การพัฒนาระบบการจัดการกระแสงานจากแผนภาพกิจกรรมของยูเอ็มแอล

นายธนวัฒน์ มหาไตรภพ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550

DEVELOPMENT OF A WORKFLOW MANAGEMENT SYSTEM FROM UML ACTIVITY DIAGRAM

Mr. Thanawat Mahatribhop

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2007

หัวข้อวิทยานิพนธ์	การพัฒนาระบบการจัดการกระแสงานจากแผนภาพกิจกรรมของยูเอ็มแอล
โดย	นายธนวัฒน์ มหาไตรภพ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

ธนวัฒน์ มหาไตรภพ : การพัฒนาระบบการจัดการกระแสงานจากแผนภาพกิจกรรมของยูเอ็มแอล. (DEVELOPMENT OF A WORKFLOW MANAGEMENT SYSTEM FROM UML ACTIVITY DIAGRAM) อ. ที่ปรึกษา : ผศ.ดร.วิวัฒน์ วัฒนาวุฒิ, 101 หน้า.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อพัฒนาระบบการจัดการกระแสงานที่รับข้อกำหนดของกระแสงานในรูปแบบแผนภาพกิจกรรมของยูเอ็มแอล โดยที่แผนภาพกิจกรรมนั้นต้องถูกจัดเก็บมาในรูปแบบของมาตรฐานเอ็กซ์เอ็มไอ ระบบนี้จะควบคุมการทำงานให้เป็นไปตามลำดับและเงื่อนไขที่แสดงมาในแผนภาพกิจกรรม กล่าวคือระบบจะแสดงรายการงานของผู้ใช้แต่ละคนในแต่ละเวลาตามบทบาทของผู้ใช้ รวมถึงแสดงสถานะของงาน และจัดเก็บเอกสารที่เป็นอินพุตหรือเอาท์พุตของงานให้

ระบบการจัดการกระแสงานที่พัฒนาขึ้นมีลักษณะเป็นเว็บแอพพลิเคชัน ทำให้สามารถรองรับการทำงานของผู้ใช้หลายคนและมากกว่าหนึ่งกระแสงานพร้อมกัน รวมถึงรองรับการทำกลับการทำงานและการแก้ไขข้อกำหนดของกระแสงานที่มีอินสแตนซ์ดำเนินการอยู่ ทั้งนี้ระบบยังทำการบันทึกข้อมูลการทำงานลงในล็อกไฟล์เพื่อการดูรายละเอียดการทำงานในภายหลัง

การทดสอบระบบทำโดยการทดลองใช้ทั้งกระแสงานจริงและกรณีทดสอบ ผลการทดสอบพบว่าระบบสามารถทำงานได้ถูกต้องตามที่ออกแบบ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....ธนวัฒน์.....
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....วิวัฒน์ วัฒนาวุฒิ.....
ปีการศึกษา 2550

4871416021 : MAJOR COMPUTER SCIENCE

KEY WORD: ACTIVITY DIAGRAM / WORKFLOW MANAGEMENT SYSTEM / XMI

THANAWAT MAHATRIBHOP : DEVELOPMENT OF A WORKFLOW MANAGEMENT SYSTEM FROM UML ACTIVITY DIAGRAM. THESIS ADVISOR : ASST.PROF. WIWAT VATANAWOOD, PH.D., 101 pp.

The purpose of this research is to develop a workflow management system that takes UML activity diagrams, saved in the XMI format, as workflow specifications. The UML activity diagrams are generally used to model business activities. The system will enact the workflows to be performed in sequence, with right conditions as stipulated in the activity diagrams by populating and presenting the users their worklists based on the roles of the users. The system also displays tasks' status and stores documents which are inputs or outputs of the tasks.

The developed workflow management system is a web application. It is designed to be able to handle multiple users and multiple workflows concurrently. Additionally, the system supports task undoing, and allows modification to be made to the specifications of workflows whose instances are active. The system also records work details into log files so that work history can be viewed later on.

The system is thoroughly tested by both using real workflows and test cases. The results show that the system works correctly as designed.

Department of Computer Engineering
Field of study Computer Science
Academic year 2007

Student's signature.....
Advisor's signature.....

กิตติกรรมประกาศ

ผู้วิจัยสามารถทำวิทยานิพนธ์ฉบับนี้จนเสร็จสมบูรณ์ได้จากความช่วยเหลือทั้งทางตรงและทางอ้อมของบุคคลต่อไปนี้ ซึ่งผู้วิจัยขอกราบขอบพระคุณเป็นอย่างมาก

ผศ.ดร.วิวัฒน์ วัฒนาวุฒิ ที่กรุณาเป็นที่ปรึกษาให้ในการทำวิทยานิพนธ์ รวมถึงคำแนะนำและความช่วยเหลือตลอดการทำวิทยานิพนธ์

กรรมการสอบวิทยานิพนธ์ทั้ง 3 ท่าน ได้แก่ รศ.ดร.พรศิริ หมื่นไชยศรี, ผศ.ดร. ธาราทิพย์ สุวรรณศาสตร์ และผศ.ดร.อาทิตย์ ทองทักษ์ ที่ได้ให้ข้อคิดเห็น คำชี้แนะ และกรุณาสละเวลามาเป็นกรรมการให้แก่วิทยานิพนธ์ฉบับนี้

บิดา มารดา และพี่สาว ที่คอยให้ความช่วยเหลือ ความรัก ความห่วงใย และเป็นกำลังใจมาโดยตลอด

ผู้วิจัยทุกท่านทั้งในประเทศและต่างประเทศสำหรับการวิจัยที่วิทยานิพนธ์นี้อ้างอิงถึง รวมถึงผู้เขียนบทความและผู้พัฒนาเครื่องมือต่างๆที่ช่วยให้ผู้วิจัยได้พัฒนาความรู้ความสามารถ และเป็นผลให้วิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์ได้

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

รูป **หน้า**

รูป **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การจัดการกระแสงาน (Workflow Management) เป็นเทคโนโลยีที่ได้รับความสนใจและได้รับการพัฒนามาอย่างต่อเนื่องตลอดระยะเวลาหลายปีที่ผ่านมา [1] จุดประสงค์ของการจัดการกระแสงานก็คือการทำให้การดำเนินงานของกระบวนการใดๆเป็นไปตามกฎหรือข้อบังคับที่กำหนดไว้ [2] โดยกระบวนการในที่นี้จะมีการกำหนดลำดับขั้นตอน และเงื่อนไขต่างๆ ของการทำงานไว้อย่างชัดเจน และเพื่อให้การควบคุมให้การดำเนินงานนี้เป็นไปตามข้อกำหนด การดำเนินงานจะถูกควบคุมด้วยระบบคอมพิวเตอร์ ซึ่งเรียกว่าระบบการจัดการกระแสงาน (Workflow Management System) [3]

หลักการของระบบการจัดการกระแสงานนั้น เริ่มต้นจากระบบจะรับข้อกำหนดของกระแสงาน (Workflow Specification) เป็นอินพุต [4] ซึ่งจะทำการกำหนดข้อมูลที่จำเป็นทั้งหมดของกระแสงานซึ่งได้แก่ ลำดับของงานในกระแสงาน เงื่อนไขของแต่ละงาน บุคคลที่เป็นผู้รับผิดชอบกระทำแต่ละงาน รวมถึงข้อมูลที่จะเป็นอินพุตและเอาท์พุตของแต่ละงาน ตัวอย่างเช่น กระแสงานการจัดการโครงการซอฟต์แวร์ (Software Project Management) อาจเริ่มด้วยการรับอินพุตเป็นข้อกำหนดของความต้องการ (Requirements Specification) ของโครงการ หลังจากนั้นหัวหน้าโครงการ (Project Leader) ก็จะเลือกสมาชิกของโครงการ และต่อมาสมาชิกก็จะช่วยกันประมาณเวลาที่ใช้ในการทำโครงการ ซึ่งหัวหน้าโครงการก็จะนำผลลัพธ์จากการประมาณมาสร้างแผนของโครงการ (Project Plan) เป็นต้น ซึ่งหลังจากที่มีการสร้างข้อกำหนดของกระแสงานแล้ว ระบบการจัดการกระแสงานก็จะทำการควบคุมกระแสงานตามที่ถูกกำหนดไว้ โดยการควบคุมนี้อาจหมายความถึงการที่ผู้ใช้สามารถตรวจสอบงานที่ตัวเองได้รับมอบหมายให้ทำ ณ ขณะใดขณะหนึ่ง โดยการล็อกอินเข้ามาสู่ระบบและตรวจดูว่าตนเองมีงานใดค้างอยู่บ้าง และเมื่อผู้ใช้ทำงานเสร็จก็จะแจ้งกับระบบว่าทำงานนั้นเสร็จเรียบร้อยแล้ว ซึ่งระบบก็จะแสดงให้เห็นถึงงานในลำดับถัดไป

การสร้างข้อกำหนดของกระแสงานนั้นสามารถทำได้หลายวิธี หรืออาจเรียกว่าสามารถสร้างมาโดยภาษาที่แตกต่างกัน ทั้งนี้ระบบการจัดการกระแสงานส่วนใหญ่มักจะรับข้อกำหนดของกระแสงานที่กำหนดมาในรูปแบบภาษาเฉพาะ (Proprietary Language) ของแต่ละระบบเอง ซึ่งแต่ละภาษาก็จะมีวากยสัมพันธ์ (Syntax) และสัญกรณ์ (Notation) ที่ใช้แตกต่างกันไป การนำข้อกำหนดของกระแสงานที่กำหนดมาโดยภาษาเฉพาะของระบบหนึ่งไปใช้เป็น

อินพุตของอีกระบบหนึ่งจึงไม่สามารถทำได้ เพื่อแก้ปัญหานี้หน่วยงานที่ชื่อ The Workflow Management Coalition ซึ่งเป็นหน่วยงานที่จัดตั้งขึ้นโดยกลุ่มผู้สนใจในเทคโนโลยีการจัดการกระแสงานจึงได้นำเสนอภาษาเอ็กซ์พีดีแอล (XPDL – XML-based Process Definition Language) [5] เพื่อให้ใช้เป็นมาตรฐานสำหรับการสร้างข้อกำหนดของกระแสงาน แต่อย่างไรก็ตามภาษานี้ไม่ได้รับความนิยมมากนักเนื่องจากเครื่องมือที่ใช้สร้างข้อกำหนดของกระแสงานในภาษาเอ็กซ์พีดีแอลยังมีน้อยมาก และนอกจากนี้เอ็กซ์พีดีแอลยังเป็นภาษาใหม่ที่ผู้ใช้ทั่วไปยังไม่คุ้นเคย ในช่วงหลังจึงมีการเสนอให้มีการนำแผนภาพกิจกรรมของยูเอ็มแอล (UML Activity Diagram) มาใช้ในการสร้างข้อกำหนดของกระแสงาน [6, 7] ซึ่งมีข้อดีคือผู้ใช้โดยมากจะคุ้นเคยและปัจจุบันก็มีเครื่องมือที่สนับสนุนการสร้างแผนภาพอยู่มาก แต่อย่างไรก็ดี ปัจจุบันยังไม่มีระบบการจัดการกระแสงานที่รับอินพุตเป็นข้อกำหนดของกระแสงานที่อยู่ในรูปแบบของแผนภาพกิจกรรมโดยตรง งานวิจัยนี้จึงนำเสนอระบบการจัดการกระแสงานที่รับข้อกำหนดของกระแสงานที่อยู่ในรูปแบบของแผนภาพกิจกรรมของยูเอ็มแอล โดยระบบนี้จะตีความสัญกรณ์ต่างๆในแผนภาพกิจกรรม เพื่อให้ระบบสามารถทำการควบคุมการทำงานให้เป็นไปตามลำดับและเงื่อนไขที่ถูกกำหนดมาโดยแผนภาพกิจกรรมนั้น

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบการจัดการกระแสงานซึ่งรับข้อกำหนดของกระแสงานที่อยู่ในรูปแบบของแผนภาพกิจกรรมของยูเอ็มแอล

1.3 ขอบเขตของการวิจัย

1) งานวิจัยนี้จะรองรับแผนภาพกิจกรรมตามมาตรฐานยูเอ็มแอล เวอร์ชัน 2.0

2) รองรับแผนภาพกิจกรรมที่ใช้แสดงกระแสงานเฉพาะที่ถูกสร้างขึ้นด้วยเครื่องมือ Visual Paradigm for UML และจัดเก็บอยู่ในรูปแบบของเอ็กซ์เอ็มไอ

3) ระบบการจัดการกระแสงานนี้สามารถจัดการควบคุมการทำงานได้มากกว่าหนึ่งกระแสงาน และหลายอินสแตนซ์ของกระแสงานพร้อมกัน

4) ระบบการจัดการกระแสงานนี้มีความสามารถดังต่อไปนี้

- กำหนดผู้ร่วมกระแสงานสำหรับผู้ใช้ระบบแต่ละคน
- แสดงรายการงาน ซึ่งรวมถึงสถานะของงานสำหรับผู้ใช้แต่ละคน
- แสดงรายละเอียดของงาน

- รองรับการแก้ไขข้อกำหนดของกระแสงานที่มีอินสแตนซ์ดำเนินการอยู่ และการแก้ไขการทำงานของแต่ละอินสแตนซ์ในลักษณะทำกลับ (Undo) การทำงานบางชิ้น ภายใต้เงื่อนไขที่กำหนด
- บันทึกข้อมูลการทำงานต่างๆลงล็อกไฟล์ (Log File) และผู้ใช้สามารถมาเรียกดูภายหลังได้

1.4 วิธีดำเนินการวิจัย

1) ศึกษาทฤษฎีของกระแสงานและแผนภาพกิจกรรม
2) ศึกษาระบบการจัดการกระแสงานในปัจจุบัน
3) ออกแบบระบบการจัดการกระแสงาน
4) พัฒนาระบบการจัดการกระแสงานตามที่ได้ออกแบบไว้
5) ทดสอบการทำงาน
6) สรุปผลการวิจัยและจัดทำรูปเล่มวิทยานิพนธ์

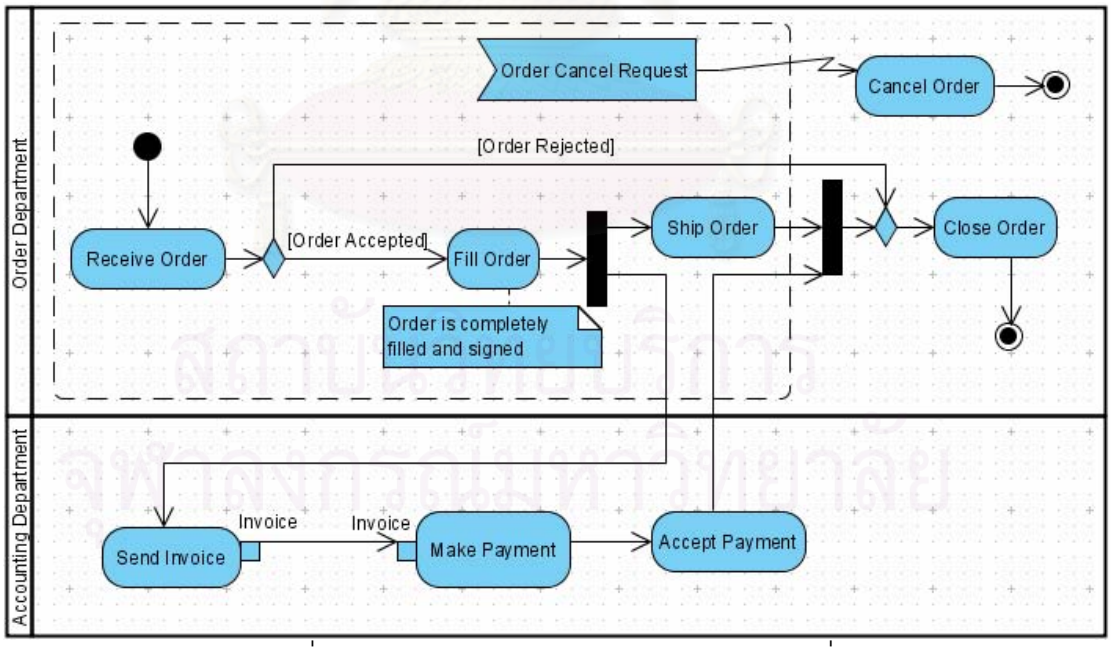
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 กระแสงาน

ก. คำจำกัดความ

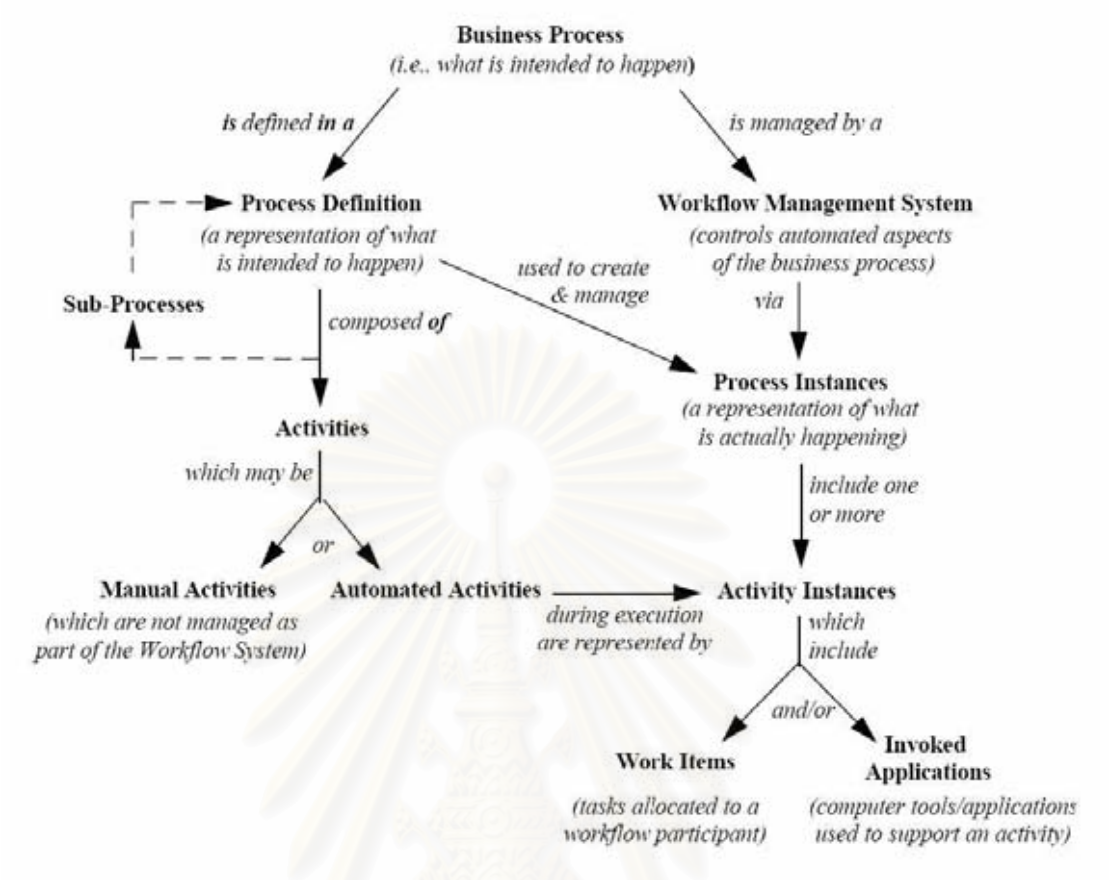
The Workflow Management Coalition ซึ่งเป็นหน่วยงานที่จัดตั้งขึ้นอันเนื่องมาจากความพยายามที่จะกำหนดมาตรฐานของการจัดการกระแสงาน ได้ให้คำจำกัดความของคำว่า “กระแสงาน” ไว้ว่าเป็น “การทำให้เป็นอัตโนมัติของกระบวนการทางธุรกิจ (Business Process) อาจจะโดยทั้งหมดหรือเพียงบางส่วน” ทั้งนี้กระบวนการทางธุรกิจดังกล่าวคือลำดับของกิจกรรม ซึ่งถูกกำหนดมาโดยนิยามของกระบวนการ (Process Definition) และอาจเรียกได้อีกอย่างหนึ่งว่าข้อกำหนดของกระแสงาน [8] ดังตัวอย่างในรูปที่ 2.1 ซึ่งเป็นข้อกำหนดของกระแสงานของการจัดการคำสั่ง



รูปที่ 2.1 ข้อกำหนดของกระแสงานการจัดการคำสั่ง

การควบคุมลำดับของการทำงานให้เป็นไปตามข้อกำหนดนั้นจะถูกทำด้วยระบบคอมพิวเตอร์ที่เรียกว่าระบบการจัดการกระแสงาน

กิจกรรมในกระแสงานแบ่งเป็นกิจกรรมที่ผู้ใช้เป็นผู้กระทำด้วยตนเอง (Manual Activities) โดยระบบการจัดการกระแสงานมีบทบาทเพียงแจ้งให้ผู้มีหน้าที่ทำกิจกรรมนั้นทราบว่าถึงเวลาเริ่มดำเนินการแล้ว เมื่อกิจกรรมนั้นถูกทำเสร็จสิ้น ผู้ใช้ก็จะทำการบันทึกกับระบบ เพื่อที่ระบบจะสามารถแจ้งให้ผู้มีหน้าที่ทำกิจกรรมถัดไปว่าถึงเวลาเริ่มดำเนินการแล้ว และกิจกรรมแบบอัตโนมัติ (Automated Activities) ซึ่งระบบการจัดการกระแสงานจะเป็นผู้ดำเนินการเองหรือเรียกแอพพลิเคชันอื่นให้ทำกิจกรรมนั้นให้ และเรียกแอพพลิเคชันนี้ว่าแอพพลิเคชันที่ถูกเรียก (Invoked Applications) ทั้งนี้ในการดำเนินงานจริงๆนั้น ระบบการจัดการกระแสงานจะทำการสร้างอินสแตนซ์ของกระบวนการ (Process Instance) หรือเรียกอีกอย่างว่าอินสแตนซ์ของกระแสงาน (Workflow Instance) ขึ้นมาจากข้อกำหนดของกระแสงานและทำการควบคุมการทำงานของอินสแตนซ์ของกระแสงานนี้ ตัวอย่างเช่นกระบวนการการจัดการโครงการซอฟต์แวร์นั้นแต่ละโครงการก็จะถือว่าเป็นคนละอินสแตนซ์แต่จะมีข้อกำหนดของลำดับการทำงานที่หมือนกัน ในอินสแตนซ์ของกระแสงานจะประกอบด้วยอินสแตนซ์ของกิจกรรมจำนวนหนึ่ง และในแต่ละกิจกรรมนั้นก็จะประกอบด้วยงานตั้งแต่หนึ่งชิ้นขึ้นไป ซึ่งงานแต่ละชิ้นที่ถูกกำหนดให้ผู้ร่วมกระแสงาน (Workflow Participant) แต่ละคนทำจะเรียกว่าชิ้นงาน (Work Item) ระบบการจัดการกระแสงานจะทำการควบคุมและแสดงให้ผู้ร่วมกระแสงานเห็นถึงกลุ่มของชิ้นงานที่ผู้ร่วมในกระแสงานเหล่านั้นถูกมอบหมายให้ทำ ณ ขณะใดขณะหนึ่ง กลุ่มของงานเหล่านี้ถูกเรียกว่ารายการงาน (Worklist) ซึ่งอาจจะแตกต่างกันไปสำหรับผู้ร่วมกระแสงานแต่ละคน รูปที่ 2.2 แสดงศัพท์ (Terminology) ต่างๆในเทคโนโลยีการจัดการกระแสงานซึ่งถูกกำหนดโดย The Workflow Management Coalition



รูปที่ 2.2 ศัพท์ต่างๆในการจัดการกระแสงาน [1]

ข. คุณภาพของกระแสงาน

ดังที่กล่าวมาแล้วว่ากระแสงานเป็นการแสดงลำดับการทำงานพร้อมทั้งเงื่อนไขต่างๆในการทำงาน การดูคุณภาพของกระแสงานจะใช้สิ่งที่เรียกว่าซาวด์เนสของกระแสงาน (Workflow Soundness) [9] ซึ่งได้ให้คุณสมบัติที่ดีของกระแสงานไว้ดังนี้

- ในกระแสงานต้องไม่มีงานที่ไม่สามารถทำไปถึงได้
- ในกระแสงานต้องไม่มีเดดล็อก (Dead Lock)
- จุดสิ้นสุดของกระแสงานต้องระบุชัดเจน

ค. ประโยชน์ของกระแสงาน

โดยทั่วไปประโยชน์ของการนำระบบการจัดการกระแสงานมาใช้สามารถแบ่งได้เป็น 3 ประเภท [10] คือ

- การประหยัดค่าใช้จ่ายโดยตรง (Direct Cost Savings) เนื่องจากการใช้ระบบการจัดการกระแสงานทำให้สามารถใช้พนักงานได้อย่างคุ้มค่ามากขึ้น รวมถึงอาจสามารถลดจำนวนพนักงานลงได้ เนื่องจากงานบางอย่างสามารถทำได้โดยอัตโนมัติ
- การประหยัดค่าใช้จ่ายที่ซ่อนอยู่ (Hidden Cost Savings) ในที่นี้เป็นค่าใช้จ่ายที่ประหยัดได้จริงแต่สามารถวัดได้ยาก ตัวอย่างเช่น สามารถลดเวลาของผู้จัดการในการควบคุมการทำงาน
- ประโยชน์ที่ไม่อาจวัดหรือจับต้องได้ (Intangible Benefits) หมายความถึงประโยชน์ที่ไม่ได้มีลักษณะเป็นตัวเลขที่แสดงออกมาให้เห็นชัดจริง ตัวอย่างเช่น ความพึงพอใจของพนักงาน

ง. โครงสร้างและความสามารถของระบบการจัดการกระแสงาน

โครงสร้างของระบบการจัดการกระแสงานนั้นสามารถแบ่งออกได้เป็นสองส่วน ได้แก่ ส่วนกำหนดแบบกระแสงาน (Workflow Modelling Component) และ ส่วนปฏิบัติการกระแสงาน (Workflow Execution Component) โดยทั่วไประบบการจัดการกระแสงานจะมีความสามารถดังนี้ [2]

- รองรับการสร้างข้อกำหนดของกระแสงาน และสามารถตีความข้อกำหนดของกระแสงานได้ถูกต้อง
- รองรับการสร้างอินสแตนซ์ของกระแสงานที่ระบบรู้จัก รวมถึงการจัดการกับอินสแตนซ์ทั้งหมด เช่นการยกเลิกอินสแตนซ์
- สามารถควบคุมลำดับและเงื่อนไขการทำงานของอินแตนซ์ได้ดังที่กำหนดในข้อกำหนดของกระแสงาน
- แสดงรายการงานที่เหมาะสมของผู้ใช้แต่ละคนในแต่ละเวลา รวมถึงข้อมูลของงานที่อาจเป็นการส่งต่อมาของงานก่อนหน้า
- สามารถเรียกแอพพลิเคชันภายนอกให้ทำงานได้ และตรวจสอบผลลัพธ์ของการทำงานนั้น
- สามารถนำเสนอข้อมูลการทำงานต่างๆที่ผ่านมาแล้วได้ โดยอาจมีลักษณะเป็นล็อกที่บันทึกข้อมูลต่างๆ เช่น การล็อกอินเข้าสู่ระบบของผู้ใช้ หรือการบันทึกการทำงานเสร็จ เป็นต้น

- รองรับการแทรกแซงการทำงานแบบทำด้วยมือ (Manual Intervention) ในกรณีที่มีเหตุการณ์ผิดพลาดเกิดขึ้น (Exception)

จ. ภาษาที่ใช้สร้างข้อกำหนดของกระแสงาน

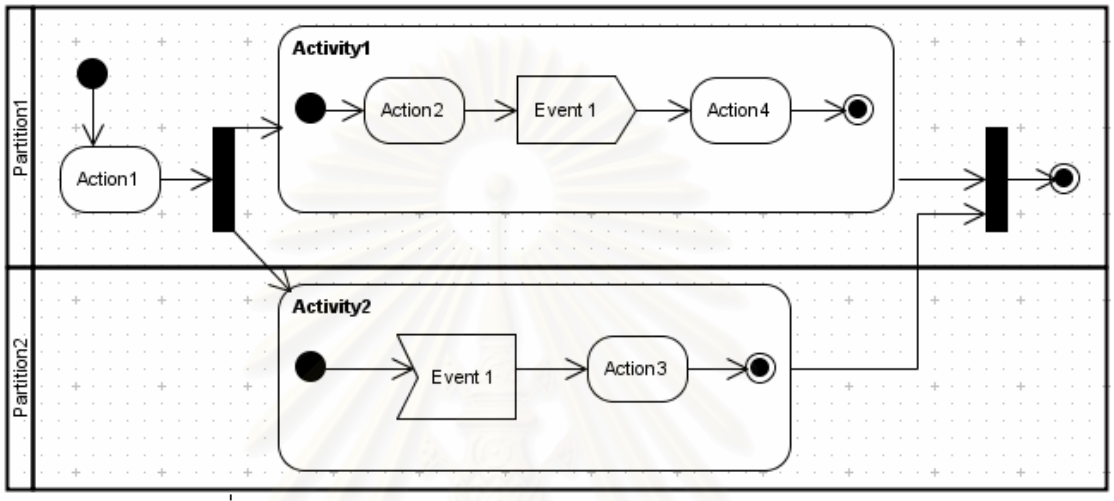
ในปัจจุบันภาษาที่เกี่ยวข้องในการใช้สร้างข้อกำหนดของกระแสงานที่เป็นที่นิยมได้แก่

- XPDL (XML Process Definition Language) [5] เป็นภาษาที่นำเสนอโดย WfMC โดยจะมีลักษณะเป็นเท็กซ์ (Text) ที่อยู่ในรูปแบบมาตรฐานเอ็กซ์เอ็มแอล โดยไม่มีสัญกรณ์เฉพาะในการสร้างข้อกำหนดของกระแสงาน
- BPML (Business Process Modelling Language) เป็นภาษาที่นำเสนอโดย BPMI (Business Process Management Initiative) โดยสัญกรณ์ที่รองรับหลักความคิดของภาษา BPML ในการใช้สร้างข้อกำหนดของกระแสงานเรียกว่า BPMN (Business Process Modelling Notation) [11]
- แผนภาพกิจกรรมของยูเอ็มแอล ถูกนำเสนอโดย OMG (Object Management Group) ซึ่งมีสัญกรณ์ที่ใช้แสดงรูปแบบของการทำงานแบบต่างๆ

ฉ. การประเมินระบบการจัดการกระแสงาน

ในการที่จะประเมินหรือเปรียบเทียบระบบการจัดการกระแสงานนั้น Jablonski และ Bussler [12] ได้เสนอหลักความคิดของทัศนมิติ (Perspective) สำหรับวิเคราะห์ระบบการจัดการกระแสงาน เช่น ทัศนมิติด้านกระแสควบคุม (Control-Flow Perspective) จะบรรยายถึงกิจกรรมและลำดับของการทำกิจกรรม ทัศนมิติด้านกระแสข้อมูล (Data-Flow Perspective) จะบรรยายถึงข้อมูล เช่นเอกสารที่จะถูกเคลื่อนย้ายไปตามกระแสของกิจกรรม และทัศนมิติด้านทรัพยากร (Resource Perspective) บรรยายถึงคนและบทบาทของคนที่จะเป็นผู้กระทำแต่ละกิจกรรม นอกจากนี้ van der Aalst และคณะ [13] ได้ทำการวิเคราะห์การทำงาน (Functionality) ของระบบการจัดการกระแสงานในปัจจุบันตามแต่ละทัศนมิติ และได้ให้แบบรูปของกระแสงาน (Workflow Patterns) ออกมา ซึ่งแต่ละแบบรูปของกระแสงานนั้นหมายถึงลักษณะการทำงานที่พบได้บ่อยในกระแสงาน ตัวอย่างเช่นแบบรูป “Milestone Pattern” ซึ่งเป็นหนึ่งในแบบรูปด้านกระแสการควบคุม หมายถึงการทำงานที่ต้องรับรู้ว่ามีเหตุการณ์ที่สนใจเกิดขึ้น เพื่อจุดประสงค์บางอย่าง ซึ่งเหตุการณ์นี้ถูกเรียกว่า Milestone ดังตัวอย่างในรูปที่ 2.3 ที่ใช้แผนภาพกิจกรรมในการเขียนแบบรูป Milestone เพื่อแสดงว่าการกระทำ Action3 ในกิจกรรม Activity2 จะ

ถูกเริ่มทำได้ก็ต่อเมื่อการกระทำ Action2 ในอีกกิจกรรมหนึ่งถูกทำเสร็จสิ้นแล้ว ทั้งนี้ในการประเมินระบบการจัดการกระแสงานนั้นคือการวิเคราะห์ว่าระบบนั้นสามารถรองรับการทำงานในแบบรูปนี้ได้หรือไม่ ปัจจุบันการใช้แบบรูปของกระแสงานเป็นวิธีที่นิยมมากที่สุดในการประเมินและเปรียบเทียบระบบการจัดการกระแสงานและภาษาที่ใช้กำหนดข้อกำหนดของกระแสงาน



รูปที่ 2.3 ตัวอย่างการใช้แผนภาพกิจกรรมเขียนแบบรูป Milestone

2.1.2 แผนภาพกิจกรรมของยูเอ็มแอล

แผนภาพกิจกรรมเป็นแผนภาพหนึ่งในแผนภาพของยูเอ็มแอล ใช้ในการบรรยายว่าแต่ละกิจกรรมมีการประสานงานกันอย่างไร ซึ่งรวมถึงลำดับและเงื่อนไขในการทำแต่ละกิจกรรม รวมถึงสามารถระบุได้ว่าผู้กระทำแต่ละกิจกรรมคือใคร หรือวัตถุที่ถูกกระทำในแต่ละกิจกรรมคืออะไร ทั้งนี้แต่ละกิจกรรมอาจประกอบด้วยการกระทำ (Action) ตั้งแต่หนึ่งการกระทำขึ้นไป ในยูเอ็มแอลเวอร์ชัน 2.0 นั้นมีสัญกรณ์ที่สามารถใช้ในแผนภาพกิจกรรมมากมาย ทั้งนี้กิจกรรมได้ถูกแบ่งเป็นหลายแพ็คเกจที่มีความซับซ้อนมากขึ้นตามลำดับ ซึ่งได้แก่

- FundamentalActivities จะกำหนดความหมายของการกระทำและกิจกรรม
- BasicActivities จะกำหนดและรองรับสัญกรณ์ของกระแสควบคุมแบบพื้นฐานของการกระทำ
- ImmediateActivities จะกำหนดและรองรับสัญกรณ์ที่ซับซ้อนขึ้นของกระแสควบคุม เช่นสัญกรณ์โหนด Fork, โหนด Decision รวมถึงกำหนดและรองรับสัญกรณ์ของกระแสข้อมูล

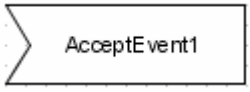
- CompleteActivities จะกำหนดและรองรับสัญกรณ์ที่มีลักษณะเป็นเหตุการณ์ เช่น AcceptEventAction
- StructuredActivities จะกำหนดส่วนประกอบที่มักพบได้ทั่วไปในการเขียนโปรแกรม เช่น ลำดับ (Sequence), เงื่อนไข (Condition) เป็นต้น
- CompleteStructuredActivities จะกำหนดกระแสข้อมูลของ StructuredActivities
- ExtraStructuredActivities จะกำหนดและรองรับสัญกรณ์ของการจัดการข้อผิดพลาด (Exception Handling)

โดยรายละเอียดของสัญกรณ์ทั้งหมดในแผนภาพกิจกรรมของยูเอ็มแอลมีดังนี้

- โหนด Action และโหนด Activity เป็นสัญกรณ์ที่ใช้แสดงการกระทำและกิจกรรมในแผนภาพกิจกรรมตามลำดับ โดยโหนด Action แสดงถึงการกระทำที่เป็นส่วนย่อยที่สุดที่ไม่สามารถมีการกระทำอื่นอยู่ภายในได้อีก แต่โหนด Activity โดยทั่วไปใช้แสดงชุดของการกระทำ กล่าวคือสามารถมีหลายโหนด Action หรือโหนดชนิดอื่นๆอยู่ซ้อนภายในได้ รูปที่ 2.4 แสดงสัญกรณ์ของโหนด Action ชื่อ Action1 และโหนด Activity ชื่อ Activity1

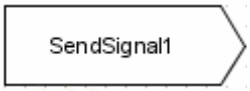
รูปที่ 2.4 สัญกรณ์ของโหนด Action และโหนด Activity

- โหนด AcceptEventAction เป็นสัญกรณ์ที่ใช้แสดงถึงการรอรับเหตุการณ์ (Event) ในแผนภาพกิจกรรม ซึ่งเมื่อเหตุการณ์ที่มีชื่อเดียวกับชื่อของโหนด AcceptEventAction เกิดขึ้นจะส่งผลให้เกิดการทำงานต่อไปยังโหนดที่เชื่อมต่อกับโหนด AcceptEventAction นั้น รูปที่ 2.5 แสดงสัญกรณ์ของโหนด AcceptEventAction ชื่อ AcceptEvent1



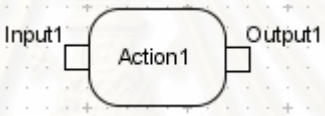
รูปที่ 2.5 สัญกรณ์ของโหนด AcceptEventAction

- โหนด SendSignalAction เป็นสัญกรณ์ที่ใช้แสดงถึงการเกิดเหตุการณ์ในแผนภาพกิจกรรม โดยเหตุการณ์ที่เกิดคือเหตุการณ์ที่มีชื่อเดียวกับชื่อของโหนด SendSignalAction รูปที่ 2.6 แสดงสัญกรณ์ของโหนด SendSignalAction ชื่อ SendSignal1



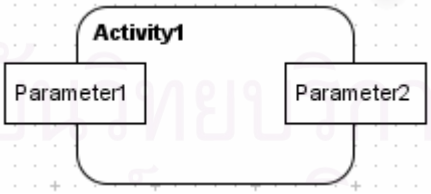
รูปที่ 2.6 สัญกรณ์ของโหนด SendSignalAction

- โหนด Input Pin และโหนด Output Pin เป็นสัญกรณ์ที่ใช้แสดงถึงวัตถุที่เป็นอินพุตและเอาท์พุตของโหนด Action ตามลำดับ ดังรูปที่ 2.7 โหนด Action ชื่อ Action1 มีโหนด Input Pin ชื่อ Input1 เป็นอินพุต และโหนด Output Pin ชื่อ Output1 เป็นเอาท์พุต



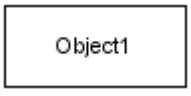
รูปที่ 2.7 สัญกรณ์ของโหนด Input Pin และโหนด Output Pin

- โหนด ActivityParameter เป็นสัญกรณ์ที่ใช้แสดงถึงวัตถุที่เป็นอินพุตหรือเอาท์พุตของโหนด Activity ดังรูปที่ 2.8 โหนด Activity ชื่อ Activity1 มีอินพุตชื่อ Parameter1 และเอาท์พุตชื่อ Parameter2



รูปที่ 2.8 สัญกรณ์ของโหนด ActivityParameter

- โหนด Object เป็นสัญกรณ์ที่ใช้แสดงถึงวัตถุที่เป็นอินพุตหรือเอาท์พุตของโหนด Action หรือโหนด Activity มีลักษณะดังรูปที่ 2.9



รูปที่ 2.9 สัญกรณ์ของโหนด Object

- โหนด Initial เป็นสัญกรณ์ที่ใช้แสดงจุดเริ่มต้นของแผนภาพกิจกรรม มีลักษณะดังรูปที่ 2.10

รูปที่ 2.10 สัญกรณ์ของโหนด Initial

- โหนด ActivityFinal เป็นสัญกรณ์ที่ใช้แสดงจุดสิ้นสุดของแผนภาพกิจกรรม โดยกิจกรรมและการกระทำทั้งหมดในแผนภาพกิจกรรมนั้นจะถูกยกเลิกไปเมื่อการทำงานดำเนินไปถึงโหนด ActivityFinal ซึ่งมีลักษณะดังรูปที่ 2.11

รูปที่ 2.11 สัญกรณ์ของโหนด ActivityFinal

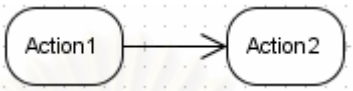
- โหนด FlowFinal เป็นสัญกรณ์ที่แสดงจุดสิ้นสุดของกระแสการทำงาน มีลักษณะดังรูปที่ 2.12

รูปที่ 2.12 สัญกรณ์ของโหนด FlowFinal

- โหนด ActivityPartition เป็นสัญกรณ์ที่แสดงบทบาทของผู้รับผิดชอบทำแต่ละการกระทำหรือกิจกรรมในแผนภาพกิจกรรม มีลักษณะดังรูปที่ 2.13 ซึ่งแสดงถึง 2 บทบาท คือ Partition1 และ Partition2

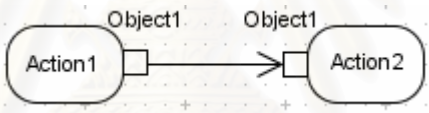
รูปที่ 2.13 สัญกรณ์ของโหนด ActivityPartition

- Control Flow เป็นสัญกรณ์ที่ใช้เชื่อมโหนด 2 โหนดเข้าด้วยกันเพื่อแสดงลำดับของการกระทำและกิจกรรมในแผนภาพกิจกรรม โดยโหนดที่ถูกเชื่อมเข้าด้วยกันต้องไม่มีโหนดที่ใช้แทนวัตถุ (ไม่ใช่โหนด Input Pin, Output Pin, ActivityParameter, Object) ดังรูปที่ 2.14 แสดง Control Flow ที่เชื่อม 2 โหนด Action เข้าด้วยกัน



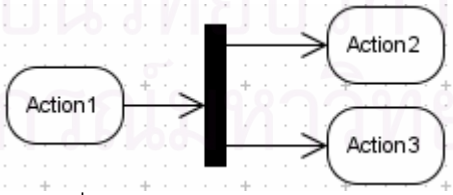
รูปที่ 2.14 สัญกรณ์ของ Control Flow

- Object Flow เป็นสัญกรณ์ที่ใช้เชื่อมโหนดที่ใช้แทนวัตถุ (Input Pin, Output Pin, ActivityParameter, Object) 2 โหนดเข้าด้วยกันเพื่อแสดงลำดับของการกระทำและกิจกรรมในแผนภาพกิจกรรม ดังรูปที่ 2.15 แสดง Object Flow ที่เชื่อมโหนด Input Pin เข้ากับโหนด Output Pin



รูปที่ 2.15 สัญกรณ์ของ Object Flow

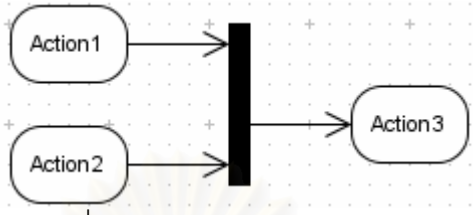
- โหนด Fork เป็นสัญกรณ์ที่แสดงลักษณะการทำงานที่แตกเป็นหลายกระแสการทำงานซึ่งสามารถดำเนินการไปได้พร้อมกัน ดังรูปที่ 2.16 แสดงโหนด Fork ที่แตกกระแสการทำงาน 1 กระแสออกเป็น 2 กระแสการทำงาน



รูปที่ 2.16 สัญกรณ์ของโหนด Fork

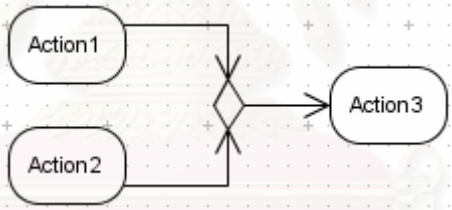
- โหนด Join เป็นสัญกรณ์ที่แสดงการรวมหลายกระแสการทำงานให้เป็นกระแสเดียว ซึ่งทุกกระแสการทำงานต้องถูกทำให้เสร็จสิ้นก่อนการเริ่มทำงานในกระแสที่ถูกรวม

จากโหนด Join นั้น ดังรูปที่ 2.17 แสดงโหนด Join ที่รวม 2 กระแสการทำงานเป็น 1 กระแสการทำงาน



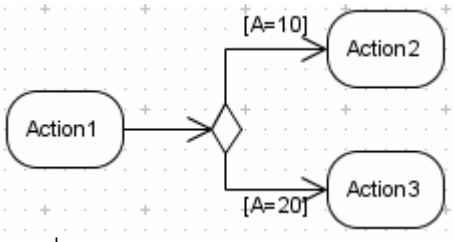
รูปที่ 2.17 สัญกรณ์ของโหนด Join

- โหนด Merge เป็นสัญกรณ์ที่แสดงลักษณะการทำงานที่รวมหลายกระแสการทำงานเป็นหนึ่งกระแส แต่ต่างจากโหนด Join ตรงที่การกระทำหรือกิจกรรมในกระแสที่ออกจากโหนด Merge สามารถเริ่มทำได้ทันทีหากกระแสการทำงานที่เข้าไปยังโหนด Merge เพียงกระแสเดียวถูกทำเสร็จสิ้น ดังรูปที่ 2.18 แสดงโหนด Merge ที่รวม 2 กระแสการทำงานเป็น 1 กระแสการทำงาน



รูปที่ 2.18 สัญกรณ์ของโหนด Merge

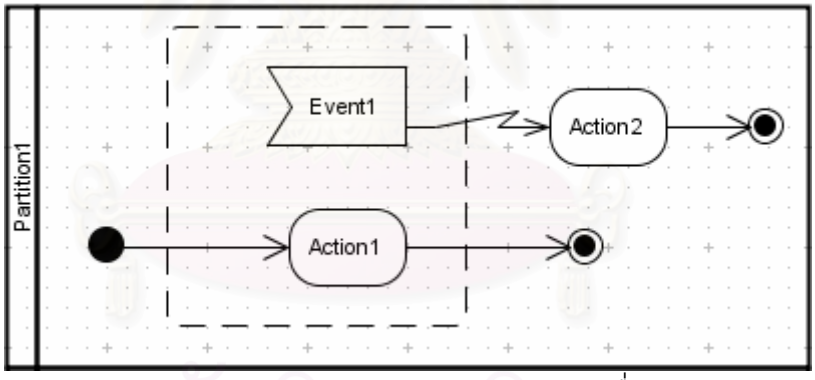
- โหนด Decision เป็นสัญกรณ์ที่ใช้แสดงการเลือกกระแสการทำงานตามเงื่อนไขที่ระบุไว้ในเส้นเชื่อมขาออก (Outgoing Edge) ทั้งนี้จะมีเงื่อนไขที่เป็นจริงได้เพียงเงื่อนไขเดียวเท่านั้น ดังรูปที่ 2.19 แสดงโหนด Decision ที่มีเงื่อนไขของเส้นเชื่อมขาออกเป็น “A=10” และ “A=20”



รูปที่ 2.19 สัญกรณ์ของโหนด Decision

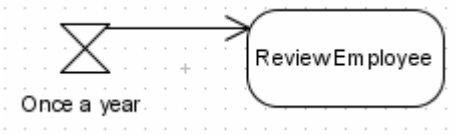
- โหนด InterruptibleActivityRegion เป็นสัญกรณ์ที่ใช้แสดงกลุ่มของการทำงานที่สามารถถูกยกเลิกไปได้หากมีเหตุการณ์ที่ระบุไว้เกิดขึ้น มีลักษณะดังรูปที่ 2.20 ทั้งนี้การวาดแบบในลักษณะนี้ต้องใช้โหนด InterruptibleActivityRegion คู่กับเส้นเชื่อมชนิด ExceptionHandler ที่เชื่อมระหว่างเหตุการณ์ที่อยู่ภายในโหนด InterruptibleActivityRegion นั้นและโหนดภายนอก ซึ่งหากเหตุการณ์นั้นเกิดขึ้น การทำงานที่ดำเนินการอยู่ในกลุ่มนั้นจะถูกยกเลิกทันที แล้วการกระทำหรือกิจกรรมถัดไปก็คือโหนดที่เป็นปลายทางของเส้นเชื่อม ExceptionHandler นั้น ดังรูปที่ 2.21 แสดงการใช้โหนด InterruptibleActivityRegion คู่กับเส้นเชื่อมชนิด ExceptionHandler ซึ่งเชื่อมระหว่างเหตุการณ์ Event1 และการกระทำ Action2

รูปที่ 2.20 สัญกรณ์ของโหนด InterruptibleActivityRegion



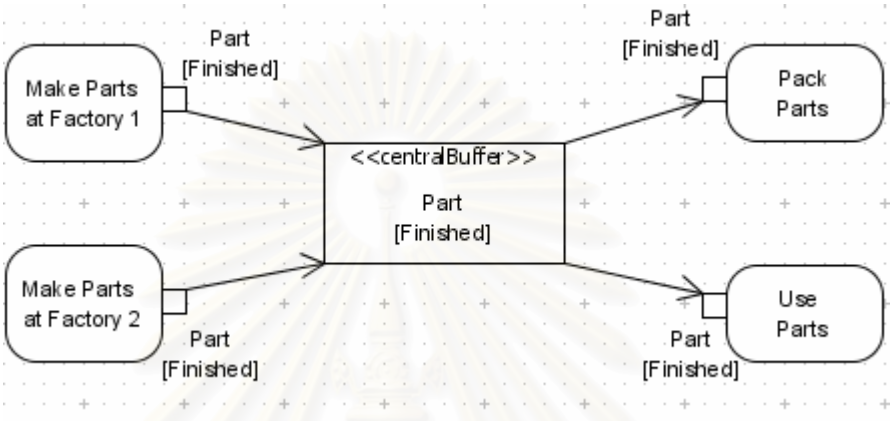
รูปที่ 2.21 การใช้สัญกรณ์ InterruptibleActivityRegion คู่กับเส้นเชื่อมชนิด ExceptionHandler

- โหนด AcceptTimeEventAction เป็นสัญกรณ์ที่ใช้แสดงเหตุการณ์ที่เกิดขึ้นเมื่อถึงเวลาที่กำหนด ดังตัวอย่างในรูปที่ 2.22 แสดงการใช้โหนด AcceptTimeEventAction ซึ่งบอกถึงการกระทำ Review Employee ที่จะถูกทำปีละครั้ง



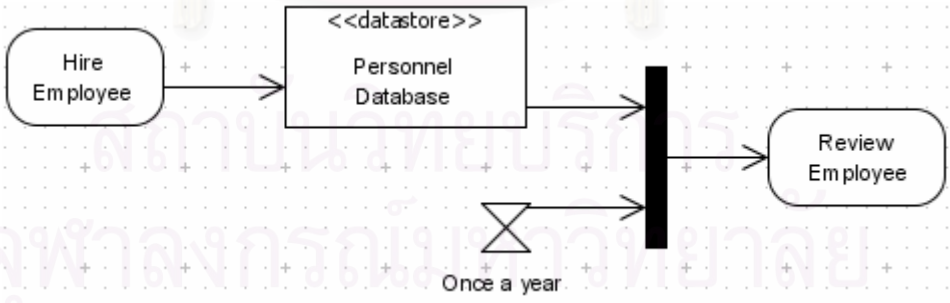
รูปที่ 2.22 สัญกรณ์ของโหนด AcceptTimeEventAction

- โหนด CentralBuffer เป็นสัญกรณ์ที่ใช้แสดงถึงวัตถุเช่นเดียวกับโหนด Object แต่แตกต่างกันตรงที่โหนด CentralBuffer ไม่ผูกติดกับโหนด Action ใดๆโดยตรง โดยการใช้เป็นไปในลักษณะที่บอกถึงการเก็บวัตถุไว้ชั่วคราวก่อนที่จะส่งวัตถุนั้นต่อไปตาม Object Flow ดังตัวอย่างในรูปที่ 2.23



รูปที่ 2.23 สัญกรณ์ของโหนด CentralBuffer

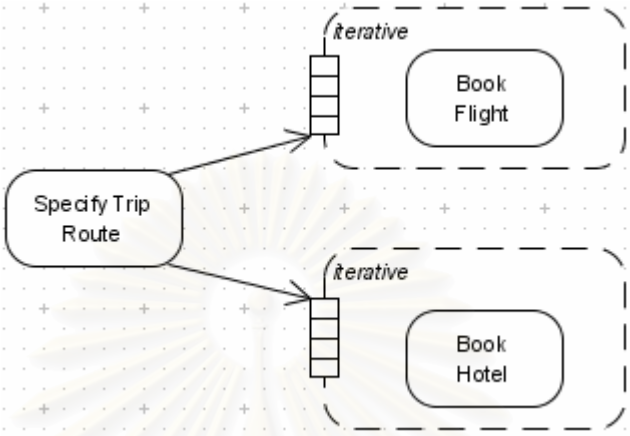
- โหนด DataStore เป็นสัญกรณ์ที่ใช้แสดงถึงการเก็บวัตถุเช่นเดียวกับโหนด CentralBuffer แต่จะบอกถึงการเก็บที่จะนำวัตถุนั้นออกมาใช้ต่อเมื่อต้องการเท่านั้น ซึ่งอาจจะเก็บไว้นานเท่าใดก็ได้ ดังตัวอย่างในรูปที่ 2.24 แสดงการใช้โหนด DataStore ที่เก็บข้อมูลของพนักงาน และจะถูกนำออกมาใช้ปีละครั้งเมื่อทำการกระทำ Review Employee



รูปที่ 2.24 สัญกรณ์ของโหนด DataStore

- โหนด ExpansionRegion เป็นสัญกรณ์ที่ใช้แสดงกลุ่มของการกระทำที่ถูกทำหลายครั้งตามจำนวนอินพุต ดังตัวอย่างในรูปที่ 2.25 แสดงการใช้โหนด ExpansionRegion ที่มีการกระทำ Book Flight และการกระทำ Book Hotel อยู่ภายใน โดยการกระทำ Specify Trip

Route อาจทำให้เกิดการจองหลายเที่ยวบินและจองหลายโรงแรม แต่จำนวนอินพุตนั้นจะไม่ถูกระบุอย่างชัดเจนในแผนภาพกิจกรรม



รูปที่ 2.25 สัญกรณ์ของโหนด ExpansionRegion

2.1.3 มาตรฐานเอ็กซ์เอ็มไอ (XMI) [14]

เอ็กซ์เอ็มไอ (XMI – XML Metadata Interchange) เป็นรูปแบบมาตรฐานที่ใช้ในการแลกเปลี่ยนวัตถุใดๆ เช่น แผนภาพของยูเอ็มแอล มาตรฐานนี้ถูกนำเสนอโดย OMG เอ็กซ์เอ็มไอจะถูกกำหนดอยู่ในรูปแบบของภาษาเอ็กซ์เอ็มแอล (XML – Extensible Markup Language) ดังตัวอย่างในรูปที่ 2.26

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1"
         xmlns:uml="http://schema.omg.org/spec/UML/2.0"
         xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">

  <xmi:Documentation xmi:Exporter="Visual Paradigm for UML"
                     xmi:ExporterVersion="6.0.1"/>

  <uml:Model name="ProcessOrderWithEvents2" xmi:id="Au0w4YiD.AAAAQAF">
    <ownedMember xmi:id="activity_id" xmi:type="uml:Activity">
      <xmi:Extension xmi:Extender="Visual Paradigm for UML">
        <diagram/>
      </xmi:Extension>

      <node name="Swimlane"
            xmi:id=".umw4YiD.AAAAQCR"
            xmi:type="activitySwimlane2">

        <horizontalPartition xmi:idref="temw4YiD.AAAAQCm"/>
        <horizontalPartition xmi:idref="N.mw4YiD.AAAAQCt"/>
```

รูปที่ 2.26 ตัวอย่างเอกสารเอ็กซ์เอ็มไอ

จากรูปที่ 2.26 เป็นตัวอย่างเอกสารเอ็กซ์เอ็มไอที่คัดลอกบางส่วนมาจากแผนภาพกิจกรรมการจัดการกับคำสั่ง ส่วนที่เป็นแท็ก <xmi:XMI> จะระบุเวอร์ชันของเอ็กซ์เอ็มไอของเอกสารนี้ ดังรูปเป็นเวอร์ชัน 2.1 ซึ่งเป็นเวอร์ชันล่าสุดในปัจจุบัน แท็ก <xmi:Documentation> จะระบุชื่อพร้อมทั้งเวอร์ชันของเครื่องมือที่สร้างเอกสารเอ็กซ์เอ็มไอ ส่วนต่อจากนั้นจะเป็นแท็กที่แสดงถึงข้อมูลของออบเจ็คท์ (Object) ดังตัวอย่างนี้เป็นแผนภาพยูเอ็มแอล ในส่วนนี้จะไม่มีการกำหนดแท็กที่แน่นอน ซึ่งทำให้แต่ละเครื่องมือที่ใช้สร้างเอกสารเอ็กซ์เอ็มไอมีแท็กที่แตกต่างกัน ในปัจจุบันเครื่องมือวาดแผนภาพยูเอ็มแอลส่วนมากจะมีความสามารถในการอิมพอร์ตและเอ็กซ์พอร์ตเอกสารเอ็กซ์เอ็มไอของแผนภาพที่วาด

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 แผนภาพกิจกรรมกับการกำหนดกระแสงาน

ในช่วงหลายปีที่ผ่านมาได้มีการศึกษาถึงความเหมาะสมที่จะนำแผนภาพกิจกรรมของยูเอ็มแอลมาใช้ในการสร้างข้อกำหนดของกระแสงาน โดยงานวิจัยแทบทั้งหมดจะใช้แบบรูปของกระแสงานในการประเมินดังที่ได้กล่าวมาแล้วในส่วนของทฤษฎีที่เกี่ยวข้องเรื่องกระแสงาน โดยในงานวิจัยของ Russell และคณะ [15] ซึ่งใช้แบบรูปของกระแสงานในการประเมินสัญกรณ์ของแผนภาพกิจกรรมว่าสามารถสร้างแบบแบบรูปใดได้บ้าง ได้ทำการประเมินอย่างครบถ้วนทั้งในทัศนมิติด้านกระแสการควบคุม ด้านกระแสข้อมูล และด้านทรัพยากร ทั้งนี้ในงานวิจัยสรุปว่าแผนภาพกิจกรรมจะเหมาะสมกับการใช้กำหนดกระแสงานในทัศนมิติด้านกระแสการควบคุมและกระแสข้อมูลเท่านั้น แต่ไม่เหมาะที่จะใช้กำหนดกระแสงานในทัศนมิติด้านอื่นๆ เช่นด้านทรัพยากร ตัวอย่างเช่นสัญกรณ์ของแผนภาพกิจกรรมจะไม่สามารถแสดงการแจกจ่ายงานในลักษณะที่ผู้ใช้เป็นผู้ระบุงานที่จะทำด้วยตนเอง โดยสรุปแล้วแผนภาพกิจกรรมสามารถรองรับได้ 17 แบบรูปจาก 20 แบบรูปด้านกระแสการควบคุม, 18 แบบรูปจาก 40 แบบรูปด้านกระแสข้อมูล และ 8 แบบรูปจาก 43 แบบรูปด้านทรัพยากร

สำหรับปัญหาของแผนภาพกิจกรรมได้ถูกนำเสนออย่างชัดเจนโดย Schattkowsky และ Forster [16] ซึ่งโดยมากเกิดจากความไม่ชัดเจนทางด้านความหมาย (Semantic) และวากยสัมพันธ์ของแผนภาพกิจกรรม ตัวอย่างเช่นกระแสข้อมูลที่ผ่านโหนด Fork ว่าข้อมูลที่เป็นอินพุตนั้นจะต้องถูกแตกออกเป็นหลายอินสแตนซ์สำหรับแต่ละงานที่ต่อกับโหนด Fork นั้น หรืองานเหล่านั้นจะอ้างอิงไปยังอินพุตที่เป็นอินสแตนซ์เดียว

นอกจากนี้ Eshuis และ Weiringa [8] ได้หาความหมายของแผนภาพกิจกรรมที่เหมาะสมกับการใช้กำหนดกระแสงานรวมถึงวิธีตรวจสอบความถูกต้องเพื่อที่ภายหลังจะสามารถใช้ตัวตรวจเช็คแบบ (Model Checker) ในการตรวจสอบความถูกต้องของระบบการจัดการกระแสงานได้

มีงานวิจัยจำนวนหนึ่งที่นำเสนอวิธีการแปลงจากแผนภาพกิจกรรมให้ไปอยู่ในรูปแบบภาษาเอ็กซ์พีดีแอล ตัวอย่างเช่น Guelfi และ Mamma [17] รวมถึง Jiang, Mair และ Newman [18] ได้นำเสนอกฎการแปลงเชิงรูปนัย (Formal Translation Rules) รวมถึงเสนอวิธีการทำให้เหมาะมากที่สุด (Optimization) ของข้อกำหนดที่ถูกแปลงมาเป็นภาษาเอ็กซ์พีดีแอลแล้ว และยังนำเสนอวิธีการตรวจสอบความถูกต้องหลังการแปลง

นอกจากนี้ยังมีการนำเสนอสัญกรณ์ใหม่ๆเข้าไปในแผนภาพกิจกรรม เพื่อให้สามารถสร้างข้อกำหนดของกระแสงานได้อย่างครบถ้วน เช่นงานวิจัยของ Bastos [6] ได้นำเสนอหลักความคิดของแผนภาพกิจกรรมสำหรับกระแสงาน (Workflow Activity Diagram) โดยมีการเพิ่มสเตอริโอไทพ์ (Stereotype) ต่างๆเข้าไปในแผนภาพกิจกรรมปกติ

2.2.2 Visual Paradigm for UML [19]

เครื่องมือ Visual Paradigm for UML เป็นเครื่องมือหนึ่งที่นิยมใช้ในการสร้างแผนภาพของยูเอ็มแอลซึ่งถูกพัฒนาโดยบริษัท Visual Paradigm โดยเครื่องมือนี้รองรับการเอ็กซ์พอร์ตแผนภาพยูเอ็มแอลให้อยู่ในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ รวมถึงอิมพอร์ตไฟล์เอ็กซ์เอ็มไอกลับคืน เครื่องมือนี้ยังรองรับการสร้างโค้ด (Code Generation) ขึ้นมาจากแผนภาพ โดยสามารถสร้างโค้ดให้อยู่ในรูปแบบภาษาได้มากกว่า 10 ภาษา

บทที่ 3
การวิเคราะห์แผนภาพกิจกรรม

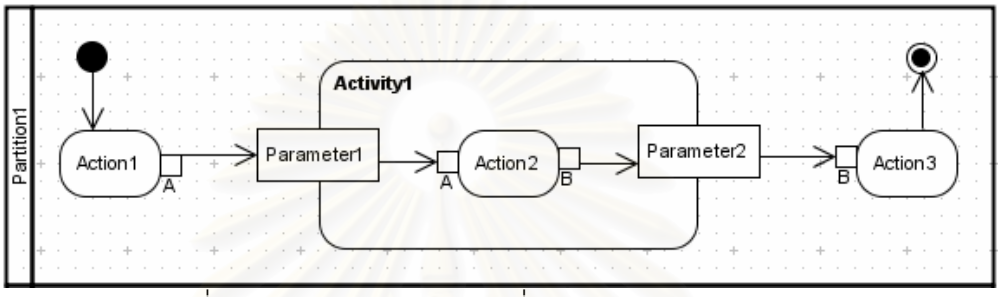
3.1 การตีความสัญกรณ์ในแผนภาพกิจกรรม

สัญกรณ์ในแผนภาพกิจกรรมของยูเอ็มแอลนั้นมีมากมายดังที่ได้สรุปไว้ในบทที่ 2 ซึ่งบางส่วนก็ไม่เกี่ยวข้องโดยตรงกับการจัดการกระแสงาน นอกจากนั้นในงานวิจัยนี้เป็นการวิเคราะห์ไฟล์เอ็กซ์เอ็มไอที่เอ็กซ์พอร์ตมาจากเครื่องมือ Visual Paradigm for UML ซึ่งการวาดแผนภาพกิจกรรมด้วยเครื่องมือนี้จะมีบางข้อมูลที่เป็นข้อมูลเฉพาะของเครื่องมือ ในบทนี้จะเป็นการอธิบายถึงรายละเอียดของการตีความสัญกรณ์ในแผนภาพกิจกรรมทั้งหมดที่ระบบรองรับ รวมถึงข้อกำหนดและข้อจำกัดที่ผู้สร้างแผนภาพกิจกรรมด้วยเครื่องมือ Visual Paradigm for UML ต้องยึดถือตาม เพื่อที่ว่าระบบการจัดการกระแสงานที่พัฒนาขึ้นจะสามารถตีความได้อย่างถูกต้อง ทั้งนี้ข้อกำหนดบางข้อถูกนำมาจากเอกสารข้อกำหนดของแผนภาพกิจกรรมของยูเอ็มแอล 2.0 [20] ซึ่งระบบก็จะทำการตรวจสอบให้ หากแผนภาพกิจกรรมไม่เป็นไปตามข้อกำหนดต่างๆ ระบบจะถือว่าแผนภาพกิจกรรมนั้นไม่ถูกต้องและไม่ยอมรับแผนภาพกิจกรรมนั้น ซึ่งระบบจะบอกข้อมูลความไม่ถูกต้องนี้กับผู้ใช้งานที่ใส่แผนภาพกิจกรรมที่อยู่ในรูปแบบไฟล์เอ็กซ์เอ็มไอนั้นเข้ามา ดังรายละเอียดในภาคผนวก ง ทั้งนี้สัญกรณ์ที่ระบบการจัดการกระแสงานนี้รองรับนั้นได้มาจากการวิเคราะห์ถึงส่วนประกอบที่สำคัญที่ใช้แสดงกระแสงาน ซึ่งได้แก่ งาน, วัตถุที่ถูกส่งผ่านไปในกระแสงาน, บทบาทของผู้รับผิดชอบทำงานแต่ละงาน, ลำดับการทำงานแบบต่างๆ และเหตุการณ์ที่อาจเกิดขึ้นได้จากภายนอกที่มีผลให้เกิดการทำงานหรือยกเลิกการทำงานบางชิ้น การวิเคราะห์นี้เป็นการพิจารณาว่าแต่ละสัญกรณ์ของแผนภาพกิจกรรมของยูเอ็มแอลสามารถนำมาใช้แทนแต่ละส่วนประกอบของกระแสงานได้อย่างไร

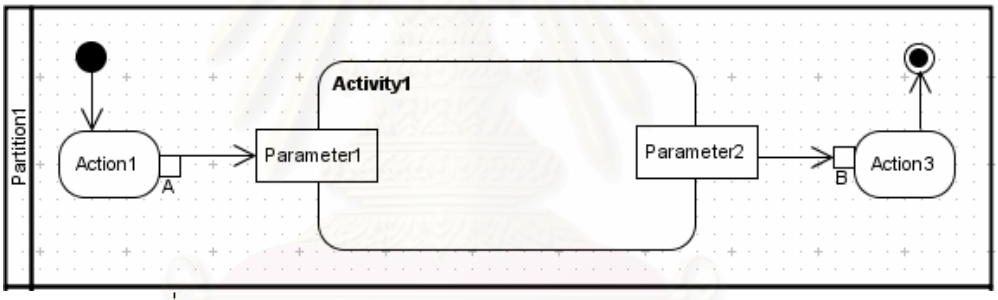
3.1.1 โหนด Action / โหนด Activity

โหนด Action และ โหนด Activity เป็นสัญกรณ์ที่ใช้แทนการกระทำและกิจกรรมในแผนภาพกิจกรรมตามลำดับ แต่โหนด Activity โดยทั่วไปใช้แสดงชุดของการกระทำ กล่าวคือสามารถมีหลายโหนด Action หรือโหนดชนิดอื่นๆอยู่ซ้อนภายในได้ อย่างไรก็ตามโดยปกติใน 1 แผนภาพกิจกรรม จะไม่มีมากกว่า 1 โหนด Activity เนื่องจากมีข้อจำกัดในข้อกำหนดของ UML 2.0 ว่าโหนด ActivityParameter ซึ่งใช้แทนอินพุตหรือเอาท์พุตของ Activity จะมีได้เพียงเส้นเชื่อมขาเข้า (Incoming Edge) หรือเส้นเชื่อมขาออก (Outgoing Edge) อย่างใดอย่างหนึ่งเท่านั้น [20] ทำให้ไม่สามารถวาดแบบให้โหนด Activity รับอินพุตมาจากงานก่อนหน้าแล้ว

ส่งผ่านเข้าไปยังโหนด Action ภายใน หรือรับเอาท์พุตจากโหนด Action ภายในแล้วส่งผ่านไปยังงานต่อไปได้ ดังรูปที่ 3.1 ซึ่งถือว่าผิดข้อกำหนดของยูเอ็มแอล ในงานวิจัยนี้จึงกำหนดให้โหนด Activity ห้ามมีโหนดอื่นๆอยู่ภายใน โดยทั้ง 2 สัญกรณ์จะใช้หมายความถึง “งาน” ในระบบการจัดการกระแสงานนี้แบบเดียวกันทุกประการ ทั้งนี้ชื่อของโหนด Action และโหนด Activity จะใช้แทนชื่อของงาน ดังรูปที่ 3.2 แสดงถึงกระแสงานที่มี 3 งาน ชื่อ Action1, Activity1, Action3



รูปที่ 3.1 การใช้โหนด Activity ที่ผิดข้อกำหนดของยูเอ็มแอล



รูปที่ 3.2 การใช้โหนด Action และ Activity แทนงานในกระแสงาน

โดยสรุปข้อกำหนดและข้อจำกัดมีดังนี้

1) ชนิดของโหนด Action จะต้องตั้งค่าในเครื่องมือ Visual Paradigm for UML เป็น “Unspecified” ซึ่งหมายถึงเป็น Action ธรรมดาที่ไม่มีแผนภาพกิจกรรมซ้อนอยู่ภายใน

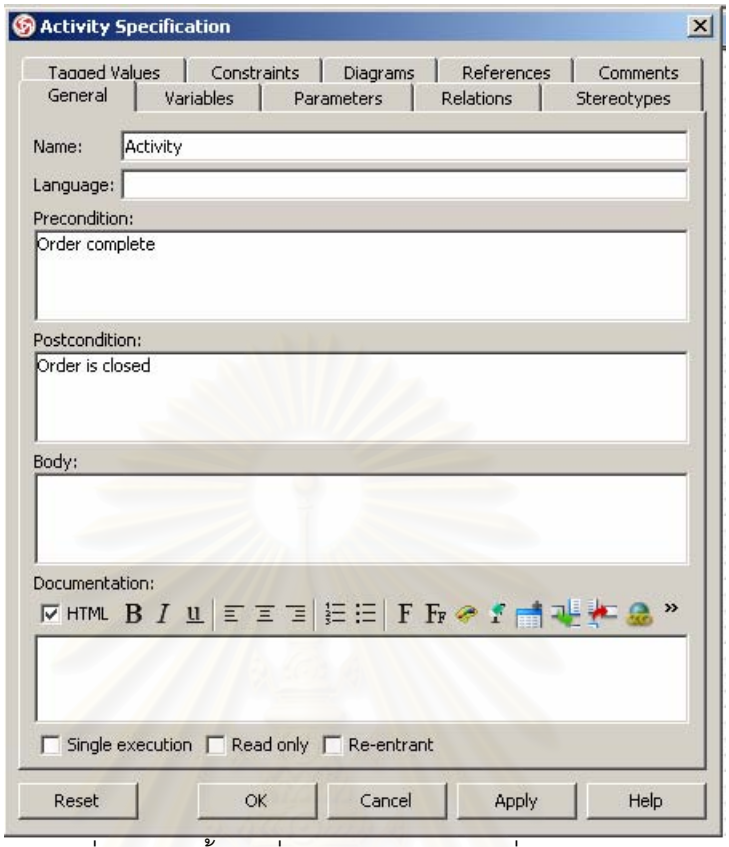
2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด Activity มีโหนดอื่นอยู่ภายใน

3) การตั้งค่าระยะเวลา (Duration) สามารถทำได้โดยการใส่ค่าในส่วนที่เป็น Summary ของข้อกำหนดคอมเมนต์ของโหนด Action หรือโหนด Activity นั้น โดยระบบจะตีความค่าที่ใส่นี้ว่าเป็นระยะเวลาที่วางแผนว่างานนั้นจะใช้เวลาทำนานเท่าใด มีหน่วยเป็นวัน ทั้งนี้วิธีการตั้งค่านี้เป็นการตั้งค่าทางอ้อม เนื่องจากโดยปกติเครื่องมือ Visual Paradigm for UML ไม่

รองรับการตั้งค่าระยะเวลา งานวิจัยนี้จึงเสนอให้ใช้วิธีการดังกล่าวเป็นการตั้งค่าระยะเวลาโดยไม่ต้องใส่หน่วย แต่ระบบจะตีความให้เป็น “วัน” ดังตัวอย่างในรูปที่ 3.3 การที่ใส่เพียงตัวเลข “4” จะมีความหมายว่า 4 วัน ทั้งนี้การกำหนดให้ใช้หน่วย “วัน” มาจากระยะเวลาของการทำงานโดยทั่วไปที่มักใช้หน่วยของระยะเวลาเป็น “วัน” ซึ่งระบบก็จะแสดงข้อมูลนี้ให้ผู้ใช้งานเห็น ทั้งนี้หากข้อมูลที่ใส่ไม่ถูกต้อง เช่นค่าที่ใส่ไม่ใช่ตัวเลข ระบบจะตัดการตั้งค่านี้ทิ้ง

รูปที่ 3.3 การตั้งค่าระยะเวลา

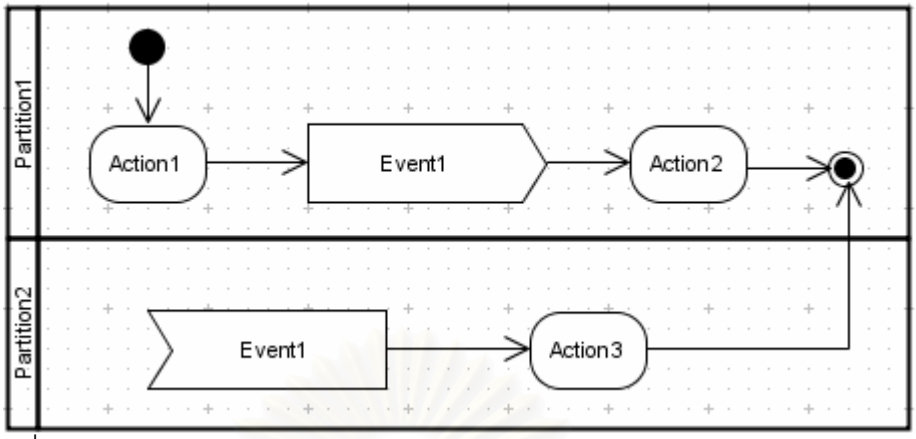
4) การตั้งค่าเงื่อนไขก่อนหน้า (Precondition) หรือเงื่อนไขภายหลัง (Postcondition) สามารถทำได้หากเป็นโหนด Activity โดยใส่ในส่วนที่เป็นข้อกำหนดของโหนด Activity ดังรูปที่ 3.4



รูปที่ 3.4 การตั้งค่าเงื่อนไขก่อนหน้าและเงื่อนไขภายหลัง

3.1.2 โหนด AcceptEventAction / โหนด SendSignalAction

โหนด AcceptEventAction ใช้ระบุการรอรับเหตุการณ์ ซึ่งอาจทำให้เกิดการทำงานบางอย่างหรือยกเลิกการทำงานบางอย่างในกระแสงาน (เมื่อใช้คู่กับ โหนด InterruptibleActivityRegion) ส่วนโหนด SendSignalAction ใช้ระบุการเกิดเหตุการณ์ที่เกิดจากภายใน เช่นเหตุการณ์ที่เกิดขึ้นเมื่องานชิ้นหนึ่งในกระแสงานถูกทำเสร็จสิ้น แล้วอาจมีผลเช่นเดียวกับการใช้โหนด AcceptEventAction โดยสรุปเหตุการณ์สามารถเกิดขึ้นได้จาก 2 กรณี ดังรูปที่ 3.5 ผู้ใช้ที่มีบทบาทเป็น Partition2 จะรอคอยเหตุการณ์ Event1 ให้เกิดขึ้นจึงจะสามารถเริ่มทำงาน Action3 ได้ ซึ่งเหตุการณ์ Event1 อาจเกิดขึ้นเมื่องาน Action1 ถูกทำเสร็จสิ้นโดยผู้ใช้ที่มีบทบาทเป็น Partition1 หรือเกิดเหตุการณ์นี้จากภายนอก โดยระบบการจัดการกระแสงานนี้จะรองรับการเกิดเหตุการณ์ทั้ง 2 แบบ ทั้งนี้ระบบจะมีช่องทางให้ผู้ใช้งานสามารถระบุได้ว่าเกิดเหตุการณ์ขึ้นจากภายนอก

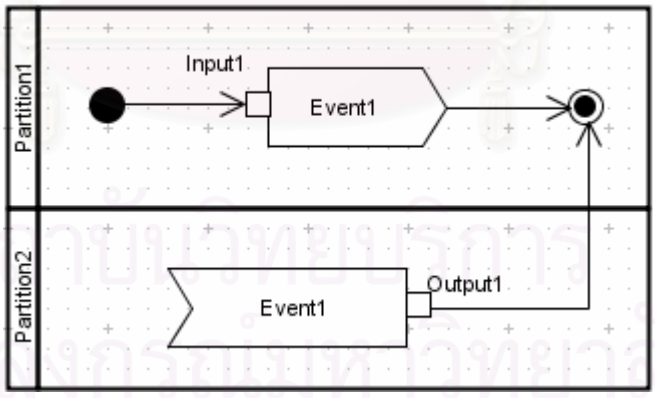


รูปที่ 3.5 ตัวอย่างการใช้โหนด AcceptEventAction และ โหนด SendSignalAction

ข้อกำหนดและข้อจำกัดของโหนด AcceptEventAction และโหนด SendSignalAction มีดังนี้

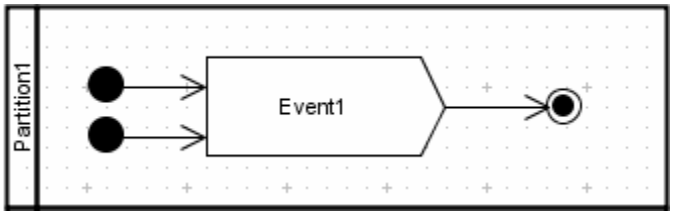
1) ชนิดของโหนดทั้งสองจะต้องตั้งค่าเป็น “Unspecified” ซึ่งหมายถึงเป็นโหนดธรรมดาที่ไม่มีแผนภาพกิจกรรมซ้อนอยู่ภายใน (เช่นเดียวกับโหนด Action)

2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด AcceptEventAction หรือโหนด SendSignalAction มีโหนด Input Pin หรือโหนด Output Pin ดังตัวอย่างในรูป 3.6



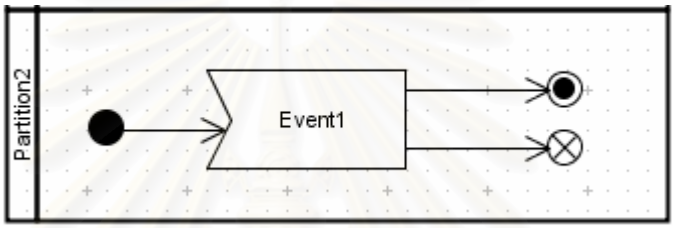
รูปที่ 3.6 การใช้โหนด AcceptEventAction หรือโหนด SendSignalAction ที่ไม่ถูกต้อง 1

3) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีมากกว่าหนึ่งเส้นเชื่อมขาเข้าไปยังโหนด AcceptEventAction หรือโหนด SendSignalAction ดังตัวอย่างในรูป 3.7



รูปที่ 3.7 การใช้โหนด AcceptEventAction หรือโหนด SendSignalAction ที่ไม่ถูกต้อง 2

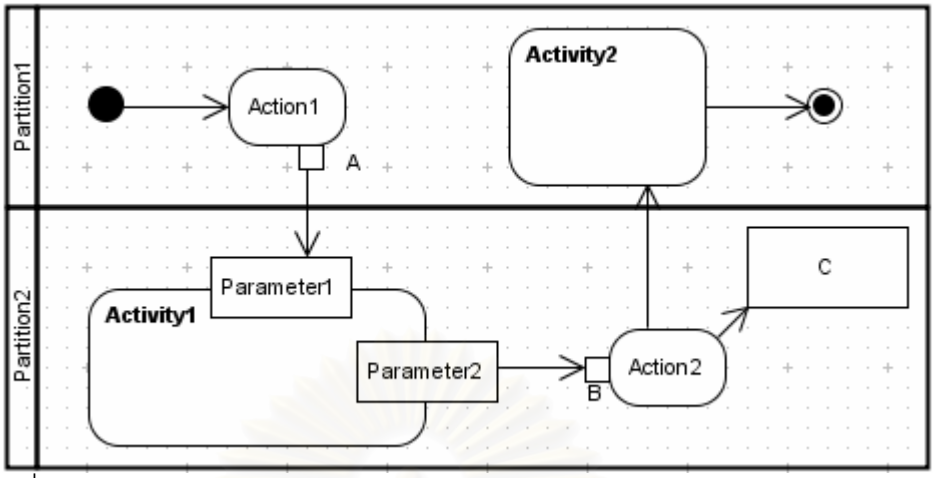
4) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออกจากโหนด AcceptEventAction หรือโหนด SendSignalAction ดังรูป 3.8



รูปที่ 3.8 การใช้โหนด AcceptEventAction หรือโหนด SendSignalAction ที่ไม่ถูกต้อง 3

3.1.3 โหนด Input Pin / โหนด Output Pin / โหนด ActivityParameter / โหนด Object

โหนดทั้ง 4 ชนิดจะใช้แสดงถึงอินพุตและเอาท์พุตของงาน โดยในระบบการจัดการกระแสงานนี้อินพุตและเอาท์พุตหมายถึงเอกสารที่เกี่ยวข้องกับการทำงาน ซึ่งระบบจะแสดงให้ผู้ใช้งานเห็นว่าอินพุตและเอาท์พุตของแต่ละงานคืออะไร ผู้ใช้งานมีหน้าที่ในการอัพโหลดเอกสารให้ระบบไปเก็บไว้เพื่อใช้เรียกดูภายหลัง หรือเพื่อเป็นอินพุตของงานถัดไป ดังตัวอย่างในรูปที่ 3.9 งาน Action1 มีเอาท์พุตชื่อ A ซึ่งจะเป็นอินพุตของงาน Activity1 ที่ชื่อ Parameter1 (การตั้งชื่ออาจตั้งให้ต่างกันได้) ในกรณีนี้จำเป็นต้องอัพโหลดเอกสารเพื่อเป็นเอาท์พุต A ก่อนจึงจะสามารถบันทึกการทำงานเสร็จสิ้นของงาน Action1 ได้ เช่นเดียวกันเอาท์พุต Parameter2 ของงาน Activity1 จะเป็นอินพุตชื่อ B ของงาน Action2 และงาน Action2 ก็มีเอาท์พุตเป็น C ซึ่งไม่เป็นอินพุตของงานใดต่อ ในกรณีต้องใช้โหนด Object แทน ทั้งนี้ระบบอนุญาตให้สามารถมีหลายอินพุตหรือเอาท์พุตสำหรับแต่ละโหนด Action หรือโหนด Activity รวมถึงให้มีชื่อซ้ำกันได้

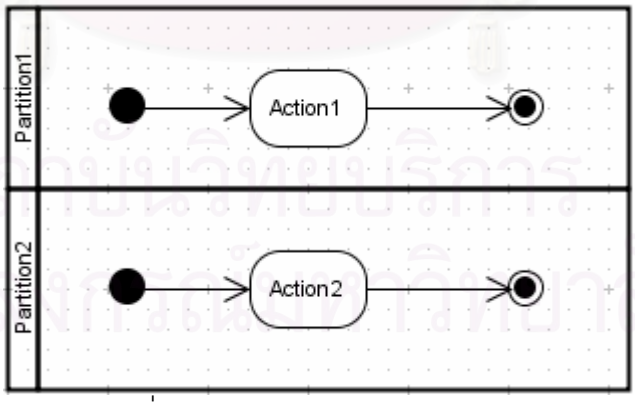


รูปที่ 3.9 ตัวอย่างการใช้โหนด Input Pin / Output Pin / ActivityParameter / Object

ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากชื่อของโหนด Input Pin / โหนด Output Pin / โหนด ActivityParameter / โหนด Object ไม่แสดงค่า (เป็นค่าว่างเปล่า)

3.1.4 โหนด Initial

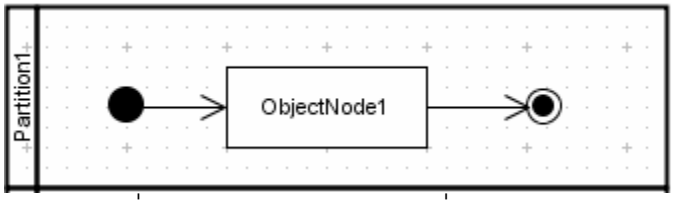
โหนด Initial ใช้แสดงถึงจุดเริ่มต้นของแผนภาพกิจกรรม ซึ่งหมายถึงจุดเริ่มต้นของกระแสงาน กล่าวคือเมื่ออินสแตนซ์ของกระแสงานเกิดขึ้น งานที่ต่อกับโหนด Initial จะถูกสร้างขึ้นทันที ทั้งนี้ระบบอนุญาตให้มีมากกว่า 1 โหนด Initial ใน 1 กระแสงานได้ ดังตัวอย่างในรูป 3.10 งาน Action1 และงาน Action2 จะเกิดขึ้นทันทีที่อินสแตนซ์ของกระแสงานนี้เกิดขึ้น



รูปที่ 3.10 ตัวอย่างการใช้โหนด Initial

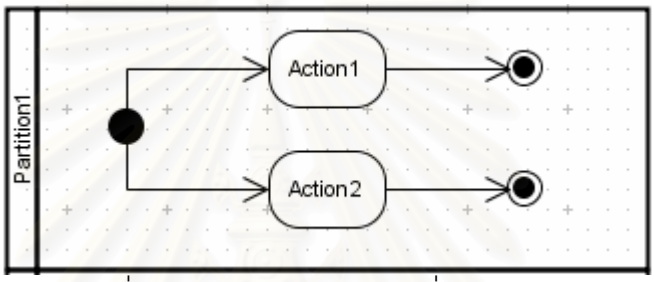
ข้อกำหนดและข้อจำกัดของโหนด Initial มีดังนี้

1) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีโหนด Object ต่อกับโหนด Initial โดยตรง ดังรูปที่ 3.11



รูปที่ 3.11 การใช้โหนด Initial ที่ไม่ถูกต้อง 1

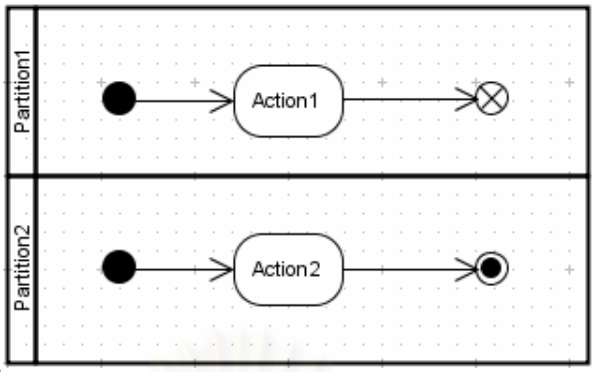
2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด Initial ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก ดังรูปที่ 3.12



รูปที่ 3.12 การใช้โหนด Initial ที่ไม่ถูกต้อง 2

3.1.5 โหนด ActivityFinal / FlowFinal

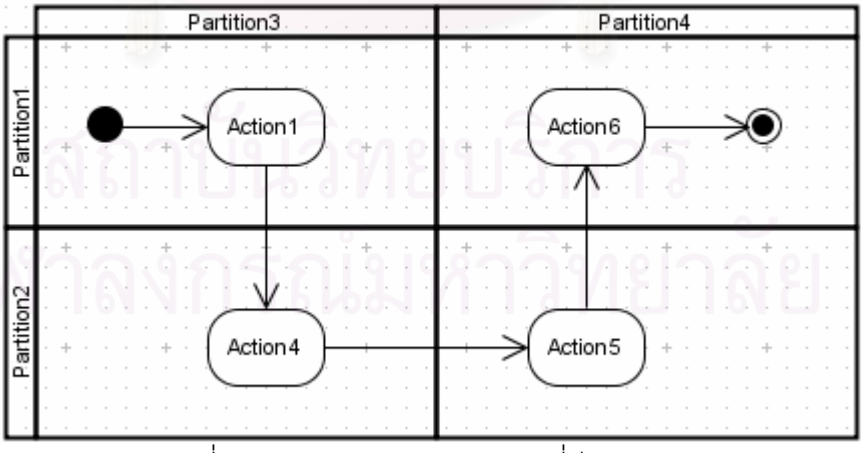
โหนด ActivityFinal ใช้แสดงจุดสิ้นสุดของกิจกรรม ซึ่งหมายถึงจุดสิ้นสุดของกระแสงาน กล่าวคือเป็นการสิ้นสุดของทุกกระแสการทำงานที่ดำเนินอยู่ในขณะนั้น ส่วนโหนด FlowFinal ใช้แสดงจุดสิ้นสุดของกระแสเพียงกระแสหนึ่ง ในระบบการจัดการกระแสงานนี้ หากการทำงานของอินสแตนซ์หนึ่งดำเนินไปถึงโหนด ActivityFinal อินสแตนซ์นั้นจะเปลี่ยนสถานะการทำงานเป็น “Done” ทันที และงานที่ค้างอยู่ในขณะนั้นจะถูกยกเลิก (เปลี่ยนสถานะเป็น “Cancelled”) ทั้งนี้ระบบอนุญาตให้มีมากกว่า 1 โหนด ActivityFinal หรือ โหนด FlowFinal ดังตัวอย่างในรูป 3.13 หากงาน Action2 ถูกทำเสร็จ ระบบจะถือว่าการทำงานของอินสแตนซ์นั้นเสร็จสิ้นทันที (เปลี่ยนสถานะเป็น “Done”) แม้ว่างาน Action1 จะยังดำเนินการอยู่ก็ตาม ในขณะที่หากงาน Action1 ถูกทำเสร็จ ระบบก็จะยังคงรอคอยให้งาน Action2 ถูกทำเสร็จสิ้นก่อนที่จะเปลี่ยนสถานะของอินสแตนซ์เป็น “Done” ทั้งนี้ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากไม่มีเส้นเชื่อมขาเข้าไปยัง โหนด ActivityFinal หรือโหนด FlowFinal เลย



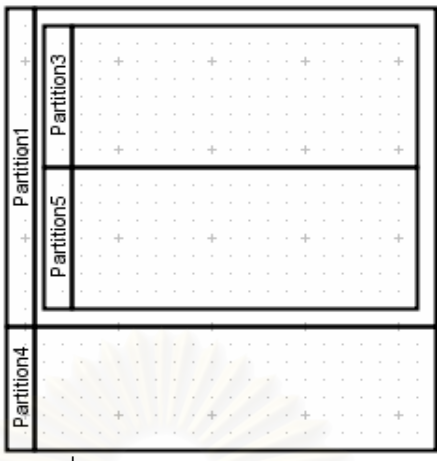
รูปที่ 3.13 ตัวอย่างการใช้โหนด ActivityFinal และ FlowFinal

3.1.6 โหนด ActivityPartition

โหนด ActivityPartition ใช้แสดงถึงบทบาทที่มีหน้าที่ในการทำงานแต่ละงานในแผนภาพกิจกรรม ทั้งนี้โหนด ActivityPartition อาจมีหลายมิติดังรูปที่ 3.14 ซึ่งในงานวิจัยนี้กำหนดให้ระบบนำชื่อบทบาทที่ระบุด้านแนวนอนมารวมกับชื่อบทบาทที่ระบุด้านแนวตั้ง ดังนั้นในกรณีนี้ระบบจะตีความได้เป็น 4 บทบาท ได้แก่ “Partition1 Partition3”, “Partition1 Partition4”, “Partition2 Partition3” และ “Partition2 Partition4” กล่าวคือดังรูปผู้ใช้ที่มีหน้าที่ทำงาน Action1 คือผู้ใช้ที่มีบทบาทเป็น “Partition1 Partion3” และผู้ใช้ที่มีหน้าที่ทำงาน Action4 คือผู้ใช้ที่มีบทบาทเป็น “Partition2 Partition3” ซึ่งการจับคู่ของผู้ใช้กับบทบาทสามารถทำได้เมื่อสร้างผู้ใช้ใหม่ ทั้งนี้ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีการใช้การแบ่งส่วนย่อย (Sub-partition) ซึ่งมีลักษณะดังรูป 3.15



รูปที่ 3.14 โหนด ActionPartition ที่มีหลายมิติ



รูปที่ 3.15 การใช้การแบ่งส่วนย่อย

3.1.7 Control Flow / Object Flow

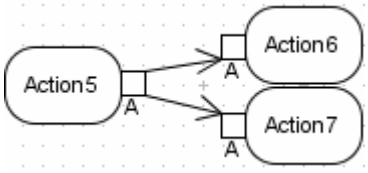
Control Flow และ Object Flow จะใช้แสดงลำดับของการทำงานในกระแสงาน

ข้อกำหนดและข้อจำกัดของ Control Flow / Object Flow มีดังนี้

1) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีมากกว่า 1 Control Flow เป็นเส้นเชื่อมขาเข้าไปยังโหนด Action หรือโหนด Activity หรือมากกว่า 1 Control Flow ที่เป็นเส้นเชื่อมขาออกจาก โหนด Action หรือโหนด Activity ดังรูปที่ 3.16

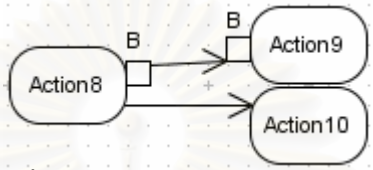
รูปที่ 3.16 โหนด Action ที่มีมากกว่า 1 Control Flow เป็นเส้นเชื่อมขาออก

2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีมากกว่า 1 Object Flow เชื่อมต่อกับ 1 โหนด Input Pin, Output Pin ดังรูปที่ 3.17 หรือโหนด ActivityParameter นอกจากนี้โหนด Object ก็จะมีได้แค่ 1 Object Flow เป็นเส้นเชื่อมขาเข้าและขาออกเท่านั้น



รูปที่ 3.17 โหนด Output Pin ที่มีมากกว่า 1 Object Flow เป็นเส้นเชื่อมขาออก

3) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากมีทั้ง Control Flow และ Object Flow เป็นเส้นเชื่อมขาเข้าไปยังโหนด Action หรือโหนด Activity หรือเป็นเส้นเชื่อมขาออกจากโหนด Action หรือโหนด Activity ดังรูปที่ 3.18 กล่าวคือโหนด Action หรือโหนด Activity จะต้องมีเส้นเชื่อมขาเข้าที่เป็น Control Flow หรือ Object Flow ทั้งหมด และมีเส้นเชื่อมขาออกที่เป็น Control Flow หรือ Object Flow ทั้งหมดเช่นเดียวกัน



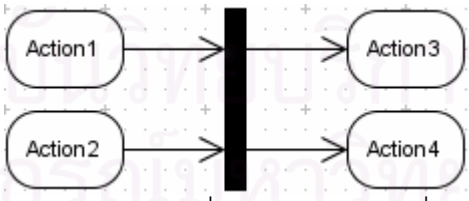
รูปที่ 3.18 โหนด Action ที่มีทั้ง Control Flow และ Object Flow เป็นเส้นเชื่อมขาออก

3.1.8 โหนด Fork / โหนด Join

โหนด Fork ใช้แสดงลักษณะการทำงานที่แตกเป็นหลายกระแสการทำงานซึ่งสามารถดำเนินการไปได้พร้อมกัน ส่วนโหนด Join แสดงการรวมหลายกระแสการทำงานให้เป็นกระแสเดียว ซึ่งทุกกระแสการทำงานต้องถูกทำให้เสร็จสิ้นก่อนการเริ่มทำงานในกระแสที่ถูกรวมจากโหนด Join นั้น

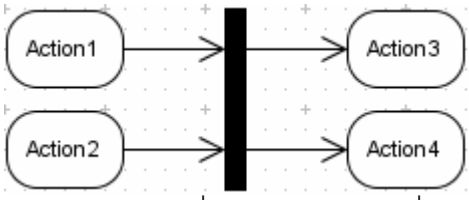
ข้อกำหนดและข้อจำกัดของโหนด Fork / โหนด Join มีดังนี้

1) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหาก 1 โหนด Fork ไม่มีหรือมีมากกว่า 1 เส้นเชื่อมขาเข้า ดังตัวอย่างในรูป 3.19



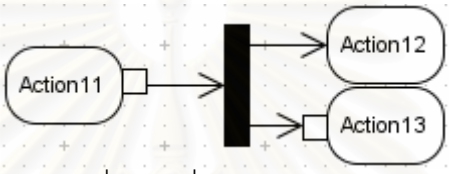
รูปที่ 3.19 โหนด Fork ที่มีมากกว่า 1 เส้นเชื่อมขาเข้า

2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหาก 1 โหนด Join ไม่มีหรือมีมากกว่า 1 เส้นเชื่อมขาออก ดังตัวอย่างในรูป 3.20 (จะมีลักษณะคล้ายกับรูป 3.19 แต่โหนดในรูปที่ 3.20 เป็นโหนด Join ในขณะที่รูป 3.19 เป็นโหนด Fork)



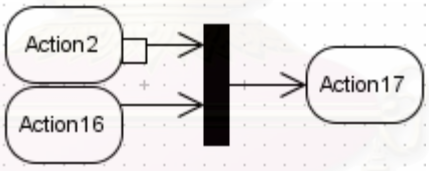
รูปที่ 3.20 โหนด Join ที่มีมากกว่า 1 เส้นเชื่อมขาออก

3) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากเส้นเชื่อมขาเข้าหรือขาออกจากโหนด Fork ไม่เป็นชนิดเดียวกันทั้งหมด ซึ่งอาจเป็น Control Flow หรือ Object Flow ดังรูปที่ 3.21



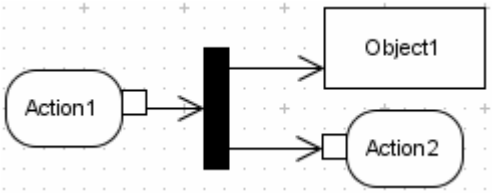
รูปที่ 3.21 โหนด Fork ที่มีเส้นเชื่อมขาออกไม่เป็นชนิดเดียวกันทั้งหมด

4) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด Join มีเส้นเชื่อมขาเข้าที่เป็น Object Flow แต่ไม่มีเส้นเชื่อมขาออกที่เป็น Object Flow ดังรูปที่ 3.22



รูปที่ 3.22 โหนด Join ที่มีเพียงเส้นเชื่อมขาเข้าเป็น Object Flow

5) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด Fork หรือโหนด Join มีเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออกที่เป็น Object Flow โดยวัตถุที่ถูกส่งผ่านนั้นไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใดๆ ดังรูปที่ 3.23 โหนด Fork มีการส่งผ่านวัตถุ Object1 ที่ไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใด



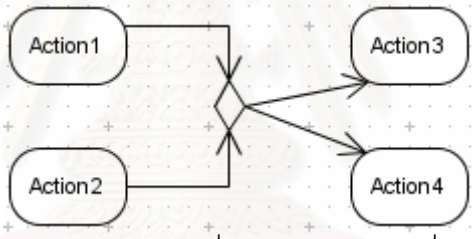
รูปที่ 3.23 โหนด Fork มีวัตถุที่ถูกส่งผ่านที่ไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใด

3.1.9 โหนด Merge / โหนด Decision

โหนด Merge เป็นการแสดงลักษณะการทำงานที่รวมหลายกระแสการทำงานเป็นหนึ่งกระแส แต่ต่างจากโหนด Join ตรงที่งานในกระแสที่ออกจากโหนด Merge สามารถเริ่มทำได้ทันทีหากกระแสการทำงานที่เข้าไปยังโหนด Merge เพียงกระแสเดียวถูกทำเสร็จสิ้น ทั้งนี้หากกระแสการทำงานที่เข้าไปยังโหนด Merge อีกกระแสหนึ่งถูกทำเสร็จภายหลัง งานในกระแสที่ออกจากโหนด Merge ก็ต้องเริ่มทำใหม่อีกครั้ง ซึ่งถือเป็นอินสแตนซ์ของงานที่ต่างกัน ในขณะที่โหนด Decision จะมีหลายกระแสการทำงานที่ออกจากโหนด Decision นั้น ซึ่งการจะไปทำงานในกระแสใดขึ้นอยู่กับเงื่อนไขที่ระบุไว้สำหรับแต่ละกระแสการทำงาน ว่าเงื่อนไขใดเป็นจริง ซึ่งระบบก็จะแสดงเงื่อนไขต่างๆเหล่านี้ให้ผู้ใช้งานเลือกเพื่อนำไปสู่งานถัดไปได้อย่างถูกต้อง

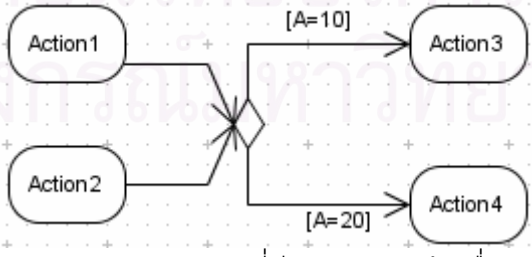
ข้อกำหนดและข้อจำกัดของ โหนด Merge / โหนด Decision มีดังนี้

1) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหาก 1 โหนด Merge มีมากกว่า 1 เส้นเชื่อมขาออก ดังรูปที่ 3.24



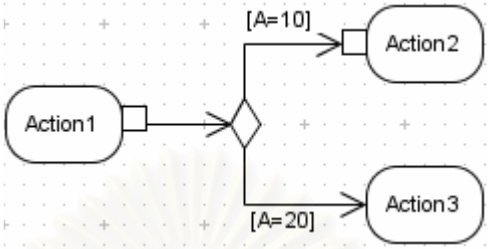
รูปที่ 3.24 โหนด Merge ที่มีมากกว่า 1 เส้นเชื่อมขาออก

2) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหาก 1 โหนด Decision มีมากกว่า 1 เส้นเชื่อมขาเข้า ดังรูปที่ 3.25



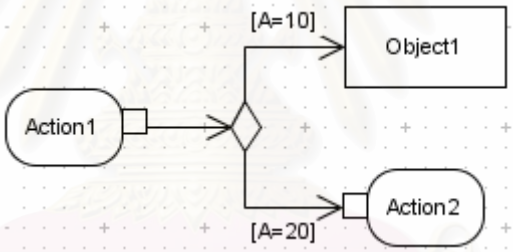
รูปที่ 3.25 โหนด Decision ที่มีมากกว่า 1 เส้นเชื่อมขาเข้า

3) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากเส้นเชื่อมขาเข้าและเส้นเชื่อมขาออกของโหนด Merge หรือโหนด Decision ไม่เป็นชนิดเดียวกันทั้งหมด ซึ่งอาจเป็น Control Flow หรือ Object Flow ดังรูปที่ 3.26



รูปที่ 3.26 โหนด Decision ที่มีเส้นเชื่อมขาออกไม่เป็นชนิดเดียวกันทั้งหมด

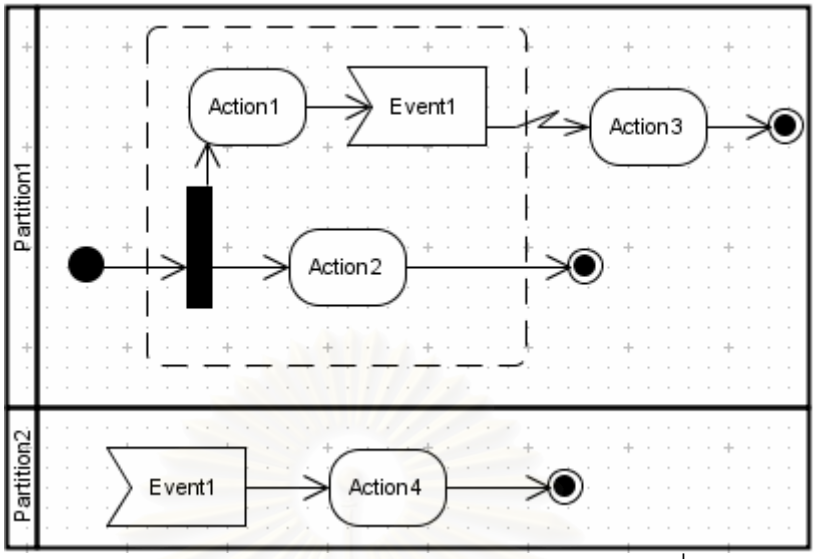
4) ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากโหนด Merge หรือโหนด Decision มีเส้นเชื่อมขาเข้าและเส้นเชื่อมขาออกที่เป็น Object Flow โดยที่วัตถุนั้นไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใด ดังรูปที่ 3.27 โหนด Merge มีการส่งผ่านวัตถุ Object1 ที่ไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใด



รูปที่ 3.27 โหนด Decisoin มีวัตถุที่ถูกส่งผ่านที่ไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใด

3.1.10 โหนด InterruptibleActivityRegion / เส้นเชื่อม ExceptionHandler

โหนด InterruptibleActivityRegion เมื่อใช้คู่กับเส้นเชื่อม ExceptionHandler จะใช้แสดงกลุ่มของงานที่จะถูกยกเลิกเมื่อมีเหตุการณ์ที่ระบุไว้เกิดขึ้น แล้วงานถัดไปก็คืองานที่เป็นปลายทางของเส้นเชื่อม ExceptionHandler นั้น ดังตัวอย่างในรูปที่ 3.28 หากเหตุการณ์ Event1 เกิดขึ้น งาน Action1 และ Action2 ที่ดำเนินการอยู่จะถูกยกเลิกโดยทันทีเพื่อไปทำงาน Action3 ทั้งนี้สถานะของงาน Action1 และ Action2 จะถูกเปลี่ยนเป็น “Interrupted” อย่างไรก็ตามหากโหนด InterruptibleActivityRegion ไม่มีเส้นเชื่อม ExceptionHandler เชื่อมต่อจากงานภายในออกไปยังโหนดที่อยู่ภายนอกโหนด InterruptibleActivityRegion นั้น จะถือว่าไม่มีความหมาย



รูปที่ 3.28 ตัวอย่างการใช้ โหนด InterruptibleActivityRegion และเส้นเชื่อม ExceptionHandler

สัญกรณ์ทั้งหมดที่ได้กล่าวถึงเป็นสัญกรณ์ของแผนภาพกิจกรรมของยูเอ็มแอลที่ระบบรองรับ ทั้งนี้สัญกรณ์อื่นๆของแผนภาพกิจกรรมที่ระบบไม่รองรับ มีรายละเอียดดังนี้

- โหนด AcceptTimeEventAction ซึ่งใช้แสดงเหตุการณ์ที่เกิดขึ้นเมื่อถึงเวลาที่กำหนด ทั้งนี้เนื่องจากระบบไม่มีความรู้ในการวิเคราะห์ข้อความที่ใช้ระบุเวลาของโหนด AcceptTimeEventAction ตัวอย่างเช่นระบบจะไม่รู้ความหมายของการระบุ “Once a year” ว่ามีความหมายถึงการให้ทำงานปีละครั้ง จึงทำให้ระบบไม่สามารถรองรับสัญกรณ์นี้ได้

- โหนด CentralBuffer และโหนด DataStore ซึ่งใช้แสดงลักษณะของการเก็บเอกสารไว้จนกระทั่งถึงเวลาที่ต้องการ เนื่องจากกระแสงานที่ระบบการจัดการกระแสงานนี้ทำการควบคุมเป็นลักษณะที่การทำงานต่อเนื่องกัน กล่าวคือเมื่อทำงานหนึ่งเสร็จ งานต่อไปจะเกิดขึ้นทันที ระบบจึงไม่รองรับทั้ง 2 สัญกรณ์นี้

- โหนด ExpansionRegion ซึ่งเป็นสัญกรณ์ที่ใช้แสดงกลุ่มของการกระทำที่ถูกทำหลายครั้งตามจำนวนอินพุต อย่างไรก็ตามจำนวนอินพุตของโหนด ExpansionRegion นี้จะไม่ได้ถูกกำหนดจำนวนแน่นอนมา กล่าวคือจากแผนภาพกิจกรรมจะไม่สามารถบอกได้ว่าจำนวนอินพุตเป็นเท่าใด และต้องทำงานที่อยู่ในโหนด ExpansionRegion นั้นกี่ครั้ง ระบบที่พัฒนาขึ้นจึงไม่สามารถรองรับการทำงานในลักษณะนี้ได้ เนื่องจากกระแสงานที่เป็นอินพุตของระบบจำเป็นต้องมีเงื่อนไขการทำงานที่แน่นอน

3.2 ข้อกำหนดและข้อจำกัดทั่วไปของระบบ

นอกจากรายละเอียดของการตีความแต่ละสัญกรณ์ของแผนภาพกิจกรรมแล้ว ยังมีข้อกำหนดและข้อจำกัดทั่วไปของระบบเกี่ยวกับแผนภาพกิจกรรมที่เป็นอินพุต ดังรายละเอียดต่อไปนี้

1. ระบบจะรองรับเพียงสัญกรณ์ที่กล่าวไปในส่วนที่แล้วเท่านั้น โดยระบบจะถือว่าแผนภาพกิจกรรมที่ใส่เข้ามาไม่ถูกต้องหากในแผนภาพกิจกรรมมีการใช้สัญกรณ์ที่ระบบไม่รองรับ ซึ่งได้แก่สัญกรณ์ AcceptTimeEventAction, DataStore, CentralBuffer และ ExpansionRegion รวมถึงระบบจะไม่รองรับแผนภาพย่อย (Sub-diagram) และสเตอริโอไทพ์

2. ระบบจะทำการตรวจสอบไฟล์เอ็กซ์เอ็มไอที่เป็นอินพุตว่าถูกต้องตามรูปแบบที่เอ็กซ์พอร์ตมาจากเครื่องมือ Visual Paradigm for UML หรือไม่ และจะถือว่าแผนภาพกิจกรรมนั้นไม่ถูกต้องหากข้อมูลในไฟล์เอ็กซ์เอ็มไอไม่อยู่ในรูปแบบที่กำหนด

3. เมื่อผู้ใช้ทำการวาดแผนภาพกิจกรรมด้วยเครื่องมือ Visual Paradigm for UML ผู้ใช้จำเป็นต้องจัดเก็บข้อมูลแผนภาพกิจกรรมนั้นก่อนทำการเอ็กซ์พอร์ตเป็นไฟล์เอ็กซ์เอ็มไอ โดยชื่อโปรเจคท์ให้ใส่ชื่อของกระแสงานที่ต้องการ

4. ระบบจะไม่ยอมรับแผนภาพกิจกรรมใหม่ที่ใส่เข้ามาหากในระบบมีแผนภาพกิจกรรมที่ใช้ชื่อนั้นอยู่แล้ว กล่าวคือในระบบจะไม่มีมากกว่า 1 กระแสงานที่มีชื่อเดียวกัน

5. ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากในแผนภาพกิจกรรมมีโหนดใดๆที่ไม่อยู่ในโหนด ActivityPartition ทั้งนี้ข้อกำหนดนี้มีขึ้นเพื่อที่ระบบจะสามารถรู้บทบาทของผู้ที่มีหน้าที่ในการทำงานทุกงานในกระแสงาน

6. ระบบจะถือว่าแผนภาพกิจกรรมไม่ถูกต้องหากแผนภาพกิจกรรมไม่มีโหนด Initial หรือโหนด ActivityFinal ทั้งนี้ข้อกำหนดนี้มีขึ้นเพื่อที่ระบบจะสามารถรู้จุดเริ่มต้นและจุดสิ้นสุดของกระแสงาน

นอกจากนี้การตั้งค่าอื่นๆด้วยเครื่องมือ Visual Paradigm for UML ที่นอกเหนือจากการตั้งค่าของแต่ละสัญกรณ์ที่ได้กล่าวมาแล้วจะไม่ถูกตีความโดยระบบ ตัวอย่างการตั้งค่านี้ เช่น คอมเมนต์ของโหนด เป็นต้น

3.3 การแก้ไขกระแสงาน

ระบบการจัดการกระแสงานที่ดีควรจะสามารถรองรับการแก้ไขกระแสงานในระหว่างที่มีอินสแตนซ์ของกระแสงานนั้นดำเนินการอยู่ได้ เพื่อที่อินสแตนซ์นั้นสามารถดำเนินการ

ต่อไปได้บนข้อกำหนดใหม่ของกระแสงาน อย่างไรก็ตามการแก้ไขนั้นต้องไม่กระทบกับงานที่ดำเนินการเสร็จสิ้นไปแล้วหรือดำเนินการอยู่ ในส่วนนี้เป็นการให้รายละเอียดของข้อกำหนดและเงื่อนไขในการแก้ไขกระแสงานของระบบการจัดการกระแสงานที่ถูกพัฒนาขึ้น

โดยทั่วไปแล้วการแก้ไขกระแสงานจะเป็นการเพิ่มงาน ลดงาน หรือเปลี่ยนแปลงลำดับการทำงานในกระแสงาน รวมถึงอาจเปลี่ยนคุณสมบัติของงาน เช่น เปลี่ยนบทบาทของผู้รับผิดชอบงานนั้น เป็นต้น ทั้งนี้การแก้ไขกระแสงานที่ไม่ส่งผลกระทบกับงานที่ดำเนินการเสร็จสิ้นไปแล้วหรือยังดำเนินการอยู่จึงเป็นการเปลี่ยนที่ไม่ทำให้ลำดับการทำงานนั้นผิดไป และไม่เป็นการเปลี่ยนคุณสมบัติใดๆของงานนั้นที่จะทำให้เกิดความขัดแย้งของการทำงานขึ้น อย่างไรก็ดีงานต่างๆที่การแก้ไขอาจส่งผลกระทบจะเป็นงานของอินสแตนซ์ที่ยังดำเนินการอยู่เท่านั้น สำหรับอินสแตนซ์ที่ดำเนินการเสร็จสิ้นไปแล้ว ระบบสามารถแก้ไขความขัดแย้งโดยการเปลี่ยนเลขเวอร์ชันของกระแสงานทุกครั้งที่มีการแก้ไขกระแสงานนั้น โดยระบบจะบันทึกข้อมูลของเวอร์ชันกระแสงานที่อินสแตนซ์ทำงานตามไปเป็นคุณสมบัติหนึ่งของอินสแตนซ์ด้วย ผู้ใช้งานจึงสามารถดูได้ภายหลังว่าแต่ละอินสแตนซ์ทำงานคู่กับเวอร์ชันใดของกระแสงาน อย่างไรก็ตามระบบจะไม่เก็บข้อมูลของกระแสงานเวอร์ชันเก่าไว้ กล่าวคือผู้ใช้ไม่สามารถเรียกดูข้อกำหนดของกระแสงานเวอร์ชันก่อนหน้าที่ถูกแก้ไขได้

กล่าวโดยสรุป การแก้ไขกระแสงานจะทำได้สำเร็จเมื่อเป็นไปตามข้อกำหนดดังต่อไปนี้

1. การแก้ไขกระแสงานจะต้องเป็นการแก้ไขโดยเครื่องมือ Visual Paradigm for UML และแก้ไขจากแผนภาพกิจกรรมเดิม หลังจากนั้นต้องเอ็กซ์พอร์ตเป็นไฟล์เอ็กซ์เอ็มไอออกมา แล้วใส่เป็นอินพุตให้กับระบบโดยระบุว่าเป็นการแก้ไขกระแสงานใด ทั้งนี้การเขียนแผนภาพกิจกรรมขึ้นมาใหม่ทั้งหมดแล้วระบุว่าเป็นการแก้ไขกระแสงานจะไม่สามารถทำได้ เนื่องจากระบบจะตรวจสอบคุณสมบัติของงานว่าถูกแก้ไขหรือไม่โดยใช้รหัสของโหนดที่ระบุมาในไฟล์เอ็กซ์เอ็มไอเป็นหมายเลขอ้างอิงระหว่างโหนดในกระแสงานเก่าและโหนดในกระแสงานใหม่ การแก้ไขแผนภาพกิจกรรมเดิมจะทำให้รหัสโหนดยังคงเป็นค่าเดิม (ยกเว้นโหนดที่ถูกเพิ่มเข้ามาใหม่) ทำให้สามารถตรวจสอบการแก้ไขได้ แต่หากเขียนแผนภาพกิจกรรมขึ้นมาใหม่ ทุกโหนดจะมีรหัสใหม่ ทำให้ไม่สามารถตรวจสอบความถูกต้องของการแก้ไขได้

2. แผนภาพกิจกรรมที่ถูกแก้ไขมาต้องถูกต้องตามข้อกำหนดต่างๆของระบบ มิฉะนั้นระบบจะไม่ทำการตรวจสอบความถูกต้องของการแก้ไขต่อไป

3. ชื่อกระแสงานในไฟล์เอ็กซ์เอ็มไอที่ใส่เป็นอินพุตเข้ามาใหม่จะต้องเป็นชื่อเดียวกับกระแสงานที่ระบุว่าถูกแก้ไข

4. จำนวนโหนด Initial ต้องเท่ากันในกระแสงานทั้งก่อนและหลังแก้ไข ทั้งนี้เนื่องจากการเพิ่มโหนด Initial เข้ามาใหม่เป็นการแสดงว่างานที่ต่อกับโหนด Initial ใหม่นั้นจะต้องถูกสร้างขึ้นทันทีที่อินสแตนซ์เกิดขึ้น ซึ่งไม่เป็นความจริง

5. ในการตรวจสอบความถูกต้อง หากกระแสงานที่ถูกแก้ไขไม่มีอินสแตนซ์ที่ดำเนินการอยู่ ระบบจะถือว่าการแก้ไขนั้นสำเร็จทันที แต่หากมีอินสแตนซ์ที่ดำเนินการอยู่ ระบบจะทำการตรวจสอบข้อมูลในกระแสงานใหม่ เทียบกับทุกอินสแตนซ์ที่ดำเนินการอยู่ดังนี้

ก. ทุกงานของอินสแตนซ์เหล่านั้นที่เกิดขึ้นมาแล้วจะต้องยังอยู่ในกระแสงานหลังการแก้ไข (โหนด Action หรือ Activity ที่ใช้แทนงานนั้นยังคงอยู่หลังการแก้ไข) และจะต้องไม่มีคุณสมบัติใดที่เปลี่ยนไป

ข. ทุกงานจะต้องถูกเชื่อมต่อมาจากโหนดเดิมเหมือนในกระแสงานก่อนถูกแก้ไข รวมถึงอินพุตของงานเหล่านั้นจะต้องไม่เปลี่ยนแปลง

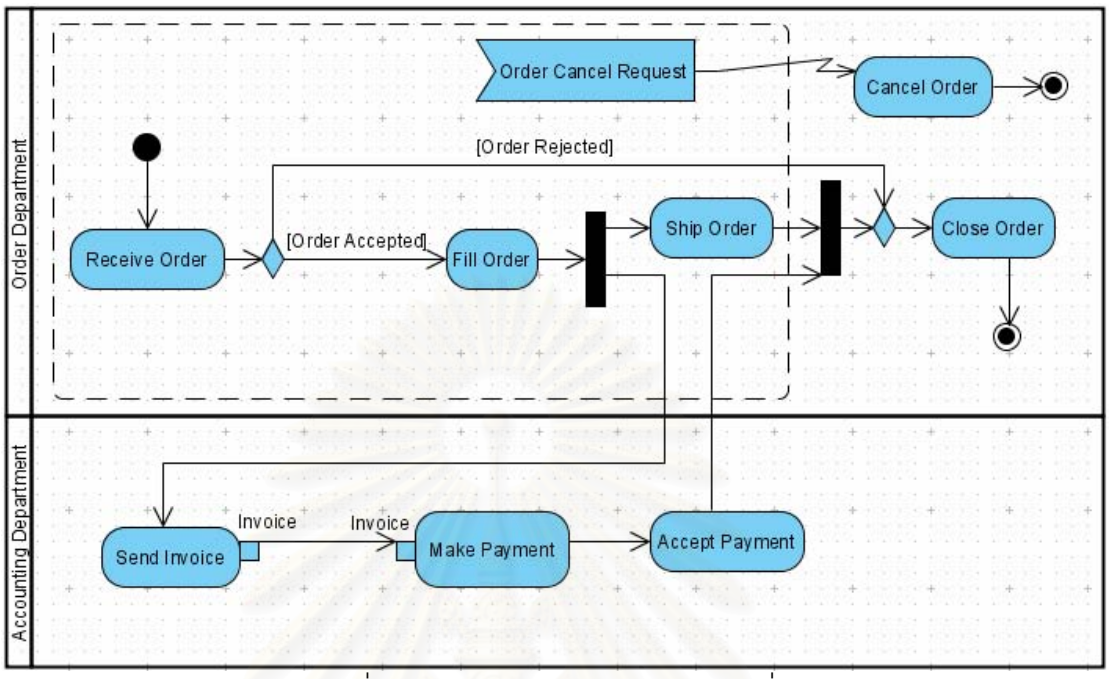
ค. สำหรับงานที่ทำเสร็จสิ้นไปแล้ว ทุกงานเหล่านั้นจะต้องเชื่อมต่อไปยังโหนดเดิมเหมือนในกระแสงานก่อนถูกแก้ไข รวมถึงเอาท์พุตของงานเหล่านั้นจะต้องไม่เปลี่ยนแปลง

ง. หากโหนดก่อนหน้างานใดเป็นโหนด Decision เส้นเชื่อมขาเข้ามายังงานเหล่านั้นต้องมีเงื่อนไขที่เหมือนเดิม ทั้งนี้เพื่อไม่ให้เกิดความขัดแย้งในกรณีที่เงื่อนไขถูกเปลี่ยนไป ซึ่งอาจตีความได้ว่าเงื่อนไขเดิมที่ทำให้เกิดงานเหล่านี้ขึ้นอาจไม่เป็นจริงอีกต่อไป

จ. สำหรับโหนด Fork ที่งานในกระแสหนึ่งที่ออกจากโหนด Fork นั้นเกิดขึ้นแล้ว จะต้องไม่มีเส้นเชื่อมขาออกของ โหนด Fork นั้นเพิ่มเข้ามา

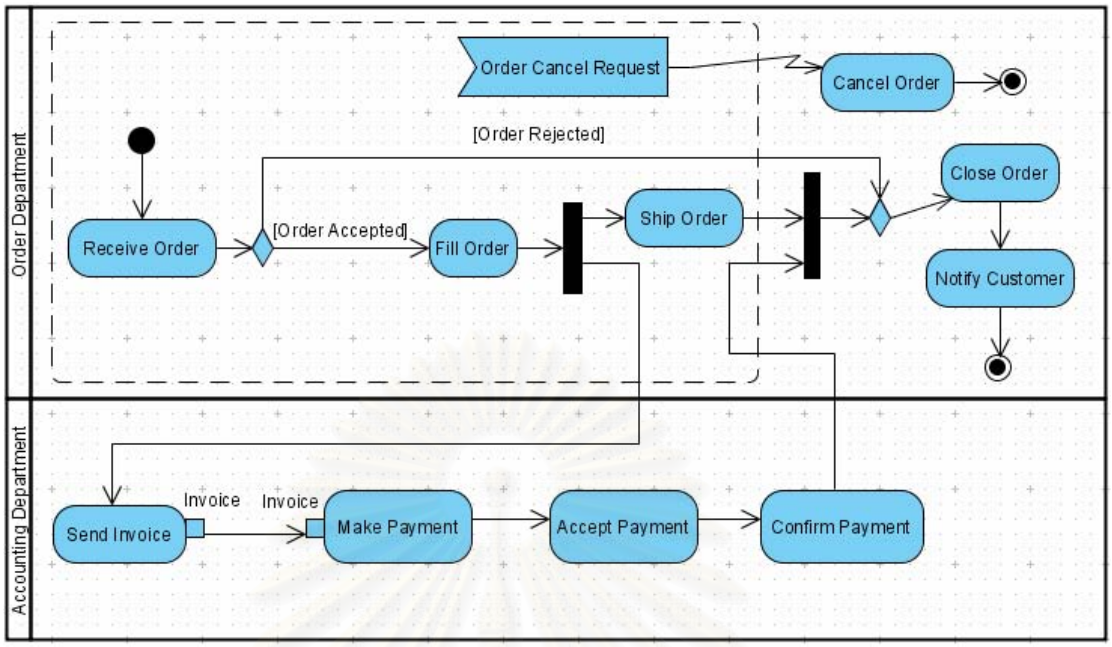
ฉ. โหนด AcceptEventAction และ โหนด SendSignalAction ต้องไม่ถูกลบออกหรือถูกแก้ไข รวมถึงหากมีการเพิ่มโหนดชนิดดังกล่าวเข้ามา จะต้องมีชื่อที่แตกต่างจากโหนดเดิมที่มีอยู่แล้ว ทั้งนี้เพื่อไม่ให้เกิดความขัดแย้งในกรณีที่เหตุการณ์บางเหตุการณ์เกิดขึ้นแล้ว การที่มีเหตุการณ์นั้นเพิ่มเข้ามาแล้วเชื่อมต่อไปยังโหนดอื่นด้วยกระแสใหม่ ผลลัพธ์คือเหตุการณ์นี้เปรียบเสมือนยังไม่เกิดขึ้นในกระแสที่เพิ่มเข้ามาใหม่นี้ ซึ่งไม่ถูกต้อง

ดังตัวอย่างในรูป 3.29 เป็นกระแสงานการจัดการกับคำสั่ง ซึ่งหาก ณ ขณะหนึ่งมีอินสแตนซ์หนึ่งดำเนินการอยู่ และงานที่เพิ่งทำเสร็จสิ้นไปคืองาน Fill Order ซึ่งทำให้เกิดงาน Ship Order ในรายการงานของผู้ใช้ที่มีบทบาทเป็น Order Department และงาน Send Invoice ในรายการงานของผู้ใช้ที่มีบทบาทเป็น Accounting Department

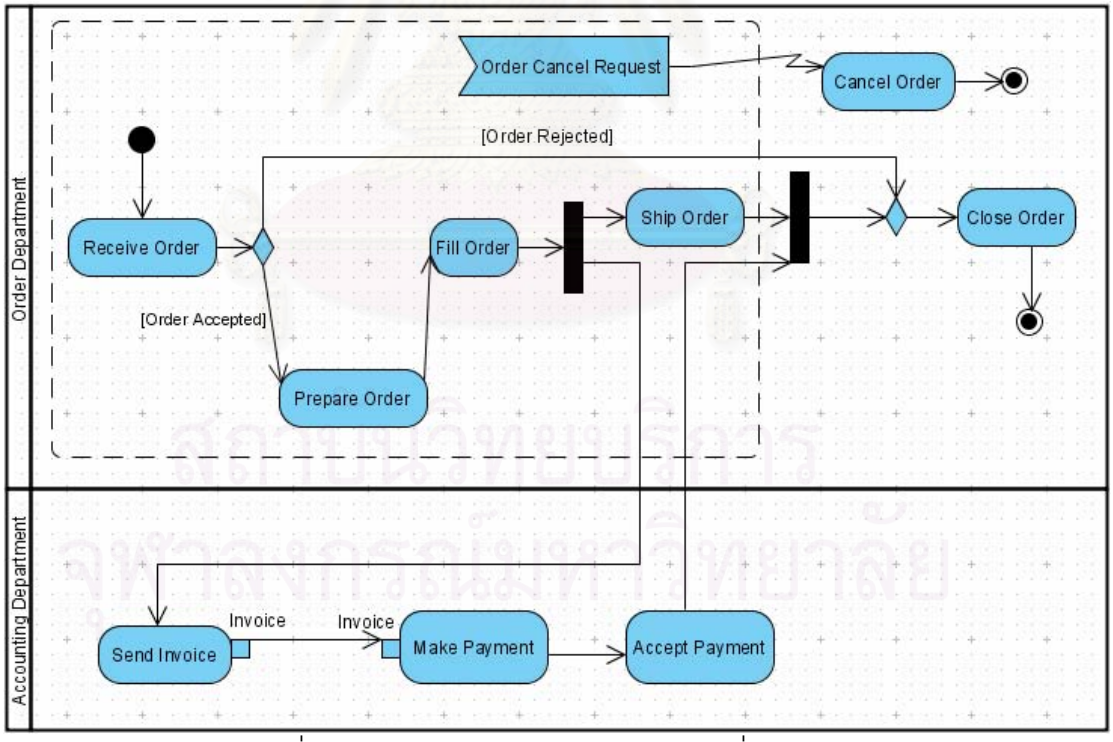


รูปที่ 3.29 กระแสงานการจัดการกับคำสั่ง

เมื่อมีการทำการแก้ไขกระแสงาน ณ เวลาดังกล่าว รูปที่ 3.30 แสดงการแก้ไขที่ถูกต้อง เป็นการเพิ่มโหนด Action ชื่อ Confirm Payment ให้ทำต่อจากโหนด Accept Payment และเพิ่มโหนด Action ชื่อ Notify Customer ให้ทำต่อจากงาน Close Order ซึ่งการเพิ่มทั้งสองงานนี้ไม่กระทบกับงานที่ทำเสร็จสิ้นไปแล้ว ในขณะที่รูปที่ 3.31 แสดงการแก้ไขที่ไม่ถูกต้อง เนื่องจากเป็นการเพิ่มโหนด Action ชื่อ Prepare Order เข้ามาก่อนงาน Fill Order ซึ่งได้ทำเสร็จสิ้นไปแล้ว



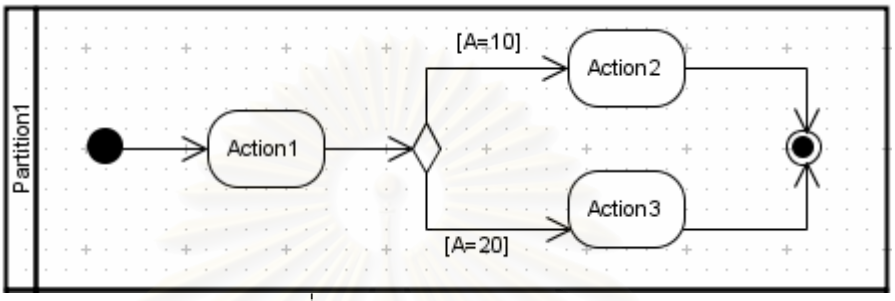
รูปที่ 3.30 การแก้ไขกระแสงานการจัดการที่ถูกต้อง



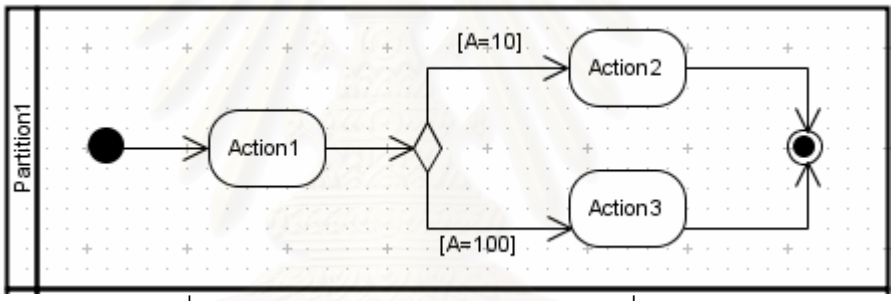
รูปที่ 3.31 การแก้ไขกระแสงานการจัดการที่ไม่ถูกต้อง

รูปที่ 3.32 และรูปที่ 3.33 แสดงตัวอย่างการแก้ไขกระแสงานที่ผิดตามเงื่อนไข ง จากรูปที่ 3.32 เป็นกระแสงานตัวอย่าง 1 ซึ่งอินสแตนซ์ที่ถูกดำเนินการอยู่มีงานในปัจจุบันคืองาน

Action3 ซึ่งมีความหมายว่าหลังจากงาน Action1 ถูกทำเสร็จ เงื่อนไขที่เป็นจริงคือ A=20 จึงทำให้มีงาน Action3 เป็นงานลำดับถัดไป เมื่อมีการแก้ไขในลักษณะดังรูป 3.33 ซึ่งเปลี่ยนเงื่อนไขของเส้นเชื่อมขาเข้ามายังงาน Action3 เป็น A=100 จึงทำให้เกิดความขัดแย้งขึ้น ดังนั้นการแก้ไขในลักษณะนี้จึงถือว่าไม่ถูกต้อง

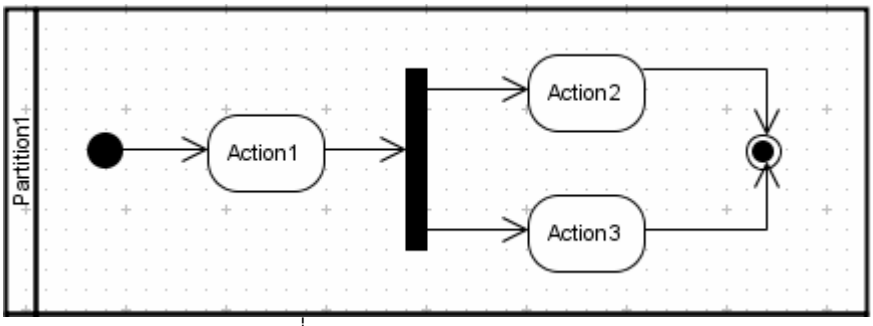


รูปที่ 3.32 กระแสงานตัวอย่าง 1

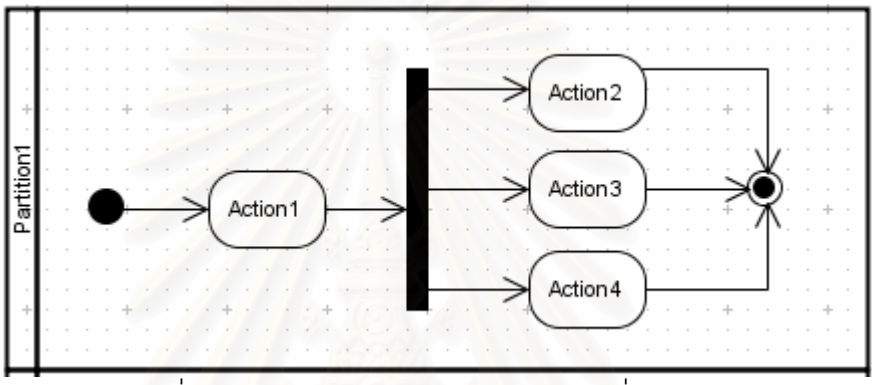


รูปที่ 3.33 การแก้ไขกระแสงานตัวอย่าง 1 ที่ไม่ถูกต้อง

นอกจากนี้รูปที่ 3.34 และรูปที่ 3.35 เป็นตัวอย่างการแก้ไขกระแสงานที่ผิดตามเงื่อนไข จ จากรูปที่ 3.34 เป็นกระแสงานตัวอย่าง 2 ซึ่งอินสแตนซ์ที่ถูกดำเนินการอยู่มีงานในปัจจุบันคืองาน Action2 และงาน Action3 การแก้ไขในรูปที่ 3.35 เป็นการเพิ่มงานกระแสใหม่ที่ออกจากโหนด Fork (งาน Action4) ซึ่งถือว่าไม่ถูกต้อง เนื่องจากเมื่อพิจารณาข้อกำหนดของกระแสงานใหม่ จะเห็นว่าในปัจจุบันงาน Action4 จะต้องถูกดำเนินการอยู่ด้วยเช่นเดียวกับงาน Action2 และงาน Action3 ตามลักษณะของโหนด Fork อย่างไรก็ตามงาน Action4 แท้จริงแล้วเป็นงานที่เพิ่งถูกเพิ่มเข้ามา ทำให้มีข้อขัดแย้งเกิดขึ้น การแก้ไขนี้จึงถือว่าไม่ถูกต้อง



รูปที่ 3.34 กระแสงานตัวอย่าง 2

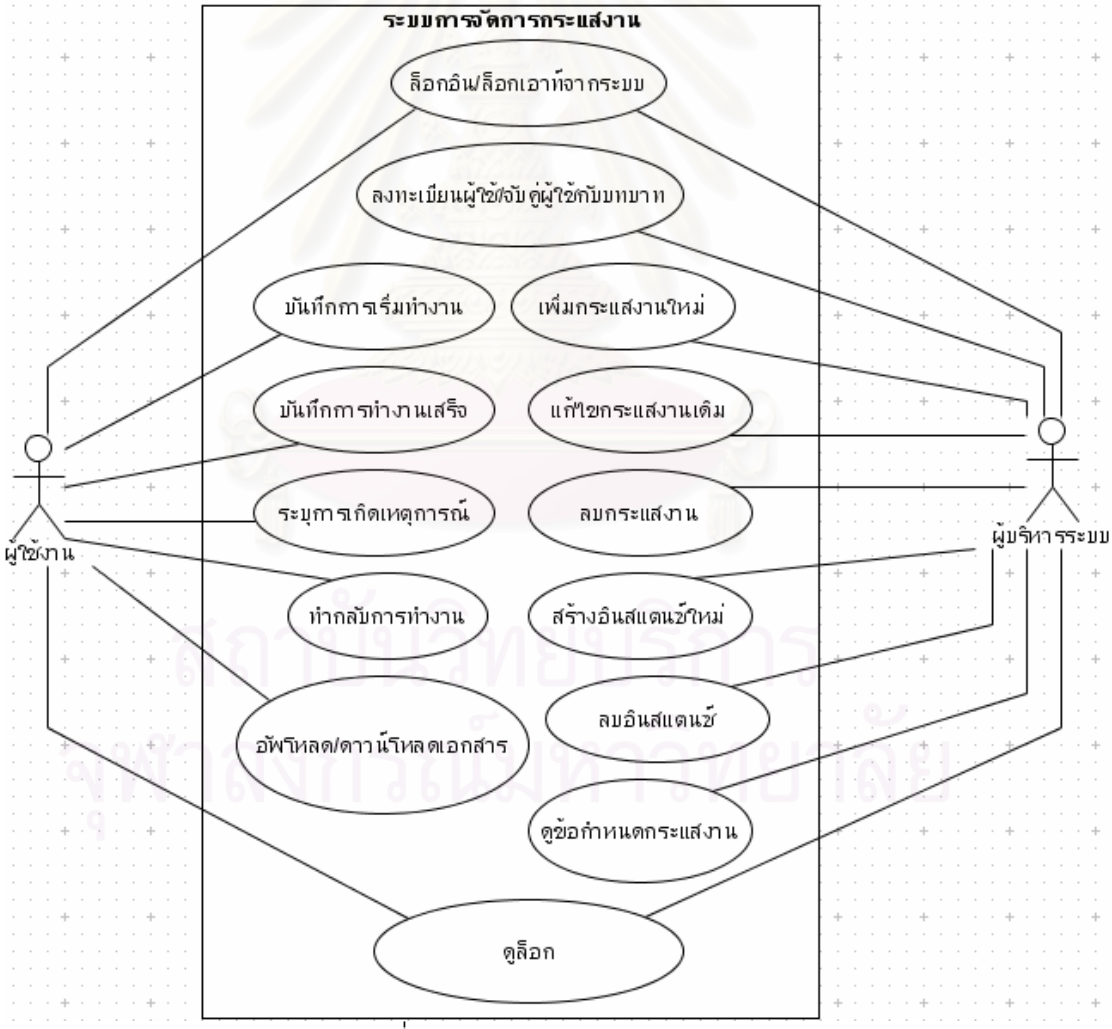


รูปที่ 3.35 การแก้ไขกระแสงานตัวอย่าง 2 ที่ไม่ถูกต้อง

บทที่ 4
การออกแบบและพัฒนาระบบ

4.1 การวิเคราะห์ความต้องการของระบบ

การวิเคราะห์ความต้องการของระบบการจัดการกระแสงานนี้ส่วนหนึ่งจะได้มาจากการวิเคราะห์ความสามารถโดยทั่วไปของระบบการจัดการกระแสงานดังที่ได้กล่าวในบทที่ 2 ทั้งนี้การวิเคราะห์ความต้องการของระบบที่พัฒนาขึ้นสามารถอธิบายได้ด้วยแผนภาพยูสเคส (Use Case Diagram) ในมุมมองของผู้ใช้งานและผู้บริหารระบบ ดังแสดงในรูปที่ 4.1

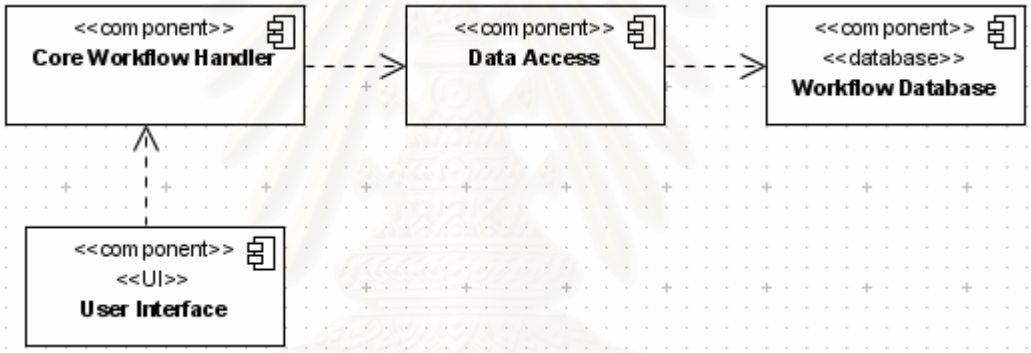


รูปที่ 4.1 แผนภาพยูสเคสของระบบ

จากรูปที่ 4.1 แสดงแผนภาพยูสเคสของระบบ ซึ่งผู้ใช้สามารถแบ่งได้เป็นผู้บริหารระบบและผู้ใช้งานปกติซึ่งมีสิทธิในการทำงานไม่เท่ากัน ทั้งนี้รายละเอียดประกอบยูสเคสทั้งหมดแสดงในภาคผนวก ค

4.2 สถาปัตยกรรมของระบบ

ระบบการจัดการกระแสงานนี้สามารถแบ่งคอมโพเนนท์ตามหน้าที่ในการทำงานได้เป็น 4 ส่วน ได้แก่แกนหลักของระบบ (Core Workflow Handler), ส่วนการเข้าถึงข้อมูล (Data Access), ฐานข้อมูล (Database) และส่วนต่อประสานกับผู้ใช้ (User Interface) ดังแสดงในรูปที่ 4.2



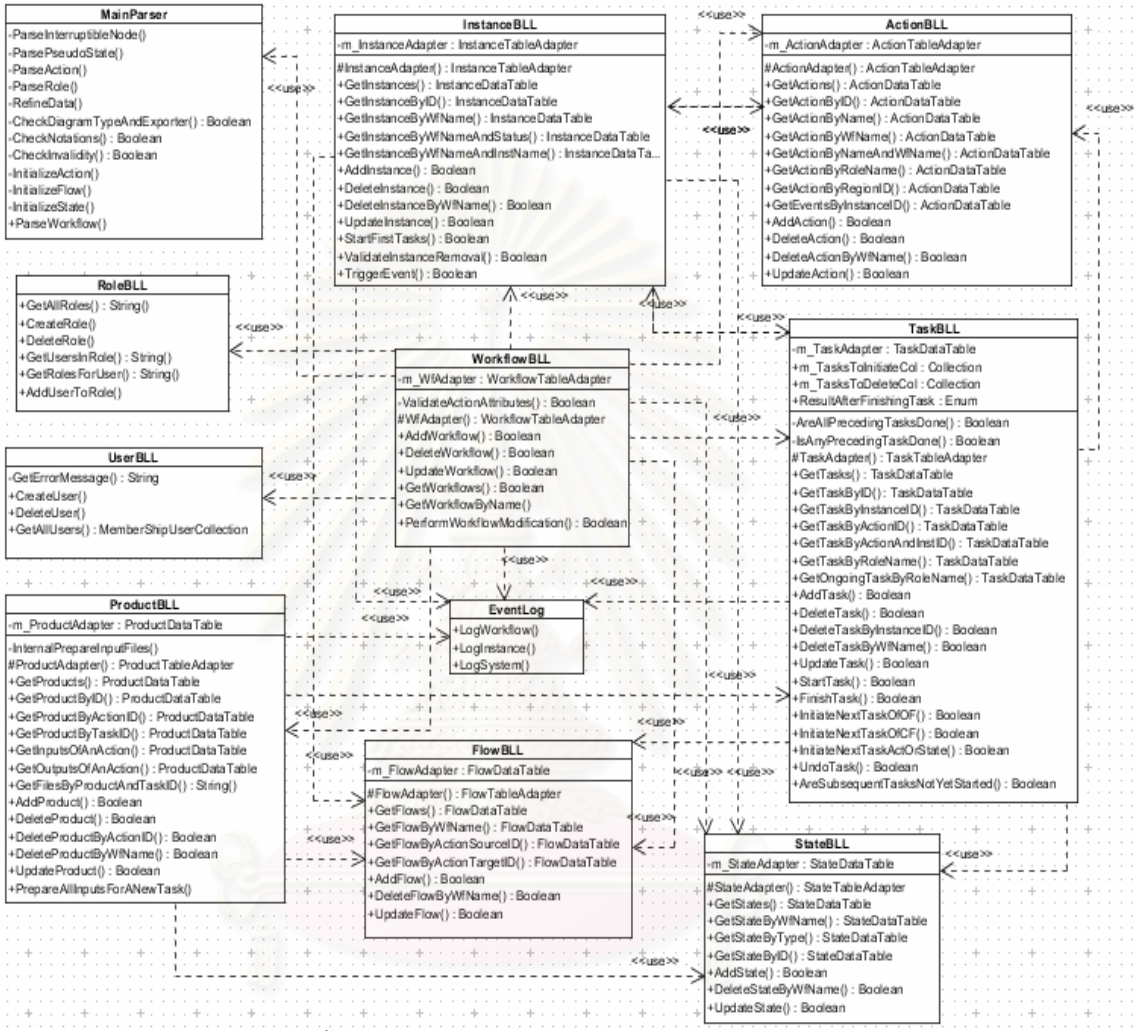
รูปที่ 4.2 แผนภาพคอมโพเนนท์แสดงสถาปัตยกรรมของระบบ

การแบ่งเช่นนี้มีข้อดีคือสามารถแยกตรรกะของการเข้าถึงข้อมูลในฐานข้อมูล และตรรกะของส่วนการแสดงผลทางหน้าจอให้เป็นอิสระกับส่วนประมวลผลหลัก ทำให้การแก้ไขและพัฒนาต่อเป็นไปได้ง่าย โดยในแต่ละส่วนมีรายละเอียดดังต่อไปนี้

4.2.1 แกนหลักของระบบ

คอมโพเนนท์นี้เป็นส่วนหลักของระบบ ซึ่งเปรียบเสมือนส่วนประมวลผลของระบบในการทำงานที่เกี่ยวข้องกับการจัดการกระแสงาน ภายในแบ่งเป็นหลายโมดูลย่อยซึ่งจะถูกเรียกใช้โดยส่วนต่อประสานกับผู้ใช้ และติดต่อกับฐานข้อมูลผ่านทางคอมโพเนนท์ส่วนการเข้าถึงข้อมูลเพื่อนำข้อมูลไปประมวลผล จากนั้นส่งผลลัพธ์ที่ได้กลับไปยังส่วนต่อประสานกับผู้ใช้เพื่อแสดงให้ผู้ใช้งานเห็นต่อไป รวมถึงอาจมีการแก้ไขข้อมูลกลับไปยังฐานข้อมูล ทั้งนี้โมดูลย่อยใน

แกนหลักของระบบมีลักษณะเป็นคลาสที่แทนเอ็นติตี (Entity) ต่างๆในกระแสงานเพื่อประมวลผลเฉพาะส่วนที่เกี่ยวข้องดังรูปที่ 4.3 ซึ่งมีรายละเอียดต่อไปนี้



รูปที่ 4.3 แผนภาพคลาสของแกนหลักของระบบ

1) คลาส WorkflowBLL มีหน้าที่ในการประมวลผลเกี่ยวกับกระแสงาน เช่นการเพิ่มกระแสงานเข้าไปในระบบโดยตรวจสอบว่ามีกระแสงานที่ชื่อซ้ำกันอยู่ในฐานข้อมูลหรือไม่ หรือการแก้ไขกระแสงานโดยตรวจสอบว่าการแก้ไขนั้นถูกต้องหรือไม่ เป็นต้น

2) คลาส InstanceBLL มีหน้าที่ในการประมวลผลเกี่ยวกับอินสแตนซ์ของกระแสงาน เช่นการเพิ่มอินสแตนซ์ใหม่ของกระแสงานหนึ่งโดยตรวจสอบว่ามีอินสแตนซ์ของกระแสงานนั้นที่ชื่อซ้ำกันหรือไม่ หรือการระบุการเกิดเหตุการณ์ในอินสแตนซ์นั้นเพื่อนำไปสู่การทำงานที่ถูกต้อง เป็นต้น

3) คลาส ActionBLL มีหน้าที่ในการประมวลผลเกี่ยวกับการกระทำที่ได้มาจากการวิเคราะห์แผนภาพกิจกรรมที่ถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ ทั้งนี้การกระทำหมายความรวมถึงโหนด Action และ โหนด Activity ของแผนภาพกิจกรรมซึ่งใช้แทน “งาน” ในระบบการจัดการกระแสงานนี้

4) คลาส StateBLL มีหน้าที่ในการประมวลผลเกี่ยวกับโหนดควบคุม (Control Node) ที่ได้มาจากการวิเคราะห์แผนภาพกิจกรรมที่ถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ ทั้งนี้โหนดควบคุมเป็นโหนดที่ใช้แสดงลักษณะกระแสของการทำงานแบบต่างๆ เช่น โหนด Decision หรือโหนด Fork เป็นต้น

5) คลาส FlowBLL มีหน้าที่ในการประมวลผลเกี่ยวกับกระแสที่ได้มาจากการวิเคราะห์แผนภาพกิจกรรมที่ถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ โดยคลาสนี้มีความสำคัญในการจัดลำดับการทำงานที่ถูกต้อง

6) คลาส ProductBLL มีหน้าที่ในการประมวลผลเกี่ยวกับอินพุตและเอาท์พุตของโหนด Action หรือโหนด Activity ที่ได้มาจากการวิเคราะห์แผนภาพกิจกรรมที่ถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ โดยในแผนภาพกิจกรรมโหนดที่ใช้แทนอินพุตและเอาท์พุตได้แก่ โหนด Input Pin, โหนด Output Pin,โหนด Activity Parameter, โหนด Object

7) คลาส TaskBLL เป็นคลาสหลักที่มีหน้าที่ในการประมวลผลเกี่ยวกับการทำงานของผู้ใช้งาน ตัวอย่างเช่นการบันทึกการเริ่มทำงาน และการบันทึกการทำงานเสร็จสิ้น ที่ต้องมีการตรวจสอบเงื่อนไขต่างๆเช่นอินพุตและเอาท์พุตถูกอัพโหลดไว้ที่ระบบเรียบร้อยแล้ว ระบบจึงจะอนุญาตให้บันทึกการทำงานเสร็จ รวมถึงตรวจสอบเงื่อนไขในกรณีผู้ใช้งานขอทำกลับการทำงานที่ได้ทำเสร็จสิ้นแล้ว ซึ่งจะอนุญาตก็ต่อเมื่องานในลำดับถัดไปยังไม่ถูกบันทึกเริ่มการทำงาน เป็นต้น

8) คลาส UserBLL มีหน้าที่ในการประมวลผลเกี่ยวกับการลงทะเบียนผู้ใช้งาน

9) คลาส RoleBLL มีหน้าที่ในการประมวลผลเกี่ยวกับการจับคู่ผู้ใช้งานกับบทบาทตามที่มีข้อมูลในฐานข้อมูล

10) คลาส MainParser มีหน้าที่ในการวิเคราะห์ไฟล์ในรูปแบบมาตรฐานเอ็กซ์เอ็มไอที่ผู้บริหารระบบใส่เข้ามา คลาสนี้จะตรวจสอบแท็กต่างๆของเอ็กซ์เอ็มไอนั้นว่าถูกต้องตามมาตรฐาน รวมถึงตรวจสอบความถูกต้องของข้อมูลแผนภาพกิจกรรม เมื่อการวิเคราะห์เสร็จสิ้นหากแผนภาพกิจกรรมถูกต้อง คลาสนี้จะทำการจัดเก็บข้อมูลของกระแสงานนั้นลงในฐานข้อมูลผ่านทางส่วนการเข้าถึงข้อมูล หรือหากมีความไม่ถูกต้องจะส่งข้อมูลไปยังส่วนต่อประสานกับผู้ใช้เพื่อแจ้งให้ผู้บริหารระบบทราบถึงความไม่ถูกต้องนั้นต่อไป

11) คลาส EventLog มีหน้าที่ในการบันทึกข้อมูลการทำงานต่างๆ สำหรับผู้บริหารระบบในการมาวิเคราะห์การทำงานภายหลัง ตัวอย่างข้อมูลที่บันทึกเช่น วันและเวลาที่อินสแตนซ์ใหม่เกิดขึ้น วันและเวลารวมถึงชื่อผู้ใช้ที่บันทึกการเริ่มทำงานและการทำงานเสร็จสิ้นเป็นต้น

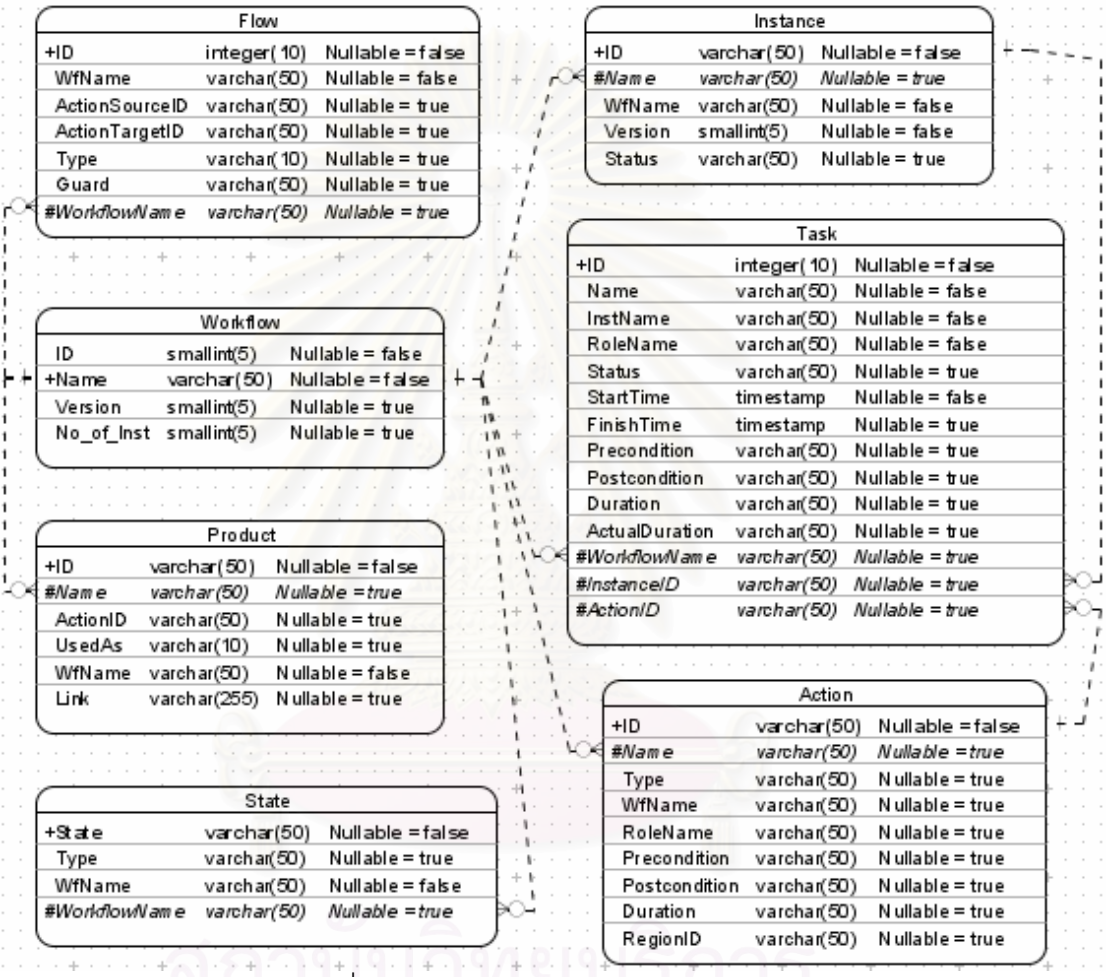
4.2.2 ส่วนการเข้าถึงข้อมูล

คอมโพเนนท์นี้ทำหน้าที่เข้าถึงข้อมูลในฐานข้อมูลและเตรียมข้อมูลเพื่อการนำไปใช้โดยแกนหลักของระบบ ส่วนการเข้าถึงข้อมูลนี้ถูกออกแบบมาเพื่อให้แกนหลักของระบบไม่ต้องมีกระบวนการในการเข้าถึงข้อมูลในฐานข้อมูลโดยตรง ในระบบการจัดการกระแสงานนี้ส่วนการเข้าถึงข้อมูลจะคิวรีข้อมูลโดยผ่านทางคลาส TableAdapter ที่ภาษา ASP.NET ซึ่งเป็นภาษาที่ใช้ในการพัฒนาระบบการจัดการกระแสงานนี้จัดเตรียมไว้ให้ [21] โดยการเข้าถึงข้อมูลวิธีนี้จะได้ออบเจ็คท์ชนิด Dataset จากฐานข้อมูลมา ซึ่งมีลักษณะเป็นการเก็บข้อมูลจากฐานข้อมูลทั้งหมดไว้ในหน่วยความจำ และข้อมูลของแต่ละตารางจะตรงตามเค้าร่างที่ระบุไว้ในฐานข้อมูล (Database Schema) จึงทำให้ง่ายต่อการอ่านและแก้ไขข้อมูล โดยในคอมโพเนนท์นี้มี 7 TableAdapter ได้แก่ WorkflowTableAdapter, InstanceTableAdapter, ActionTableAdapter, StateTableAdapter, FlowTableAdapter, ProductTableAdapter, TaskTableAdapter สำหรับการเข้าถึงข้อมูลในแต่ละตารางในฐานข้อมูล

4.2.3 ฐานข้อมูล

ฐานข้อมูลของระบบการจัดการกระแสงานนี้ประกอบด้วย 7 ตาราง ซึ่งมีความสัมพันธ์ดังแผนภาพความสัมพันธ์ของเอ็นติตีในรูปที่ 4.4 ทั้งนี้นอกจาก 7 ตารางนี้แล้ว ยังมีอีก 2 ตารางคือ User และ Role ที่ถูกสร้างให้โดยอัตโนมัติสำหรับเว็บแอพพลิเคชันที่ถูกสร้างด้วยภาษา ASP.NET ซึ่งเป็นภาษาที่ใช้ในการพัฒนาระบบการจัดการกระแสงานนี้

เมื่อกระแสงานใหม่ถูกใส่เข้ามาในระบบ ข้อมูลของกระแสงานนั้นจะถูกใส่ไปในตาราง Workflow, ตาราง Action, ตาราง Flow, ตาราง Product, ตาราง State จากนั้นเมื่อผู้บริหารระบบสร้างอินสแตนซ์ใหม่ของกระแสงานนี้ซึ่งหมายความว่างานแรกเกิดขึ้นทันที ข้อมูลจะถูกใส่ไปในตาราง Instance และตาราง Task



รูปที่ 4.4 แผนภาพความสัมพันธ์ของเอ็นติตี

สำหรับรายละเอียดของแต่ละฟิลด์ในแต่ละตารางเป็นดังตารางที่ 4.1 ถึงตารางที่ 4.7

ตารางที่ 4.1 รายละเอียดของตาราง Workflow

ฟิลด์	ความหมาย
ID	รหัสกระแสงาน
Name	ชื่อกระแสงาน ซึ่งใช้เป็นตัวระบุกระแสงานในฐานข้อมูล ทั้งนี้ระบบไม่อนุญาตให้มีหลายกระแสงานใช้ชื่อเดียวกัน
Version	เวอร์ชันของกระแสงาน ค่าเริ่มต้นคือ 1 และจะถูกเพิ่มขึ้นเมื่อมีการแก้ไขกระแสงาน
No_of_Inst	จำนวนอินสแตนซ์ของกระแสงาน

ตารางที่ 4.2 รายละเอียดของตาราง Action

ฟิลด์	ความหมาย
ID	รหัสการกระทำ ใช้เป็นตัวระบุการกระทำในฐานข้อมูล รหัสการกระทำนี้ได้จากการนำรหัสของโหนด Action หรือโหนด Activity ที่ระบุอยู่ในไฟล์เอ็กซ์เอ็มไอ มาต่อกับชื่อของกระแสงาน เพื่อให้เป็นหมายเลขที่ไม่ซ้ำกันสำหรับแต่ละการกระทำ
Type	ชนิดของการกระทำ ซึ่งมีอยู่ 4 ชนิด ขึ้นอยู่กับว่าการกระทำนี้ในแผนภาพกิจกรรมถูกสร้างมาด้วยโหนดชนิดใด ดังนี้ ถ้าถูกสร้างมาด้วยโหนด Action ชนิดจะเป็น “Action” ถ้าถูกสร้างมาด้วยโหนด Activity ชนิดจะเป็น “Activity” ถ้าถูกสร้างมาด้วยโหนด AcceptEventAction ชนิดจะเป็น “Event” ถ้าถูกสร้างมาด้วยโหนด SendSignalAction ชนิดจะเป็น “SignalEvent”
WfName	ชื่อกระแสงานของการกระทำ
RoleName	ชื่อบทบาทที่มีหน้าที่ทำการกระทำนี้ ตามที่ระบุมาในแผนภาพกิจกรรม
Precondition	เงื่อนไขก่อนหน้าของการกระทำ ซึ่งสามารถระบุได้ผ่านเครื่องมือ Visual Paradigm for UML สำหรับโหนด Activity เท่านั้น
Postcondition	เงื่อนไขภายหลังของการกระทำ ซึ่งสามารถระบุได้ผ่านเครื่องมือ Visual Paradigm for UML สำหรับโหนด Activity เท่านั้น
Duration	ระยะเวลาทำงานที่วางแผนไว้ หน่วยเป็นวัน ตามที่ระบุมาในแผนภาพกิจกรรม
RegionID	รหัสของโหนด InterruptibleActivityRegion ที่มีการกระทำนี้อยู่ภายในตามแผนภาพกิจกรรม โดยรหัสนี้ได้มาจากหมายเลขที่ระบุอยู่ในไฟล์เอ็กซ์เอ็มไอ

ตารางที่ 4.3 รายละเอียดของตาราง Flow

ฟิลด์	ความหมาย
ID	รหัสกระแส
WfName	ชื่อกระแสงานของกระแส
ActionSourceID	รหัสของโหนดที่เป็นต้นทางของกระแสนี้
ActionTargetID	รหัสของโหนดที่เป็นปลายทางของกระแสนี้
Guard	เงื่อนไขของกระแส ซึ่งเงื่อนไขนี้จะมีค่าเมื่อเป็นกระแสที่โหนดต้นทางคือโหนด Decision ทั้งนี้เงื่อนไขจะถูกนำไปใช้เมื่อการทำงานดำเนินมาถึงโหนด Decision นั้นและระบบจะแสดงเงื่อนไขให้ผู้ใช้งานเลือกว่าเงื่อนไขใดเป็นจริง เพื่อนำไปสู่งานถัดไปตามกระแสที่ถูกเลือก

ตารางที่ 4.4 รายละเอียดของตาราง Product

ฟิลด์	ความหมาย
ID	รหัสโพรดักส์ใช้เป็นตัวระบุโพรดักส์ในฐานข้อมูล รหัสนี้ได้จากการนำรหัสของโหนด Input Pin, Output Pin, โหนด ActivityParameter หรือโหนด Object ที่ระบุอยู่ในไฟล์เอ็กซ์เอ็มไอมาต่อกับชื่อของโหนด Action หรือโหนด Activity ที่เป็นเจ้าของโพรดักส์นั้น เพื่อให้เป็นรหัสที่ไม่ซ้ำกันสำหรับแต่ละโพรดักส์
ActionID	รหัสของโหนด Action หรือ Activity ที่เป็นเจ้าของโพรดักส์นี้
UsedAs	ใช้ระบุว่าโพรดักส์เป็นอินพุตหรือเอาท์พุต โดยมี 2 ค่าที่เป็นไปได้คือ “Input” และ “Output”
WfName	ชื่อกระแสงาน
Link	พาธ (Path) บนเว็บเซิร์ฟเวอร์ไปยังแฟ้มที่เก็บไฟล์ของโพรดักส์นี้

ตารางที่ 4.5 รายละเอียดของตาราง State

ฟิลด์	ความหมาย
ID	รหัสสเตทใช้เป็นตัวระบุสเตทในฐานข้อมูล รหัสสเตทนี้ได้จากการนำรหัสของโหนดที่ระบุอยู่ในไฟล์เอ็กซ์เอ็มไอ มาต่อกับชื่อของกระแสงาน เพื่อให้เป็นรหัสที่ไม่ซ้ำกันสำหรับแต่ละสเตท

ตารางที่ 4.5 รายละเอียดของตาราง State (ต่อ)

ฟิลด์	ความหมาย
Type	ชนิดของสเตท ซึ่งมีอยู่ 6 ชนิด ขึ้นอยู่กับว่าการกระทำนี้ในแผนภาพกิจกรรมถูกสร้างมาด้วยโหนดชนิดใด ดังนี้ ถ้าถูกสร้างมาด้วยโหนด Initial ชนิดจะเป็น “initial” ถ้าถูกสร้างมาด้วยโหนด FlowFinal ชนิดจะเป็น “flowfinal” ถ้าถูกสร้างมาด้วยโหนด ActivityFinal ชนิดจะเป็น “activityfinal” ถ้าถูกสร้างมาด้วยโหนด Fork ชนิดจะเป็น “fork” ถ้าถูกสร้างมาด้วยโหนด Join ชนิดจะเป็น “join” ถ้าถูกสร้างมาด้วยโหนด Decision หรือโหนด Merge ชนิดจะเป็น “junction”
WfName	ชื่อกระแสงาน

ตารางที่ 4.6 รายละเอียดของตาราง Instance

ฟิลด์	ความหมาย
ID	รหัสอินสแตนซ์
Name	ชื่ออินสแตนซ์
WfName	ชื่อกระแสงานของอินสแตนซ์
Version	เวอร์ชันของกระแสงานของอินสแตนซ์
Status	สถานะของอินสแตนซ์ซึ่งได้แก่ “In Progress” และ “Done”

ตารางที่ 4.7 รายละเอียดของตาราง Task

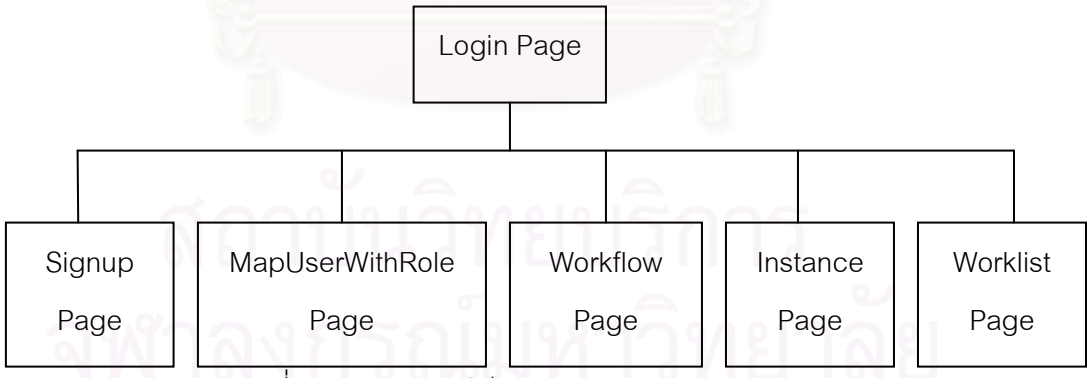
ฟิลด์	ความหมาย
ID	รหัสของงาน
Name	ชื่อของงาน ซึ่งเป็นชื่อเดียวกับการกระทำที่เป็นต้นแบบของงาน
ActionID	รหัสของการกระทำที่เป็นแบบของงาน
WfName	ชื่อกระแสงานของงาน
InstID	รหัสอินสแตนซ์ของงาน
InstName	ชื่ออินสแตนซ์ของงาน
RoleName	ชื่อบทบาทที่มีหน้าที่รับผิดชอบทำงาน

ตารางที่ 4.7 รายละเอียดของตาราง Task (ต่อ)

ฟิลด์	ความหมาย
Status	สถานะของงาน ซึ่งได้แก่ “New”, “In Progress”, “Done”, “Awaiting”, “Cancelled” และ “Interrupted”
StartTime	เวลาที่งานถูกเริ่มทำ
FinishTime	เวลาที่งานถูกทำเสร็จสิ้น
Precondition	เงื่อนไขก่อนหน้าของงาน ซึ่งได้มาจากการกระทำที่เป็นต้นแบบของงาน
Postcondition	เงื่อนไขภายหลังของงาน ซึ่งได้มาจากการกระทำที่เป็นต้นแบบของงาน
Duration	ระยะเวลาทำงานที่วางแผนไว้ หน่วยเป็นวัน ซึ่งได้มาจากการกระทำที่เป็นต้นแบบของงาน
ActualDuration	ระยะเวลาการทำงานจริง หน่วยเป็นวัน

4.2.4 ส่วนต่อประสานกับผู้ใช้

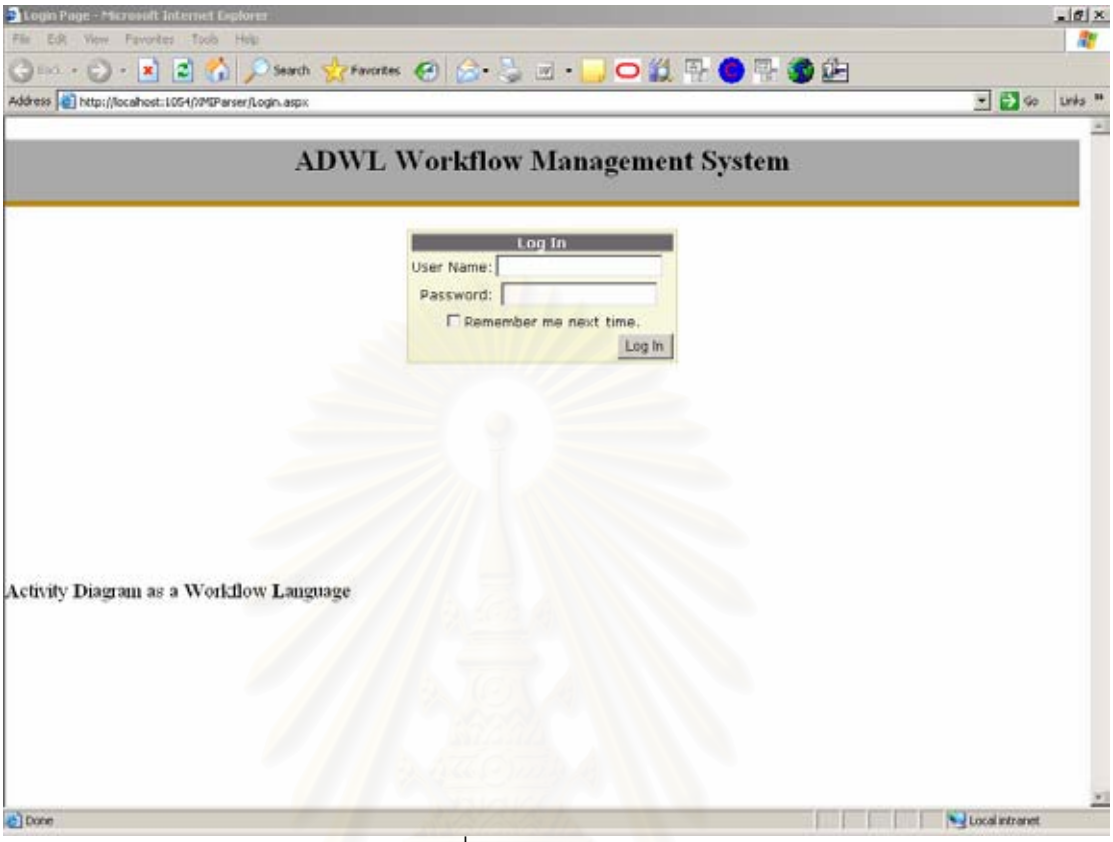
ส่วนต่อประสานกับผู้ใช้ในระบบการจัดการกระแสงานนี้คือหน้าเว็บต่างๆ ที่ผู้ใช้งานและผู้บริหารระบบต้องใช้ในการจัดการกระแสงาน โดยในระบบมีทั้งหมด 6 หน้าเว็บ ดังแสดงในแผนภาพเว็บไซต์ในรูปที่ 4.5



รูปที่ 4.5 แผนภาพเว็บไซต์ระบบการจัดการกระแสงาน

สำหรับรายละเอียดของหน้าเว็บทั้งหมดเป็นดังแสดงในรูปที่ 4.6 – 4.11

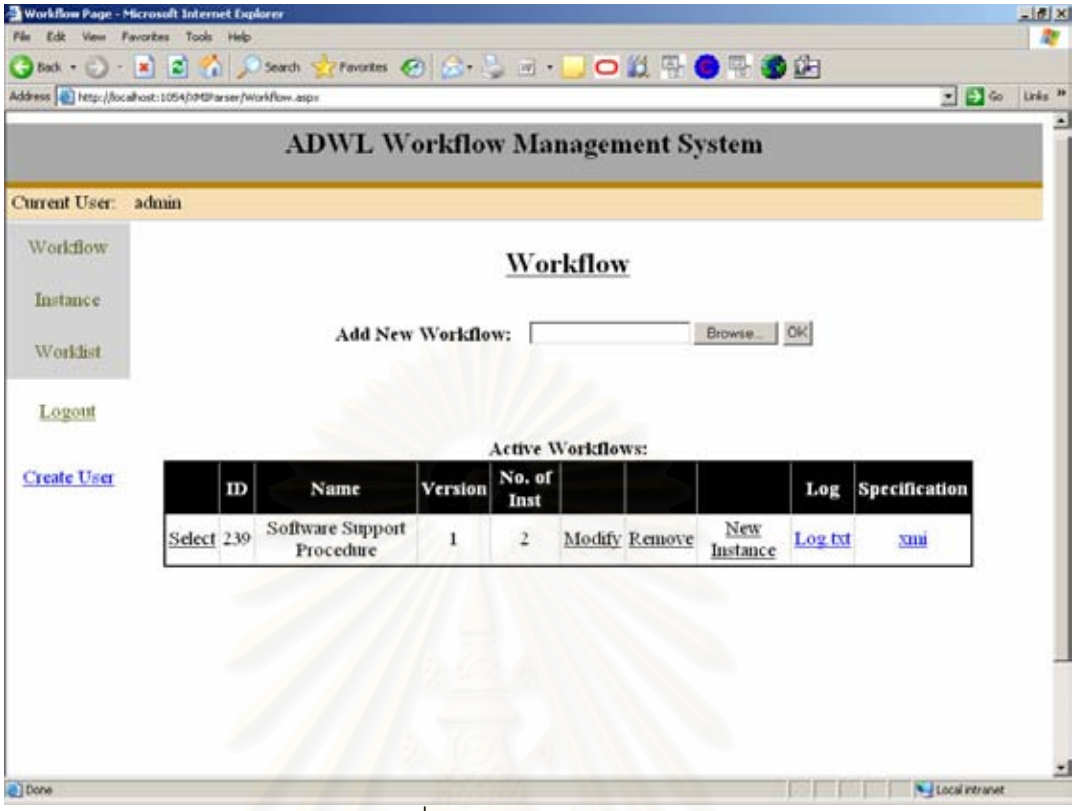
หน้าเว็บ Login เป็นหน้าเว็บสำหรับผู้ใช้งานในการล็อกอินเข้าสู่ระบบ โดยผู้ใช้งานต้องใส่ชื่อและรหัสผ่าน



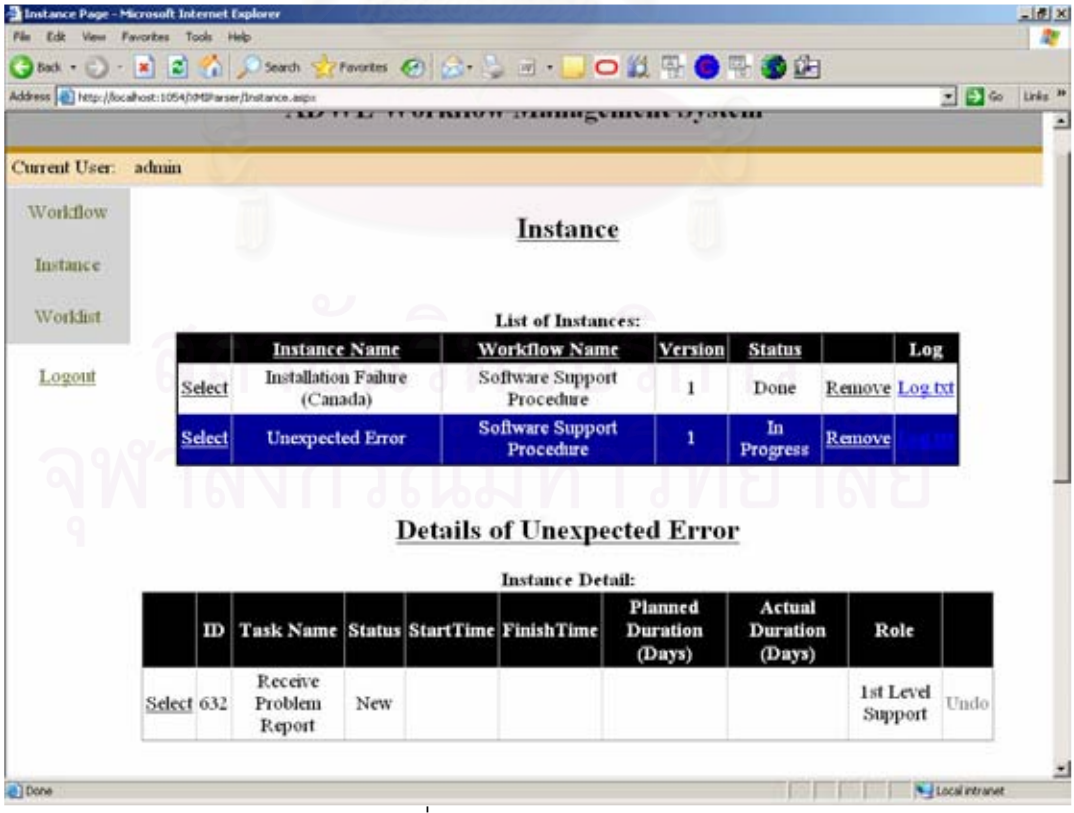
รูปที่ 4.6 หน้าเว็บ Login

เมื่อผู้ใช้งานทำการล็อกอินเข้าสู่ระบบเรียบร้อย ผู้ใช้จะเห็นหน้าเว็บ Workflow ซึ่งแสดงรายละเอียดของกระแสงานทั้งหมดที่ระบบรู้จักในขณะนั้น รวมถึงเลขเวอร์ชันและจำนวนอินสแตนซ์ของแต่ละกระแสงาน ที่หน้าเว็บนี้ผู้บริหารระบบสามารถเพิ่มกระแสงานใหม่เข้าไปในระบบ ลบหรือแก้ไขกระแสงานเดิม รวมถึงสามารถดูล็อกไฟล์และข้อกำหนดของกระแสงานในรูปแบบไฟล์เอ็กซ์เอ็มไอของแต่ละกระแสงานได้

หน้าเว็บ Instance เป็นหน้าที่แสดงอินสแตนซ์ทั้งหมดที่ระบบรู้จัก และสถานะในขณะนั้น รวมถึงแสดงรายละเอียดของการทำงานในแต่ละอินสแตนซ์

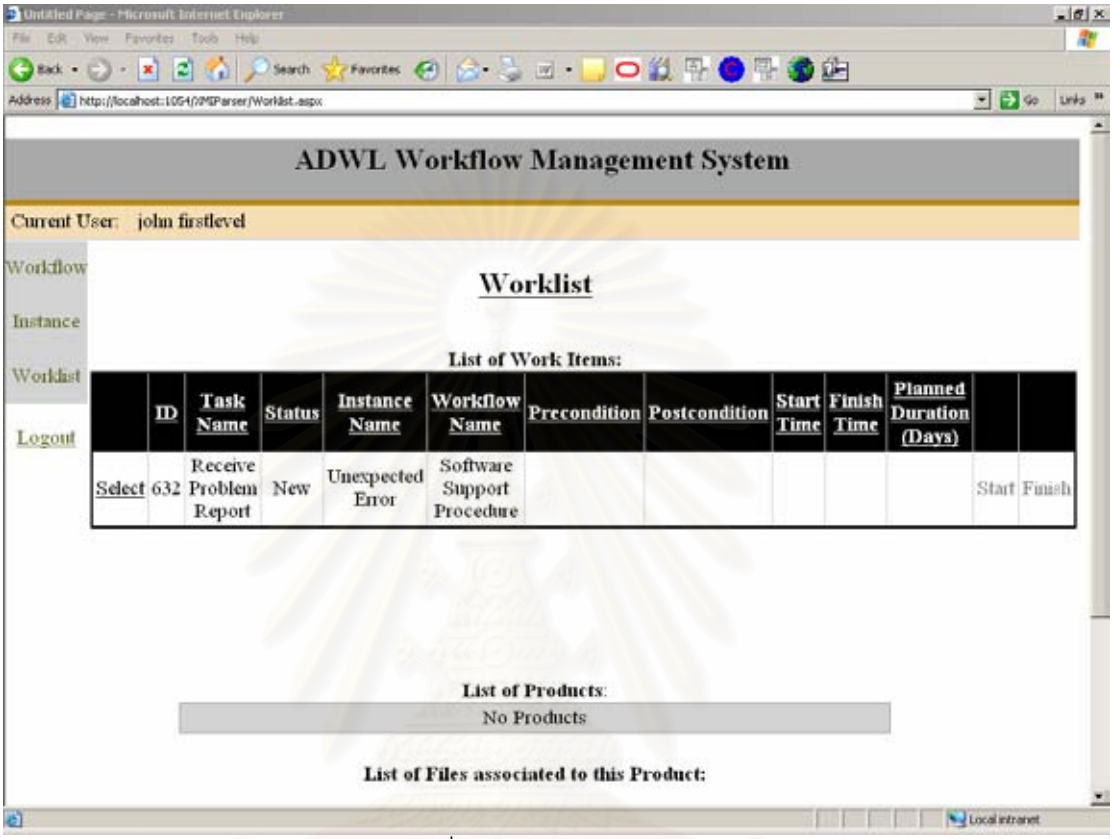


รูปที่ 4.7 หน้าเว็บ Workflow



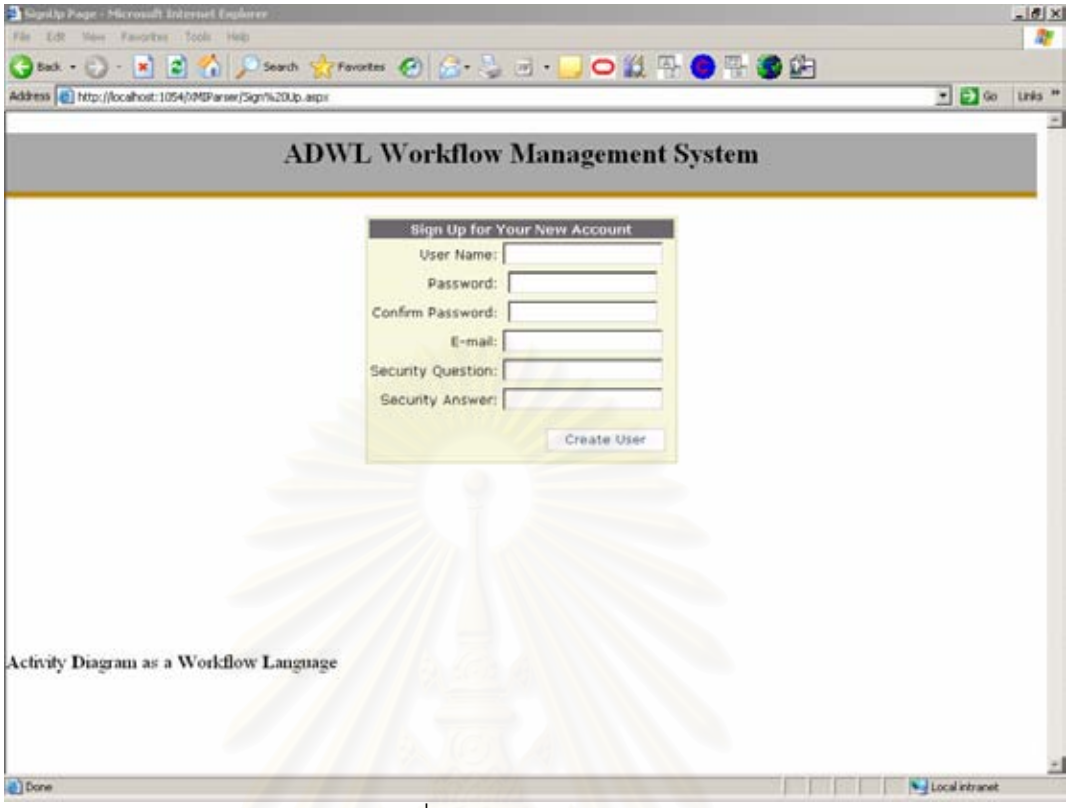
รูปที่ 4.8 หน้าเว็บ Instance

หน้าเว็บ Worklist เป็นหน้าที่แสดงรายการงานของผู้ใช้ที่ล็อกอินอยู่ในขณะนั้น

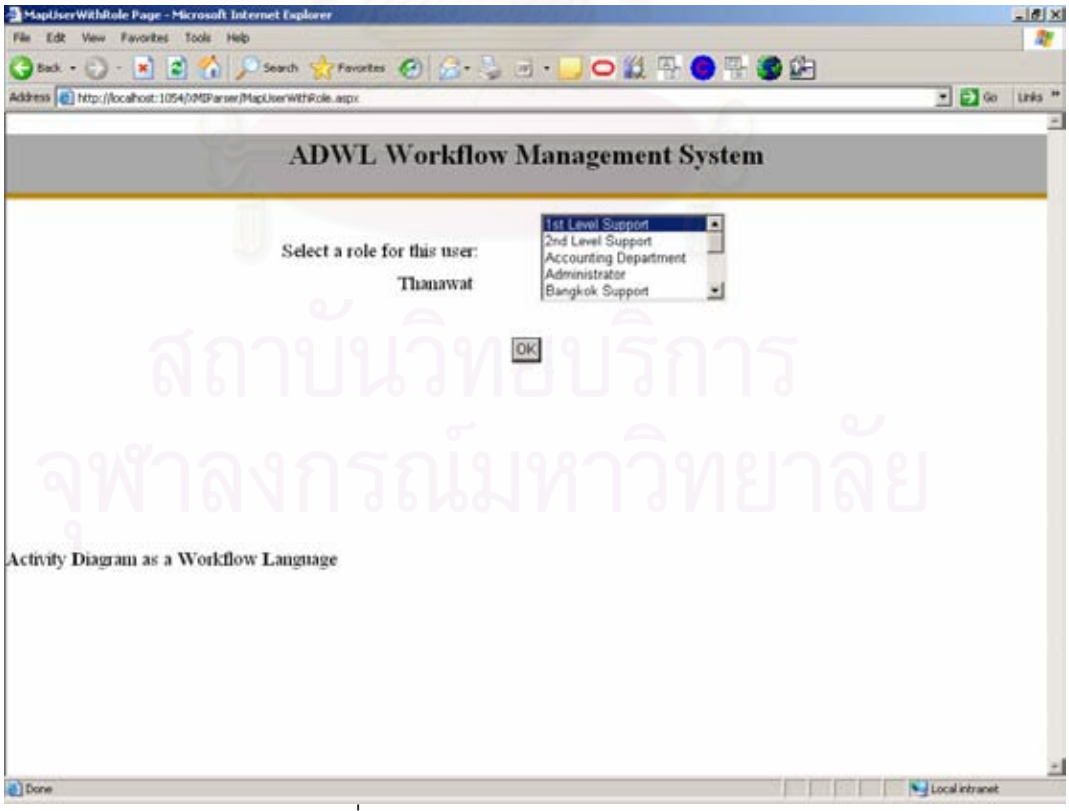


รูปที่ 4.9 หน้าเว็บ Worklist

ทั้งนี้ผู้บริหารระบบสามารถลงทะเบียนผู้ใช้ใหม่กับระบบ รวมถึงจับคู่ผู้ใช้กับบทบาทได้ โดยบทบาทที่มีให้เลือกจับคู่จะได้มาจากบทบาทในทุกกระแสงานที่อยู่ในระบบขณะนั้น



รูปที่ 4.10 หน้าเว็บ Signup



รูปที่ 4.11 หน้าเว็บ MapUserWithRole

บทที่ 5
การทดสอบระบบ

เนื้อหาของบทนี้แสดงการทดสอบระบบการจัดการกระแสงานที่ได้พัฒนาขึ้นจากการทำวิทยานิพนธ์นี้ ผู้วิจัยได้แบ่งการทดสอบออกเป็น 3 ส่วน ได้แก่ การทดสอบโดยใช้กรณีทดสอบ การทดลองใช้ระบบการจัดการกระแสงาน และการประเมินระบบการจัดการกระแสงานโดยใช้แบบรูปของกระแสงาน

5.1 การทดสอบโดยใช้กรณีทดสอบ

การทดสอบโดยใช้กรณีทดสอบแบ่งเป็นการทดสอบการตีความสัญกรณ์ต่างๆ ของแผนภาพกิจกรรมและการแก้ไขข้อกำหนดของกระแสงานตามที่ได้ระบุไว้ในส่วนการวิเคราะห์แผนภาพกิจกรรมในบทที่ 3 ทั้งนี้การทดสอบการตีความแบ่งเป็นกรณีแผนภาพกิจกรรมถูกต้องและกรณีแผนภาพกิจกรรมไม่ถูกต้อง

5.1.1 กรณีแผนภาพกิจกรรมถูกต้อง

กรณีทดสอบในกรณีนี้มีรายละเอียดดังในตารางที่ 5.1 ซึ่งแยกตามชนิดของสัญกรณ์

ตารางที่ 5.1 กรณีทดสอบกรณีแผนภาพกิจกรรมถูกต้อง

กรณี	ผลลัพธ์
1. โหนด Initial	
1.1 งานที่ติดกับ Initial Node ถูกสร้างขึ้นทันทีที่อินสแตนซ์เกิดขึ้น	ถูกต้อง
2. โหนด Action / โหนด Activity / Control Flow / Object Flow	
2.1 เมื่องานหนึ่งทำเสร็จ งานในลำดับถัดไปตาม Control Flow ถูกสร้างอย่างถูกต้อง	ถูกต้อง
2.2 เมื่องานหนึ่งทำเสร็จ งานในลำดับถัดไปตาม Object Flow ถูกสร้างอย่างถูกต้อง	ถูกต้อง
3. โหนด FlowFinal	
3.1 กระแสการทำงานนั้นจะสิ้นสุดเมื่อเจอโหนด FlowFinal	ถูกต้อง
4. โหนด ActivityFinal	

ตารางที่ 5.1 กรณีทดสอบกรณีแผนภาพกิจกรรมถูกต้อง (ต่อ)

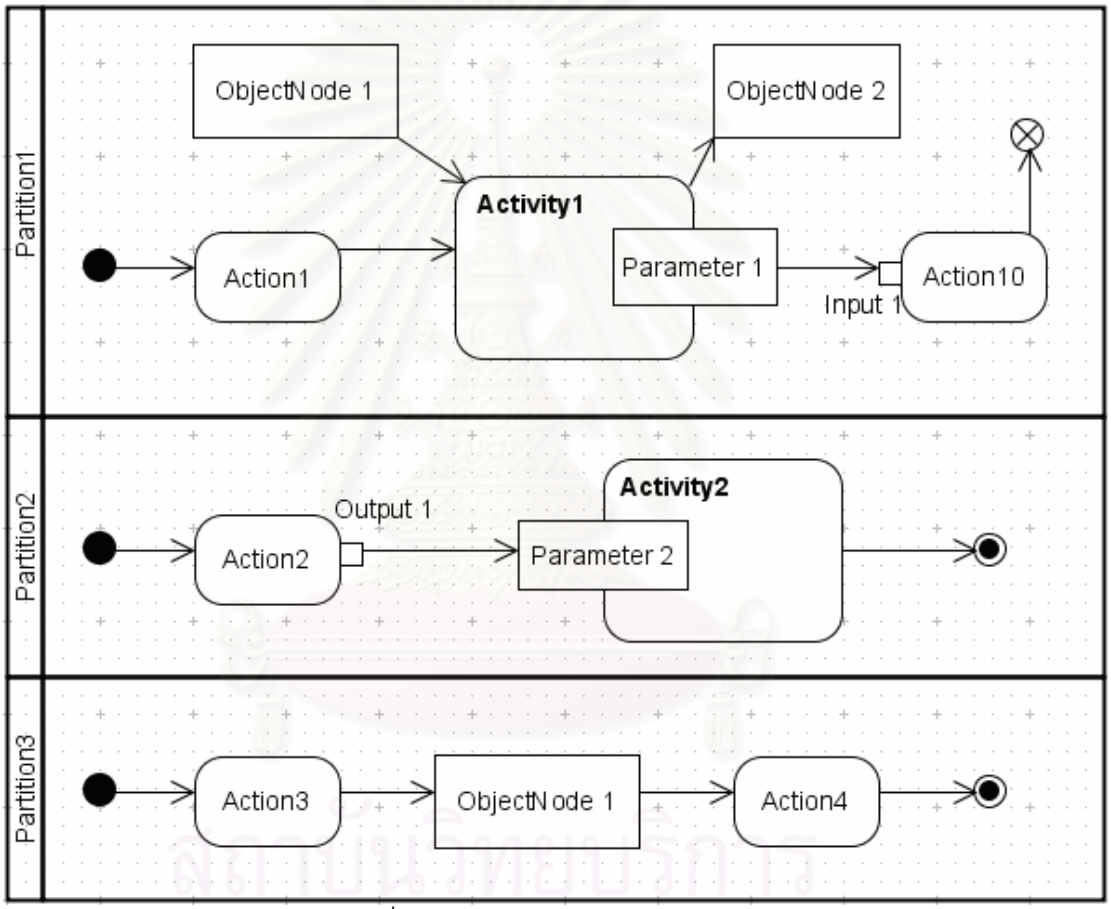
กรณี	ผลลัพธ์
4.1 กระแสงานนั้นจะสิ้นสุดเมื่อเจอโหนด ActivityFinal	ถูกต้อง
4.2 เมื่อเจอโหนด ActivityFinal งานที่ดำเนินการอยู่ในกระแสอื่นๆจะเปลี่ยนสถานะเป็น “Cancelled” ทั้งหมด	ถูกต้อง
5. โหนด AcceptEventAction / โหนด SendSignalAction	
5.1 เมื่อมีการระบุการเกิดเหตุการณ์ ระบบจะรับรู้และสร้างงานถัดไปอย่างถูกต้อง	ถูกต้อง
5.2 เมื่องานหนึ่งทำเสร็จสิ้นและโหนดต่อไปเป็นโหนด AcceptEventAction ระบบจะรออยู่ที่สถานะ “Awaiting”	ถูกต้อง
5.3 เมื่องานหนึ่งทำเสร็จสิ้นและงานต่อไปเป็นโหนด SendSignalAction จะเป็นการระบุการเกิดเหตุการณ์ตามชื่อโหนดนั้นขึ้นในระบบ	ถูกต้อง
6. โหนด InterruptibleActivityRegion / เส้นเชื่อม ExceptionHandler	
6.1 ถ้ามีการระบุการเกิดของเหตุการณ์ที่อยู่ใน InterruptibleActivityRegion และเหตุการณ์นั้นมีเส้นเชื่อม ExceptionHandler ไปยังงานนอกพื้นที่นั้น ระบบจะยกเลิกงานที่ทำอยู่ในพื้นที่นั้น แล้วสร้างงานที่เป็นปลายทางของ ExceptionHandler ขึ้น สำหรับงานที่ถูกยกเลิกจะถูกเปลี่ยนสถานะเป็น “Interrupted”	ถูกต้อง
7. โหนด ActivityPartition	
7.1 ระบบมีการจัดบทบาทของผู้ที่รับผิดชอบทำงานแต่ละงานได้ตามที่ระบุมาด้วยโหนด ActivityPartion ซึ่งอาจมีหลายมิติ	ถูกต้อง
8. โหนด Fork	
8.1 การทำงานแตกออกเป็นหลายกระแสเมื่อเจอโหนด Fork ซึ่งสามารถทำงานไปได้พร้อมกัน	ถูกต้อง
8.2 ในกรณีเป็น Object Flow เอกสารจะถูกส่งไปให้กับทุกกระแสการทำงานได้ถูกต้อง	ถูกต้อง
9. โหนด Decision	
9.1 ระบบมีให้เลือกเงื่อนไขตามที่ระบุใน Decision Node และเมื่อเลือกเงื่อนไขที่เป็นจริงแล้ว ระบบจะสร้างงานถัดไปได้ถูกต้อง รวมถึงหากมีเอกสารก็ส่งไปเป็นอินพุตของงานนั้นด้วย	ถูกต้อง

ตารางที่ 5.1 กรณีทดสอบกรณีแผนภาพกิจกรรมถูกต้อง (ต่อ)

กรณี	ผลลัพธ์
10. โหนด Join	
10.1 การทำงานที่ต่อไปจากโหนด Join ต้องรอให้ทุกกระแสที่เป็นอินพุตทำเสร็จสิ้นก่อน	ถูกต้อง
10.2 ในกรณีเป็น Object Flow เอกสารที่ต่อกับโหนด Join ในทุกกระแสจะเป็นอินพุตของงานในกระแสเอาท์พุตจากโหนด Join	ถูกต้อง
11. โหนด Merge	
11.1 เมื่องานในกระแสเพียงกระแสเดียวที่เป็นอินพุตทำเสร็จ งานที่ต่อจาก MergeNode จะถูกสร้างขึ้นทันที	ถูกต้อง
11.2 เมื่องานในกระแสใดๆที่เป็นอินพุตของ MergeNode ทำเสร็จ งานที่ต่อจาก MergeNode จะถูกสร้างขึ้น โดยงานนั้นถือเป็นงานที่แตกต่างกัน แม้จะมีชื่อเดียวกัน และในกรณี Object Flow เอกสารที่เป็นอินพุตจะมาจากเอกสารของกระแสที่ทำเสร็จในครั้งนั้นๆ	ถูกต้อง

ทั้งนี้แผนภาพกิจกรรมที่ใช้ทดสอบพร้อมคำอธิบายเป็นดังรูปที่ 5.1 ถึง 5.4

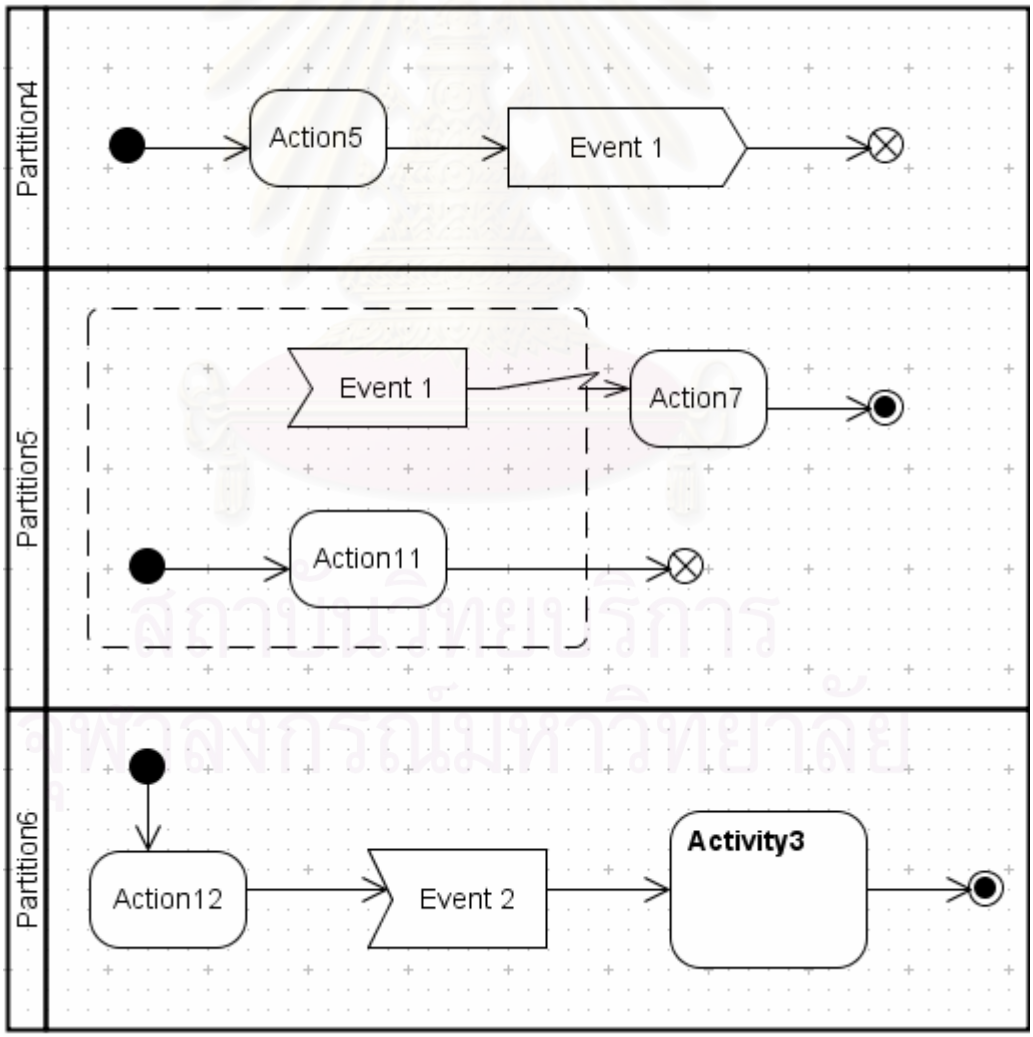
รูปที่ 5.1 เป็นแผนภาพกิจกรรมที่ใช้ทดสอบกรณีทดสอบ 1, 2, 3, 4 โดยงาน Action1, Action2 และ Action3 จะเกิดขึ้นทันทีที่อินสแตนซ์ของกระแสงานนี้ถูกสร้างขึ้น ทั้งนี้หากมีการทำงานไปจนถึงโหนด ActivityFinal โดยยังมีงานในกระแสอื่นค้างอยู่ ตัวอย่างเช่นการทำงาน Activity2 เสร็จสิ้นในขณะที่งาน Action4 ยังคงถูกดำเนินการอยู่ ในกรณีนี้โหนดถัดไปของ Activity2 เป็นโหนด ActivityFinal ซึ่งส่งผลให้งาน Action4 ถูกยกเลิกไป โดยผู้ใช้จะเห็นสถานะของงาน Action4 เป็น “Cancelled”



รูปที่ 5.1 แผนภาพกิจกรรมทดสอบ 1

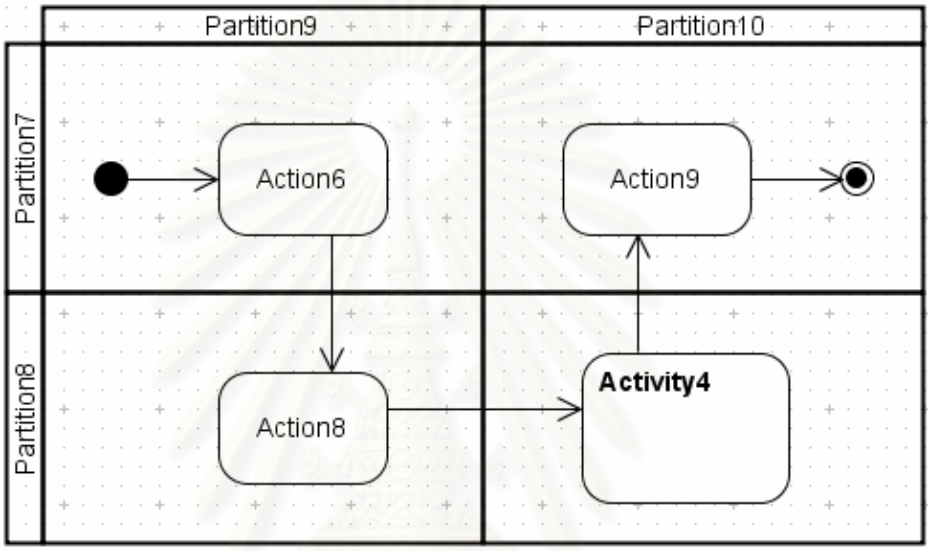
รูปที่ 5.2 เป็นแผนภาพกิจกรรมที่ใช้ทดสอบกรณีทดสอบ 5 และ 6 โดยเมื่องาน Action5 ถูกทำเสร็จสิ้น เหตุการณ์ Event1 จะถูกทำให้เกิดขึ้นโดยอัตโนมัติ ซึ่งหากในขณะนั้นงาน Action11 ซึ่งอยู่ในโหนด InterruptibleActivityRegion ยังทำไม่เสร็จสิ้น งานนี้จะถูกยกเลิกทันที โดยผู้ใช้จะเห็นสถานะของงาน Action11 เป็น “Interrupted” และงานที่เกิดขึ้นถัดไปคืองาน Action7 ซึ่งเป็นโหนดปลายทางของเหตุการณ์ Event1 ที่ต่อมาจากเส้นเชื่อม ExceptionHandler

ในขณะเดียวกัน เมื่องาน Action12 ถูกทำเสร็จสิ้น ระบบจะรอเพื่อให้เหตุการณ์ Event2 เกิดขึ้นโดยผู้ใช้งาน เพื่อเกิดงาน Activity3 ต่อไป ทั้งนี้ในขณะที่ระบบรอเหตุการณ์ Event2 อยู่นั้น ที่หน้าเว็บอินสแตนซ์จะแสดงงาน Event2 เป็นสถานะ “Awaiting” ซึ่งหมายความถึงการรอเหตุการณ์



รูปที่ 5.2 แผนภาพกิจกรรมทดสอบ 2

รูปที่ 5.3 เป็นแผนภาพกิจกรรมที่ใช้ทดสอบกรณีทดสอบ 7 ซึ่งเป็นโหนด ActivityPartition ที่มี 2 มิติ ซึ่งในกรณีนี้ชื่อบทบาทจะเป็นการนำชื่อบทบาทในมิติแนวนอนมาบวกรวมกับชื่อมิติในแนวตั้ง รวมเป็น 4 บทบาท ได้แก่ “Paritition7 Partition9”, “Partition7 Partition10”, “Partition8 Partition9”, “Partition8 Partition10”

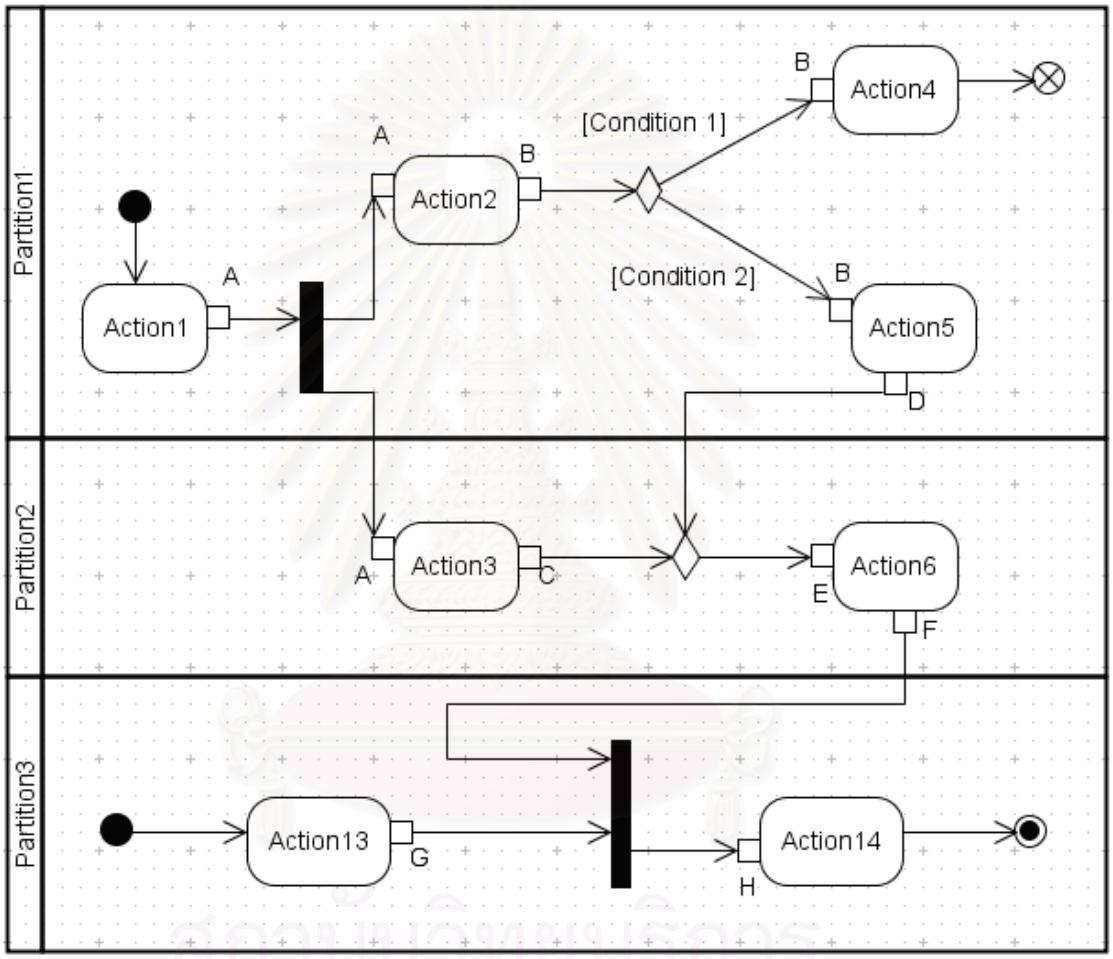


รูปที่ 5.3 แผนภาพกิจกรรมทดสอบ 3

รูปที่ 5.4 เป็นแผนภาพกิจกรรมที่ใช้ทดสอบกรณีทดสอบ 8, 9, 10 และ 11 เริ่มจากเมื่องาน Action ถูกทำเสร็จสิ้นพร้อมทั้งเอาท์พุตชื่อ A ได้ถูกอัพโหลดไปเก็บไว้ที่ระบบเรียบร้อยแล้ว เอาท์พุตนี้จะไปเป็นอินพุตของงาน 2 งาน ได้แก่ อินพุต A ของงาน Action2 และอินพุต A ของงาน Action3 ด้วยโหนด Fork จากนั้นเมื่อ Action2 ถูกทำเสร็จ ระบบจะเปิดหน้าเว็บให้ผู้งานเลือกข้อความที่เป็นจริงระหว่าง “Condition1” และ “Condition2” โดยเมื่อเลือกข้อความใด เอาท์พุต B ของงาน Action2 จะไปเป็นอินพุตของงานถัดไปตามข้อความที่เลือกนั้น (ตัวอย่างเช่นอินพุต B ของงาน Action5)

จากนั้นหากงาน Action5 หรืองาน Action3 ในอีกหนึ่งกระแสที่ออกจากโหนด Fork ถูกทำเสร็จสิ้น งาน Action6 จะถูกสร้างขึ้นทันที และมีอินพุตที่ได้มาจากเอาท์พุตของงานที่ทำเสร็จนั้น ตัวอย่างเช่นหากงาน Action3 ถูกทำเสร็จ เอาท์พุต C ของงาน Action3 จะไปเป็นอินพุต E ของงาน Action6 ทั้งนี้หากหลังจากนั้น Action5 ถูกทำเสร็จ จะเกิดงาน Action6 ขึ้นอีกครั้งซึ่งถือว่าเป็นคนละงานกับ Action6 ที่เกิดขึ้นก่อนหน้า โดยงาน Action6 ที่เกิดขึ้นครั้งหลังนี้จะได้อินพุตมาจากเอาท์พุต D ของ Action5

ในขณะเดียวกันงาน Action13 ในอีกกระแสหนึ่ง เมื่อถูกทำเสร็จสิ้น ระบบจะต้องรอให้งาน Action6 ถูกทำเสร็จเช่นเดียวกันจึงจะสร้างงาน Action14 ขึ้น ในกรณีนี้ทั้งเอาท์พุต G ของงาน Action13 และเอาท์พุต F ของงาน Action6 จะไปเป็นอินพุต H ของงาน Action14 กล่าวคือเอกสารที่อัพโหลดไปสำหรับทั้งสองเอาท์พุตจะถือเป็นอินพุต H



รูปที่ 5.4 แผนภาพกิจกรรมทดสอบ 4

5.1.2 กรณีแผนภาพกิจกรรมไม่ถูกต้อง

กรณีทดสอบนี้เป็นการทดสอบว่าระบบมีการตรวจสอบความไม่ถูกต้องและเตือนให้ผู้ใช้งานทราบอย่างถูกต้อง รายละเอียดเป็นดังในตารางที่ 5.2 ทั้งนี้ข้อความเตือนในกรณีที่แผนภาพกิจกรรมไม่ถูกต้องทั้งหมดถูกระบุในภาคผนวก ง

ตารางที่ 5.2 กรณีทดสอบกรณีแผนภาพกิจกรรมไม่ถูกต้อง

กรณี	ผลลัพธ์
1. แผนภาพกิจกรรมมีสัญกรณ์ที่ไม่รองรับ หรือมีแผนภาพย่อย, สเตอริโอไทพ์	ถูกต้อง
2. ไฟล์เอ็กซ์เอ็มไอไม่ได้เอ็กพอร์ตมาจากเครื่องมือ Visual Paradigm for UML	ถูกต้อง
3. แผนภาพกิจกรรมมีโหนดที่ไม่ได้อยู่ในโหนด ActivityPartition	ถูกต้อง
4. แผนภาพกิจกรรมไม่มีโหนด Initial หรือโหนด ActivityFinal	ถูกต้อง
5. โหนด Activity มีโหนดอื่นอยู่ภายใน	ถูกต้อง
6. โหนด AcceptEventAction หรือ SendSignalAction มีอินพุตหรือเอาท์พุต	ถูกต้อง
7. โหนด Action, Activity, SendSignalAction ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออก	ถูกต้อง
8. โหนด AcceptEventAction ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก หรือมีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	ถูกต้อง
9. โหนด Input Pin, Output Pin, ActivityParameter, Object ไม่แสดงชื่อ (เป็นค่าว่างเปล่า)	ถูกต้อง
10. โหนด Input Pin, Output Pin, ActivityParameter, Object มีมากกว่าหนึ่งเส้นเชื่อมขาเข้าหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก	ถูกต้อง
11. โหนด Object ต่อกับโหนด Initial โดยตรง	ถูกต้อง
12. โหนด Initial ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก	ถูกต้อง
13. โหนด FlowFinal, ActivityFinal ไม่มีเส้นเชื่อมขาเข้า	ถูกต้อง
14. โหนด Action, Activity มีเส้นเชื่อมขาเข้าที่ต่างชนิดกัน หรือเส้นเชื่อมขาออกที่ต่างชนิดกัน	ถูกต้อง
15. โหนด Fork ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	ถูกต้อง
16. โหนด Fork มีเส้นเชื่อมขาเข้าต่างชนิดกับเส้นเชื่อมขาออก	ถูกต้อง
17. โหนด Join ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก	ถูกต้อง
18. โหนด Join มีเส้นเชื่อมขาเข้าที่เป็น Object Flow แต่ไม่มีเส้นเชื่อมขาออกที่เป็น Object Flow	ถูกต้อง
19. โหนด Fork หรือ Join มีเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออกที่เป็น Object Flow โดยที่วัตถุนั้นไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใดๆ	ถูกต้อง
20. โหนด Merge มีมากกว่าหนึ่งเส้นเชื่อมขาออก	ถูกต้อง
21. โหนด Decision มีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	ถูกต้อง

ตารางที่ 5.2 กรณีทดสอบกรณีแผนภาพกิจกรรมไม่ถูกต้อง (ต่อ)

กรณี	ผลลัพธ์
22. โหนด Merge หรือ Decision มีเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออกที่ไม่เป็นชนิดเดียวกันทั้งหมด (มีเส้นเชื่อมที่เป็นชนิด Control Flow และเส้นเชื่อมที่เป็นชนิด Object Flow)	ถูกต้อง
23. โหนด Merge หรือ Decision มีเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออกที่เป็น Object Flow โดยที่วัตถุนั้นไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใดๆ	ถูกต้อง

5.1.3 กรณีแก้ไขกระแสงานและทำกลับการทำงาน

กรณีทดสอบนี้เป็นการทดสอบว่าระบบอนุญาตให้แก้ไขกระแสงานได้ในกรณีที่ทำตามเงื่อนไข และทำงานต่อได้อย่างถูกต้อง หรือหากการแก้ไขไม่ถูกต้องก็ไม่อนุญาตให้แก้ไขพร้อมทั้งบอกข้อมูลความไม่ถูกต้องนั้นให้ผู้ใช้งานทราบ รวมถึงการทำกลับการทำงานซึ่งระบบจะอนุญาตให้ทำได้เมื่องานถัดไปยังไม่ถูกบันทึกการเริ่มทำงาน รายละเอียดเป็นดังในตารางที่ 5.3

ตารางที่ 5.3 กรณีทดสอบกรณีแก้ไขกระแสงานและทำกลับการทำงาน

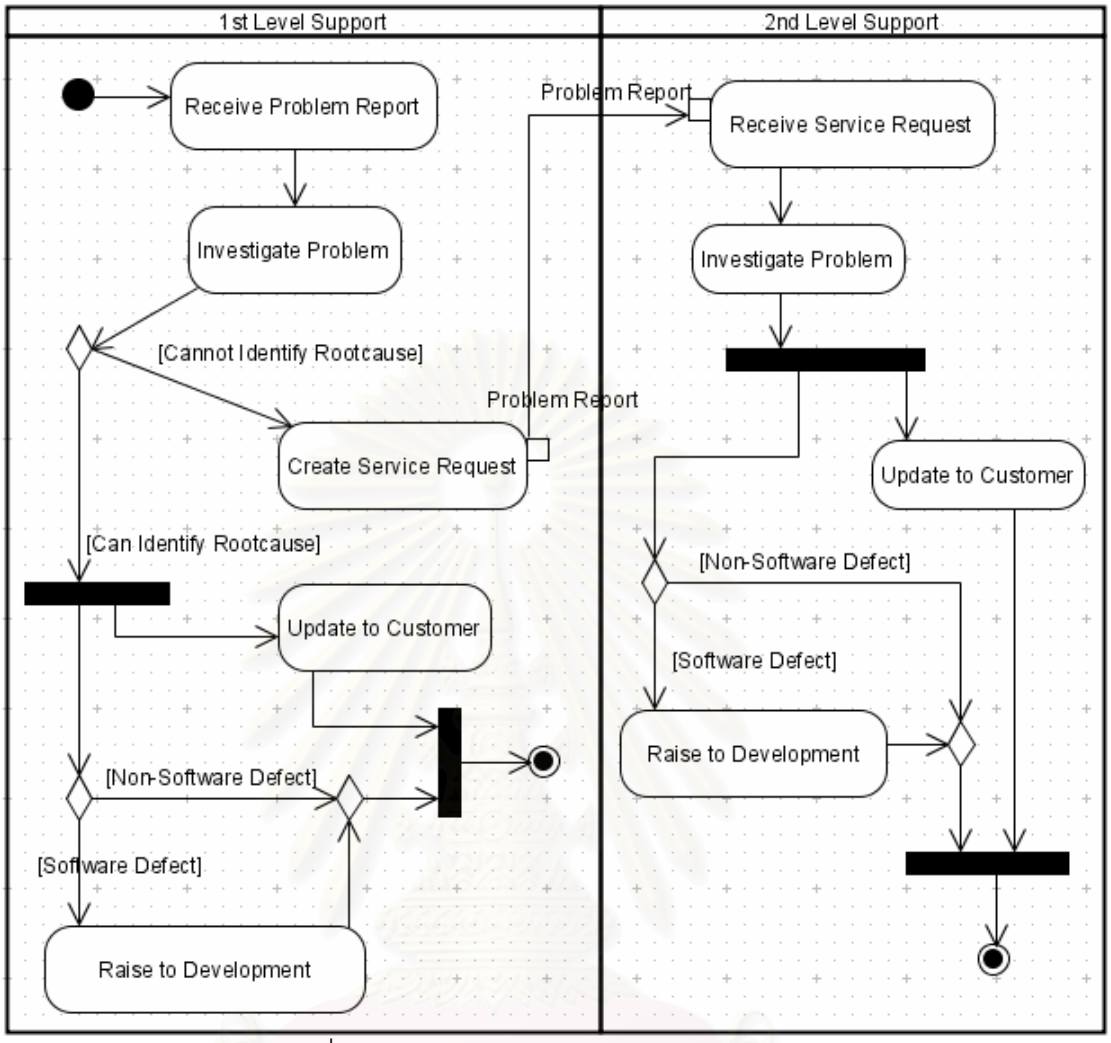
กรณี	ผลลัพธ์
1. ระบบอนุญาตให้แก้ไขกระแสงานที่ไม่มีอินสแตนซ์ที่ยังดำเนินการอยู่	ถูกต้อง
2. ระบบอนุญาตให้แก้ไขกระแสงานเมื่อการแก้ไขทำตามเงื่อนไข	ถูกต้อง
3. เมื่อระบบอนุญาตให้แก้ไขกระแสงาน ระบบเพิ่มเลขเวอร์ชันของกระแสงานและแสดงให้เห็นว่าอินสแตนซ์ที่ดำเนินการอยู่เป็นการทำงานบนเวอร์ชันใหม่ของกระแสงาน แต่ไม่เปลี่ยนแปลงอินสแตนซ์ที่ดำเนินการอยู่	ถูกต้อง
4. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อกระแสงานใหม่ไม่ถูกต้อง (ตามเงื่อนไขของสัญกรณ์)	ถูกต้อง
5.ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อการแก้ไขไม่ได้ทำบนไฟล์เอ็กซ์เอ็มไอเดิม	ถูกต้อง
6. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อกระแสงานในไฟล์เอ็กซ์เอ็มไอใหม่มีชื่อต่างไปจากชื่อกระแสงานที่ระบุว่าแก้ไข	ถูกต้อง
7. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อจำนวนโหนด Initial ในกระแสงานหลังการแก้ไขไม่เท่ากับกระแสงานก่อนการแก้ไข	ถูกต้อง

ตารางที่ 5.3 กรณีทดสอบกรณีแก้ไขกระแสงานและทำกลับการทำงาน (ต่อ)

กรณี	ผลลัพธ์
8. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อมีการแก้ไขคุณสมบัติของงานบางงานในอินสแตนซ์ที่ยังดำเนินการอยู่ของกระแสงานนั้น (แก้ไขคุณสมบัติของโหนด Action, Activity ที่เป็นต้นแบบของงานเหล่านั้น)	ถูกต้อง
9. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่องานบางงานในอินสแตนซ์ที่ยังดำเนินการอยู่ของกระแสงานนั้นมีโหนดก่อนหน้าต่างจากเดิมก่อนแก้ไข	ถูกต้อง
10. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่องานที่เสร็จสิ้นไปแล้วบางงานในอินสแตนซ์ที่ยังดำเนินการอยู่ของกระแสงานนั้นมีโหนดต่อท้ายต่างจากเดิมก่อนแก้ไข	ถูกต้อง
11. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่องานบางงานในอินสแตนซ์ที่ยังดำเนินการอยู่ และมีโหนด Decision อยู่ก่อนหน้า มีเงื่อนไขของเส้นเชื่อมที่เชื่อมต่อจากโหนด Decision มางานนี้เปลี่ยนไปหลังการแก้ไข	ถูกต้อง
12. ระบบไม่อนุญาตให้แก้ไขกระแสงานที่มีโหนด Fork และมีอินสแตนซ์ที่ยังดำเนินการอยู่ที่การทำงานผ่านเลยจากโหนด Fork นั้นแล้ว แต่เส้นเชื่อมขาออกจากโหนด Fork นั้นมีจำนวนเพิ่มขึ้น	ถูกต้อง
13. ระบบไม่อนุญาตให้แก้ไขกระแสงานเมื่อมีการแก้ไขหรือลบเหตุการณ์บางเหตุการณ์ในอินสแตนซ์ที่ยังดำเนินการอยู่ รวมถึงหากมีการเพิ่มเหตุการณ์จะต้องมีชื่อที่ต่างไปกับทุกเหตุการณ์ที่มีอยู่เดิมในกระแสงานก่อนการแก้ไข	ถูกต้อง
14. ระบบอนุญาตให้ทำกลับการทำงานที่ทำเสร็จสิ้นไปแล้วในกรณีที่งานถัดไปยังไม่ถูกบันทึกเริ่มทำงาน และอินสแตนซ์นั้นยังไม่จบการทำงาน	ถูกต้อง
15. ระบบไม่อนุญาตให้ทำกลับการทำงานที่ทำเสร็จสิ้นไปแล้วในกรณีที่งานถัดไปถูกบันทึกเริ่มทำงานแล้ว หรืออินสแตนซ์นั้นจบการทำงานแล้ว	ถูกต้อง

5.2 การทดลองใช้งาน

ในส่วนนี้เป็นการทดลองใช้งานระบบตั้งแต่เริ่มต้น โดยตัวอย่างกระแสงานที่ถูกควบคุมคือกระแสงานการรองรับปัญหาซอฟต์แวร์ของลูกค้า ดังรูปที่ 5.5

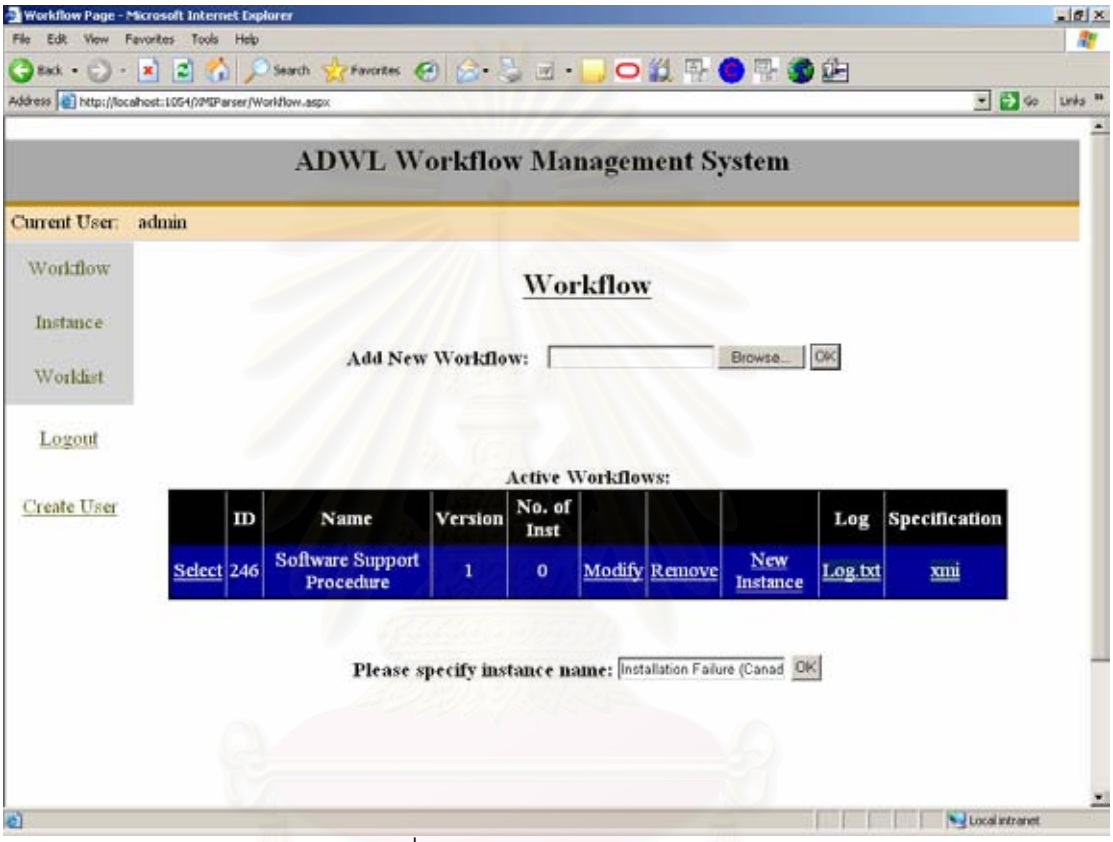


รูปที่ 5.5 กระแสงานการรองรับปัญหาของลูกค้า

ในกระแสงานนี้มีบทบาทอยู่สองบทบาท คือ 1st Level Support และ 2nd Level Support โดยปัญหาจะถูกรายงานผ่านมาทาง 1st Level Support เพื่อทำการสืบหาสาเหตุ หากไม่สามารถบอกถึงสาเหตุของปัญหาได้ภายในสามวัน (ในแผนภาพมีการใส่ข้อมูลระยะเวลา) ก็จะต้องส่งผ่านไปยัง 2nd Level Support ซึ่งมีความชำนาญในซอฟต์แวร์มากกว่าเพื่อสืบหาสาเหตุต่อไป ในที่สุดเมื่อค้นพบสาเหตุจะต้องรายงานสาเหตุของปัญหานั้นไปยังลูกค้า ในขณะเดียวกันก็ต้องดูว่าสาเหตุนั้นเป็นข้อผิดพลาดของซอฟต์แวร์หรือไม่ ถ้าใช่ก็ต้องรายงานไปยังทีมพัฒนา เพื่อทำการแก้ไขต่อไป ซึ่งถือว่าเป็นส่วนที่อยู่นอกเหนือกระแสงานนี้

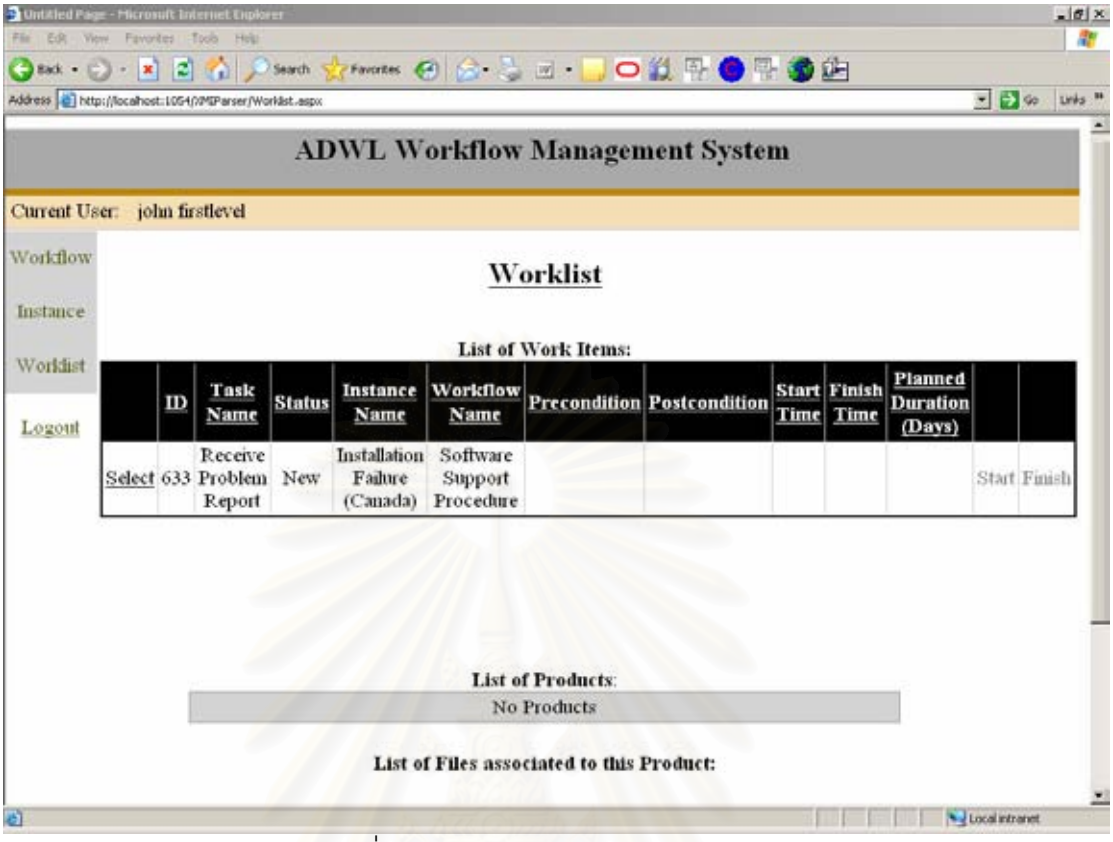
การทำงานจะเริ่มจากผู้บริหารระบบล็อกอินเข้าสู่ระบบที่หน้าล็อกอินเพื่อเพิ่มกระแสงานการรองรับปัญหาของลูกค้าเข้าสู่ระบบ รวมถึงลงทะเบียนผู้ใช้ใหม่ที่มีบทบาทเป็น 1st Level Support และ 2nd Level Support จากนั้นเมื่อมีลูกค้ารายงานปัญหาเข้ามา ผู้บริหารระบบ

ต้องเป็นผู้สร้างอินสแตนซ์ใหม่ของกระแสงานโดยการคลิกปุ่ม ‘New Instance’ บนกระแสงานที่ต้องการสร้างอินสแตนซ์ จากนั้นระบบจะให้ใส่ชื่อของอินสแตนซ์ ดังรูปที่ 5.6 ซึ่งชื่อนี้จะต้องไม่ซ้ำกับอินสแตนซ์เดิมของกระแสงานนี้ เมื่อคลิก ‘OK’ จะเป็นการเพิ่มอินสแตนซ์เข้าสู่ระบบ ในที่นี้อินสแตนซ์ชื่อ Installation Failure (Canada)



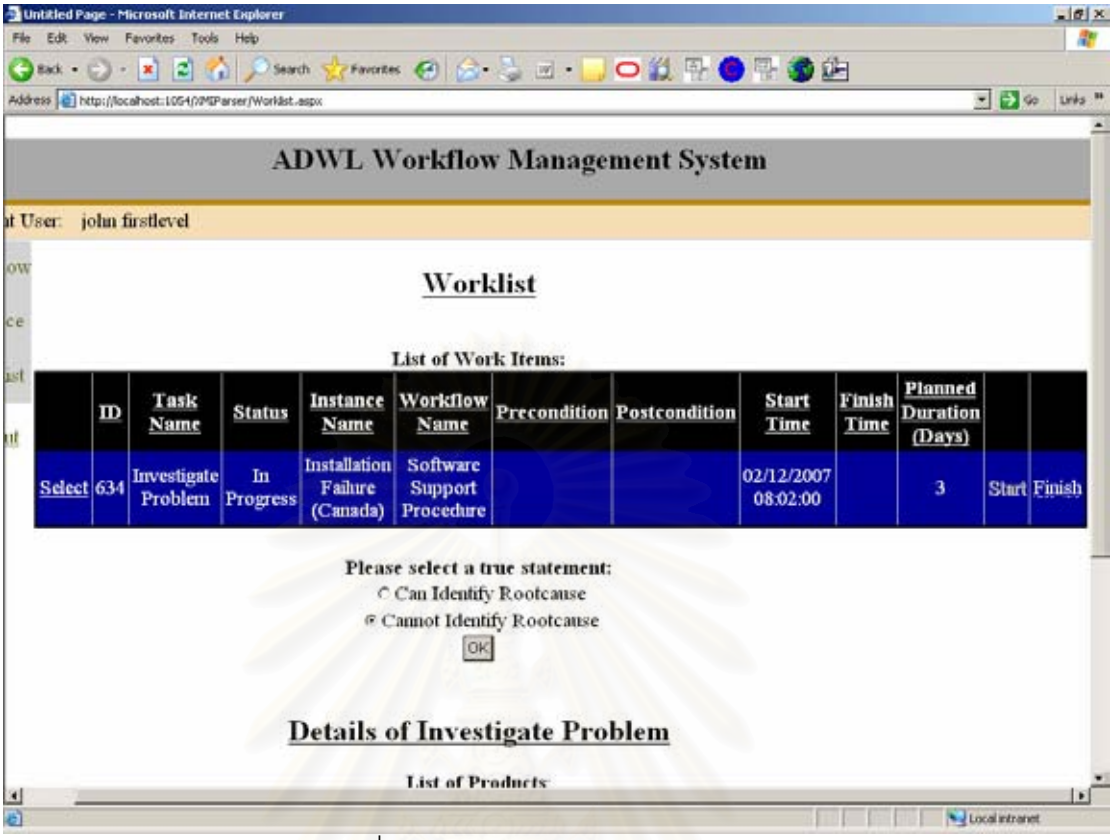
รูปที่ 5.6 การสร้างอินสแตนซ์ใหม่

เมื่อเริ่มทำงาน โดยในที่นี้คือเมื่อผู้ใช้ John FirstLevel ที่มีบทบาทเป็น 1st Level Support ล็อกอินเข้าสู่ระบบ ผู้ใช้จะเห็นรายการงานซึ่งมีงาน Receive Problem อยู่ ดังรูปที่ 5.7 ซึ่งมีสถานะเป็น “New” เนื่องจากเป็นงานใหม่ เมื่อคลิกปุ่ม Start เป็นการบอกว่าได้รับรู้ถึงงานนั้นแล้ว และสถานะของงานจะเปลี่ยนเป็น “In Progress” สุดท้ายเมื่อคลิกปุ่ม Finish เป็นการระบุการทำงานเสร็จสิ้น งานนี้จะถูกเปลี่ยนสถานะเป็น “Done” และถูกลบออกไปจากรายการงาน



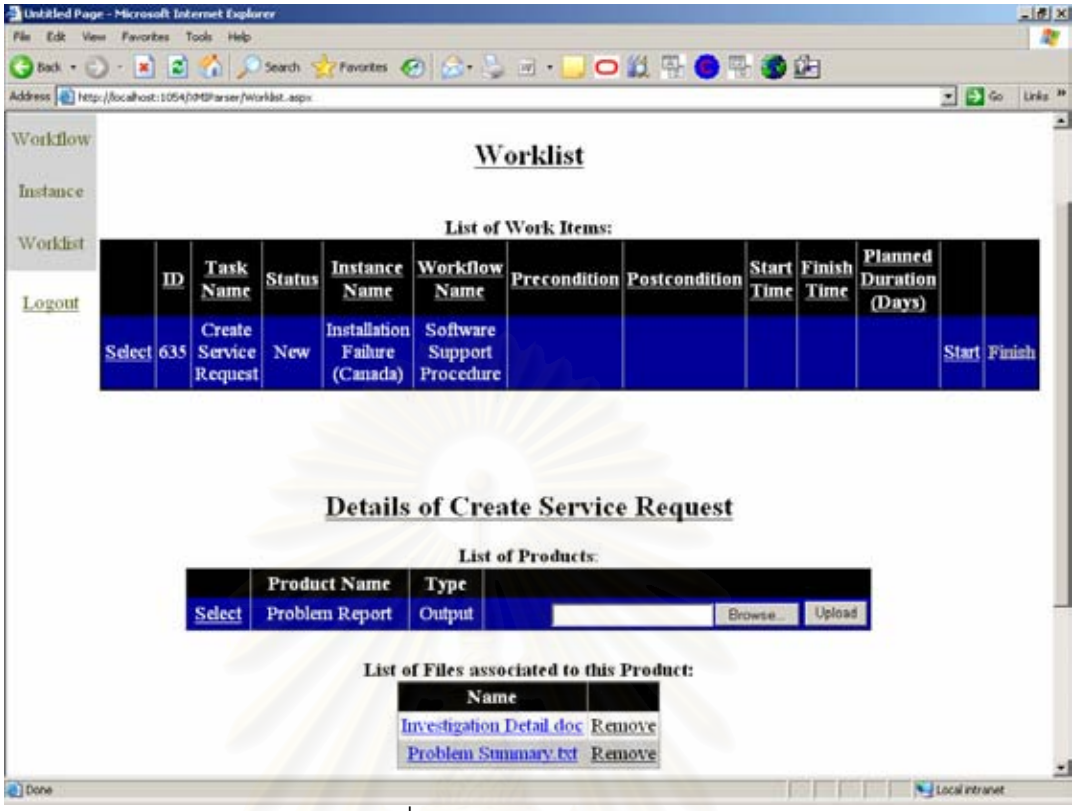
รูปที่ 5.7 รายการงานสถานะเป็น “New”

ในลักษณะเดียวกัน งานถัดไปตามที่ระบุในแผนภาพกิจกรรมคืองาน Investigate Problem จะถูกใส่เข้ามาในรายการงานแทน และทันทีที่ทำงาน Investigate Problem เสร็จ จากข้อกำหนดของกระแสงานจะเป็นการตัดสินใจระหว่าง “Can Identify Rootcause” และ “Cannot Identify Rootcause” ซึ่งมีงานถัดไปที่ต่างกัน ระบบจะแสดงข้อความเหล่านี้ให้ผู้ใช้เห็นเพื่อเลือกข้อความที่เป็นจริง ดังรูปที่ 5.8 ในที่นี้ผู้ใช้เลือก “Cannot Identify Rootcause”

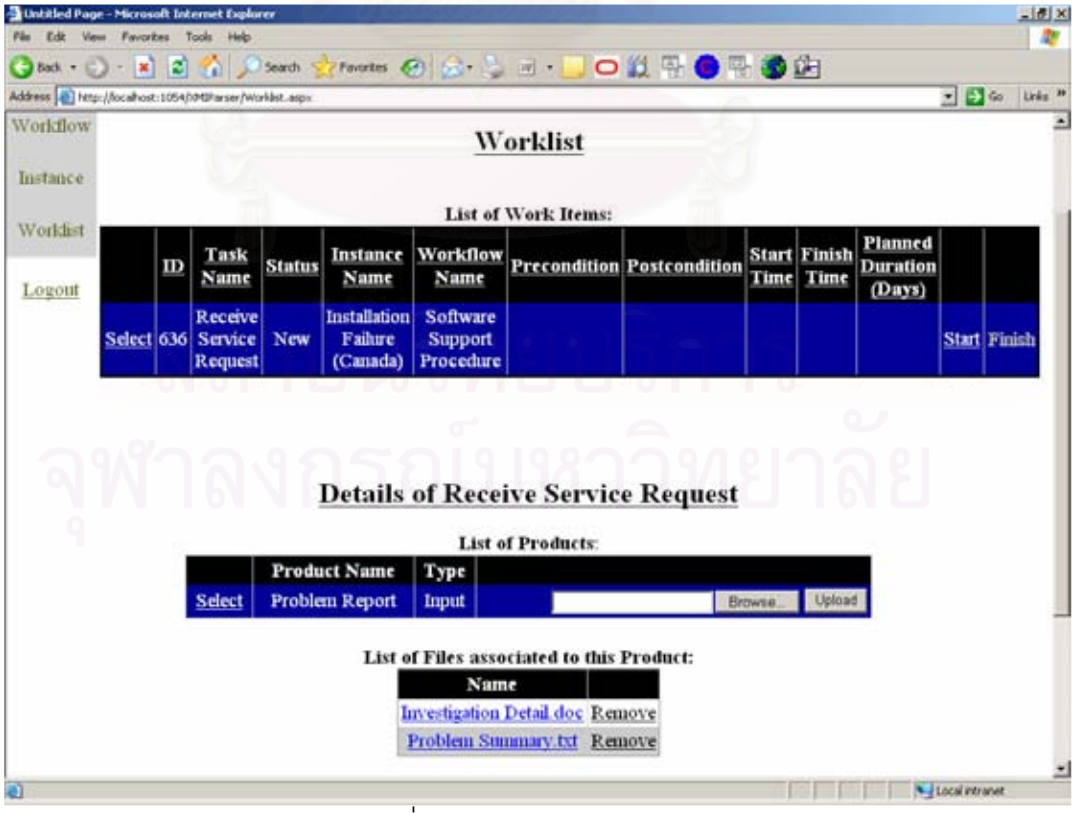


รูปที่ 5.8 การแสดงข้อความให้ผู้ใช้เลือก

เมื่อข้อความ Cannot Identify Rootcause ถูกเลือก จะนำไปสู่งานถัดไปคือ Create Service Request ซึ่งงานนี้มีเอาท์พุตชื่อ Problem Report เพื่อเป็นอินพุตไปยังงาน Receive Service Request ต่อไป การอัพโหลดเอกสารสามารถทำได้โดยการเลือกเอาท์พุตและคลิก ‘Browse’ ไปยังไฟล์ที่ต้องการ ซึ่งสามารถอัพโหลดได้มากกว่า 1 ไฟล์ต่อ 1 เอาท์พุตหรืออินพุต ดังตัวอย่างในรูปที่ 5.9 เอกสาร 2 ไฟล์ได้ถูกอัพโหลดเป็นเอาท์พุต Problem Report ซึ่งทั้ง 2 ไฟล์นี้ก็จะไปเป็นอินพุตชื่อ Problem Report ของงาน Receive Service Request ซึ่งรับผิดชอบโดยบทบาท 2nd Level Support ดังรูปที่ 5.10 ในที่นี้ผู้ใช้ที่มีบทบาทนี้คือ Dan SecondLevel

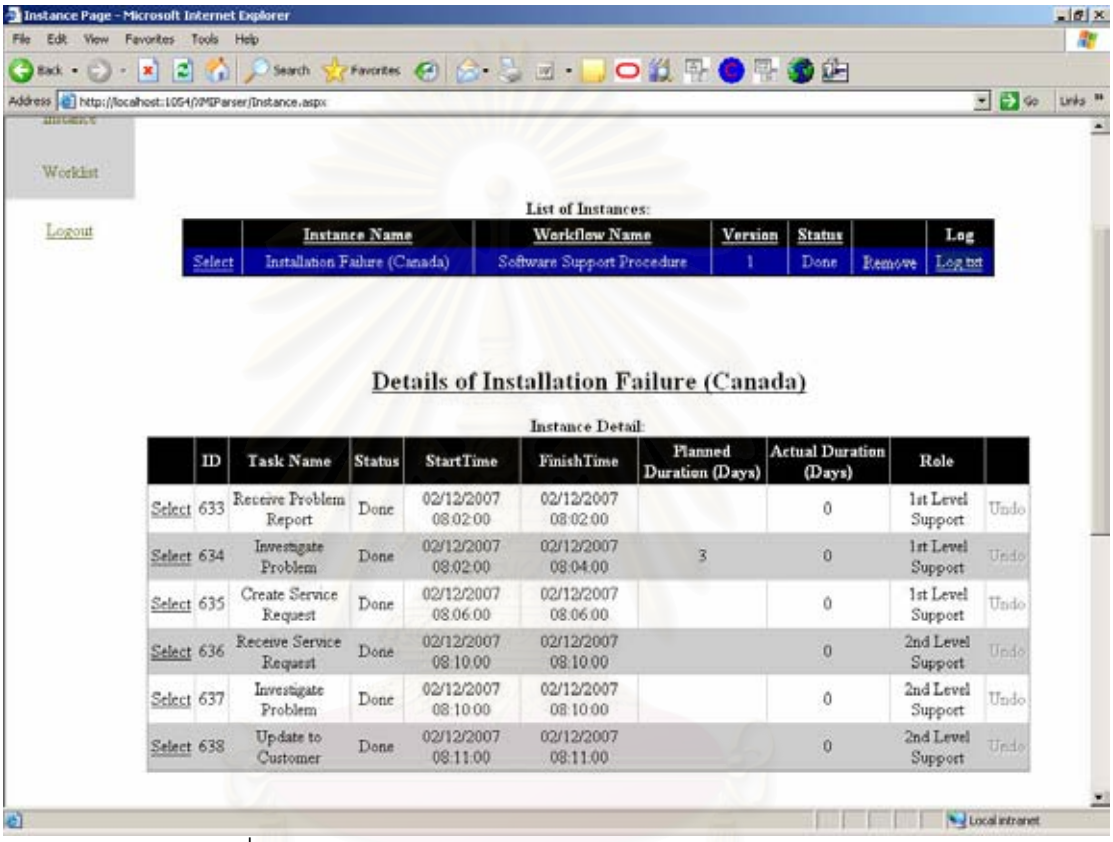


รูปที่ 5.9 หน้าจอแสดงเอาท์พุต



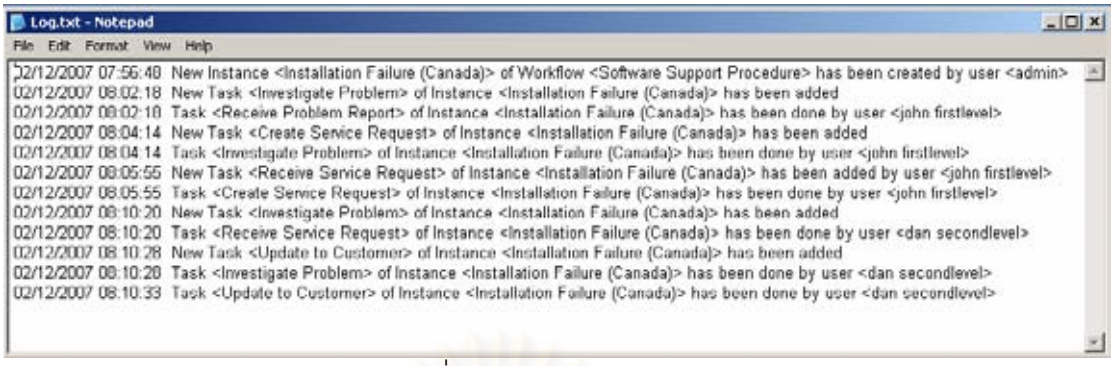
รูปที่ 5.10 หน้าจอแสดงอินพุต

เมื่อการทำงานดำเนินไปจนกระทั่งระบบพบว่าโหนดถัดไปคือโหนด ActivityFinal ระบบจะจบการทำงานของอินสแตนซ์นี้ โดยผู้ใช้สามารถดูการทำงานที่ผ่านมาของทั้งอินสแตนซ์ในภาพรวมได้โดยการเลือกเมนู ‘Instance’ เพื่อไปที่หน้าเว็บ Instance ดังรูปที่ 5.11 จะเห็นรายละเอียดของการทำงานที่ผ่านมาของอินสแตนซ์ Installation Failure (Canada)



รูปที่ 5.11 หน้าเว็บแสดงรายละเอียดการทำงานของอินสแตนซ์

หากต้องการดูรายละเอียดที่มากขึ้น ผู้ใช้สามารถดูได้จากล็อกไฟล์โดยการเลือกที่ Log.txt ของอินสแตนซ์ที่ต้องการ หรือหากต้องการจัดเก็บก็สามารถทำได้โดยการคลิกขวาที่ Log.txt แล้วเลือกเมนู ‘Save Target As…” โดยตัวอย่างในรูป 5.12 แสดงตัวอย่างข้อมูลในล็อกไฟล์ของอินสแตนซ์ Installation Failure (Canada) ซึ่งสังเกตว่าจะมีข้อมูลตั้งแต่ที่อินสแตนซ์ถูกสร้างขึ้น รวมถึงมีชื่อผู้ใช้ที่เป็นผู้บันทึกการเริ่มทำงาน และทำงานเสร็จสิ้น



รูปที่ 5.12 ตัวอย่างล็อกไฟล์

5.3 การประเมินโดยใช้แบบรูปของกระแสงาน

ในส่วนนี้จะเป็นการประเมินระบบการจัดการกระแสงานที่ได้พัฒนาขึ้นโดยใช้แบบรูปของกระแสงาน อย่างไรก็ตามข้อจำกัดต่างๆของระบบการจัดการกระแสงานจะขึ้นอยู่กับความสามารถของภาษาที่ใช้สร้างข้อกำหนดของกระแสงานว่าสามารถรองรับแบบรูปใดได้บ้าง สำหรับแผนภาพกิจกรรมของยูเอ็มแอลนั้นได้มีการวิเคราะห์อย่างละเอียดในงานวิจัยของ Russell และคณะ [15] ดังได้กล่าวมาแล้วในบทที่ 2 ถึงแบบรูปที่แผนภาพกิจกรรมสามารถแสดงได้ ซึ่งแบ่งออกเป็น 17 แบบรูปด้านกระแสการควบคุม 18 แบบรูปด้านกระแสข้อมูล และ 8 แบบรูปด้านทรัพยากร แต่สิ่งที่ต่างออกไปของระบบการจัดการกระแสงานที่พัฒนาขึ้นในงานวิจัยนี้และเป็นผลให้ต้องมีการวิเคราะห์เพิ่มมีสาเหตุมาจากข้อจำกัดของระบบที่รองรับทั้งสิ้น 20 สัญกรณ์ของแผนภาพกิจกรรม และมี 4 สัญกรณ์ที่ระบบไม่รองรับ ทำให้อาจมีบางแบบรูปที่ระบบไม่สามารถแสดงได้ ทั้งนี้การวิเคราะห์จะจำกัดอยู่เพียงแบบรูปที่แผนภาพกิจกรรมของยูเอ็มแอลทั่วไปรองรับเท่านั้น รวมทั้งสิ้น 43 แบบรูป (17 แบบรูปด้านกระแสการควบคุม 18 แบบรูปด้านกระแสข้อมูล และ 8 แบบรูปด้านทรัพยากร)

อย่างไรก็ตาม โดยการอ้างอิงตามงานวิจัยของ Russell และคณะ เกือบทั้งหมดของแบบรูปที่แผนภาพกิจกรรมทั่วไปรองรับนั้นสามารถถูกแสดงได้โดยใช้สัญกรณ์เพียง 20 สัญกรณ์ที่ระบบรองรับ กล่าวคือไม่มีแบบรูปใดที่จำเป็นต้องใช้สัญกรณ์ AcceptTimeEventAction, Datastore, CentralBuffer หรือ ExpansionRegion ซึ่งระบบไม่รองรับในการแสดง ทั้งนี้มีเพียงหนึ่งแบบรูปที่ระบบการจัดการกระแสงานนี้ไม่รองรับ คือแบบรูปการทำงานอย่างอัตโนมัติ (Automatic Execution) ซึ่งเป็นแบบรูปในทัศนมิติด้านทรัพยากร (ทัศนมิติที่บรรยายถึงคนที่มีหน้าที่ทำงาน) และเป็นการแสดงถึงงานที่ทำได้โดยอัตโนมัติโดยไม่ต้องอาศัยคน เนื่องจากในกระแสงานที่เป็นอินพุตของระบบนี้มีข้อจำกัดคือทุกโหนดต้องอยู่ใน

โหนด ActivityPartition ซึ่งจะระบุบทบาทของผู้มีหน้าที่รับผิดชอบทำงาน กล่าวคือระบบไม่มีความสามารถในการทำงานใดๆให้ ผู้ใช้งานที่มีบทบาทตามที่กำหนดมาในแผนภาพกิจกรรมต้องเป็นผู้รับผิดชอบทำงานนั้นเอง ระบบจึงไม่รองรับแบบรูปนี้

กล่าวโดยสรุป ระบบการจัดการกระแสงานที่พัฒนาขึ้นสามารถรองรับได้ 17 แบบรูปด้านกระแสการควบคุม 18 แบบรูปด้านกระแสข้อมูล และ 7 แบบรูปด้านทรัพยากร รวมทั้งสิ้นเป็น 42 แบบรูปจากทั้งหมด 43 แบบรูปที่แผนภาพกิจกรรมทั่วไปรองรับ

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอการพัฒนาระบบการจัดการกระแสงานที่มีข้อกำหนดของกระแสงานเป็นแผนภาพกิจกรรมของยูเอ็มแอลเวอร์ชัน 2.0 โดยแผนภาพกิจกรรมนั้นต้องถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอที่เอ็กซ์พอร์ตมาจากเครื่องมือ Visual Paradigm for UML ทั้งนี้การที่ต้องกำหนดชนิดเครื่องมือเป็นผลมาจากมาตรฐานเอ็กซ์เอ็มไอที่มีแท็กแตกต่างกันไปสำหรับแต่ละเครื่องมือ

ระบบการจัดการกระแสงานที่พัฒนาขึ้นมีลักษณะเป็นเว็บแอพพลิเคชัน ซึ่งถือว่าเป็นลักษณะที่เหมาะสมกับระบบการจัดการกระแสงานที่ต้องมีผู้ใช้งานหลายคนเข้าใช้งานพร้อมกัน โดยสถาปัตยกรรมของระบบจะแบ่งเป็น 4 คอมโพเนนต์คือ ส่วนแกนหลักของระบบ, ส่วนการเข้าถึงข้อมูล, ฐานข้อมูล และส่วนต่อประสานกับผู้ใช้ การแบ่งเช่นนี้มีข้อดีคือสามารถแยกตรรกะของการเข้าถึงข้อมูลในฐานข้อมูล และตรรกะของส่วนการแสดงผลทางหน้าจอให้เป็นอิสระกับส่วนประมวลผลหลัก ทำให้การแก้ไขและพัฒนาต่อเป็นไปได้ง่าย

ส่วนหลักของการวิจัยได้เน้นไปที่การวิเคราะห์แผนภาพกิจกรรมเพื่อนำสัญกรณ์ของแผนภาพกิจกรรมมาใช้สร้างข้อกำหนดของกระแสงาน ซึ่งค้นพบว่าจำเป็นต้องสร้างข้อกำหนดและข้อจำกัดขึ้นมาหลายข้อเพื่อที่ทำให้สามารถตีความและควบคุมการทำงานให้เป็นไปตามข้อกำหนดของกระแสงานที่เป็นแผนภาพกิจกรรมนั้นเป็นไปได้อย่างถูกต้อง ทั้งนี้โดยสรุประบบรองรับทั้งสิ้น 20 สัญกรณ์ของแผนภาพกิจกรรม

ผู้วิจัยได้ทำการทดสอบระบบที่พัฒนาขึ้นเพื่อให้สามารถนำไปใช้งานได้จริง ซึ่งพบว่าระบบมีความสามารถพื้นฐานเช่นเดียวกับระบบการจัดการกระแสงานโดยทั่วไป กล่าวคือ ระบบรองรับการจับคู่ระหว่างผู้ใช้งานกับบทบาทที่มีอยู่ในแผนภาพกิจกรรมทั้งหมดที่ระบบรู้จัก และแสดงรายการงานของผู้ใช้แต่ละคนในแต่ละเวลาได้อย่างถูกต้อง รวมถึงจัดเก็บเอกสารที่เป็นอินพุตหรือเอาท์พุตของแต่ละงานให้ และรองรับการทำกลับการทำงานและการแก้ไขกระแสงานที่มีอินสแตนซ์ของกระแสงานที่ยังดำเนินการอยู่ภายใต้เงื่อนไขที่กำหนด ทั้งนี้ระบบจะทำการบันทึกข้อมูลการทำงานต่างๆ เช่น เวลาที่ผู้ใช้บันทึกการเริ่มทำงานและทำงานเสร็จสิ้นลงในล็อกไฟล์เพื่อการดูรายละเอียดการทำงานในภายหลัง นอกจากนี้ผู้วิจัยยังได้ทำการวิเคราะห์ระบบตามแบบรูป

ของกระแสงานโดยอ้างอิงตามงานวิจัยก่อนหน้าที่สรุปว่าแผนภาพกิจกรรมของยูเอ็มแอลรองรับ 43 แบบรูปของกระแสงาน จากการวิเคราะห์พบว่าระบบการจัดการกระแสงานที่พัฒนาขึ้นรองรับทั้งสิ้น 42 แบบรูป แบ่งเป็น 17 แบบรูปด้านกระแสการควบคุม 18 แบบรูปด้านกระแสข้อมูล และ 7 แบบรูปด้านทรัพยากร ขาดเพียงแบบรูปการทำงานอย่างอัตโนมัติซึ่งเป็นแบบรูปด้านทรัพยากรที่แผนภาพกิจกรรมของยูเอ็มแอลโดยทั่วไปรองรับแต่ระบบการจัดการกระแสงานที่พัฒนาขึ้นไม่รองรับ

ทั้งนี้บางส่วนของงานวิจัยนี้ได้รับการตีพิมพ์ในวารสาร Proceedings ของการประชุมทางวิชาการวิทยาการคอมพิวเตอร์ และวิศวกรรมคอมพิวเตอร์แห่งชาติครั้งที่ 11 (NCSEC 2007) ดังรายละเอียดในภาคผนวก ก

6.2 ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำระบบที่ถูกพัฒนาขึ้นมาไปเป็นตัวอย่างหนึ่งในการสร้างระบบการจัดการกระแสงานที่รับข้อกำหนดของกระแสงานในรูปแบบของแผนภาพกิจกรรมเพื่อการพัฒนาในขั้นต่อไป ซึ่งทั้งนี้จะทำให้เทคโนโลยีของระบบการจัดการกระแสงานเป็นที่แพร่หลายมากขึ้น รวมถึงเป็นพื้นฐานของการวิจัยเพื่อหาวิธีการที่เหมาะสมมากที่สุดในการสร้างข้อกำหนดของกระแสงานในอนาคต

6.3 ข้อเสนอแนะ

ผู้วิจัยพบว่าระบบจะถูกนำไปใช้ประโยชน์ในวงกว้างได้มากขึ้นหากมีการเพิ่มเติมส่วนงานดังต่อไปนี้

1) สามารถรับข้อมูลในรูปแบบมาตรฐานเอ็กซ์เอ็มไอได้หลายรูปแบบมากขึ้น ทั้งนี้ในระบบการจัดการกระแสงานนี้รองรับเพียงไฟล์เอ็กซ์เอ็มไอที่เอ็กซ์พอร์ตมาจากเครื่องมือ Visual Paradigm for UML เท่านั้น

2) สามารถรองรับสัญกรณ์อื่นๆทั้งหมดของแผนภาพกิจกรรมของยูเอ็มแอล

3) สามารถรองรับการตั้งค่าอื่นๆที่มีประโยชน์กับการจัดการกระแสงาน ตัวอย่างเช่น Escalation ซึ่งเป็นการส่งงานต่อให้ผู้อื่นในกรณีที่ไม่สามารถทำงานเสร็จได้ทันตามกำหนด

4) มีการนำระบบการจัดการกระแสงานนี้ไปต่อเข้ากับระบบเมล์ขององค์กร เพื่อให้สามารถแจ้งผู้ใช้งานทราบได้ทันทีเมื่อมีงานใหม่เข้ามาในรายการงานของผู้ใช้นั้น

รายการอ้างอิง

[1] Stohr, E.A., Zhao, J.L. Workflow Automation: Overview and Research Issues, 2001.

[2] Muehlen, M.Z. Workflow-based Process Controlling. Germany: Logos Verlag, 2004.

[3] Petkov, S., Oren, E., Haller, A. Aspects in Workflow Management. DERI Technical Report, 2005.

[4] The Workflow Management Coalition. The Workflow Reference Model, 1995.

[5] The Workflow Management Coalition. Process Definition Interface - - XML Process Definition Language, 2005.

[6] Dumas, M., Hofstede, A. UML Activity Diagrams as a Workflow Specification Language. Proceedings of the UML'01 Conference, 2001.

[7] Bastos, R. M., Ruiz, D.D.A. Extending UML Activity Diagram for Workflow Modeling in Production Systems. Proceedings of the 35th Hawaii International Conference on System Sciences, 2002.

[8] Eshuis, R. Semantics and Verification of UML Activity Diagrams for Workflow Modelling. Ph.D. Thesis, University of Twente, 2002.

[9] van der Aalst, W.M.P., van Hee K. Workflow Management – Models, Methods, and Systems. The MIT Press, 2001.

[10] Dutta, S. Design and Development of a Workflow Management System Using UML and C++. Master's Thesis, University of Delhi, 2005.

[11] Object Management Group. Business Process Modelling Notations Specification v1.1, 2008.

[12] Jablonski, S., Bussler, C. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press, 1996.

[13] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P. Workflow Patterns, 2002.

[14] Object Management Group. MOF2.0/XMI Mapping Specification, v2.1, 2005.

[15] Russell, N., van der Aals, W.M.P., ter Hofstede, A.H.M., Wohed P. On the Suitability of UML 2.0 Activity Diagram for Business Process Modelling

Proceedings of the 3rd Asia-Pacific on Conceptual Modelling, 2006.

[16] Schattkowsky, T., Forster, A. On the Pitfalls of UML 2 Activity Modelling Proceedings of MISE'07, 2007.

[17] Guelfi, N., Mamma, A. A Formal Framework to Generate XPDL Specifications from UML Activity Diagrams. Proceedings of SOC'06, 2006.

[18] Jiang, P., Mair, Q., Newman, J. Using UML to Design Distributed Collaborative Workflows: from UML to XPDL Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.

[19] Visual Paradigm International. Visual Paradigm for UML. Available from: http://www.visual-paradigm.com/product/vpuml [2007, Mar 15]

[20] Object Management Group (OMG). Unified Modeling Language: Superstructure, version 2.0, 2004

[21] Microsoft Corporation. 2006. Creating a Data Access Layer [Online]. Available from: http://msdn2.microsoft.com/en-us/library/aa581778.aspx [2007, October 1]

ภาคผนวก

ภาคผนวก ก
เอกสารที่ได้รับการตีพิมพ์

วารสารระดับชาติ

1) Mahatribhop, T. and Vatanawood, W. Development of a Workflow Management System from UML Activity Diagram. Proceedings of the 11th National Computer Science and Engineering Conference (NCSEC 2007). Bangkok, Thailand, November 19-21, 2007.

Development of a Workflow Management System from UML Activity Diagram

Thanawat Mahatribhop and Wiwat Vatanawood
Department of Computer Engineering, Faculty of Engineering
Chulalongkorn University, Bangkok, Thailand
Email: thanawat.m@student.chula.ac.th, wiwatv@gmail.com

Abstract

Currently there exists a numerous number of workflow management systems, both of commercial and open-source type. However, those systems usually apply their own proprietary format and terminology on workflow modelling which causes the interchange of workflow specifications to be very difficult. Standardization efforts in this area, such as the XPDL language proposed by Workflow Management Coalition to be the standard workflow modelling language have essentially failed to gain universal acceptance. Recently, UML activity diagrams, which are generally used to model business activities, have been proposed for defining workflows. Although several research works have been done on the suitability of activity diagrams to workflow modelling, none of such workflow management systems have really been existed. As a proof of concept, we have developed a workflow management system that takes activity diagrams as workflow specifications. In this paper, the architecture and the functionality of the system are described.

Key Words: Workflow, Workflow Management System, UML Activity Diagrams

1. Introduction

Workflow management is a technology for automating and controlling business processes. According to Workflow Management Coalition (WfMC), an industrial standardization body for workflow management, a workflow can be defined as "the computerized facilitation or automation of a business process, in whole or part" [1].

A workflow management system (WfMS) is an information system that supports modelling, execution, management and monitoring of workflows [15]. An important function of WfMSs is to enforce certain rules between business activities [9]. Examples of rules are ordering rules, which specify the sequence of activities, and allocation rules, which state to which actor in the organization a WfMS may allocate an activity. For instance, in a workflow that handles order processing, after an Order Form is completely filled-in and signed by the Order Department, the Accounting Department can then send out an invoice. These rules are specified in a workflow specification. Enforcement of rules in a workflow specification by a WfMS is called enactment. Activities in a workflow may be automated activities, which can be carried out by the system itself, or manual activities, which the system merely notifies the actors who are responsible for working on those activities, anyway the activities have to be done manually [11].

Workflow specifications in which rules are defined can completely be constructed in various ways, or we may say that workflow specifications can be modelled with different languages. One crucial problem in the workflow management technology is that each WfMS in the market uses its own proprietary language, which significantly differs from other languages in terms of concepts, constructs and semantics. This means that it is infeasible to reuse a workflow specification amongst different WfMSs [16]. A standardization effort in this area has then been initiated by WfMC, as the XPDL (XML Process Definition Language) [13] has been proposed to be the standard workflow modelling language. Unfortunately, the language has failed to gain acceptance [12]. Mainly this is because a workflow specification needs to be communicated to different groups of people, like managers, end users and technical staff. Hence, it is desirable that workflow specifications are written in a language that is understandable to all these groups, and preferably be graphical [8]. XPDL is textual-based and has no associated graphical elements so it is pretty difficult to be understood by most people. Moreover, XPDL is somewhat a new language, so even technical staff has to spend significant effort in order to get acquainted with its syntax and semantic.

Recently, another language has been proposed for modelling workflows, namely UML activity diagrams. Unified Modeling Language (UML) is a

de-facto industry standard for representing software system designs. Activity diagrams are graphical languages containing plenty of notations and are used to model business activities. A number of researches have since been conducted on the expressiveness of UML activity diagrams in modeling workflow processes [3, 4, 6, 8, 9, 10, 11], mainly based on the workflow patterns concept [5]. Nonetheless, to the best of our knowledge, so far there has been no activity-diagram-based workflow management system that really existed. In this paper, we present the architecture and functionality of the ADWL (Activity Diagram as a Workflow Language) system, which is a WfMS that takes standard UML 2.0 activity diagrams, without any special extension, as the workflow specifications and enact them.

The remainder of this paper proceeds as the followings. Section 2 provides the basic concepts and definitions in the workflow technology. Section 3 discusses how activity diagram notations are mapped into workflow elements in the ADWL system. Section 4 briefly describes the ADWL architecture. Section 5 provides an example of workflow enactment in the ADWL. Section 6 concludes the paper.

2. Basic workflow concepts

In the previous section, we already explained some concepts of workflows. This section provides more terminology of workflow products [7].

A business process consists of a sequence of activities. An activity is a discrete process step performed either by a machine or human. An activity may consist of one or more tasks. A workflow is the automation of a business process in whole or in part, during which documents, information or tasks are passed from one participant to another according to a set of procedural rules defined in a workflow specification (or process definition). A set of tasks to be performed by a user in a WfMS is called a worklist. The worklist is prepared by the WfMS and displayed to the workflow participant who is to perform the task according to the workflow specification. The individual tasks on the worklist are called work items. What is actually happening in a WfMS when executing a workflow specification is represented through a workflow instance. For example, in the Process Order workflow, a new Process Order instance has to be created every time a new order comes in.

3. UML Activity Diagrams as workflow specifications in ADWL

UML 2.0 activity diagram notations have a vast majority of features but most of which make no sense for workflow definition. Therefore, we select a list of recommended activity diagram features for workflow definition in the ADWL system. In UML 2.0, activity diagram definition is structured into packages with ever growing complexity which are Fundamental, Basic, Intermediate, Complete, Structured, CompleteStructured, and ExtraStructured [2]. The ADWL system supports most of the concepts in every package except for those in the ExtraStructured package where the Exception Handling concept is presented, as it is not directly relevant to workflow modeling. Basically, the required aspects of workflow specifications are as follows:

- Control flows – support for branching, decisions, parallelization, synchronization and loops in order to specify ordering rules
- Data flows
- Different types of tasks such as automated or manual
- Resource management – the definition of human task performers

In this section, we detail the uses of UML 2.0 activity diagram notations in the ADWL system for workflow enactment; they are presented in Figure 1. However, the detail of each notation is not described here as it goes beyond the scope of this paper.

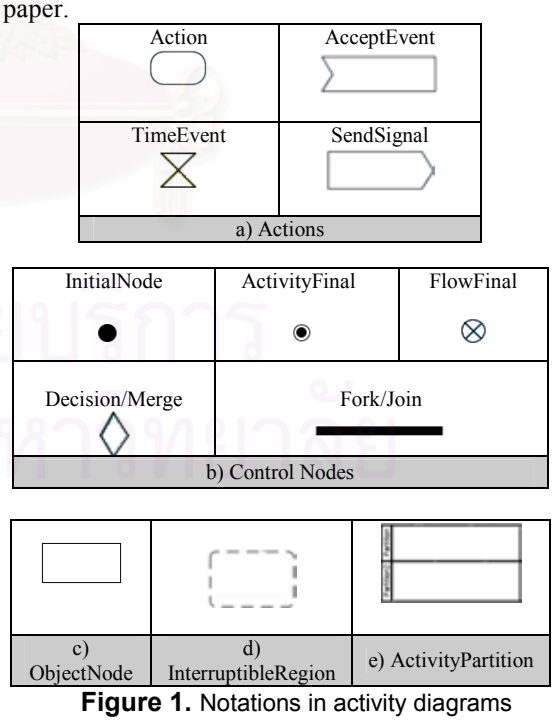


Figure 1. Notations in activity diagrams

We start with Actions which are equivalent to tasks in the workflow domain. In this work, only the generic Action notation, AcceptEventAction, AcceptTimeEventAction, SendSignalAction, CallBehaviorAction are supported. Note that CallBehaviorAction is the notation representing the Activities concept, in which it is composed of actions and/or other activities. The ADWL system also recognizes action constraints which are symbolized in activity diagrams as localPreconditions and localPostconditions.

Control nodes in UML 2.0 activity diagrams have been analyzed in previous researches and found to be truly meaningful for defining workflows as most workflow patterns such as parallel split, synchronization, looping can be articulated [10]. The ADWL system supports all basic control nodes that are presented in Figure 1 as well as ActivityPartition that defines the participant to be working on each task. The InterruptibleRegion notation can also be used to define the area where events may occur and cause the ongoing tasks in the area to be terminated.

4. ADWL architecture

This section describes the overall architecture and main functionality of the ADWL system using Figure 2.

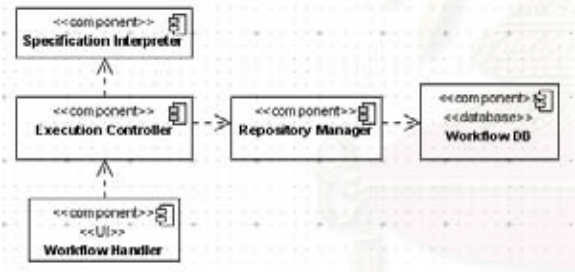


Figure 2. ADWL architecture

The ADWL system is a web application which can be accessed through web browsers. It enacts a workflow specification by enforcing the sequence and conditions of tasks defined in the workflow specifications, which have to be modelled with UML 2.0 activity diagrams and saved in the XMI format. Note that XMI is a textual representation of UML diagrams, based on XML technology, and emerging as a standard interchange format of UML diagrams [14]. Most of UML tools nowadays have a feature to export the diagrams into XMI files. Unfortunately, different tools implement the concept in a different way, which means the exported XMI files are generally unlike other different tools. The ADWL system recognizes files in the format of XMI 2.1 that are exported from Visual Paradigm for UML (http://www.visual-paradigm.com).

The core of the system is formed by the Execution Controller. Once the system obtains an XMI file from a user via the Workflow Handler web interface, the Execution Controller invokes the Specification Interpreter in order to parse the file to extract the rules defined in the workflow specification, then stores the data in the database through the Repository Manager. The system can stock up many workflow specifications and can enact more than one workflow at the same time. Table 1 presents the list of tables in ADWL's database that stores the workflow rules which are used to enact workflow processes.

After that when a user logs in through the Workflow Handler web interface, the Execution Controller retrieves the data from the database and populates the web screens to the users, as well as updates the data and saves back into the database. An example of workflow enactment in the ADWL system will be described in section 5.

Table 1. Tables in ADWL's database

Table	Definition
Workflow	Holds the definition of each workflow process
Task	Holds the details for each task in all the workflows known by the system
Flow	Holds the details for each task flow such as the source and target task.
Instance	Holds the details for each workflow instance in each workflow
Worklist	Holds the details for each workflow participant's worklist
Object	Holds the details for each object; either the input or output of each task
Workflow Participant	Holds the details for each workflow participant
User	Holds the details for each user of the system

5. Workflow enactment in the ADWL system

Applying the concept of manual activities, the role of the ADWL system is to notify the users of their worklists according to the rules defined in the workflow specifications. The system clearly displays the status of each task either New, In

Progress, or Done as well as allows the user to view the history and the input and output of each task (if any).

User configuration is required before enactment. Specifically, the mappings between users of the system and workflow participants specified in workflow specifications have to be defined. This step is needed such that the system knows who are to carry out what tasks and consequently can update the worklist of each user correctly. One user may be assigned to more than one workflow participant.

It is also realized that business processes commonly keep changing overtime and thus it is demanded that a WfMS be capable of supporting the changes while the workflow is being carried out [17]. The ADWL system supports the Edit Workflow functionality to alter workflow specifications provided that the modification does not cause discrepancy to the ongoing instances of that modified workflows. Moreover, the system allows the user to undo particular tasks on condition that the subsequent tasks have yet to start. More details can be found later in this section from the example.

Event is another concept in UML activity diagrams which is used to define actions to execute when certain events occur. This can be modelled with the AcceptEvent, the TimeEvent, and the InterruptibleRegion notations. The ADWL system allows the user to raise events for each workflow instance as defined in the specification.

With all the aforementioned functionality, the ADWL system can be practically used to enact workflows. Below is an example of an enactment, using the Process Order workflow illustrated in Figure 3 as an example.

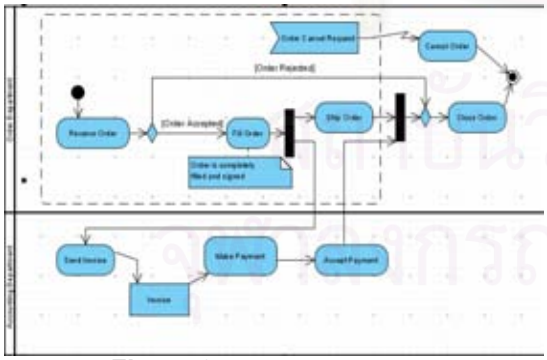


Figure 3. Process Order workflow

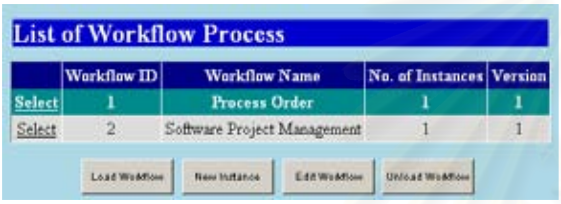
The workflow has two workflow participants, which are Order Department and Accounting Department. The process commences when Order Department receives an order and then proceeds to either the Fill Order task if the order is accepted or the Close Order task if the order is rejected. At the Send Invoice task, it produces an Invoice as the output which in turn will be the input of the Make Payment task. Once there is an Order Cancel Request coming in, the ongoing Fill Order or Ship Order will be terminated in order to perform the Cancel Order task instead.

Figure 4 shows some current screenshots of the ADWL system that is still under development. Figure 4a illustrates the Workflow Handler screen where the list of workflow processes and their instances are displayed. A user then may load a new workflow process into the system by specifying the path to an XMI file; or unload a workflow process out of the system. Besides, the user can edit existing workflows as long as the modification does not bring about discrepancy. For example, at the First Order instance of the Process Order workflow, the Receive Order task has been completely done and the Fill Order task is being carried out, the system does not allow to remove the Receive Order task from the Process Order specification as discrepancy will arise. If an alteration has been done successfully, the system will increment the version of the workflow specification so that we can distinguish the instances that have been done according to the original specification from the instances that work according to the modified specification. The user can also view details or edit individual instance of each workflow as illustrated in Figure 4b where the details of each instance such as the current status (either New, In Progress, Done or Cancelled), the start time and finish time, as well as the input and output are presented. As for undoing a task, it is allowed to be done if subsequent tasks have not started. For instance, in Figure 4c the Receive Order task has been done and the subsequent task, Fill Order, has already started (the status is In Progress); in this case the system allows to undo only the Fill Order task. This is to avoid conflicts with the specification.

Figure 4d shows the worklist of a user mapped to Order Department that will be displayed once the user logs in. The status of each task is New when it first comes into the worklist (the preceding task is done) and will be changed to In Progress once the user presses the Start button, and eventually will be Done upon pressing Register Completion (the task will not be shown in the worklist screen at the next log-in). The user can also view the details like pre- and post-conditions, and the required input and output of the tasks. Note that the input and output are files provided by users, and will be stored in the ADWL system. Upon a completion registration, in

case that the next node is a Decision node bearing guard conditions, the system will ask the user to select which condition is true so that the system knows what the next task is. This is illustrated in Figure 4e that appears when the Receive Order task is done.

As illustrated in Figure 4b, the user can raise events by pressing the Raise Event button and select the event to raise as in Figure 4f (in this example, only one event is defined in the specification so only one event is available). If the Order Cancel Request event is raised, all the ongoing tasks in the Interruptible region will be updated with the status Cancelled (displayed in the corresponding user's worklist) and the ongoing task will turn to Cancel Order according to the workflow specification.

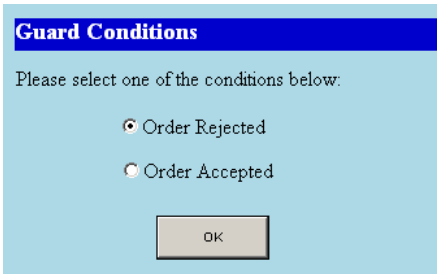


(a)

(b)

(c)

(d)



(e)

(f)

Figure 4. Some screenshots of the ADWL system

6. Conclusions

In this paper we presented how workflow specifications modelled with UML 2.0 activity diagrams are enacted in the ADWL system. Although the implementation has not completed, we consider it as a proof of concept that UML activity diagrams can be used to model workflows and be enacted. The contributions of this paper are approaches to handle the notations in UML activity diagrams in the sense of workflow. A challenging problem is to be more generic with the input XMI files that represent activity diagrams. This may be achieved by allowing users to provide the system with a document format file together with an XMI file. Another possible future work is to integrate the ADWL system with a mail server in the organization in order that a user will be notified of a new task added to his/her worklist immediately and automatically.

12. References

[1] The Workflow Management Coalition, "Terminology and glossary," 1999.

[2] Object Management Group (OMG), "Unified Modeling Language: Superstructure, version 2.0," 2004.

[3] Dumas, M., Hofstede ,A., "UML Activity Diagrams as a Workflow Specification Language," 2001

[4] Bastos, R. M., Ruiz, D.D.A., "Extending UML Activity Diagram for Workflow Modeling in Production Systems," 2002

[5] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P., "Workflow Patterns," 2002.

[6] Wohed P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N., "Pattern-based Analysis of UML Activity Diagrams," 2005.
[7] Stohr, E.A., Zhao, J.L., "Workflow Automation: Overview and Research Issues," 2001.
[8] Eshuis, R., Weiringa, R., "A Formal Semantics for UML Activity Diagrams – Formalising Workflow Model," 2000.
[9] Eshuis, R. "Semantics and Verification of UML Activity Diagrams for Workflow Modelling," 2002.
[10] Russell, N., van der Aals, W.M.P., ter Hofstede, A.H.M., Wohed P., "On the Suitability of UML 2.0 Activity Diagram for Business Process Modelling," 2006.
[11] Kalnins, A., Vitolins, V., "Use of UML and Model Transformations for Workflow Process Definitions," 2006.
[12] van der Aalst, W.M.P., Aldred, L., Dumas, M., ter Hofstede, A.H.M., "Design and implementation of the YAWL system," 2003.
[13] The Workflow Management Coalition., "Process Definition Interface - - XML Process Definition Language," 2005.
[14] Object Management Group (OMG), "XMI Metadata Interchange (XMI) Specification," 2003.
[15] Petkov, S., Oren, E., Haller, A., "Aspects in Workflow Management," 2005.
[16] zur Muehlen, M., "Workflow-based Process Controlling," 2004.
[17] Dias, P., Vieira, P., Rito-Silva, A., "Dynamic Evolution in Workflow Management Systems," 2003

ภาคผนวก ข
เอ็กซ์เอ็มไอแท็กของแผนภาพกิจกรรม

รายละเอียดของเอ็กซ์เอ็มไอแท็กของแผนภาพกิจกรรมที่เอ็กซ์พอร์ตมาจากเครื่องมือ Visual Paradigm มีรายละเอียดดังตาราง ข.1 ทั้งนี้โดยส่วนใหญ่แท็กที่แสดงแต่ละโหนดจะประกอบด้วยแอททริบิวต์ (Attribute) คือ name, xmi:id, xmi:type เพื่อใช้ระบุชื่อโหนด, รหัสโหนด และชนิดของโหนดตามลำดับ หากชนิดของโหนดเป็น “uml:PseudoState” โหนดนั้นจะมีแอททริบิวต์ kind เพิ่มขึ้นมา ไว้ระบุประเภทของโหนด

ตาราง ข.1 เอ็กซ์เอ็มไอแท็กของแผนภาพกิจกรรม

ข้อมูล	แท็ก
โหนด Action	`<node name="Investigate Problem" xmi:id="UUBvsWiD.AAAAQFx" xmi:type="uml:CallBehaviorAction">` `</node>`

โหนด Activity	`<ownedMember isReadOnly="false" isReentrant="false" isSingleExecution="false" name="Activity1"`
	`postcondition="" precondition="" xmi:id="liLscWiD.AAAAQEu" xmi:type="uml:Activity">`
	`</ownedMember>`

โหนด AcceptEventAction	`<node name="Event1" xmi:id="u_iGcWiD.AAAAQKj" xmi:type="uml:AcceptEventAction">` `</node>`
โหนด SendSignalAction	`<node name="Event2" xmi:id="zQlGcWiD.AAAAQUL" xmi:type="uml:SendSignalAction">` `</node>`
โหนด Input Pin	`<argument name="A" ordering="FIFO" type="null_id" xmi:id="pPsxcWiD.AAAAQMi" xmi:type="uml:InputPin">` `</argument>`
โหนด Output Pin	`<result name="B" ordering="FIFO" type="null_id" xmi:id="HgkxcWiD.AAAAQHH" xmi:type="uml:OutputPin">` `</result>`

โหนด ActivityParameter	`<node isControlType="false" name="Parameter2" ordering="FIFO" type="null_id" xmi:id="2PnscWiD.AAAAQJY"`
	`xmi:type="uml:ActivityParameterNode">`
	`</node>`

โหนด Object	`<ownedMember name="C" ordering="FIFO" type="null_id" xmi:id="cOPe8WiD.AAAASGx"`
	`xmi:type="uml:CentralBufferNode">`
	`</ownedMember>`

โหนด Initial	`<node kind="initial" name="InitialNode" xmi:id="IBKvsWiD.AAAAQD5" xmi:type="uml:Pseudostate">` `</node>`
โหนด ActivityFinal	`<node kind="final" name="ActivityFinalNode" xmi:id="X89dCWiD.AAAAQId" xmi:type="uml:Pseudostate">` `</node>`
โหนด FlowFinal	`<node kind="final" name="FlowFinalNode" xmi:id="Qk4t8WiD.AAAARne" xmi:type="uml:Pseudostate">` `</node>`

ตาราง ข.1 เอ็กซ์เอ็มไอแท็กของแผนภาพกิจกรรม (ต่อ)

ข้อมูล	แท็ก
โหนด ActivityPartition	<node name="Swimlane" xmi:id="wEesiWiD.AAAAQFv" xmi:type="activitySwimlane2"> <verticalPartition xmi:idref="gkesiWiD.AAAAQF1"/> <ownedMember name="Role1" xmi:id="gkesiWiD.AAAAQF1" xmi:type="uml:ActivityPartition"> <containedNode xmi:idref="blwS8WiD.AAAAQc8"/> <containedNode xmi:idref="OP3i8WiD.AAAAQbD"/> </ownedMember> </node> (ในที่นี้มีหนึ่งบทบาทชื่อ Role1 ซึ่งมี 2 โหนดอยู่ภายใน)
โหนด Fork	<ownedMember kind="fork" name="ForkNode" xmi:id="xrPfsWiD.AAAAQdm" xmi:type="uml:Pseudostate"> </ownedMember>
โหนด Join	<ownedMember kind="join" name="JoinNode2" xmi:id="BCY18WiD.AAAAQ7G" xmi:type="uml:Pseudostate"> </ownedMember>
โหนด Decision	<node kind="junction" name="DecisionNode3" xmi:id="zZIDCWiD.AAAAQsl" xmi:type="uml:Pseudostate"> </node>
โหนด Merge	<node kind="junction" name="MergeNode" xmi:id="wIZb8WiD.AAAASRH" xmi:type="uml:Pseudostate"> </node> (สังเกตว่าโหนด Merge และ Decision ไม่สามารถแยกความแตกต่างได้โดยตรง จำเป็นต้องดูจำนวนเส้นเชื่อมขาเข้าและขาออก ถ้าเป็นโหนด Merge จะมีเส้นเชื่อมขาออกแค่หนึ่งเส้นเชื่อม ในขณะที่โหนด Decision จะมีเส้นเชื่อมขาเข้าหนึ่งเส้นเชื่อม)
โหนด InterruptibleActivityRegion	<node name="Region" xmi:id="6zr01YiD.AAAAQGN" xmi:type="InterruptibleActivityRegion"> </node>
ExceptionHandler	<node exceptionInput="xQXM1YiD.AAAAQi3" handlerBody="BC8M1YiD.AAAAQPY" name="" xmi:id="pz3M1YiD.AAAAQja" xmi:type="uml:ExceptionHandler"/>

ภาคผนวก ค
รายละเอียดประกอบยูสเคสของระบบ

รายละเอียดประกอบยูสเคสทั้งหมดของระบบแสดงในตาราง ค.1 – ค.14

ตาราง ค.1 รายละเอียดประกอบยูสเคสล็อกอิน / ล็อกเอาท์จากระบบ

Use Case Name: ล็อกอิน / ล็อกเอาท์จากระบบ
Primary Actor: ผู้บริหารระบบ, ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบหรือผู้ใช้งานต้องการล็อกอินเข้าสู่ระบบหรือล็อกเอาท์ออกจากระบบ
Normal Flow of Events: 1. ล็อกอิน 2. บนหน้าเว็บ 'Login.aspx' ผู้บริหารระบบหรือผู้ใช้งานใส่ชื่อและรหัสผ่าน 3. ผู้บริหารระบบเลือกปุ่ม 'Log In' 2. ล็อกเอาท์ 1. บนหน้าเว็บ 'Workflow.aspx' ผู้บริหารระบบหรือผู้ใช้งานเลือกเมนู 'Logout'
Alternate/Exceptional Flows: 1-E ในกรณีล็อกอิน ถ้าชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง ให้ทำการล็อกอินใหม่โดยใส่ชื่อผู้ใช้และรหัสผ่านให้ถูกต้อง

ตาราง ค.2 รายละเอียดประกอบยูสเคสลงทะเบียนผู้ใช้ / จับคู่ผู้ใช้กับบทบาท

Use Case Name: ลงทะเบียนผู้ใช้ / จับคู่ผู้ใช้กับบทบาท
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการลงทะเบียนผู้ใช้ใหม่ในระบบ และจับคู่ผู้ใช้กับบทบาท

ตาราง ค.2 รายละเอียดประกอบยูสเคสลงทะเบียนผู้ใช้ / จับคู่ผู้ใช้กับบทบาท (ต่อ)

Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกปุ่ม 'Create User' บนหน้าเว็บ 'Workflow.aspx' 2. ผู้บริหารระบบกรอกรายละเอียดของผู้ใช้ใหม่ 3. ผู้บริหารระบบเลือกปุ่ม 'Create User' 4. ผู้บริหารระบบเลือกบทบาทที่จะจับคู่กับผู้ใช้ 5. ผู้บริหารระบบเลือกปุ่ม 'OK'
Alternate/Exceptional Flows: 1-E ถ้าชื่อผู้ใช้ซ้ำกับผู้ใช้เดิมในระบบ หรือกรอกรายละเอียดไม่ครบถ้วน ผู้บริหารระบบแก้ไขชื่อผู้ใช้และกรอกรายละเอียดให้ครบถ้วน

ตาราง ค.3 รายละเอียดประกอบยูสเคสเพิ่มกระแสงานใหม่

Use Case Name: เพิ่มกระแสงานใหม่
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการเพิ่มกระแสงานใหม่เข้าสู่ระบบ โดยกระแสงานต้องถูกจัดเก็บมาในรูปแบบมาตรฐานเอ็กซ์เอ็มไอ
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกปุ่ม 'Browse' บนหน้าเว็บ 'Workflow.aspx' และเลือกไปยังไฟล์เอ็กซ์เอ็มไอที่ต้องการ ซึ่งได้มาจากการเอ็กซ์พอร์ตด้วยเครื่องมือ Visual Paradigm for UML 2. ผู้บริหารระบบเลือกปุ่ม 'OK'

ตาราง ค.3 รายละเอียดประกอบยูสเคสเพิ่มกระแสงานใหม่ (ต่อ)

Alternate/Exceptional Flows: 1-E ถ้าชื่อกระแสงานที่เพิ่มซ้ำกับชื่อกระแสงานในระบบ ผู้บริหารระบบกลับไปแก้ไขชื่อกระแสงานด้วยเครื่องมือ Visual Paradigm for UML แล้วเริ่มทำยูสเคสเพิ่มกระแสงานใหม่อีกครั้งหนึ่ง 2-E ถ้ากระแสงานไม่ถูกต้องซึ่งระบบจะเตือนให้ทราบ ผู้บริหารระบบกลับไปแก้ไขกระแสงานด้วยเครื่องมือ Visual Paradigm for UML แล้วเริ่มทำยูสเคสเพิ่มกระแสงานใหม่อีกครั้งหนึ่ง

ตาราง ค.4 รายละเอียดประกอบยูสเคสแก้ไขกระแสงานเดิม

Use Case Name: แก้ไขกระแสงานเดิม
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการแก้ไขกระแสงานเดิมที่มีอยู่ในระบบ
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกกระแสงานที่ต้องการแก้ไขและเลือกปุ่ม ‘Modify’ บนหน้าเว็บ ‘Workflow.aspx’ 2. ผู้บริหารระบบเลือกไฟล์เอ็กซ์เอ็มไอใหม่ และเลือกปุ่ม ‘OK’
Alternate/Exceptional Flows: 1-E ถ้ากระแสงานที่แก้ไขไม่ถูกต้องตามเงื่อนไขของการแก้ไขซึ่งระบบจะเตือนให้ทราบ ผู้บริหารระบบกลับไปแก้ไขกระแสงานด้วยเครื่องมือ Visual Paradigm for UML แล้วเริ่มทำยูสเคสแก้ไขกระแสงานเดิมอีกครั้งหนึ่ง

ตาราง ค.5 รายละเอียดประกอบยูสเคสลบกระแสงาน

Use Case Name: ลบกระแสงาน
Primary Actor: ผู้บริหารระบบ

ตาราง ค.5 รายละเอียดประกอบยูสเคสลบกระแสงาน (ต่อ)

Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการลบกระแสงานเดิมที่มีอยู่ในระบบ
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกกระแสงานที่ต้องการลบ และเลือกปุ่ม 'Remove' บนหน้าเว็บ 'Workflow.aspx'
Alternate/Exceptional Flows: -

ตาราง ค.6 รายละเอียดประกอบยูสเคสสร้างอินสแตนซ์ใหม่

Use Case Name: สร้างอินสแตนซ์ใหม่
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการสร้างอินสแตนซ์ใหม่ของกระแสงาน
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกกระแสงานที่ต้องการสร้างอินสแตนซ์ใหม่บนหน้าเว็บ 'Workflow.aspx' 2. ผู้บริหารระบบเลือกปุ่ม 'New Instance' 3. ผู้บริหารระบบใส่ชื่ออินสแตนซ์ใหม่ที่ต้องการ และเลือกปุ่ม 'OK'
Alternate/Exceptional Flows: 1-E ถ้าชื่ออินสแตนซ์ใหม่ซ้ำกับชื่ออินสแตนซ์ที่มีอยู่แล้วของกระแสงานนั้น ผู้บริหารระบบต้องระบุชื่อของอินสแตนซ์ใหม่ และเลือกปุ่ม 'OK'

ตาราง ค.7 รายละเอียดประกอบยูสเคสลบอินสแตนซ์

Use Case Name: ลบอินสแตนซ์
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการสร้างลบอินสแตนซ์ออกจากระบบ
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วเลือกอินสแตนซ์ที่ต้องการลบบนหน้าเว็บ ‘Instance.aspx’ 2. ผู้บริหารระบบเลือกปุ่ม ‘Remove’
Alternate/Exceptional Flows: -

ตาราง ค.8 รายละเอียดประกอบยูสเคสดูล็อก

Use Case Name: ดูล็อก
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการดูล็อกของระบบ
Normal Flow of Events: 1. ผู้บริหารระบบที่ล็อกอินเข้าสู่ระบบแล้วคลิก ‘Log.txt’ ของกระแสงานที่ต้องการดูล็อกบนหน้าเว็บ ‘Workflow.aspx’

ตาราง ค.8 รายละเอียดประกอบยูสเคสดูล็อก (ต่อ)

Alternate/Exceptional Flows: 1-A ถ้าผู้บริหารระบบต้องการดูล็อกของอินสแตนซ์ให้ไปที่หน้าเว็บ ‘Instance.aspx’ และคลิก ‘Log.txt’ ของอินสแตนซ์ที่ต้องการดูล็อก 2-A ถ้าผู้บริหารระบบต้องการจัดเก็บล็อก ให้คลิกขวาที่ ‘Log.txt’ และเลือกเมนู ‘Save Target As…’ 3-A ผู้ใช้งานสามารถดูล็อกของอินสแตนซ์ได้ที่หน้า ‘Instance.aspx’ และจัดเก็บล็อกได้โดยวิธีเดียวกับผู้บริหารระบบ

ตาราง ค.9 รายละเอียดประกอบยูสเคสบันทึกการเริ่มทำงาน

Use Case Name: บันทึกการเริ่มทำงาน
Primary Actor: ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้ใช้งานต้องการบันทึกการเริ่มทำงาน
Normal Flow of Events: 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกงานที่ต้องการบันทึกการเริ่มทำงานบนหน้าเว็บ ‘Worklist.aspx’ 2. ผู้ใช้งานเลือกปุ่ม ‘Start’
Alternate/Exceptional Flows: -

ตาราง ค.10 รายละเอียดประกอบยูสเคสบันทึกการทำงานเสร็จ

Use Case Name: บันทึกการทำงานเสร็จ
Primary Actor: ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้ใช้งานต้องการบันทึกการทำงานเสร็จ

ตาราง ค.10 รายละเอียดประกอบยูสเคสบันทึกการทำงานเสร็จ (ต่อ)

Normal Flow of Events: 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกงานที่ต้องการบันทึกการทำงานเสร็จซึ่งถูกบันทึกการเริ่มทำงานไว้แล้ว ที่หน้าเว็บ 'Worklist.aspx' 2. ผู้ใช้งานเลือกปุ่ม 'Finish'
Alternate/Exceptional Flows: 1-E ถ้างานนั้นมีเอกสารที่ยังไม่ถูกอัพโหลด ซึ่งอาจเป็นอินพุตหรือเอาท์พุต ระบบจะไม่อนุญาตให้ผู้ใช้งานบันทึกการทำงานเสร็จ ให้ผู้ใช้งานอัพโหลดเอกสารให้ครบ และทำยูสเคสบันทึกการทำงานเสร็จอีกครั้ง

ตาราง ค.11 รายละเอียดประกอบยูสเคสระบุการเกิดเหตุการณ์

Use Case Name: ระบุการเกิดเหตุการณ์
Primary Actor: ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้ใช้งานต้องการระบุการเกิดเหตุการณ์
Normal Flow of Events: 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกอินสแตนซ์ที่ต้องการระบุการเกิดเหตุการณ์ ที่หน้าเว็บ 'Instance.aspx' 2. ผู้ใช้งานเลือกเหตุการณ์ที่ต้องการ แล้วเลือกปุ่ม 'Trigger Event'
Alternate/Exceptional Flows: -

ตาราง ค.12 รายละเอียดประกอบยูสเคสอัพโหลด / ดาวน์โหลดเอกสาร

Use Case Name: อัพโหลด / ดาวน์โหลดเอกสาร
Primary Actor: ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้ใช้งานต้องการอัพโหลด / ดาวน์โหลดเอกสาร
Normal Flow of Events: 1. อัพโหลด 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกงานที่ต้องการอัพโหลดเอกสารที่หน้าเว็บ 'Worklist.aspx' 2. ผู้ใช้งานเลือกอินพุตหรือเอาท์พุตที่ต้องการอัพโหลดเอกสาร 3. ผู้ใช้เลือกปุ่ม 'Browse' และเลือกเอกสารที่ต้องการอัพโหลด จากนั้นผู้ใช้เลือกปุ่ม 'Upload' 2. ดาวน์โหลด 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกงานที่ต้องการดาวน์โหลดเอกสารที่หน้าเว็บ 'Worklist.aspx' 2. ผู้ใช้งานเลือกอินพุตหรือเอาท์พุตที่ต้องการดาวน์โหลดเอกสาร 3. ผู้ใช้งานคลิกขวาที่เอกสารที่ต้องการดาวน์โหลดและเลือกเมนู 'Save Target As...' เพื่อจัดเก็บเอกสาร
Alternate/Exceptional Flows: -

ตาราง ค.13 รายละเอียดประกอบยูสเคสดูข้อกำหนดกระแสงาน

Use Case Name: ดูข้อกำหนดกระแสงาน
Primary Actor: ผู้บริหารระบบ
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้บริหารระบบต้องการดูข้อกำหนดของกระแสงาน

ตาราง ค.13 รายละเอียดประกอบยูสเคสดูข้อกำหนดกระแสงาน (ต่อ)

Normal Flow of Events: 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วคลิก ‘XMI’ ของกระแสงานที่ต้องการ ระบบจะเปิดไฟล์เอ็กซ์เอ็มไอที่เป็นข้อกำหนดของกระแสงานนั้น
Alternate/Exceptional Flows: 1-A ถ้าผู้บริหารระบบต้องการจัดเก็บข้อกำหนดของกระแสงาน ให้คลิกขวาที่ ‘XMI’ และเลือกเมนู ‘Save Target As…’

ตาราง ค.14 รายละเอียดประกอบยูสเคสทำกลับการทำงาน

Use Case Name: ทำกลับการทำงาน
Primary Actor: ผู้ใช้งาน
Brief Description: ยูสเคสนี้อธิบายการทำงานของระบบเมื่อผู้ใช้งานต้องการทำกลับการทำงาน
Normal Flow of Events: 1. ผู้ใช้งานที่ล็อกอินเข้าสู่ระบบแล้วเลือกงานที่ต้องการทำกลับการทำงานที่หน้าเว็บ ‘Instance.aspx’ แล้วคลิก ‘Undo’
Alternate/Exceptional Flows: 1-E ถ้างานถัดไปถูกบันทึกเริ่มทำงานหรือถูกบันทึกการทำงานเสร็จสิ้นไปแล้ว (สถานะไม่เป็น “New”) ระบบจะไม่อนุญาตให้ผู้ใช้งานทำกลับการทำงาน

ภาคผนวก ง
รายละเอียดของข้อความเตือนผู้ใช้

รายละเอียดของข้อความเตือนผู้ใช้กรณีที่ผู้บริหารระบบใส่แผนภาพกิจกรรมที่ไม่ถูกต้อง และกรณีที่ทำการแก้ไขกระแสงานแบบไม่ถูกต้อง มีรายละเอียดดังตาราง ง.1

ตารางที่ ง.1 รายละเอียดของข้อความเตือนผู้ใช้

กรณี	ข้อความเตือนผู้ใช้
แผนภาพกิจกรรมมีสัญกรณ์ AcceptTimeEventAction	"The system does not support AcceptTimeEventAction"
แผนภาพกิจกรรมมีสัญกรณ์ CentralBuffer	"The system does not support CentralBufferNode"
แผนภาพกิจกรรมมีสัญกรณ์ DataStore	"The system does not support DataStoreNode"
แผนภาพกิจกรรมมีสัญกรณ์ ExpansionRegion	"The system does not support ExpansionRegion"
แผนภาพกิจกรรมมีแผนภาพย่อย	"The system does not support sub-diagram"
แผนภาพกิจกรรมมีการใช้สเตอริโอไทพ์	"The system does not support Stereotype"
ไฟล์เอ็กซ์เอ็มไอไม่ได้เอ็กพอร์ตมาจากเครื่องมือ Visual Paradigm for UML	"Invalid XMI: The exporter is not Visual Paradigm for UML"
แผนภาพกิจกรรมมีโหนดที่ไม่ได้อยู่ในโหนด ActivityPartition	"Invalid workflow: Some Action or Activity nodes are not in a Partition"
แผนภาพกิจกรรมไม่มีโหนด Initial	"Invalid workflow: InitialNode not found"
แผนภาพกิจกรรมไม่มีโหนด ActivityFinal	"Invalid workflow: ActivityFinalNode not found"
โหนด Activity มีโหนดอื่นอยู่ภายใน	"Invalid workflow: Activity nodes must not contain other nodes inside"
โหนด AcceptEventAction หรือ SendSignalAction มีอินพุตหรือเอาท์พุต	"Invalid workflow: AcceptEvent/SendSignalEvent must not have inputs or outputs"
โหนด Action, Activity, SendSignalAction ไม่มีเส้นเชื่อมขาเข้า	"Invalid workflow: Action nodes (apart from Event) must have an incoming flow"
โหนด Action, Activity, SendSignalAction, AcceptEventAction มีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	"Invalid workflow: Action nodes cannot have multiple incoming flows"
โหนด Action, Activity, SendSignalAction, AcceptEventAction ไม่มีเส้นเชื่อมขาออก	"Invalid workflow: Action nodes must have an outgoing flow"

ตารางที่ ง.1 รายละเอียดของข้อความเตือนผู้ใช้ (ต่อ)

โหนด Action, Activity, SendSignalAction, AcceptEventAction มีมากกว่าหนึ่งเส้นเชื่อมขาออก	"Invalid workflow: Action nodes cannot have multiple outgoing flows"
โหนด Input Pin, Output Pin, ActivityParameter, Object ไม่แสดงชื่อ (เป็นค่าว่างเปล่า)	"Invalid workflow: Pin name cannot be blank"
โหนด Input Pin, Output Pin, ActivityParameter, Object มีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	"Invalid workflow: InputPin cannot have multiple incoming flows"
โหนด Input Pin, Output Pin, ActivityParameter, Object มีมากกว่าหนึ่งเส้นเชื่อมขาออก	"Invalid workflow: OutputPin cannot have multiple outgoing flows"
โหนด Object ต่อกับโหนด Initial โดยตรง	"Invalid workflow: InitialNode cannot connect to ObjectNode"
โหนด Initial ไม่มีเส้นเชื่อมขาออก	"Invalid workflow: InitialNode must have an outgoing flow"
โหนด Initial มีมากกว่าหนึ่งเส้นเชื่อมขาออก	"Invalid workflow: InitialNode cannot have multiple outgoing flows"
โหนด FlowFinal, ActivityFinal ไม่มีเส้นเชื่อมขาเข้า	"Invalid workflow: FinalNode must have an incoming flow"
โหนด Action, Activity มีเส้นเชื่อมขาเข้าที่ต่างชนิดกัน หรือเส้นเชื่อมขาออกที่ต่างชนิดกัน	"Invalid workflow: All flows of Actoin/Activity must be of the same type"
โหนด Fork ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาเข้า	"Invalid workflow: Fork node must have only one incoming flow"
โหนด Fork, Join, Decision, Merge มีเส้นเชื่อมขาเข้าต่างชนิดกับเส้นเชื่อมขาออก	"Invalid workflow: All flows of Decision/Merge/Fork/Join must be of the same type"
โหนด Join ไม่มีหรือมีมากกว่าหนึ่งเส้นเชื่อมขาออก	"Invalid workflow: Join node must have only one outgoing flow"
โหนด Join มีเส้นเชื่อมขาเข้าที่เป็น Object Flow แต่ไม่มีเส้นเชื่อมขาออกที่เป็น Object Flow	"Invalid workflow: If a Join node has an incoming object flow, it must have an outgoing object flow"
โหนด Fork , Join, Decision, Merge มีเส้นเชื่อมขาเข้าหรือเส้นเชื่อมขาออกที่เป็น Object Flow โดยที่วัตถุนั้นไม่ได้เป็นอินพุตหรือเอาท์พุตของงานใดๆ	"Invalid workflow: A Control node must not connect to an Object node that has no incoming flow "
การแก้ไขกระแสงานมีการเปลี่ยนคุณสมบัติของงานบางงานในอินสแตนซ์ที่ยังดำเนินการอยู่ของกระแสงานนั้น	"Failed to modify workflow: Some attributes of an ongoing/finished task are modified"
การแก้ไขกระแสงานมีการเปลี่ยนโหนดก่อนหน้าของงานในอินสแตนซ์ที่ยังดำเนินการอยู่	"Failed to modify workflow: Incoming edges of an ongoing/finished task are modified"

ตารางที่ ง.1 รายละเอียดของข้อความเตือนผู้ใช้ (ต่อ)

การแก้ไขกระแสงานมีการเปลี่ยนอินพุตของงานในอินสแตนซ์ที่ยังดำเนินการอยู่	"Failed to modify workflow: Inputs of an ongoing/finished task are modified"
การแก้ไขกระแสงานมีการเปลี่ยนโหนดต่อท้ายของงานที่ทำเสร็จไปแล้วในอินสแตนซ์ที่ยังดำเนินการอยู่	"Failed to modify workflow: Outgoing edges of a finished task are modified"
การแก้ไขกระแสงานมีการเปลี่ยนเอาท์พุตของงานที่ทำเสร็จไปแล้วในอินสแตนซ์ที่ยังดำเนินการอยู่	"Failed to modify workflow: Outputs of a finished task are modified"
การแก้ไขกระแสงานมีการเปลี่ยนแปลงเงื่อนไขของเส้นเชื่อมขาเข้าของงานที่ต่อจากโหนด Decision ของอินสแตนซ์ที่ดำเนินการอยู่	"Failed to modify workflow: Guard of an Incoming edge of an ongoing/finished task is modified"
การแก้ไขกระแสงานมีการเพิ่มเส้นเชื่อมขาออกของโหนด Fork ที่การทำงานผ่านเลยจากโหนด Fork นั้นไปแล้วของอินสแตนซ์ที่ดำเนินการอยู่	"Failed to modify workflow: New flows are added to a Fork node that subsequent tasks have started"
การแก้ไขกระแสงานมีการลบเหตุการณ์ออก และมีอินสแตนซ์ที่ดำเนินการอยู่	"Failed to modify workflow: Some Events are removed"
การแก้ไขกระแสงานมีการแก้ไขคุณสมบัติเหตุการณ์และมีอินสแตนซ์ที่ดำเนินการอยู่	"Failed to modify workflow: Some attributes of an Event are modified"
การแก้ไขกระแสงานมีการเพิ่มเหตุการณ์ที่มีชื่อเดียวกับเหตุการณ์ที่มีอยู่เดิม และมีอินสแตนซ์ที่ดำเนินการอยู่	"Failed to modify workflow: Duplicated Event names are added"
ผู้ใช้ขอทำกลับการทำงานโดยที่งานถัดไปถูกบันทึกการเริ่มทำงานแล้ว	"Cannot undo: Subsequent task(s) has already started"

ประวัติผู้เขียนวิทยานิพนธ์

นายธนวัฒน์ มหาไตรภพ เกิดเมื่อวันที่ 21 ตุลาคม พ.ศ. 2524 สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิตจากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2546 จากนั้นจึงเข้าศึกษาต่อปริญญาวิทยาศาสตรมหาบัณฑิตสาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2548 ปัจจุบันทำงานในตำแหน่ง Technical Specialist บริษัทรอยเตอร์ ซอฟท์แวร์ (ประเทศไทย) จำกัด