เจเอฟทีเอ–ตัวสร้างกรณีทดสอบบนพื้นฐานการวิเคราะห์ต้นไม้ความผิดพลาด
JFTA–A Fault Tree Analysis-based Test Case Generator

พิมพ์ทิพย์ ไพบูลย์เกษมสุทธิ์ (Pimthip Paiboonkasemsut)* และ ญาใจ ลิมปิยะกรณ์ (Yachai Limpiyakorn)*

บทคัดย่อ

ความน่าเชื่อถือเป็นมิติหนึ่งของคุณภาพซอฟต์แวร์ การทดสอบความน่าเชื่อถือมีแนวโน้มสามารถตรวจพบความล้มเหลวส่วนมากที่จะเกิดขึ้นในการใช้งานจริงได้แต่เนิ่นๆ การวิเคราะห์ต้นไม้ความผิดพลาด (เอฟทีเอ) เป็นเทคนิคหนึ่งที่ถูกใช้อย่างแพร่หลายสำหรับการประเมินความปลอดภัยและความน่าเชื่อถือ ก่อนที่จะประยุกต์ใช้เอฟทีเอ วิธีต้นไม้การจำแนกเงื่อนไขมักถูกใช้สำหรับระบุเส้นทางที่ถูกต้องและไม่ถูกต้อง เพื่อใช้สร้างกรณีทดสอบการทำงานและกรณีทดสอบความน่าเชื่อถือตามลำดับ ในงานวิจัยนี้ ตัวสร้างกรณีทดสอบ เจเอฟทีเอ ได้ถูกพัฒนาขึ้นด้วยแนวคิดการออกแบบโมเดล วิว คอนโทรลเลอร์ และพัฒนาด้วยภาษาจาวา โดยใช้ไลบรารีเจกราฟเอกซ์ เพื่อช่วยการแสดงผลลัพธ์แผนภาพต้นไม้การจำแนกเงื่อนไข ระบบที่พัฒนาขึ้นทำงานโดยรับข้อมูลเข้าเป็นแผนภาพกิจกรรมยูเอ็มแอลในรูปแบบเอกซ์เอ็มไอที่สร้างจากปลั๊กอินอีคลิปส์ ปาปิรุส มาร์ส ร่วมกับการระบุเส้นทางที่ระบบสามารถทำงานได้อย่างถูกต้อง ซึ่งกำหนดโดยผู้ใช้ จากนั้นแผนภาพต้นไม้การจำแนกเงื่อนไข แผนภาพต้นไม้ความสำเร็จ และแผนภาพต้นไม้ความผิดพลาดจะถูกสร้างเป็นผลลัพธ์จากการประมวลผล ระบบที่พัฒนาขึ้นในงานนี้จะช่วยลดต้นทุนค่าใช้จ่ายและความผิดพลาดของมนุษย์ในการสร้างกรณีทดสอบ

คำสำคัญ: การวิเคราะห์ต้นไม้ความผิดพลาด การสร้างกรณีทดสอบ ความน่าเชื่อถือ แผนภาพกิจกรรม

Abstract

Reliability is one of the dimensions of software quality. Reliability testing tends to uncover earlier those failures which are most likely in actual operations. Fault Tree Analysis (FTA) is one of the techniques widely used for safety and reliability evaluations. Prior to applying FTA, the condition classification tree method will be generally carried out to identify the valid and invalid paths that shall be used for creating functionality test cases and reliability test cases, respectively. In this research, the test case generator, JFTA, is developed with the Model-View-Controller design concept. The JFTA is implemented in Java and uses JGraphX library to help rendering the output diagrams of Condition Classification Tree. The implemented system works with the input of UML activity diagrams in XMI format, generated by Papyrus Mars (Eclipse Plug-in). In addition to the activity diagram, the success goals are also required to be input by the user. Afterwards, Condition-Classification Tree diagram, Success Tree Diagram, and Fault Tree diagram will be generated as the results. The implementation in this work would reduce the cost and human-error in test case construction.

Keywords: Fault Tree Analysis, Test Case Generation, Reliability, Activity Diagram.

1. บทนำ

ปัจจุบัน เครื่องมือในการสร้างกรณีทดสอบได้ถูกใช้อย่างแพร่หลายในหลายหน่วยงานหรือองค์กร ทั้งนี้กรณีทดสอบมีหลายประเภท แต่ละประเภทมีเป้าหมายและจุดประสงค์ของการทดสอบที่แตกต่างกัน ได้แก่ การทดสอบการทำงานของฟังก์ชัน (Functionality Testing) ซึ่งเป็นการทดสอบการทำงานที่ถูกต้องหรือเส้นทางที่ถูกต้อง (Valid paths)

* ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

* Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University.

ส่วนการทดสอบความน่าเชื่อถือ (Reliability Testing) มีจุดประสงค์เพื่อทดสอบการทำงานที่ผิดพลาดหรือเส้นทางที่ไม่ถูกต้อง (Invalid paths) ซึ่งจะบ่งบอกความน่าเชื่อถือของตัวผลิตภัณฑ์หรือซอฟต์แวร์ [1] โดยทั่วไป การวิเคราะห์ต้นไม้ความผิดพลาดหรือเอฟทีเอ (Fault Tree Analysis: FTA) [2] เป็นเทคนิคหนึ่งที่มักนิยมใช้อย่างแพร่หลายสำหรับการทดสอบซอฟต์แวร์ที่ต้องการความน่าเชื่อถือ ด้วยวิธีการวิเคราะห์เหตุการณ์จากบนลงล่าง (Top-down event) เพื่อสร้างกรณีทดสอบความผิดพลาดสำหรับการทดสอบซอฟต์แวร์ในสภาวะการทำงานไม่ปกติ ได้ผลลัพธ์เป็นแผนภาพต้นไม้ความผิดพลาดหรือเอฟทีดี (Fault Tree Diagram: FTD) ซึ่งจะถูกสร้างมาจากกระบวนการทำงานในสภาวะปกติ สำหรับการทำงานสภาวะปกติจะนิยมนำมาใช้เพื่อทดสอบการทำงานของฟังก์ชัน นำเสนอในรูปแบบของแผนภาพต้นไม้ความสำเร็จหรือเอสทีดี (Success Tree Diagram: STD)

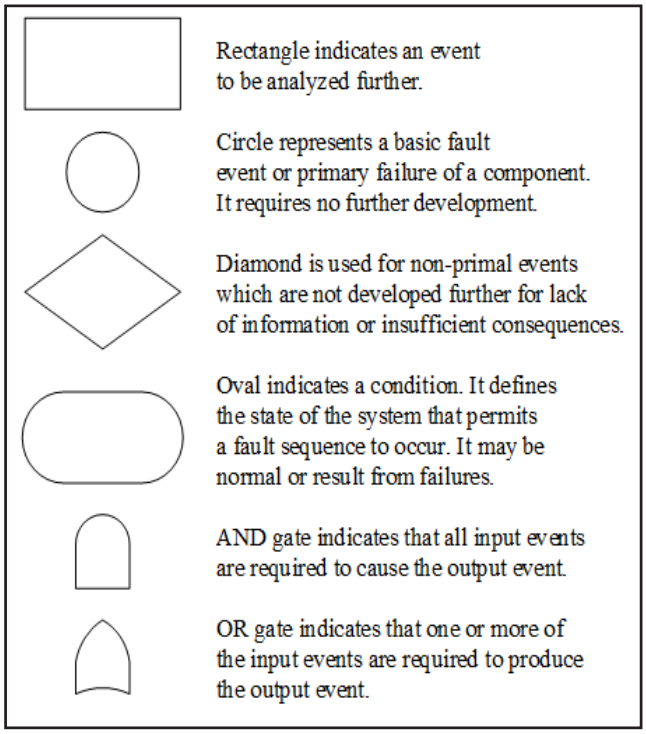
บทความ [3], [4] ได้นำเสนอวิธีการสร้างกรณีทดสอบการทำงานที่ไม่ถูกต้องด้วยเทคนิคการวิเคราะห์ต้นไม้ความผิดพลาดบนพื้นฐานการสร้างแบบจำลองต้นไม้การจำแนกเงื่อนไขหรือซีซีทีเอ็ม (Condition-Classification Tree Model: CCTM) งานวิจัยนี้ได้ประยุกต์ใช้แนวทางดังกล่าวและพัฒนาเป็นระบบต้นแบบสำหรับสร้างต้นไม้ความผิดพลาดจากเดิมที่ทำด้วยมือ เพื่อให้การสร้างกรณีทดสอบการทำงานที่ไม่ถูกต้องมีประสิทธิภาพและลดความผิดพลาดของการสร้างกรณีทดสอบด้วยมือ

2. งานวิจัยที่เกี่ยวข้อง

2.1 การวิเคราะห์ต้นไม้ความผิดพลาดหรือเอฟทีเอ

การวิเคราะห์ต้นไม้ความผิดพลาด [2] เป็นเครื่องมือสำหรับการวิเคราะห์ จำลองแผนภาพ และประเมินเส้นทางของความผิดพลาดในระบบ การวิเคราะห์ด้วยวิธีนี้จะมีกระบวนการที่มีประสิทธิภาพในการประเมินความเสี่ยงของระบบ ปัจจุบันมีหลายองค์กรและบุคลากรที่มีความคุ้นเคยกับเครื่องมือดังกล่าว ซึ่งมักถูกนำไปใช้ในระดับสากลเพื่อประเมินความปลอดภัยและความน่าเชื่อถือ การประเมินความเสี่ยง หรือการวิเคราะห์เหตุการณ์ที่ระบบทำงานผิดพลาด [5], [6]

พื้นฐานของการวิเคราะห์ต้นไม้ความผิดพลาด คือ การแปลงความผิดพลาดในระบบใช้งานจริงหรือสถานการณ์จริงออกมาอยู่ในรูปแบบของแผนภาพและแบบจำลองเชิงตรรกะซึ่งจะอำนวยความสะดวกในการหาความสัมพันธ์และสาเหตุของปัญหา ภาพที่ 1 แสดงบางส่วนของสัญลักษณ์ที่ใช้ในแผนภาพต้นไม้ความผิดพลาด ซึ่งใช้อธิบายเหตุการณ์การทำงานที่ไม่ถูกต้อง



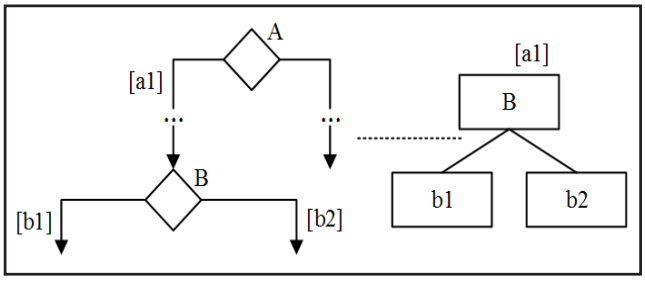
ภาพที่ 1 *ตัวอย่างสัญลักษณ์ที่ใช้ในแผนภาพต้นไม้ความผิดพลาด [5]*

เอฟทีเอเป็นการประยุกต์เงื่อนไขต่างๆ ที่จะนำไปสู่เหตุการณ์ที่แสดงอยู่บนสุดหรือต้นเหตุของเหตุการณ์ที่ทำให้การทำงานไม่ถูกต้อง/ไม่สำเร็จ ทั้งนี้การสร้างต้นไม้ความผิดพลาด และทำการประเมินเป็นกระบวนการที่ค่อนข้างง่ายและตรงไปตรงมา อย่างไรก็ตามกระบวนการดังกล่าวจะยากขึ้นเมื่อต้นไม้มีขนาดใหญ่ซับซ้อนขึ้น กล่าวคือความสามารถในการประเมินต้นไม้ความผิดพลาดนั้นแปรผกผันกับขนาดความซับซ้อนของต้นไม้

2.2 Generating test cases from UML activity diagrams using the Condition-Classification Tree Method [3]

บทความนี้ได้นำเสนอเทคนิคที่ใช้ในกระบวนการสร้างกรณีทดสอบจากแผนภาพกิจกรรม ด้วยวิธีการสร้างแบบ

จำลองต้นไม้การจำแนกเงื่อนไขหรือซีซีทีเอ็ม โดยจะสร้างแผนภาพต้นไม้การจำแนกเงื่อนไข (Condition-Classification Tree) จากจุดตัดสินใจของซอฟต์แวร์ที่ได้จากแผนภาพกิจกรรมยูเอ็มแอล (UML activity diagram)



ภาพที่ 2 *ตัวอย่างการพิจารณาจุดตัดสินใจบนแผนภาพกิจกรรม [3]*

ตัวอย่างการพิจารณาจุดตัดสินใจ จุดบี (B) บนแผนภาพกิจกรรม (ภาพที่ 2) เมื่อนำมาสร้างต้นไม้การจำแนกเงื่อนไขจะได้กรณี b1 และ b2 สำหรับการสร้างซีซีทีเอ็มจะประกอบด้วย 3 ขั้นตอนหลัก ดังนี้

1) วิเคราะห์และสร้างต้นไม้การจำแนกเงื่อนไข ขั้นตอนนี้จะทำการวิเคราะห์แผนภาพกิจกรรมที่อยู่ในรูปแบบยูเอ็มแอลเพื่อจัดสร้างต้นไม้การจำแนกเงื่อนไข

2) สร้างตารางกรณีทดสอบ โดยการนำต้นไม้การจำแนกที่ได้จากข้อ 1 มาวางเรียงกันเพื่อสร้างตารางทดสอบ จากนั้นทำการระบุเงื่อนไขของแต่ละต้นและสร้างเส้นแถวของตารางเพื่อใช้ในการกำหนดกรณีทดสอบ

3) สร้างกรณีทดสอบจากแต่ละแถวของตาราง แบบจำลองนี้เป็นวิธีหนึ่งที่ใช้เทคนิคการทดสอบแบบกล่องดำ (Black Box Testing) ซึ่งช่วยให้การสร้างกรณีทดสอบมีประสิทธิภาพและลดเวลาในการวิเคราะห์เอกสารความต้องการ ผู้วิจัยได้ประยุกต์ใช้วิธีการดังกล่าวข้างต้นเพื่อพิจารณาสกัดจุดตัดสินใจ และนำไปสร้างแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาด

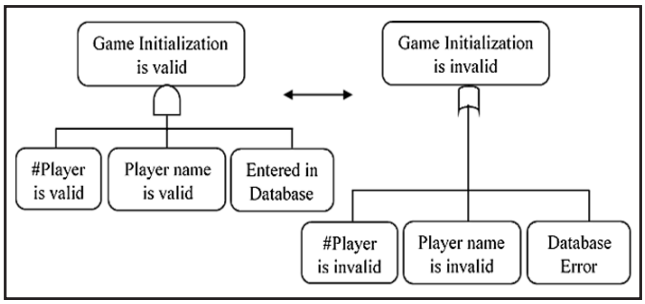
2.3 An Approach to Generate Safety Validation Test Cases from UML Activity Diagram [4]

บทความนี้ได้นำเสนอการตรวจสอบการสร้างโมเดลของระบบซอฟต์แวร์ และการวิเคราะห์โมเดลเพื่อการค้นหาและกำจัดข้อผิดพลาด ซึ่งรวมถึงการใช้วิธีการวิเคราะห์ต้นไม้ความผิดพลาดสำหรับการพิจารณาตรวจสอบด้วย พบว่าเป็นวิธีที่มีประสิทธิภาพในการวิเคราะห์ จำลองรูปแบบ และประเมินข้อผิดพลาดในระบบ

วิธีการวิเคราะห์ต้นไม้ความผิดพลาดเป็นการวิเคราะห์โดยอิงจากเหตุการณ์ข้างบนสุดเป็นหลัก ซึ่งจะเป็นเหตุการณ์ที่ไม่ผิดพลาดและเกิดจากการรวมกันของหลายๆ เหตุการณ์ในโหนดลูก ในทางทฤษฎีวิธีการวิเคราะห์ต้นไม้ความผิดพลาดจะระบุเหตุการณ์พื้นฐานที่สามารถนำไปสู่เหตุการณ์ที่ทำให้เกิดความผิดพลาดในชั้นบนสุดได้

การวิเคราะห์ต้นไม้ความผิดพลาดของระบบซอฟต์แวร์สามารถนำมาใช้เป็นเครื่องมือในการวิเคราะห์เหตุการณ์ที่สามารถนำไปสู่เหตุการณ์ขั้นสูงสุดหรือเหตุการณ์ที่ทำให้เกิดความผิดพลาด ซึ่งนำเสนอในรูปแบบของต้นไม้ความผิดพลาด แผนภาพต้นไม้ความผิดพลาดจะประกอบด้วยสัญลักษณ์ 3 ประเภท คือ เหตุการณ์พื้นฐาน เหตุการณ์ขั้นกลาง และลอจิกเกต (Logic gate) โดยที่เหตุการณ์พื้นฐาน และเหตุการณ์ขั้นกลาง แสดงถึงต้นเหตุของเหตุการณ์บนสุดหรือเหตุการณ์ที่ไม่ปกติ ขณะที่ลอจิกเกตแสดงถึงความสัมพันธ์ที่จำเป็นที่จะทำให้เกิดเหตุการณ์ขั้นสูงต่อไป ลอจิกเกตพื้นฐานมี 2 ประเภท: OR gate และ AND gate

การสร้างต้นไม้ความสำเร็จจะใช้สัญลักษณ์เช่นเดียวกันกับต้นไม้ความผิดพลาด แต่แตกต่างกันที่เหตุการณ์ขั้นสูงสุดของต้นไม้ความสำเร็จจะเป็นเหตุการณ์ที่ทำให้ระบบทำงานได้อย่างถูกต้อง ภาพที่ 3 แสดงตัวอย่างการแปลงจากต้นไม้ความสำเร็จเป็นต้นไม้ความผิดพลาด



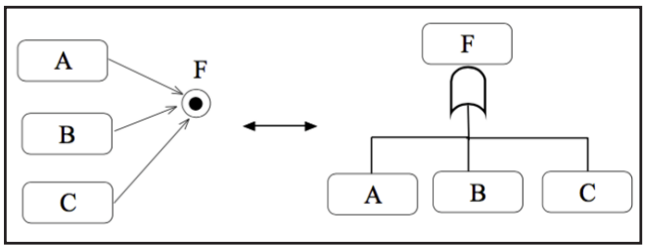
ภาพที่ 3 *ตัวอย่างการแปลงต้นไม้ความสำเร็จเป็นต้นไม้ความผิดพลาด [4]*

บทความได้นำเสนอตัวอย่างการแปลงแผนภาพกิจกรรมเป็นต้นไม้ความสำเร็จ มีองค์ประกอบดังนี้

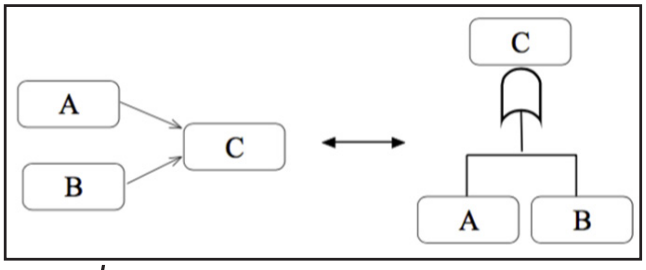
1) การแปลงแผนภาพกิจกรรมเป็นต้นไม้ความสำเร็จ จะเชื่อมต่อด้วย OR gate ตัวอย่างดังภาพที่ 4 อธิบายได้ว่า F จะเป็นจริง ก็ต่อเมื่อ A หรือ B หรือ C เป็นจริง

2) การเปลี่ยนสถานะในแผนภาพกิจกรรมจะถูกแปลงเป็น

ข้อมูลนำเข้าของ OR gate ในแผนภาพต้นไม้ความสำเร็จตัวอย่างดังภาพที่ 5



ภาพที่ 4 *ตัวอย่างการแปลงแผนภาพกิจกรรมเป็นต้นไม้ความสำเร็จด้วย OR gate [4]*



ภาพที่ 5 *ตัวอย่างการแปลงเป็นต้นไม้ความสำเร็จจากการเปลี่ยนสถานะในแผนภาพกิจกรรม [4]*

กฎการสร้างต้นไม้ที่นำเสนอในบทความสามารถครอบคลุมหลายกรณีที่เกิดขึ้นในแผนภาพกิจกรรม เมื่อเทียบกับงานวิจัยอื่นแล้ว พบว่าวิธีดังกล่าวสามารถนำมาพัฒนาเป็นเครื่องมือในการสร้างกรณีทดสอบได้อย่างมีประสิทธิภาพ จากเดิมที่เป็นการวิเคราะห์โดยมนุษย์ซึ่งย่อมมีความผิดพลาด เนื่องจากการพิจารณาตรวจดูแผนภาพกิจกรรมต้องใช้ความละเอียดเป็นอย่างมาก

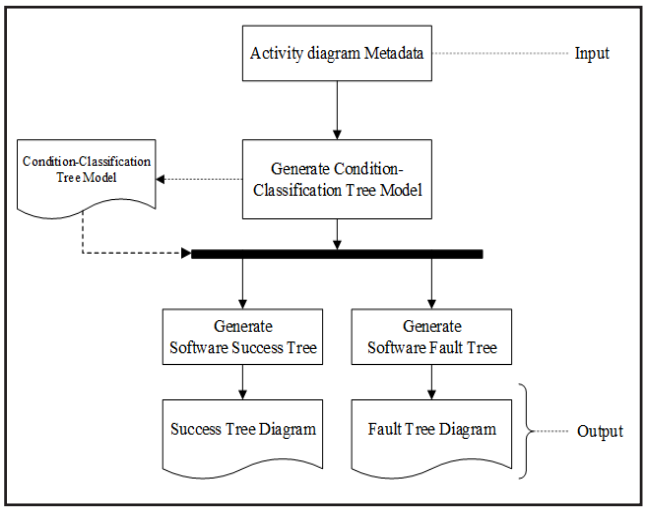
การทดสอบความคงทน (Robustness) จะทดสอบระบบด้วยกรณีทดสอบแบบปกติและแบบพิเศษ ถึงกระนั้นการสร้างกรณีทดสอบข้อผิดพลาดนั้นกระทำได้ยาก ต้นไม้ความผิดพลาดจึงถูกนำมาใช้สำหรับสร้างกรณีทดสอบและระบุถึงความสัมพันธ์ของข้อมูลนำเข้าและเงื่อนไข ซึ่งการทดสอบในรูปแบบนี้สามารถนำไปเพิ่มความน่าเชื่อถือของระบบซอฟต์แวร์ได้อย่างมีประสิทธิภาพ ข้อดีอีกประการของวิธีการนี้คือ การระบุความผิดพลาดของระบบซอฟต์แวร์ได้ในขั้นตอนแรกๆ ของการพัฒนาซอฟต์แวร์ ซึ่งทำให้สามารถแก้ไขหรือป้องกันได้แต่เนิ่นๆ เนื่องจากถ้าความผิดพลาดพื้นฐานไม่เกิดขึ้น ความผิดพลาดของระบบย่อมไม่เกิดขึ้นเช่นเดียวกัน

2.4 แผนภาพกิจกรรม

แผนภาพกิจกรรมใช้อธิบายลำดับงานต่างๆ ที่เกิดขึ้น ในลักษณะของกระแสงาน (Workflow) ในโดเมนวิศวกรรมซอฟต์แวร์ มักนิยมใช้แผนภาพกิจกรรมสำหรับออกแบบกระบวนการสำหรับองค์กรและการพัฒนาซอฟต์แวร์ เพื่ออธิบายตรรกะกระบวนการ กระบวนการทางธุรกิจ (Business process) และกระแสงานต่างๆ ทั้งกระบวนการที่ทำงานเป็นลำดับ หรือทำงานแบบพร้อมกัน (Concurrency) โครงสร้างแผนภาพกิจกรรมประกอบด้วยลำดับและเงื่อนไขต่างๆ ที่ใช้ในการควบคุมพฤติกรรมของระบบ โดยมีสายงานควบคุม (Control flow) และสายงานวัตถุ (Object flow) อธิบายลำดับของการกระทำ (Action) และมีบัพควบคุม (Control node) อธิบายพฤติกรรมของระบบ [7], [8]

3. แนวคิดและวิธีวิจัย

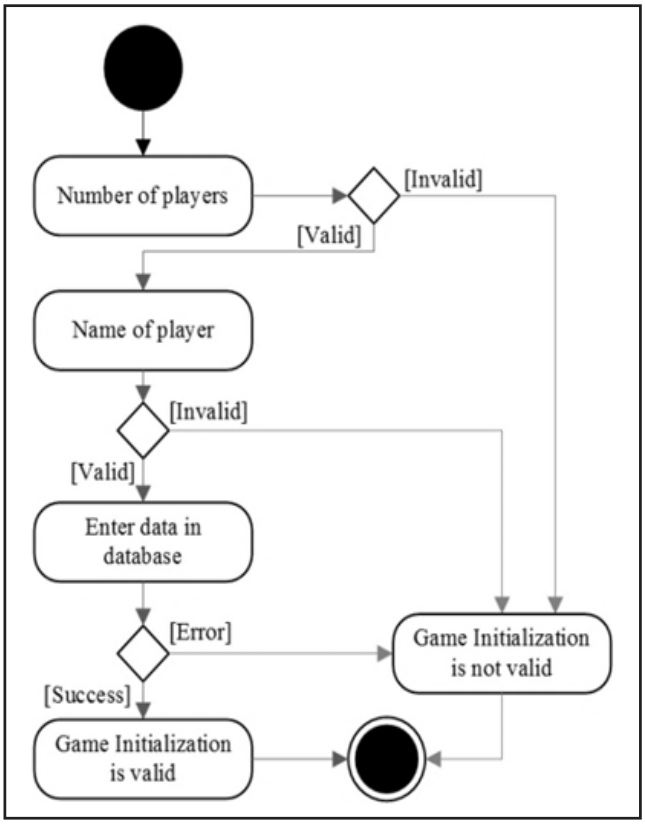
งานวิจัยนี้นำเสนอแนวทางและพัฒนาเครื่องมือสำหรับสร้างกรณีทดสอบแบบอัตโนมัติจากแผนภาพกิจกรรม โดยจะแบ่งการทำงานหลักออกเป็น 4 ส่วน คือ 1) ส่วนนำเข้าเมทาเดตา (Metadata) ของแผนภาพกิจกรรม 2) สร้างแผนภาพต้นไม้การจำแนกเงื่อนไข 3) สร้างกรณีทดสอบด้วยแผนภาพต้นไม้การจำแนกเงื่อนไข และ 4) แปลงกรณีทดสอบที่ได้จากแผนภาพต้นไม้การจำแนกเงื่อนไขให้เป็นแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาด ภาพรวมแนวคิดงานวิจัยแสดงดังภาพที่ 6



ภาพที่ 6 *ภาพรวมของระเบียบวิธีวิจัย*

3.1 ส่วนนำเข้าเมทาเดตาของแผนภาพกิจกรรม

ข้อมูลนำเข้าเป็นเมทาเดตาที่สกัดได้จากแผนภาพกิจกรรมซึ่งสร้างจากอีคลิปโมเดลลิ่งเฟรมเวิร์ก งานวิจัยนี้เลือกใช้ปลั๊กอินอีคลิปส์ ปาปิรุส มาร์ส (Papyrus Mars) สำหรับสร้างแผนภาพกิจกรรม จากนั้นทำการแปลงให้เป็นภาษามาตรฐานสำหรับยูเอ็มแอลซึ่งอยู่ในรูปแบบเอกซ์เอ็มไอ ภาพที่ 7 แสดงแผนภาพกิจกรรมแอปพลิเคชันการเริ่มต้นเกมที่ใช้เป็นกรณีสาธิตวิธีการที่นำเสนอ ภาพที่ 8 แสดงบางส่วนของเอกซ์เอ็มไอที่ได้จากแผนภาพกิจกรรมในภาพที่ 7 ซึ่งไฟล์เอกซ์เอ็มไอจะถูกใช้เพื่อการวิเคราะห์และสร้างต้นไม้การจำแนกเงื่อนไขในขั้นตอนต่อไป



ภาพที่ 7 *แผนภาพกิจกรรมแอปพลิเคชันการเริ่มต้นเกม*

ภาพที่ 8 *ตัวอย่างเอกซ์เอ็มไอที่ได้จากแผนภาพกิจกรรม*

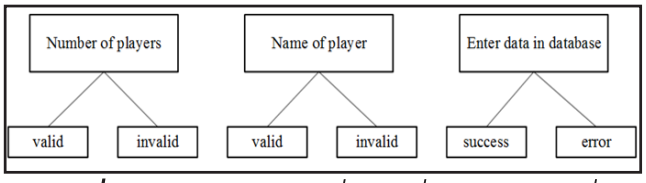
3.2 สร้างแผนภาพต้นไม้การจำแนกเงื่อนไข

วิเคราะห์เมทาเดตาของแผนภาพกิจกรรม โดยสืบค้นแท็กประเภท xmi:type = “uml:DecisionNode” (ตัวอย่างแสดงในภาพที่ 8) เพื่อระบุจุดตัดสินใจซึ่งใช้สร้างต้นไม้การจำแนกเงื่อนไข และกำหนดทางเลือกดำเนินการกิจกรรมถัดไป โดยดำเนินการดังนี้

1. สร้างโหนดพ่อแม่ (Parents Node) โดยการพิจารณาจากจุดตัดสินใจ จุด B (ภาพที่ 2 ซ้าย) จากนั้นจึงสร้างโหนดตั้งต้น B ขึ้นมา

2. สร้างโหนดลูก (Child Node) โดยการพิจารณาจากเส้นทางเงื่อนไขออกจากจุดตัดสินใจ จุด B (ภาพที่ 2 ซ้าย) ประกอบด้วย 2 เส้นทาง คือ b1 และ b2 นำมาสร้างเป็นโหนดลูก b1 และ b2 ใต้โหนดตั้งต้น B (ภาพที่ 2 ขวา)

พิจารณาตัวอย่างแผนภาพกิจกรรมในภาพที่ 7 ซึ่งประกอบด้วย 3 จุดตัดสินใจ จากผลลัพธ์การทำงานโหนดหลัก: จำนวนผู้เล่น (Number of players) ชื่อผู้เล่น (Name of player) และเก็บข้อมูลลงฐานข้อมูล (Enter data in database) การสร้างต้นไม้การจำแนกเงื่อนไขเริ่มจากการพิจารณาจุดตัดสินใจแต่ละจุด และใช้วิธีการข้างต้นเพื่อสร้างต้นไม้จากจุดตัดสินใจทุกๆจุดในแผนภาพกิจกรรม จะได้ผลลัพธ์ดังแสดงในภาพที่ 9



ภาพที่ 9 *ต้นไม้การจำแนกเงื่อนไขที่สร้างจากภาพที่ 7*

3.3 สร้างกรณีทดสอบ

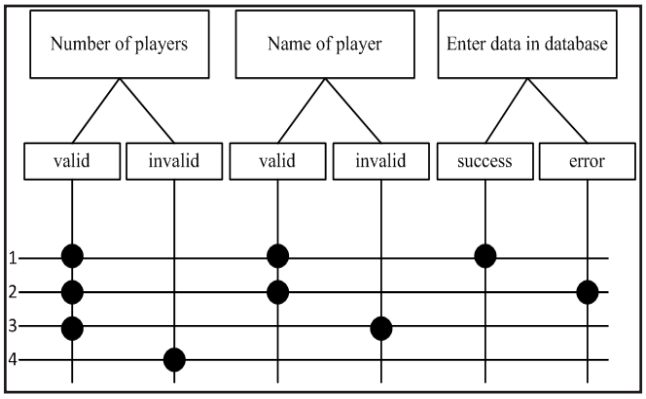
ขั้นตอนนี้จะรวบรวมข้อมูลจากทุกจุดตัดสินใจมาวางเรียงกันตามลำดับเพื่อสร้างเป็นตารางกรณีทดสอบก่อน โดยลากเส้นจากโหนดลูกของต้นไม้ และพิจารณาเงื่อนไขของโหนดลูกจากต้นไม้การจำแนกเงื่อนไขว่าสามารถเลือกจับกับโหนดลูกต้นใดได้ จากนั้นจึงทำจุดลงบนเส้นที่ลากจากโหนดลูก โดยเส้นแถวแนวนอนคือหมายเลขของกรณีทดสอบจะลากผ่านเส้นของโหนดลูกทุกๆ เส้น เพื่อให้ทราบว่าจุดนั้นอยู่ในกรณีทดสอบหมายเลขอะไร

ขั้นตอนต่อไปคือการสร้างกรณีทดสอบ ประกอบด้วย:

1. ระบุกรณีทดสอบแต่ละกรณี โดยมีการระบุหมายเลขหรือชื่อในแต่ละแถวของตารางซึ่งจะพิจารณาจากจุดตัดสินใจ

ที่ได้ถูกแปลงเป็นต้นไม้การจำแนกเงื่อนไข ในที่นี้จะเริ่มจากซ้ายไปขวา การพิจารณาเงื่อนไขจะถูกใช้เพื่อระบุว่าต้นไม้การจำแนกเงื่อนไขสามารถเชื่อมต่อกับกรณีทดสอบ โดยวิเคราะห์จากแผนภาพกิจกรรม (ภาพ 7) และต้นไม้การจำแนกเงื่อนไข (ภาพ 9) จะสามารถสร้างตารางกรณีทดสอบได้ดังภาพที่ 10 เห็นได้ว่า กรณีทดสอบแรกมีการเชื่อมต่อแต่ละโหนด โดยโหนดแรกซึ่งเริ่มจากทางด้านซ้ายจะเชื่อมต่อกับโหนดลูกของจำนวนผู้เล่น

2. กรณีที่มีโหนดลูกในต้นไม้การจำแนกที่ไม่ถูกเลือก หลังจากทำการพิจารณาครบทุกต้นไม้การจำแนกแล้ว สามารถกำหนดให้เป็นโหนดเดียวในกรณีทดสอบนั้นได้ เช่น กรณีทดสอบที่ 4 ในภาพที่ 10



ภาพที่ 10 *แผนภาพต้นไม้การจำแนกเงื่อนไขที่ได้จากแอปพลิเคชันการเริ่มต้นเกม*

ตารางที่ 1 *สรุปเปอร์เซ็นต์ความถูกต้องโมเดลจำแนกกิจกรรม*

Case	players	Name of player	Enter data in database	Result
1	Valid	Valid	Success	Valid
2	Valid	Valid	Error	Invalid
3	Valid	Invalid	don't care	Invalid
4	Invalid	don't care	don't care	Invalid

ตารางที่ 1 สรุปกรณีทดสอบที่ได้จากภาพที่ 10 ทั้งหมด 4 กรณี แบ่งเป็น 1 กรณีที่โปรแกรมทำงานได้อย่างถูกต้อง และ 3 กรณีที่โปรแกรมไม่สามารถทำงานได้

3.4 แปลงกรณีทดสอบเป็นแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาด

ตารางกรณีทดสอบที่ถูกสร้างขึ้นสามารถนำมาใช้วิเคราะห์และสร้างแผนภาพ 2 ชนิด คือ แผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาด ซึ่งทั้งสองแผนภาพนี้จะถูกแสดงในรูปของวงจรเชิงตรรกะ โดยมีจุดประสงค์ คือ แผนภาพต้นไม้ความสำเร็จใช้สำหรับทดสอบความถูกต้องของการทำงาน และแผนภาพต้นไม้ความผิดพลาดใช้ทดสอบความน่าเชื่อถือของซอฟต์แวร์

การสร้างแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาด ประกอบด้วยขั้นตอนดังนี้

1. หาเส้นทางที่ถูกต้อง จากแผนภาพต้นไม้การจำแนกเงื่อนไขและกรณีทดสอบ ซึ่งในตัวอย่างที่ยกมานี้จะเป็นกรณีทดสอบที่ 1 ของภาพที่ 10

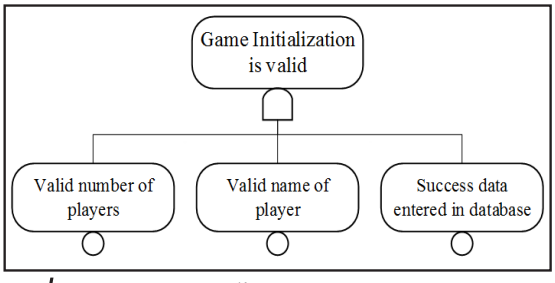
2. สร้างเป้าหมายของแผนภาพต้นไม้ความสำเร็จซึ่งอยู่ด้านบนสุดของแผนภาพด้วยเป้าหมายจากขั้นตอนที่ 1

3. สร้างแผนภาพต้นไม้ความสำเร็จโดยใช้ AND gate เชื่อมต่อกับโหนดในกรณีทดสอบที่ 1 ในกรณีนี้ ข้อมูลนำเข้าประกอบด้วย จำนวนผู้เล่นที่ถูกต้อง ชื่อผู้เล่นที่ถูกต้อง และการบันทึกข้อมูลลงฐานข้อมูลที่สำเร็จ

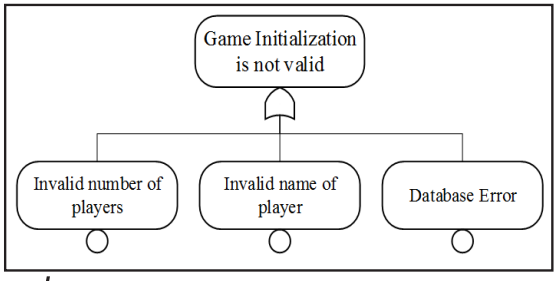
4. กรณีที่เส้นทางที่ถูกต้องมีมากกว่าหนึ่งหรือกรณีทดสอบที่ไปสู่เป้าหมายที่สำเร็จมีมากกว่าหนึ่ง จะต้องนำ OR gate มาใช้เพิ่มเติม

5. สร้างแผนภาพต้นไม้ความผิดพลาดโดยใช้ OR gate ในกรณีนี้คือกรณีทดสอบที่ 2, 3 และ 4

เมื่อประยุกต์ใช้ขั้นตอนข้างต้นกับตัวอย่างโปรแกรมการเริ่มต้นเกม สามารถสร้างแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาดได้ดังภาพที่ 11 และ ภาพที่ 12



ภาพที่ 11 *แผนภาพต้นไม้ความสำเร็จของแอปพลิเคชันการเริ่มต้นเกม [4]*



ภาพที่ 12 *แผนภาพต้นไม้ความผิดพลาดแอปพลิเคชันการเริ่มต้นเกม [4]*

4. การออกแบบและพัฒนาระบบ

4.1 การออกแบบสถาปัตยกรรม

เจเอฟทีเอถูกออกแบบด้วยสถาปัตยกรรมเอ็มวีซี ซึ่งมักนิยมนำมาสร้างเว็บแอปพลิเคชันหรืออินเทอร์เน็ตแอปพลิเคชัน โครงสร้างของสถาปัตยกรรมประกอบด้วย 3 ส่วนประกอบเชิงตรรกะ คือ 1) ส่วนชั้นธุรกิจ ประกอบด้วยโมเดลเชิงตรรกะ 2) ส่วนการแสดงผล ประกอบด้วยวิวเชิงตรรกะ และ 3) ส่วนควบคุมข้อมูลนำเข้า หรือส่วนควบคุมเชิงตรรกะ กล่าวโดยย่อวิวจะแสดงถึงการแสดงผลในเว็บไซต์ ในขณะที่คอนโทรลเลอร์จะรับหรือส่ง จากนั้นประมวลผลด้วยข้อมูลในโมเดลเพื่อแสดงผลต่อไป โดยที่โมเดลจะประกอบไปด้วยกฎธุรกิจสำหรับการประมวลผลและการแสดงผล [9], [10] โครงสร้างดังกล่าวจะช่วยให้นักพัฒนาสามารถทำงานได้อย่างมีประสิทธิภาพ เนื่องจากมีการแบ่งชั้นส่วนที่ชัดเจน กล่าวคือ

โมเดลในเอ็มวีซี แสดงถึงแกนหลักของแอปพลิเคชันที่รองรับข้อมูลเชิงตรรกะ ตัวอย่างเช่น ข้อมูลการซื้อ การคำนวณวันเกิดของสมาชิก หรือ ผลรวมของภาษี เป็นต้น

วิว ในเอ็มวีซี เป็นส่วนจัดการเรื่องการแสดงข้อมูล โดยทั่วไป วิว จะถูกสร้างจากข้อมูลในโมเดล ซึ่งข้อมูลในหนึ่งโมเดลแสดงออกมาได้หลายวิว เนื่องจากวัตถุประสงค์คือการแสดงผลติดต่อกับผู้ใช้งาน

คอนโทรลเลอร์ ในเอ็มวีซี ทำหน้าที่ประมวลผลข้อมูลนำเข้าจากผู้ใช้งานและใช้งานอัลกอริทึมต่างๆ ด้วยวัตถุประสงค์ที่ต่างกันไป โดยแต่ละข้อมูลในโมเดลมีหน้าที่สนับสนุนอัลกอริทึม และคอนโทรลเลอร์สามารถส่งผลตอบสนองให้กับวิวเพื่อแสดงผลให้กับผู้ใช้งานได้อีกด้วย

ข้อจำกัดของระบบในงานวิจัยนี้ ประกอบด้วย

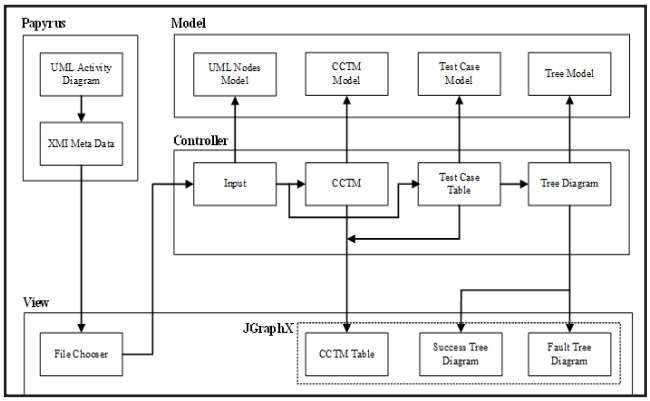
1) โปรแกรม ปาปิรุส มาร์ส ถูกใช้เพื่อแปลงแผนภาพกิจกรรมให้เป็นเมทาเดตาเอกซ์เอ็มไอ

2) เจกราฟเอกซ์ (JGraphX) ได้ถูกนำมาใช้เป็นไลบรารีสำหรับวิวเพื่อการสร้างแผนภาพ

3) ข้อมูลนำเข้าของแผนภาพกิจกรรมจำเป็นต้องมีเป้าหมายที่สำเร็จหนึ่งเดียวและต้องมีการระบุจากผู้ใช้งาน

4) แผนภาพต้นไม้ความผิดพลาดจะถูกสร้างจากจุดตัดสินใจในแผนภาพกิจกรรมเท่านั้น

ภาพที่ 13 แสดงส่วนประกอบระบบที่ออกแบบโดยสถาปัตยกรรมเอ็มวีซีและเครื่องมือที่ใช้ในการพัฒนา



ภาพที่ 13 *สถาปัตยกรรมระบบและเครื่องมือที่ใช้พัฒนา*

ภาพรวมการทำงานของเจเอฟทีเอ ประกอบด้วยขั้นตอนดังนี้

4.1.1 การแปลงข้อมูลนำเข้า

ขั้นตอนแรกปาปิรุสจะแปลงแผนภาพกิจกรรมให้เป็นเอกซ์เอ็มไอเมทาเดตา ซึ่งเอกซ์เอ็มไอจะประกอบไปด้วยข้อมูลของแผนภาพกิจกรรมที่อยู่ในรูปแบบของเอกซ์เอ็มแอล แต่ละเอกซ์เอ็มแอลประกอบด้วยคุณลักษณะของ ชื่อ รหัสเป้าหมาย และต้นกำเนิด โดยเอกซ์เอ็มแอลจะมีสองประเภทด้วยกันคือ แบบโหนด และเส้นการควบคุม

4.1.2 คอนโทรลเลอร์ - โมเดล

ลำดับต่อมาเมื่อได้เอกซ์เอ็มไอเมทาเดตาแล้ว โปรแกรมจะทำหน้าที่แปลงข้อมูลเป็นโมเดลของแผนภาพต้นไม้การจำแนกเงื่อนไข ดังแสดงในส่วนคอนโทรลเลอร์ของภาพที่ 13 ซึ่งประกอบด้วย 1) ข้อมูลนำเข้า 2) ต้นไม้การจำแนกเงื่อนไข 3) กรณีทดสอบ และ 4) แผนภาพต้นไม้ โดยที่แต่ละส่วนจะแสดงถึงคอนโทรลเลอร์คลาส และโมเดลของแต่ละตัว ซึ่งจะถูกประมวลผลวิเคราะห์ตามลำดับดังนี้

1. ข้อมูลนำเข้า

วัตถุประสงค์ของส่วนนี้คือ การแปลงเอกซ์เอ็มไอไฟล์กลายเป็นอ็อบเจกต์ ซึ่งมีโครงสร้างคล้ายคลึงกับ ยูเอ็มแอลโหนด โดยที่โครงสร้างของยูเอ็มแอลโหนดนั้นสามารถสนับสนุนโหนดได้หลายประเภท เช่น โหนดการตัดสินใจ โหนดเริ่มต้น โหนดกิจกรรม และโหนดสิ้นสุด ซึ่งแต่ละโหนดนั้นควรจะประกอบไปด้วยคำอธิบายที่แสดงถึง ชื่อ เป้าหมาย ต้นทาง และคุณลักษณะโดยทั่วไป โดยข้อมูลนำเข้าควรจัดเก็บอยู่ในรูปแบบกลุ่มของตัวแปร

2. ต้นไม้การจำแนกเงื่อนไข

ส่วนประกอบนี้มีหน้าที่สร้างกรณีทดสอบด้วยวิธี

แผนภาพต้นไม้การจำแนกเงื่อนไข จากจุดตัดสินใจซึ่งสกัดได้จากข้อมูลนำเข้า และแปลงเป็นต้นไม้การจำแนกเงื่อนไข ซึ่งรากของแต่ละต้นไม้การจำแนกเกิดจากจุดตัดสินใจที่มีความสัมพันธ์ของโหนดตัดสินใจนั้นๆ และจุดตัดสินใจทั้งหมดควรจะถูกประมวลผล

3. กรณีทดสอบ

หลักการสร้างกรณีทดสอบของแผนภาพต้นไม้การจำแนกเงื่อนไขนั้นเป็นพื้นฐานสำคัญในการสร้างแผนภาพต้นไม้ความผิดพลาดในงานวิจัยนี้ ส่วนดังกล่าวจะมีลักษณะการทำงานหรือการคำนวณที่ให้ผลลัพธ์แบบย้อนกลับหรือทำซ้ำขั้นตอนเดิม โดยผลลัพธ์จะถูกเก็บในรูปแบบตัวแปรอ็อบเจกต์

4. แผนภาพต้นไม้

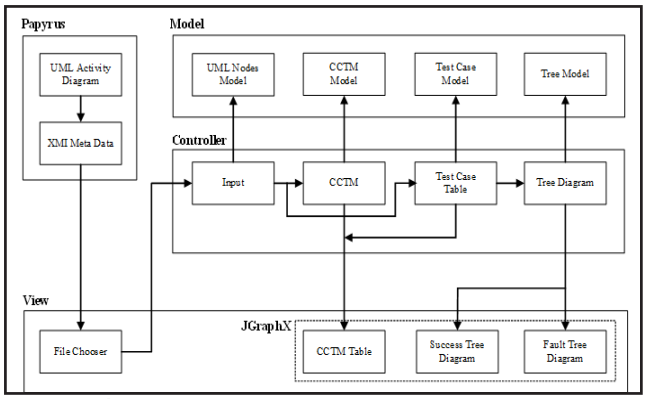
ขั้นต่อมาผลลัพธ์ทั้งหมดจากส่วนที่กล่าวมาข้างต้นจะถูกนำไปใช้สร้างแผนภาพต้นไม้ความสำเร็จและแผนภาพต้นไม้ความผิดพลาดซึ่งเป็นผลลัพธ์สุดท้าย

4.1.3 วิว

ทำหน้าที่ติดต่อและแสดงผลแผนภาพต้นไม้ ระบบที่พัฒนาขึ้นใช้โปรแกรมเจกราฟเอกซ์ เป็นตัวช่วยในการสร้างแผนภาพต้นไม้ต่างๆ เจกราฟเอกซ์ เป็นซอฟต์แวร์โอเพนซอร์ส มีข้อดีคือ มีความยืดหยุ่นในการสร้างแผนภาพและการปรับปรุง รวมทั้งสามารถสร้างสัญลักษณ์ตามที่ต้องการ โดยเฉพาะการสร้างสัญลักษณ์ OR gate และ AND gate

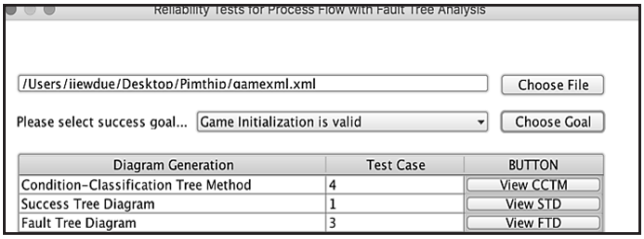
4.2 การพัฒนาระบบ

เจเอฟทีเอ ถูกพัฒนาขึ้นด้วยภาษาจาวา โดยใช้ปลั๊กอินอีคลิปส์ ปาปิรุส มาร์สในการสร้างแผนภาพกิจกรรมและแปลงเป็นข้อมูลนำเข้าในรูปแบบเอกซ์เอ็มไอเมทาเดตาของแผนภาพกิจกรรม การแสดงผลลัพธ์แผนภาพต้นไม้ต่างๆ จะใช้ไลบรารี เจกราฟเอกซ์สำหรับสร้างส่วนต่อประสานกราฟิกกับผู้ใช้ ภาพที่ 14 แสดงแผนภาพกิจกรรมที่สร้างจากปาปิรุส ซึ่งใช้เป็นตัวอย่างสาธิตการทำงานของระบบที่พัฒนาขึ้น

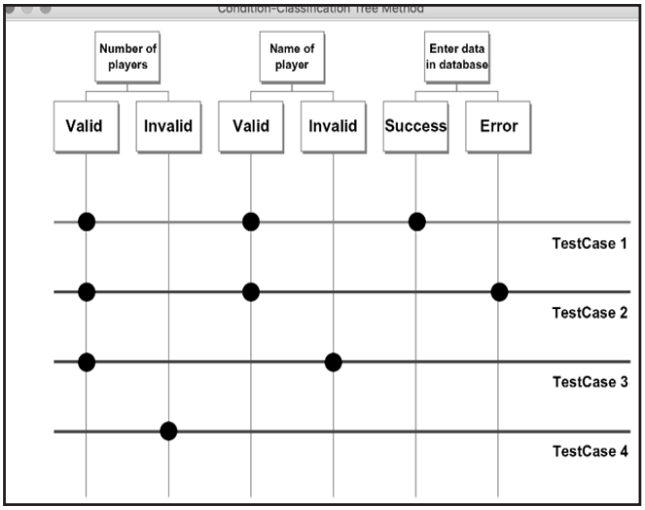


ภาพที่ 14 *ตัวอย่างแผนภาพกิจกรรมที่ได้จากปาปิรุส*

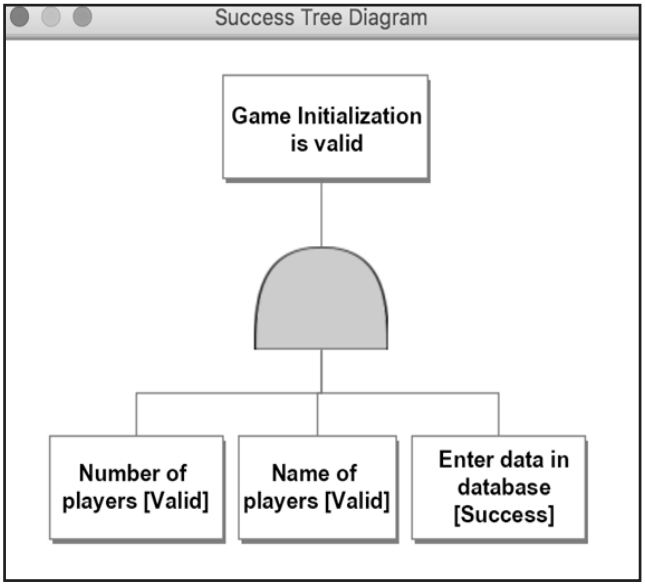
เมื่อได้เอกซ์เอ็มไอไฟล์มาแล้ว ขั้นแรกส่วนรับข้อมูลจะทำการเลือกไฟล์และเลือกเป้าหมาย จากนั้นระบบจะแสดงจำนวนกรณีทดสอบที่เป็นไปได้ ดังภาพที่ 15



ภาพที่ 15 *หน้าจอแสดงจำนวนกรณีทดสอบ*



ภาพที่ 16 *แผนภาพต้นไม้การจำแนกเงื่อนไขที่สร้างจากระบบ*



ภาพที่ 17 *แผนภาพต้นไม้ ความสำเร็จที่สร้างจากระบบ*

จากหน้าจอภาพที่ 15 เมื่อระบบทำการประมวลผลเสร็จเรียบร้อย สามารถดูแผนภาพต้นไม้การจำแนกเงื่อนไข แผนภาพต้นไม้ความสำเร็จ และแผนภาพต้นไม้ความผิดพลาดได้ดังภาพที่ 16-18

5. สรุปผลและข้อเสนอแนะการวิจัย

งานวิจัยนี้นำเสนอวิธีการและพัฒนาเครื่องมือสร้างกรณีทดสอบความน่าเชื่อถือแบบอัตโนมัติในโดเมนวิศวกรรมซอฟต์แวร์ โดยใช้เทคนิควิเคราะห์ต้นไม้ความผิดพลาดบนพื้นฐานของการสร้างแบบจำลองต้นไม้การจำแนกเงื่อนไข ซึ่งนิยมใช้กันแพร่หลายในระดับสากล ระบบต้นแบบหรือเจเอฟทีเอที่พัฒนาขึ้นในงานวิจัยนี้จะช่วยลดต้นทุนค่าใช้จ่ายและความผิดพลาดของมนุษย์ในการสร้างกรณีทดสอบ ความถูกต้องและความล้มเหลวของการทำงานของระบบซอฟต์แวร์

การประเมินผลงานวิจัยนี้ได้ทดลองเปรียบเทียบผลลัพธ์ของการสร้างกรณีทดสอบ และแผนภาพหรือผลลัพธ์ที่ได้จากกรณีทดสอบจากแผนภาพกิจกรรมที่นำมาทดสอบทั้งหมด 8 แผนภาพ ด้วยวิธีการสร้างด้วยมือและจากงานวิจัยอื่นๆ พบว่าโปรแกรมสามารถสร้างกรณีทดสอบได้อย่างถูกต้องครบถ้วนและรวดเร็วกว่าวิธีการทำด้วยมือ ซึ่งจะช่วยลดการใช้ทรัพยากรเวลาและแรงงานของโครงการ รวมทั้งอำนวยความสะดวกต่อการจัดทำเอกสารในลำดับต่อไป

อย่างไรก็ตาม เจเอฟทีเอ สนับสนุนการสร้างข้อมูลนำเข้าเมทาเดตาแผนภาพกิจกรรม ด้วยปาปิรุส มาร์ส 1.1.4 เท่านั้น นอกจากนี้ เครื่องมือสามารถประมวลผลได้เฉพาะจุดตัดสินใจบนแผนภาพกิจกรรม และจะต้องเลือกเป้าหมายที่ถูกต้องเพื่อให้ได้ผลลัพธ์ที่ถูกต้อง ข้อเสนอแนะเพื่อพัฒนาต่อยอดงานวิจัยนี้ ได้แก่ การขยายขอบเขตความสามารถระบบเพื่อให้รองรับส่วนประกอบแผนภาพกิจกรรมให้ครบถ้วนทุกประเภท ปรับปรุงส่วนแสดงผลแผนภาพต้นไม้ให้รองรับกรณีทดสอบจำนวนมาก รวมทั้งพัฒนาผลลัพธ์จากงานวิจัยนี้ให้อยู่ในรูปแบบไฟล์เอกซ์เอ็มแอลเพื่อให้สามารถนำไปใช้ในแอปพลิเคชันอื่นๆ ต่อไปได้

6. เอกสารอ้างอิง

[1] M. Grochtmann and K. Grimm. “Classification trees for partition testing.” *Software Testing, Verification and Reliability*, Vol. 3, pp. 63-82, 1993.

[2] C. Ericson. “Fault Tree Analysis - A History.” *In The Proceedings of The 17th International System Safety Conference*, pp. 1-9, 1999.

[3] S. Kansomkeat, P. Thiket, and J. Offutt. “Generating test cases from UML activity diagrams using the Condition-Classification Tree Method.” *In Software Technology and Engineering (ICSTE) 2nd International Conference on*, 2010, pp. V1-62-V1-66, 2010.

[4] S. Tiwari and A. Gupta. “An Approach to Generate Safety Validation Test Cases from UML Activity Diagram.” *In Software Engineering Conference (APSEC) 20th Asia- Pacific Conference on, 2013*, pp. 189-198, 2013.

[5] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz. “A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System.” *Requirements Engineering*, Vol. 7, pp. 207-220, 2002.

[6] K. M. Hansen, A. P. Ravn, and V. Stavridou. “From safety analysis to software requirements.” *IEEE Transactions on Software Engineering*, Vol. 24, pp. 573-584, 1998.

[7] OMG Unified Modeling Language (OMG UML). in Superstructure Version 2.3, ed: Object Management Group 2011.

[8] D. Tegarden, A. Dennis, and B. H. Wixom. “Systems Analysis and Design with UML.” *Hoboken,* N.J.: Wiley, 2013.

[9] Tutorialspoint.com. *MVC Framework - Introduction*. Available Online at http://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm, accessed on 12 April 2016.

[10] J. Atwood. *Understanding Model-View-Controller*. Available Online at https://blog.codinghorror.com/understanding-model-view-controller., accessed on 23 March 2016.