การทดสอบแบบมิวเทชันสำหรับตัวดำเนินการดัดแปลงนิพจน์ของบีเพล

นายนัฐพล ไทยสาครพันธ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2552

MUTATION TESTING FOR EXPRESSION MODIFICATION OPERATORS OF BPEL

Mr. Natthapol Thaisakonpun

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2009

หัวข้อวิทยานิพนธ์	การทดสอบแบบมิวเทชันสำหรับตัวดำเนินการดัดแปลงนิพจน์ของบีเพล
โดย	นายนัฐพล ไทยสาครพันธ์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์)

..... กรรมการ
(รองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ นครทิพย์ พร้อมพูล)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร. ภัทรชัย ลลิตโรจน์วงศ์)

นัฐพล ไทยสาครพันธ์ : การทดสอบแบบมิวเทชันสำหรับตัวดำเนินการดัดแปลงนิพจน์ของบีเพล. (MUTATION TESTING FOR EXPRESSION MODIFICATION OPERATORS OF BPEL) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. ธาราทิพย์ สุวรรณศาสตร์, 121 หน้า.

ดับเบิลยูเอสบีเพล หรือบีเพล เป็นภาษากระแสงานทางธุรกิจที่ได้รับการออกแบบมาสำหรับเว็บเซอร์วิซ บีเพลนั้นใช้สำหรับกำหนดการทำงานร่วมกันของเว็บเซอร์วิซ และมีหน้าที่ประสานการทำงานร่วมกันของเซอร์วิซเหล่านั้น วิทยานิพนธ์นี้ได้นำเสนอวิธีการทดสอบบีเพลโดยใช้วิธีการทดสอบแบบมิวเทชัน การทดสอบแบบมิวเทชัน หรือการวิเคราะห์แบบมิวเทชัน เป็นวิธีการทดสอบที่มีรากฐานอยู่บนความผิดพร่องของโปรแกรม ซึ่งใช้สำหรับวัดความเพียงพอของกรณีทดสอบ วิทยานิพนธ์นี้ได้ประยุกต์ใช้การทดสอบแบบมิวเทชันกับภาษาบีเพลโดยการใส่ความผิดพร่องเข้าไปในเอกสารบีเพลเพื่อที่จะสร้างมิวแตนท์ วิทยานิพนธ์นี้ได้มีการนิยามตัวดำเนินการมิวเทชันสำหรับภาษาบีเพลโดยทำตามหลักการเลือกตัวดำเนินการมิวเทชัน เพื่อช่วยในการลดจำนวนตัวดำเนินการมิวเทชัน นอกจากนี้วิทยานิพนธ์นี้ยังสร้างเครื่องมือการทดสอบมิวเทชันสำหรับภาษาบีเพลซึ่งมีความสามารถในการสร้างมิวแตนท์ ดีพลอยโปรแกรมลงในเครื่องประมวลผลบีเพล เรียกใช้เซอร์วิซของโปรแกรมทดสอบ และรายงานผลการทดลองได้อย่างอัตโนมัติ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต นัฐพล ไทยสาครพันธ์

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก ธาราทิพย์ สุวรรณศาสตร์

ปีการศึกษา 2552

4970385021: MAJOR SOFTWARE ENGINEERING

KEY WORDS: BPEL / SOFTWARE TESTING / MUTATION TESTING / MUTATION OPERATOR

NATTHAPOL THAISAKONPUN : MUTATION TESTING FOR EXPRESSION MODIFICATION OPERATORS OF BPEL. THESIS ADVISOR: ASSOC. PROF. TARATIP SUWANNASART, Ph.D., 121 pp.

Business Process Execution Language (BPEL) is an XML-based language used for the definition and execution of business process by using Web Services. BPEL is a coordination and composition language for Web Services. We propose a technique for testing BPEL by using mutation testing. Mutation testing or mutation analysis is a fault-based testing method for measuring the adequacy of test cases. We apply mutation testing to BPEL by injecting fault to BPEL document in order to generate mutants. We identify mutation operators by following selective mutation to decrease number of mutation operators. Mutation testing is a difficult testing method because mutation testing generates large number of mutants. Therefore, a prototype of a mutation testing tool for BPEL aims at real software projects is implemented. This tool is used for automatically generating mutants, deploys mutants to BPEL server, executes test cases, and reports test results.

Department : Computer Engineering Student's Signature ณัฐพล ไทยสกลนันท์
Field of Study : Software Engineering Advisor's Signature ทาราทิพ สุวรรณสาร
Academic Year : 2009

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จไปได้ด้วยดี เนื่องจากความกรุณาของผู้เกี่ยวข้องทุกฝ่ายผู้เขียนขอกราบขอบพระคุณ รศ. ดร. ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาเอาใจใส่ มอบความรู้ คำแนะนำต่างๆ ที่เป็นประโยชน์อย่างยิ่งต่อวิทยานิพนธ์ รวมทั้งตรวจแก้ไขข้อบกพร่องให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์ยิ่งขึ้น

ขอกราบขอบพระคุณ ผศ.ดร.อาทิตย์ ทองทักษ์ ที่กรุณาสละเวลามาเป็นประธานกรรมการการสอบวิทยานิพนธ์ ขอกราบขอบพระคุณ รศ.ดร.วิวัฒน์ วัฒนาวุฒิ ผศ.นครทิพย์ พร้อมพูล และ ผศ.ดร.ภัทรชัย ลลิตโรจน์วงศ์ ที่กรุณาสละเวลามาเป็นกรรมการการสอบวิทยานิพนธ์ รวมทั้งให้คำแนะนำและแก้ไขข้อบกพร่องให้วิทยานิพนธ์ฉบับนี้มีความสมบูรณ์ยิ่งขึ้น

ขอขอบคุณ คุณชิดชนก อัศวโภคี และเพื่อนๆวิศวกรรมซอฟต์แวร์ทุกคน ที่คอยช่วยเหลือ แก้ไขปัญหา ให้คำปรึกษา และให้กำลังใจจนวิทยานิพนธ์เล่มนี้สำเร็จ

สุดท้ายขอกราบขอบพระคุณ คุณวรรณาและคุณประภาส ไทยสาครพันธ์ คุณแม่และคุณพ่อที่แสนดี และทุกคนในครอบครัว ที่ให้ทุกสิ่งทุกอย่าง ทั้งความรัก ความห่วงใย และกำลังใจมาตลอด

สารบัญ

หน้า

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

รูปที่ หน้า

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

กระบวนการทดสอบซอฟต์แวร์เป็นขั้นตอนที่สำคัญของการพัฒนาซอฟต์แวร์ ซึ่งเป็นขั้นตอนที่ทำให้มั่นใจได้ว่าซอฟต์แวร์ที่สร้างขึ้นมามีคุณภาพและมีความน่าเชื่อถือ วิธีการที่ใช้ทดสอบซอฟต์แวร์นั้นมีหลายวิธี เช่น การทดสอบแบบมิวเทชัน (Mutation Testing) หรือการวิเคราะห์แบบมิวเทชัน (Mutation Analysis) เป็นวิธีการหนึ่งที่ใช้ในการทดสอบซอฟต์แวร์และประเมินประสิทธิภาพของกรณีทดสอบ (Test Case) ที่สร้างขึ้นมา

การทดสอบแบบมิวเทชันเป็นการทดสอบระดับหนึ่งหน่วย (Unit Testing) ที่มีประสิทธิภาพที่สูงและมีค่าใช้จ่ายสูงด้วยเช่นกัน สิ่งนี้เป็นข้อจำกัดที่ทำให้การทดสอบแบบมิวเทชันไม่สามารถนำไปประยุกต์ใช้ได้จริง ดังนั้นจึงมีการวิจัยเพื่อคิดค้นเทคนิคและขั้นตอนวิธี (Algorithm) เพื่อลดค่าใช้จ่ายของการทดสอบแบบมิวเทชัน เช่น หลักการการทดสอบแบบมิวเทชันโดยการคัดเลือก (Selective Mutation) เพื่อช่วยในการลดจำนวนตัวดำเนินการมิวเทชัน

ปัจจุบันสถาปัตยกรรมเชิงบริการหรือเอสโอเอ (SOA-Service Oriented Architecture) ได้มีบทบาทที่สำคัญมากยิ่งขึ้นทุกที โดยเอสโอเอเป็นรูปแบบของสถาปัตยกรรมทางซอฟต์แวร์ที่แบ่งซอฟต์แวร์ออกเป็นองค์ประกอบย่อยๆ โดยแต่ละส่วนถูกเรียกว่าเซอร์วิซ (Service) ซึ่งแต่ละส่วนมีหน้าที่รับผิดชอบการทำงานเฉพาะ งานบางงานอาจจะใช้เพียงเซอร์วิซเดียวก็สามารถบรรลุผล แต่งานบางงานต้องประกอบขึ้นจากหลายๆเซอร์วิซ โดยที่รูปแบบการติดต่อของแต่ละเซอร์วิซจะติดต่อกันอย่างหลวมๆ (Loose Coupling) ปัจจุบันมีการประยุกต์ใช้เอสโอเอที่เป็นมาตรฐานและเป็นที่ยอมรับอย่างแพร่หลายคือเว็บเซอร์วิซ (Web Service)

ดับเบิลยูเอสบีเพล (WSBPEL-Web Services Business Process Execution Language) หรือบีเพล (BPEL) เป็นภาษากระแสงานทางธุรกิจที่ถูกออกแบบมาสำหรับเว็บเซอร์วิซ (Web Service) บีเพลนั้นใช้สำหรับกำหนดการทำงานร่วมกันของเว็บเซอร์วิซ (Service Composition) และมีหน้าที่ประสานการทำงานร่วมกัน (Orchestration) ของเซอร์วิซเหล่านั้น

มีงานวิจัยที่ทำเกี่ยวกับการทดสอบสำหรับภาษาบีเพล เช่น [1] ได้เสนอแนวทางการค้นหากราฟ (Graph-Search) มาประยุกต์ใช้กับการสร้างกรณีทดสอบสำหรับบีเพล ซึ่งใช้ได้ดีกับลักษณะภาษาของบีเพลที่มีการทำงานแบบพร้อมกัน (Concurrent) ซึ่งงานนี้ได้นิยามกราฟการไหลสำหรับบีเพล (BFG-BPEL Flow Graph) โดยเพิ่มเติมจากกราฟการควบคุมการไหล (CFG-Control Flow Graph) เพื่อจะแสดงภาษาบีเพลในแบบจำลองเชิงแผนภาพสำหรับวิถีของการทดสอบแบบพร้อมกัน (Concurrent Test Paths) ซึ่งสามารถแปลงได้จากแบบจำลองกราฟควบคุมการไหลสำหรับภาษาบีเพล และข้อมูลการทดสอบสามารถสร้างได้โดยใช้วิธีแก้ปัญหาด้วยข้อจำกัด (Constraint Solving Method)

งานวิจัยที่ทำเกี่ยวกับการทดสอบภาษาบีเพลที่ผ่านมาไม่มีงานวิจัยใดเลยที่ใช้การทดสอบแบบมิวเทชันกับภาษาบีเพล ดังนั้นงานวิจัยนี้จึงมีแนวคิดที่จะนำวิธีการทดสอบแบบมิวเทชันมาใช้กับภาษาบีเพล

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อประยุกต์การทดสอบแบบมิวเทชันกับภาษาบีเพล

1.2.2 เพื่อสร้างเครื่องมือที่สนับสนุนการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล

1.3 ขอบเขตของการวิจัย

1.3.1 วิทยานิพนธ์นี้สนใจเฉพาะตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์ของภาษาบีเพลซึ่งนิพจน์ที่สนใจคือ นิพจน์บูลลีน นิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย และนิพจน์ทั่วไป

1.3.2 เอกสารบีเพล เอกสารดับเบิลยูเอสดีแอล และเอกสารเอ็กซ์เอ็มแอลสคีมา ซึ่งเป็นข้อมูลนำเข้าต้องถูกต้องตรงตามมาตรฐานของโอเอซิส และดับเบิลยูสามซี (W3C)

1.3.3 ภาษาในการสอบถามข้อมูล (Query) ของภาษาบีเพลต้องใช้ภาษาเอ็กซ์พาธ (Xpath) เท่านั้น

1.3.4 รุ่นของภาษาบีเพลที่โปรแกรมทดสอบแบบมิวเทชันสำหรับภาษาบีเพลสามารถรองรับได้คือรุ่น 2.0 เท่านั้น (WS-BPEL 2.0)

1.3.5 ผู้ใช้ต้องเป็นผู้สร้างกรณีทดสอบเองเครื่องมือนี้ไม่สามารถสร้างกรณีทดสอบได้

1.3.6 ตัวดำเนินการมิวเทชันที่เครื่องมือนี้สามารถรองรับได้เป็นตัวดำเนินการมิวเทชันที่นิยามในวิทยานิพนธ์นี้เท่านั้นผู้ใช้ไม่สามารถเพิ่มเติมได้

1.3.7 เครื่องมือนี้ไม่สามารถระบุมิวแตนท์สมมูลได้

1.3.8 เครื่องมือสามารถรองรับได้เฉพาะเซอร์วิซแบบร้องขอ-ตอบสนองเท่านั้น

1.3.9 วิทยานิพนธ์นี้สร้างโปรแกรมตัวอย่างขึ้นมา 5 โปรแกรมเพื่อใช้ในการทดสอบ

1.3.10 เครื่องมือที่สร้างขึ้นมีความสามารถดังต่อไปนี้เป็นอย่างน้อย

1) สามารถสร้างโปรแกรมมิวแตนท์ได้อย่างอัตโนมัติ

2) ผู้ใช้สามารถเลือกชนิดของตัวดำเนินการมิวเทชันที่ผู้ใช้ต้องการใช้ทดสอบได้

3) สามารถติดตั้งโปรแกรมมิวแตนท์และโปรแกรมต้นฉบับลงเครื่องประมวลผลบีเพลได้อย่างอัตโนมัติ

4) สามารถเรียกใช้งานเซอร์วิซของโปรแกรมมิวแตนท์และเซอร์วิซของโปรแกรมต้นฉบับด้วยกรณีทดสอบที่ผู้ใช้นำเข้าได้อย่างอัตโนมัติ

5) สามารถรายงานจำนวนมิวแตนท์ที่ยังคงมีชีวิตอยู่ มิวแตนท์ที่ถูกกำจัด และค่าร้อยละของมิวแตนท์ที่ถูกกำจัดได้

6) สามารถระบุโปรแกรมมิวแตนท์ที่ยังคงมีชีวิตอยู่ได้

1.4 ประโยชน์ที่จะได้รับ

1.4.1 สามารถประยุกต์ใช้การทดสอบแบบมิวเทชันกับภาษาบีเพล

1.4.2 เครื่องมือช่วยในการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล

1.5 วิธีดำเนินการวิจัย

1.5.1 ศึกษางานวิจัย ทฤษฎีและเครื่องมือการทดสอบแบบมิวเทชันที่เกี่ยวข้อง

1.5.2 ศึกษาภาษาบีเพล

1.5.3 ศึกษาวิธีการทดสอบแบบมิวเทชัน

1.5.4 สร้างโปรแกรมตัวอย่างและสร้างกรณีทดสอบสำหรับโปรแกรมตัวอย่าง

1.5.5 ออกแบบเครื่องมือ

1.5.6 สร้าง ทดสอบและปรับปรุงเครื่องมือ

1.5.7 บันทึก สรุปผลการวิจัยและข้อเสนอแนะ

1.5.8 จัดทำรายงานวิทยานิพนธ์

1.6 ลำดับขั้นตอนการเสนอผลการวิจัย

วิทยานิพนธ์ฉบับนี้แบ่งเนื้อหาออกเป็น 6 บทดังต่อไปนี้ บทที่ 1 เป็นบทนำซึ่งกล่าวถึงความเป็นมาและความสำคัญของปัญหา รวมถึงวัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีการดำเนินการวิจัย ลำดับขั้นตอนการเสนอผลการวิจัย และผลงานตีพิมพ์จากวิทยานิพนธ์ บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานและงานวิจัยที่เกี่ยวข้องกับงานวิจัยนี้ บทที่ 3 กล่าวถึงแนวคิด วิธีการสร้างมิวแตนท์ ซึ่งจะกล่าวถึงตัวดำเนินการมิวเทชันสำหรับภาษาบีเพล การดัดแปลงนิพจน์ของภาษาบีเพล และการสร้างและออกแบบเครื่องมือ บทที่ 4 กล่าวถึงสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ และโครงสร้างของเครื่องมือ บทที่ 5 กล่าวถึงสภาพแวดล้อมที่ใช้ในการทดสอบเครื่องมือ ขั้นตอนการทดสอบเครื่องมือ การสร้างโปรแกรมที่ใช้ในการทดสอบ และผลการทดสอบ และบทที่ 6 เป็นข้อสรุปและข้อเสนอแนะจากการวิจัย

1.7 ผลงานตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “Mutation Testing for Expression Modification Operator of BPEL” โดย นัฐพล ไทยสาครพันธ์ และรศ.ดร.ธาราทิพย์ สุวรรณศาสตร์ ในงานประชุมทางวิชาการ 13th National Computer Science and Engineering Conference (NCSEC 2009) ณ จังหวัดกรุงเทพฯ ประเทศไทย ระหว่างวันที่ 4-6 พฤศจิกายน 2552 โดยงานวิจัยนี้ได้รับการคัดเลือกให้ได้รับรางวัล Best Paper Award ในหมวด Computer Software

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 การทดสอบแบบมิวเทชันหรือการวิเคราะห์แบบมิวเทชัน

การทดสอบแบบมิวเทชันเป็นการทดสอบจากความผิดพร่อง (Fault) ของโปรแกรม ซึ่งเป็นวิธีการที่ใช้วัดความพอเพียงของกรณีทดสอบ [2] การทดสอบแบบมิวเทชันนั้นจะทำให้โปรแกรมเกิดความผิดพร่องโดยสร้างโปรแกรมขึ้นมาหลายๆรุ่นแต่ละรุ่นมีความผิดพร่องอยู่หนึ่งอย่าง ซึ่งเรียกโปรแกรมเหล่านี้ว่ามิวแตนท์ (Mutant) มิวแตนท์จะถูกกำจัด (Kill) ก็ต่อเมื่อข้อมูลออกของมิวแตนท์ต่างจากข้อมูลออกของโปรแกรมต้นฉบับ ซึ่งจากผลของคัปปิ้ง (Coupling Effect) ทำให้ถ้าสามารถค้นพบความผิดพร่องที่ใส่เข้าไปในโปรแกรมจะทำให้สามารถพบความผิดพร่องที่มีอยู่ในโปรแกรมอยู่แล้วส่วนใหญ่ด้วย [3]

ตัวดำเนินการมิวเทชัน (Mutation Operator) คือ กฎที่ใช้ในการสร้างมิวแตนท์ ตัวอย่างของตัวดำเนินการมิวเทชัน เช่น การแทนที่ตัวดำเนินการ การเพิ่มตัวดำเนินการใหม่ และการลบข้อความสั่ง (Statement) หลังจากที่ได้มิวแตนท์แล้วจะนำกรณีทดสอบมากระทำการ (Execute) กับมิวแตนท์ โดยมีเป้าหมายเพื่อที่จะแบ่งแยกระหว่างโปรแกรมที่มีความผิดพร่องกับโปรแกรมต้นฉบับ มิวแตนท์จะถูกกำจัด (Kill) ถ้าข้อมูลออกมีความแตกต่างกับข้อมูลออกของโปรแกรมต้นฉบับ ส่วนมิวแตนท์ที่ยังมีข้อมูลออกเหมือนกับข้อมูลออกของโปรแกรมต้นฉบับเรียกว่ามิวแตนท์ที่ยังคงมีชีวิตอยู่ (Live Mutant) ส่วนมิวแตนท์ที่ไม่สามารถหากรณีทดสอบมากำจัดได้เรียกว่ามิวแตนท์สมมูล (Equivalent Mutant) การทดสอบแบบมิวเทชันนี้สามารถวัดประสิทธิภาพของชุดของกรณีทดสอบได้จากคะแนนมิวเทชัน (Mutation Score) สามารถหาค่าได้จากสูตร

$$\text{คะแนนมิวเทชัน} = \frac{\text{จำนวนของมิวแตนท์ที่ถูกกำจัด}}{(\text{จำนวนของมิวแตนท์ทั้งหมด} - \text{จำนวนมิวแตนท์สมมูล})}$$

ดังนั้นค่าที่ดีที่สุดคือ 1 ซึ่งแสดงให้เห็นว่ามิวแตนท์ทั้งหมดถูกกำจัด ในรูปที่ 2.1 แสดงโปรแกรมต้นฉบับและแสดงตัวอย่างของโปรแกรมมิวแตนท์ในรูปที่ 2.2 ซึ่งโปรแกรมมิวแตนท์นี้

เกิดขึ้นจากการแทนที่ตัวดำเนินการ + ด้วยตัวดำเนินการ * ถ้าข้อมูลนำเข้าคือ 2 และ 2 จะทำให้เกิดมิวแตนท์ที่ยังคงมีชีวิตอยู่เพราะ 2*2=4 และ 2+2=4

```
package mutation.example;
public class Example {
        public static void main(String args[]){
                int x = Integer.parseInt(args[0]);
                int y = Integer.parseInt(args[1]);
                int result = x+y;
                System.out.println(result);
        }
}
```

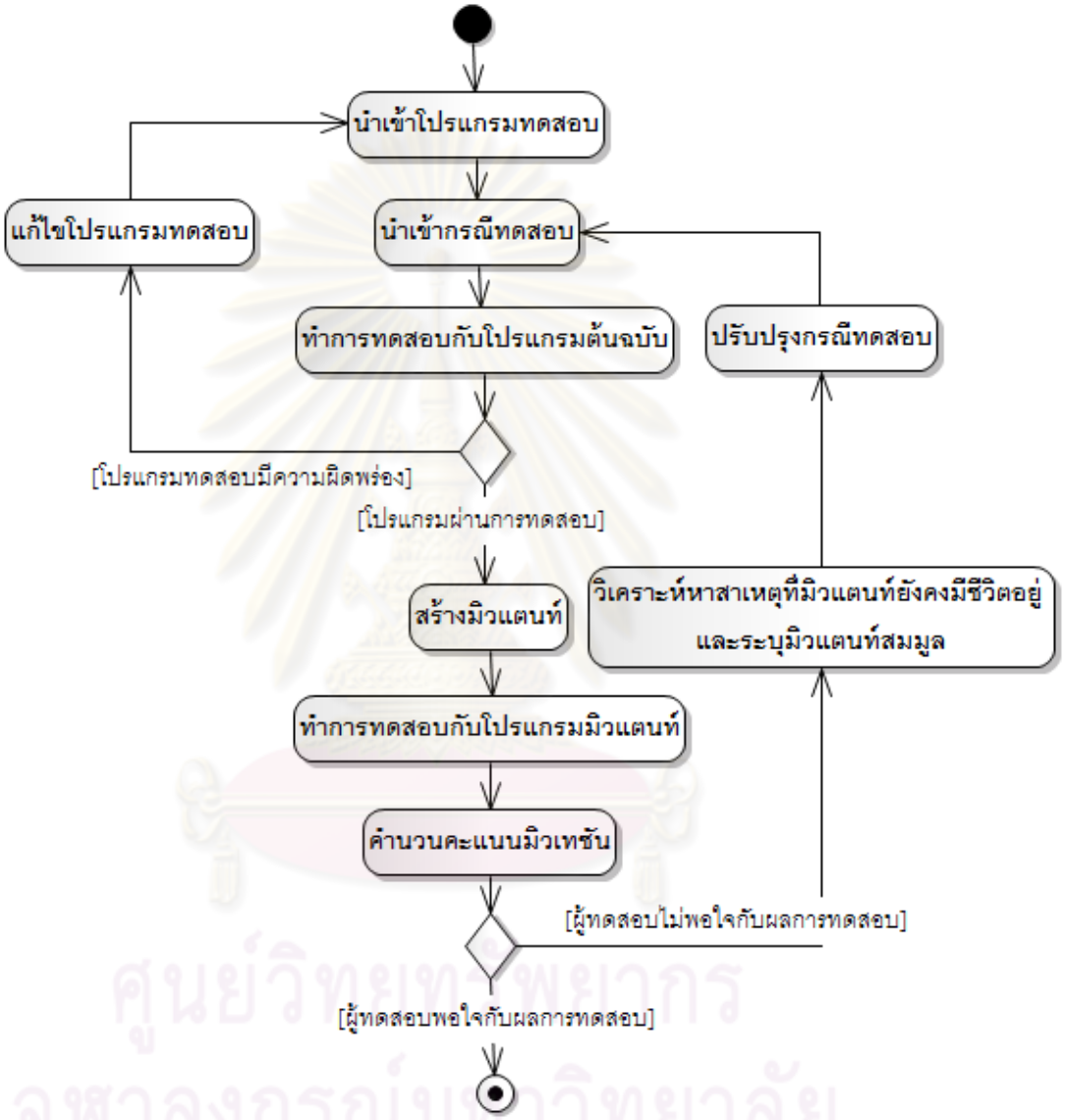
รูปที่ 2.1 ตัวอย่างของโปรแกรมต้นฉบับ

```
package mutation.example;
public class Example {
        public static void main(String args[]){
                int x = Integer.parseInt(args[0]);
                int y = Integer.parseInt(args[1]);
                int result = x*y;
                System.out.println(result);
        }
}
```

รูปที่ 2.2 ตัวอย่างของโปรแกรมมิวแตนท์

กระบวนทดสอบแบบมิวเทชันแสดงดังรูปที่ 2.3 นั้นเริ่มต้นด้วยการนำเข้าโปรแกรมทดสอบ นำเข้ากรณีทดสอบและกระทำการทดสอบโปรแกรมต้นฉบับเพื่อหาความผิดพร่องของโปรแกรมต้นฉบับและแก้ไขให้โปรแกรมทำงานถูกต้อง เมื่อโปรแกรมต้นฉบับผ่านการทดสอบแล้วสร้างโปรแกรมมิวแตนท์จากโปรแกรมต้นฉบับและกระทำการทดสอบกับทุก ๆ มิวแตนท์และหาค่าคะแนนมิวเทชัน ถ้าผู้ทดสอบไม่พอใจกับผลของคะแนนมิวเทชันให้ทำการวิเคราะห์สาเหตุที่ทำ

ให้เกิดมิวแตนท์ที่ยังคงมีชีวิตอยู่และทำการปรับปรุงกรณีทดสอบ ซึ่งจะนำกรณีทดสอบใหม่นี้เข้าสู่ขั้นตอนนำเข้ากรณีทดสอบเพื่อเริ่มกระบวนการทดสอบรอบใหม่จนผู้ทดสอบพอใจกับผลของคะแนนมิวเทชัน



รูปที่ 2.3 กระบวนการทดสอบแบบมิวเทชัน

ถึงแม้การทดสอบแบบมิวเทชันจะมีประสิทธิภาพที่สูงแต่ก็มีข้อเสียที่ต้องใช้ค่าใช้จ่ายที่สูงเนื่องจากการทดสอบแบบมิวเทชันสร้างมิวแตนท์ออกมาจำนวนมากทำให้ต้องใช้เวลาในการทดสอบมาก วิธีหนึ่งที่จะลดค่าใช้จ่ายได้คือลดจำนวนของโปรแกรมมิวแตนท์ที่ถูกสร้างขึ้นมาโดยใช้การทดสอบแบบมิวเทชันโดยการคัดเลือกซึ่งเสนอโดย [4]

จำนวนของตัวดำเนินการมิวเทชันขึ้นอยู่กับภาษาโปรแกรมของระบบที่จะทดสอบและระบบมิวเทชันที่ใช้สำหรับการทดสอบ เช่น ระบบ Mothra [5] ใช้ 22 ตัวดำเนินการมิวเทชันซึ่งส่วนใหญ่ได้มาจากการศึกษาความผิดพลาดของโปรแกรมเมอร์ ซึ่งตัวดำเนินการเหล่านี้ผ่านการศึกษามาระยะเวลานานมากกว่า 10 ปี จากหลายระบบมิวเทชัน

ตัวดำเนินการมิวเทชันแต่ละตัวนั้นสามารถนำไปสร้างเป็นโปรแกรมมิวแตนท์ได้จำนวนไม่เท่ากัน งานวิจัย [4] ได้เสนอการทดสอบแบบมิวเทชันโดยการคัดเลือก โดยการหาค่าร้อยละของจำนวนโปรแกรมมิวแตนท์ที่ถูกสร้างขึ้นจากตัวดำเนินการมิวเทชันแต่ละตัวจากระบบ Mothra และทำการทดลองการทดสอบแบบมิวเทชันโดยไม่ใช้ตัวดำเนินการมิวเทชันที่มีค่าร้อยละของการนำไปสร้างมิวแตนท์ที่สูงที่สุดสองลำดับและได้ผลการทดลองว่าแม้จะไม่ใช้ตัวดำเนินการมิวเทชันสองตัวที่กล่าวถึงก็ยังให้ประสิทธิภาพของการทดสอบใกล้เคียงกับการทดสอบมิวเทชันแบบดั้งเดิม (Strong Mutation) ซึ่งต่อมางานวิจัย [6] เรียกวิธีการนี้ว่าการเลือกตัวดำเนินการมิวเทชันแบบ 2 ตัว (2-Selective Mutation) และขยายงานวิจัยออกเป็นการทดสอบแบบมิวเทชันโดยการคัดเลือกแบบ N ตัว ซึ่งเป็นการเลือกตัวดำเนินการมิวเทชันที่สามารถนำไปสร้างโปรแกรมมิวแตนท์ได้มากที่สุด N ลำดับออก และได้ทำการทดลองกับการทดสอบแบบมิวเทชันโดยการคัดเลือกแบบ 4 และ 6 ตัว (4- Selective, 6- Selective Mutation) ผลการทดลองแสดงให้เห็นว่าการทดสอบแบบมิวเทชันโดยการคัดเลือกแบบ 4 และ 6 ตัว ยังคงให้ประสิทธิภาพที่ใกล้เคียงกับการทดสอบแบบมิวเทชันแบบดั้งเดิม

ต่อมางานวิจัย [7] ได้ทำการแบ่งตัวดำเนินการมิวเทชันของระบบ Mothra ออกเป็น 3 กลุ่ม คือ กลุ่มตัวดำเนินการมิวเทชันแทนที่ตัวถูกดำเนินการ (Operand Replacement Operators) มี 11 ตัวดำเนินการ กลุ่มตัวดำเนินการมิวเทชันดัดแปลงนิพจน์ (Expression Modification Operators) มี 5 ตัวดำเนินการ และกลุ่มตัวดำเนินการมิวเทชันดัดแปลงข้อความสั่ง (Statement Modification Operators) มี 6 ตัวดำเนินการ งานวิจัย [7] ได้นิยามการทดสอบแบบมิวเทชันโดยการคัดเลือกแบบกลุ่มโดยแบ่งออกเป็น 4 กลุ่มดังต่อไปนี้

1) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มดัดแปลงนิพจน์และกลุ่มดัดแปลงข้อความสั่ง (ES-Selective Mutation) คือ การทดสอบแบบมิวเทชันที่ใช้ตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์และตัวดำเนินการมิวเทชันกลุ่มดัดแปลงข้อความสั่ง

2) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงข้อความสั่ง (RS-Selective Mutation) คือ การทดสอบแบบมิวเทชันที่ใช้เฉพาะตัวดำเนินการมิวเทชันกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงข้อความสั่ง

3) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงนิพจน์ (RE-Selective Mutation) คือ การทดสอบมิวเทชันที่ใช้เฉพาะตัวดำเนินการมิวเทชันกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงนิพจน์

4) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มดัดแปลงนิพจน์ (E-Selective Mutation) คือ การทดสอบมิวเทชันที่ใช้เฉพาะตัวดำเนินการมิวเทชันที่อยู่ในกลุ่มดัดแปลงนิพจน์

ผลการทดลองแสดงให้เห็นว่าตัวดำเนินการมิวเทชันที่แทนที่ตัวถูกดำเนินการและตัวดำเนินการมิวเทชันที่ดัดแปลงข้อความสั่งให้ประสิทธิภาพที่ต่ำต่อการทดสอบแบบมิวเทชัน ดังนั้นระบบ Mothra จึงสามารถใช้ตัวดำเนินการมิวเทชันเพียงกลุ่มเดียวคือ ตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์ซึ่งมีตัวดำเนินการมิวเทชันอยู่เพียง 5 ตัวแต่สามารถให้ประสิทธิภาพใกล้เคียงกับการทดสอบมิวเทชันแบบดั้งเดิมซึ่งใช้ตัวดำเนินการมิวเทชันถึง 22 ตัว

2.2 เอ็กซ์เอ็มแอลและเอ็กซ์เอ็มแอลสคีมา

2.2.1 ภาษาเอ็กซ์เอ็มแอล (XML: Extensible Markup Language)

เอ็กซ์เอ็มแอล [8] เป็นภาษาที่ใช้อธิบายข้อมูล ได้รับการออกแบบให้มีความสามารถที่จะอธิบายความหมายของตัวเองได้หรือนิยามข้อมูลได้ ทำให้ยืดหยุ่น ผู้พัฒนาสามารถขยายข้อมูลเพิ่มเติมได้มากเท่าที่ต้องการ ทำให้เอ็กซ์เอ็มแอลกลายเป็นภาษามาตรฐานเพื่ออธิบายข้อมูลสำหรับการแลกเปลี่ยนข้อมูลระหว่างแพลตฟอร์มและเทคโนโลยีที่แตกต่างกัน

เอกสารเอ็กซ์เอ็มแอลมีกฎเกณฑ์ โครงสร้าง และการตรวจสอบความถูกต้อง สรุปได้ดังนี้

1) กฎเกณฑ์เบื้องต้นสำหรับการสร้างเอกสารเอ็กซ์เอ็มแอล

1.1) อีลีเมนต์ ประกอบด้วยแท็กเปิด แท็กปิดและเนื้อหา เช่น <id>001</id>

1.2) แท็กของเอ็กซ์เอ็มแอลนั้น ตัวอักษรตัวใหญ่และตัวเล็กมีความแตกต่างกัน (Case Sensitive)

1.3) อีลีเมนต์ จะต้องซ้อนกันเป็นลำดับ เช่น <persondata><id>001</id></persondata>

1.4) อีลีเมนต์ว่าง คืออีลีเมนต์ที่ไม่มีเนื้อหา สามารถเขียนได้ 2 แบบ คือ <id></id> หรือ <id/>

1.5) เอกสารเอ็กซ์เอ็มแอลมีอีลีเมนต์ราก (Root Element) เพียงอีลีเมนต์เดียว ส่วนอีลีเมนต์ลูกอื่นๆจะซ้อนกันเป็นลำดับชั้นภายในอีลีเมนต์ราก

2) โครงสร้างของเอกสารเอ็กซ์เอ็มแอล

เอกสารเอ็กซ์เอ็มแอลประกอบด้วย 2 ส่วนหลัก คือ การประกาศส่วนที่เรียกว่า Prolog หรือ XML Declaration หมายถึงส่วนหัวของเอกสารเอ็กซ์เอ็มแอลที่ต้องการใช้งาน ส่วนที่ 2 คือส่วนพื้นที่แสดงแหล่งข้อมูลเอกสารเอ็กซ์เอ็มแอลเรียกว่า Document Elements ซึ่งประกอบด้วยอีลีเมนต์ราก และอีลีเมนต์ลูก ดังแสดงในรูปที่ 2.4

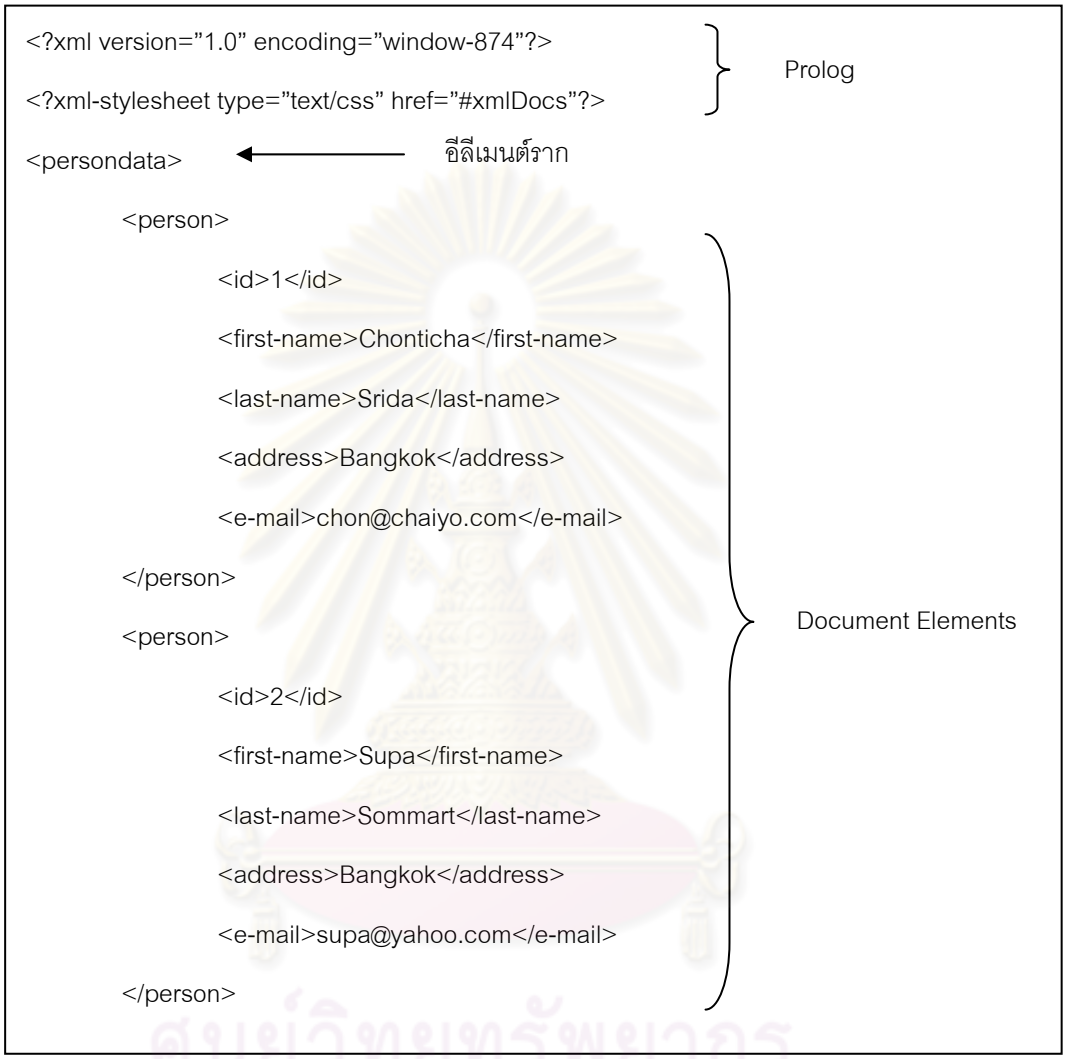
3) การตรวจสอบความถูกต้องในเอกสารเอ็กซ์เอ็มแอล

สำหรับเอกสารเอ็กซ์เอ็มแอลที่ทำตามกรอบกฎเกณฑ์เบื้องต้นภาษาเรียกว่า Well-Formed และหากเอกสารเอ็กซ์เอ็มแอลมีการทำตามกรอบของ DTD (Document Type Definition) หรือ เอ็กซ์เอ็มแอลสคีมาด้วย จะเรียกเอกสารเอ็กซ์เอ็มแอลนั้นว่าเป็นเอกสารที่ถูกต้องสมบูรณ์ (Valid XML Document)

2.2.2 เอ็กซ์เอ็มแอลสคีมา

เอ็กซ์เอ็มแอลสคีมา [9] ใช้นิยามโครงสร้างข้อมูลในเอกสารเอ็กซ์เอ็มแอล เช่น กำหนดชนิดข้อมูล ขอบเขตข้อมูล เป็นเสมือนข้อตกลงสำหรับกำหนดกฎเกณฑ์ไวยากรณ์โครงสร้างข้อมูลในภาษาให้สอดคล้องเข้าใจความหมายตรงกัน ใช้รูปแบบไวยากรณ์

เดียวกับเอ็กซ์เอ็มแอล เป็นภาษาที่ทำหน้าที่ตรวจสอบความถูกต้องให้กับข้อมูลที่จัดเก็บในรูปแบบเอกสารเอ็กซ์เอ็มแอล ตัวอย่างเอกสารเอ็กซ์แอ็มแอลสคีมาแสดงดังรูปที่ 2.5



รูปที่ 2.4 โครงสร้างของเอกสารเอ็กซ์เอ็มแอล

2.3 **ดับเบิลยูเอสดีแอล** (WSDL: Web Service Description Language)

ดับเบิลยูเอสดีแอล [10] เป็นรูปแบบเอกสารเอ็กซ์เอ็มแอลสำหรับอธิบายเว็บเซอร์วิซ ซึ่งมีโครงสร้างดังรูปที่ 2.6 ประกอบไปด้วยอีลีเมนต์ดังต่อไปนี้

1) อีลีเมนต์<types> แสดงชนิดของข้อมูลที่ใช้ในเซอร์วิซ
2) อีลีเมนต์<message> แสดงแมสเสจที่ต้องใช้ติดต่อกับเซอร์วิซ
3) อีลีเมนต์<operation> แสดงการดำเนินการที่เซอร์วิซนี้มีให้

4) อีลีเมนต์<portType> แสดงกลุ่มของการดำเนินการที่เซอร์วิซนี้มีให้

5) อีลีเมนต์<binding> แสดงโพรโทคอลและรูปแบบของข้อมูลสำหรับ<port Type> ของ เซอร์วิซ

6) อีลีเมนต์<port> แสดงการติดต่อและที่อยู่ของเครือข่าย

7) อีลีเมนต์<service> แสดงกลุ่มของ Port

2.3.1 อีลีเมนต์ <types>

อีลีเมนต์ <types> ทำให้สามารถประกาศชนิดของข้อมูลในเอกสารดับเบิลยูเอสดี-แอลได้ ดังตัวอย่างในรูปที่ 2.7 เป็นการประกาศชนิดข้อมูลแบบแถวลำดับชนิดสายอักขระซึ่งถูกนิยามอยู่ในรูปเอ็กซ์เอ็มแอลสคีมา

2.3.2 อีลีเมนต์<message>

แมสเสจภายในเว็บเซอร์วิซนั้นถูกใช้เป็น ข้อมูลนำเข้า ข้อมูลนำออก และชนิดของข้อความแสดงความผิดพร่องที่จะถูกรายงานออกมาเมื่อเกิดความผิดพร่องขึ้นระหว่างที่เว็บเซอร์วิซกำลังทำงาน อีลีเมนต์<message> นั้นจะประกอบไปด้วยชื่อของแมสเสจและอีลีเมนต์ <part> ภายใน ซึ่งอีลีเมนต์ <part> แต่ละตัวก็เปรียบเทียบได้กับอาร์กิวเมนต์ที่จะถูกส่งไปในฟังก์ชันดังนั้นอีลีเมนต์ <part> แต่ละตัวจึงประกอบไปด้วย ชื่อ และชนิดของตัวแปร ซึ่งตัวอย่างของการประกาศอีลีเมนต์ <message> แสดงดังรูปที่ 2.8 เป็นการประกาศแมสเสจชื่อว่า creditInformationMessage ซึ่งประกอบไปด้วย 3 <part>

```
<?xml version=”1.0”?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/xmlschema>
   <xsd:element name=”persondata”
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref=”person” minOccurs=”1” maxOccurs=”unbounded”/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name=”person”
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref=”id” minOccurs=”1” maxOccurs=”1”/>
            <xsd:element ref=”first-name” minOccurs=”1” maxOccurs=”1”/>
            <xsd:element ref=”last-name” minOccurs=”1” maxOccurs=”1”/>
            <xsd:element ref=”address” minOccurs=”1” maxOccurs=”1”/>
            <xsd:element ref=”e-mail” minOccurs=”1” maxOccurs=”1”/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name=”id” type=”xsd:int”/>
   <xsd:element name=”first-name” type=”xsd:string” minLength value=”1” maxLength value=”20”/>
    <xsd:element name=”last-name” type=”xsd:string” minLength value=”1” maxLength value=”50”/>
   <xsd:element name=”address” type=”xsd:stringt” minLength value=”5” maxLength value=”200”/>
   <xsd:element name=”e-mail” type=”xsd:string” minLength value=”5” maxLength value=”30”/>
<xsd:schema>
```

รูปที่ 2.5 ตัวอย่างเอกสารเอ็กซ์เอ็มแอลสคีมา

```
<definitions name = " . . . ">
<types>
        [ . . . ]
</types>
<message name = ". . . ">
        [ . . . ]
</message>
<portType name = " . . . " >
        <operation name="…..">
        [ . . . ]
        </operation>
</portType>
<binding>
        [ . . . ]
</binding>
<service>
        <port>
        [ . . . ]
        </port>
</service>
```

รูปที่ 2.6 โครงสร้างของเอกสารดับเบิลยูเอสดีแอล

```
<types name =  “StringArray ”>
        <xsd:schema targetNamespace= “http://typeSample.org ”>
                <xsd:complexType name =  “stringarray”>
                        <xsd:choice maxOccurs= “20”>
                                <xsd:elementname = “item” type = “xsd:string ”/>
                        </xsd:choice>
                </ xsd:complexType>
        </ xsd:schema>
</ types>
```

รูปที่ 2.7 ตัวอย่างการประกาศข้อมูลแบบแถวลำดับชนิดสายอักขระในอีลีเมนต์ <types>

```
<message name = “creditInformationMessage”>
        <part name =  “firstName” type = “xsd:string”/>
        <part name = “lastName” type = “xsd:string”/>
        <part name = “amount” type = “xsd:integer”/>
</message>
```

รูปที่ 2.8 ตัวอย่างการประกาศชนิดตัวแปรแมสเสจของเอกสารดับเบิลยูเอสดีแอล

2.3.3 อีลีเมนต์ <portTypes> และ อีลีเมนต์<operation>

อีลีเมนต์ <portTypes> จะรวมกลุ่ม <operation> ไว้ในชื่อของ <portTypes> เดียวกัน <operation> นั้นสามารถพิจารณาเป็นฟังก์ชันในภาษาโปรแกรมแบบดั้งเดิมได้ ซึ่ง <operation> สามารถแบ่งได้เป็น 4 ชนิด

1) แบบทางเดียว (One-way) เซอร์วิซเป็นผู้รับแมสเสจเท่านั้น

2) แบบร้องขอ-ตอบสนอง (Request-Response) เซอร์วิซเป็นผู้รับแมสเสจและส่งแมสเสจตอบกลับ

3) แบบเซอร์วิซเป็นผู้ร้องขอและรอการตอบสนอง (Solicit-response) เซอร์วิซเป็นผู้ส่งแมจแสจและรอรับแมจแสจส่งกลับมา

4) แบบส่งเท่านั้น (Notification) เซอร์วิซเป็นผู้ส่งแมสเสจเท่านั้น

การนิยาม <operation> นั้นต้องมี ชื่อ แมสเสจนำเข้า แมสเสจนำออก และชนิดข้อมูลของความผิดพร่องที่จะถูกส่งมาให้ผู้ใช้เมื่อเกิดความผิดพร่องขึ้นขณะที่เว็บเซอร์วิซทำงาน ซึ่งชนิดข้อมูลเหล่านี้คือข้อมูลที่นิยามอยู่ในอีลีเมนต์ <message> ของเอกสารดับเบิลยูเอสดีแอลนั่นเอง ดังตัวอย่างในรูปที่ 2.9 เป็นการประกาศ <portType> ที่มีชื่อว่า loanApprovalPT โดยที่ <portType> นี้ประกอบไปด้วย 1 <operation> มีชื่อว่า approve โดย <operation> นี้มีชนิดแมสเสจนำเข้าเป็นชนิด creditInformationMessage มีแมสเสจนำออกเป็นชนิด approvalMessage และมีความผิดพร่องที่จะถูกส่งให้ผู้ใช้เมื่อเซอร์วิซเกิดความผิดพร่องขึ้นขณะทำงานชื่อ loanProcessFault ซึ่งมีชนิดของแมสเสจเป็น loanRequestErrorMessage

```
<portType name=”loanApprovalPT”>
        <operation name= “approve”>
                <input message = “loandef:credit InformationMessage”/>
                <output message =  “tns:approvalMessage”/>
                <faultname = “loanProcessFault”
                message = “loandef:loanRequestErrorMessage”/>
        </operation>
</portType>
```

รูปที่ 2.9 ตัวอย่างการประกาศ <portType>

2.4 ภาษาบีเพล (BPEL)

ภาษาบีเพล [11] ปัจจุบันได้รับการดูแลและปรับปรุงเพิ่มเติมอย่างต่อเนื่องโดยหน่วยงานโอเอซิส (OASIS) และมีชื่อเรียกอย่างเป็นทางการว่าดับเบิลยูเอสบีเพล จากชื่อเดิมคือบีเพลโฟดับเบิลยูเอส (BPEL4WS) [5] ซึ่งภาษาบีเพลประกอบไปด้วยส่วนต่างๆดังต่อไปนี้

2.4.1 อีลีเมนต์ <process>

อีลีเมนต์ <process> เป็นอีลีเมนต์รากของเอกสารบีเพล ซึ่งประกอบไปด้วยอีลีเมนต์ลูกที่เป็นทางเลือก คือ <import> <partnerLinks> <messageExchanges> <variables> <correlationSets> <faultHandlers> <eventHandlers> นอกจากนี้ภายในอีลีเมนต์ Process ยังประกอบไปด้วย แอ็คติวิตี (Activity) ซึ่งสามารถเป็นได้ทั้งแอ็คติวิตีอย่างง่าย (Basic Activity) หรือ แอ็คติวิตีเชิงโครงสร้าง (Structured Activity) ซึ่งแอ็คติวิตีเชิงโครงสร้างนี้สามารถมีแอ็คติวิตีอื่นๆ อยู่ภายในได้ โครงสร้างของอีลีเมนต์ Process แสดงในรูปที่ 2.10

```
<process>
        <import> [.....]</import>
        <partnerLinks> [.....]</partnerLinks>
        <messageExchanges> [.....]</messageExchanges>
        <variables> [.....]</variables>
        <correlationSets>[.....] </correlationSets>
        <faultHandlers> [.....]</faultHandlers>
        <eventHandlers> [.....]</eventHandlers>
        [activity]
</process>
```

รูปที่ 2.10 โครงสร้างของอีลีเมนต์ <process>

2.4.2 ตัวแปร (Variables)

ชนิดของตัวแปรที่สามารถนำมาใช้กับอีลีเมนต์ <variable> ได้คือ ตัวแปรชนิดแมสเสจของเอกสารดับเบิลยูเอสดีแอล ชนิดตัวแปรของเอ็กซ์เอ็มแอลสคีมา (XML Schema Type) และชนิดตัวแปรอีลีเมนต์ของเอ็กซ์เอ็มแอลสคีมา (XML Schema Element) ดังรูปที่ 2.11 แสดงการประกาศตัวแปรเพื่อใช้ในการเก็บข้อมูลที่อยู่ซึ่งมีชนิดของข้อมูลเป็นชนิดสายอักขระและมีการกำหนดค่าเริ่มต้นโดยนำค่ามาจากตัวแปร quoteRequest

```
<variable name="address" type="xsd:string">
    <from>
        < variable="quoteRequest" property="emailAddr" />
    </from>
</variable>
```

รูปที่ 2.11 ตัวอย่างของการประกาศและกำหนดค่าเริ่มต้นให้กับตัวแปร

2.4.3 นิพจน์

ภายในเอกสารบีเพลสามารถมีนิพจน์ได้ 5 ชนิด คือ

1) นิพจน์บูลลีน (Boolean Expressions) สามารถพบได้ในข้อความสั่งเงื่อนไข เช่น แอ็คติวิตี <while> และ <if>

2) นิพจน์ค่าเส้นตาย (Deadline-valued) สามารถพบได้ในนิพจน์ until ของแอ็คติวิตี <onAlarm> และ <wait>

3) นิพจน์ค่าช่วงเวลา (Duration-valued) สามารถพบได้ในนิพจน์ for ของแอ็คติวิตี <onAlarm> <wait> และนิพจน์ <repeatEvery> ของแอ็คติวิตี <onAlarm>

4) นิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย (Unsigned Integer) สามารถพบได้ในอีลีเมนต์ <startCounterValue> <finalCounterValue> และ <brance> ซึ่งอยู่ภายในแอ็คติวิตี <forEach>

5) นิพจน์ทั่วไป สามารถพบได้ในการกำหนดค่าสำหรับการคำนวณอย่างง่ายในตรรกะของกระบวนการ

```
bpel:getVariableProperty('stockResult', 'level')>0                    (นิพจน์บูลลีน)
bpel:getLinkStatus('AtoB') and bpws:getLinkStatus('CtoB')             (นิพจน์บูลลีน)
bpel:getVariableProperty('assessment', 'tryCount') – 1                (นิพจน์ทั่วไป)
'P3DT10H'                                                             (นิพจน์ค่าช่วงเวลา)
'2002-12-24T18:00+01:00'                                              (นิพจน์ค่าเส้นตาย)
```

รูปที่ 2.12 ตัวอย่างของนิพจน์แบบต่างๆ ของภาษาบีเพล

2.4.4 แอ็คติวิตี

แอ็คติวิตีของภาษาบีเพลเป็นส่วนที่จัดการด้านตรรกะของกระบวนการ ซึ่งสามารถแบ่งแอ็คติวิตีได้เป็น 2 ชนิดคือ แอ็ตติวิตีอย่างง่าย และแอ็คติวิตีเชิงโครงสร้าง

1) แอ็คติวิตีอย่างง่าย

แอ็คติวิตีอย่างง่ายเป็นแอ็คติวิตีที่ทำภารกิจที่มีลักษณะไม่ต่อเนื่อง ในภาษาบีเพล 2.0 มีแอ็คติวิตีอย่างง่ายดังต่อไปนี้

1.1) แอ็คติวิตี <assign> มีหน้าที่สำเนาข้อมูลต้นทางไปยังข้อมูลปลายทาง ซึ่งในแอ็คติวิตี <assign> อาจจะมีการสำเนาข้อมูลหลายค่า ตัวอย่างของแอ็คติวิตี <assign> แสดงในรูปที่ 2.13 เป็นการสำเนาข้อมูลสองค่าในส่วนแรกเป็นการสำเนาข้อมูลจากตัวแปรชื่อ quoteRequest ใน part ชื่อ placeQuoteParameter ไปยังตัวแปรชื่อ highlightQuote ใน part ชื่อ placeQuoteResult ในส่วนที่สองเป็นการสำเนาข้อมูล จากตัวแปรชื่อ mvRecord ใน part ชื่อ licenseNumber ไปยังตัวแปรชื่อ vahicleLicense

1.2) แอ็คติวิตี <empty> เป็นแอ็คติวิตีที่บอกว่าไม่มีการทำอะไร ซึ่งอาจใช้ในกรณีที่มีการตรวจจับความผิดพร่อง หรืออาจจะใช้เป็นจุดประสาน (Synchronization) ในแอ็คติวิตี <flow> ดังรูปที่ 2.14 แสดงตัวอย่างการใช้แอ็คติวิตี <empty> เพื่อใช้เป็นจุดประสานเพื่อไปยังจุดเป้าหมายที่ชื่อ manyDrivers ถ้าค่าของ numberOfDrivers มากกว่า 5

1.3) แอ็คติวิตี <exit> เป็นการหยุดกระบวนการแบบทันที โดยไม่มีการเรียกการจัดการความผิดพร่อง (Fault Handler) ดังตัวอย่างในรูปที่ 2.15 แสดงการใช้งานแอ็คติวิตี <exit> เพื่อหยุดกระบวนการทันทีถ้าหากค่าของ numberOfDriver มากกว่า 10

```
<assign name="DefaultQuoteAssignment">
        <copy>
                <from variable="quoteRequest" part="placeQuoteParameters">
                        <query>/quoteInformation</query>
                </from>
                <to variable="highlightQuote" part="placeQuoteResult">
                        <query>/quote/customerInformation</query>
                </to>
        </copy>
        <copy>
                <from variable="mvRecord" part="licenseNumber"/>
                <to variable="vehicleLicense"/>
        </copy>
</assign>
```

รูปที่ 2.13 ตัวอย่างของแอ็คติวิตี <assign>

```
<if>
        <condition><![CDATA[$numberOfDrivers > 5]]> </condition>
        <empty>
                <sources> <source linkName="manyDrivers"/> </sources>
        </empty>
</if>
```

รูปที่ 2.14 ตัวอย่างของแอ็คติวิตี <empty>

```
<if>
        <condition>><![CDATA[$numberOfDrivers > 10]]></condition>
        <exit/>
</if>
```

รูปที่ 2.15 ตัวอย่างของแอ็คติวิตี <exit>

1.4) แอ็คติวิตี <invoke> ใช้ในการเรียกใช้เซอร์วิซได้ทั้งแบบซิงโครนัส (Synchronous) และ อะซิงโครนัส (Asynchronous) ดังรูปที่ 2.16 แสดงการเรียกใช้เซอร์วิซของผู้มีส่วนร่วมที่ระบุอยู่ใน Partner Link ที่มีชื่อว่า agent โดยการดำเนินการที่จะถูกกระบวนการเรียกใช้ชื่อ requestQuote ซึ่งอยู่ภายใน Port Type ที่มีชื่อว่า policyPT ข้อมูลที่กระบวนการจะส่งไปเรียกใช้เซอร์วิซเก็บอยู่ในตัวแปรชื่อ quoteRequest และข้อมูลที่กระบวนการได้รับเก็บอยู่ในตัวแปรชื่อ requestInfo

```
<invoke partnerLink="agent"
        portType="policyPT"
        operation="requestQuote"
        inputVariable="quoteRequest"
        outputVariable="requestInfo">
</invoke>
```

รูปที่ 2.16 ตัวอย่างของแอ็คติวิตี <invoke>

1.5) แอ็คติวิตี <receive> เป็นแอ็คติวิตีที่รอรับแมสเสจที่เข้ากันได้กับที่นิยามไว้ในเอกสารดับเบิลยูเอสดีแอล ดังรูปที่ 2.17 แสดงตัวอย่างของแอ็คติวิตี <receive> ที่มีชื่อว่า getQuote ซึ่งเป็นการรอรับแมสเสจที่ส่งมาให้การดำเนินการชื่อ requestQuote จากเซอร์วิซของผู้มีส่วนร่วมที่ระบุอยู่ใน Partner Link ที่มีชื่อว่า agent โดยข้อมูลนำเข้านี้จะเก็บอยู่ในตัวแปรชื่อ quoteDetails และการที่กำหนดให้ createInstance=”yes” แสดงว่าเมื่อมีการเรียกใช้แอ็คติวิตีนี้แล้วจะทำให้มีการสร้างกรณีตัวอย่าง (Instance) ของกระบวนการนี้ หรืออาจกล่าวได้ว่าแอ็คติวิตีนี้เป็นแอ็คติวิตีเริ่มต้นของกระบวนการ

```
<receive name="getQuote"
        partnerLink="agent"
        operation="requestQuote"
        variable="quoteDetails"
        createInstance="yes" />
```

รูปที่ 2.17 ตัวอย่างของแอ็คติวิตี <receive>

1.6) แอ็คติวิตี <reply> เป็นแอ็คติวิตีที่ตอบสนองต่อแมสเสจที่ได้รับมา ดังรูปที่ 2.18 แสดงตัวอย่างของแอ็คติวิตี <reply> ที่มีชื่อว่า StatusReply ซึ่งเป็นการตอบสนองแมสเสจที่ได้รับมาจากเซอร์วิซของผู้มีส่วนร่วมที่ระบุอยู่ใน Partner Link ที่ชื่อว่า ProcessPolicy โดยแอ็คติวิตีนี้เป็นแอ็คติวิตีที่ตอบสนองการดำเนินการที่มีชื่อว่า status ซึ่งการดำเนินการนี้อยู่ใน Port Type ที่มีชื่อว่า NewPolicySystem โดยข้อมูลที่จะตอบกลับถูกเก็บอยู่ในตัวแปรที่มีชื่อว่า statusResponse

```
<reply name="StatusReply"
      partnerLink="ProcessPolicy"
      portType="NewPolicySystem"
      operation="status"
      variable="statusResponse" />
```

รูปที่ 2.18 ตัวอย่างของแอ็คติวิตี <reply>

1.7) แอ็คติวิตี <rethrow> เป็นแอ็คติวิตีที่สามารถใช้ได้ภายในการจัดการความผิดพร่อง เท่านั้น โดยที่แอ็คติวิตีนี้จะส่งความผิดพร่องไปสู่ขอบเขตแม่ (Parent Scope) ดังรูปที่ 2.19 แสดงการใช้งานแอ็คติวิตี <rethrow> ที่มีชื่อว่า invalid เมื่อเกิดความผิดพร่องขึ้นกับกระบวนการ

```
<faultHandlers>
        <catchAll>
                <compensate />
                <rethrow name="invalid" />
        </catchAll>
</faultHandlers>
```

รูปที่ 2.19 ตัวอย่างของแอ็คติวิตี <rethrow>

1.8) แอ็คติวิตี <throw> เป็นแอ็คติวิตีที่ใช้ในการส่งข้อมูลความผิดพร่องไปสู่ขอบเขตแม่ วัตถุประสงค์เพื่อระบุความผิดพลาดทางธุรกิจ (Business Error) ดังตัวอย่างในรูปที่ 2.20 แสดงการใช้งานแอ็คติวิตี <throw> ที่มีชื่อว่า CancelPurchese โดยที่

กระบวนการนี้จะส่งข้อมูลความผิดพร่องที่มีชื่อว่า cancelPurchese ก็ต่อเมื่อมีเหตุการณ์ที่เกิดขึ้นตามที่กำหนดไว้ในแอ็คติวิตี <onEvent>

```
<eventHandlers>
  <onEvent messageType="abandonRequestMsg"
        partnerLink="ProcessPolicyPL"
        portType="ProcessPolicyPT"
        operation="abandon"
        variable="abandonVar">
            <throw faultName="cancelPurchase" name="CancelPurchase" />
  </onEvent>
</eventHandlers>
```

รูปที่ 2.20 ตัวอย่างของแอ็คติวิตี <throw>

1.9) แอ็คติวิตี <validate> เป็นแอ็คติวิตีที่ตรวจสอบข้อมูลที่อยู่ในตัวแปรสอดคล้องกับชนิดของข้อมูลที่ประกาศไว้หรือไม่ ดังรูปที่ 2.21 แสดงตัวอย่างของการใช้แอ็คติวิตี <validate> ตรวจสอบข้อมูลของตัวแปร applicationInfo และ spouseInfo

```
<validate variables="applicantInfo spouseInfo"/>
```

รูปที่ 2.21 ตัวอย่างของแอ็คติวิตี <validate>

1.10) แอ็คติวิตี <wait> เป็นแอ็คติวิตีที่หน่วงเวลาตามค่าช่วงเวลาหนึ่งหรือหน่วงเวลาจนถึงค่าของเวลาที่กำหนดไว้ ดังรูปที่ 2.22 แสดงตัวอย่างของการใช้แอ็คติวิตี <wait> เพื่อให้กระบวนการรอจนถึงวันที่ 25 เดือน 12 ปี 2010 เวลา 18.00 น.

```
<sequence>
  <!- wait until 6:00 PM Pacific Standard Time
      on 25 December 2010 ->
  <wait>
    <until>'2010-12-25T18:00-08:00'</until>
  </wait>
  <invoke ... />
</sequence>
```

รูปที่ 2.22 ตัวอย่างของแอ็คติวิตี <wait>

2) แอ็คติวิตีเชิงโครงสร้าง

แอ็คติวิตีเชิงโครงสร้างเป็นแอ็คติวิตีที่เป็นตัวกำหนดลำดับหรือเงื่อนไขของการทำงานของ แอ็คติวิตีภายใน ภาษาบีเพล 2.0 มีแอ็คติวิตีเชิงโครงสร้างดังต่อไปนี้

2.1) แอ็คติวิตี <flow> ภายในแอ็คติวิตี <flow> สามารถกำหนดกระบวนการที่ทำงานแบบพร้อมกันได้ ดังรูปที่ 2.23 แสดงตัวอย่างการใช้งานแอ็คติวิตี <flow> เพื่อเรียกใช้งานเซอร์วิซของสายการบิน American Airlines และเซอร์วิซของสายการบิน Delta Airlines แบบพร้อมกัน

2.2) แอ็คติวิตี <forEach> แอ็คติวิตีนี้สามารถกระทำการแอ็คติวิตีที่อยู่ภายในได้ทั้งในรูปแบบเรียงลำดับหรือแบบพร้อมกันในรูปแบบกระทำซ้ำ ดังรูปที่ 2.24 แสดงตัวอย่างของการใช้แอ็คติวิตี <forEach> ซึ่งมีการกำหนด parallel=yes แสดงว่าเป็นการกระทำการแอ็คติวิตีภายในแบบพร้อมกัน โดยตั้งชื่อตัวแปรนับจำนวนให้ชื่อว่า loopIndex และกำหนดค่าเริ่มต้นให้เป็น 1 ซึ่งจำนวนของการวนรอบนั้นถูกกำหนดให้เท่ากับค่าของ vehicle

2.3) แอ็คติวิตี <if> แอ็คติวิตีนี้จะกระทำการแอ็คติวิตีภายในก็ต่อเมื่อนิพจน์บูลลีนมีค่าออกมาเป็น true ภายในแอ็คติวิตี <if> อาจจะมีแอ็คติวิตี <elseIf> หรือ <else> ก็ได้ ดังรูปที่ 2.25 แสดงตัวอย่างของแอ็คติวิตี <if> ซึ่งในตัวอย่างมีทั้งการใช้งาน <if> <elseIf> และ else

```
<flow>
    <sequence>
     <invoke partnerLink="AmericanAirlines"
        portType="aln:FlightAvailabilityPT"
        operation="FlightAvailability"
        inputVariable="FlightDetails" />
     <receive partnerLink="AmericanAirlines"
        portType="aln:FlightCallbackPT"
        operation="FlightTicketCallback"
        variable="FlightResponseAA" />
    </sequence>
    <sequence>
     <invoke partnerLink="DeltaAirlines"
        portType="aln:FlightAvailabilityPT"
        operation="FlightAvailability"
        inputVariable="FlightDetails" />
     <receive partnerLink="DeltaAirlines"
        portType="aln:FlightCallbackPT"
        operation="FlightTicketCallback"
        variable="FlightResponseDA" />
    </sequence>
</flow>
```

รูปที่ 2.23 ตัวอย่างแอ็คติวิตี <flow>

2.4) แอ็คติวิตี <pick> เป็นแอ็คติวิตีที่รอการเกิดของเหตุการณ์ที่กำหนดไว้ ซึ่งภายในแอ็คติวิตี <pick> สามารถมีเหตุการณ์ได้ 2 ชนิดคือ <onMessage> และ <onAlarm> แต่อย่างน้อยที่สุดต้องมี 1 <onMessage> แต่ <onAlarm> จะมีเท่าไรก็ได้ เมื่อมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นแล้วเหตุการณ์ที่เหลือจะไม่สามารถเกิดได้ ดังรูปที่ 2.26 แสดงการใช้งานแอ็คติวิตี <pick> ถ้าแมสเสจตามที่ระบุรายละเอียดไว้ในอีลีเมนต์ <onMessage> เกิดขึ้นจะมีการกระทำการแอ็ตติวิตีภายใน แต่ถ้ารอจนเวลาผ่านไป 5 นาทีแล้วไม่มีแมจแสจตามที่กำหนดไว้ใน <onMessage> จะมีการกระทำการแอ็คติวิตีภายใน <onAlam>

```
<forEach counterName="loopIndex" parallel="yes">
<startCounterValue>1</startCounterValue>
<finalCounterValue>
    <query>count(/CustomerQuoteInformation/vehicle)</query>
</finalCounterValue>
  <scope>
    <assign>
      <copy>
        <from variable="inputRecord">
          <query>
            (/CustomerQuoteInformation/vehicle)[loopIndex]
           </query>
        </from>
        <to variable="currentVehicle"/>
      </copy>
    </assign>
    <invoke name="CheckVehicleWithDMV"
          partnerLink="dmvPL"
          portType="dmvPT">
          operation="retrieveLicenseStatus"
          inputVariable="currentVehicle"
          outputVariable="receiveDetail"
    </invoke>
  </scope>
</forEach>
```

รูปที่ 2.24 ตัวอย่างของแอ็คติวิตี <forEach>

```
<if>
      <condition>
           <![CDATA[bpel:getVariableProperty('mvStatusVariable','AutoStatus') = 'OWN']]>
      </condition>
      <flow>
              <!-- perform fulfillment work -->
      </flow>
<elseif>
       <condition>
  <![CDATA[bpel:getVariableProperty(mvStatusVariable', 'AutoStatus') = 'INCOMPLETE']]>
       </condition>
       <throw faultName="IncompleteFault" />
</elseif>
<else>
       <throw faultName="StatusFault" />
</else>
</if>
```

รูปที่ 2.25 ตัวอย่างของแอ็คติวิตี <if>

2.5) แอ็คติวิตี <repeatUntil> เป็นแอ็คติวิตีที่ใช้สำหรับนิยามการทำซ้ำ ซึ่งต้องทำอย่างน้อย 1 ครั้งและทำซ้ำจนกว่านิพจน์บูลลีนจะได้ผลลัพธ์เป็น true ดังตัวอย่างในรูปที่ 2.27 แสดงการใช้งานแอ็คติวิตี <repeatUtil> ซึ่งจะมีการกระทำการแอ็คติวิตีภายในจนกว่าค่าของนิพจน์ $underwriterStatus.result > 100 จะเป็น true

2.6) แอ็คติวิตี <sequence> แอ็คติวิตีนี้จะบรรจุแอ็คติวิตีที่กระทำการอย่างเป็นลำดับ ดังตัวอย่างในรูปที่ 2.28 การทำงานเป็นลำดับโดยเริ่มจากการรอเป็นเวลา 10 ปี และจะมีการเรียกใช้งานเซอร์วิซของผู้มีส่วนร่วม

2.7) แอ็คติวิตี <while> แอ็คติวิตีนี้จะกระทำการแบบทำซ้ำจนกว่านิพจน์บูลลีนจะมีค่าเป็น false ดังตัวอย่างในรูปที่ 2.29 จะมีการทำซ้ำภายในแอ็คติวิตี <while> ไปจนกว่าค่าของนิพจน์ $numberOfDrivers < 5 จะเป็น false

```
<pick name="CustomerResponseToQuote">
  <onMessage operation="requestMore"
            partnerLink="ProcessQuotePL"
            portType="ProcessQuotePT"
            variable="furtherConsiderationReq">
    <correlations>
      <correlation initiate="no" set="applicantEMailAddr" />
    </correlations>
    <sequence name="customerWantsUnderwriterToContactPostReview">
      <assign name="copyQuoteForReview">
        <copy>
          .......
        </copy>
      </assign>
       <invoke name="underwriterFollowupAndCall"
              partnerLink="UnderwriterPL"
              portType=" UnderwriterPT"
              operation="retrieveAndFollowUp"
              inputVariable="underwriterReview"/>
    </sequence>
  </onMessage>
  <onAlarm>
    <for>'PT5M'</for>
    <invoke .../>
  </onAlarm>
</pick>
```

รูปที่ 2.26 ตัวอย่างของแอ็คติวิตี <pick>

```
<repeatUntil>
  <invoke name="GetUnderwriterWorkLoad"
        partnerLink="UnderwriterPL"
        operation="getWorkLoad"
        outputVariable="underwriterStatus" />
  <condition>
    <![CDATA[$underwriterStatus.result > 100]]>
  </condition>
</repeatUntil>
```

รูปที่ 2.27 ตัวอย่างของแอ็คติวิตี <repeatUntil>

```
<sequence>
  <!- wait for 1 year ->
  <wait>
    <for>'P1Y'</for>
  </wait>
  <invoke name="ContactCustomer"
        partnerLink="PrintShopPL"
        portType="PrintShopPT"
        operation="sendFormLetterByEmail"
        inputVariable="customerDetail"/>
</sequence>
```

รูปที่ 2.28 ตัวอย่างของแอ็คติวิตี <sequence>

```
<while>
  <condition><![CDATA[$numberOfDrivers < 5]]></condition>
  <invoke ... />
</while>
```

รูปที่ 2.29 ตัวอย่างของแอ็คติวิตี <while>

2.5 งานวิจัยที่เกี่ยวข้อง

2.5.1 งานวิจัย An Experimental Determination of Sufficient Mutation Operators โดย Offuttและคณะ [7]

งานวิจัยนี้ได้ทำการปรับปรุงงานวิจัย [4, 6] ที่ได้ทดลองใช้วิธีการเลือกตัวดำเนินการมิวเทชันกับระบบ Mothra โดยมีการนิยามและทดลองวิธีการเลือกตัวดำเนินมิวเทชันแบบ 2 3 และ 6 ซึ่งงานวิจัยนี้ได้นิยามการทดสอบแบบมิวเทชันโดยการคัดเลือกขึ้นมาใหม่โดยแบ่งตัวดำเนินการมิวเทชันออกเป็น 3 กลุ่ม คือ กลุ่มตัวดำเนินการมิวเทชันแทนที่ตัวถูกดำเนินการมี 11 ตัวดำเนินการ กลุ่มตัวดำเนินการมิวเทชันดัดแปลงนิพจน์มี 5 ตัวดำเนินการ และกลุ่มตัวดำเนินการมิวเทชันดัดแปลงข้อความสั่งมี 6 ตัวดำเนินการ และนิยามการทดสอบแบบมิวเทชันโดยการคัดเลือกตามกลุ่มที่แบ่งไว้ซึ่งงานวิจัยนี้นิยามการทดสอบแบบมิวเทชันโดยการคัดเลือกออกเป็น 4 วิธี คือ

1) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มนิพจน์และกลุ่มดัดแปลงข้อความสั่ง

2) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงข้อความสั่ง

3) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มแทนที่ตัวถูกดำเนินการและกลุ่มดัดแปลงนิพจน์

4) การเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มดัดแปลงนิพจน์

งานวิจัยนี้ได้ทำการทดลองเปรียบเทียบประสิทธิภาพระหว่างการทดสอบแบบมิวเทชันแบบดั้งเดิมกับการทดสอบแบบมิวเทชันที่ใช้การทดสอบแบบมิวเทชันโดยการคัดเลือกแบบกลุ่ม โดยที่ประสิทธิภาพของการทดสอบสามารถวัดได้จากค่าของคะแนนมิวเทชันของการทดสอบแบบมิวเทชันที่ใช้ตัวดำเนินการมิวเทชัน 22 ตัวที่กระทำการทดสอบด้วยกรณีทดสอบที่สามารถกำจัดมิวแตนท์ได้ทั้งหมดจากการทดสอบแบบมิวเทชันในแต่ละกลุ่ม ซึ่งจากผลการทดลองสามารถสรุปได้ว่าตัวดำเนินการมิวเทชันกลุ่มแทนที่ตัวถูกดำเนินการและตัวดำเนินการมิวเทชันกลุ่มดัดแปลงข้อความสั่งให้ประสิทธิภาพที่ต่ำต่อการทดสอบแบบมิวเทชัน ดังนั้นระบบ Mothra จึงสามารถใช้ตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์เพียงกลุ่มเดียวก็สามารถให้

ประสิทธิภาพได้ใกล้เคียงกับการทดสอบแบบมิวเทชันแบบดั้งเดิมแต่สามารถลดจำนวนมิวแตนท์ได้ถึง 4 เท่า

วิทยานิพนธ์นี้มีการประยุกต์ใช้การทดสอบแบบมิวเทชันโดยการคัดเลือกแบบกลุ่มเพื่อลดจำนวนตัวดำเนินการมิวเทชัน โดยจะสนใจเพียงตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์เท่านั้น

2.5.2 งานวิจัย “Mutation Operator for Ada” โดย Offutt และคณะ [12]

งานวิจัยนี้ได้นิยามตัวดำเนินการมิวเทชันสำหรับภาษาโปรแกรมเอดา ซึ่งได้สร้างตัวดำเนินการ สำหรับทุกไวยากรณ์ของภาษาเอดา รวมไปถึงการควบคุมสิ่งผิดปกติ (Exception handling) เจเนริก (Generics) และทาส์กกิ้ง (Tasking) สำหรับตัวดำเนินการมิวเทชันที่งานวิจัยนี้นิยามสำหรับภาษาโปรแกรมเอดาสามารถแบ่งได้ 5 ชนิด คือ

1) ตัวดำเนินการที่ใช้ในการแทนที่ตัวถูกดำเนินการ (Operand Replacement Operators) มี 30 ตัวดำเนินการ

2) ตัวดำเนินการสำหรับข้อความสั่ง (Statement Operators) มี 14 ตัวดำเนินการ

3) ตัวดำเนินการสำหรับนิพจน์ (Expression Operators) มี 14 ตัวดำเนินการ

4) ตัวดำเนินการสำหรับความครอบคลุม (Coverage Operators) มี 4 ตัวดำเนินการ

5) ตัวดำเนินการสำหรับทาส์กกิ้ง (Tasking Operators) มี 3 ตัวดำเนินการ

งานวิจัยนี้ได้นำการทดสอบแบบมิวเทชันโดยการคัดเลือกมาใช้ทำให้สามารถลดจำนวนตัวดำเนินการมิวเทชันเหลือเพียง 16 ตัว

งานวิจัยนี้ได้นิยามตัวดำเนินการมิวเทชันสำหรับภาษาเอดาซึ่งตัวดำเนินการมิวเทชันเหล่านี้ใช้ได้กับภาษาเอดาเท่านั้น

เนื่องจากวิทยานิพนธ์นี้นำวิธีการทดสอบแบบมิวเทชันมาใช้กับภาษาบีเพลซึ่งจำเป็นต้องมีการนิยามตัวดำเนินการมิวเทชันสำหรับภาษาบีเพล โดยวิทยานิพนธ์นี้สนใจ

เฉพาะตัวดำเนินการมิวเทชันในกลุ่มดัดแปลงนิพจน์ ซึ่งการนิยามตัวดำเนินการมิวเทชันสำหรับวิทยานิพนธ์นี้จะดัดแปลงมาจากตัวดำเนินการสำหรับนิพจน์ของภาษาเอดา

บทที่ 3
การสร้างมิวแตนท์

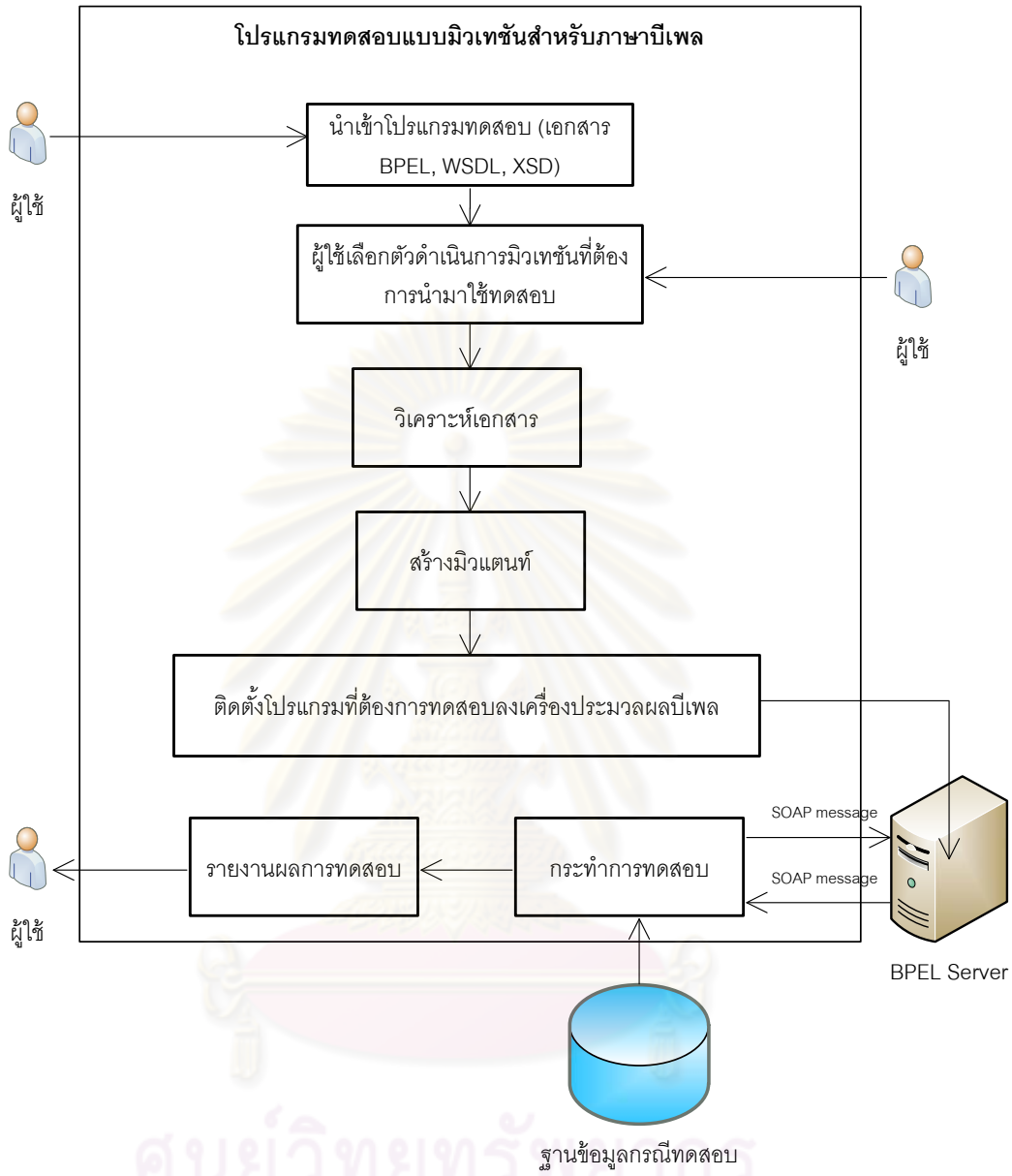
ในบทนี้จะเริ่มต้นด้วยการนำเสนอภาพรวมของวิธีการที่นำเสนอ วิธีการคัดเลือกตัวดำเนินการมิวเทชันในกลุ่มดัดแปลงนิพจน์สำหรับภาษาบีเพล การดัดแปลงนิพจน์ของภาษาบีเพล การวิเคราะห์และออกแบบเครื่องมือ และรูปแบบของรายงาน ซึ่งในส่วนของการวิเคราะห์และออกแบบเครื่องมือจะอธิบายด้วย แผนภาพยูสเคส (Use case diagram) แผนภาพคลาส (Class diagram) และแผนภาพลำดับ (Sequence diagram)

3.1 ภาพรวมของวิธีการที่นำเสนอ

วิทยานิพนธ์ฉบับนี้เสนอการประยุกต์ใช้การทดสอบแบบมิวเทชันกับภาษาบีเพล ซึ่งจากผลการทดลองของงานวิจัย [7] แสดงให้เห็นว่าการเลือกตัวดำเนินการมิวเทชันเฉพาะกลุ่มดัดแปลงนิพจน์ ให้ประสิทธิภาพใกล้เคียงกับการทดสอบแบบมิวเทชันแบบดั้งเดิมดังนั้นงานวิจัยนี้จึงสนใจเฉพาะตัวดำเนินการมิวเทชันเฉพาะกลุ่มดัดแปลงนิพจน์ ซึ่งสามารถนิยามตัวดำเนินการมิวเทชันสำหรับภาษาบีเพลได้ 5 ตัวดำเนินการ โดยรายละเอียดอยู่ในหัวข้อที่ 3.2

การสร้างมิวแตนท์นั้นทำได้โดยการแปลงเอกสารบีเพลเป็นเอกสารดอม (DOM document) ซึ่งจะทำให้สามารถจัดการและสามารถค้นหานิพจน์เพื่อจะดัดแปลงนิพจน์โดยการใช้ตัวดำเนินการมิวเทชันทั้ง 5 ตัวที่นิยามไว้ในข้อ 3.2 หลังจากดัดแปลงนิพจน์แล้วจึงแปลงเอกสารดอมกลับเป็นเอกสารบีเพลเพื่อนำไปทดสอบต่อไป โดยรายละเอียดของการสร้างมิวแตนท์อยู่ในหัวข้อที่ 3.3

เนื่องจากการทดสอบแบบมิวเทชันนั้นสร้างมิวเตนท์ออกมาจำนวนมากทำให้ยากต่อการทดสอบ ดังนั้นในการนำไปประยุกต์ใช้จริงจึงต้องสร้างเครื่องมือสำหรับการทดสอบแบบมิวเทชัน ซึ่งมีหน้าที่ดัดแปลงโปรแกรมต้นฉบับให้เป็นโปรแกรมมิวแตนท์ได้อย่างอัตโนมัติ เรียกใช้งานโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์ด้วยกรณีทดสอบที่ผู้ใช้สร้างขึ้นได้อย่างอัตโนมัติ และรายงานผลการทดสอบ โครงสร้างการทำงานของเครื่องมือแสดงดังรูปที่ 3.1 และรายละเอียดต่างๆ ของเครื่องมืออยู่ในหัวข้อที่ 3.4



รูปที่ 3.1 โครงสร้างการทำงานของเครื่องมือ

3.1.1 ข้อมูลนำเข้า

ข้อมูลนำเข้านี้ผู้ใช้ระบบจะเป็นผู้นำเข้าโดยประกอบไปด้วย เอกสารโปรแกรมต้นฉบับ และตัวดำเนินการมิวเทชันที่ผู้ใช้เลือก

1) ไฟล์โปรแกรมต้นฉบับ ประกอบไปด้วย เอกสารบีเพล เอกสารดับเบิลยูเอสดีแอล และเอกสารเอกซ์เอ็มแอลสคีมา

2) ตัวดำเนินการมิวเทชันที่ผู้ใช้เลือก ผู้ใช้เป็นผู้เลือกว่าต้องการใช้ตัวดำเนินการมิวเทชันใดกับโปรแกรมทดสอบบ้าง ซึ่งเครื่องมือจะนำตัวดำเนินการมิวเทชันเหล่านี้ไปใช้ในการสร้างมิวแตนท์

3.1.2 ส่วนวิเคราะห์เอกสาร

ส่วนการวิเคราะห์เอกสารนี้จะนำเอกสารที่นำเข้าจากหัวข้อ 3.1.1 ซึ่งเอกสารนำเข้าคือ เอกสารเอกสารบีเพล เอกสารดับเบิลยูเอสดีแอล และเอกสารเอ็กซ์เอ็มแอลสคีมาที่ผู้ใช้เป็นผู้นำเข้าโดยที่การวิเคราะห์เอกสารบีเพลนั้นจะเป็นการแยกองค์ประกอบของกระบวนการ ตัวแปร ข้อความสั่ง และนิพจน์ ดังรูปที่ 3.2 แสดงตัวอย่างของเอกสารบีเพล ซึ่งภายในเอกสารนี้มีการประกาศตัวแปร 2 ตัวคือ outputVar เป็นตัวแปรชนิด responseMessage และ inputVar เป็นตัวแปรชนิด requestMessage ตัวอย่างของข้อความสั่งคือข้อความสั่ง if ซึ่งข้อความสั่งนี้จะขึ้นอยู่กับผลของนิพจน์บูลลีน ($inputVar.inputType/ns2:paramA = 'Natthapol') ที่อยู่ภายในภายในอีลีเมนต์ <condition> ถ้าค่าที่อยู่ในตัวแปร inputVar ใน part paramA คือ Natthapol นิพจน์บูลลีน ($inputVar.inputType/ns2:paramA = 'Natthapol') จะได้ผลลัพธ์เป็น true ส่งผลให้กระทำการข้อความสั่ง assign ซึ่งจะทำการสำเนาข้อความ Hello Natthapol ไปยังตัวแปร outputVar ใน part paramA ถ้าค่าที่อยู่ในตัวแปร inputVar ใน part paramA ไม่ใช่ Natthapol นิพจน์บูลลีน($inputVar.inputType/ns2:paramA = 'Natthapol') จะได้ผลลัพธ์เป็น false ส่งผลให้กระทำทำการข้อความสั่ง assign ซึ่งจะทำการสำเนาข้อความที่ได้จากนิพจน์ concat('Hello ', $inputVar.inputType/ns2:paramA) ไปยัง outputVar ใน part paramA

ในส่วนของการวิเคราะห์เอกสารดับเบิลยูเอสดีแอลและเอกสารเอ็กซ์เอ็มแอลสคีมานั้นจะเป็นการวิเคราะห์ การดำเนินการ แมสเสจ และชนิดของพารามิเตอร์ เพื่อนำข้อมูลเหล่านี้มาสร้างส่วนของการรับค่าของกรณีทดสอบจากผู้ใช้

```
……..
   <partnerLinks>
      <partnerLink name="SynchronousSample"
                   partnerLinkType="ns1:partnerlinktype1"
                   myRole="partnerlinktyperole1"/>
   </partnerLinks>
   <variables>
      <variable name="outputVar" messageType="ns1:responseMessage"/>
      <variable name="inputVar" messageType="ns1:requestMessage"/>
   </variables>
    <sequence>
      <receive name="start"
               partnerLink="SynchronousSample"
               operation="operation1"
               portType="ns1:portType1"
               variable="inputVar"
               createInstance="yes">
      </receive>
      <if name="If1">
         <condition> ( $inputVar.inputType/ns2:paramA = 'Natthapol' ) </condition>
         <assign name="Assign1">
            <copy>
               <from>
                  <literal>Hello Natthapol</literal>
               </from>
               <to>$outputVar.resultType/ns2:paramA</to>
            </copy>
         </assign>
         <else>
            <assign name="Assign2">
               <copy>
                  <from>concat('Hello ', $inputVar.inputType/ns2:paramA)</from>
                  <to>$outputVar.resultType/ns2:paramA</to>
               </copy>
            </assign>
         </else>
      </if>
```

รูปที่ 3.2 ตัวอย่างเอกสารบีเพล

3.1.3 ตัวสร้างมิวแตนท์ (Mutant Generator)

โปรแกรมส่วนนี้มีหน้าที่ในการนำข้อมูลจากการวิเคราะห์เอกสาร และตัวดำเนินการมิวเทชันที่ผู้ใช้เลือก มาใช้ในการดัดแปลงเอกสารโปรแกรมต้นฉบับให้เป็นโปรแกรมมิวแตนท์ ดังตัวอย่างในรูปที่ 3.3 แสดงตัวอย่างโปรแกรมต้นฉบับ และรูปที่ 3.4 แสดงตัวอย่างของโปรแกรมมิวแตนท์ที่แทนที่ตัวดำเนินการ = ด้วย != ที่เครื่องมือสร้างออกมา

```
<if name="If1">
    <condition> ( $inputVar.inputType/ns2:paramA = 'Natthapol' ) </condition>
    <assign name="Assign1">
      <copy>
        <from>
          <literal>Hello Natthapol</literal>
        </from>
        <to>$outputVar.resultType/ns2:paramA</to>
      </copy>
    </assign>
    <else>
      <assign name="Assign2">
        <copy>
          <from>concat('Hello ', $inputVar.inputType/ns2:paramA)</from>
          <to>$outputVar.resultType/ns2:paramA</to>
        </copy>
      </assign>
    </else>
  </if>
```

รูปที่ 3.3 ตัวอย่างของโปรแกรมต้นฉบับภาษาบีเพล

```
<if name="If1">
  <condition> ( $inputVar.inputType/ns2:paramA != 'Natthapol' ) </condition>
  <assign name="Assign1">
    <copy>
      <from>
        <literal>Hello Natthapol</literal>
      </from>
      <to>$outputVar.resultType/ns2:paramA</to>
    </copy>
  </assign>
  <else>
    <assign name="Assign2">
      <copy>
        <from>concat('Hello ', $inputVar.inputType/ns2:paramA)</from>
        <to>$outputVar.resultType/ns2:paramA</to>
      </copy>
    </assign>
  </else>
</if>
```

เปลี่ยนจากตัวดำเนินการ = เป็น !=

รูปที่ 3.4 ตัวอย่างโปรแกรมมิวแตนท์ที่เครื่องมือสร้างออกมา

3.1.4 ส่วนติดตั้งโปรแกรม

โปรแกรมส่วนนี้มีหน้าที่ในการติดตั้งโปรแกรมลงในเครื่องประมวลผลบีเพล (BPEL Engine) ซึ่งโปรแกรมเหล่านี้คือ โปรแกรมต้นฉบับ และโปรแกรมมิวแตนท์ เครื่องมือจะเรียกใช้เซอร์วิซของโปรแกรมมิวแตนท์และเปรียบเทียบกับผลลัพธ์กับโปรแกรมต้นฉบับ

3.1.5 ส่วนเรียกใช้เซอร์วิซและรายงานผลการทดสอบ

ส่วนนี้มีหน้าที่ในการเรียกใช้เซอร์วิซ ที่ส่วนติดตั้งกระบวนการ ได้ติดตั้งลงเครื่องประมวลผลบีเพลไว้แล้วด้วยกรณีทดสอบที่อยู่ในฐานข้อมูลกรณีทดสอบ ทั้งโปรแกรม

ต้นฉบับและโปรแกรมมิวแตนท์ และนำผลลัพธ์ที่ได้จากโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์มาเปรียบเทียบกัน ถ้าข้อมูลนำออกของโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์ต่างกันแสดงว่ามิวแตนท์ตัวนั้นถูกกำจัดเรียบร้อยแล้ว แต่ถ้าหากข้อมูลนำออกของโปรแกรมมิวแตนท์เหมือนกับข้อมูลนำออกของโปรแกรมต้นฉบับแสดงว่ามิวแตนท์ตัวนั้นเป็นมิวแตนท์ที่ยังคงมีชีวิตอยู่และคำนวณหาค่าร้อยละของมิวแตนท์ที่ถูกกำจัด ซึ่งสุดท้ายโปรแกรมส่วนนี้จะสรุปข้อมูลและรายงานให้กับผู้ใช้

3.2 การคัดเลือกตัวดำเนินการมิวเทชันในกลุ่มดัดแปลงนิพจน์

จากผลของงานวิจัย [7] แสดงให้เห็นว่าการเลือกตัวดำเนินการเฉพาะกลุ่มดัดแปลงนิพจน์ ให้ประสิทธิภาพใกล้เคียงกับการทดสอบแบบมิวเทชันแบบดั้งเดิม ดังนั้นวิทยานิพนธ์นี้จึงสนใจเฉพาะตัวดำเนินการกลุ่มดัดแปลงนิพจน์ ซึ่งภาษาบีเพลมีนิพจน์อยู่ 5 ชนิดตามที่ระบุไว้ในหัวข้อ 2.5.3 แต่นิพจน์ของภาษาบีเพลที่งานวิจัยนี้สนใจ คือ นิพจน์บูลลีน นิพจน์ทั่วไป และนิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย เนื่องจากนิพจน์ค่าเส้นตาย และนิพจน์ค่าช่วงเวลา ไม่อยู่ในกลุ่มดัดแปลงนิพจน์ แต่อยู่ในกลุ่มตัวดำเนินการมิวเทชันแทนที่ตัวถูกดำเนินการ และกลุ่มตัวดำเนินการมิวเทชันดัดแปลงข้อความสั่งของตัวดำเนินการมิวเทชันในภาษาเอดา [12]

ตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์จะไปเปลี่ยนแปลงตัวดำเนินการในภาษาโปรแกรมซึ่งตัวดำเนินการในภาษาเอดามีดังต่อไปนี้

1) ตัวดำเนินการเชิงตรรกะ (Logical operators)
   มี 3 ตัว คือ and or และ xor
2) ตัวดำเนินการเชิงสัมพันธ์ (Relational operators)
   มี 6 ตัว คือ /= = < <= > และ >=
3) ตัวดำเนินการการเพิ่มแบบทวิภาค (Binary adding operators)
   มี 3 ตัว คือ + - และ &
4) ตัวดำเนินการการเพิ่มแบบเอกภาค (Unary adding operators)
   มี 2 ตัว คือ + และ -
5) ตัวดำเนินการการคูณ (Multiplying operator)
   มี 4 ตัว คือ * / mod และ rem

6) ตัวดำเนินการทำก่อนสูงสุด (Highest precedence operator)
มี 3 ตัว คือ ** not และ abs

7) ตัวดำเนินการทางลัด (Shortcut operators)
มี 2 ตัว คือ and then และ or else

8) การทดสอบการเป็นสมาชิก (Membership tests)
มี 2 ตัว คือ in และ not in

นิพจน์บูลลีนและนิพจน์ทั่วไปของภาษาบีเพลสนับสนุนตัวดำเนินการดังต่อไปนี้

1) ตัวดำเนินการเชิงคณิตศาสตร์ (Arithmetic Operators)
มี 5 ตัว คือ + - * div และ mod

2) ตัวดำเนินการเชิงสัมพันธ์ (Relational Operators)
มี 6 ตัวคือ < <= > >= != และ =

3) ตัวดำเนินการเชิงตรรกะ (Conditional Operators)
มี 3 ตัวคือ AND OR และ not()

เนื่องจากภาษาเอดามีตัวดำเนินการครอบคลุมตัวดำเนินการของภาษาบีเพล ดังนั้นตัวดำเนินการมิวเทชันของภาษาบีเพลจึงสามารถนำมาจากภาษาเอดาได้แต่มีบางตัวดำเนินการที่ภาษาเอดามีมากกว่าภาษาบีเพลดังนั้นจึงต้องมีการคัดเลือกตัวดำเนินการมิวเทชันกลุ่มดัดแปลงนิพจน์จากภาษาเอดาอีกทีหนึ่ง ซึ่งตัวดำเนินการมิวเทชันของภาษาบีเพลที่คัดเลือกมามีดังต่อไปนี้

1) AOR (Arithmetic Operator Replacement)

เป็นการทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงคณิตศาสตร์ด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่นๆ (+ - * div และ mod) ซึ่ง AOR สามารถดัดแปลงนิพจน์บูลลีน นิพจน์ทั่วไป และนิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย โดยมีจำนวนมิวแตนท์ที่สร้างขึ้นมาในแต่ละนิพจน์เท่ากับ 4N เมื่อ N คือ จำนวนตัวดำเนินการเชิงคณิตศาสตร์ในนิพจน์

2) ROR (Relational Operator Replacement)

เป็นการทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงสัมพันธ์ด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่นๆ(< <= > >= != และ =) ซึ่ง ROR สามารถดัดแปลงนิพจน์

บูลลีน โดยมีจำนวนมิวแตนท์ที่สร้างขึ้นมาในแต่ละนิพจน์เท่ากับ 5N เมื่อ N คือ จำนวนตัวดำเนินการเชิงสัมพันธ์ในนิพจน์

3) LOR (Logical Operator Replacement)

เป็นการทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงตรรกะด้วยตัวดำเนินการเชิงตรรกะตัวอื่นๆ(AND และ OR) ซึ่ง LOR สามารถดัดแปลงนิพจน์บูลลีน โดยมีจำนวนมิวแตนท์ที่สร้างขึ้นมาในหนึ่งนิพจน์เท่ากับ N เมื่อ N คือ จำนวนตัวดำเนินการเชิงตรรกะในนิพจน์

4) LOD (Logical Operator Delete)

เป็นการทำให้เกิดความผิดพร่องโดยการลบตัวดำเนินการเชิงตรรกะ(not()) LOD สามารถดัดแปลงนิพจน์บูลลีน โดยมีจำนวนมิวแตนท์ที่สร้างขึ้นมาในหนึ่งนิพจน์เท่ากับ N เมื่อ N คือ จำนวนตัวดำเนินการ not() ในนิพจน์

5) LOI (Logical Operator Insertion)

เป็นการทำให้เกิดความผิดพร่องโดยการแทรกตัวดำเนินการเชิงตรรกะ (not()) LOD สามารถดัดแปลงนิพจน์บูลลีน โดยมีจำนวนมิวแตนท์ที่สร้างขึ้นมาในหนึ่งนิพจน์เท่ากับ N+1 เมื่อ N คือ จำนวนตัวดำเนินการเชิงตรรกะในนิพจน์

3.3 การดัดแปลงนิพจน์ของภาษาบีเพล

การสร้างมิวแตนท์นั้นทำได้โดยการแปลงเอกสารบีเพลเป็นเอกสารดอม (DOM document) ซึ่งจะทำให้สามารถจัดการและทำการค้นหานิพจน์เพื่อจะดัดแปลงนิพจน์โดยการใช้ตัวดำเนินการมิวเทชันทั้ง 5 ตัวที่นิยามไว้ในข้อ 3.1 หลังจากดัดแปลงนิพจน์แล้วจึงแปลงเอกสารดอมกลับเป็นเอกสารบีเพล ซึ่งการดัดแปลงนิพจน์โดยใช้ตัวดำเนินการมิวเทชันแต่ละตัวมีรายละเอียดดังต่อไปนี้

3.3.1 การดัดแปลงนิพจน์ของบีเพลโดยใช้ตัวดำเนินการ AOR

ตัวดำเนินการมิวเทชัน AOR เป็นตัวดำเนินการที่ทำให้เกิดความผิดพร่องโดยการดัดแปลงตัวดำเนินการเชิงคณิตศาสตร์ของภาษาบีเพล ซึ่งตัวดำเนินการเหล่านี้ คือ + - * div และ mod โดยที่ตัวดำเนินการเชิงคณิตศาสตร์เหล่านี้สามารถอยู่ใน นิพจน์บูลลีน นิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย และนิพจน์ทั่วไป ซึ่งขั้นตอนการดัดแปลงมีดังต่อไปนี้

1) สำหรับทุกๆแอ็คติวิตี if elseif while และ repeatUtil ในเอกสารบีเพลทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ condition โดยการแทนที่ทุกๆตัวดำเนินการเชิงคณิตศาสตร์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

2) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ target และภายในอีลีเมนต์ target มีอีลีเมต์ joinCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ joinCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงคณิตศาสตร์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ sources ซึ่งภายในอีลีเมนต์ sources มีอีลีเมนต์ source อยู่ภายในอย่างน้อย 1 อีลีเมนต์และภายในอีลีเมต์ source มีอีลีเมนต์ transitionCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ transitionCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงคณิตศาสตร์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

4) สำหรับทุกๆแอ็คติวิตี forEach ในเอกสารบีเพล ทำการดัดแปลงนิพจน์จำนวนเต็มไม่ระบุเครื่องหมายภายในอีลีเมนต์ startCounterValue และ อีลีเมนต์ finalCounterValue โดยการแทนที่ทุกๆตัวดำเนินการเชิงคณิตศาสตร์ที่อยู่ในนิพจน์จำนวนเต็มไม่ระบบเครื่องหมายด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่น โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3.3.2 การดัดแปลงนิพจน์ของบีเพลโดยใช้ตัวดำเนินการ ROR

ตัวดำเนินการมิวเทชัน ROR เป็นตัวดำเนินการที่ทำให้เกิดความผิดพร่องโดยการดัดแปลงตัวดำเนินการเชิงสัมพันธ์ของภาษาบีเพล ซึ่งตัวดำเนินการเหล่านี้ คือ < <= > >= และ = โดยที่ตัวดำเนินการเชิงสัมพันธ์เหล่านี้สามารถอยู่ใน นิพจน์บูลลีน และนิพจน์ทั่วไป ซึ่งขั้นตอนการดัดแปลงมีดังต่อไปนี้

1) สำหรับทุกๆแอ็คติวิตี if elseif while และ repeatUtil ในเอกสารบีเพลทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ condition โดยการแทนที่ทุกๆตัวดำเนินการเชิงสัมพันธ์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

2) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ target และภายในอีลีเมนต์ target มีอีลีเมต์ joinCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์

joinCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงสัมพันธ์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ sources ซึ่งภายในอีลีเมนต์ sources มีอีลีเมนต์ source อยู่ภายในอย่างน้อย 1 อีลีเมนต์และภายในอีลีเมต์ source มีอีลีเมนต์ transitionCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ transitionCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงสัมพันธ์ที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่น โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3.3.3 การดัดแปลงนิพจน์ของบีเพลโดยใช้ตัวดำเนินการ LOR

ตัวดำเนินการมิวเทชัน LOR เป็นตัวดำเนินการที่ทำให้เกิดความผิดพร่องโดยการดัดแปลงตัวดำเนินการเชิงตรรกะของภาษาบีเพล ซึ่งตัวดำเนินการเหล่านี้ คือ AND และ OR โดยที่ตัวดำเนินการเชิงตรรกะเหล่านี้สามารถอยู่ใน นิพจน์บูลลีน นิพจน์จำนวนเต็มไม่ระบุเครื่องหมาย และนิพจน์ทั่วไป ซึ่งขั้นตอนการดัดแปลงมีดังต่อไปนี้

1) สำหรับทุกๆแอ็คติวิตี if elseif while และ repeatUtil ในเอกสารบีเพลทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ condition โดยการแทนที่ทุกๆตัวดำเนินการเชิงตรรกะที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงตรรกะตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

2) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ target และภายในอีลีเมนต์ target มีอีลีเมต์ joinCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ joinCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงตรรกะที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงตรรกะตัวอื่นๆ โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ sources ซึ่งภายในอีลีเมนต์ sources มีอีลีเมนต์ source อยู่ภายในอย่างน้อย 1 อีลีเมนต์และภายในอีลีเมต์ source มีอีลีเมนต์ transitionCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ transitionCondition โดยการแทนที่ทุกๆตัวดำเนินการเชิงตรรกะที่อยู่ในนิพจน์บูลลีนด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่น โดยการแทนที่แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3.3.4 การดัดแปลงนิพจน์ของบีเพลโดยใช้ตัวดำเนินการ LOD

ตัวดำเนินการมิวเทชัน LOD เป็นตัวดำเนินการที่ทำให้เกิดความผิดพร่องโดยการลบตัวดำเนินการเชิงตรรกะ not ของภาษาบีเพล ซึ่งตัวดำเนินการ not นี้สามารถอยู่ใน นิพจน์บูลลีน และนิพจน์ทั่วไป ซึ่งขั้นตอนการดัดแปลงมีดังต่อไปนี้

1) สำหรับทุกๆแอ็คติวิตี if elseif while และ repeatUtil ในเอกสารบีเพลทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ condition โดยการลบตัวดำเนินการ not ที่อยู่ในนิพจน์บูลลีน โดยการลบตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

2) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ target และภายในอีลีเมนต์ target มีอีลีเมต์ joinCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ joinCondition โดยการลบตัวดำเนินการ not ที่อยู่ในนิพจน์บูลลีน โดยการลบตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ sources ซึ่งภายในอีลีเมนต์ sources มีอีลีเมนต์ source อยู่ภายในอย่างน้อย 1 อีลีเมนต์และภายในอีลีเมต์ source มีอีลีเมนต์ transitionCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ transitionCondition โดยการลบตัวดำเนินการ not ที่อยู่ในนิพจน์บูลลีน โดยการลบตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3.3.5 การดัดแปลงนิพจน์ของบีเพลโดยใช้ตัวดำเนินการ LOI

ตัวดำเนินการมิวเทชัน LOI เป็นตัวดำเนินการที่ทำให้เกิดความผิดพร่องโดยการเพิ่มตัวดำเนินการเชิงตรรกะ not ของภาษาบีเพล ซึ่งตัวดำเนินการ not นี้สามารถอยู่ในนิพจน์บูลลีน และนิพจน์ทั่วไป ซึ่งขั้นตอนการดัดแปลงมีดังต่อไปนี้

1) สำหรับทุกๆแอ็คติวิตี if elseif while และ repeatUtil ในเอกสารบีเพลทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ condition โดยการแทรกตัวดำเนินการ not เข้าเข้าไปในนิพจน์บูลลีนและในทุกๆนิพจน์ย่อยในนิพจน์บูลลีน โดยการแทรกตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

2) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ target และภายในอีลีเมนต์ target มีอีลีเมต์ joinCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ joinCondition โดยการแทรกตัวดำเนินการ not เข้าไปในนิพจน์บูลลีนและในทุกๆนิพจน์ย่อยในนิพจน์บูลลีน โดยการแทรกตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

3) สำหรับทุกๆแอ็คติวิตีในเอกสารบีเพลถ้ามีอีลีเมนต์ sources ซึ่งภายในอีลีเมนต์ sources มีอีลีเมนต์ source อยู่ภายในอย่างน้อย 1 อีลีเมนต์และภายในอีลีเมต์ source มีอีลีเมนต์ transitionCondition ทำการดัดแปลงนิพจน์บูลลีนภายในอีลีเมนต์ โดยการแทรกตัวดำเนินการ not เข้าเข้าไปในนิพจน์บูลลีนและในทุกๆนิพจน์ย่อยในนิพจน์บูลลีน โดยการแทรกตัวดำเนินการ not แต่ละครั้งจะได้มิวแตนท์ขึ้นมา 1 โปรแกรม

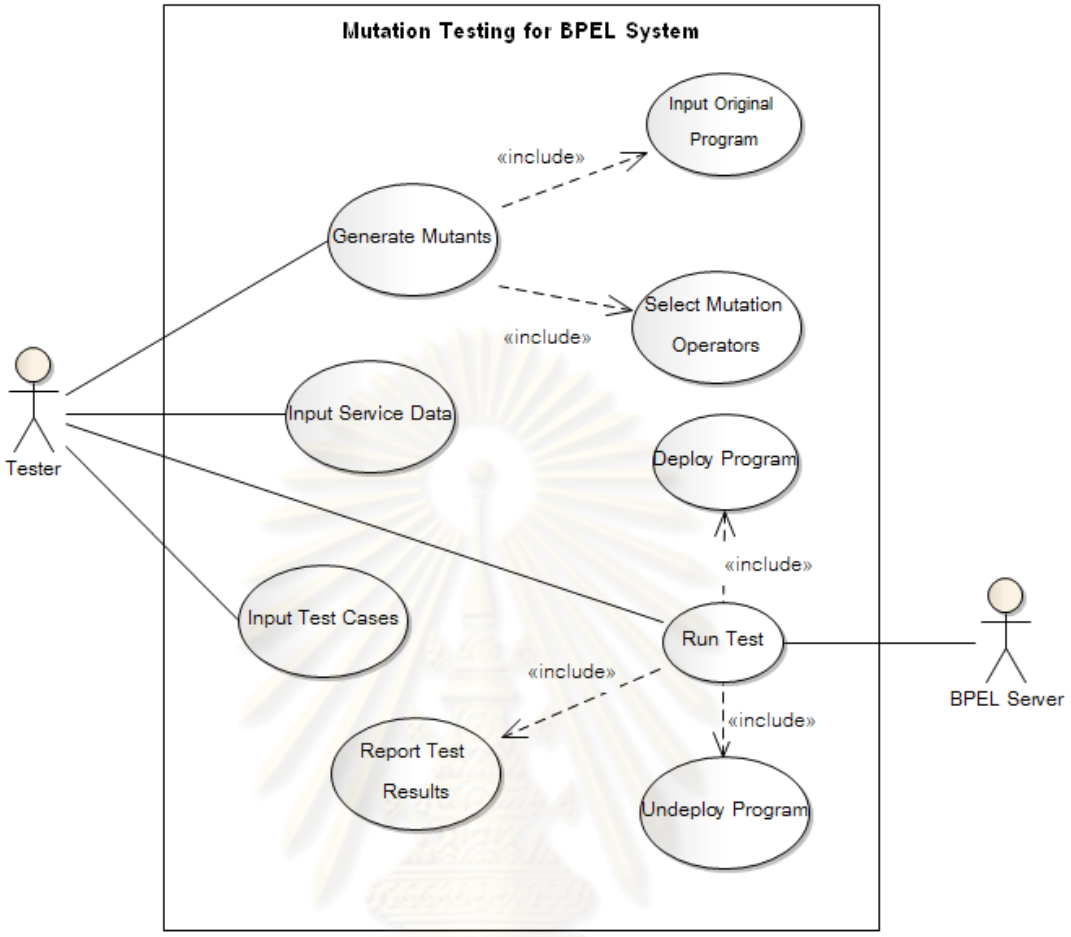
3.4 การวิเคราะห์และออกแบบเครื่องมือ

เครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลนั้นถูกสร้างขึ้นมาเพื่ออำนวยความสะดวกในการทดสอบแบบมิวเทชัน ซึ่งมีหน้าที่ดัดแปลงโปรแกรมต้นฉบับให้เป็นโปรแกรมมิวแตนท์ เรียกใช้งาน เปรียบเทียบผลลัพธ์ของโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์ และรายงานผล ซึ่งมีรายละเอียดต่อไปนี้

3.4.1 แผนภาพยูสเคส

แผนภาพยูสเคสอธิบายขอบเขตและฟังก์ชันการทำงานพื้นฐานของระบบในมุมมองของผู้ใช้ แผนภาพยูสเคสของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลแสดงดังรูปที่ 3.5

จากแผนภาพยูสเคสรูปที่ 3.5 เริ่มต้นด้วยผู้ใช้นำเข้าโปรแกรมต้นฉบับ (Input Original Program) ซึ่งอยู่ในรูปไฟล์ซิฟ หลังจากนั้นผู้ใช้เลือกตัวดำเนินการมิวเทชัน (Select Mutation Operators) ที่ต้องการนำมาใช้สร้างมิวเตนท์ หลังจากผู้ใช้เลือกตัวดำเนินการมิวเทชันแล้วระบบจะทำการสร้างมิวแตนท์ (Generate Mutants) หลังจากนั้นระบบจะวิเคราะห์เอกสารดับเบิลยูเอสดีแอลเพื่อสร้างหน้าต่างสำหรับนำเข้าข้อมูลเว็บเซอร์วิซ (Input Service Data) และหน้าต่างสำหรับนำเข้ากรณีทดสอบ (Input Test Cases) หลังจากนั้นระบบจะกระทำการทดสอบ (Run Test) โดยการติดตั้งโปรแกรม (Deploy Program) ที่ต้องการทดสอบลงเครื่องประมวลผลบีเพลและเรียกใช้เซอร์วิซของโปรแกรมทดสอบหลังจากนั้นบันทึกผลการทดสอบและยกเลิกการติดตั้งโปรแกรม (Undeploy Program) หลังจากกระทำการทดสอบเสร็จเรียบร้อยระบบจะรายงานผลการทดสอบ (Report Test Result) ส่วนรายละเอียดของแต่ละยูสเคสแสดงในตารางที่ 3.1 – 3.9



รูปที่ 3.5 แผนภาพยูสเคสของเครื่องมือ

ตารางที่ 3.1 รายละเอียดยูสเคสนำเข้าโปรแกรมต้นฉบับ

ยูสเคส	Input Original Program
แอคเตอร์	ผู้ทดสอบ (Tester)
เป้าหมาย	นำเข้าโปรแกรมต้นฉบับสู่ระบบ
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	โปรแกรมต้นฉบับเก็บอยู่ในรูปไฟล์ซิฟ
ขั้นตอน	1. ผู้ใช้เรียกใช้โปรแกรมผ่านทางเว็บเบราว์เซอร์ 2. ระบบสร้างหน้าต่างสำหรับนำเข้าโปรแกรมต้นฉบับ 3. ผู้ใช้อัพโหลดโปรแกรมต้นฉบับ 4. ระบบบันทึกโปรแกรมต้นฉบับไว้บนเครื่องบริการเว็บ(Web Server)
เงื่อนไขภายหลัง	โปรแกรมต้นฉบับได้รับการบันทึกอยู่บนเครื่องบริการเว็บ

ตารางที่ 3.2 รายละเอียดยูสเคสเลือกตัวดำเนินการมิวเทชัน

ยูสเคส	Select Mutation Operators
แอคเตอร์	ผู้ทดสอบ
เป้าหมาย	ให้ผู้ใช้เลือกตัวดำเนินการมิวเทชันที่ต้องการนำไปสร้างมิวแตนท์
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	
ขั้นตอน	1. ระบบสร้างหน้าต่างสำหรับเลือกตัวดำเนินการมิวเทชัน 2. ผู้ใช้เลือกตัวดำเนินการมิวเทชัน 3. ระบบบันทึกตัวดำเนินการมิวเทชันที่ผู้ใช้เลือก
เงื่อนไขภายหลัง	

ตารางที่ 3.3 รายละเอียดยูสเคสสร้างมิวเตนท์

ยูสเคส	Generate Mutants
แอคเตอร์	
เป้าหมาย	สร้างมิวเตนท์จากโปรแกรมต้นฉบับ
ยูสเคสที่สัมพันธ์	Includes : Input Original Program, Select Mutation Operators
เงื่อนไขก่อนหน้า	โปรแกรมต้นฉบับได้รับการบันทึกไว้บนเครื่องบริการเว็บ
ขั้นตอน	1. นำเข้าโปรแกรมต้นฉบับโดยใช้ยูสเคส Input Original Program 2. นำเข้าชนิดของตัวดำเนินการมิวเทชันที่ต้องการใช้สร้างมิวแตนท์โดยใช้ยูสเคส Select Mutation Operators 3. ระบบสร้างโปรแกรมมิวแตนท์จากโปรแกรมต้นฉบับ โดยใช้ตัวดำเนินการมิวเทชันที่ผู้ใช้ได้เลือกไว้แล้ว 4. ระบบบันทึกโปรแกรมมิวแตนท์ทั้งหมดลงบนเครื่องบริการเว็บ
เงื่อนไขภายหลัง	มิวแตนท์ทั้งหมดได้รับการบันทึกไว้บนเครื่องบริการเว็บ

ตารางที่ 3.4 รายละเอียดยูสเคสนำเข้าข้อมูลเว็บเซอร์วิซ

ยูสเคส	Input Service Data
แอคเตอร์	ผู้ทดสอบ
เป้าหมาย	เพื่อนำเข้ากรณีทดสอบสู่ระบบ
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	
ขั้นตอน	1. ระบบสร้างหน้าต่างสำหรับกรอกข้อมูลเว็บเซอร์วิซ 2. ผู้ใช้กรอกข้อมูลเว็บเซอร์วิซและกดปุ่มบันทึก 3. ระบบบันทึกข้อมูลเว็บเซอร์วิซ
เงื่อนไขภายหลัง	

ตารางที่ 3.5 รายละเอียดยูสเคสนำเข้ากรณีทดสอบ

ยูสเคส	Input Test Cases
แอคเตอร์	ผู้ทดสอบ
เป้าหมาย	เพื่อนำเข้ากรณีทดสอบจากผู้ใช้
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	
ขั้นตอน	1. ระบบสร้างหน้าต่างสำหรับกรอกกรณีทดสอบ 2. ผู้ใช้กรอกกรณีทดสอบจนครบแล้วกดบันทึก 3. ระบบบันทึกกรณีทดสอบ
เงื่อนไขภายหลัง	

ตารางที่ 3.6 รายละเอียดยูสเคสติดตั้งโปรแกรม

ยูสเคส	Deploy Program
แอคเตอร์	เครื่องประมวลผลบีเพล (BPEL Server)
เป้าหมาย	เพื่อติดตั้งโปรแกรมลงบนเครื่องประมวลผลบีเพล (BPEL server)
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	โปรแกรมถูกบันทึกลงบนเครื่องบริการเว็บแล้ว
ขั้นตอน	ระบบเรียกใช้บริการติดตั้งโปรแกรมจากเครื่องประมวลผลบีเพล
เงื่อนไขภายหลัง	โปรแกรมได้รับการติดตั้งลงบนเครื่องประมวลผลบีเพล

ตารางที่ 3.7 รายละเอียดยูสเคสยกเลิกการติดตั้งโปรแกรม

ยูสเคส	Undeploy Program
แอคเตอร์	เครื่องประมวลผลบีเพล
เป้าหมาย	เพื่อยกเลิกการติดตั้งโปรแกรมบนเครื่องประมวลผลบีเพล
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	โปรแกรมได้รับการติดตั้งลงบนเครื่องประมวลผลบีเพลแล้ว
ขั้นตอน	ระบบเรียกใช้เซอร์วิซยกเลิกการติดตั้งโปรแกรมจากเครื่องประมวลผลบีเพล
เงื่อนไขภายหลัง	โปรแกรมได้รับการยกเลิกการติดตั้ง

ตารางที่ 3.8 รายละเอียดยูสเคสกระทำการทดสอบ

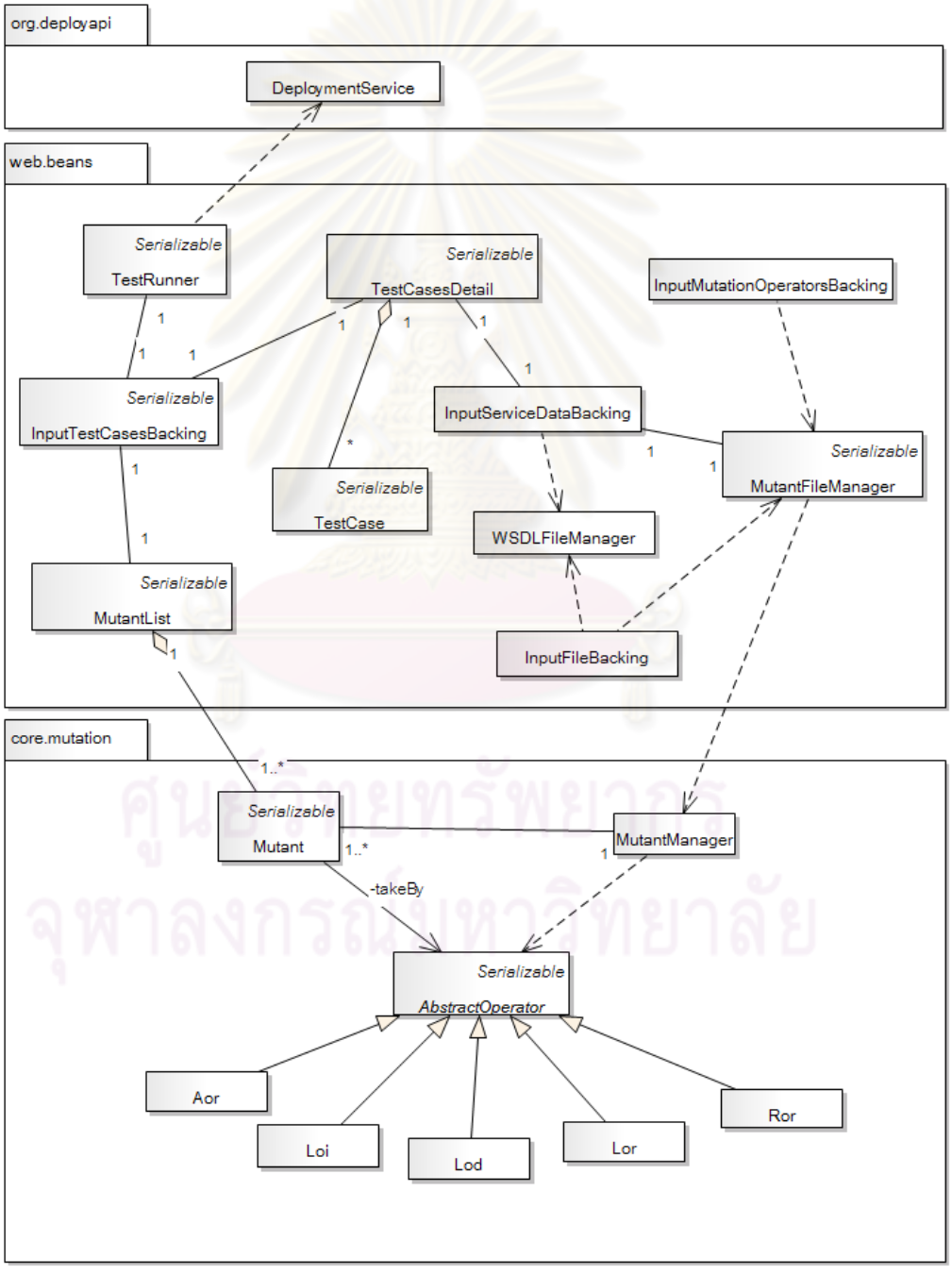
ยูสเคส	Run Test
แอคเตอร์	เครื่องประมวลผลบีเพล
เป้าหมาย	เพื่อสั่งโปรแกรมทำงานโดยใช้กรณีทดสอบของผู้ใช้
ยูสเคสที่สัมพันธ์	Includes : Deploy Program, Undeploy Program, Report Test Result
เงื่อนไขก่อนหน้า	โปรแกรมต้องได้รับการติดตั้งลงบนเครื่องประมวลผลบีเพลแล้ว
ขั้นตอน	1. ผู้ใช้กดปุ่มเริ่มทดสอบโปรแกรม 2. ติดตั้งโปรแกรมโดยใช้ยูสเคส Deploy Program 3. ระบบเรียกใช้เว็บเซอร์วิซของโปรแกรมทดสอบโดยใช้กรณีทดสอบของผู้ใช้ 4. ระบบบันทึกผลการทดสอบ 5. ยกเลิกการติดตั้งโปรแกรมโดยใช้ยูสเคส Undeploy Program 6. ทำซ้ำจากข้อ 3-6 จนครบทุกๆมิวแตนท์
เงื่อนไขภายหลัง	

ตารางที่ 3.9 รายละเอียดยูสเคสรายงานผลการทดสอบ

ยูสเคส	Report Test Result
แอคเตอร์	ผู้ทดสอบ
เป้าหมาย	เพื่อรายงานผลการทดสอบ
ยูสเคสที่สัมพันธ์	
เงื่อนไขก่อนหน้า	ทุกๆมิวแตนท์ได้รับการทดสอบแล้ว
ขั้นตอน	ระบบสร้างตารางผลการทดสอบ
เงื่อนไขภายหลัง	

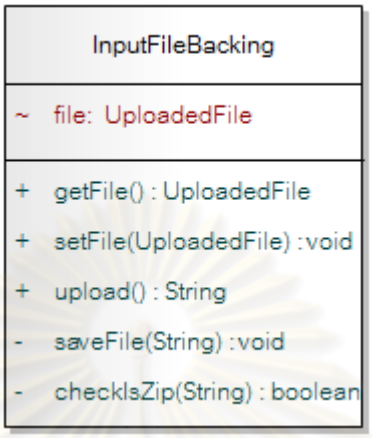
3.4.2 แผนภาพคลาส

แผนภาพคลาสใช้แสดงรายละเอียดคลาสและความสัมพันธ์ระหว่างคลาสต่างๆ เพื่อจำลองภาพการออกแบบส่วนที่เป็นโครงสร้างคงที่ของระบบ โดยในรูปที่ 3.6 เป็นแผนภาพคลาสของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล ซึ่งแต่ละคลาสสามารถแบ่งได้ตามแพคเกจ (Package) และมีรายละเอียดดังต่อไปนี้



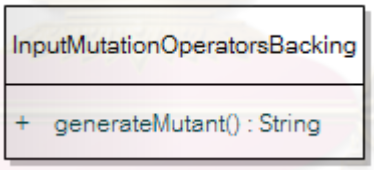
รูปที่ 3.6 แผนภาพคลาสของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล

1) คลาส InputFileBacking คือ คลาสที่ทำหน้าที่ติดต่อกับผู้ใช้ในส่วนของการนำเข้าโปรแกรมต้นฉบับ รายละเอียดของคลาสแสดงดังรูปที่ 3.7



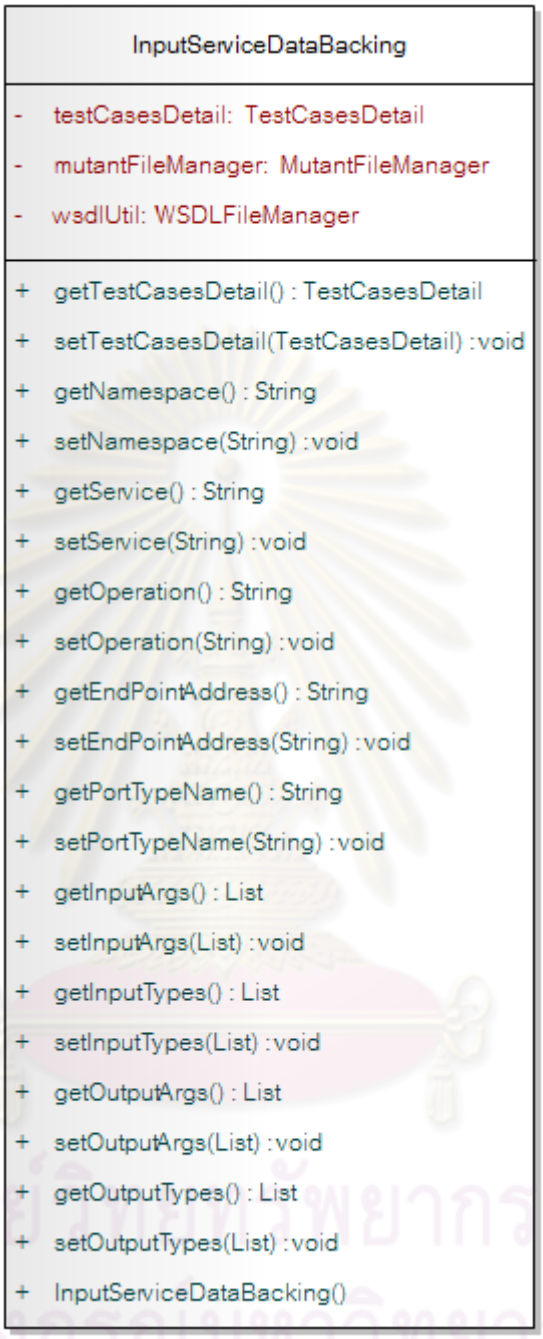
รูปที่ 3.7 คลาส InputFileBacking

2) คลาส InputMutationOperatorsBacking คือ คลาสที่ทำหน้าที่ติดต่อกับผู้ใช้ โดยการให้ผู้ใช้เลือกชนิดของตัวดำเนินการมิวเทชันที่ต้องการใช้ในการสร้างมิวแตนท์ รายละเอียดของคลาสแสดงดังรูปที่ 3.8



รูปที่ 3.8 คลาส InputMutationOperatorsBacking

3) คลาส InputServiceDataBacking คือ คลาสที่ทำหน้าที่ติดต่อกับผ็ใช้ในส่วนของการนำเข้าข้อมูลของเว็บเซอร์วิซที่ต้องการทดสอบ รายละเอียดของคลาสแสดงดังรูปที่ 3.9

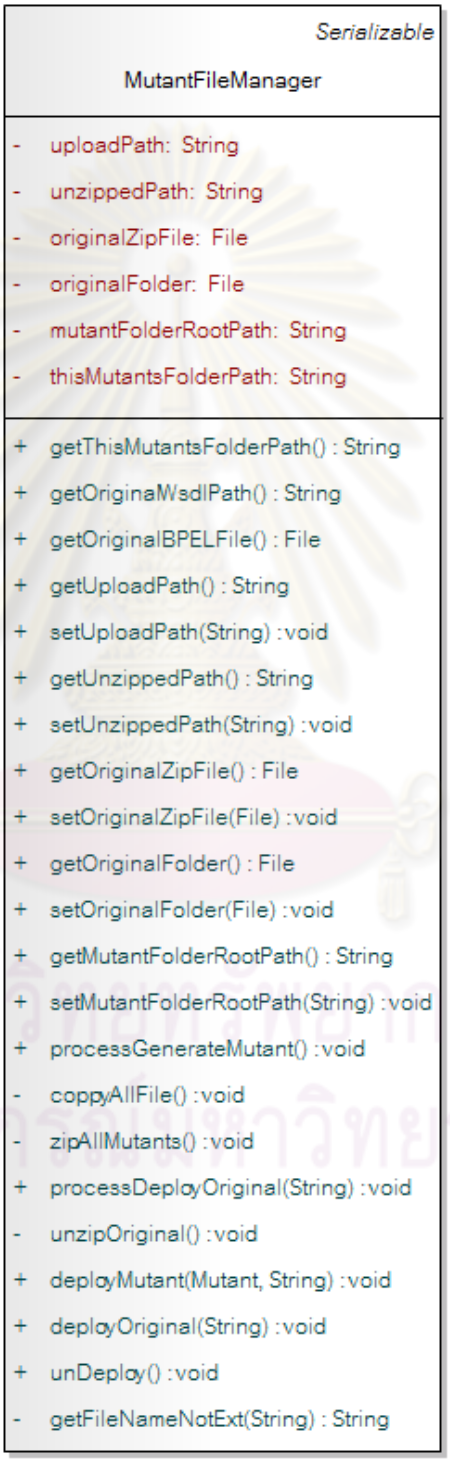


รูปที่ 3.9 คลาส InputServiceDataBacking

4) คลาส InputTestCasesBacking คือ คลาสที่ทำหน้าที่ติดต่อกับผู้ใช้ในส่วนของการนำเข้ากรณีทดสอบ และการดำเนินการทดสอบโปรแกรมทั้งโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์ รายละเอียดของคลาสแสดงดังรูปที่ 3.10

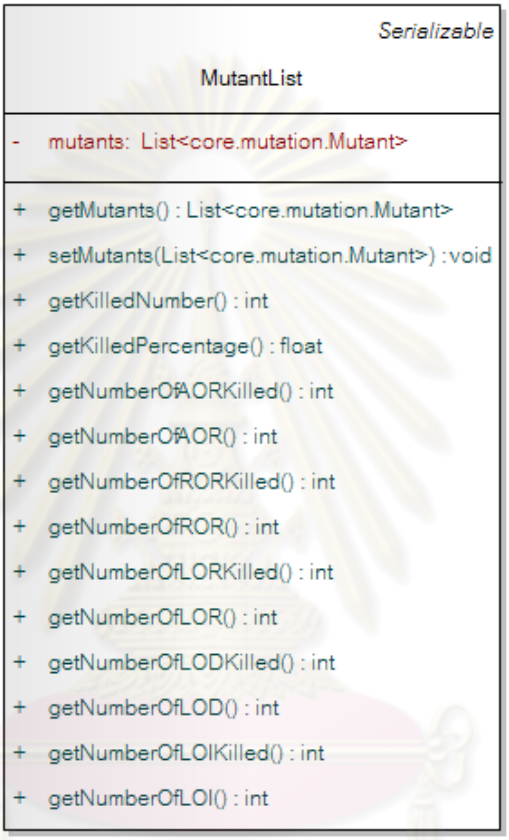
รูปที่ 3.10 คลาส InputTestCasesBacking

5) คลาส MutantFileManager คือ คลาสที่ทำหน้าที่เก็บรายละเอียดไฟล์มิวแตนท์ จัดการไฟล์มิวแตนท์ ติดตั้งมิวแตนท์ลงเครื่องประมวลผลบีเพล และยกเลิกการติดตั้งมิวแตนท์ลงเครื่องประมวลผลบีเพล รายละเอียดของคลาสแสดงดังรูปที่ 3.11



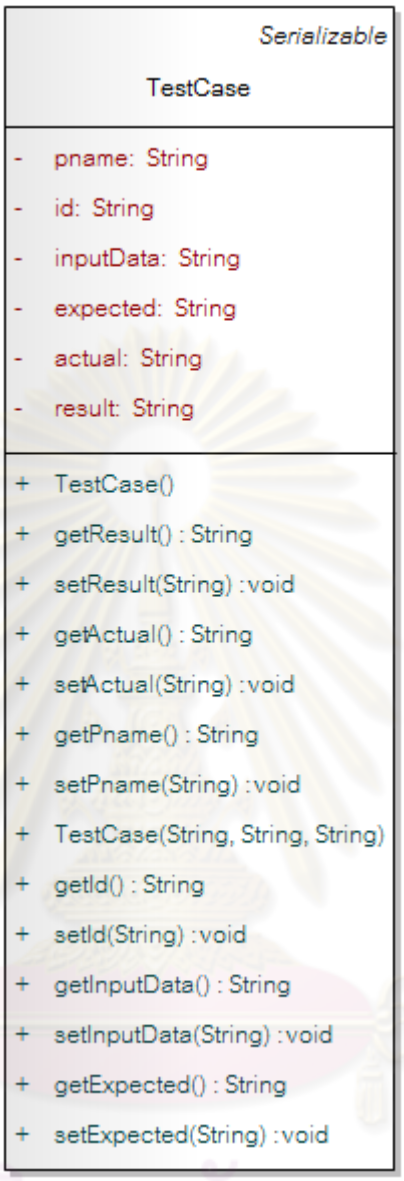
รูปที่ 3.11 คลาส MutantFileManager

6) คลาส MutantList คือคลาสที่ทำหน้าที่เก็บข้อมูลรายการของมิวแตนท์ทั้งหมด และสามารถหาจำนวนของมิวแตนท์ที่ถูกกำจัดทั้งหมด ค่าร้อยละของจำนวนมิวเตนท์ที่ถูกกำจัดต่อจำนวนมิวแตนท์ทั้งหมด และค่าของจำนวนมิวแตนท์ที่ถูกกำจัดตามตัวดำเนินการมิวเทชัน รายละเอียดของคลาสแสดงดังรูปที่ 3.12



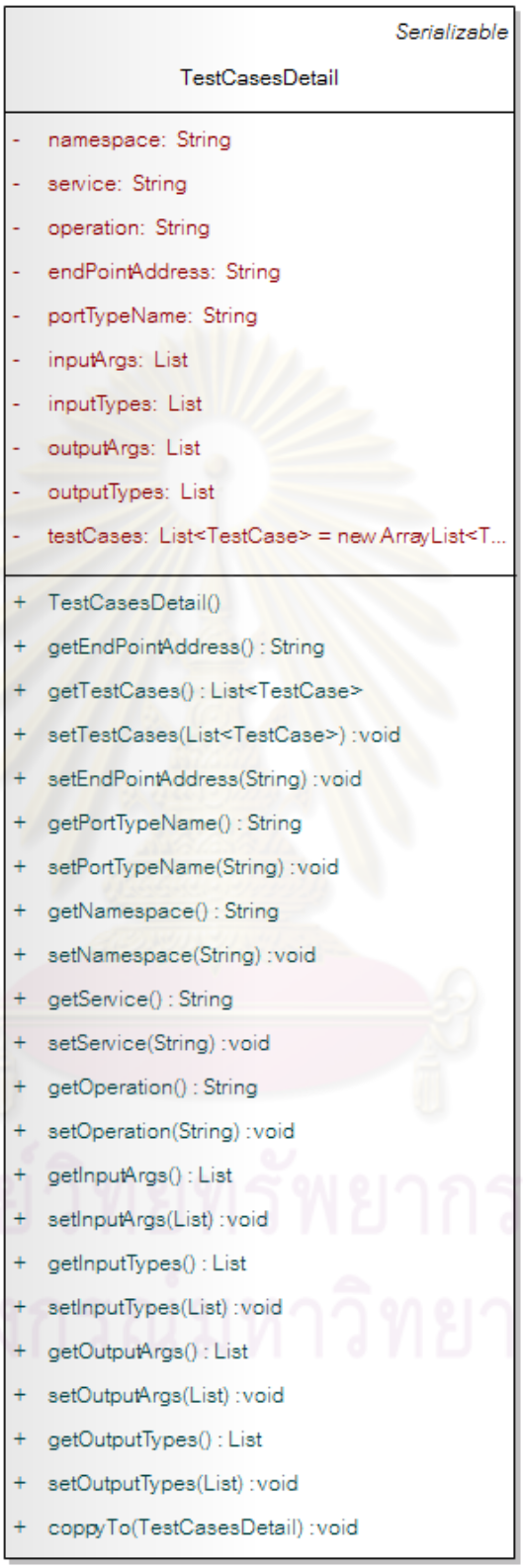
รูปที่ 3.12 คลาส MutantList

7) คลาส TestCase คือ คลาสที่เก็บรายละเอียดของกรณีทดสอบ รายละเอียดของคลาสแสดงดังรูปที่ 3.13



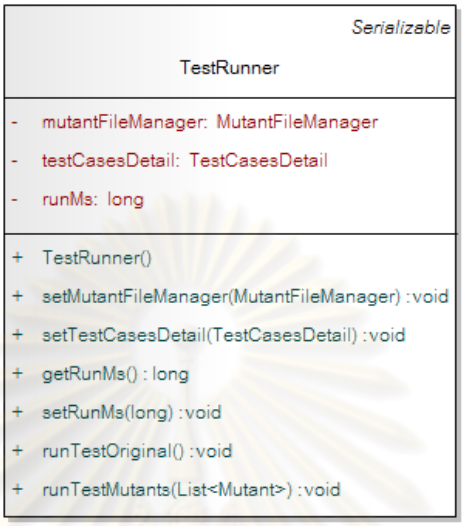
รูปที่ 3.13 คลาส TestCase

8) คลาส TestCasesDetail คือ คลาสที่เก็บรายละเอียดของข้อมูลที่ต้องใช้ในการทดสอบซึ่งภายในจะประกอบไปด้วยรายการของกรณีทดสอบทั้งหมด และข้อมูลที่ต้องใช้ในการเรียกใช้เว็บเซอร์วิซ รายละเอียดของคลาสแสดงดังรูปที่ 3.14

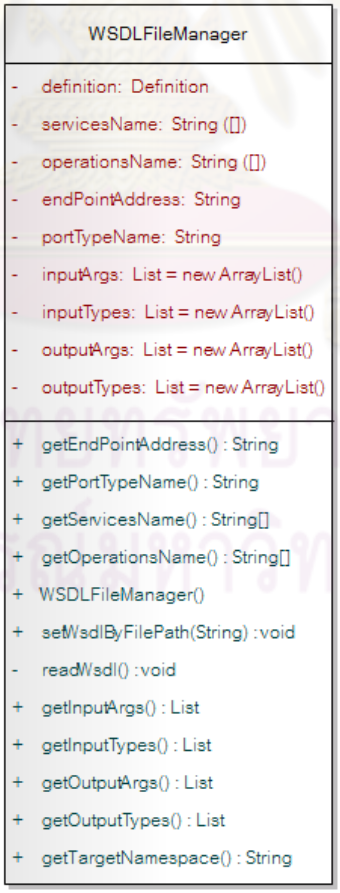


รูปที่ 3.14 คลาส TestCaseDetail

9) คลาส TestRunner คือ คลาสที่ทำหน้าที่ดำเนินการทดสอบโปรแกรมต้นฉบับ และดำเนินการทดสอบโปรแกรมมิวแตนท์ รายละเอียดของคลาสแสดงดังรูปที่ 3.15



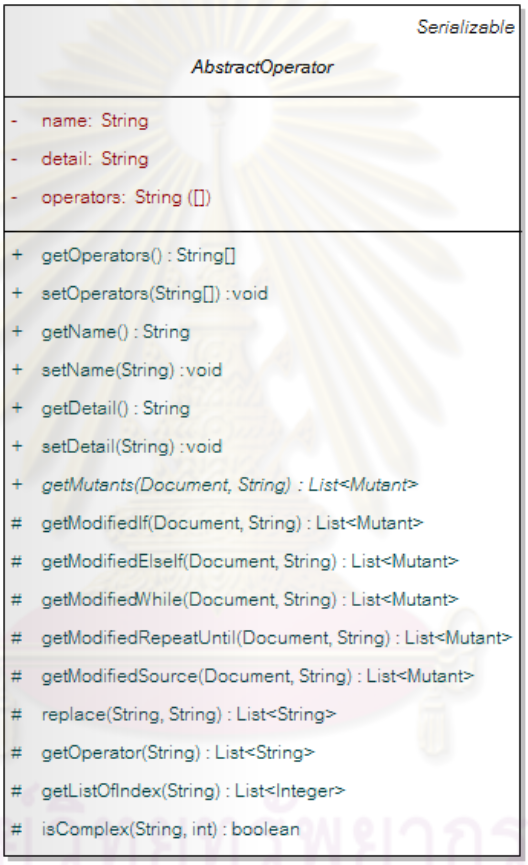
รูปที่ 3.15 คลาส TestRunner



รูปที่ 3.16 คลาส WSDLFileManager

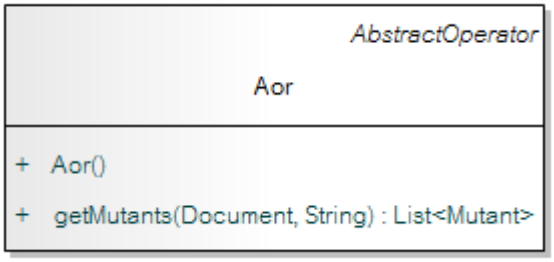
10) คลาส WSDLFileManager คือ คลาสที่ทำหน้าที่เก็บข้อมูลไฟล์ดับเบิลยูเอสดีแอล และวิเคราะห์ส่วนประกอบต่างๆของไฟล์ดับเบิลยูเอสดีแอล รายละเอียดของคลาสแสดงดังรูปที่ 3.16

11) คลาส AbstractOperator คือ คลาสที่นิยามข้อมูลพื้นฐาน และเมท็อด (Method) พื้นฐานของตัวดำเนินการมิวเทชัน รายละเอียดของคลาสแสดงดังรูปที่ 3.17



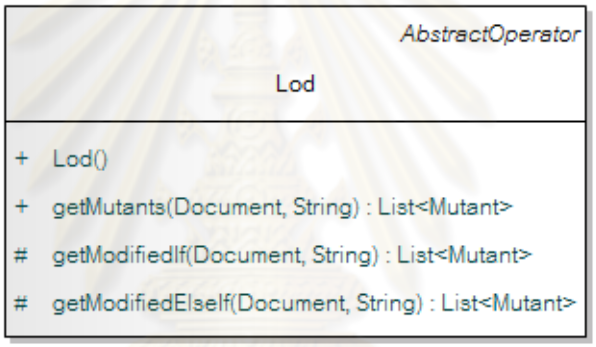
รูปที่ 3.17 คลาส AbstractOperator

12) คลาส AOR คือ คลาสที่ใช้ในการเก็บข้อมูลของตัวดำเนินการมิวเทชันที่ทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงคณิตศาสตร์ด้วยตัวดำเนินการเชิงคณิตศาสตร์ตัวอื่นๆ รายละเอียดของคลาสแสดงดังรูปที่ 3.18



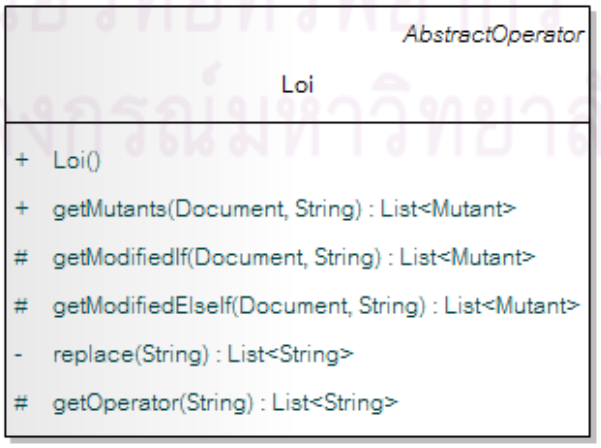
รูปที่ 3.18 คลาส AOR

13) คลาส LOD คือ คลาสที่ใช้ในการเก็บข้อมูลของตัวดำเนินการมิวเทชันที่ทำให้เกิดความผิดพร่องโดยการลบตัวดำเนินการเชิงตรรกะ รายละเอียดของคลาสแสดงดังรูปที่ 3.19



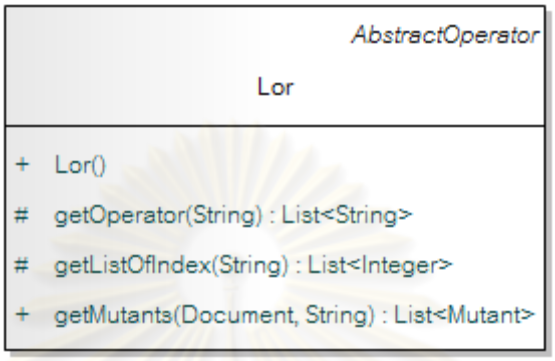
รูปที่ 3.19 คลาส LOD

14) คลาส LOI คือ คลาสที่เก็บข้อมูลของตัวดำเนินการมิวเทชันที่ทำให้เกิดความผิดพร่องโดยการแทรกตัวดำเนินการเชิงตรรกะ รายละเอียดของคลาสแสดงดังรูปที่ 3.20



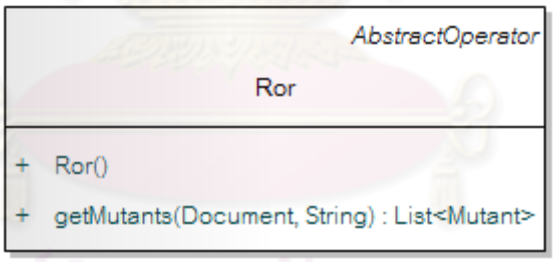
รูปที่ 3.20 คลาส LOI

15) คลาส LOR คือ คลาสที่เก็บข้อมูลของตัวดำเนินการมิวเทชันที่ทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงตรรกะด้วยตัวดำเนินการเชิงตรรกะตัวอื่นๆ รายละเอียดของคลาสแสดงดังรูปที่ 3.21



รูปที่ 3.21 คลาส LOR

16) คลาส ROR คือ คลาสที่เก็บข้อมูลตัวดำเนินการมิวเทชันที่ทำให้เกิดความผิดพร่องโดยการแทนที่ตัวดำเนินการเชิงสัมพันธ์ด้วยตัวดำเนินการเชิงสัมพันธ์ตัวอื่นๆ รายละเอียดของคลาสแสดงดังรูปที่ 3.22

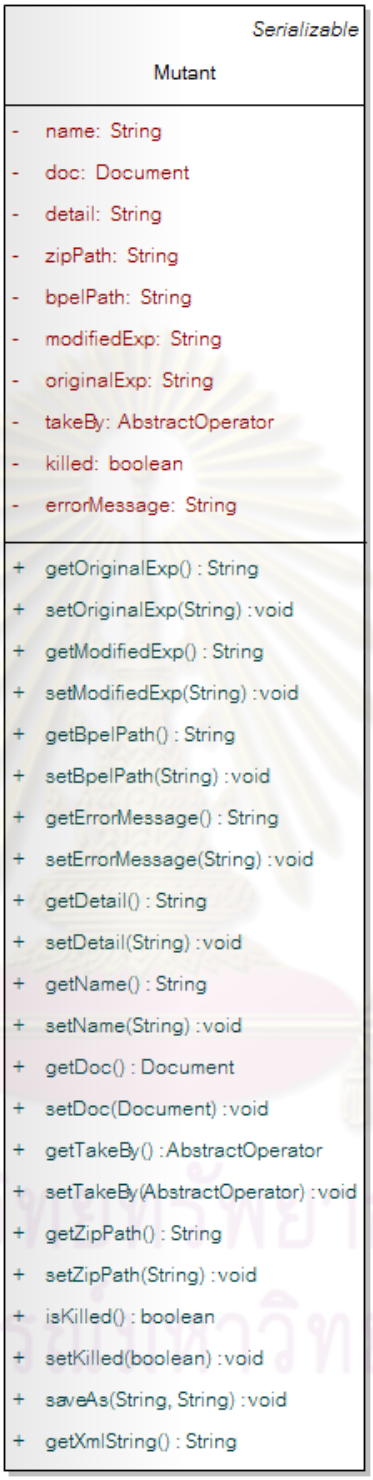


รูปที่ 3.22 คลาส ROR

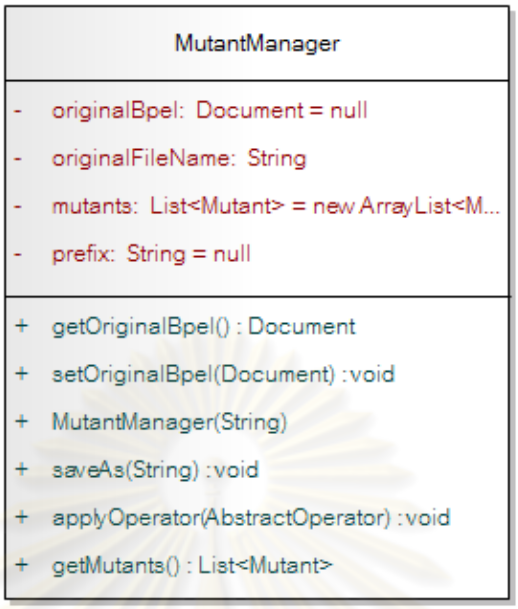
17) คลาส Mutant คือ คลาสที่เก็บข้อมูลและบันบันทึกมิวแตนท์ รายละเอียดของคลาสแสดงดังรูปที่ 3.23

18) คลาส MutantManager คือ คลาสที่ใช้สำหรับจัดการมิวแตนท์ รายละเอียดของคลาสแสดงดังรูปที่ 3.24

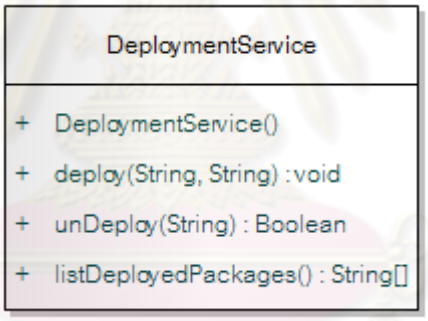
19) คลาส DeploymentService คือ คลาสที่ทำหน้าที่จัดการการติดตั้งโปรแกรมลงเครื่องประมวลผลบีเพล รายละเอียดของคลาสแสดงดังรูปที่ 3.25



รูปที่ 3.23 คลาส Mutant



รูปที่ 3.24 คลาส MutantManager



รูปที่ 3.25 คลาส DeploymentService

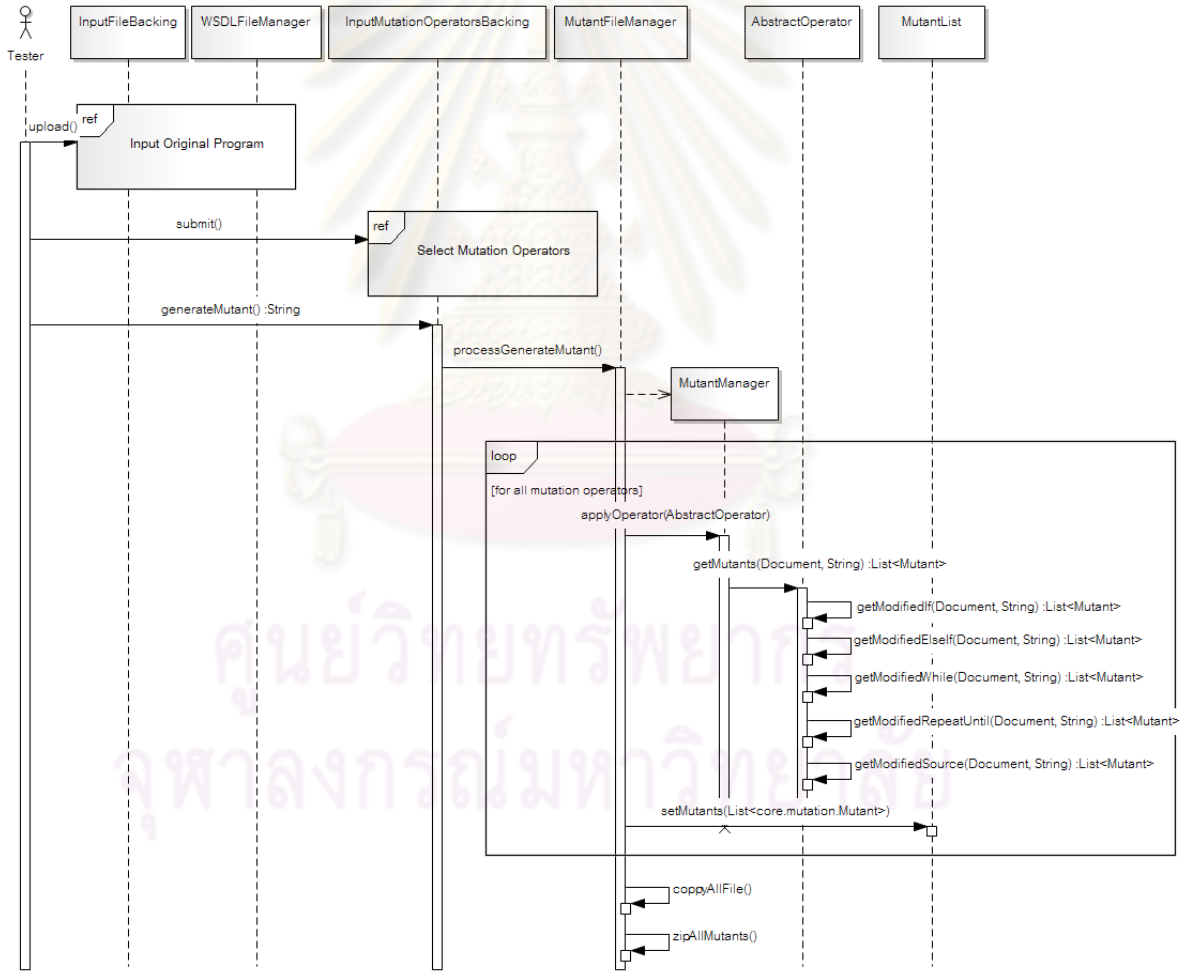
3.4.3 แผนภาพลำดับและแผนภาพลำดับกิจกรรม

แผนภาพลำดับเป็นแผนภาพที่แสดงปฏิสัมพันธ์ระหว่างวัตถุต่างๆที่อยู่ในระบบ โดยแผนภาพลำดับของเครื่องมือมีดังต่อไปนี้

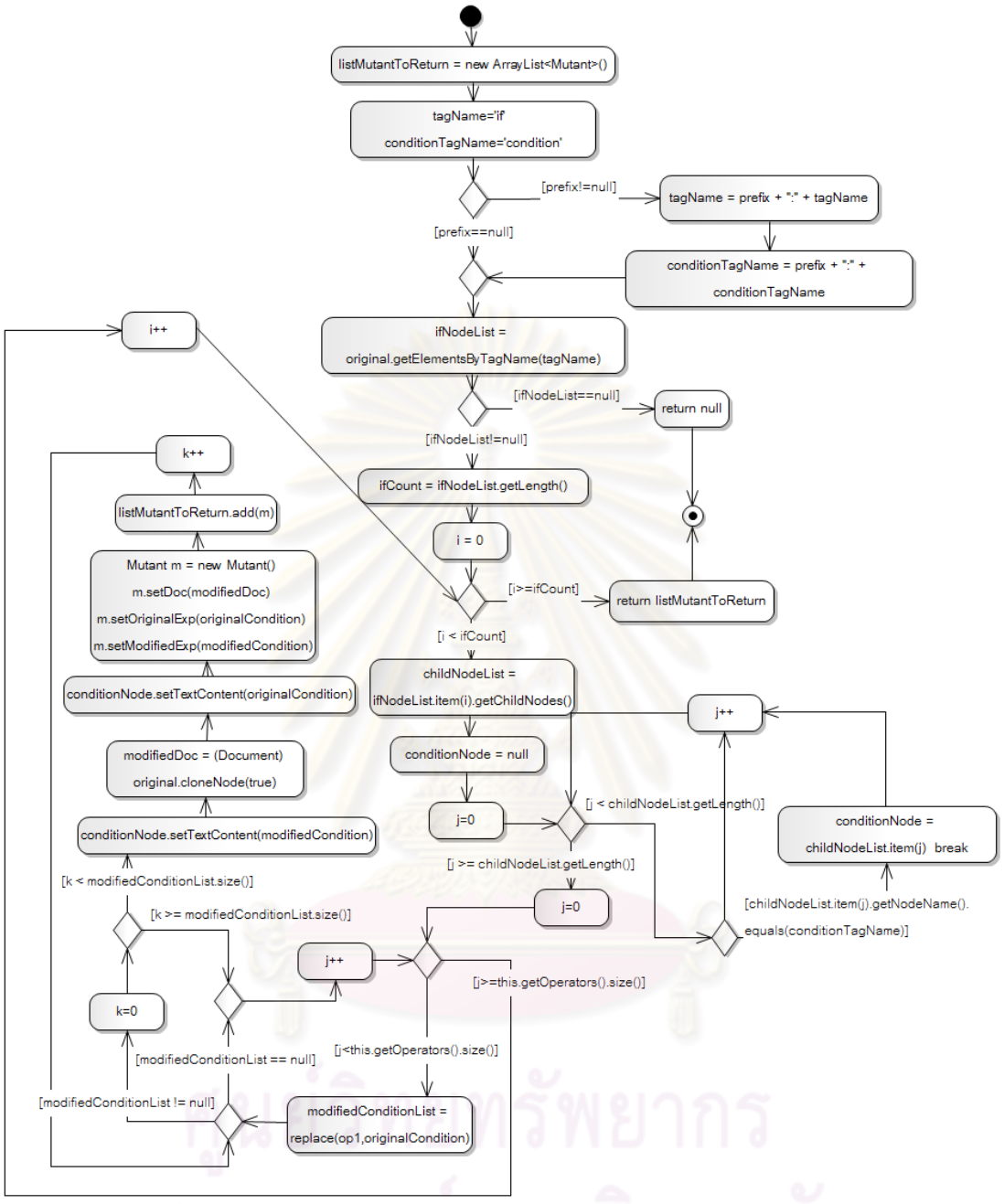
1) แผนภาพลำดับ Generate Mutants ดังรูปที่ 3.26 เป็นการสร้างมิวแตนท์ โดยเริ่มจากการนำเข้าโปรแกรมต้นฉบับ หลังจากนั้นนำเข้าชนิดของตัวดำเนินการมิวเทชันจากผู้ใช้ และทำการสร้างมิวแตนท์โดยการสร้างอ็อบเจกต์ MutantManager หลังจากนั้น

เรียกใช้เมท็อด applyOps เพื่อใช้ตัวดำเนินการมิวเทชันในการสร้างมิวแตนท์ ซึ่งใช้แผนภาพกิจกรรมดังต่อไปนี้ในการอธิบายการดัดแปลงนิพจน์ของบีเพล

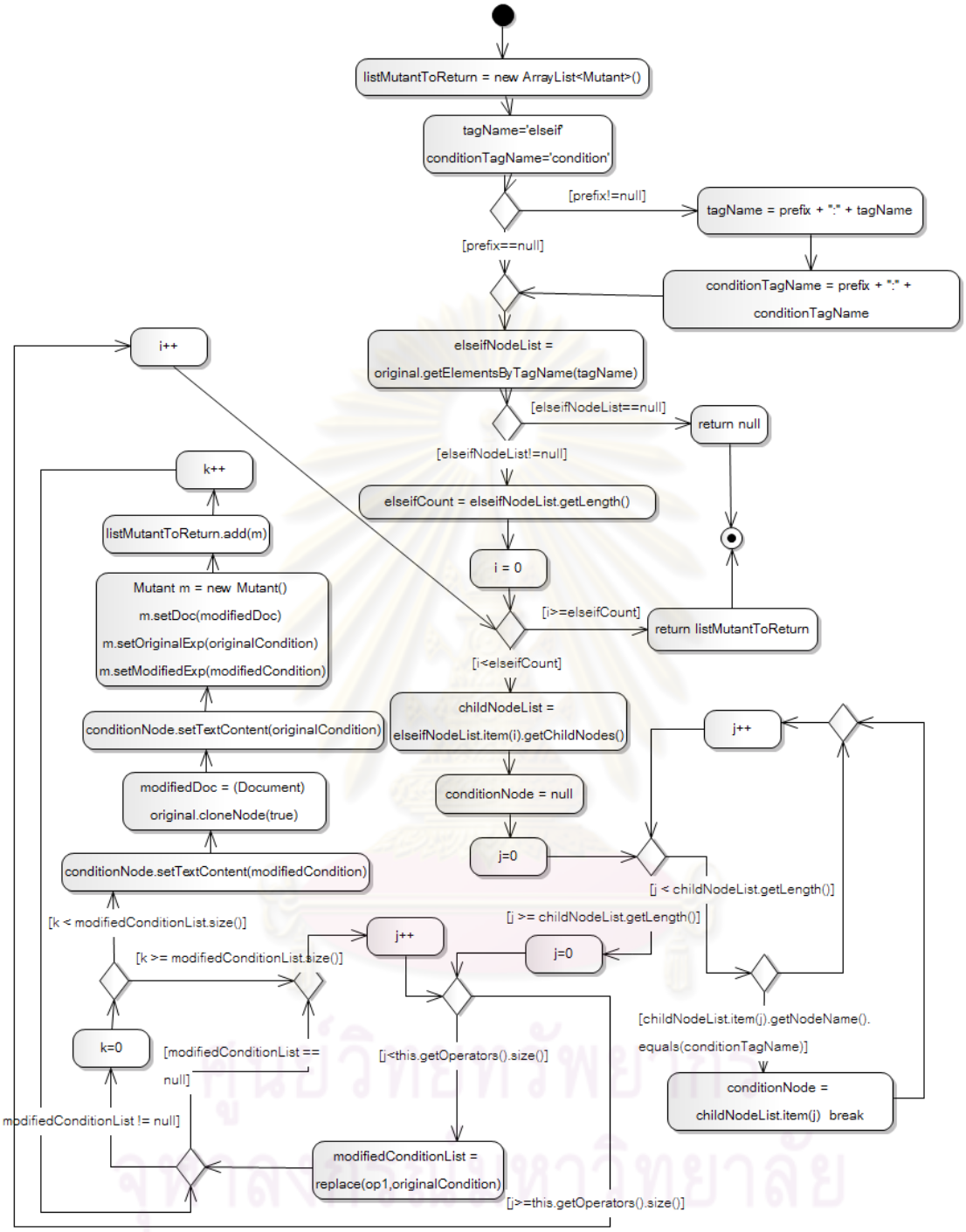
1.1) ในส่วนของแผนภาพกิจกรรมในรูปที่ 3.27 – 3.31 เป็นตัวอย่างแผนภาพที่อธิบายเมท็อด getModifiedIf (Document original,String prefix) getModifiedElseIf(Document original,String prefix) getModifiedWhile(Document original ,String prefix) getModifiedRepeatUtil(Document original,String prefix) และ getModifiedSource(Document original,String prefix) สำหรับตัวดำเนินการมิวเทชัน AOR LOR และ ROR



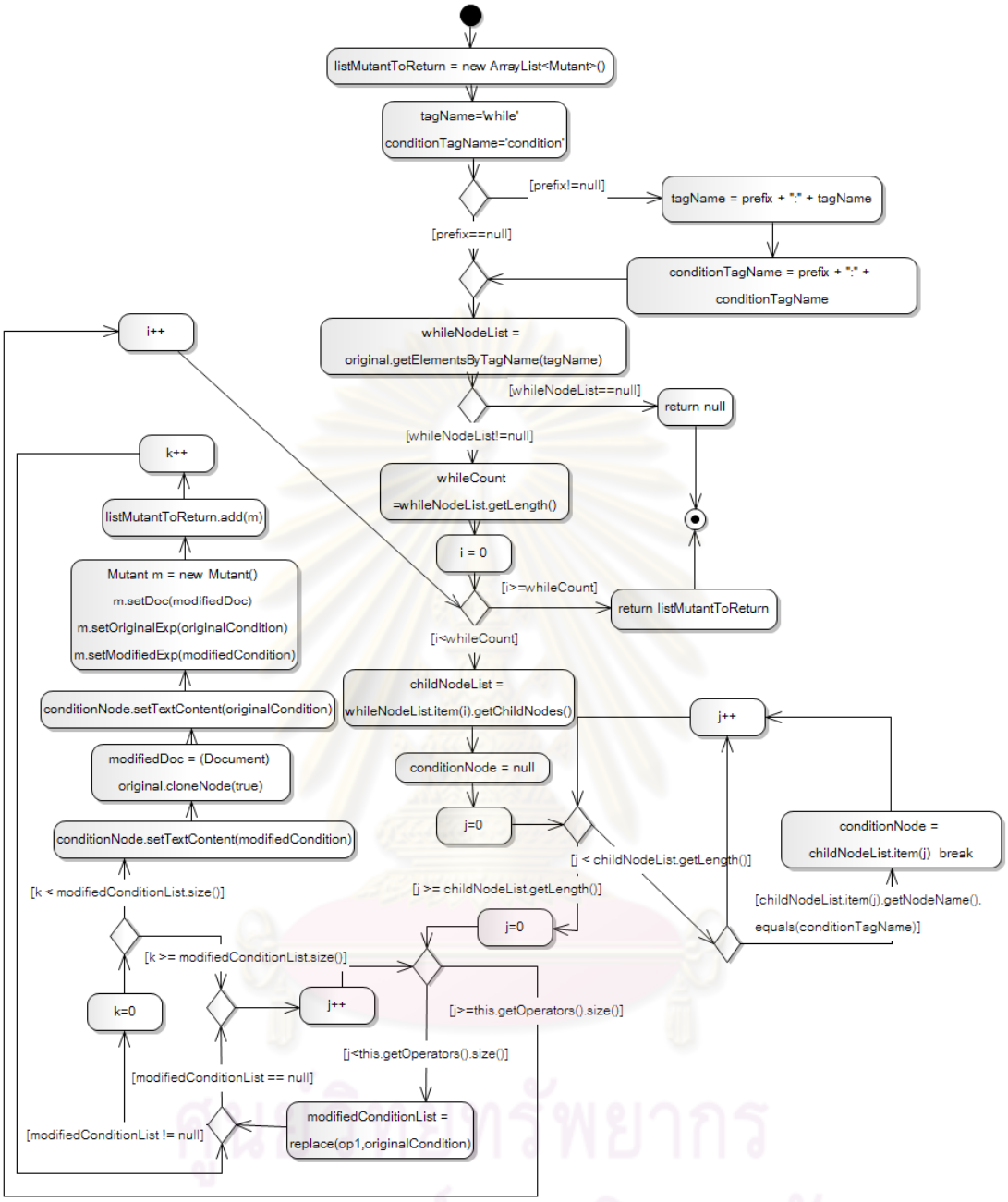
รูปที่ 3.26 แผนภาพลำดับ Generate Mutants



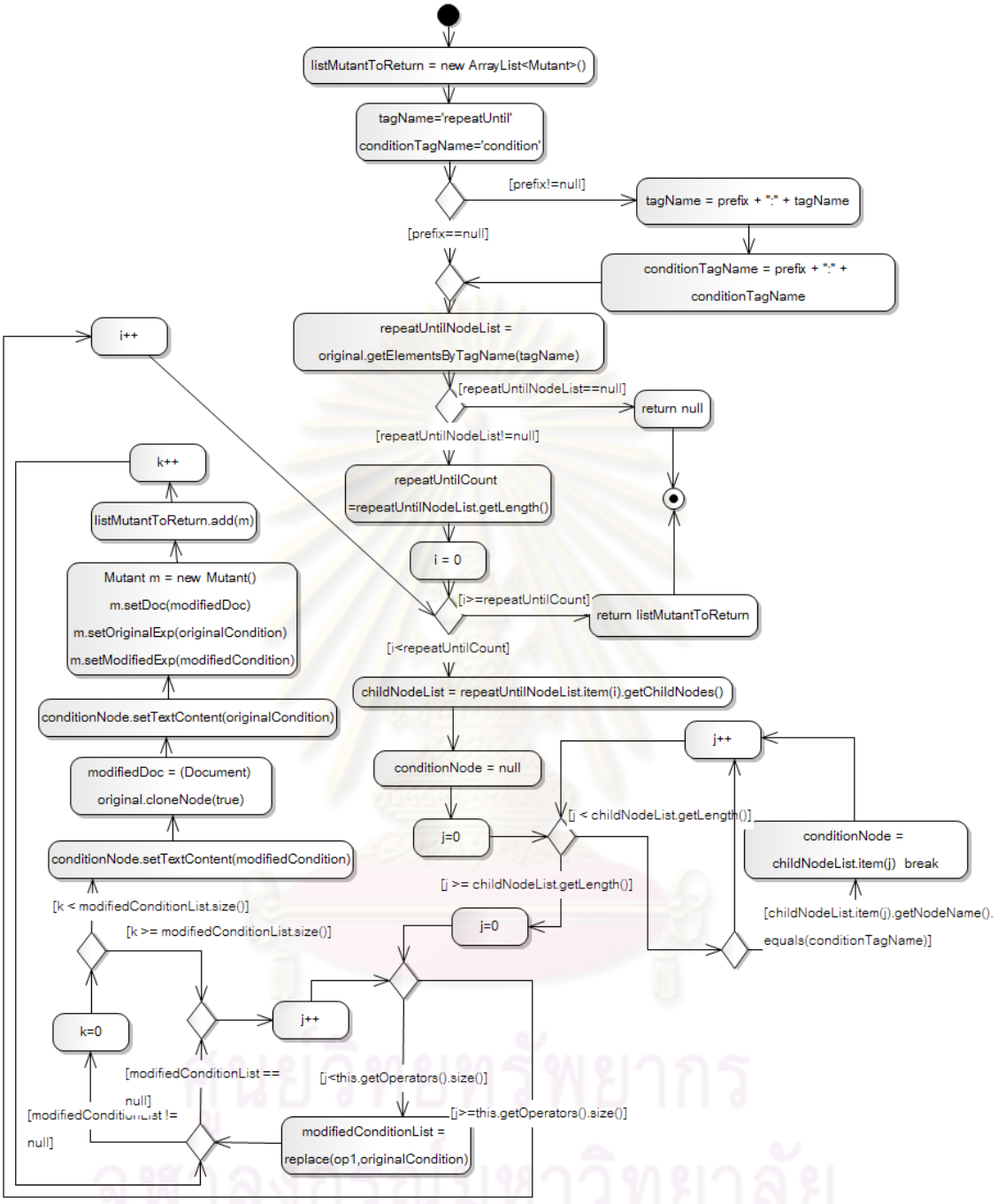
รูปที่ 3.27 แผนภาพกิจกรรมของเมท็อด getModifiedIf(Document,String) (AOR LOR และ ROR)



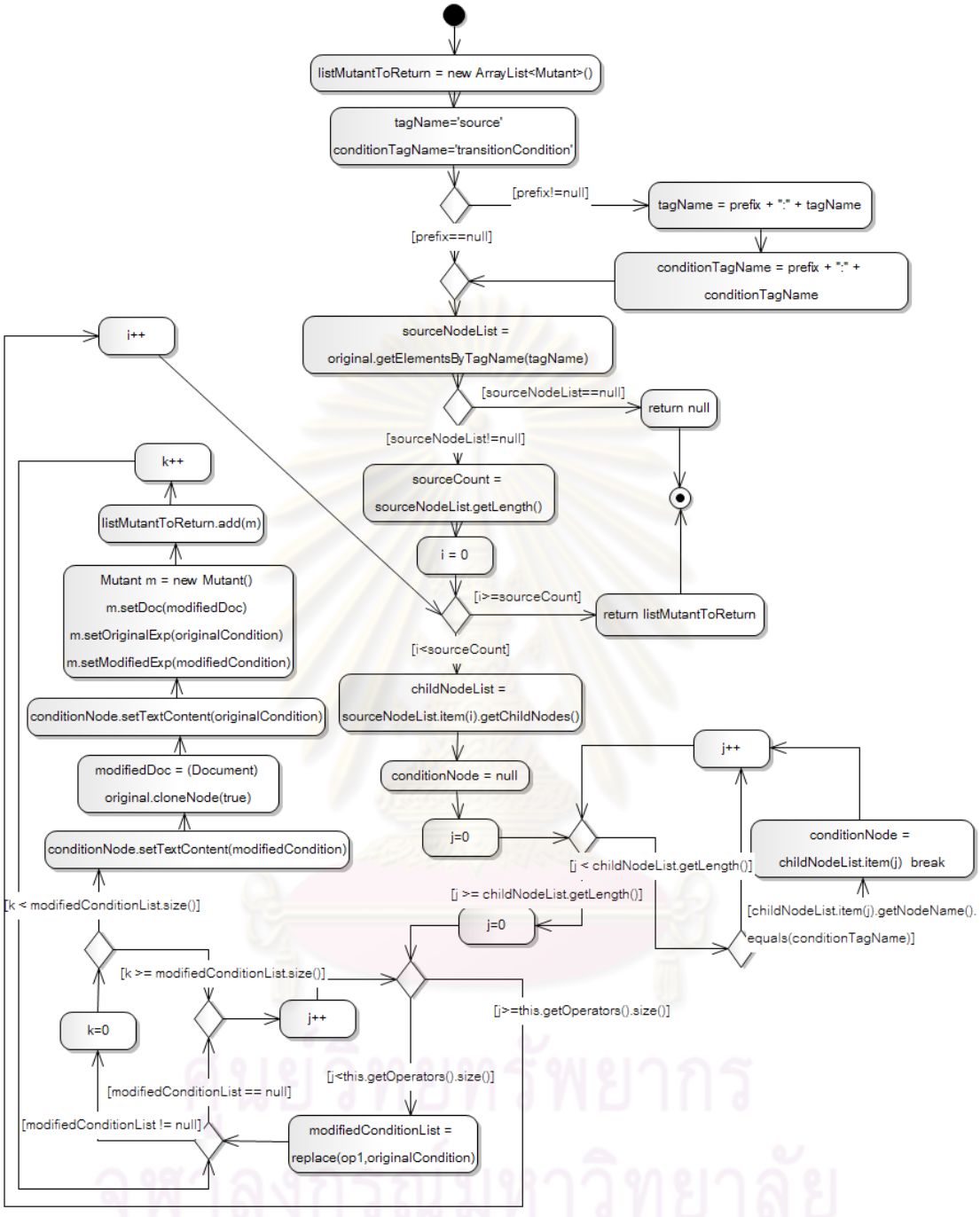
รูปที่ 3.28 แผนภาพกิจกรรมของเมท็อด getModifiedElseIf(Document,String) (AOR LOR และ ROR)



รูปที่ 3.29 แผนภาพกิจกรรมของเมท็อด getModifiedWhile(Document,String) (AOR LOR และ ROR)



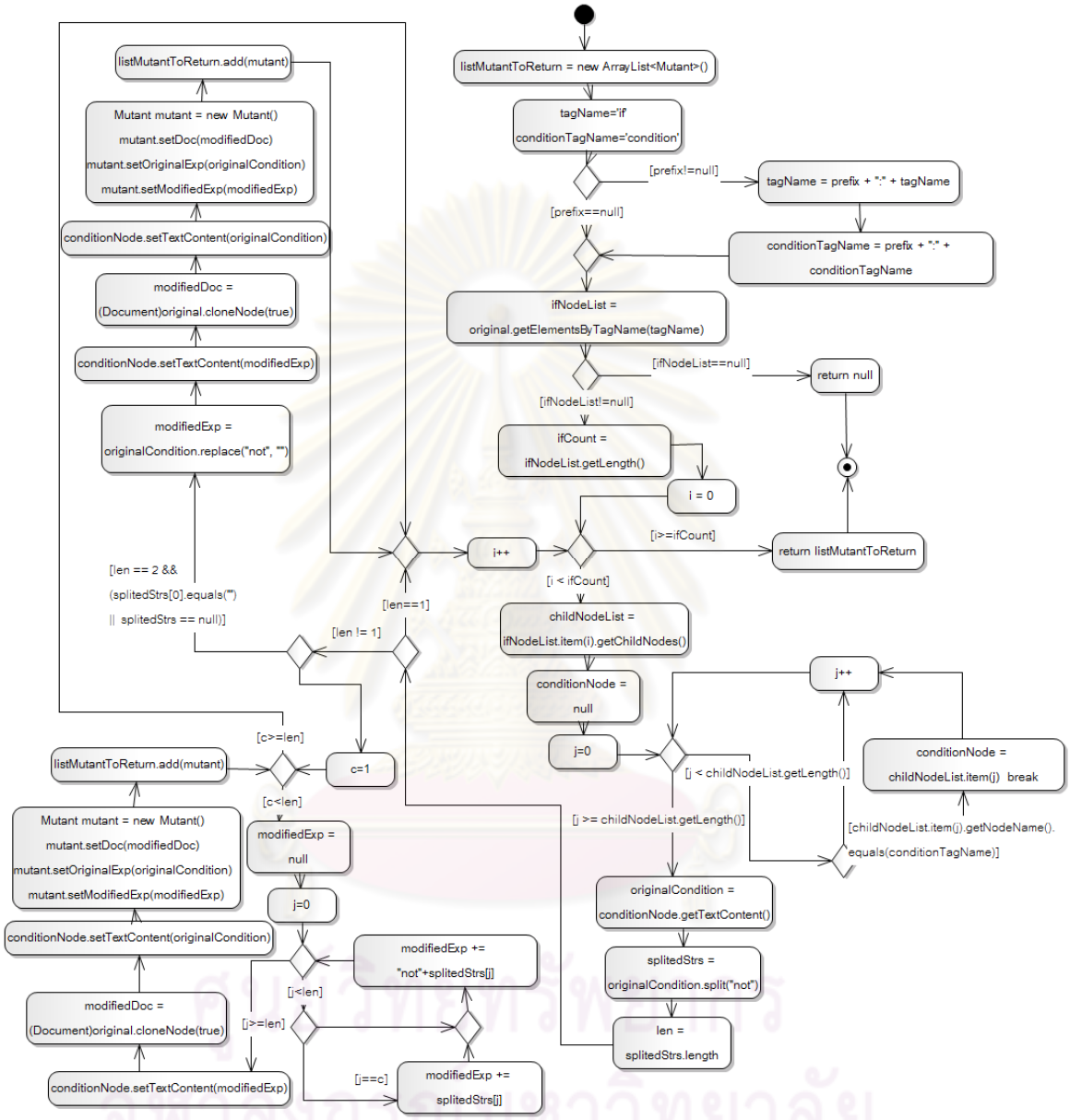
รูปที่ 3.30 แผนภาพกิจกรรมของเมท็อด getModifiedRepeatUtil(Document,String) (AOR LOR และ ROR)



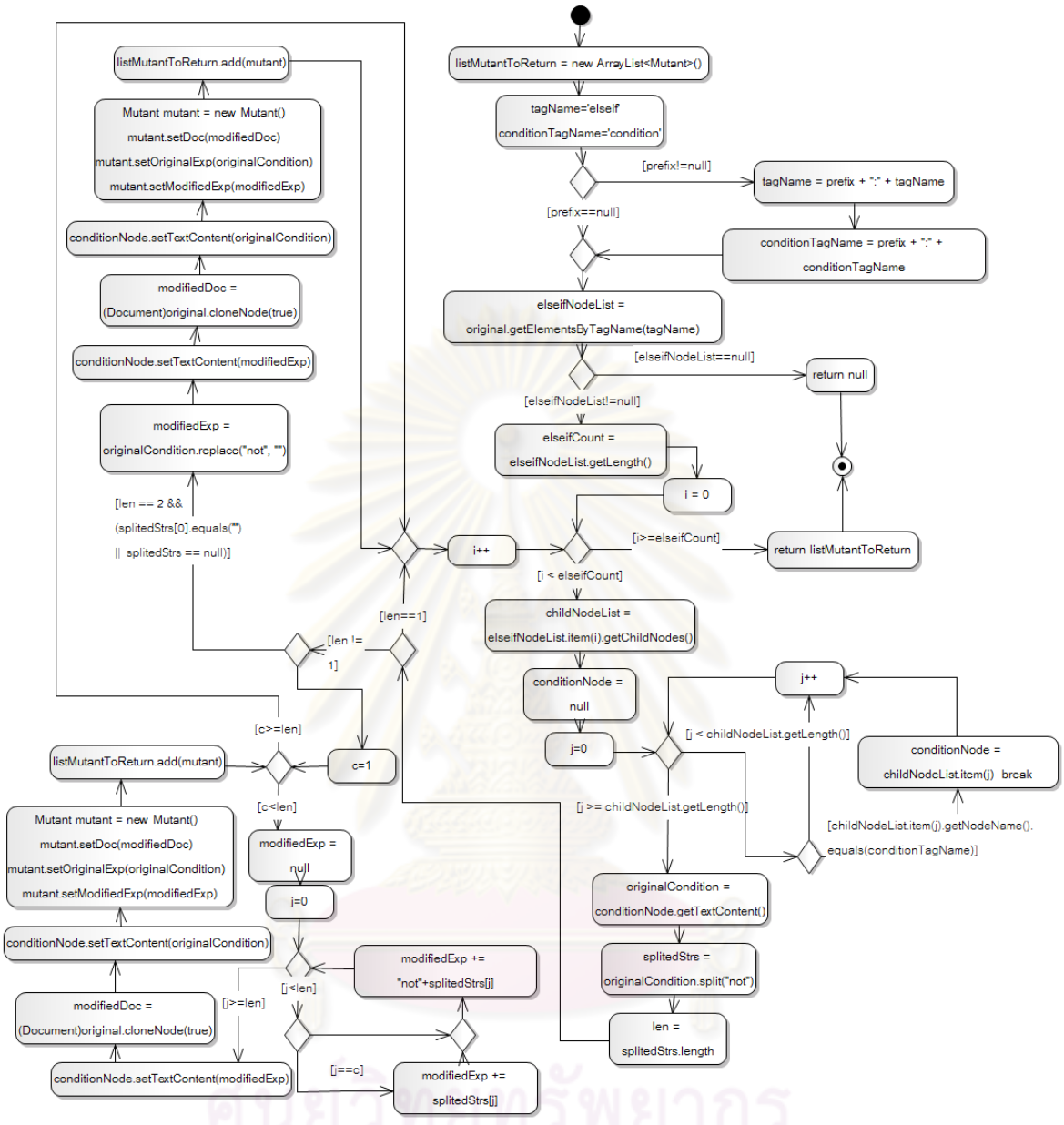
รูปที่ 3.31 แผนภาพกิจกรรมของเมท็อด getModifiedSource(Document,String) (AOR LOR และ ROR)

1.2) ในส่วนของแผนภาพกิจกรรมในรูปที่ 3.32 – 3.36 เป็นตัวอย่างแผนภาพที่อธิบายเมท็อด getModifiedIf(Document original,String prefix) getModifiedElseIf(Document original,String prefix) getModifiedWhile(Document

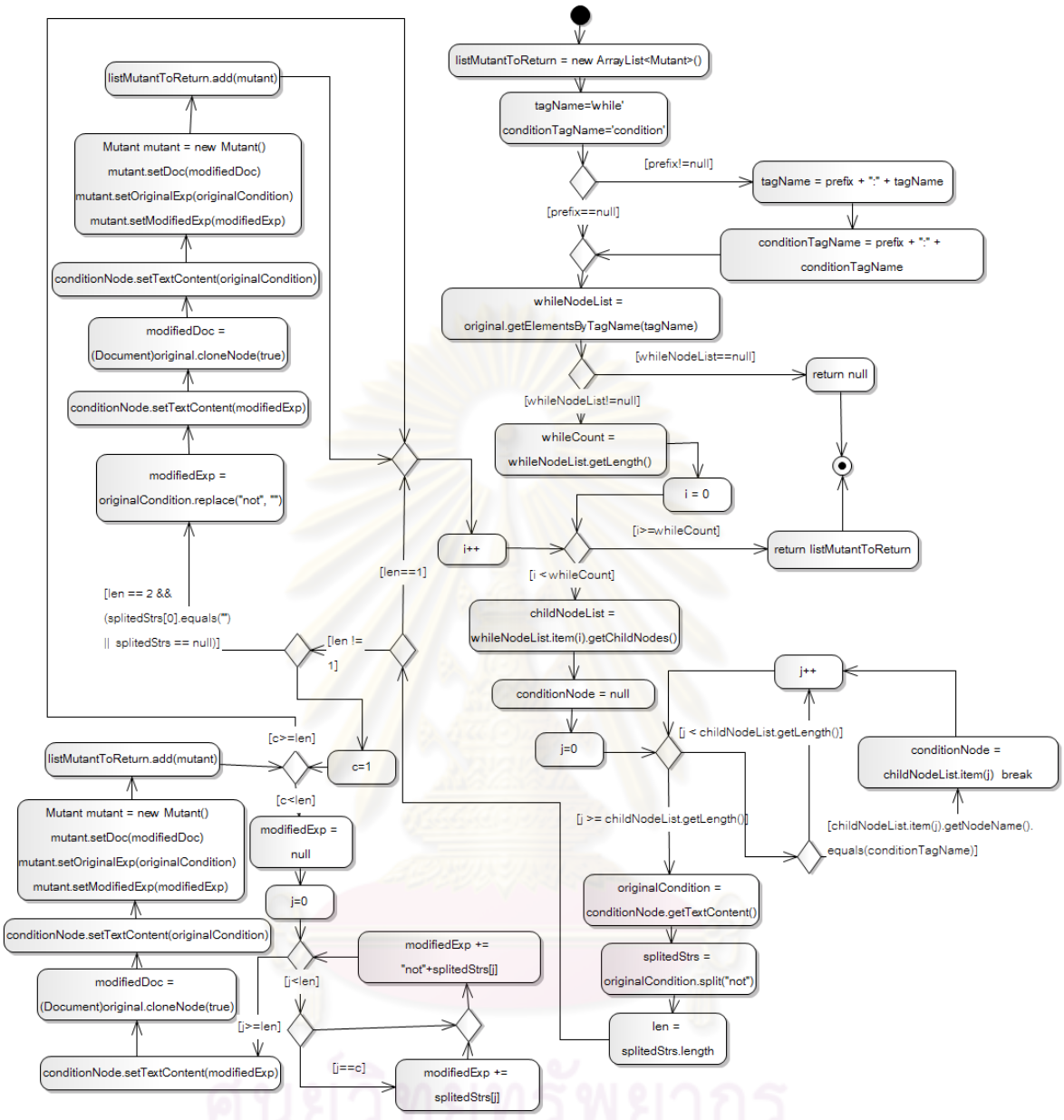
original ,String prefix) getModifiedRepeatUtil(Document original,String prefix) และ getModifiedSource(Document original,String prefix) สำหรับตัวดำเนินการมิวเทชัน LOD



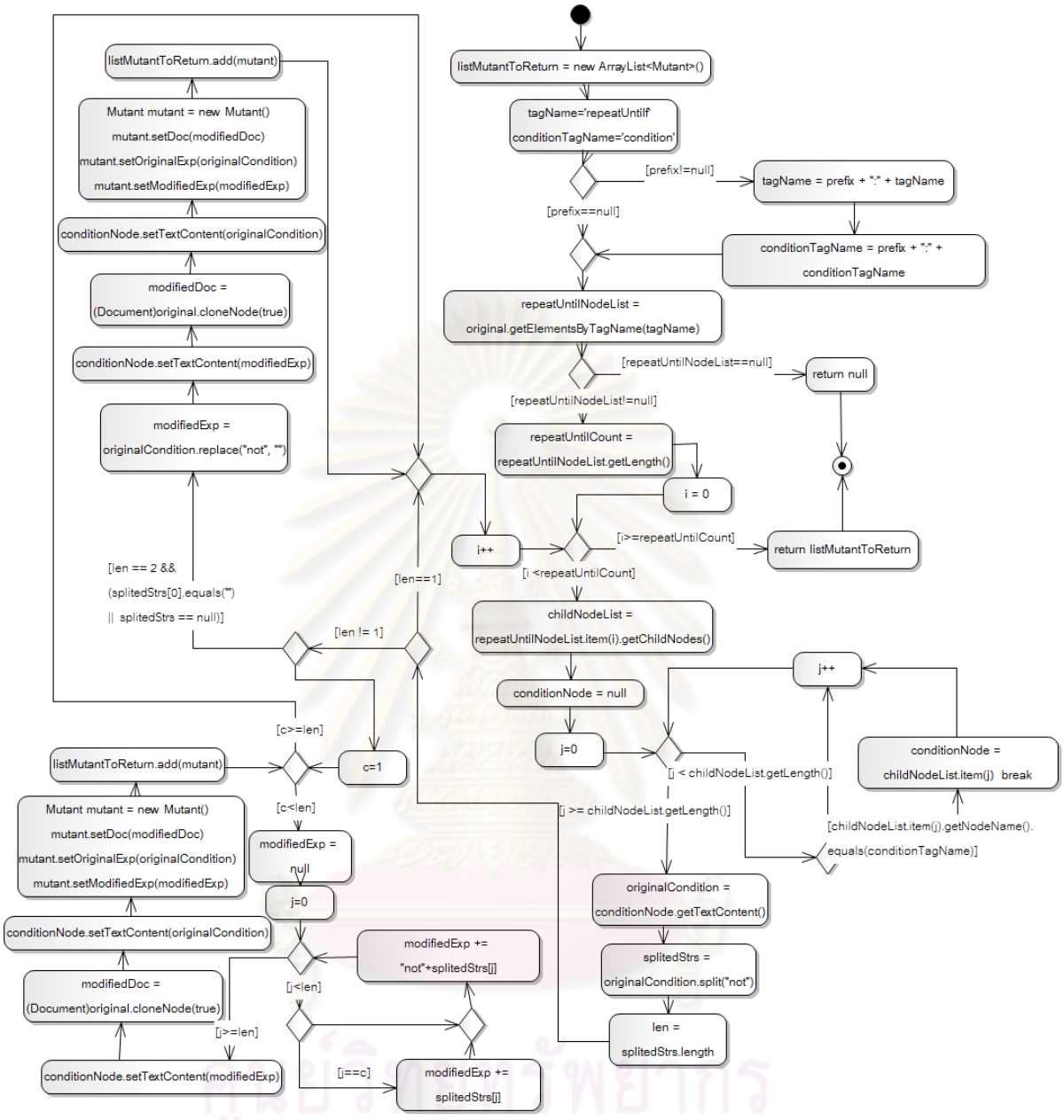
รูปที่ 3.32 แผนภาพกิจกรรมของเมท็อด getModifiedIf(Document,String) (LOD)



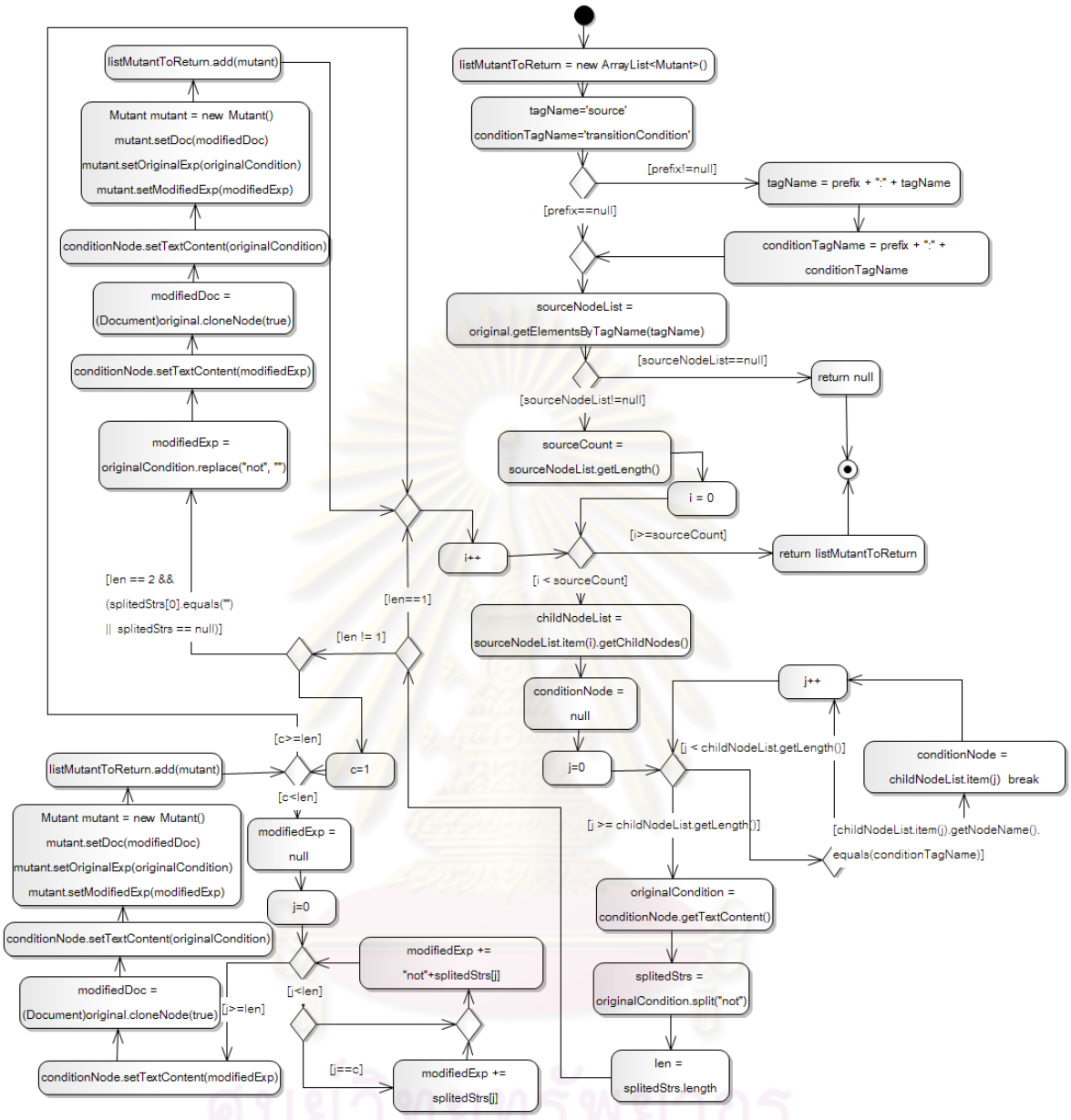
รูปที่ 3.33 แผนภาพกิจกรรมของเมท็อด getModifiedElseIf(Document,String) (LOD)



รูปที่ 3.34 แผนภาพกิจกรรมของเมท็อด getModifiedWhile(Document,String) (LOD)

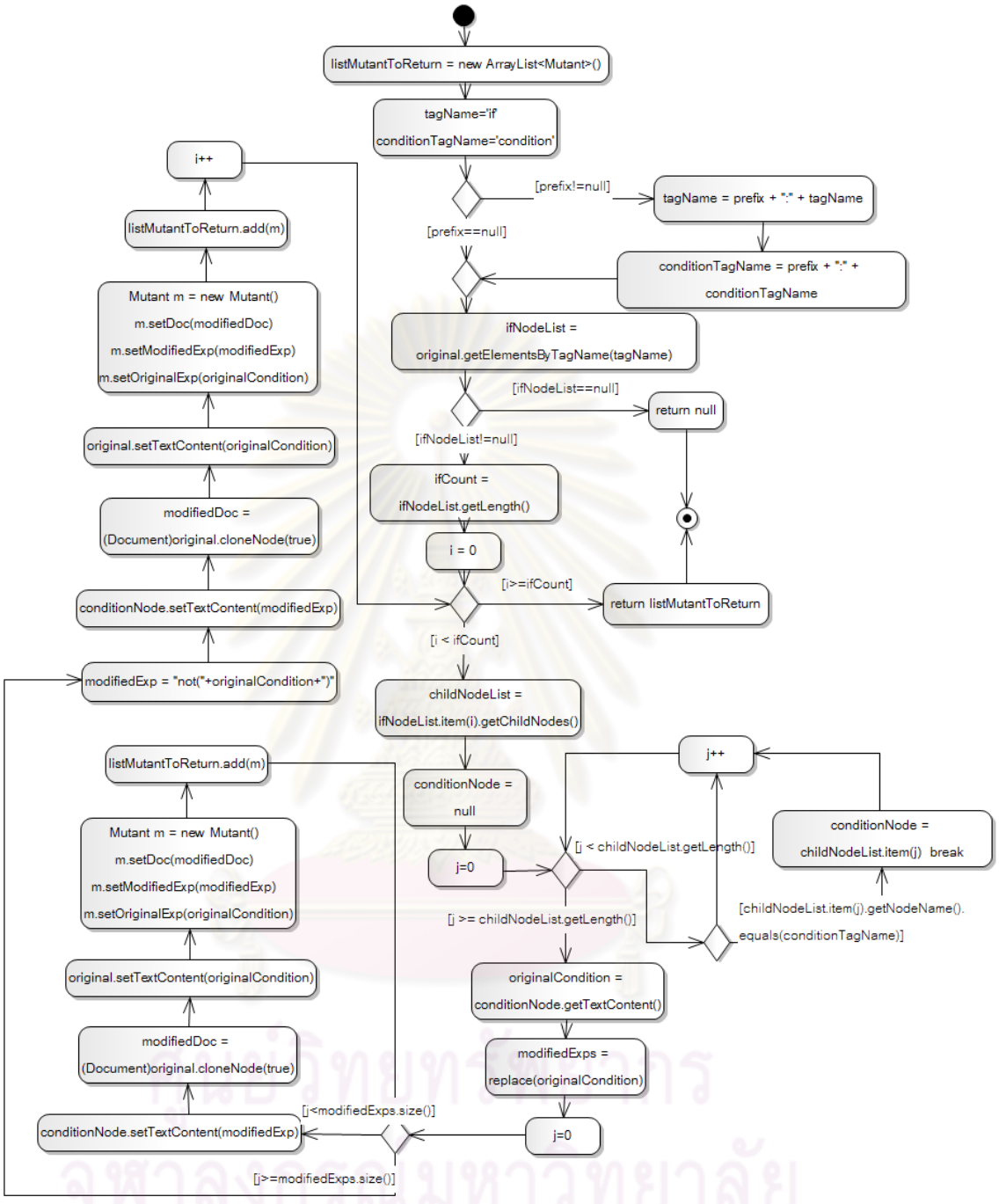


รูปที่ 3.35 แผนภาพกิจกรรมของเมท็อด getModifiedRepeatUtil(Document,String) (LOD)

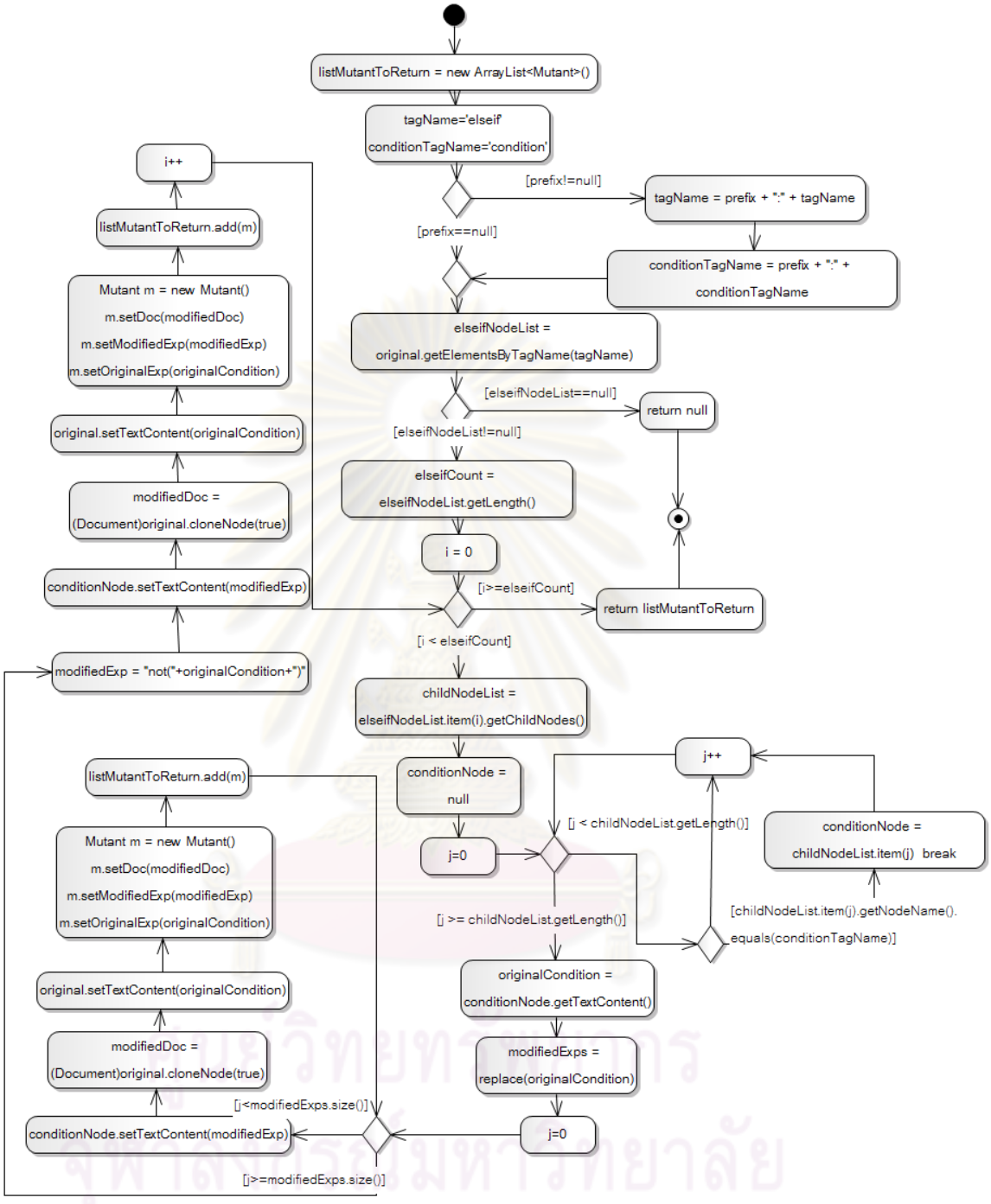


รูปที่ 3.36 แผนภาพกิจกรรมของเมท็อด getModifiedSource(Document,String) (LOD)

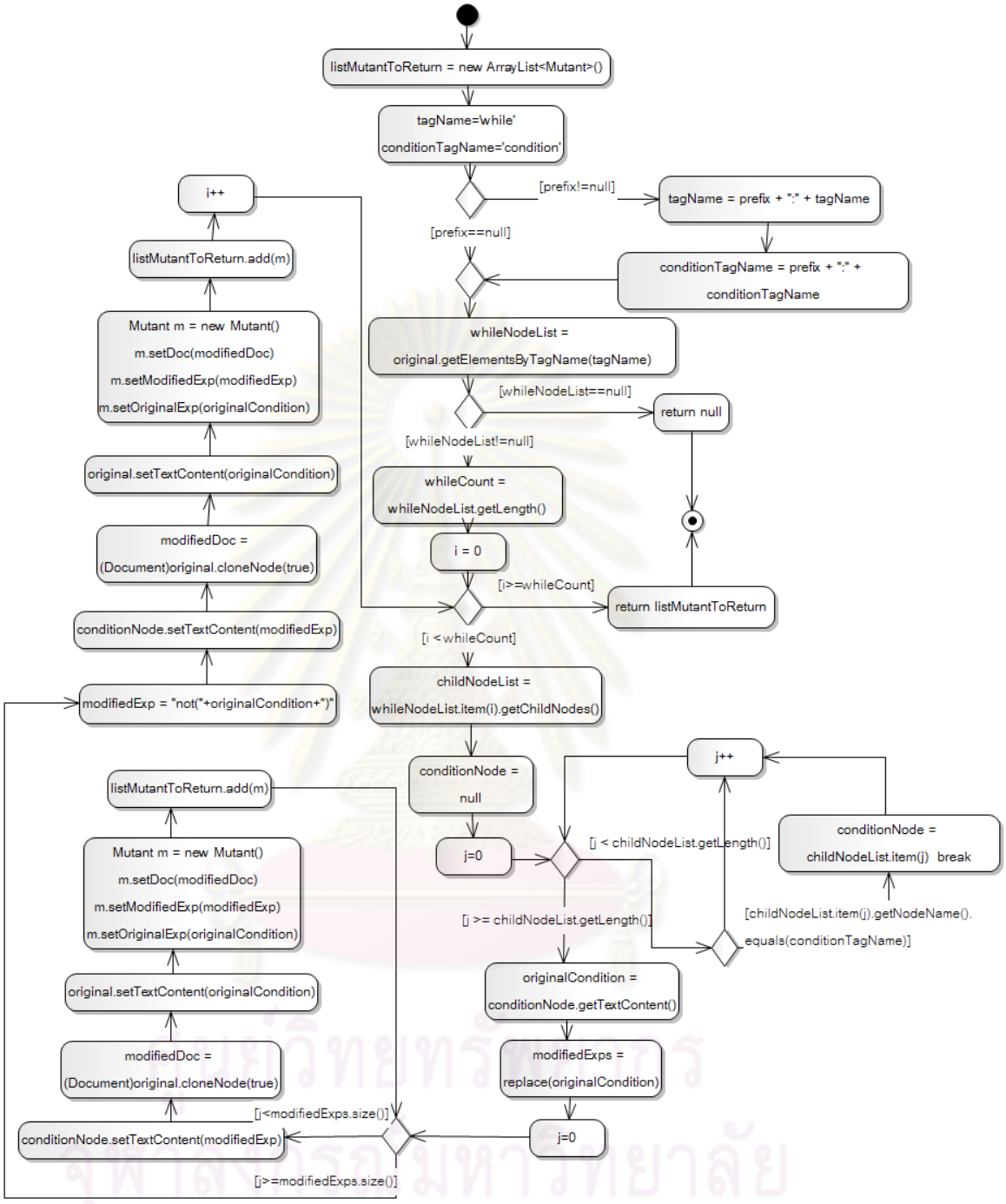
1.3) ในส่วนของแผนภาพกิจกรรมในรูปที่ 3.37 – 3.41 เป็นตัวอย่างแผนภาพที่อธิบายเมท็อด getModifiedIf(Document original,String prefix) getModifiedElseIf(Document original,String prefix) getModifiedWhile(Document original ,String prefix) getModifiedRepeatUtil(Document original,String prefix) และ getModifiedSource(Document original,String prefix) สำหรับตัวดำเนินการมิวเทชัน LOI



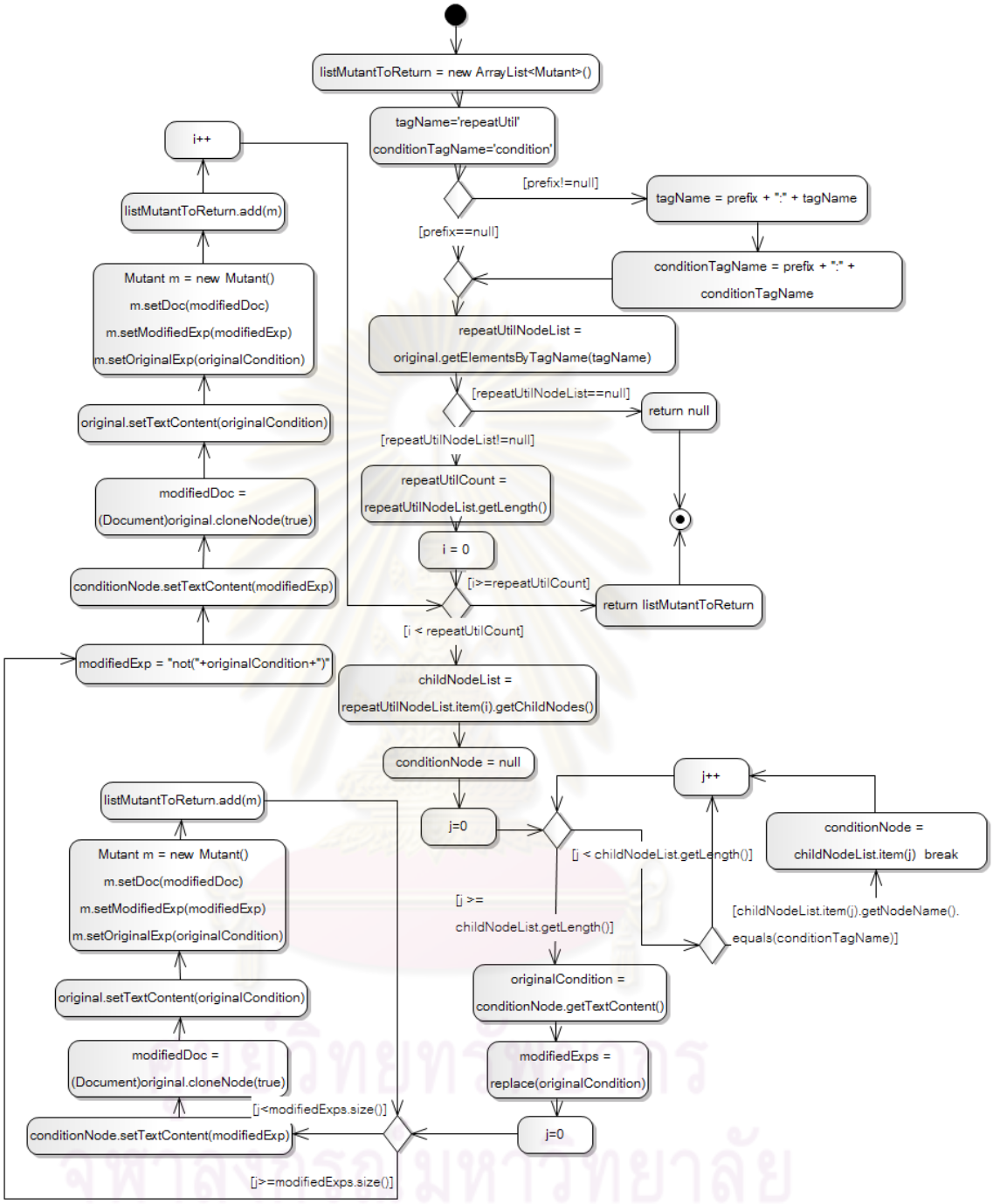
รูปที่ 3.37 แผนภาพกิจกรรมของเมท็อด getModifiedIf(Document,String) (LOI)



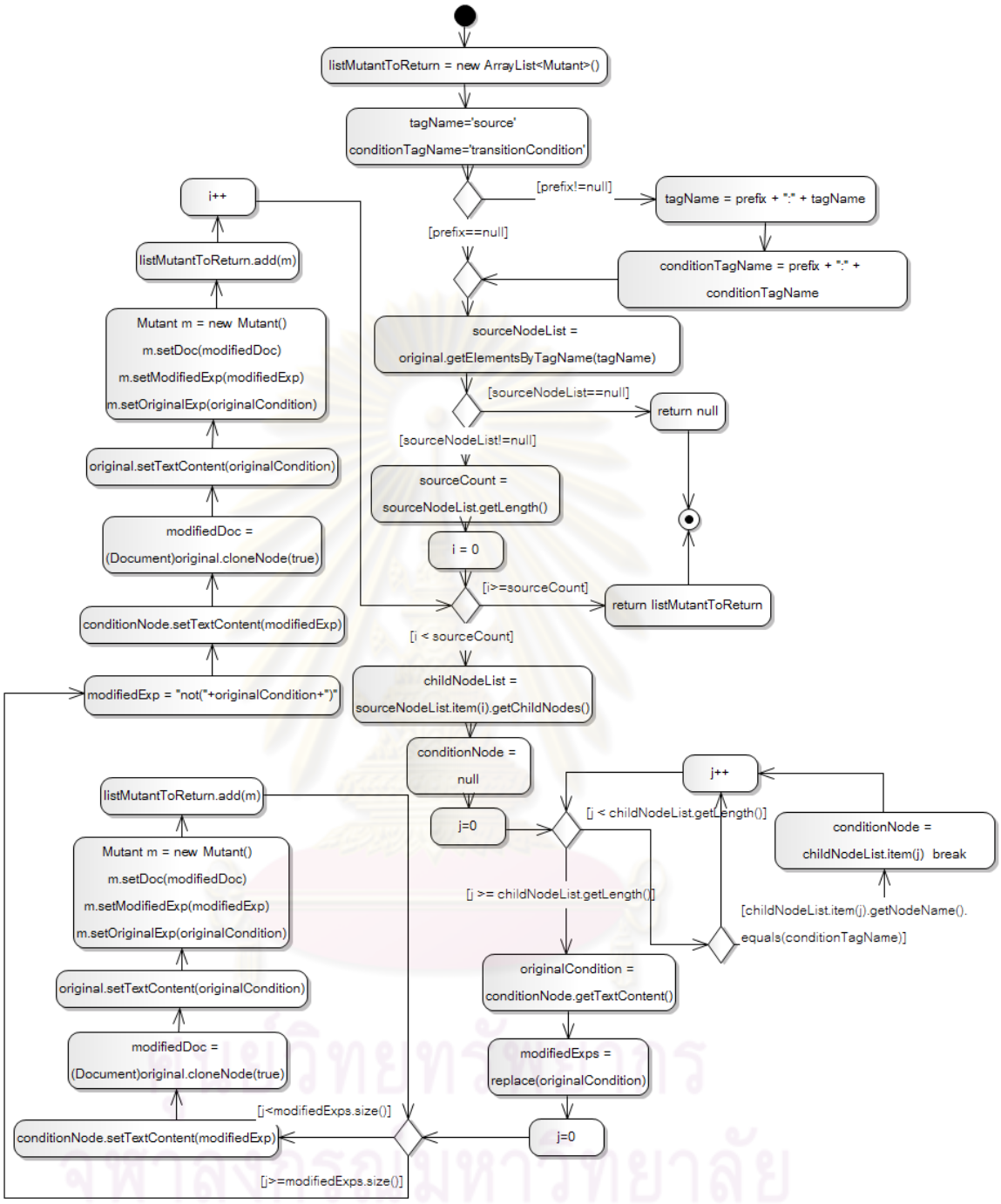
รูปที่ 3.38 แผนภาพกิจกรรมของเมท็อด getModifiedElseIf(Document,String) (LOI)



รูปที่ 3.39 แผนภาพกิจกรรมของเมท็อด getModifiedWhile(Document,String) (LOI)

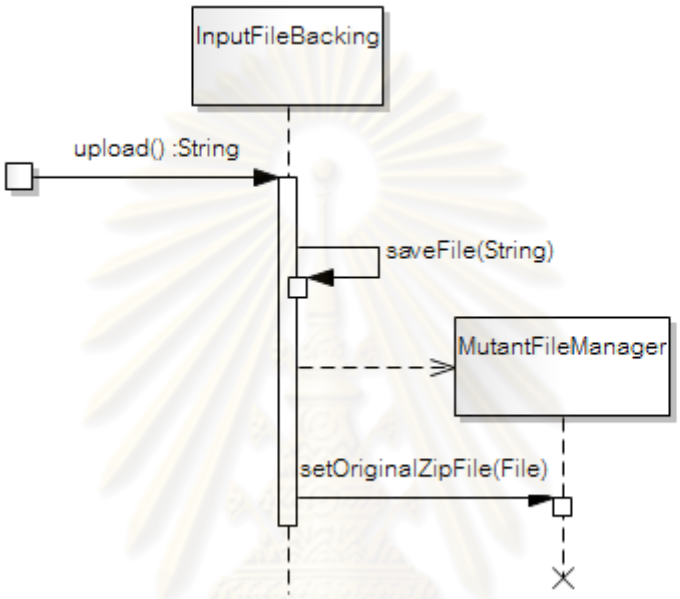


รูปที่ 3.40 แผนภาพกิจกรรมของเมท็อด getModifiedRepeatUtil(Document,String) (LOI)



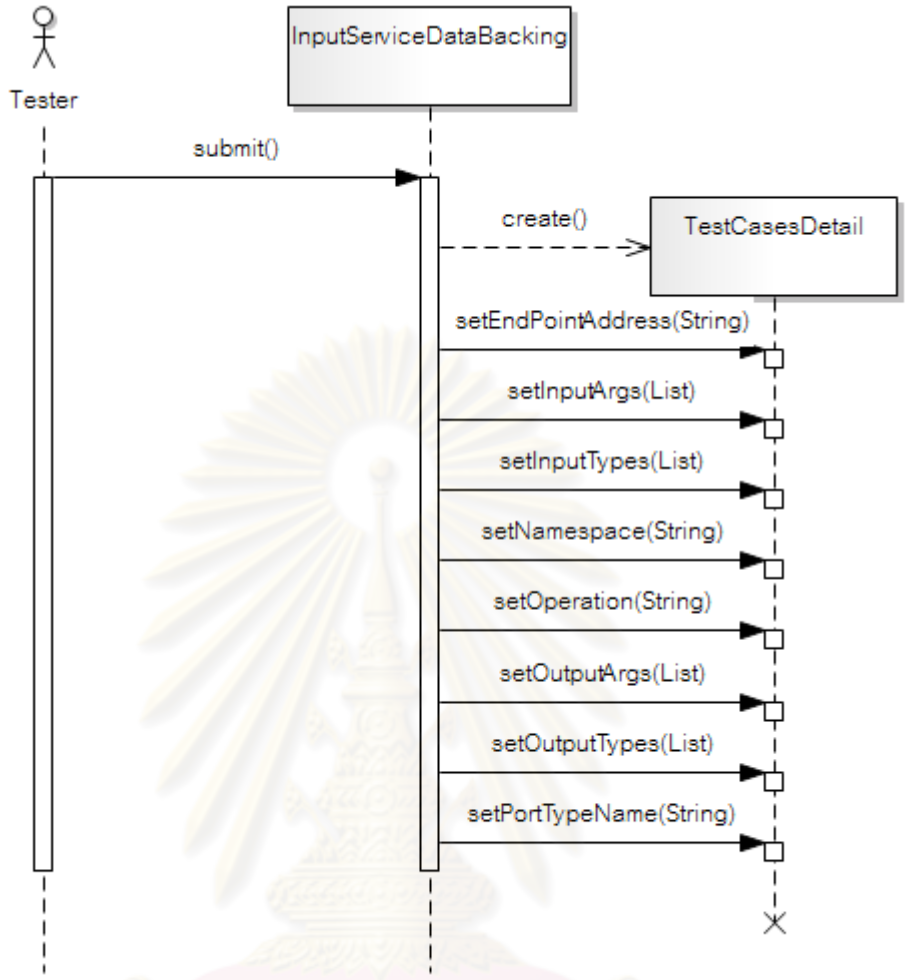
รูปที่ 3.41 แผนภาพกิจกรรมของเมท็อด getModifiedSource(Document,String) (LOI)

2) แผนภาพลำดับ Input Original Program ดังรูปที่ 3.42 เป็นการนำเข้าโปรแกรมต้นฉบับ โดยเริ่มจากระบบสร้างหน้าต่างสำหรับอัพโหลดโปรแกรมต้นฉบับ หลังจากนั้นผู้ใช้อัพโหลดโปรแกรมต้นฉบับหลังจากการอัพโหลดเรียบร้อยโปรแกรมจะได้รับการบันทึกบนเครื่องบริการเว็บ



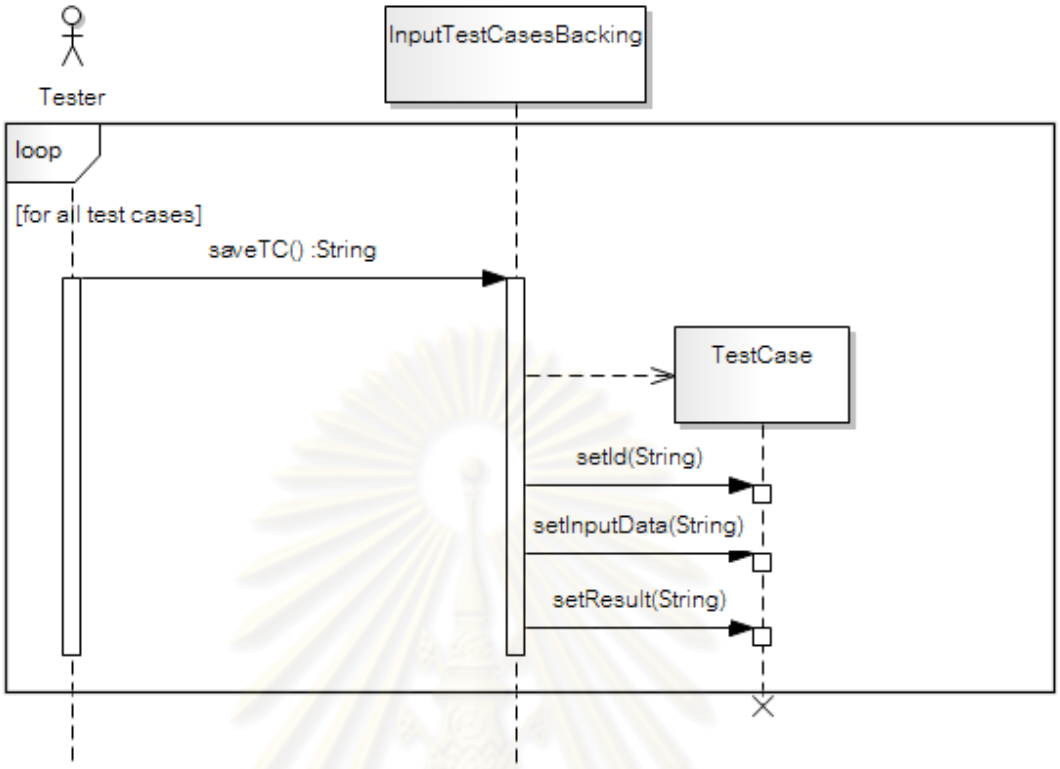
รูปที่ 3.42 แผนภาพลำดับ Input Original Program

3) แผนภาพลำดับ Input Service Data ดังรูปที่ 3.43 หลังจากที่ผู้ทดสอบนำเข้าโปรแกรมต้นฉบับ เลือกตัวดำเนินการมิวเทชัน และระบบทำการสร้างมิวแตนท์แล้ว ระบบจะทำการวิเคราะห์เอกสารดับเบิลยูเอสดีแอลเพื่อทำการหาจำนวนและชนิดของข้อมูลนำเข้า และทำการกำหนดค่าข้อมูลต่างๆที่จำเป็นต้องใช้ในการดำเนินการทดสอบเข้าไปไว้ในอ็อบเจกต์ TestCaseDetail



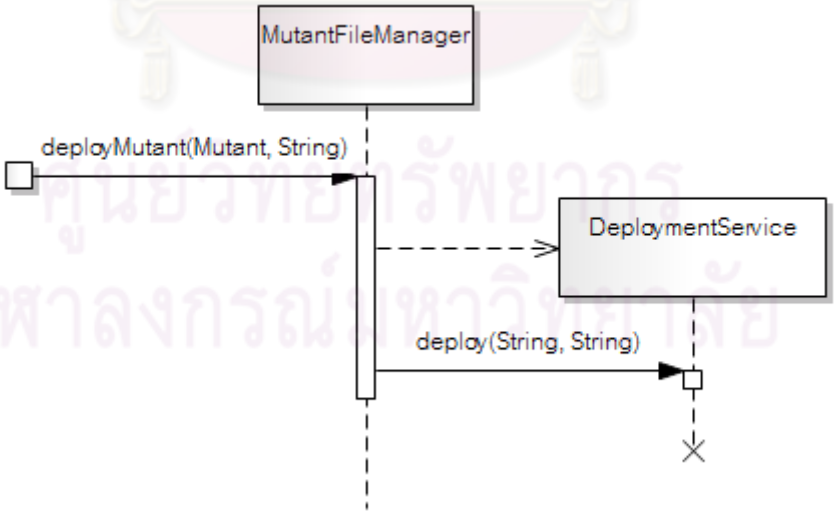
รูปที่ 3.43 แผนภาพลำดับ Input Service Data

4) แผนภาพลำดับ Input Test Cases ดังรูปที่ 3.44 หลังจากที่ได้วิเคราะห์เอกสารดับเบิลยูเอสดีแอลแล้วระบบจะแสดงหน้านำเข้ากรณีทดสอบ เมื่อผู้ทดสอบกรอกข้อมูลกรณีทดสอบแล้วระบบจะทำการสร้างอ็อบเจกต์ TestCase และกำหนดค่ารหัสของกรณีทดสอบ ข้อมูลนำเข้า และผลลัพธ์คาดหวัง และถ้าผู้ทดสอบกดปุ่มบันทึกระบบจะทำการแปลงอ็อบเจกต์ TestCase ไปเป็นเอกสารเอ็กซ์เอ็มแอลซึ่งผู้ทดสอบสามารถโหลดข้อมูลกรณีทดสอบกลับมาใช้ใหม่ได้ในการทดสอบครั้งต่อไป



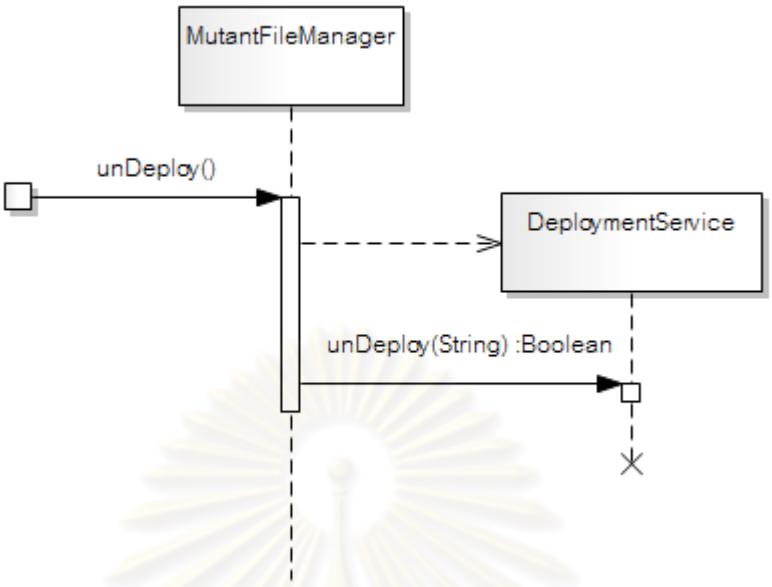
รูปที่ 3.44 แผนภาพลำดับ Input Test Cases

5) แผนภาพลำดับ Deploy Program ดังรูปที่ 3.45 เป็นการติดตั้งโปรแกรม ซึ่งทำโดยผ่านอ็อบเจกต์ DeploymentService โดยเรียกเมท็อด deploy



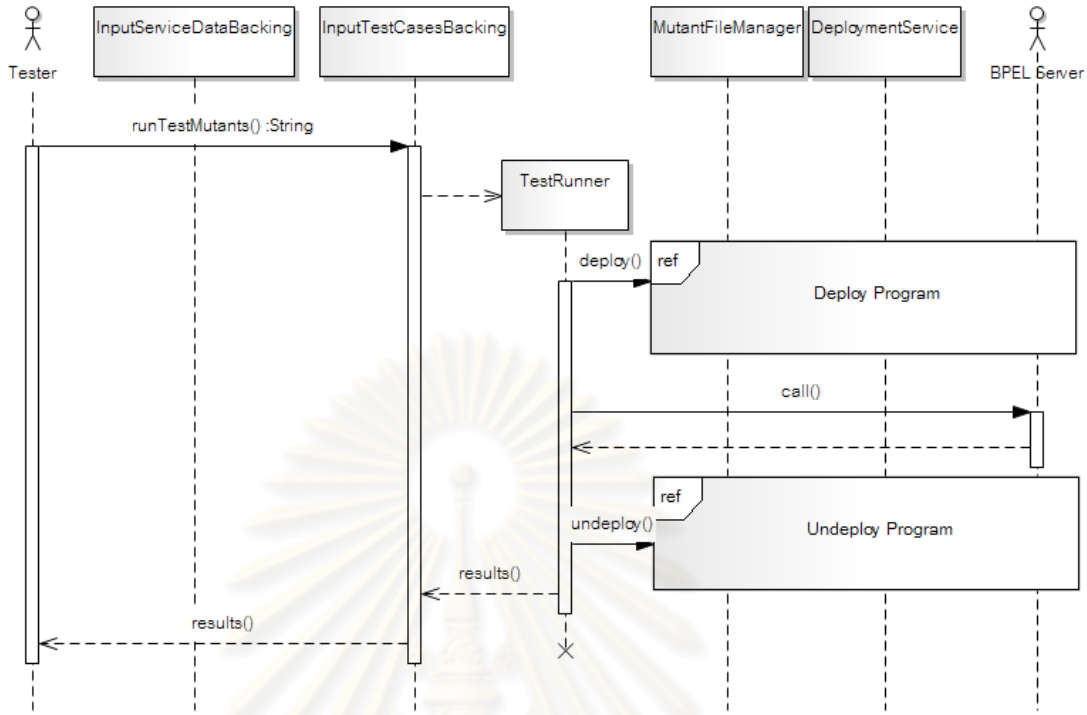
รูปที่ 3.45 แผนภาพลำดับ Deploy Program

6) แผนภาพลำดับ Undeploy Program ดังรูปที่ 3.46 เป็นการติดตั้งโปรแกรม ซึ่งทำโดยผ่านอ็อบเจกต์ DeploymentService โดยเรียกเมท็อด undeploy



รูปที่ 3.46 แผนภาพลำดับ Undeploy Program

7) แผนภาพลำดับ Run Test ดังรูปที่ 3.47 การทดสอบโปรแกรมต้นฉบับและโปรแกรมมิวแตนท์นั้นสามารถทำได้ผ่านอ็อบเจกต์ TestRuner โดยอ็อบเจกต์นี้จะสั่งให้อ็อบเจกต์ DeploymentService ติดตั้งโปรแกรม แล้วจึงทำการเรียกใช้เซอร์วิซของโปรแกรมทดสอบด้วยกรณีทดสอบที่ผู้ทดสอบได้สร้างเอาไว้หลังจากได้ผลการทดสอบแล้ว จึงสั้งให้อ็อบเจกต์ DeploymentService ยกเลิกการติดตั้งโปรแกรม เมื่อทำการทดสอบจนครบมิวแตนท์ทุกตัวแล้วระบบจะรายงานสรุปผลการทดสอบ



รูปที่ 3.47 แผนภาพลำดับ Run Test

3.4.4 รายงานผลการทดสอบ

หลังจากทำการทดสอบโปรแกรมมิวแตนท์ เครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลจะแสดงรายงานการทดสอบซึ่งมีรูปแบบดังตารางที่ 3.10

ตารางที่ 3.10 รูปแบบของรายงานผลการทดสอบ

ตัวดำเนินการมิวเทชัน	**จำนวนมิวแตนท์**	**จำนวนมิวแตนท์ที่ถูกกำจัด**
AOR		
ROR		
LOR		
LOD		
LOI		
รวม	(ผลรวมของมิวแตนท์)	(ผลรวมของจำนวนมิวแตนท์ที่ถูกกำจัด)
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด		
เวลาที่ใช้ในการทดสอบ (วินาที)		

ส่วนประกอบของรายงานผลการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลดังตารางที่ 3.13 มีรายละเอียดดังต่อไปนี้

1) ตัวดำเนินการมิวเทชัน คือ ตัวดำเนินการมิวเทชันที่งานวิจัยนี้ได้นิยามขึ้นเพื่อใช้สำหรับภาษาบีเพลทั้ง 5 ตัว คือ AOR ROR LOR LOD และ LOI

2) จำนวนมิวแตนท์ คือ ผลรวมของจำนวนมิวแตนท์ที่สร้างขึ้นมาจากตัวดำเนินการมิวเทชันแต่ละตัว

3) จำนวนมิวแตนท์ที่ถูกกำจัด คือ ผลรวมของจำนวนมิวแตนท์ที่สร้างขึ้นมาจากตัวดำเนินการมิวเทชันแต่ละตัวที่ถูกกำจัดโดยกรณีทดสอบที่ผู้ทดสอบสร้างขึ้น

4) รวม คือ ผลรวมของมิวแตนท์ทั้งหมดและผลรวมของมิวแตนท์ที่ถูกกำจัดทั้งหมด

5) จำนวนร้อยละของมิวแตนท์ที่ถูกกำจัด คือ ค่าที่คำนวณได้จากสูตร

$$\text{ร้อยละของมิวแตนท์ที่ถูกกำจัด} = \frac{\text{ผลรวมของจำนวนมิวแตนท์ที่ถูกกำจัด}}{\text{ผลรวมของมิวแตนท์ทั้งหมด}} \times 100$$

6) เวลาที่ใช้ในการทดสอบ คือ ผลรวมของเวลาที่ใช้ในการทดสอบมิวแตนท์ทั้งหมดซึ่งมีหน่วยเป็นวินาที

บทที่ 4
การพัฒนาเครื่องมือ

ในบทนี้จะกล่าวถึงสภาพแวดล้อม และโครงสร้างส่วนต่อประสานกับผู้ใช้ของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล ซึ่งมีรายละเอียดดังต่อไปนี้

4.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

1) ฮาร์ดแวร์ (Hardware)

1.1) เครื่องคอมพิวเตอร์แบบพีซี (PC) หน่วยประมวลผลอินเทลคอร์ทูดูโอ ที 9600 2.80 กิกะเอิร์ทซ์ (Intel Core 2 Dual T9600 2.8 GHz)

1.2) หน่วยความจำสำรอง (RAM) 4.00 กิกะไบต์ (4.00 GB)

1.3) ฮาร์ดดิสก์ (Hard disk) 320 กิกะไบต์ (320 GB)

2) ซอฟต์แวร์ (Software)

2.1) ระบบปฏิบัติการ (Operating system) ไมโครซอฟท์วินโดวส์เอ็กซ์พีโพรเฟสชันแนล เซอร์วิซแพค 3 (Microsoft Windows XP Professional Service Pack 3)

2.2) เครื่องมือที่ใช้พัฒนา อีคลิป ไอดีอี รุ่นสำหรับนักพัฒนาภาษาจาวาระดับองค์กร (Eclipse IDE for Java EE Developers)

2.3) ภาษาและเฟรมเวิร์ก (Frameworks) ที่ใช้ในการพัฒนา

- ภาษาจาวา (Java)
- จาวาเซอร์เวอร์เฟส (Java Server Faces)

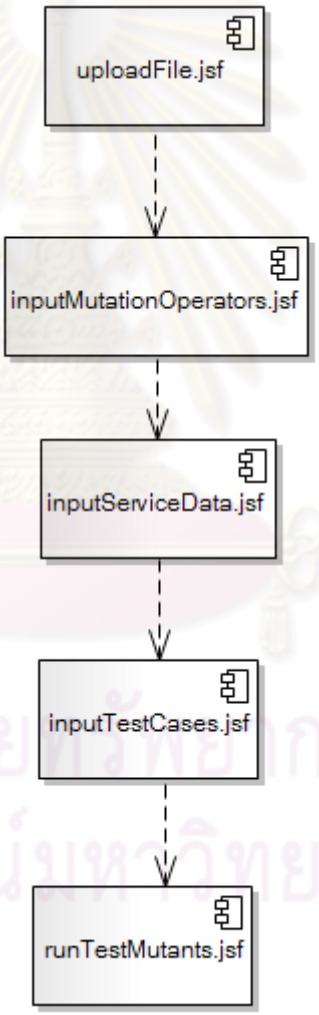
2.4) เครื่องบริการเว็บ (Web server) อาพาเช่ทอมแคท 6.0 (Apache Tomcat 6.0)

2.5) เครื่องประมวลผลบีเพล (BPEL server) อาพาเช่โอดีอี 1.3 (Apache ODE 1.3) ติดตั้งบน อาพาเช่ทอมแคท 6.0

2.6) เว็บบราวเซอร์ (Web browser) มอสซิลล่าไฟล์ฟอกซ์ 3.5 (Mozilla Firefox 3.5)

4.2 โครงสร้างส่วนต่อประสานกับผู้ใช้ของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล

โครงสร้างในส่วนของส่วนต่อประสานกับผู้ใช้ของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลจะได้รับการอธิบายด้วยแผนภาพส่วนประกอบ (Component Diagram) ซึ่งเป็นแผนภาพที่ใช้อธิบายส่วนประกอบต่างๆ ในระบบ โดยแผนภาพส่วนประกอบของส่วนต่อประสานกับผู้ใช้ของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลแสดงดังรูปที่ 4.1

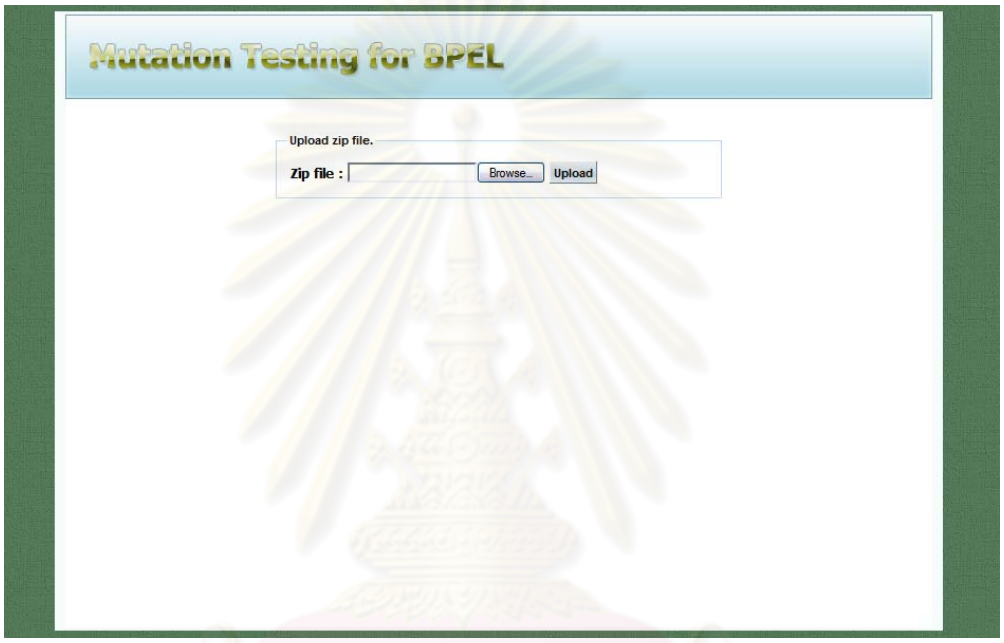


รูปที่ 4.1 แผนภาพส่วนประกอบของเครื่องมือ

จากรูปที่ 4.1 ไฟล์นามสกุล .jsf แต่ละไฟล์จะแทนแต่ละหน้าของส่วนต่อประสานกับผู้ใช้ซึ่งมีรายละเอียดดังต่อไปนี้

4.2.1 หน้า uploadFile.jsf

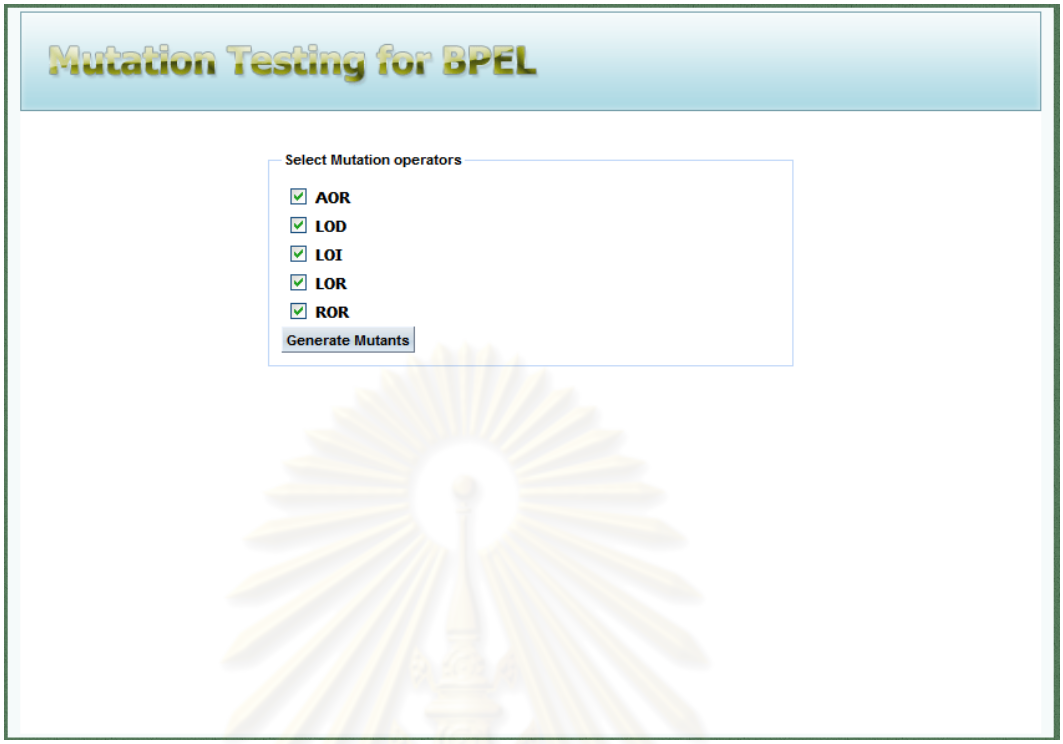
หน้า uploadFile.jsf แสดงดังรูปที่ 4.2 ใช้ในการอัพโหลดไฟล์ต้นฉบับที่อยู่ในรูปไฟล์ซิฟ(Zip) ซึ่งภายในประกอบไปด้วยเอกสารบีเพล เอกสารดับเบิลยูเอสดีแอล และเอกสารเอกซ์เอ็มแอลสคีมา เมื่ออัพโหลดเรียบร้อยแล้วเครื่องมือจะทำการแตกไฟล์และบันทึกลงบนเครื่องบริการเว็บ



รูปที่ 4.2 หน้า uploadFile.jsf

4.2.2 หน้า inputMutationOperators.jsf

หน้า inputMutationOperators.jsf แสดงดังรูปที่ 4.3 มีไว้เพื่อให้ผู้ใช้ได้เลือกตัวดำเนินการมิวเทชันที่ต้องการใช้ในการสร้างมิวแตนท์ ซึ่งตัวดำเนินการมิวเทชันที่ใช้เป็นตัวดำเนินการมิวเทชันที่ได้นิยามไว้ในงานวิจัยนี้ทั้ง 5 ตัว และเมื่อผู้ใช้คลิกที่ปุ่ม Generate Mutants เครื่องมือจะทำการสร้างมิวแตนท์ทั้งหมดและบันทึกลงบนเครื่องบริการเว็บ

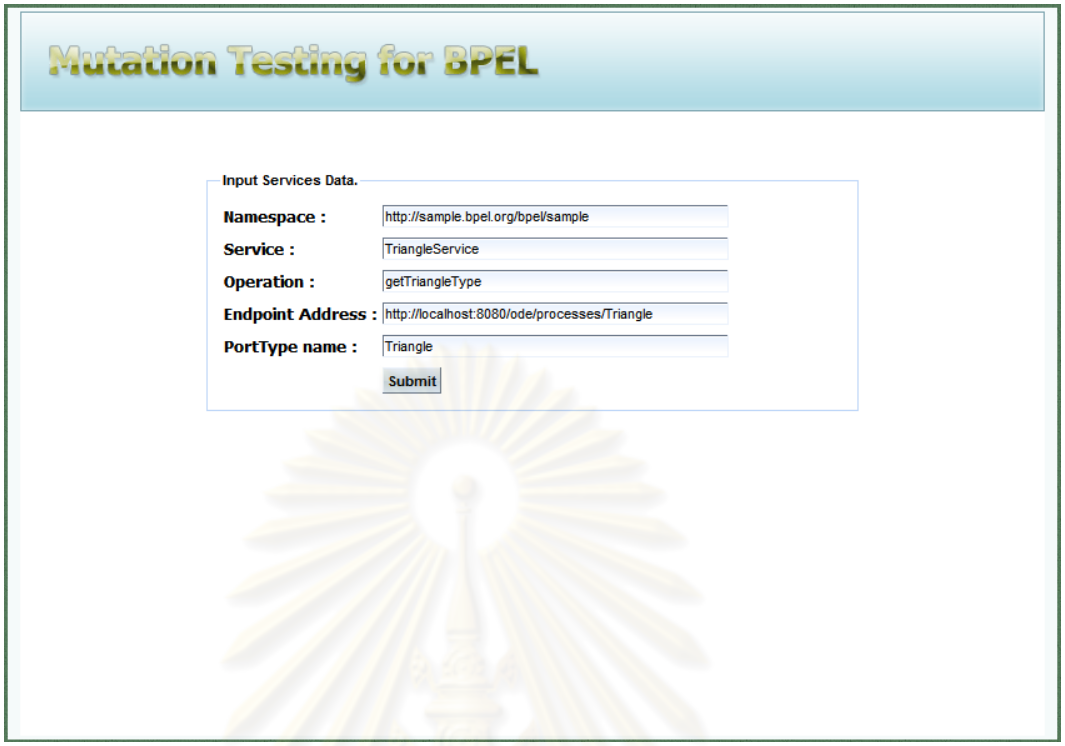


รูปที่ 4.3 หน้า inputMutationOperators.jsf

4.2.3 หน้า inputServiceData.jsf

หน้า inputServiceData.jsf แสดงดังรูปที่ 4.4 มีไว้เพื่อให้ผู้ใช้ได้กรอกข้อมูลเกี่ยวกับเซอร์วิซของบีเพล ซึ่งมีข้อมูลดังต่อไปนี้

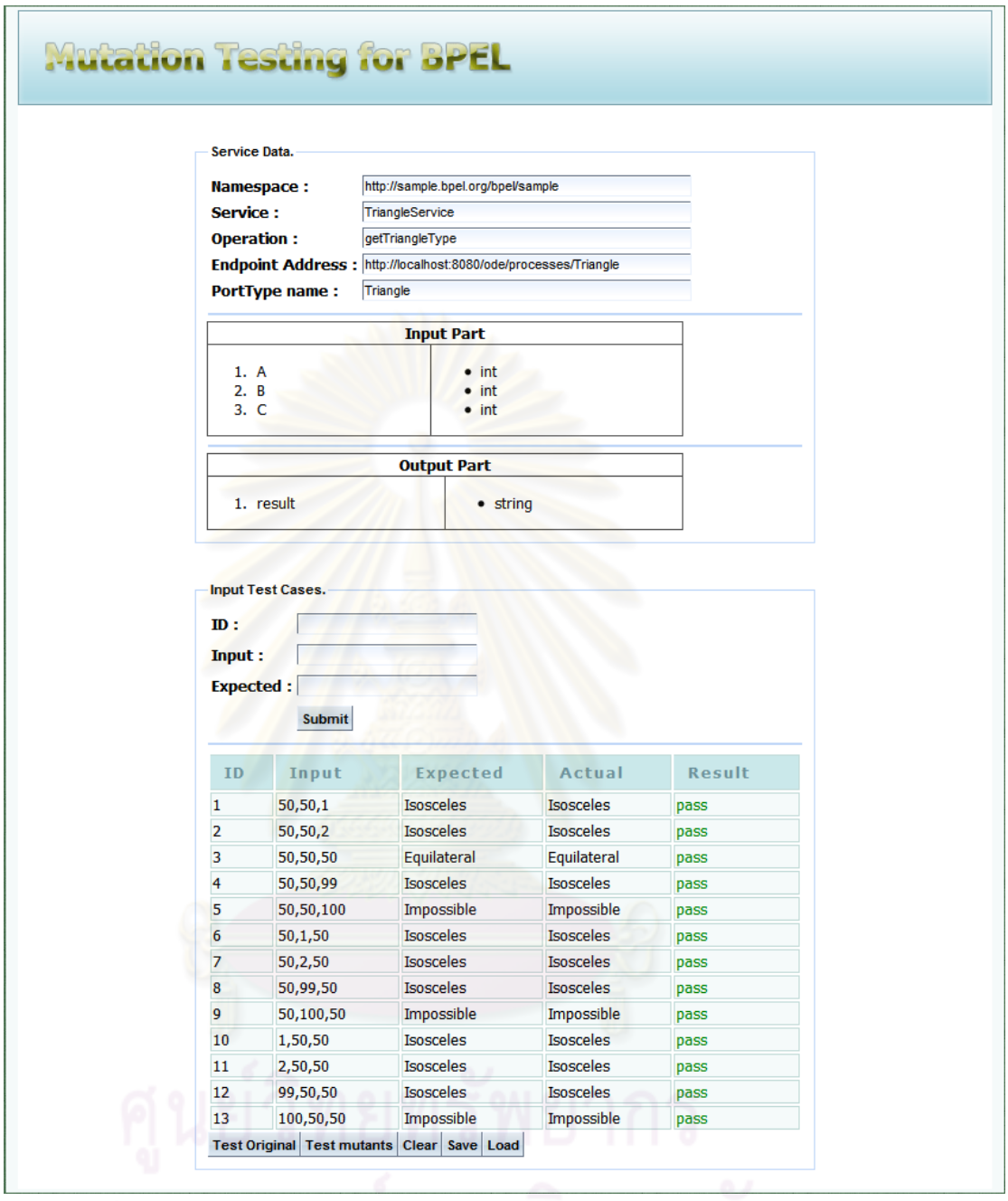
1) Namespace คือ เนมสเปซของเอกสารดับเบิลยูเอสดีแอล
2) Service คือ ชื่อของเซอร์วิซที่ต้องการทดสอบ
3) Operation คือ ชื่อของการดำเนินการที่ต้องการทดสอบ
4) Endpoint Address คือ ที่อยู่ของเซอร์วิซที่ต้องการทดสอบ
5) Port Type Name คือ ชื่อของอินเตอร์เฟสที่ต้องการทดสอบ



รูปที่ 4.4 หน้า inputServiceData.jsf

4.2.4 หน้า inputTestCases.jsf

หน้า inputTestCases.jsf ดังรูปที่ 4.5 จะแสดงจำนวนและชนิดของพารามิเตอร์ของเซอร์วิซที่ต้องการทดสอบทั้งในส่วนของข้อมูลนำเข้าและนำออก และในส่วนด้านล่างจะเป็นส่วนของการนำเข้ากรณีทดสอบซึ่งกรณีทดสอบเหล่านี้เมื่อนำเข้าสู่ระบบแล้วผู้ใช้สามารถบันทึกเก็บไว้ได้ด้วยการคลิกที่ปุ่ม Save โดยระบบจะเก็บกรณีทดสอบเหล่านี้อยู่รูปของเอกสารเอ็กซ์เอ็มแอล ซึ่งผู้ใช้สามารถนำกรณีทดสอบเหล่านี้กลับมาใช้ได้อีกในภายหลังโดยการคลิกที่ปุ่ม Load หลังจากที่นำเข้ากรณีทดสอบเรียบร้อยแล้วผู้ใช้สามารถทดสอบโปรแกรมต้นฉบับโดยการคลิกที่ปุ่ม Test Original หลังจากที่ทดสอบเรียบร้อยแล้วผลการทดสอบจะถูกแสดงในตารางดังรูปที่ 4.5



รูปที่ 4.5 หน้า inputTestCases.jsf

4.2.5 หน้า runTestMutants.jsf

ดังรูปที่ 4.6 หน้านี้จะมาจากการคลิกที่ปุ่ม Test mutants ของหน้า inputTestCases.jsf ดังรูปที่ 4.5 หน้า runTestMutant.jsf นี้มีไว้เพื่อสั่งให้โปรแกรมทำการทดสอบ

มิวแตนท์ทั้งหมดและหลังจากทดสอบเรียบร้อยแล้วเครื่องมือจะทำการแสดงรายงานผลการทดสอบดังตัวอย่างในรูปที่ 4.7

Mutation Testing for BPEL

Run test mutants.

Run Test Mutans

Name	Killed	Original Exp.	Modified Exp.
LOR_If_0	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C and $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_1	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B and $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_2	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A and $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_3	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 and $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_4	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 and $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_5	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 and $input.A > 100 or $input.B > 100 or $input.C > 100
LOR_If_6	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 and $input.B > 100 or $input.C > 100
LOR_If_7	false	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 or $input.C > 100	$input.A + $input.B <= $input.C or $input.A + $input.C <= $input.B or $input.B + $input.C <= $input.A or $input.A <= 0 or $input.B <= 0 or $input.C <= 0 or $input.A > 100 or $input.B > 100 and $input.C > 100

รูปที่ 4.6 หน้า runTestMutants.jsf

Mutation operators	Number of Mutants	Number of Killed Mutants
AOR	60	12
ROR	120	63
LOR	19	7
LOD	0	0
LOI	29	29
Total	228	111
Total killed/Total mutants	48.684208 %	
Test time	264 s	

รูปที่ 4.7 รายงานผลการทดสอบ

บทที่ 5
การทดสอบ

การทดสอบเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลนั้นจะทำได้โดยการสร้างโปรแกรมตัวอย่างซึ่งสร้างขึ้นมาจากภาษาบีเพลขึ้นมา 5 โปรแกรม คือ โปรแกรมหาชนิดของสามเหลี่ยม [13] โปรแกรมขออนุมัติเงินกู้ โปรแกรมค้นหาสินค้าราคาถูก โปรแกรมบวกเลขแบบใช้แอคติวิตี while และ โปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil

5.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

เป็นสภาพแวดล้อมเดียวกับที่ใช้ในการพัฒนาเครื่องมือในบทที่ 4

5.2 ขั้นตอนการทดสอบเครื่องมือ

1) สร้างโปรแกรมทดสอบโดยใช้ภาษาบีเพล สร้างส่วนต่อประสานกับโปรแกรมโดยใช้เอกสารดับเบิลยูเอสดีแอลและเอกสารเอ็กซ์เอ็มแอลสคีมา

2) สร้างเว็บเซอร์วิซที่ถูกเรียกโดยโปรแกรมทดสอบ

3) สร้างกรณีทดสอบสำหรับโปรแกรมทดสอบ

4) ติดตั้งเว็บเซอร์วิซที่สร้างไว้ในหัวข้อ 5.2.2

5) รวบรวมเอกสารของโปรแกรมทดสอบที่สร้างไว้ในข้อ 5.2.1 ให้อยู่ในรูปไฟล์ซิฟ

6) เริ่มทำการทดสอบโดยใช้เครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพล

7) รวบรวมและสรุปผลการทดสอบ

5.3 โปรแกรมที่ใช้ในการทดสอบ

งานวิจัยนี้จะสร้างโปรแกรมตัวอย่างขึ้นมา 5 โปรแกรม คือ โปรแกรมหาชนิดของสามเหลี่ยม (Triangle Process) โปรแกรมขออนุมัติเงินกู้ (Loan Approval Process) โปรแกรมค้นหาสินค้าราคาถูก (Shopping process) โปรแกรมบวกเลขแบบใช้แอคติวิตี while และโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil เพื่อใช้ในการทดสอบเครื่องมือ โดยโปรแกรมเหล่านี้ถูกสร้างขึ้นด้วยภาษาบีเพล ซึ่งโปรแกรมจะประกอบไปด้วยเอกสารบีเพล เอกสารดับเบิลยู

เอสดีแอล และเอกสารเอ็กซ์เอ็มแอลสคีมา เอกสารเหล่านี้จะถูกนำเข้าสู่เครื่องมือเพื่อเข้าสู่กระบวนการทดสอบแบบมิวเทชันต่อไป

5.3.1 โปรแกรมหาชนิดของสามเหลี่ยม

ดังรูปที่ 5.1 แสดงแผนภาพกิจกรรมของโปรแกรมหาชนิดของสามเหลี่ยม โดยโปรแกรมนี้เริ่มต้นการทำงานด้วยการรับค่าความยาวของด้านทั้งสามของสามเหลี่ยมเป็นข้อมูลชนิดจำนวนเต็มซึ่งแทนด้วยค่า A B และ C ซึ่งมีค่าระหว่าง 1 ถึง 100 หลังจากนั้นตรวจสอบว่าสามเหลี่ยมที่รับมาเป็นสามเหลี่ยมชนิดใดโดยมีขั้นตอนการตรวจสอบดังต่อไปนี้

1) ตรวจสอบว่าด้านทั้งสามที่นำเข้าสามารถประกอบเป็นสามเหลี่ยมได้หรือไม่โดยมีเงื่อนไขการตรวจสอบดังนี้ A + B <= C or A + C <= B or B + C <= A or A <= 0 or B <= 0 or C <= 0 or A > 100 or B > 100 or C > 100 ถ้าด้านทั้งสามไม่สามารถประกอบเป็นสามเหลี่ยมได้โปรแกรมจะทำการกำหนดค่า Impossible ให้กับข้อมูลนำออก

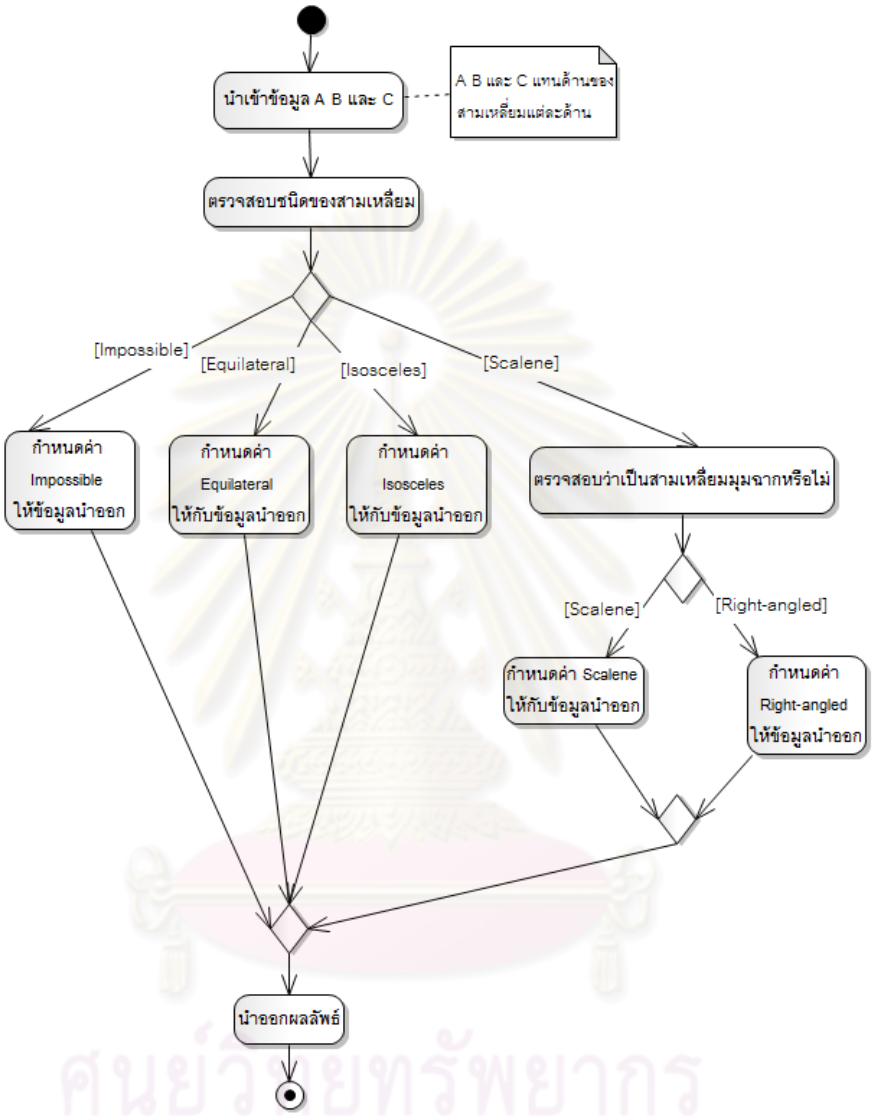
2) ตรวจสอบว่าเป็นสามเหลี่ยมด้านเท่าหรือไม่โดยมีเงื่อนไขการตรวจสอบดังนี้ A = B and B = C and C = A ถ้าเป็นสามเหลี่ยมด้านเท่าโปรแกรมจะทำการกำหนดค่า Equilateral ให้กับข้อมูลนำออก

3) ตรวจสอบว่าเป็นสามเหลี่ยมหน้าจั่วหรือไม่โดยมีเงื่อนไขการตรวจสอบดังต่อไปนี้ (A = B and A != C) or (C = B and C != A) or (A = C and A != B) ถ้าเป็นสามเหลี่ยมหน้าจั่วโปรแกรมจะกำหนดค่า Isosceles ให้กับข้อมูลนำออก

4) ตรวจสอบว่าเป็นสามเหลี่ยมด้านไม่เท่าหรือไม่โดยมีเงื่อนไขการตรวจสอบดังต่อไปนี้ A != B and A != C and B != C ถ้าเป็นสามเหลี่ยมด้านไม่เท่าโปรแกรมจะกำหนดค่า Scalene ให้กับข้อมูลนำออก

4.1) ตรวจสอบว่าเป็นสามเหลี่ยมมุมฉากหรือไม่โดยมีเงื่อนไขการตรวจสอบดังต่อไปนี้ ((A * A) + (B * B) = (C * C)) or ((B * B) + (C * C) = (A * A)) or ((A * A) + (C * C) = (A * A)) ถ้าเป็นสามเหลี่ยมมุมฉากโปรแกรมจะกำหนดค่า Right-angled ให้กับข้อมูลนำออก

หลังจากตรวจสอบชนิดของสามเหลี่ยมเรียบร้อยแล้วโปรแกรมจะคืนค่าชนิดของสามเหลี่ยมให้กับผู้เรียกใช้เซอร์วิซ

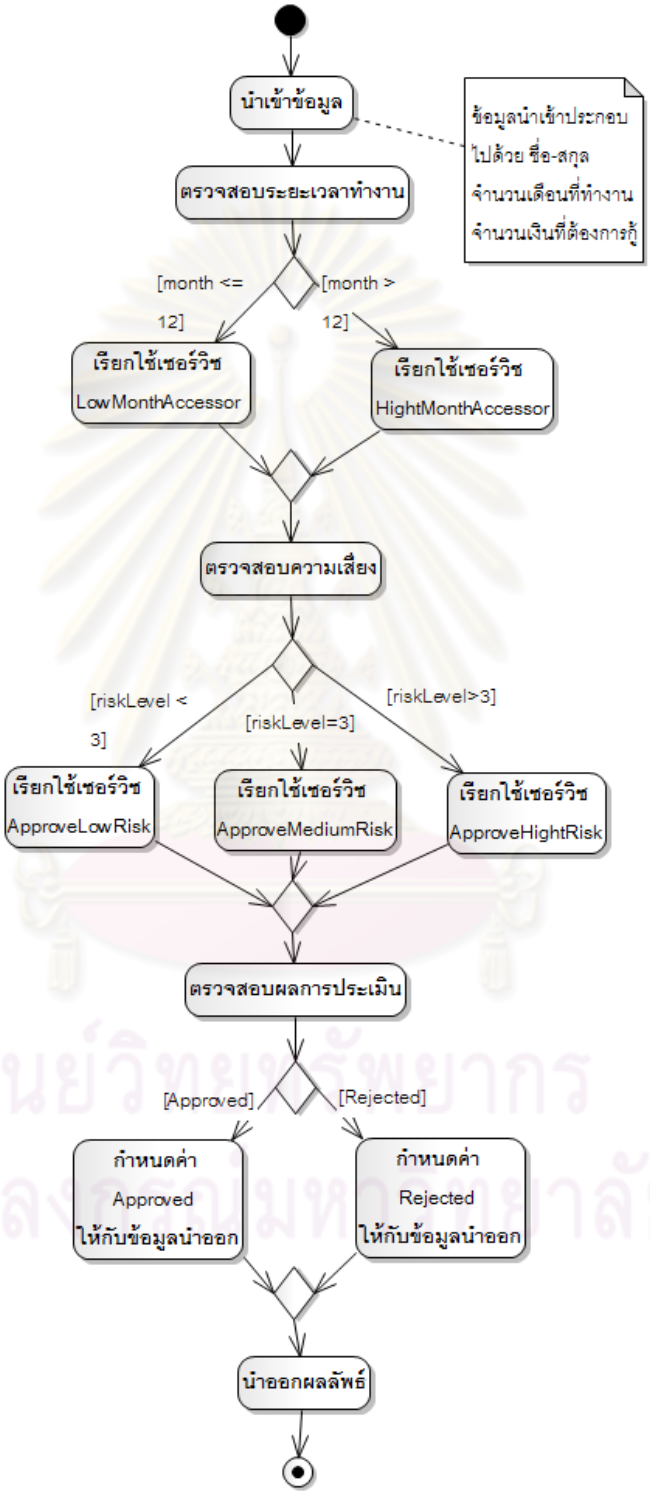


รูปที่ 5.1 แผนภาพกิจกรรมของโปรแกรมหาชนิดของสามเหลี่ยม

5.3.2 โปรแกรมขออนุมัติเงินกู้

ดังรูปที่ 5.2 แสดงแผนภาพกิจกรรมของโปรแกรมขออนุมัติเงินกู้ ซึ่งเริ่มต้นการทำงานด้วยการรับข้อมูลชื่อ-สกุลซึ่งเป็นข้อมูลชนิด String จำนวนเงินที่ต้องการกู้เป็นข้อมูลชนิด Float และจำนวนเดือนที่ทำงานเป็นข้อมูลชนิด Integer หลังจากนั้นนำข้อมูลเหล่านี้ไปเรียกใช้เว็บเซอร์วิซเพื่อประเมินความเสี่ยงในแต่ละระดับ หลังจากนั้นจะได้ระดับของความเสี่ยงและนำค่า

ระดับความเสี่ยงที่ได้ไปหาว่าควรจะเรียกใช้เว็บเซอร์วิซที่ใช้ในการอนุมัติการกู้เงินในความเสี่ยงระดับใด ซึ่งหลังจากโปรแกรมประมวลผลแล้วจะได้ผลลัพธ์ Approved หรือ Rejected



รูปที่ 5.2 แผนภาพกิจกรรมของโปรแกรมขออนุมัติเงินกู้

5.3.3 โปรแกรมค้นหาสินค้าราคาถูก

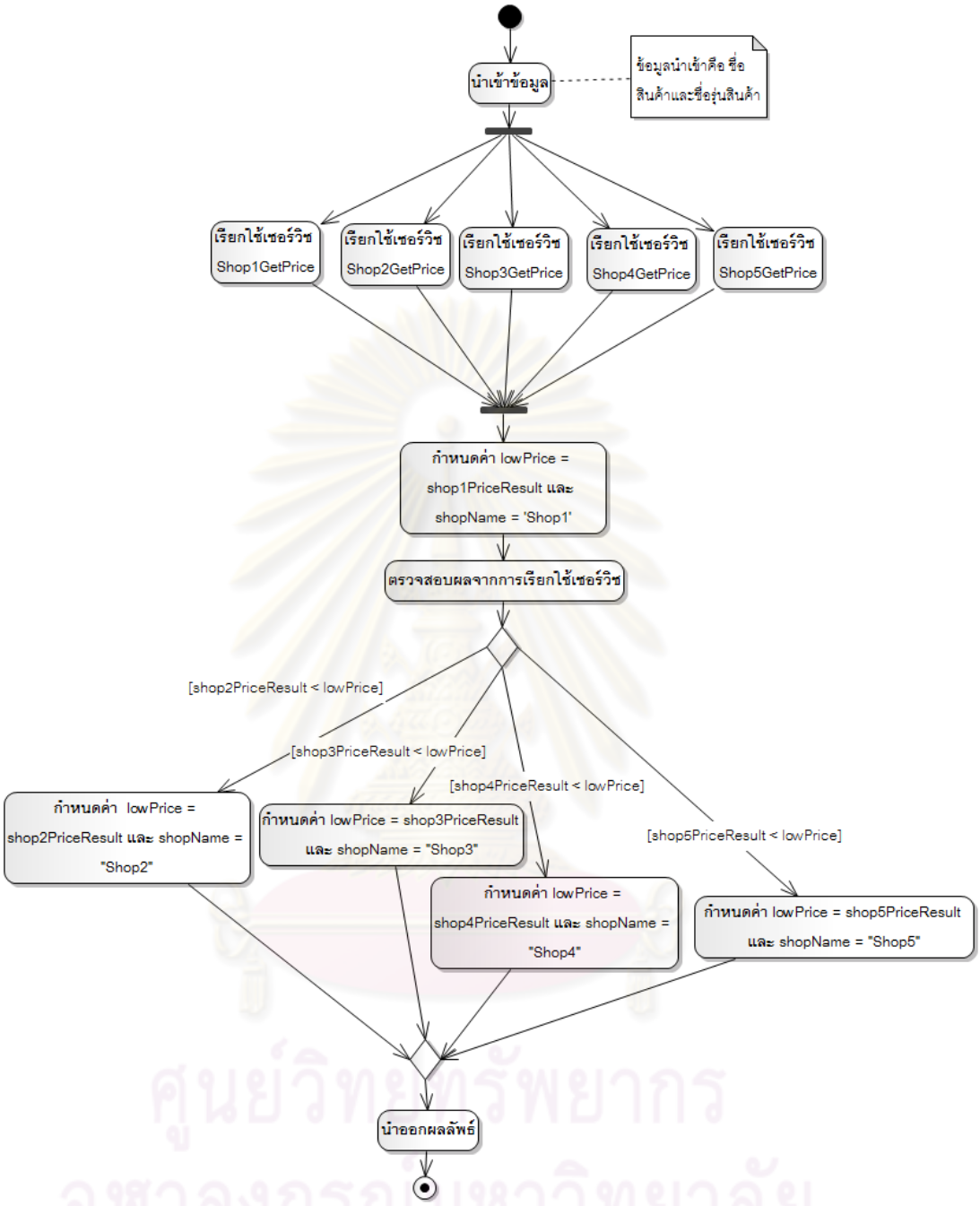
ดังรูปที่ 5.3 แสดงแผนภาพกิจกรรมของโปรแกรมค้นหาสินค้าราคาถูก โดยเริ่มต้นด้วยการรับข้อมูลชื่อสินค้าและชื่อรุ่นสินค้าซึ่งเป็นข้อมูลชนิด String จากผู้เรียกใช้เว็บเซอร์วิซแล้วนำข้อมูลเหล่านี้ไปตรวจสอบราคาสินค้าจากเว็บเซอร์วิซของร้านจำหน่ายสินค้าแต่ละร้าน เมื่อประมวลผลเสร็จจะได้ผลลัพธ์เป็นชื่อร้านค้าที่จำหน่ายสินค้าราคาถูกที่สุด

5.3.4 โปรแกรมบวกเลขแบบใช้แอคติวิตี while

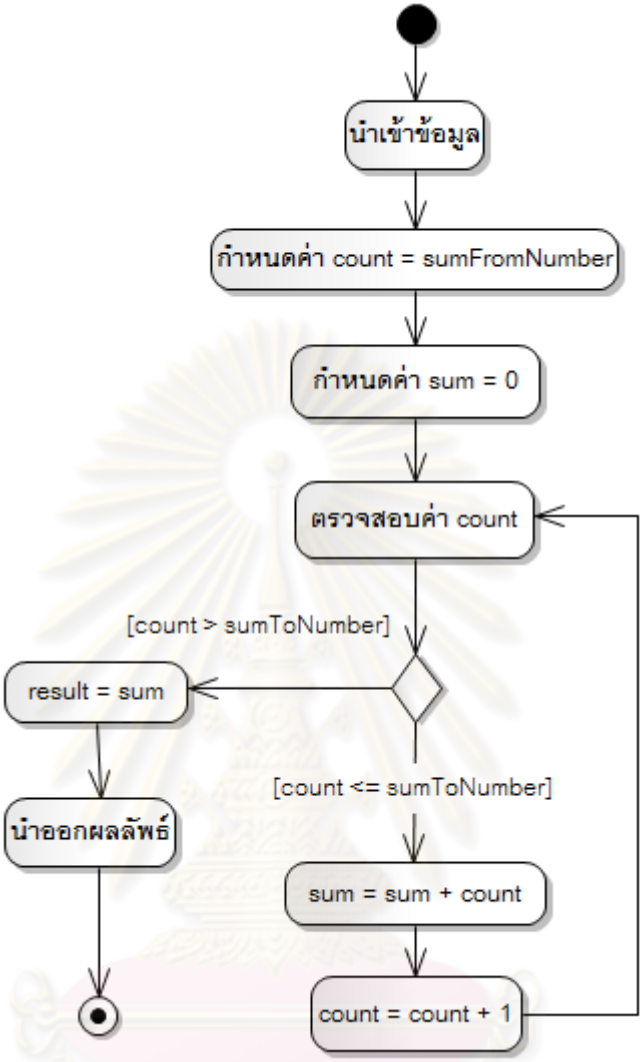
ดังรูปที่ 5.4 แสดงแผนภาพกิจกรรมของโปรแกรมบวกเลขแบบใช้แอคติวิตี while ซึ่งโปรแกรมทำงานโดยรับค่านำเข้าเป็นข้อมูลชนิด Integer สองค่าคือ ค่าเริ่มต้นของการบวก และค่าสุดท้ายของการบวก โดยทั้งสองค่าจะมีค่าอยู่ระหว่าง 1 ถึง 20 เมื่อรับค่าเข้ามาแล้วโปรแกรมจะทำการหาผลรวมของค่าตั้งแต่ ค่าเริ่มต้นของการบวกถึงค่าสุดท้ายของการบวก เช่น ถ้าค่าเริ่มต้นของการบวกคือ 3 และค่าสุดท้ายของการบวกคือ 7 โปรแกรมจะทำการหาค่า 3+4+5+6+7 ซึ่งมีค่าเท่ากับ 25 โดยโปรแกรมนี้ใช้แอคติวิตี while ในการหาผลรวม

5.3.5 โปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil

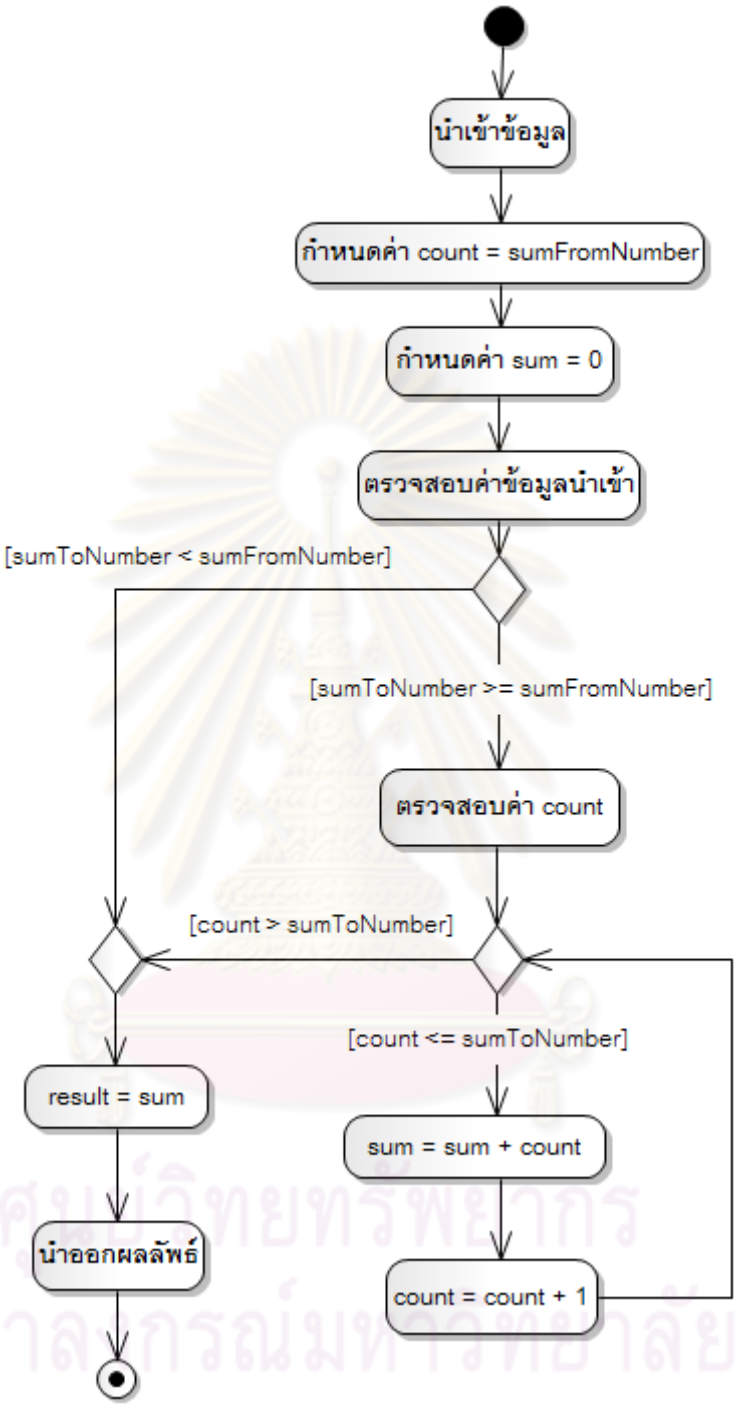
ดังรูปที่ 5.5 แสดงแผนภาพกิจกรรมของโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil ซึ่งหลักการทำงานเหมือนกับโปรแกรมในหัวข้อ 5.3.4 แตกต่างที่โปรแกรมนี้ใช้แอคติวิตี repeatUntil ในการหาผลรวม



รูปที่ 5.3 แผนภาพกิจกรรมของโปรแกรมค้นหาสินค้าราคาถูก



รูปที่ 5.4 แผนภาพกิจกรรมของโปรแกรมบวกเลขแบบใช้แอคติวิตี while



รูปที่ 5.5 แผนภาพกิจกรรมของโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil

5.4 กรณีทดสอบสำหรับโปรแกรมตัวอย่าง

เพื่อที่จะทดสอบความถูกต้องของการรายงานผลการทดสอบของเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลที่สร้างขึ้นมา ดังนั้นจึงต้องสร้างกรณีทดสอบสำหรับโปรแกรมหาชนิดของสามเหลี่ยม โปรแกรมขออนุมัติเงินกู้ และโปรแกรมค้นหาสินค้าราคาถูก ดังตารางที่ 5.1 5.2 5.3 ตามลำดับ และกรณีทดสอบของโปรแกรมบวกเลขทั้งแบบใช้แอคติวิตี while และแบบใช้แอคติวิตี repeatUntil แสดงดังตารางที่ 5.4

ตารางที่ 5.1 กรณีทดสอบของโปรแกรมหาชนิดของสามเหลี่ยม

รหัส	ข้อมูลนำเข้า	ผลลัพธ์คาดหวัง
1	50,50,1	Isosceles
2	50,50,2	Isosceles
3	50,50,50	Equilateral
4	50,50,99	Isosceles
5	50,50,100	Impossible
6	50,1,50	Isosceles
7	50,2,50	Isosceles
8	50,99,50	Isosceles
9	50,100,50	Impossible
10	1,50,50	Isosceles
11	2,50,50	Isosceles
12	99,50,50	Isosceles
13	100,50,50	Impossible

ตารางที่ 5.2 กรณีทดสอบของโปรแกรมขออนุมัติเงินกู้

รหัส	ข้อมูลนำเข้า	ผลลัพธ์คาดหวัง
1	500000,D,3	Approved
2	500000,D,4	Approved
3	500000,D,12	Approved
4	500000,D,35	Approved
5	500000,D,36	Approved
6	500000,A,12	Rejected
7	500000,B,12	Approved
8	500000,C,12	Approved
9	10000,D,12	Approved
10	10001,D,12	Approved
11	999999,D,12	Approved
12	1000000,D,12	Approved

ตารางที่ 5.3 กรณีทดสอบของโปรแกรมค้นหาสินค้าราคาถูก

รหัส	ข้อมูลนำเข้า	ผลลัพธ์คาดหวัง
1	A1,ProductA	Shop1
2	A2,ProductA	Shop2
3	A3,ProductA	Shop3
4	B1,ProductB	Shop1
5	B2,ProductB	Shop2
6	B3,ProductB	Shop3
7	C1,ProductC	Shop1
8	C2,ProductC	Shop3
9	C3,ProductC	Shop2

ตารางที่ 5.4 กรณีทดสอบของโปรแกรมบวกเลขทั้งแบบใช้แอคติวิตี while และ แบบใช้แอคติวิตี repeatUntil

รหัส	ข้อมูลนำเข้า	ผลลัพธ์คาดหวัง
1	1, 20	210
2	10, 20	165
3	20, 20	20
4	20, 1	0
5	20, 10	0

5.5 ผลการทดสอบ

หลังจากทำการทดสอบโปรแกรมตัวอย่างทั้ง 5 โปรแกรมแล้วได้ผลการทดสอบดังต่อไปนี้

5.5.1 ผลการทดสอบของโปรแกรมหาชนิดของสามเหลี่ยม

เมื่อทดสอบโปรแกรมหาชนิดของสามเหลี่ยมโดยใช้กรณีทดสอบที่สร้างไว้ดังตารางที่ 5.1 ได้ผลการทดสอบดังตารางที่ 5.5

ตารางที่ 5.5 ผลการทดสอบของโปรแกรมหาชนิดของสามเหลี่ยม

ตัวดำเนินการมิวเทชัน	จำนวนมิวแตนท์	จำนวนมิวแตนท์ที่ถูกกำจัด
AOR	60	12
ROR	120	63
LOR	19	7
LOD	0	0
LOI	29	29
รวม	228	111
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด	48.68 %	
เวลาที่ใช้ในการทดสอบ	264 วินาที	

5.5.2 ผลการทดสอบของโปรแกรมขออนุมัติเงินกู้

เมื่อทดสอบโปรแกรมขออนุมัติเงินกู้โดยใช้กรณีทดสอบที่สร้างไว้ดังตารางที่ 5.2 ได้ผลการทดสอบดังตารางที่ 5.6

ตารางที่ 5.6 ผลการทดสอบของโปรแกรมขออนุมัติเงินกู้

ตัวดำเนินการมิวเทชัน	**จำนวนมิวแตนท์**	**จำนวนมิวแตนท์ที่ถูกกำจัด**
AOR	0	0
ROR	35	20
LOR	1	1
LOD	0	0
LOI	8	8
รวม	44	29
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด	65.91 %	
เวลาที่ใช้ในการทดสอบ	73 วินาที	

5.5.3 ผลการทดสอบของโปรแกรมค้นหาสินค้าราคาถูก

เมื่อทดสอบโปรแกรมค้นหาสินค้าราคาถูกโดยใช้กรณีทดสอบที่สร้างไว้ดังตารางที่ 5.3 ได้ผลการทดสอบดังตารางที่ 5.7

ตารางที่ 5.7 ผลการทดสอบของโปรแกรมค้นหาสินค้าราคาถูก

ตัวดำเนินการมิวเทชัน	จำนวนมิวแตนท์	จำนวนมิวแตนท์ที่ถูกกำจัด
AOR	0	0
ROR	20	15
LOR	0	0
LOD	0	0
LOI	4	4
รวม	24	19
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด	79.17 %	
เวลาที่ใช้ในการทดสอบ	144 วินาที	

5.5.4 ผลการทดสอบของโปรแกรมบวกเลขแบบใช้แอคติวิตี while โดยใช้กรณีทดสอบที่สร้างไว้ดังตารางที่ 5.4 ได้ผลการทดสอบดังตารางที่ 5.8

ตารางที่ 5.8 ผลการทดสอบของโปรแกรมบวกเลขแบบใช้แอคติวิตี while

ตัวดำเนินการมิวเทชัน	จำนวนมิวแตนท์	จำนวนมิวแตนท์ที่ถูกกำจัด
AOR	0	0
ROR	20	14
LOR	3	3
LOD	1	1
LOI	5	5
รวม	29	23
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด	79.31 %	
เวลาที่ใช้ในการทดสอบ	165 วินาที	

5.5.5 ผลการทดสอบของโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil โดยใช้กรณีทดสอบที่สร้างไว้ดังตารางที่ 5.4 ได้ผลการทดสอบดังตารางที่ 5.9

ตารางที่ 5.9 ผลการทดสอบของโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil

ตัวดำเนินการมิวเทชัน	**จำนวนมิวแตนท์**	**จำนวนมิวแตนท์ที่ถูกกำจัด**
AOR	0	0
ROR	20	16
LOR	2	2
LOD	1	1
LOI	5	5
รวม	28	24
จำนวนร้อยละของมิว-แตนท์ที่ถูกกำจัด	85.71 %	
เวลาที่ใช้ในการทดสอบ	157 วินาที	

5.6 สรุปผลการทดสอบ

หลังจากที่ทำการทดสอบเครื่องมือโดยใช้โปรแกรมตัวอย่างที่สร้างขึ้นมาทั้ง 5 โปรแกรม โดยให้เครื่องมือสร้างมิวแตนท์ขึ้นมาอย่างอัตโนมัติด้วยตัวดำเนินการมิวเทชันทั้ง 5 ตัว ดำเนินการจากผลการทดสอบในตารางที่ 5.5 – 5.9 ผลการสร้างมิวแตนท์ของเครื่องมือแสดงให้เห็นว่ามีการสร้างมิวแตนท์โดยการดัดแปลงนิพจน์ ของโปรแกรมต้นฉบับได้อย่างถูกต้องตรงตามหลักการทดสอบแบบมิวเทชัน และหลังจากทำการทดสอบแล้วเครื่องมือแสดงผลการทดสอบได้อย่างถูกต้อง สามารถรายงานจำนวนมิวเตนท์ที่ยังมีชีวิตอยู่ มิวแตนท์ที่ถูกกำจัด และสามารถระบุมิวแตนท์ที่ยังมีชีวิตอยู่และมิวแตนท์ที่ถูกกำจัดได้อย่างถูกต้อง นอกจากนี้เครื่องมือสามารถคำนวณค่าร้อยละของจำนวนมิวแตนท์ที่ถูกกำจัดต่อมิวแตนท์ทั้งหมดได้อย่างถูกต้อง

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้ได้เสนอวิธีการประยุกต์ใช้การทดสอบแบบมิวเทชันสำหรับภาษาบีเพล โดยวิทยานิพนธ์นี้ได้นิยามตัวดำเนินการมิวเทชันสำหรับภาษาบีเพลโดยการนิยามตัวดำเนินการมิวเทชันนั้นใช้การทดสอบแบบมิวเทชันโดยการคัดเลือกทำให้ได้ตัวดำเนินการมิวเทชันสำหรับภาษาบีเพล 5 ตัว คือ AOR ROR LOR LOD และ LOI

เพื่อที่จะทดสอบเครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลวิทยานิพนธ์นี้ได้สร้างโปรแกรมตัวอย่างขึ้นมา 5 โปรแกรม คือ โปรแกรมหาชนิดของสามเหลี่ยม โปรแกรมขออนุมัติเงินกู้ โปรแกรมค้นหาสินค้าราคาถูก โปรแกรมบวกเลขแบบใช้แอคติวิตี while และโปรแกรมบวกเลขแบบใช้แอคติวิตี repeatUntil หลังจากที่ทำการทดสอบเครื่องมือโดยใช้โปรแกรมตัวอย่างที่สร้างขึ้นมาทั้ง 5 โปรแกรม โดยให้เครื่องมือสร้างมิวแตนท์ขึ้นมาอย่างอัตโนมัติด้วยตัวดำเนินการมิวเทชันทั้ง 5 ตัวดำเนินการ ผลการสร้างมิวเตนท์ของเครื่องมือแสดงให้เห็นว่ามีการสร้างมิวแตนท์โดยการดัดแปลงนิพจน์ ของโปรแกรมต้นฉบับได้อย่างถูกต้องตรงตามหลักการทดสอบแบบมิวเทชัน และหลังจากทำการทดสอบแล้วเครื่องมือแสดงผลการทดสอบได้อย่างถูกต้อง ซึ่งสามารถรายงานจำนวนมิวเตนท์ที่ยังมีชีวิตอยู่ มิวแตนท์ที่ถูกกำจัด และสามารถระบุมิวแตนท์ที่ยังมีชีวิตอยู่และมิวแตนท์ที่ถูกกำจัดได้อย่างถูกต้อง นอกจากนี้เครื่องมือสามารถคำนวณค่าร้อยละของจำนวนมิวแตนท์ที่ถูกกำจัดต่อมิวแตนท์ทั้งหมดได้อย่างถูกต้อง

6.2 ข้อจำกัดของงานวิจัย

1) วิทยานิพนธ์นี้สนใจเพียงภาษาบีเพลที่เป็นมาตรฐานเท่านั้น ซึ่งในการนำไปใช้จริงบริษัทผู้ผลิตเครื่องประมวลผลบีเพลได้สร้างส่วนต่อขยายสำหรับใช้กับเครื่องประมวลผลบีเพลของตน เช่น เครื่องประมวลผลบีเพลเว็บสเฟียร์โปรเซสเซอร์เวอร์ (WebSphere Process Server) ของบริษัทไอบีเอ็มสามารถประมวลผลภาษานิพจน์ที่เขียนโดยใช้ภาษาจาวาได้ ซึ่งไม่มีในภาษา

บีเพลที่เป็นมาตรฐาน ดังนั้นจึงควรปรับปรุงงานวิจัยเพื่อให้สามารถรองรับกับการใช้งานจริงได้ดียิ่งขึ้น

2) วิทยานิพนธ์นี้ไม่ได้สนใจการทำงานแบบพร้อมกัน (Concurrency) ซึ่งสามารถทำให้เกิดข้อผิดพร่องในรูปแบบที่นอกเหนือจากที่นิยามไว้ในวิทยานิพนธ์นี้ได้

3) ในส่วนของข้อมูลนำเข้าและข้อมูลนำออกของเซอร์วิซของโปรแกรมทดสอบในวิทยานิพนธ์นี้รองรับเฉพาะข้อมูลนำเข้าและข้อมูลนำออกชนิด Byte Decimal Double Float Integer Long Short และ String เท่านั้น ดังนั้นจึงควรปรับปรุงให้สามารถรองรับชนิดข้อมูลชนิดอื่นๆได้เพื่อให้มีประสิทธิภาพมากขึ้น

4) เครื่องมือการทดสอบแบบมิวเทชันสำหรับภาษาบีเพลที่สร้างขึ้นจากวิทยานิพนธ์นี้ไม่สามารถสร้างกรณีทดสอบได้อย่างอัตโนมัติ ดังนั้นจึงควรปรับปรุงเครื่องมือให้สามารถสร้างกรณีทดสอบได้เพื่อเพิ่มประสิทธิภาพการทำงานของเครื่องมือ

รายการอ้างอิง

[1] Yuan, Y., Li, Z., and Sun, W. A Graph-search based approach to BPEL4WS test generation. Proceedings of the International Conference on Software Engineering Advances (ICSEA'06) (2006): 14.

[2] Demillo, R. A., and Offutt, A. J. Constraint-based automatic test data generation. IEEE transaction on software engineering (1991): 900-910.

[3] Offutt, A. J., and Untch, R.H. Mutation 2000: uniting the orthogonal. Mutation testing in the twentieth and twenty-first centuries (2000): 45-55.

[4] Mathur, A. P. Performance effectiveness and reliability issue in software testing. In Proceeding of The Fifteenth Annual International Computer Software and Applications Conference Tokyo, Japan (September 1991): 604-605.

[5] King, K.N., and Offutt, A.J. A fortran language system for mutation-base software testing. Software-Practice and Experience 21 (July 1991): 685-718.

[6] Offutt, A.J., Rothermel, G., and Zapf, C., An experimental evaluation of selective mutation. In Proceedings of The Fifteenth International Conference on Software Engineering, IEEE Computer Society Press, Baltimore, MD (May 1993): 100-107.

[7] Offutt, A.J., Lee, A., Rothermel, G., Untch, R., and Zapf, C. An experimental determination of sufficient mutation operators. ACM Transactions on Software Engineering Methodology 5 (April 1996): 99-118.

[8] Vlist, E. Using W3C XML Schema[Online]. Available from: http://www.xml.com/pub/a/2000/11/29/Schemas/part1.html [2001, October 17]

[9] Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. XML Schema Part1 : Structures Second Edition[Online]. Available from: http://www.w3.org/TR/xmlschema-1/ [2004, October 28]

[10] Christesen, E., Curbera, F., Meredith, G., and Weerawarana, S. Web Services Description Language (WSDL) 1.1[Online]. Available from: http://www.w3.org/TR/wsdl [2001, March]

[11] Sharpe, B. M. SOA for the Business Developer: Concepts, BPEL, and SCA. MC Press, 2007.

[12] Offutt, A.J., Voas, J., Payne, J. Mutation operator for Ada. Technical Report ISSE-TR-96-09, Information and Software Systems Engineering , George Mason University, Fairfax, Virgina (March 1996)

[13] Jorgensen, P.C. Software Testing: A Craftsman's Approach, Second Edition. CRC Press LLC, 2002.

ภาคผนวก

ผลงานที่ตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “Mutation Testing for Expression Modification Operator of BPEL” โดย นัฐพล ไทยสาครพันธ์ และรศ.ดร.ธาราทิพย์ สุวรรณศาสตร์ ในงานประชุมทางวิชาการ 13th National Computer Science and Engineering Conference (NCSEC 2009) ณ จังหวัดกรุงเทพฯ ประเทศไทย ระหว่างวันที่ 4-6 พฤศจิกายน 2552 โดยงานวิจัยนี้ได้รับการคัดเลือกให้ได้รับรางวัล Best Paper Award ในหมวด Computer Software

Mutation Testing for Expression Modification Operator of BPEL

Natthapol Thaisakonpun[1] and Taratip Suwannasart[2]
Software Engineering Laboratory
Center of Excellence in Software Engineering
Faculty of Engineering
Chulalongkorn University
Bangkok, Thailand
t_natthapol@hotmail.com[1] , Taratip.S@chula.ac.th[2]

Abstract

Business Process Execution Language (BPEL) is an XML-based language used for the definition and execution of business process by using Web Services. BPEL is a coordination and composition language for Web Services. We propose a technique for testing BPEL by using mutation testing. Mutation testing or mutation analysis is a fault-based testing method for measuring the adequacy of test cases. We apply mutation testing to BPEL by injecting fault to BPEL document in order to generate mutants. We identify mutation operators by following selective mutation to decrease number of mutation operators. Mutation testing is the difficult testing method because mutation testing generates large number of mutants. Therefore, a prototype of a mutation testing tool for BPEL aims at real software projects is implemented. This tool is used for automatically generating mutants, deploys mutants to BPEL server, executes test cases, and reports test results.

Key Words: BPEL, Mutation Operator, Mutation Testing, Software Testing

1. Introduction

Service Oriented Architecture (SOA) is software architecture that separates software to several parts by responsibility. Each part is called service. Each service uses a loosely coupled integration model to other. Nowadays, SOA is implemented by Web Services. Web Services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks [1]. Web Services can be accessed over the network by using XML messages that follow the SOAP standard. The operations provided by Web Services are written in Web Services Description Language (WSDL).

WSDL [2] is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used for communicating.

Business Process Execution Language (BPEL) is an XML-based language used for the definition and execution of business process by using Web Services. BPEL is a coordination and composition language for Web Services.

There are several publications that have proposed testing technique for BPEL [3][4][5].

The mutation testing has been widely applied to many programming languages [6][7][8] except BPEL. Thus we apply the mutation testing for BPEL, which identifies mutation operators by following selective mutation for decrease number of mutants and creates a prototype of a mutation testing tool for BPEL for demonstration of our approach.

This paper is organized as follows. Section 2 introduces background related to some basics of BPEL, mutation testing, and selective mutation. The proposed applying mutation testing to BPEL is elaborated in section 3. The prototype of a mutation testing tool for BPEL is shown in section 4.

2. Background

2.1. BPEL

Business Process Execution Language (BPEL) is an XML-based language for creating a process, which is a set of logical steps called activities that guide a workflow. Each BPEL activity is equivalent to statement or function call in structural programming language. Activities are categorized in basic activities and structured activities. Basic activities such as receive, reply, assign, and invoke do discrete tasks. Structured activities such as if, repeatUntil, foreach, and while specify an order or condition that affects the circumstance for running a set of other, embedded activities, which may be basic, structured, or both [9].

2.2. Mutation testing

Mutation testing or mutation analysis is a fault-based testing method used for measuring the adequacy of a set of created test cases [10]. Mutation testing is done by applying mutation operators to a source program. The result of applying the mutation operators are new programs, each containing one fault and is called a mutant. A mutant is killed when the test result is fail. The mutants are limited to simple changes on the basis of coupling effect, which imply that complex faults are coupled to simple faults in such a way that a test data set that detect all simple faults in a program will detect most complex faults[10].

The mutation operators are rule for modifying original program, for example, would replace an operand by every other syntactically legal operand, or modify expressions by replacing operators and inserting new operators, or deleting the entire statement.

After executing all mutants the tester is left two kind of information. The portions of the mutants that die indicate how well the program has been tested [10]. The live mutants that could not be distinguished by test cases from original program are called equivalent mutant. To assess the adequacy of a test set, the mutation score is computed as follows.

$$M = \frac{D}{T - E}$$

M = Mutation score, D = Number of dead mutants, T = Number of total mutants, E = Number of equivalence mutants

The tester's main goal is to improve the mutation score to 100% which indicating all mutants has been detected [11].

2.3. Selective mutation

The 22 mutation operators used by Mothra system [12], each generates mutants at difference rates. Mathur [13] has proposed selective mutation by without using 2 mutation operators that generate most mutants. This method is called 2-selective mutation by Offutt [14]. Offutt [14] has extended to 3-selective, 4-selective and 6-selective mutation.

Since, results from 3-selective, 4-selective and 6-selective mutation are not significantly difference from the original mutation. Offutt [15] has extended selective mutation by divide the mutation operators that used by Mothra system to 3 categories based on the syntactic elements that they modify.

1) Operand Replacement Operators
2) Expression Modification Operators
3) Statement Modification Operators

Specifically, the results indicate that the mutation operators that replace all operands with all syntactically legal operands (Operand Replecement Operators) and the mutation operators that modify entire statements (Statement Modification Operators) add very little to the effectiveness of mutation testing [15].

3. Applying mutation testing to BPEL

3.1. Expression modification operators for BPEL

We designed mutation operators for BPEL by following the selective mutation approach. The selective mutation results found that the mutation operators, that replace all operands with all syntactically legal operands, and that modify entire statements add very little to the effectiveness of mutation testing. Therefore we consider mutation operator that modify expressions. We use 5 mutation operators that are summarized in table 1, and then discuss each operator in detail.

Table 1: 5 Mutation operators for BPEL

AOR	Arithmetic operator replacement
ROR	Relational operator replacement
LOR	Logical operator replacement
LOD	Logical operator delete
LOI	Logical operator insertion

3.1.1. AOR: Arithmetic Operator Replacement Inject fault by replacing each arithmetic operator (+, -, *, div, mod) with each other arithmetic operator that is syntactically legal.

3.1.2. ROR: Relational Operator Replacement Inject fault by replacing each relational operator (=, !=, >, <, >=, <=) with each other relational operator that is syntactically legal.

3.1.3. LOR: Logical Operator Replacement Inject fault by replacing each conditional operator (and, or) with each other conditional operator that is syntactically legal.

3.1.4. LOD: Logical Operator Delete Inject fault by deleting unary logical operator (not).

3.1.5. LOI: Logical Operator Insertion Inject fault by inserting unary logical operator (not).

3.2. Modifying BPEL Expressions

WS-BPEL specification [16] defined 5 expressions. These expressions may occur in both the basic and the structured activities.

3.2.1. Boolean expressions may occur in transition, join, while, and if activities.

3.2.2. Deadline expressions may occur in until expression of onAlarm and wait activities.

3.2.3. Duration expressions may occur in for expression of onAlarm and wait activities, and repeatEvery expression of onAlarm activities.

3.2.4. Unsigned Integer expressions may occur in startCounterValue, finalCounterValue, and branches element in forEach activities.

3.2.5 General expressions may occur in assign activities.

The mutation operators can apply to Boolean and Unsigned integer expressions. Thus, these expressions will be analyzed by our tool which we will explain in detail in the following part.

For original program P, each mutant of P is forms as result of a single modification of some expression in P. Therefore, each mutant of P differs from the original by only one mutates expression.

An example of a simple Boolean expression as figure 1 and 5 mutated expressions that are results of applying the ROR operator as figure 2 by replacing The more than operator (>) with each other relational operator (=, !=, <, >=).

```
<bpws:if name="If">
  <bpws:assign name="Assign1"validate="no">
    <bpws:copy>
      <bpws:from>
        <bpws:literal>
           A larger than B
        </bpws:literal>
      </bpws:from>
     <bpws:to part="outputMessage"
            variable="output" />
   </bpws:copy>
  </bpws:assign>
 <bpws:condition>
     <![CDATA[$input.APart > $input.BPart]]>
 </bpws:condition>
</bpws:if>
```

Figure 1: A simple Boolean expression

The Boolean expression can be applied by the AOR, ROR, LOR, LOD and LOI operator. The Unsigned Integer expressions can be applied by the AOR operator.

4. Mutation testing tool for BPEL

The mutation testing tool is web-based system that is implemented by using JSF (Java Server Faces) Framework. The BPEL server is Apache ODE is installed on Apache Tomcat. The BPEL mutation tool differs from other mutation tools, since BPEL is executed on BPEL server while others are executed on the local machine. Therefore our mutation testing tool uses a Web Service to deploy the original program and the mutated programs to BPEL server.

```
<bpws:condition>
<![CDATA[$input.APart = $input.BPart]]>
</bpws:condition>
..
<bpws:condition>
<![CDATA[$input.APart != $input.BPart]]>
</bpws:condition>
..
<bpws:condition>
<![CDATA[$input.APart < $input.BPart ]]>
</bpws:condition>
..
<bpws:condition>
<![CDATA[$input.APart >= $input.BPart]]>
</bpws:condition>
--
<bpws:condition>
<![CDATA[$input.APart <= $input.BPart]]>
</bpws:condition>
```

Figure 2: 5 mutated expressions

4.1. Architecture of mutation testing tool for BPEL

Architecture of mutation testing tool for BPEL is shown in figure 3 and the detail as follows.

4.1.1. Mutant generator accepts an original program (BPEL, WSDL and XSD document) which is zipped and uploaded by tester. Mutant generator unzips the original program and generates mutants by converting the BPEL document to DOM object. The Mutant generator finds the expressions and applies the mutation operators.

4.1.2. Test execution controller deploys both the original and the mutants to BPEL server by using DeploymentService Web Service which is provided by Apache ODE. Test cases are designed, created, and input to our tool by tester. The Test execution controller retrieves the test cases from test cases database which are encoded in an xml document that illustrates in figure 4. The test execution controller invokes Web Service by using the test cases, collects test results, and report test results. The test result is total mutant number, killed mutant number, percentage of killed mutant number in total mutant number, and execution time.

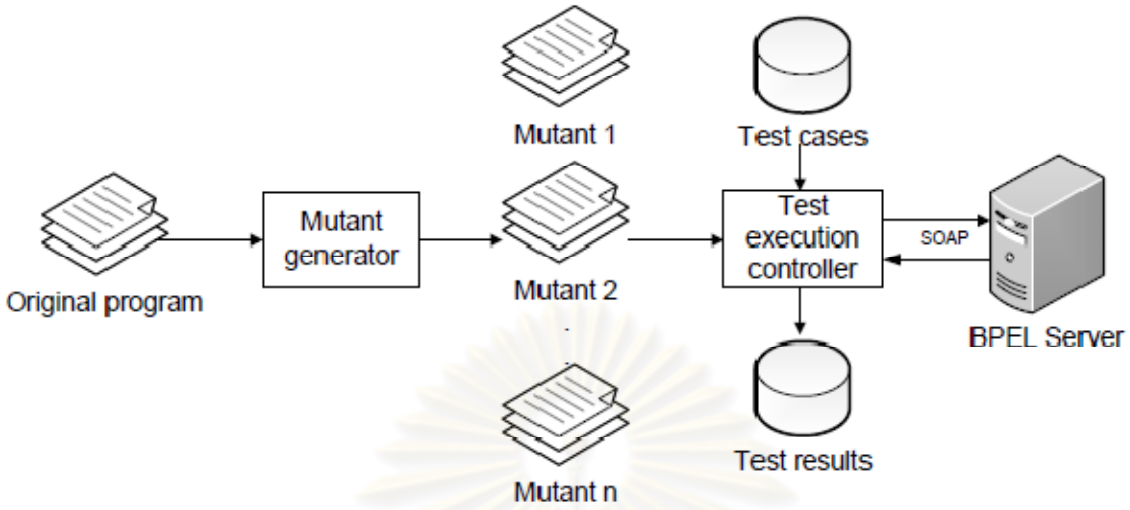


Figure 3: Architecture of mutation testing tool for BPEL

```
<void property="testCases">
        <void method="add">
                <object class="beans.TestCase">
                        <void property="expected">
                                <string>Equilateral</string>
                        </void>
                        <void property="id">
                                <string>1</string>
                        </void>
                        <void property="inputData">
                                <string>50,50,50</string>
                        </void>
                        <void property="pname">
                                <string>Original</string>
                        </void>
                </object>
        </void>
</void>
```

Figure 4: XML-encoded test cases

5. Example

For the demonstration of our approach we create 3 BPEL process: Triangle Process, Loan Approval Process, and Shopping Process.

5.1. Triangle Process

Triangle Process accepts tree integer parameter a, b, and c as input. They are taken to be sides of a triangle. The value of each side is between 1 and 100. The output of the process is the type of triangle: Scalene, Equilateral, Isosceles, Right-angled, and Impossible.

Test cases for the Triangle Process are designed as table 2. Mutant number of each Mutation operator is summarized as table 3. The total mutant's number is 224, which are killed 106 and the execution time is 403343 ms.

Table 2: Test cases for Triangle Process

ID	Input	Expected Result
1	50,50,1	Isosceles
2	50,50,2	Isosceles
3	50,50,50	Equilateral
4	50,50,99	Isosceles
5	50,50,100	Impossible
6	50,1,50	Isosceles
7	50,2,50	Isosceles
8	50,99,50	Isosceles
9	50,100,50	Impossible
10	1,50,50	Isosceles
11	2,50,50	Isosceles
12	99,50,50	Isosceles
13	100,50,50	Impossible

Table 3: Mutant number of Triangle Process

Operator	Mutants	Killed Mutants
AOR	60	12
ROR	120	63
LOR	19	7
LOD	0	0
LOI	24	24

5.2. Loan Approval Process

Loan Approval Process is a simple loan approval Web Service that accepts loan request from customers. Customers of the service send their loan requests, including personal information and amount being requested. The process result is Approved or Rejected.

Test cases for the Loan Approval Process are designed as table 4. Mutant number of each mutation operator is summarized as table 5. The total mutant's number is 43, which are killed 28 and the execution time is 113218 ms.

Table 4: Test cases for Loan Approval Process

ID	Input	Expected Result
1	500000,D,3	Approved
2	500000,D,4	Approved
3	500000,D,12	Approved
4	500000,D,35	Approved
5	500000,D,36	Approved
6	500000,A,12	Rejected
7	500000,B,12	Approved
8	500000,C,12	Approved
9	10000,D,12	Approved
10	10001,D,12	Approved
11	999999,D,12	Approved
12	1000000,D,12	Approved

Table 5: Mutant number of Loan Approval Process

Operator	Mutants	Killed Mutants
AOR	0	0
ROR	35	20
LOR	1	1
LOD	0	0
LOI	7	7

5.3. Shopping Process

Shopping process as is a shopping Web Service that accepts goods information and returns the lowest price shop name.

Test cases for the Shopping Process are designed as table 6. Mutant number of each mutation operator is summarizes as table 7. The total mutant's number is 24, which are killed 19 and the execution time is 178078 ms.

Table 6: Test cases for Shopping Process

ID	Input	Expected Result
1	A1,ProductA	Shop1
2	A2,ProductA	Shop2
3	A3,ProductA	Shop3
4	B1,ProductB	Shop1
5	B2,ProductB	Shop2
6	B3,ProductB	Shop3
7	C1,ProductC	Shop1
8	C2,ProductC	Shop3
9	C3,ProductC	Shop2

Table 7: Mutant number of Shopping Process

Operator	Mutants	Killed Mutants
AOR	0	0
ROR	20	15
LOR	0	0
LOD	0	0
LOI	4	4

6. Conclusion

We have proposed a technique to testing BPEL by using mutation testing. We have identified five mutation operators by following the selective approach that can be applied to the BPEL document by modifying the BPEL expressions. For the demonstration our approach, we have created a prototype of a mutation testing tool for BPEL, which is able to automatically generating mutants, deploys mutants to BPEL server, invokes service by using test cases, and reports test results. Finally, we have created 3 BPEL processes for the demonstration our tool. The testing results from our 3 BPEL processes indicate that our tool can detect inadequately of our test cases, although use only 5 mutation operators which the execution time is satisfactory.

7. Reference

[1] W3C. “Web Services Architecture”, W3C Working Group Note 11 February, http://www.w3.org/TR/ws-arch/, 2004.

[2] W3C. “Web Services Description Language (WSDL) 1.1”, W3C Note 15 March, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001

[3] Garcia-Fanjul, J. de la Riva, C. Tuya, J. “Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking”, Testing: Academic and Industrial Conference - Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings**,** 29-31 Aug. 2006, 127-130.

[4] Yuan, Y. Li, Z. and Sun, W. “A Graph-search Based Approach to BPEL4WS Test Generation”, Proceedings of the International Conference on Software Engineering Advances (ICSEA'06), 2006.

[5] Yan, J. Li, Z. Yuan, Y. Sun, W. and Zhang, J. “BPEL4WS Unit Testing: Test Case Generation Using a Concurrent Path Analysis Approach”, Software Reliability Engineering, 2006. ISSRE '06. 17th International Symposium on**,** 7-10 Nov. 2006, 75-84.

[6] Offutt, A.J., Voas, J. and Payne, J. “Mutation Operator for Ada”, Technical Report ISSE-TR-96-09, Information and Software Systems Engineering , George Mason University, Fairfax, Virgina, March, 1996.

[7] Alexander, R.T. Bieman, J.M. Ghosh, S. and Bixia Ji “Mutation of Java objects”, Software Reliability Engineering, 2002. ISSRE 2002. Proceedings. 13th International Symposium, November, 2002.

[8] Ferrari, F.C. Maldonado, J.C. and Rashid, A. “Mutation Testing for Aspect-Oriented Programs”, Software Testing, Verification, and Validation, 2008 1st International Conference, April, 2008.

[9] Sharpe, B. M. “SOA for the Business Developer: Concepts, BPEL, and SCA” , MC Press, 2007.

[10] Demillo, R. A. and Offutt , A. J. “Constraint-based Automatic Test Data Generation”, IEEE transaction on software engineering, 17(9), 1991, 900-910.

[11] Siblini, R. and Mansour, N. “Testing Web Services”, Computer Systems and Applications, The 3rd ACS/IEEE International ,2005.

[12] King, K.N. and Offutt, A.J. “A Fortran Language System for Mutation-Base Software Testing” Software-Practice and Experience, July, 1991, 21(7):685-718.

[13] Mathur, A.P. “Performance Effectiveness and Reliability Issue in Software Testing" In Proceeding of The Fifteenth Annual International Computer Software and Applications Conference, Tokyo, Japan, September, 1991, 604-605.

[14] Offutt, A.J., Rothermel, G. and Zapf, C. “An Experimental Evaluation of Selective Mutation” In Proceedings of The Fifteenth International Conference on Software Engineering, IEEE Computer Society Press, Baltimore, MD, May, 1993, 100-107.

[15] Offutt, A.J., Lee, A., Rothermel, G., Untch, R., and Zapf C. “An Experimental Determination of Sufficient Mutation Operators” ACM Transactions on Software Engineering Methodology, April, 1996, 5(2), 99-118.

[16] OASIS, “Web Services Business Process Execution Language Version 2.0”, OASIS Standard, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, 11 April, 2007.

ประวัติผู้เขียนวิทยานิพนธ์

นัฐพล ไทยสาครพันธ์ เกิดวันที่ 24 มกราคม พ.ศ.2525 ที่อำเภอสามพราน จังหวัดนครปฐม สำเร็จการศึกษาในหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาฟิสิกส์ ภาควิชาฟิสิกส์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร ในปี พ.ศ. 2547 และในปี พ.ศ.2549 เข้าศึกษาต่อในระดับปริญญาหมาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย