การระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์

นายสิทธิพล ลิ้มชัยชะดา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2557

IDENTIFYING COMMON ASSET CANDIDATES IN SOURCE CODE BY COMPARING SOFTWARE TESTING PATH

Mr. Sitthipon Limchaichada

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 2014

หัวข้อวิทยานิพนธ์	การระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์
โดย	นายสิทธิพล ลิ้มชัยชะดา
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

.....กรรมการ
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.มฑุปายาส ทองมาก)

สิทธิพล ลิ้มชัยชะดา : การระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ (IDENTIFYING COMMON ASSET CANDIDATES IN SOURCE CODE BY COMPARING SOFTWARE TESTING PATH) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.พรศิริ หมื่นไชยศรี, 96 หน้า.

การระบุตัวแทนสินทรัพย์ทั่วไปเป็นการนำเอาส่วนประกอบร่วมของผลิตภัณฑ์ในอดีตที่มีความสามารถในการตอบสนองต่อความต้องการในขอบเขตของการทำงานเดียวกันมาแปลงให้เป็นสินทรัพย์หลักเพื่อตอบสนองความหลากหลายของผลิตภัณฑ์ สำหรับนำไปใช้ซ้ำในการประกอบเป็นผลิตภัณฑ์ที่อยู่บนความต้องการเดียวกัน

ในปัจจุบันนั้นการระบุตัวแทนสินทรัพย์ทั่วไปในประเภทของซอร์สโค้ดที่มีการนิยามเชิงพฤติกรรมที่เหมือนกันแต่วิธีการเขียนที่แตกต่างกันนั้นยังมีข้อจำกัด เนื่องจากเทคนิคการตรวจหาสำเนาโค้ดโดยส่วนใหญ่อาศัยความสัมพันธ์ของโครงสร้าง ในการตรวจหา ซึ่งไม่สามารถตรวจหาสำเนาโค้ดที่มีการเขียนที่แตกต่างกันแต่มีพฤติกรรมการทำงานที่เหมือนกัน

งานวิจัยนี้มีจุดประสงค์เพื่อออกแบบวิธีการแก้ปัญหาในการระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดดังกล่าวโดยวิธีการตรวจหาความเหมือนของโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4) โดยใช้เส้นทางของวิธีการทดสอบซอฟต์แวร์

จากผลการทดลองพบได้ว่า การระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดด้วยวิธีดังกล่าวสามารถสร้างตัวแทนสินทรัพย์ทั่วไปเพื่อนำมาเป็นสินทรัพย์หลักในการพัฒนาผลิตภัณฑ์ที่มีลักษณะการทำงานที่เหมือนกันของเมท็อดที่จะเกิดขึ้นในอนาคตได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2557

5470416721 : MAJOR SOFTWARE ENGINEERING
KEYWORDS: SOFTWARE TESTING / CORE ASSET

SITTHIPON LIMCHAICHADA: IDENTIFYING COMMON ASSET CANDIDATES IN SOURCE CODE BY COMPARING SOFTWARE TESTING PATH. ADVISOR: ASSOC. PROF. PORNSIRI MUENCHAISRI, Ph.D., 96 pp.

Identifying common assets is to find common components of the existing products with same requirements in the scope of work and then convert them into core assets for software reuse.

Currently, there are several approaches that can identify common assets from source code fragments but they have limitations. They mainly look at the structure of the code fragments which cannot be used to detect different code fragments that have the same behavior.

The purpose of this research is to design an approach to identify a common asset of source code fragments that perform the same function but have different source code (Type 4) by comparing software testing paths. A tool supporting the approach is developed.

The result shows that the proposed approach can specify common assets from different code with the same behavior. With the use of the approach and its tool, identified common assets may be reused and software development will be faster in the future.

Department: Computer Engineering Student's Signature .....
Field of Study: Software Engineering Advisor's Signature .....
Academic Year: 2014

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ประสบความสำเร็จและครบถ้วนสมบูรณ์ได้ด้วยความร่วมมือและความช่วยเหลือจาก รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี ผู้ซึ่งเป็นอาจารย์ที่ปรึกษาที่คอยให้คำปรึกษาทั้งทางด้านวิชาการและความรู้ต่างๆ รวมไปจนกระทั่งด้านคุณธรรมจริยธรรม ซึ่งถือเป็นสิ่งสำคัญที่ทำให้งานวิจัยและวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้

ขอขอบพระคุณ รองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์ ประธานกรรมการสอบวิทยานิพนธ์ รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา และ ผู้ช่วยศาสตราจารย์ ดร.มฑุ ปายาส ทองมาก กรรมการสอบวิทยานิพนธ์ สำหรับข้อชี้แนะและแนวทางต่างๆ ที่ช่วยเพิ่มพูนให้งานวิจัยมีคุณภาพและประสิทธิภาพที่ดีขึ้น

ขอขอบพระคุณคณาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยทุกท่าน ที่ให้ความรู้และคำสั่งสอนแก่ผู้วิจัย

ขอขอบคุณบุคลากรในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยทุกท่าน ที่คอยให้ข้อมูล คำปรึกษา และความช่วยเหลือต่างๆ ในระหว่างการสอบวิทยานิพนธ์ให้สำเร็จลุล่วงไปด้วยดี

สุดท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา และญาติๆ ที่คอยสนับสนุนและให้กำลังใจแก่ผู้วิจัยเสมอมา

สารบัญ

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

สารบัญรูปภาพ

ภาคผนวก

บทที่ 1
บทนำ

ในบทนี้จะกล่าวถึงภาพรวมของงานวิจัยการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ ซึ่งประกอบด้วย ที่มาและความสำคัญของการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ ปัญหา วัตถุประสงค์ ขอบเขต ประโยชน์ ขั้นตอนและวิธีดำเนินการ ประโยชน์ที่คาดว่าได้รับ โครงสร้างของเนื้อหาในวิทยานิพนธ์ ผลงานที่ตีพิมพ์

1.1 ที่มาและความสำคัญของปัญหา

การพัฒนาสินทรัพย์หลัก (Core asset development) [1] เป็นหนึ่งในกิจกรรมที่มีความสำคัญมากในการสนับสนุนให้เกิดกระบวนการในการผลิต ให้มีความหลากหลาย (variety of product) และปรับให้เข้ากับผู้บริโภคที่มีความหลากหลายได้เป็นอย่างดี (highly customizable) โดยการพัฒนาสินทรัพย์หลักนั้นเป็นการนำเอาส่วนประกอบร่วม (Common component) ของผลิตภัณฑ์ในอดีตที่มีความสามารถในการตอบสนองต่อความต้องการในขอบเขตของการทำงานเดียวกันมาแปลงให้เป็นสินทรัพย์หลัก (core asset) โดยอยู่บนพื้นฐานของการนำซอฟต์แวร์มาใช้ซ้ำ (Software reuse) ซึ่งเป็นการสนับสนุนการพัฒนาซอฟต์แวร์ให้มีการนำกลับมาใช้ใหม่อีกครั้งอย่างเป็นระบบ

ในการพัฒนาสินทรัพย์หลัก (Core asset development) [1] นั้นจะมีหลักการของการพัฒนาออกเป็น 2 วิธี ได้แก่รูปแบบหลักของการพัฒนาสินทรัพย์ในแนวทางเชิงเชิงรุก (proactive approach) คือ กรณีที่พัฒนาสินทรัพย์หลักขึ้นมาก่อนแล้วจึงค่อยนำมาใช้ในผลิตภัณฑ์ และแนวทางเชิงปฏิกิริยา (Reactive approach) คือ กรณีที่ผลิตภัณฑ์พัฒนาออกมาก่อนแล้วค่อยสกัดในการพัฒนาสินทรัพย์หลักออกมา อย่างไรก็ตามการพัฒนาในแนวทางเชิงรุกนั้นค่อนข้างที่จะทำได้ยาก เนื่องมาจากจะต้องทำนายอนาคตในการผลิตของผลิตภัณฑ์นั้นๆ ซึ่งเราไม่สามารถรู้ได้เลยว่าจะประสบความสำเร็จกับผลิตภัณฑ์ที่มีลักษณะเดียวกันมาหรือไม่ แต่ แนวทางเชิงปฏิกิริยานั้นจะมีความยากในส่วนของเอกสารแล้วความล้าสมัยของการออกแบบทำให้เราสามารถศึกษาจากซอร์สโค้ดเป็นหลักโดยส่วนใหญ่ [2],[3]

ในแนวทางการพัฒนาเชิงปฏิกิริยา (Reactive approach) สามารถจัดการกับความยากของเอกสารและความล้าสมัยของการออกแบบโดยการการศึกษาจากซอร์สโค้ดโดยใช้เทคนิคของการตรวจสอบหาความตรงกัน (Matching Technique) [4] ซึ่งเป็นเทคนิคที่ใช้ในการวิเคราะห์เวอร์ชันของซอฟต์แวร์ซึ่งมีเทคนิคที่ได้ทำการสำรวจในงานวิจัยที่ผ่านมา ได้แก่ Entity Name Matching, String Matching, Syntax Tree Matching, Control Flow Graph Matching, Program Dependence Graph Matching, Binary Code Matching, Clone Detection, Origin Analysis Tools โดยแต่ละเทคนิคนั้นมีข้อดีข้อเสียและผลกระทบที่แตกต่างกัน

การตรวจหาสำเนาโค้ดเป็นหนึ่งในเทคนิคที่มีความเหมาะสมต่อการระบุส่วนร่วมของผลิตภัณฑ์ โดยใช้วิธีการ ตรวจหาชิ้นส่วนโค้ดที่มีลักษณะเหมือนหรือคล้ายกัน ที่เรียกว่า สำเนาโค้ด ซึ่งโดยส่วนใหญ่มักพบได้ทั่วไปในการพัฒนาซอฟต์แวร์ [5],[6] โดยการตรวจหาสำเนาโค้ดถูกนำไปใช้ในหลายขอบเขตและวัตถุประสงค์ เช่น การตรวจหาการลอกเลียนวรรณกรรม (Plagiarism Detection) การตรวจหาโค้ดซ้ำซ้อนในระบบ (Duplicated code reduction) เป็นต้น โดยส่วนใหญ่สำเนาโค้ดมัก จะเกิดจากการสำเนาและแปะ แต่นักเขียน โปรแกรมมักมีแนวทางในการเขียนที่มีความแตกต่างกันออกไปทำให้เกิดความแตกต่างระหว่างคู่สำเนาโค้ด ซึ่งเป็นอุปสรรคต่อการสร้างสินทรัพย์ทั่วไปจากสำเนาโค้ดที่ตรวจหาได้

อย่างไรก็ตามในบทความ [5],[6] ได้ทำการจำแนกประเภทของสำเนาโค้ดแบ่งออกเป็น 4 ประเภท ซึ่งได้แก่ สำเนาโค้ดประเภทที่ 1 คือโค้ดที่ถูกเขียนเหมือนกันมาโดยสมบูรณ์ ไม่มีความแตกต่างใดๆ นอกเหนือจากหมายเหตุ และ ช่องว่าง สำเนาโค้ดประเภทที่ 2 ชิ้นส่วนของโค้ดมีความคล้ายกันในลักษณะเชิงโครงสร้าง แต่จะมีการเปลี่ยนแปลงในส่วนของชื่อหรอตัวแปรของข้อมูล ในแต่ละบรรทัดให้มีความแตกต่าง และยังรวมไปถึง ช่องว่างในบรรทัด และหมายเหตุที่กำกับไว้ สำเนาโค้ดประเภทที่ 3 คือโค้ดมีการเปลี่ยนแปลง อันได้แก่การเพิ่มข้อความของคำสั่ง ลบข้อความของคำสั่ง หรือเปลี่ยนแปลงข้อความของคำสั่ง เพื่อทำให้เกิดความแตกต่างในแต่บรรทัด ในชิ้นส่วนของโค้ด สำเนาโค้ดประเภทที่ 4 ชิ้นส่วนโค้ดที่ใช้หลักการหรือรูปแบบของเขียนที่มีความแตกต่างกัน แต่ให้ผลลัพธ์หรือพฤติกรรมของการทำงานที่เหมือนกัน เช่น for<>recursive , if-else<>switch case , while<>recursive เป็นต้น

ต่อมาในบทความ [7] ได้มีวิธีการออกแบบวิธีการและพัฒนาเครื่องมือระบุตัวเลือกสินทรัพย์ร่วมจากซอร์สโค้ดระหว่างผลิตภัณฑ์โดยประยุกต์ใช้เทคนิคการตรวจหาสำเนาโค้ดในการระบุส่วนร่วมและใช้แบบอย่างการออกแบบประเภทเมท็อดแม่เพื่อแยกความแตกต่างออกจากสำเนาโค้ดที่ตรวจพบและก่อแบบเป็นตัวเลือกสินทรัพย์ร่วม อย่างไรก็ตามบทความ[7]เสนอเพียงการระบุตัวเลือกสินทรัพย์ร่วมจากสำเนาโค้ดประเภทที่ 3 เท่านั้นคือ สำเนาโค้ดที่ทำการเปลี่ยนแปลง เพิ่มและ ลบคำสั่ง แต่ไม่รวมสำเนาโค้ดประเภทที่ 4 ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน ที่ยังคงเหลืออยู่ในซอฟต์แวร์และไม่ได้มีการกระทำใดๆกับสำเนาโค้ดเหล่านี้เลย

งานวิจัยนี้ได้นำเสนอวิธีการแก้ปัญหาในการระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดดังกล่าวโดยวิธีการตรวจหาความเหมือนของโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4) โดยใช้วิธีการตรวจสอบหาความเหมือนกันของ กราฟการไหลของข้อมูล(Control Flow Graph) โดยวิธีการสร้าง กราฟการไหลของข้อมูล (Control Flow Graph) จาก โค้ด แล้วหาเส้นทางทั้งหมดที่เกิดขึ้นโดยวิธีการจำแนกเส้นทางของการทดสอบ (Software Testing Path) หลังจากนั้นทำการเปรียบเทียบพฤติกรรมที่เกิดขึ้น แล้วทำการก่อตัวแบบตัวเลือกสินทรัพย์ เพื่อใช้สำหรับการพัฒนาผลิตภัณฑ์ที่มีลักษณะการทำงานที่เหมือนกันของเมท็อดที่จะเกิดขึ้นในอนาคต

1.2 ปัญหาของงานวิจัย

สำเนาโค้ดประเภทที่ 4 ที่ชิ้นส่วนโค้ดใช้วิธีการเขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกันนั้นไม่ยังถูกยกมารวมไว้ ณ งานวิจัยในอดีตเนื่องจากเทคนิคการตรวจหาสำเนาโค้ดโดยส่วนใหญ่อาศัยความสัมพันธ์ของโครงสร้าง หรือวากยสัมพันธ์ของโค้ดในการตรวจหา และสำเนาโค้ดประเภทที่ 4 นั้นต้องตรวจหาสำเนาโค้ดในนิยามเชิงพฤติกรรม เนื่องจากวิธีการเขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน ทำให้ไม่สามารถตรวจสอบเชิงโครงสร้างได้

1.3 วัตถุประสงค์ของงานวิจัย

1) ออกแบบวิธีการระบุสินทรัพย์ทั่วไปจากซอร์สโค้ดระหว่างสองผลิตภัณฑ์ที่ใช้วิธีการเขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4)

2) พัฒนาเครื่องมือเพื่อสนับสนุนการออกแบบวิธีการระบุสินทรัพย์ทั่วไปจากสำเนาโค้ดประเภทที่ 4

1.4 ขอบเขตของงานวิจัย

1) งานวิจัยนี้ครอบคลุมการออกแบบวิธีการระบุสินทรัพย์ทั่วไปจากซอร์สโค้ดระหว่างสองผลิตภัณฑ์ที่ใช้วิธีการ เขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4) เท่านั้น

2) พัฒนาเครื่องมือเพื่อสนับสนุนการออกแบบวิธีการระบุสินทรัพย์ทั่วไปจากสำเนาโค้ดประเภทที่ 4

3) ข้อมูลนำเข้าที่ใช้ คือ ซอร์สโค้ดของผลิตภัณฑ์ที่จะนำมาเปรียบเทียบในภาษาจาวา(Java) ที่ได้ผ่านการ compile

4) ข้อมูลส่งออกของงานวิจัยนี้ ผลลัพธ์ของการตรวจหาส่วนทั่วไปในสำเนาโค้ดประเภทที่ 4 เพื่อใช้ในการระบุ สินทรัพย์ทั่วไป

5) ทำการประเมินวิธีการระบุสินทรัพย์ทั่วไปจากซอร์สโค้ดระหว่างสองผลิตภัณฑ์ที่ใช้วิธีการเขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4) โดยการนำซอฟต์แวร์สำโค้ดประเภทที่ 4 มาทำการทดสอบกับเครื่องมือที่พัฒนาขึ้นโดยข้อมูลส่งออกที่ได้จะต้องค้นหาคำสั่งทีทำงานที่เหมือนกันออกมา ในกรณีที่มีจุดแตกต่างจะต้องแยกจุดแตกต่างดังกล่าวออกมา

1.5 ประโยชน์ของงานวิจัย

1) ลดภาระของผู้เชี่ยวชาญในการวิเคราะห์โค้ดของผลิตภัณฑ์เพื่อระบุตัวแทนสินทรัพย์ทั่วไป

2) นำเสนอแนวทางจัดการกับชิ้นส่วนโค้ดที่มีวิธีการเขียนที่แตกต่างกันแต่ให้พฤติกรรมที่เหมือนกันที่ตรวจหาได้ เพื่อใช้สร้างเป็นตัวแทนสินทรัพย์ทั่วไป

1.6 ขั้นตอนและวิธีดำเนินการวิจัย

1) ศึกษางานวิจัยที่เกี่ยวข้องกับรูปแบบของการสร้างสินทรัพย์หลักของผลิตภัณฑ์ซอฟต์แวร์

2) ศึกษาขั้นตอนของเทคนิคการตรวจหาสำเนาโค้ดเชิงโครงสร้างและพฤติกรรมที่มีการวิจัยอยู่ในปัจจุบัน

3) ศึกษาประเภทและขั้นตอนของเทคนิคการเปรียบเทียบกราฟการไหลของการควบคุมที่มีอยู่ในปัจจุบัน

4) ศึกษางานวิจัยที่เกี่ยวกับการใช้นำเทคนิคการตรวจหาสำเนาโค้ดที่นำมาใช้ตรวจสอบเชิงพฤติกรรม

5) ศึกษาทฤษฎี และเทคนิคอื่นๆ ที่จะนำมาใช้ร่วมกับงานวิจัย

6) กำหนดหัวข้อรายละเอียดของวิธีการและขั้นตอนงานวิจัย

7) ทำการทดลองวิธีการที่ได้ทำการวิจัยกับตัวอย่างซอฟต์แวร์

8) วิเคราะห์ผลลัพธ์ที่ได้จากการทดลองวิธีการที่ได้ทำการวิจัยกับตัวอย่างซอฟต์แวร์

9) เขียนบทความงานวิจัย

1.7 ประโยชน์ที่คาดว่าได้รับ

1) ลดภาระของผู้เชี่ยวชาญในการวิเคราะห์โค้ดของผลิตภัณฑ์เพื่อระบุตัวแทนสินทรัพย์ทั่วไป

2) นำเสนอแนวทางจัดการกับชิ้นส่วนโค้ดที่มีวิธีการเขียนที่แตกต่างกันแต่ให้พฤติกรรมที่เหมือนกันที่ตรวจหาได้ เพื่อใช้สร้างเป็นตัวแทนสินทรัพย์ทั่วไป

1.8 โครงสร้างของเนื้อหาในวิทยานิพนธ์

เนื้อหาของวิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 6 บทคือ บทที่ 1 เป็นบทนำ ซึ่งเป็นบทที่กำลังกล่าวถึงอยู่ในขณะนี้ บทที่ 2 เป็นการกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 เป็นบทที่อธิบายถึงการแก้ปัญหาในการระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดดังกล่าวโดยวิธีการตรวจหาความเหมือนของโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4 บทที่ 4 เป็นบทที่อธิบายการออกแบบและพัฒนาเครื่องมือตามแนวคิดที่นำเสนอ บทที่ 5 เป็นบทที่อธิบายการทดลองและผลการทดลองของแนวคิดที่นำเสนอ รวมถึงการวิเคราะห์ผลการ

ทดลอง และบทที่ 6 เป็นบทสุดท้าย ซึ่งจะเป็นบทสรุปและข้อจำกัดของงานวิจัย รวมทั้งงานวิจัยในอนาคต และบทความวิชาการที่ตีพิมพ์

1.9 ผลงานที่ตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “Identifying Common Asset Candidates In Source Code By Comparing Software Testing Path” ในงานประชุมวิชาการ “The 10th National Conference on Computing and Information Technology” ณ จังหวัดภูเก็ต ประเทศไทย ระหว่างวันที่ 8-9 พฤษภาคม 2557

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะอธิบายถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง ซึ่งเป็นส่วนที่สำคัญสำหรับนำไปประยุกต์ใช้กับงานวิจัยฉบับนี้ บทนี้ประกอบด้วยสองส่วน ส่วนแรกเป็นการศึกษาทฤษฎีและหลักการทางวิศวกรรมซอฟต์แวร์ที่เกี่ยวข้องที่เกี่ยวกับการตรวจสอบเชิงโครงสร้างและพฤติกรรมอันได้แก่ สายผลิตภัณฑ์ซอฟต์แวร์และการพัฒนาสินทรัพย์ การตรวจหาสำเนาโค้ด กราฟการไหลของการควบคุม Decision-to-Decision path การทดสอบเส้นทาง เมท็อดแม่แบบ และส่วนที่สองเป็นการศึกษางานวิจัยที่เกี่ยวข้องและสอดคล้องกับงานวิจัยการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ โดยจะแสดงรายละเอียดของทั้งสองส่วนได้ดังนี้

2.1 ความรู้และทฤษฎีที่เกี่ยวข้อง

ในส่วนนี้อธิบายเกี่ยวกับความรู้พื้นฐาน ทฤษฎี และหลักการทางวิศวกรรมซอฟต์แวร์ที่เกี่ยวข้องกับงานวิจัยทั้งหมดด้วยกัน 6 ส่วนย่อยๆ คือ สายผลิตภัณฑ์ซอฟต์แวร์และการพัฒนาสินทรัพย์ การตรวจหาสำเนาโค้ด กราฟการไหลของการควบคุม [8] Decision-to-Decision path การทดสอบเส้นทาง [9] เมท็อดแม่แบบ โดยจะแสดงรายละเอียดของแต่ละส่วนดังนี้

2.1.1 สายผลิตภัณฑ์ซอฟต์แวร์และการพัฒนาสินทรัพย์ [1]

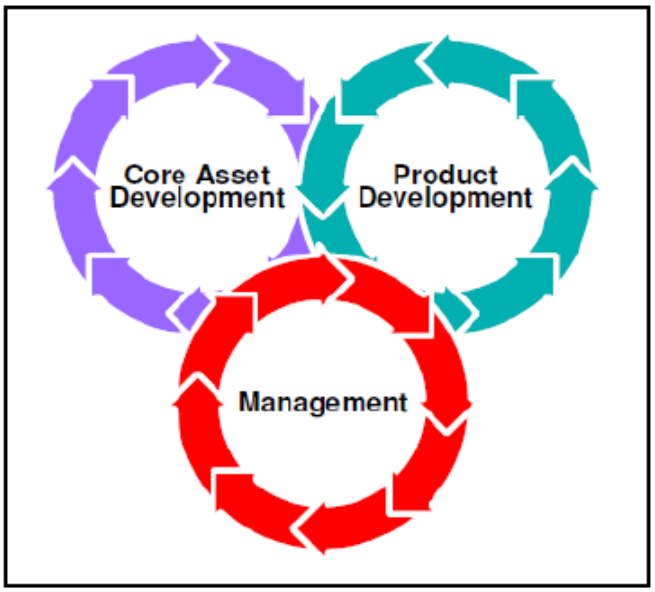
สายผลิตภัณฑ์ซอฟต์แวร์ คือ การนำแนวคิดการผลิตเชิงอุตสาหกรรมมาใช้ โดยนำหลักการที่สำคัญคือ การออกแบบและการจัดการการผลิต ซึ่งเริ่มต้นตั้งแต่การทำวิจัยและการทำการสำรวจและออกแบบผลิตภัณฑ์วางแผนการผลิต จนถึงเริ่มต้นผลิตสินค้า ดังรูปที่ 2.1

ประโยชน์ที่ได้รับจากการจัดการสายการผลิตที่ดีคือ ช่วยลดต้นทุนและเวลาในการผลิต สนับสนุนให้เกิดกระบวนการการผลิตสินค้าให้มีความหลากหลาย (variety of product) และปรับให้เข้ากับผู้บริโภคที่มีความหลากหลายได้เป็นอย่างดี (highly customizable) และช่วยเพิ่มคุณภาพให้กับตัวสินค้า

นอกจากนี้ยังมีประโยชน์อื่นอีกมากมาย โดยเฉพาะการสร้างสินทรัพย์ร่วม (core asset development) ซึ่งเป็นการนำ ส่วนประกอบทั่วไป (common component) มาแปลงให้เป็น

สินทรัพย์ (core asset) โดยอยู่บนพื้นฐานของการ นำกลับมาใช้ใหม่ (reuse) แต่การ นำกลับมาใช้ใหม่ (reuse) ต้องเป็นการ นำกลับมาใช้ใหม่ (reuse) ที่มีจัดการและการวางแผนอย่างดี

เมื่อจะพัฒนาระบบฯ ใหม่ ก็ต้องเก็บความต้องการ (requirements) วิเคราะห์ (Analysis) ออกแบบ (Design) และสร้างใหม่หมดตั้งแต่ต้น (เริ่มตั้งแต่ศูนย์ทุกครั้ง) โดยไม่สามารถ นำกลับมาใช้ใหม่(reuse) สิ่งที่มีอยู่แล้ว



รูปที่ 2.1 กิจกรรมที่จำเป็นในการพัฒนาสายผลิตภัณฑ์ซอฟต์แวร์ [1]

กิจกรรมที่จำเป็นในสายผลิตภัณฑ์ซอฟต์แวร์ ประกอบด้วย 3 กิจกรรมหลัก ได้แก่

1) การพัฒนาสินทรัพย์ (Core Asset Development)

เป็นการสร้างสินทรัพย์เพื่อนำมาใช้ในสายผลิตภัณฑ์ จากปัจจัยหลายอย่าง เช่น ข้อจำกัดของผลิตภัณฑ์และการผลิต จะถูกนำมาใช้ทำให้เกิดเป็นผลลัพธ์ ได้แก่ ขอบเขตของสายผลิตภัณฑ์ สินทรัพย์ทั่วไป

2) การพัฒนาผลิตภัณฑ์ (Product Development)

เป็นการใช้สินทรัพย์ที่มีอยู่มาประกอบกันเป็นผลิตภัณฑ์ กิจกรรมนี้จะนำผลลัพธ์ที่ได้จากการพัฒนาสินทรัพย์ทั่วไปมาใช้ตั้งต้นด้วยส่วนทั่วไป แล้วเพิ่มเติม ตามความต้องการเฉพาะของผลิตภัณฑ์นั้นๆ

3) การจัดการ (Management)

เป็นกิจกรรมที่ทำการจัดสรรทรัพยากรที่จำเป็น, ประสานงาน และควบคุมกิจกรรมอื่นๆในสายผลิตภัณฑ์ ซึ่งจะทำทั้งในเชิงเทคนิค และเชิงองค์กร

ในการพัฒนาสินทรัพย์ (Core Asset Development) นั้นจะแบ่งออกเป็น 2 วิธี ได้แก่

1) แนวทางเชิงรุก (proactive approach) เราจะเริ่มต้นโดยการพัฒนาสินทรัพย์หลัก (Core Asset base) ขึ้นมาก่อนแล้วจะนำไปผลิตหรือดัดแปลงเป็นส่วนประกอบร่วมในอนาคต

2) แนวทางเชิงปฏิกิริยา (Reactive approach) เราจะเริ่มต้นโดยการผลิตภัณฑ์พัฒนาออกมาก่อนแล้วค่อยสกัดในการพัฒนาสินทรัพย์หลักออกมาโดยการนำส่วนทั่วไปในผลิตภัณฑ์มาสกัดเป็นส่วนประกอบร่วม

2.1.2 การตรวจหาสำเนาโค้ด

2.1.2.1 นิยามและประเภทของสำเนาโค้ด

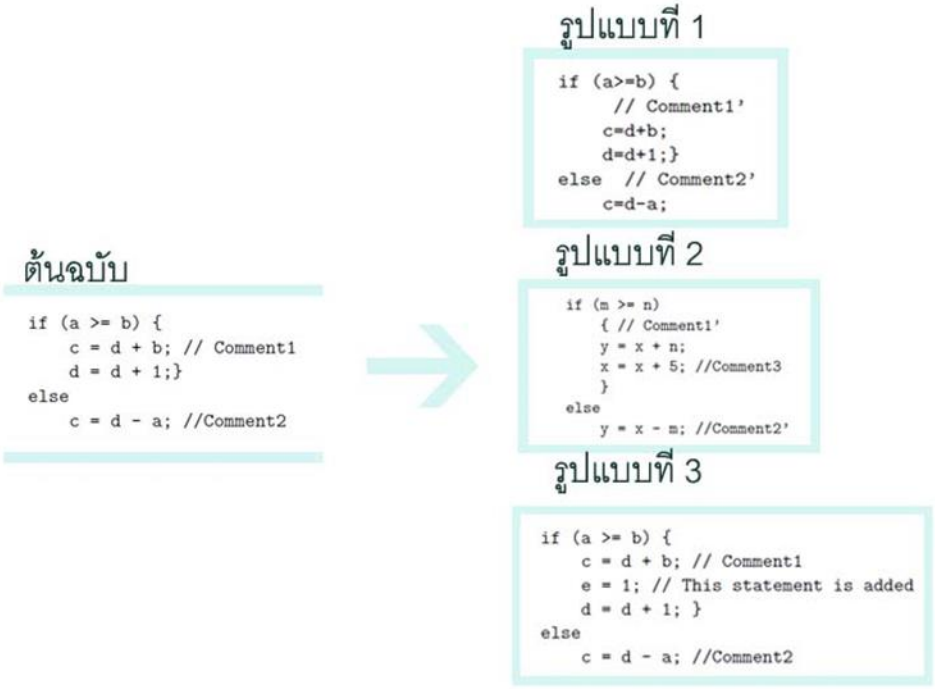
สำเนาโค้ด คือ ชิ้นส่วนซอร์สโค้ดที่ถูกสำเนาเพื่อการใช้ซ้ำและอาจทำการแก้ไขให้เข้ากับการใช้งานใหม่ พบได้ในการพัฒนาซอฟต์แวร์โดยทั่วไปซึ่งนิยมใช้การทำสำเนาและแปะ โดยอาจจะมีการแก้ไขหรือไม่ก็ได้ โดยได้มีการจำแนกประเภทของสำเนาโค้ด ออกเป็น 3 ประเภทในเชิงข้อความและอีกหนึ่งประเภทในลักษณะเชิงฟังก์ชัน [4],[5],[6] ดังรูปที่ 2.2 และ 2.3 ได้แก่

1) ประเภทที่ 1 เป็นเป็นลักษณะของชิ้นส่วนของโค้ดที่ถูกคัดลอกมาอย่างสมบูรณ์ ซึ่งไม่มีความแตกต่างใดๆเลยนอกจาก ช่องว่างในบรรทัด และ หมายเหตุที่ได้กำกับไว้

2) ประเภทที่ 2 ชิ้นส่วนโค้ดมีลักษณะเดียวกันในรูปแบบเชิงโครงสร้าง แต่อาจมีการสร้างความเปลี่ยนแปลงในแต่ละบรรทัด อันได้แก่ ชื่อ หรือชนิดของตัวแปร (type) รวมช่องว่างในบรรทัดและ หมายเหตุที่ได้กำกับไว้

3) ประเภทที่ 3 ชิ้นส่วนโค้ดมีการเปลี่ยนแปลง อันได้แก่การเพิ่มข้อความของคำสั่ง ลบข้อความของคำสั่ง หรือเปลี่ยนแปลงข้อความของคำสั่ง เพื่อทำให้เกิดความแตกต่างในแต่บรรทัดในชิ้นส่วนของโค้ด

4) ประเภทที่ 4 ชิ้นส่วนโค้ดที่ใช้หลักการหรือรูปแบบของเขียนที่มีความแตกต่างกัน แต่ให้ผลลัพธ์หรือพฤติกรรมของการทำงานที่เหมือนกัน เช่น if-else<>switch case , for<>recursive , while<>recursive เป็นต้น

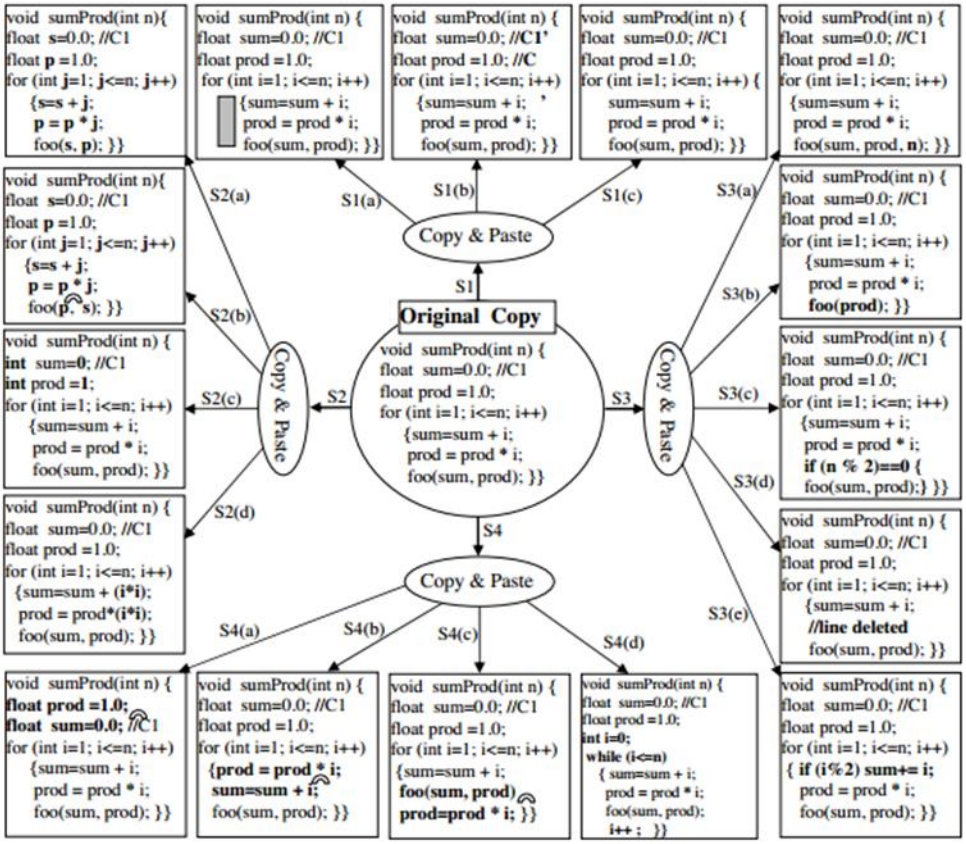


รูปที่ 2.2 ตัวอย่างคู่สำเนาโค้ดเชิงข้อความ ประเภทที่ 1, 2 และ และ 3 [4]


```
int i, j=1;
  for (i=1; i<=VALUE; i++)
       j=j*i;
```

```
int factorial(int n) {
 if (n == 0) return 1 ;
 else        return n * factorial(n-1) ;
}
```

รูปที่ 2.3 ตัวอย่างคู่สำเนาโค้ดเชิงฟังก์ชัน (ประเภทที่ 4) [4]



รูปที่ 2.4 ตัวอย่างคู่สำเนาโค้ดประเภทต่างๆ ที่มีการแก้ไขให้มีความแตกต่าง และสำเนาต้นแบบ [4]

ในการจัดการโค้ดประเภทที่ 4 ซึ่ง C.K. Roy และ J.R. Cordy ได้ทำการจำแนกโค้ดที่เกี่ยวข้องประเภทที่ 4 ออกเป็นประเภทต่าง [5] ดังต่อไปนี้ทั้งหมด 5 ประเภท

สำเนาเชิงโครงสร้าง (Structural clone)

จัดเป็นสำเนาที่จำแนกจากความคล้ายในเชิงโครงสร้าง โดยอาจจะมีขอบเขตของสำเนาในหลายระดับไวยกรณ์ ซึ่งจะครอบคลุมถึงสำเนาโค้ดทั้ง 4 ประเภท

สำเนาเชิงโครงสร้างในระดับการออกแบบ (Design level structural clone)

เป็นสำเนาโค้ดที่ประกอบด้วยสำเนาโค้ดที่ได้อธิบายไว้ก่อนหน้านี้ มารวมกันทำให้เกิดสำเนาที่คล้ายกันของโครงสร้างในระดับการออกแบบ

สำเนาสลับลำดับ (Reordered clone)

เป็นสำเนาโค้ดที่มีการสลับลำดับของข้อความคำสั่งหรือส่วนประกอบของโค้ด โดยที่ยังมีการขึ้นต่อกันของโครงสร้างและการควบคุมหรือทำให้ลำดับของการทำงานของข้อมูลไม่มีการเปลี่ยนแปลง ซึ่งก็คือสำเนาโค้ดประเภทที่ 4

```
Fragment 1:
++ fp1 = LA+i*tokensetsize;
++ fp2 =lookaheadset;
++ while (fp2 < fp3 )
++     *fp2++ |= *fp1++;
```

```
Fragment 2:
++ fp1 = base ;
++ fp2 = F + j * tokensetsize;
++ while (fp1 < fp3)++
++     ++ *fp1++ |=*fp2++;
```

รูปที่ 2.5 ตัวอย่างคู่สำเนาสลับลำดับ (Reordered clone) [4]

สำเนาฟังก์ชัน (Function clone)

จัดเป็นสำเนาเชิงโครงสร้างที่สนใจการทำงานอยู่ในระดับฟังก์ชัน ที่มีลักษณะของการทำงานที่เหมือนกันของฟังชันก์ ซึ่งก็คือสำเนาโค้ดประเภทที่ 4

สำเนาสานต่อกัน (Intertwined clone)

เป็นสำเนาโค้ดเกิดจากการรวมชิ้นส่วนของโค้ด 2 ชิ้นนำมาสานต่อให้ลักษณะที่มีความเหลื่อมกันต่อกัน จัดได้ว่าเป็นสำเนาโค้ดประเภทที่ 4

```
Fragment 1:
 tmpa = UCHAR(*a);
 while (blanks[tmpa])
   tmpa = UCHAR(*++a);
   if (tmpa == '-'
 ...
```

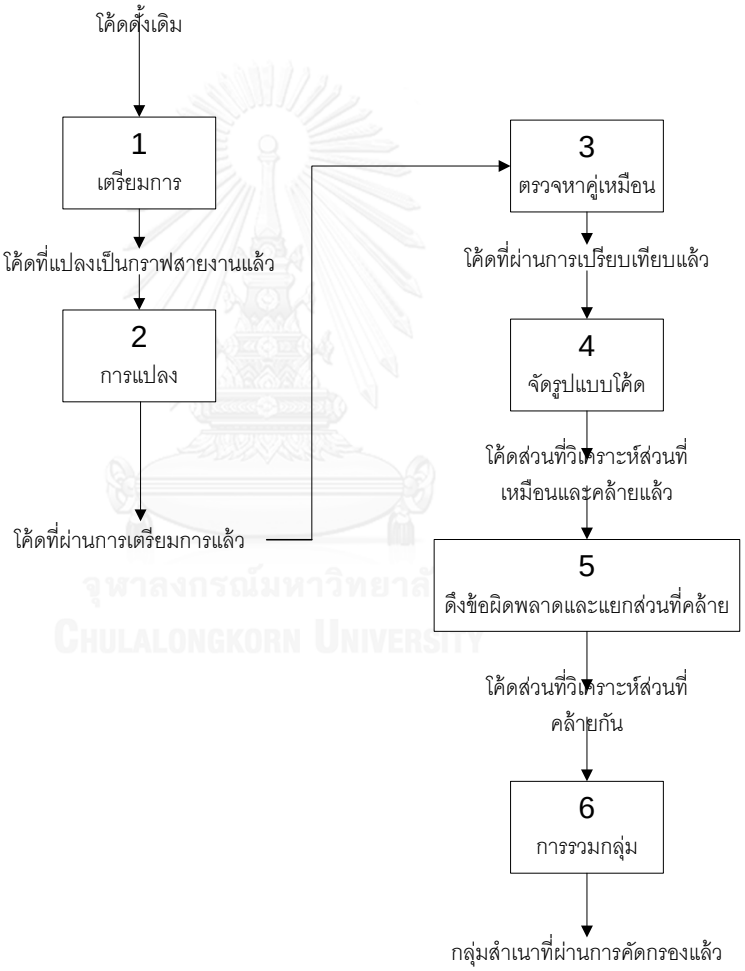
```
Fragment 2:
tmpb = UCHAR(*b);
while (blanks[tmpb])
tmpb = UCHAR(*++b);

else if (tmpb == '-' ..
```

```
-----------------------------------------------------------------
-----------------------------------------------------------------
 ++ tmpa = UCHAR(*a);
xx tmpb = UCHAR(*b);
++ while (blanks[tmpa])
++   tmpa = UCHAR(*++a);
xx while (blanks[tmpb])
xx   tmpb = UCHAR(*++b);
++ if (tmpa == '-'
 ...
xx else if (tmpb == '-' ..
```

รูปที่ 2.6 ตัวอย่างคู่สำเนาสานต่อกัน (Intertwined clone) [4]

2.1.2.2 กระบวนการในการตรวจหาสำเนาโค้ด

ในการตรวจหาสำเนาโค้ด มักจะทำการแปลงโค้ดจากรูปแบบข้อความ ที่อยู่ในลักษณะของโค้ดแต่ละประเภท ให้อยู่ในรูปแบบเชิงโครงสร้างก่อน แล้วทำการเปรียบเทียบส่วนประกอบของโค้ด โค้ดแล้วดำเนินการสรุปผลการตรวจหาสำเนาโค้ด และอาจจะมีขบวนการในการเพิ่มประสิทธิภาพของผลลัพ์ให้มีค่าสูงยิ่งขึ้น ดังที่ได้จำแนกไว้โดยสังเขปใน [10] ดังแสดงในรูปที่ 7



รูปที่ 2.7 ภาพรวมของการตรวจหาสำเนาโค้ด [10]

1) การเตรียมการ

เริ่มต้นด้วยขั้นตอนที่ทำการแบ่งส่วนต่างๆของโค้ด และพิจารณาระดับของการเปรียบเทียบ โดยจะลบซอร์สโค้ดส่วนที่ไม่สนใจออก เช่น ส่วนของภาษาอื่นๆที่ได้มีการฝังตัวอยู่ในโค้ด หรือโค้ดที่อาจมีสาเหตุที่จะทำให้การตรวจหาสำเนาโค้ดเกิดความผิดพลาด

2) การแปลงโค้ดเป็นกราฟการไหลของการควบคุม

เป็นการปรับโค้ดทั้งสองที่ใช้ในการเปรียบเทียบนำมาทำเป็นกราฟการไหลของการควบคุม เพื่อใช้ในการตรวจสอบโครงสร้างของข้อมูลซอร์สโค้ดจะใช้แนวคิดของการทดสอบเส้นทาง ซึ่งในการแปลงโค้ดกราฟการไหลของการควบคุมจะมีลักษณะและวิธีการที่แตกต่างกันออกไป [11],[12],[8] ในงานวิจัยนี้ได้รวบรวมรูปแบบของการแปลงโค้ดเป็นกราฟการไหลของการควบคุมที่เหมาะสมต่อการเปรียบเทียบลักษณะโครงสร้าง

3) ตรวจหาคู่เหมือน

การตรวจหาคู่เหมือนเป็นขั้นตอนที่จะทำการเปรียบเทียบซอร์สโค้ดที่ผ่านขั้นตอนการตรวจสอบเชิงโครงสร้างมาแล้วระหว่างทุกหน่วยย่อยเพื่อหาสำเนาโค้ดที่มีอยู่ โดยข้อมูลที่ใช้อาจจะเป็นซอร์สโค้ดโดยตรง หรือข้อมูลรูปแบบอื่นที่ได้จากขั้นตอนการแปลง เป็นกราฟการไหลของการควบคุม (Control flow graph Matching) ข้างต้น

ในการตรวจหาคู่เหมือนของการตรวจหาสำเนาโค้ดประเภทที่ 4 จะใช้วิธีการของการจำแนกกราฟการไหลของการควบคุมทั้งสองออกเป็นเส้นทางทั้งหมดที่เป็นไปได้แล้วทำการจับคู่เส้นทางที่มีการทำงานที่เหมือนกัน ดังนั้นแล้ว สิ่งที่ได้จากขั้นตอนของการหาคู่เหมือนคือคู่สำเนาโค้ดของเส้นทางที่มีการทำงานที่เหมือนกัน [13]

4) จัดรูปแบบโค้ดเป็นส่วนๆ

คู่สำเนาโค้ดที่ได้จากการตรวจหาคู่เหมือนอาจจะอยู่ในรูปที่ถูกแปลงไปแล้ว ในในรูปแบบข้างต้นไม้ ทำโดยในขั้นตอนนี้จะสามารถมองโค้ดออกเป็นกลุ่มๆซึ่งจะแบ่งออกเป็นกลุ่มทีเหมือนกับกลุ่มที่มีความคล้ายกันออกจากกัน

5) ดึงข้อผิดพลาดออกจากโค้ด

ขั้นตอนในการจัดลำดับหรือคัดกรองสำเนาที่ตรวจหามาได้ในส่วนที่มีความคล้ายกัน เพื่อดึงส่วนที่ตรวจผิดพลาดออก และปรับส่วนที่คล้ายให้มีความเหมือนกัน

6) การรวมกลุ่ม (Aggregation)

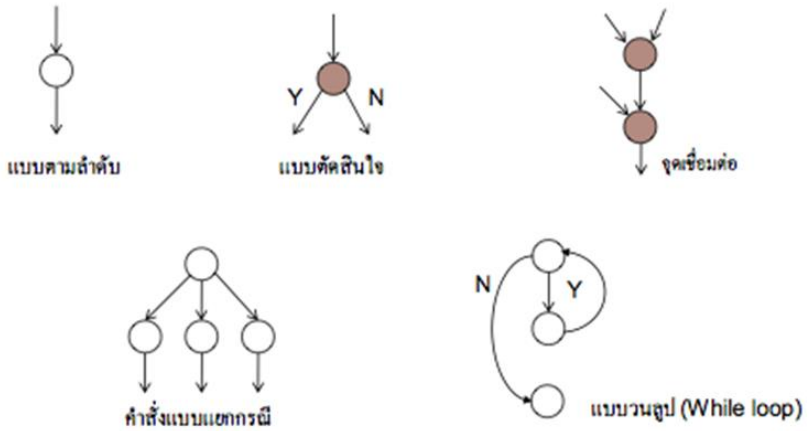
การรวมกลุ่มของข้อมูลการตรวจสอบสำเนา โดยอาจจะรายงานเป็นคู่สำเนา หรือคลาสสำเนา (Clone class) ก็ได้ เพื่อสรุปเป็นรายงานแสดงถึงภาพรวม

2.1.3 กราฟการไหลของการควบคุม [14]

กราฟการไหลของการควบคุม (Control Flow graphs) เป็นการแสดงโครงสร้างการควบคุมของโปรแกรมในลักษณะเชิงโครง สร้างของโปรแกรม ซึ่งคล้ายกับแผนภาพ โฟร์ชาร์ท แต่เราสามารถไม่ต้องใส่รายละเอียดทุกคำสั่งของขบวนการ

เส้นทาง (Path) เป็นไปตามลำดับของคำสั่งโปรแกรมที่เริ่มจากโหนดใดโหนด หนึ่ง เช่น อาจจะเริ่มด้วยโหนดลำดับ, โหนดแบบจุดเชื่อมต่อ หรือ โหนดตัดสินใจ และจบด้วยโหนดจบ เส้นทาง (Path) หนึ่งสามารถผ่านโหนดลำดับ โหนดจุดเชื่อมต่อ หรือโหนด ตัดสินใจหนึ่งครั้งหรือมากกว่านั้นก็ได้

เส้นเชื่อมโยง เป็นเส้นเชื่อมระหว่าง 2 โหนด คือ ระหว่างโหนด-โหนด หรือ โหนด ตัดสินใจ – โหนด ดังนั้นความยาวของเส้นทาง คือ จำนวนของเส้นเชื่อมโยงที่อยู่ในเส้นทางนั้นๆ



รูปที่ 2.8 ส่วนประกอบของกราฟการไหลของการควบคุม (Flow graph Elements)

กราฟการไหลของการควบคุมจะมีการแบ่งประเภทของโหนดออกจากกัน ซึ่งแต่ละประเภทนั้นจะมีความแตกต่างกันไป ขึ้นอยู่กับลักษณะของคำสั่งการทำงานของโหนด ซึ่งแต่ละประเภทจะแจกแจงรายละเอียดดังต่อไปนี้

1) โหนดแบบตามลำดับ

โหนดเป็นไปตามลำดับของคำสั่งโปรแกรม อย่างเช่น ถ้าไม่มีโหนดใดโหนดหนึ่งถูกขัดจังหวะให้ประมวลผล ดังนั้นคำสั่งทั้งหมดก็จะถูกประมวลผลทั้งหมด การถูกขัดจังหวะโดยมีโหนดแบบตัดสินใจ หรือโหนดเชื่อมต่อ ดังนั้นลักษณะแบบลำดับ คือ มี 1 ทางเข้า และ 1 ทางออก โปรแกรมไม่กระโดดเข้าหรือออกจากโหนดทันที

2) โหนดแบบตัดสินใจ

เป็นจุดทางเลือกของโปรแกรม ซึ่งสามารถควบคุมทางแยกได้ ส่วนใหญ่แล้วจะแยกเป็น 2 ทาง เรียกว่าแบบ binary

3) โหนดแบบแยกกรณี

มีหลายทางเลือกอาจมีได้มากกว่า 2 ทาง สำหรับการออกแบบการทดสอบ ไม่แตกต่างกันระหว่างโหนดตัดสินใจและ โหนดแบบทางเลือก

4) โหนดจุดเชื่อมต่อ

เป็นโหนดของโปรแกรมซึ่งควบคุมกราฟและสามารถเป็นจุดรวมกันได้

การสร้างกราฟโปรแกรม (Constructing a Program Graph) จะมีขบวนการในการสร้างจากลำดับของขั้นตอนดังต่อไปนี้

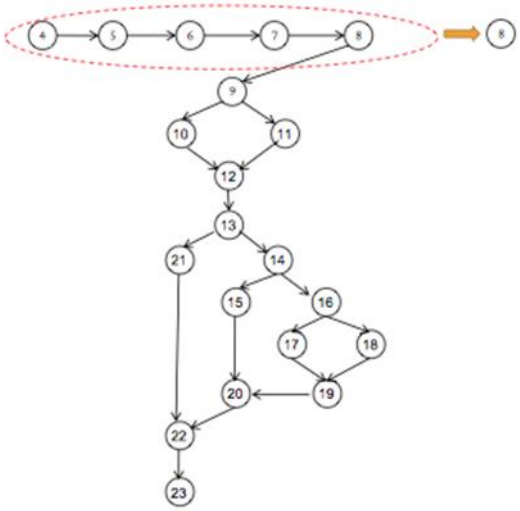
• แต่ละคำสั่งโปรแกรม (Statement) หรือการแตกคำสั่งโปรแกรม (Statement) ถูกกำหนดเป็นเลขบรรทัด (Line number)

• วาดกราฟการไหลของการควบคุม (Control flow graph) ให้สัมพันธ์กับโปรแกรม

ตัวอย่างการสร้างกราฟโปรแกรมวิเคราะห์ชนิดของสามเหลี่ยม

```
1.  Program triangle2 'Structured
    programming version of simpler
    specification'
2.  Dim a, b, c As Integer
3.  Dim IsATriangle As Boolean
        'Step 1: Get Input'
4.  Output ("Enter 3 integers which are sides
    of a triagle")
5.  Input (a, b, c)
6.  Output ("Side A is", a)
7.  Output ("Side B is", b)
8.  Output ("Side C is", c)
        'Step 2: Is A Triangle?'
9.  If (a < b+c) AND (b < a+c) AND (c < a+b)
10. Then IsATriangle = True
11. Else IsATriangle = False
12. EndIf
        'Step3: Determine Triangle Type'
13. If IsATriangle
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 2.9 โค้ดโปรแกรมตัวอย่าง



รูปที่ 2.10 กราฟการไหลของการควบคุม (Control Flow Graphs) ของโปรแกรมตัวอย่าง

การทดสอบแบบ Basis Path [15] จะต้องรู้จำนวนเส้นทางความซับซ้อนของปัญหา (cyclomatic complexity หรือแทนด้วยกราฟ V(G)) โดย V(G) สามารถหาได้จาก 3 วิธี ดังนี้

วิธีที่ 1. V(G) = จำนวนของพื้นที่แบบปิด (enclosed area) ของผังงานโปรแกรม +1

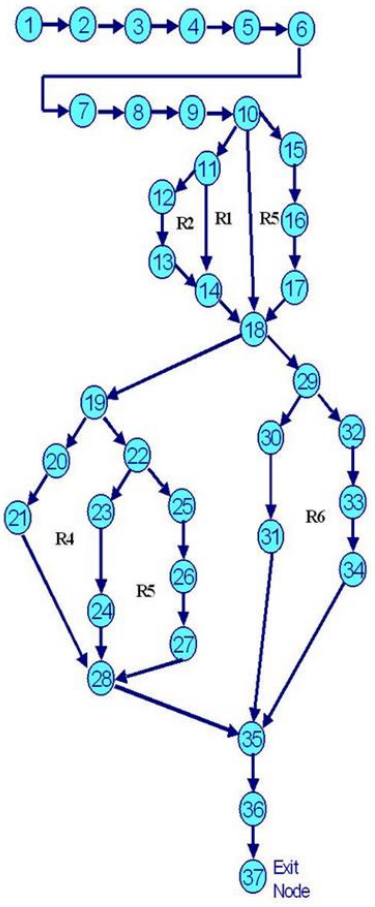
วิธีที่ 2 V(G) = จำนวนของสัญลักษณ์ตัดสินใจในผังงานโปรแกรม (simple decisions) +1

วิธีที่ 3 V(G) = Edge (จำนวนเส้นเชื่อมระหว่าง Node) – Node (จำนวน Node ภายในกราฟ) +2

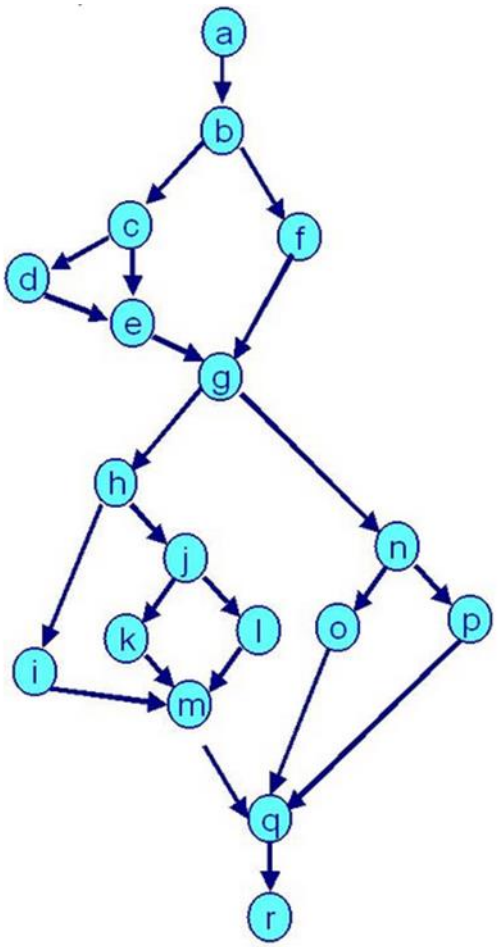
2.1.4 Decision-to-Decision path [14]

DD-Path ย่อมาจาก Decision-to-Decision path เริ่มจากการออกจากคำสั่งเริ่ม (Start) แล้วต่อด้วยคำสั่งแบบตัดสินใจ (Decision) และจบด้วยทางเข้าของคำสั่งตัดสินใจต่อไป เส้นทางของโหนดต้องเป็นกราฟแบบมีทิศทาง (Directed graph) และมีการนับการเข้า (indegree) และ ออกของข้อมูล (out degree)

ในงานวิจัยนี้ได้นำเอาทฤษฎีของ DD-Path มาใช้ในการปรับรูปร่างของกราฟการไหลของการควบคุมเพื่อง่ายต่อการมองลักษณะโครงสร้างเพื่อนำมาเปรียบเทียบความคล้ายกันของแผนภาพโดยลักษณะของ DD-Path จะมีการยุบรวมกันของกราฟที่มีลักษณะของเส้นตรงให้อยู่ในโหนดเดียวกัน ทำให้เราสามารถมองโครงสร้างหลักที่เป็นการจำแนกการทำงานของพฤติกรรมได้อย่างชัดเจนดังตัวอย่างข้างต้น



รูปที่ 2.11 CFG ก่อนที่เป็น DD-Path



รูปที่ 2.12 CFGเป็น DD-Path

ลักษณะของ DD-Path มี 5 กรณี คือ

- Case 1: การประกอบของโหนดเดี่ยว (Single node) ที่มี indeg =0

- Case 2: การประกอบของโหนดเดี่ยว ที่มี outdeg = 0

- Case 3: การประกอบของโหนดเดี่ยว ที่มีindeg >= 2 หรือ outdeg >=2

- Case 4: การประกอบของโหนดเดี่ยว ที่มี indeg = 1 และ outdeg =1

- Case 5: เป็นการเชื่อมโยงของโหนดที่มีความยาว >=1

ซึ่งค่าเหล่านี้เราสามารถนำค่าของทั้งสองกราฟมาเปรียบเทียบกันเพื่อหาความคล้ายกันของโครงสร้างทั่วไปในกราฟการไหลของการควบคุมได้

ซึ่งเราได้นำตัวอย่างของการหา DD-Path ของโปรแกรมหาประเภทสามเหลี่ยม ดังรูปที่ 2.13 แสดงซอร์สโค้ดของโปรแกรมประเภทสามเหลี่ยม ต่อมารูปที่ 2.14 จะแสดงค่า DD-Path ของ

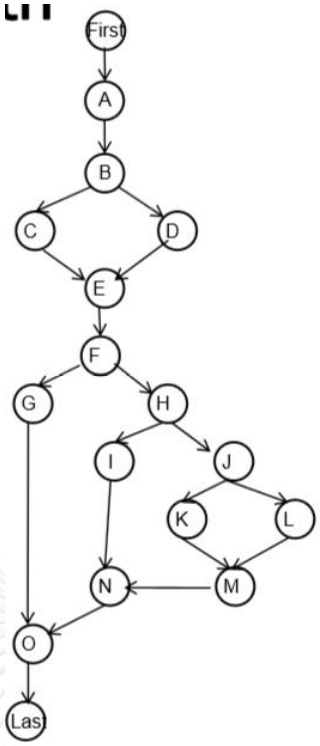
โปรแกรมประเภทสามเหลี่ยม และรูปที่ 2.15 จะแสดงกราฟ DD-Path ของโปรแกรมหาประเภทสามเหลี่ยม

```
1.  Program triangle2 'Structured programming version of simpler specification'
2.  Dim a, b, c As Integer
3.  Dim IsATriangle As Boolean
        'Step 1: Get Input'
4.  Output ("Enter 3 integers which are sides of a triagle")
5.  Input (a, b, c)
6.  Output ("Side A is", a)
7.  Output ("Side B is", b)
8.  Output ("Side C is", c)
        'Step 2: Is A Triangle?'
9.  If (a < b+c) AND (b < a+c) AND (c < a+b)
10. Then IsATriangle = True
11. Else IsATriangle = False
12. EndIf
        'Step3: Determine Triangle Type'
13. If IsATriangle
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 2.13 ซอร์สโค้ดของโปรแกรมสร้างสามเหลี่ยม

Program Graph Node	DD-Path Name	Case of DD-Path
4	First	1
5-8	A	5
9	B	3
10	C	4
11	D	4
12	E	3
13	F	3
14	H	3
15	I	4
16	J	3
17	K	4
18	L	4
19	M	3
20	N	3
21	G	4
22	O	3
23	Last	2

รูปที่ 2.14 ค่า DD-Path ของโปรแกรมประเภทสามเหลี่ยม



รูปที่ 2.15 กราฟ DD-Path ของโปรแกรมหาประเภทสามเหลี่ยม

2.1.5 การทดสอบเส้นทาง [14]

การทดสอบเส้นทาง (Path Testing) เป็นวิธีการหนึ่งสำหรับการทำการทดสอบขนาดย่อย (Unit Testing) ซึ่งอยู่ในรูปแบบประเภทการ ทดสอบ แบบกล่องขาว (White Box) เพื่อต้องการตรวจสอบการทำงานของ โปรแกรมว่าครอบคลุมทุกคำสั่งการ ทำงานหรือไม่ (Coverage Testing) ดังนั้นแล้ว การทดสอบเส้นทาง (Path Testing) จึงสามารถตรวจสอบการเขียนโปรแกรมของโปรแกรมเมอร์ หรือ นักพัฒนา (Developer) ตรงกับความต้องการของผู้ใช้ที่ได้กำหนดสเปคของโปรแกรม (Program Specification)

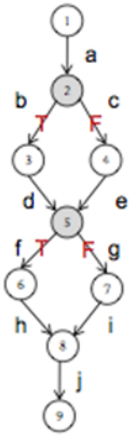
การทำการทดสอบเส้นทาง (Path Testing) นั้นจำเป็นจะต้องเป็น โปรแกรมที่เป็นกราฟแบบมีทิศทาง (directed graph) ซึ่งมีโนด (Node) ที่สามารถแตกคำสั่งจากโปรแกรม (Statement) และมีเส้น (Edges) แสดงการ ควบคุมสาย งาน (Flow of control)

กลยุทธ์ของการทดสอบแบบมีเส้นทาง (Path Testing Criteria or strategies) แบ่งออกเป็นทั้งหมด 3 ประเภทดังต่อไปนี้

1) Statement Testing การประมวลผลคำสั่งโปรแกรม (statement) ในโปรแกรมอย่างน้อย 1 คำสั่ง (100% statement / node converage)

2) Branch Testing การประมวลผลให้ทดสอบเพียงพอเพื่อให้มั่นใจว่าเส้นเชื่อมหรือกิ่งสาขา (branch) ที่มีทางเลือกจะมีการ exercised อย่างน้อย 1 กิ่งสาขา (100% branch/ link coverage)

3) Path Testing (C3) การประมวลผลทุกสายงานควบคุมที่เป็นไปได้ทั้งหมดตลอดทั้งโปรแกรม อย่างน้อย 1 เส้นทาง (100% path coverage)



Paths	Decisions		Process-Link									
	2	5	a	b	c	d	e	f	g	h	i	j
1-2-3-5-6-8-9 (a-b-d-f-h-j)	T	T	✓	✓		✓		✓		✓		✓
1-2-4-5-7-8-9 (a-c-e-g-i-j)	F	F	✓		✓		✓		✓		✓	✓
1-2-3-5-7-8-9 (a-b-d-g-i-j)	T	F	✓	✓		✓			✓		✓	✓
1-2-4-5-6-8-9 (a-c-e-f-h-j)	F	T	✓		✓		✓	✓		✓		✓

รูปที่ 2.16 ตัวอย่างการหาเส้นทางที่เป็นไปได้ทั้งหมดจากกราฟการไหลของการควบคุม

ดังนั้นการหาเส้นทางที่เกิดขึ้นทั้งหมดเราสามารถเรียกดูพฤติกรรมต่างๆของโปรแกรมได้ว่าสิ่งที่เกิดขึ้นนั้นจะรวมไปถึงสิ่งที่เป็นไปได้และสิ่งที่เป็นไปไม่ได้ทำให้เราทราบถึงขบวนการของการทำงานเชิงพฤติกรรมได้ทั้งหมดซึ่งจะสามารถนำมาวิเคราะห์โครงสร้างการทำงานภายในเชิงลักษณะของพฤติกรรมของการทำงานได้

2.1.6 เมท็อดแม่แบบ (Template method)

เมท็อดแม่แบบเป็นลักษณะของรูปแบบการออกแบบอย่างหนึ่งในการออกแบบเชิงวัตถุ โดยมีจุดประสงค์ในการทำงานเพื่อลดความซ้ำซ้อนเมื่อเมท็อดนั้นๆมีทั้งโค้ดส่วนที่เหมือนกันและแตกต่างกัน จึงใช้เมท็อดแม่แบบเพื่อแยกส่วนทั้งสองที่แตกต่างกันออกไปให้อยู่ในคลาสลูก และทำให้สามารถนำโค้ดอื่นๆ ที่เหมือนกันไปรวมไว้ยังคลาสแม่

โครงสร้างเมท็อดแม่แบบคือ รวมกลุ่มเป็นโค้ดส่วนที่เหมือนกัน เพื่อย้ายโค้ดเหล่านั้นไปไว้ในคลาสแม่ หรือที่เรียกว่าเมท็อดฮุคซึ่งจะเป็นเพียงเมท็อดนามธรรมในคลาสแม่และโค้ดที่แตกต่างจะถูกย้ายไปไว้ในคลาสลูก เพื่อโอเวอร์ไรด์เมท็อดฮุคที่ถูกประกาศไว้ในคลาสแม่นั่นเอง

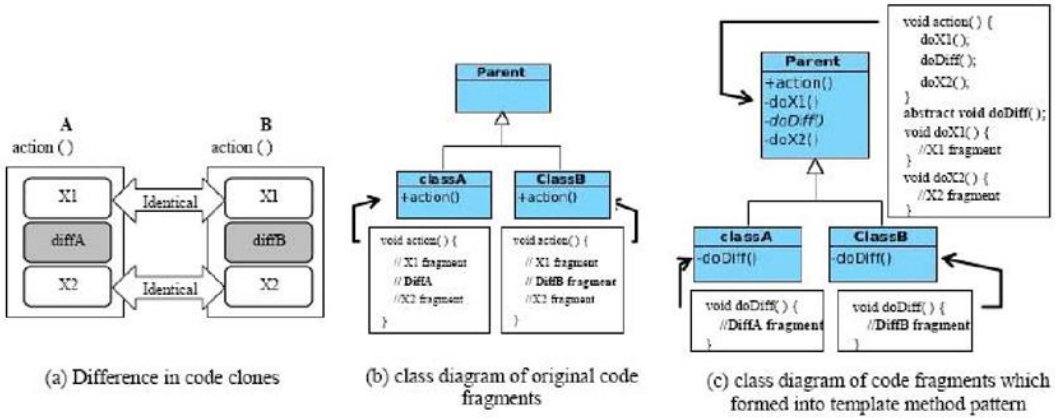
2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยนี้ได้ศึกษางานวิจัยที่เกี่ยวข้องด้วยกันทั้งหมด 5 งานวิจัย จุดประสงค์หลักของการศึกษางานวิจัยที่เกี่ยวข้อง เพื่อศึกษาแนวทางและวิธีคิดต่างๆ ของงานวิจัยก่อนหน้า เพื่อนำมาปรับใช้ในการทำงานวิจัยนี้ให้มีประสิทธิภาพยิ่งขึ้น โดยจะแสดงรายละเอียดงานวิจัยที่เกี่ยวข้องได้ดังต่อไปนี้

2.2.1 IDENTIFYING COMMON ASSET CANDIDATES IN SOFTWARE PRODUCT LINE BY CODE CLONE DETECTION [7]

บทความนี้ได้มีการนำเสนอแนวคิดที่ถูกทำขึ้นเพื่อลดค่าใช้จ่ายและแรงงานในการพัฒนาซอฟต์แวร์ซึ่งได้มีการพัฒนาหลักสินทรัพย์ เพื่อนำไปใช้ในการประกอบและปรับแต่งเป็นส่วนหนึ่งของผลิตภัณฑ์ที่จะทำการสร้างขึ้นมา โดยผู้เชี่ยว ชาญโดเมน อาจวิเคราะห์และสร้างสินทรัพย์ทั่วไปขึ้น มาใหม่ตั้งแต่เริ่มแรก หรือระบุสินทรัพย์ทั่วไปจาก ผลิตภัณฑ์ที่มี อยู่แล้วก็ได้ ซึ่งในบทความนี้ได้มีการประยุกต์ใช้การตรวจหาสำเนาโค้ด ระหว่างผลิตภัณฑ์ เพื่อระบุตัวแทนสินทรัพย์ ทั่วไปจากผลิตภัณฑ์ที่มีอยู่ โดยใช้การตรวจหาสำเนาโค้ดประเภทที่ 3 คือ สำเนาโค้ดที่ตรวจพบมีความแตกต่างอยู่บ้างในแต่ละบรรทัด และ ความแตกต่างนั้น จำเป็นต้องถูกแยกออกมา เพื่อให้ สามารถนำสำเนาโค้ดนั้น ไปพัฒนาเป็นสินทรัพย์หลักทั่วไปได้

งานวิจัยนี้มี จุดประสงค์เพื่อแยกส่วนที่ความแตกต่างดังกล่าวออกจากคู่สำเนาโค้ดที่ตรวจพบ โดยการจำแนกสำเนาโค้ดที่มีความ แตกต่างในระดับ เมท็อด แล้วได้ทำการประยุกต์ใช้ เมท็อดแม่แบบ เพื่อทำการพัฒนาหรือแปลงให้ได้ตัวแทนสินทรัพย์หลักทั่วไป ในรูปแบบคลาสแม่ ซึ่งจะมีการโอเวอร์ไรด์แตกต่างจากคลาสลูกสำหรับแต่ละผลิตภัณฑ์ด้วยโค้ดในส่วนที่แตกต่างกันของทั้งสองผลิตภัณฑ์ ทำให้คลาสแม่ที่ได้มานั้น จะสามารถนำไปสกัดเป็นหลักสินทรัพย์ ที่จะนำมาปรับแต่งเป็นผลิตภัณฑ์ที่จะเกิดขึ้นในอนาคต ทำให้สามารถลดเวลาของการผลิตสินค้า และลดต้นทุนของการวิเคราะห์สินค้าในด้านของการพัฒนาและการผลิตได้อย่างดียิ่งขึ้น



(a) Difference in code clones

(b) class diagram of original code fragments

(c) class diagram of code fragments which formed into template method pattern

รูปที่ 2.17 การค้นหาตัวแทนสินทรัพย์ ทั่วไปจากผลิตภัณฑ์ที่มีอยู่ [2]

อย่างไรก็ตามบทความ[7]เสนอเพียงการค้นหาสำเนาโค้ดประเภทที่ 3 เท่านั้นว่าจะทำอย่างไรกับโค้ดที่ทำการเปลี่ยนแปลง และเพิ่มลบคำสั่ง แต่ไม่ได้รวมถึงสำเนาโค้ดประเภทที่ 4 ที่ชิ้นส่วนโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน

2.2.2 Program Element Matching for Multi-Version Program Analyses [4]

บทความงานวิจัยของ Miryung Kim, David Notkin [4] ได้ รวบรวมและสำรวจเกี่ยวกับความสามารถและ คุณสมบัติของ Matching Technique ซึ่งเป็นเทคนิคที่ใช้ในการวิเคราะห์ เวอร์ชันของซอฟต์แวร์โดยเทคนิคที่ได้ ทำการสำรวจมีดังนี้ Entity Name Matching, String Matching, Syntax Tree Matching, Control Flow Graph Matching, Program Dependence Graph Matching, Binary Code Matching, Clone Detection, Origin Analysis Tools โดยแต่ละเทคนิค นั้นมีข้อดีข้อเสียและผลกระทบที่แตกต่างกัน และในงานวิจัยนี้ได้มีการทดสอบโดยการนำข้อมูลที่มีความแตกต่างกันมาทำการทดลองและสรุปผลได้ว่าในแต่ละวิธีนั้นจะทีข้อดีและข้อเสียแตกต่างกันไปขึ้นอยู่กับ องค์ประกอบต่างๆของโปรแกรมที่มีอยู่

Table 1: Comparison of Matching Techniques

Program Representation	Citation	Granularity	Assumed Correspondence	Multiplicity	Heuristics			Application
					N	P	S	
A set of entities	[20, 15, 37]	Module		1:1	√			Fault proneness
	Bevan et al. [11]	File		1:1	√			Instability
	Ying et al. [48]	File		1:1	√			Co-change
	Zimmermann et al. [51]	File Procedure Field		1:1	√			
String	*diff* [25]	Line	File	1:1			√	Merging Clone genealogy [29] Fix inducing code [43]
	bdiff [45]	Line	File	1:n			√	Merging
AST	*cdiff* [47]	AST Node	Procedure	1:1	√			
	Neamtiu et al.[38]	Type, Var		1:1	√	√		Type change
	Hunt, Tichy[24, 35]	Token	File	1:1		√	√	Merging
CFG	*Jdiff* [5]	CFG node		1:1	√		√	Regression testing Impact analysis
Binary	BMAT [46]	Code block		1:1 (procedure) n:1 (block)	√	√	√	Profile propagation Regression testing
Hybrid	Zou, Godfrey [52]	Procedure		1:1 or 1:n or n:1	√		√	Origin analysis
	Kim et al. [30]	Procedure		1:1	√		√	Signature change [31] Renaming analysis

N: Name-based heuristics, P: Position-based heuristics, S: Similarity-based heuristics

รูปที่ 2.18 แสดงการทดลองที่ได้นำวิธีการต่างๆ ไปใช้การแต่ละรูปแบบเพื่อหาความเหมาะสม [4]

Program Representation	Citation	Scenario		Transformations				Strength and Weakness
				Split/Merge		Rename		
		1	2	Proc	File	Proc	File	
String	*diff* [25]	☒	□	□	□	☒	□	− sensitive to file name changes
	bdiff [45]	■	□	☒	□	☒	□	+ can trace copied blocks
AST	*cdiff* [47]	□	□	□	□	□	□	− sensitive to nested level change − require procedure level mappings
	Neamtiu et al. [38]	□	□	□	□	□	□	− partial AST matching
	Hunt, Tichy [24, 35]	☒	□	□	□	■	□	− require file level mappings + can identify procedure renaming
CFG	*JDiff* [5]	■	☒	□	□	☒	☒	+ robust to signature changes − sensitive to control structure changes
Binary	BMAT [46]	□	■	□	□	■	■	+ robust to procedure renaming − assume 1:1 procedure correspondence − only applicable to binary programs
Hybrid	Zou, Godfrey [52]	□	■	■	■	■	■	− semi-automatic, manual analysis
	Kim et al. [30]	□	■	□	□	■	■	− assume 1:1 procedure correspondence

■ good ☒ mediocre □ poor

รูปที่ 2.19 แสดงผลการทดลองที่ได้นำวิธีการต่างๆ ไปใช้การแต่ละรูปแบบเพื่อหาความเหมาะสม [4]

จากตารางแสดงการทดลองและผลการทดลองที่ได้นำวิธีการต่างๆ ไปใช้การแต่ละรูปแบบ พบว่าแต่ละวิธีการจะมีข้อดีข้อเสียที่แตกต่างกันออกไปดังที่กล่าวไว้ในตาราง แต่สิ่งที่น่าสนใจกลับพบว่า CFG มีความเหมาะสมในการเปรียบเทียบเชิงโครงสร้างมากที่สุด เนื่องจากสามารถรองรับการเปลี่ยนแปลงของซอร์สโค้ดได้โดยการตรวจสอบโครงสร้างที่คงอยู่เดิม

2.2.3 NICAD: Accurate Detection of Near-miss Intentional Clones Using Flexible Pretty-printing and Code Normalization [10]

NiCad เป็นเครื่องมือตรวจหาสำเนาโค้ดประเภทหนึ่ง ที่มีการใช้เทคนิคการตรวจหาสำเนาโค้ด โดยใช้เทคนิคในลักษณะของการเปรียบเทียบเชิงข้อความ (String Matching) โดยใช้ภาษาที

เอ็กซ์แอล (TXL) ในการแปลงโค้ดเพื่อทำการเปรียบเทียบชิ้นส่วนชุดคำสั่งของสำเนาโค้ด สามารถแบ่งเป็นสามขั้นตอนหลักสำคัญต่างๆ ได้แก่

1) การแจงส่วน

ในขั้นตอนนี้โค้ดจะถูกแบ่งออกเพื่อแยกโค้ดออกเป็นชิ้นส่วนเพื่อใช้ในขบวนการของการเปรียบเทียบ เช่น คลาส หรือฟังก์ชัน รวมถึงการจัดชิ้นให้อยู่ในรูปแบบของมาตรฐานเดียวกัน โดยช่องว่างที่เกิดขึ้นและหมายเหตุที่มีการเพิ่มเติมมาจะถูกตัดออกไป โดยในขั้นตอนนี้จะมีการตรวจนับบรรทัดเพื่อกำหนดจำนวนบรรทัดของโค้ดที่น้อยที่สุดและที่มากที่สุด เพื่อสะดวกและลดการเปรียบเทียบสิ่งที่มีความไม่จำเป็นลง แล้วทำการเก็บอยู่ในรูปเอ็กเอ็มแอล

2) การทำให้เป็นบรรทัดฐาน

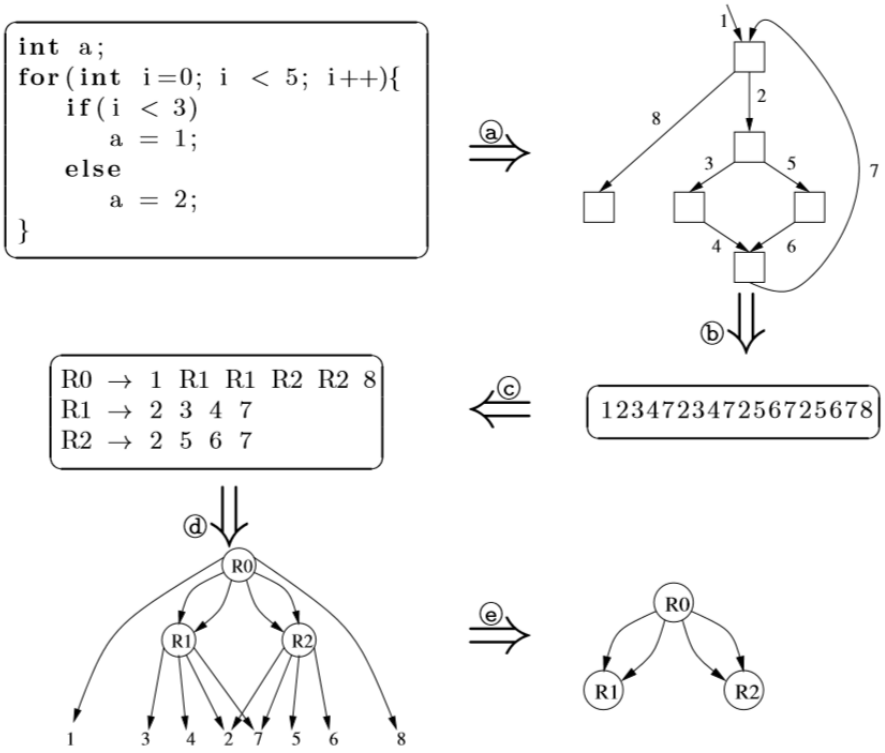
คือขั้นตอนทางเลือกที่เพิ่มเติมขึ้นมาเพื่อช่วยให้ผลการตรวจหาสำเนาโค้ดดียิ่งขึ้นด้วยเงื่อนไขเฉพาะ ก่อนทำการเปรียบเทียบระหว่างชิ้นส่วนโค้ดในขั้นตอนต่อไป ซึ่งรูปแบบในการทำให้เป็นบรรทัดฐานเดียวกันมีหลายอย่าง ได้แก่ การเปลี่ยนชื่อ เช่น การเปลี่ยนชื่อตัวแปรและข้อมูลให้อยู่ในรูปแบบเดียวกันเพื่อการตรวจสอบ หรือ การกรอง เป็นการกรองข้อความประเภทที่ไม่มีความสำคัญออกจากชิ้นส่วนที่อาจเป็นชิ้นส่วนของสำเนาโค้ด

3) การเปรียบเทียบ

ในการเปรียบเทียบจะทำการเปรียบเทียบระหว่างชิ้นส่วนโค้ดในแต่ละบรรทัด ในขั้นตอนแรกจะใช้ขั้นตอนวิธีหาลำดับย่อยร่วมกันที่ยาวที่ โดยกำหนดค่าขีดเริ่มเพื่อใช้ในการหาช่วงในการเปรียบเทียบความแตกต่างที่มากที่สุดเพื่อใช้ในการเปรียบเทียบ และทำการเปรียบเทียบเพื่อหาสำเนาโค้ดออกมา นอกจากนี้แล้วค่าขีดเริ่มยังมีส่วนช่วยลดภาระในการเปรียบเทียบ

2.2.4 Detecting Software Theft via Whole Program Path Birthmarks [16]

บทความนี้เป็นอีกหนึ่งเทคนิคที่ใช้ในการตรวจสอบการโจรกรรมซอฟต์แวร์โดยในบทความนี้จะใช้วิธีการสังเกต เส้นทางการทำงานของโปรแกรมโดยใช้การตามรอยการควบคุมการไหล โดยในบทความนี้จะมีการตามรอยเส้นทางเพื่อที่ จะหาความคล้ายคลึงกันของพฤติกรรม และหลังจากการประเมินพบว่า การตรวจจับการคัดลอกของโปรแกรม แบบ Whole Program Path birthmarks มีความยืดหยุ่นกกว่าเทคนิคต่างๆที่เกิดขึ้นก่อนหน้านี้



รูปที่ 2.20 แสดงตัวอย่างการใช้งานตามขั้นตอนของ Whole Program Path birthmarks [16]

2.2.5 Design patterns based Pre-processing of Source Code for Plagiarism Detection[17]

บทความนี้ได้กล่าวถึงรูปแบบของการตรวจจับการขโมยความคิดในโปรแกรมซอฟต์แวร์ โดยปัจจุบันนี้ได้มีเครื่อง มือมากมายในการช่วยในการตรวจจับเอกสาร ข้อความ และแหล่งที่มา แต่ประเด็นสำคัญของบทความฉบับนี้ได้นำเสนอ ว่า โดยส่วนใหญ่ของเครื่องมือตรวจจับการขโมยความคิดสำหรับซอร์สโค้ดจะต้องทำการจำแนกประเภทก่อนการประมวล ผล ในบทความฉบับนี้ได้รวบรวมวิธีการค้นโค้ดที่เหมือนกันและสร้าง ขั้นตอนก่อนการประมวลผล (pre-processing steps) และอธิบายเกี่ยวกับรูปแบบของการออกแบบบนพื้นฐานของขั้นตอนก่อนการประมวลผล (Design Patterns base pre-processing steps) และทำให้พบว่า รูปแบบของการออก (Design Pattern) มีความสำคัญในการตรวจ จับการคัดลอก โดยการตรวจสอบโดยใช้ uniform tokens ในกรณีที่เราไม่สามารถมองเห็นรูปแบบของการออกแบบ ขั้นตอนก่อนการประมวลผล (pre-processing steps) ถือเป็นขบวนการ สำคัญ และเป็นประโยชน์ในการตรวจจับของกรณีที่ไม่สามารถมองเห็น รูปแบบของการออก (Design Pattern)

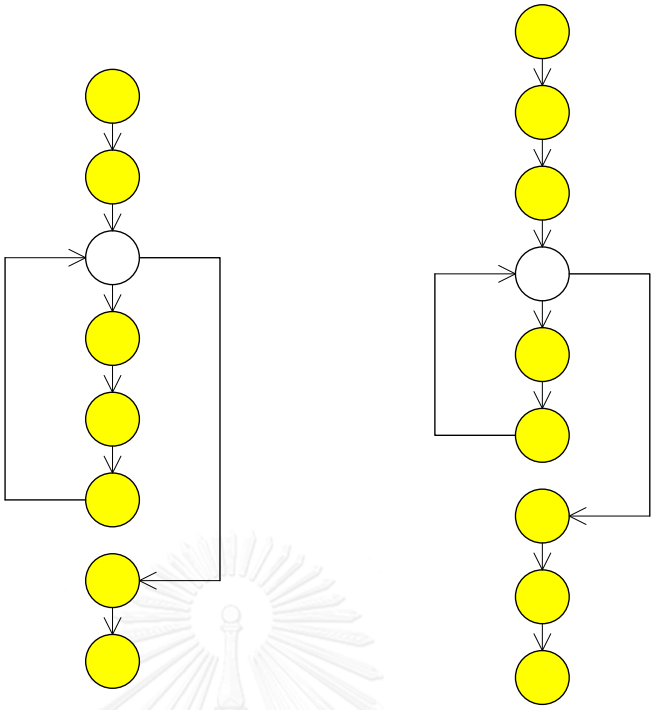
บทที่ 3
การออกแบบวิธีการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์

ในบทนี้จะกล่าวถึงการออกแบบวิธีการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ของการแก้ปัญหาในการระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดดังกล่าวโดยวิธีการตรวจหาความเหมือนของโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน โดยใช้วิธีการใช้เส้นทางของการทดสอบซอฟต์แวร์ โดยแบ่งการทำงานออกเป็น 2 ขั้นตอนการตรวจหาสำเนาโค้ดที่มีลักษณะของโครงสร้างการทำงานที่เหมือนกันโดยการเปรียบเทียบกราฟการไหลของการควบคุม และการระบุเป็นสำเนาโค้ดประเภทที่ 4 เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์

3.1 ที่มา ภาพรวมของขั้นตอน และวัตถุประสงค์ของการออกแบบวิธีการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์

การพัฒนาสินทรัพย์หลักถือเป็นกิจกรรมหลักสำหรับการพัฒนาผลิตภัณฑ์ซอฟต์แวร์เพื่อนำมาใช้ซ้ำ โดยเฉพาะอย่างยิ่งในการพัฒนาสินทรัพย์หลักในแนวทางปฏิกิริยา ซึ่งใช้การหาส่วนทั่วไประหว่างผลิตภัณฑ์ที่มีอยู่แล้วเพื่อพัฒนาเป็นสินทรัพย์ทั่วไป และถึงแม้ว่าจะมีงานวิจัยในหัวข้อดังกล่าวมากมายก็ตาม แต่ส่วนใหญ่มักจะมุ่งเน้นการพัฒนาสินทรัพย์หลักไปที่ระดับของการนำการออกแบบและการวิเคราะห์กลับมาใช้ใหม่ โดยไม่ได้สนใจในส่วนรายละเอียดระดับโค้ดหรือขั้นตอนของการพัฒนา

ในงานวิจัยที่ผ่านมา[7] ได้มีการริเริ่มวิธีการระบุตัวเลือกสินทรัพย์ร่วมจากซอร์สโค้ดระหว่างผลิตภัณฑ์โดยการนำเสนอ การออกแบบวิธีการเพื่อทำการระบุตัวเลือกสินทรัพย์ร่วมจากซอร์สโค้ดระหว่างผลิตภัณฑ์โดย ใช้เทคนิคของการตรวจหาสำเนาโค้ดเพื่อทำการระบุส่วนร่วมของคู่สำเนาโค้ด และใช้รูปแบบการออกแบบประเภทเมท็อดแม่ (Template method) ในการแยกความแตกต่างออกจากสำเนาโค้ดที่ตรวจพบและก่อแบบเป็นตัวเลือกสินทรัพย์ร่วมในผลิตภัณฑ์

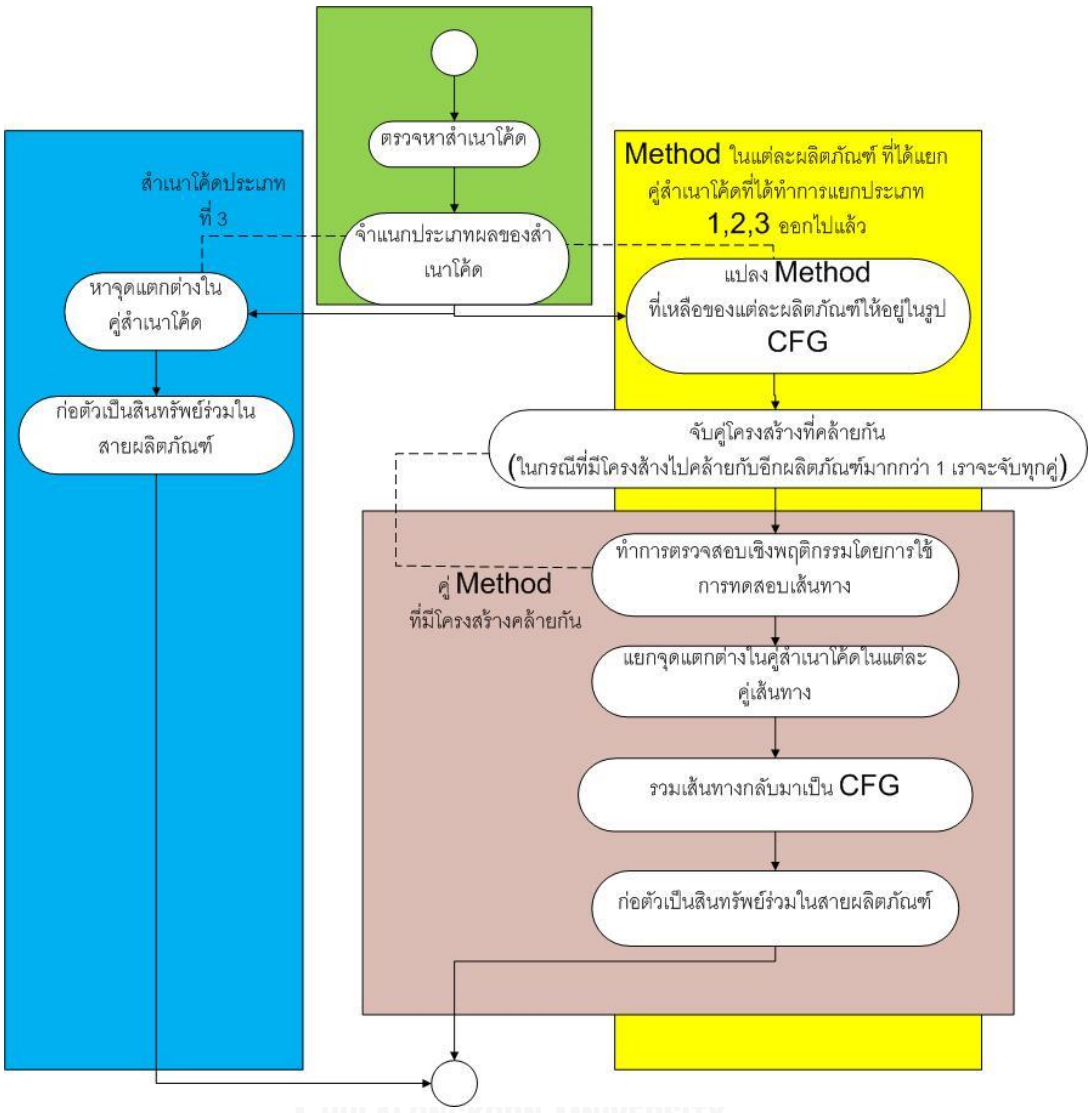


(ก) ผลิตภัณฑ์ A (ข) ผลิตภัณฑ์ B

รูปที่ 3.1 ตัวอย่าง CFG ของ (ก) ผลิตภัณฑ์ A (ข) ผลิตภัณฑ์ B ที่มีพฤติกรรมการทำงานที่เหมือนกัน

อย่างไรก็ตามบทความ[7]เสนอการระบุตัวเลือกสินทรัพย์ร่วมจากสำเนาโค้ดประเภทที่ 3 เท่านั้นคือสำโค้ดที่ทำการเปลี่ยนแปลง และเพิ่มลบคำสั่ง แต่ไม่ได้รวมถึงสำเนาโค้ดประเภทที่ 4 ที่ชิ้นส่วนโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน เพื่ออธิบายถึงแนวความคิดของปัญหา ในที่นี้ได้ยกตัวอย่างโดยสมมุติให้มีผลิตภัณฑ์ A ซึ่งมีการแปลงจากโค้ดเป็นกราฟการไหลของการควบคุมดังแสดงในรูปที่ 3.1(ก) และ ผลิตภัณฑ์ B ซึ่งมีการแปลงจากโค้ดเป็นกราฟการไหลของการควบคุมดังแสดงไว้ในรูปที่ 3.1(ข) เช่นกัน จะเห็นได้ว่ากราฟทั้งสองมีลักษณะคล้ายกันแต่จะมีความแตกต่างกันอยู่บ้างในบางส่วนซึ่งจำเป็นจะต้องแยกออกมาเพื่อพัฒนาต่อเป็นสินทรัพย์ร่วม ในที่นี้ผลิตภัณฑ์ A มีการเขียนป็นแบบ While-loop ส่วนผลิตภัณฑ์ B มีการเขียนเป็นแบบ for-loop ซึ่งมีการแตกต่างด้านการเขียนแต่มีวัตถุประสงค์ของการทำงานที่เหมือนกัน

ในงานนี้จะนำเสนอการจัดการพัฒนาต่อเป็นสินทรัพย์ร่วมที่กล่าวมาข้างต้น แต่ถึงอย่างไรก็ตามเราจำเป็นจะต้องจำแนกให้ได้ซึ่งสำเนาโค้ดประเภทที่ 4 ออกมาก่อน ถึงจะมาการระบุส่วนร่วมที่อยู่ภายในโค้ดเพื่อพัฒนาต่อสินทรัพย์ร่วม เมื่อเปรียบเทียบกับงานวิจัยที่ผ่านมาจะแสดงให้เห็นภาพรวมของงานวิจัยที่เปรียบเทียบกันดังรูปที่ 3.2



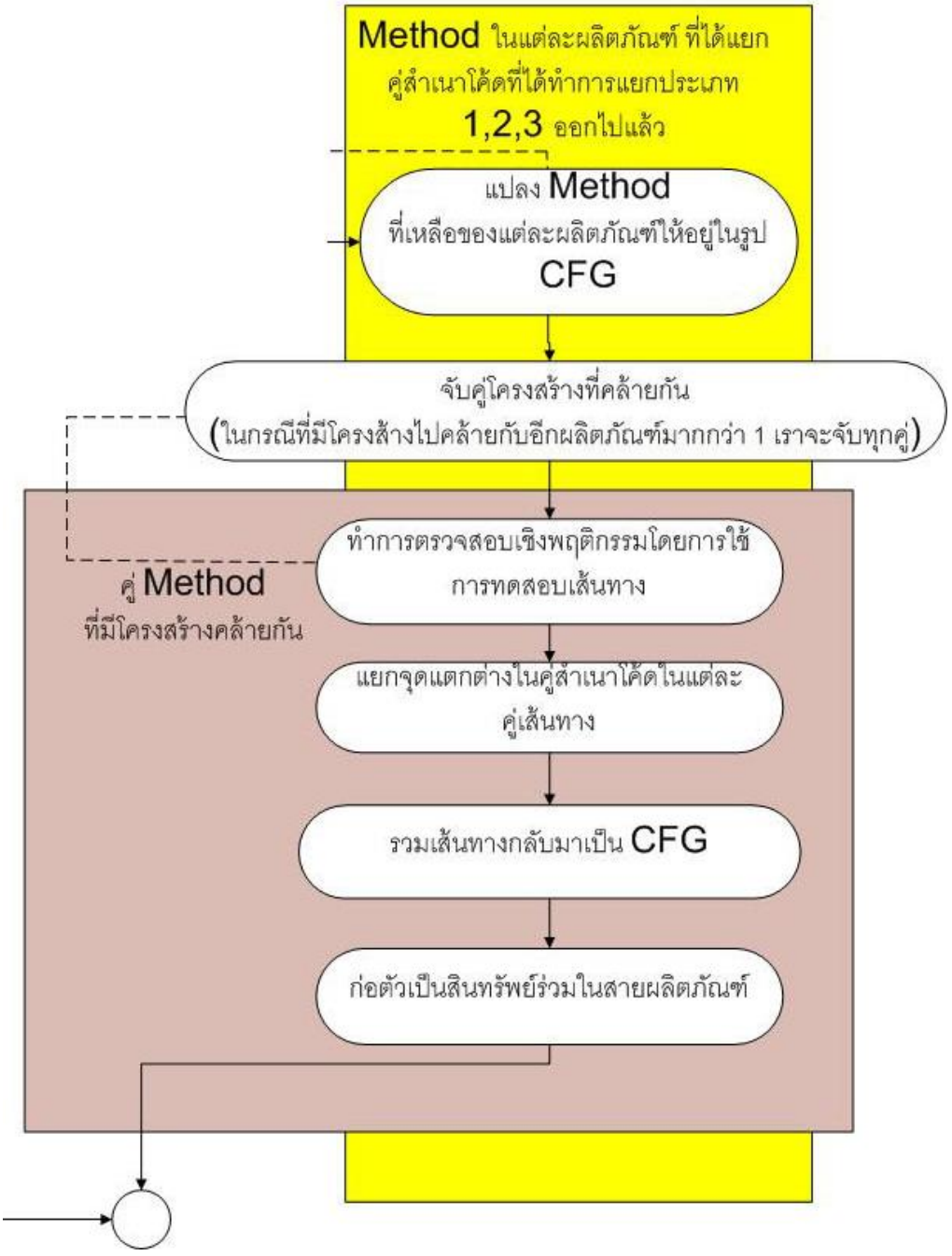
(3.2ก) (3.2ข) (3.2ค)

รูปที่ 3.2 ภาพรวมเปรียบเทียบขั้นตอนของงานวิจัยในปัจจุบัน

จากรูปที่ 3.2 แสดงให้เห็นว่า ในงานวิจัยที่ผ่านมาได้ทำการจำแนกสำเนาโค้ดได้ออกมาทั้งหมด 3 ประเภทแล้วนำสำเนาโค้ดประเภทที่ 3 ไปใช้แยกจุดแตกต่างออก อย่างไรก็ตามผู้วิจัยได้มีความเห็นว่าโค้ดที่ไม่ถูกมาระบุในสำเนาโค้ดทั้ง 3 ประเภทนั้นอาจจะมีสำเนาโค้ดประเภทที่ 4 ซ่อนภายในเมท็อดที่เหลืออยู่ดังนั้นแล้วในงานวิจัยนี้เราจึงได้แบ่งแนวทางของการวิจัยออกเป็น 2 ส่วนดังดังรูปที่ 3.2ค กรอบบน และรูป 3.2ค กรอบล่าง ต่อไปนี้

1).การตรวจสอบเชิงโครงสร้างด้วยวิธีการตรวจหาสำเนาโค้ดที่มีลักษณะของโครงสร้างการทำงานที่เหมือนกันโดยการเปรียบเทียบกราฟการไหลของการควบคุม (รูป (3.2ค) กรอบบน)

2).การตรวจสอบเชิงพฤติกรรมด้วยวิธีการระบุเป็นสำเนาโค้ดประเภทที่ 4 โดยใช้เส้นทางของการทดสอบซอฟต์แวร์เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์ (รูป (3.2ค) กรอบล่าง)

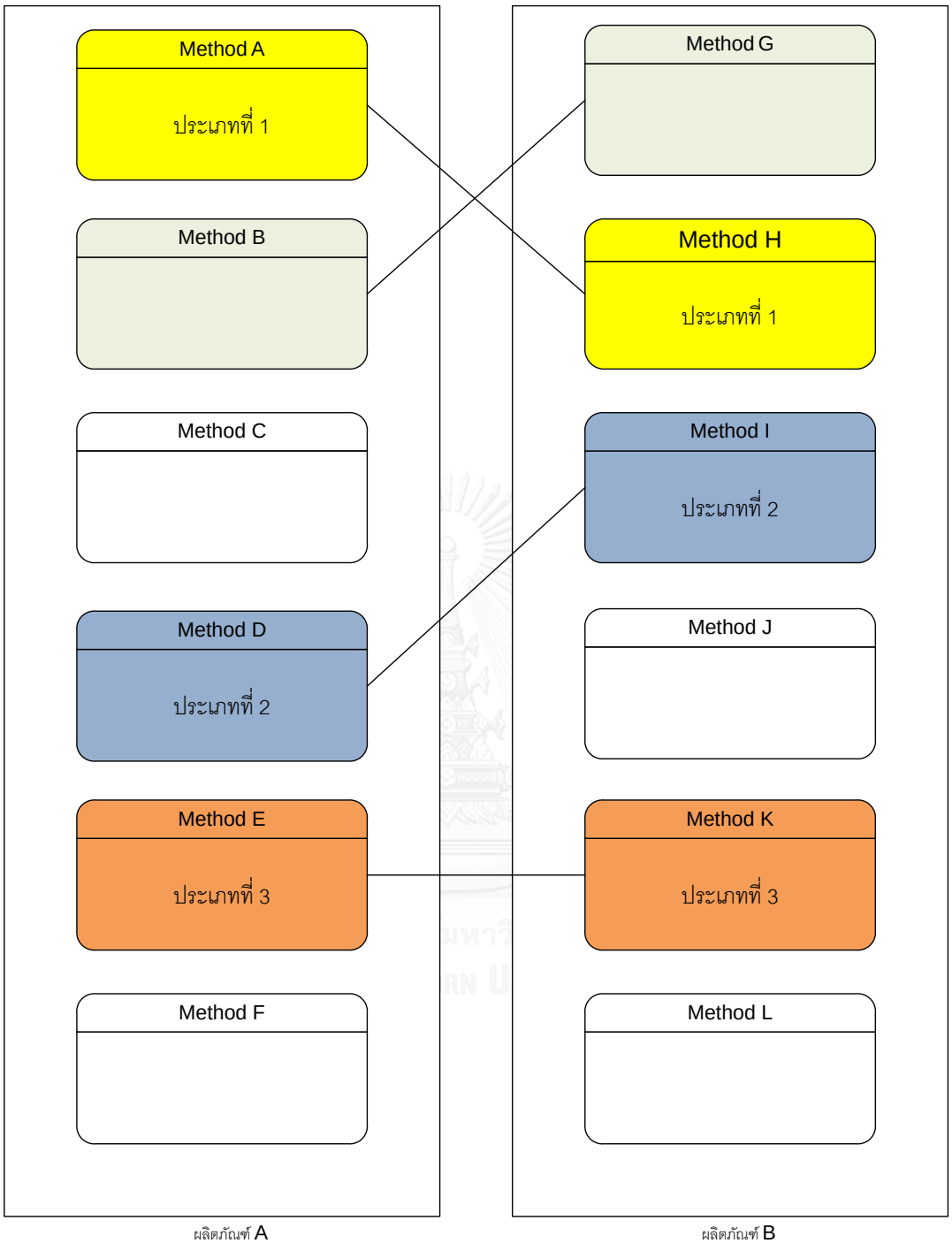


รูปที่ 3.3 ภาพรวมของงานวิจัยในปัจจุบัน

ดังนั้นงานวิจัยนี้มีจุดประสงค์เพื่อแก้ปัญหาในการระบุตัวเลือกสินทรัพย์ร่วมในระดับโค้ดดังกล่าวโดยวิธีการตรวจหาความเหมือนของโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือ

ผลลัพธ์การทำงานที่เหมือนกัน โดยใช้วิธีการตรวจสอบหาความเหมือนกันของโครงสร้าง โดยวิธีการสร้างกราฟการไหลของข้อมูล (Control Flow Graph) จาก โค้ด เนื่องจากวิธีดังกล่าวสามารถเข้าใจถึงโครงสร้างโดยรวมได้เป็นอย่างดี แล้วใช้วิธีการทดสอบเส้นทางเพื่อหาเส้นทางทั้งหมดที่เกิดขึ้น เนื่องจากการทดสอบเส้นทางทำให้เข้าถึงพฤติกรรมที่จะเกิดขึ้นในการทำงานทั้งหมด หลังจากนั้นทำการเปรียบเทียบพฤติกรรมที่เกิดขึ้น แล้วทำการก่อตัวแบบตัวเลือกสินทรัพย์ เพื่อใช้สำหรับการพัฒนาผลิตภัณฑ์ที่มีลักษณะการทำงานที่เหมือนกันของเมท็อดที่จะเกิดขึ้นในอนาคต

โดยการออกแบบวิธีการนี้เริ่มจากเราได้ทำการจำแนกประเภทของสำเนาโค้ด (รูป 3.2ข) ในงานวิจัยที่ผ่านมาจะสามารถเห็นได้ชัดว่ามีเมท็อดที่เหลืออยู่ใน 2 ผลิตภัณฑ์ที่ไม่สามารถจำแนกได้ว่าเป็นสำเนาโค้ดประเภทไหน ดังนั้นในงานวิจัยนี้เราจะมีการเริ่มต้นจากข้อมูลนำเข้าที่เป็นเมท็อดที่เหลืออยู่จากการสำเนาโค้ดที่มีการจำแนกทั้ง 3 ประเภทดังรูปที่ 3.4



รูปที่ 3.4 เมท็อดที่เหลืออยู่ใน 2 ผลิตภัณฑ์ที่ไม่สามารถจำแนกได้ว่าเป็นสำเนาโค้ดประเภทไหน (C,J,F,L)

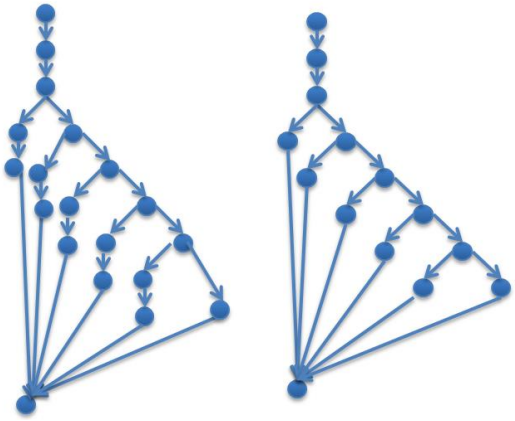
3.2 การตรวจสอบเชิงโครงสร้างด้วยวิธีการตรวจหาสำเนาโค้ดที่มีลักษณะของโครงสร้างการทำงานที่เหมือนกันโดยการเปรียบเทียบกราฟการไหลของการควบคุม(รูป3.2ค กรอบบน)

3.2.1 แปลงโค้ดที่เหลืออยู่ทุกเมท็อดของทั้งสองผลิตภัณฑ์ให้อยู่ในกราฟการไหลของการควบคุมแล้วทำการปรับกราฟให้อยู่ในรูป DD-Path

ใช้ทฤษฎีของ Path Testing มาเปลี่ยนโค้ดให้อยู่ในรูปกราฟการไหลของการควบคุมโดยในการแปลงกราฟนั้นจะต้องอยู่ภายใต้รูปแบบตามงานวิจัยที่ผ่านมา [11],[12],[18],[19],[20],[9] เพื่อจะให้เข้าใจในแนวทางเดียวกันและง่ายในการปรับเปลี่ยนและเปรียบเทียบ

แล้วใช้ทฤษฎี DD-Path [14] มาทำการปรับกราฟเพื่อให้ง่ายต่อความเข้าใจโครงสร้าง มากยิ่งขึ้นรวมถึงการเปรียบเทียบที่จะมีความชัดเจนมากยิ่งขึ้นโดย DD-Path จะทำการยุบรวมประโยคควบคุมที่เป็นเส้นตรงให้ง่ายต่อการเปรียบเทียบมากยิ่งขึ้นโดยจากรูปที่ 3.5(ก) และ 3.5 (ข) แสดงให้เห็นการเปลี่ยนแปลงจากการแปลงโดยใช้ทฤษฎีพบว่ากราฟมีขนาดเล็กลงเนื่องจากมีการย่อประโยคควบคุมที่เป็นเส้นตรงทำให้การนำไปเปรียบเทียบโครงสร้างกับโค้ดที่มีการทำงานแบบเป็นเส้นตรงที่มีขนาดยาวของบรรทัดง่ายยิ่งขึ้นเนื่องจากการเปรียบเทียบขั้นต้นนี้จะเป็นดูลักษณะของโครงสร้างมากกว่ารายละเอียดของโค้ด

หลักของการแปลงเป็น DD-Path จะมีหลักการแปลงดังต่อไปนี้ คือ จุดที่มีพฤติกรรมการทำงานที่เป็นเส้นตรงจะถูกยุบรวมเป็นจุดเดียวกันเพื่อลดความยาวของขั้นตอนในช่วงนั้นทำให้ง่ายต่อการมองโครงสร้างในภาพรวม จุดที่จะไม่ถูกยุบรวมกับจัดอื่นคือจุดเริ่มต้นและสิ้นสุด เมื่อยุบรวมกันแล้วสามารถนำมาเขียนค่าของแต่ละจุดได้ และ สามารถนำค่าดังกล่าวของแต่ละกราฟมาเปรียบเทียบด้วยชุดตัวเลขได้ เพื่อหาโครงสร้างที่มีการทำงานที่คล้ายกันดังที่กล่าวมาในบทที่ 2 หัวข้อที่ 2.1.4



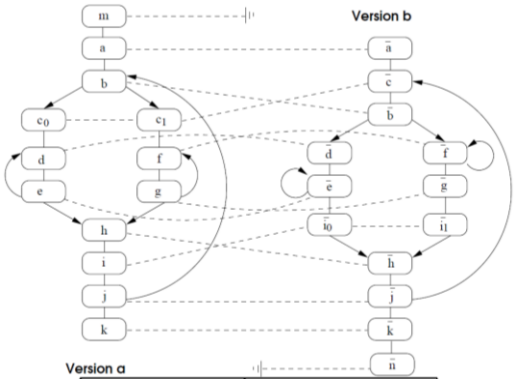
ก) ก่อนแปลงเป็น DD-Path ข) หลังแปลงเป็น DD-Path

รูปที่ 3.5 การเปลี่ยนแปลงจากการแปลงกราฟควบคุมโดยใช้ทฤษฎี DD-Path

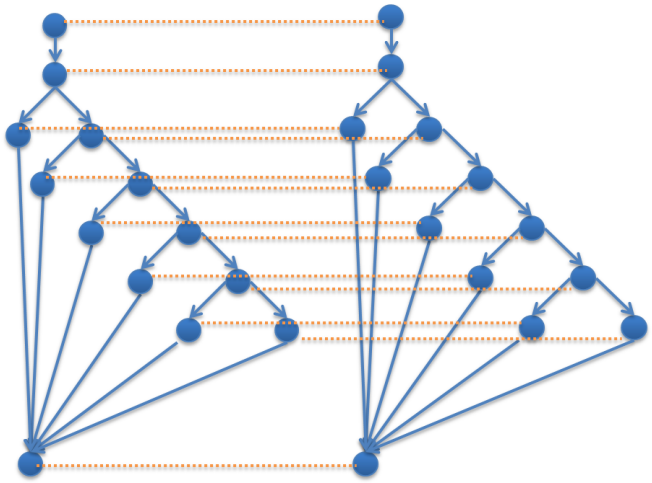
3.2.2 เปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม

เนื่องจากเราไม่สามารถตรวจหาสำเนาโค้ดในการวัดโครงสร้างได้ดังนั้นเบื้องต้นแล้วเราสามารถดูโครงสร้างจากโค้ดประเภทที่ 4 ได้โดยการดูจากกราฟการไหลของการควบคุมโดยในบทความ[21]ได้นำการเปรียบเทียบความคล้ายกันโดยเป็นการเทียบกันระหว่างโหนดดังรูปที่ 3.6 เพื่อหาความเหมือนกันในกราฟการไหลของการควบคุมให้สามารถจับคู่เมท็อดที่มีโครงสร้างที่มีความคล้ายกัน [22]

โดยในบทความดังกล่าวยังไม่ได้มีการปรับกราฟให้อยู่ในรูปแบบของ DD-Path ทำให้เห็นได้อย่างชัดเจนว่าปริมาณการเปรียบเทียบกันระหว่าง โหนดสูงกว่าแบบที่มีการปรับกราฟเป็น DD-Path แล้วดังรูปที่ 3.7



รูปที่ 3.6 การเปรียบเทียบความคล้ายกันระหว่างโหนดโดยทั่วไป



รูปที่ 3.7 การเปรียบเทียบความคล้ายกันระหว่างโหนดที่มีการปรับกราฟเป็น DD-Path

ในการเปรียบเทียบกราฟการไหลของการควบคุมมีวิธีต่างๆดังต่อไปนี้

1).การนำทฤษฎีของ DD-Path เข้ามาช่วยสามารถโดยใช้ค่าเคสของ DD-Path มาทำการเปรียบเทียบแล้วทำการค้นหาชุดของตัวเลขที่มีการเรียนกันที่เหมือนกันของระหว่างสองกราฟการไหลของการควบคุมเพื่อเปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม[14]

2).ใช้การวัดค่าการคำนวนเส้นทางจาก Cyclomatic Complexity เพื่อใช้ในการเปรียบเทียบโครงสร้าง[14]

3).การเปรียบเทียบแผนภาพความสัมพันธ์จากรูปภาพแบบโหนด ต่อ โหนด ในงานวิจัยที่ผ่านมา[13],[21]

วิธีการในการเปรียบเทียบในงานวิจัยนี้จะใช้วิธีการนำทฤษฎีของ DD-Path เข้ามาช่วยสามารถโดยใช้ค่าเคสของ DD-Path มาทำการเปรียบเทียบแล้วทำการค้นหาชุดของตัวเลขที่มีการเรียนกันที่เหมือนกันของระหว่างสองกราฟการไหลของการควบคุมเพื่อเปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม เนื่องจากมีความละเอียดในเชิงโครงสร้างมากที่สุดเมื่อเปรียบเทียบกับอีก 2 วิธี

การนำทฤษฎีของ DD-Path เข้ามาช่วยสามารถโดยใช้ค่าเคสของ DD-Path มาทำการเปรียบเทียบแล้วทำการค้นหาชุดของตัวเลขที่มีการเรียนกันที่เหมือนกันของระหว่างสองกราฟการไหลของการควบคุมเพื่อเปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม จะให้ความสำคัญใน

แต่ละโหนดทำให้การเปรียบเทียบเชิงโครงสร้างจะได้ความละเอียดมากเพราะจะสามารถเข้าถึงพฤติกรรมและโครงสร้างในแต่ละโหนดได้

3.2.3 การเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม

สิ่งที่มักจะสร้างข้อแตกต่างในการเขียนโค้ดที่ต่างแต่ผลลัพธ์ที่เหมือนกันมักจะเป็นจุดตัดสินใจในโค้ดเช่น for-while , if else-switch case , for-recursive , while-recursive เป็นต้น ดังนั้นการเปรียบเทียบดังกล่าวอาจทำให้เราสามารถลดขั้นตอนหรือตัดสินใจได้ว่าโครงสร้างต่างกันหรือไม่ เช่น

กรณีที่ for-while

กรณีที่โค้ด for → for(i=1,i<5,i++)

กรณีที่โค้ด while → i=1;while(i<6){i++;}

จะแสดงให้เห็นว่ากรณีดังกล่าวอาจมีโครงสร้างของกราฟการไหลของการควบคุมที่เหมือนกัน แต่ถ้านำมา reuse อาจเกิดปัญหาได้ เนื่องจากจุดตัดสินใจมีความแตกแต่งกันของการแยกกันของเส้นทาง ทำให้พฤติกรรมที่เกิดขึ้นจะมีความแตกต่างกัน ดังนั้นแล้วการวัดจุดตัดสินใจถือเป็นหนึ่งในขบวนการการวิเคราะห์โครงสร้างที่สำคัญอีกอย่างหนึ่ง

อีกประการหนึ่งที่สำคัญของการเปรียบเทียบการทำงานของจุดตัดสินใจคือกรณีที่ช่วงของการวนซ้ำใช้ช่วงที่แตกต่างกันแต่ให้ค่าจำนวนของรอบที่เท่ากันอาจส่งผลต่อการแสดงพฤติกรรมของการทำงานซึ่งจะแบ่งออกเป็นสองกรณี

1.กรณีที่ค่าเริ่มต้นและค่าสิ้นสุดของทั้งสองกราฟการไหลของการควบคุมต่างกันแต่จำนวนรอบของการวนเท่ากันและค่าที่ควบคุมแต่ละรอบไม่ส่งผลต่อการทำงานภายใน ในกรณีนี้ถือว่ามีลักษณะการทำงานที่เหมือนกันเช่น

```
for(i=1,i<5,i++){          for(i=0,i<4,i++){
j++;                       j++;
}                          }
```

2.กรณีที่ค่าเริ่มต้นและค่าสิ้นสุดของทั้งสองกราฟการไหลของการควบคุมต่างกันแต่จำนวนรอบของการวนเท่ากันและค่าที่ควบคุมแต่ละรอบส่งผลต่อการทำงานภายในและตัวแปรอื่นๆ ในกรณีนี้ถือว่ามีลักษณะการทำงานที่แตกต่างกันเช่น

```
for(i=1,i<5,i++){                    for(i=0,i<4,i++){
j = j+i ;                            j = j+i ;
}                                    }
```

ดังนั้นสิ่งที่เราจะสนใจในการเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม [14] ได้แก่ค่าต่างๆดังต่อไปนี้ จำนวนรูปแบบในการแบ่งเส้นทาง(ค่าจำนวนเต็ม), เส้นทางของการแบ่งแยก (กรณีที1,กรณีที่ 2,กรณีที่ 3,...), มีการวนซ้ำหรือไม่ (Y,N), จำนวนรอบของการวนซ้ำ (ค่าจำนวนเต็ม), ค่าตัวแปรในลูปส่งผลกระทบต่อการวนซ้ำ (Y,N) สิ่งเหล่านี้จะช่วยในการเปรียบเทียบจุดตัดสินใจบนกราฟการไหลของการควบคุม

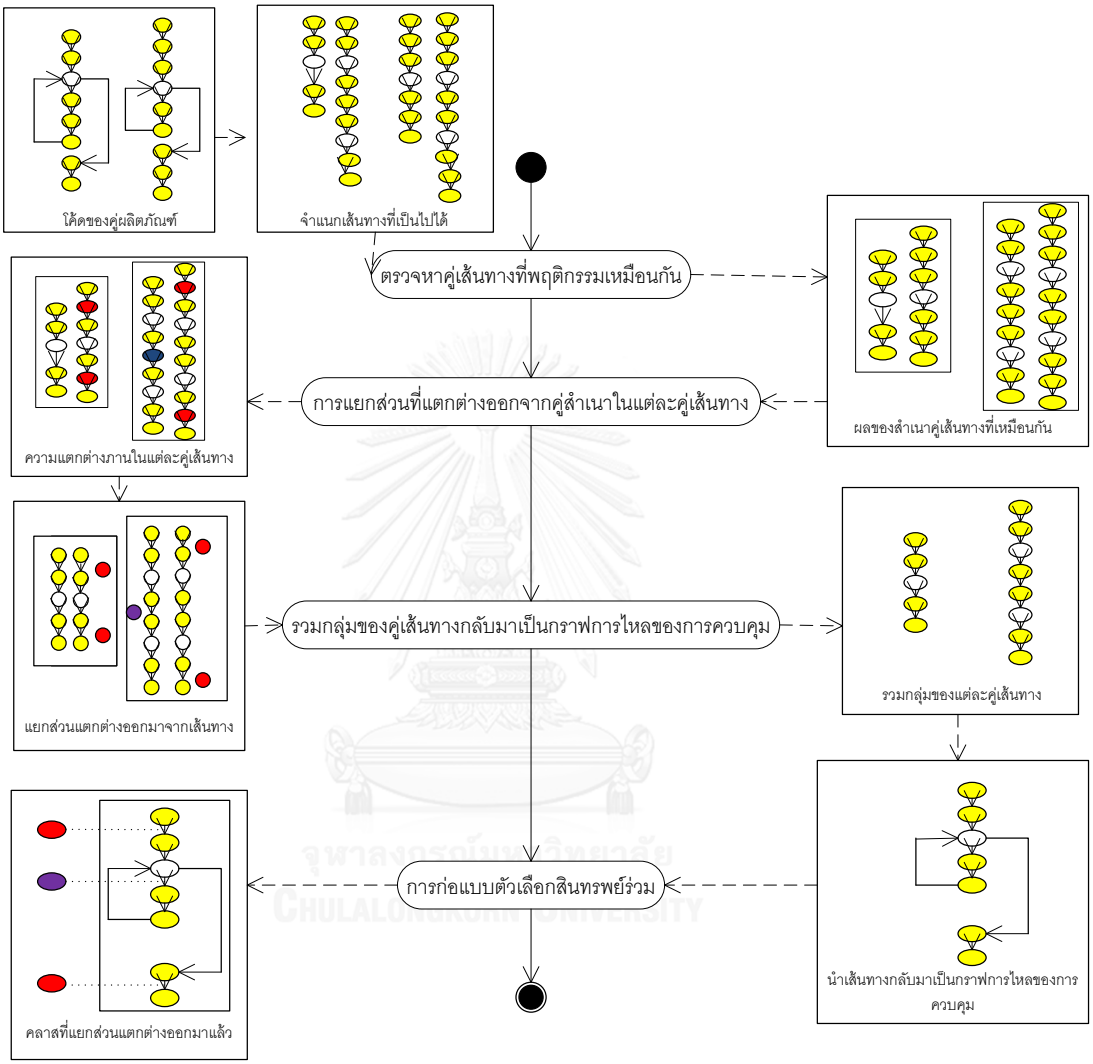
3.3 การตรวจสอบเชิงพฤติกรรมด้วยวิธีการระบุเป็นสำเนาโค้ดประเภทที่ 4 โดยใช้เส้นทางของการทดสอบซอฟต์แวร์เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์

หลังจากที่เราตรวจหาสำเนาโค้ดที่มีลักษณะของโครงสร้างการทำงานที่เหมือนกันโดยการเปรียบเทียบกราฟการไหลของการควบคุมแล้ว สิ่งที่เราจะได้มาหลังจากขั้นตอนข้างต้นคือคู่เมท็อดที่มีโครงสร้างเหมือนกันและจะนำไปเป็นข้อมูลนำเข้าในขั้นตอนนี้

การใช้เส้นทางของการทดสอบซอฟต์แวร์จะทำให้เข้าใจถึงขบวนการของการทำงานที่เกิดขึ้นทั้งหมดดังนั้นแล้ว ถ้าสามารถตรวจสอบการทำงานให้ครอบคลุมทั้งหมดก็จะทำให้เข้าใจถึงพฤติกรรมทั้งหมดและสามารถนำมาเปรียบเทียบการทำงานในต่ละพฤติกรรมได้อย่างครบถ้วน

ภาพรวมของขั้นตอนการแก้ปัญหาของงานวิจัยนี้ในขั้นตอนการตรวจสอบเชิงพฤติกรรมด้วยวิธีการระบุเป็นสำเนาโค้ดประเภทที่ 4 โดยใช้เส้นทางของการทดสอบซอฟต์แวร์เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์ ถูกแบ่งออกเป็น 4 ขั้นตอนได้แก่การตรวจหาสำเนาโค้ดประเภทที่ 4 การแยกส่วนที่แตกต่างออกจากคู่สำเนาในแต่ละคู่เส้นทาง การรวมกลุ่มของคู่เส้นทางกลับมาเป็นกราฟการไหลของการควบคุม และการก่อแบบตัวเลือกสินทรัพย์ร่วมระหว่างผลิตภัณฑ์ทั้งสอง ดังแสดงตามแผนภาพกิจกรรมรูปที่ 3.8 โดยเริ่มต้นจากการตรวจหาสำเนาโค้ดระหว่างคู่เมท็อดในผลิตภัณฑ์ที่มีอยู่แล้ว แล้ว

นำผลของคู่เส้นทางที่ได้มาแยกส่วนที่ต่างกันออกจากเส้นทาง แล้วทำการรวมกลุ่มคู่เส้นทางกลับมีเป็นกราฟการไหลของการควบคุม จากนั้นจึงก่อแบบตัวเลือกสินทรัพย์ร่วมในในรูปแบบของคลาสซึ่งประกอบด้วยส่วนที่สามารถปรับแต่งตามความแตกต่างของแต่ละผลิตภัณฑ์



รูปที่ 3.8 ภาพรวมขั้นตอนการแก้ปัญหาการตรวจสอบเชิงพฤติกรรมด้วยวิธีการระบุเป็นสำเนาโค้ดประเภทที่ 4 โดยใช้เส้นทางของการทดสอบซอฟต์แวร์เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์

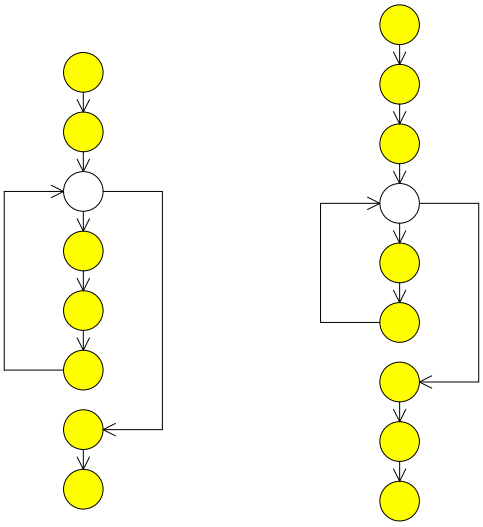
3.3.1 การตรวจหาคู่เส้นทางที่ทำงานเหมือนกัน (สำเนาโค้ดประเภทที่ 4)

งานวิจัยนี้ได้ใช้เทคนิคการตรวจหาคู่เส้นทางที่ทำงานเหมือนกัน เพื่อค้นหาส่วนทั่วไปในโค้ดระหว่างเมท็อดในผลิตภัณฑ์ที่มีอยู่ โดยเป็นการหาสำเนาโค้ดประเภทที่ 4 ที่ชิ้นส่วนโค้ดที่ใช้วิธีการเขียนแตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน โดยใช้การตรวจสอบพฤติกรรมของเส้นทางในการทดสอบซอฟต์แวร์

การตัดสินใจเลือกเทคนิคนี้เนื่องจากการศึกษาวิธีการเขียนโค้ดที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกันนั้นไม่สามารถตรวจสอบได้เพียงความสัมพันธ์เชิงโครงสร้างได้เพียงอย่างเดียว ซึ่งจำเป็นจะต้องอาศัยนิยามเชิงพฤติกรรมเข้ามาร่วมด้วย ดังนั้นแล้วการศึกษาในเรื่องของเส้นทางของการทดสอบซอฟต์แวร์นั้นจะช่วยของพฤติกรรมของการทำงานทั้งหมดตั้งแต่เริ่มต้นจนสิ้นสุดการทำงาน

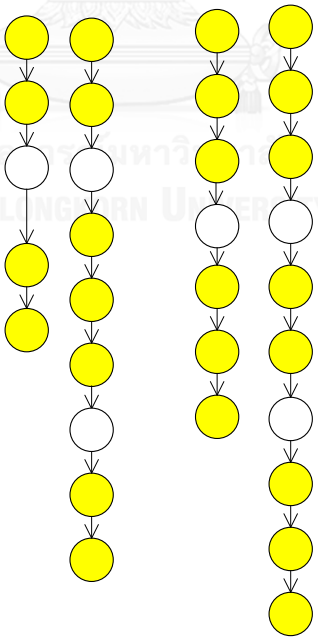
โดยการตรวจหาคู่เส้นทางที่ทำงานเหมือนกันนั้นจะเริ่มขั้นตอนแรกโดยการใช้ทฤษฎีพื้นฐานของการทดสอบซอฟต์แวร์คือการเปลี่ยนจากโค้ดของทั้งสองซอฟต์แวร์ไปเป็นกราฟการไหลของการควบคุมดังรูปที่ 3.9 หลังจากนั้นก็ทำการแจกแจงเส้นทางทั้งหมดที่เป็นไปได้จากกราฟการไหลของการควบคุมดังรูปที่ 3.10 และขั้นตอนถัดมาของการตรวจหาสำเนาหรือพฤติกรรมที่เหมือนกันคือ การจับคู่เส้นทางที่เหมือนกันโดยในที่นี้จะมีข้อจำกัดคือ ในกรณีที่มีการวนซ้ำจะเอามาเพียงการวนซ้ำแค่ครั้งเดียวเพื่อต้องการดูเพียงแค่พฤติกรรมภายในลูป เนื่องจากในการตรวจสอบโครงสร้างนั้น ได้มีการเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุมดังที่อธิบายในหัวข้อ 2.5 ไม่จำเป็นจะต้องทำซ้ำ เมื่อทำการเปรียบเทียบพฤติกรรมแล้วจะได้คู่สำเนาดังรูปที่ 3.11 เป็นคู่ที่มีพฤติกรรมที่เหมือนกัน

ในการตรวจสอบความเหมือนของคู่เส้นทางนั้นมีวิธีการตรวจสอบต่างๆมากมาย แต่ในงานวิจัยนี้จะเลือกใช้วิธี การเปรียบเทียบแบบเชิงข้อความเนื่องจากกราฟที่มีพฤติกรรมซ้ำซ้อนได้มีการจัดรูปแบบให้อยู่ในรูปการทำงานที่เป็นเส้นตรงแล้ว การสลับบรรทัดหรือรูปแบบของคำสั่งในการตัดสินใจจะไม่ส่งผลกระทบในการตรวจสอบ หรือในที่นี้อาจจะใช้วิธีการของการจำแนกการจับคู่พฤติกรรมของการตัดสินใจ แล้วค่อยทำการเปรียบเทียบเชิงข้อความ [23] กับคำสั่งทั่วไป



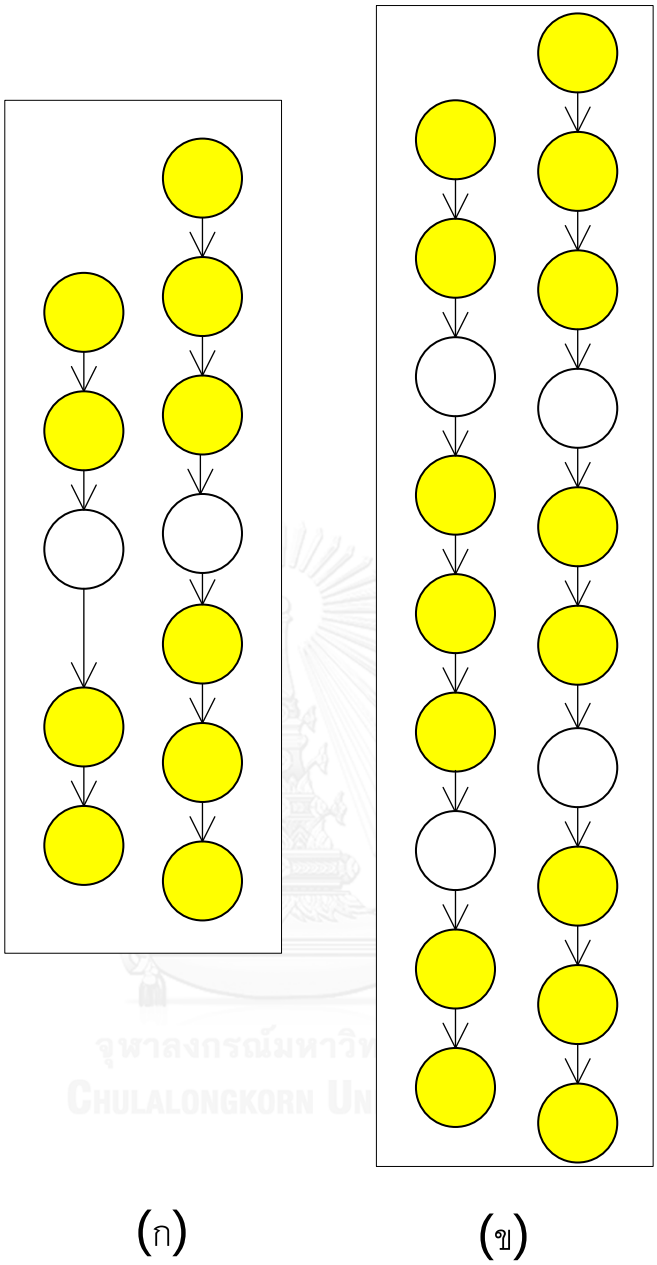
(ก) ผลิตภัณฑ์ A (ข) ผลิตภัณฑ์ B

รูปที่ 3.9 โค้ดของคู่ผลิตภัณฑ์ที่ได้มีการเปลี่ยนเป็นกราฟการไหลของการควบคุม



(ก) ผลิตภัณฑ์ A (ข) ผลิตภัณฑ์ B

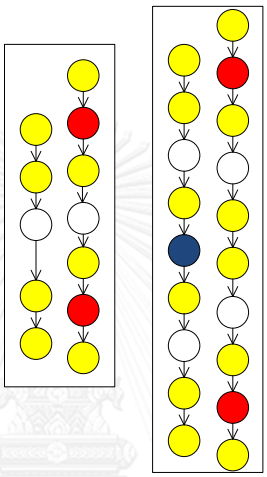
รูปที่ 3.10 การแจกแจงเส้นทางที่เป็นไปได้ของสองผลิตภัณฑ์



รูปที่ 3.11 สำเนาคู่เส้นทางที่มีพฤติกรรมที่เหมือนกัน

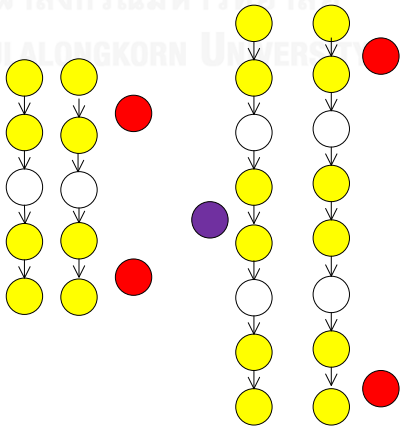
3.3.2 การแยกส่วนที่แตกต่างออกจากคู่สำเนาในแต่ละคู่เส้นทาง

ในการแยกส่วนที่แตกต่างออกจากคู่สำเนาในแต่ละคู่เส้นทาง เราได้จับคู่เส้นทางที่มีพฤติกรรมที่มีการทำงานที่เหมือนกันแล้วในแต่ละคู่กูจะสามารถเปรียบเทียบการทำงานได้ง่ายขึ้นทำให้เราสามารถเห็นข้อแตกต่างของคำสั่งในแต่ละคู่เส้นทางนั้นๆ ดังรูปที่ 3.12 เมื่อเราแยกคำสั่งที่มีความแตกต่างกันออกมาได้ก็จะทำให้มองเห็นภาพของการทำงานได้อย่างชัดเจนยิ่งขึ้นว่าทั้งสองเส้นทางในแต่ละคู่มีพฤติกรรมที่ใกล้เคียงกันมาก ดังรูปที่ 3.13



(ก) (ข)

รูปที่ 3.12 ความแตกต่างของคำสั่งในแต่ละคู่เส้นทางการทดสอบ

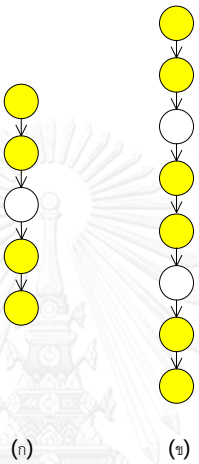


(ก) (ข)

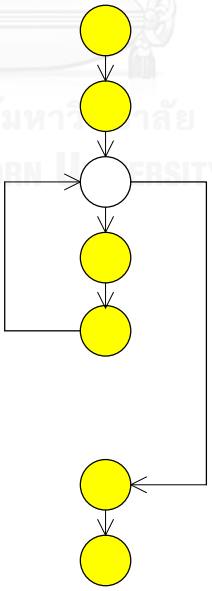
รูปที่ 3.13 การแยกความแตกต่างของคำสั่งออกจากแต่ละคู่เส้นทางการทดสอบ

3.3.3 การรวมกลุ่มของคู่เส้นทางกลับมาเป็นกราฟการไหลของการควบคุม

ในการรวมกลุ่มของคู่เส้นทางกลับมาเป็นกราฟการไหลของการควบคุม โดยเริ่มต้นเราจะได้ตัวแทนของแต่ละคู่เส้นทางการทำงานที่เหมือนกันเมื่อทำการตัดจุดที่แตกต่างออกไปแล้ว ดังรูปที่ 3.14 หลังจากนั้นก็ทำการนำเส้นทางทั้งหมดมารวมกันให้เป็นกราฟการไหลของการควบคุมเพียงกราฟเดียวดังรูปที่ 3.15 แล้วนำไปเปรียบเทียบกับกราฟของผลิตภัณฑ์ทั้งสองจะเห็นได้ว่ากราฟทั้งสามนั้นจะมีโครงสร้างที่คล้ายกัน



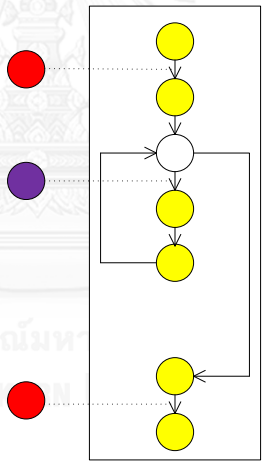
รูปที่ 3.14 ตัวแทนของแต่ละคู่เส้นทางการทำงานที่เหมือนกันเมื่อทำการตัดจุดที่แตกต่างออกไป



รูปที่ 3.15 การนำเส้นทางทั้งหมดมารวมกันให้เป็นกราฟการไหลของการควบคุมเพียงกราฟเดียว

3.3.4 การก่อแบบตัวเลือกสินทรัพย์ร่วม

ในการก่อแบบตัวเลือกสินทรัพย์ร่วม แนวทางการแยกส่วนที่แตกต่างออกจากส่วนทั่วไปนั้น ใช้เมท็อดแม่แบบ ซึ่งเป็นการแก้ปัญหาความซ้ำซ้อนของคำสั่งระหว่างคลาสทั้งสองที่มีพฤติกรรมการทำงานที่เหมือนกัน ในระดับเมท็อดที่มีลักษณะพฤติกรรมของการทำงานที่เหมือนกัน แต่มีความแตกต่างของการทำงานในบางขั้นตอน ทำให้ไม่สามารถรวมทั้งคลาสทั้งสองเข้าด้วยกันได้ เมท็อดแม่แบบจึงทำการแยกชิ้นส่วนของโค้ดส่วนที่เหมือนกันและชิ้นส่วนของโค้ดแตกต่างกันออกจากกัน โดยทำการย้ายชิ้นส่วนของโค้ดในส่วนที่มีคำสั่งของการทำงานเหมือนกันไปไว้ในเมท็อดที่ถูกสร้างขึ้นใหม่ซึ่งจะเป็นคลาสแม่ของคลาสทั้งสอง ซึ่งจะถูกเรียกว่า เมท็อดแม่แบบ ส่วนจุดแตกต่างที่เกิดขึ้นจะถูกแทนที่จุดที่แตกต่างด้วยคำสั่งเรียกใช้เมท็อดซึ่งถูกเรียกว่า เมท็อดฮุค ซึ่งเมท็อดฮุคจะถูกเขียนขึ้นเป็นเมท็อดรูปธรรมในแต่ละคลาสลูกที่มีความต่างกัน ด้วยโค้ดส่วนที่แตกต่างนั้นเอง เพื่อโอเวอร์ไรด์เมท็อดฮุคในคลาสแม่อีกทีหนึ่ง



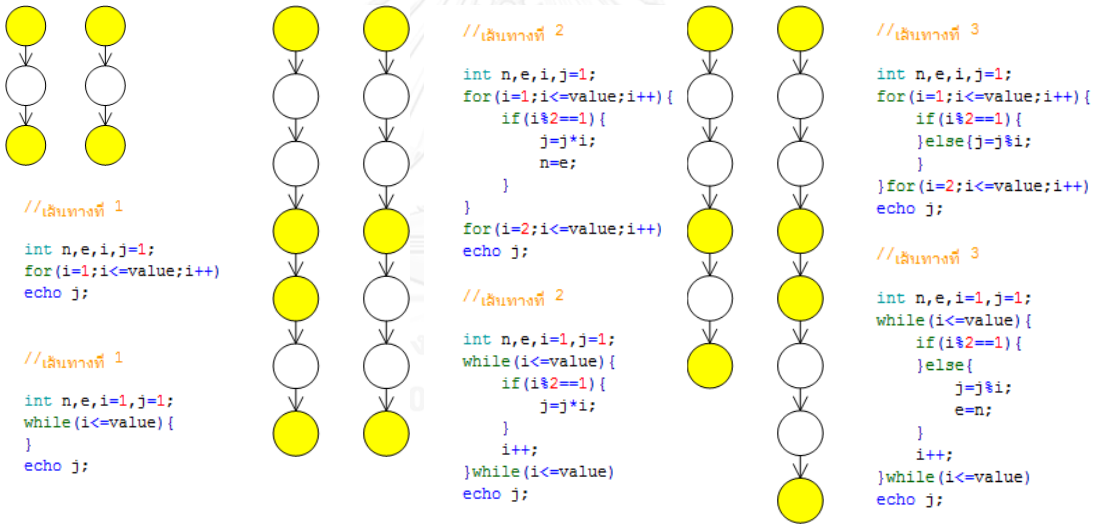
รูปที่ 3.16 แสดงภาพของเมท็อดแม่แบบที่ได้จากการแยกส่วนที่แตกต่างกันออกมา

จากรูปที่ 3.16 เราสามารถเห็นได้ว่าเมื่อนำส่วนที่มีความแตกต่างในจุดที่เป็นสีแดง(สีอ่อน) กลับเข้าไปรวมกับกราฟการไหลของการควบคุมหลักจะพบว่ามีลักษณะที่เหมือนกันกับผลิตภัณฑ์ที่ 2 และเช่นกันถ้านำส่วนที่มีความแตกต่างในจุดที่เป็นสีน้ำเงิน(สีเข้ม) กลับเข้าไปรวมกับกราฟการไหลของการควบคุมหลักจะพบว่ามีลักษณะที่เหมือนกันกับผลิตภัณฑ์ที่ 1 ดังรูปที่ 3.16

3.4 ตัวอย่างการตรวจสอบเชิงพฤติกรรมด้วยวิธีการระบุเป็นสำเนาโค้ดประเภทที่ 4 โดยใช้เส้นทางของการทดสอบซอฟต์แวร์เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์โดยซอร์สโค้ด และ กราฟการไหลของการควบคุม

โดยการเริ่มต้นการทำงานที่โค้ดมีพฤติกรรมของการทำงานที่เหมือนกัน มาทำการตรวจสอบดูแล้วว่าสามารถจับคู่กันได้ ดังรูปที่ 3.17 แล้วนำเอามาแยกส่วนแตกต่างในแต่ละคู่ออกมาเพื่อให้ได้เป็นตัวแทนของทั้งสองเส้นทางในแต่ละคู่ ดังรูปที่ 3.18

เมื่อได้ตัวแทนของแต่ละคู่เส้นทาง ออกมาแล้วขั้นตอนถัดมาจะนำเอาตัวแทนของแต่ละเส้นทางนำกลับมารวมกลับเป็นกราฟ ดังรูปที่ 3.19 ซึ่งกราฟดังกล่าวที่ได้มานี้จะเป็นลักษณะของคลาสแม่ที่มีการนำเอาส่วนต่างๆ ออกมาแล้ว และแยกช่องว่างไว้ให้ส่วนเดิมของทั้งสองผลิตภัณฑ์เพื่อเป็นการรอโอเวอร์ไรด์จากคลาสลูก ดังรูปที่ 3.20

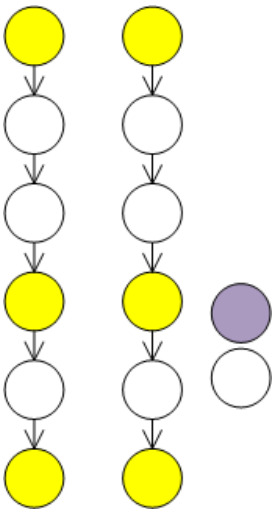


```
//เส้นทางที่ 1

int n,e,i,j=1;
for(i=1;i<=value;i++)
echo j;

//เส้นทางที่ 1

int n,e,i=1,j=1;
while(i<=value){
}
echo j;
```

```
//เส้นทางที่ 2

int n,e,i,j=1;
for(i=1;i<=value;i++){
    if(i%2==1){
        j=j*i;
        n=e;
    }
}
for(i=2;i<=value;i++)
echo j;

//เส้นทางที่ 2

int n,e,i=1,j=1;
while(i<=value){
    if(i%2==1){
        j=j*i;
    }
    i++;
}while(i<=value)
echo j;
```

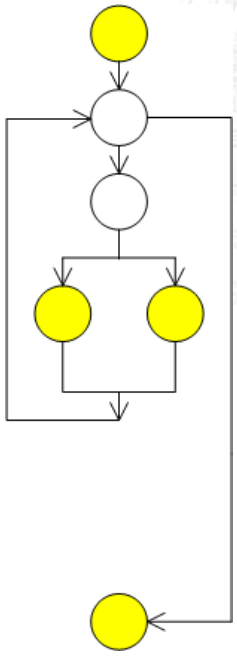
```
//เส้นทางที่ 3

int n,e,i,j=1;
for(i=1;i<=value;i++){
    if(i%2==1){
    }else{j=j%i;
    }
}for(i=2;i<=value;i++)
echo j;

//เส้นทางที่ 3

int n,e,i=1,j=1;
while(i<=value){
    if(i%2==1){
    }else{
        j=j%i;
        e=n;
    }
    i++;
}while(i<=value)
echo j;
```

รูปที่ 3.17 แสดงพฤติกรรมของการทำงานที่เหมือนกัน มาทำการตรวจสอบดูแล้วว่าสามารถจับคู่กันได้



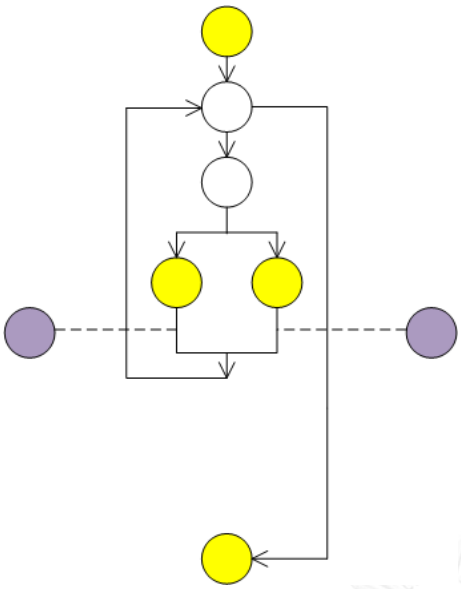
```
//เส้นทางที่ 3

int n,e,i,j=1;
for(i=1;i<=value;i++){
    if(i%2==1){
    }else{j=j%i;
    }
}for(i=2;i<=value;i++)
echo j;

//เส้นทางที่ 3

int n,e,i=1,j=1;
while(i<=value){
    if(i%2==1){
    }else{
        j=j%i;
                                //      e=n;
    }
                                //      i++;
}while(i<=value)
echo j;
```

รูปที่ 3.18 แสดงการนำเอามาแยกส่วนแตกต่างในแต่ละคู่ออกมาเพื่อให้ได้เป็นตัวแทนของทั้งสองเส้นทางในแต่ละคู่



```
//รวมเส้นทาง

int n,e,i,j=1;
for(i=1;i<=value;i++){
    if(i%2==1){
        j=j*i;
    }else{
        j=j%i;
    }
}
echo j;
```

รูปที่ 3.19 แสดงขั้นตอนถัดมาจะนำเอาตัวแทนของแต่ละเส้นทางนำกลับมารวมกลับเป็นกราฟ



```
//รวมแล้วเทาง

int n,e,i,j=1;
for(i=1;i<=value;i++){
    if(i%2==1){
        j=j*i;
                                //      e=n;
    }else{
        j=j%i;
                                //      e=n;
    }
}
echo j;
```

รูปที่ 3.20 แสดงลักษณะของคลาสแม่ที่มีการนำเอาส่วนต่างๆ ออกมาแล้วโอเวอร์ไรด์โดยคลาสลูก

บทที่ 4
การประเมินผลวิธีการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์

ในบทนี้จะกล่าวถึงการทดลองและการวิเคราะห์ผลการทดลองตามแนวคิดที่นำเสนอในบทที่ 3 โดยมีส่วนหลักๆ คือ โปรแกรมที่นำมาใช้ในการทดลอง วัตถุประสงค์ของการทดลอง วิธีการประเมินผล วิธีการทดลองและขั้นตอนการทดลอง และสรุปผลการทดลอง

โดยการทดลองจะทำการนำโปรแกรม 3 ประเภทมาทำการทดลอง ซึ่งแต่ละโปรแกรมจะมีเวอร์ชั่นที่แตกต่างกันออกไป เพื่อทำการทดลองว่าตัวแทนสินทรัพย์หลักที่ได้ออกมาสามารถเป็นตัวแทนของผลิตภัณฑ์ได้หรือไม่ แล้วนำผลการทดลองที่ได้มาประเมินผลโดยการใช้ กรณีทดสอบของผลิตภัณฑ์เบื้องต้นมาทดสอบกับสินทรัพย์ร่วมว่าได้ผลลัพธ์ตามที่ระบุไว้หรือไม่

4.1 โปรแกรมที่นำมาใช้ในการทดลอง

ในที่นี้ได้ใช้โปรแกรมแบ่งออกเป็น 3 ประเภท ที่จะนำมาเป็นการจำลองตัวแทนของผลิตภัณฑ์ที่มีลักษณะของความแตกต่าง การใช้โปรแกรม 3 ประเภทเนื่องจากแต่ละประเภทนั้นมีข้อจำกัดในบางประการอยู่ ซึ่งโปรแกรมแต่ละประเภทได้แก่

- โปรแกรม JabRef ซึ่งเป็นโปรแกรมเดียวกับงานวิจัยในอดีตที่มีใบอนุญาตเป็นแบบโอเพนซอร์สซึ่งเป็นโปรแกรมจัดการฐานข้อมูล ภาษาจาวาจำนวนทั้งสิ้น 17 เวอร์ชัน

- โปรแกรมโจทย์เกี่ยวกับระบบการจัดการเว็ปเมล์ ซึ่งเป็นการจัดการกับระบบกล่องจดหมายจากทั้งหมด 2 เวอร์ชัน

- โปรแกรม TCP2Serial ซึ่งเป็นโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย ซึ่งเป็นเวอร์ชัน 2 และ 3

เนื่องจากประเภทของซอฟต์แวร์ในงานวิจัยในอดีตที่ใช้ในลักษณะของเวอร์ชันซึ่งมีความคล้ายกัน แต่อย่างไรก็ตามพบว่าซอฟต์แวร์ในลักษณะนี้พบสำเนาโค้ดประเภทที่ 4 น้อยมากหรือแทบจะไม่มีเลย เนื่องจากการเปลี่ยนแปลงของเวอร์ชันโดยส่วนใหญ่จะมีแค่การเพิ่มคำสั่งและเพิ่มฟีเจอร์ขึ้นมาใหม่ ไม่มีการเปลี่ยนแปลงทั้งเมท็อดและอยู่ในรูปของการทำงานเดิม ดังนั้นก็จะพบเพียงแค่

สำเนาโค้ดประเภทที่ 3 เพียงเท่านั้น ประเภทของซอฟต์แวร์ในรูปแบบต่อมาคือ ซอฟต์แวร์ที่ถูกสร้างขึ้นโดยนักศึกษาที่มาจากภายใต้ขอบเขตของการทำงานเดียวกันสามารถเห็นผลทดลองได้อย่างชัดเจนมากยิ่งขึ้น

4.2 วัตถุประสงค์ของการทดลอง

วัตถุประสงค์ของการทดลองในงานวิจัยนี้คือ เพื่อวัดประสิทธิผลของการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ เพราะว่าผลของระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ด จะต้องมีความสัมพันธ์กัน กับผลิตภัณฑ์ตั้งต้น ผลลัพธ์ที่มีความแม่นยำและถูกต้อง จะส่งผลให้การระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ มีความถูกต้องด้วย สำหรับในส่วนของวัตถุประสงค์ของการทดลองในงานวิจัยนี้แบ่งออกเป็น 2 วัตถุประสงค์ คือ

1) ออกแบบวิธีการระบุสินทรัพย์ทั่วไปจากซอร์สโค้ดระหว่างสองผลิตภัณฑ์ที่ใช้วิธีการเขียนที่แตกต่างกัน แต่ให้พฤติกรรมหรือผลลัพธ์การทำงานที่เหมือนกัน (สำเนาโค้ดประเภทที่ 4) ให้ได้ผลลัพธ์สอดคล้องกับผลิตภัณฑ์ตั้งต้น

2) จำแนกสำเนาโค้ดประเภทที่ 4 ออกมาได้อย่างแม่นยำและมีความแตกต่างจาก 3 ประเภทอย่างชัดเจน

4.3 ขั้นตอนการทดลองและผลการทดลอง

จากการทดลองในงานวิจัยนี้จะเริ่มจากการนำโปรแกรม 3 ประเภท ที่จะนำมาเป็นการจำลองตัวแทนของผลิตภัณฑ์ที่มีลักษณะของความแตกต่าง การใช้โปรแกรม 3 ประเภทเนื่องจากแต่ละประเภทนั้นมีข้อจำกัดในบางประการอยู่ ซึ่งโปรแกรม JabRef ซึ่งเป็นโปรแกรมเดียวกับงานวิจัยในอดีตจะนำเอาเวอร์ชัน 1.8 และ 2.0 มาทำการทดลอง โปรแกรมโจทย์เกี่ยวกับระบบการจัดการเว็ปเมล์ ซึ่งเป็นการจัดการกับระบบกล่องจดหมาย 2 เวอร์ชั่นมาทำการทดลอง และ โปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่ายจะนำเอาทั้งสองเวอร์ชันมาทดลอง

จากการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ข้างต้น ผู้วิจัยได้แบ่งการทดลองออกเป็น 3 ส่วนคือ การทดลองการจำแนกเพื่อหาสำเนา

โค้ดที่เหลืออยู่จากทั้ง 3 ประเภท การทดลองการตรวจสอบเชิงโครงสร้าง การทดลองการตรวจสอบเชิงพฤติกรรม ซึ่งสามารถแสดงรายละเอียดได้ดังนี้

ตารางที่ 4.1 รายละเอียดข้อมูลสินทรัพย์ที่จัดเก็บโปรแกรม JabRef ในแต่ละเวอร์ชั่นหลัก

เวอร์ชันของ JabRef	คุณลักษณะ		
	จำนวนบรรทัดโค้ด	จำนวนเมท็อด	จำนวนแฟ้มโค้ด
1.0	11352	855	91
1.1	13450	978	101
1.2	16827	1251	149
1.3.1	18750	1311	155
1.4	57517	3918	402
1.5	22041	1542	178
1.6	29016	1953	224
1.7	34320	2438	294
1.8	38680	2698	311
2.0	44444	3322	360
2.1	46263	3424	377
2.2	58152	4039	470
2.3	64730	4453	511
2.4	68953	4773	537
2.5	72512	4976	562
2.6	74264	5112	578
2.7	77814	5492	597

ตารางที่ 4.2 รายละเอียดข้อมูลสินทรัพย์ที่จัดเก็บระบบการจัดการเว็ปเมล์

โปรแกรมของนิสิต	คุณลักษณะ		
	จำนวนบรรทัดโค้ด	จำนวนเมท็อด	จำนวนคลาส
02	11936	994	21
03	12284	1010	21

ตารางที่ 4.3 รายละเอียดข้อมูลสินทรัพย์ที่จัดเก็บโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

โปรแกรม	คุณลักษณะ		
	จำนวนบรรทัดโค้ด	จำนวนเมท็อด	จำนวนคลาส
02	325	18	2
03	242	14	2

4.3.1 การทดลองโปรแกรม JabRef ในเวอร์ชัน 1.8 และ เวอร์ชัน 2.0

4.3.1.1 การทดลองการจำแนกเพื่อหาสำเนาโค้ดที่เหลืออยู่จากทั้ง 3 ประเภท

จากการตรวจหาสำเนาโค้ดโดยวิธีการใช้เทคนิคและเครื่องมือของงานวิจัยในอดีตที่ผ่านมา พบว่า สามารถจำแนกสำเนาโค้ดทั้งสามประเภทตามตารางที่ 4.4 ของโปรแกรม และจะนำข้อมูลดังกล่าวเป็นข้อมูลนำเข้าในขั้นตอนถัดมาเพื่อจะทำการค้นหาสำเนาโค้ดที่หลงเหลืออยู่ของผลิตภัณฑ์ทั้งสอง ตามตารางที่ 4.5

ตารางที่ 4.4 จำแนกสำเนาโค้ดทั้งสามประเภทของโปรแกรม JabRef

ประเภทสำเนาโค้ด	จำนวนเมท็อดสำเนาโค้ดที่ตรวจพบ
ประเภทที่ 1	986
ประเภทที่ 2	117
ประเภทที่ 3	378

ตารางที่ 4.5 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรม JabRef

ประเภท/เวอร์ชัน	เวอร์ชัน 1.8	เวอร์ชัน 2.0
สำเนาโค้ดเริ่มต้น	2698	3322
ประเภทที่ 1	986	986
ประเภทที่ 2	117	117
ประเภทที่ 3	378	378
สำเนาโค้ดที่เหลืออยู่	1217	1841

4.3.1.2 การทดลองการตรวจสอบเชิงโครงสร้าง

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่หลงเหลืออยู่มาทำการตรวจสอบหาโครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันซึ่งขั้นตอนนั้นได้กล่าวไว้ในบทที่ 3 ซึ่งแบ่งออกเป็น 4 ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม ปรับกราฟให้อยู่ในรูปของ DD-Path เปรียบเทียบโครงสร้างของกราฟ เปรียบเทียบจุดตัดสินใจภายในกราฟ และจะได้ผลลัพธ์ในการเปรียบเทียบตามตัวอย่างตารางดัง 4.6

ตารางที่ 4.6 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรม JabRef

ประเภท/เวอร์ชัน	เวอร์ชัน 1.8	เวอร์ชัน 2.0
สำเนาโค้ดเริ่มต้น	2698	3322
ประเภทที่ 1	986	986
ประเภทที่ 2	117	117
ประเภทที่ 3	378	378
สำเนาโค้ดที่เหลืออยู่	1217	1841
โค้ดที่มีโครงสร้างเหมือนกัน	8 (ตัดโครงสร้างแบบเส้นตรง)	8 (ตัดโครงสร้างแบบเส้นตรง)

4.3.1.3 การทดลองการตรวจสอบเชิงพฤติกรรม

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่โครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันมาทำการตรวจสอบเชิงพฤติกรรม ซึ่งขั้นตอนนั้นได้กล่าวไว้ในบทที่ 3 ซึ่งแบ่งออกเป็น ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม การจำแนกเส้นทางของทั้งสองเมท็อดทั้งหมด จับคู่เส้นทางการทำงานที่เหมือนกัน และจะได้ผลลัพธ์ในการเปรียบเทียบตามตัวอย่างตารางดัง 4.7

ตารางที่ 4.7 รายละเอียดการเปรียบเทียบเส้นทางของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรม JabRef

เมท็อด V1.8		เมท็อด V2.0		ผลการเปรียบเทียบ
เมท็อด	จำนวน Path	เมท็อด	จำนวน Path	
Entry file	4	Custom exports	4	Match
Base line	11	Base line	11	Match
Link to internal	5	Link to internal	5	Match
Using exports	4	Entry time stamps	4	Match

ตารางที่ 4.8 รายละเอียดการหาสำเนาโค้ดประเภทที่ 4 อยู่ของผลิตภัณฑ์ทั้งสองโปรแกรม JabRef

ประเภท/เวอร์ชัน	เวอร์ชัน 1.8	เวอร์ชัน 2.0
สำเนาโค้ดเริ่มต้น	2686	3310
ประเภทที่ 1	986	986
ประเภทที่ 2	117	117
ประเภทที่ 3	378	378
สำเนาโค้ดที่เหลืออยู่	1217	1829
โค้ดที่มีโครงสร้างเหมือนกัน	8 (ตัดโครงสร้างแบบเส้นตรงและ LOC <10)	8 (ตัดโครงสร้างแบบเส้นตรงและ LOC <10)

จำนวนสำเนาโค้ดประเภทที่ 4	4	4

4.3.2 การทดลองโปรแกรมโจทย์เกี่ยวกับระบบการจัดการเว็บเมล์

4.3.2.1 การทดลองการจำแนกเพื่อหาสำเนาโค้ดที่เหลืออยู่จากทั้ง 3 ประเภท

จากการตรวจหาสำเนาโค้ดโดยวิธีการใช้เทคนิคและเครื่องมือของงานวิจัยในอดีตที่ผ่านมา พบว่า สามารถจำแนกสำเนาโค้ดทั้งสามประเภทตามตารางที่ 4.9 ของโปรแกรม และจะนำข้อมูลดังกล่าวเป็นข้อมูลนำเข้าในขั้นตอนถัดมาเพื่อจะทำการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองที ตามตารางที่ 4.10

ตารางที่ 4.9 จำแนกสำเนาโค้ดทั้งสามประเภทของโปรแกรมโจทย์เกี่ยวกับระบบการจัดการเว็บเมล์

ประเภทสำเนาโค้ด	จำนวนเมท็อดสำเนาโค้ดที่ตรวจพบ
ประเภทที่ 1	231
ประเภทที่ 2	4
ประเภทที่ 3	18

ตารางที่ 4.10 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์

ประเภท/เวอร์ชัน	V 02	V 03
สำเนาโค้ดเริ่มต้น	994	1010
ประเภทที่ 1	231	231
ประเภทที่ 2	4	4
ประเภทที่ 3	18	18
สำเนาโค้ดที่เหลืออยู่	691	757

4.3.2.2 การทดลองการตรวจสอบเชิงโครงสร้าง

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่หลงเหลืออยู่มาทำการตรวจสอบหาโครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันซึ่งขั้นตอนนั้นได้กล่าวไว้ใน

บทที่ 3 ซึ่งแบ่งออกเป็น 4 ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม ปรับกราฟให้อยู่ในรูปของ DD-Path เปรียบเทียบโครงสร้างของกราฟ เปรียบเทียบจุดตัดสินใจภายในกราฟ และจะได้ผลลัพธ์ในการเปรียบเทียบตามตัวอย่างตารางดัง 4.11

ตารางที่ 4.11 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์

ประเภท/เวอร์ชัน	V 02	V 03
สำเนาโค้ดเริ่มต้น	994	1010
ประเภทที่ 1	231	231
ประเภทที่ 2	4	4
ประเภทที่ 3	18	18
สำเนาโค้ดที่เหลืออยู่	691	757
โค้ดที่มีโครงสร้างเหมือนกัน	10 (ตัดโครงสร้างแบบเส้นตรงและ LOC <10)	10 (ตัดโครงสร้างแบบเส้นตรงและ LOC <10)

4.3.2.3 การทดลองการตรวจสอบเชิงพฤติกรรม

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่โครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันมาทำการตรวจสอบเชิงพฤติกรรม ซึ่งขั้นตอนนั้นได้กล่าวไว้ในบทที่ 3 ซึ่งแบ่งออกเป็น ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม การจำแนกเส้นทางของทั้งสองเมท็อดทั้งหมด จับคู่เส้นทางการทำงานที่เหมือนกัน และจะได้ผลลัพธ์ในการเปรียบเทียบตามตัวอย่างตารางดัง 4.12

ตารางที่ 4.12 รายละเอียดการเปรียบเทียบเส้นทางของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์

V 02		V 03		ผลการเปรียบเทียบ
เมท็อด	จำนวน Path	เมท็อด	จำนวน Path	
Del Mail	4	Del mail	4	Match
Time st	2	Time st	2	Match

Get time	3	Get time	3	Match
extract	2	extract	2	Match
Input view	5	Input view	5	Match
Delete His	3	Delete His	3	Match

ตารางที่ 4.13 รายละเอียดการหาสำเนาโค้ดประเภทที่ 4 ของผลิตภัณฑ์ทั้งสองโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์

ประเภท/เวอร์ชัน	V 02	V 03
สำเนาโค้ดเริ่มต้น	994	1010
ประเภทที่ 1	231	231
ประเภทที่ 2	4	4
ประเภทที่ 3	18	18
สำเนาโค้ดที่เหลืออยู่	691	757
โค้ดที่มีโครงสร้างเหมือนกัน	10 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)	10 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)
จำนวนสำเนาโค้ดประเภทที่ 4	6	6

4.3.3 การทดลองโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

4.3.3.1 การทดลองการจำแนกเพื่อหาสำเนาโค้ดที่เหลืออยู่จากทั้ง 3 ประเภท

จากการตรวจหาสำเนาโค้ดโดยวิธีการใช้เทคนิคและเครื่องมือของงานวิจัยในอดีตที่ผ่านมา พบว่า สามารถจำแนกสำเนาโค้ดทั้งสามประเภทตามตารางที่ 4.14 ของโปรแกรม และจะนำข้อมูลดังกล่าวเป็นข้อมูลนำเข้าในขั้นตอนถัดมาเพื่อจะทำการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองตามตารางที่ 4.15

ตารางที่ 4.14 จำแนกสำเนาโค้ดทั้งสามประเภทของโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

ประเภทสำเนาโค้ด	จำนวนเมท็อดสำเนาโค้ดที่ตรวจพบ
ประเภทที่ 1	5
ประเภทที่ 2	0
ประเภทที่ 3	0

ตารางที่ 4.15 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

ประเภท/เวอร์ชัน	เวอร์ชัน 2	เวอร์ชัน 3
ประเภทที่ 1	5	5
ประเภทที่ 2	0	0
ประเภทที่ 3	0	0
สำเนาโค้ดที่เหลืออยู่	15	11

4.3.1.2 การทดลองการตรวจสอบเชิงโครงสร้าง

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่หลงเหลืออยู่มาทำการตรวจสอบหาโครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันซึ่งขั้นตอนนั้นได้กล่าวไว้ในบทที่ 3 ซึ่งแบ่งออกเป็น 4 ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม ปรับกราฟให้อยู่ในรูปของ DD-Path เปรียบเทียบโครงสร้างของกราฟ เปรียบเทียบจุดตัดสินใจภายใน

ตารางที่ 4.16 รายละเอียดการหาสำเนาโค้ดที่เหลืออยู่ของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

ประเภท/เวอร์ชัน	เวอร์ชัน 2	เวอร์ชัน 3
ประเภทที่ 1	5	5
ประเภทที่ 2	0	0

ประเภทที่ 3	0	0
สำเนาโค้ดที่เหลืออยู่	15	11
โค้ดที่มีโครงสร้างเหมือนกัน	2 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)	2 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)

4.3.1.3 การทดลองการตรวจสอบเชิงพฤติกรรม

การทดลองการตรวจสอบเชิงโครงสร้างในการทดลองส่วนนี้เราจะนำเอาคู่เมท็อดที่โครงสร้างที่มีลักษณะของการทำงานที่เหมือนกันมาทำการตรวจสอบเชิงพฤติกรรม ซึ่งขั้นตอนนั้นได้กล่าวไว้ในบทที่ 3 ซึ่งแบ่งออกเป็น ขั้นตอนได้แก่ การแปลงเป็นกราฟการไหลของการควบคุม การจำแนกเส้นทางของทั้งสองเมท็อดทั้งหมด จับคู่เส้นทางการทำงานที่เหมือนกัน และจะได้ผลลัพธ์ในการเปรียบเทียบตามตัวอย่างตารางดัง 4.17

ตารางที่ 4.17 รายละเอียดการเปรียบเทียบเส้นทางของผลิตภัณฑ์ทั้งสองประเภทของโปรแกรมเกี่ยวกับระบบการจัดการเว็ปเมล์

เมท็อด V2		เมท็อด V3		ผลการเปรียบเทียบ
เมท็อด	จำนวน Path	เมท็อด	จำนวน Path	
SerialClient	5	SerialClient	5	Match

ตารางที่ 4.18 รายละเอียดการหาสำเนาโค้ดประเภทที่ 4 อยู่ของผลิตภัณฑ์ทั้งสองโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย

ประเภท/เวอร์ชัน	เวอร์ชัน 2	เวอร์ชัน 3
ประเภทที่ 1	5	5
ประเภทที่ 2	0	0
ประเภทที่ 3	0	0
สำเนาโค้ดที่เหลืออยู่	15	11

โค้ดที่มีโครงสร้างเหมือนกัน	2 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)	2 (ตัดโครงสร้างแบบเส้นตรง และ LOC <10)
จำนวนสำเนาโค้ดประเภทที่ 4	1	1

4.3.4 การทดลองการก่อสินทรัพย์ร่วมของผลิตภัณฑ์

เมื่อทำการจำแนกสำเนาโค้ดออกมาแล้ว สิ่งที่ได้จากขั้นตอนก่อนหน้านี้คือคู่เส้นทางที่มีพฤติกรรมของการทำงานที่เหมือนกัน จะนำเอามาก่อแบบตัวเลือกสินทรัพย์ร่วม ดังเช่นตัวอย่างนำเอาคู่เส้นทางเมท็อด Get time ของผลิตภัณฑ์ทั้งสองโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์ที่มีพฤติกรรมของการทำงานที่เหมือนกัน มาทำการตรวจสอบดูแล้วว่าสามารถจับคู่กันได้ แล้วนำเอามาแยกส่วนแตกต่างในแต่ละคู่ออกมาเพื่อให้ได้เป็นตัวแทนของทั้งสองเส้นทางในแต่ละคู่

เมื่อได้ตัวแทนของแต่ละคู่เส้นทาง ออกมาแล้วขั้นตอนถัดมาจะนำเอาตัวแทนของแต่ละเส้นทางนำกลับมารวมกลับเป็นกราฟ ซึ่งกราฟดังกล่าวที่ได้มานี้จะเป็นลักษณะของคลาสแม่ที่มีการนำเอาส่วนต่างๆ ออกมาแล้ว และแยกช่องว่างไว้ให้ส่วนเดิมของทั้งสองผลิตภัณฑ์วิอยู่เพื่อเป็นการรอโอเวอร์ไรด์จากคลาสลูก

ในส่วนของคลาสลูกนั้นถูกสร้างขึ้นจากความแตกต่างของผลิตภัณฑ์ทั้งสองที่ถูกแยกออกมาก่อนหน้านี้ ซึ่งจะเป็นของ V.02 และ V.03

4.4 บทวิเคราะห์ผลการทดลอง

ผลการทดลองของงานวิจัยนี้พบว่าการนำซอฟต์แวร์ทั้งสองมาสร้างเป็นสินทรัพย์หลักสามารถทำได้โดยผลลัพธ์ที่ได้คือ ตัวแทนสินทรัพย์สามารถนำมาเป็นตัวแทนของซอฟต์แวร์ทั้งสองได้โดยที่สามารถนำฟังชันก์ของการทำงานที่เหมือนกันมาสร้างเป็นซอฟต์แวร์ใหม่ได้ โดยลดภาระของผู้พัฒนา และสามารถแต่งเติมส่วนเสริมตามความต้องการ

โดยที่ผลิตภัณฑ์ทั้งสองโปรแกรม JabRef จากเวอร์ชัน 1.8 จำนวน 2696 เมท็อด และเวอร์ชัน 2.0 จำนวน 3318 เมท็อด สามารถนำมาเป็นสำเนาโค้ดประเภทที่ 4 ได้เพียง 4 เมท็อด

โดยที่ผลิตภัณฑ์ทั้งสองโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์ จาก V.2 จำนวน 994 เมท็อด และV.3 จำนวน 1010 เมท็อด สามารถนำมาเป็นสำเนาโค้ดประเภทที่ 4 ได้เพียง 6 เมท็อด

โดยที่ผลิตภัณฑ์ทั้งสองโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่ายจากเวอร์ชัน 2 จำนวน 15 เมท็อด และเวอร์ชัน 3 จำนวน 11 เมท็อด สามารถนำมาเป็นสำเนาโค้ดประเภทที่ 4 ได้เพียง 1 เมท็อด

4.5 วิธีการประเมินผล

การประเมินผลของการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์จะแบ่งการทดลองออกเป็น 3 ประเภทคือ ขั้นตอนแรกการประเมินผลการทดลองของการตรวจหาพฤติกรรมเชิงโครงสร้าง โดยจะทำการประเมินผลจากผลการเปรียบเทียบกราฟและการเทียบค่า DD-Path ของการมาเปรียบเทียบ ในขั้นตอนถัดมาคือการประเมินผลการทดลองของการตรวจหาพฤติกรรม โดยจะทำการประเมินผลจากการตรวจสอบความเหมือนของเส้นทางการทดสอบ และขั้นตอนสุดท้ายคือ การประเมินผลของการระบุตัวแทนสินทรัพย์ทั่วไป จะนำเอาผลลัพธ์ของกรณีทดสอบของตัวแทนสินทรัพย์ทั่วไปที่สร้างขึ้น นำมาเปรียบเทียบกับผลลัพธ์ของกรณีทดสอบของผลิตภัณฑ์ตั้งต้นทั้งสองว่ามีค่าตรงกันหรือไม่

การประเมินผลของการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ ผลลัพธ์ของกรณีทดสอบของตัวแทนสินทรัพย์ทั่วไปที่สร้างขึ้นจะต้องมีค่าเท่ากับผลลัพธ์ของกรณีทดสอบของผลิตภัณฑ์ตั้งต้นทุกค่า

4.6 ประเมินผลสินทรัพย์ร่วมของผลิตภัณฑ์

การสร้างสินทรัพย์ทั่วไปในผลิตภัณฑ์จะได้ผลลัพธ์สูงสุดก็ต่อเมื่อเมท็อดทั้งสองที่มีโครงสร้างของการทำงานที่คล้ายกันในเชิงพฤติกรรมสามารถสกัดสิ่งที่เป็นแม่แบบของการทำงานร่วมกันแยกส่วนที่แตกต่างออกจากส่วนทั่วไปนั้นโดยใช้เมท็อดแม่แบบ

ในงานวิจัยนี้ได้ใช้โปรแกรมแบ่งออกเป็น 3 ประเภท ที่จะนำมาเป็นการจำลองตัวแทนของผลิตภัณฑ์ที่มีลักษณะของความแตกต่าง การใช้โปรแกรม 3 ประเภทเนื่องจากแต่ละประเภทนั้นมีข้อจำกัดในบางประการอยู่ ซึ่งโปรแกรมแต่ละประเภทได้แก่ โปรแกรม JabRef ซึ่งเป็นโปรแกรมเดียวกับงานวิจัยในอดีตที่มีใบอนุญาตเป็นแบบโอเพนซอร์สซึ่งเป็นโปรแกรมจัดการฐานข้อมูล ภาษาจาวาจำนวนทั้งสิ้น 17 เวอร์ชัน โปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์ จากทั้งหมด 2 เวอร์ชั่น และโปรแกรม TCP2Serial ซึ่งเป็นโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่าย

ถึงแม่ข่ายบนเครือข่าย ซึ่งเป็นเวอร์ชัน 2 และ 3 ทั้งสามประเภทที่มีการแทรกโค้ดในลักษณะเมท็อดของการเขียนโค้ดที่มีพฤติกรรมเหมือนกันแต่เขียนต่างกันไว้ทั้งสองผลิตภัณฑ์เพื่อที่จะทำการทดสอบการจับคู่และสร้างสินทรัพย์ร่วมของทั้งสองผลิตภัณฑ์

จากการทดลองในส่วนแรกพบว่า ในขบวนการเปรียบเทียบความเหมือนของกราฟการไหลของการควบคุมเพื่อให้แน่ใจว่าทั้งสองเมท็อดมีโครงสร้างที่เหมือนกันดังผลลัพธ์ในขั้นตอนก่อนหน้านี้ หลังจากนั้นจะเข้าสู่ขบวนการระบุเป็นสำเนาโค้ดประเภทที่ 4 เพื่อก่อเป็นสินทรัพย์ร่วมในผลิตภัณฑ์ เมื่อทำสร้างสินทรัพย์ร่วมของทั้ง 3 ผลิตภัณฑ์จะพบว่ามีบางคู่สำเนาโค้ดที่ไม่สามารถมาสร้างเป็นสินทรัพย์ร่วมได้ เนื่องจากมีจำนวนลูปที่เท่ากันแต่คำสั่งภายในลูปแตกต่างกัน จึงส่งผลให้มีเส้นทางที่เกิดขึ้นไม่เท่ากันและคำสั่งภายในเส้นทางมีความแตกต่างกันอย่างชัดเจน จึงต้องทำการสกัดออกไป

เมื่อนำไปประเมินผลระหว่างเมท็อดเดิมของทั้งสองผลิตภัณฑ์เปรียบเทียบกับสินทรัพย์ร่วมที่เป็นเมท็อดแม่แบบโดยการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์ร่วม ดังตารางที่ 2 พบว่าผลลัพธ์ของการนำกรณีทดสอบเบื้องต้นของทั้งสองผลิตภัณฑ์ไปทดสอบกับสินทรัพย์ร่วมของทั้งสองผลิตภัณฑ์ ผลลัพธ์ที่ได้จะมีค่าความถูกต้องเฉลี่ยเท่ากับ 1 ซึ่งไม่มีความผิดพลาดที่เกิดขึ้นเลย แต่ในส่วนของคู่เมท็อดที่มีความซับซ้อนซึ่งมีความต่างกันอยู่ในส่วนของความแตกต่างที่ไม่มีความเหมือนกันตลอดคู่เส้นทางใดเส้นทางหนึ่งจะถูกสกัดออกในส่วนขั้นตอนที่ 2 จะทำให้ไม่สามารถนำมาสร้างเป็นสินทรัพย์ร่วมได้

ตารางที่ 4.19 แสดงผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์

เปรียบเทียบเมท็อด	Test case ผลิตภัณฑ์ V1.8	Test case ผลิตภัณฑ์ V2.0	ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V1.8		ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V2.0	
Entry file - Custom exports	4	4	4/4	1	4/4	1
Base line - Base line	11	11	11/11	1	11/11	1
Link to internal - Link to internal	5	5	5/5	1	5/5	1

Using exports - Using exports	4	4	4/4	1	4/4	1

ตารางที่ 4.20 แสดงผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์

เปรียบเทียบเมท็อด	Test case V. 02	Test case V. 03	ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V.02		ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V.03	
Del Mail - Del Mail	4	4	4/4	1	4/4	1
Time st - Time st	2	2	2/2	1	2/2	1
Get time - Get time	3	3	3/3	1	3/3	1
extract - extract	2	2	2/2	1	2/2	1
Input view - Input view	5	5	5/5	1	5/5	1
Delete His - Delete His	3	3	3/3	1	3/3	1

ตารางที่ 4.21 แสดงผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์

เปรียบเทียบเมท็อด	Test case V 02	Test case V 03	ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V 02		ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์/ V 03	
SerialClient - SerialClient	5	5	5/5	1	5/5	1

4.7 บทวิเคราะห์ผลการประเมิน

งานวิจัยนี้สามารถนำเสนอแนวทางจัดการกับชิ้นส่วนโค้ดที่มีวิธีการเขียนที่แตกต่างกันแต่ให้พฤติกรรมที่เหมือนกันที่ตรวจหาได้โดยใช้วิธีการของการทดสอบซอฟต์แวร์เพื่อใช้สร้างเป็นตัวแทนสินทรัพย์ทั่วไปทำให้ลดภาระของผู้เชี่ยวชาญในการวิเคราะห์โค้ดของผลิตภัณฑ์เพื่อระบุตัวแทน

สินทรัพย์ทั่วไป

ผลการประเมินของงานวิจัยนี้พบว่าการทดสอบซอฟต์แวร์สามารถตรวจสอบพฤติกรรมของการทำงานของโค้ดในระดับของเมท็อดได้ดีในทุกกรณียกเว้นการเปลี่ยนแปลงทั้งเส้นทางการทดสอบที่ไม่สามารถตรวจสอบได้ โดยจัดการกับชิ้นส่วนโค้ดที่มีวิธีการเขียนที่แตกต่างกันแต่ให้พฤติกรรมที่เหมือนกันที่ตรวจหาได้โดยใช้วิธีการของการทดสอบซอฟต์แวร์ สามารถจัดการค้นหาสำเนาโค้ดประเภทที่ 4 ได้อย่างถูกต้องเมื่อนำมาเปรียบเทียบกับผลลัพธ์ที่ได้จากการก่อตัวเลือกสินทรัพย์ร่วม แนวทางการดำเนินงานต่อไปในอนาคตคือต้องการที่จะระบุสินทรัพย์ร่วมระหว่างผลิตภัณฑ์ตั้งแต่สองผลิตภัณฑ์ขึ้นไปรวมถึงการครอบคลุมสายผลิตภัณฑ์ซอฟต์แวร์

ผลลัพธ์ของการประเมินผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์ของผลิตภัณฑ์ทั้งสองของโปรแกรม JabRef พบว่า ค่า ความถูกต้องของสินทรัพย์ร่วม 2 ผลิตภัณฑ์ กับ กรณีทดสอบตั้งต้น คือ 1 ซึ่งเป็นค่าที่สูงมาก

ผลลัพธ์ของการประเมินผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์ของผลิตภัณฑ์ทั้งสองของโปรแกรมเกี่ยวกับระบบการจัดการเว็บเมล์ พบว่า ค่า ความถูกต้องของ สินทรัพย์ร่วม 2 ผลิตภัณฑ์ กับ กรณีทดสอบตั้งต้น คือ 1 ซึ่งเป็นค่าที่สูงมาก

ผลลัพธ์ของการประเมินผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์ของผลิตภัณฑ์ทั้งสองของโปรแกรมเกี่ยวกับการจัดการข้อมูลการรับส่งข้อมูลระหว่างลูกข่ายถึงแม่ข่ายบนเครือข่าย ซึ่งเป็นเวอร์ชัน 2 และ 3 พบว่า ค่า ความถูกต้องของ สินทรัพย์ร่วม 2 ผลิตภัณฑ์ กับ กรณีทดสอบตั้งต้น คือ 1 ซึ่งเป็นค่าที่สูงมาก

บทที่ 5
การพัฒนาเครื่องมือระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ดด้วยการเปรียบเทียบ

ในบทนี้อธิบายวิธีการพัฒนาเครื่องมือ ซึ่งมีรายละเอียดคือการอธิบายความต้องการเชิงหน้าที่ การออกแบบการทำงาน โดยมุ่งเน้นการจำแนกโค้ดที่มีความแตกต่าง แล้วจึงแยกจุดที่แตกต่างนั้นออกจากส่วนร่วม แล้วสร้างเมท็อดแม่แบบ

ตารางที่ 5.1 ข้อกำหนดเบื้องต้นของระบบ

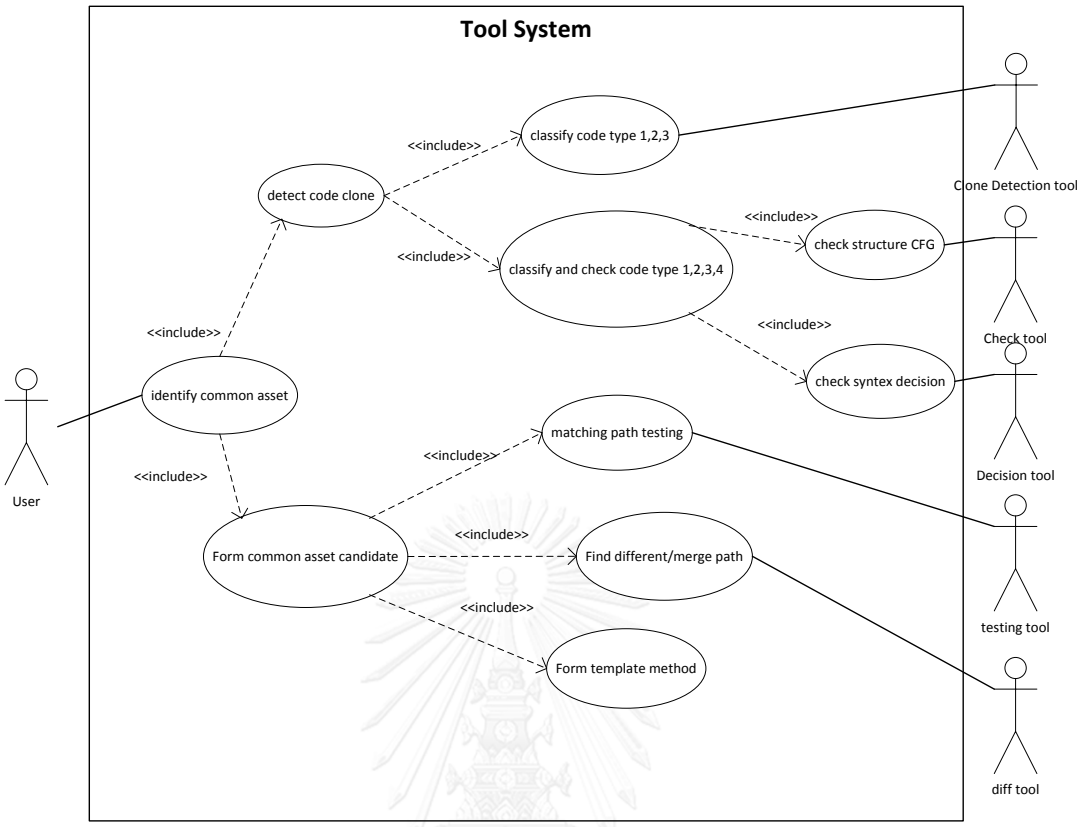
ข้อกำหนดของระบบเบื้องต้น	รายละเอียด
ผู้ใช้งานระบบ	เป็นผู้มีหน้าที่วิเคราะห์ผลิตภัณฑ์ ซึ่งมีความรู้ความสามารถและรู้ขอบเขตการทำงานของผลิตภัณฑ์นั้นเป็นอย่างดี
ข้อมูลนำเข้า	โค้ดของผลิตภัณฑ์ทั้งสองที่จะใช้เปรียบเทียบ
ข้อมูลนำออก	โค้ดซึ่งเป็นสินทรัพย์ร่วมระหว่างผลิตภัณฑ์ทั้งสองในรูปแบบเมท็อดแม่แบบ
สภาพแวดล้อมของระบบ	ระบบปฏิบัติการต่างๆ

ผู้วิจัยได้กำหนดข้อมูลเบื้องต้นของระบบ ซึ่งได้แก่ ผู้ใช้งานระบบ ข้อมูลนำเข้า ข้อมูลนำออก สภาพแวดล้อมของระบบ ซึ่งผู้ใช้งานระบบจะต้องเป็นผู้มีหน้าที่วิเคราะห์ผลิตภัณฑ์ ซึ่งมีความรู้ความสามารถและรู้ขอบเขตการทำงานของผลิตภัณฑ์นั้นเป็นอย่างดี เพื่อเข้าใจถึงปัญหาและความต้องการในการใช้งานเป็นอย่างดี ข้อมูลนำเข้า จะต้องเป็นโค้ดของผลิตภัณฑ์ทั้งสองที่จะใช้เปรียบเทียบ ซึ่งอยู่ภายใต้การทำงานที่มีลักษณะคล้ายๆกัน ข้อมูลนำออก จะแสดงเป็นโค้ดซึ่งเป็นสินทรัพย์ร่วมระหว่างผลิตภัณฑ์ทั้งสองในรูปแบบเมท็อดแม่แบบ ซึ่งคลาสแม่จะเป็นการทำงานที่เหมือนกัน และคลาสลูกจะเป็นส่วนที่แตกต่างกัน

5.1 แผนภาพยูสเคส (Use case diagram)

ความต้องการเชิงหน้าที่ของเครื่องมือที่พัฒนาสามารถเขียนให้อยู่ในรูปของแผนภาพยูสเคส ดังแสดงในรูปที่ 5.1 และอธิบายได้ว่า เครื่องมือที่พัฒนาประกอบด้วยส่วนการทำงานหลักๆ ซึ่งจะบอกถึงขบวนของการทำงาน 10 ยูสเคส ได้แก่

- Identify common asset
- Detect code clone
- From common asset candidate
- Classify code type 1,2,3
- Classify and check type 4
- Matching path testing
- Find different and merge path
- From template method
- Check structure CFG
- Check syntax decision



รูปที่ 5.1 แสดงแผนภาพยูสเคสของระบบเครื่องมือสนับสนุน

ตารางที่ 5.2 คำอธิบายยูสเคส Identify common asset

Use case Name : Identify common asset	**ID :** U01
Primary Actor : User	
Description : ควบคุมการทำงานโดยรวมของการระบุตัวแทนสินทรัพย์ร่วมระหว่างผลิตภัณฑ์ด้วยการใช้ เส้นทางของการทดสอบซอฟต์แวร์	
Pre Condition : ผู้ใช้นำโค้ดของทั้งสองผลิตภัณฑ์มาทำการเปรียบเทียบเพื่อระบุตัวแทนสินทรัพย์ร่วมของทั้งสองผลิตภัณฑ์	
Relationships : **Association :** User **Include :** detect code clone ,From common asset candidate **Extend :** -	
Normal Flow of Event : 1. แปลงโค้ดที่เหลืออยู่ทุกเมท็อดของทั้งสองผลิตภัณฑ์ให้อยู่ในกราฟการไหลของการควบคุมแล้วทำการปรับกราฟให้อยู่ในรูป DD-Path 2. เปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม 3. การเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม 4. การตรวจหาคู่เส้นทางที่ทำงานเหมือนกัน (สำเนาโค้ดประเภทที่ 4) 5. การแยกส่วนที่แตกต่างออกจากคู่สำเนาในแต่ละคู่เส้นทาง 6. การรวมกลุ่มของคู่เส้นทางกลับมาเป็นกราฟการไหลของการควบคุม 7. การก่อแบบตัวเลือกสินทรัพย์ร่วม	
Sub flow : -	
Alternative/Exception Flow : ผู้ใช้สร้างกระบวนการทำงานไม่ถูกต้อง A. มีการแจ้งข้อผิดพลาด ผู้ใช้กำหนดรูปแบบของพารามิเตอร์ไม่ถูกต้อง A. มีการแจ้งข้อผิดพลาด ผู้ใช้กำหนดข้อมูลนำเข้าไม่ถูกต้อง A. มีการแจ้งข้อผิดพลาด	
Post Condition : ตัวเลือกสินทรัพย์ร่วม	

ตารางที่ 5.3 คำอธิบายยูสเคส Detect code clone

Use case Name : Detect code clone	**ID :** U02
Primary Actor : -	
Description : ตรวจหาสำเนาโค้ดระหว่างสองผลิตภัณฑ์	
Pre Condition : ผลิตภัณฑ์ทั้งสองถูกกำหนดให้มีการตรวจหาสำเนาโค้ด	
Relationships : **Association :** - **Include :** Classify Code type 1,2,3 , Classify and check type 1,2,3,4 **Extend :** -	
Normal Flow of Event : 1. เรียกเครื่องมือตรวจหาสำเนาโค้ดที่ตรวจหาด้วยพารามิเตอร์ในงานวิจัยที่ผ่านมา 2. แปลงโค้ดที่เหลืออยู่ทุกเมท็อดของทั้งสองผลิตภัณฑ์ให้อยู่ในกราฟการไหลของการควบคุมแล้วทำการปรับกราฟให้อยู่ในรูป DD-Path 3. เปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม 4. การเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม 5. การตรวจหาคู่เส้นทางที่ทำงานเหมือนกัน (สำเนาโค้ดประเภทที่ 4)	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : ผลของสำเนาโค้ดประเภทที่ 4	

ตารางที่ 5.4 คำอธิบายยูสเคส From common asset candidate

Use case Name : From common asset candidate	**ID :** U03
Primary Actor : -	
Description : ก่อแบบสินทรัพย์ร่วมระหว่างผลิตภัณฑ์	
Pre Condition : นำคู่เมท็อดที่จำแนกเป็นสำเนาโค้ดประเภทที่ 4 มาเป็นข้อมูล	
Relationships : **Association :** - **Include :** matching path testing ,Find different and merge path ,From template method **Extend :** -	
Normal Flow of Event : 1. การแยกส่วนที่แตกต่างออกจากคู่สำเนาในแต่ละคู่เส้นทาง 2. การรวมกลุ่มของคู่เส้นทางกลับมาเป็นกราฟการไหลของการควบคุม 3. การก่อแบบตัวเลือกสินทรัพย์ร่วม	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : คลาสแม่ที่เป็นเมท็อดแม่แบบ และคลาสลูกที่เป็นส่วนแตกต่าง	

ตารางที่ 5.5 คำอธิบายยูสเคส Classify code type 1,2,3

Use case Name : Classify code type 1,2,3	**ID :** U04
Primary Actor : Clone Detection tools	
Description : เรียกใช้เครื่องมือเพื่อจำแนกสำเนาโค้ดประเภท 1,2,3	
Pre Condition : สำเนาโค้ดที่มีการกำหนดของทั้งสองผลิตภัณฑ์	
Relationships : **Association :** Clone detection tool **Include :** - **Extend :** -	
Normal Flow of Event : 1. เรียกใช้เครื่องมือเพื่อระบุสำเนาโค้ดประเภทที่ 1,2,3 2. จัดเก็บสำเนาโค้ดที่เหลืออยู่จากการจำแนก 1,2,3	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : แฟ้มสำเนาโค้ดที่เหลืออยู่จากการจำแนกสำเนาโค้ดประเภทที่ 1,2,3	

ตารางที่ 5.6 คำอธิบายยูสเคส Classify and check type 4

Use case Name : Classify and check type 4	**ID :** U05
Primary Actor : -	
Description : จำแนกสำเนาโค้ดประเภทที่ 4	
Pre Condition : สำเนาโค้ดที่เหลืออยู่จากการจำแนกสำเนาโค้ดประเภทที่ 1,2,3	
Relationships : **Association :** - **Include :** Check structure CFG ,Check syntax Decision **Extend :** -	
Normal Flow of Event : 1. แปลงโค้ดที่เหลืออยู่ทุกเมท็อดของทั้งสองผลิตภัณฑ์ให้อยู่ในกราฟการไหลของการควบคุมแล้วทำการปรับกราฟให้อยู่ในรูป DD-Path 2. เปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม 3. การเปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม 4. การตรวจหาคู่เส้นทางที่ทำงานเหมือนกัน (สำเนาโค้ดประเภทที่ 4)	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : คู่เมท็อดที่มีโครงสร้างของการทำงานที่เหมือนกัน	

ตารางที่ 5.7 คำอธิบายยูสเคส Matching path testing

Use case Name : Matching path testing	**ID :** U06
Primary Actor : testing tool	
Description : ตรวจสอบและจับคูเส้นทางที่มีการทำงานที่เหมือนกัน	
Pre Condition : คู่เมท็อดที่มีโครงสร้างการทำงานที่เหมือนกัน	
Relationships : **Association :** Testing tools **Include :** - **Extend :** -	
Normal Flow of Event : 1. ทำการการเปลี่ยนเป็นกราฟการไหลของการควบคุม 2. แจกแจงเส้นทางที่เป็นไปได้ของสองผลิตภัณฑ์ 3. แสดงผลลัพธ์สำเนาคู่เส้นทางที่มีพฤติกรรมที่เหมือนกัน	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : คู่เส้นทางของการทำงานที่มีลักษณะคล้ายกันตลอดเส้นทาง	

ตารางที่ 5.8 คำอธิบายยูสเคส Find different and merge path

Use case Name : Find different and merge path	**ID :** U07
Primary Actor : diff tools	
Description : แยกส่วนที่แตกต่างออกจากตัวแทนเส้นทางและรวมเส้นทางที่แยกจุดต่างออกแล้วกลับมา	
Pre Condition : คู่เส้นทางของการทำงานที่มีลักษณะคล้ายกันตลอดเส้นทาง	
Relationships : **Association :** diff tools **Include :** - **Extend :** -	
Normal Flow of Event : 1. ค้นหาความแตกต่างของคำสั่งในแต่ละคู่เส้นทางการทดสอบ 2. ทำการแยกความแตกต่างของคำสั่งออกจากแต่ละคู่เส้นทางการทดสอบ 3. แสดงผลลัพธ์ตัวแทนของแต่ละคู่เส้นทางการทำงานที่เหมือนกันเมือทำการตัดจุดที่แตกต่าง 4. ทำการนำเส้นทางทั้งหมดมารวมกันให้เป็นกราฟการไหลของการควบคุมเพียงกราฟเดียว	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : CFG ที่ถูกจำแนกเส้นทางที่มีความแตกต่างออกจากกัน และจุดแตกต่างที่แยกออกมา	

ตารางที่ 5.9 คำอธิบายยูสเคส From template method

Use case Name : From template method	**ID :** U08
Primary Actor : -	
Description : สร้างคลาสแม่แบบที่มีการโอเวอร์ไรด์คลาสลูกที่มีจุดแตกต่าง	
Pre Condition : ที่ถูกจำแนกเส้นทางที่มีความแตกต่างออกจากกัน และจุดแตกต่างที่แยกออกมา	
Relationships : **Association :** - **Include :** - **Extend :** -	
Normal Flow of Event : 1. แสดงเมท็อดแม่แบบที่ได้จากการแยกส่วนที่แตกต่างกันออกมา 2. แปลงข้อมูลที่ได้ให้อยู่ในลักษณะคลาสแม่ที่เป็นเมท็อดแม่แบบ และคลาสลูกที่เป็นส่วนแตกต่าง	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : คลาสแม่ที่เป็นเมท็อดแม่แบบ และคลาสลูกที่เป็นส่วนแตกต่าง	

ตารางที่ 5.10 คำอธิบายยูสเคส Check structure CFG

Use case Name : Check structure CFG	**ID :** U09
Primary Actor : Checking Structure tool	
Description : ตรวจสอบหาความคล้ายของกราฟการไหลของการควบคุม	
Pre Condition : แฟ้มสำเนาโค้ดที่เหลืออยู่จากการจำแนกสำเนาโค้ดประเภทที่ 1,2,3	
Relationships : **Association :** Checking Structure tool **Include :** - **Extend :** -	
Normal Flow of Event : 1. ทำการการนำทฤษฎีของ DD-Path เข้ามาช่วยสามารถโดยใช้ค่าเคสของ DD-Path มาทำการเปรียบเทียบแล้วทำการค้นหาชุดของตัวเลขที่มีการเรียนกันที่เหมือนกันของระหว่างสองกราฟการไหลของการควบคุมเพื่อเปรียบเทียบความเหมือนของกราฟการไหลของการควบคุม 2. ทำการการวัดค่าการคำนวณเส้นทางจาก Cyclomatic Complexity เพื่อใช้ในการเปรียบเทียบโครงสร้าง	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : คู่เมท็อดที่มีโครงสร้างการทำงานที่เหมือนกัน	

ตารางที่ 5.11 คำอธิบายยูสเคส Check syntax decision

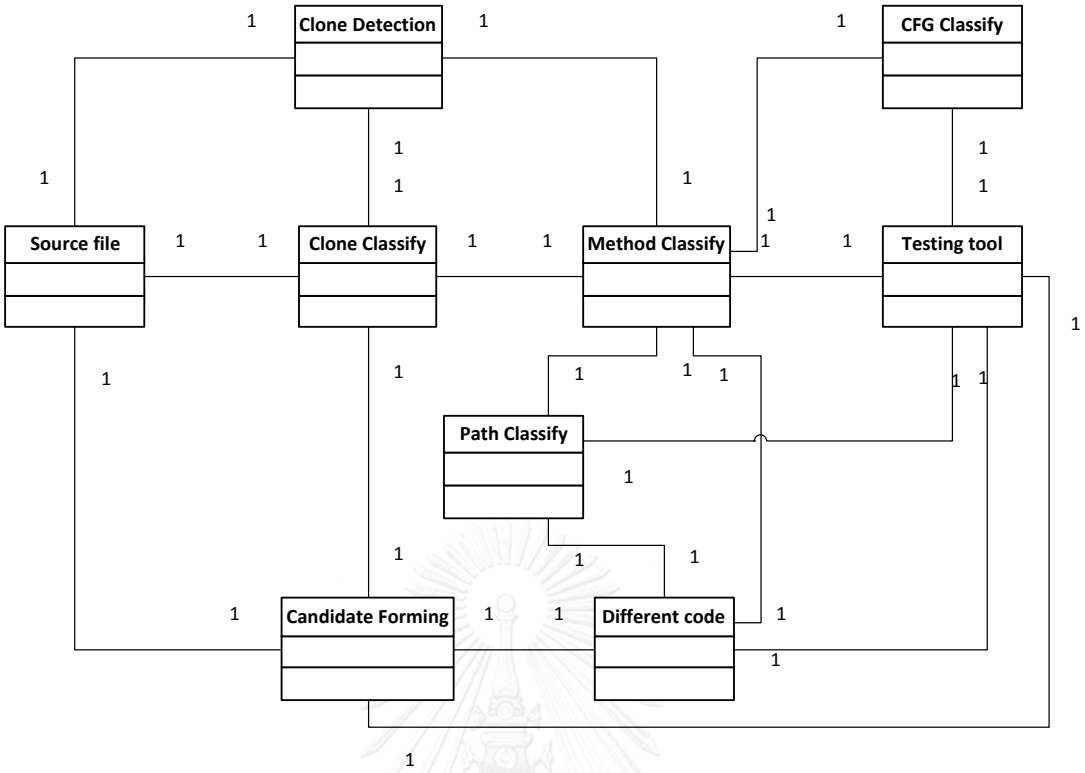
Use case Name : Check syntax decision	**ID :** U10
Primary Actor : syntax tool	
Description : เปรียบเทียบโค้ดในส่วนของจุดตัดสินใจบนกราฟการไหลของการควบคุม	
Pre Condition : แฟ้มสำเนาโค้ดที่เหลืออยู่จากการจำแนกสำเนาโค้ดประเภทที่ 1,2,3	
Relationships : **Association :** syntax tool **Include :** - **Extend :** -	
Normal Flow of Event : 1. เปรียบเทียบค่าดังต่อไปนี้ จำนวนรูปแบบในการแบ่งเส้นทาง(ค่าจำนวนเต็ม),เส้นทางของการแบ่งแยก(กรณีที่ 1,กรณีที่ 2,กรณีที่ 3,...),มีการวนซ้ำหรือไม่(Y,N),จำนวนรอบของการวนซ้ำ(ค่าจำนวนเต็ม),ค่าตัวแปรในลูปส่งผลกระทบต่อการวนซ้ำ(Y,N) 2. ใช้เครื่องมือในการเปรียบเทียบแล้วทำการจัดเก็บข้อมูล 3. จัดเก็บข้อมูลที่มีจุดตัดสินใจเหมือนกัน	
Sub flow : -	
Alternative/Exception Flow : -	
Post Condition : เมท็อดที่มีโครงสร้างการทำงานและจุดตัดสินใจที่เหมือนกัน	

5.2 แผนภาพคลาส (Use case diagram)

แผนภาพคลาสถูกแสดงดังรูปที่ 5.2 ซึ่งประกอบด้วยคลาสดังต่อไปนี้

- Clone detection tool
- Control flow classify
- Source file
- Clone Classify
- Method Classify
- Testing tool
- Path classify
- Candidate forming
- Differrent code

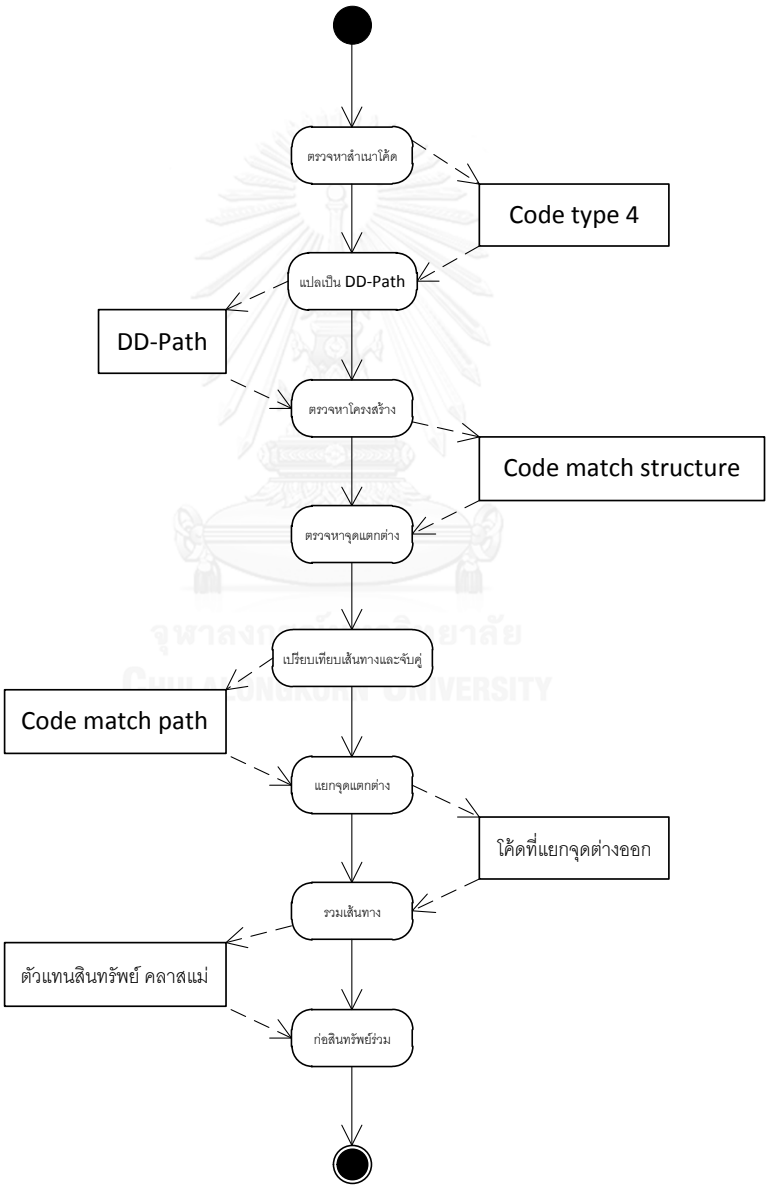
แผนภาพคลาสรูปที่ 5.2 จะอธิบายถึงความสัมพันธ์ที่เกิดขึ้นระหว่าง คลาสของเครื่องมือที่จะเกิดขึ้น โดยเริ่มจากความสำพันธ์ของคลาส clone detection tool ซึ่งจะมีความสัมพันธ์กับคลาส source file ซึ่งเป็นแฟ้มโค้ดผลลัพธ์ของเอกสารคลาส clone classify เป็นแฟ้มของการแยกประเภทของสำเนาโค้ด คลาส method classify เป็นแฟ้มของเมท็อดที่ผ่านการแยก และในส่วนของคลาส method classify นั้นคลาส control flow classify ,Path classify ,different code และ testing tool ที่เป็นคลาสที่อ้างถึง การตรวจสอบหาความเหมือนของโครงสร้าง ,การจำแนกเส้นทางของการทดสอบ ,การหาส่วนต่างที่เกิดขึ้น และเครื่องมือการทดสอบ และ different code นั้นจะเป็น คลาสลูกของ candidate forming ซึ่งเป็นคลาสของการสร้างสินทรัพย์ทั่วไปของงานวิจัยนี้



รูปที่ 5.2 แสดงแผนภาพคลาสของระบบเครื่องมือสนับสนุน

5.3 แผนภาพกิจกรรม (Activity diagram)

แผนภาพกิจกรรมแสดงการทำงานทั้งหมดที่เกิดขึ้นทั้งหมดของเครื่องมือ โดยสามารถแสดงขั้นตอนตามกิจกรรมที่เกิดขึ้นในแผนภาพกิจกรรมดังรูปที่ 5.3 แ มีการเริ่มขั้นตอนตั้งแต่การตรวจหาสำเนาโค้ด การตรวจสอบเชิงโครงสร้าง การหาจุดตัดสินใจ การจำแนกเส้นทางเพื่อเปรียบเทียบละจับคู่ การแยกจุดแตกต่าง การรวมเส้นทางกลับเป็น CFG และสุดท้ายการก่อสินทรัพย์ร่วมระหว่างผลิตภัณฑ์



รูปที่ 5.3 แสดงแผนภาพกิจกรรมของระบบเครื่องมือสนับสนุน

5.4 ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software)

ฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software) ที่ใช้ในการพัฒนาเครื่องมือของงานวิจัยนี้ มีรายละเอียดดังนี้

1) ด้านฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ส่วนบุคคล (Intel Quad core 2.75 GHz)
- หน่วยความจำ (Memory) 8 กิกะไบต์
- จานบันทึกแบบแข็ง (Hard disk) ความจุ 1 เทอร์ราไบต์

2) ด้านซอฟต์แวร์

- เครื่องมือพัฒนาโปรแกรมภาษาจาวาอีคลิปส์ (Eclipse)
- เครื่องมือจำแนกสำเนาโค้ดภาษาจาวา (NiCad3)

บทที่ 6
สรุปผลการวิจัย

ในบทนี้ซึ่งเป็นบทสุดท้ายของวิทยานิพนธ์ฉบับนี้ จะกล่าวถึงการสรุปผลการวิจัย งานวิจัยในอนาคต และบทความวิชาการที่ได้รับการตีพิมพ์ โดยมีรายละเอียดดังนี้

6.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ด ด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ โดยได้นำเอาซอฟต์แวร์ของทั้งสองผลิตภัณฑ์ที่มีลักษณะของขอบเขตการทำงานที่ใกล้เคียงกันนำมาสร้างเป็นสินทรัพย์ที่สามารถนำไปพัฒนาต่อในอนาคตโดยใช้เทคนิควิธีการของการทดสอบซอฟต์แวร์ซึ่งมีประสิทธิภาพของการตรวจสอบที่มีความครอบคลุมในด้านของโครงสร้างซอฟต์แวร์รวมไปถึงสามารถตรวจสอบพฤติกรรมการทำงานของซอฟต์แวร์ได้เป็นอย่างดี นอกจากนี้ยังสามารถนำวิธีการและกระบวนการต่างๆ ของงานวิจัยไปสร้างเป็นเครื่องมือเพื่อสนับสนุนการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ด ด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ดังกล่าวได้

การจำแนกประเภทผลสำเนาโค้ด จากงานวิจัยในอดีตได้ทำการจำแนก สำเนาโค้ดออกมาเป็นทั้งสามประเภทและยังมีสำเนาโค้ดอีกส่วนหนึ่งที่ไม่สามารถระบุได้ว่าเป็นสำเนาโค้ดประเภทใด อันเนื่องมาจากข้อจำกัดเชิงพฤติกรรมดังนั้น งานวิจัยนี้จึงได้นำเอาสำเนาโค้ดประเภทดังกล่าวทั้งหมดมาเป็นข้อมูลนำเข้าเพื่อตรวจสอบหาสำเนาโค้ดประเภทที่ 4

การสร้างสินทรัพย์ร่วมของทั้งสองผลิตภัณฑ์จากสำเนาโค้ดประเภทที่ 4 นั้นเพื่อให้ลดระยะเวลาในการตรวจสอบจะต้องมีการตรวจสอบเชิงโครงสร้างเพื่อลดขั้นตอนในการตรวจสอบเชิงพฤติกรรม ดังนั้นการตรวจสอบเชิงโครงสร้างจะต้องนำเอาเมท็อดทั้งหมดที่ไม่สามารถระบุประเภทสำเนาโค้ดได้มาทำการตรวจแปลงเมท็อดให้อยู่ในรูปแบบของกราฟก่อนแล้วทำการย่อกราฟให้มีขนาดเล็กลงเพื่อง่ายต่อการตรวจสอบหาคู่ที่มีโครงสร้างของการทำงานที่เหมือนกัน เมื่อจบขั้นตอนนี้เราจะได้คู่เมท็อดที่มีโครงสร้างการทำงานที่เหมือนกันเพื่อที่จะนำไปพิสูจน์ในส่วนของการทำงานเชิงพฤติกรรมต่อไป

การตรวจสอบลักษณะเชิงพฤติกรรมของสำเนาโค้ดประเภทที่ 4 นั้น จะต้องนำคู่เมท็อดที่มีลักษณะของโครงสร้างที่เหมือนกันมาทำการจำแนกเส้นทางที่เป็นไปได้ทั้งหมด เพื่อดูพฤติกรรมของการทำงาน เมื่อทำการจำแนกออกมาแล้วจับคู่เส้นทางได้กันได้พอดี ก็จะทำให้พบได้ว่าเมท็อดทั้งสองเป็นสำเนาโค้ดประเภทที่ 4 ที่มีลักษณะของการเขียนที่แตกต่างกันแต่มีพฤติกรรมของการทำงานที่เหมือนกัน

ในขั้นตอนของการก่อแบบตัวเลือกสินทรัพย์ร่วมของทั้งสองผลิตภัณฑ์ซึ่งเป็นขั้นตอนสุดท้ายด้วยแนวทางการออกแบบเมท็อดแม่แบบ คลาสแม่จะถูกสร้างเก็บไว้เพื่อเก็บการทำงานที่เหมือนกัน ส่วนคลาสลูกจะเก็บในส่วนของการทำงานที่แตกต่างกัน โดยที่จะสร้างขั้นจากการนำคู่เส้นทางที่มีการทำงานที่เหมือนกันมาแยกจุดแตกต่างออกจากกัน แล้วกลับไปรวมกันในรูปแบบของกราฟ ซึ่งกราฟที่ถูกสร้างขึ้นใหม่จะกลายเป็นคลาสแม่ซึ่งโอเวอร์ไรด์ด้วยคลาสลูกที่เป็นจุดที่มีความแตกต่างที่ถูกแยกออกมาในขั้นตอนก่อนหน้านี้ ซึ่งเมท็อดแม่แบบนี้จะสามารถนำไปใช้และปรับแต่งกับผลิตภัณฑ์ที่จะเกิดขึ้นในอนาคตที่มีขอบเขตการทำงานที่คล้ายกันกับผลิตภัณฑ์ร่วมดังกล่าว

เนื่องจากการหาซอร์สโค้ดของผลิตภัณฑ์ที่มีสำเนาโค้ดประเภทที่ 4 ปะปนอยู่นั้นเป็นไปได้ยาก รวมถึงผลิตภัณฑ์โดยส่วนใหญ่ที่เป็นประเภทของซอฟต์แวร์ที่มีลักษณะของการทำงานที่คล้ายกันมักจะไม่มีการเปิดเผยข้อมูล งานวิจัยนี้จึงได้ใช้ซอฟต์แวร์ออกเป็นสองแบบในการทดลองคือ ประเภทของซอฟต์แวร์ในงานวิจัยในอดีตที่ใช้ในลักษณะของเวอร์ชันซึ่งมีความคล้ายกัน แต่อย่างไรก็ตามพบว่าซอฟต์แวร์ในลักษณะนี้พบสำเนาโค้ดประเภทที่ 4 น้อยมากหรือแทบจะไม่มีเลย เนื่องจากการเปลี่ยนแปลงของเวอร์ชันโดยส่วนใหญ่จะมีแค่การเพิ่มคำสั่งและเพิ่มฟีเจอร์ขึ้นมาใหม่ ไม่มีการเปลี่ยนแปลงทั้งเมท็อดและอยู่ในรูปของการทำงานเดิม ดังนั้นก็จะพบเพียงแค่สำเนาโค้ดประเภทที่ 3 เพียงเท่านั้น ประเภทของซอฟต์แวร์ในรูปแบบต่อมาคือ ซอฟต์แวร์ที่ถูกสร้างขึ้นโดยนักศึกษาที่มาจากภายใต้ขอบเขตของการทำงานเดียวกัน ซึ่งเสมือนกับลักษณะของผลิตภัณฑ์ประเภทเดียวกันแต่ต่างผู้พัฒนามากที่สุด ซึ่งเลือกซอฟต์แวร์ที่มีขนาดกลางที่มีขนาด 10,000-50,000 บรรทัดโค้ด

จากผลการทดลองดังกล่าวพบว่าการระบุตัวแทนสินทรัพย์ทั่วไปในซอร์สโค้ด ด้วยการเปรียบเทียบเส้นทางของการทดสอบซอฟต์แวร์ สามารถแยกสำเนาโค้ดประเภทที่ 4 ได้และนำไปสร้างซึ่งสินทรัพย์ร่วมที่อยู่ในรูปแบบเมท็อดแม่แบบนี้จะสามารถนำไปใช้และปรับแต่งกับผลิตภัณฑ์ที่จะเกิดขึ้นในอนาคต อย่างไรก็ตาม การนำไปใช้งานจริงยังต้องมีการปรับแต่งข้อจำกัดเพื่อให้ใช้กับ

ซอฟต์แวร์ที่มีขนาดใหญ่ให้มีความรวดเร็วในการตรวจสอบ รวมถึงการเพิ่มความยืดหยุ่นในการวิเคราะห์ในแต่ละจุดให้สามารถแจงส่วนและลดข้อจำกัดในการวิเคราะห์นั้นๆให้ละเอียดอย่างยิ่งขึ้นเพื่อความชัดเจนและถูกต้องมากยิ่งขึ้น

เมื่อทำการนำผลการทดลองที่ได้ไปประเมินผลความถูกต้องพบว่าผลลัพธ์ของการประเมินผลการนำกรณีทดสอบของเมท็อดเดิมของทั้งสองผลิตภัณฑ์ไปประมวลผลกับสินทรัพย์ของผลิตภัณฑ์ทั้ง 6 ระบบ พบว่า ค่า ความถูกต้องของ สินทรัพย์ร่วม 2 ผลิตภัณฑ์ กับ กรณีทดสอบตั้งต้น ทั้ง 6 ระบบ คือ 1 ซึ่งเป็นค่าที่สูงมาก จึงสามารถนำเมท็อดแม่แบบนี้จะสามารถนำไปใช้และปรับแต่งกับผลิตภัณฑ์ที่จะเกิดขึ้นในอนาคตที่มีขอบเขตการทำงานที่คล้ายกันกับผลิตภัณฑ์ร่วมดังกล่าว

6.2 ข้อจำกัด

ข้อจำกัดของงานวิจัยนี้ประกอบด้วย

1) โค้ดผลิตภัณฑ์ที่นำมาเปรียบเทียบ ใช้ได้เฉพาะในภาษาจาวาเท่านั้น เนื่องจากภายใต้ข้อจำกัดที่มีเป็นจำนวนมากของจาวาและไม่ครอบคลุมถึงภาษาอื่นๆ

2) งานวิจัยนี้มุ่งเน้นการจัดการสำเนาโค้ดประเภทที่ 4 ซึ่งจะครอบคลุมไปยังสำเนาโค้ดประเภทที่ 3 ส่วนสำเนาโค้ดประเภทที่ 2 นั้นบางส่วนอาจจะต้องมีการวิเคราะห์ว่าการเปลี่ยนชนิดและชื่อของตัวแปรนั้นจะอ้างอิงคนละตัวหรือไม่ แต่ภาพรวมของการตรวจสอบยังสามารถตรวจสอบได้

3) ไม่รองรับการทำงานที่เปลี่ยนแปลงการทำงานตลอดเส้นทาง ซึ่งจะไม่ระบุเป็นสำเนาโค้ดประเภทที่ 4 ซึ่งการเปลี่ยนแปลงเชิงการเขียนโค้ดจะต้องไม่ส่งผลกระทบต่อพฤติกรรม

4) ไม่รับรองเป็นสำเนาโค้ดประเภทที่ 4 ถ้าเมื่อมีการจำแนกเส้นทางออกมาแล้วมีจำนวนเส้นทางที่ไม่เท่ากัน ถือว่าเป็นการเพิ่มคำสั่งให้มีความแตกต่างกันเชิงโครงสร้าง

5) การหาซอร์สโค้ดของผลิตภัณฑ์ที่มีสำเนาโค้ดประเภทที่ 4 ปะปนอยู่นั้นเป็นไปได้ยาก รวมถึงผลิตภัณฑ์โดยส่วนใหญ่ที่เป็นประเภทของซอฟต์แวร์ที่มีลักษณะของการทำงานที่คล้ายกันมักจะไม่มีการเปิดเผยข้อมูล

6.3 งานวิจัยในอนาคต

สามารถแบ่งการทำงานวิจัยในอนาคตได้ดังนี้

1) การระบุตัวเลือกสินทรัพย์ร่วมให้สามารถทำในระดับที่ใหญ่ขึ้น ซึ่งอาจจะรวมไปถึงการพิจารณาของข้อมูลในส่วนของการออกแบบมาเปรียบเทียบให้ก่อประสิทธิภาพเพิ่มขึ้น

2) การระบุตัวแทนที่มากกว่าสำเนาโค้ดประเภทที่ 4 อาจมีการสกัดสำเนาโค้ดที่เป็นส่วนหนึ่งของเมท็อดที่มีขนาดใหญ่หรือ การวิเคราะห์ในส่วนของการเพิ่มขึ้นของเส้นทางการทดสอบที่ยังเป็นข้อจำกับในปัจจุบันที่ทำให้ไม่สามารถตรวจสอบได้ เพื่อแสดงให้เห็นถึงความเป็นอิสระของผลิตภัณฑ์มากยิ่งขึ้น

3) การรวมในส่วนขั้นตอนของการเปรียบเทียบเชิงโครงสร้างและการเปรียบเทียบของเชิงพฤติกรรมเข้าด้วยกันเพื่อลดระยะเวลาในการวิเคราะห์และสามารถผนวกรวมเครื่องมือให้อยู่ในชิ้นเดียวกันทำให้สามารถประหยัดเวลาในการวิเคราะห์ และลดบางขั้นตอนให้ทำด้วยเครื่องมือแทนการใช้แรงงานคนในการตรวจสอบและวิเคราะห์

4) ทำการเปรียบเทียบระหว่างผลิตภัณฑ์มากกว่า 2 ผลิตภัณฑ์ขึ้นไป จุดแตกต่างอาจจะมีมากขึ้นจนทำให้การวิเคราะห์ความแตกต่างมากขึ้นแต่จะได้ในส่วนของสินทรัพย์หลักที่มีความสมบูรณ์มากที่สุด และตัดจุดแตกต่างออกไปอย่างมีประสิทธิภาพ

รายการอ้างอิง

1. Northrop, P.C.a.L., *Software product lines: practices and patterns.* 2001.
2. T. Mende, F.B., R. Koschke, and G. Meier, *Supporting the Grow-and-Prune Model in Software Product Lines Evolution Using Clone Detection.* Proceedings of the 2008 12th European Conference on Software Maintenance and Reengineering, IEEE Computer Society, 2008: p. 163-172.
3. Cordy, C.K.R.a.J.R., *A Survey on Software Clone Detection Research.* SCHOOL OF COMPUTING TR 2007-541, QUEEN'S UNIVERSITY, 2007. **115**.
4. Miryung Kim, D.N., *Program Element Matching for Multi-Version Program Analyses.* MSR'06, 2006.
5. Koschke, R., *Survey of Research on Software Clones, Duplication, Redundancy, and Similarity in Software.* Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), 2007.
6. S. Bellon, R.K., G. Antoniol, J. Krinke, and E. Merlo, *Comparison and Evaluation of Clone Detection Tools.* ," IEEE Transactions on Software Engineering, 2007. **33**(591, 577).
7. Reantragoon, T., *Identifying Common Asset Candidates in Software Product Line by Code Clone Detection* 4th International Conference on Computer Research and Development, 2012.
8. L. A. Clarke, A.P., D. J. Richardson, and S. J. Zeil, *A Formal Evaluation of Data Flow Path Selection Criteria.* IEEE Trans. Softw. Eng. 15, 1989: p. 1318-1332.
9. Yan Wang, Z.B., Miao Zhang, Wen Du, Ying Qin, and Xiyang Liu, *Fitness calculation approach for the switch-case construct in evolutionary testing.* Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO '08), 2008: p. 1767-1774.
10. Cordy, C.K.R.a.J.R., *NICAD: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization.* Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on, 2008, 2008: p. 172–181.

11. Gold, R., *Control flow graphs and code coverage.* Int. J. Appl. Math. Comput. Sci., 2010: p. 739-749.
12. Ullman, S.H.a.J.D., *Control flowgraphs and code coverage by ROBERT GOLD*

Flow graph reducibility by Matthew Software verification and validation by Lionel Briand, 2008.

13. Vijayanand Nagarajan, R.G., Matias Madou, Xiangyu Zhang, Bjorn De Sutter, *Matching Control Flow of Program Versions.* ICSM 2007. **84-93**.
14. C.Jorgensen, P., *Software testing a craftsman approach.* Boca Raton New York. Auerbach Publications, 2007. **3**.
15. T. Apiwattanapong, A.O., and M. J. Harrold, *A differencing algorithm for object-oriented programs.* IEEE Computer Society, 2004: p. 213.
16. Ginger Myles, C.C., *Detecting Software Theft via Whole Program Path Birthmarks.* 2004.
17. Asadullah, A., *Design Patterns Based Pre-processing of Source Code for Plagiarism Detection.* Software Engineering Conference (APSEC), 2012
18. T. Apiwattanapong, A.O., and M. J. Harrold, *Effcient and precise dynamic impact analysis using execute-after sequences.* CSE, 2005: p. 432.
19. A. Orso, N.S., and M. J. Harrold, *Scaling regression testing to large software systems.* SIGSOFT '04/FSE-12, 2004.
20. Szermer, J.L.a.W., *Identification of program modifications and its applications in software maintenance.* ICSM, 1992.
21. Gupta, X.Z.a.R., *Matching execution histories of program versions.* SIGSOFT Softw. Eng. Notes 30, 2005: p. 197-206.
22. Horwitz, R.K.a.S., *Tool demonstration: Finding duplicatedcode using program dependences.* Proceedings of the European Symposium on Pro-gramming (ESOP'01), 2001: p. 383-386.
23. S. Bellon, R.K., G. Antoniol, J. Krinke, and E. Merlo, *Comparison and Evaluation of Clone Detection Tools.* IEEE Transactions on Software Engineering, 2007: p. 591, 577.

ภาคผนวก

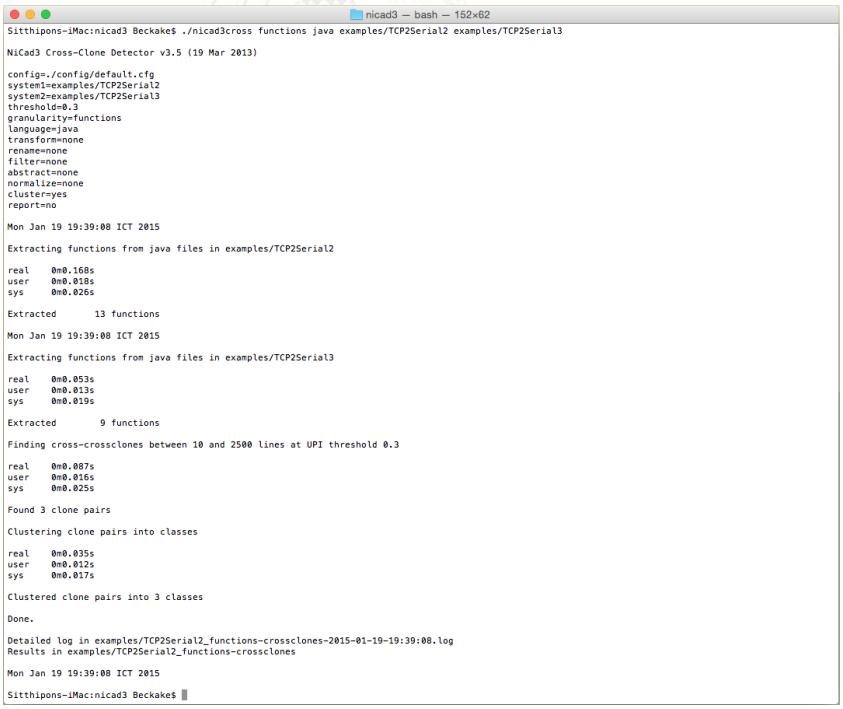
ภาคผนวก ก

การทำงานและส่วนต่อประสานผู้ใช้งาน

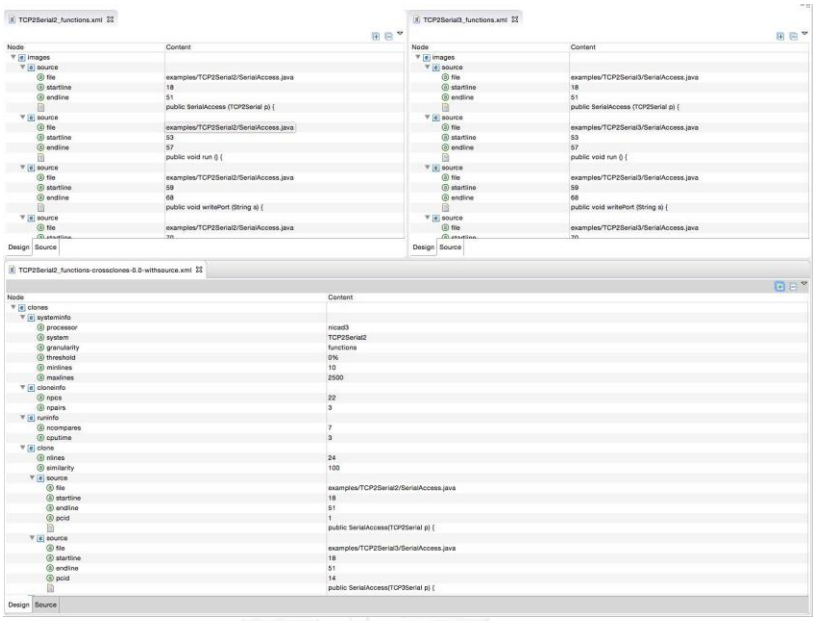
เครื่องมือของงานวิจัยนี้ได้พัฒนาบนวินโดว์แอปพลิเคชัน ซึ่งการทำงานหลักๆ ประกอบด้วย 3 ส่วนคือ ส่วนการตรวจหาสำเนาเชิงโครงสร้างและการตรวจสอบเชิงพฤติกรรม และการก่อสินทรัพย์ร่วม โดยสามารถแสดงรายละเอียดและหน้าจอส่วนต่อประสานผู้ใช้ได้ดังนี้

ก.1 ส่วนการตรวจหาสำเนาเชิงโครงสร้าง

ส่วนการตรวจหาสำเนาเชิงโครงสร้างจะมีส่วนต่อประสานของหน้าจอโดยเริ่มผู้ใช้จะต้องทำการนำโค้ดของทั้งสองผลิตภัณฑ์มาเป็นข้อมูลนำเข้าของเครื่องมือจำแนกสำเนาโค้ดภาษาจาวา ดังรูปที่ ก.1 หลังจากนั้นจะได้เป็นข้อมูลนำออกอยู่ในรูปของเมท็อดต่างๆ คือ เมท็อดที่เป็นประเภท 1,2,3 ของทั้งสองผลิตภัณฑ์ เมท็อดผลิตภัณฑ์ที่ 1 เมท็อดผลิตภัณฑ์ที่ 2 ดังรูปที่ ก.2

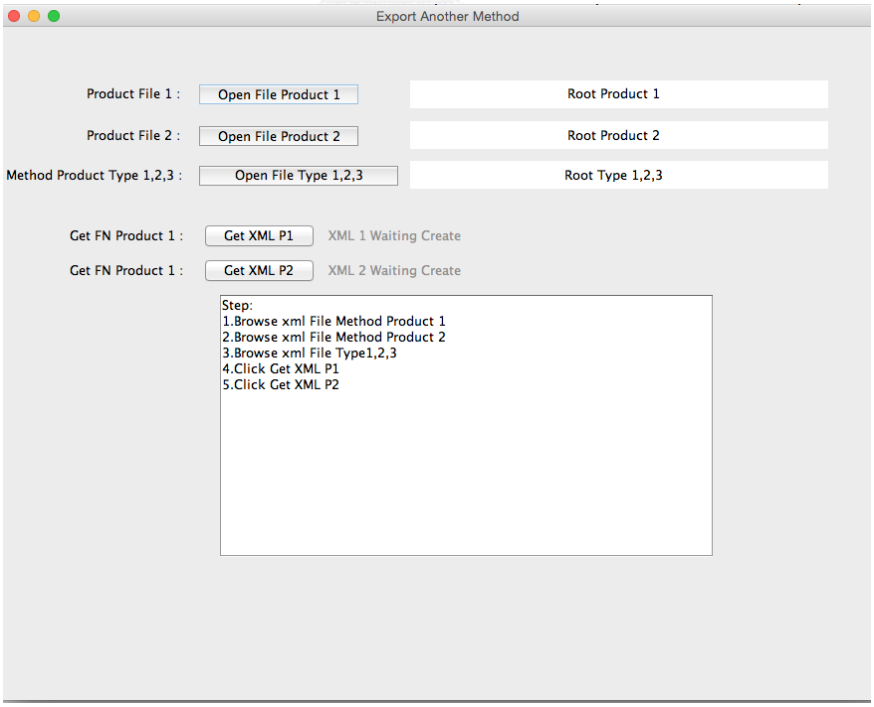


รูปที่ ก.1 การนำโค้ดของทั้งสองผลิตภัณฑ์มาเป็นข้อมูลนำเข้าของเครื่องมือจำแนกสำเนาโค้ดภาษาจาวา

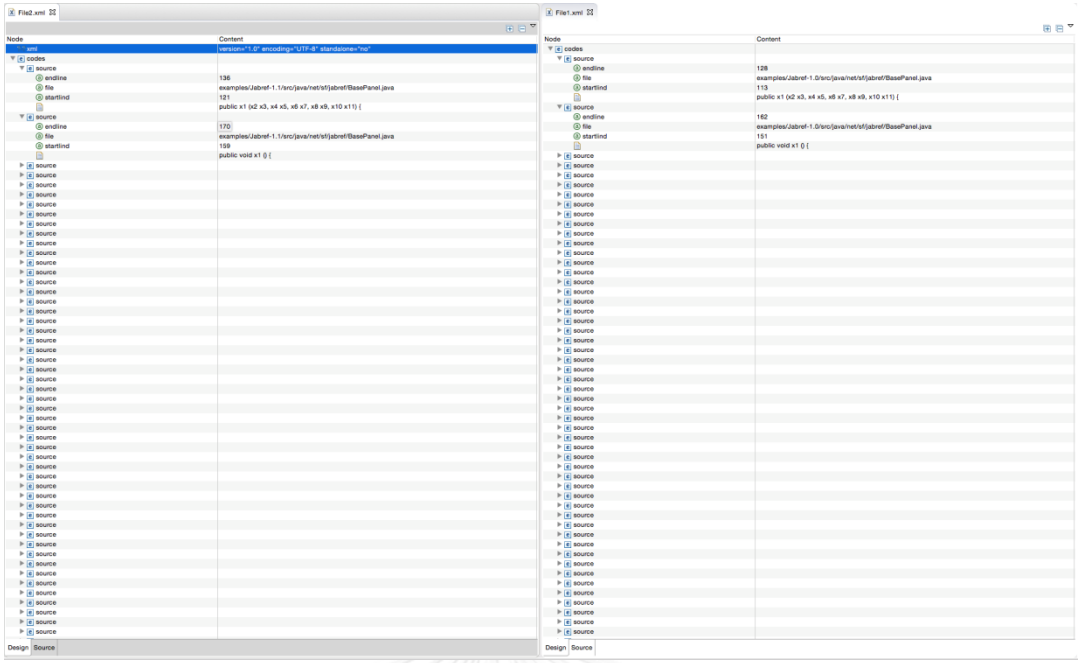


รูปที่ ก.2 ข้อมูลนำออกอยู่ในรูปของเมท็อดต่างๆ

ขั้นตอนต่อมาจะนำเอาข้อมูลที่ได้จากขั้นตอนที่แล้วมาเป็นข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Export Another method เพื่อทำการสกัดเอาเมท็อดที่เหลืออยู่จากประเภทที่ 1,2,3 ออกมา ดังรูปที่ ก.3 และจะได้ผลลัพธ์ออกมาเป็นเมท็อดที่เหลืออยู่ของทั้งสองผลิตภัณฑ์ ดังรูปที่ ก.4

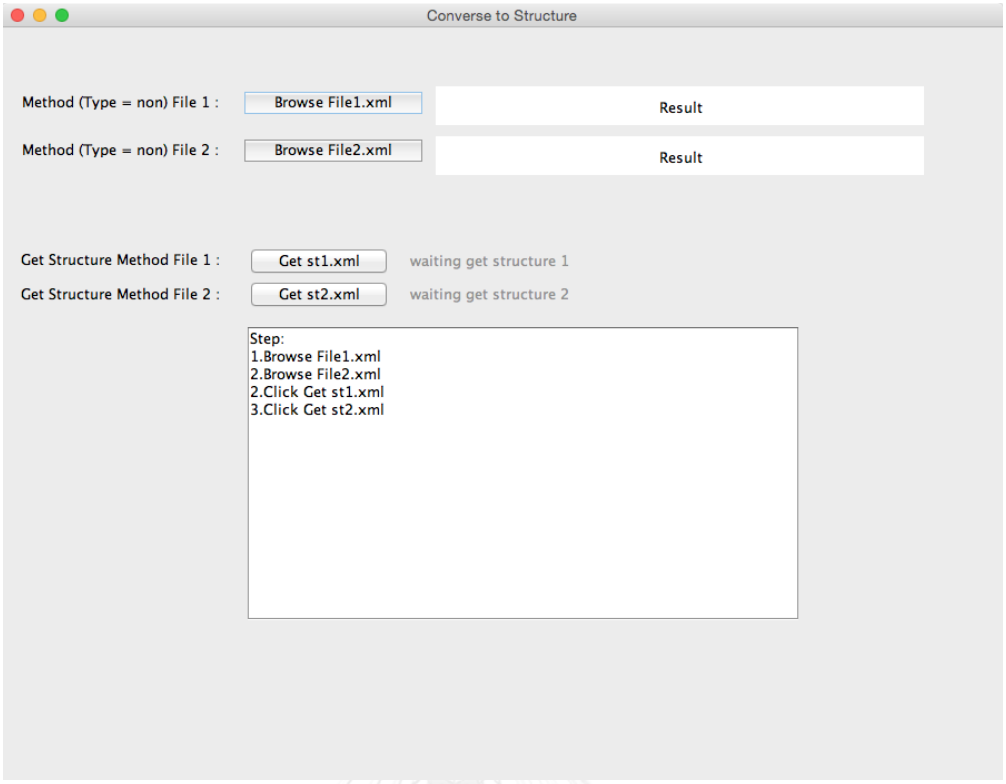


รูปที่ ก.3 ข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Export Another method เพื่อทำการสกัดเอาเมท็อดที่เหลืออยู่

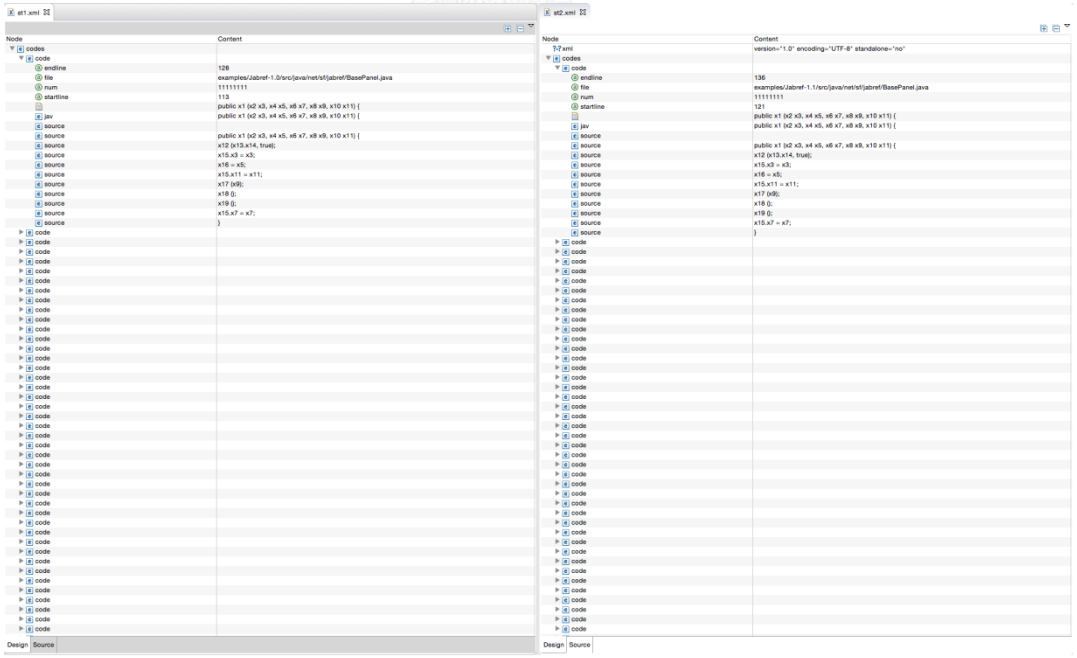


รูปที่ ก.4 ผลลัพธ์ออกมาเป็นเมท็อดที่เหลืออยู่ของทั้งสองผลิตภัณฑ์

ถัดมาจะนำเอาข้อมูลที่ได้จากขั้นตอนที่แล้วมาเป็นข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Comverse to Structure เพื่อทำการแปลงเส้นทางการทำงานเพื่อหาเมท็อดที่มีโครงสร้างที่เหมือนกัน ดังรูปที่ ก.5 และจะได้ผลลัพธ์ออกมาเป็นคู่เมท็อดที่แปลงเส้นทางการทำงานของทั้งสองผลิตภัณฑ์ ดังรูปที่ ก.6

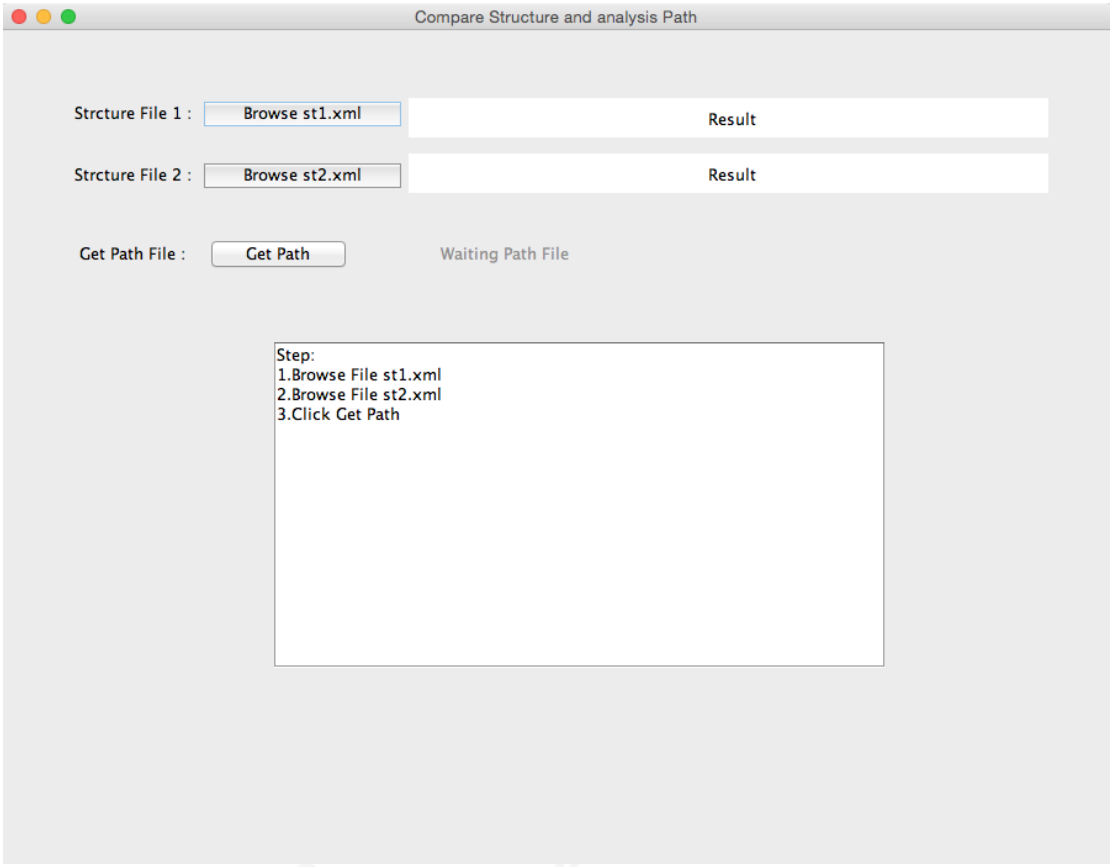


รูปที่ ก.5 ข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Comverse to Structure เพื่อทำการแปลงเส้นทางการทำงานเพื่อหาเมท็อดที่มีโครงสร้างที่เหมือนกัน

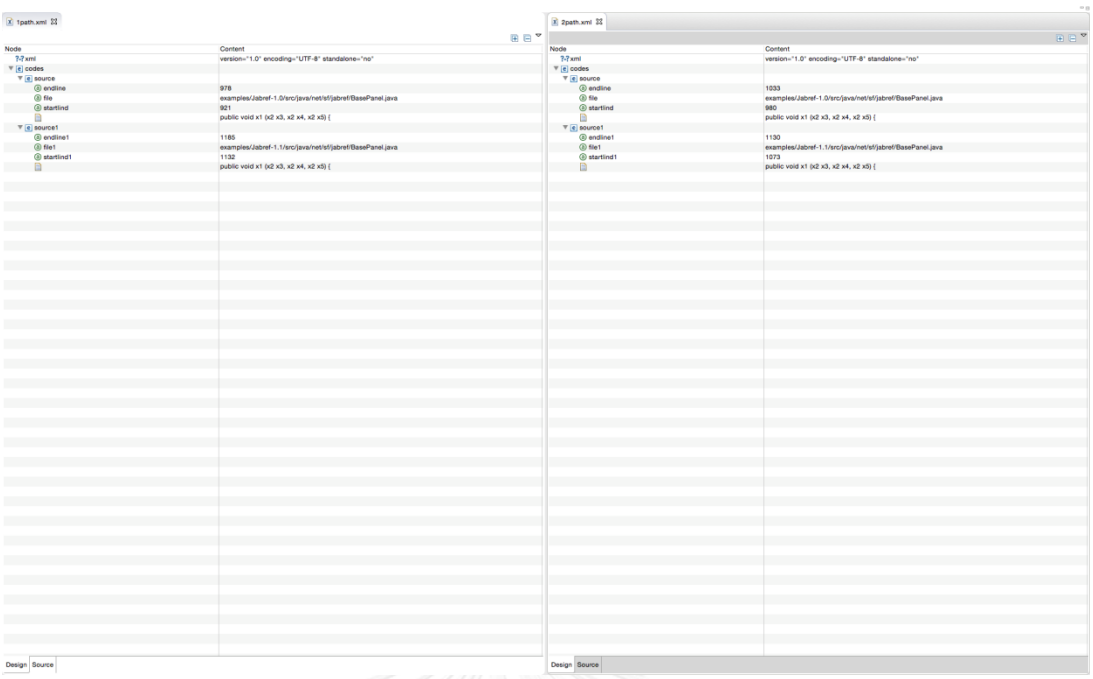


รูปที่ ก.6 ผลลัพธ์ออกมาเป็นคู่เมท็อดที่มีการแปลงเส้นทางการทำงานเพื่อหาเมท็อดที่มีโครงสร้างที่เหมือนกัน

ถัดมาจะนำเอาข้อมูลที่ได้จากขั้นตอนที่แล้วมาเป็นข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Compare Structure and analysis Path เพื่อทำการเปรียบเทียบเส้นทางการทำงานเพื่อหาเมท็อดที่มีโครงสร้างที่เหมือนกัน ดังรูปที่ ก.7 และจะได้ผลลัพธ์ออกมาเป็นคู่เมท็อดที่มีโครงสร้างที่เหมือนกันของทั้งสองผลิตภัณฑ์ ดังรูปที่ ก.8



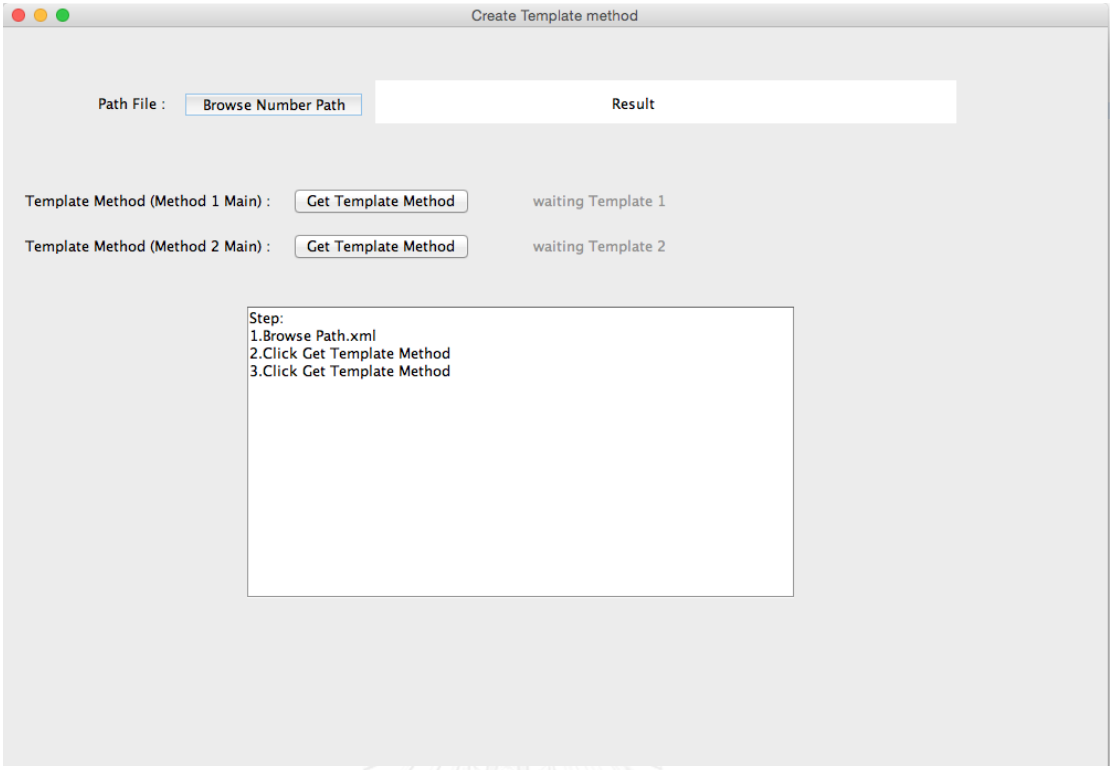
รูปที่ ก.7 การนำคู่เมท็อดที่มีโครงสร้างที่เหมือนกันของทั้งสองผลิตภัณฑ์มาเป็นข้อมูลนำเข้าของเครื่องมือจำแนกเส้นทาง



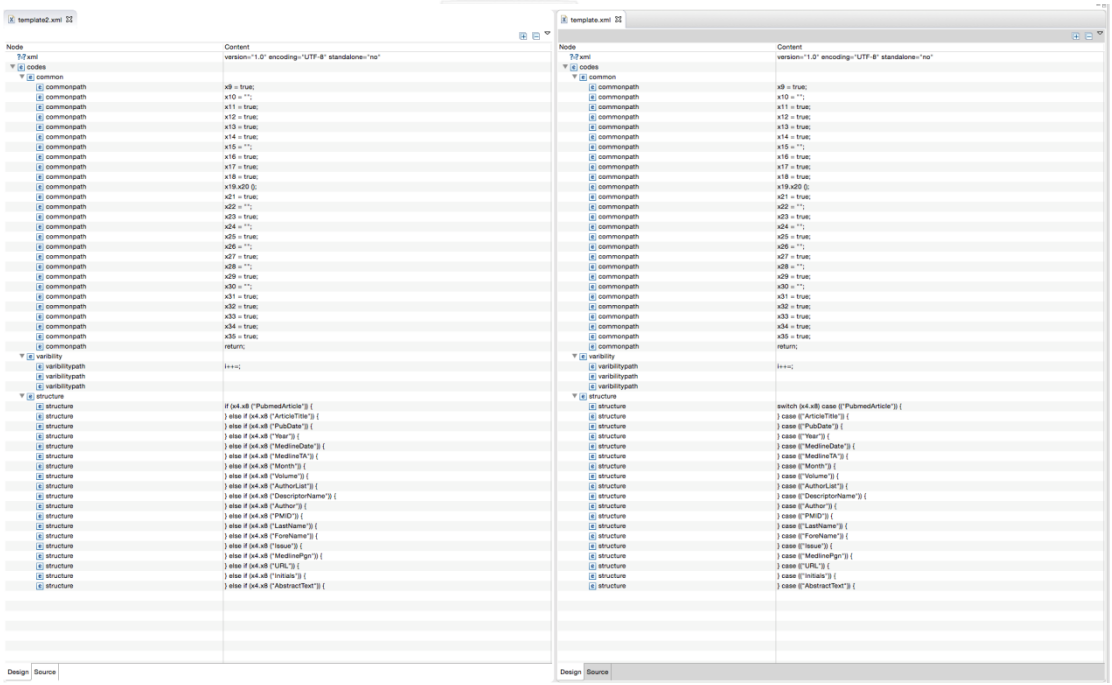
รูปที่ ก.8 ข้อมูลนำออกอยู่ในโครงสร้างที่เหมือนกันของทั้งสองผลิตภัณฑ์มาเป็นข้อมูลนำเข้าของเครื่องมือจำแนกเส้นทาง

ก.2 ส่วนการตรวจสอบเชิงพฤติกรรม

ส่วนการตรวจสอบเชิงพฤติกรรมจะมีส่วนต่อประสานของหน้าจอโดยเริ่มผู้ใช้จะต้องทำการนำคู่เมท็อดที่มีโครงสร้างที่เหมือนกันของทั้งสองผลิตภัณฑ์มาเป็นข้อมูลนำเข้าของเครื่องมือจำแนกเส้นทาง เพื่อทำการจับคู่เส้นทางที่มีพฤติกรรมที่เหมือนกัน แล้วสร้างเป็น Template method ดังรูปที่ ก.9 และจะได้ผลลัพธ์ออกมาเป็น Template method ดังรูปที่ ก.10



รูปที่ ก.9 ข้อมูลนำเข้าในเครื่องมือ ICT4 ในส่วนของเมนู Create Template method เพื่อทำการจับคู่เส้นทางที่มีพฤติกรรมที่เหมือนกัน แล้วสร้างเป็น Template method



รูปที่ ก.10 ผลลัพธ์ออกมาเป็นคู่เส้นทางที่มีการจับคู่เส้นทางที่มีพฤติกรรมที่เหมือนกัน แล้วสร้างเป็น Template method

ประวัติผู้เขียนวิทยานิพนธ์

นายสิทธิพล ลิ้มชัยชะดา เกิดเมื่อวันที่ 23 กรกฎาคม พ.ศ. 2531 ที่โรงพยาบาลมหาราช อำเภอเมือง จังหวัดนครศรีธรรมราช โดยมีภูมิลำเนาอยู่จังหวัดนครศรีธรรมราช สำเร็จการศึกษาระดับประถมศึกษาจากโรงเรียนเทศบาลวัดใหญ่ อำเภอเมือง จังหวัดนครศรีธรรมราช สำเร็จการศึกษาระดับมัธยมศึกษาตอนต้นจากโรงเรียนสาธิตเทศบาลวัดเพชรจริก อำเภอเมือง จังหวัดนครศรีธรรมราช สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนสาธิตเทศบาลวัดเพชรจริก อำเภอเมือง จังหวัดนครศรีธรรมราช สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาการพัฒนาซอฟต์แวร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย จังหวัดกรุงเทพมหานคร ในปีการศึกษา 2553 และเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย จังหวัดกรุงเทพมหานคร ในปี 2554