป็นเลขอ้างอิงโทเค็น
lstToken	รายการของโทเค็นที่ได้จากบีเอ็นเอฟ
lstTag	รายการของนอนเทอร์มินอลที่ได้จากบีเอ็นเอฟ
lstTokSeq	รายการของโทเค็นที่ได้จากรหัสต้นฉบับ
startPos	ตำแหน่งเริ่มต้นของ TokSeq ที่กำลังตรวจสอบไวยากรณ์
endPos	ตำแหน่งปัจจุบันของ TokSeq ที่กำลังตรวจสอบไวยากรณ์
lstMeasur	รายการของกฎในการตรวจสอบซึ่งถูกแปลงให้อยู่ในรูปของโทเค็นแล้ว
lstCall	รายการของการเรียกทำงานระหว่างมอดูลที่ได้จากการตรวจสอบการใช้งาน
lstVar	รายการของพารามิเตอร์และตัวแปรที่ได้จากการตรวจสอบการใช้งาน

ตารางที่ ก-26 รายการคุณลักษณะของคลาส Cmeasurer (ต่อ)

ชื่อคุณลักษณะ	คำอธิบาย
lstStruct	รายการของโครงสร้างข้อมูลที่ได้จากการตรวจสอบการใช้งาน
lstExternal	รายการของแฟ้มข้อมูลแบบภายนอกที่ได้จากการตรวจสอบการใช้งาน

ตารางที่ ก-27 รายการวิธีทำงานของคลาส CMeasurer

ชื่อวิธีทำงาน	คำอธิบาย
DiscoveryTag()	สร้างรายการของนอนเทอร์มินอลจากบีเอ็นเอฟ
DiscoveryKeyword()	สร้างรายการโทเค็น (ตารางสัญลักษณ์) จากบีเอ็นเอฟ
ParseModule()	ตรวจสอบไวยากรณ์ของคำอธิบายการทำงานของมอดูล โดยแปลงจากรหัสต้นฉบับเป็นโทเค็น
TestTokenWithTag()	เปรียบเทียบโทเค็นของรหัสต้นฉบับกับนอนเทอร์มินอล
TestTokenWithToken()	เปรียบเทียบโทเค็นของรหัสต้นฉบับกับโทเค็นคำหลัก
TestTokenWithIdentifier()	เปรียบเทียบโทเค็นของรหัสต้นฉบับกับโทเค็นตัวระบุ
TestTokenWithConstant()	เปรียบเทียบโทเค็นของรหัสต้นฉบับกับโทเค็นค่าคงที่
AnalyzeSentense()	ตรวจสอบประเภทและการใช้งานของพารามิเตอร์
CalCoupling()	คำนวณค่าความสัมพันธ์ระหว่างมอดูลของผังโครงสร้าง

ภาคผนวก ข.

บีเอ็นเอฟ

<module> ::= module <module identifier> (<parameter declare part>) <block>

<parameter declare part> ::= <type identifier> <parameter identifier> { , <type identifier> <parameter identifier> } |
<empty>

<type identifier> ::= <simple type> | <array type>

<simple type> ::= integer | float | char | datetime | <struct identifier>

<array type> ::= <simple type> [<index>]

<index> ::= <number constant>

<block> ::= begin <statement list> end

<statement list> ::= <statement> { <statement> }

<statement> ::= <simple statement> | <structure statement>

<simple statement> ::= <struct declare statement> | <variable declare statement> |
<assign statement> | <call statement> | <read statement> | <write statement> |
<input statement> | <message statement> | <prompt statement> | <print statement> |
<break statement> | <continue statement> | <exit statement>

<struct declare statement> ::= define type <struct identifier> as <type identifier> <attribute identifier> { ,
<type identifier> <attribute identifier> } ;

<variable declare statement> ::= global <type identifier> <identifier-list> ; |
shared <type identifier> <identifier-list> ; |
<type identifier> <identifier-list> ;

<identifier-list> ::= <variable identifier> { , <variable identifier> }

<assign statement> ::= <variable> = <call statement> |
<variable> = <calculation expression> ;

<variable> ::= <variable identifier> . <attribute identifier> | <variable identifier> [<index>] | <variable identifier>

<calculation expression> ::= <calculation term> <calculation operator> <calculation term> | <calculation term>

<calculation term> ::= (<calculation expression>) | <variable> | <cmd constant>

<calculation operator> ::= + | - | * | / | ^

<call statement> ::= call <module identifier> (<parameter list>) ;

<parameter list> ::= <variable> { , <variable> } | <empty>

<read statement> ::= read <variable list> from <device type> <device constant> ;

<variable list> ::= <variable identifier> { , <variable identifier> }

<device type> ::= file | table | database | tape | disk | <empty>

<write statement> ::= write <variable list> into <device type> <device constant> ;

<input statement> ::= input <variable list> ;

<message statement> ::= message <string expression> ; |
message <calculation expression> ;

<string expression> ::= <string variable> + <string expression> |
<string variable>

<string variable> ::= <variable identifier> | <string constant>

<prompt statement> ::= <variable> = prompt <string constant> ;

<print statement> ::= print <variable list> ;

<break statement> ::= break ;

<continue statement> ::= continue ;

<exit statement> ::= exit <exit expression> ;

<exit expression> ::= <calculation expression> | <empty>

<structure statement> ::= <if statement> | <while statement> | <for statement> | <repeat statement> |
<case statement>

<if statement> ::= if <relation expression> then <statement list> { elseif <relation expression> then
<statement list> } else <statement list> endif ; |
if <relation expression> then <statement list> else <statement list> endif ; |
if <relation expression> then <statement list> endif ;

<relation expression> ::= (<relation term> <relation operator> <relation term>) |
<relation term> <relation operator> <relation term>

<relation term> ::= <variable identifier> | <cmd constant> | <module identifier>

<relation operator> ::= = | <> | >= | > | <= | < | and | or

<while statement> ::= while <relation expression> do <statement list> endwhile ;

<for statement> ::= for <variable identifier> = <integer variable> to <integer variable> <statement list> endfor ;

<integer variable> ::= <variable identifier> | <number constant>

<repeat statement> ::= repeat <statement list> until <relation expression> ;

<case statement> ::= switch <variable identifier> case : <cmd constant> <statement list> { case : <cmd constant>
<statement list> } endcase ; |
switch <variable identifier> case : <cmd constant> <statement list> { case : <cmd constant>
<statement list> } default : <statement list> endcase ;

<module identifier> ::= <identifier>

<parameter identifier> ::= <identifier>

<variable identifier> ::= <identifier>

<struct identifier> ::= <identifier>

<attribute identifier> ::= <identifier>

<device constant> ::= <string constant>

<cmd constant> ::= <number constant> | <string constant>

<number constant> ::= <constant>

<string constant> ::= <constant>

ประวัติผู้วิจัย

นาย ทวีเกียรติ เอี่ยมงามทรัพย์ เกิดเมื่อวันที่ 27 กุมภาพันธ์ พ.ศ. 2515 สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต สาขาวิชาฟิสิกส์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2535 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2539 โดยก่อนเข้าศึกษาในหลักสูตรวิทยาศาสตร์มหาบัณฑิต ได้ทำงานที่บริษัทไมโครเอ็กซ์จำกัดในตำแหน่งผู้สนับสนุนทางเทคนิค