เครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง

นายวิทยา เหลืองหิรัญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

A Tool for Generating Test Cases from Javascript based on Statement Coverage Criteria

Mr. Witthaya Luanghirun

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 2015

หัวข้อวิทยานิพนธ์	เครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง
โดย	นายวิทยา เหลืองหิรัญ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร. สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร. วิวัฒน์ วัฒนาวุฒิ)

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

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

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

วิทยา เหลืองหิรัญ : เครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง (A Tool for Generating Test Cases from Javascript based on Statement Coverage Criteria) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร. ธาราทิพย์ สุวรรณศาสตร์, 78 หน้า.

ในการพัฒนาเว็บแอปพลิเคชันให้ทันสมัย จาวาสคริปต์เป็นภาษาที่มีความสำคัญสูงสุดในการเขียนโปรแกรมสำหรับการพัฒนา และ เฟรมเวิร์คการทดสอบอัตโนมัติมักจะถูกใช้ในการทดสอบเว็บแอปพลิเคชันระดับหน่วย อย่างไรก็ตามผู้พัฒนาใช้เวลาอย่างมากในการสร้างชุดคำสั่งทดสอบด้วยมือ ดังนั้นเครื่องมือสร้างชุดคำสั่งทดสอบอัตโนมัติ สามารถสนับสนุนผู้พัฒนาสร้างชุดคำสั่งอย่างมีประสิทธิภาพ ถึงกระนั้น เครื่องมือสำหรับสร้างชุดคำสั่งทดสอบโดยสร้างค่านำเข้าแบบสุ่มไม่สามารถรับประกันได้ว่าทุกทางเดินของซอร์สโค้ดจะถูกดำเนินการและใช้เวลาอย่างมีนัยในการทดสอบเพื่อให้ได้ความครอบคลุมสูง

งานวิจัยนี้นำเสนอเครื่องมือสำหรับการสร้างกรณีทดสอบจากฟังก์ชันจาวาสคริปต์ และดำเนินการกรณีทดสอบให้ครอบคลุมประโยคคำสั่ง เครื่องมือจะวิเคราะห์และแทรกโปรแกรมตรวจวัดความครอบคลุมในซอร์สโค้ดจาวาสคริปต์ เพื่อสร้างคอนโทรลโฟลว์กราฟและสร้างกรณีทดสอบ โดยสร้างข้อมูลที่เจาะจงบนทางเดิน และ เวกเตอร์นำเข้าจะทดสอบทางเดิน ประเมินความครอบคลุมดำเนินการกรณีทดสอบ และรายงานผลการทดสอบ หลังจากผู้วิจัยได้ทดสอบเครื่องมือกับไฟล์จาวาสคริปต์ตัวอย่างจำนวน 5 ไฟล์ เครื่องมือสามารถสร้างกรณีทดสอบ ดำเนินการกรณีทดสอบ และเครื่องมือสามารถทดสอบทุกทางเดินของไฟล์จาวาสคริปต์ตัวอย่างได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2558

5670378721 : MAJOR SOFTWARE ENGINEERING
KEYWORDS: SOFTWARE TESTING / AUTOMATED TESTING / JAVASCRIPT / INPUT VECTOR / PATH PREDICATE EXPRESSION

WITTHAYA LUANGHIRUN: A Tool for Generating Test Cases from Javascript based on Statement Coverage Criteria. ADVISOR: ASSOC. PROF. TARATIP SUWANNASART, Ph.D., 78 pp.

In modern web application development, JavaScript is the most important programming language for web application implementation and test framework automation is usually applied in unit web application testing. However, developers spend a lot of time to create test script manually. Thus, creating automated test script tool can support creating of test script efficiently. Nonetheless, a tool for generating test script by randomly creating test input cannot guarantee that all paths of the code is executed and it takes significant of time on testing to reach a high code coverage.

This paper proposes a tool for generating test cases from JavaScript function and executing test cases to cover all statements coverage criteria. The tool can analyze and instrument JavaScript code to generate a control flow graph and test cases by selecting data based on test paths and input vector to drive the paths, evaluate coverage, execute test cases, as well as display a test report. Finally, we test this tool with five JavaScript example files. The tool can generate test cases, execute the test cases, and it can test all paths in the JavaScript example files.

Department: Computer Engineering Student's Signature .....
Field of Study: Software Engineering Advisor's Signature .....
Academic Year: 2015

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลืออย่างดียิ่งจาก รองศาสตราจารย์ ดร. ธาราทิพย์ สุวรรณศาสตร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้คำแนะนำปรึกษา ให้แนวความคิด กำหนดกรอบเวลา และปรับปรุงแก้ไขข้อบกพร่องต่างๆ ด้วยความตั้งใจ จนวิทยานิพนธ์นี้สำเร็จไปได้ด้วยดี

ขอขอบพระคุณ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ ประธานกรรมการสอบ ผู้ช่วยศาสตราจารย์ ดร. อาทิตย์ ทองทักษ์ และผู้ช่วยศาสตราจารย์ ดร. ภัทรชัย ลลิตโรจน์วงศ์ คณะกรรมการสอบ ที่กรุณาตรวจสอบและชี้แนะข้อบกพร่องของวิทยานิพนธ์ฉบับนี้

ขอขอบคุณอาจารย์ทุกท่าน ที่ให้ความรู้ในด้านต่างๆ ที่ทำให้ข้าพเจ้ามีความรู้เพียงพอที่จะดำเนินงานวิจัยขึ้นมาได้

ขอบคุณน้องๆ พี่ๆ เพื่อนๆ ทุกท่านที่ให้คำแนะนำ และช่วยเหลือ และให้กำลังใจกันระหว่างทำวิจัย

ขอขอบคุณบิดา มารดา ของข้าพเจ้าและคุณณัชชา แม้นอ่วม ที่สนับสนุนในด้านต่างๆ เป็นกำลังใจ และเป็นแรงผลักดันให้ข้าพเจ้าดำเนินงานวิจัยจนสำเร็จ

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

กรณีทดสอบเป็นผลิตภัณฑ์งานที่สำคัญในการทดสอบซอฟต์แวร์ โดยเฉพาะการทดสอบซอฟต์แวร์ในระดับหน่วย ถ้ากรณีทดสอบที่ถูกสร้างขึ้นไม่พบข้อบกพร่อง (Defect) ที่แฝงในซอฟต์แวร์ระดับหน่วยแล้ว การทดสอบซอฟต์แวร์ในระดับบูรณาการ การทดสอบระบบ รวมไปถึงการใช้งานซอฟต์แวร์จริง อาจเกิดปัญหาจากข้อบกพร่องที่แฝงนั้นได้

เงื่อนไขคำสั่งจะปรากฏการพัฒนาซอฟต์แวร์อยู่บ่อยครั้ง เนื่องจากเงื่อนไขคำสั่งที่จะใช้ตรวจสอบค่านำเข้าและตัดสินใจกำหนดทางเดินของคำสั่ง ให้คำสั่งดำเนินการตามทางเดินที่เงื่อนไขกำหนดเฉพาะได้ ถ้าเงื่อนไขคำสั่งไม่สามารถตัดสินใจกำหนดทางเดินของคำสั่งจากค่านำเข้าได้ จะทำให้คำสั่งภายใต้เงื่อนไขไม่สามารถถูกดำเนินการได้และเกิดข้อบกพร่องขึ้น

จากโครงงานมหาบัณฑิต “เครื่องมือสร้างมอดูลทดสอบสำหรับจาวาสคริปต์บนเงื่อนไขความครอบคลุมคำสั่ง”[1] ได้นำเสนอเครื่องมือสร้างมอดูลทดสอบไฟล์จาวาสคริปต์โดยสร้างข้อมูลทดสอบด้วยวิธีการสุ่มค่า ทำให้ทางเดินคำสั่งบางทางเดินไม่ได้ถูกทดสอบ จึงทำให้การทดสอบมีความครอบคลุมประโยคคำสั่งน้อย งานวิจัยที่นำเสนอนี้จึงได้ปรับปรุงวิธีการสร้างข้อมูลทดสอบโดยการสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง ให้ได้ข้อมูลทดสอบเฉพาะสามารถดำเนินการบนทางเดินคำสั่งมากขึ้น เพื่อทำให้การทดสอบไฟล์จาวาสคริปต์มีความครอบคลุมประโยคคำสั่งมากขึ้น

ดังนั้นงานวิจัยนี้จึงนำเสนอเครื่องมือที่ใช้สำหรับสร้างกรณีทดสอบจากซอฟต์แวร์ที่ถูกพัฒนาด้วยภาษาจาวาสคริปต์ โดยสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง โดยเครื่องมือนี้จะวิเคราะห์ซอร์สโค้ดเพื่อสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง ดำเนินการทดสอบ และรายงานการทดสอบ

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อพัฒนาเครื่องมือสร้างกรณีทดสอบสำหรับจาวาสคริปต์โดยสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง

1.3 ขอบเขตของงานวิจัย

1. พัฒนาเครื่องมือสร้างกรณีทดสอบสำหรับจาวาสคริปต์ให้เป็นไปตามความครอบคลุมประโยคคำสั่งโดยเลือกทางเดินทดสอบครอบคลุมประโยคคำสั่งจากกราฟการไหลของความควบคุม ข้อมูลทดสอบที่ถูกสร้างมาจากการสุ่มค่าหรือเลือกค่าจากเงื่อนไขคำสั่ง ผลลัพธ์จะได้มอดูลทดสอบที่เป็นไปตามกรณีทดสอบที่ถูกสร้างขึ้น
2. เครื่องมือจะทดสอบโค้ดภายในฟังก์ชันจาวาสคริปต์เท่านั้น ไม่รองรับทดสอบโค้ดนอกไฟล์จาวาสคริปต์ และรองรับพารามิเตอร์ที่เป็นตัวเลข สตริง และบูลีน เท่านั้น
3. เครื่องมือจะกำหนดให้แต่ละคำสั่งที่มีเครื่องหมายอัฒภาค (“ ; ”) สิ้นสุด และมีความยาวหนึ่งบรรทัด เท่ากับหนึ่งคำสั่ง
4. เครื่องมือไม่รองรับคำสั่งที่เกี่ยวข้องกับการจัดการเอกสารอ็อบเจกต์โมเดล (Document Object Model)
5. เครื่องมือนี้เป็นวินโดว์แอปพลิเคชันซึ่งพัฒนาด้วยภาษาจาวา
6. เครื่องมือที่พัฒนาเสร็จแล้วจะถูกนำไปทดสอบกับไฟล์ตัวอย่างจาวาสคริปต์จำนวน 5 ไฟล์ โดยที่หนึ่งไฟล์จาวาสคริปต์ตัวอย่างจะมีหนึ่งฟังก์ชันเท่านั้น

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาโครงสร้างภาษาจาวาสคริปต์
2. ศึกษาเครื่องมือที่เกี่ยวข้อง
3. ศึกษางานวิจัยที่เกี่ยวข้อง
4. กำหนดคุณลักษณะและขอบเขตความสามารถของเครื่องมือ
5. พัฒนาเครื่องมือ
6. ทดสอบการทำงานของเครื่องมือ
7. สรุปผลการวิจัยและประเมินการทดสอบ
8. จัดทำบทความตีพิมพ์
9. จัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

เครื่องมือต้นแบบจากงานวิจัยนี้จะช่วยเพิ่มความครอบคลุมประโยคคำสั่งของการทดสอบซอฟต์แวร์ระดับหน่วย และช่วยให้การทดสอบซอฟต์แวร์มีโอกาสพบข้อบกพร่องมากขึ้น

1.6 ลำดับขั้นตอนในการนำเสนองานวิจัย

วิทยานิพนธ์ฉบับนี้แบ่งเนื้อหาออกเป็น 5 บท ดังต่อไปนี้ บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา รวมทั้งวัตถุประสงค์ของงานวิจัย ขอบเขตของงานวิจัย ขั้นตอนและวิธีการดำเนินการวิจัย และประโยชน์ที่คาดว่าจะได้รับ บทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้องรวมถึงงานวิจัยที่เกี่ยวข้อง บทที่ 3 นำเสนอการวิเคราะห์และออกแบบเครื่องมือ รวมทั้งภาพรวมการทำงานของเครื่องมือ บทที่ 4 การพัฒนาเครื่องมือ บทที่ 5 การทดสอบและสภาพแวดล้อมการทดสอบ และบทที่ 6 เป็นบทสรุปผลการวิจัยและข้อเสนอแนะ

1.7 ผลงานตีพิมพ์งานวิจัย

งานวิจัยนี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการดังต่อไปนี้

1. บทความวิชาการเรื่อง “การสร้างกรณีทดสอบจากจาวาสคริปตบนเงื่อนไขความครอบคลุมประโยคคำสั่ง” โดย วิทยา เหลืองหิรัญ และธาราทิพย สุวรรณศาสตร ได้รับคัดเลือกและตีพิมพ์ในการประชุมวิชาการ “19th International Computer Science and Engineering Conference (ICSEC 2015)” ที่ถูกจัดขึ้นในระหว่างวันที่ 23-26 พฤศจิกายน 2558 ณ โรงแรมดวงตะวัน จังหวัดเชียงใหม่ ประเทศไทย

2. บทความวิชาการเรื่อง “Test Cases Generation Tool for JavaScript Based on Statement Coverage Criteria” โดย Witthaya Luanghirun และ Taratip Suwannasart ได้รับคัดเลือกและตีพิมพ์ในการประชุมวิชาการ “24th International MultiConference of Engineers and Computer Scientists 2016 (IMECS 2016)” ที่ถูกจัดขึ้นในระหว่างวันที่ 16-18 มีนาคม 2559 ณ โรงแรม Royal Garden เขตเกาลูน ประเทศฮ่องกง

บทที่2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 จาวาสคริปต์ (JavaScript)

จาวาสคริปต์เป็นภาษาสคริปต์ (Scripting Language) [2] มีจุดประสงค์เพื่อเพิ่มประสิทธิภาพและความสามารถในการพัฒนาส่วนต่อประสานผู้ใช้งาน จาวาสคริปต์จะถูกประมวลผลโดยเว็บเบราว์เซอร์ และสามารถเข้าถึงอ็อบเจกต์ต่างๆของเว็บเบราว์เซอร์ได้เช่นหน้าต่าง เมนู กล่องข้อความ ลิงค์ และคุกกี้ เป็นต้น จาวาสคริปต์สามารถสั่งการอ็อบเจกต์เหล่านั้นได้ เช่นสั่งให้โฟกัสที่กล่องข้อความ ซอร์สโค้ดภาษาจาวาสคริปต์จะปรากฏอยู่ในรหัส HTML และสามารถนาเข้าซอร์สโค้ดภาษาจาวาสคริปต์จากไฟล์อื่นมาได้ ข้อดีของจาวาสคริปต์คือสามารถดำเนินการได้โดยไม่ต้องคอมไพล์ (Compile) โดยใช้วิธีแปลความ (Interpret) และดำเนินการ ทำให้จาวาสคริปต์สามารถทำงานได้บนเว็บเบราว์เซอร์ทุกชนิด

นอกจากนี้จาวาสคริปต์เป็นภาษาที่มีการพัฒนาไปอย่างรวดเร็ว เพราะภาษานี้มีวัตถุประสงค์ในการใช้งานที่หลากหลาย และถูกกำหนดขึ้นมาตรฐานของเทคโนโลยีเวิลด์ไวด์เว็บ (World Wide Web) ทำให้ทุกการเปลี่ยนแปลงในภาษานี้จะมีผลกระทบสำคัญในการพัฒนาเว็บไซต์เป็นวงกว้าง [3]

2.1.2 การทดสอบเพื่อความครอบคลุมประโยคคำสั่ง (Statement coverage testing)

การทดสอบเพื่อความครอบคลุมประโยคคำสั่ง [4] เป็นส่วนหนึ่งของเทคนิคการทดสอบซอฟต์แวร์แบบไวท์บอกซ์ (White box testing technique) ซึ่งเป็นเทคนิคสำหรับทดสอบซอฟต์แวร์โดยสนใจที่โครงสร้างของซอฟต์แวร์ ซึ่งสามารถใช้กราฟการไหลของการควบคุมในการวิเคราะห์โครงสร้างของซอฟต์แวร์นั้น

แนวคิดของการทดสอบเพื่อความครอบคลุมประโยคคำสั่งคือ ทุกประโยคคำสั่งที่อยู่ในโปรแกรมต้องถูกดำเนินการอย่างน้อยหนึ่งครั้งโดยไม่สนใจว่าคำสั่งนั้นจะทำอะไร การทดสอบเพื่อความครอบคลุมประโยคคำสั่งนี้จะต้องมีการแทรกโปรแกรมตรวจวัดความครอบคลุมเข้าไปในซอร์สโค้ดที่จะถูกทดสอบ (Instrument code) เพื่อระบุว่าประโยคคำสั่งใดบ้างที่ถูกดำเนินการไปแล้ว รูปที่ 2-1

แสดงซอร์สโค้ดที่ถูกแทรกด้วยโปรแกรมตรวจวัดความครอบคลุมที่บรรทัดที่ 1, 3, 5, 9 , 13, 15 และ 17

```
1  coverage[1]++;
2  Encoder = {EncodeType: "entity", isEmpty: (function (val) {
3    coverage[7]++;
4    if (val) {
5      coverage[8]++;
6      return ((val === null) || (val.length == 0) || /^\s+$/.test(val));
7    }
8    else {
9      coverage[10]++;
10     return true;
11   }
12 }), HTML2Numerical: (function (s) {
13   coverage[15]++;
14   var arr1 = new Array(" ", "&iexcl;", "&cent;", "&pound;", "&curren
15   coverage[16]++;
16   var arr2 = new Array(" ", "&#161;", "&#162;", "&#163;", "&#164;",
17   coverage[17]++;
18   return this.swapArrayVals(s, arr1, arr2);
```

รูปที่ 2-1 ซอร์สโค้ดที่ถูกแทรกด้วยโปรแกรมตรวจวัดความครอบคลุม

2.1.3 **เวกเตอร์นำเข้า (Input Vector)**

เวกเตอร์นำเข้า [5] คือ ข้อมูลนำเข้าทั้งหมดใช้ในการดำเนินการฟังก์ชันของโปรแกรม โดยข้อมูลนำเข้าสามารถมีได้หลายรูปแบบ เช่น ตัวแปรโกลบอล ค่าคงที่ ไฟล์ และการเชื่อมโยงเครือข่าย เป็นต้น ในงานวิจัยนี้จะใช้พารามิเตอร์ของฟังก์ชันจาวาสคริปต์ ที่มีชนิดข้อมูล (Data type) เป็น ตัวเลข สตริง และบูลีน เป็นเวกเตอร์นำเข้า จากรูปที่ 2-2 แสดงตัวอย่างฟังก์ชันจาวาสคริปต์ใช้พารามิเตอร์ซึ่งเป็นเวกเตอร์นำเข้าในรับค่าตัวแปร a , b แล้วคืนค่าผลลัพธ์จากฟังก์ชันเป็นค่าตัวแปร a คูณกับค่าตัวแปร b

```
1  function myFunction(a, b) {
2      return a * b;
3  }
```

รูปที่ 2-2 ตัวอย่างซอร์สโค้ดของฟังก์ชันในจาวาสคริปต์

2.1.4 **เพรดิเคต (Predicate)**

เพรดิเคต [6] คือ เงื่อนไขที่ต้องดำเนินการตรวจสอบตรงโหนดตัดสินใจ โดยผลลัพธ์ที่ได้จากเพรดิเคตจะมีค่าจริง (True) หรือ เท็จ (False) เท่านั้น จึงทำให้เพรดิเคตสามารถนำไปใช้งานประยุกต์หลากหลาย และช่วยในการตีความทางคณิตศาสตร์และตรรกศาสตร์ รวมไปถึงการเขียนโปรแกรมคอมพิวเตอร์ที่จำเป็นในการใช้เพรดิเคตช่วยการติดสินใจเพื่อโปรแกรมสามารถทำงานในกิจกรรมต่างๆได้อย่างถูกต้อง

เพรดิเคต สามารถจำแนกได้เป็น 2 ประเภท ได้แก่ เพรดิเคตเดี่ยว และเพรดิเคตประกอบ โดยที่เพรดิเคตเดี่ยว คือ เพรดิเคตที่มีตัวแปรเป็นตัวแปรบูลีน (Boolean variable) หรือ เพรดิเคตที่มีนิพจน์เชิงสัมพันธ์ (Relational expression) ซึ่งนิพจน์เชิงสัมพันธ์นี้เกิดจาก ค่าตัวแปรหรือค่าคงที่สองค่า มาเปรียบเทียบกัน ด้วย ตัวดำเนินการเชิงสัมพันธ์ (Relational operator) หนึ่งตัวจากทั้งหมด 6 ตัวดำเนินการ ได้แก่ $=, \neq, <, >, \geq, \leq$ เพรดิเคตประกอบ จะประกอบ อย่างน้อยหนึ่งตัวดำเนินการบูลีนแบบทวิภาค (Binary boolean operator) และเพรดิเคตเดี่ยวสองเพรดิเคต ทำหน้าที่เป็นตัวถูกดำเนินการ ตัวดำเนินการบูลีนแบบทวิภาคที่ใช้ในงานวิจัยนี้ เป็น ตัวดำเนินการ AND (“&&”) และ OR (“||”)

จากรูปที่ 2-3 เป็นตัวอย่างเงื่อนไขคำสั่งในการตัดเกรดจากคะแนน โดยคำสั่งเงื่อนไขจะแสดงเพรดิเคต-ประกอบในการตัดสินใจค่าของตัวแปรตัวเลข n เพื่อให้ได้ผลลัพธ์ค่าตัวแปรสตริง grade ตามแต่ละเกรดที่ได้ระบุไว้

```
if ((n >= 80) && (n <= 100)){grade = "A";
} else if ((n >= 60) && (n < 80)){grade = "B";
} else if ((n >= 50) && (n < 60)){grade = "C";
} else if ((n >= 40) && (n < 50)){grade = "D";
} else if ((n >= 0) && (n < 40)){grade = "F";
} else { grade= "no grade"; }
```

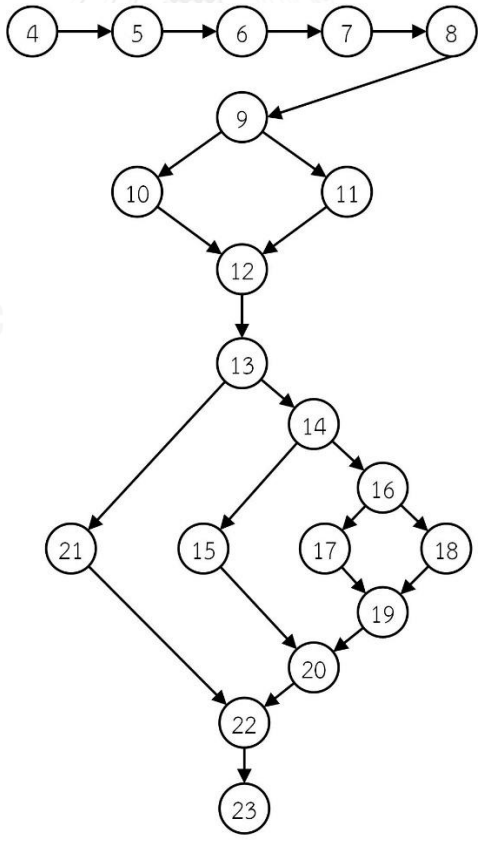
รูปที่ 2-3 ตัวอย่างเพรดิเคตประกอบในซอร์สโค้ดคำสั่งเงื่อนไข (if)

2.1.5 **กราฟการไหลของการควบคุม (Control flow graph)**

กราฟการไหลของการควบคุม [7] เป็น กราฟระบุทิศทาง (Directed graph) ใช้แสดงโครงสร้างของโปรแกรม บอกถึงลำดับการประมวลผลของคำสั่งในโปรแกรม รู้จักกันในอีกชื่อหนึ่งว่ากราฟกระแสการไหล (Flow graph) หรือ โปรแกรมกราฟ (Program graph) โดยกราฟการไหลของการควบคุมจะประกอบด้วย โหนดแทนประโยคคำสั่ง (Statement) และเส้นทิศทางแทนกระแสการไหลของการควบคุมการดำเนินการของโปรแกรม โดยตัวอย่างของโปรแกรมและกราฟการไหลของการควบคุมของโปรแกรมสามารถแสดง ได้ดังรูปที่ 2-4 และรูปที่ 2-5

```
1       Program triangle2 'Structured programming version of
        simpler specification'
2       Dim a,b,c AsInteger
3       Dim IsAtriangle As Boolean
4       Output ('Enter 3 Integers which are sides of a triangle")
5       Input (a,b,c)
6       Output ("Side A is ",a)
7       Output ("Side B is ",b)
8       Output ("Side C is ",c)
9       If(a < b + c) AND (b < a + c) AND (c < a + b)
10             Then IsATriangle = True
11      Else IsATriangle = False
12      EndIf
13      If IsTriangle
14             Then IF (a = b) AND (b = c)
15                    Then Output ("Equilateral")
16                    Else If (a <> b) AND (a <> c) AND (b <> c)
17                           Then Output ("Scalene")
18                           Else Output ("Isosceles")
19                         EndIf
20                    EndIf
21             Else Output ("Not a Triangle")
22      EndIf
23      End triangle2
```

รูปที่ 2-4 ตัวอย่างโปรแกรมสามเหลี่ยม [8]



รูปที่ 2-5 ตัวอย่างกราฟการไหลของการควบคุมของโปรแกรมสามเหลี่ยม [8]

2.1.6 **ทางเดินที่เป็นไปไม่ได้ (Infeasible path)**

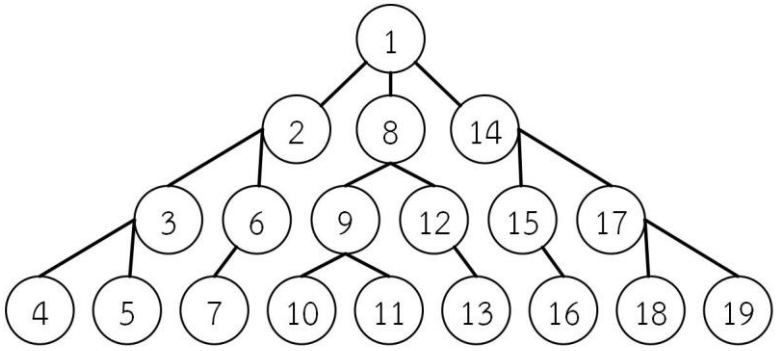
ทางเดินที่เป็นไปไม่ได้ [5] คือ ทางเดินคำสั่งที่ไม่สามารถหาค่านำเข้ามาดำเนินการทุกคำสั่งในทางเดินของคำสั่งนั้น เนื่องจากเงื่อนไขบนทางเดินคำสั่งไม่สามารถเข้าถึงได้ จากรูปที่ 2-6 ได้แสดงซอร์สโค้ดคำสั่งเงื่อนไข ที่มีเพรดิเคตประกอบ a > b และ a < b ซึ่งเงื่อนไขนี้ไม่สามารถหาค่านำเข้า a และ b ดำเนินการคำสั่งเงื่อนไขตัดสินใจเป็นจริงได้

```
if (a > b && a < b) {
    return a;
}
```

รูปที่ 2-6 ตัวอย่างซอร์สโค้ดคำสั่งเงื่อนไขที่ดำเนินการคำสั่งไม่ได้

2.1.7 **การค้นหาแนวลึกก่อน (Depth first search)**

การค้นหาแนวลึกก่อน [9] เป็นขั้นตอนวิธีที่ใช้ในการเดินทางเข้าไปในกราฟ เริ่มต้นจากโหนดราก (Root node) ที่อยู่บนสุด แล้วเดินลงมาให้ลึกที่สุด เมื่อถึงโหนดล่างสุด (Terminal node) ให้ย้อนขึ้นมาที่จุดสูงสุดของกิ่งเดียวกันที่มีกิ่งแยกและยังไม่ได้เดินผ่าน แล้วเริ่มเดินลงจนถึงโหนดลึกสุดอีก ทำเช่นนี้สลับไปเรื่อยจนพบโหนดที่ต้องการหาครบทุกโหนดแล้ว ดังรูปที่ 2-7 การค้นหาแบบลึกก่อนจะมีลำดับการเดิน ตามโหนดดังตัวเลขที่กำกับไว้ในแต่ละโหนด



รูปที่ 2-7 ลำดับการเดินทางบนโหนดของการค้นหาแนวลึกบนโครงสร้างต้นไม้

2.1.8 **เรกูลาร์เอกซ์เพรสชัน (Regular expression)**

เรกูลาร์เอกซ์เพรสชัน [10] คือ รูปแบบอักขระที่ใช้ในการค้นหาข้อความจากสตริงที่ต้องการ โดยใช้รูปแบบ (Pattern) ของสตริงในการค้นหา รูปแบบของสตริงนั้นจะถูกเขียนขึ้นตามหลักไวยากรณ์ของเรกูลาร์เอกซ์เพรสชัน ดังตารางที่ 2-1

ในงานวิจัยนี้จะนำเรกูลาร์เอกซ์เพรสชันและแอนท์เลอร์ ใช้ในการวิเคราะห์โครงสร้างคำสั่งในไฟล์จาวาสคริปต์ที่ถูกนำเข้ามาในเครื่องมือ

ตารางที่ 2-1 อักขระพิเศษในเรกูลาร์เอกซ์เพรสชันพื้นฐาน

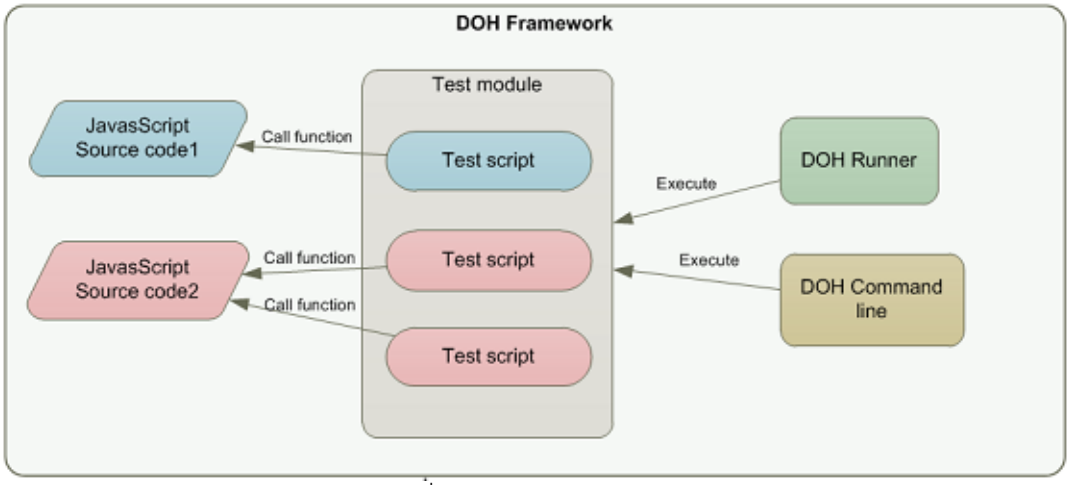
อักขระ	ความหมาย
\	เป็นการบอกว่า อักขระตัวต่อไปไม่เป็นอักขระพิเศษ และให้แปลความหมายตามที่เขียน เช่น รูปแบบ /a*/ ที่ใช้อักขระพิเศษ '*' จะตรงกับข้อความที่มีหรือไม่มี a ก็ได้ ในทางตรงกันข้าม, รูปแบบ /a*/ จะทำให้ '*' ไม่เป็นอักขระพิเศษ และจะได้สตริงที่มี 'a*' อยู่ข้างใน
^	ค้นหาข้อความในสตริง ด้วยอักขระที่อยู่หลัง '^' เพื่อหาข้อความที่ขึ้นต้นด้วยอักขระนั้น เช่น /^A/ จะไม่ตรงกับ 'A' ใน "an A" แต่จะตรงกับ 'A' ใน "An E"
$	ค้นหาข้อความในสตริง ด้วยอักขระที่อยู่หน้า '$' เพื่อหาข้อความที่ลงท้ายด้วยอักขระนั้น เช่น /t$/ จะไม่ตรงกับ 't' ใน "eater" แต่จะตรงกับ "eat"
*	ค้นหาข้อความในสตริง ด้วยอักขระที่อยู่หน้า '*' เพื่อหาข้อความที่มีหรือไม่มีอักขระนั้น เช่น /bo*/ จะตรงกับ 'boooo' ใน "A ghost booooed"
+	ค้นหาข้อความในสตริง ด้วยอักขระที่อยู่หน้า '+' เพื่อหาข้อความที่มีอักขระนั้นอย่างน้อยหนึ่งตัวขึ้นไป เช่น /a+/ จะตรงกับ 'a' ใน "candy" และหลายๆ a ใน "caaaaaaandy"
?	ค้นหาข้อความในสตริง ด้วยอักขระที่อยู่หน้า '?' เพื่อหาข้อความที่ไม่มีอักขระนั้นหรือมีอักขระนั้นอยู่หนึ่งตัว เช่น /e?le?/ จะตรงกับ 'el' ใน "angel" และ 'le' ใน "angle" รวมทั้ง 'l' ใน "oslo"
.	(จุดทศนิยม) ค้นหาข้อความในสตริง ที่มีอักขระตัวใดก็ได้ เช่น /.n/ จะตรงกับ 'an' และ 'on' ใน "nay, an apple is on the tree" แต่ไม่ตรงกับ 'nay'
(x)	ค้นหาข้อความในสตริงด้วย 'x' และจำข้อความที่พบ เช่น อักขระ '(foo)' และ '(bar)' ที่อยู่ใน regular expression รูปแบบ /(foo) (bar) \1 \2/ จะตรงกับสตริง "foo bar foo bar"

ตารางที่ 2-2 อักขระพิเศษในเรกูลาร์เอกซ์เพรสชันพื้นฐาน (ต่อ)

อักขระ	ความหมาย
{n,m}	ค้นหาอักขระอย่างน้อย n ตัว เกิดขึ้นไม่เกิน m ครั้ง โดยที่ n และ m เป็นเลขจำนวนเต็มบวก เช่น /a {1,3} / จะไม่พบใน "cndy" พบ 'a' หนึ่งตัวใน "candy," พบ a สองตัวแรกใน "caandy เป็นต้น
[xyz]	เป็นกลุ่มอักขระ ค้นหาอักขระตัวใดตัวหนึ่งที่ตรงกับในวงเล็บ เช่น [abcd] จะตรงกับ 'b' ใน "brisket" และ 'c' ใน 'city'

2.1.9 **ดีโอเอช (D.O.H)**

ดีโอเอช [1, 11] เป็นเฟรมเวิร์คสำหรับทดสอบซอฟต์แวร์ระดับหน่วยสำหรับจาวาสคริปต์ที่ถูกพัฒนาตามโดโจเฟรมเวิร์ค ดีโอเอชรองรับการทำงานบนเว็บเบราว์เซอร์ การทำงานโดยไม่ใช้เว็บเบราว์เซอร์ การทำงานแบบไม่ประสานเวลา (Asynchronous) การทำงานกับส่วนต่อประสานผู้ใช้ รวมถึงการใช้งานเพื่อทดสอบประสิทธิภาพซอฟต์แวร์



รูปที่ 2-8 ดีโอเอชเฟรมเวิร์ค

จากรูปที่ 2-8 แสดงถึงความสัมพันธ์ของซอร์สโค้ดสคริปต์ทดสอบ มอดูลทดสอบ และส่วนดำเนินการทดสอบ ดีโอเอชรันเนอร์ (DOH runner) และดีโอเอชคอมมานด์ไลน์ (DOH Command line) ซอร์สโค้ดจาวาสคริปต์ทางด้านซ้ายจะถูกเรียกใช้งานโดยสคริปต์ทดสอบ ซึ่งสคริปต์ทดสอบนั้นจะถูกบรรจุรวมกันอยู่ในมอดูลทดสอบ ส่วนดำเนินการทดสอบ จะดำเนินการมอดูลทดสอบเพื่อทำการทดสอบจาวาสคริปต์

ในการใช้งานดีโอเอชเฟรมเวิร์คเพื่อทดสอบฟังก์ชัน ผู้ใช้งานต้องเตรียมองค์ประกอบ สำคัญสำหรับการทดสอบ 2 ส่วนดังนี้

1) ซอร์สโค้ดของฟังก์ชันที่จะถูกทดสอบ ดังรูปที่ 2-9 บรรทัดที่ 6-8 คือฟังก์ชัน “demo.doh.toBeTestedFunction.getLength” ที่จะถูกทดสอบ

```
1 dojo.provide("demo.doh.toBeTestedFunction");
2
3
4 //This file contains a to be tested javascript function
5
6 demo.doh.toBeTestedFunction.getLength = function(string) {
7     var length = string.length;
8 };
```

รูปที่ 2-9 ตัวอย่างซอร์สโค้ดของฟังก์ชันที่จะถูกทดสอบ

2) สคริปต์ทดสอบของฟังก์ชัน ดังรูปที่ 2-10 บรรทัดที่ 4 มีการนำเข้าไฟล์จาวาสคริปต์ของฟังก์ชันที่จะถูกทดสอบ บรรทัดที่ 12 แสดงถึงการสั่งดำเนินการฟังก์ชันพร้อมทั้งใส่ค่าข้อมูลทดสอบให้เป็น “This is test input”

```
1  dojo.provide("demo.doh.tests.functions.sampleFunctions");
2
3  //Import in the code being tested.
4  dojo.require("demo.doh.toBeTestedFunction");
5
6  doh.register("demo.doh.tests.functions.sampleFunctions", [
7
8      function executeTest(){
9          //  description:
10         //  Call the function getLength in file toBeTestedFunction
11         //  Put  "This is test input" as parameter
12         demo.doh.toBeTestedFunction.getLength("This is test input");
13     }
14 ]);
```

รูปที่ 2-10 ตัวอย่างสคริปต์ทดสอบในรูปแบบดีโอเอชเฟรมเวิร์ค

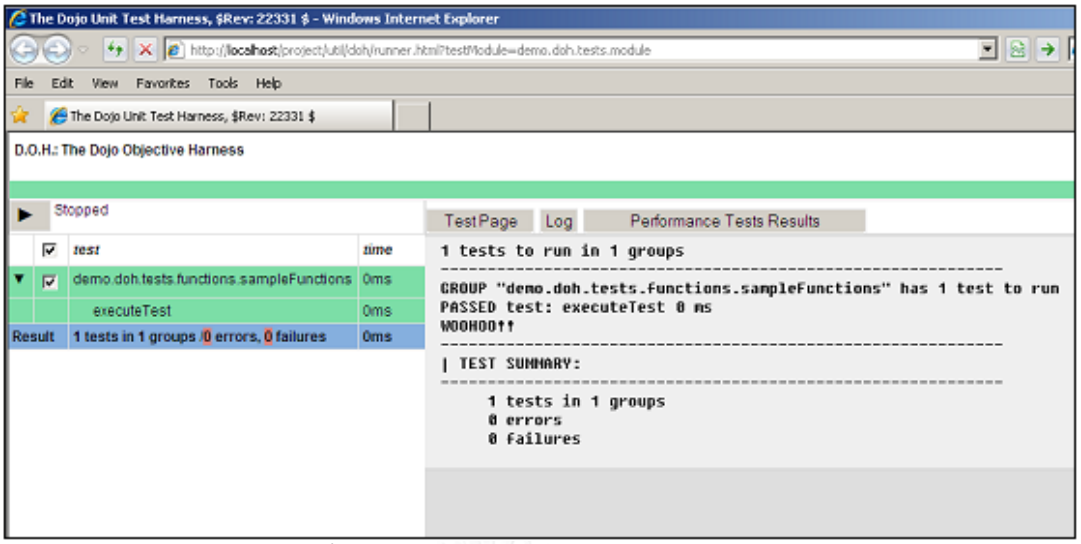
การดำเนินการทดสอบนั้น ดีโอเอชจะรับคำสั่งสองวิธีคือวิธีเรียกผ่านดีโอเอชรันเนอร์ และผ่านดีโอเอช คอมมานด์ไลน์ ผู้ใช้งานต้องส่งค่าพารามิเตอร์เป็นมอดูลทดสอบ มีรายละเอียดดังนี้

1) ดำเนินการทดสอบผ่านดีโอเอชรันเนอร์

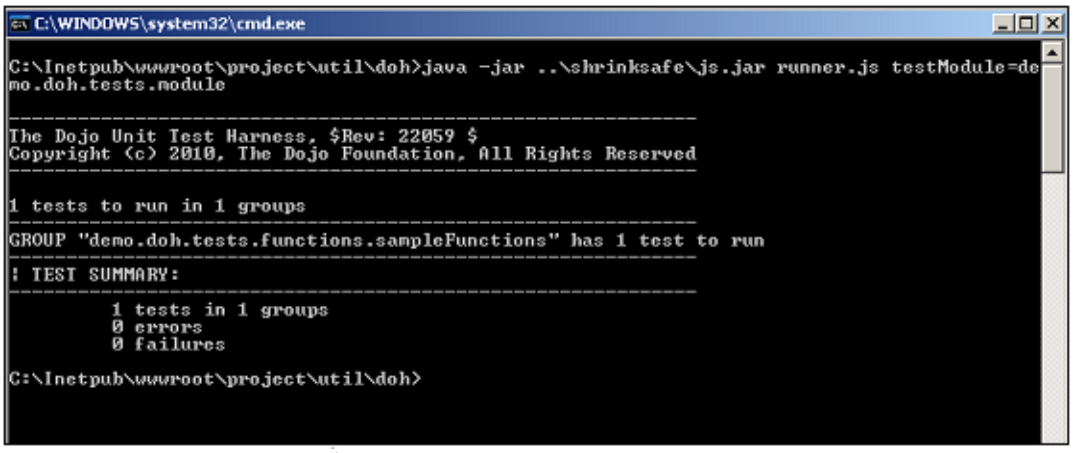
ผู้ใช้งานต้องเปิด URL .../util/doh/runner.html?t estModule=<test module> ในเว็บเบราว์เซอร์ ดังรูปที่ 2-11 ดีโอเอชรันเนอร์จะดำเนินการทดสอบและแสดงผลการทดสอบ โดยทางด้านซ้ายคือมอดูลทดสอบและสคริปต์ทดสอบภายในมอดูลนั้นและทางขวามือคือผลการทดสอบ

2) ดำเนินการทดสอบผ่านคอมมานด์ไลน์

ใช้คำสั่ง java -jar ..\shrinksafe\js.jar runner.js testModule=<test module> ในคอมมานด์ไลน์ ซึ่งผลการทดสอบที่ได้จะอยู่ในรูปแบบเดียวกับผลการทดสอบจากดีโอเอชรันเนอร์ ดังรูปที่ 2-12



รูปที่ 2-11 ผลการทดสอบผ่านดีโอเอชรันเนอร์



รูปที่ 2-12 ผลการทดสอบผ่านคอมมานด์ไลน์

2.1.10 **แอปทานา (Aptana)**

แอปทานา [12] เป็นเครื่องมือโอเพนซอร์สในการพัฒนาเว็บแอปพลิเคชัน ซึ่งมีไลบรารี (Library) ช่วยสนับสนุนในการเขียนซอร์สโค้ดสำหรับพัฒนาเว็บแอปพลิเคชันต่างๆได้แก่ ภาษาเอชทีเอ็มแอล (HTML) ภาษาซีเอสเอส (CSS) และภาษาจาวาสคริปต์ ซึ่งในงานวิจัยนี้จึงได้นำไลบรารีบางส่วนของเครื่องมือนี้มาปรับแต่ง มาใช้ในการแจงส่วน (Parse) ไฟล์จาวาสคริปต์ให้อยู่ในรูปแบบของไฟล์เอกซ์เอ็มแอล (XML) เพื่อใช้ในการวิเคราะห์โครงสร้างไฟล์จาวาสคริปต์

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 โครงงานมหาบัณฑิต“เครื่องมือสร้างมอดูลทดสอบสำหรับจาวาสคริปต์บนเงื่อนไขความครอบคลุมคำสั่ง” [1]

โครงงานมหาบัณฑิตนี้นำเสนอเครื่องมือสร้างมอดูลทดสอบไฟล์จาวาสคริปต์โดยสร้างข้อมูลทดสอบด้วยวิธีการสุ่มค่าตามประเภทของพารามิเตอร์ ซึ่งมีการใช้เวลาสุ่มข้อมูลทดสอบค่อนข้างนาน และข้อมูลทดสอบที่ถูกสุ่มไม่สามารถดำเนินการทดสอบในบางเงื่อนไขคำสั่งได้ ทำให้ทางเดินคำสั่งบางทางเดินไม่ได้ถูกทดสอบ จึงทำให้การทดสอบมีความครอบคลุมประโยคคำสั่งน้อย งานวิจัยที่นำเสนอนี้จึงได้ปรับปรุงวิธีการสร้างข้อมูลทดสอบโดยการสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง เพื่อให้ได้ข้อมูลทดสอบเฉพาะสามารถดำเนินการบนทางเดินคำสั่งมากขึ้น เพื่อทำให้การทดสอบไฟล์จาวาสคริปต์ครอบคลุมประโยคคำสั่งมากขึ้น

2.2.2 งานวิจัย “Increase in Modified Condition/Decision Coverage Using Program Code Transformer” [13]

งานวิจัยนี้นำเสนอ ความเป็นไปได้ในการดำเนินการ 4 ตัวแปรบูลีนในหนึ่งเงื่อนไข ที่เป็นเพรดิเคตประกอบ ((A OR B) AND (C OR D)) มีความเป็นไปได้ในการสร้างเงื่อนไขของตัวแปรบูลีนเพื่อให้ได้ผลลัพธ์ทั้งหมด 16 แบบ เพื่อให้ได้ความครอบคลุมเงื่อนไขและการตัดสินใจของเงื่อนไขนี้ ดังนั้นงานวิจัยที่นำเสนอนี้จึงใช้เทคนิคนี้ในการสร้างค่าเวกเตอร์นำเข้า ใช้ในการเลือกสร้างค่าเวกเตอร์นำเข้าให้ความครอบคลุมของการตัดสินใจของทางเดินที่ถูกเลือกมา

2.2.3 งานวิจัย “Computation of the minimal set of paths for observability-based statement coverage” [14]

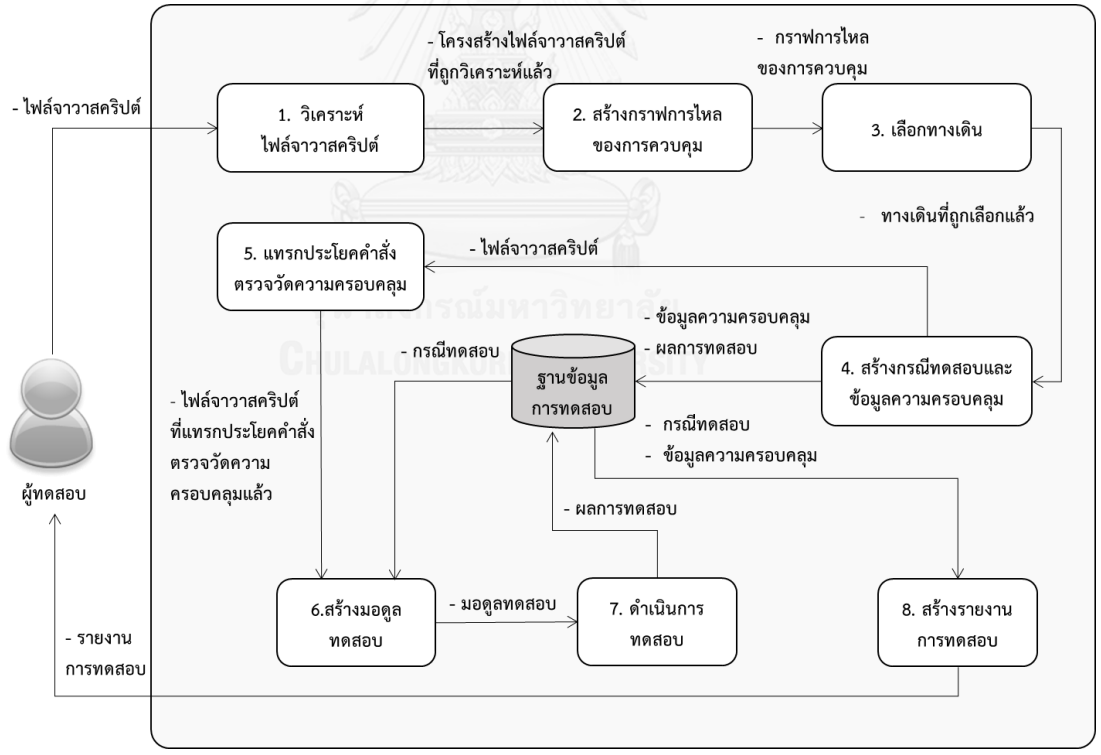
งานวิจัยนี้ได้นำเสนอเทคนิคทดสอบซอฟต์แวร์แบบไวท์บอกซ์ เพื่อใช้สร้างเวกเตอร์นำเข้า และทดสอบกับตัวอย่างโปรแกรมเชิงสถิติทั้งหมด 5 โปรแกรม โดยแต่ละโปรแกรมมีความยาวเฉลี่ยไม่เกิน 200 บรรทัด งานวิจัยที่นำเสนอนี้จึงได้นำ เทคนิคสร้างค่าเวกเตอร์นำเข้าและตัวอย่างทดสอบงานวิจัยนี้ มาประยุกต์ใช้ในการสร้างและทดสอบเครื่องมือนี้

บทที่ 3
การวิเคราะห์และออกแบบเครื่องมือ

ในบทนี้จะอธิบายการวิเคราะห์และการออกแบบเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง โดยจะกล่าวถึงภาพรวมของเครื่องมือ และอธิบายการวิเคราะห์เครื่องมือและออกแบบเครื่องมือด้วยแผนภาพยูสเคส (Use case diagram) แผนภาพคลาส (Class diagram) และแผนภาพลำดับ (Sequence diagram) รวมทั้งอธิบายโครงสร้างของฐานข้อมูลโดยใช้แผนภาพความสัมพันธ์เอนทิตี (ER diagram) ซึ่งมีรายละเอียดดังต่อไปนี้

3.1 ภาพรวมของเครื่องมือ

ภาพรวมการทำงานของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง สามารถแสดงได้ในรูปที่ 3-1



รูปที่ 3-1 แผนภาพแนวคิดของเครื่องมือ

จากรูปที่ 3-1 แสดงถึงแผนภาพแนวคิดของเครื่องมือ ซึ่งสามารถแบ่งออกได้เป็น 8 ส่วนการทำงาน โดยเริ่มจาก ผู้ทดสอบนำเข้าไฟล์จาวาสคริปต์เข้าสู่ระบบ จากนั้นระบบจะวิเคราะห์ไฟล์จาวา-สคริปต์ เมื่อได้ผลลัพธ์เป็นโครงสร้างไฟล์จาวาสคริปต์ที่ถูกวิเคราะห์แล้ว จากนั้นนำมาสร้างกราฟการไหลของการควบคุม โดยตรวจจับตำแหน่งคำสั่งเพื่อใช้ในการสร้างกราฟการไหลของการควบคุม หลังจากนั้นระบบจะเลือกทางเดินจากกราฟการไหลของความควบคุมโดยใช้เกณฑ์การครอบคลุมประโยคคำสั่ง เมื่อเลือกทางเดินเสร็จแล้ว ระบบจะสร้างกรณีทดสอบและข้อมูลความครอบคลุม เมื่อได้กรณีทดสอบและข้อมูลความครอบคลุมแล้ว จึงเก็บไว้ในฐานข้อมูลการทดสอบ จากนั้นกรณีทดสอบจากฐานข้อมูลการทดสอบและไฟล์จาวาสคริปต์ที่แทรกประโยคคำสั่งตรวจวัดความครอบคลุมจะถูกนำไปใช้ในการสร้างมอดูลทดสอบ เพื่อดำเนินการทดสอบ เมื่อดำเนินการทดสอบเสร็จสิ้น จึงสร้างรายงานการทดสอบ โดยนำผลการทดสอบและข้อมูลความครอบคลุมมาประกอบการสร้างรายงานการทดสอบ จากนั้นจึงแสดงรายงานการทดสอบให้ผู้ทดสอบรับทราบ

3.1.1 **วิเคราะห์ไฟล์จาวาสคริปต์**

เมื่อผู้ใช้นำเข้าไฟล์จาวาสคริปต์ในเครื่องมือ ระบบจะใช้ไลบรารีแจงส่วน (Parse) ที่ปรับแต่งแล้วของแอปทานา แจงส่วนไฟล์นำเข้าให้อยู่ในรูปแบบของไฟล์เอกซ์เอ็มแอล เพื่อใช้ในการวิเคราะห์โครงสร้างของไฟล์จาวาสคริปต์ ซึ่งตัวอย่างซอร์สโค้ดของไฟล์จาวาสคริปต์แสดงในรูปที่ 3-2 และตัวอย่างซอร์สโค้ดไฟล์เอกซ์เอ็มแอลที่แจงส่วนไฟล์จาวาสคริปต์แล้ว จะแสดงในรูปที่ 3-3


```
1    function main(a,b){
2         var x = a + b;
3         var y = a - b;
4         if (x > y){
5              z = x + y;
6         }
7         else      {
8              z = x - y;
9         }
10   }
```

รูปที่ 3-2 ตัวอย่างซอร์สโค้ดจาวาสคริปต์

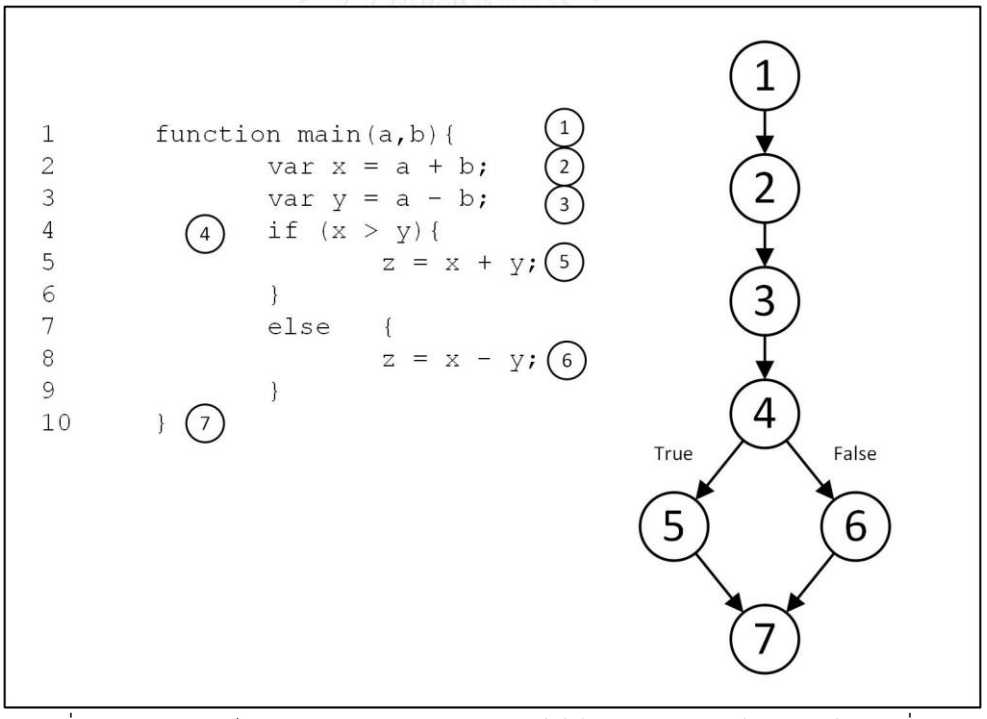
```
1 <JSParseRootNode value="function main (a, b) {var x = a + b;var y = a - b;if (x $gt y) {z
= x + y;} else {z = x - y;}}">
2  <function value="function main (a, b) {var x = a + b;var y = a - b;if (x $gt y) {z = x +
y;} else {z = x - y;}}" name="main">
3    <identifier value="main">
4    </identifier>
5    <parameters value="(a, b)">
6      <identifier value="a">
7      </identifier>
8      <identifier value="b">
9      </identifier>
10    </parameters>
11    <statements value="{var x = a + b;var y = a - b;if (x $gt y) {z = x + y;} else {z = x
- y;}}">
12      <var value="var x = a + b;">
13        <declaration value="x = a + b">
14          <identifier value="x">
15          </identifier>
16          <add value="a + b">
17            <identifier value="a">
18            </identifier>
19            <identifier value="b">
20            </identifier>
21          </add>
22        </declaration>
23      </var>
24      <var value="var y = a - b;">
25        <declaration value="y = a - b">
26          <identifier value="y">
27          </identifier>
28          <subtract value="a - b">
29            <identifier value="a">
30            </identifier>
31            <identifier value="b">
32            </identifier>
33          </subtract>
34        </declaration>
35      </var>
36      <if value="if (x $gt y) {z = x + y;} else {z = x - y;}">
37        <greater_than value="x $gt y">
38          <identifier value="x">
39          </identifier>
40          <identifier value="y">
41          </identifier>
42        </greater_than>
43        <statements value="{z = x + y;}">
44          <assign value="z = x + y;">
45            <identifier value="z">
46            </identifier>
47            <add value="x + y">
48              <identifier value="x">
49              </identifier>
50              <identifier value="y">
51              </identifier>
52            </add>
53          </assign>
54        </statements>
55        <statements value="{z = x - y;}">
56          <assign value="z = x - y;">
57            <identifier value="z">
58            </identifier>
59            <subtract value="x - y">
60              <identifier value="x">
61              </identifier>
62              <identifier value="y">
63              </identifier>
64            </subtract>
65          </assign>
66        </statements>
67      </if>
68    </statements>
69  </function>
70 </JSParseRootNode>
```

รูปที่ 3-3 ตัวอย่างซอร์สโค้ดไฟล์เอกซ์เอ็มแอลที่แจงส่วนไฟล์จาวาสคริปต์แล้ว

จากซอร์สโค้ดไฟล์เอกซ์เอ็มแอลที่แจงส่วนไฟล์จาวาสคริปต์แล้วในรูปที่ 3-3 เครื่องมือจะอ่านไฟล์เอกซ์เอ็มแอลและสกัดประโยคคำสั่งในแต่ละบรรทัดให้อยู่ในรูปของโครงสร้างวัตถุโหนด (Object Node) ซึ่งมีรายละเอียดตัวแปรตามแผนภาพคลาส ObjectNode ในรูปที่ 3-37 เครื่องมือจะเรียงลำดับประโยคคำสั่งตามประโยคคำสั่งจริง เพื่อใช้ในการสร้างกราฟในขั้นตอนต่อไป

3.1.2 **การสร้างกราฟการไหลของการควบคุม**

ในหัวข้อนี้จะกล่าวถึงการสร้างกราฟการไหลของการควบคุม เริ่มจากระบบจะนำไฟล์จาวาสคริปต์ที่แทรกโปรแกรมตรวจวัดความครอบคลุมแล้ว สร้างโหนดขึ้นมาจากตำแหน่งฟังก์ชัน เงื่อนไข และประโยคคำสั่งที่ถูกระบุไว้ในโปรแกรมตรวจวัดความครอบคลุม จากนั้นลากเส้นเชื่อมโยงระหว่างโหนด เมื่อนำเส้นเชื่อมโยงระหว่างโหนดแต่ละอันมาต่อกันจะได้กราฟการไหลของการควบคุม ดังรูปที่ 3-4



รูปที่ 3-4 กราฟการไหลของการควบคุมของซอร์สโค้ดจาวาสคริปต์ตัวอย่างในรูปที่ 3-2

3.1.3 **การเลือกทางเดิน**

ในขั้นตอนนี้ ระบบจะเลือกทางเดินจากกราฟการไหลของการควบคุม โดยใช้วิธีการค้นหาแนวลึกก่อน เพื่อให้ทางเดินผ่านทุกโหนดตามเกณฑ์การครอบคลุมประโยคคำสั่ง เริ่มจากการเลือกโหนด

เริ่มต้น เชื่อมทางเดินไปโหนดต่อไปที่เป็นไปได้ในแนวลึกจนสิ้นสุดโหนดสุดท้าย ทางเดินที่ผ่านจากโหนดเริ่มต้นถึงโหนดสุดท้ายคือหนึ่งทางเดินถูกเลือกมา จากนั้นเลือกทางเดินใหม่ โดยเริ่มจากโหนดเริ่มต้น ค้นหาเชื่อมโหนดต่อไปที่เป็นไปได้ในแนวลึก ให้เชื่อมทางเดินจากโหนดที่ยังไม่ได้ผ่าน ให้ทางเดินผ่านโหนดทุกโหนดอย่างน้อยหนึ่งครั้ง สุดท้ายผลลัพธ์ที่ได้จากขั้นตอนนี้คือทางเดินทุกทางเดินที่ผ่านทุกโหนดอย่างน้อยหนึ่งครั้ง

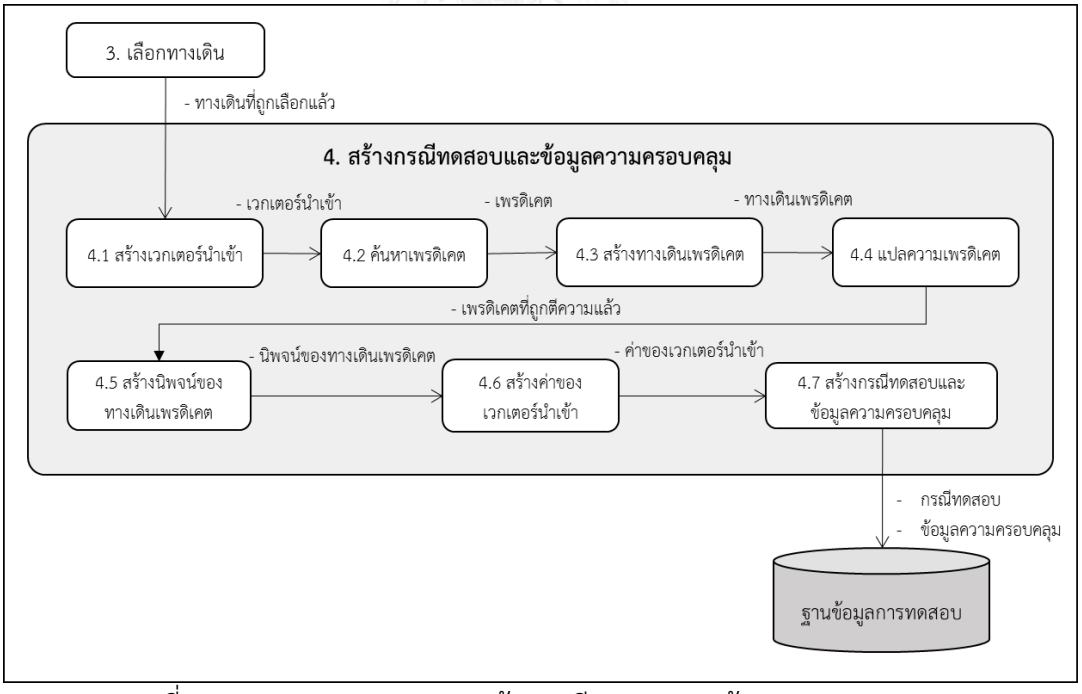
ทางเดินของซอร์สโค้ดจาวาสคริปต์ตัวอย่างในรูปที่ 3-4 เพื่อให้ครอบคลุมประโยคคำสั่ง สามารถแสดงได้ 2 ทางเดิน ได้แก่

ทางเดินแรก : 1 - > 2 -> 3 -> 4 -> 5 -> 7

ทางเดินที่สอง : 1 -> 2 -> 3 - > 4 -> 6 -> 7

3.1.4 **การสร้างกรณีทดสอบและข้อมูลความครอบคลุม**

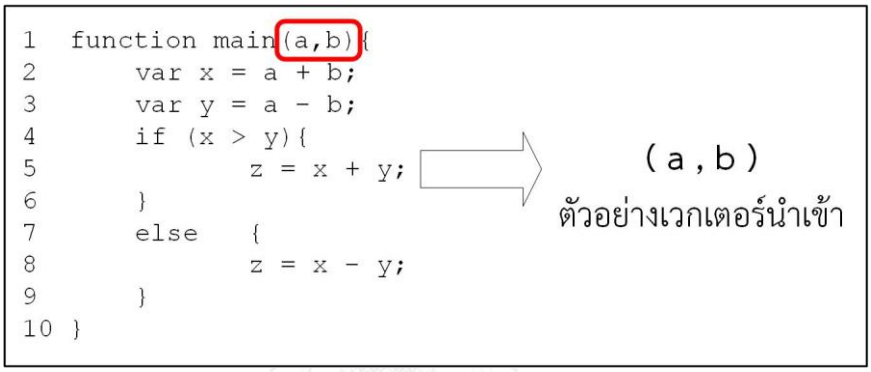
ในขั้นตอนนี้จะกล่าวถึงการสร้างกรณีทดสอบและข้อมูลความครอบคลุม โดยประกอบด้วย 7 ส่วนการทำงานย่อย โดยแสดง ดังรูปที่ 3-5 และมีรายละเอียดดังต่อไปนี้



รูปที่ 3-5 แผนภาพกระบวนการสร้างกรณีทดสอบและข้อมูลความครอบคลุม

1) สร้างเวกเตอร์นำเข้า

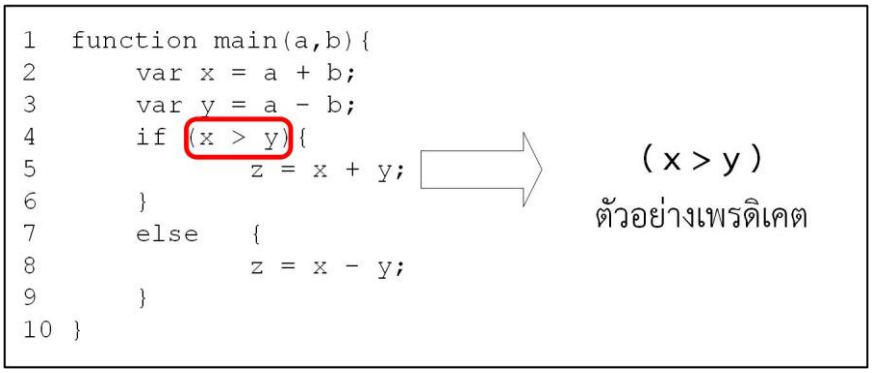
ระบบจะค้นหาพารามิเตอร์ของฟังก์ชันต้นทางหรือโหนดเริ่มต้นของทางเดินที่เลือก ว่ามีจำนวนพารามิเตอร์กี่ตัวและเป็นมีพารามิเตอร์ประเภทอะไรบ้าง ที่ใช้ในการดำเนินการในไฟล์จาวาสคริปต์ที่นำเข้ามา จากรูปที่ 3-6 ได้แสดงตำแหน่งพารามิเตอร์ของฟังก์ชันจาวาสคริปต์ และนำค่าพารามิเตอร์นั้นมาสกัดเป็นเวกเตอร์นำเข้า



รูปที่ 3-6 ตัวอย่างเวกเตอร์นำเข้า จากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

2) ค้นหาเพรดิเคต

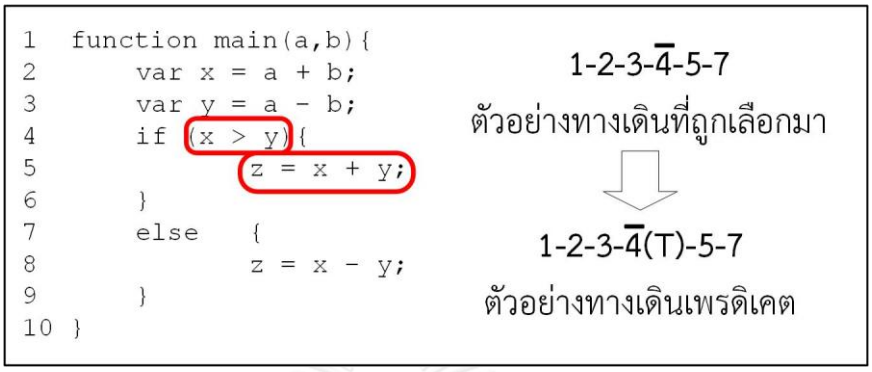
ระบบจะค้นหาเพรดิเคตจากเงื่อนไขคำสั่ง เพื่อหาจุดตัดสินใจของทางเดินที่เลือกเดิน จากรูปที่ 3-7 แสดงตำแหน่งเพรดิเคตของฟังก์ชันจาวาสคริปต์ตัวอย่าง และนำเพรดิเคตนั้นมาดำเนินการขั้นตอนต่อไป



รูปที่ 3-7 ตัวอย่างเพรดิเคต จากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

3) สร้างทางเดินเพรดิเคต

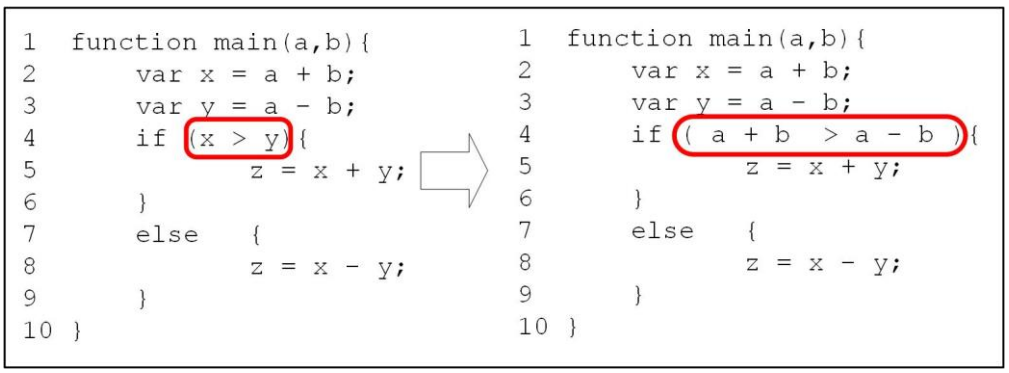
ในขั้นตอนนี้จะนำทางเดินที่เลือกมาสร้างทางเดินเพรดิเคต โดยวิเคราะห์แต่ละเงื่อนไขมีการตัดสินใจอย่างไร และบันทึกผลลัพธ์การตัดสินใจลงไปในทางเดิน ดังตัวอย่างในรูปที่ 3-8 เป็นตัวอย่างแสดงการสร้างทางเดินเพรดิเคตในเงื่อนไขที่เป็นจริง ของทางเดินแรกจากขั้นตอนการเลือกทางเดิน ในหัวข้อ 3.1.3



รูปที่ 3-8 ตัวอย่างทางเดินเพรดิเคต จากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

4) แปลความเพรดิเคต

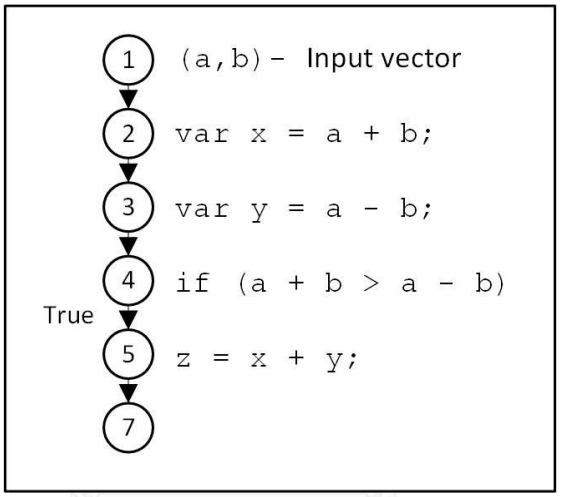
การแปลความเพรดิเคต จะเป็นการแปลตัวแปรภายในเพรดิเคตว่ามาจากแหล่งนำเข้าจากที่ใด โดยแหล่งนำเข้าสามารถเป็น เวกเตอร์นำเข้า ตัวแปรในฟังก์ชัน ค่าคงที่ หรือฟังก์ชันระบบ ที่สามารถเรียกภายในไฟล์จาวาสคริปต์ได้ ซึ่งฟังก์ชันระบบ เป็นเมท็อดสำเร็จรูปในภาษาจาวาสคริปต์ที่กระทำต่ออ็อบเจกต์ชนิดข้อมูลต่างๆ โดยงานวิจัยที่นำเสนอนี้จะทดลองกับฟังก์ชันระบบบางฟังก์ชัน จากนั้นแทนค่าแหล่งนำเข้าเกี่ยวข้องกับเพรดิเคตลงในเพรดิเคตนั้น จากรูปที่ 3-9 เพรดิเคต x > y สามารถ แปลความได้เป็น a + b > a – b เนื่องจากมีการกำหนดค่า x และ y จากเวกเตอร์นำเข้า (a , b) ในบรรทัดที่ 2 และ 3



รูปที่ 3-9 ตัวอย่างการแปลความเพรดิเคต จากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

5) สร้างนิพจน์ของทางเดินเพรดิเคต

นิพจน์ของทางเดินเพรดิเคต คือ การรวมกันระหว่างทางเดินเพรดิเคตกับเพรดิเคตที่ถูกแปลความ เพรดิเคตที่ปรากฏอยู่ในทางเดินจะต้องถูกทดสอบเงื่อนไขทั้งหมดจึงจะสามารถทดสอบทางเดินนั้นได้ ถ้าการทดสอบเพรดิเคตที่ปรากฏอยู่ในทางเดินนั้นไม่ผ่านระบบจะบันทึกทางเดินนั้นว่าไม่สามารถสร้างเวกเตอร์นำเข้าได้ และทางเดินนี้เป็นทางเดินที่เป็นไปไม่ได้ จากรูปที่ 3-10 เป็นตัวอย่างการนำทางเดินเพรดิเคต และการแปลความเพรดิเคต ในรูปที่ 3-8 และรูปที่ 3-9 มารวมกันเพื่อให้ได้นิพจน์ของทางเดินเพรดิ-เคต



รูปที่ 3-10 ตัวอย่างการสร้างนิพจน์ของทางเดินเพรดิเคต จากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

6) สร้างค่าเวกเตอร์นำเข้า

การสร้างเวกเตอร์นำเข้า สามารถแยกออกเป็น 3 ประเภทตามชนิดของพารามิเตอร์ ได้แก่ ตัวเลข สตริง และบูลีน ซึ่งมีรายละเอียดสร้างค่าเวกเตอร์นำเข้าดังนี้

1. กรณีสร้างค่าเวกเตอร์นำเข้าเป็นตัวเลข (Number)

1.1 สร้างค่าแบบไม่สุ่ม

- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าเท่ากันกับค่าคงที่ ระบบจะสร้าง ค่านำเข้าแบบไม่สุ่ม เพื่อเลือกเดินในทางเดินของเพรดิเคตกำหนด เช่น ถ้ามีเพรดิเคต x == 10 ดังนั้นค่านำเข้าที่ถูกสร้างให้ตัวแปร x จะเป็น 10

1.2 สร้างค่าแบบสุ่ม

- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่ามากกว่า น้อยกว่า มากกว่าเท่ากับ น้อยกว่าเท่ากับ ค่าคงที่ ระบบจะสุ่มค่าให้ตรงตามเงื่อนไขของการเปรียบเทียบ เช่น ถ้ามีเพรดิเคต x > 5 ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x ให้มีค่ามากกว่า 5
- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าที่เป็นช่วง ระบบจะสุ่มค่าระหว่างช่วงให้ตรงตามเงื่อนไขของการเปรียบเทียบ เช่น ถ้ามีเพรดิเคต x > 6 && x < 100 ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x จะอยู่ในช่วง 6 ถึง 100
- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าไม่เท่ากันกับค่าคงที่ ระบบจะสร้างค่านำเข้าแบบสุ่ม โดยไม่เท่ากับค่าคงที่นั้น เช่น ถ้ามีเพรดิเคต x != 5 ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x ให้มีค่าไม่เท่ากับ 5
- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าไม่ได้เปรียบเทียบเกี่ยวข้องกับค่าคงที่ ระบบจะนำเพรดิเคตไปทำการสุ่มค่า ให้ได้ค่าตามเงื่อนไขของเพรดิเคต เช่น ถ้ามีเพรดิเคต (y - x > x + y) ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x และ y จะได้ค่าตัวเลขที่สามารถผ่านเงื่อนไขของเพรดิเคตนี้ได้

2. กรณีสร้างค่าเวกเตอร์นำเข้าที่เป็นสตริง (String)

2.1 สร้างค่าแบบไม่สุ่ม

- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าเท่ากับค่าตัวแปรสตริงที่ระบุไว้ชัดเจน เช่น ถ้ามีเพรดิเคต (x == “David”) ดังนั้นค่านำเข้าที่ถูกสร้างค่าให้ตัวแปร x เป็น David

2.2 สร้างค่าแบบสุ่ม

- กรณีค่าเวกเตอร์นำเข้า ไม่ได้เปรียบเทียบในเพรดิเคต เช่น ถ้ามีฟังก์ชัน function show (x); print x; ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x เป็นค่าสตริงแบบสุ่ม
- กรณีค่าเวกเตอร์นำเข้าเปรียบเทียบค่าเท่ากับค่าตัวแปรสตริงค่าที่ไม่เท่ากับค่าเปรียบเทียบ เช่น ถ้ามีเพรดิเคต x != “aaa” ; ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x มีค่าไม่เท่ากับ aaa

- กรณีค่าเวกเตอร์นำเข้าเปรียบเทียบค่าที่ไม่ได้กำหนดค่าที่แน่นอนไว้เช่น ถ้ามีเพรดิเคต x != null ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x เป็นค่าสตริงแบบสุ่ม

3 กรณีสร้างค่าเวกเตอร์นำเข้าที่เป็นบูลีน (Boolean)

3.1 สร้างค่าแบบไม่สุ่ม

- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าเท่ากับค่าตัวแปรบูลีนที่ระบุไว้ เช่น ถ้ามีเพรดิเคต x == true ดังนั้นค่านำเข้าที่ถูกสร้างค่าให้ตัวแปร x จะมีค่าเป็นจริง
- กรณีเพรดิเคตที่มีเวกเตอร์นำเข้าเปรียบเทียบค่าไม่เท่ากับค่าตัวแปรบูลีนที่ระบุไว้ เช่น ถ้ามีเพรดิเคต x != true ดังนั้นค่านำเข้าที่ถูกสร้างค่าให้ตัวแปร x จะมีค่าเป็นเท็จ

3.2 สร้างค่าแบบสุ่ม

- กรณีค่าเวกเตอร์นำเข้าเปรียบเทียบค่าที่ไม่ได้กำหนดค่าที่แน่นอนไว้ เช่น ถ้ามีเพรดิเคตx != null ดังนั้นค่านำเข้าที่ถูกสร้างค่าแบบสุ่มให้ตัวแปร x มีค่าเป็นจริงหรือเท็จก็ได้

7) สร้างกรณีทดสอบและข้อมูลความครอบคลุม

เมื่อได้ค่าเวกเตอร์นำเข้าและทางเดินทดสอบ ระบบจะสร้างกรณีทดสอบ โดยมีรายละเอียด ดังต่อไปนี้

1. หมายเลขกรณีทดสอบ (Test case no.)
2. ชื่อฟังก์ชัน (Function)
3. เวกเตอร์นำเข้า (Input vector)
4. ค่าเวกเตอร์นำเข้า (Input value)
5. ทางเดินที่ถูกทดสอบ (Tested path)

จากตารางที่ 3-1 แสดงตัวอย่างกรณีทดสอบที่ถูกสร้างขึ้น เมื่อใส่รายละเอียดกรณีทดสอบแล้ว

ตารางที่ 3-1 ตารางตัวอย่างแสดงกรณีทดสอบ

Test case no.	Function	Input vector	Input value	Tested Path
1	Main	a , b	a = 836 , b = 37	1-2-3-4(T)-5-7
2	Main	a , b	a = -578 , b = -24	1-2-3-4(F)-6-7

ข้อมูลความครอบคลุม สามารถหาได้จากการคำนวณเปอร์เซ็นต์ความครอบคลุม โดยใช้สูตร

$$\text{เปอร์เซ็นต์ความครอบคลุม} = \frac{\text{จำนวนคำสั่งออกมาใช้อย่างน้อยหนึ่งครั้ง}}{\text{จำนวนคำสั่งที่ดำเนินการทั้งหมด}} \times 100$$

โดยที่ จำนวนคำสั่งออกมาใช้อย่างน้อยหนึ่งครั้ง สามารถหาได้จากจำนวนคำสั่งที่ดำเนินการทั้งหมด ลบจาก จำนวนคำสั่งที่ไม่ได้ออกมาใช้จากทางเดินที่เป็นไปไม่ได้ เมื่อคำนวณได้ค่าเปอร์เซ็นต์ความครอบคลุมแล้ว ระบบจะนำทางเดินที่เป็นไปไม่ได้ ค่าเปอร์เซ็นต์ความครอบคลุม และกรณีทดสอบ ไปเก็บไว้ในฐานข้อมูลการทดสอบ เพื่อใช้ในขั้นตอนสร้างมอดูลทดสอบ

3.1.5 **การแทรกประโยคคำสั่งตรวจวัดความครอบคลุม**

ในขั้นตอนนี้จะแทรกประโยคคำสั่งตรวจวัดความครอบคลุมลงในซอร์สโค้ดของไฟล์จาวาสคริปต์เพื่อตรวจสอบทางเดินในการดำเนินการมอดูลทดสอบ โดยตัวอย่างไฟล์จาวาสคริปต์ในรูปที่ 3-2 จะถูกแทรกคำสั่งตรวจวัดความครอบคลุมจะแสดงได้ในรูปที่ 3-11 โดยประโยคคำสั่งตรวจวัดความครอบคลุมจะถูกกำหนดด้วยตัวแปร “jsstringcounter” และตัวแปรนี้จะเป็นตัวนับในแต่ละประโยคคำสั่ง เพื่อบันทึกทางเดินของคำสั่งในขณะดำเนินการทดสอบ และใช้ตรวจสอบว่าค่าเวกเตอร์นำเข้าที่ถูกสร้างขึ้นสามารถดำเนินการตามทางเดินทดสอบที่ถูกกำหนดไว้ในกรณีทดสอบได้จริง

```
1  function main(a,b){
2     var jsstringcounter =  "1";
3         var x = a + b;
4     jsstringcounter += "-2";
5         var y = a - b;
6     jsstringcounter += "-3";
7     jsstringcounter += "-4";
8         if (x > y){
9            z = x + y;
10    jsstringcounter += "-5";
11        }
12        else {
13           z = x - y;
14    jsstringcounter += "-6";
15        }
16    jsstringcounter += "-End";
17    console.log(jsstringcounter);
18    console.save(jsstringcounter,"16-05-14_22-33_TC01.txt");
19    window.close();
20 }
```

รูปที่ 3-11 ตัวอย่างไฟล์จาวาสคริปต์ที่ถูกแทรกประโยคคำสั่งตรวจวัดความครอบคลุมแล้ว

3.1.6 **การสร้างมอดูลทดสอบ**

ในขั้นตอนนี้ มอดูลทดสอบจะถูกสร้างขึ้นจากกรณีทดสอบโดยใช้แผนแบบ (Template) เป็นตัวเปลี่ยนโครงสร้างกรณีทดสอบให้เป็นมอดูลทดสอบ โดยระบบจะอ่านแผนแบบ แล้วนำค่ากรณีทดสอบและรายละเอียดจาวาสคริปต์ไปแทนค่าในแผ่นแบบ ซึ่งในแผ่นแบบจะมีเพลซโฮลเดอร์ (Placeholder)อยู่ 3 เพลซโฮลเดอร์ ดังนี้

1) [TEST_NAME] คือเพลซโฮลเดอร์สำหรับชื่อการทดสอบ

2) [FUNCTION_NAME] คือเพลซโฮลเดอร์สำหรับชื่อฟังก์ชันที่จะถูกทดสอบ

3) [PARAMETERS] คือเพลซโฮลเดอร์สำหรับค่าของพารามิเตอร์จากค่าเวกเตอร์นำเข้าในกรณีทดสอบ

ระบบจะแทนที่ค่ากรณีทดสอบจากฐานข้อมูลการทดสอบลงในเพลซโฮลเดอร์ทั้ง 3 เพลซโฮลเดอร์ ในรูปที่ 3-12 แสดงมอดูลทดสอบที่สมบูรณ์ ซึ่งเกิดจากการแทนค่าลงในเพลซโฮลเดอร์ และไฟล์จาวาสคริปต์ที่ถูกแทรกประโยคคำสั่งตรวจวัดความครอบคลุมแล้วจะนำมาใช้ในการมอดูลทดสอบด้วย

```
define(["doh"],
   function(doh, test) {

       doh.register("test", [{
               name: "16-05-14_22-33_TC01",
               setUp: (function(console) {

                   console.save = function(data, filename) {

                       if (!data) {
                           console.error('Console.save: No data')
                           return;
                       }

                       if (!filename) filename = 'console.json'

                       if (typeof data === "object") {
                           data = JSON.stringify(data, undefined, 4)
                       }

                       var blob = new Blob([data], {
                               type: 'text/json'
                           }),
                           e = document.createEvent('MouseEvents'),
                           a = document.createElement('a')

                       a.download = filename
                       a.href = window.URL.createObjectURL(blob)
                       a.dataset.downloadurl = ['text/json', a.download,
         a.href].join(':')
                      e.initMouseEvent('click', true, false, window, 0, 0,
         0, 0, 0, 0, false, false, false, false, 0, null)
                      a.dispatchEvent(e)
                   }
               })(console),

               runTest: function(t) {
                   var res = main(-18296.1322, 1524030.2678);
               },
               tearDown: function() {

               }

           }

       ]);
   });

 function main(a, b) {
   var jsstringcounter = "1";
   var x = a + b;
   jsstringcounter += "-2";
   var y = a - b;
   jsstringcounter += "-3";
   jsstringcounter += "-4";
   if (x > y) {
       z = x + y;
       jsstringcounter += "-5";
   } else {
       z = x - y;
       jsstringcounter += "-6";
   }
   jsstringcounter += "-End";
   console.log(jsstringcounter);
   console.save(jsstringcounter, "16-05-14_22-33_TC01.txt");
   window.close();
 }
```

รูปที่ 3-12 ตัวอย่างมอดูลทดสอบ

3.1.7 **การดำเนินการทดสอบ**

เมื่อระบบสร้างมอดูลทดสอบเสร็จแล้ว ระบบจะเรียกการทำงานของดีโอเอชรันเนอร์ และส่งมอดูลทดสอบเข้าดีโอเอชรันเนอร์เพื่อดำเนินการทดสอบ เมื่อการทดสอบเสร็จสิ้น ผลดำเนินการทดสอบจะถูกประเมิน จากนั้นระบบจะส่งผลการผลการทดสอบไปฐานข้อมูลการทดสอบ เพื่อใช้ในการสร้างรายงานการทดสอบ

3.1.8 **การสร้างรายงานการทดสอบ**

ในขั้นตอนนี้ ระบบจะนำข้อมูลความครอบคลุมและผลการทดสอบ จากฐานข้อมูลการทดสอบ มาสร้างรายงานการทดสอบ โดยจะแสดงรายละเอียดกรณีทดสอบสำหรับแต่ละกรณีทดสอบ รวมไปถึงเปอร์เซ็นต์ความครอบคลุมประโยคคำสั่งในการทดสอบ

3.2 การวิเคราะห์และออกแบบเครื่องมือ

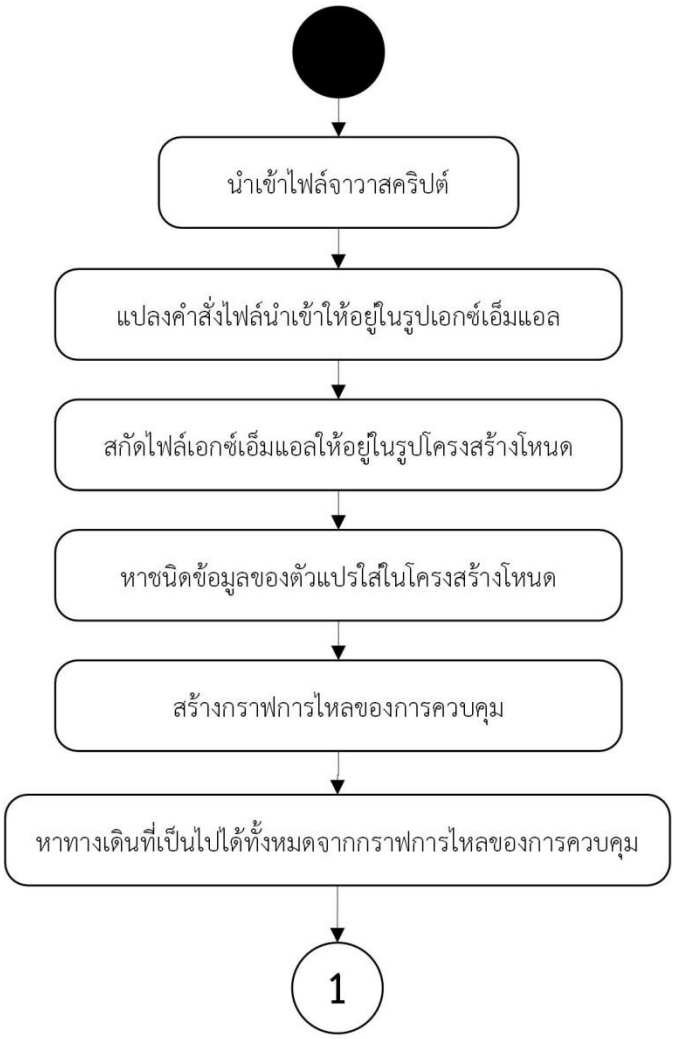
การวิเคราะห์และออกแบบเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง สามารถแสดงได้ด้วยแผนภาพยูสเคส แผนภาพคลาส แผนภาพลำดับ และแผนภาพความสัมพันธ์เอนทิตี

3.2.1 **แผนภาพกิจกรรม**

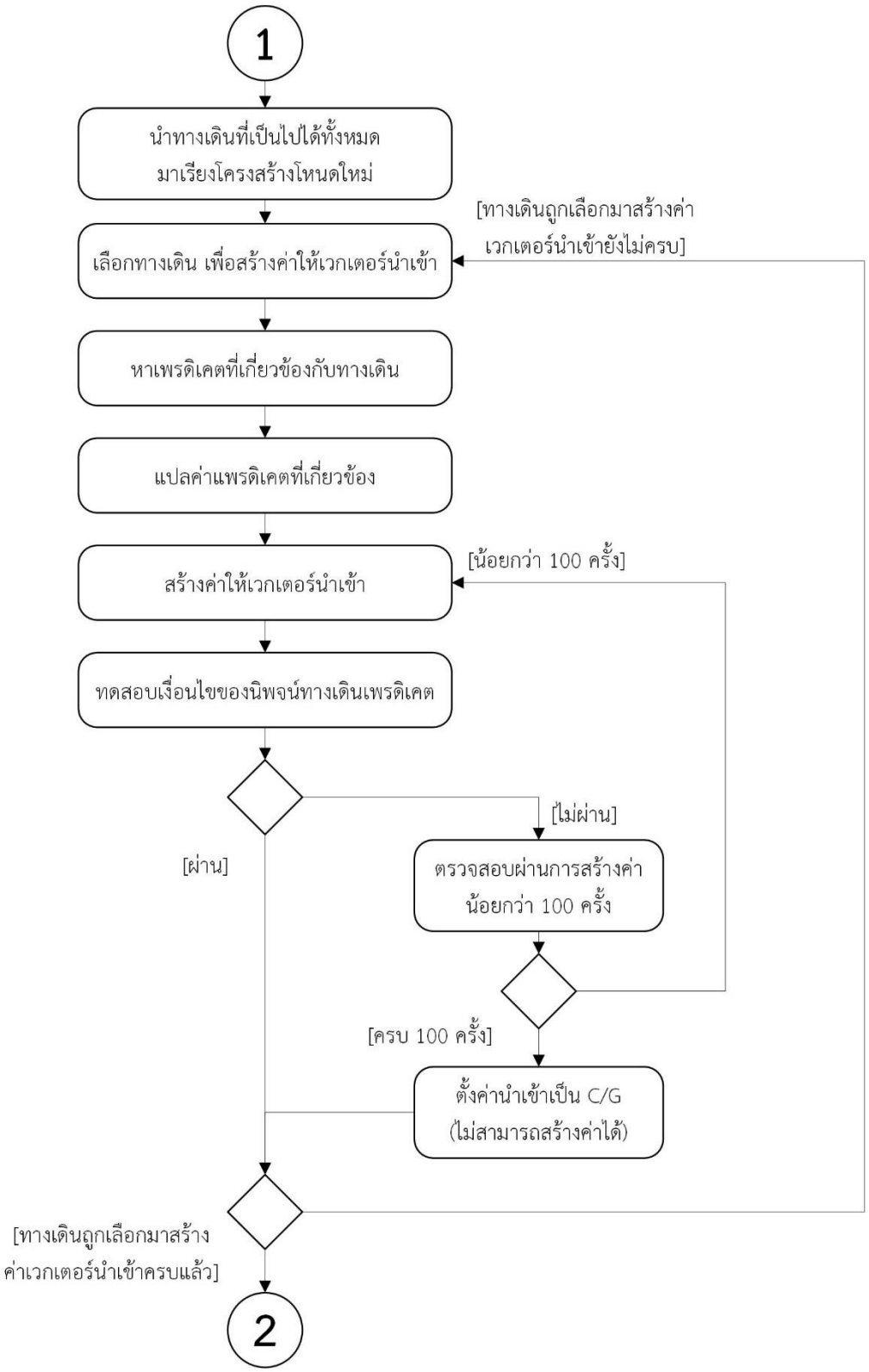
แผนภาพกิจกรรมเป็นแผนภาพที่แสดงกิจกรรมเบื้องต้นและความสัมพันธ์ระหว่างกิจกรรมของเครื่องมือ แผนภาพกิจกรรมหลักของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง สามารถแสดงดังรูปที่ 3-13

จากแผนภาพกิจกรรมในรูปที่ 3-13 แสดงขั้นตอนหลักของเครื่องมือผู้ใช้นำเข้าไฟล์จาวาสคริปต์ เริ่มจากเครื่องมือแปลงไฟล์จาวาสคริปต์ให้อยู่ในรูปเอกซ์เอ็มแอล และสกัดไฟล์ในแต่ละบรรทัดให้อยู่ในรูปโครงสร้างโหนดที่เครื่องมือกำหนดให้ จากนั้นเครื่องมือจะสร้างกราฟการไหลของการควบคุมเพื่อสกัดหาทางเดินทั้งหมดที่ครอบคลุมประโยคคำสั่ง จากนั้นเครื่องมือจะนำแต่ละทางเดินไปสกัดหาค่าของเวกเตอร์นำเข้า เมื่อได้ค่าของเวกเตอร์นำเข้าของทุกทางเดินแล้ว เครื่องมือจะคำนวณเปอร์เซ็นต์ความครอบคลุมประโยคคำสั่งจากการสร้างค่านำเข้า สร้างกรณีทดสอบจากชื่อฟังก์ชันจาวาสคริปต์ ค่าความครอบคลุม ทางเดินทุกทางเดินและค่าของเวกเตอร์นำเข้าทุกค่าเก็บลงฐานข้อมูล

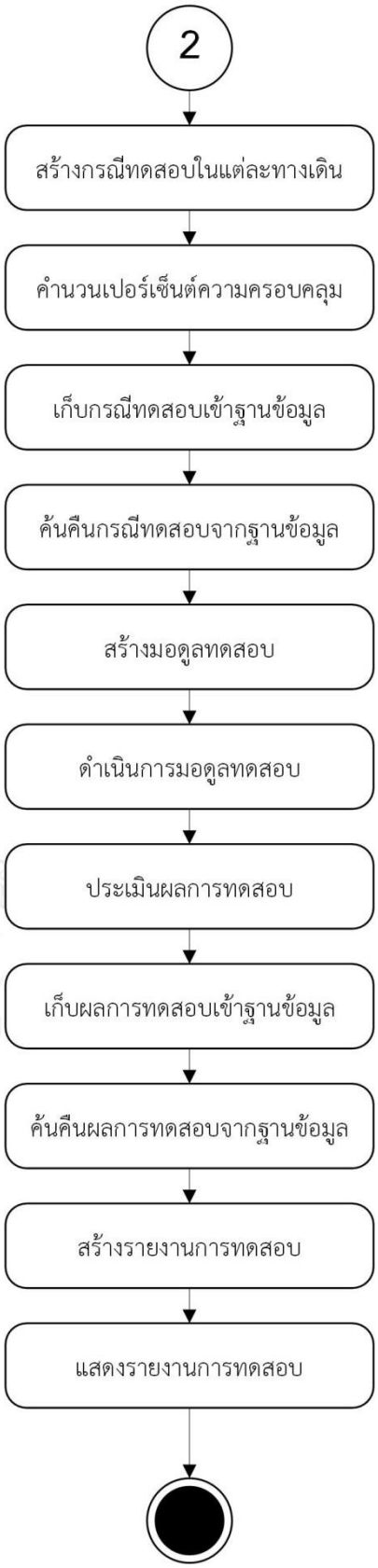
การทดสอบของเครื่องมือ จากนั้นเครื่องมือสร้างมอดูลทดสอบจากกรณีทดสอบในฐานข้อมูลการทดสอบและไฟล์จาวาสคริปต์ที่ถูกแทรกคำสั่งตรวจวัดความครอบคลุม เพื่อดำเนินการกรณีทดสอบที่ถูกสร้างขึ้น โดยมอดูลทดสอบจะทดสอบผ่านโปรแกรมดีโอเอช เมื่อทดสอบเสร็จเครื่องมือจะประเมินคำสั่งตรวจวัดความครอบคลุมว่าตรงกับทางเดินในกรณีทดสอบนั้นๆหรือไม่ จากนั้นจึงบันทึกผลการทดสอบลงในฐานข้อมูลเครื่องมือ และแสดงรายงานการทดสอบให้ผู้ใช้ทราบ



รูปที่ 3-13 แผนภาพกิจกรรมสร้างการทดสอบ



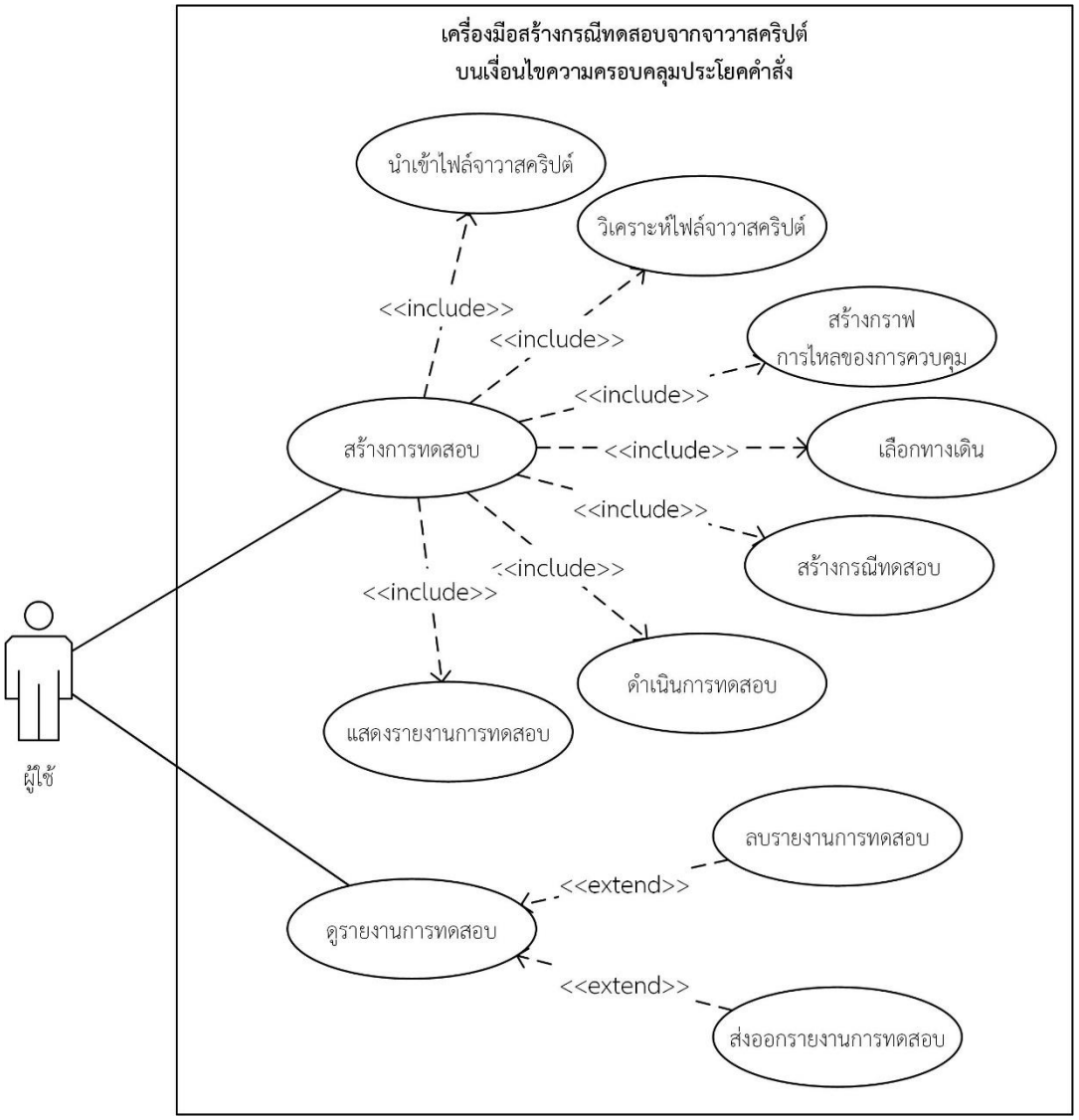
รูปที่ 3-13 แผนภาพกิจกรรมสร้างการทดสอบ(ต่อ)



รูปที่ 3-13 แผนภาพกิจกรรมสร้างการทดสอบ(ต่อ)

3.2.2 **แผนภาพยูสเคส**

แผนภาพยูสเคสเป็นแผนภาพที่ใช้แสดงการทำงานของเครื่องมือที่ผู้ใช้และผู้เกี่ยวข้องปฏิสัมพันธ์ต่อระบบหรือซอฟต์แวร์ โดยแผนภาพยูสเคสของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง ประกอบด้วย 11 ยูสเคส คือ ยูสเคสนำเข้าไฟล์จาวาสคริปต์ ยูสเคสสร้างการทดสอบ ยูสเคสวิเคราะห์ไฟล์จาวาสคริปต์ ยูสเคสสร้างกราฟการไหลของการควบคุม ยูสเคสเลือกทางเดิน ยูสเคสสร้างกรณีทดสอบ ยูสเคสดำเนินการทดสอบ ยูสเคสแสดงรายงานการทดสอบ ยูสเคสดูรายงานการทดสอบ ยูสเคสลบรายงานการทดสอบและยูสเคสส่งออกรายงานการทดสอบ โดยยูสเคสทั้งหมดจะแสดงในรูปที่ 3-14



รูปที่ 3-14 แผนภาพยูสเคสของเครื่องมือ

รายละเอียดยูสเคสด้านข้างนี้จะอธิบายรายละเอียดยูสเคสทั้งหมดในแผนภาพ โดย รายละเอียดยูสเคสสร้างการทดสอบแสดงในตารางที่ 3-2 รายละเอียดยูสเคสนำเข้าไฟล์จาวาสคริปต์แสดงในตารางที่ 3-3 รายละเอียดยูสเคสวิเคราะห์ไฟล์จาวาสคริปต์แสดงในตารางที่ 3-4 รายละเอียดยูสเคสสร้างกราฟการไหลของการควบคุมแสดงในตารางที่ 3-5 รายละเอียดยูสเคสเลือกทางเดินแสดงในตารางที่ 3-6 รายละเอียดยูสเคสสร้างกรณีทดสอบแสดงในตารางที่ 3-7 รายละเอียดยูสเคสดำเนินการทดสอบแสดงในตารางที่ 3-8 รายละเอียดยูสเคสแสดงรายงานการทดสอบแสดงในตารางที่ 3-9 รายละเอียดยูสเคสดูรายงานการทดสอบแสดงใน ตารางที่ 3-10 รายละเอียดยูสเคสลบรายงานการทดสอบแสดงในตารางที่ 3-11 รายละเอียดยูสเคสส่งออกรายงานการทดสอบแสดงในตารางที่ 3-12

ตารางที่ 3-2 รายละเอียดยูสเคสสร้างการทดสอบ

ยูสเคส	สร้างการทดสอบ
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	สร้างการทดสอบใหม่
ยูสเคสที่สัมพันธ์	Include : นำเข้าไฟล์จาวาสคริปต์, วิเคราะห์ไฟล์จาวาสคริปต์, สร้างกราฟการไหลของการควบคุม, เลือกทางเดิน, สร้างกรณีทดสอบ, ดำเนินการทดสอบ, แสดงรายงานการทดสอบ
เงื่อนไขก่อนหน้า	ผู้ใช้เปิดหน้าต่างหลักของเครื่องมือ
ขั้นตอน	1. ยูสเคสนี้จะเรียกยูสเคสอื่นใช้งานเรียงตามลำดับ ดังนี้ 1.1 ยูสเคสนำเข้าไฟล์จาวาสคริปต์ 1.2 ยูสเคสวิเคราะห์ไฟล์จาวาสคริปต์ 1.3 ยูสเคสสร้างกราฟการไหลของการควบคุม 1.4 ยูสเคสเลือกทางเดิน 1.5 ยูสเคสสร้างกรณีทดสอบ 1.6 ยูสเคสดำเนินการทดสอบ 1.7 ยูสเคสแสดงรายงานการทดสอบ 2. ผลลัพธ์ที่ได้แต่ละยูสเคสจะส่งกลับมาที่ยูสเคสนี้ 3. ยูสเคสนี้จะส่งค่านำเข้าให้แต่ละยูสเคสที่เรียกใช้งาน
เงื่อนไขภายหลัง	-

ตารางที่ 3-3 รายละเอียดยูสเคสนำเข้าไฟล์จาวาสคริปต์

ยูสเคส	นำเข้าไฟล์จาวาสคริปต์
แอคเตอร์	ผู้ใช้
รายละเอียดยูสเคส	นำเข้าไฟล์จาวาสคริปต์ เพื่อสร้างการทดสอบใหม่
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	ผู้ใช้เปิดหน้าต่างหลักของเครื่องมือ
ขั้นตอน	1. ผู้ใช้กดปุ่ม “New Test” 2. เครื่องมือแสดงหน้าต่าง Select File 3. ผู้ใช้เลือกไฟล์จาวาสคริปต์ที่ต้องการทดสอบและกดปุ่ม “OK” 4. เครื่องมือแสดงหน้าต่างป๊อบอัพแสดงว่า สถานะเครื่องมือจะมีการเรียกใช้โปรแกรมดีโอเอชให้ผู้ใช้รับทราบ 5. ผู้ใช้คลิกหน้าต่างป๊อบอัพ เพื่อให้เครื่องมือดำเนินการขั้นตอนต่อไป
เงื่อนไขภายหลัง	เครื่องมือจะเริ่มสร้างการทดสอบไฟล์จาวาสคริปต์

ตารางที่ 3-4 รายละเอียดยูสเคสวิเคราะห์ไฟล์จาวาสคริปต์

ยูสเคส	วิเคราะห์ไฟล์จาวาสคริปต์
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือวิเคราะห์ไฟล์จาวาสคริปต์ ที่ผู้ใช้นำเข้า
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	ผู้ใช้ต้องนำเข้าไฟล์จาวาสคริปต์
ขั้นตอน	1. เครื่องมือเรียกไลบรารีแจงส่วนของแอปทานา ใช้ในการแจง-ส่วนไฟล์จาวาสคริปต์เป็นไฟล์เอกซ์เอ็มแอล 2. เครื่องมืออ่านไฟล์เอกซ์เอ็มแอล สกัดข้อมูลให้อยู่ในโครงสร้างวัตถุโหนด
เงื่อนไขภายหลัง	เครื่องมือนำโครงสร้างวัตถุโหนดมาสร้างกราฟจาวาสคริปต์

ตารางที่ 3-5 รายละเอียดยูสเคสสร้างกราฟการไหลของการควบคุม

ยูสเคส	สร้างกราฟการไหลของการควบคุม
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือสร้างกราฟ จากโครงสร้างวัตถุโหนด
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	โครงสร้างวัตถุโหนด ที่ถูกสร้างจากยูสเคสวิเคราะห์ไฟล์จาวาสคริปต์
ขั้นตอน	1. เครื่องมือเรียกไลบรารีเจกราฟที ใช้ในการสร้างกราฟการไหลของการควบคุม
เงื่อนไขภายหลัง	เครื่องมือนำกราฟการไหลของการควบคุมมาหาทางเดินทั้งหมด

ตารางที่ 3-6 รายละเอียดยูสเคสเลือกทางเดิน

ยูสเคส	เลือกทางเดิน
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือเลือกทางเดิน
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	การไหลของการควบคุม ถูกสร้างจากยูสเคสสร้างกราฟการไหลของการควบคุม
ขั้นตอน	1. เครื่องมือหาทางเดินทั้งหมดจากการไหลของการควบคุมโดยวิธีการค้นหาแนวลึกก่อน 2. เครื่องมือเลือกทางเดินคำสั่งจากทางเดินทั้งหมด เพื่อสร้างกรณี-ทดสอบ
เงื่อนไขภายหลัง	เครื่องมือนำทางเดินที่ถูกเลือก มาสร้างกรณีทดสอบ

ตารางที่ 3-7 รายละเอียดยูสเคสสร้างกรณีทดสอบ

ยูสเคส	สร้างกรณีทดสอบ
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือสร้างกรณีทดสอบ
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	ทางเดินคำสั่ง ที่ถูกเลือกจากยูสเคสเลือกทางเดิน
ขั้นตอน	1. เครื่องสร้างหาชื่อฟังก์ชัน ชื่อเวกเตอร์นำเข้า เพรดิเคตที่เกี่ยวข้องจากทางเดินที่เลือกมา 2. เครื่องมือสร้างนิพจน์ของทางเดินเพรดิเคต 3. เครื่องมือสร้างค่าเวกเตอร์นำเข้า 4. เครื่องมือสร้างกรณีทดสอบ 5. เครื่องมือคำนวณค่าความครอบคลุมประโยคคำสั่ง เมื่อทางเดินทั้งหมดถูกสร้างกรณีทดสอบทั้งหมดแล้ว 6. เครื่องมือส่งออกกรณีทดสอบทั้งหมดลงฐานข้อมูลการทดสอบ
เงื่อนไขภายหลัง	เครื่องมือนำกรณีทดสอบมาดำเนินการทดสอบ

ตารางที่ 3-8 รายละเอียดยูสเคสดำเนินการทดสอบ

ยูสเคส	ดำเนินการทดสอบ
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือดำเนินการทดสอบ
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	กรณีทดสอบในฐานข้อมูล ถูกสร้างขึ้นจากยูสเคสสร้างกรณีทดสอบ
ขั้นตอน	1. เครื่องมือแทรกประโยคคำสั่งตรวจคำสั่งตรวจวัดความครอบคลุม 2. เครื่องมือค้นคืนกรณีทดสอบในฐานข้อมูล 3. เครื่องมือสร้างมอดูลทดสอบจากกรณีทดสอบ 4. เครื่องมือดำเนินการทดสอบ ด้วยโปรแกรมดีโอเอชผ่านเว็บบราวเซอร์ 5. เว็บบราวเซอร์บันทึกไฟล์ทางเดิน ที่มาจากการดำเนินการทดสอบประโยคคำสั่งตรวจวัดความครอบคลุม
เงื่อนไขภายหลัง	เครื่องมืออ่านทางเดิน ประเมินผล และแสดงรายงานการทดสอบ

ตารางที่ 3-9 รายละเอียดยูสเคสแสดงรายงานการทดสอบ

ยูสเคส	แสดงรายงานการทดสอบ
แอคเตอร์	เครื่องมือ
รายละเอียดยูสเคส	เครื่องมือแสดงรายงานการทดสอบ
ยูสเคสที่สัมพันธ์	Include : สร้างการทดสอบ
เงื่อนไขก่อนหน้า	ไฟล์ทางเดิน ที่ถูกบันทึกจากยูสเคสดำเนินการทดสอบ
ขั้นตอน	1. เครื่องมืออ่านไฟล์ทางเดิน 2. เครื่องมือประเมินผลทางเดินการดำเนินการทดสอบจริงเทียบกับทางเดินที่ถูกเลือกใช้ในการสร้างกรณีทดสอบ 3. เครื่องมือบันทึกผลการทดสอบลงในฐานข้อมูล 4. เครื่องมือค้นคืนกรณีทดสอบและผลการทดสอบในฐานข้อมูล 5. เครื่องมือแสดงรายงานการทดสอบ
เงื่อนไขภายหลัง	-

ตารางที่ 3-10 รายละเอียดยูสเคสดูรายงานการทดสอบ

ยูสเคส	ดูรายงานการทดสอบ
แอคเตอร์	ผู้ใช้
รายละเอียดยูสเคส	ผู้ใช้เลือกรายงานการทดสอบที่ต้องการดู จากรายการบันทึกผลการทดสอบ ที่ถูกสร้างและบันทึกลงในฐานข้อมูลก่อนหน้านี้
ยูสเคสที่สัมพันธ์	Extend : ลบรายงานการทดสอบ ส่งออกรายงานการทดสอบ
เงื่อนไขก่อนหน้า	รายงานการทดสอบจะต้องถูกสร้างขึ้น บันทึกลงในฐานข้อมูล และแสดงให้เห็นในรายการบันทึกผลการทดสอบ
ขั้นตอน	เลือกรายงานการทดสอบที่ต้องการดู จากรายการบันทึกผลการทดสอบ
เงื่อนไขภายหลัง	เครื่องมือดึงรายงานการทดสอบจากฐานข้อมูลการทดสอบ และแสดงรายงานการทดสอบ

ตารางที่ 3-11 รายละเอียดยูสเคสลบรายงานการทดสอบ

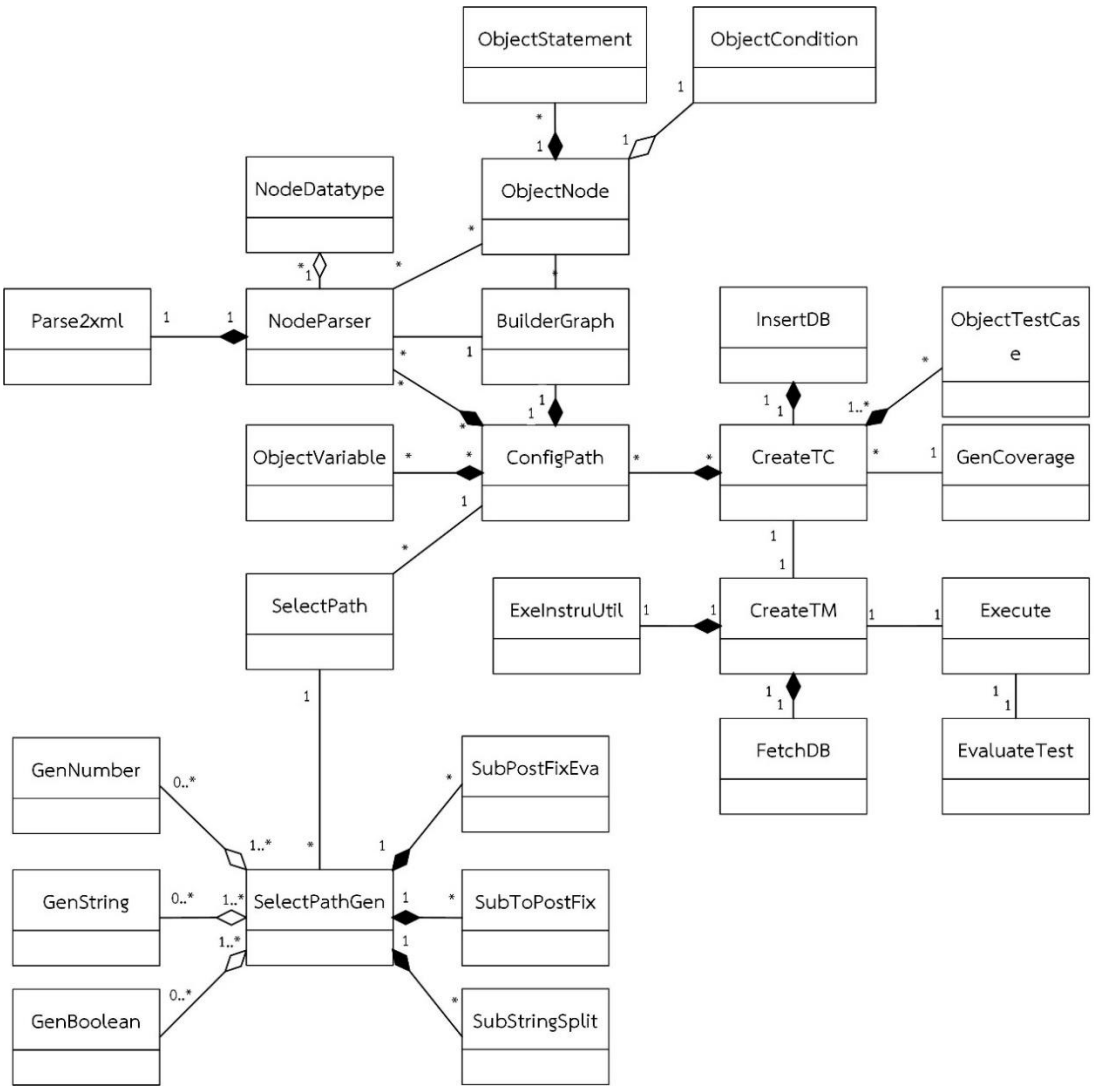
ยูสเคส	ลบรายงานการทดสอบ
แอคเตอร์	ผู้ใช้
รายละเอียดยูสเคส	ลบรายงานการทดสอบ ที่ถูกเลือกในรายการบันทึกผลการทดสอบ
ยูสเคสที่สัมพันธ์	Extend : ดูรายงานการทดสอบ
เงื่อนไขก่อนหน้า	ผู้ใช้เลือกรายงานการทดสอบที่ต้องการลบ จากรายการบันทึกผลการทดสอบ จากยูสเคสดูรายงานการทดสอบ
ขั้นตอน	ผู้ใช้กดปุ่ม “Delete”
เงื่อนไขภายหลัง	1. เครื่องมือลบรายการการทดสอบที่ถูกเลือก ในฐานข้อมูลการทดสอบ 2. เครื่องมือปรับปรุงรายการบันทึกผลการทดสอบ

ตารางที่ 3-12 รายละเอียดยูสเคสส่งออกรายงานการทดสอบ

ยูสเคส	ส่งออกรายงานการทดสอบ
แอคเตอร์	ผู้ใช้
รายละเอียดยูสเคส	ส่งออกรายงานการทดสอบ ที่ถูกเลือกในรายการบันทึกผลการทดสอบ
ยูสเคสที่สัมพันธ์	Extend : ดูรายงานการทดสอบ
เงื่อนไขก่อนหน้า	ผู้ใช้เลือกรายงานการทดสอบที่ต้องการส่งออก จากรายการบันทึกผลการทดสอบ จากยูสเคสดูรายงานการทดสอบ
ขั้นตอน	ผู้ใช้กดปุ่ม “Export”
เงื่อนไขภายหลัง	เครื่องมือส่งออกรายการทดสอบ โดยใช้รูปแบบไฟล์นามสกุลเอกซ์แอลเอส (*.xls) สำหรับให้โปรแกรม Excel อ่านได้

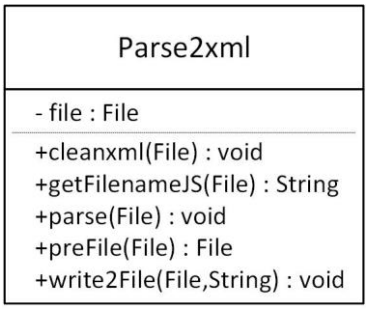
3.2.3 **แผนภาพคลาส**

แผนภาพคลาสเป็นแผนภาพที่แสดงรายละเอียดของคลาสและความสัมพันธ์ระหว่างคลาส ซึ่งแผนภาพคลาสของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง แสดงได้ดังรูปที่ 3-15 ซึ่งมีรายละเอียดดังต่อไปนี้

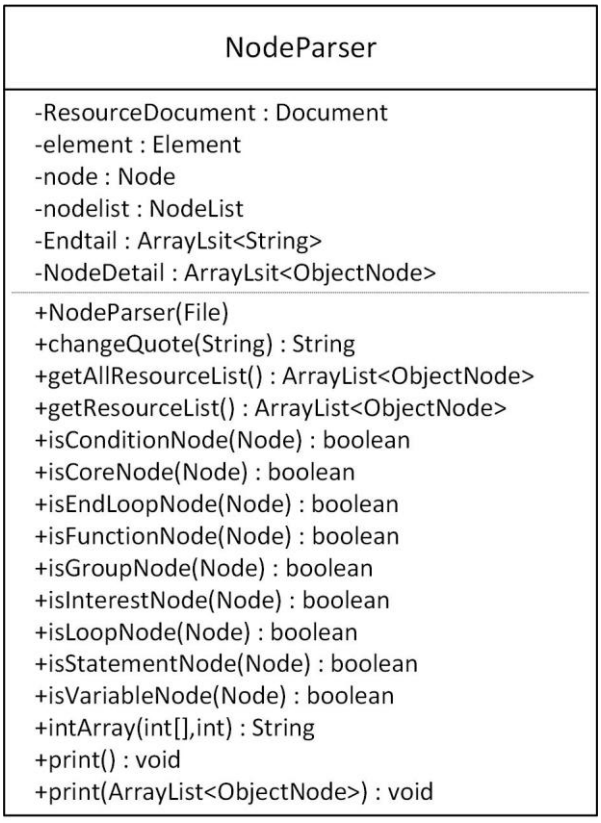


รูปที่ 3-15 แผนภาพคลาสของเครื่องมือ

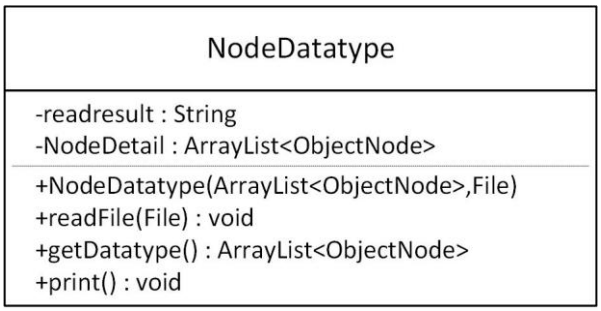
1) คลาส Parse2xml เป็นคลาสที่ทำหน้าที่อ่านไฟล์จาวาสคริปต์นำเข้าแล้วแปลงซอร์สโค้ดและบันทึกให้อยู่รูปของไฟล์เอกซ์เอ็มแอล แสดงดังรูปที่ 3-16
2) คลาส NodeParser เป็นคลาสที่ทำหน้าที่สกัดค่าไฟล์เอกซ์เอ็มแอลให้อยู่ในโครงสร้างของโหนด เพื่อที่จะใช้ในการสร้างกราฟ แสดงดังรูปที่ 3-17
3) คลาส NodeDatatype เป็นคลาสที่ทำหน้าที่หาค่าชนิดของข้อมูลของฟังก์ชันพารามิเตอร์จากไฟล์เอกซ์เอ็มแอล เพื่อระบุชนิดของข้อมูลที่จะใช้ในการสร้างเวกเตอร์นำเข้า แสดงดังรูปที่ 3-18
4) คลาส BuildGraph เป็นคลาสที่ทำหน้าที่สร้างกราฟการไหลของการควบคุม และหาทางเดินทั้งหมดของกราฟ ด้วยวิธีการค้นหาแนวลึกก่อน แสดงดังรูปที่ 3-19



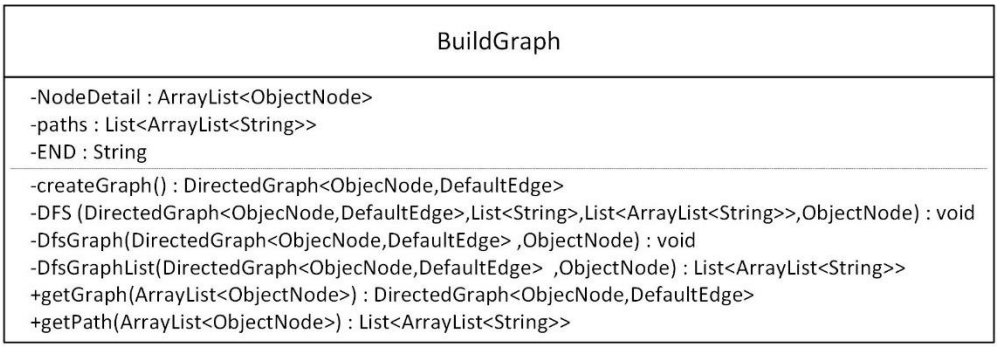
รูปที่ 3-16 คลาส Parse2xml



รูปที่ 3-17 คลาส NodeParser

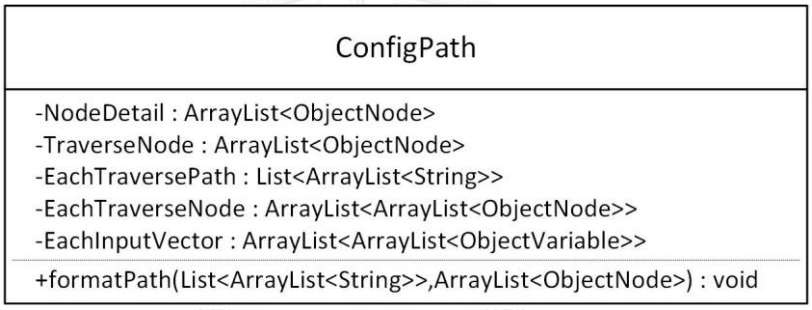


รูปที่ 3-18 คลาส NodeDatatype



รูปที่ 3-19 คลาส BuildGraph

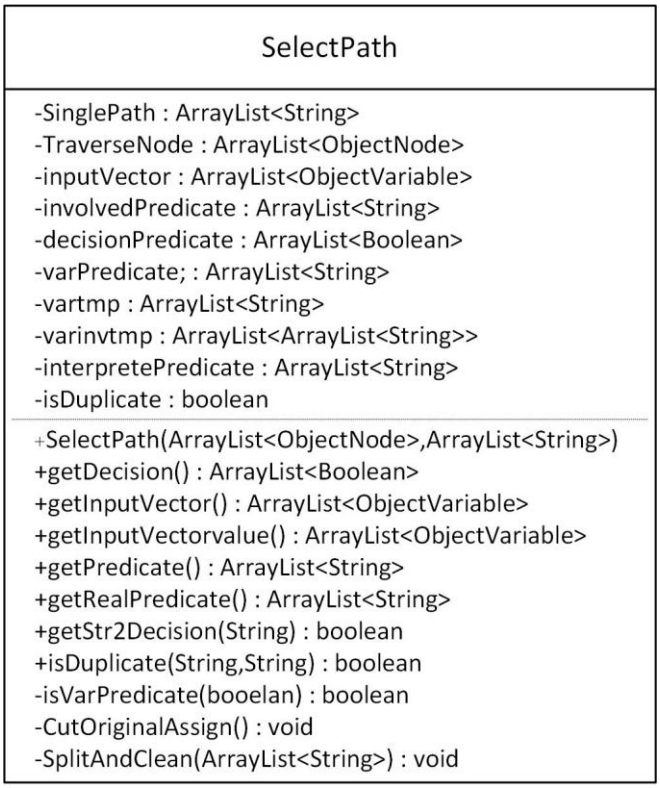
5) คลาส ConfigPath เป็นคลาสที่ทำหน้าที่จัดเรียงข้อมูลทางเดินทั้งหมด มาจับคู่กับแต่ละโหนดที่อยู่ในทางเดิน และนำแต่ละทางเดินที่ถูกจับคู่กับโหนดแล้วมาสร้างเวกเตอร์นำเข้า แสดงดังรูปที่ 3-20



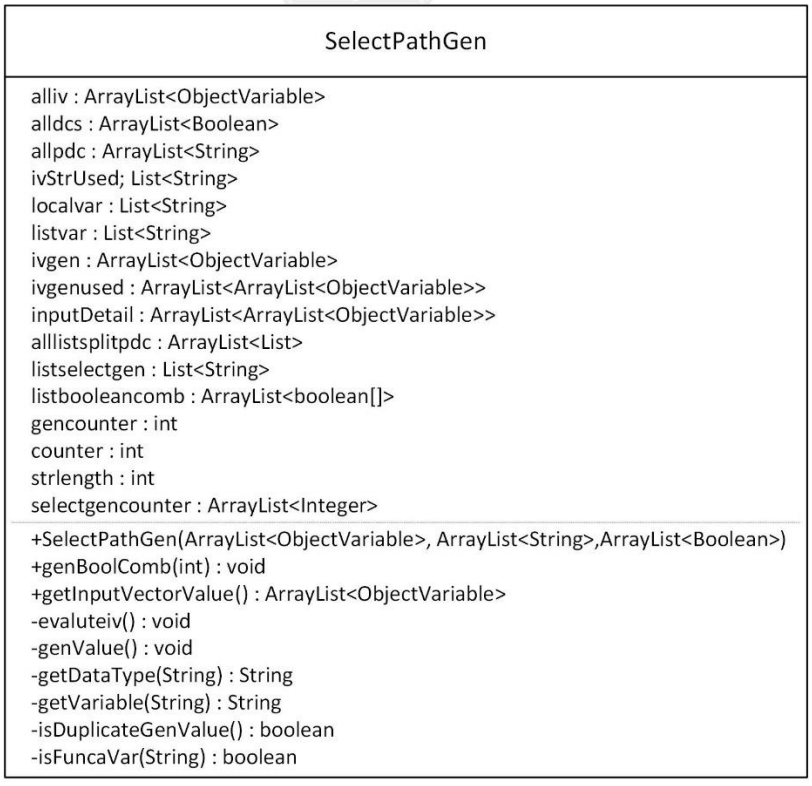
รูปที่ 3-20 คลาส ConfigPath

6) คลาส SelectPath เป็นคลาสที่ทำหน้าที่จัดเรียงข้อมูลจากทางเดินที่ถูกเลือก ให้อยู่ในรูปเวกเตอร์นำเข้า ค่าของเวกเตอร์นำเข้า เพรดิเคตที่เกี่ยวข้อง และการตัดสินของเพรดิเคต รวมไปถึงการแปลเพรดิเคตที่เกี่ยวข้องด้วย แสดงดังรูปที่ 3-21

7) คลาส SelectPathGen เป็นคลาสที่ทำหน้าที่สร้างค่านำเข้าจากข้อมูลที่ได้ในคลาส SelectPath โดยทำหน้าที่สร้างค่านำเข้าให้กับเวกเตอร์นำเข้าตามชนิดข้อมูลที่ระบุมา และพิสูจน์กับเพรดิเคตที่เกี่ยวข้องให้มีการตัดสินใจตรงกับทางเดินที่ถูกเลือกมา แสดงดังรูปที่ 3-22

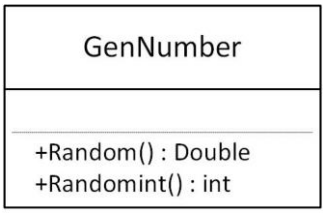


รูปที่ 3-21 คลาส SelectPath



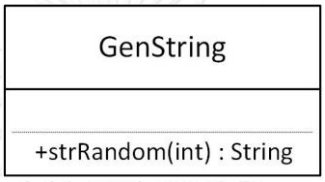
รูปที่ 3-22 คลาส SelectPathGen

8) คลาส GenNumber เป็นคลาสที่ทำหน้าที่สร้างข้อมูลนำเข้าให้เวกเตอร์นำเข้าที่มีชนิดข้อมูลเป็นตัวเลข แสดงดังรูปที่ 3-23



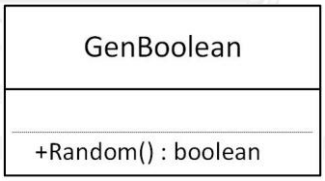
รูปที่ 3-23 คลาส GenNumber

9) คลาส GenString เป็นคลาสที่ทำหน้าที่สร้างข้อมูลนำเข้าให้เวกเตอร์นำเข้าที่มีชนิดข้อมูลเป็นสตริง แสดงดังรูปที่ 3-24



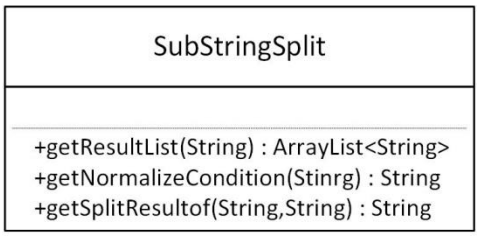
รูปที่ 3-24 คลาส GenString

10) คลาส GenBoolean เป็นคลาสที่ทำหน้าที่สร้างข้อมูลนำเข้าให้เวกเตอร์นำเข้าที่มีชนิดข้อมูลเป็นบูลีน แสดงดังรูปที่ 3-25



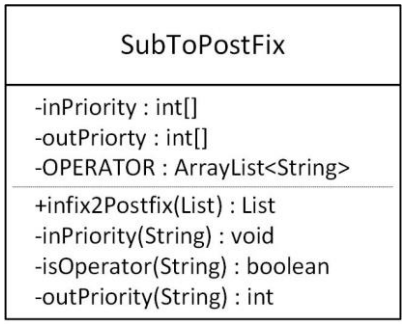
รูปที่ 3-25 คลาส GenBoolean

11) คลาส SubStringSplit เป็นคลาสที่ทำหน้าที่แบ่งสตริงเพรดิเคตเพื่อใช้ทำ PostFix แสดงดังรูปที่ 3-26



รูปที่ 3-26 คลาส SubStringSplit

12) คลาส SubToPostFix เป็นคลาสที่ทำหน้าที่แปลงเพรดิเคตที่อยู่ในรูป PostFix เป็น Prefix แสดงดังรูปที่ 3-27

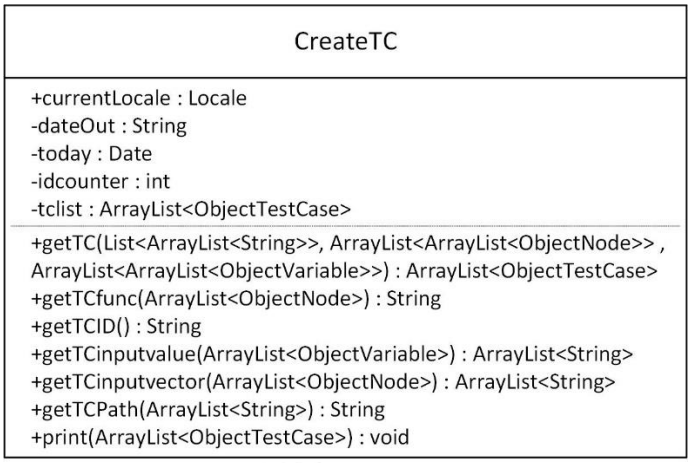


รูปที่ 3-27 คลาส SubToPostFix

13) คลาส SubPostFixEva เป็นคลาสที่ทำหน้าที่พิสูจน์เพรดิเคตที่อยู่ในรูป Postfix แสดงรูปที่ 3-28

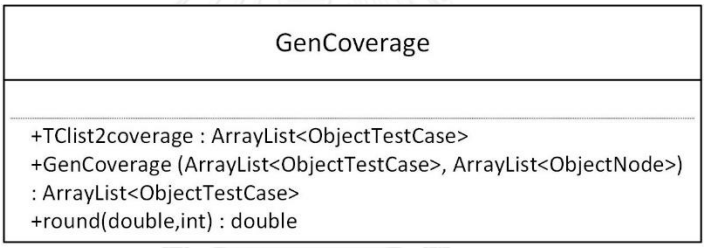
รูปที่ 3-28 คลาส SubPostFixEva

14) คลาส CreateTC เป็นคลาสที่ทำหน้าที่สร้างกรณีทดสอบทั้งหมด แสดงดังรูปที่ 3-29



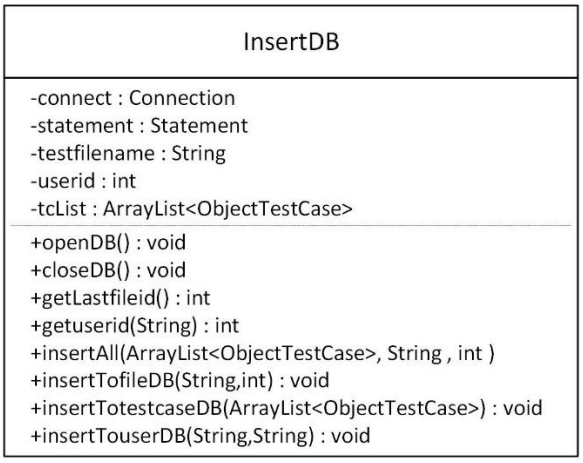
รูปที่ 3-29 คลาส CreateTC

15) คลาส GenCoverage เป็นคลาสที่ทำหน้าที่คำนวณความครอบคลุมประโยคคำสั่งจากกรณีทดสอบทั้งหมด แสดงดังรูปที่ 3-30



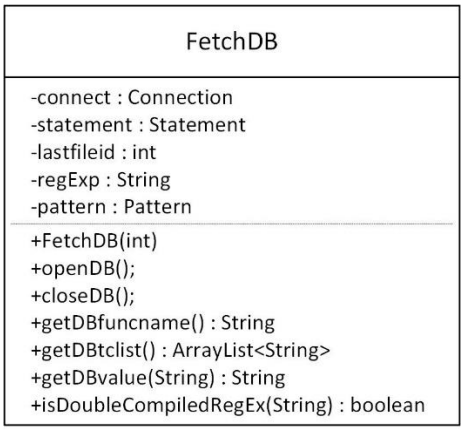
รูปที่ 3-30 คลาส GenCoverage

16) คลาส InsertDB เป็นคลาสที่ทำหน้าที่ส่งค่าในกรณีทดสอบ เข้าฐานข้อมูลทดสอบของเครื่องมือ แสดงดังรูปที่ 3-31



รูปที่ 3-31 คลาส InsertDB

17) คลาส FetchDB เป็นคลาสที่ทำหน้าที่ดึงค่ากรณีทดสอบจากฐานข้อมูลการทดสอบของเครื่องมือ แสดงดังรูปที่ 3-32

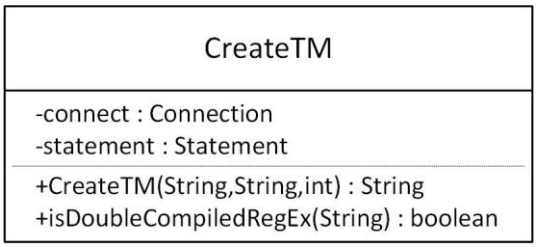


รูปที่ 3-32 คลาส FetchDB

18) คลาส ExeInstruUtil เป็นคลาสที่ทำหน้าที่แทรกคำสั่งตรวจวัดความครอบคลุมกับไฟล์จาวาสคริปต์ที่จะดำเนินการกรณีทดสอบ แสดงดังรูปที่ 3-33

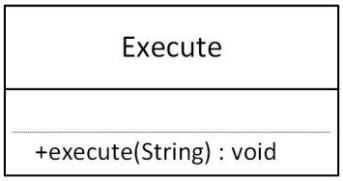
รูปที่ 3-33 คลาส ExeInstruUtil

19) คลาส CreateTM เป็นคลาสที่ทำหน้าที่สร้างมอดูลทดสอบจากกรณีทดสอบ โดยไฟล์มอดูลทดสอบจะใช้ข้อมูลจาก กรณีทดสอบในฐานข้อมูลจากคลาส FetchDB และไฟล์จาวาสคริต์ที่แทรกคำสั่งตรวจวัดความครอบคลุมแล้วจากคลาส ExeInstruUtil ไฟล์มอดูลทดสอบจะใช้เป็นไฟล์ตัวกลางในการดำเนินการกรณีทดสอบใน โปรแกรมดีโอเอช แสดงดังรูปที่ 3-34



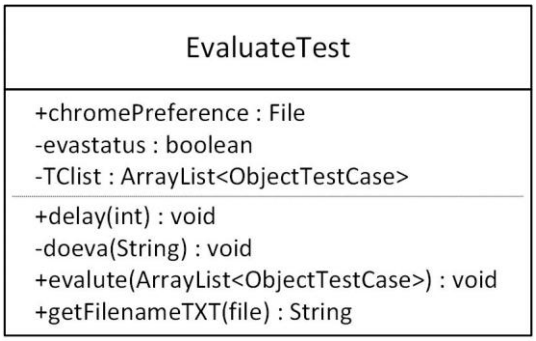
รูปที่ 3-34 คลาส CreateTM

20) คลาส Execute เป็นคลาสที่ทำหน้าที่ดำเนินการมอดูลทดสอบ โดยใช้โปรแกรมดีโอเอช แสดงดังรูปที่ 3-35



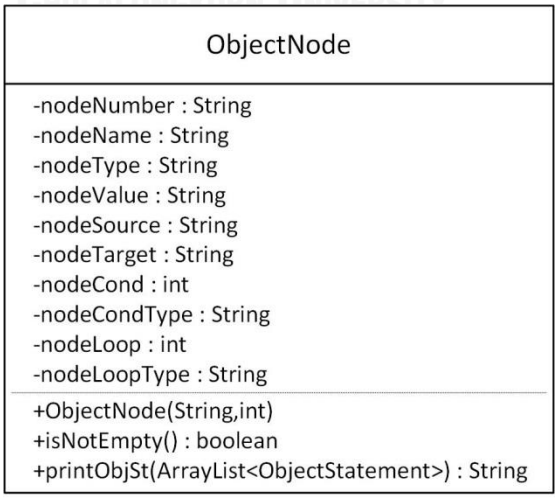
รูปที่ 3-35 คลาส Execute

21) คลาส EvaluateTest เป็นคลาสที่ทำหน้าที่ประเมินผลดำเนินการทดสอบแสดงดังรูปที่ 3-36



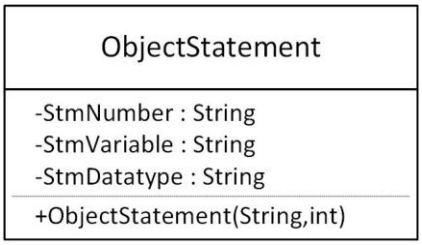
รูปที่ 3-36 คลาส EvaluateTest

22) คลาส ObjectNode เป็นคลาสที่ทำหน้าที่เป็นโครงสร้างโหนดเพื่อรับข้อมูลในการสกัดไฟล์เอกซ์เอ็มแอล ในอยู่ในรูปแบบของโหนด เพื่อใช้ในการสร้างกราฟการไหลของการควบคุม แสดงดังรูปที่ 3-37



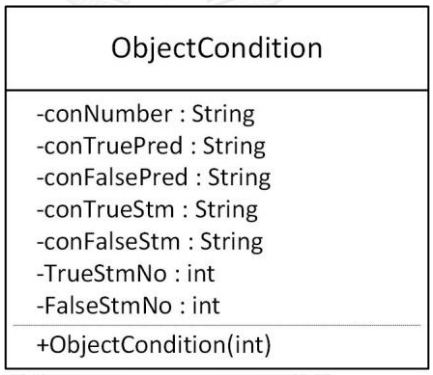
รูปที่ 3-37 คลาส ObjectNode

23) คลาส ObjectStatement เป็นคลาสที่ทำหน้าที่เป็นโครงสร้างตัวแปรเพื่อใช้เก็บตัวแปรและค่าของตัวแปรในแต่ละโหนดที่ถูกสกัดมา แสดงดังรูปที่ 3-38



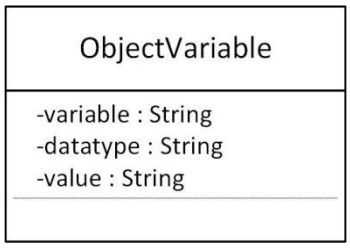
รูปที่ 3-38 คลาส ObjectStatement

24) คลาส ObjectCondition เป็นคลาสที่ทำหน้าที่เป็นโครงสร้างเงื่อนไขเพื่อใช้เก็บโหนดที่มีคำสั่งเงื่อนไข แสดงดังรูปที่ 3-39



รูปที่ 3-39 คลาส ObjectCondition

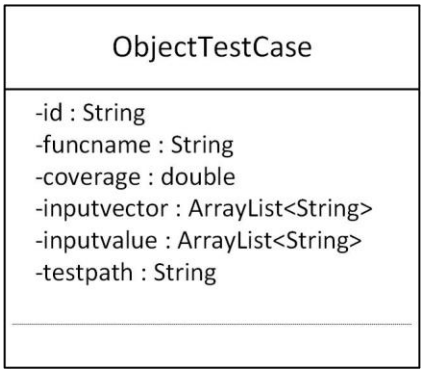
25) คลาส ObjectVariable เป็นคลาสที่ทำหน้าที่เป็นโครงสร้างตัวแปรเพื่อใช้เก็บตัวแปรและค่าของตัวแปรในแต่ละเวกเตอร์นำเข้า แสดงดังรูปที่ 3-40



รูปที่ 3-40 คลาส ObjectVariable

26) คลาส ObjectTestCase เป็นคลาสที่ทำหน้าที่เป็นโครงสร้างกรณีทดสอบเพื่อใช้เก็บค่าทั้งหมดที่เกี่ยวข้องกับกรณีทดสอบได้แก่ รหัสกรณีทดสอบ ฟังก์ชันของไฟล์จาวาสคริปต์ที่ใช้

ทดสอบ เปอร์เซ็นต์ความครอบคลุมประโยคคำสั่ง ค่าเวกเตอร์นำเข้าที่ถูกสร้างขึ้น ทางเดินที่ใช้ในการทดสอบ แสดงดังรูปที่ 3-41



รูปที่ 3-41 คลาส ObjectTestCase

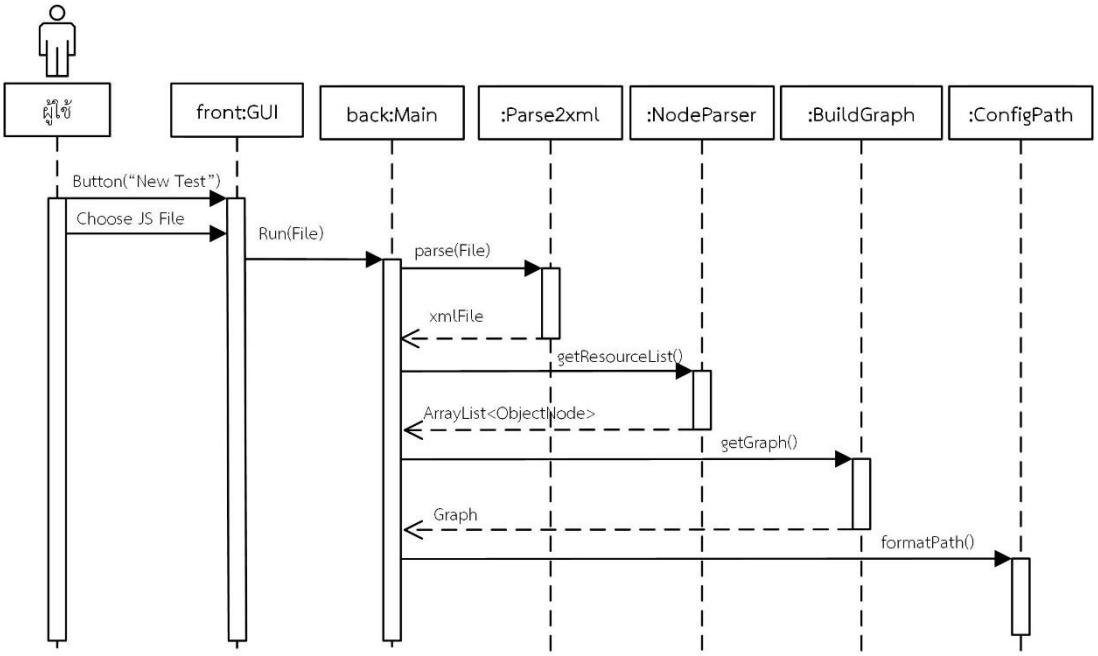
3.2.4 **แผนภาพลำดับ**

แผนภาพลำดับเป็นแผนภาพที่ใช้แสดงปฏิสัมพันธ์ระหว่างวัตถุต่างๆในระบบ แผนภาพลำดับที่อธิบายกิจกรรมหลักในการสร้างกรณีทดสอบและดำเนินการทดสอบของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง มีทั้งหมด 11 แผนภาพ คือ แผนภาพลำดับการสร้างการทดสอบ แสดงดังรูปที่ 3-42 แผนภาพลำดับการนำเข้าไฟล์จาวาสคริปต์ แสดงดังรูปที่ 3-43 แผนภาพลำดับการวิเคราะห์ไฟล์จาวาสคริปต์ แสดงดังรูปที่ 3-44 แผนภาพลำดับการสร้างกราฟการไหลของการควบคุม แสดงดังรูปที่ 3-45 แผนภาพลำดับการนำเข้าเลือกทางเดิน แสดงดังรูปที่ 3-46 แผนภาพลำดับสร้างกรณีทดสอบ แสดงดังรูปที่ 3-47 และรูปที่ 3-48 แผนภาพลำดับดำเนินการทดสอบ แสดงดังรูปที่ 3-49 แผนภาพลำดับแสดงรายงานการทดสอบ แสดงดังรูปที่ 3-50 แผนภาพลำดับการดูรายงานทดสอบ แสดงดังรูปที่ 3-51 แผนภาพลำดับการลบรายงานการทดสอบ แสดงดังรูปที่ 3-52 แผนภาพลำดับการส่งออกรายงานการทดสอบ แสดงดังรูปที่ 3-53

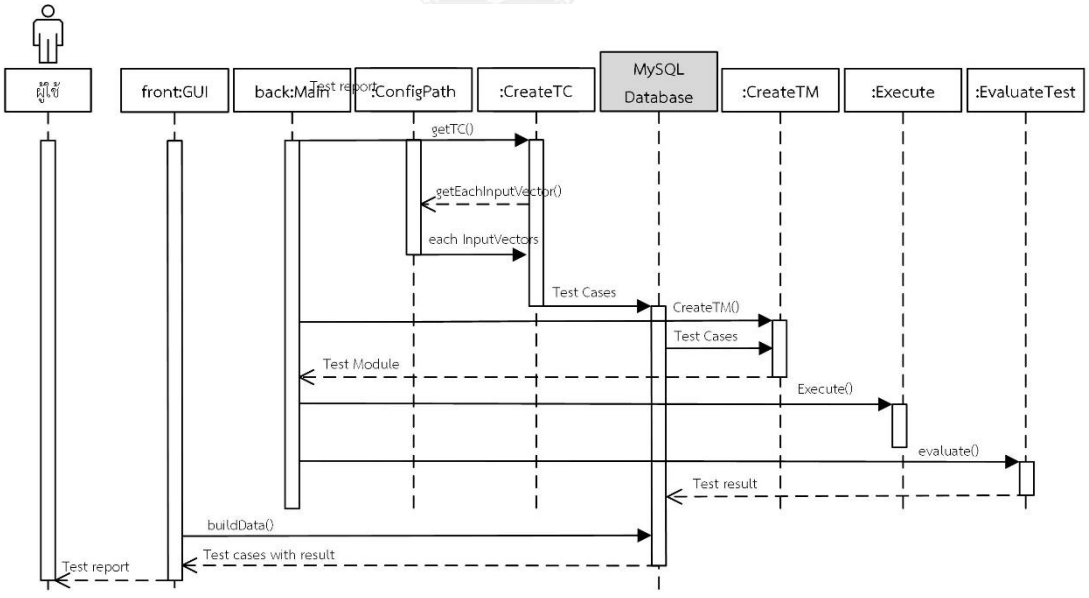
1) แผนภาพลำดับการสร้างการทดสอบ

จากรูปที่ 3-42 ผู้ใช้ต้องการเริ่มสร้างการทดสอบ ผู้ใช้จะต้องกดปุ่ม “New Test” และนำเข้าไฟล์จาวาสคริปต์เข้าเครื่องมือ เมื่อนำเข้าไฟล์จาวาสคริปต์แล้ว วัตถุ back:Main จะเป็นวัตถุส่วนกลางของเครื่องมือใช้เรียกการทำงานวัตถุต่างๆที่ใช้ในการสร้างการทดสอบไฟล์จาวาสคริปต์ที่นำเข้ามา โดยรูปที่ 3-43 แสดงแผนภาพลำดับที่เป็นภาพรวมในการรับส่งค่าของวัตถุ back:Main เพื่อสร้างการทดสอบเริ่มจากวัตถุ back:Main โดยเรียกทำงาน ของวัตถุ Parse2xml วัตถุ NodeParser วัตถุ BuildGraph วัตถุ ConfigPath วัตถุ CreateTC วัตถุ CreateTM วัตถุ Execute และวัตถุ EvaluateTest

ตามลำดับ เมื่อดำเนินการทดสอบเสร็จแล้ว วัตถุ front:GUI จะเรียกการทำงาน buildData() เพื่อดึงข้อมูลจากฐานข้อมูลในวัตถุ MySQL Database มาสร้างเป็นรายงานผลการทดสอบให้ผู้ใช้รับทราบ



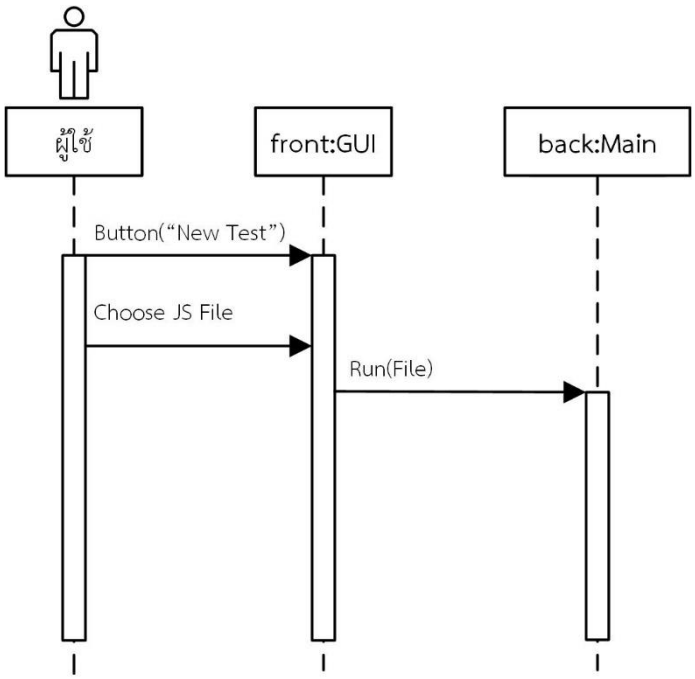
รูปที่ 3-42 แผนภาพลำดับการสร้างการทดสอบ



รูปที่ 3-42 แผนภาพลำดับการสร้างการทดสอบ (ต่อ)

2) แผนภาพลำดับการนำเข้าไฟล์จาวาสคริปต์

จากรูปที่ 3-43 เริ่มต้นด้วยผู้ใช้กดปุ่ม “NewTest” วัตถุ front:GUI ซึ่งเป็นส่วนต่อประสานผู้ใช้จะแสดงหน้าต่างเลือกไฟล์ ผู้ใช้เลือกไฟล์จาวาสคริปต์ที่ต้องการทดสอบ front:GUI จะเรียกการทำงาน run(File) ผ่านวัตถุ back:Main ซึ่งโปรแกรมหลักของเครื่องมือ และ back:Main จะเรียกการทำงานอื่นต่อไป สามารถอธิบายในแผนภาพลำดับถัดไป



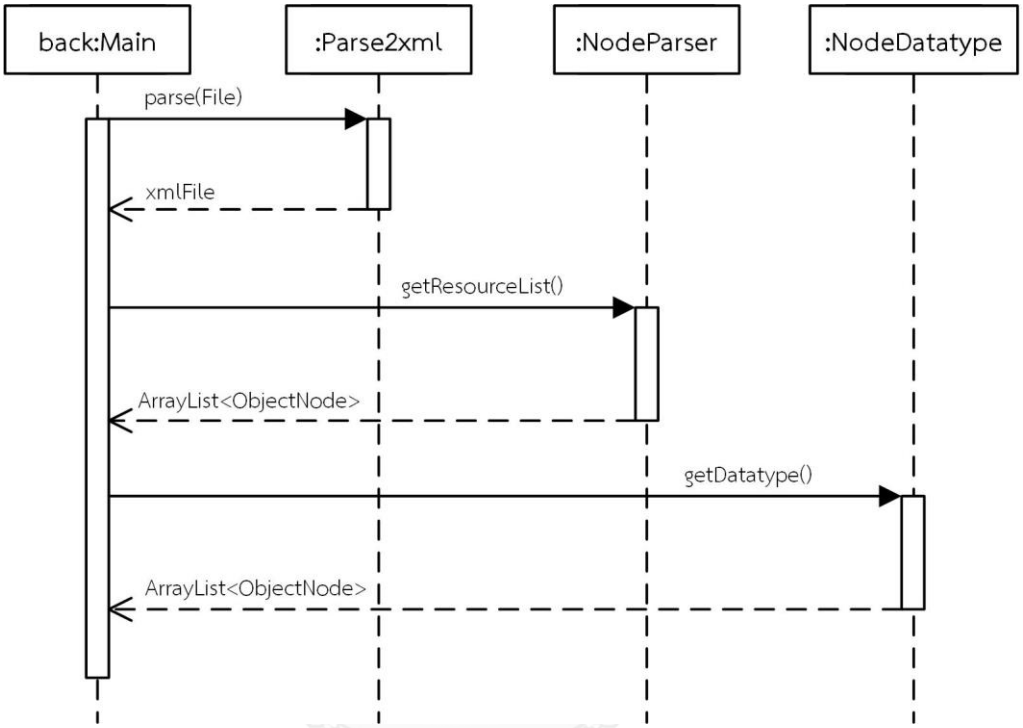
รูปที่ 3-43 แผนภาพลำดับการนำเข้าไฟล์จาวาสคริปต์

3) แผนภาพลำดับการวิเคราะห์ไฟล์จาวาสคริปต์

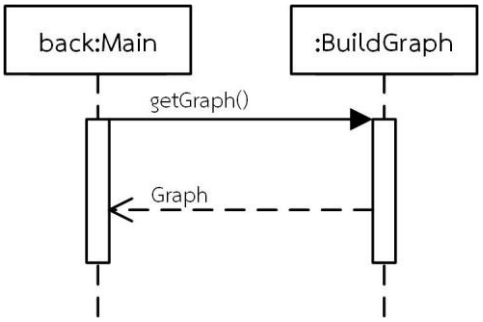
จากรูปที่ 3-44 เมื่อผู้ใช้นำเข้าไฟล์จาวาสคริปต์ที่ต้องการทดสอบลงใน back:Main แล้ว วัตถุ back:Main จะเรียกการทำงาน parse(File) จากวัตถุ Parse2xml และวัตถุ Parse2xml ดำเนินการแจงส่วนไฟล์จาวาสคริปต์และส่งผลลัพธ์กลับมาในรูปแบบไฟล์เอกซ์เอ็มแอล จากไฟล์เอกซ์เอ็มแอลที่ได้รับกลับมา วัตถุ back:Main จะเรียกการทำงาน getResourceList() จากวัตถุ NodeParser และวัตถุ NodeParser จะส่งโครงสร้างวัตถุโหนดที่วิเคราะห์โครงสร้างไฟล์เอกซ์เอ็มแอลแล้ว จากนั้นวัตถุ back:Main จะเรียกการทำงาน getDatatype() จากวัตถุ NodeDatatype เพื่อหาชนิดข้อมูลของแต่ละตัวแปรในโครงสร้างวัตถุโหนด

4) แผนภาพลำดับการสร้างกราฟการไหลของการควบคุม

จากรูปที่ 3-45 เมื่อเครื่องมือได้โครงสร้างวัตถุโหนดที่สมบูรณ์แล้ว วัตถุ back:Main จะเรียกการทำงาน getGraph() จากวัตถุ BuildGraph และวัตถุ BuildGraph จะดำเนินการเชื่อมโครงสร้างวัตถุโหนดให้เป็นกราฟการไหลของการควบคุม



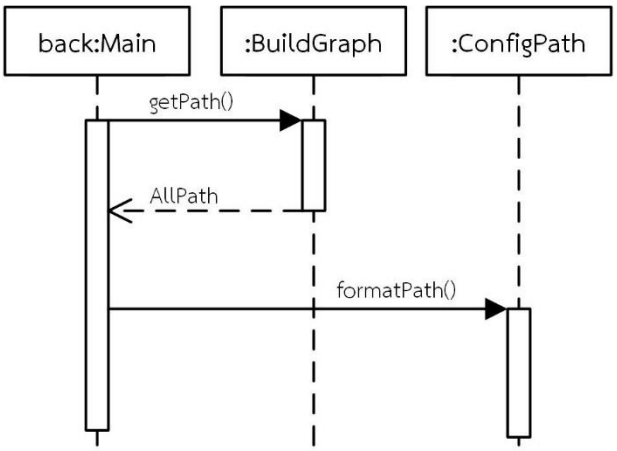
รูปที่ 3-44 แผนภาพลำดับการวิเคราะห์ไฟล์จาวาสคริปต์



รูปที่ 3-45 แผนภาพลำดับการสร้างกราฟการไหลของการควบคุม

5) แผนภาพลำดับการนำเข้าเลือกทางเดิน

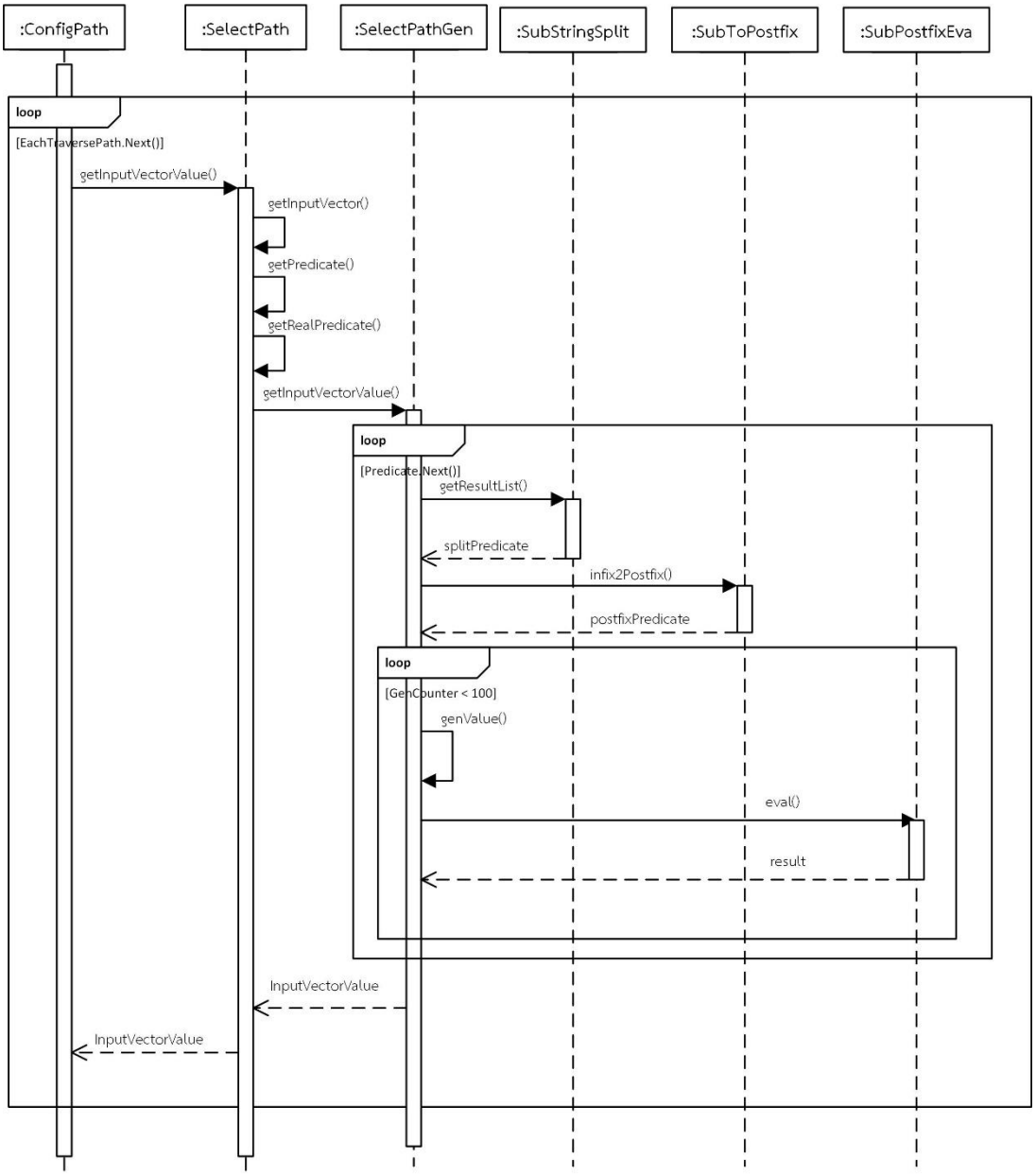
จากรูปที่ 3-46 เมื่อได้กราฟการไหลของการควบคุมแล้ว วัตถุ back:Main จะเรียกการทำงาน getPath() จากวัตถุ BuildGraph และวัตถุ BuildGraph จะท่องกราฟใช้วิธีการค้นหาแนวลึกก่อน เพื่อหาทางเดินทั้งหมด และส่งค่าทางเดินทั้งหมดกลับมาที่ วัตถุ back:Main จากนั้นวัตถุ back:Main จะเรียกการทำงาน formatPath() จากวัตถุ ConfigPath และวัตถุ ConfigPath จะเลือกทางเดินแต่ละทางเดินมาสร้างกรณีทดสอบในลำดับต่อไป



รูปที่ 3-46 แผนภาพลำดับการนำเข้าเลือกทางเดิน

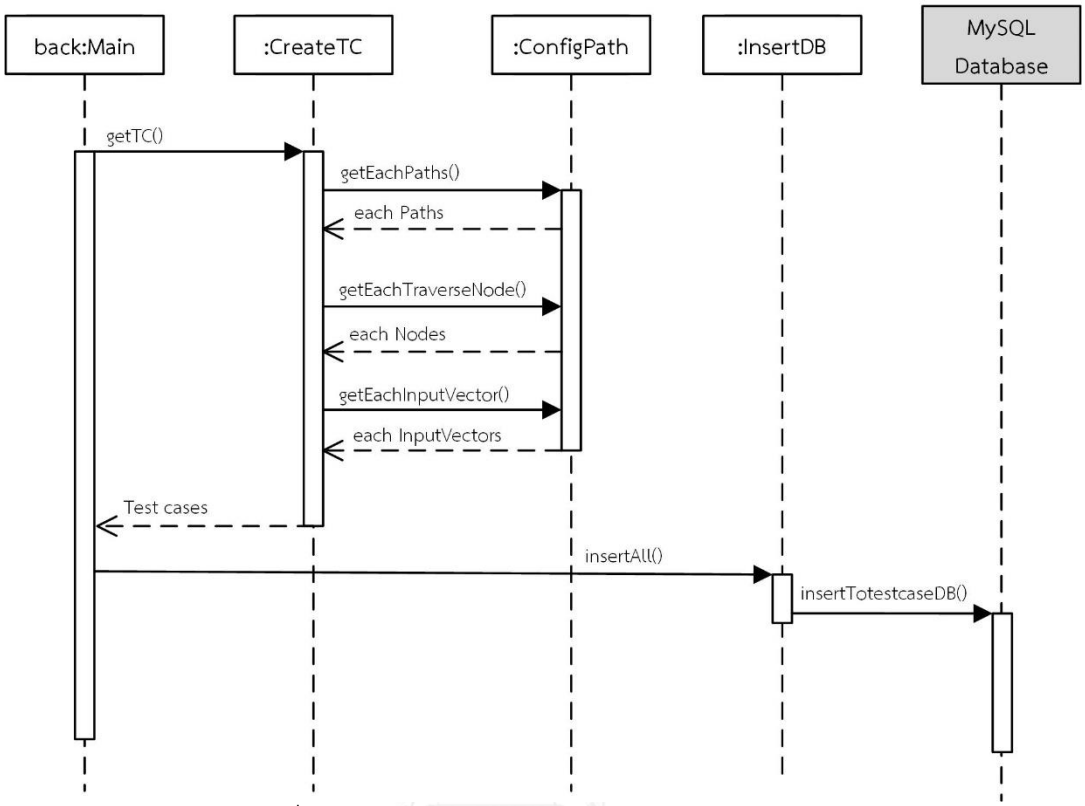
6) แผนภาพลำดับการสร้างกรณีทดสอบ

จากรูปที่ 3-47 วัตถุ ConfigPath จะเลือกทางเดินมาสร้างกรณีทดสอบ โดยเรียกการทำงาน getInputVectorValue() จากวัตถุ SelectPath จากนั้นวัตถุจะเรียกการทำงาน getInputVector() getPredicate() และ getRealPredicate() เพื่อสกัดเวกเตอร์นำเข้า เพรดิเคตที่เกี่ยวข้องที่ใช้ในการสร้างกรณีทดสอบและหาค่าเวกเตอร์นำเข้า จากนั้นจะนำค่าที่สกัดได้มาหาค่าเวกเตอร์นำเข้าโดยการเรียกการทำงาน getInputVectorValue() จากวัตถุ SelectPathGen จากนั้นวัตถุ SelectPathGen จะวิเคราะห์แต่ละเพรดิเคตในทางเดินที่ถูกเลือกมา แบ่งเพรดิเคตเป็นส่วน จากการทำงาน getResultList() จากวัตถุ SubStringSplit และทำเพรดิเคตให้อยู่ในรูป PostFix จากการทำงาน infix2Postfix() จากวัตถุ SubToPostfix จากนั้นวัตถุจะสร้างค่าเวกเตอร์นำเข้าให้กับเวกเตอร์นำเข้า และตรวจสอบการตัดสินใจของเพรดิเคตจากการใส่ค่าเวกเตอร์นำเข้าว่าตรงกับที่ตัดสินใจไว้ในทางเดินที่เลือกมาไว้หรือไม่ โดยเรียกการทำงาน eval() จากวัตถุ SubPostfixEva วัตถุ SelectPathGen จะสร้างค่าเวกเตอร์จนกว่าผ่านการตรวจสอบหรือสร้างค่าซ้ำจนครบ 100 ครั้ง จากนั้นวัตถุ SelecPathGen จะส่งค่าเวกเตอร์นำเข้า (InputVectorValue) คืนกลับ มาที่วัตถุ SelectPath และ วัตถุ ConfigPath ตามลำดับ



รูปที่ 3-47 แผนภาพลำดับการสร้างกรณีทดสอบ

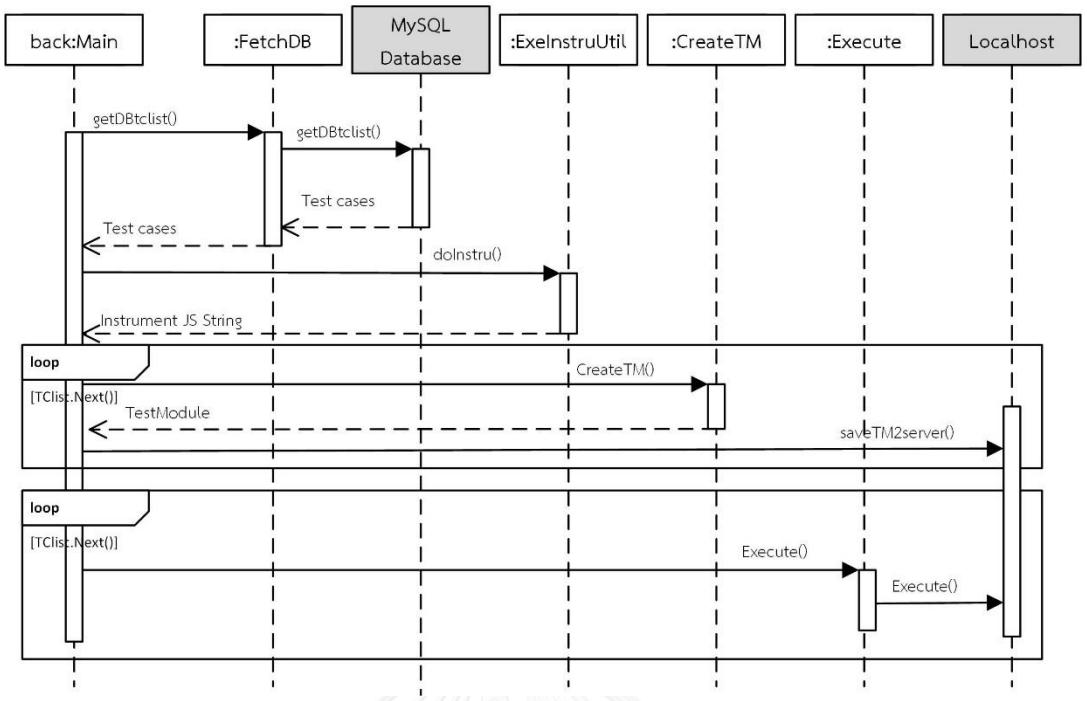
จากรูปที่ 3-48 เมื่อได้ค่าทั้งหมดที่ใช้สร้างกรณีทดสอบในวัตถุ ConfigPath แล้ว วัตถุ back:Main จะเรียกการทำงาน getTC() จากวัตถุ CreateTC โดยวัตถุ CreatTC จะเรียกการทำงาน getEachPath() getEachNodeTraverseNode() และ getEachInputVector() ในการดึงค่าทั้งหมดจากวัตถุ ConfigPath มาสร้างกรณีทดสอบ จากนั้นวัตถุ CreateTC จะคืนค่ากรณีทดสอบทั้งหมด (Test cases) กลับมาที่วัตถุ back:Main และวัตถุ back:Main จะเรียกการทำงาน insertAll() จากวัตถุ InsertDB และ วัตถุ InsertDB เรียกการทำงาน insertTotestcaseDB() เพื่อส่งออกกรณีทดสอบทั้งหมดเข้าสู่ฐานข้อมูลการทดสอบ



รูปที่ 3-48 แผนภาพลำดับการสร้างกรณีทดสอบ (ต่อ)

7) แผนภาพลำดับการดำเนินการทดสอบ

จากรูปที่ 3-49 เมื่อส่งออกกรณีทดสอบทั้งหมดเข้าสู่ฐานข้อมูลการทดสอบ แล้ววัตถุ back:Main จะค้นคืนกรณีทดสอบจากฐานข้อมูลการทดสอบ โดยเรียกการทำงาน getDBtclist() จากวัตถุ FetchDB จากนั้นวัตถุ back:Main จะแทรกประโยคคำสั่งตรวจวัดความครอบคลุมในไฟล์จาวาสคริปต์ที่นำเข้าในเครื่องมือ เพื่อจะตรวจสอบทางเดินขณะดำเนินการทดสอบจริง โดยเรียกการทำงาน doInstru() จาก วัตถุ ExeInstruUtil จากนั้นวัตถุ back:Main จะเรียกการทำงาน CreateTM() จากวัตถุ CreateTM ใช้ในการสร้างมอดูลทดสอบ และบันทึกลงเซิร์ฟเวอร์ Localhost จากนั้นวัตถุ back:Main เรียกการทำงาน Execute() วัตถุ Execute เพื่อสั่งดำเนินการทดสอบผ่านโปรแกรมดีโอเอชในเว็บบราวเซอร์



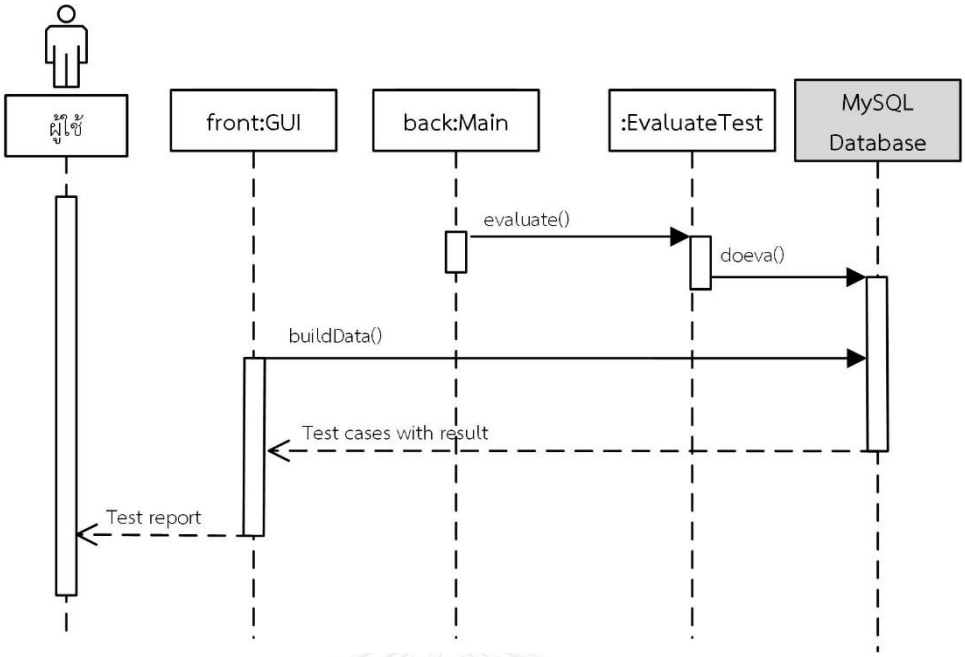
รูปที่ 3-49 แผนภาพลำดับการดำเนินการทดสอบ

8) แผนภาพลำดับการแสดงรายงานการทดสอบ

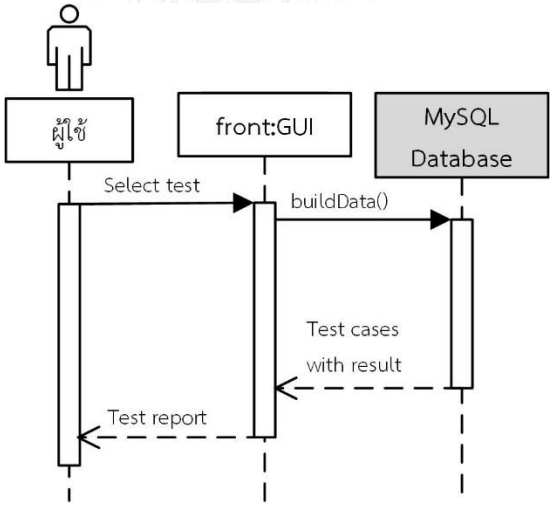
จากรูปที่ 3-50 เมื่อดำเนินการทดสอบเสร็จสิ้น วัตถุ back:Main จะเรียกการทำงาน evaluate() ผ่านวัตถุ EvaluateTest จากนั้นวัตถุ EvaluateTest จะเรียกการทำงาน doeva() เพื่อประเมินผลการทดสอบแต่ละมอดูลทดสอบโดยเปรียบทางเดินที่ได้จากการดำเนินการทดสอบกับทางเดินที่ได้จากการสร้างกรณีทดสอบ เมื่อได้ผลการทดสอบแล้วจะส่งผลไปที่ฐานข้อมูลการทดสอบ จากนั้นวัตถุ front:GUI จะค้นคืนกรณีทดสอบและผลการทดสอบผ่านการทำงาน buildData() สร้างตารางรายงานการทดสอบและแสดงรายงานการทดสอบให้ผู้ใช้รับทราบ

9) แผนภาพลำดับการดูรายงานการทดสอบ

จากรูปที่ 3-51 เมื่อผู้ใช้ต้องการดูรายงานการทดสอบที่ได้สร้างการทดสอบไปแล้ว ผู้ใช้สามารถเลือกการทดสอบต้องการ ในรายการบันทึกผลการทดสอบ จากนั้น วัตถุ front:GUI ที่เป็นส่วนต่อประสานผู้ใช้จะเรียกการทำงาน buildGraph() เพื่อค้นคืนกรณีทดสอบและผลการทดสอบจากฐานข้อมูลการทดสอบ เมื่อวัตถุ front:GUI ได้รับผลลัพธ์จากฐานข้อมูลแล้ว จะสร้างตารางรายงานการทดสอบและแสดงผลให้ผู้ใช้รับทราบ



รูปที่ 3-50 แผนภาพลำดับการแสดงรายงานการทดสอบ

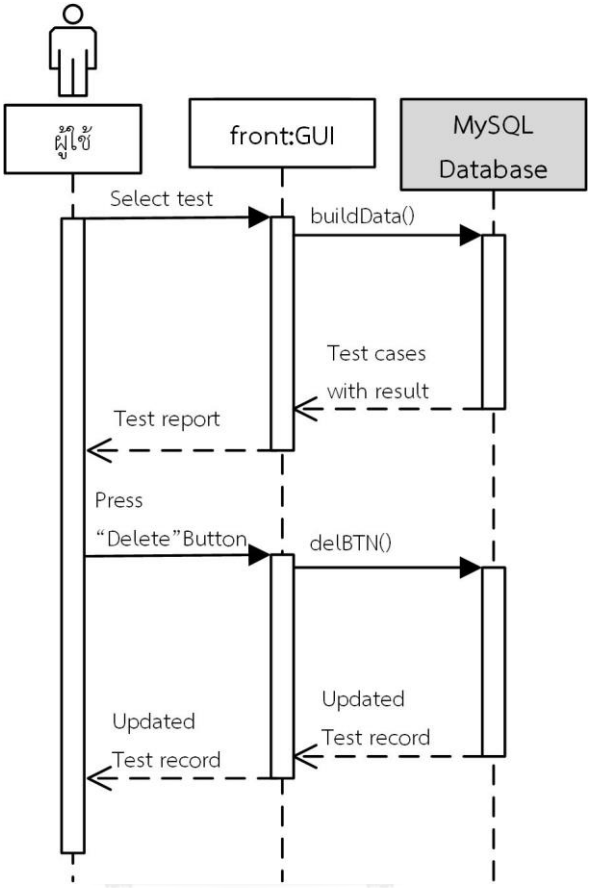


รูปที่ 3-51 แผนภาพลำดับการดูรายงานการทดสอบ

10) แผนภาพลำดับการลบรายงานการทดสอบ

จากรูปที่ 3-52 เมื่อผู้ใช้ต้องการจะลบรายงานการทดสอบ ผู้ใช้ต้องเลือกการทดสอบที่ต้องการลบ วัตถุ front:GUI จะแสดงรายงานการทดสอบนั้น จากขั้นตอนการทำงานในแผนภาพลำดับการดูการทดสอบ จากนั้นผู้ใช้ต้องกดปุ่ม Delete เพื่อสั่งให้เครื่องมือลบรายงานการทดสอบที่ถูกเลือก วัตถุ

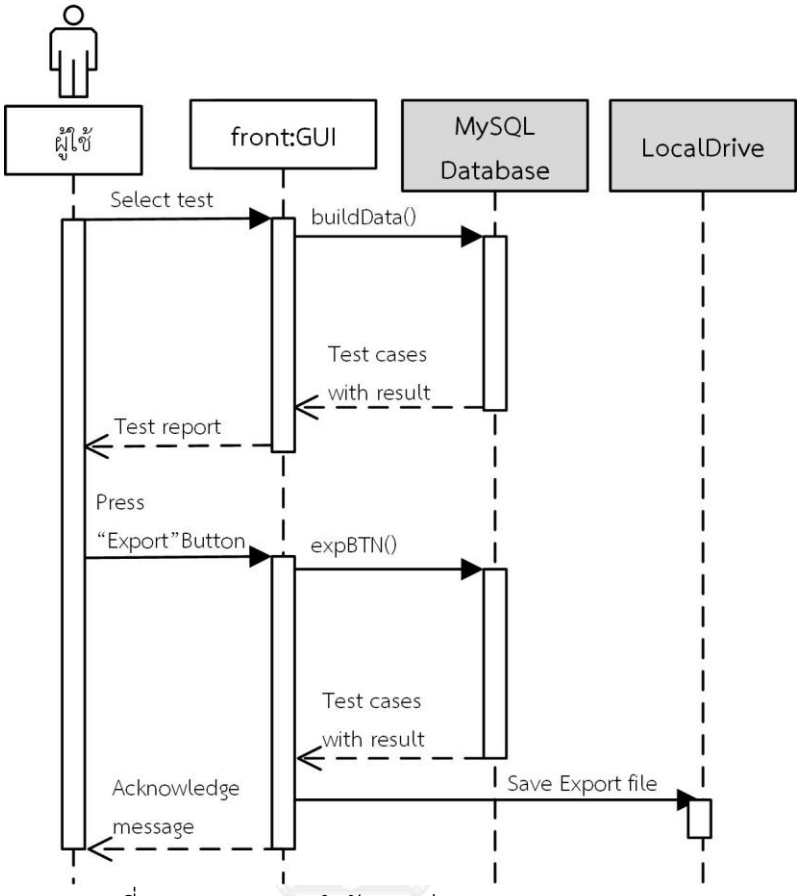
front:GUI จะเรียนการทำงาน delBTN() เพื่อสร้างคำสั่งลบข้อมูลการทดลองจากฐานข้อมูล เมื่อลบการทดสอบเสร็จแล้ว front:GUI จะปรับปรุงรายการบันทึกผลการทดสอบ



รูปที่ 3-52 แผนภาพลำดับการลบรายงานการทดสอบ

11) แผนภาพลำดับการส่งออกรายงานการทดสอบ

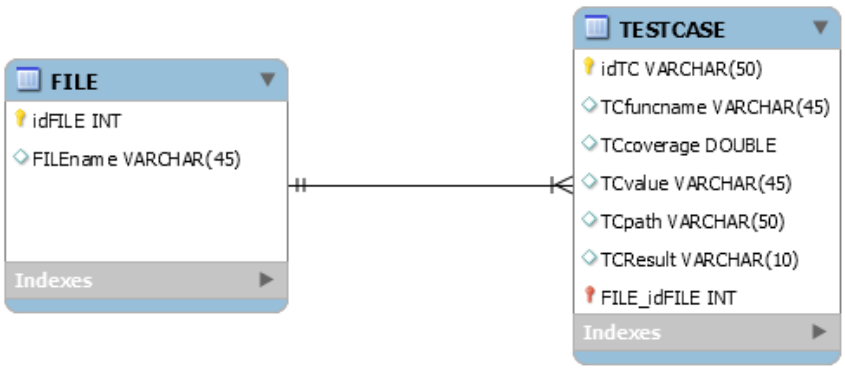
จากรูปที่ 3-53 เมื่อผู้ใช้ต้องการจะส่งออกรายงานการทดสอบ ผู้ใช้ต้องเลือกการทดสอบที่ต้องการส่งออก วัตถุ front:GUI จะแสดงรายงานการทดสอบนั้น จากขั้นตอนการทำงานในแผนภาพลำดับการดูการทดสอบ จากนั้นผู้ใช้ต้องกดปุ่ม Export เพื่อสั่งให้เครื่องมือส่งออกผลการทดสอบที่ถูกเลือก วัตถุ front:GUI จะเรียนการทำงาน expBTN() เพื่อสร้างคำสั่งส่งออกข้อมูลการทดลอง โดยค้น-คืนรายงานการทดสอบที่ต้องการจากฐานข้อมูล และบันทึกรายงานการทดสอบเป็นไฟล์โดยใช้รูปแบบไฟล์นามสกุลเอกซ์แอลเอส (*.xls) ลงในไดร์ฟที่เครื่องมือทำงาน จากนั้นเครื่องมือจะส่งข้อมูลให้ผู้ใช้งานรับทราบ



รูปที่ 3-53 แผนภาพลำดับการส่งออกรายงานการทดสอบ

3.2.5 **โครงสร้างของฐานข้อมูล**

โครงสร้างฐานข้อมูลของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง แสดงโดยแผนภาพความสัมพันธ์เอนทิตี (ER diagram) ได้รูปที่ 3-54



รูปที่ 3-54 แผนภาพความสัมพันธ์เอนทิตีของโครงสร้างฐานข้อมูลเครื่องมือ

จากแผนภาพความสัมพันธ์เอนทิตี ในรูปที่ 3-54 จะมีตารางทั้งหมด 2 ตาราง สามารถแสดงรายละเอียดเบื้องต้นในแต่ละตารางด้านล่างนี้ โดยรายละเอียดในแต่ละตารางจะถูกอธิบายอย่างละเอียดในพจนานุกรมในภาคผนวก ก

1) ตาราง FILE เป็นตารางที่ใช้เก็บชื่อไฟล์จาวาสคริปต์ ที่จะใช้แสดงรายการการทดสอบในบันทึกผลการทดสอบ
2) ตาราง TESTCASE เป็นตารางที่ใช้เก็บรายละเอียดข้อมูลกรณีทดสอบที่ถูกสร้างขึ้น เป็นตารางข้อมูลหลักที่ใช้การค้นคืนกรณีทดสอบ เพื่อสร้างมอดูลทดสอบ บันทึกผลการทดสอบ และแสดงรายงานการทดสอบ

บทที่ 4
การพัฒนาเครื่องมือ

ในบทการพัฒนาเครื่องมือนี้จะกล่าวถึงสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ รวมทั้งโครงสร้างส่วนต่อประสานกับผู้ทดสอบของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง ซึ่งมีรายละเอียดดังต่อไปนี้

4.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ แบ่งได้เป็น 2 ส่วนได้แก่ ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software) ซึ่งมีรายละเอียดต่างๆดังต่อไปนี้

4.1.1 **ฮาร์ดแวร์**

1) เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)

   หน่วยประมวลผล (CPU) อินเทลคอร์ทูดูโอ (Intel Core 2 Duo 2.53 GHz)

2) ฮาร์ดดิสก์ (Harddisk) 250 กิกะไบต์ (250 GB)

3) หน่วยความจำสำรอง (RAM) 4 กิกะไบต์ (4 GB)

4.1.2 **ซอฟต์แวร์**

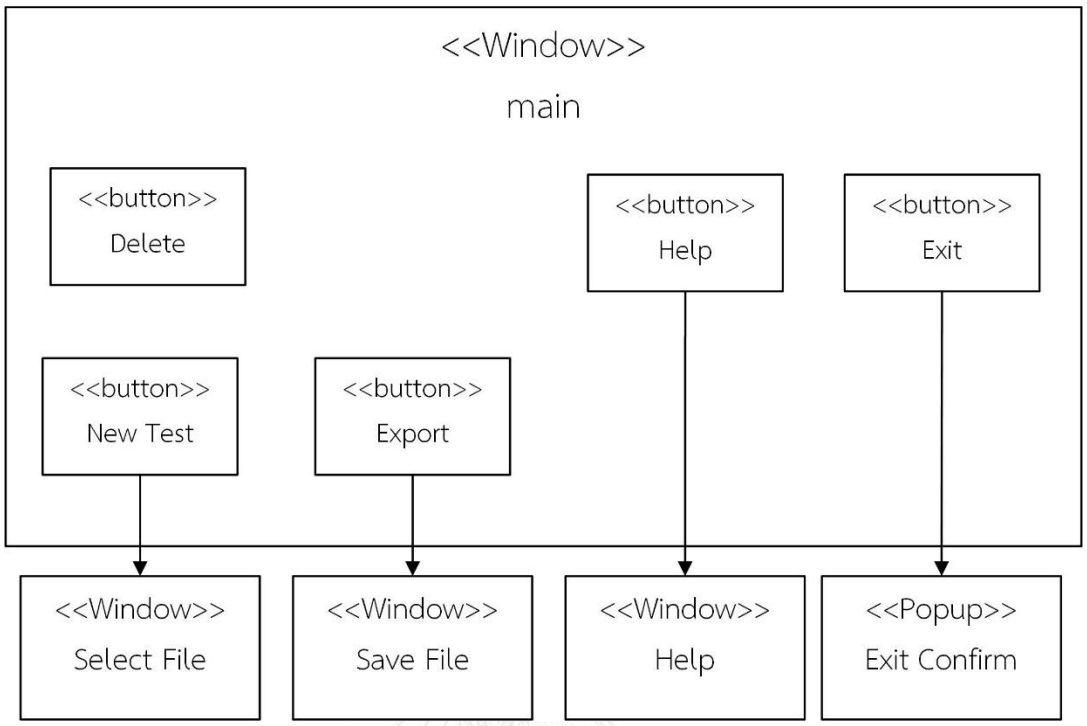
1) ระบบปฏิบัติการ (Operating System) ไมโครซอฟต์วินโดว์ เอท (Microsoft Window 8)

2) ซอฟต์แวร์เขียนโปรแกรม Eclipse Mars Release (4.5.0)

3) ฐานข้อมูล MySQL เวอร์ชัน 5.617 (MySQL 5.6.17)

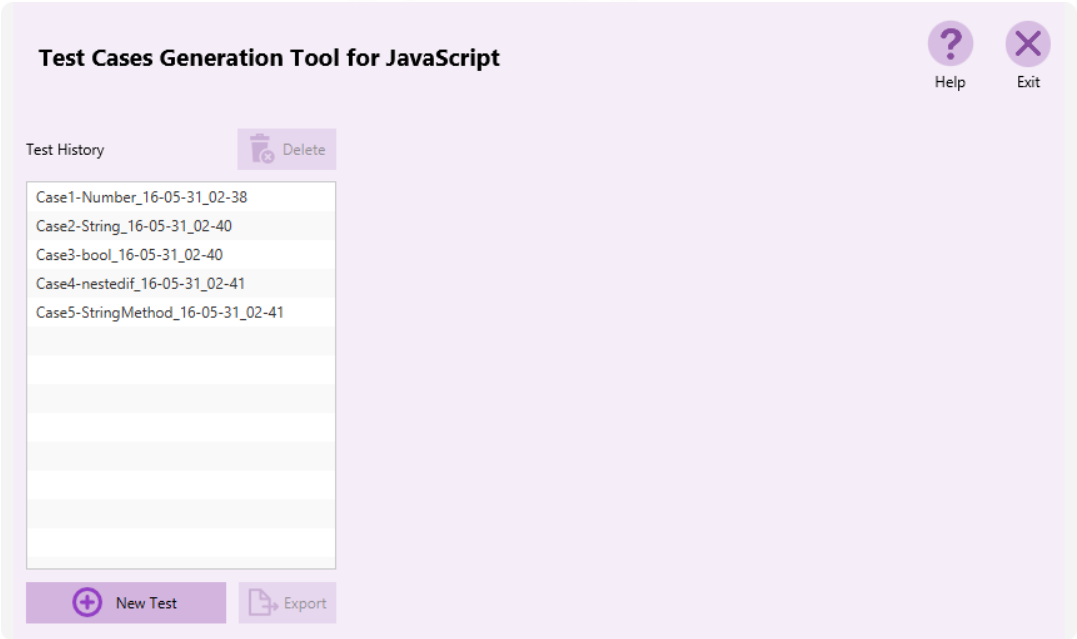
4.2 โครงสร้างส่วนต่อประสานกับผู้ใช้ของเครื่องมือ

โครงสร้างส่วนต่อประสานของเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง จะถูกอธิบายด้วยแผนภาพวินโดวส์เนวิเกชัน (Windows navigation diagram) ซึ่งใช้อธิบายถึงความสัมพันธ์ต่างๆของส่วนต่อประสานทั้งหมดในโปรแกรมแบ่งตามหน้าที่ต่างๆดังรูปที่ 4-1



รูปที่ 4-1 แผนภาพวินโดวส์เนวิเกชันของเครื่องมือ

จากรูปที่ 4-1 แผนภาพวินโดวส์เนวิเกชันนี้ ได้แสดงส่วนประกอบส่วนต่อประสานของเครื่องมือ ซึ่งประกอบไปด้วย หน้าต่างและปุ่ม โดยแต่ละส่วนประกอบจะมีรายละเอียดดังต่อไปนี้



รูปที่ 4-2 หน้าต่าง Main

1) หน้าต่าง Main เป็นหน้าต่างหลักของเครื่องมือ ซึ่งประกอบไปด้วย 5 ปุ่มหลัก ได้แก่ ปุ่ม New Test ปุ่ม Delete ปุ่ม Help และปุ่ม Exit สามารถแสดงดังรูปที่ 4-2

   ปุ่มที่ปรากฏในหน้าต่าง Main มีหน้าที่ดังต่อไปนี้

   1.1) ปุ่ม New Test เมื่อกดปุ่มนี้ จะแสดงหน้าต่างให้เลือกไฟล์จาวาสคริปต์สำหรับสร้างการทดสอบใหม่

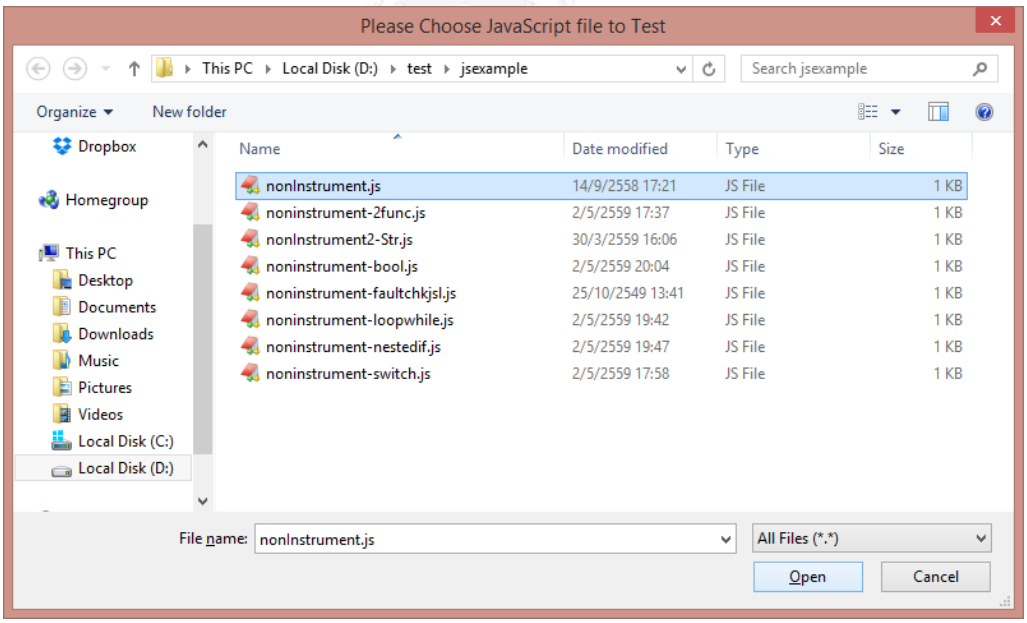
   1.2) ปุ่ม Export เมื่อกดปุ่มนี้ จะแสดงหน้าต่างให้บันทึกไฟล์นามสกุลเอกซ์แอลเอสเอกซ์ (*.xlsx)

   1.3) ปุ่ม Delete เมื่อกดปุ่มนี้ จะลบการทดสอบที่ถูกเลือกในรายการ Test record

   1.4) ปุ่ม Help เมื่อกดปุ่มนี้ จะแสดงหน้าต่าง Help ที่แสดงวิธีการใช้งานเครื่องมือ

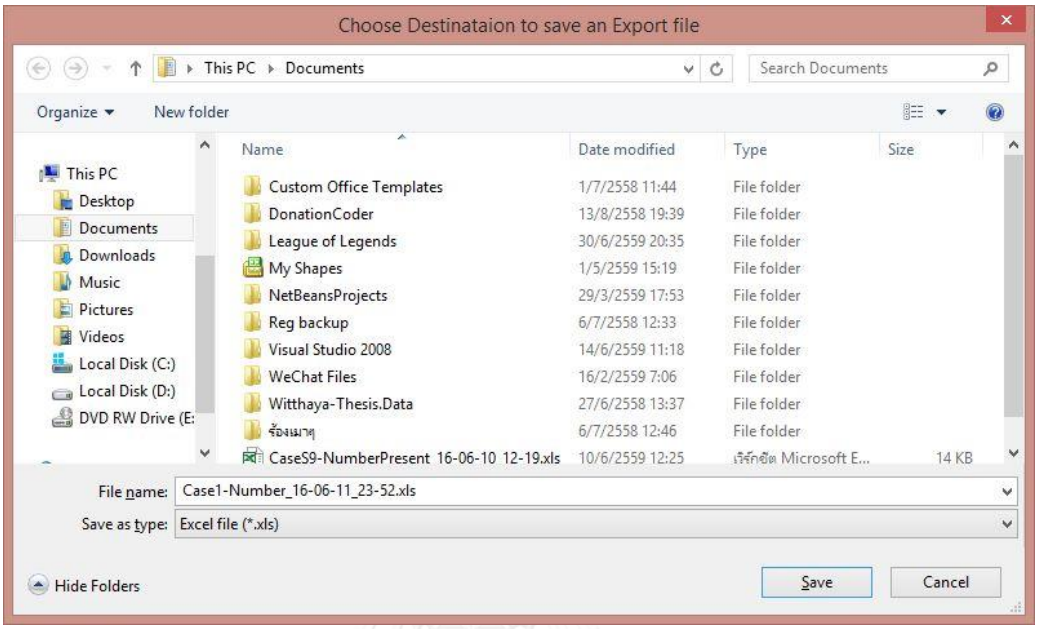
   1.5) ปุ่ม Exit เมื่อกดปุ่มนี้ จะออกจากหน้าต่าง Main โดยจะมีหน้าต่างป๊อบอัพเพื่อยืนยันการออกจากโปรแกรมเครื่องมือ

2) หน้าต่าง Select File เป็นหน้าต่างเลือกไฟล์ เพื่อนำเข้าไฟล์จาวาสคริปต์เข้าเครื่องมือ ดังรูปที่ 4-3



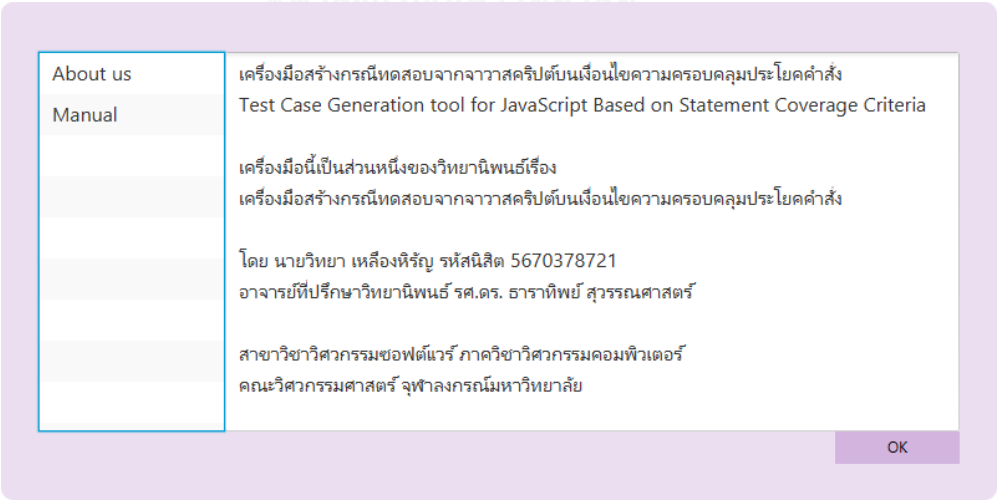
รูปที่ 4-3 หน้าต่าง Select File

3) หน้าต่าง Export File เป็นหน้าต่างส่งออกไฟล์รายงานการทดสอบ ให้อยู่ในรูปแบบไฟล์เอกซ์แอลเอสเอกซ์ (.xlsx) ดังรูปที่ 4-4



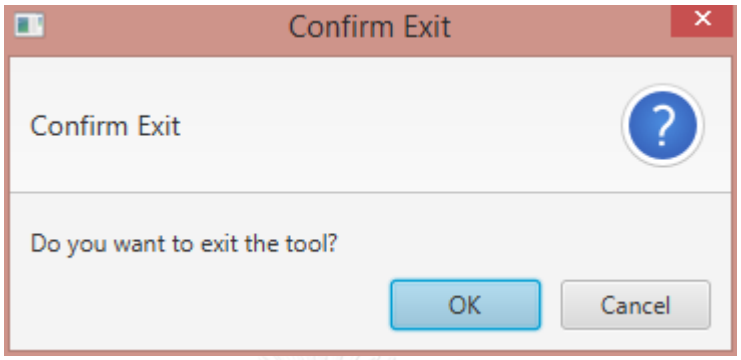
รูปที่ 4-4 หน้าต่าง Export file

4) หน้าต่าง Help ให้ความช่วยเหลือ โดยจะแสดงวิธีการใช้งานเครื่องมือ และรายละเอียดเกี่ยวกับผู้สร้างเครื่องมือ ดังรูปที่ 4-5



รูปที่ 4-5 หน้าต่าง Help

5) หน้าต่างป๊อบอัพ Exit Confirm เป็นหน้าต่างยืนยันผู้ใช้ก่อนออกจากโปรแกรมเครื่องมือ รูปที่ 4-6



รูปที่ 4-6 หน้าต่างป๊อบอัพ Exit Confirm

บทที่ 5
การทดสอบเครื่องมือ

หัวข้อนี้กล่าวถึงการทดสอบเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง เพื่อทดสอบความถูกต้อง โดยจะกล่าวถึงสภาพแวดล้อมที่ใช้ ในการทดสอบการทำงานของเครื่องมือ รายละเอียดของการทดสอบเครื่องมือ และผลการทดสอบเครื่องมือ โดยมีรายละเอียดการทดสอบดังต่อไปนี้

5.1 สภาพแวดล้อมที่ใช้ทดสอบ

สภาพแวดล้อมที่ใช้ทดสอบเครื่องมือ แบ่งได้เป็น 2 ส่วนได้แก่ ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software) ซึ่งมีรายละเอียดต่างๆดังต่อไปนี้

5.1.1 ฮาร์ดแวร์

1) เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)

   หน่วยประมวลผล (CPU) อินเทลคอร์ทูดูโอ (Intel Core 2 Duo 2.53 GHz)

2) ฮาร์ดดิสก์ (Harddisk) 250 กิกะไบต์ (250 GB)

3) หน่วยความจำสำรอง (RAM) 4 กิกะไบต์ (4 GB)

5.1.2 ซอฟต์แวร์

1) ระบบปฏิบัติการ (Operating System) ไมโครซอฟต์วินโดว์ เอท (Microsoft Window 8)

2) ซอฟต์แวร์เขียนโปรแกรม Eclipse Mars Release (4.5.0)

3) ฐานข้อมูล MySQL เวอร์ชัน 5.617 (MySQL 5.6.17)

5.2 การทดสอบเครื่องมือ

การทดสอบเครื่องมือเป็นสิ่งจำเป็นที่ต้องทำ เพื่อตรวจสอบความถูกต้องของการทำงานของเครื่องมือว่ามีความสามารถสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง โดยการทดสอบเครื่องมือนี้จะทดสอบกับไฟล์จาวาสคริปต์ตัวอย่าง 5 ไฟล์ ได้แก่ ไฟล์ Case1-Number.js ไฟล์Case2-String.js ไฟล์Case3-bool.js ไฟล์Case4-nestedif.js และไฟล์ Case5-StringMethod.js โดยแต่ละไฟล์มีรายละเอียดดังต่อไปนี้

1) ไฟล์ Case1-Number.js จะใช้ทดสอบฟังก์ชันที่มีชนิดข้อมูลของพารามิเตอร์ที่เป็นตัวเลข สามารถแสดงซอร์สโค้ดในรูปที่ 5-1

```
1     function main(a,b){
2           var x = a + b;
3           var y = a - b;
4           if (x > y){
5                 z = x + y;
6           } else {
7           z = x - y;
8           }
9     }
```

รูปที่ 5-1 ซอร์สโค้ดไฟล์ Case1-Number.js

2) ไฟล์ Case2-String.js จะใช้ทดสอบฟังก์ชันที่มีชนิดข้อมูลของพารามิเตอร์ที่เป็นสตริง สามารถแสดงซอร์สโค้ดในรูปที่ 5-2

```
1     function String2(a){
2       if (a=="aaaaa"){
3          console.log(true);
4       }else{
5          console.log(false);
6       }
7     }
```

รูปที่ 5-2 ซอร์สโค้ดไฟล์ Case2-String.js

3) ไฟล์ Case3-bool.js จะใช้ทดสอบฟังก์ชันที่มีชนิดข้อมูลของพารามิเตอร์ที่เป็นบูลีน สามารถแสดงซอร์สโค้ดในรูปที่ 5-3

```
1  function boolchk (boo,far) {
2     if (boo === true) {
3           console.log (true);
4     } else {
5           console.log(false);
6     }
7     if (boo === false &&far === false )     {
8           console.log ("boo & far = false" );
9           console.log ("boo & far = false" );
10          console.log ("boo & far = false" );
11    } else if (boo === true &&far === false ) {
12          console.log ("boo = true , far = false" );
13          console.log ("boo = true , far = false" );
14          console.log ("boo = true , far = false" );
15    } else {
16          console.log (“boo = false or far = true");
17    }
18 }
```

รูปที่ 5-3 ซอร์สโค้ดไฟล์ Case3-bool.js

4) ไฟล์ Case4-nestedif.js จะใช้ทดสอบฟังก์ชันที่มีเงื่อนไข ประกอบด้วย 4 เพรดิเคต และชนิดข้อมูลของพารามิเตอร์ที่เป็นตัวเลข สามารถแสดงซอร์สโค้ดในรูปที่ 5-4

```
1  function nestedif (Type,PageCount){
2     if ((Type < 2 && PageCount != 6) ||
          (Type > 2 && PageCount < 2)) {
3            z = 0;
4     }else if ((Type < 3 && PageCount <= 4) ||
                (Type < 3 && PageCount > 5)){
5                    z = 5;
6                } else{
7                    z = 2;
8                }
9  }
```

รูปที่ 5-4 ซอร์สโค้ดไฟล์ Case4-nestedif.js

5) ไฟล์ Case5-StringMethod.js จะใช้ทดสอบฟังก์ชันที่มีชนิดข้อมูลของพารามิเตอร์ที่เป็นสตริงและมีการเรียกใช้เมท็อดของสตริง สามารถแสดงซอร์สโค้ดในรูปที่ 5-5

```
1  function stringMethod (line){
2     if(line.length <= 100){
3           console.log("good");
4     }
5     if(line.indexOf("steve") <= 100){
6           console.log("good");
8     }
7     if(line.search("Alan") <= 100){
9           console.log("good");
10    }
11 }
```

รูปที่ 5-5 ซอร์สโค้ดไฟล์ Case5-StringMethod.js

5.3 ผลการทดสอบ

จากการทดสอบเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่งโดยทดสอบไฟล์จาวาสคริปต์ตัวอย่างจำนวน 5 ไฟล์ ผลลัพธ์ของการทดสอบจะแสดงอยู่ในรูปแบบรายงานการทดสอบซึ่งรายงานการทดสอบไฟล์จาวาสคริปต์ทั้งหมดจะแสดงไว้ในภาคผนวก ข และผลการทดสอบจากการทดสอบไฟล์จาวาสคริปต์ตัวอย่าง สามารถแสดงได้ดังตารางที่ 5-1

ตารางที่ 5-1 ผลการทดสอบเครื่องมือกับไฟล์จาวาสคริปต์ตัวอย่าง

ไฟล์จาวาสคริปต์ที่ทดสอบ	จำนวนกรณีทดสอบ	ความครอบคลุมประโยคคำสั่ง (ร้อยละ)
Case1-Number.js	2	100
Case2-String.js	2	100
Case3-bool.js	4	100
Case4-nestedif.js	3	100
Case5-StringMethod.js	1	100

5.4 สรุปผลการทดสอบ

จากผลการทดสอบเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่งในตารางที่ 5-1 สามารถสรุปผลการทดลองได้ว่าเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่งสามารถสร้างกรณีทดสอบและดำเนินการทดสอบไฟล์จาวาสคริปต์ตัวอย่างให้ครอบคลุมทุกคำสั่งได้

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

จากการวิเคราะห์ ศึกษา วิจัยและพัฒนาเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง สามารถสรุปผลการวิจัย ข้อจำกัดขอบเขตความสามารถของเครื่องมือและแนวทางในการพัฒนาต่อในอนาคต มีรายละเอียดดังต่อไปนี้

6.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอเครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง โดยเครื่องมือนี้จะรองรับการทดสอบไฟล์จาวาสคริปต์ที่มีหนึ่งฟังก์ชันและไม่มีการเรียกใช้งานฟังก์ชันอื่นทั้งภายในและภายนอกไฟล์จาวาสคริปต์ ซึ่งเป็นการทดสอบไฟล์จาวาสคริปต์ในระดับหน่วย

เครื่องมือนี้สามารถวิเคราะห์โครงสร้างไฟล์จาวาสคริปต์ สร้างกรณีทดสอบให้ครอบคลุมประโยคคำสั่ง และดำเนินการทดสอบไฟล์จาวาสคริปต์ โดยแสดงผลเป็นรายงานการทดสอบซึ่งมีรายละเอียดของแต่ละกรณีทดสอบ และค่าความครอบคลุมประโยคคำสั่ง

นอกจากนี้ การใช้เครื่องมือนี้สามารถช่วยลดเวลาในการสร้างกรณีทดสอบและสร้างมอดูลทดสอบเพื่อใช้ในดำเนินการทดสอบไฟล์จาวาสคริปต์อีกด้วย

6.2 ข้อจำกัดของเครื่องมือ

เครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง มีข้อจำกัดดังต่อไปนี้

1) เครื่องมือรองรับการทดสอบไฟล์จาวาสคริปต์ 1 ไฟล์ต่อครั้งเท่านั้น
2) เครื่องมือไม่รองรับไฟล์จาวาสคริปต์ที่มีฟังก์ชันมากกว่า 1 ฟังก์ชัน
3) เครื่องมือไม่รองรับไฟล์จาวาสคริปต์ที่มีฟังก์ชันที่มีตัวแปรภายในเรียกฟังก์ชันอื่น
4) เครื่องมือไม่รองรับการทดสอบที่มีคำสั่งเงื่อนไขซ้อนกันมากกว่า 4 ชั้น
5) เครื่องมือไม่รองรับการทดสอบที่มีเพรดิเคตในเงื่อนไขมากกว่า 4 เพรดิเคต
6) เครื่องมือไม่รองรับการทดสอบที่มีคำสั่งวงวน (Loop) ซ้อนกันมากกว่า 1 ชั้น

7) เครื่องมือไม่รองรับฟังก์ชันจาวาสคริปต์ที่มีพารามิเตอร์ที่มีชนิดข้อมูลเป็นอ็อบเจกต์ (Object) สัญลักษณ์ (Symbol) แถวลำดับ (Array) และค่าอนิยาม (Undefined)

6.3 แนวทางการพัฒนาต่อ

เครื่องมือสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง สามารถพัฒนาต่อ ในหลายประเด็นได้ดังต่อไปนี้

1) พัฒนาเครื่องมือให้สามารถรองรับชนิดข้อมูลทุกแบบในภาษาจาวาสคริปต์
2) พัฒนาเครื่องมือให้สามารถรองรับการทดสอบไฟล์จาวาสคริปต์หลายไฟล์ต่อครั้ง
3) จากแนวคิดในการสร้างกรณีทดสอบ พัฒนาเครื่องมือให้สามารถรองรับการสร้างกรณีทดสอบในโปรแกรมที่มีภาษาโปรแกรมอื่นๆได้
4) พัฒนาเครื่องมือให้สามารถรองรับความซับซ้อนของคำสั่งเงื่อนไขและคำสั่งวงวนที่เพิ่มขึ้นได้
5) พัฒนาเครื่องมือให้สามารถรองรับการทดสอบแบบบูรณาการ คือการทดสอบไฟล์จาวา-สคริปต์ที่สามารถเรียกฟังก์ชันอื่นได้

รายการอ้างอิง

[1] ปานนวัต จานทอง, "เครื่องมือสร้างโมดูลทดสอบสำหรับจาวาสคริปต์บนเงื่อนไขความครอบคลุมคำสั่ง," วิทยานิพนธ์ปริญญามหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์, จุฬาลงกรณ์มหาวิทยาลัย, 2556.

[2] M. d. Kunder. (Apr). *"The size of the World Wide Web"* [Online]. Available: http://worldwidewebsize.com

[3] S. Gude, M. Hafiz, and A. Wirfs-Brock, "JavaScript: The Used Parts," in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*, 2014, pp. 466-475.

[4] B. Beizer, *Software Testing Techniques*, 2nd ed.: Van Nostrand Reinhold, 1990.

[5] K. Naik and P. Tripathy, *Software Testing and Quality Assurance Theory and practice*. Hoboken, New Jersey: John Wiley & Sons, 2008.

[6] A. Paradkar, K. C. Tai, and M. A. Vouk, "Automatic Test-Generation for Predicates," *IEEE TRANSACTIONS ON RELIABILITY,* vol. 45, pp. 515-530, 1996.

[7] M. Pezze and M. Young, *Software testing and analysis : process, principles, and techniques*. Hoboken, New Jersey: Wiley, 2008.

[8] P. C. Jorgensen, *Software testing : a craftsman's approach*. Boca Raton,Florida: CRC Press, 2002.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, Second ed.: MIT Press and McGraw-Hill, 2001.

[10] Zarazi (Mozillar Developer Network Contributor). *Regular Expression* [Online]. Available: https://developer.mozilla.org/th/docs/Web/JavaScript/Guide/Regular_Expressions

[11] The Dojo Foundation. *D.O.H.: Dojo Objective Harness* [Online]. Available: https://dojotoolkit.org/reference-guide/1.9/util/doh.html

[12] Appcelerator. *Aptana | Studio* [Online]. Available: http://www.aptana.com/products/studio3.html

[13] S. Godboley, G. S. Prashanth, D. P. Mohapatro, and B. Majhi, "Increase in Modified Condition/Decision Coverage using program code transformer," in *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, 2013, pp. 1400-1407.

[14] J. Costa and J. Monteiro, "Computation of the minimal set of paths for observability-based statement coverage," in *Mixed Design of Integrated Circuits and Systems, 2008. MIXDES 2008. 15th International Conference on*, 2008, pp. 587-592.

ภาคผนวก

ภาคผนวก ก
พจนานุกรมข้อมูล

ตารางที่ ก-1 พจนานุกรมข้อมูลตาราง file

ชื่อ	คำอธิบาย	ชนิดข้อมูล	คีย์	ค่าว่าง
idFile	ลำดับการนำเข้าไฟล์จาวาสคริปต์	Integer	PK	ไม่ว่าง
FILEname	ชื่อไฟล์จาวาสคริปต์ที่ใช้ทดสอบ	Varchar(45)		ว่าง

ตารางที่ ก- 2 พจนานุกรมข้อมูลตาราง testcase

ชื่อ	คำอธิบาย	ชนิดข้อมูล	คีย์	ค่าว่าง
idTC	หมายเลขกรณีทดสอบ	Varchar(50)	PK	ไม่ว่าง
TCfuncname	ชื่อฟังก์ชัน	Varchar(50)		ว่าง
TCcoverage	ค่าความครอบคลุมประโยคคำสั่ง	Double		ว่าง
TCvalue	ค่าพารามิเตอร์	Varchar(2000)		ว่าง
TCpath	ทางเดินใช้ในการทดสอบ	Varchar(50)		ว่าง
TCResult	ผลการทดสอบ	Varchar(20)		ว่าง
FILE_idFILE	ลำดับการนำเข้าไฟล์จาวาสคริปต์	Integer	FK	ไม่ว่าง

ภาคผนวก ข
รายงานการทดสอบไฟล์จาวาสคริปต์ตัวอย่าง

ตารางที่ ข-1 รายงานการทดสอบไฟล์ Case1-Number.js

Function : main			
% Statement Coverage : 100 %			
idTC	TCValue	TCPath	TCResult
16-05-31_02-38-49_TC01	a = 33.0 , b = 89.0	1->2->3->4->5->End	Exercised
16-05-31_02-38-49_TC02	a = 30.0 , b = 0.0	1->2->3->4->6->End	Exercised

ตารางที่ ข-2 รายงานการทดสอบไฟล์ Case2-String.js

Function : String2			
% Statement Coverage : 100 %			
idTC	TCValue	TCPath	TCResult
16-05-31_02-40-20_TC01	a = aaaaa	1->2->3->End	Exercised
16-05-31_02-40-20_TC02	a = SonRT	1->2->4->End	Exercised

ตารางที่ ข-3 รายงานการทดสอบไฟล์ Case3-bool.js

Function : boolchk			
% Statement Coverage : 100 %			
idTC	TCValue	TCPath	TCResult
16-05-31_02-40-46_TC01	boo = C/G , far = C/G	1->2->3->5->6->7->8->End	not Exercised
16-05-31_02-40-46_TC02	boo = true , far = false	1->2->3->5->9->10->11->12->End	Exercised
16-05-31_02-40-46_TC03	boo = true , far = true	1->2->3->5->9->13->End	Exercised
16-05-31_02-40-46_TC04	boo = false , far = false	1->2->4->5->6->7->8->End	Exercised

ตารางที่ ข-4 รายงานการทดสอบไฟล์ Case4-nestedif.js

Function : nestedif			
% Statement Coverage : 100 %			
idTC	TCValue	TCPath	TCResult
16-05-31_02-41-16_TC01	Type = 1.0 , PageCount = 63.0	1->2->3->End	Exercised
16-05-31_02-41-16_TC02	Type = 2 , PageCount = 2	1->2->4->5->End	Exercised
16-05-31_02-41-16_TC03	Type = 2 , PageCount = 5	1->2->4->6->End	Exercised

ตารางที่ ข-5 รายงานการทดสอบไฟล์ Case5-StringMethod.js

Function : stringMethod			
% Statement Coverage : 100 %			
idTC	TCValue	TCPath	TCResult
16-05-31_02-41-47_TC01	line = mTcrTywSlikgLoIJtibEgLqOEhMAMIqmeWSCRusqUXNUzzMMKokTeSHLxGwCqUPpRsteveIbeAlanJsbqTyqhd,	1->2->3->4->5->6->7->End	Exercised

ประวัติผู้เขียนวิทยานิพนธ์

นายวิทยา เหลืองหิรัญ เกิดวันที่ 12 กรกฎาคม 2527 ที่จังหวัดนนทบุรี สำเร็จการศึกษาในหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาปิโตรเคมีและวัสดุพอลิเมอร์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยศิลปากร เมื่อปีการศึกษา 2549 หลังจากนั้นได้ทำงานที่บริษัท โฟมเทค อินเตอร์เนชันเนล จำกัด บริษัท ลักกี้เทค จำกัด และ บริษัท สยามมิชลิน จำกัด รวมระยะการทำงาน 4 ปี 6 เดือน จากนั้นเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปี 2556