19th International Computer Science and Engineering Conference (ICSEC)
Chiang Mai, Thailand, 23-26 November, 2015

การสร้างกรณีทดสอบจากจาวาสคริปต์บนเงื่อนไขความครอบคลุมประโยคคำสั่ง
Test Cases Generation from Javascript based on Statement Coverage Criteria

วิทยา เหลืองหิรัญ[1] และ ธาราทิพย์ สุวรรณศาสตร์[2]
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
E-mail: Witthaya.L@student.chula.ac.th[1] , taratip.s@chula.ac.th[2]

บทคัดย่อ

กรณีทดสอบเป็นส่วนสำคัญในการทดสอบซอฟต์แวร์ ถ้ากรณีทดสอบที่ถูกสร้างขึ้นใช้ทดสอบแล้วไม่พบข้อบกพร่อง ทำให้การใช้งานซอฟต์แวร์จริง อาจเกิดปัญหาจากข้อบกพร่องที่แฝงนั้นได้ เงื่อนไขคำสั่งจะถูกใช้ในการพัฒนาซอฟต์แวร์ส่วนใหญ่ คำสั่งภายใต้เงื่อนไขไม่สามารถถูกดำเนินการได้และอาจเกิดข้อบกพร่องขึ้น ถ้าการทดสอบไม่ครอบคลุมเงื่อนไขของคำสั่ง งานวิจัย [1] นำเสนอเครื่องมือสร้างมอดูลทดสอบไฟล์จาวาสคริปต์โดยสร้างข้อมูลทดสอบด้วยวิธีการสุ่มค่า ทำให้ทางเดินคำสั่งบางทางเดินไม่ได้ถูกทดสอบ งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอแนวคิดการสร้างกรณีทดสอบจากซอฟต์แวร์ที่ถูกพัฒนาด้วยภาษาจาวาสคริปต์ โดยวิเคราะห์ซอร์สโค้ดเพื่อสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง และสร้างกรณีทดสอบเพื่อใช้ดำเนินการทดสอบ ทำให้การทดสอบมีความครอบคลุมประโยคคำสั่งมากขึ้น

คำสำคัญ: การทดสอบระดับหน่วย, การสร้างกรณีทดสอบ, ความครอบคลุมของการทดสอบ, เงื่อนไขคำสั่ง , จาวาสคริปต์

Abstract

Test Case is an important part in Software testing. If generated test cases cannot find defects, the delivered software may have problems from the hidden defects. Conditional statements are usually used in software development. If testing cannot cover conditional statements, statements under conditional statements cannot be executed and they can cause defects. In [1], they presented a tool for generating test module to test a javascript file with random generating test data that causes some program paths cannot be tested. Our research proposes an approach for test cases generation from Javascript by analyzing source code for generating test cases and test data from conditional statements in order to gain higher statement coverage.

Keywords: Unit testing, Test case generation, Test coverage, Condition Statement, Javascript

1. บทนำ

กรณีทดสอบเป็นส่วนสำคัญในการทดสอบซอฟต์แวร์โดยเฉพาะการทดสอบซอฟต์แวร์ในระดับหน่วย ถ้ากรณีทดสอบที่ถูกสร้างขึ้นไม่พบข้อบกพร่อง (Defect) ที่แฝงในซอฟต์แวร์ระดับหน่วยแล้ว การทดสอบซอฟต์แวร์ในระดับบูรณาการ การทดสอบระบบ ร่วมไปถึงการใช้งานซอฟต์แวร์จริง อาจเกิดปัญหาจากข้อบกพร่องที่แฝงนั้นได้

เงื่อนไขคำสั่งจะปรากฏการพัฒนาซอฟต์แวร์อยู่บ่อยครั้ง เนื่องจากเงื่อนไขคำสั่งที่จะใช้ตรวจสอบค่านำเข้าและตัดสินใจกำหนดทางเดินของคำสั่ง ให้คำสั่งดำเนินการตามทางเดินที่เงื่อนไขกำหนดเฉพาะได้ ถ้าเงื่อนไขคำสั่งไม่สามารถตัดสินใจกำหนดทางเดินของคำสั่งจากค่านำเข้าได้ จะทำให้คำสั่งภายใต้เงื่อนไขไม่สามารถถูกดำเนินการได้และเกิดข้อบกพร่องขึ้น

จากการศึกษาทฤษฎีและงานวิจัย [2-5] เกี่ยวกับเสนอเทคนิคการสร้างค่าข้อมูลทดสอบเป็นส่วนหนึ่งของเทคนิคการทดสอบซอฟต์แวร์แบบไวท์บอกซ์ (White box testing technique) ซึ่งเป็นเทคนิคสำหรับทดสอบซอฟต์แวร์โดยสนใจที่โครงสร้างของซอฟต์แวร์ โดยสร้างกราฟการไหลของการควบคุม [6, 7] เพื่อวิเคราะห์โครงสร้างซอฟต์แวร์และสร้างค่านำเข้าเพื่อทดสอบซอฟต์แวร์นั้น

งานวิจัย [1] ได้นำเสนอเครื่องมือสร้างมอดูลทดสอบไฟล์จาวาสคริปต์โดยสร้างข้อมูลทดสอบด้วยวิธีการสุ่มค่าทำให้ ซึ่งมีการใช้เวลาสุ่มข้อมูลทดสอบค่อนข้างนาน และทางเดินคำสั่งในบางทางเดินไม่ได้ถูกทดสอบจากข้อมูลทดสอบที่ถูกสุ่มมา จึงทำให้การทดสอบมีความครอบคลุมประโยคคำสั่ง (Statement coverage) น้อย

ดังนั้นงานวิจัยนี้จึงนำเสนอแนวคิด การสร้างกรณีทดสอบจากซอฟต์แวร์ที่ถูกพัฒนาด้วยภาษาจาวาสคริปต์ โดยสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง โดยการวิเคราะห์ซอร์สโค้ด สร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง เพื่อให้ได้กรณีทดสอบเฉพาะสามารถดำเนินการบนทางเดินคำสั่งมากขึ้น ทำให้การทดสอบไฟล์จาวาสคริปต์มีครอบคลุมประโยคคำสั่งมากขึ้น

2. งานวิจัยที่เกี่ยวข้อง

2.1 “เครื่องมือสร้างมอดูลทดสอบสำหรับจาวาสคริปต์บนเงื่อนไขความครอบคลุมคำสั่ง” [1]

งานวิจัย [1] นำเสนอเครื่องมือสร้างมอดูลทดสอบไฟล์จาวา-สคริปต์โดยสร้างข้อมูลทดสอบด้วยวิธีการสุ่มค่าตามประเภทของพารามิเตอร์ ซึ่งมีการใช้เวลาสุ่มข้อมูลทดสอบค่อนข้างนาน และข้อมูลทดสอบที่ถูกสุ่มไม่สามารถดำเนินการทดสอบในบางเงื่อนไขคำสั่งได้ ทำให้ทางเดินคำสั่งบางทางเดินไม่ได้ถูกทดสอบ จึงทำให้การทดสอบมีความครอบคลุมประโยคคำสั่งน้อย ผลลัพธ์ที่ได้จากงานวิจัย คือ การแสดงรายงานการทดสอบของฟังก์ชันจาวาสคริปต์ ซึ่งมีรายละเอียดทางเดินทดสอบ ค่าข้อมูลทดสอบที่ถูกสุ่มมาใช้ทดสอบ และเปอร์เซ็นต์ความครอบคลุมในแต่ละฟังก์ชัน

งานวิจัยที่นำเสนอนี้จึงได้ปรับปรุงวิธีการสร้างข้อมูลทดสอบโดยการสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง เพื่อให้ได้ข้อมูลทดสอบเฉพาะสามารถดำเนินการบนทางเดินคำสั่งมากขึ้น เพื่อทำให้การทดสอบไฟล์จาวาสคริปต์มีครอบคลุมประโยคคำสั่งมากขึ้น

2.2 “Computation of the minimal set of paths for observability-based statement coverage” [2]

งานวิจัย [2] ได้นำเสนอเทคนิคทดสอบซอฟต์แวร์แบบไวท์-บอกซ์ โดยวิเคราะห์หาเซตทางเดินที่สั้นที่สุดให้ครอบคลุมประโยคคำสั่งที่สังเกตในโปรแกรมฝังตัวเพื่อใช้สร้างเวกเตอร์นำเข้าสำหรับการทดสอบและทดสอบกับตัวอย่างโปรแกรมฝังตัวเชิงสถิติทั้งหมด 5 โปรแกรม ผลลัพธ์ที่ได้จากงานวิจัย คือ การสร้างค่าข้อมูลทดสอบ เพื่อดำเนินการทดสอบกับเซตทางเดินคำสั่งที่สั้นที่สุดของแต่ละโปรแกรมฝังตัวให้ครอบคลุมประโยคคำสั่งที่สังเกต

งานวิจัยที่นำเสนอนี้จึงได้นำเทคนิคสร้างค่าเวกเตอร์นำเข้าและตัวอย่างทดสอบงานวิจัยนี้ มาประยุกต์ใช้ในการสร้างและทดสอบแนวความคิดนี้

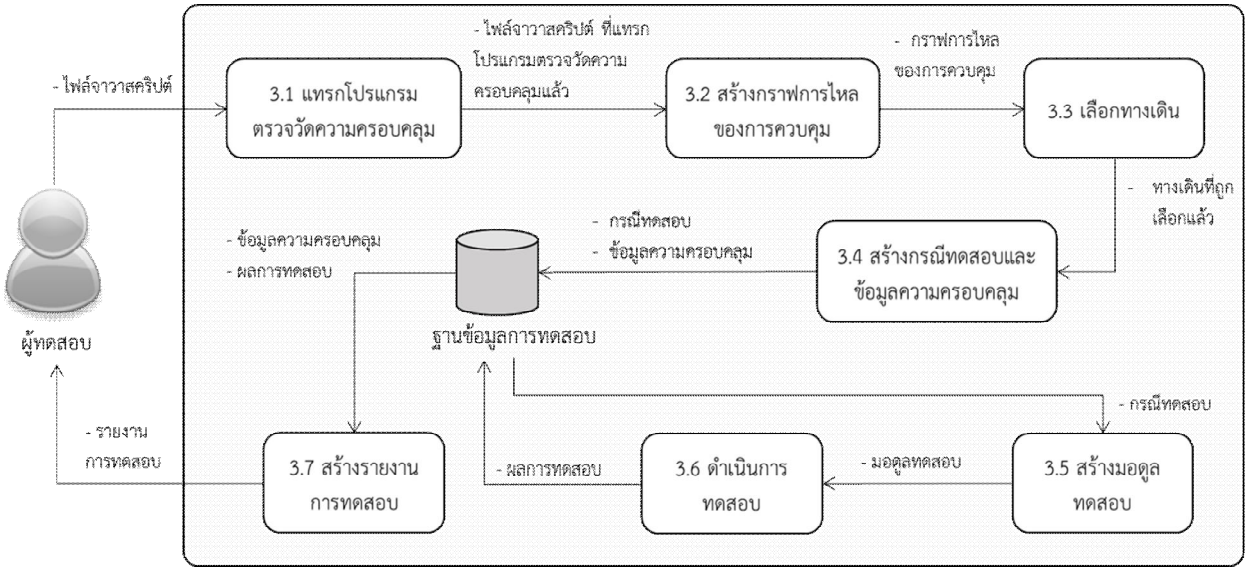
3. วิธีดำเนินงานวิจัย

งานวิจัยนี้นำเสนอการสร้างกรณีทดสอบซอฟต์แวร์สำหรับจาวาสคริปต์ โดยมีภาพรวมงานวิจัย ดังรูปที่ 1 ซึ่งสามารถแบ่งขั้นตอนออกได้เป็น 7 ขั้นตอน ได้แก่ การแทรกโปรแกรมตรวจวัดความครอบคลุม สร้างกราฟการไหลของการควบคุม เลือกทางเดิน สร้างกรณีทดสอบและข้อมูลความครอบคลุม สร้างมอดูลทดสอบ ดำเนินการทดสอบ และการรายงานการทดสอบ ในการสร้างกรณีทดสอบจะใช้ 4 ขั้นตอนแรก ส่วน 3 ขั้นตอนที่เหลือจะใช้สำหรับการดำเนินการทดสอบ เพื่อรองรับการพัฒนาเครื่องมือในลำดับต่อไป

จากงานวิจัยนี้จะได้กรณีทดสอบที่ดำเนินการทดสอบบนทางเดินคำสั่งมากขึ้น ทำให้การทดสอบไฟล์จาวาสคริปต์มีครอบคลุมประโยคคำสั่งมากขึ้น โดยเสนอวิธีการดำเนินงานหลัก 4 ขั้นตอน ในแต่ละขั้นตอนอธิบายได้ดังนี้

3.1 การแทรกโปรแกรมตรวจวัดความครอบคลุม

โปรแกรมตรวจวัดความครอบคลุมจะแทรกลงในซอร์สโค้ดของไฟล์จาวาสคริปต์ที่ผู้ใช้นำเข้า โดยใช้เทคนิคจากเรกูลาร์เอกซ์เพรสชันและเครื่องมือแอนท์เลอร์(ANTLR) ใช้ในการวิเคราะห์โครงสร้างไฟล์จาวาสคริปต์ ซึ่งตัวอย่างซอร์สโค้ดของไฟล์จาวาสคริปต์แสดงในรูปที่ 2 โดยผลลัพธ์ในขั้นตอนนี้คือไฟล์จาวาสคริปต์ที่แทรกโปรแกรมตรวจวัดความครอบคลุมแล้ว แสดงในรูปที่ 3



รูปที่ 1 ภาพรวมของงานวิจัย

```
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  }
```

รูปที่ 2 ตัวอย่างซอร์สโค้ดจาวาสคริปต์

จากรูปที่ 3 โปรแกรมตรวจวัดความครอบคลุมแทรกในไฟล์จาวาสคริปต์แบ่งออกเป็น 2 ส่วนหลัก ได้แก่

1.ส่วนแรกใช้วิเคราะห์โครงสร้างของไฟล์จาวาสคริปต์ โดยแทรกประโยคคำสั่งตัวอย่างที่ขึ้นต้นด้วย _jscover แล้วตามด้วยรายละเอียดคำสั่งของแต่ละบรรทัด ได้แก่ ชื่อไฟล์ ชื่อฟังก์ชัน เลขบรรทัด ประโยคคำสั่งหรือพารามิเตอร์ของฟังก์ชัน ในบรรทัดนั้นๆ ตามลำดับ เพื่อใช้สร้างกราฟการไหลของการควบคุม และวิเคราะห์หาค่าเวกเตอร์นำเข้าให้กับกรณีทดสอบที่จะถูกสร้างขึ้นต่อไป

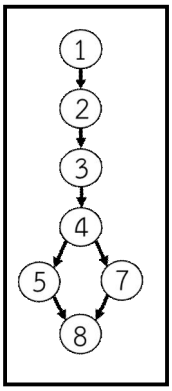
2.ส่วนที่สองใช้ตรวจสอบทางเดินของโปรแกรมในไฟล์จาวาสคริปต์ในขั้นตอนดำเนินการทดสอบ โดยกำหนดตัวแปร jsstringcounter และแทรกตัวแปรนี้เป็นตัวนับ (Counter) ในแต่ละคำสั่ง เพื่อบันทึกทางเดินของคำสั่งในขณะดำเนินการทดสอบ โดยตรวจสอบว่าค่าเวกเตอร์นำเข้าที่ถูกสร้างขึ้นสามารถดำเนินการตามทางเดินทดสอบที่กำหนดไว้กรณีทดสอบได้จริง

3.2 การสร้างกราฟการไหลของการควบคุม

ไฟล์จาวาสคริปต์ที่แทรกโปรแกรมตรวจวัดความครอบคลุมแล้วจะนำมาสร้างกราฟ โดยสร้างโหนดขึ้นมาจากตำแหน่งตำแหน่งฟังก์ชัน เงื่อนไข และประโยคคำสั่งที่ถูกระบุไว้ในโปรแกรมตรวจวัดความครอบคลุม จากนั้นลากเส้นเชื่อมโยงระหว่างโหนด เมื่อนำเส้นเชื่อมโยงระหว่างโหนดแต่ละกันมาต่อกันจะได้กราฟการไหลของการควบคุม ดังแสดงในรูปที่ 4

```
function main(a, b) {
    _jscover_func("TestExample.js","main","1:2:3:4:5:6:7:8",
                  "a:number","b:number");
        var jsstingcounter = "1";
            var x = a + b;
        jsstingcounter += "-2" ;
    _jscover_line("TestExample.js","main","2","var x = a + b;");
            var y = a - b;
        jsstingcounter += "-3" ;
    _jscover_line("TestExample.js","main","3","var x = a - b;");
        jsstingcounter += "-4" ;
            if (x > y)
    _jscover_line("TestExample.js","main","4","if (x > y)");
    {           z = x + y;
        jsstingcounter += "-5" ;
    }
    _jscover_line("TestExample.js","main","5","z = x + y;");
            else
    _jscover_line("TestExample.js","main","6","else");
    {           z = x - y;
        jsstingcounter += "-7" ;
    }
    _jscover_line("TestExample.js","main","7","z = x - y;");
        jsstingcounter += "-8" ;
        console.log(jsstingcounter);
}
```

รูปที่ 3 ตัวอย่างซอร์สโค้ดจาวาสคริปต์ที่แทรกโปรแกรมตรวจวัดความคลอบคลุมแล้ว



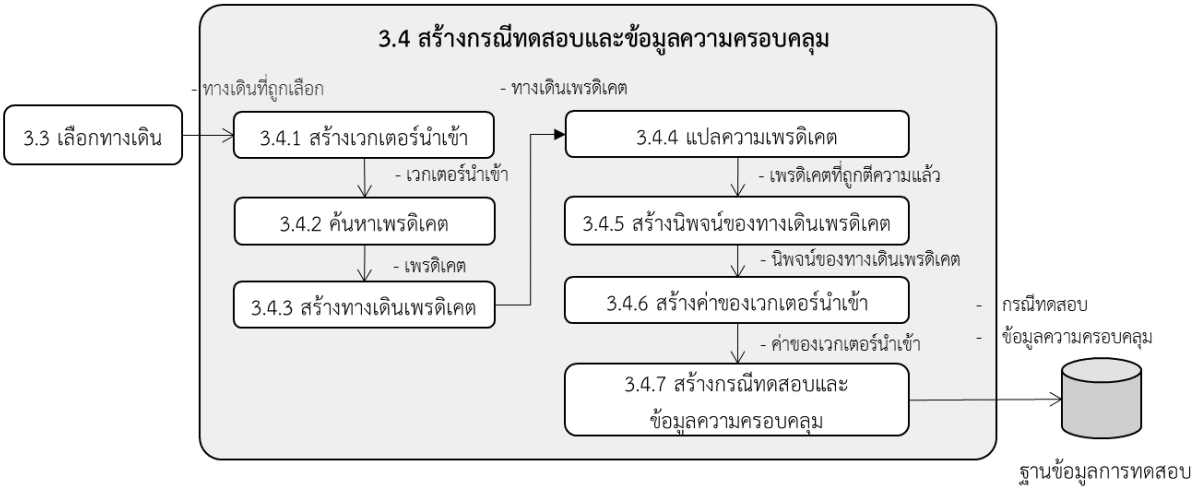
รูปที่ 4 กราฟการไหลของการควบคุมของซอร์สโค้ดจาวาสคริปต์ตัวอย่างในรูปที่ 2

3.3 การเลือกทางเดิน

การเลือกทางเดินจากกราฟการไหลของการควบคุม โดยใช้วิธีการค้นหาแนวลึกก่อน (Depth First Search) เพื่อให้ทางเดินผ่านทุกโหนดตามเกณฑ์การครอบคลุมประโยคคำสั่ง ทางเดินของซอร์สโค้ดจาวาสคริปต์ตัวอย่างในรูปที่ 4 เพื่อให้ครอบคลุมประโยคคำสั่ง สามารถแสดงได้ 2 ทางเดิน ได้แก่

ทางเดินแรก : 1 - > 2 -> 3 -> 4 -> 5 -> 8

ทางเดินที่สอง : 1 -> 2 -> 3 - > 4 -> 7 -> 8



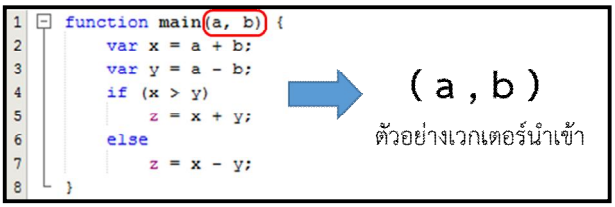
รูปที่ 5 แผนภาพกระบวนการสร้างกรณีทดสอบและข้อมูลความครอบคลุม

3.4 การสร้างกรณีทดสอบและข้อมูลความครอบคลุม

ในขั้นตอนนี้จะกล่าวถึงการสร้างกรณีทดสอบและข้อมูลความครอบคลุม โดยประกอบด้วย 7 ส่วนการทำงานย่อย โดยแสดงดังรูปที่ 5 และมีรายละเอียดดังต่อไปนี้

3.4.1 สร้างเวกเตอร์นำเข้า

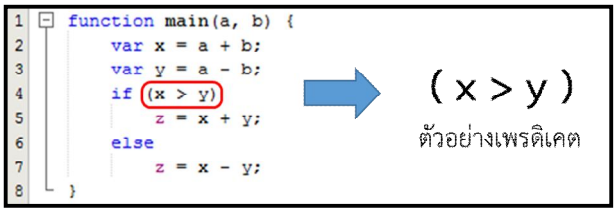
พารามิเตอร์ของฟังก์ชันต้นทางหรือโหนดเริ่มต้นของทางเดินที่เลือกจะถูกวิเคราะห์ว่ามีจำนวนพารามิเตอร์กี่ตัวและเป็นมีพารามิเตอร์ประเภทอะไรบ้าง ที่ใช้ในการดำเนินการในไฟล์จาวาสคริปต์ที่นำเข้ามา จากรูปที่ 6 ได้แสดงตำแหน่งพารามิเตอร์ของฟังก์ชันจาวาสคริปต์ และนำค่าพารามิเตอร์นั้นมาสร้างเป็นเวกเตอร์นำเข้า



รูปที่ 6 ตัวอย่างเวกเตอร์นำเข้าจากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

3.4.2 ค้นหาเพรดิเคต

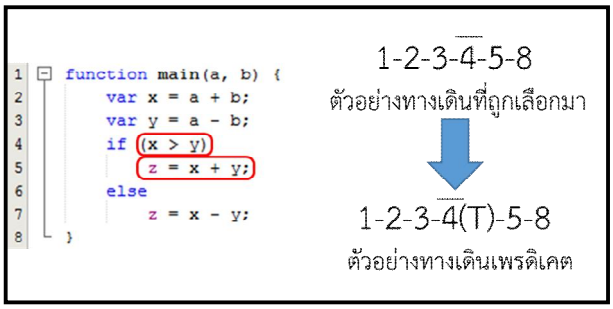
การค้นหาเพรดิเคตจากเงื่อนไขคำสั่งจะหาจากจุดตัดสินใจของทางเดินที่เลือกเดิน จากรูปที่ 7 แสดงตำแหน่งเพรดิเคตของฟังก์ชันจาวาสคริปต์ตัวอย่าง และนำเพรดิเคตนั้นมาดำเนินการขั้นตอนต่อไป



รูปที่ 7 ตัวอย่างเพรดิเคตจากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

3.4.3 สร้างทางเดินเพรดิเคต

ทางเดินที่ถูกเลือกมาจะนำไปสร้างทางเดินเพรดิเคต โดยวิเคราะห์แต่ละเงื่อนไขมีการตัดสินใจอย่างไร และบันทึกผลลัพธ์การตัดสินใจลงไปในทางเดิน ดังตัวอย่างรูปที่ 8 เป็นตัวอย่างแสดงการสร้างทางเดินเพรดิเคตในเงื่อนไขที่เป็นจริงของทางเดินแรก จากขั้นตอนการเลือกทางเดิน ในหัวข้อ 3.3

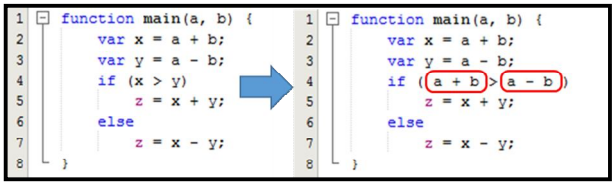


รูปที่ 8 ตัวอย่างทางเดินเพรดิเคตจากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

3.4.4 แปลความเพรดิเคต

การแปลความเพรดิเคต เป็นการแปลความตัวแปรภายในเพรดิเคตว่ามาจากแหล่งนำเข้าไหน โดยแหล่งนำเข้านี้สามารถเป็น เวกเตอร์นำเข้า ตัวแปรในฟังก์ชัน ค่าคงที่ หรือฟังก์ชันระบบ ที่สามารถเรียกภายในไฟล์จาวาสคริปต์ได้ ซึ่งฟังก์ชันระบบ เป็น

เมท็อดสำเร็จรูปในภาษาจาวาสคริปต์ที่กระทำต่ออ็อบเจกต์ชนิดข้อมูลต่างๆ โดยงานวิจัยที่นำเสนอนี้จะทดลองกับฟังก์ชันระบบบางฟังก์ชันจากนั้นแทนค่าแหล่งนำเข้าเกี่ยวข้องกับเพรดิเคต ลงในเพรดิเคตนั้น ดังตัวอย่างในรูปที่ 9 เพรดิเคต x > y สามารถ แปลความได้เป็น a + b > a – b เนื่องจากมีการกำหนดค่า x และ y จากเวกเตอร์นำเข้า (a , b) ในบรรทัดที่ 2 และ 3



รูปที่ 9 ตัวอย่างการแปลความเพรดิเคตจากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

3.4.5 สร้างนิพจน์ของทางเดินเพรดิเคต

นิพจน์ของทางเดินเพรดิเคต คือ การรวมกันระหว่างทางเดินเพรดิเคตกับเพรดิเคตที่ถูกแปลความแล้ว เพรดิเคตที่ปรากฏอยู่ในทางเดินจะต้องถูกทดสอบเงื่อนไขทั้งหมดจึงจะสามารถทดสอบทางเดินนั้นได้ ถ้าการทดสอบเพรดิเคตที่ปรากฏอยู่ในทางเดินนั้นไม่ผ่าน บันทึกทางเดินนั้นว่าไม่สามารถสร้างเวกเตอร์นำเข้าได้ และทางเดินนี้เป็นทางเดินที่เป็นไปไม่ได้

โหนด	
1	Input vector (a , b)
2	x = a + b;
3	y = a – b;
4(T)	(a + b > a – b)
5	z = x + y;
8	

รูปที่ 10 ตัวอย่างการสร้างนิพจน์ของทางเดินเพรดิเคตจากซอร์สโค้ดจาวาสคริปต์ตัวอย่าง

จากรูปที่ 10 เป็นตัวอย่างการนำทางเดินเพรดิเคต และการแปลความเพรดิเคต ในรูปที่ 8 และ 9 มารวมกันเพื่อให้ได้นิพจน์ของทางเดินเพรดิเคต

3.4.6 สร้างค่าเวกเตอร์นำเข้า

สร้างการสร้างเวกเตอร์นำเข้า สามารถแยกออกเป็น 3 ประเภทตามชนิดของพารามิเตอร์ ได้แก่ ตัวเลข สตริง และบูลีน โดยสร้างค่าเวกเตอร์นำเข้าให้ตรงตามเงื่อนไขของนิพจน์ของทางเดินเพรดิเคต

3.4.7 สร้างกรณีทดสอบและข้อมูลความครอบคลุม

จากตารางที่ 1 แสดงตัวอย่างกรณีทดสอบที่ถูกสร้างขึ้นจากค่าเวกเตอร์นำเข้าและทางเดินทดสอบในขั้นตอนก่อนหน้านี้ โดยมีรายละเอียดของกรณีทดสอบ ดังต่อไปนี้

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

ตารางที่ 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)-5-8
2	Main	a , b	a = -578 , b = -24	1-2-3-4(F)-7-8

ข้อมูลความครอบคลุม จะประกอบด้วยทางเดินคำสั่งทั้งหมดในฟังก์ชันและ เปอร์เซ็นต์ความครอบคลุมแต่ละฟังก์ชัน การคำนวณเปอร์เซ็นต์ความครอบคลุมสามารถหาได้จากสูตร

$$\text{เปอร์เซ็นต์ความครอบคลุม} = \frac{\text{จำนวนคำสั่งออกมาใช้อย่างน้อยหนึ่งครั้ง}}{\text{จำนวนคำสั่งที่ดำเนินการทั้งหมด}} \times 100 \quad (1)$$

4. สรุปและแนวทางพัฒนาต่อ

งานวิจัยนี้นำเสนอแนวคิดการสร้างกรณีทดสอบจากซอฟต์แวร์ที่ถูกพัฒนาด้วยภาษาจาวาสคริปต์ โดยวิเคราะห์ซอร์สโค้ด เพื่อสร้างข้อมูลทดสอบจากเงื่อนไขคำสั่ง ให้ได้กรณีทดสอบที่ความครอบคลุมประโยคคำสั่งมากขึ้น สำหรับแนวทางการพัฒนาต่อไปในอนาคต ผู้วิจัยจะนำแนวคิดที่ได้เสนอนี้ในพัฒนาเป็นเครื่องมือเพื่อนำไปประยุกต์ใช้กับงานทดสอบในลำดับต่อไป

เอกสารอ้างอิง

[1] P. Janthong and T. Suwannasart, "Tool for generating test module for JavaScript based on statement coverage criteria," in Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on, 2014, pp. 331-336.

[2] 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.

[3] 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.

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

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

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

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