ข้อกำหนดรูปนัยเพื่อทวนสอบแผนภาพกิจกรรมของกระแสกระบวนการ

นายเจริญศักดิ์ นาคงาม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

FORMAL SPECIFICATION FOR VERIFYING ACTIVITY DIAGRAM OF PROCESS FLOW

Mr. Charoensak Narkngam

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

หัวข้อวิทยานิพนธ์	ข้อกำหนดรูปนัยเพื่อทวนสอบแผนภาพกิจกรรมของกระแสกระบวนการ
โดย	นายเจริญศักดิ์ นาคงาม
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

……. ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

……. กรรมการ
(อาจารย์ ดร.นัทที นิภานันท์)

……. กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

เจริญศักดิ์ นาคงาม : ข้อกำหนดรูปนัยเพื่อทวนสอบแผนภาพกิจกรรมของกระแสกระบวนการ. (FORMAL SPECIFICATION FOR VERIFYING ACTIVITY DIAGRAM OF PROCESS FLOW). อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.ญาใจ ลิ่มปิยะกรณ์, 89 หน้า.

ภาษาจำเพาะโดเมนหรือดีเอสแอล คือ ภาษาข้อกำหนดที่อยู่ในรูปแบบของข้อความหรือแบบจำลองที่ถูกออกแบบโดยเฉพาะเจาะจงสำหรับโดเมนปัญหาหนึ่งๆ งานวิจัยนี้ได้สร้างภาษาอธิบายการกระทำ หรือเอดีแอล ซึ่งจัดอยู่ในประเภทภาษาจำเพาะโดเมน เพื่อป้องกันมโนทัศน์ที่ผิดและความไม่ตรงกันของพฤติกรรมในแผนภาพกิจกรรม นอกจากนี้ ยังได้พัฒนาวิธีการอิงวัตถุสำหรับการสร้างแผนภาพกิจกรรมบนบทคำสั่งเอดีแอล โดยหลังจากการแจงส่วนบทคำสั่งเอดีแอลแบบจำลองความหมายจะถูกสร้างขึ้นบนพื้นฐานของเอดีแอลเมทาโมเดล วิธีการที่นำเสนอสามารถสร้างแผนภาพกิจกรรมได้อย่างถูกต้อง เริ่มต้นจากการสกัดข้อมูลจากแต่ละแอ็คชั่น แล้วตรวจจับหาความสัมพันธ์ระหว่างวัตถุในการกำหนดบัพควบคุมต่างๆ ได้แก่ บัพแยก บัพรวม บัพตัดสินใจ บัพผสาน บัพเริ่มต้น บัพหยุดสายงาน และบัพหยุดกิจกรรม แนวทางที่นำเสนอยังได้จัดสร้างกฎตรวจสอบความสมเหตุสมผลสำหรับเอดีแอลเพื่อป้องกันความไม่ตรงกันของข้อมูลและพฤติกรรมในแผนภาพกิจกรรม รวมทั้งกฎการทวนสอบเพื่อให้แผนภาพถูกสร้างขึ้นอย่างถูกต้องตามข้อกำหนดมากยิ่งขึ้น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2554

5170266621 : MAJOR SOFTWARE ENGINEERING
KEYWORDS : ACTIVITY DIAGRAM MODELING / DOMAIN SPECIFIC LANGUAGE / METAMODEL / QUALITY CONTROL

CHAROENSAK NARKNGAM : FORMAL SPECIFICATION FOR VERIFYING ACTIVITY DIAGRAM OF PROCESS FLOW. ADVISOR : ASSOC. PROF. YACHAI LIMPIYAKORN, Ph.D., 89 pp.

A domain specific language or DSL is a specification language in textual format or model that is dedicated to a particular problem domain. In this research, the action description language or ADL, which is a domain specific language, has been invented to prevent misconception and inconsistencies of behaviors residing activity diagrams. Additionally, the object-based method has been developed for diagram generation with the underlying ADL scripts. Once the ADL script has been parsed, the semantic model is created based on the ADL metamodel. Starting from extracting data from actions, then detecting object relations to determine the controls, the proposed method could properly construct the activity diagram, which contains all control nodes including fork, join, decision, merge, initial, flow final and activity final. The approach also establishes the validation rules for ADL to prevent the activity diagram from inconsistent data and behaviors, as well as the verification rules to promote conformance to specifications.

Department : Computer Engineering Student's Signature .....
Field of Study : Software Engineering Advisor's Signature .....
Academic Year : 2011

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความอนุเคราะห์อย่างดียิ่งของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้สละเวลาให้ความรู้ ให้คำปรึกษา ตรวจสอบ ให้คำแนะนำแนวทางการวิจัย และสนับสนุน จนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี ข้าพเจ้าจึงขอกราบระลึกถึงพระคุณของอาจารย์ไว้ ณ ที่นี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. สุกรี สินธุภิญโญ อาจารย์ ดร.นัทที นิภานันท์ และอาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลา ให้คำแนะนำ ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ท้ายที่สุด ผู้เสนอวิทยานิพนธ์ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และครอบครัว สำหรับกำลังใจที่มีค่ายิ่ง รวมถึงขอขอบพระคุณผู้บังคับบัญชาในสายงาน เพื่อนร่วมงาน และมิตรสหาย ที่คอยติดตามให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ และท่านอื่นๆ ที่มิได้กล่าวชื่อไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์ของข้าพเจ้าสำเร็จไปได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

แผนภาพกิจกรรม (Activity Diagram) เป็นแผนภาพที่นิยมนำไปใช้กันอย่างกว้างขวางในหลายสายงานและหลายสาขาวิชาชีพ ซึ่งแผนภาพกิจกรรมทำหน้าที่ในการอธิบายลำดับขั้นตอนการดำเนินงานและเงื่อนไขที่เกี่ยวข้องกับการดำเนินงานในขั้นตอนต่าง ๆ ของระบบที่สนใจ อย่างไรก็ตามการออกแบบแผนภาพกิจกรรมที่มีขนาดใหญ่และซับซ้อนให้มีความถูกต้องสมบูรณ์นั้น จำเป็นต้องใช้เวลาที่นานและมีโอกาสที่จะเกิดความผิดพลาดระหว่างออกแบบแผนภาพกิจกรรมได้ง่าย ทำให้เกิดความไม่สอดคล้องต้องกันในแต่ละกิจกรรมได้ เป็นผลให้กระบวนการจากแผนภาพกิจกรรมและการดำเนินงานจริงมีความขัดแย้งกันเกิดขึ้น ซึ่งจะนำไปสู่ความล้มเหลวในการใช้กระบวนการในที่สุด

ข้อกำหนดรูปนัยเป็นหนึ่งในวิธีที่นิยมมาใช้ในการกำหนดรายละเอียดและการตรวจสอบความถูกต้องของข้อมูลของแบบจำลอง เพราะสามารถกำหนดรายละเอียดการดำนินงานได้มากกว่าการสร้างแบบจำลองด้วยสัญกรณ์กราฟิก (Graphic Notation) เพียงอย่างเดียว อย่างไรก็ตามการออกแบบแผนภาพกิจกรรมด้วยข้อกำหนดรูปนัยเชิงคณิตศาสตร์นั้น ไม่เป็นที่นิยมมากนัก เนื่องจากยากต่อการเรียนรู้ และการนำไปใช้งานจริง

งานวิจัยชิ้นนี้จึงมีจุดมุ่งหมายที่จะเปลี่ยนแปลงวิธีการออกแบบแผนภาพกิจกรรมที่ใช้สัญกรณ์กราฟิก มาเป็นการใช้ข้อกำหนดรูปนัยในลักษณะของดีเอสแอล (Domain-Specific Language - DSL) ซึ่งจะช่วยลดความผิดพลาดที่เกิดขึ้นในการออกแบบแผนภาพกิจกรรม เช่น การออกแบบโดยสัญกรณ์กราฟิกผิดความหมาย การออกแบบแผนภาพกิจกรรมไม่ตรงตามมาตรฐาน เป็นต้น นอกจากนี้ยังสามารถลดความยุ่งยากในการกำหนดการไหลของข้อมูลในกิจกรรมที่มีความซับซ้อนหรือกิจกรรมที่มีข้อมูลที่ง่ายต่อการเกิดความสับสน โดยดีเอสแอลที่ออกแบบขึ้นมานั้นจะเน้นที่การออกแบบให้รองรับกิจกรรมในเชิงธุรกิจเป็นหลัก ซึ่งช่วยให้ง่ายต่อการเรียนรู้และการอ่านลำดับการทำงานของกิจกรรม

1.2 วัตถุประสงค์ของการวิจัย

เพื่อนำเสนอแนวทางการสร้างและการออกแบบข้อกำหนดรูปนัยในลักษณะของดีเอสแอลที่สามารถผสมผสานการออกแบบแผนภาพกิจกรรมและการอธิบายการทำงานในเชิงธุรกิจเข้าด้วยกัน และเพื่อนำแสนอแนวทางการป้องกันข้อผิดพลาดที่เกี่ยวข้องกับการออกแบบแผนภาพกิจกรรมและการไม่ต้องกันของกระบวนการอย่างมีประสิทธิภาพ

1.3 ขอบเขตของการวิจัย

1. สร้างข้อกำหนดรูปนัยในลักษณะของดีเอสแอลเพื่อรองรับการออกแบบด้วยสัญกรณ์ทางกราฟิกต่าง ๆ ของแผนภาพกิจกรรม
2. ระบบที่พัฒนาขึ้นสามารถสร้างแผนภาพกิจกรรมระดับปานกลางเชิงป้องกันจากข้อกำหนดรูปนัยได้
3. ระบบที่พัฒนาขึ้นประกอบด้วยซอฟต์แวร์สำหรับแปลงข้อมูลจากข้อกำหนดรูปนัยเป็นแผนภาพกิจกรรม

1.4 ข้อตกลงเบื้องต้น

1. แผนภาพกิจกรรมเป็นไปตามข้อกำหนดของยูเอ็มแอลเวอร์ชั่น 2.3 (UML 2.3)
2. ซอฟต์แวร์สามารถรายงานข้อผิดพลาดหากพบว่าการร้อยกระบวนการจากข้อกำหนดรูปนัยมีการไม่ต้องกันของกระบวนการเกิดขึ้น และสามารถสร้างแผนภาพกิจกรรมได้ในกรณีที่ไม่มีข้อผิดพลาด
3. ประเมินผลงานวิจัยจากการตรวจสอบความถูกต้องของแผนภาพกิจกรรม ด้วยวิธีการเปรียบเทียบระหว่างตัวอย่างข้อมูลกับผลลัพธ์ที่ได้จากระบบที่พัฒนาขึ้น

1.5 ข้อจำกัดของการวิจัย

1. ซอฟต์แวร์ไม่สามารถแปลงกลับแผนภาพกิจกรรมเป็นข้อกำหนดรูปนัยได้
2. ข้อมูลในรูปแบบของเอ็กซ์เอ็มไอ (XML Metadata Interchange - XMI) ที่ได้จากระบบที่พัฒนาขึ้น ไม่สามารถใช้เป็นข้อมูลนำเข้าให้กับซอฟต์แวร์อื่นได้
3. เอดีแอล (Action Description Language - ADL) ที่สร้างขึ้นในงานวิจัยนี้ รองรับเฉพาะแผนภาพกิจกรรมเชิงธุรกิจเท่านั้น กล่าวคือยังไม่รองรับแผนภาพกิจกรรมเชิงโครงสร้างหรือแผนภาพกิจกรรมที่ใช้สำหรับการเขียนหรือพัฒนาโปรแกรม

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. แนวทางการออกแบบข้อกำหนดรูปนัยจากข้อกำหนดในลักษณะของดีเอสแอลสำหรับแผนภาพกิจกรรม
2. แนวทางในการสร้างแผนภาพกิจกรรมจากข้อกำหนดรูปนัย
3. แนวทางในการป้องกันความไม่ต้องกันของกระบวนการจากการออกแบบแผนภาพกิจกรรม

1.7 วิธีดำเนินงานวิจัย

1. ออกแบบข้อกำหนดรูปนัยในลักษณะของดีเอสแอลจากแผนภาพกิจกรรม
2. ศึกษาข้อจำกัดการใช้งานที่พบในข้อ 1 และปรับปรุงเพื่อให้สอดคล้องกับมาตรฐานในการออกแบบแผนภาพกิจกรรม
3. ศึกษาแนวทางการสร้างแผนภาพกิจกรรม
4. สร้างซอฟต์แวร์สำหรับทดสอบแนวคิด
5. ตรวจสอบความถูกต้องของแผนภาพกิจกรรมที่ได้จากข้อ 4 โดยเปรียบเทียบกับข้อมูลตัวอย่าง
6. ตีพิมพ์ผลงานวิชาการ
7. ปรับปรุงซอฟต์แวร์และสร้างระบบส่วนต่อประสานกับผู้ใช้
8. ประเมินผลที่ได้จากการสร้างแผนภาพกิจกรรมด้วยข้อกำหนดรูปนัย
9. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.8 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บท ดังต่อไปนี้ บทที่ 1 เป็นบทนำกล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับและผลงานตีพิมพ์ บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 กล่าวถึงวิธีดำเนินการวิจัย บทที่ 4 กล่าวถึง การออกแบบและพัฒนาระบบตามแนวทางการวิจัยที่นำเสนอ บทที่ 5 กล่าวถึงวิธีการประเมินและวัดผลการทดลองและบทที่ 6 สรุปผลการวิจัย ข้อเสนอแนะและแนวทางสำหรับการวิจัยต่อในอนาคต

1.9 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวข้อเรื่อง “Designing a Domain Specific Language for UML Activity Diagram”, Charoensak Narkngam and Yachai Limpiyakorn in 2012 4TH International Conference on Computer Engineering and Technology (ICCET 2012)

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวข้อเรื่อง “ภาษาจำเพาะโดเมนสำหรับแผนภาพกิจกรรม” โดย เจริญศักดิ์ นาคงาม และ ญาใจ ลิ่มปิยะกรณ์ ในวารสารรามคำแหง ฉบับวิศวกรรมศาสตร์ (Ramkhamhaeng Journal of Engineering) ปีที่ 6 ฉบับที่ 1

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

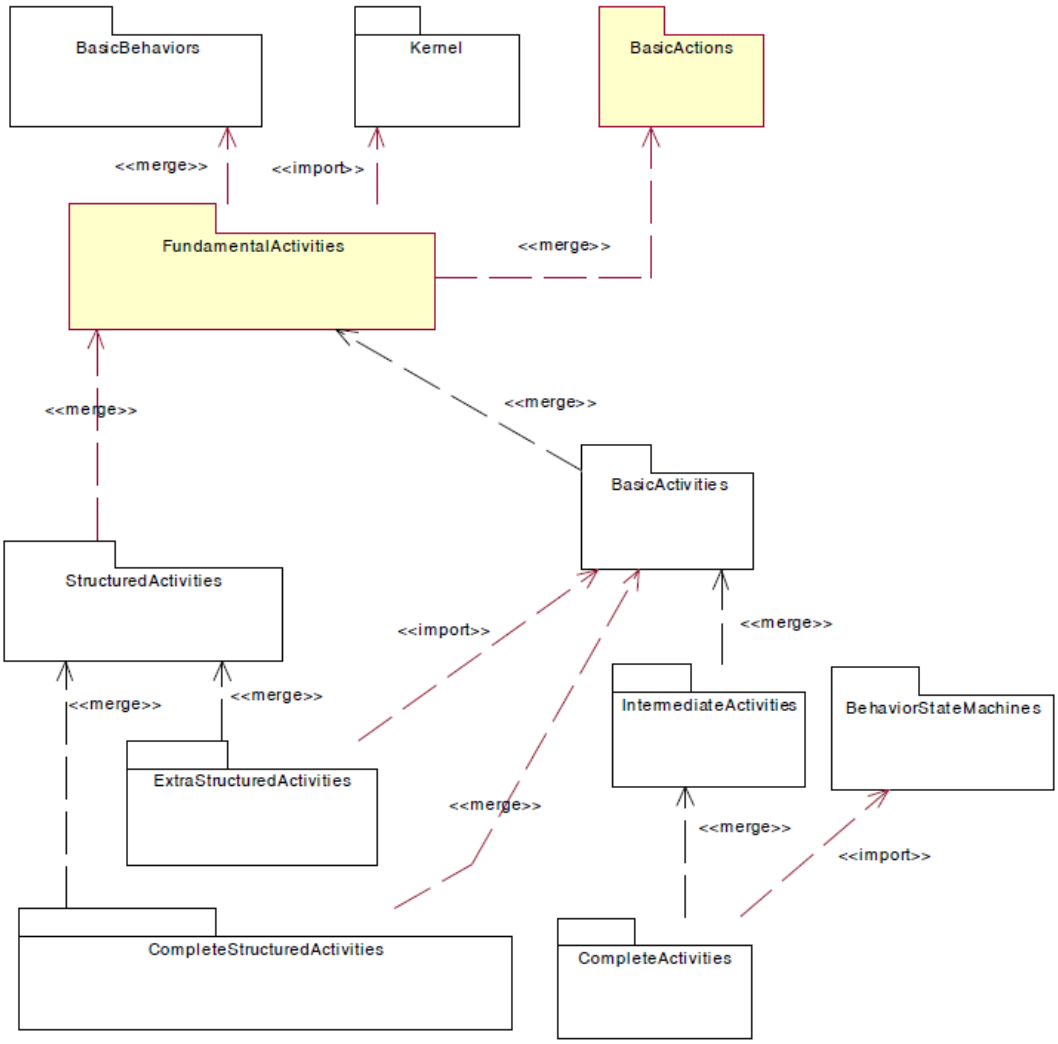
2.1.1 แผนภาพกิจกรรม

แผนภาพกิจกรรมถูกออกแบบและกำหนดเป็นมาตรฐานโดยโอเอ็มจี (Object Management Group - OMG) [1] ถูกออกแบบขึ้นเพื่ออธิบายลำดับการทำงานและเงื่อนไขต่าง ๆ ที่เกี่ยวข้องกับพฤติกรรมของระบบ แผนภาพกิจกรรมจะประกอบด้วยสายงานและบัพต่าง ๆ โดยสายงานจะบ่งบอกถึงลำดับการทำงาน ส่วนบัพจะเป็นตัวกำหนดการกระทำและพฤติกรรมของระบบ

แผนภาพกิจกรรมสามารถแบ่งกรอบงานได้ออกเป็นเจ็ดประเภทดังภาพที่ 1 โดยมีรายละเอียดดังต่อไปนี้

1. กิจกรรมระดับรากฐาน (Fundamental Activities) ใช้สำหรับกำหนดแอ็คชั่นหรือส่วนกระทำในกิจกรรม ซึ่งอาจมีการจัดกลุ่มแอ็คชั่นหรือไม่ก็ได้
2. กิจกรรมระดับพื้นฐาน (Basic Activities) ใช้สำหรับกำหนดลำดับการทำงานของแอ็คชั่นในกิจกรรมระดับรากฐาน และมีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของสายงาน
3. กิจกรรมระดับปานกลาง (Intermediate Activities) ใช้สำหรับกำหนดรายละเอียดลำดับการทำงานที่ได้จากกิจกรรมระดับพื้นฐาน โดยอธิบายถึงรายละเอียดของพฤติกรรมที่เกี่ยวข้องกับระบบ เช่น การอธิบายการทำงานที่สามารถทำพร้อมกันได้ การทำงานที่มีการตัดสินใจหรือเงื่อนไขเข้ามาเกี่ยวข้อง เป็นต้น
4. กิจกรรมระดับสมบูรณ์ (Complete Activities) ใช้สำหรับกำหนดรายละเอียดเพิ่มเติมจากกิจกรรมระดับปานกลาง โดยจะเน้นการกำหนดค่าน้ำหนักบนสายงานข้อมูลต่าง ๆ เพื่อให้มีรายละเอียดการดำเนินงานที่ชัดเจนขึ้น
5. กิจกรรมเชิงโครงสร้าง (Structured Activities) ใช้สำหรับการออกแบบจำลองเพื่อการเขียนหรือพัฒนาโปรแกรมเป็นหลัก โดยจะมีการกำหนดลำดับการทำงาน การวนซ้ำและเงื่อนไขที่เกี่ยวข้องกับลำดับการทำงานของแอ็คชั่นที่กำหนดขึ้นในกิจกรรมระดับรากฐาน
6. กิจกรรมเชิงโครงสร้างระดับสมบูรณ์ (Complete Structured Activities) ใช้สำหรับกำหนดรายละเอียดอินพุตและเอาต์พุตของแต่ละแอ็คชั่นในกิจกรรมเชิงโครงสร้าง

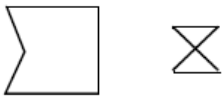
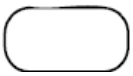
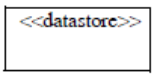
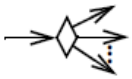
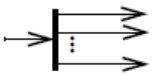
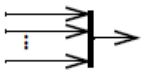
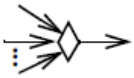
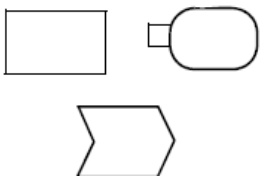
7. กิจกรรมเชิงโครงสร้างระดับพิเศษ (Extra Structured Activities) ใช้สำหรับกำหนดเงื่อนไขพิเศษต่าง ๆ ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ ซึ่งประกอบด้วยการจัดการสิ่งผิดปกติและการเรียกทำงานแอ็คชั่นผ่านเงื่อนไขหรือตัวควบคุมเหตุการณ์ต่าง ๆ ที่กำหนดขึ้น

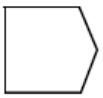
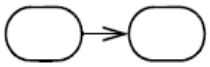
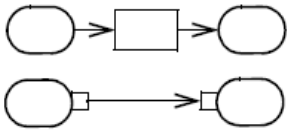
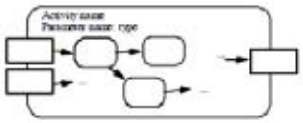
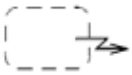
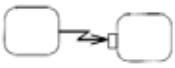
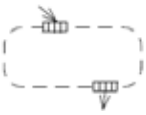
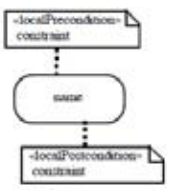


ภาพที่ 1 โครงสร้างความสัมพันธ์ของกิจกรรมแต่ละประเภท

การออกแบบแผนภาพกิจกรรมโดยปกแล้ว จำเป็นต้องอาศัยซอฟต์แวร์เฉพาะในการออกแบบโดยการออกแบบจะเป็นการเชื่อมสัญกรณ์กราฟิกต่าง ๆ เข้าด้วยกัน เพื่ออธิบายเป็นลำดับการทำงานและพฤติกรรมของระบบ โดยสัญกรณ์กราฟิกที่ใช้ในการออกแบบ ประเภทของสัญกรณ์กราฟิก และความสัมพันธ์ระหว่างสัญกรณ์กราฟิกกับระดับของกรอบงานแผนภาพกิจกรรมที่รองรับ แสดงดังตารางที่ 1

ตารางที่ 1 ประเภทและสัญลักษณ์ที่ใช้ในแผนภาพกิจกรรม

ประเภท	สัญลักษณ์	Fundamental	Basic	Intermediate	Complete	Structured	CompleteStructured	ExtraStructured
NODE								
AcceptEventAction					X			X
Action		X	X	X	X	X	X	X
ActivityFinal			X	X	X	X	X	X
DataStore					X			
DecisionNode				X	X	X	X	X
FlowFinal				X	X	X	X	X
ForkNode				X	X	X	X	X
InitialNode			X	X	X	X	X	X
JoinNode				X	X	X	X	X
MergeNode				X	X	X	X	X
ObjectNode			X	X	X			

ประเภท	สัญลักษณ์	Fundamental	Basic	Intermediate	Complete	Structured	CompleteStructured	ExtraStructured
SendSignalAction					X			X
PATH								
ControlFlow			X	X	X	X	X	X
ObjectFlow			X	X	X			
OTHER ELEMENTS								
Activity		X	X	X	X	X	X	X
ActivityPartition				X	X			
InterruptibleActivityRegion					X			X
ExceptionHandler								X
ExpansionRegion								X
LocalPrecondition, LocalPostcondition		X	X	X	X	X	X	X

ประเภท	สัญลักษณ์	Fundamental	Basic	Intermediate	Complete	Structured	CompleteStructured	ExtraStructured
ParameterSet					X		X	X

2.1.2 **ดีเอสแอล** (Domain Specific Language – DSL)

ดีเอสแอล คือ ภาษาข้อกำหนดที่อยู่ในรูปแบบของข้อความหรือแบบจำลองที่ถูกออกแบบโดยเฉพาะเจาะจงสำหรับโดเมนปัญหาหนึ่งๆ การศึกษาเกี่ยวกับดีเอสแอลเกิดขึ้นจากปัญหาต่าง ๆ ที่เกี่ยวข้องกับตัวต่อประสานระหว่างคอมพิวเตอร์กับโลกแห่งความเป็นจริงที่มีความยุ่งยากในการจัดเตียมข้อมูลดังตัวอย่างการเตรียมไฟล์โครงแบบในภาพที่ 2

```
<stateMachine start = "idle">
  <event name="doorClosed" code="D1CL"/>
  <event name="drawerOpened" code="D2OP"/>
  <event name="lightOn" code="L1ON"/>
  <event name="doorOpened" code="D1OP"/>
  <event name="panelClosed" code="PNCL"/>
  <command name="unlockPanel" code="PNUL"/>
  <command name="lockPanel" code="PNLK"/>
  <command name="lockDoor" code="D1LK"/>
  <command name="unlockDoor" code="D1UL"/>
  <state name="idle">
    <transition event="doorClosed"
        target="active"/>
    <action command="unlockDoor"/>
    <action command="lockPanel"/>
  </state>
  <state name="active">
    <transition event="drawerOpened"
        target="waitingForLight"/>
    <transition event="lightOn"
        target="waitingForDrawer"/>
  </state>
  <state name="waitingForLight">
    <transition event="lightOn"
        target="unlockedPanel"/>
  </state>
  <state name="waitingForDrawer">
    <transition event="drawerOpened"
        target="unlockedPanel"/>
  </state>
  <state name="unlockedPanel">
    <action command="unlockPanel"/>
    <action command="lockDoor"/>
    <transition event="panelClosed"
        target="idle"/>
  </state>
  <resetEvent name = "doorOpened"/>
</stateMachine>
```

```
events
  doorClosed D1CL
  drawerOpened D2OP
  lightOn L1ON
  doorOpened D1OP
  panelClosed PNCL
end
resetEvents
  doorOpened
end
commands
  unlockPanel PNUL
  lockPanel PNLK
  lockDoor D1LK
  unlockDoor D1UL
end
state idle
  actions {unlockDoor lockPanel}
  doorClosed => active
end
state active
  drawerOpened => waitingForLight
  lightOn => waitingForDrawer
end
state waitingForLight
  lightOn => unlockedPanel
end
state waitingForDrawer
  drawerOpened => unlockedPanel
end
state unlockedPanel
  actions {unlockPanel lockDoor}
  panelClosed => idle
end
```

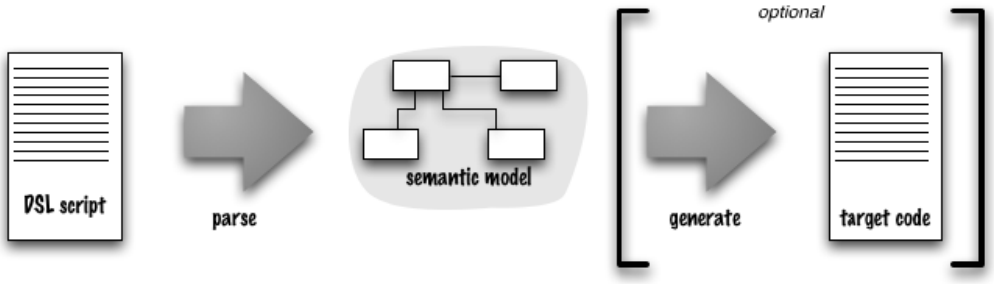
ภาพที่ 2 ตัวอย่างความแตกต่างในการเตรียมไฟล์โครงแบบ แบบปกติ (ซ้าย) กับแบบดีเอสแอล (ขวา)

ลักษณะของดีเอสแอลประกอบด้วยคุณลักษณะสี่ประการ [2] ดังต่อไปนี้

1. “กรอบงานจะต้องมีความชัดเจน” โดยทั่วไปแล้วดีเอสแอลสามารถสร้างขึ้นมาได้ในสองลักษณะคือ หนึ่งใช้คณิตศาสตร์เข้ามาช่วยอธิบาย และอีกอย่างหนึ่งคือการใช้ภาษาทั่วไปเข้ามาช่วยอธิบาย ซึ่งการกำหนดรายละเอียดกรอบนั้น จำเป็นต้องอาศัยผู้เชี่ยวชาญเข้ามาช่วยในการระบุงานที่เกี่ยวข้องทั้งหมด
2. “สัญกรณ์ที่ใช้ต้องชัดเจน” เนื่องจากสัญกรณ์ คือ รูปแบบทางภาษาอย่างหนึ่งที่เป็นตัวกลางในการสื่อสารระหว่างต้นทางกับปลายทางได้ ดังนั้นการกำหนดภาษาที่ดีจะช่วยให้สามารถเข้าใจกันได้อย่างถูกต้องและชัดเจนมากขึ้น ตัวอย่างเช่นในภาษาพูดเราจะใช้เสียงเป็นตัวกลางในการสื่อสาร ส่วนในทางดนตรีเราจะใช้ตัวโน้ตเป็นตัวกลางในการสื่อสาร เป็นต้น
3. “ความหมายแบบไม่เป็นทางการจะต้องชัดเจน” ภาษาที่ใช้ต้องชัดเจนและตรงประเด็น ตัวอย่างเช่น ตัวโน้ตทางดนตรี สัญญาณจราจรต่าง ๆ เป็นต้น โดยภาษาเหล่านี้จะไม่ได้ลงรายละเอียดเชิงลึก แต่เป็นรายละเอียดที่สามารถเข้าใจได้โดยผู้ใช้งานที่เกี่ยวข้อง
4. “ความหมายแบบทางการจะต้องชัดเจนและสามารถทำให้เกิดผลได้” ภาษาที่นำมาใช้จะต้องสามารถนำมาทำให้เกิดผลได้ด้วยซอฟต์แวร์หรือคอมพิวเตอร์ กล่าวคือ จะต้องเป็นภาษาที่เครื่องจักรสามารถเข้าใจได้ เช่น ตัวโน้ตทางดนตรีอาจกำหนดให้อยู่ในรูปแบบของคณิตศาสตร์โดยประกอบด้วยความถี่ ระดับเสียง และระยะเวลา เพื่อใช้ในการสร้างเสียงสำหรับตัวโน้ตนั้น ๆ ออกมา เป็นต้น

สถาปัตยกรรมของดีเอสแอลสามารถแบ่งออกได้เป็นสองแบบ [3] คือ ดีเอสแอลแบบภายใน และดีเอสแอลแบบภายนอก โดยดีเอสแอลแบบภายในจะใช้ในการอธิบายงานตามโดเมนที่กำหนดไว้ซึ่งจะเป็นภาษาที่เกี่ยวข้องกับงานในโดเมนนั้น ๆ ส่วนดีเอสแอลแบบภายนอก คือ การแปลงดีเอสแอลแบบภายในให้อยู่ในภาษาที่คอมพิวเตอร์สามารถนำไปใช้ประโยชน์หรือประมวลผลได้ (โดยส่วนใหญ่ดีเอสแอลแบบภายนอกจะอยู่ในรูปแบบของไฟล์เอ็กซ์เอ็มแอล)

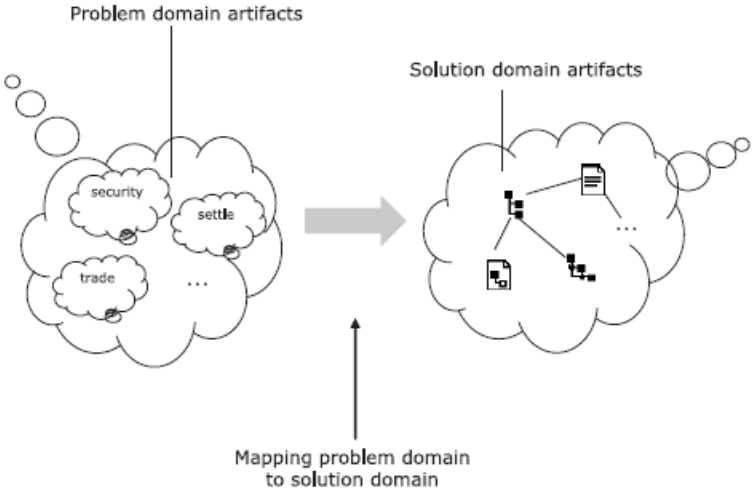
สำหรับการแปลงข้อมูลจากดีเอสแอลแบบภายนำไปยังดีเอสแอลภายนอกสามารถทำได้หลายวิธี แต่โดยทั่วไปแล้วจะใช้ตัวแจงส่วน (Parser) เข้ามาช่วยในการสกัดข้อมูลจากดีเอสแอลภายในให้อยู่ในรูปของแบบจำลองความหมาย (Semantic Model) แล้วจึงแปลงเป็นดีเอสแอลภายนอกเพื่อนำไปใช้งานต่อไป ดังภาพที่ 3



ภาพที่ 3 การจัดเตรียมข้อมูลจากดีเอสแอล

การออกแบบโดเมน [4] เพื่อจะทำดีเอสแอล เป็นการกำหนดกรอบงานและความสัมพันธ์ของกระบวนการที่เกี่ยวข้องกันเพื่อให้สามารถวิเคราะห์หาสิ่งที่เกี่ยวข้องกับโดเมนสำหรับใช้ในการจัดทำดีเอสแอล โดยการออกแบบโดเมนจะเริ่มต้นจากการกำหนดโดเมนปัญหา (Problem Domain) และการกำหนดความสัมพันธ์ระหว่างเอนทิตีต่าง ๆ ภายในโดเมนปัญหานั้น ๆ โดยทั่วไปสิ่งที่ได้จากโดเมนปัญหาจะเป็นคำศัพท์พื้นฐานต่าง ๆ ที่เกี่ยวข้องกับสิ่งของหรือกิจกรรมในโดเมนนั้น ๆ

หลังจากกำหนดโดเมนปัญหาได้แล้ว ขั้นตอนต่อไปคือการกำหนดโซลูชั่นโดเมน (Solution Domain) โดยโซลูชั่นโดเมนสามารถอยู่ในรูปอะไรก็ได้ที่เราต้องการ ตัวอย่างเช่น ถ้าต้องการให้โซลูชั่นโดเมนอยู่ในรูปแบบของระเบียบวิธีเชิงวัตถุ (OOP) ดังนั้นคลาส ออบเจ็กต์ และเมท็อด จะเป็นส่วนประกอบของโซลูชั่นโดเมนดังตัวอย่างภาพที่ 4



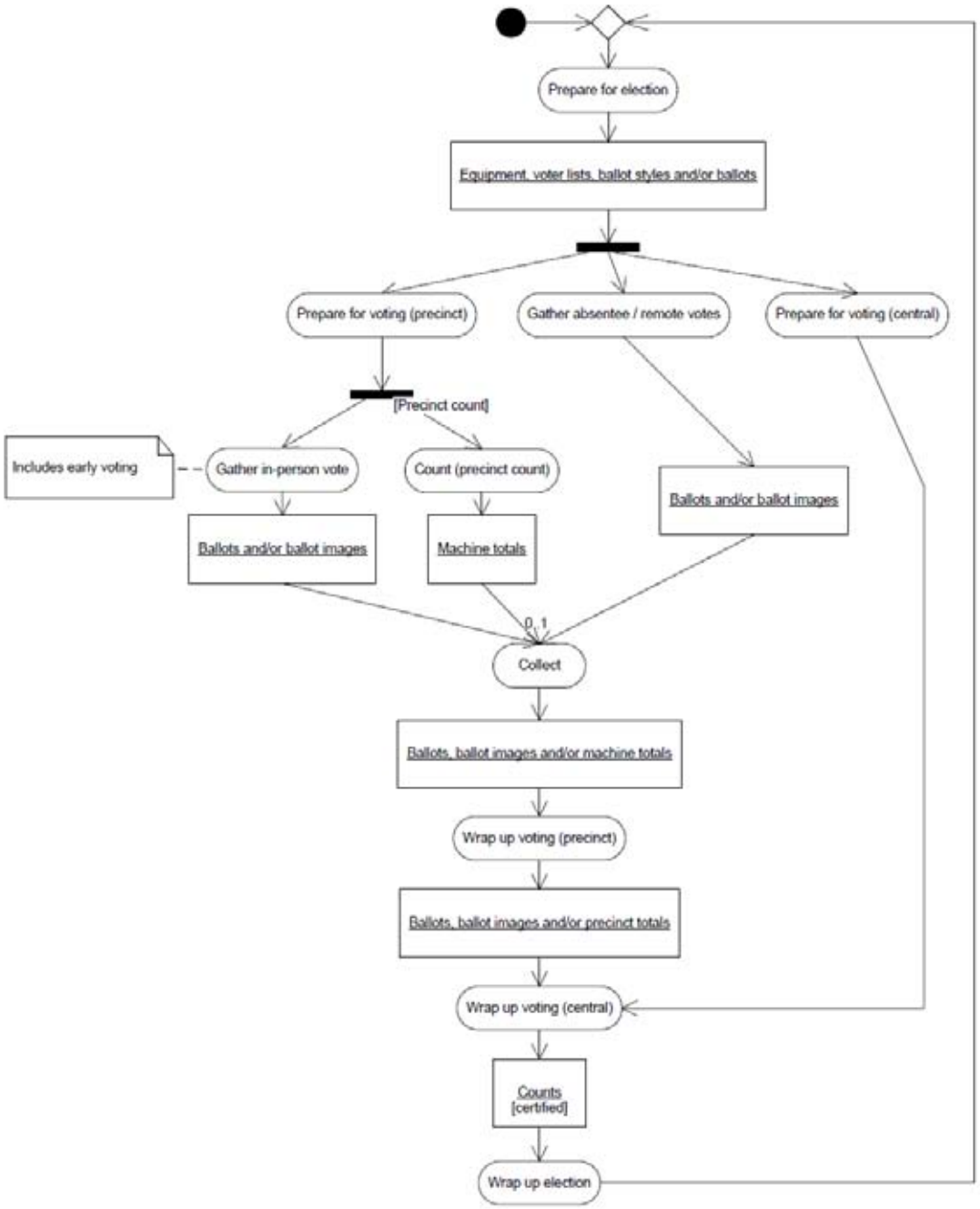
ภาพที่ 4 ตัวอย่างการจำแนกข้อมูลจากโดเมนปัญหาไปยังโดเมนผลเฉลย

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 Rendering UML Activity Diagrams As Human-Readable Text [5]

ในงานวิจัยนี้ได้นำเสนอการอธิบายแผนภาพกิจกรรมในรูปแบบสัญกรณ์ข้อความ ซึ่งสัญ

กรณ์ข้อความดังกล่าวเรียกว่า เอดีแอลเอฟ (Activity Diagram Linear Form - ADLF) เนื่องจากเอดีแอลเอฟอยู่ในรูปแบบของข้อความตัวอักษรจึงทำให้ง่ายต่อการนำไปใช้งานในส่วนอื่น ๆ โดยเฉพาะอย่างยิ่งการจัดทำเอกสาร จากตัวอย่างแผนภาพกิจกรรมในภาพที่ 5 สามารถเขียนในรูปของเอดีแอลเอฟได้ดังแสดงในภาพที่ 6



ภาพที่ 5 ตัวอย่างแผนภาพกิจกรรมจากเอดีแอลเอฟ

```
<InitialNode>-><MergeNode *merge>->("Prepare for election")
  ->["Equipment, voter lists, ballot styles and/or ballots"]-><ForkNode>
    { ->("Prepare for voting (precinct)")-><ForkNode>
        { ->("Gather in-person vote") // Includes early voting.
            ->["Ballots and/or ballot images"]->(Collect *c),
          "Precinct count"->("Count (precinct count)")
            ->["Machine totals"]->0..1(*c)
        },
      ->("Gather absentee / remote votes")->["Ballots and/or ballot images"]
        ->(*c),
      ->("Prepare for voting (central)")->("Wrap up voting (central)" *w)
    };

(*c)->["Ballots, ballot images and/or machine totals"]
  ->("Wrap up voting (precinct)")
    ->["Ballots, ballot images and/or precinct totals"]->(*w)
      ->["Counts" state=certified]->("Wrap up election")-><*merge>.
```

ภาพที่ 6 ตัวอย่างเอดีแอลเอฟ

อย่างไรก็ตามเอดีแอลเอฟคงมุ่งเน้นเฉพาะการออกแบบแผนภาพกิจกรรมในรูปของข้อความเท่านั้น ซึ่งยังไม่มีการป้องกันความผิดพลาดเนื่องมาจากการร้อยกระบวนผิด และการออกแบบแผนภาพกิจกรรมผิดไปจากมาตรฐาน ทำให้ไม่สามารถรับรองได้ว่าแผนภาพกิจกรรมที่สร้างขึ้นนั้นจะมีความถูกต้องของมากน้อยเพียงใด

2.2.2 The Formal Semantics of UML Activity Diagram Based on Process Algebra [6]

ในการสร้างแผนภาพกิจกรรมนั้นจะพบว่าจะเกิดปัญหาหลายอย่างขึ้น โดยเฉพาะอย่างยิ่งการอธิบายกระบวนการที่ไม่สมบูรณ์ คลุมเครือ หรือมีความไม่สอดคล้องต้องกันภายในแผนภาพกิจกรรม งานวิจัยนี้จึงได้นำเสนอการวิเคราะห์ความหมายของแผนภาพกิจกรรมเชิงรูปนัย ซึ่งประกอบด้วยคำนิยามโครงสร้าง กฎ และการทำงาน โดยการวิเคราะห์จะใช้กระบวนการเชิงพีชคณิตในการตรวจสอบความถูกต้องของแผนภาพกิจกรรม

ขอบเขตของแผนภาพกิจกรรมที่งานวิจัยนี้ได้นำเสนอ แบ่งออกเป็นสองส่วน คือ ส่วนของบัพ (Node) และส่วนของเส้นเชื่อมกิจกรรม (Edge) ซึ่งในแต่ละบัพจะประกอบด้วย สถานะ (State) เส้นทาง (Route) ขอบเขต (Border) และ กิจกรรม (Activity) สำหรับเส้นเชื่อมกิจกรรมจะแบ่งการวิเคราะห์ออกเป็นสองลักษณะ คือ แบบธรรมดา (Simple) และแบบที่มีความซับซ้อน (Complex) โดยแบบธรรมดา คือ มีเส้นเชื่อมเข้าและออกอย่างละหนึ่งเส้น ส่วนแบบที่มีความซับซ้อน คือ มีเส้นเชื่อมเข้าหรือเส้นเชื่อมออกมากกว่าหนึ่งเส้น

นิยามโครงสร้างแผนภาพกิจกรรมมีรายละเอียดดังนี้

1. Nodes = SN U BN = AS U WS U {Initial, Final} ; AS U WS ≠ Ø, AS ∩ WS = Ø
   โดยที่ AS คือ Activity State, WS คือ Wait State, SN = AS U WS คือ State Node

และ BN = {Initial, Final} คือ เซตของปลายกิจกรรม

2. Activity = (ID, Role, State, Rdata, Time) โดยที่ ID คือ ชื่อของกิจกรรม, Role คือ ผู้ที่ทำหน้าที่ดำเนินการกิจกรรม, State คือ สถานะของกิจกรรมว่าได้ดำเนินการเสร็จสิ้นแล้วหรือยัง, Rdata คือ ข้อมูลอ้างอิงต่าง ๆ ที่จะต้องนำมาใช้ในกิจกรรม, Time คือ เวลาที่ต้องใช้ในการดำเนินกิจกรรม
3. ev = (ID, Time, Type, Origin) โดยที่ ID คือ ชื่ออ้างอิงของแต่ละเหตุการณ์, Time คือ เวลา, Type คือ ประเภท, Origin คือ ต้นทางของกิจกรรม
4. Events = CE U ME U TE U CCE U EE โดยที่ CE คือ Completion Event, ME คือ Message Event, TE คือ Time Event, CCE คือ Codition Change Event, EE คือ Exception Event, CEvents = Current Events, CEvents $\subset$ Events
5. a = (ID, ev, c, Content), a $\in$ Actions โดยที่ ID คือ ชื่อของแอ็คชั่น, ev คือ เหตุการณ์ที่เป็นตัวกระตุ้นให้แอ็คชั่นทำงาน, c คือ เงื่อนไขในการดำเนินการ, Content คือ รายละเอียดของการดำเนินการแอ็คชั่นว่ามีการเปลี่ยนแปลงสถานะอะไรบ้างและมีการเริ่มทำงานของกิจกรรมอะไรบ้าง
6. Condition ของเส้นเชื่อมต่าง ๆ จะต้องถูกกำหนดอยู่ในรูปแบบของนิพจน์บูลีน
7. Edges = Nodes X Rules X Nodes = Nodes X Events X Conditions X Actions X Nodes โดยที่ Event (e) = ev $\in$ Events, Condition (e) = c $\in$ Conditions, Action (e) = a $\in$ Actions, e = (sn, ev, c, a, tn), source (e) = sn $\in$ Nodes, target (e) = tn $\in$ Nodes
8. Configuration = AS U WS U BN U CEvents โดยที่ Configuration คือ ตัวอธิบายสถานะของกิจกรรม ณ เวลาหนึ่ง ๆ
9. AD = (Nodes, Edges, Rules, Configuration)

นอกจากนี้ทางผู้เขียนยังได้สร้างกฎขึ้นมาเพื่อให้มั่นใจว่าแผนภาพกิจกรรมจะสามารถดำเนินการได้อย่างถูกต้อง โดยแต่ละแอ็คชั่นจะถูกกำกับด้วยกฎ r = (ev, c, a) ดังภาพที่ 7 และได้กำหนดเงื่อนไขของกิจกรรมเบื้องต้นดังต่อไปนี้

1. $(\forall e \in Edges) \Rightarrow (\{Initial\} \in target(e)) \wedge (\{Final\} \notin source(e))$
2. $(\forall e \in Edges) \wedge (\{Initial\} \in source(e)) \Rightarrow (source(e) = \{Initial\})$
3. $(\forall e \in Edges) \wedge (\{Final\} \in target(e)) \Rightarrow (target(e) = \{Initial\})$
4. $(\forall e \in Edges) \Rightarrow VALUE(Condition(e)) = TRUE$
5. $(\forall e \in Edges) \wedge (source(e) = \{Initial\}) \Rightarrow (event(e) = NULL) \wedge$

(VALUE(Condition(e)) = TRUE)

เงื่อนไขข้อ 1-3 เป็นการกำหนดให้กิจกรรมมีจุดเริ่มต้นเพียงหนึ่งจุดและจุดสิ้นสุดกิจกรรมเพียงหนึ่งจุดเท่านั้น สำหรับเงื่อนไขข้อ 4 เป็นการกำหนดให้ผลของเงื่อนไขบนเส้นเชื่อมกิจกรรมซึ่งจะต้องมีนิพจน์เป็นความจริงเท่านั้น กิจกรรมจึงจะสามารถดำเนินการต่อได้ สำหรับเงื่อนไขข้อ 5 เป็นการกำหนดเส้นทางออกจากจุดเริ่มต้นซึ่งจะต้องไม่ใช่เส้นเหตุการณ์และเส้นเชื่อมจะต้องสามารถเดินทางผ่านได้

```
WHEN HAPPENED (ev)
      IF VALUE(c) = TRUE THEN
            DO a
      END IF
END WHEN
```

ภาพที่ 7 กฎการดำเนินงานของแอ็คชั่น

สำหรับการดำเนินงานของเหตุการณ์จะถูกนิยามด้วยกฎดังภาพที่ 8

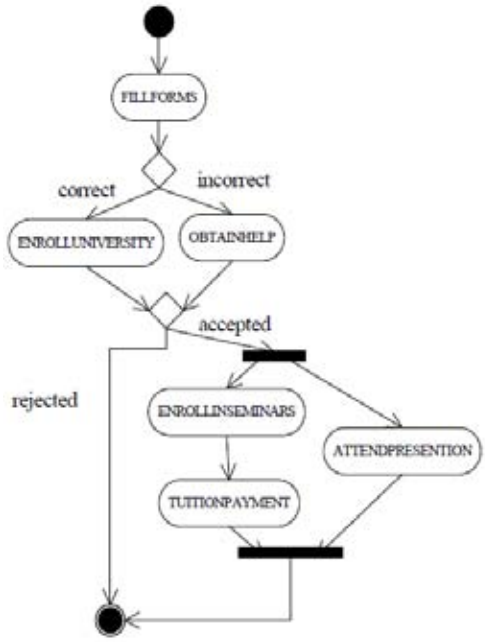
```
∀AD = (Nodes, Edges, Rules, Configuration)
IF e ∈ Edges AND Configuration(e)
WHEN HAPPENED (Event(e))
      Configuration + {Event(e)}
      IF VALUE(Condition(e)) = TRUE THEN
            DO a (a={Taken(e) AND Configuration = {source(e),
            Event(e)} + {Target(e)}})
      END IF
END WHEN
END IF
```

ภาพที่ 8 กฎการดำเนินงานของเหตุการณ์

อย่างไรก็ตามงานวิจัยชิ้นนี้เป็นการเพิ่มกฎให้แผนภาพกิจกรรม เพื่อควบคุมให้แผนภาพกิจกรรมมีความหมายถูกต้องเท่านั้น ซึ่งพบว่ากฎบางอย่างนั้นขัดแย้งกับความเป็นจริง เช่น การกำหนดให้จุดสิ้นสุดกิจกรรมมีเพียงจุดเดียว ซึ่งในความเป็นจริงนั้นแผนภาพกิจกรรมอาจมีได้ทั้งจุดสิ้นสุดกิจกรรมและจุดสิ้นสุดสายงาน ซึ่งในงานวิจัยนี้ไม่ได้ทำการพิจารณาถึงจุดหยุดสายงานเข้าไปในทฤษฎีด้วย

2.2.3 Model Checking UML Activity Diagrams in FDR [7]

เนื่องจากแผนภาพกิจกรรมของยูเอ็มแอลมีการนำไปใช้งานอย่างแพร่หลาย แต่ยังขาดการวิเคราะห์การตรวจสอบอย่างมีแบบแผน ดังนั้นงานวิจัยนี้จึงได้นำเสนอการแปลงแผนภาพกิจกรรมให้อยู่ในรูปแบบของซีเอสพี (Communicating Sequential Process- CSP) [8] เพื่อให้สามารถนำไปวิเคราะห์ในโปรแกรมเอฟดีอาร์ได้ดังภาพที่ 9 โดยโปรแกรมเอฟดีอาร์จะทำการวิเคราะห์ความสัมพันธ์ที่เกี่ยวข้องกับลำดับการทำงานของกิจกรรมได้




```
channel incorrect, correct, accepted, rejected
P1 = incorrect -> OBTAINHELP -> ENROLLUNIVERSITY
P2 = correct -> ENROLLUNIVERSITY
P0 = P1 [] P2
P3 = FILLFORMS -> P0
P4 = ATTENDPRESENTION [{}||{}] (ENROLLINSEMINARS ->
TUITIONPAYMENT)
P5 = accepted -> P4 -> SKIP
P6 = rejected -> SKIP
P7 = P5|~|P6
ENROLLMENT = P3 -> P7
assert P0 [T= P1
assert P0 [T= P2
assert ENROLLMENT [T= P3
```

ภาพที่ 9 ตัวอย่างการแปลงแผนภาพกิจกรรมให้อยู่ในรูปแบบซีเอสพีเอ็ม

จากงานวิจัยชิ้นนี้ทำให้เห็นว่าเราสามารถที่จะตรวจสอบแผนภาพกิจกรรมได้ด้วยเครื่องมือที่มีอยู่แล้วแต่ว่าการจัดเตรียมข้อมูลสำหรับวิเคราะห์นั้นยังทำได้ยาก เนื่องจากมาตรฐานและรูปแบบการจัดเก็บยังคงขาดการจัดการและการควบคุมมาตราฐานที่ดี ตัวอย่างเช่น ในงานวิจัยชิ้นนี้ใช้ข้อมูลจากเอ็กซ์เอ็มไอมาแปลงให้อยู่ในรูปแบบข้อมูลซีเอสพีเอ็มเพื่อที่จะสามารถนำไปวิเคราะห์ในโปรแกรมเอฟดีอาร์ได้ แต่พบว่าการจัดเก็บข้อมูลเอ็กซ์เอ็มไอของแผนภาพกิจกรรมในปัจจุบันนั้น ในแต่ละโปรแกรมอาจใช้รูปแบบการจัดเก็บที่แตกต่างกันจึงทำให้ยากต่อการนำเข้าข้อมูลไฟล์เอ็กซ์เอ็มไออย่างมีประสิทธิผล

บทที่ 3
วิธีดำเนินการวิจัย

3.1 แนวคิดในการพัฒนา

งานวิจัยนี้จะเป็นการนำเสนอการสร้างข้อกำหนดรูปนัยในลักษณะของดีเอสแอล เพื่อนำมาใช้ในการสร้างแบบจำลองความหมายของแผนภาพกิจกรรม โดยการพัฒนาจะเริ่มต้นจากการออกแบบเอดีแอลสำหรับใช้ในการอธิบายแผนภาพกิจกรรม ซึ่งการออกแบบจะเน้นการพัฒนาภาษาให้เหมาะสมกับแผนภาพกิจกรรมในแต่ละกรอบงาน ซึ่งประกอบด้วยแผนภาพกิจกรรมระดับรากฐาน แผนภาพกิจกรรมระดับพื้นฐาน และแผนภาพกิจกรรมระดับปานกลาง ตามลำดับ หลังจากนั้นจึงนำภาษาที่ได้มากำหนดเป็นวากยสัมพันธ์สำหรับใช้ในการแจงส่วน ซึ่งการแจงส่วนจะเป็นการแปลงข้อมูลจากบทคำสั่งที่เขียนด้วยเอดีแอลเป็นแบบจำลองความหมายของเอดีแอล โดยแบบจำลองความหมายของเอดีแอลจะประกอบด้วยวัตถุและความสัมพันธ์ต่าง ๆ ที่เกี่ยวข้องกับวัตถุนั้น ๆ หลังจากนั้นจึงทำการสกัดข้อมูลจากแบบจำลองความหมายดังกล่าว เพื่อสร้างแบบจำลองความหมายสำหรับสร้างแผนภาพกิจกรรม (ตัวสร้างแผนภาพกิจกรรม) โดยการแปลงแบบจำลองความหมายต่าง ๆ จะเลือกใช้คิววีทีเชิงปฏิบัติการ (Operational QVT) [9] ในการอธิบายรายละเอียดขั้นตอนการแปลงแบบจำลองความหมาย

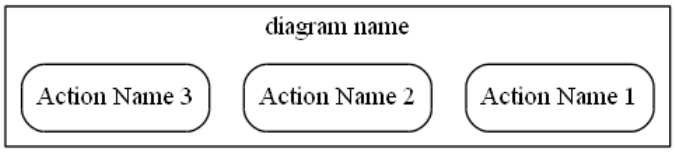
3.2 เอดีแอลสำหรับแผนภาพกิจกรรม

การออกแบบเอดีแอลจะเริ่มทำการออกแบบจากแผนภาพกิจกรรมระดับรากฐาน แผนภาพกิจกรรมระดับพื้นฐาน และแผนภาพกิจกรรมระดับปานกลาง ตามลำดับ ทั้งนี้เพื่อให้มั่นใจได้ว่าภาษาที่ออกแบบนั้นมีคุณสมบัติความเข้ากันได้ย้อนหลัง ซึ่งจะช่วยให้สามารถสร้างแบบจำลองความหมายของแผนภาพกิจกรรมในระดับต่าง ๆ ได้

3.2.1 แผนภาพกิจกรรมระดับรากฐาน

แผนภาพกิจกรรมระดับรากฐานจะเป็นการอธิบายว่ากิจกรรมประกอบด้วยแอ็คชั่นอะไรบ้างเท่านั้น ซึ่งยังไม่มีการแสดงข้อมูลในส่วนของเส้นเชื่อมหรือสายงานแต่อย่างใด (ไม่มีลำดับการทำงาน) ดังนั้นการออกแบบภาษาสำหรับใช้ในการอธิบายแผนภาพกิจกรรม จึงอยู่ที่การสร้างตัวบรรจุสำหรับบรรจุแอ็คชั่นต่าง ๆ เพื่อความสะดวกในการสร้างแอ็คชั่นดังกล่าวและการป้องกันข้อผิดพลาดอันเนื่องมาจากการตัดคำ เราจะใช้ลักษณะการสร้างชื่อแอ็คชั่นแบบคาเมลเคส (Camel Case) ดังตัวอย่างภาพที่ 10

```
diagram 'diagram name'
      actionName1
      actionName2
      actionName3
end
```



ภาพที่ 10 ตัวอย่างเอดีแอลและผลลัพธ์สำหรับแผนภาพกิจกรรมระดับรากฐาน

3.2.2 แผนภาพกิจกรรมระดับพื้นฐาน

แผนภาพกิจกรรมระดับพื้นฐานจะเป็นการกำหนดลำดับขั้นตอนการทำงานของแอ็คชั่นที่สร้างขึ้นในกิจกรรมระดับรากฐาน โดยลำดับการทำงานจะถูกกำหนดในลักษณะของเส้นเชื่อมกิจกรรม นอกจากการกำหนดลำดับการทำงานของแอ็คชั่นแล้ว แผนภาพกิจกรรมระดับนี้ยังมีการแสดงจุดเริ่มต้นและจุดสิ้นสุดของกิจกรรมอีกด้วย

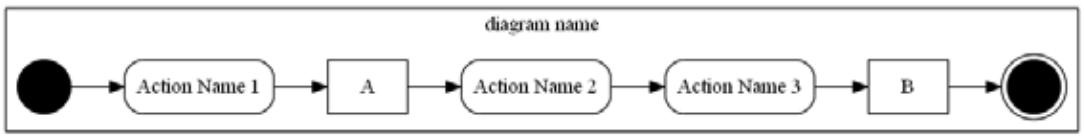
สำหรับการอธิบายลำดับขั้นตอนการทำงานเราจะใช้เครื่องหมาย “->” หรือ “then” ซึ่งจะเป็นการอธิบายลำดับการทำงานจากซ้ายไปขวา สาเหตุที่เราไม่เลือกใช้ลำดับการสร้างแอ็คชั่นเป็นตัวกำหนดลำดับการทำงาน เพราะว่าจะส่งผลให้ภาษาขาดความยืดหยุ่นทำให้ไม่สามารถอธิบายลำดับการทำงานที่ซับซ้อนได้ ดังตัวอย่างภาพที่ 10 หากกิจกรรมทำงานจากบนลงล่างจะทำให้ไม่สามารถอธิบายการทำงานจาก “actionName3” ไป “actionName2” หรือ “actionName1” ได้

สำหรับจุดเริ่มต้นและจุดสิ้นสุดของกิจกรรมนั้น สามารถกำหนดได้สองวิธี คือ แบบกำหนดด้วยตนเองและแบบอัตโนมัติ โดยทั้งสองวิธีนี้เราจะทำการกำหนดชื่อสำหรับอ้างอิงจุดเริ่มต้นและจุดสิ้นสุดกิจกรรม คือ “initiate” และ “terminate” ตามลำดับ ดังนั้นสำหรับการกำหนดด้วยตนเองเราจะใช้วิธีเช่นเดียวกันกับการอธิบายลำดับขั้นตอนการทำงาน ตัวอย่างเช่น เราสามารถกำหนด “initiate->actionName1” เพื่อระบุว่า “actionName1” คือจุดเริ่มต้น เป็นต้น สำหรับแบบอัตโนมัติระบบจะทำการค้นหาแอ็คชั่นที่ไม่เป็นปลายทางของแอ็คชั่นใด ๆ เพื่อผูกแอ็คชั่นเหล่านั้นกับ “initiate” เช่นเดียวกันกับจุดสิ้นสุดกิจกรรมเราจะสามารถหาจุดสิ้นสุดกิจกรรมได้จากแอ็คชั่นที่ไม่เป็นต้นทางของแอ็คชั่นใด ๆ เพื่อผูกแอ็คชั่นเหล่านั้นกับ “terminate”

นอกจากการกำหนดลำดับการทำงานแล้วแผนภาพกิจกรรมในระดับนี้ ยังมีความสามารถในการแสดงให้เห็นถึงวัตถุที่เกี่ยวข้องแอ็คชั่นนั้น ๆ ด้วย ดังนั้นการออกแบบเอดีแอลในระดับนี้ควรมีการกำหนดไวยากรณ์ภาษาให้รองรับการแสดงความสัมพันธ์ทั้งแบบแสดงวัตถุระหว่างกิจกรรม

และไม่แสดงวัตถุ โดยชื่อของวัตถุเราจะใช้กำหนดชื่อแบบปาสคาล (Pascal Case) เพื่อป้องกันความสับสนระหว่างชื่อวัตถุและชื่อแอ็คชั่น สำหรับตัวอย่างของเอดีแอลที่ใช้อธิบายกิจกรรมในระดับนี้จะแสดงในภาพที่ 11

```
diagram 'diagram name'
      action actionName1
            outputs A
      end
      action actionName3
            outputs B
      end
      actionName1->actionName2->actionName3
end
```



ภาพที่ 11 ตัวอย่างเอดีแอลและผลลัพธ์สำหรับแผนภาพกิจกรรมระดับพื้นฐาน

3.2.3 แผนภาพกิจกรรมระดับปานกลาง

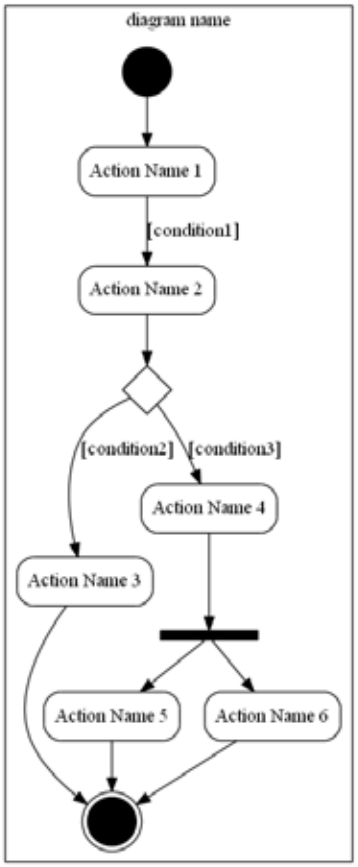
แผนภาพกิจกรรมระดับปานกลางจะเป็นการกำหนดส่วนแสดงพฤติกรรมของระบบ ซึ่งประกอบด้วยการทำงานคู่ขนาน การทำงานแบบมีเงื่อนไข การรวมสายงานแบบประสานเวลา การรวมสายงานแบบไม่ประสานเวลา และการหยุดสายงาน

การอธิบายการทำงานคู่ขนานเราสามารถอธิบายได้จากลำดับสายงานที่กำหนดขึ้น ตัวอย่างเช่น ถ้าต้องการให้ “actionName2” และ “actionName3” ทำงานพร้อมกันหลังจาก “actionName1” ทำงานเสร็จ สามารถกำหนดได้โดยการสร้างความสัมพันธ์ “actionName1-> actionName2” และ “actionName1-> actionName3” เป็นต้น อย่างไรก็ตามเพื่อความสะดวกในการใช้งานและการอ่านลำดับการทำงานจากเอดีแอล ดังนั้นเราจึงควรเพิ่มไวยากรณ์ให้สามารถอ่านเข้าใจได้ง่ายขึ้น ในที่นี้เราจะใช้ “and” เข้ามาช่วยในการอธิบายความสัมพันธ์ของแอ็คชั่นที่เกิดขึ้นพร้อมกัน ดังนั้นจากตัวอย่างข้างต้นเราสามารถอธิบายความสัมพันธ์ดังกล่าวด้วย “actionName1-> actionName2 and actionName3”

สำหรับการทำงานแบบมีเงื่อนไข เราจะใช้การกำกับข้อความบนเส้นเชื่อมกิจกรรมเพื่อแสดงเงื่อนไขที่ยอมรับให้แอ็คชั่นต่อไปสามารถทำงานได้ ซึ่งการกำกับข้อความจะใช้เครื่องหมายก้ามปู “[ ]” กำกับหลังการกำหนดลำดับการทำงาน เช่น ถ้าต้องการให้ “actionName2” จะทำงาน

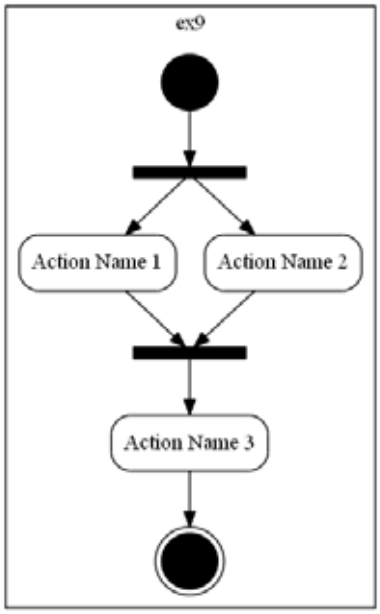
หลังจาก “actionName1” โดยจะทำงานได้ก็ต่อเมื่อ “condition” ดังนั้นความสัมพันธ์ระหว่าง “actionName1” และ “actionName2” สามารถกำหนดได้ด้วย “actionName1->actionName2 [‘condition’]” อย่างไรก็ตามพบว่าการอธิบายเงื่อนไขเช่นนี้ไม่เพียงพอต่อการอธิบายการตัดสินใจแบบเงื่อนไขซ้อนใน (Nested Condition) ได้ ดังนั้นเพื่อให้รองรับรูปแบบการตัดสินใจดังกล่าวเราจึงได้เลือกใช้ “if...then...else...endif” เข้ามาประยุกต์ใช้กับเอดีแอลเพื่อให้อธิบายการตัดสินใจแบบเงื่อนไขซ้อนในได้ ดังตัวอย่างภาพที่ 12

```
diagram ‘diagram name’
      actionName1->actionName2 [‘condition1’]
      decision from actionName2
            if ‘condition2’ then actionName3
            else
                  if ‘condition3’
                  then actionName4
                  endif
            endif
      end
      actionName4->actionName5 and actionName6
end
```



ภาพที่ 12 ตัวอย่างการตัดสินใจแบบเงื่อนไขซ้อนใน

ส่วนการรวมสายงานทั้งแบบประสานเวลาและไม่ประสานเวลานั้น เราจะสามารถสร้างได้จากการกำหนดลำดับของแอ็คชั่น โดยให้แอ็คชั่นมากกว่าหนึ่งแอ็คชั่นมีลำดับการทำงานต่อไปเป็นแอ็คชั่นเดียวกัน ตัวอย่างเช่น ถ้าต้องการให้ “actionName3” รวมสายงานจาก “actionName1” และ “actionName2” เข้าด้วยกัน เราจะสามารถกำหนดได้ด้วย “actionName1->actionName3” และ “actionName2->actionName3” ซึ่งจะทำให้ได้ผลลัพธ์ดังภาพที่ 13

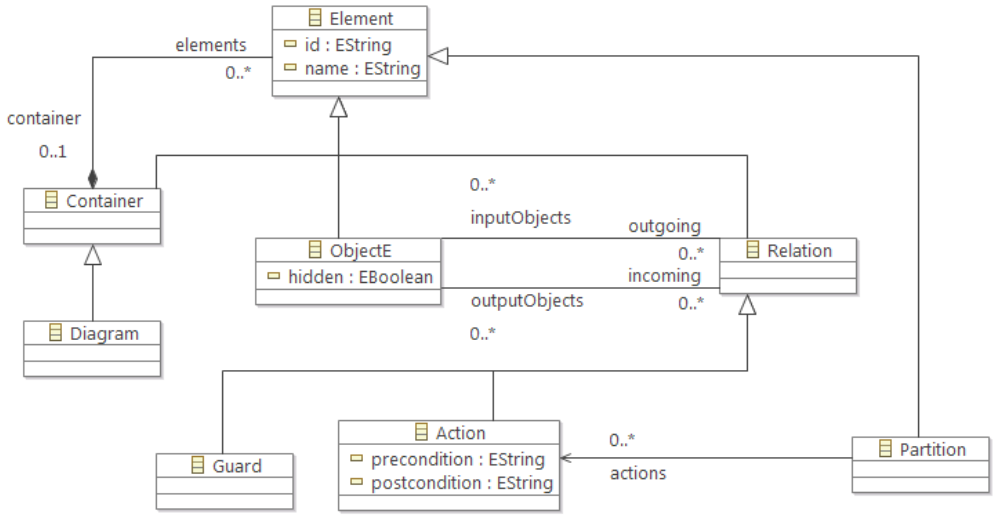


ภาพที่ 13 ตัวอย่างแผนภาพกิจกรรมที่มีการรวมสายงาน

3.3 การสร้างแบบจำลองความหมายของเอดีแอล

แบบจำลองความหมายของเอดีแอลประกอบด้วยวัตถุและความสัมพันธ์ระหว่างวัตถุ โดยความสัมพันธ์จะสามารถแบ่งออกได้เป็นสองแบบ คือ แบบปกติ และแบบมีเงื่อนไข โดยความสัมพันธ์แบบปกติจะเปรียบเสมือนกับแอ็คชั่น ส่วนความสัมพันธ์แบบมีเงื่อนไขจะเปรียบเสมือนกับตัวป้องกัน (Guard) ดังนั้นแอ็คชั่นและตัวป้องกันในแบบจำลองความหมายของเอดีแอลจะทำหน้าที่เป็นตัวเปลี่ยนสถานะของวัตถุเท่านั้น

เนื่องจากวัตถุสามารถสร้างขึ้นมาได้สองแบบ คือ แบบชัดแจ้ง และแบบโดยนัย โดยวัตถุแบบชัดแจ้งจะถูกสร้างขึ้นมาจากการระบุอินพุตหรือเอาต์พุตของแอ็คชั่น ดังตัวอย่างภาพที่ 11 “A” และ “B” เกิดจากการถูกระบุเป็นเอาต์พุตของ “actionName1” และ “actionName3” ตามลำดับ ส่วนวัตถุโดยนัยจะเกิดขึ้นจากการสร้างความสัมพันธ์โดยตรงหรือไม่มีการกำหนดอินพุตและเอาต์พุตให้กับแอค็คชั่น โดยเมื่อเราตรวจพบความสัมพันธ์ในลักษณะดังกล่าววัตถุโดยนัยจะถูกสร้างขึ้นมาเพื่อเป็นเอาต์พุตของแอ็คชั่นต้นทาง และเป็นอินพุตของแอ็คชั่นปลายทาง ซึ่งความสัมพันธ์ดังกล่าวจะรวมถึงตัวป้องกันด้วย ดังภาพที่ 14

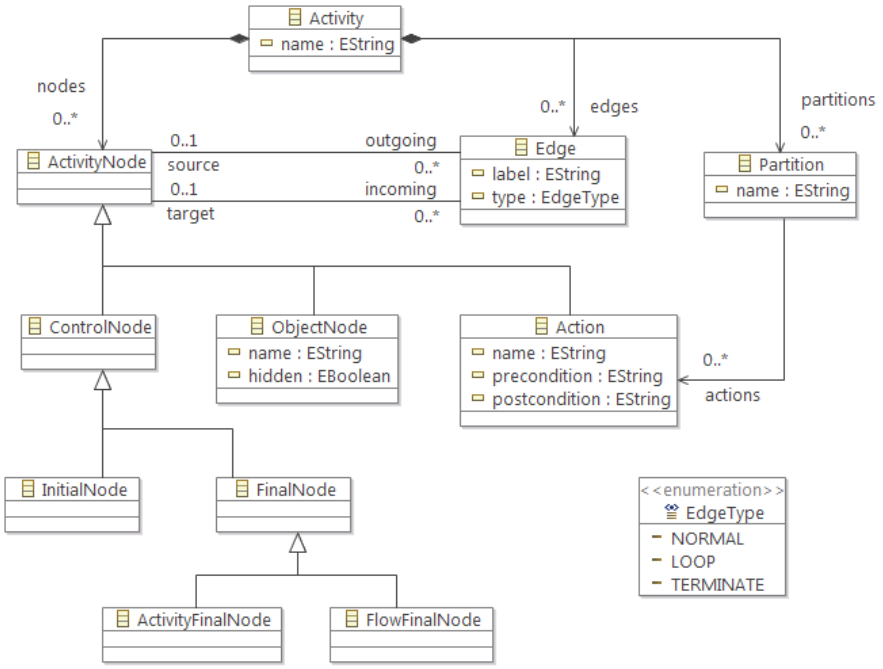


ภาพที่ 14 เมทาโมเดลของเอดีแอล

3.4 การแปลงจากแบบจำลองความหมายเอดีแอลสู่ตัวสร้างแผนภาพกิจกรรม

แบบจำลองสำหรับการสร้างแผนภาพกิจกรรมหรือตัวสร้างแผนภาพกิจกรรม ทำหน้าที่ลดความซับซ้อนและเพิ่มความสะดวกในการแปลงแผนภาพกิจกรรมระดับต่าง ๆ โดยตัวสร้างแผนภาพกิจกรรมจะถูกสร้างขึ้นจากการแปลงข้อมูลจากแบบจำลองความหมายของเอดีแอล บนพื้นฐานของเมทาโมเดลภาพที่ 15 โดยมีรายละเอียดการแปลงแบบโดยสังเขปดังนี้

- Diagram :: Activity แปลงจากกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นกิจกรรม
- ObjectE :: ObjectNode แปลงจากวัตถุของกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นวัตถุในกิจกรรม
- Action :: Action แปลงจากแอ็คชั่นของกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นแอ็คชั่นในกิจกรรม
- Action :: Edge แปลงจากแอ็คชั่นของกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นเส้นเชื่อมกิจกรรมในกิจกรรม
- Guard :: Edge แปลงจากแอ็คชั่นของกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นเส้นเชื่อมกิจกรรมแบบมีชื่อกำกับในกิจกรรม
- Partition :: Partition แปลงจากตัวแบ่งกลุ่มของกิจกรรมในแบบจำลองความหมายของเอดีแอล เป็นตัวแบ่งกลุ่มในกิจกรรม โดยแอ็คชั่นที่ตัวแบ่งกลุ่มอ้างอิงถึงจะต้องเป็นแอ็คชั่นที่มีตัวตนอยู่จริงในแผนภาพกิจกรรมนั้น ๆ



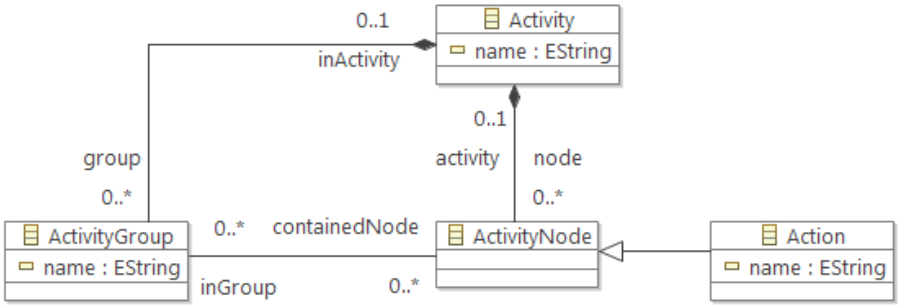
ภาพที่ 15 เมทาโมเดลของตัวสร้างแผนภาพกิจกรรม

3.5 การสร้างแผนภาพกิจกรรมระดับรากฐาน

แบบจำลองความหมายของแผนภาพกิจกรรมระดับรากฐานเกิดจากการแปลงข้อมูลจากตัวสร้างแผนภาพกิจกรรมบนพื้นฐานของเมทาโมเดลภาพที่ 16 โดยมีรายละเอียดการแปลงแบบดังต่อไปนี้

- Activity :: Activity แปลงจากกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นกิจกรรม
- Action :: Action แปลงจากแอ็คชั่นของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นแอ็คชั่นในกิจกรรม

เนื่องจากตัวสร้างแผนภาพกิจกรรมมีข้อมูลที่เพียงพอต่อการสร้างแผนภาพกิจกรรมระดับรากฐานอยู่แล้ว ดังนั้นจึงสามารถทำการแปลงแบบหนึ่งต่อหนึ่งได้เลย โดยไม่ต้องมีการสกัดข้อมูลเพิ่มเติมแต่อย่างใด



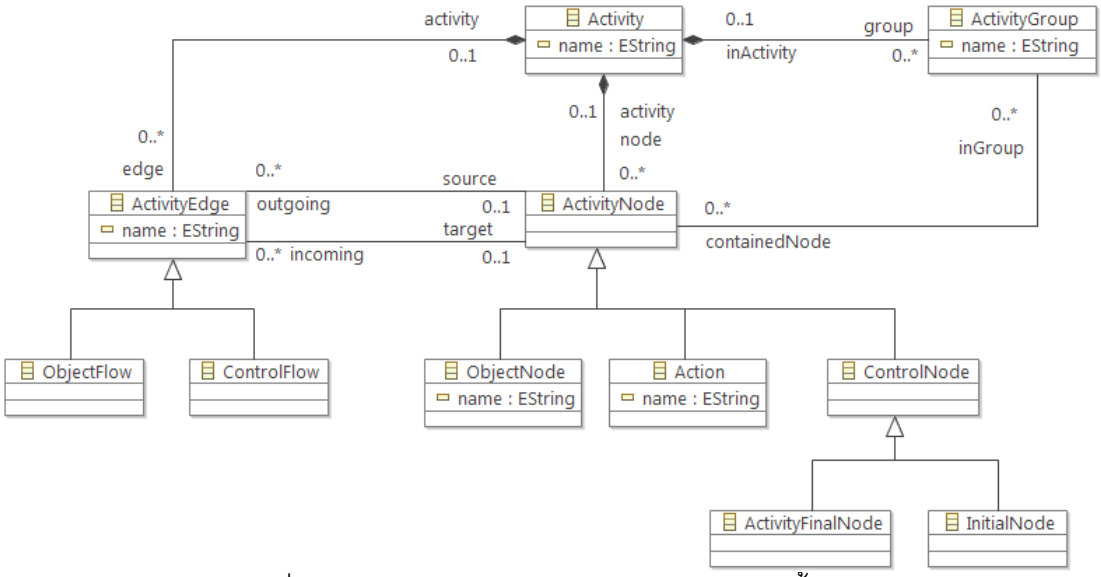
ภาพที่ 16 เมทาโมเดลของแผนภาพกิจกรรมระดับรากฐาน

3.6 การสร้างแผนภาพกิจกรรมระดับพื้นฐาน

แบบจำลองความหมายของแผนภาพกิจกรรมระดับพื้นฐานสามารถสร้างได้จากการแปลงแบบจากตัวสร้างแผนภาพกิจกรรม บนพื้นฐานของเมทาโมเดลภาพที่ 17 ซึ่งเมทาโมเดลทั้งสองมีความคล้ายคลึงกันมาก จะแตกต่างกันตรงที่แผนภาพกิจกรรมระดับพื้นฐานไม่มีการแสดงวัตถุโดยนัยเท่านั้น ดังนั้นการแปลงแบบมีรายละเอียดการแปลงดังต่อไปนี้

- Activity :: Activity แปลงจากกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นกิจกรรม
- ObjectNode :: ObjectNode แปลงจากวัตถุของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นวัตถุในกิจกรรม โดยวัตถุที่ถูกสร้างขึ้นมาจะใช้เฉพาะวัตถุแบบชัดแจ้งเท่านั้น เนื่องจากวัตถุโดยนัยจะไม่ถูกนำมาแสดงในแผนภาพกิจกรรม
- Action :: Action แปลงจากแอ็คชั่นของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นแอ็คชั่นในกิจกรรม
- InitialNode :: InitialNode แปลงจากจุดเริ่มต้นของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นจุดเริ่มต้นในกิจกรรม
- FinalNode :: ActivityFinalNode แปลงจากจุดสิ้นสุดของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นจุดสิ้นสุดในกิจกรรม โดยจุดสิ้นสุดสายงานจะถือเป็นจุดสิ้นสุดกิจกรรมด้วย

สำหรับการสร้างเส้นเชื่อมนั้นเนื่องจากไม่สามารถแปลงข้อมูลได้โดยตรงเพราะจะต้องมีการข้ามวัตถุโดยนัย ดังนั้นการสร้างเส้นเชื่อมจึงใช้การแวะผ่านไปยังแต่ละบัพโดยเริ่มจากจุดเริ่มต้นกิจกรรม



ภาพที่ 17 เมทาโมเดลของแผนภาพกิจกรรมระดับพื้นฐาน

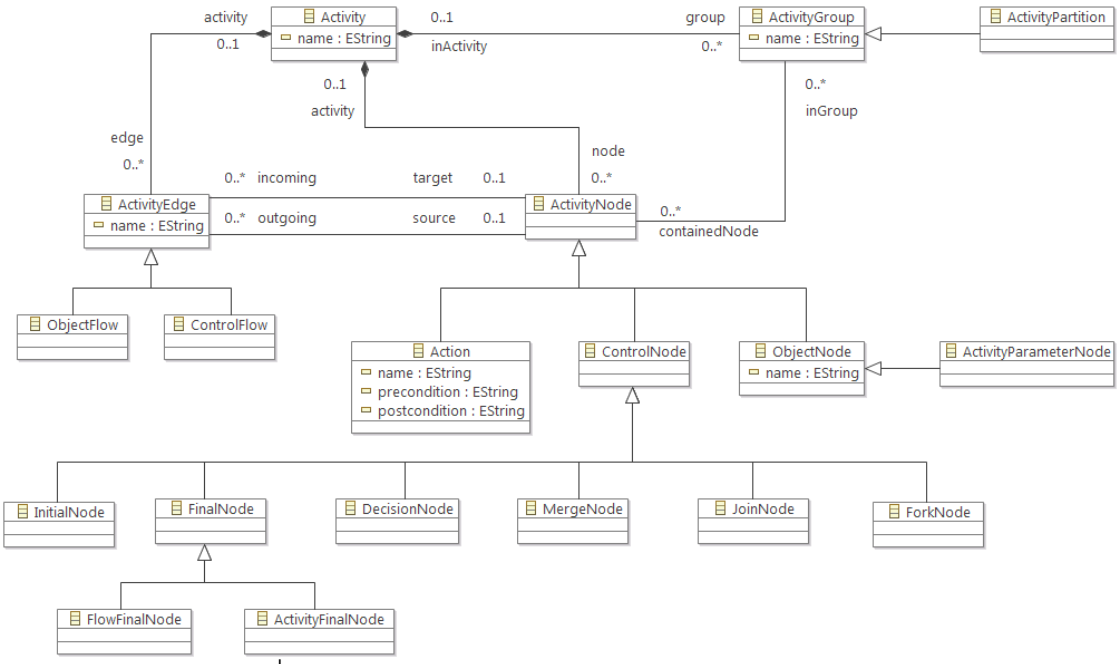
3.7 การสร้างแผนภาพกิจกรรมระดับปานกลาง

แบบจำลองความหมายแผนภาพกิจกรรมระดับปานกลางสามารถสร้างได้จากการแปลงแบบจากตัวสร้างแผนภาพ บนพื้นฐานของเมทาโมเดลภาพที่ 18 ซึ่งการแปลงแบบจะคล้ายกับการแปลงแบบของแผนภาพกิจกรรมระดับพื้นฐาน แต่จะแตกต่างกันตรงที่การสร้างบัพควบคุมต่าง ๆ จะต้องใช้วิธีสกัดข้อมูลจากเส้นเชื่อมเข้าและเส้นเชื่อมออกของแต่ละบัพในแบบจำลองความหมาย ดังนั้นการสร้างแบบจำลองความหมายจึงประกอบด้วยสองขั้นตอน คือ ขั้นตอนที่หนึ่งการแปลงแบบ และขั้นตอนที่สองการสกัดข้อมูลเพื่อสร้างบัพควบคุมต่าง ๆ

สำหรับขั้นตอนที่หนึ่งการแปลงแบบจะมีรายละเอียดการแปลงดังต่อไปนี้

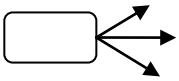
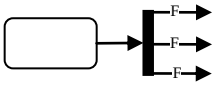
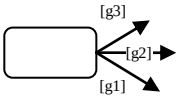
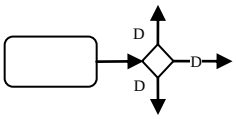
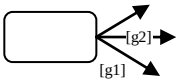
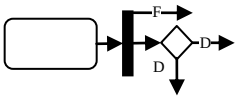
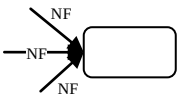
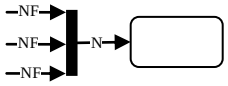
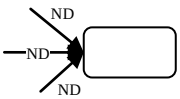
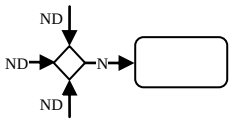
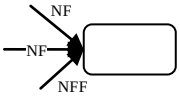
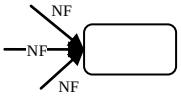
- Activity :: Activity แปลงจากกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นกิจกรรม
- ObjectNode :: ObjectNode แปลงจากวัตถุแบบชัดแจ้งและวัตถุที่เกิดจากการสร้างเงื่อนไขของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นวัตถุในกิจกรรม
- Action :: Action แปลงจากแอ็คชั่นของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นแอ็คชั่นในกิจกรรม
- InitialNode :: InitialNode แปลงจากจุดเริ่มต้นของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นจุดเริ่มต้นในกิจกรรม
- ActivityFinalNode :: ActivityFinalNode แปลงจากจุดสิ้นสุดกิจกรรมของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นจุดสิ้นสุดในกิจกรรม
- FlowFinalNode :: FlowFinalNode แปลงจากจุดสิ้นสายงานของกิจกรรมในตัวสร้างแผนภาพกิจกรรม เป็นจุดสิ้นสุดสายงานในกิจกรรม

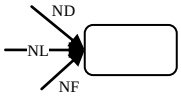
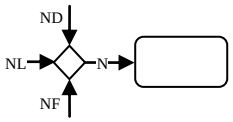
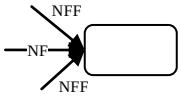
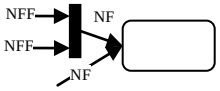
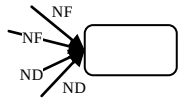
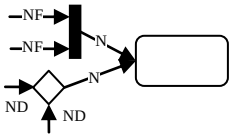
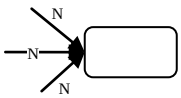
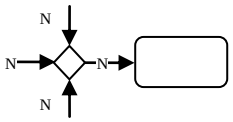
สำหรับขั้นตอนที่สองการสกัดข้อมูลเพื่อสร้างบัพควบคุมต่าง ๆ จะแบ่งได้ออกเป็นสองกรณี คือ การสร้างบัพแยกและบัพตัดสินใจ และการสร้างบัพรวมและบัพผสาน โดยการสร้างบัพแยกและบัพตัดสินใจจะพิจารณาจากเส้นเชื่อมออกของแต่ละบัพในแบบจำลองความหมายของแผนภาพกิจกรรมระดับปานกลาง ส่วนการสร้างบัพรวมและบัพผสานจะพิจารณาจากเส้นเชื่อมเข้าของแต่ละบัพ โดยการสร้างบัพรวมและบัพผสานนั้นจะต้องพิจารณาจากบัพควบคุมก่อนหน้าและระดับความลึกของบัพนั้น ๆ (พิจารณาจากจำนวนและประเภทของบัพควบคุมก่อนหน้า) ซึ่งการสร้างบัพทั้งสองกรณีสามารถสรุปได้ดังตารางที่ 2



ภาพที่ 18 เมทาโมเดลของแผนภาพกิจกรรมระดับปานกลาง

ตารางที่ 2 แพทเทิร์นสำหรับการสร้างบัพควบคุมต่าง ๆ

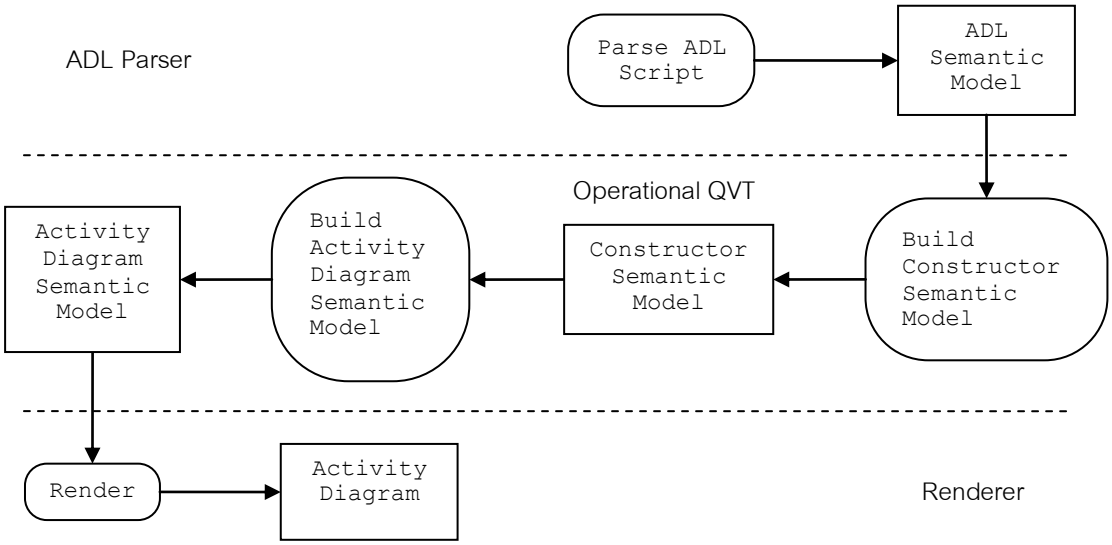
	แพทเทิร์น	ชื่อแพทเทิร์น	รายละเอียด	แพทเทิร์นผลลัพธ์
บัพแยกและบัพตัดสินใจ		Fork	เส้นเชื่อมออกทั้งหมดเป็นเส้นเชื่อมปกติ	
		Decision	เส้นเชื่อมออกทั้งหมดเป็นเส้นเชื่อมแบบมีเงื่อนไข	
		Fork-Decision	เส้นเชื่อมออกบางส่วนเป็นเส้นเชื่อมปกติและบางส่วนมีเงื่อนไข	
บัพผสานและบัพตัดสินใจ		Join	เส้นเชื่อมเข้าทั้งหมดเป็นประเภท F (บัพแยก)	
		Merge	เส้นเชื่อมเข้าทั้งหมดเป็นประเภท D (บัพตัดสินใจ)	
		Invalid	เส้นเชื่อมเข้าที่อยู่ในระดับลึกสุดไม่สามารถจับคู่ได้	

		Different	เส้นเชื่อมเข้าแต่ละเส้นเป็นคนละประเภทกัน	
		Nested Control	เส้นเชื่อมเข้ามีหลายระดับและสามารถจับคู่ในระดับลึกสุดได้	
		Set	เส้นเชื่อมเข้าในระดับเดียวกันจับคู่ได้มากกว่าหนึ่งกลุ่ม	
		Normal	เส้นเชื่อมเข้าทั้งหมดเป็นประเภท N (บัพเริ่มต้น)	

บทที่ 4
การออกแบบและพัฒนาระบบ

4.1 สถาปัตยกรรมระบบ

ระบบที่พัฒนาขึ้นแบ่งออกเป็นสามมอดูลหลัก คือ มอดูลการแจงส่วนเอดีแอล มอดูลการแปลงแบบจำลอง และมอดูลการแสดงผลแผนภาพกิจกรรม โดยทั้งสามมอดูลมีความสัมพันธ์กัน ดังภาพที่ 19 การแจงส่วนเอดีแอลจะทำหน้าที่ในการแปลงบทคำสั่งที่เขียนขึ้นในรูปของเอดีแอลให้อยู่ในรูปของแบบจำลองความหมายของเอดีแอล หลังจากนั้นมอดูลการแปลงแบบจำลองจะทำหน้าที่แปลงแบบจำลองความหมายของเอดีแอลให้อยู่ในรูปของแบบจำลองความหมายของแผนภาพกิจกรรมต่าง ๆ เมื่อได้แบบจำลองความหมายของแผนภาพกิจกรรมแล้วมอดูลการแสดงผลแผนภาพกิจกรรมจะหน้าที่ในการสร้างแผนภาพกิจกรรมในรูปแบบกราฟวิซเพื่อแปลงเป็นรูปภาพจากข้อมูลดังกล่าว



ภาพที่ 19 การทำงานของระบบ

4.2 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒาระบบประกอบด้วยรายการฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

4.2.1 สภาพแวดล้อม

1. หน่วยประมวลผลอินเทล คอร์ 2 ดูโอ 2.20 กิกะเฮิร์ต (CPU Intel Core 2 Duo 2.20GHz)
2. หน่วยความจำ 4 กิกะไบต์ (4 GB RAM)

3. ฮาร์ดดิสก์ความจุ 500 กิกะไบต์ (500 GB HDD)
4. ระบบปฏิบัติการไมโครซอฟต์วินโดว์ 7 (Microsoft Windows 7) แบบ 64 บิท

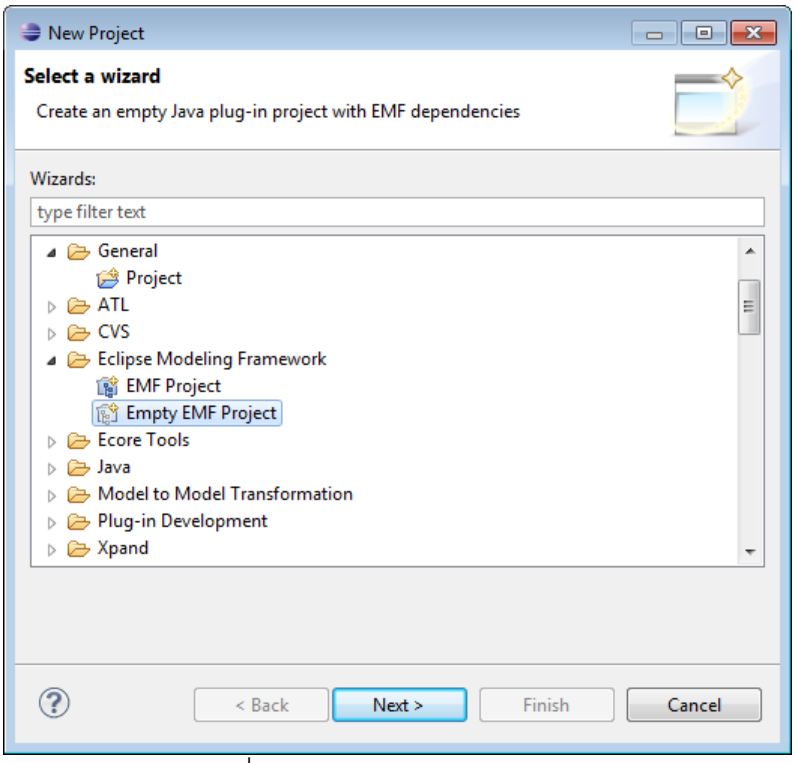
4.2.2 เครื่องมือที่ใช้ในการพัฒนา

1. อีคลิปส์ 3.7.2 (Eclipse 3.7.2)
2. ชุดเครื่องมือพัฒนาจาวา 7 (Java Development Kit 7)
3. อีคลิปส์โมเดลลิ่งทูล 1.4.2 (Eclipse Modeling Tools 1.4.2)
4. ชุดเครื่องมือพัฒนาคิววีทีเชิงปฏิบัติการ 3.1.0 (Operational QVT SDK 3.1.0)
5. เครื่องมือพัฒนาเอ็กซ์เทค 2.0.1 (Xtext SDK 2.0.1)
6. เครื่องมือพัฒนาเอ็กซ์แพน 1.1.1 (Xpand SDK 1.1.1)
7. กราฟวิซ 2.28.0 (Graphviz 2.28.0)

4.3 การพัฒนาระบบ

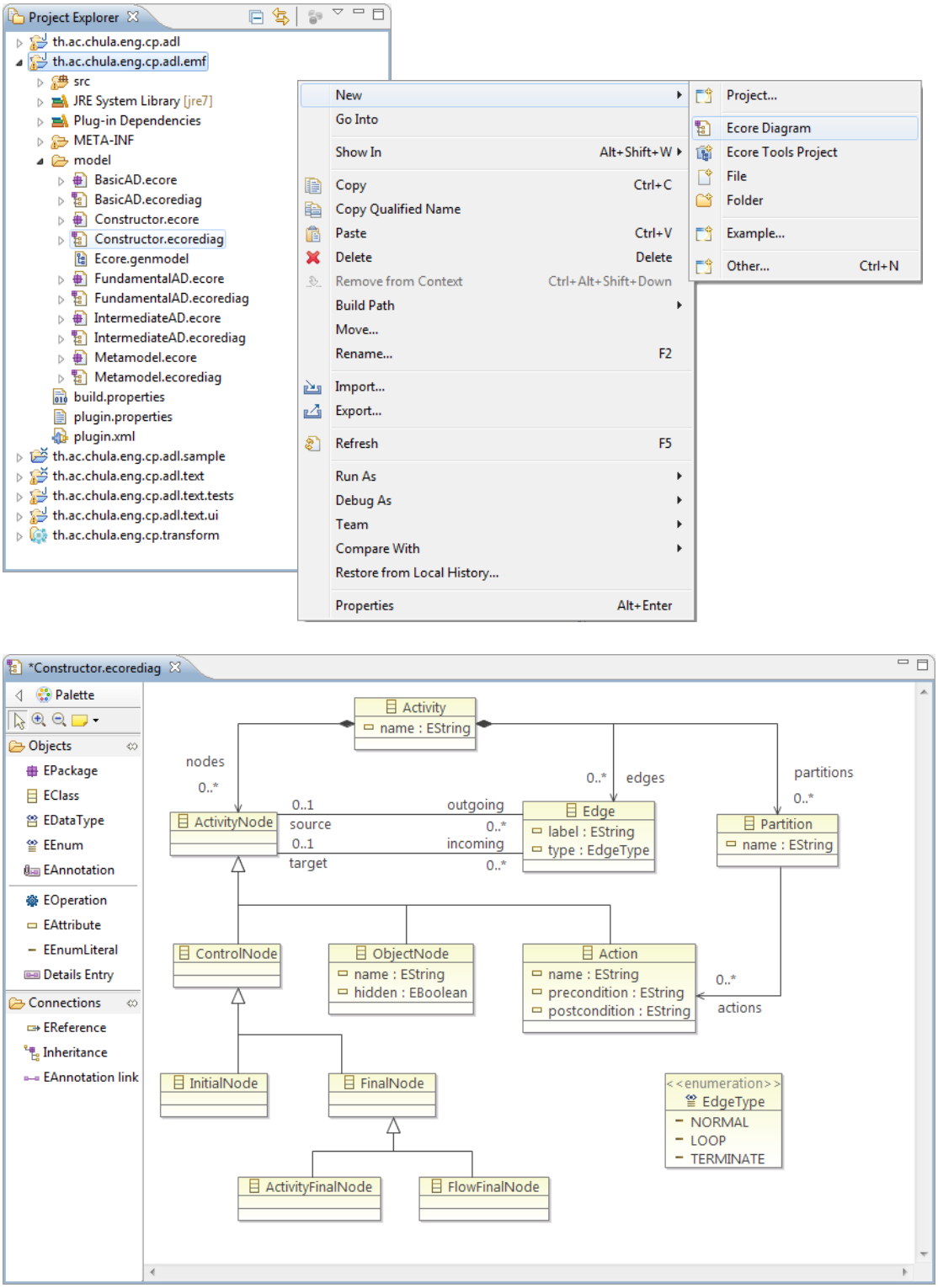
4.3.1 การพัฒนาเมทาโมเดล

การพัฒนาเมทาโมเดลจะใช้อีคลิปส์โมเดลลิ่งทูลในการพัฒนา โดยเราจะทำการเลือกสร้างโพรเจกต์ใหม่ แล้วเลือกอีเอ็มเอฟโพรเจกต์ (EMF Project) จากหัวข้ออีคลิปส์โมเดลลิ่งเฟรมเวิร์ค (Eclipse Modeling Framework) ดังภาพที่ 20



ภาพที่ 20 การสร้างอีเอ็มเอฟโพรเจกต์

หลังจากนั้นให้ทำการเลือกสร้างแผนภาพอีคอร์ (Ecore Diagram) ภายใต้หัวข้อโมเดล เพื่อทำการสร้างเมทาโมเดลให้กับระบบ (เมทาโมเดลที่ใช้อ้างอิงจากบทที่ 3) ดังตัวอย่างภาพที่ 21



ภาพที่ 21 การสร้างแผนภาพอีคอร์

หลังจากที่ทำการสร้างแผนภาพสำหรับเมทาโมเดลของเอดีแอล เมทาโมเดลของตัวสร้างแผนภาพกิจกรรม เมทาโมเดลของแผนภาพกิจกรรมระดับรากฐาน เมทาโมเดลของแผนภาพกิจกรรมระดับพื้นฐาน และเมทาโมเดลของแผนภาพกิจกรรมระดับปานกลางเสร็จแล้ว ในขั้นตอนต่อไปคือการสร้างไฟล์คลาสต่าง ๆ ที่เกี่ยวข้องด้วยเครื่องมืออีเอ็มเอฟเจเนอเรเตอร์โมเดล (EMF Generator Model) จากหัวข้ออีคลิปส์โมเดลลิ่งเฟรมเวิร์ค หลังจากนั้นจึงทำการสร้างส่วนเสริม (Plug-In) เพื่อนำไปลงทะเบียนให้อีคลิปส์สามารถเข้าถึงได้ในการพัฒนาลำดับต่อ ๆ ไป

4.3.2 **การพัฒนาตัวแจงส่วน** (Parser)

การพัฒนาตัวแจงส่วนจะใช้เครื่องมือพัฒนาเอ็กซ์เทคและเครื่องมือพัฒนาเอ็กซ์แพน เพื่อสร้างไวยากรณ์ของระบบและตัวแจงส่วนตามลำดับ การพัฒนาวากยสัมพันธ์ให้กับเครื่องมือพัฒนาเอ็กซ์เทคนั้นจะใช้หลักการเช่นเดียวกันกับการพัฒนาวากยสัมพันธ์บนเครื่องมือพัฒนาภาษาแอนท์เลอร์ [10] เนื่องจากเอ็กซ์เทคนั้นถูกสร้างขึ้นเพื่อห่อหุ้มเครื่องมือพัฒนาภาษาแอนท์เลอร์อีกทีหนึ่ง ดังนั้นรูปแบบวากยสัมพันธ์สำหรับไวยากรณ์ที่ออกแบบไว้ในบทที่ 3 จึงสามารถกำหนดได้ดังภาพที่ 22 โดยมีรายละเอียดดังตารางที่ 3

```
grammar th.ac.chula.eng.cp.adl.text.ADLDSL with
      org.eclipse.xtext.common.Terminals

generate adldsl "http://www.chula.ac.th/eng/cp/adl/text/ADLDSL"

import "http://www.eclipse.org/emf/2002/Ecore" as ecore

Diagram:
    'diagram' name=STRING elements+=Element* 'end';

Element:
    Action | Decision | Partition | Sequence;

Action:
    'action' id=ACTIONID options+=Action_Opts* 'end';

Action_Opts:
    'name' name=STRING |
    ('<-' | 'inputs') inputObjects=MultiOID |
    ('->' | 'outputs') outputObjects=MultiOID |
    'precondition' precondition=STRING |
    'postcondition' postcondition=STRING;

Decision:
    'decision' 'from' id=ACTIONID guards+=Guard* 'end';

Guard:
    'if' condition=STRING 'then' (then+=Expression)+
    ('else' (els+=Expression)+)*
    'endif';
```

ภาพที่ 22 ชุดข้อมูลวากยสัมพันธ์สำหรับสร้างไวยากรณ์ของเอดีแอล

```
Expression:
    MultiAID | Guard;

Partition:
    'partition' name=STRING actions+=ACTIONID+ 'end'
;

Sequence:
    id=ACTIONID (('->' | 'then') next+=SequenceRight)+
;

SequenceRight:
    id=(MultiAID) ('[' expression=STRING ']')?
;

MultiAID: id+=ACTIONID ((',' | 'and') id+=ACTIONID)*;
MultiOID: id+=OBJECTID ((',' | 'and') id+=OBJECTID)*;

terminal BOOLEAN returns ecore::EBoolean:('true'|'false');
terminal ACTIONID: '^'?('a'..'z') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;
terminal OBJECTID: '^'?('A'..'Z') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*;
```

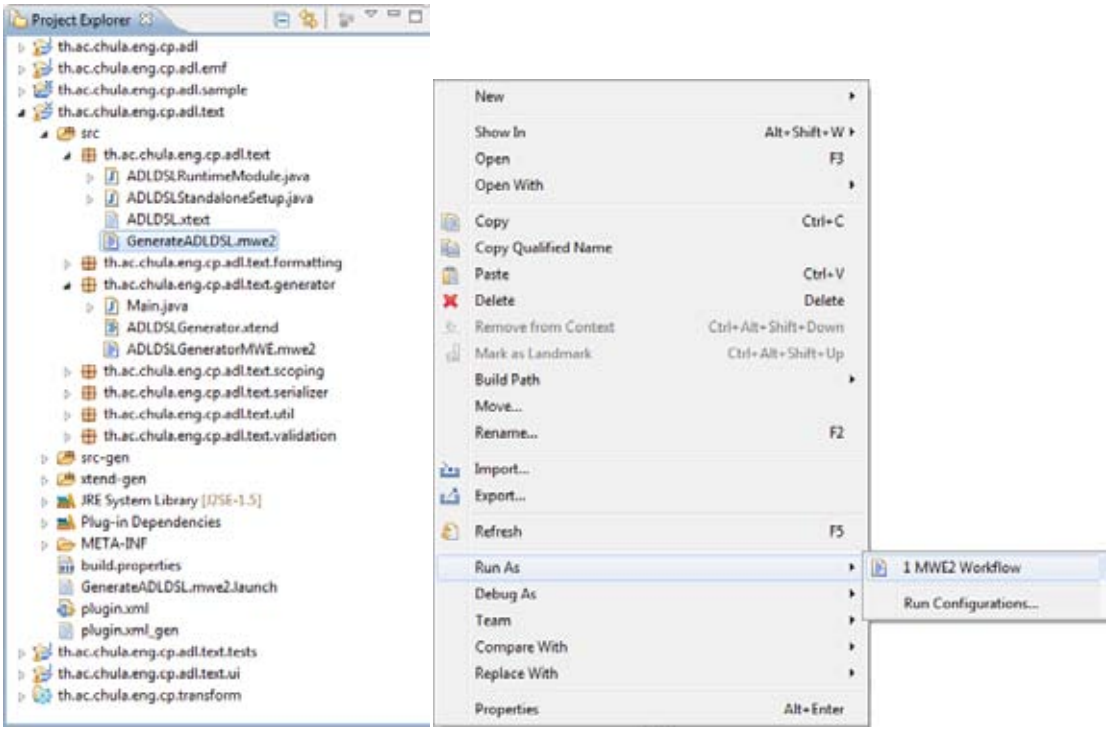
ภาพที่ 22 ชุดข้อมูลวากยสัมพันธ์สำหรับสร้างไวยากรณ์ของเอดีแอล (ต่อ)

ตารางที่ 3 คำสำคัญของเอดีแอลและรายละเอียดการใช้งาน

คำสำคัญ	**รายละเอียด**	**ตัวอย่างการใช้งาน**	**ความหมาย**
diagram	สร้างแผนภาพกิจกรรมตามชื่อที่กำหนด	diagram ‘name’ ... end	สร้างแผนภาพกิจกรรมที่มีชื่อว่า “name”
action	สร้างแอ็คชั่นตามชื่อที่กำหนด	action ‘id’ name ‘action name’ inputs ObjectIn outputs ObjectOut end	สร้างแอ็คชั่นที่มีตัวชี้ คือ “id”และกำหนดให้แอ็คชั่นดังกล่าวมีชื่อเป็น “action name” โดยมี “ObjectIn” และ “ObjectOut” เป็นอินพุตและเอาต์พุตของแอ็คชั่นตามลำดับ
decision	สร้างการตัดสินใจที่ซับซ้อนในกิจกรรม	decision ‘action1’ if ‘condition’ then ‘action2’ endif end	สร้างการตัดสินใจหลัง “action1” โดยถ้ามีเงื่อนไขเป็น “condition” ให้ทำงาน “action2”
then	สร้างลำดับการทำงานในกิจกรรม	action1 then action2	หลังจากที่ทำงาน “action1” เสร็จแล้วให้ทำงาน “action2”
initiate	กำหนดจุดเริ่มต้นของกิจกรรม	initiate then action1	กำหนดให้ “action1” เป็นจุดเริ่มต้นกิจกรรม

break	กำหนดจุดสิ้นสุดของสายงาน	action1 then break	หยุดการทำงานของสายงานหลังจากที่ทำงาน “action1” เสร็จแล้ว
terminate	กำหนดจุดสิ้นสุดกิจกรรม	action1 then terminate	หยุดการทำงานของกิจกรรมหลังจากที่ทำงาน “action1” เสร็จแล้ว

สำหรับการพัฒนาตัวแจงส่วนนั้น เราสามารถสร้างโครงสร้างพื้นฐานสำหรับตัวแจงส่วนได้โดยดำเนินงานคำสั่งจากไฟล์เอ็มดับบลิวอีทู (Modeling Workflow Engine 2 - MWE2) ซึ่งเป็นไฟล์ที่สร้างขึ้นมาพร้อมกับไฟล์ไวยากรณ์ของเอดีแอล โดยการดำเนินงานคำสั่งสามารถดำเนินงานได้ดังภาพที่ 23



ภาพที่ 23 การดำเนินงานคำสั่งจากไฟล์เอ็มดับบลิวอีทู

หลังจากนั้นจึงพัฒนาชุดคำสั่งสำหรับการแจงส่วนเพื่อแปลงบทคำสั่งเอดีแอลเป็นแบบจำลองความหมายโดยมีรายละเอียดพอสังเขปดังภาพที่ 24 หลังจากนั้นเราสามารถบันทึกแบบจำลองความหมายให้อยู่ในรูปแบบของไฟล์เอ็กซ์เอ็มไอได้โดยใช้ชุดคำสั่งดังภาพที่ 25

```
package th.ac.chula.eng.cp.adl.text.generator

import org.eclipse.emf.ecore.resource.Resource

import th.ac.chula.eng.cp.adl.emf.metamodel.*
import th.ac.chula.eng.cp.adl.emf.metamodel.impl.*

class ADLDSLGenerator implements IGenerator {
    MetamodelFactory metamodelFactory
    Diagram rootContainer

    override void doGenerate(Resource resource, IFileSystemAccess
            fsa) {
        for (e: resource.contents.filter(typeof(Diagram))) {
            this.metamodelFactory = MetamodelFactoryImpl::init()
            e.compile
    }

    def compile(Diagram diagram) {
        val d = metamodelFactory.createDiagram()
        d.name = diagram.name

        this.rootContainer = d

        for (e: diagram.elements) {
            if (e instanceof Action) compile(e as Action)
            else if (e instanceof Sequence) compile(e as Sequence)
            else if (e instanceof Decision) compile(e as Decision)
            else if (e instanceof Partition) compile(e as Partition)
        }
    }

    def compile(Action action)
    def compile(Partition p)
    def compile(Sequence s)
    def compile(Decision d)
    def compile(Guard guard, Collection<ObjectE> o)
}
```

ภาพที่ 24 ชุดคำสั่งการแปลงบทคำสั่งเอดีแอลเป็นแบบจำลองความหมาย

```
package th.ac.chula.eng.cp.adl.text.util;

import java.io.File;
import java.io.IOException;

import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.resource.Resource;
import org.eclipse.emf.ecore.resource.ResourceSet;
import org.eclipse.emf.ecore.resource.impl.ResourceSetImpl;
import org.eclipse.emf.ecore.xmi.impl.XMIResourceFactoryImpl;
import th.ac.chula.eng.cp.adl.emf.metamodel.Diagram;

public class TextUtil {

    private static final String OUTPUT =
            "../th.ac.chula.eng.cp.adl.sample/output/";

    public static void saveXMI(Diagram d) {
        ResourceSet resourceSet = new ResourceSetImpl();
        resourceSet
            .getResourceFactoryRegistry()
            .getExtensionToFactoryMap()
            .put("xmi", new XMIResourceFactoryImpl());
        String filename = OUTPUT + d.getName();

        Resource res = resourceSet
            .createResource(URI.createFileURI(filename
            + ".metamodel.xmi"));
        res.getContents().add(d);

        try {
            res.save(null);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

ภาพที่ 25 ชุดคำสั่งการบันทึกแบบจำลองความหมายให้อยู่ในรูปแบบเอ็กซ์เอ็มไอ

4.3.3 การพัฒนาตัวแปลงแบบจำลอง

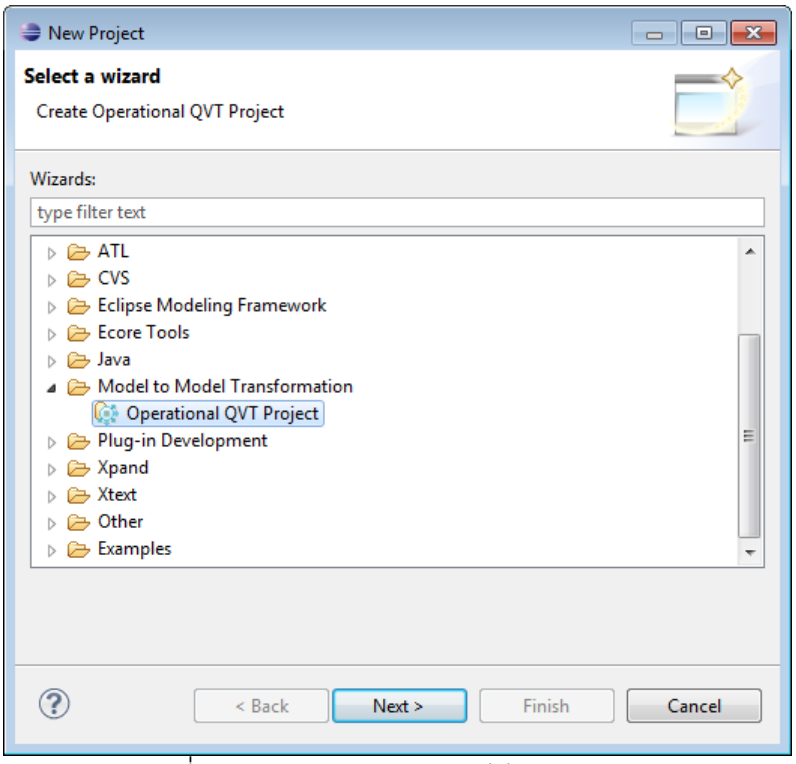
การพัฒนาตัวแปลงแบบจำลองจะใช้เครื่องมือพัฒนาคิววีทีเชิงปฏิบัติการ ในการแปลงแบบจำลองต่าง ๆ สาเหตุที่ใช้คิววีทีเชิงปฏิบัติการเนื่องจากการแปลงแบบจำลองมีลักษณะการแปลงในทิศทางเดียว (ไม่สามารถแปลงกลับได้) ซึ่งมีข้อดีคือทำให้สามารถกำหนดการแปลงในลักษณะพิเศษได้ง่ายและสะดวก อาทิเช่น การสร้างบัพควบคุมต่าง ๆ เป็นต้น โดยการพัฒนาจะเริ่มจากการสร้างโพรเจ็กต์จากโอเปอร์เรชั่นนอลคิววีทีโพรเจ็กต์ (Operational QVT Project) ดังภาพที่ 27 หลังจากนั้นจึงทำการสร้างตัวแปลงแบบดังภาพที่ 28 ซึ่งชุดคำสั่งที่ใช้ในการแปลงจะใช้ชุดคำสั่งแบบเดียวกับมาตราฐานที่โอเอ็มจี (Object Management Group - OMG) เป็นคนกำหนดขึ้น โดยสามารถดูตัวอย่างชุดคำสั่งการแปลงแบบจากแบบจำลองความหมายของตัวสร้าง

แผนภาพกิจกรรมไปยังแผนภาพกิจกรรมระดับรากฐานได้ดังภาพที่ 26

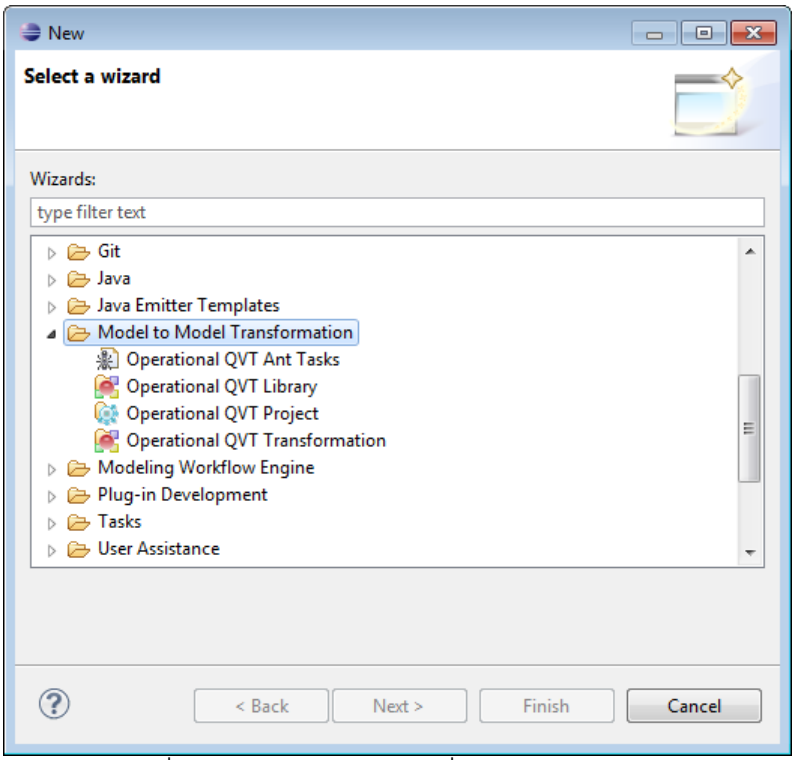
```
modeltype Constructor "strict" uses
'http://th.ac.chula.eng.cp.adl.emf/constructor/1.0';
modeltype FundamentalAD "strict" uses
'http://th.ac.chula.eng.cp.adl.emf/fundamentalad/1.0';
transformation ConstructorToFundamentalAD(in source:Constructor, out
target:FundamentalAD);
main() {
      source.rootObjects()[Activity].map activity2activity();
}
mapping Activity::activity2activity() : FundamentalAD::Activity {
      name := self.name;
      node += self.nodes->select(e | e.oclIsKindOf(Action))[Action]
  -    >map action2action();
      group += self.partitions->map partition2group();
}
mapping Action::action2action() : FundamentalAD::Action {
      name := self.name;
}
mapping Partition::partition2group() : FundamentalAD::ActivityGroup {
      name := self.name;
      self.actions->forEach(a) {
            target.rootObjects()[FundamentalAD::Activity].node
   -         >forEach(n | n.oclIsKindOf(FundamentalAD::Action)
                  and n.oclAsType(FundamentalAD::Action).name =
                  a.name) {
                  containedNode += n;
            };
      };
}
```

ภาพที่ 26 ตัวอย่างการสร้างกิจกรรมระดับรากฐานจากตัวสร้างแผนภาพกิจกรรม

หลังจากนั้นเราสามารถเริ่มการแปลงแบบได้โดยการเพิ่มรายละเอียดการดำเนินงานในรันคอนฟิกกูเรชั่น (Run Configuration) ของอีคลิปส์ดังภาพที่ 30 เมื่อทำการดำเนินงานเราจะได้ไฟล์เอ็กซ์เอ็มไอของแบบจำลองความหมายของเมทาโมเดลที่ได้เลือกไว้ เพื่อนำไปใช้ในการสร้างแผนภาพกิจกรรมในลำดับต่อไป



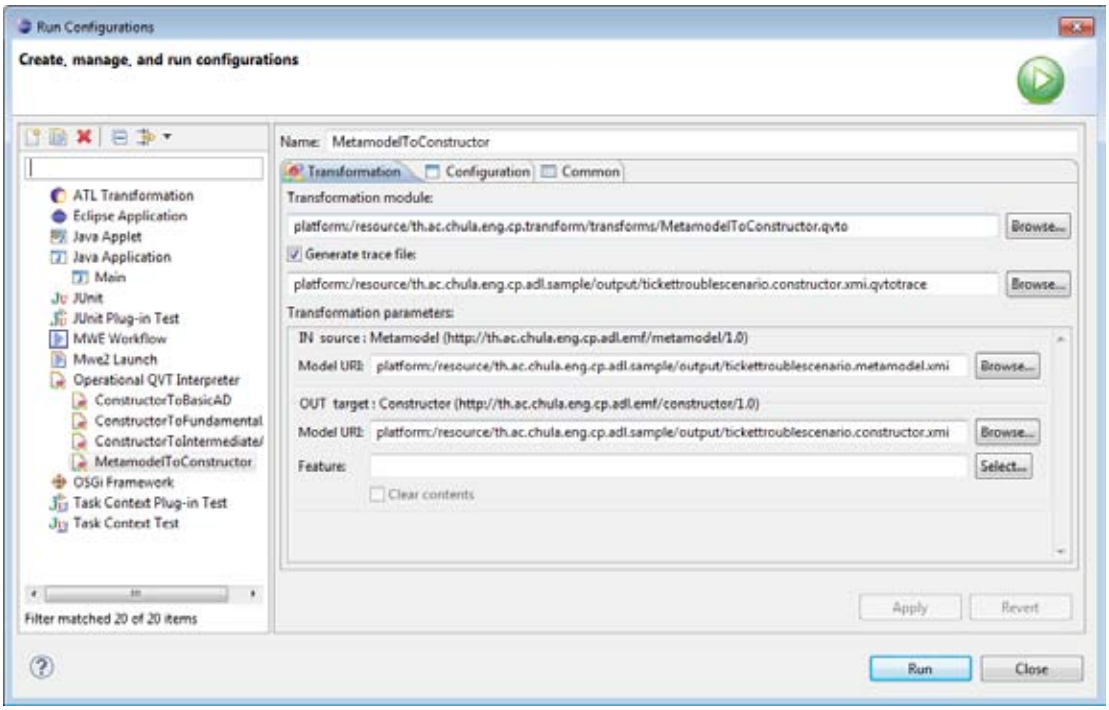
ภาพที่ 27 การสร้างโพรเจ็กต์คิววีทีเชิงปฏิบัติการ



ภาพที่ 28 การสร้างไฟล์ชุดคำสั่งคิววีทีเชิงปฏิบัติการ

```
modeltype Constructor "strict" uses
      'http://th.ac.chula.eng.cp.adl.emf/constructor/1.0';
modeltype FundamentalAD "strict" uses
      'http://th.ac.chula.eng.cp.adl.emf/fundamentalad/1.0';
transformation ConstructorToFundamentalAD(in source:Constructor, out
      target:FundamentalAD);
main() {
      source.rootObjects()[Activity].map activity2activity();
}
mapping Activity::activity2activity() : FundamentalAD::Activity {
      name := self.name;
      node += self.nodes->select(e | e.oclIsKindOf(Action))[Action]
            ->map action2action();
      group += self.partitions->map partition2group();
}
mapping Action::action2action() : FundamentalAD::Action {
      name := self.name;
}
mapping Partition::partition2group() : FundamentalAD::ActivityGroup {
      name := self.name;
      self.actions->forEach(a) {
            target.rootObjects()[FundamentalAD::Activity].node
                  ->forEach(n | n.oclIsKindOf(FundamentalAD::Action)
                        and n.oclAsType(FundamentalAD::Action).name =
                        a.name) {
                  containedNode += n;
            };
      };
}
```

ภาพที่ 29 ตัวอย่างชุดคำสั่งคิวีทีเชิงปฏิบัติการ



ภาพที่ 30 การดำเนินการชุดคำสั่งคิวีทีเชิงปฏิบัติการ

4.3.4 การพัฒนาตัวแสดงผล

การพัฒนาตัวแสดงผลแผนภาพกิจกรรมในงานวิจัยนี้ได้เลือกใช้เครื่องมือกราฟวิซเข้ามาช่วยในการวาดกราฟ โดยการสร้างไฟล์กราฟวิซขึ้นมาจากแบบจำลองความหมายของแผนภาพกิจกรรมที่ได้จากจากนำเข้าข้อมูลไฟล์เอ็กซ์เอ็มไอ หลังจากนั้นจึงใช้โปรแกรมกราฟวิซวาดกราฟจากไฟล์ที่สร้างขึ้นดังกล่าว

หลักการที่ใช้ในการสร้างไฟล์กราฟวิซนั้นจะใช้วิธีการสร้างบัพในกราฟวิซจากบัพต่าง ๆ ที่เกิดขึ้นในแบบจำลองความหมายของแผนภาพกิจกรรม หลังจากนั้นจึงทำการสร้างเส้นเชื่อมระหว่างบัพในกราฟวิซจากเส้นเชื่อมกิจกรรมที่ระบุไว้ในแบบจำลองความหมายของแผนภาพกิจกรรม โดยลักษณะการสร้างบัพและเส้นเชื่อมในกราฟวิซจะใช้ลักษณะการแปลงแบบหนึ่งต่อหนึ่ง ทั้งนี้เพื่อเป็นการพิสูจน์ว่าแบบจำลองความหมายของแผนภาพกิจกรรมที่สร้างขึ้นนั้นมีความหมายตรงกับที่ต้องการจริง ๆ ไม่ได้ถูกบิดเบือนจากการสร้างตัวแสดงผลแต่อย่างใด ซึ่งตัวอย่างผลลัพธ์ที่ได้จากการแปลงแบบจำลองความหมายของแผนภาพกิจกรรมเป็นไฟล์กราฟวิซแสดงดังภาพที่ 31 และมีผลลัพธ์แผนภาพกิจกรรมดังภาพที่ 12

```
digraph "diagram name" {
subgraph cluster_0 {
label="diagram name";
color=black;
NODE_244071D [shape=box, style=rounded, label="Action Name 1"];
NODE_402F0214 [shape=box, style=rounded, label="Action Name 2"];
NODE_4E318FF7 [shape=box, style=rounded, label="Action Name 3"];
NODE_17E2C93C [shape=box, style=rounded, label="Action Name 4"];
NODE_2A098E64 [shape=box, style=rounded, label="Action Name 5"];
NODE_59635CCB [shape=box, style=rounded, label="Action Name 6"];
NODE_A614146 [shape=circle, fillcolor=black, style=filled,
fixedsize=true, width=.5, label=""];
NODE_77045FFC [shape=doublecircle, fillcolor=black, style=filled,
fixedsize=true, width=.5, label=""];
NODE_4039D66F [shape=diamond, fixedsize=true, width=.5, height=.5,
label=""];
NODE_152D325A [shape=box, fillcolor=black, style=filled,
fixedsize=true, width=1, height=.1, label=""];
NODE_A614146->NODE_244071D;
NODE_244071D->NODE_402F0214 [label="[condition1]"];
NODE_4039D66F->NODE_4E318FF7 [label="[condition2]"];
NODE_4E318FF7->NODE_77045FFC;
NODE_4039D66F->NODE_17E2C93C [label="[condition3]"];
NODE_152D325A->NODE_2A098E64;
NODE_2A098E64->NODE_77045FFC;
NODE_152D325A->NODE_59635CCB;
NODE_59635CCB->NODE_77045FFC;
NODE_402F0214->NODE_4039D66F;
NODE_17E2C93C->NODE_152D325A;
}}
```

ภาพที่ 31 ตัวอย่างไฟล์กราฟวิซ

บทที่ 5
การประเมินและการวัดผล

5.1 แนวทางการประเมินและการวัดผล

แนวทางการประเมินจะใช้วิธีการแปรียบเทียบระหว่างประชากรตัวอย่างกับผลลัพธ์ที่ได้จากซอฟต์แวร์ของระบบ โดยข้อมูลตัวอย่างจะใช้ข้อมูลแผนภาพกิจกรรมจากเอกสารยูเอ็มแอลของโอเอ็มจี [1] ทั้งนี้เพื่อให้มั่นใจได้ว่าข้อมูลตัวอย่างมีความถูกต้องตามมาตรฐานที่กำหนด สำหรับการเลือกข้อมูลตัวอย่างจะใช้เกณฑ์การเลือกโดยประชากรที่เลือกมาจะสอดคล้องและครอบคลุมกับแนวคิดในบทที่ 3 โดยแนวทางการประเมินประกอบด้วยรายละเอียดต่าง ๆ ดังต่อไปนี้

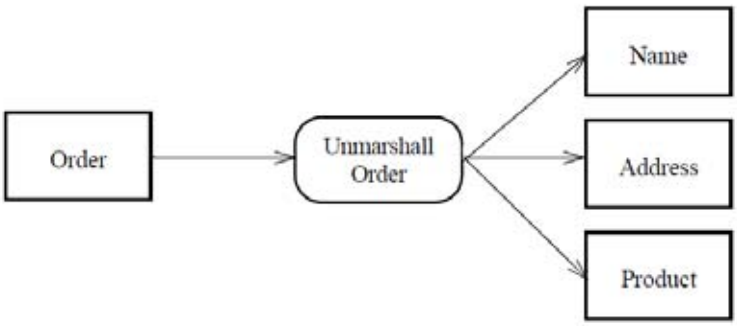
1. คำอธิบายแผนภาพกิจกรรม
2. ข้อมูลอธิบายแผนภาพกิจกรรมด้วยเอดีแอล
3. ผลลัพธ์จากซอฟต์แวร์ของระบบ
4. ข้อสังเกต

สำหรับการวัดผลนั้นจะใช้การเปรียบเทียบความแตกต่างของแผนภาพกิจกรรม และความแตกต่างของความหมายของแผนภาพกิจกรรมระหว่างประชากรตัวอย่างกับผลลัพธ์ที่ได้จากซอฟต์แวร์ของระบบ โดยการอธิบายความหมายจะเป็นการอธิบายลำดับขั้นตอนการดำเนินงานของแผนภาพกิจกรรมนั้น ๆ

5.2 ผลการเปรียบเทียบตัวอย่างที่ 1

5.2.1 คำอธิบาย

ตัวอย่างที่ 1 แสดงให้เห็นถึงความสามารถของแผนภาพกิจกรรมที่สามารถแสดงให้เห็นถึงการแจงรายละเอียดวัตถุได้ โดยการแสดงให้เห็นถึงอินพุตและเอาต์พุตของแอ็คชั่นดังภาพที่ 32



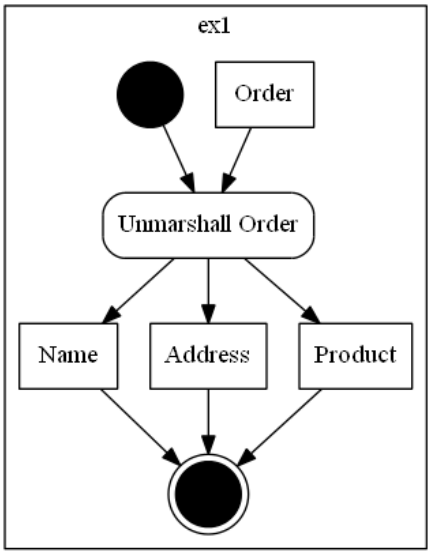
ภาพที่ 32 แผนภาพกิจกรรมตัวอย่างที่ 1

5.2.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'ex1'
      action unmarshallOrder
            <- Order
  -          > Name, Address, Product
      end
end
```

5.2.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 33 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 1

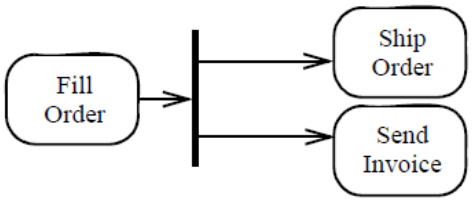
5.2.4 ข้อสังเกต

แผนภาพกิจกรรมที่ได้จากผลลัพธ์ของซอฟต์แวร์มีการแสดงให้เห็นถึงจุดเริ่มต้นและจุดสิ้นสุดของกิจกรรม ซึ่งจะแตกต่างจากตัวอย่างข้อมูล เนื่องจากเราสามารถประมาณได้ว่าจุดเริ่มต้นของแผนภาพกิจกรรม คือ “Unmarshall Order” จากวิธีการหาจุดเริ่มต้นในบทที่ 3 อย่างไรก็ตามผลลัพธ์ยังคงให้ความหมายเช่นเดียวกันกับแผนภาพตัวอย่าง

5.3 ผลการเปรียบเทียบตัวอย่างที่ 2

5.3.1 คำอธิบาย

ตัวอย่างที่ 2 แสดงให้เห็นถึงความสามารถในการสร้างการดำเนินงานแบบคู่ขนานดังตัวอย่างภาพที่ 34 แสดงให้เห็นว่าจะมีการดำเนินงาน “Ship Order” และ “Send Invoice” ในการดำเนินงานลำดับถัดไปหลังจากแอ็คชั่น “Fill Order” ทำงานเสร็จสิ้นแล้ว



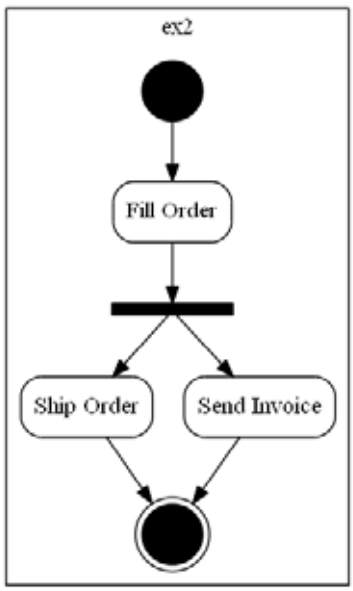
ภาพที่ 34 แผนภาพกิจกรรมตัวอย่างที่ 2

5.3.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'ex2'
      fillOrder->shipOrder and sendInvoice
end
```

5.3.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 35 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 2

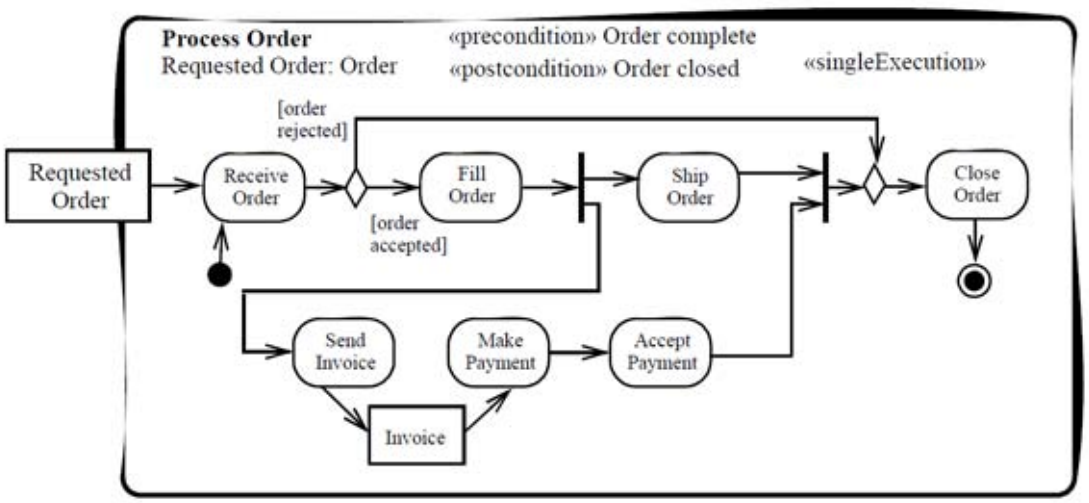
5.3.4 ข้อสังเกต

จะเห็นว่าจากแผนภาพกิจกรรมตัวอย่างไม่มีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดกิจกรรม ซึ่งทำให้เราสามารถอนุมานได้ว่ากิจกรรมเริ่มทำงานจาก “Fill Order” และกิจกรรมจะสิ้นสุดหลังจากดำเนินงาน “Ship Order” และ “Send Invoice” ซึ่งสอดคล้องกันกับแผนภาพกิจกรรมที่ได้จากผลลัพธ์ของซอฟต์แวร์

5.4 ผลการเปรียบเทียบตัวอย่างที่ 3

5.4.1 คำอธิบาย

ตัวอย่างที่ 3 แสดงให้เห็นถึงการสร้างแผนภาพกิจกรรมทั่ว ๆ ไปซึ่งส่วนมากจะประกอบด้วยการสร้างลำดับการทำงาน การส่งผ่านวัตถุ การตัดสินใจ การทำงานแบบคู่ขนาน และการรวมสายงานมากกว่าหนึ่งสายงานเข้าด้วยกัน



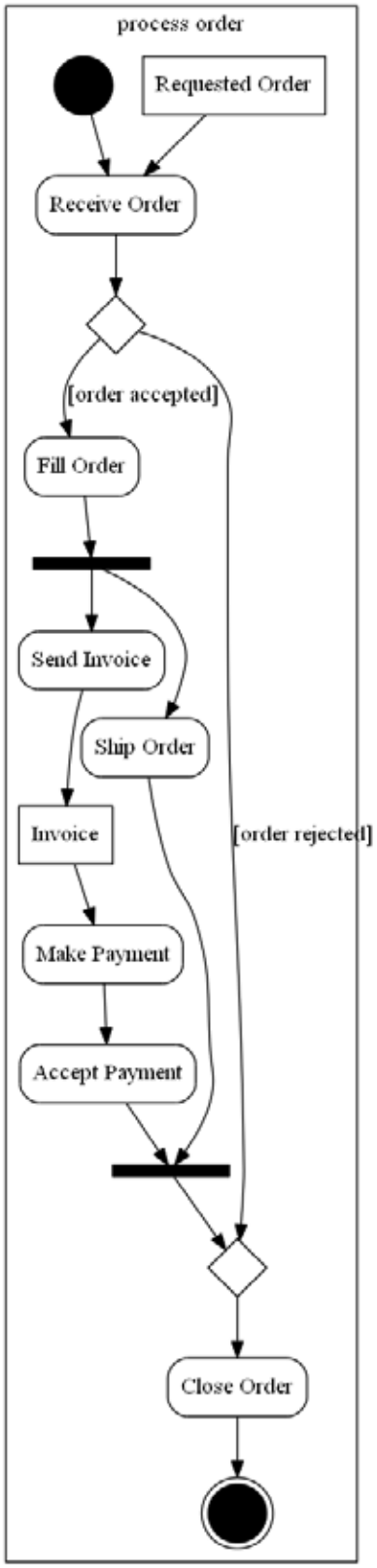
ภาพที่ 36 แผนภาพกิจกรรมตัวอย่างที่ 3

5.4.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'process order'
       action receiveOrder
            <- RequestedOrder
       end
       action sendInvoice
            -> Invoice
       end

       decision from receiveOrder
            if 'order accepted' then fillOrder
            else
                  if 'order rejected' then closeOrder
                  endif
            endif
       end
       fillOrder->shipOrder and sendInvoice
       sendInvoice->makePayment->acceptPayment->closeOrder
       shipOrder->closeOrder
end
```

5.4.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 37 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 3

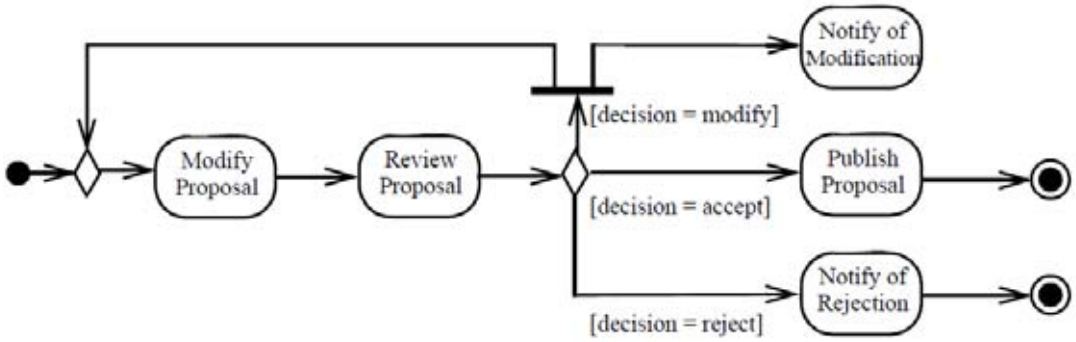
5.4.4 ข้อสังเกต

แผนภาพกิจกรรมตัวอย่างและแผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์นั้นมีองค์ประกอบของแผนภาพกิจกรรมเหมือนกันโดยเฉพาะการจัดวางตำแหน่งบัพควบคุม ซึ่งเป็นตัวบ่งชี้ให้เห็นว่าหลักการที่ใช้ในการสร้างบัพควบคุมต่าง ๆ ที่กล่าวถึงในบทที่ 3 สามารถนำไปใช้ได้จริงในเชิงปฏิบัติ

5.5 ผลการเปรียบเทียบตัวอย่างที่ 4

5.5.1 คำอธิบาย

ตัวอย่างที่ 4 แสดงให้เห็นถึงการหยุดสายงานกิจกรรมและการวนซ้ำภายในกิจกรรม



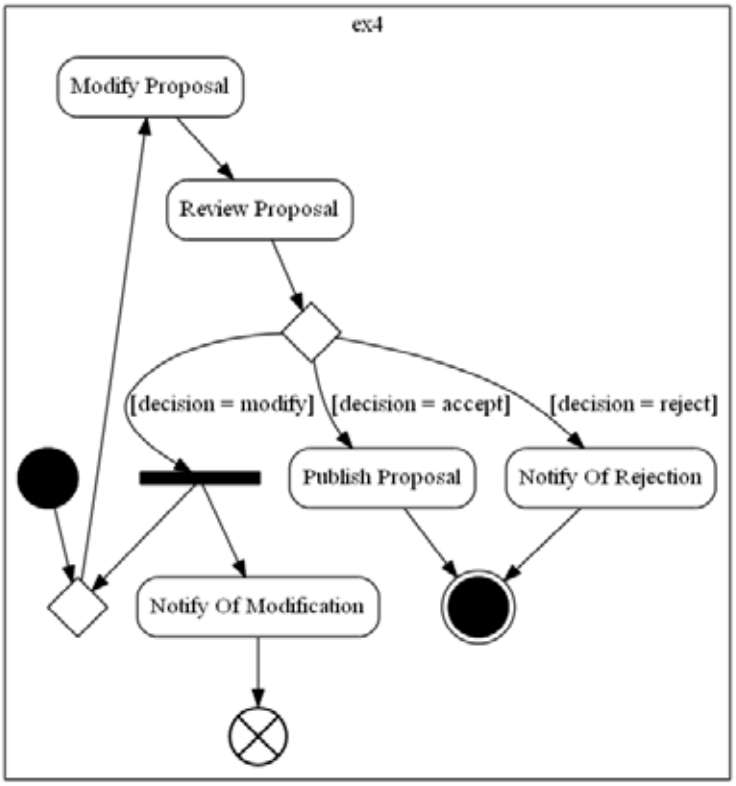
ภาพที่ 38 แผนภาพกิจกรรมตัวอย่างที่ 4

5.5.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'ex4'
       modifyProposal->reviewProposal
       decision from reviewProposal
            if 'decision = modify'
            then notifyOfModification and modifyProposal
            else
                  if 'decision = accept' then publishProposal
                  else
                        if 'decision = reject'
                        then notifyOfRejection
                        endif
                  endif
            endif
       end
       notifyOfModification->break
end
```

5.5.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 39 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 4

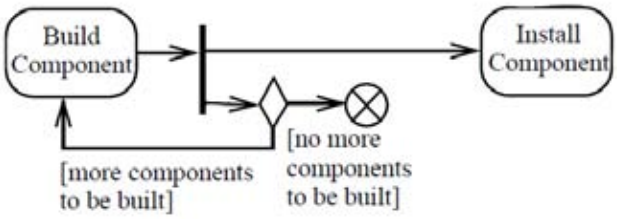
5.5.4 ข้อสังเกต

เนื่องจากแผนภาพกิจกรรมตัวอย่างไม่มีการกำหนดการทำงานหลังจากแอ็คชั่น “Notify of Modification” ทำให้สามารถอนุมานได้ว่าจะเกิดการหยุดสายงานขึ้นหลังจากดำเนินงานแอ็คชั่นดังกล่าว ดังนั้นความหมายของแผนภาพกิจกรรมตัวอย่างและแผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์จึงมีความหมายเหมือนกัน อย่างไรก็ตามจากจุดนี้จะเห็นว่าแผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์จะอธิบายการทำงานได้ชัดเจน เพราะไม่ต้องใช้การอนุมานแต่อย่างใด

5.6 ผลการเปรียบเทียบตัวอย่างที่ 5

5.6.1 คำอธิบาย

ตัวอย่างที่ 5 แสดงให้เห็นถึงจุดเริ่มต้นการทำงาน ซึ่งสามารถอนุมานได้จากแอ็คชั่นที่ไม่เป็นแอ็คชั่นปลายทางของแอ็คชั่นใด ๆ (สามารถเป็นแอ็คชั่นปลายทางของแอ็คชั่นตนเองได้)



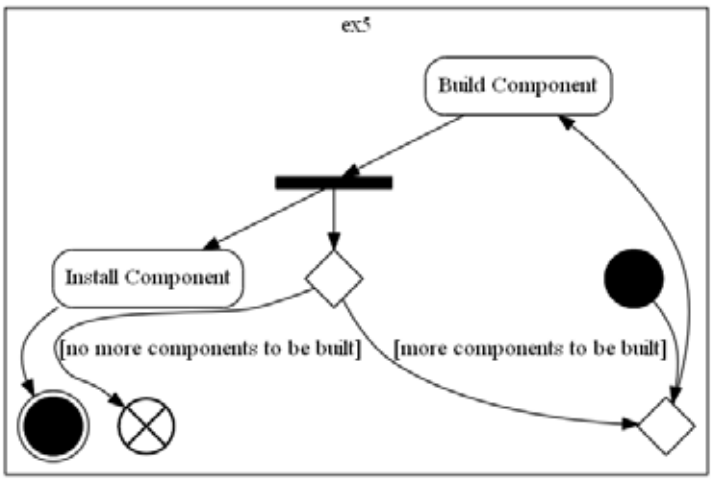
ภาพที่ 40 แผนภาพกิจกรรมตัวอย่างที่ 5

5.6.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'ex5'
      buildComponent->installComponent

      decision from buildComponent
            if 'no more components to be built' then break
            else
                  if 'more components to be built'
                  then buildComponent
                  endif
            endif
      end
end
```

5.6.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 41 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 5

5.6.4 ข้อสังเกต

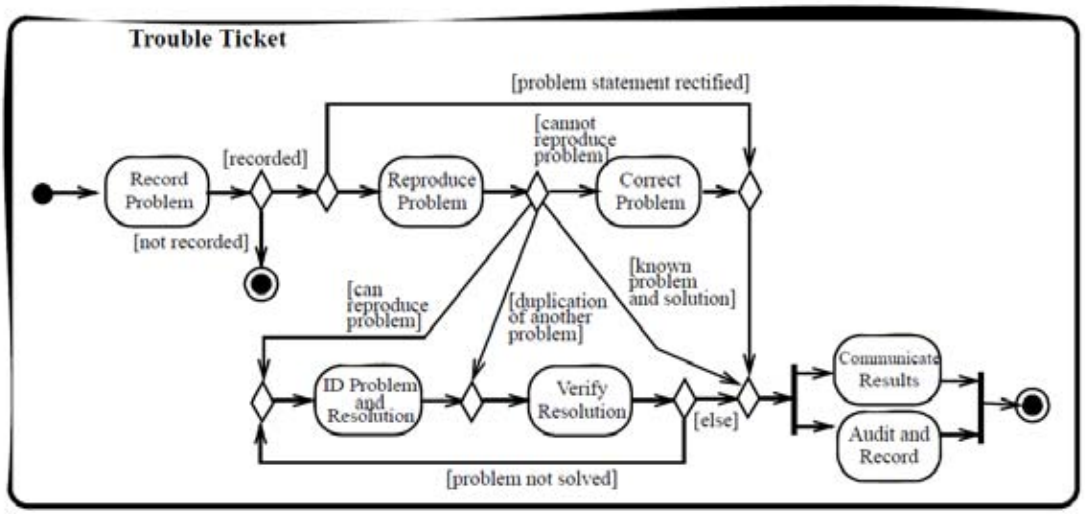
ผลลัพธ์ในเชิงความหมายของการดำเนินงานที่ได้จากแผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์จะเหมือนกันกับแผนภาพกิจกรรมตัวอย่าง แต่แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์จะมีการเพิ่มบัพผสาน (จากการวิเคราะห์เส้นเชื่อมเข้าที่ระบุไว้ในบทที่ 3) เข้ามาก่อนการ

ทำงานแอ็คชั่น “Build Component” ซึ่งเป็นการบ่งบอกว่าเมื่อมีโทเค็นให้สามารถดำเนินการผ่านได้ทันที จึงทำให้การทำงานของแผนภาพกิจกรรมทั้งสองเหมือนกัน

5.7 ผลการเปรียบเทียบตัวอย่างที่ 6

5.7.1 คำอธิบาย

ตัวอย่างที่ 6 แสดงให้เห็นถึงกิจกรรมที่มีการระบุเงื่อนไขการตัดสินใจเป็นลำดับชั้น ซึ่งจากตัวอย่างภาพที่ 42 แสดงให้เห็นถึงความยากในการออกแบบแผนภาพกิจกรรมให้มีความถูกต้องตามต้องการ



ภาพที่ 42 แผนภาพกิจกรรมตัวอย่างที่ 6

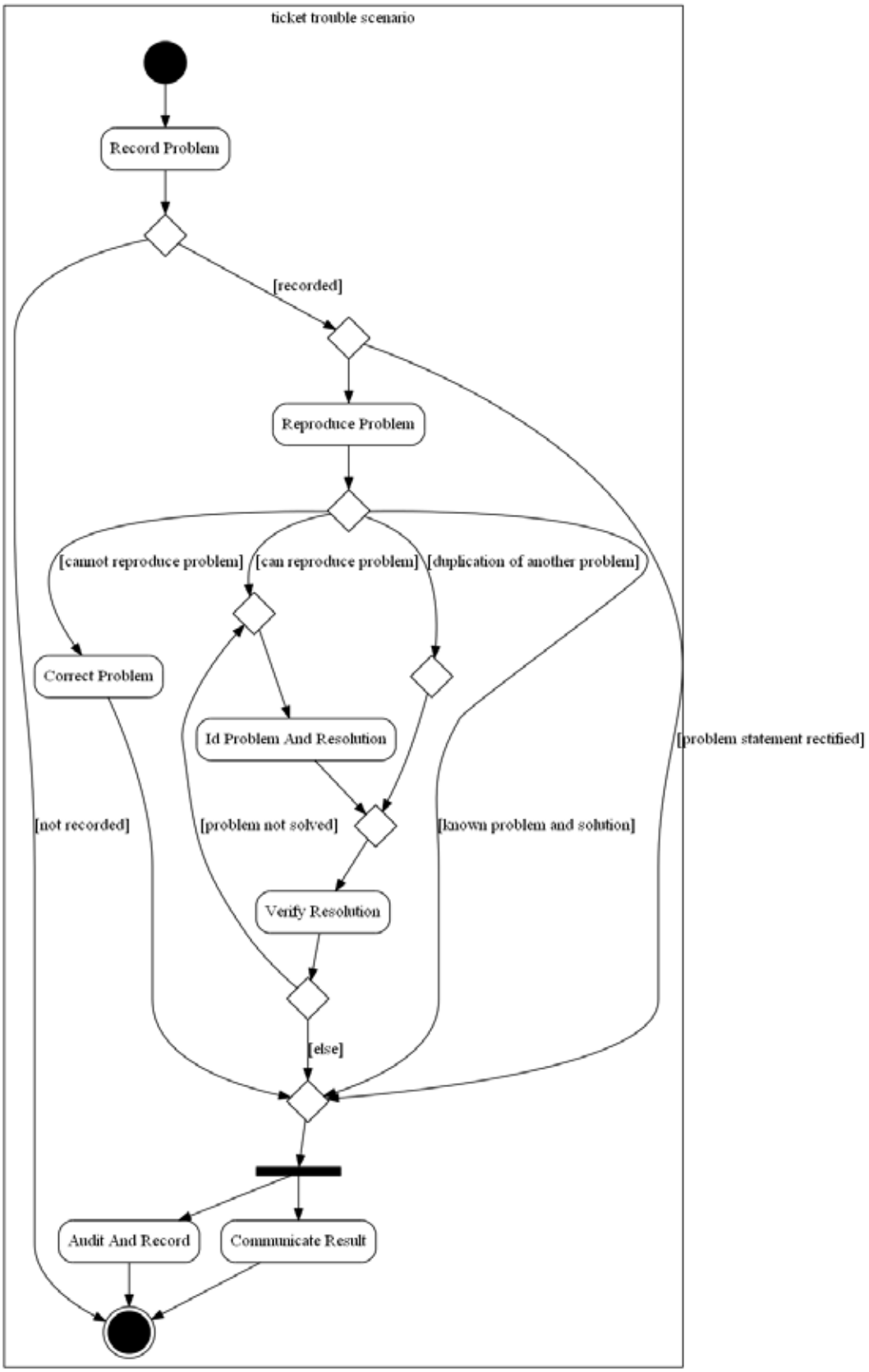
5.7.2 ข้อมูลอินพุตสำหรับทดสอบ

จากตัวอย่างข้างต้นสามารถเขียนให้อยู่ในรูปบทคำสั่งเอดีแอลได้ดังต่อไปนี้

```
diagram 'ticket trouble scenario'
      action recordProblem end
      action reproduceProblem end
      action correctProblem end
      action idProblemAndResolution end
      action verifyResolution end
      action auditAndRecord end
      action communicateResult end

      decision from recordProblem
            if 'recorded' then
            reproduceProblem
                  if 'problem statement rectified'
                  then auditAndRecord and communicateResult
                  endif
            else
```

```
                  if 'not recorded' then terminate endif
            endif
      end

      decision from reproduceProblem
            if 'cannot reproduce problem' then correctProblem
            else
                  if 'can reproduce problem'
                  then idProblemAndResolution
                  endif
            else
                  if 'duplication of another problem'
                  then verifyResolution
                  endif
            else
                  if 'known problem and solution'
                  then auditAndRecord and communicateResult
                  endif
            endif
      end

      idProblemAndResolution -> verifyResolution

      correctProblem -> auditAndRecord
      correctProblem -> communicateResult

      decision from verifyResolution
            if 'problem not solved' then idProblemAndResolution
            else
                  auditAndRecord, communicateResult
            endif
      end
end
```

5.7.3 ผลลัพธ์จากซอฟต์แวร์ของระบบ



ภาพที่ 43 แผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์ตัวอย่างที่ 6

5.7.4 ข้อสังเกต

แผนภาพกิจกรรมตัวอย่างและแผนภาพกิจกรรมจากผลลัพธ์ของซอฟต์แวร์นั้นมีองค์ประกอบของแผนภาพกิจกรรมที่ใช้ในการสร้างเงื่อนไขการตัดสินใจเหมือนกัน ซึ่งเป็นตัวบ่งชี้ให้เห็นว่าเอดีแอลสามารถใช้อธิบายแผนภาพกิจกรรมที่มีการตัดสินใจเป็นลำดับชั้นได้อย่างถูกต้อง อย่างไรก็ตามพบว่าเอดีแอลไม่สามารถอธิบายบัพรวมก่อนการสิ้นสุดกิจกรรมได้ เนื่องจากเราได้อนุมานไว้ว่าแอ็คชั่นที่ไม่เป็นต้นทางของแอ็คชั่นใด ๆ คือจุดสิ้นสุดกิจกรรม ดังนั้นจึงทำให้การสร้างบัพดังกล่าวมีความคลาดเคลื่อนไปจากที่ต้องการ สำหรับวิธีการแก้ปัญหาดังกล่าวสามารถทำได้โดยการทำให้ “Audit And Record” และ “Communicate Result” ไม่ใช่จุดสุดท้ายของกิจกรรม กล่าวคือ ต้องสร้างแอ็คชั่นต้องมารองรับการสร้างบัพรวมดังกล่าว

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอวิธีการเชิงป้องกันความผิดพลาดสำหรับการสร้างแผนภาพกิจกรรมด้วยการสร้างจากเอดีแอลซึ่งจัดอยู่ในประเภทดีเอสแอล วิธีการสร้างแผนภาพกิจกรรมที่นำเสนอจะสามารถลดทรัพยากรที่ใช้ในการออกแบบ และเพิ่มความตรงกันของพฤติกรรมในระบบได้

ซึ่งจากการทดสอบการสร้างแผนภาพในบทที่ 5 ชี้ให้เห็นกว่าเอดีแอลสามารถใช้อธิบายพฤติกรรมต่าง ๆ ของกิจกรรมได้อย่างถูกต้องในการอธิบายความหมายของลำดับการทำงานในแผนภาพกิจกรรม และการสร้างตัวควบคุมต่าง ๆ ที่จำเป็นต่อการอธิบายลำดับการทำงาน โดยระบบที่พัฒนาขึ้นสามารถสร้างตัวควบคุมสำหรับกำหนดพฤติกรรมของระบบตามหลักการสร้างบัพควบคุมจากบทที่ 3 และได้ผลลัพธ์ที่ถูกต้องตามข้อปฏิบัติ

6.2 ข้อจำกัด

1. การแสดงผลภาพจากกราฟวิซไม่สามารถควบคุมได้ ทำให้บางครั้งแผนภาพที่ออกมาอ่านยากกว่าที่ควรเป็น
2. แผนภาพที่ออกมาเป็นรูปภาพทำให้ไม่สามารถแก้ไขข้อมูลได้
3. งานวิจัยชิ้นนี้ยังไม่สามารถร้อยแผนภาพกิจกรรมที่มีมากกว่าหนึ่งกิจกรรมได้
4. ซอฟต์แวร์ยังไม่รองรับการใช้งานภาษาไทย แต่รองรับความสามารถในการระบุชื่อที่แตกต่างจากข้อมูลชื่อของตัวระบุได้

6.3 แนวทางการวิจัยต่อ

- สำหรับการอธิบายแผนภาพใด ๆ งานวิจัยนี้ชี้ให้เห็นว่าเราสามารถออกแบบดีเอสแอลสำหรับแผนภาพใด ๆ ที่สนใจได้ โดยการสกัดเอนทิตีที่เกี่ยวข้องกับแผนภาพนั้น ๆ ออกมา
- สำหรับการลงรายละเอียดของแผนภาพกิจกรรม เนื่องจากเอดีแอลยังไม่สามารถอธิบายแผนภาพกิจกรรมเชิงโครงสร้างที่มีการสร้างการขัดจังหวะกิจกรรมหรือมีการเรียกเหตุการณ์ได้ ดังนั้นการปรับปรุงเอดีแอลเพื่อให้รองรับกับความสามารถดังกล่าวจะช่วยให้เอดีแอลสามารถใช้งานได้ในหลายโดเมนมากขึ้น
- สำหรับการร้อยแผนภาพกิจกรรม เอดีแอลที่ออกแบบขึ้นในงานวิจัยนี้ยังไม่รวมถึงแผนภาพกิจกรรมระดับที่ความซับซ้อนหรือที่มีมากกว่าหนึ่งกิจกรรม ถ้าเราสามารถ

เชื่อมกิจกรรมมากกว่าหนึ่งกิจกรรมเข้าด้วยกันได้จะสามารถช่วยเพิ่มคุณภาพของแผนภาพกิจกรรมในภาพรวมได้เป็นอย่างดี อย่างไรก็ตามการร้อยกิจกรรมควรจะมีการบริหารการจัดการกิจกรรมต่าง ๆ ด้วย

- สำหรับการนำไปใช้งานเชิงธุรกิจ เราสามารถนำการแปลงแบบจำลองความหมายเข้ามาประยุกต์ใช้ในการสร้างผลลัพธ์ไฟล์เอ็กซ์เอ็มแอลสำหรับโปรแกรมที่ต้องการได้ เพื่อนำไปใช้ประโยชน์ต่อไป อาทิเช่น ใช้ในการดำเนินงานกระแสงาน ใช้ในการออกแบบแผนภาพสำหรับโปรแกรมที่รองรับการนำเข้าไฟล์เอ็กซ์เอ็มไอ เป็นต้น

รายการอ้างอิง

[1] Object Management Group, Inc. Unified Modeling Language™ (OMG UML): Superstructure Version 2.3 [Online]. 2010. Available from : http://www.omg.org/spec/UML/2.3/Superstructure/PDF/ [2011, May 17]

[2] Taha, W. Domain-Specific Languages [Online]. Available from : http://www.cs.rice.edu/~taha/ publications/conference/icces08.pdf [2012, January 20]

[3] Fowler, M. Domain-Specific Languages. Addison-Wesley Professional, 2010.

[4] Ghosh, D. DSLs in Action. Manning Publications Co., 2011.

[5] Flater D., Martin P. A., and Crane M. L., Rendering UML Activity Diagrams as Human-Readable Text. National Institute of Standards and Technology, 2007.

[6] Xu Y. J. The formal semantics of UML activity diagram based on Process Algebra. Computer Science and Service System (CSSS) (2011) : 2729-2732.

[7] Xu D., Miao H., and Phibert N. Model Checking UML Activity Diagrams in FDR. Computer and Information Science (ICIS 2009). Eighth IEEE/ACIS Internation Conference (2009) : 1035-1040.

[8] Baeten J. C. M, Basten T., and Reniers M. A. Process Algebra: Equational Theories of Commnicating Processes. Cambridge University Press, 2010.

[9] Object Management Group, Inc. Meta Object Facility (MOF) Core Specification Version 2.0 [Online]. 2006. Available from http://www.omg.org/spec/MOF/2.0/PDF/ [2011, December 3]

[10] ANTLR Parser Generator [Online]. 2012. Available from http://www.antlr.org/ [2012, January 12]

[11] PlantUML [Online]. 2012. Available from http://plantuml.sourceforge.net [2012, January 12]

[12] yUML [Online]. 2012. Available from http://yuml.me [2012, Febuary 2]

[13] Prinz A., Scheidgen M., and Tveit M. S. A Model-Based Standard for SDL. In SDL 2007, LNCS 4745, pp.1-18. Heidelberg : Springer-Verlag, 2007.

[14] Gjøsaeter T., Isfeldt I. F., and Prinz A. Sudoku – A Language Description Case Study. In SLE 2008, LNCS 5452, pp.305-321. Berlin Heidelberg : Springer-Verlag, 2009.

[15] Grune D., and Jacobs C. Parsing Techniques: A Practical Guide, 2nd ed. Springer, 2008.

[16] Object Management Group, Inc. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification Version 1.1 [Online]. 2011. Available from : http://www.omg.org/spec/QVT/1.1/PDF/ [2012, March 23]

[17] Bang-Jensen J., and Gutin G. Digraphs: Theory, Algorithms and Applications. Springer-Verlag, 2007.

[18] Börger E., and Stärk R. Abstract State Machines: A Method for High-Level System Design and Analysis. Heidelberg : Springer-Verlag, 2003.

[19] Booch G., Rumbaugh J., and Jacobson I. The Unified Modeling Language User Guide, 2nd ed. Addison-Wesley Professional, 2005.

[20] Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Addison-Wesley, 2003.

[21] Spinellis D. Notable Design Patterns for Domain Specific Languages. Journal of Systems and Software 56 (2001) : 91-99.

ภาคผนวก

ภาคผนวก ก.
สัญกรณ์กราฟิกและความหมายที่เกี่ยวข้องกับแผนภาพกิจกรรม

ตัวรับสัญญาณ (AcceptEventAction)

ลักษณะประจำ

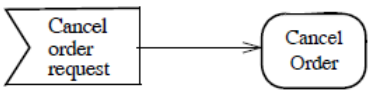
ไม่มีลักษณะประจำ

ข้อจำกัด

ไม่มีข้อจำกัด

ตัวอย่างการใช้งานและความหมาย

ในภาพที่ 44 แสดงให้เห็นถึงการยกเลิกการสั่งซื้อ ซึ่งการรับสัญญาณจะมาจากการเรียกการทำงานในส่วนของการร้องขอยกเลิกการสั่งซื้อ ทั้งนี้ทั้งนั้นตัวรับสัญญาณจะไม่มีเส้นเชื่อมเข้าปรากฏอยู่

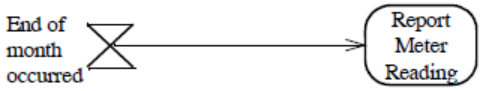


ภาพที่ 44 การรับสัญญาณในระดับบนสุด

ในภาพที่ 45 กระบวนการชำระเงินถูกเรียกหลังจากมีการสั่งซื้อ ดังนั้นตัวกิจกรรมถัดไปจะทำการรอจนกว่าจะมีการจ่ายเงิน (รอจนกว่าได้รับการยืนยันการจ่ายเงิน) โดยการยืนยันการจ่ายเงินจะถูกส่งหลังจากที่มีการชำระเงินแล้วเท่านั้น เมื่อได้การชำระเงินได้รับการยืนยันแล้วจึงจะมีการส่งของในกระบวนการถัดไป

ภาพที่ 45 การรับสัญญาณแบบแสดงให้เห็นถึงการทำงานอย่างชัดแจ้ง (Explicit Enable)

ในภาพที่ 46 แสดงตัวอย่างการสร้างสัญญาณสิ้นสุดของเดือน ซึ่งจะทำงานเมื่อถึงเวลาสิ้นเดือน (ในทุก ๆ สิ้นเดือน)



ภาพที่ 46 การทำงานที่เกี่ยวข้องกับเวลา

แอ็คชั่น (Action)

แอ็คชั่น (Action) ใช้สำหรับอธิบายถึงหนึ่งขั้นตอนภายในแผนภาพกิจกรรมเท่านั้น กล่าวคือจะไม่สามารถมีการแตกย่อยของแอ็คชั่นได้อีก โดยการทำงานของแอ็คชั่นสามารถที่จะเรียกกิจกรรมถัดไปให้ทำงานหรือให้ตัวมันเองทำงานอีกครั้งก็ได้ อย่างไรก็ตามแอ็คชั่นอาจดูไม่ซับซ้อนและดูเรียบง่ายในภาพรวมของแผนภาพกิจกรรม แต่ผู้ออกแบบยังคงต้องตระหนักเสมอว่าการประกาศแอ็คชั่นที่ไม่ได้มีการวางแผนไว้ก่อนอาจเป็นส่วนทำให้แผนภาพกิจกรรมมีความซับซ้อนและเกิดความสับสนได้ง่าย นอกจากนี้ลักษณะการทำงานของแอ็คชั่นในความเป็นจริงนั้นอาจไม่ได้ทำงานในลักษณะเชิงหน่วย (Atomic) เสมอไป

แอ็คชั่นสามารถที่จะประกอบด้วยเส้นเชื่อมเข้าและเส้นเชื่อมออกซึ่งอาจจะมีมากกว่าหนึ่งเส้นก็ได้ โดยเส้นเชื่อมเหล่านั้นจะทำหน้าที่เป็นสายงานควบคุม (Control Flow) และสายงานข้อมูล (Data Flow) จากบัพ (Node) หนึ่งไปยังอีกบัพหนึ่ง

แอ็คชั่นในแต่ละแอ็คชั่นจะทำงานก็ต่อเมื่อได้รับข้อมูลนำเข้าทั้งหมดตามเงื่อนไขที่ระบุไว้ โดยเมื่อการทำงานของแอ็คชั่นหนึ่ง ๆ สิ้นสุดลงจะกระตุ้นให้เกิดการทำงานของแอ็คชั่นถัดไป และข้อมูลนำเข้าที่ได้รับจะมาจากข้อมูลนำออกของแอ็คชั่นก่อนหน้า

ลักษณะประจำ

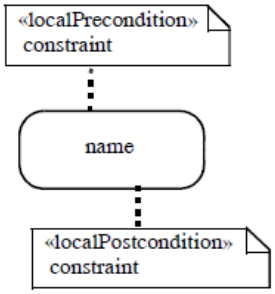
- isLocallyReentrant : Boolean [1..1] = false
  ถ้าเป็น true แอ็คชั่นจะสามารถเริ่มต้นใหม่อีกครั้งในลักษณะที่เป็นการทำงานขนานกันไปถึงแม้ว่าแอ็คชั่นก่อนหน้ายังทำงานไม่เสร็จ แต่ถ้าเป็น false แอ็คชั่นจะไม่สามารถเริ่มทำงานใหม่ได้จนกระทั่งแอ็คชั่นที่ทำงานก่อนหน้าจะทำงานเสร็จเสียก่อน

ข้อจำกัด

- localPrecondition : Constraint [0..*]
  จะต้องเป็นไปตามเงื่อนไขข้อบังคับที่ระบุไว้ทั้งหมดเสียก่อนถึงจะเริ่มงานได้
- localPostcondition : Constraint [0..*]
  จะถือว่าการทำงานเสร็จสิ้นก็ต่อเมื่อเป็นตามเงื่อนไขข้อบังคับที่ระบุไว้ทั้งหมด

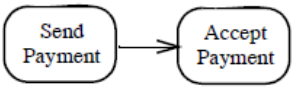
ตัวอย่างการใช้งานและความหมาย

ใกรณีที่แอ็คชั่นมีการระบุเงื่อนไขก่อนและหลังการทำงาน ให้มีการกำหนดลักษณะตามภาพที่ 47 ซึ่งในส่วนของการระบุเงื่อนไขจะใช้เครื่องหมายเดียวกับสัญลักษณ์หมายเหตุ (Note) แต่จะมีคำว่า <<localPrecondition>> และ <<localPostcondition>> กำกับอยู่



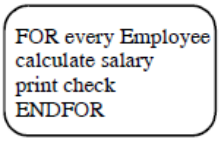
ภาพที่ 47 ส่วนแสดงที่มีการระบุเงื่อนไข

ในภาพที่ 48 เป็นการแสดงการทำงานของแอ็คชั่น โดยการทำงานที่เกิดขึ้นนี้แสดงให้เห็นว่ามีการแจ้งชำระเงินก่อนแล้วจึงมีการรับชำระตามลำดับ



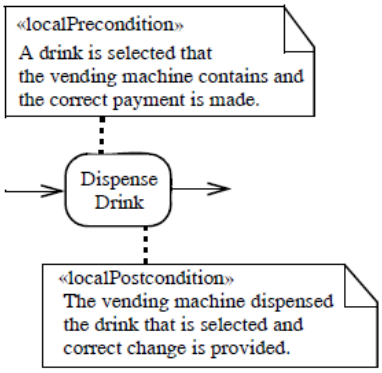
ภาพที่ 48 ตัวอย่างของแอ็คชั่น

ในกรณีที่แอ็คชั่นต้องการแสดงให้เห็นถึงการทำงานบางอย่างที่เกี่ยวข้องกับโปรแกรมประยุกต์ (Application) ในเชิงอิสระทางภาษาให้ระบุส่วนแสดงในลักษณะตามภาพที่ 49



ภาพที่ 49 ส่วนแสดงกับการระบุการทำงานที่เกี่ยวข้องกับโปรแกรมประยุกต์

สำหรับการกำหนดเงื่อนไขให้กับแอ็คชั่นนั้น จะสามารถกำหนดได้ก็ต่อเมื่อเป็นเงื่อนไขที่เกี่ยวข้องกับแอ็คชั่นโดยตรงเท่านั้น ยกตัวอย่างเช่น กิจกรรมที่เกี่ยวข้องกับเครื่องขายของอัตโนมัติ เมื่อช่างเทคนิคต้องการที่จะนำของออกมาจากเครื่องจ่ายของอัตโนมัติจะสามารถทำได้อยู่สองวิธี คือ วิธีที่หนึ่งช่างเทคนิคใช้กุญแจเปิดตู้ขายเพื่อที่จะหยิบของออกมา (ซึ่งวิธีนี้จะเห็นว่าไม่จำเป็นต้องมีการหยอดเหรียญหรือการคำนวนเงินทอนแต่อย่างใด) สำหรับอีกวิธีหนึ่งคือการซื้อของผ่านเครื่อง ซึ่งสามารถทำได้โดยการหยอดเหรียญลงไปในเครื่องแล้วเลือกสินค้า หลังจากนั้นเครื่องจะคำนวนเงินทอนที่ถูกต้องเพื่อให้เงินทอนในขั้นตอนถัดไป ซึ่งเมื่อพิจารณาแล้วจะเห็นว่าเงื่อนไขที่เกี่ยวข้องกับการจ่ายสินค้าออกจากตู้นั้นคือ ก่อนจ่ายสินค้าจะต้องมีการเลือกสินค้าและใส่จำนวนเงินที่ถูกต้องลงไปเสียก่อน และกระบวนการจ่ายสินค้าจะเสร็จสิ้นก็ต่อเมื่อสินค้าที่ถูกเลือกได้จ่ายออกไปแล้วและมีการคำนวนเงินทอน เพื่อทอนเงินนั้นตอนถัดไป ดังตัวอย่างภาพที่ 50



ภาพที่ 50 ส่วนแสดงกับการระบุเงื่อนไขที่เกี่ยวข้อง

กิจกรรม (Activity)

กิจกรรมคือการกำหนดการทำงานเชิงตัวแปร โดยการทำงานเหล่านั้นคือการแสดงถึงลำดับการกระทำของแอ็คชั่นต่าง ๆ สำหรับการเรียกกิจกรรมให้ทำงานนั้นสามารถเรียกได้สองทาง คือทางตรงและทางอ้อม, ซึ่งการเรียกทางตรงคือการเรียกผ่านลำดับสายงานปกติ ส่วนทางอ้อมนั้นจะเป็นการเรียกผ่านการทำงานของเหตุการณ์บางอย่างซึ่งสามารถกระตุ้นกิจกรรมให้เริ่มทำงานได้

ลักษณะประจำ

BasicActivities

isReadOnly : Boolean [1..1] = false

ถ้าค่าเป็น true กิจกรรมนี้จะต้องไม่กระทำการใด ๆ ที่ส่งผลให้เกิดการเปลี่ยนแปลงค่าวัตถุที่อยู่นอกขอบเขตของกิจกรรม แต่ถ้าค่าเป็น false หมายถึงกิจกรรมนี้สามารถที่เปลี่ยนค่าวัตถุภายนอกหรือไม่ก็ได้

CompleteActivities

- isSingleExecution : Boolean [1..1] = false

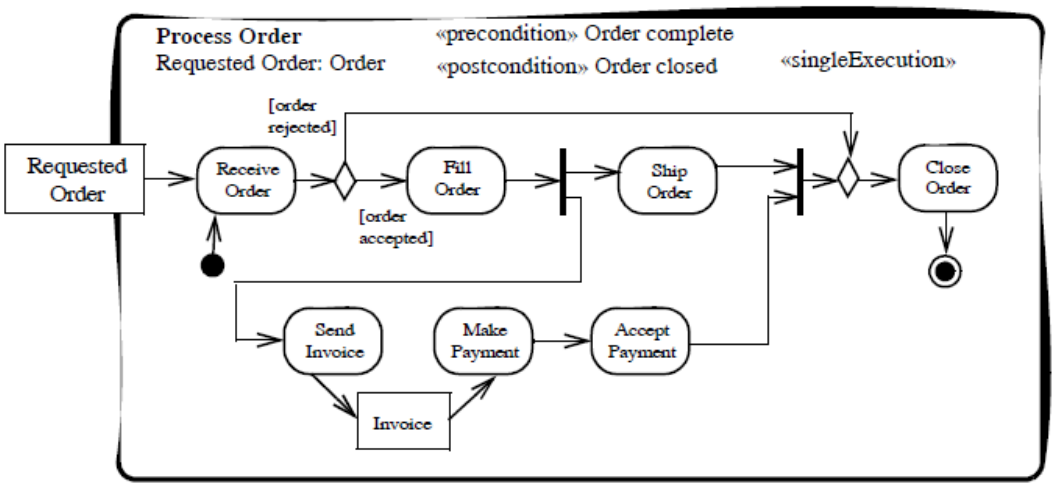
ถ้าเป็น true หมายถึงการเรียกให้แอ็คชั่นต่าง ๆ ทำงานพร้อมกัน โดยกิจกรรมจะต้องรอข้อมูลนำเข้าทั้งหมดให้พร้อมในแต่ละแอ็คชั่นเสียก่อนจึงจะสามารถสั่งให้แอ็คชั่นทำงานได้ ลักษณะการทำงานเช่นนี้กล่าวได้ว่าเป็นการรอเพื่อให้มีการกระตุ้นการทำงานเพียงครั้งเดียว

ข้อจำกัด

1. บัพกิจกรรมจะต้องมี ActivityParameterNode ต่อหนึ่งพารามิเตอร์เท่านั้น และไม่สามารถใช้ ActivityParameterNode ร่วมกันได้
2. เมื่อมีการเรียกกิจกรรมแบบอิสระ (แบบทางตรง) แล้วจะไม่สามารถเรียกกิจกรรมด้วยคุณลักษณะการทำงานได้ (แบบทางอ้อม) กล่าวคือสามารถเรียกกิจกรรมได้แบบใดแบบหนึ่งเท่านั้น
3. จะต้องไม่มีกลุ่มกิจกรรมซ้อนกันเป็นลำดับชั้น

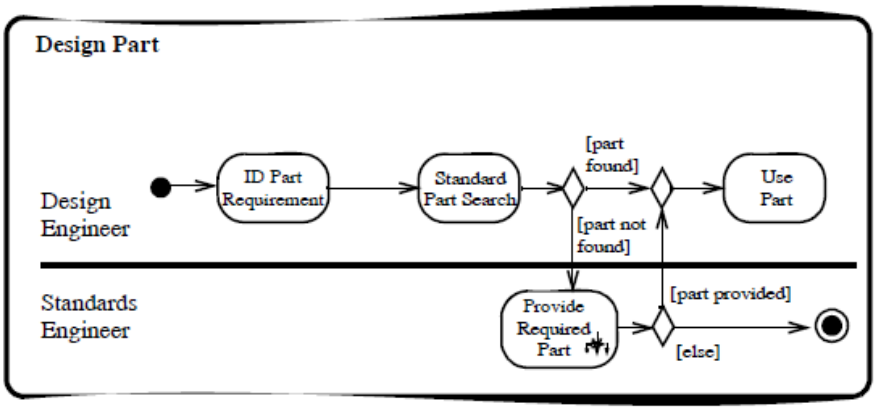
ตัวอย่างการใช้งานและความหมาย

การกำหนดกิจกรรมสามารถใช้เส้นขอบมาเป็นตัวระบุขอบเขตของกิจกรรมได้ ตัวอย่างภาพที่ 51 เป็นการกำหนดกิจกรรมของการสั่งซื้อแบบมีเงื่อนไขและตัวแปรนำเข้า

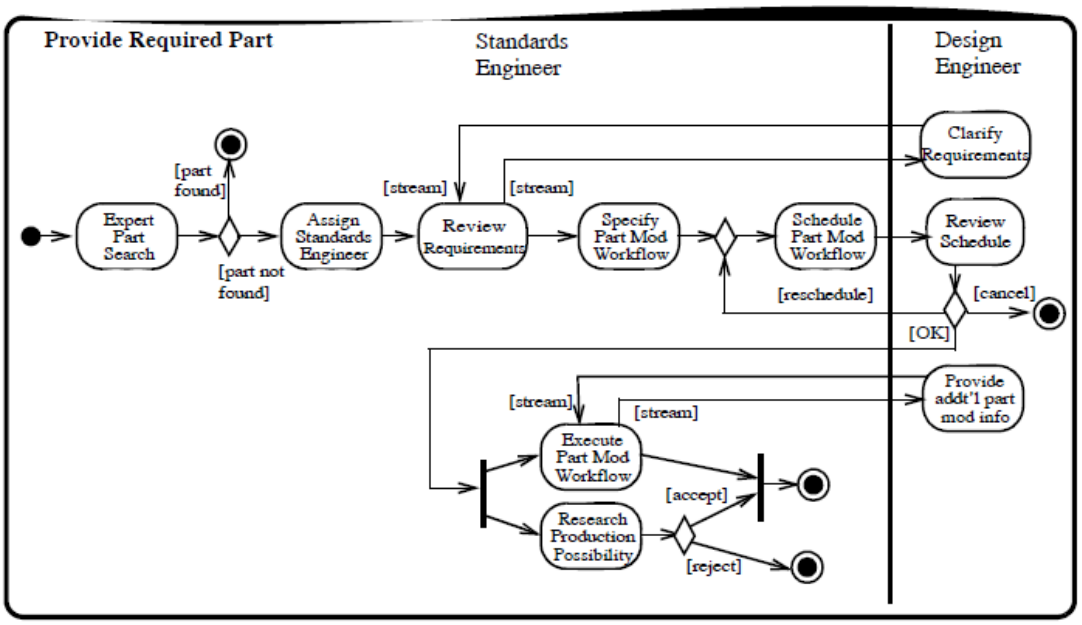


ภาพที่ 51 ตัวอย่างกิจกรรมที่มีการระบุตัวแปรนำเข้า

แผนภาพภาพที่ 52 และภาพที่ 53 แสดงถึงกระบวนการทำงานของการเลือกอะไหล่มาตราฐานในการออกแบบสายการบิน มีการใช้การแบ่งส่วน (Partition) เพื่อระบุว่าขั้นตอนไหนใครเป็นผู้ดำเนินการ ซึ่งวิศวกรมาตรฐานจะต้องมั่นใจว่าขั้นตอนกระบวนการในการจัดหาอะไหล่ที่ต้องการ เป็นไปตามลำดับขั้นตอนและเงื่อนไขที่กำหนดไว้ โดยขั้นตอนจัดหาอะไหล่จะไม่ได้ทำโดยวิศวกรมาตรฐานแต่จะทำโดยวิศวกรออกแบบแทน ในการจัดหาอะไหล่มีความเป็นไปได้ที่จะหาพบหรือไม่พบก็ได้ ซึ่งในกรณีที่หาไม่พบวิศวกรออกแบบจะต้องส่งมอบงานกลับไปยังวิศวกรมาตรฐานเพื่อดำเนินการดัดแปลงอะไหล่ ซึ่งในขั้นตอนการดัดแปลงอะไหล่มีความจำเป็นที่จะต้องมีการวางแผน จัดทำและวิจัย (ขั้นตอนการจัดและการวิจัยสามารถทำควบคู่กันไปได้) เพื่อให้ได้อะไหล่ที่เป็นไปตามความต้องการและทันต่อการใช้งาน



ภาพที่ 52 กระบวนการออกแบบอะไหล่



ภาพที่ 53 กระบวนการจัดหาอะไหล่

<u>เส้นเชื่อมกิจกรรม (ActivityEdge)</u>

เส้นเชื่อมกิจกรรม คือ การเชื่อมกิจกรรมสองกิจกรรมด้วยเส้นที่มีการกำหนดทิศทางการไหล

ลักษณะประจำ

ไม่มีลักษณะประจำ

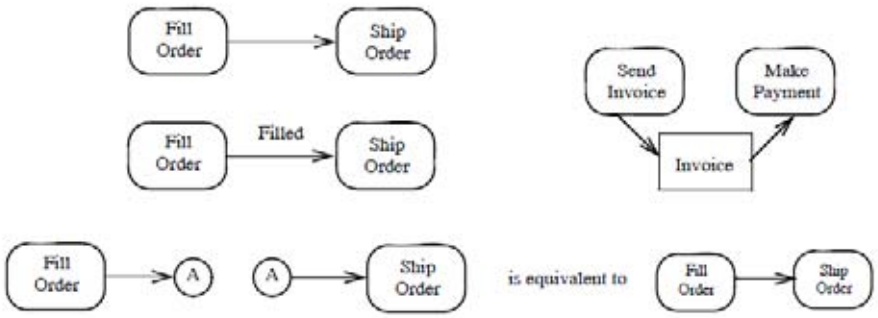
ข้อจำกัด

1. ต้นทางและปลายทางของเส้นเชื่อมจะต้องอยู่ภายในกิจกรรมเดียวกัน
2. เส้นเชื่อมกิจกรรมจะสามารถมีกิจกรรมหรือกลุ่มเป็นเจ้าของได้เท่านั้น CompleteStructuredActivities
3. เส้นเชื่อมกิจกรรมจะสามารถมีบัฟโครงสร้างเป็นเจ้าของได้มากสุดหนึ่งจุดเท่านั้น

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างเส้นเชื่อมกิจกรรมภาพที่ 54 แสดงให้เห็นถึงลักษณะความเป็นไปได้ในการเชื่อมกิจกรรมด้วยกันสามแบบ คือ แบบที่หนึ่งจากตัวอย่างด้านซ้ายบนเป็นการบอกลำดับการทำงานด้วยสายงานควบคุม (Control Flow) ซึ่งจากตัวอย่างแสดงให้เห็นว่าจะต้องมีการกรอกใบสั่งของก่อนถึงจะมีการจัดส่งสินค้าตามใบสั่งของได้ แบบที่สองจากตัวอย่างด้านล่างจะมี

ความหมายเช่นเดียวกับแบบที่หนึ่ง แต่จะเป็นการชี้ให้เห็นว่าเส้นเชื่อมสามารถลากถึงกันโดยใช้จุดต่อ (Connector) เป็นตัวเชื่อมได้ ส่วนแบบที่สามจากตัวอย่างด้านขวาบนเป็นการแสดงให้เห็นถึงลำดับการทำงานผ่านสายงานวัตถุ (Object Flow) โดยการใช้วัตถุเป็นตัวเชื่อมกิจกรรม ในตัวอย่างมีวัตถุคือใบส่งของ ซึ่งแสดงให้เห็นว่าการที่จะสามารถจ่ายเงินได้นั้นจะต้องใช้ใบส่งของเป็นตัวยืนยันกิจกรรมก่อนหน้า



ภาพที่ 54 ตัวอย่างเส้นเชื่อมกิจกรรม

<u>บัพสิ้นสุดกิจกรรม</u> <u>(ActivityFinal)</u>

บัพสิ้นสุดกิจกรรมใช้สำหรับหยุดกิจกรรม (สายงาน) ทั้งหมดในกิจกรรมนั้น ๆ

ลักษณะประจำ

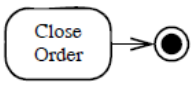
ไม่มีลักษณะประจำ

ข้อจำกัด

ไม่มีข้อจำกัด

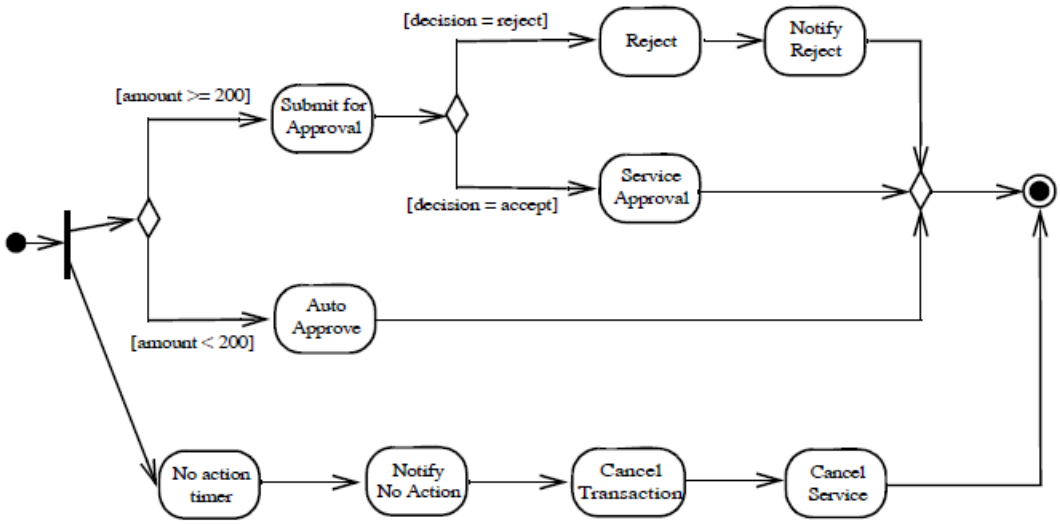
ตัวอย่างการใช้งานและความหมาย

จากตัวอย่าง**ภาพที่** 55 เป็นการบอกว่าให้หยุดการทำงาน (โทเค็น) ทั้งหมดหลังจากมีการปิดการสั่งซื้อ กล่าวคือโทเค็นทั้งหมดจะถูกทำลายลงหลังจากที่มีการปิดการสั่งซื้อ เมื่อโทเค็นถูกทำลายลงกระบวนการก็จะไม่สามารถดำเนินการต่อได้



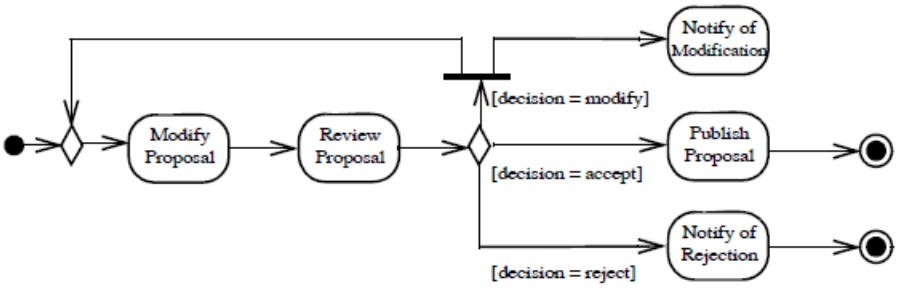
ภาพที่ 55 ตัวอย่างการสิ้นสุดกิจกรรม

จากภาพที่ 56 กิจกรรมจะมีสองสายงานทำงานลักษณะขนานกันไป เป็นการบ่งบอกว่าสองสายงานนี้ทำงานในเชิงแข่งขันเพื่อให้ไปถึงจุดสิ้นสุดกิจกรรม หากสายงานใดสายงานหนึ่งถึงจุดสิ้นสุดกิจกรรม อีกสายงานหนึ่งจะถือว่าสิ้นสุดทันที ซึ่งจากรูปจะเห็นว่าสายงานแรกคือการรับรองส่วนอีกสายงานจะเป็นการตรวจสอบการรับรองว่าเป็นไปตามกำหนดเวลาหรือไม่ หากเลยกำหนดเวลา (No Action Timer) ขั้นตอนการรับรองที่ทำงานอยู่ให้ถือว่าเป็นการสิ้นสุดหรือยกเลิกการดำเนินการรับรองทันที



ภาพที่ 56 ตัวอย่างการหยุดกิจกรรม

จากภาพที่ 57 เป็นการใช้บัพสิ้นสุดกิจกรรมกับการตัดสินใจ (Decision) ซึ่งจะแตกต่างจากตัวอย่างภาพที่ 56 ที่เป็นการทำงานแบบขนาน จากรูปจะเห็นว่ามีการใช้บัพสิ้นสุดกิจกรรมสองอัน (ซึ่งมีความหมายเช่นเดียวกันกับการใช้บัพสิ้นสุดกิจกรรมเพียงอันเดียว) โดยกิจกรรมจะเกี่ยวข้องกับการนำเสนอโครงงาน แล้วมีการตัดสินใจที่จะยอมรับ กลับไปแก้ไข หรือปฏิเสธโครงงานอย่างใดอย่างหนึ่ง ในกรณีที่มีการส่งกลับไปแก้ไขจะมีการทำงานสองส่วนขนานกัน คือการแก้ไขโครงงานและการแจ้งให้ทราบว่ามีการแก้ไขโครงงาน ซึ่งในส่วนของการแจ้งให้ทราบนั้นจะต้องใช้เวลาที่ไม่นานเกินไปหรือจะต้องเสร็จสิ้นก่อนที่จะมีการสิ้นสุดกิจกรรม



ภาพที่ 57 ตัวอย่างการหยุดกิจกรรมที่มีบัพสิ้นสุดกิจกรรมมากกว่าหนึ่งอัน

<u>บัพกิจกรรม (ActivityNode)</u>

ลักษณะประจำ

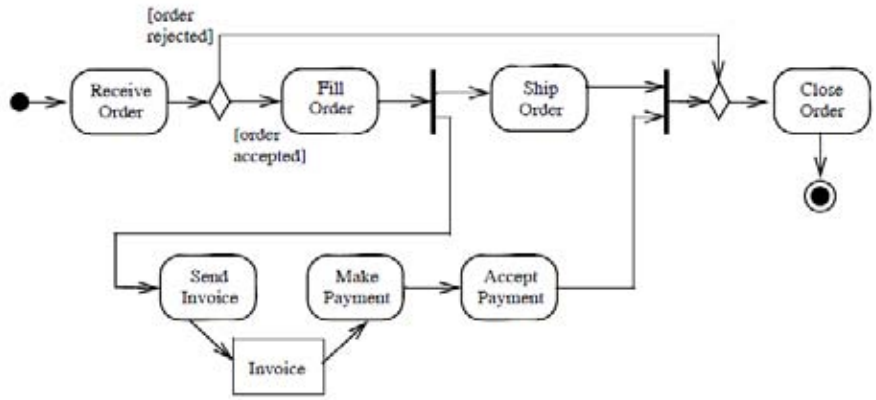
ไม่มีลักษณะประจำ

ข้อจำกัด

1. บัพกิจกรรมจะมีกิจกรรมหรือกลุ่มกิจกรรมเป็นเจ้าของได้เท่านั้น StructuredAcitivities
2. บัพกิจกรรมสามารถมีบัพโครงสร้างเป็นเจ้าของได้มากสุดหนึ่งอันเท่านั้น

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 58 แสดงตัวอย่างที่เป็นไปได้ของบัพกิจกรรมโดยประกอบด้วยแอ็คชั่น (รับสินค้า กรอกใบสั่งซื้อ ส่งสินค้า เป็นต้น) บัพวัตถุ (ใบส่งของ) และบัพควบคุม (การตัดสินใจก่อนที่จะมีการกรอกใบสั่งซื้อ การแยกทำงานคู่ขนานหลังจากมีการกรอกใบสั่งซื้อ การรวมการทำงานหลังจากที่มีการส่งสินค้า เป็นต้น)



ภาพที่ 58 ตัวอย่างบัพกิจกรรม

ตัวแบ่งส่วนกิจกรรม (ActivityPartition)

การแบ่งส่วนกิจกรรมคล้ายกลับการจัดกลุ่มกิจกรรมแต่จะต่างกันตรงที่การแบ่งส่วนกิจกรรมจะมีการเพิ่มคุณลักษณะบางอย่างให้กับส่วนแสดงด้วย ซึ่งการแบ่งส่วนนี้จะไม่ส่งผลต่อการเปลี่ยนแปลงการไหลของโทเค็น แต่อย่างไรก็ตามก็ยังคงมีข้อจำกัดเกี่ยวกับการแบ่งส่วนซึ่งจะขึ้นอยู่กับวิธีการนำไปใช้งาน

ลักษณะประจำ

- isDimension : Boolean [1..1] = false
  เป็นการบอกว่าการแบ่งส่วนนี้กับการแบ่งส่วนอื่น ๆ สามารถอยู่บนมิติเดียวกันได้หรือไม่
- isExternal : Boolean [1..1] = false
  เป็นการบอกว่าการแบ่งส่วนนี้สามารถนำไปใช้กับการแบ่งส่วนโครงสร้างได้หรือไม่

ข้อจำกัด

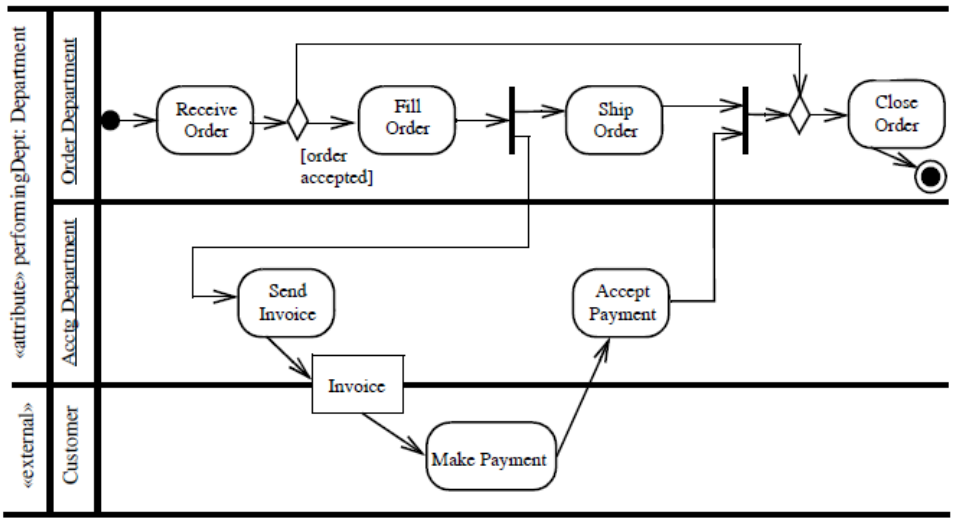
1. การแบ่งส่วนที่มีค่า isDimension = true นั้นจะไม่สามารถนำไปใช้ร่วมกับการแบ่งส่วนอื่น ๆ ได้
2. เมื่อใช้การแบ่งส่วน (Partition) เป็นการแบ่งเป็นส่วน (Part) จะต้องทำการแบ่งส่วนที่ไม่ใช่การแบ่งส่วนภายนอกภายที่อยู่ในมิติและระดับความลึกเดียวกัน โดยให้เห็นเป็นโครงสร้างการจัดกลุ่มที่อยู่ด้วยกันชัดเจน
3. เมื่อใช้การแบ่งส่วน (ที่ไม่ใช่การแบ่งส่วนภายนอก) ให้แสดงถึงการจัดกลุ่ม (Classifier) เมื่อนั้นจะต้องมีการใช้ตัวแบ่งแสดงให้เห็นถึงขอบเขตของกิจกรรมที่

เกี่ยวข้อง และหากมีการแบ่งส่วนย่อยอีกก็ต้องมีการแสดงขอบเขตของกิจกรรมที่เกี่ยวข้องในกลุ่มย่อยให้มีความชัดเจนเช่นเดียวกัน

4. เมื่อการแบ่งส่วนประกอบด้วยการแบ่งส่วนอื่นอีกที เมื่อนั้นให้ถือว่าส่วนที่แบ่งออกมานั้นอยู่ในกลุ่มของการแบ่งส่วนที่อยู่เท่านั้น กล่าวคือให้ยึดคำกำกับของการแบ่งส่วนที่อยู่ภายใต้เท่านั้น

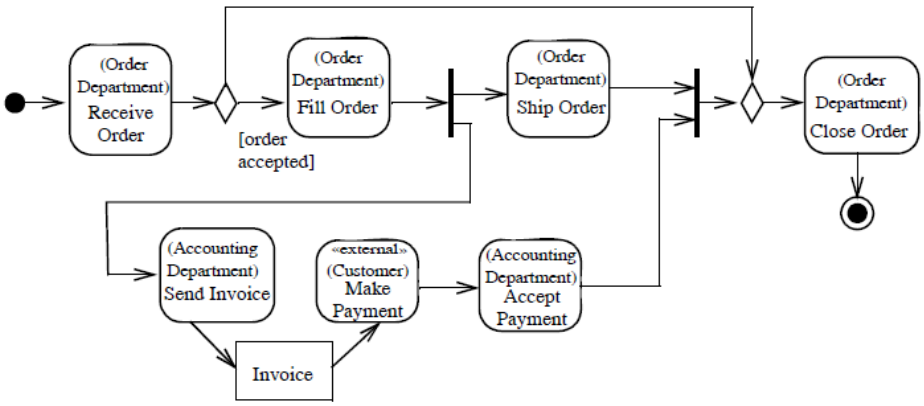
ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 59 แสดงการแบ่งส่วนกิจกรรมด้วยสวิมเลน ซึ่งเป็นการแบ่งความรับผิดชอบต่อกิจกรรม โดยด้านบนจะมีแผนกจัดซื้อ (Order Department) เป็นผู้รับผิดชอบ ส่วนตอนกลางจะมีแผนกบัญชี (Accounting Department) เป็นผู้รับผิดชอบ และสุดท้ายด้านล่างจะมีเป็นกิจกรรมที่เกี่ยวข้องกับลูกค้า (Customer) ที่มีการระบุเป็นปัจจัยภายนอก (External) นอกจากนี้จะเห็นว่าใบส่งของ (Invoice) ไม่ได้อยู่ในการแบ่งส่วนด้วยเพราะว่าใบส่งของเป็นวัตถุจึงไม่จำเป็นที่ต้องระบุในการแบ่งส่วน



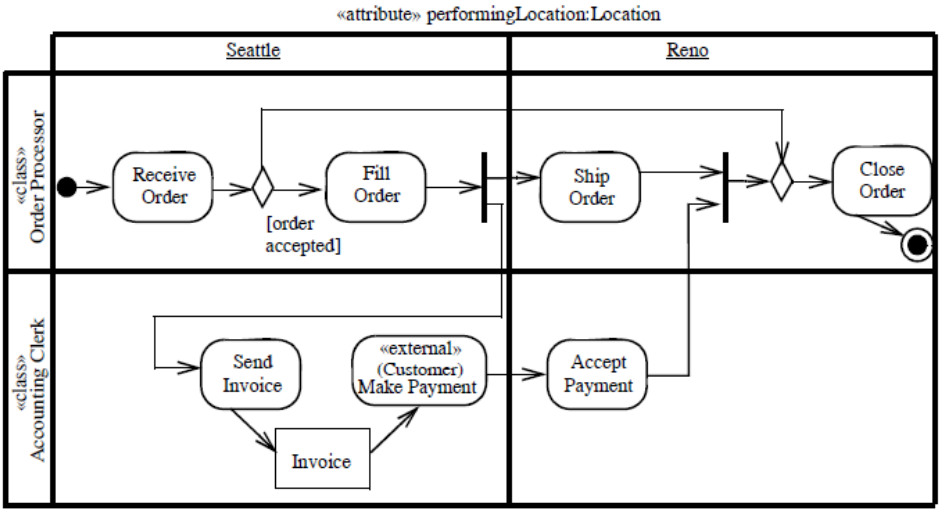
ภาพที่ 59 การแบ่งส่วนโดยใช้เทคนิคสวิมเลน

สำหรับภาพที่ 60 จะมีความหมายเช่นเดียวกันกับภาพที่ 59 แต่จะต่างกันตรงที่ภาพที่ 60 ใช้คำกำกับกิจกรรมแทนการแบ่งส่วนในรูปแบบของ Swimlane



ภาพที่ 60 การแบ่งส่วนโดยใช้คำกำกับ

สำหรับการใช้สวิมเลนหลายมิติดังภาพที่ 61 แสดงให้เห็นถึงการรับใบสั่งซื้อและการกรอกใบสั่งซื้อจะถูกดำเนินการโดยคลาสของตัวประมวลผลการสั่งซื้อ (Order Processor) ที่ตั้งอยู่ใน Seattle ส่วนขั้นตอนของการจ่ายเงินนั้นถึงแม้ว่าจะปรากฏอยู่ในความรับผิดชอบของพนักงานบัญชี (Accounting Clerk) และมีการดำเนินการอยู่ใน Seattle ก็ตาม แต่ก็ไม่ได้ดำเนินการตามที่กิจกรรมระบุแต่อย่างใด เพราะว่ามีการระบุทับด้วยคำกำกับ <<external>> Customer จึงให้ถือว่าการทำงานเป็นไปตามที่ระบุอย่างหลัง กล่าวคือ ให้ลูกค้าเป็นผู้รับผิดชอบ



ภาพที่ 61 การแบ่งส่วนโดยใช้สวิมเลนหลายมิติ

<u>สายงานควบคุม</u> <u>(ControlFlow)</u>

สายงานควบคุมทำหน้าที่เริ่มต้นการทำงานของกิจกรรมหลังจากที่กิจกรรมก่อนหน้าสิ้นสุดลง

ลักษณะประจำ

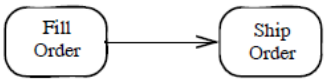
ไม่มีลักษณะประจำ

ข้อจำกัด

1. สายงานควบคุมไม่สามารถเชื่อมกับมีบัพวัตถุ (Object Node) ได้ ยกเว้นบัพวัตถุจะมีการกำหนดให้เป็นประเภทควบคุม

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 62 เป็นการเชื่อมแอ็คชั่นสองอันด้วยสายงานควบคุม ซึ่งแสดงให้ถึงขั้นตอนการทำงานอย่างเป็นลำดับ ในที่นี้หมายความว่าเมื่อมีการกรอกใบสั่งซื้อ (Fill Order) เสร็จแล้ว จึงสั่งให้มีการเริ่มจัดส่งใบสั่งซื้อ (Ship Order)



ภาพที่ 62 ตัวอย่างสายงานควบคุม

<u>คลังข้อมูล</u> <u>(DataStore)</u>

คลังข้อมูลใช้สำหรับเก็บโทเค็นทั้งหมดที่เข้ามา และจะส่งข้อมูล (เฉพาะข้อมูลที่ถูกเลือกเท่านั้น) ไปยังแอ็คชั่นถัดไป

ลักษณะประจำ

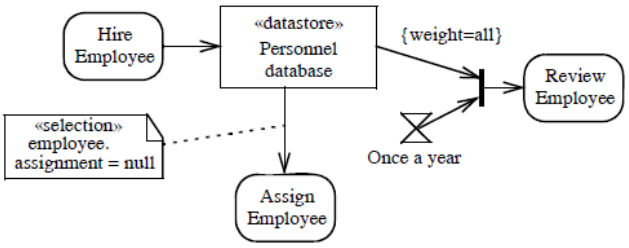
ไม่มีลักษณะประจำ

ข้อจำกัด

ไม่มีข้อจำกัด

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 63 เป็นตัวอย่างการใช้งานคลังข้อมูล (Data Store) โดยแสดงให้เห็นว่า เมื่อมีการจ้างพนักงานเข้ามา ข้อมูลจะถูกเก็บลงอยู่ในฐานข้อมูลบุคคล (Personnel Database) หลังจากนั้นจะทำการเลือกข้อมูลพนักงานที่ยังไม่มีการมอบหมายงานให้ (<<selection>> employee.assignment = null) ส่งไปยังกิจกรรมมอบหมายงานให้กับพนักงาน (Assign Employee) ส่วนข้อมูลพนักงานทั้งหมดจะถูกส่งไปยังขั้นตอนพิจารณาตรวจสอบพนักงาน (Review Employee) ซึ่งจะต้องทำปีละครั้ง



ภาพที่ 63 ตัวอย่างการใช้คลังข้อมูล

<u>บัพตัดสินใจ</u> (DecisionNode)

บัพตัดสินใจใช้สำหรับเลือกเส้นทางการไหลของข้อมูล

ลักษณะประจำ

ไม่มีลักษณะประจำ

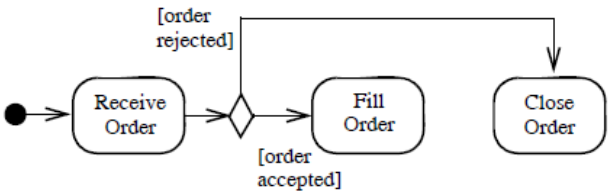
ข้อจำกัด

1. บัพตัดสินใจสามารถมีได้เส้นเชื่อมเข้าได้เพียงหนึ่งหรือสองเส้นเท่านั้น และจะต้องมีเส้นเชื่อมออกอย่างน้อยหนึ่งเส้นเสมอ
2. เส้นเชื่อมเข้าหรือออกจากบัพตัดสินใจ นอกจากเหนือจากสายงานนำเข้าของการตัดสินใจ (ถ้ามี) จะต้องเป็นสายงานควบคุมทั้งหมดหรือสายงานวัตถุทั้งหมดอย่างใดอย่างหนึ่ง
3. สายงานนำเข้าของการตัดสินใจจะต้องเป็นเส้นเชื่อมเข้าของบัพตัดสินใจเท่านั้น
4. การตัดสินใจที่ต้องใช้ข้อมูลนำเข้าจะต้องไม่มีตัวแปรนำออก ตัวแปรไหลผ่าน หรือข้อมูลส่งกลับแต่อย่างใด

5. ถ้าบัพตัดสินใจไม่มีสายงานนำเข้าของการตัดสินใจ (Decision Input Flow) และสายงานควบคุมแล้ว ให้ถือว่าการตัดสินใจนี้ไม่มีตัวแปรนำเข้า
6. ถ้าบัพตัดสินใจไม่มีสายงานนำเข้าของการตัดสินใจและสายงานวัตถุแล้ว ให้ถือว่าการตัดสินใจมีเพียงหนึ่งตัวแปรเข้าคือโทเค็นวัตถุที่มาจากเส้นเชื่อมเข้า (Incoming Edge)
7. ถ้าบัพตัดสินใจมีทั้งสายงานนำเข้าของการตัดสินใจและสายงานควบคุมแล้ว ให้ถือว่าการตัดสินใจมีเพียงหนึ่งตัวแปรเข้าคือโทเค็นวัตถุที่มาจากสายงานนำเข้าของการตัดสินใจ (Decision Input Flow)
8. ถ้าบัพตัดสินใจมีทั้งสายงานนำเข้าของการตัดสินใจและสายงานวัตถุแล้ว ให้ถือว่าการตัดสินใจมีสองตัวแปร โดยตัวแปรแรกคือโทเค็นวัตถุที่ไม่ได้มาจากสายงานนำเข้าของการตัดสินใจ ส่วนอีกตัวแปรคือโทเค็นวัตถุที่มาจากสายงานนำเข้าของการตัดสินใจ (Decision Input Flow)

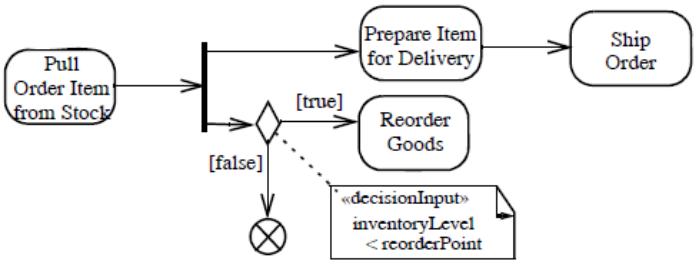
ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 64 แสดงการตัดสินใจหลังจากที่การรับการสั่งซื้อ โดยถ้ามีการยอมรับการสั่งซื้อก็จะทำงานในส่วนของการกรอกข้อมูลใบสั่งซื้อ แต่ถ้าปฏิเสธการสั่งซื้อก็จะทำการปิดการสั่งซื้อนั้นโดยทันที กล่าวคือทิศทางการไหลจะขึ้นอยู่การการยอมรับหรือปฏิเสธการสั่งซื้อ



ภาพที่ 64 ตัวอย่างการใช้บัพตัดสินใจ

จากตัวอย่างภาพที่ 65 แสดงการตัดสินใจแบบต้องอาศัยข้อมูลนำเข้าประกอบการตัดสินใจ โดยหลังจากที่มีการดึงไอเท็มที่มีการสั่งซื้อออกจากคลังสินค้าแล้ว จะแบ่งการทำงานออกเป็นสองสายคือ สายแรกจะทำหน้าที่จัดเตรียมไอเท็มนั้นเพื่อจัดส่งสินค้า อีกสายจะเป็นการตรวจสอบสินค้าคงคลังซึ่งจะเป็นการนับจำนวนสินค้าว่าเหลือน้อยกว่าระดับที่ตั้งไว้หรือไม่ ถ้าเหลือน้อยกว่าที่ตั้งไว้ก็ให้ทำการสั่งสินค้ามาเก็บไว้ในคลังสินค้าเพิ่ม



ภาพที่ 65 ตัวอย่างการใช้บัพตัดสินใจที่มีการใช้ข้อมูลนำเข้าสำหรับการตัดสินใจ

ชุดคำสั่งจัดการสิ่งผิดปกติ (ExceptionHandler)

ลักษณะประจำ

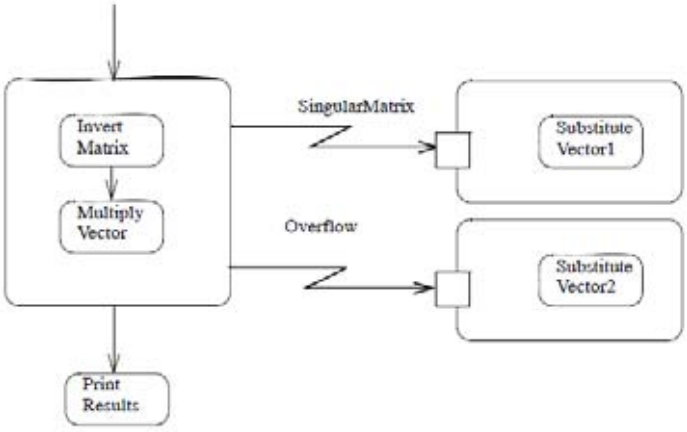
ไม่มีลักษณะประจำ

ข้อจำกัด

1. ชุดคำสั่งจัดการสิ่งผิดปกติ (Exception Handler) และข้อมูลนำเข้าที่เกี่ยวข้องจะต้องไม่เป็นต้นทางหรือปลายทางของเส้นเชื่อมใด ๆ
2. ถ้าเส้นเชื่อมมีจุดกำเนิดอยู่ในชุดคำสั่งจัดการสิ่งผิดปกติ จะต้องมีปลายทางอยู่ในขอบเขตของชุดคำสั่งจัดการสิ่งผิดปกตินั้น
3. ถ้าชุดคำสั่งจัดการสิ่งผิดปกติเป็นจุดนำออกของ StructuredActivityNode ตัวบอดีของชุดคำสั่งจัดการสิ่งผิดปกติจะต้องอยู่ที่จุดนำออกของ StructuredActivityNode นั้น ๆ ตามชนิดและจำนวนของบัพป้องกันเหล่านั้น
4. ตัวบอดีจะมีเพียงหนึ่งข้อมูลนำเข้าเท่านั้น โดยข้อมูลนำเข้านั้นจะเป็นอันเดียวกันกับข้อมูลนำเข้าของสิ่งผิดปกติ (Exception)

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 66 แสดงการทำงานของการคำนวนทางเมทิกซ์ ซึ่งกิจกรรมแรกที่ทำงานคือการสร้างเมทริกซ์ผกผัน (Invert Matrix) แล้วจึงทำการคูณเมทริกซ์ด้วยเวกเตอร์ ขั้นตอนสุดท้ายจึงแสดงผลลัพธ์ออกมา ถ้าหากมีสิ่งผิดปกติเกิดขึ้นขณะทำการผกผันหรือการคูณ กล่าวคือมีการตรวจพบว่าเป็น SingularMatrix ตัวชุดคำสั่งจัดการสิ่งผิดปกติจะทำขั้นตอน SubstituteVector1 แต่ถ้าตรวจพบว่าเกิด Overflow จะทำขั้นตอน SubstituteVector2 ซึ่งผลที่ได้จากการจัดการสิ่งผิดปกติไม่ว่าอย่างใดก็ตามจะถูกนำไปแทนบัพป้องกัน (บัพที่เป็นตัวดักสิ่งผิดปกติที่จะเกิดขึ้น) หลังจากนั้นก็จะทำงานในขั้นตอนถัดไปตามปกติ (ในที่นี้คือแสดงผลลัพธ์)



ภาพที่ 66 ตัวอย่างชุดคำสั่งจัดการสิ่งผิดปกติ

ส่วนขยาย (ExpansionRegion)

ส่วนขยายคือบริเวณของกิจกรรมเชิงโครงสร้างที่สามารถทำงานได้หลายครั้ง ทั้งนี้จะขึ้นอยู่กับกลุ่มจำนวนของข้อมูลนำเข้า

ลักษณะประจำ

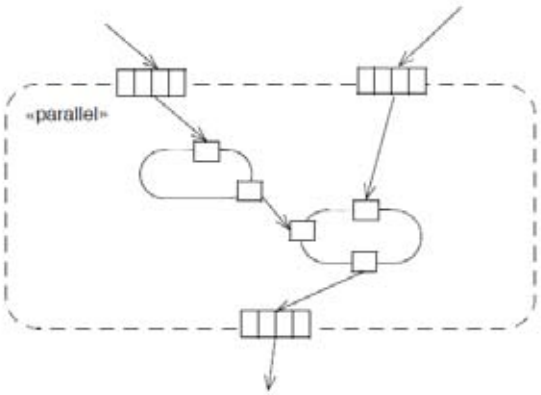
- mode : ExpansionKind = iterative
  แสดงวิธีการดำเนินการของส่วนขยาย โดยค่าที่เป็นไปได้มีดังนี้
  - parallel เป็นการบอกว่าการทำงานของกิจกรรมนี้ ทำงานแบบขนานกันไป
  - iterative เป็นการบอกว่าการทำงานของกิจกรรมนี้ ทำงานแบบวนซ้ำอย่างเป็นลำดับ
  - stream เป็นการบอกว่าการทำงานของกิจกรรมนี้ จะดำเนินการกับข้อมูลนำเข้าทั้งหมดพร้อมกัน (จะมีการสั่งหรือกระตุ้นการดำเนินงานเพียงครั้งเดียว)

ข้อจำกัด

1. ส่วนขยายจะต้องมีอย่างน้อยหนึ่งอาร์กิวเมนต์ โดยอาจจะมีหรือไม่มีผลลัพธ์กลับมาก็ได้ (สามารถมีได้มากกว่าหนึ่งผลลัพธ์)

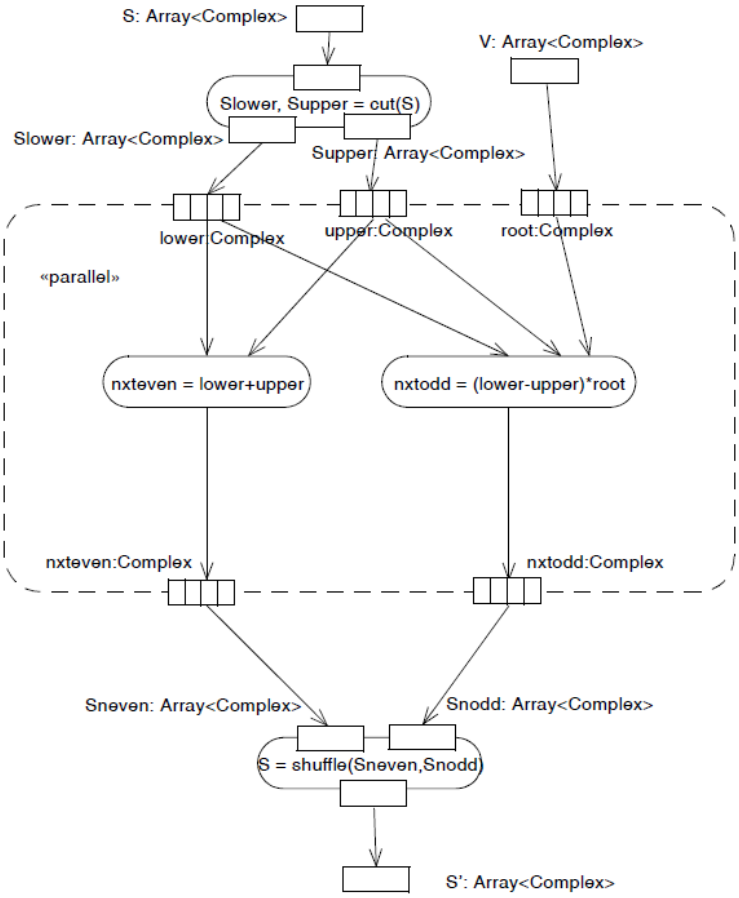
ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 67แสดงส่วนขยายที่มีข้อมูลนำเข้าสองกลุ่มและข้อมูลนำออกหนึ่งกลุ่มโดยมีการกำกับการทำงานให้เป็นแบบขนาน ดังนั้นส่วนขยายจึงจะยังไม่ทำงานจนกระทั่งจะได้รับข้อมูลครบทั้งสองตัวกลุ่มก่อน ทั้งนี้จำนวนข้อมูลทั้งสองกลุ่มจะต้องมีจำนวนเท่ากันด้วย เมื่อได้ข้อมูลครบเท่ากันแล้วแอ็คชั่นจะทำหน้าที่ดึงข้อมูลออกมาจากข้อมูลนำเข้าของแต่ละกลุ่มที่เกี่ยวข้องกับแอ็คชั่นนั้น ๆ เพื่อทำการดำเนินงานต่อไปจนกระทั่งได้ข้อมูลนำออกเพื่อนำไปเก็บไว้ที่กลุ่มข้อมูลนำออก โดยจะทำเช่นนี้ซ้ำไปเรื่อย ๆ ตามจำนวนข้อมูลนำเข้า สุดท้ายจะได้กลุ่มข้อมูลนำออกหนึ่งกลุ่มที่มีสมาชิกเท่ากับจำนวนสมาชิกของข้อมูลนำเข้า



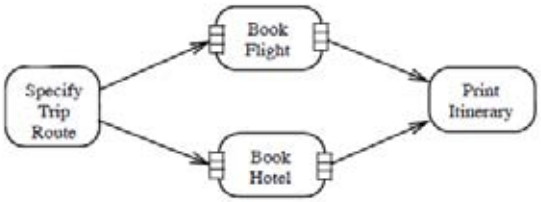
ภาพที่ 67 ส่วนขยายที่ประกอบด้วยข้อมูลนำเข้าสองกลุ่มและข้อมูลนำออกหนึ่งกลุ่ม

จากตัวอย่างส่วนขยายภาพที่ 68แสดงทำงานของฟาสต์ฟูเรียร์ทรานส์ฟอร์ม (Fast Fourier Transform - FFT) บริเวณนอกส่วนขยายจะมีการประกาศจำนวนเชิงซ้อน S, Slower Supper และ V ขึ้นมาในรูปแบบของแถวลำดับ โดย Slower และ Supper นั้นจะเกิดจากการตัดแบ่งจาก S สำหรับภายในส่วนขยายจะเกี่ยวข้องกับการคำนวนทางคณิตศาสตร์บนข้อมูลนำเข้าทั้งสามตัวและจะให้ข้อมูลนำออกสองตัว โดยข้อมูลนำออกทั้งสองตัวนี้จะถูกนำไปใช้ในการสลับที่เพื่อสร้างเป็นจำนวนเชิงซ้อน S’ ในลำดับต่อไป



ภาพที่ 68 ส่วนขยายแสดงการทำงานของฟาสต์ฟูเรียร์ทรานส์ฟอร์ม

จากภาพที่ 69เป็นการใช้ส่วนขยายในลักษณะย่อด้วยการลดรูปให้เหลือเป็นเพียงแอ็คชั่น ซึ่งจากรูปเป็นการแสดงการทำงานของการวางแผนการเดินทาง โดยจะต้องเริ่มต้นด้วยการกำหนดเส้นทางการเดินทาง ดังนั้นจึงจะต้องมีการจองตั๋วเครื่องบินและโรงแรมตลอดการเดินทางซึ่งเป็นลักษณะของกลุ่มข้อมูล จะเห็นว่าการจองตั๋วเครื่องบินและโรงแรมสามารถทำงานเป็นอิสระต่อกัน จึงมีการกำหนดสายงานให้เป็นแบบขนาน



ภาพที่ 69 ตัวอย่างการใช้ส่วนขยาย

จากภาพที่ 70เป็นการจองตั๋วเครื่องบินในหลาย ๆ เที่ยวบิน โดยแต่ละเที่ยวบินสามารถจองแยกจากกันได้ ซึ่งเป็นผลให้ส่วนแสดงในส่วนของการจองเที่ยวบินนั้นจะเกิดซ้ำกันหลาย ๆ ครั้ง

ซึ่งการวนซ้ำในแต่ละครั้งจะหมายถึงการจองเที่ยวบินหนึ่งเที่ยวบิน

ภาพที่ 70 ตัวอย่างการใช้ส่วนขยายแบบย่อ

<u>บัพหยุดสายงาน</u> (FlowFinal)

บัพหยุดสายงานใช้สำหรับหยุดสายงานในกิจกรรม

ลักษณะประจำ

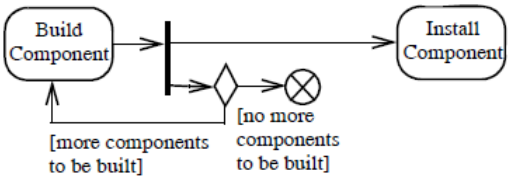
ไม่มีลักษณะประจำ

ข้อจำกัด

1. บัพหยุดสายงานไม่มีเส้นเชื่อมออก

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 71กำหนดให้มีส่วนประกอบที่จะต้องทำการติดตั้งหลายส่วน โดยจะต้องทำการสร้างส่วนประกอบขึ้นมาตามจำนวนที่ต้องการจะสร้าง เมื่อทำการสร้างเสร็จแล้วจึงให้ทำการติดตั้งส่วนประกอบดังกล่าว การสร้างและการติดตั้งส่วนประกอบจะทำการวนซ้ำไปเรื่อย ๆ จนกระทั่งไม่มีส่วนประกอบที่ต้องการจะสร้างเพิ่ม เมื่อไม่มีส่วนประกอบที่ต้องการจะสร้างเพิ่มแล้วให้ทำการสิ้นสุดสายงานดังกล่าว ทั้งนี้ทั้งนั้นการติดตั้งส่วนประกอบและกิจกรรมอื่น ๆ ยังสามารถที่จะดำเนินการได้ต่อไป



ภาพที่ 71 ตัวอย่างบัพสิ้นสุดสายงานที่ไม่มีบัพผสาน

บัพแยก (ForkNode)

บัพแยกทำหน้าที่ในการแยกสายงานออกจากกันให้สามารถทำงานพร้อม ๆ กันได้

ลักษณะประจำ

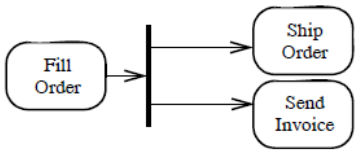
ไม่มีลักษณะประจำ

ข้อจำกัด

1. บัพแยกมีได้เพียงหนึ่งเส้นเชื่อมเข้าเท่านั้น
2. เส้นเชื่อมเข้าและออกจากบัพแยพ จะต้องเป็นสายงานวัตถุทั้งหมดหรือสายงานควบคุมทั้งหมดอย่างใดอย่างหนึ่ง

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 72 เมื่อมีการกรอกใบสั่งซื้อเสร็จให้ทำการส่งใบสั่งซื้อและใบส่งของ (ทั้งสองแอ็คชั่นจะเกิดการทำงานได้พร้อมกันหลังจากกรอกใบสั่งซื้อเสร็จ)



ภาพที่ 72 ตัวอย่างการใช้บัพแยก

บัพเริ่มต้น (InitialNode)

บัพเริ่มต้นใช้แสดงจุดเริ่มต้นของสายงานในกิจกรรม โดยแต่ละกิจกรรมอาจมีมากกว่าหนึ่งบัพเริ่มต้นก็ได้

ลักษณะประจำ

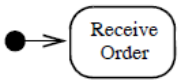
ไม่มีลักษณะประจำ

ข้อจำกัด

1. บัพเริ่มต้นไม่มีเส้นเชื่อมเข้า
2. เส้นเชื่อมที่ต่อกับบัพเริ่มต้น มีบัพเริ่มต้นเป็นต้นทาง จะต้องเป็นเส้นเชื่อมควบคุมเท่านั้น

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 73 บัพเริ่มต้นจะกระตุ้นส่วนเพื่อเริ่มต้นส่วนแสดงการรับการสั่งซื้อทันที หลังจากที่กิจกรรมเริ่มทำงาน



ภาพที่ 73 ตัวอย่างการใช้งานบัพเริ่มต้น

<u>ส่วนกิจกรรมที่สามารถขัดจังหวะได้</u> <u>(InterruptibleActivityRegion)</u>

ส่วนกิจกรรมที่สามารถขัดจังหวะได้ คือ กลุ่มของกิจกรรมที่รองรับการทำลายหรือหยุดโทเค็นในสายงาน

ลักษณะประจำ

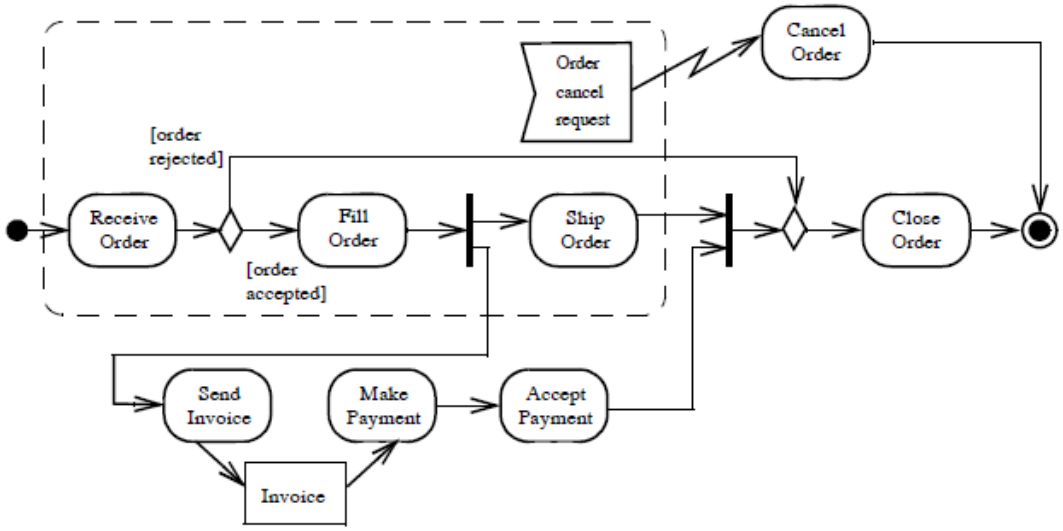
ไม่มีลักษณะประจำ

ข้อจำกัด

1. เส้นเชื่อมของส่วนกิจกรรมที่สามารถขัดจังหวะได้ (InterruptibleActivityRegion) จะต้องมีบัพต้นทางอยู่ในส่วนกิจกรรมหรือบริเวณที่รองรับการขัดจังหวะ และต้องมีบัพปลายทางอยู่ด้านนอกส่วนกิจกรรมหรือบริเวณของบัพต้นทาง

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 74 ถ้ามีการขอยกเลิกการสั่งซื้อขณะที่มีการรับใบสั่งซื้อ กรอกใบสั่งซื้อ หรือจัดส่งใบสั่งซื้อ ส่วนแสดงยกเลิกการสั่งซื้อจะถูกเรียกให้ทำงานทันที เพื่อทำการยกเลิกการสั่งซื้อดังกล่าว



ภาพที่ 74 ตัวอย่างการใช้งานส่วนกิจกรรมที่สามารถขัดจังหวะได้

บัพรวม (JoinNode)

บัพรวมทำหน้าที่ประสานสายงานหลาย ๆ สายงานเข้าด้วยกัน

ลักษณะประจำ

CompleteActivities

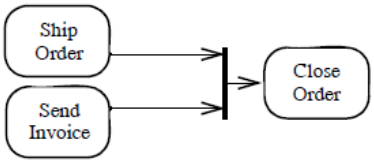
- isCombineDuplicate : Boolean [1..1] = true
  เป็นการบ่งบอกว่าโทเค็นที่เป็นวัตถุประเภทเดียวกัน ให้ทำการรวมกันเหลือเพียงวัตถุเดียว

ข้อจำกัด

1. บัพรวมมีเส้นเชื่อมออกได้เพียงหนึ่งเส้นเท่านั้น
2. ถ้าบัพรวมมีเส้นเชื่อมเข้าเป็นสายงานวัตถุจะต้องมีเส้นเชื่อมออกเป็นสายงานวัตถุเช่นเดียวกันถ้าเส้นเชื่อมเข้าเป็นสายงานควบคุมจะต้องมีเส้นเชื่อมออกเป็นสายงานควบคุม

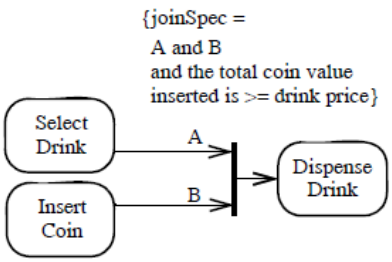
ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 75 จะมีการทำการประสานสายงานโดยใช้บัพรวม หลังจากที่มีการส่งของตามใบสั่งซื้อและใบส่งของ ซึ่งบัพรวมจะทำหน้าที่รอการทำงานจนกระทั่งส่วนแสดงทั้งสองอย่างที่กล่าวมาข้างต้นทำงานเสร็จสิ้นทั้งสองส่วนแสดง จึงทำงานในส่วนแสดงปิดการสั่งซื้อ



ภาพที่ 75 ตัวอย่างบัพรวม

จากภาพที่ 76 เป็นการใช้ข้อกำหนดการรวมเพื่อให้มั่นใจว่าการจ่ายเครื่องดื่มนี้เป็นไปตามเงื่อนไข โดยจะสามารถจ่ายเครื่องดื่มได้ก็ต่อเมื่อมีการเลือกเครื่องดื่ม และมีการหยอดเหรียญ และเหรียญที่หยอดจะต้องมีมูลค่าเท่ากับหรือมากกว่ามูลค่าของเครื่องดื่ม ซึ่งข้อกำหนดการรวมนี้จะเป็นตัวกำหนดว่าโทเค็นจะสามารถทำงานในสายงานนี้ต่อได้หรือไม่



ภาพที่ 76 ตัวอย่างบัพรวมแบบมีการใช้ข้อกำหนดการรวม

<u>บัพผสาน (MergeNode)</u>

บัพผสานเป็นการนำสายงานทางเลือกต่าง ๆ มารวมกันเพื่อให้สายงานที่ถูกเลือก (หนึ่งในสายงานทางเลือกต่าง ๆ) สามารถดำเนินงานต่อไปได้

ลักษณะประจำ

ไม่มีลักษณะประจำ

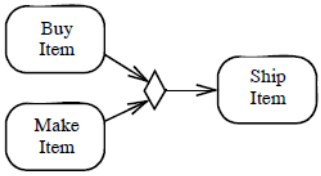
ข้อจำกัด

1. บัพผสานจะมีเส้นเชื่อมออกได้เพียงหนึ่งเส้นเท่านั้น

2. เส้นเชื่อมเข้าและเส้นออกของบัพผสาน จะต้องเป็นสายงานวัตถุทั้งหมดหรือสายงานควบคุมทั้งหมดอย่างใดอย่างหนึ่ง

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 77 มีความเป็นได้ที่จะมีการทำงานของสายงานใดสายงานหนึ่งหรือทั้งสองสายงานเลยก็ได้ โดยหากมีการทำงานของสายงานใดสายงานหนึ่งเสร็จสิ้นจะมีการเรียกแอ็คชั่นส่งสินค้าให้ทำงาน ดังนั้นหากมีการซื้อสินค้าและจัดทำสินค้า (มีการทำงานทั้งสองสายงาน) ขั้นตอนการส่งสินค้าก็จะถูกเรียกสองครั้ง



ภาพที่ 77 ตัวอย่างการใช้งานบัพผสาน

<u>สายงานวัตถุ</u> (ObjectFlow)

สายงานวัตถุ คือ เส้นเชื่อมกิจกรรมที่มีการส่งผ่านวัตถุหรือข้อมูล

ลักษณะประจำ

CompleteActivities

- isMulticast : Boolean [1..1] = false
  เป็นการบอกว่าวัตถุในสายงานถูกส่งด้วยวิธีการกระจาย (Multicasting)
- isMultireceive : Boolean [1..1] = false
  เป็นการบอกว่าวัตถุในสายงานถูกรวบรวมได้มาจากการกระจาย

ข้อจำกัด

BasicActivities

1. สายงานวัตถุไม่สามารถมีต้นทางและปลายทางเป็นส่วนแสดงทั้งสองทางได้
2. บัพวัตถุที่เชื่อมต่อกับสายงานวัตถุ บางกรณีอาจมีบัพควบคุมเข้ามาเกี่ยวข้อง ต้องเป็นชนิดที่สอดคล้องกัน กล่าวคือวัตถุปลายทางจะต้องเป็นวัตถุที่เป็นชนิดเดียวกันหรือสืบทอดมาจากวัตถุต้นทาง

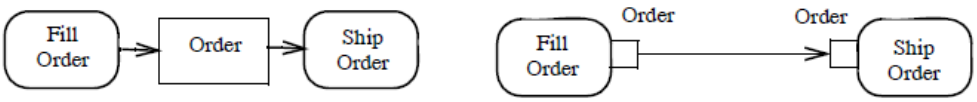
3. บัพวัตถุที่เชื่อมต่อกับสายงานวัตถุ บางกรณีอาจมีบัพควบคุมเข้ามาเกี่ยวข้อง จะต้องมีขอบเขตบน (Upper Bound) เดียวกัน

CompleteActivities

1. เส้นเชื่อมที่มีน้ำหนัก (Weight) กำหนดจะต้องไม่เชื่อมกับบัพวัตถุที่มีขอบเขตบนน้อยกว่าน้ำหนักที่ระบุไว้
2. กิจกรรมที่มีการเปลี่ยนแปลงตัวแปรนำเข้าหนึ่งตัวแปร ไปสู่ตัวแปรนำออกหนึ่งตัวแปร ตัวแปรนำเข้าจะต้องสอดคล้องกับโทเค็นวัตถุต้นทาง (สุดของต้นทาง) และตัวแปรนำออกจะต้องสอดคล้องกับโทเค็นวัตถุที่จะเกิดขึ้นในส่วนแสดงต่อ ๆ ไป
3. สายงานวัตถุสามารถใช้กิจกรรมที่เกี่ยวข้องกับการเลือกได้ หากมีบัพวัตถุเป็นต้นทาง
4. กิจกรรมที่เกี่ยวข้องกับการเลือกที่มีตัวแปรนำเข้าหนึ่งตัวแปรและตัวแปรนำออกหนึ่งตัวแปร ตัวแปรนำเข้าและตัวแปรนำออกนั้นจะต้องสอดคล้องกันกับวัตถุต้นทาง
5. ลักษณะประจำ isMulticast และ isMultireceive สามารถเป็นจริงได้เพียงลักษณะใดลักษณะหนึ่งเท่านั้น

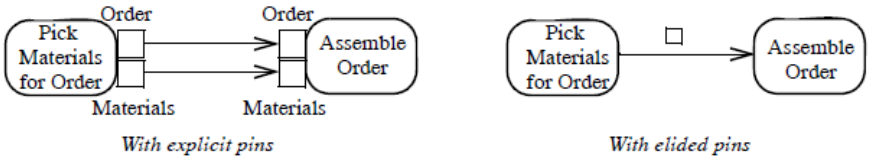
ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 78 แผนภาพทางซ้ายแสดงการส่งผ่านวัตถุจากแอ็คชั่นกรอกใบสั่งซื้อไปยังส่งใบสั่งซื้อโดยมีใบสั่งซื้อเป็นวัตถุเชื่อมสองแอ็คชั่นเข้าด้วยกัน ส่วนแผนภาพด้านขวามีความหมายเช่นเดียวกันกับแผนภาพด้านซ้าย แต่จะแตกต่างกันตรงที่แผนภาพด้านขวาใช้สัญลักษณ์หมุดวัตถุแทนการใช้บัพวัตถุ



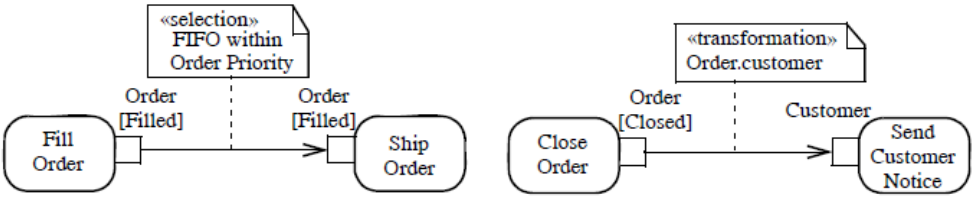
ภาพที่ 78 ตัวอย่างสายงานวัตถุ

จากภาพที่ 79 แสดงการทำงานของสายงานวัตถุ โดยส่วนแสดงการเลือกวัตถุดิบตามใบสั่งซื้อจะทำการส่งวัตถุดิบและใบสั่งซื้อให้กับการประกอบสินค้า ในบางกรณีเป็นไปได้ที่ว่าแอ็คชั่นบางอย่างเป็นที่เข้าใจกันดีอยู่แล้วว่าจะต้องใช้หรือส่งอะไรระหว่างกันบ้าง ทำให้สามารถลดรูป (ละเว้นการแสดงวัตถุ) เพื่อให้แผนภาพง่ายต่อการอ่านมากยิ่งขึ้น



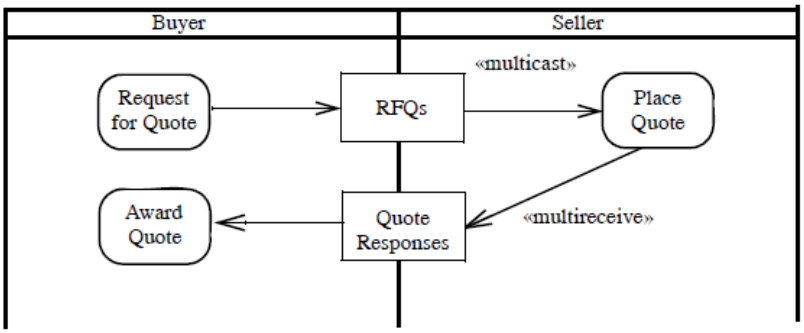
ภาพที่ 79 การละเว้นการแสดงวัตถุในสายงานวัตถุ

จากภาพที่ 80 แผนภาพด้านซ้ายแสดงการส่งใบสั่งซื้อ ซึ่งจะทำการส่งใบสั่งซื้อตามลำดับความสำคัญของใบสั่งซื้อ หากใบสั่งซื้อมีความสำคัญเท่ากันให้ยึดตามหลัก FIFO แทน สำหรับแผนภาพด้านขวาแสดงการทำงานของการปิดการสั่งซื้อ โดยหลังจากที่ปิดการสั่งซื้อแล้วจะต้องทำการแจ้งไปยังลูกค้าให้ทราบด้วย ซึ่งการแจ้งดังกล่าวจำเป็นต้องอาศัยข้อมูลเกี่ยวกับลูกค้าด้วย ข้อมูลลูกค้าสามารถหาได้จากใบสั่งซื้อ



ภาพที่ 80 ตัวอย่างการใช้การเลือกและการเปลี่ยนแปลงวัตถุบนสายงานวัตถุ

จากตัวอย่างภาพที่ 81แสดงกิจกรรมของการขอใบเสนอราคา โดยการขอใบเสนอราคาจะทำการขอไปยังแต่ละผู้ขาย (เนื่องจากมีการส่งไปยังหลายผู้ขายจึงจำเป็นต้องมีการระบุ <<multicast>>) หลังจากนั้นผู้ขายบางรายหรือทั้งหมด (เนื่องจากมีการรับมาจากหลายผู้ขายจึงจำเป็นต้องระบุ <<multireceive>>) จะทำการส่งใบเสนอราคากลับมายังผู้ซื้อ เพื่อให้ผู้ซื้อได้ทำการประกวดราคาในลำดับต่อไป



ภาพที่ 81 การกำหนด <<multicast>> และ <<multireceive>> ในสายงานวัตถุ

<u>บัพวัตถุ (ObjectNode)</u>

ลักษณะประจำ

CompleteActivities

- ordering : ObjectNodeOrderingKind [1..1] = FIFO
  เป็นการบอกว่าโทเค็นที่อยู่ในวัตถุมีลำดับอย่างไร เพื่อใช้ในการเลือกเส้นเชื่อมออกจากวัตถุ โดยจะมีค่าอย่างใดอย่างหนึ่งต่อไปนี้
  unordered ไม่มีลำดับการเรียง
  ordered มีลำดับการเรียงตามเงื่อนไข
  LIFO เรียงลำดับตามหลักการ Last-In-First-Out
  FIFO เรียงลำดับตามหลักการ First-In-First-Out

- isControlType : Boolean [1..1] = false
  เป็นการบ่งบอกว่าให้ใช้วัตถุนี้เสมือนเป็นบัพควบคุม

ข้อจำกัด

BasicActivities

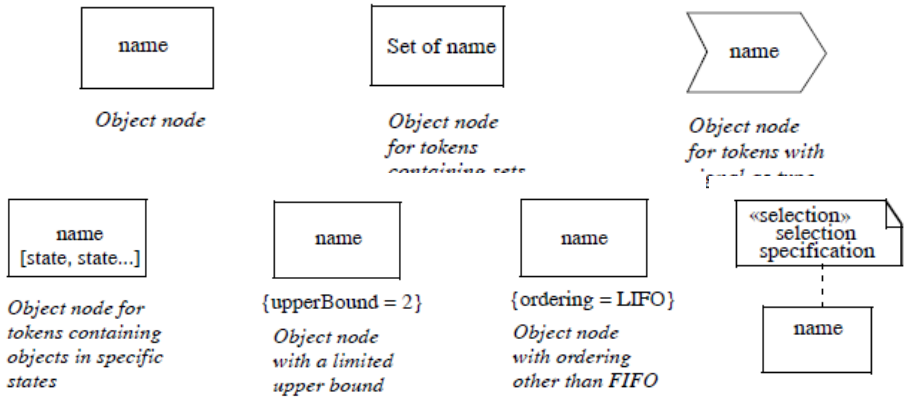
1. เส้นเชื่อมเข้าและเส้นเชื่อมออกทั้งหมดของบัพวัตถุจะต้องเป็นสายงานวัตถุ

CompleteActivities

2. เมื่อมีการเลือกวัตถุเกิดขึ้น ให้เลือกตามลำดับของวัตถุที่ระบุไว้
3. ถ้ามีการระบุกิจกรรมการเลือกจะต้องมีเพียงตัวแปรนำเข้าหนึ่งตัวแปรและตัวแปรนำออกหนึ่งตัวแปร โดยตัวแปรนำเข้าและตัวแปรนำออกจะต้องเป็นโทเค็นวัตถุที่สอดคล้องกับบัพวัตถุ

ตัวอย่างการใช้งานและความหมาย

สัญลักษณ์ที่ใช้แทนบัพวัตถุจะใช้สี่เหลี่ยมผืนผ้า โดยตรงกลางจะระบุเป็นชื่อของวัตถุดังภาพที่ 82 ชื่อของวัตถุอาจระบุเป็นชื่อเซตหรือมีสถานะกำกับใต้ชื่อก็ได้



ภาพที่ 82 ตัวอย่างบัพวัตถุ

<u>เซตตัวแปร (ParameterSet)</u>

เซตตัวแปรจะเป็นตัวทำหน้าที่กำหนดขอบเขตของข้อมูลนำเข้าและข้อมูลนำออก ซึ่งจะเป็นตัวกระตุ้นให้กิจกรรมสามารถเริ่มทำงานได้ เมื่อมีข้อมูลนำเข้าในขอบเขตที่ระบุไว้

ลักษณะประจำ

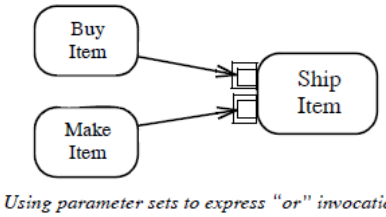
ไม่มีลักษณะประจำ

ข้อจำกัด

1. ตัวแปรที่มีการจัดเป็นกลุ่มตัวแปร ตัวแปรทั้งหมดที่อยู่ในกลุ่มจะต้องเป็นตัวแปรนำเข้าทั้งหมดหรือตัวแปรนำออกทั้งหมด โดยตัวแปรที่อยู่ในกลุ่มเดียวกันจะต้องเป็นตัวแปรประเภทเดียวกัน
2. ถ้ากิจกรรมมีตัวแปรที่อยู่ในเซตตัวแปรแล้ว ตัวแปรอื่น ๆ ที่อยู่นอกเซตตัวแปรจะต้องทำงานแบบทันทีทันใด (Streaming)
3. การจัดกลุ่มตัวแปรจะต้องจัดกลุ่มตัวแปรที่ไม่ซ้ำกัน

ตัวอย่างการใช้งานและความหมาย

จากภาพที่ 83สินค้าจะถูกส่งทันทีหลังจากที่มีการซื้อสินค้าหรือจัดทำสินค้า



ภาพที่ 83 การแบ่งสายงานทางเลือกด้วยเซตตัวแปร

ตัวส่งสัญญาณ (SendSignalAction)

ลักษณะประจำ

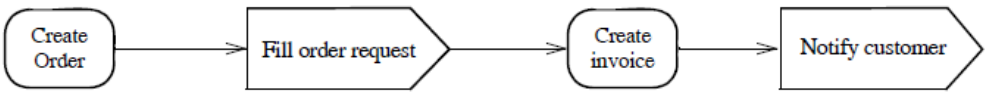
ไม่มีลักษณะประจำ

ข้อจำกัด

ไม่มีข้อจำกัด

ตัวอย่างการใช้งานและความหมาย

จากตัวอย่างภาพที่ 84 แสดการทำงานในการสั่งของแต่ละครั้งซึ่งในแผนภาพนี้จะมีการใช้สัญญาณด้วยกันสองครั้ง ครั้งแรกเป็นส่งสัญญาณแจ้งให้มีการเตรียมของตามใบสั่งซื้อหลังจากที่มีการสร้างใบสั่งซื้อ และอีกครั้งคือการส่งสัญญาณแจ้งให้ลูกค้ารับทราบหลังจากที่ได้จัดทำใบส่งของเสร็จแล้ว



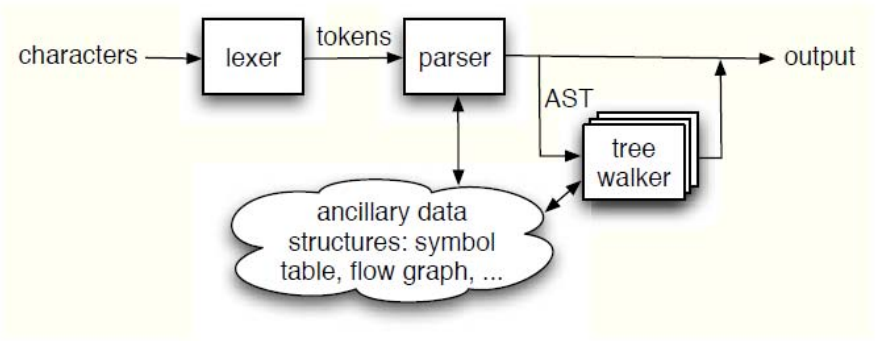
ภาพที่ 84 ตัวอย่างการใช้ตัวส่งสัญญาณ

ภาคผนวก ข.

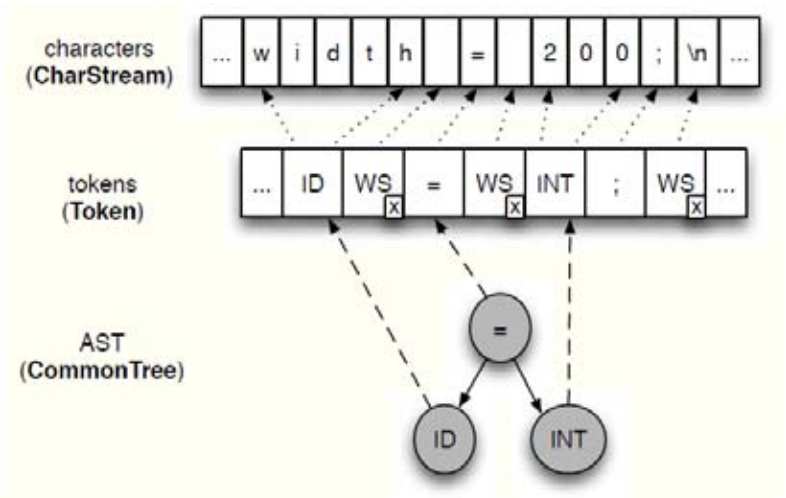
แอนท์เลอร์ (ANother Tool for Language Recognition - ANTLR)

หลักการทำงาน

แอนท์เลอร์ คือ เครื่องมือทางภาษาที่สามารถแปลบทคำสั่งให้ดำเนินงานตามภาษาเป้าหมายที่ต้องการได้ โดยการทำงานของแอนท์เลอร์ (ดังภาพที่ 86) จะเริ่มจากการนำเข้าชุดตัวอักษร แล้วผ่านเข้าสู่ตัวแบ่งคำเพื่อสกัดออกมาเป็นเป็นโทเค็นในรูปแบบของเอเอสที (Abstract Syntax Tree - AST) ดังภาพที่ 86 สำหรับใช้ในการแจงส่วนเพื่อสั่งให้ภาษาเป้าหมายดำเนินงานตามที่กำหนดไว้ได้



ภาพที่ 85 การทำงานของแอนท์เลอร์



ภาพที่ 86 การแปลงข้อมูลชุดตัวอักษรให้อยู่ในรูปแบบเอเอสที

ในบางกรณีหากเราทราบรูปแบบข้อมูลนำเข้าอยู่แล้ว เราสามารถที่จะรายละเอียดข้อมูลเหล่านั้นนำมาแปลงเป็นวากยสัมพันธ์ได้ ดังตัวอย่างสมการอนุพันธ์เชิงพหุนามภาพที่ 87 สามารถแปลงเป็นวากยสัมพันธ์ได้ดังภาพที่ 88 และวากยสัมพันธ์แบบต้นไม้ได้ดังภาพที่ 89

```
d/dx(n) = 0
d/dx(x) = 1
d/dx(nx) = n
d/dx(nx^m) = nmx^m-1
d/dx(foo + bar) = d/dx(foo) + d/dx(bar)
```

ภาพที่ 87 ตัวอย่างสมการอนุพันธ์เชิงพหุนาม

```
grammar Poly;
options {output=AST;}
tokens { MULT; } // imaginary token

poly: term ('+'^ term)*
    ;

term: INT ID  -> ^(MULT["*"] INT ID)
    | INT exp -> ^(MULT["*"] INT exp)
    | exp
    | INT
    | ID
    ;
exp : ID '^'^ INT
    ;

ID  : 'a'..'z'+ ;
INT : '0'..'9'+ ;
WS  : (' '|'\t'|'\r'|'\n')+ {skip();} ;
```

ภาพที่ 88 ตัวอย่างวากยสัมพันธ์สำหรับสมการอนุพันธ์เชิงพหุนาม

```
tree grammar PolyDifferentiator;
options {
    tokenVocab=Poly;
    ASTLabelType=CommonTree;
    output=AST;
//  rewrite=true; // works either in rewrite or normal mode
}

poly:   ^('+' poly poly)
    |   ^(MULT INT ID)      -> INT
    |   ^(MULT c=INT ^('^' ID e=INT))
        {
        String c2 = String.valueOf($c.int*$e.int);
        String e2 = String.valueOf($e.int-1);
        }
                            -> ^(MULT["*"] INT[c2] ^('^' ID INT[e2]))
    |   ^('^' ID e=INT)
        {
        String c2 = String.valueOf($e.int);
        String e2 = String.valueOf($e.int-1);
        }
                            -> ^(MULT["*"] INT[c2] ^('^' ID INT[e2]))
    |   INT                 -> INT["0"]
    |   ID                  -> INT["1"]
    ;
```

ภาพที่ 89 ตัวอย่างวากยสัมพันธ์แบบต้นไม้สำหรับสมการอนุพันธ์เชิงพหุนาม

ประวัติผู้เขียนวิทยานิพนธ์

นายเจริญศักดิ์ นาคงาม เกิดเมื่อวันที่ 11 ธันวาคม พ.ศ. 2526 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยมหิดล ในปีการศึกษา 2550 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2551