การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

น.ส.พจนารถ จันทน์วัฒนวงษ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

Software Remodularization using Tabu Search

Miss Photjanat Janvattanavong

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม
โดย	น.ส.พจนารถ จันทน์วัฒนวงษ์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

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

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

..... กรรมการ
(ศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.บัณฑิต ฐานะโสภณ)

พจนารถ จันทน์วัฒนวงษ์ : การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม. (Software Remodularization using Tabu Search) อ.ที่ปรึกษาหลัก : รศ. ดร.พรศิริ หมื่นไชยศรี

วิทยานิพนธ์นี้นำเสนอการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม เพื่อค้นหารูปแบบการจัดสรรคลาสไปยังแพ็กเกจที่เหมาะสมที่สุด ซึ่งเป็นการปรับปรุงสภาพมอดุลาร์ของซอฟต์แวร์ โดยทำการค้นหาคลาสที่ไม่เหมาะสมกับแพ็กเกจ เพื่อย้ายไปยังแพ็กเกจที่เหมาะสมมากขึ้น

การรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้ามนี้ ประกอบไปด้วยขั้นตอนการตรวจสอบว่า ซอฟต์แวร์นั้นมีความจำเป็นต้องทำการรีมอดุลาไรเซชันหรือไม่ ด้วยเกณฑ์ค่าสัมประสิทธิ์ซิลูเอทของระบบซอฟต์แวร์ และหากซอฟต์แวร์จำเป็นต้องรีมอดุลาไรเซชัน ขั้นตอนถัดไปคือทำการค้นหาด้วยการค้นหาต้องห้ามจะค้นหารูปแบบการจัดสรรคลาสที่เหมาะสม เพื่อปรับปรุงระบบซอฟต์แวร์ให้มีคุณภาพที่ดีขึ้น ซึ่งวิธีการนี้สามารถเป็นเครื่องมือช่วยวิศวกรซอฟต์แวร์ตัดสินใจในการทำรีมอดุลาไรเซชันได้

เพื่อสนับสนุนวิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม จึงได้พัฒนาเครื่องมือเพื่อทดสอบกับซอฟต์แวร์และกรณีตัวอย่างที่พัฒนาด้วยภาษาจาวารวมเจ็ดตัวอย่าง จากการทดสอบพบว่า เครื่องมือสามารถตรวจสอบได้ว่าซอฟต์แวร์ควรมีการรีมอดุลาไรเซชันหรือไม่ และเครื่องมือสามารถค้นหาและแนะนำวิธีมูฟคลาสรีแฟคทอริงเพื่อทำให้ระบบซอฟต์แวร์มีคุณภาพที่ดีขึ้น โดยประเมินจากการใช้ตัววัดสัมประสิทธิ์ซิลูเอทและเทอร์โบเอ็มคิว เมื่อเปรียบเทียบก่อนและหลังการรีมอดุลาไรเซชันแล้วพบว่ามีค่าเพิ่มขึ้น

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2563 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

5970449921 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Software Remodularization, Tabu Search, Silhouette Coefficient

Photjanat Janvattanavong : Software Remodularization using Tabu Search. Advisor: Assoc. Prof. PORNSIRI MUENCHAISRI, Ph.D.

This thesis proposes an approach to remodularize software using Tabu search. The aim is to find an optimal solution to improve software modularity by reorganizing software structure based on class relationships of all packages. An unsuitable class in current package should be moved into more suitable package to improve the software modularity.

Software remodularization using Tabu search is divided into two steps. Firstly, measuring silhouette coefficient to determine if the software needs remodularization or not. Secondly, if remodularization is needed, some classes should be moved to new packages. Tabu search is used to search for the best solution to increase the modularity of software. This approach can suggest software engineer to decide on software remodularization.

To support that software remodularization using Tabu search can be implemented, a tool is developed to support the study. The tool is evaluated with seven java software examples using the Silhouette coefficient and the Turbo MQ measurement , when compares the results from the two metric before and after remodularization, the results show that the modularity of software is improved.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2020 Advisor's Signature .....

กิตติกรรมประกาศ

ในการเรียนระดับบัณฑิตศึกษาของข้าพเจ้าสำเร็จได้ด้วยการสนับสนุนจากผู้คนรอบตัว พ่อแม่ ครูอาจารย์ที่ให้สติ ให้ปัญญา ครอบครัว คณาจารย์ เจ้าหน้าที่ภาควิชาทุกๆท่าน กัลยาณมิตร และน้ำใจจากทุกๆคน

กราบขอบพระคุณ รศ.ดร. พรศิริ หมื่นไชยศรี อาจารย์ที่ปรึกษา ที่เสียสละเวลาให้คำแนะนำ ความรู้ ความเมตตาและปรารถนาดี คอยกระตุ้นเตือนให้ข้าพเจ้ายังมีชีวิตการเรียนที่ราบรื่น นับแต่วันแรกที่สอบสัมภาษณ์จนถึงวันสุดท้ายของการเรียนอาจารย์เป็นผู้ให้โอกาส สนับสนุนและชี้เส้นทางที่ดีให้เกิดแก่ข้าพเจ้า

ขอบพระคุณ รศ.ดร.ธาราทิพย์ สุวรรณศาสตร์ ศ.ดร.บุญเสริม กิจศิริกุล รศ.ดร.วิวัฒน์ วัฒนาวุฒิ และ ผศ.ดร.บัณฑิต ฐานะโสภณ ที่กรุณามาเป็นคณะกรรมการในการสอบ ให้คำแนะนำในการพัฒนาวิทยานิพนธ์และตัวเอง และท่านอาจารย์ทุกท่านในภาควิชาที่ให้วิชาความรู้ เอาใจช่วย ให้กำลังใจอยู่เสมอ นับเป็นความโชคดีของข้าพเจ้าที่ได้รับมาตลอดช่วงเวลาที่อยู่ที่แห่งนี้

ขอบคุณกัลยาณมิตร ความมีน้ำใจของทุกๆคน ที่ให้ทั้งกำลังใจ ความช่วยเหลือทุกอย่าง ข้าพเจ้าซาบซึ้งใจมาก ขอบคุณเพื่อนๆพี่ๆในแลบทุกๆคนที่ผลัดกันให้กำลังใจกันตลอดมา ขอบคุณเพื่อนๆทุกคนยังคอยถามไถ่ ส่งเสียงให้กำลังใจมาตลอด

สุดท้ายความสำเร็จนี้ของข้าพเจ้ามาจากการสนับสนุนที่ดีของครอบครัว กราบขอบพระคุณคุณย่า พ่อแม่ พี่เมและทุกๆคนที่บ้าน ที่เป็นที่พึ่งพิงที่แข็งแรงมั่นคง และเฝ้าคอยความสำเร็จของข้าพเจ้าอย่างอดทน ยังคงเชื่อมั่นในตัวข้าพเจ้าเสมอแม้ในวันที่ข้าพเจ้าไม่เหลือความเชื่อมั่นนั้นแล้วก็ตาม ขอบคุณที่ยังพร้อมเข้าใจและสนับสนุนข้าพเจ้าอยู่เสมอ

พจนารถ จันทน์วัฒนวงษ์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

การเปลี่ยนแปลงของซอฟต์แวร์ (Software changes) เป็นสิ่งที่เกิดขึ้นในวัฏจักรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle) อย่างหลีกเลี่ยงไม่ได้ โดยเกิดขึ้นได้จากหลายสาเหตุ อาทิ การเปลี่ยนแปลงความต้องการ (Requirement changes) การปรับปรุงแก้ไข การเพิ่มเติมการทำงานบางอย่าง หรือการเกิดข้อผิดพลาด (Error) ซึ่งการแก้ไขปัญหาเหล่านี้ทำให้ซอฟต์แวร์ถูกเปลี่ยนแปลง ซึ่งกระทบต่อโครงสร้างและสภาพมอดุลาร์ของซอฟต์แวร์ (Software modularity) ซึ่งสภาพมอดุลาร์ (Modularity) [1] เป็นหนึ่งในปัจจัยบ่งชี้คุณภาพของซอฟต์แวร์ ระบบที่มีสภาพมอดุลาร์ต่ำมีแนวโน้มที่จะก่อให้เกิดปัญหาในด้านต่างๆ ตามมาภายหลัง เช่น ในกระบวนการบำรุงรักษา การปรับปรุงแก้ไขจะมีความลำบากมากขึ้นทำให้สูญเสียทรัพยากรในการแก้ไข แต่อย่างไรก็ตามส่วนหนึ่งที่ซอฟต์แวร์มีสภาพมอดุลาร์ต่ำนั้นเกิดขึ้นจากพัฒนาซอฟต์แวร์ที่ไม่ได้มีการจัดองค์ประกอบที่ดี หรือเกิดการเปลี่ยนแปลงการจัดองค์ประกอบทำให้กระทบต่อสภาพมอดุลาร์ [2, 3] ของซอฟต์แวร์

การปรับปรุงซอฟต์แวร์ให้มีสภาพมอดุลาร์ที่เหมาะสม เป็นการปรับปรุงคุณภาพอย่างหนึ่งของซอฟต์แวร์ ซึ่งทำได้โดยการปรับปรุงโครงสร้างของซอฟต์แวร์ให้มีการจัดสรรองค์ประกอบที่เหมาะสม การรีมอดุลาไรเซชัน (Software Remodularization) [4] จึงถูกนำมาแก้ปัญหานี้ โดยจะจัดสรรองค์ประกอบของซอฟต์แวร์ใหม่ (Software Reorganizing) [3, 5] ให้องค์ประกอบของซอฟต์แวร์นั้นอยู่ในระบบย่อยที่เหมาะสม โดยจะพิจารณาซอฟต์แวร์จากโครงสร้างในระดับความสัมพันธ์ขององค์ประกอบภายในระบบซอฟต์แวร์

ในช่วงกว่า 20 ปีที่ผ่านมา พบว่ามีการศึกษาและงานวิจัยเกี่ยวกับการรีมอดุลาไรเซชันจำนวนมาก โดยนำวิธีการที่แตกต่างกันมาใช้เพื่อหาวิธีการที่ทำให้การทำรีมอดุลาไรเซชันมีประสิทธิภาพและเหมาะสมมากที่สุด โดยงานวิจัยจำนวนหนึ่งมุ่งเน้นไปที่การออกแบบวิธีการจัดกลุ่มขององค์ประกอบภายในระบบ ด้วยการใช้ตัววัดความสัมพันธ์ภายในของระบบ โดยการใช้เทคนิคต่างๆ เช่น การแบ่งกลุ่ม (Clustering) [6, 7] การวิเคราะห์แนวคิด (Concept analysis) [8] นอกจากนี้มีงานวิจัยจำนวนหนึ่งที่เลือกวิธีการหาค่าที่เหมาะสมที่สุด (Optimization Algorithms) ซึ่งเป็นวิธีการที่มีเทคนิคที่หลากหลาย โดยมีงานวิจัยที่เลือกใช้ เทคนิคทางพันธุกรรม (Genetic algorithm) [5, 9] การค้นหาแบบฮาร์โมนี (Harmony Search) [10] การจำลองการอบเหนียว (Simulated Annealing) [11] และ วิธีการอาณาจักรมด (Ant colony) [12] ซึ่งแต่ละวิธีสามารถช่วยให้การทำรี

มอดุลาไรเซชันนั้นประสบความสำร็จโดยที่มีซอฟต์แวร์มีคุณภาพที่ดีขึ้น ซึ่งแต่ละวิธีก็จะมีข้อจำกัดและความสามารถแตกต่างกัน จึงเป็นความท้าทายที่จะทำการศึกษาเพื่อหาวิธีการอื่นๆ ที่จะสามารถพัฒนาประสิทธิภาพในการทำรีมอดุลาไรเซชันซอฟต์แวร์ ให้มีประสิทธิภาพการทำงานมากขึ้น

วิทยานิพนธ์นี้จะทำการศึกษาและออกแบบวิธีการ การทำรีมอดุลาไรเซชันด้วยวิธีใหม่ คือ ใช้การค้นหาต้องห้าม (Tabu search) [13, 14] โดยจะพิจารณาโครงสร้าง และวิเคราะห์ความสัมพันธ์ระหว่างคลาส (Class) และแพ็กเกจ (Package) ภายในระบบซอฟต์แวร์ซึ่งมีความสัมพันธ์กัน หากคลาสใดๆ มีความสัมพันธ์ที่ไม่เหมาะสม ก็อาจจะทำให้ซอฟต์แวร์มีสภาพมอดุลาร์ที่ต่ำ การค้นหาต้องห้ามจะทำการค้นหาแพ็กเกจที่เหมาะสมให้คลาส เพื่อแนะนำการมูฟคลาสรีแฟคทอริงให้คลาสแต่ละคลาสได้อยู่ในแพ็กเกจที่เหมาะสม อันจะทำให้สภาพมอดุลาร์ของซอฟต์แวร์นั้นอยู่ในระดับที่เหมาะสมได้ จากงานวิจัยที่ผ่านมาการประเมินมอดุลาร์ไรเซชันซอฟต์แวร์จะประเมินโดยใช้ตัววัดโดยใช้การเปรียบเทียบเพื่อตัดสินใจ ซึ่งการเปรียบเทียบและตัดสินใจนี้ขึ้นอยู่กับประสบการณ์และพื้นฐานเฉพาะตัวของนักพัฒนาทำให้ผลลัพธ์ที่ได้เป็นผลที่อาจจะมีการคลาดเคลื่อน เนื่องจากประสบการณ์ของผู้พัฒนา ดังนั้นงานวิจัยนี้ได้นำเอาตัววัดคือสัมประสิทธิ์ซิลูเอท (Silhouette Coefficients) [15, 16] ซึ่งเป็นตัววัดที่มีผลลัพธ์เป็นช่วงคำตอบที่สามารถแปลความหมายได้มาใช้เพื่อประเมินและตัดสินใจในการทำรีมอดุลาไรเซชัน ทำให้สามารถตัดสินใจและประเมินผลได้ด้วยเกณฑ์เดียวกัน ไม่ขึ้นอยู่กับผู้ใช้งาน การทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามนี้จะทำการประเมินสภาพมอดุลาร์ก่อนและหลังการทำรีมอดุลาไรเซชัน ดังนั้นในงานวิจัยนี้จะเสนอวิธีการทำรีมอดุลาร์ไรเซชันด้วยการค้นหาต้องห้าม โดยที่จะสามารถช่วยให้วิศวกรซอฟต์แวร์สามารถตัดสินใจในการรีมอดุลาร์ไรเซชันเพื่อปรับปรุงคุณภาพของซอฟต์แวร์ได้

1.2 วัตถุประสงค์

1. เพื่อเสนอวิธีการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการแบบค้นหาต้องห้าม
2. เพื่อพัฒนาเครื่องมือช่วยสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการแบบค้นหาต้องห้าม

1.3 ขอบเขตการดำเนินงาน

1. วิทยานิพนธ์นี้นำเสนอการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม สำหรับซอฟต์แวร์ที่ถูกแปลงเป็นกราฟพึ่งพาแล้ว
2. ระบบซอฟต์แวร์ที่นำมาใช้จะต้องแปลงเป็นไฟล์กราฟพึ่งพาได้และไม่พิจารณาถึงลักษณะการสืบทอด

3. การพิจารณาความสัมพันธ์ของคลาสในระบบซอฟต์แวร์โดยไม่พิจารณาทิศทางความสัมพันธ์

4. ระบบซอฟต์แวร์ที่นำมาใช้ในการทดลองจะต้องประกอบด้วยระบบย่อยตั้งแต่ 3 ระบบย่อยและแต่ละระบบย่อยประกอบด้วยคลาสตั้งแต่ 3 คลาสขึ้นไป

5. การประเมินคุณภาพของการรีมอดุลาไรเซชันก่อนและหลังการรีมอดุลาไรเซชันจะประเมินจากการค่าสัมประสิทธิ์ซิลูเอทและค่าเทอร์โบเอ็มคิว

6. เครื่องมือที่พัฒนาขึ้นจะต้องสามารถนำเข้าข้อมูลกราฟพึ่งพาได้

7. เครื่องมือที่พัฒนาขึ้นจะต้องสามารถประเมินคุณภาพของการรีมอดุลาไรเซชันได้

8. เครื่องมือจะต้องสามารถค้นหาและแนะนำเพื่อการปรับปรุงการจัดระบบซอฟต์แวร์ได้

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาข้อมูลและงานวิจัยที่เกี่ยวข้องกับการทำรีมอดุลาไรเซชัน เพื่อให้เข้าใจลักษณะหลักการ และประโยชน์ของการทำรีมอดุลาไรเซชัน รวมถึงวิวัฒนาการและการพัฒนารูปแบบที่แตกต่างกันของการทำรีมอดุลาไรเซชันที่มีหลากหลายรูปแบบ

2 ศึกษางานวิจัยที่เกี่ยวข้องกับการค้นหาต้องห้าม เพื่อศึกษารูปแบบการนำไปใช้งานกระบวนการค้นหา และเงื่อนไขของการค้นหาต้องห้าม

3. ศึกษางานวิจัยที่เกี่ยวข้องกับการมูฟคลาส รีแฟคทอริง เพื่อศึกษากระบวนการรีแฟคทอริง

4. ศึกษางานวิจัยเกี่ยวข้องกับมาตรวัดการแบ่งกลุ่ม

5. ศึกษาข้อมูลเครื่องมือที่ใช้สนับสนุนในงานวิจัย และกำหนดรูปแบบลักษณะข้อมูลที่ต้องการ

6. พัฒนาวิธีการรีมอดุลาไรเซชัน โดยการค้นหาต้องห้าม กำหนดเงื่อนไขการค้นหา

7. ออกแบบและพัฒนาเครื่องมือการทำรีมอดุลาไรเซชัน

8. ทำการทดลองโดยใช้เครื่องมือ สรุปและวิเคราะห์ผลที่ได้จากเครื่องมือ

9. วิเคราะห์และประเมินผล

10. จัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

สามารถเป็นเครื่องมือและวิธีการที่ช่วยสนับสนุนการทำงานของวิศวกร เพื่อช่วยให้สามารถใช้เป็นข้อมูลในการตัดสินใจในการปรับปรุงคุณภาพของซอฟต์แวร์ และเป็นวิธีการทำรีมอดุลาไรเซชันซอฟต์แวร์เพื่อปรับปรุงคุณภาพซอฟต์แวร์ได้โดยใช้วิธีค้นหาต้องห้าม โดยที่จะสามารถแนะนำรูปแบบวิธีการจัดองค์ประกอบซอฟต์แวร์ให้มีสภาพมอดุลาร์อยู่ในเกณฑ์ที่ดีขึ้นได้

1.6 บทความวิจัยที่ได้รับการตีพิมพ์

วิทยานิพนธ์นี้มีผลงานส่วนหนึ่งได้รับการตอบรับและตีพิมพ์เป็นบทความวิชาการเรื่อง Software Remodularizations Using Tabu Search โดยนางสาวพจนารถ จันทน์วัฒนวงษ์ และรองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี ในงานประชุมวิชาการ “The 13th International Conference on Advanced Computer Theory and Engineering (ICACTE2020)” ระหว่างวันที่ 18-20 กันยายน 2563 ที่เมืองหางโจว สาธารณรัฐประชาชนจีน (นำเสนอในรูปแบบออนไลน์)

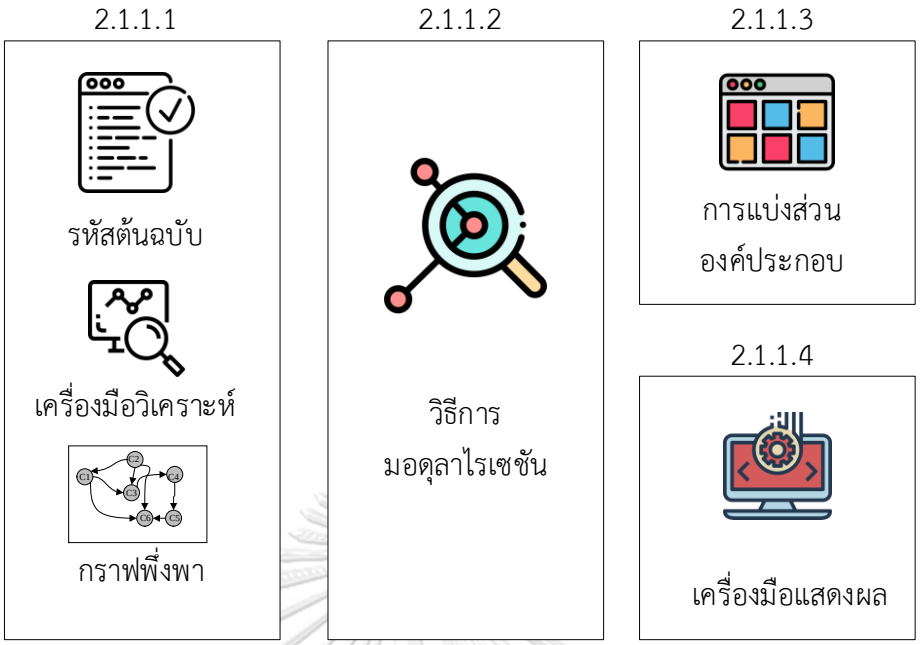
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

วิทยานิพนธ์นี้ได้นำเสนอการจัดองค์ประกอบซอฟต์แวร์ใหม่ เพื่อปรับปรุงสภาพมอดุลาร์ (Modularity) ของซอฟต์แวร์โดยใช้การค้นหาต้องห้าม เพื่อจัดสรรการคลาสให้อยู่ในแพ็กเกจที่เหมาะสม ดังนั้นจึงต้องมีความเข้าใจพื้นฐานที่เกี่ยวข้องกับ การจัดองค์ประกอบของซอฟต์แวร์ ซึ่งจะอธิบายไว้ในหัวข้อที่ 2.1.1 มอดุลาไรเซชันซอฟต์แวร์ (Software Modularization) เพื่อให้ได้คำตอบที่เหมาะสมที่สุด ในวิทยานิพนธ์นี้ได้ใช้วิธีการฮิวริสติกรูปแบบหนึ่งเพื่อค้นหาคำตอบซึ่งจะอธิบายในหัวข้อที่ 2.1.2 การค้นหาต้องห้าม (Tabu Search) โครงสร้างและความเข้าใจในระบบมีอิทธิพลต่อการเปลี่ยนแปลงคุณภาพซอฟต์แวร์ จึงวิเคราะห์ความสัมพันธ์ภายในของซอฟต์แวร์ด้วยความสัมพันธ์ของกราฟพึ่งพาในหัวข้อที่ 2.1.3 กราฟการพึ่งพา (Dependency Graph) และข้อกำหนดที่สำคัญที่จะทำให้การจัดองค์ประกอบนี้เหมาะสม จึงต้องมีการวัดคุณภาพของซอฟต์แวร์ เพื่อหารูปแบบการจัดกลุ่มใหม่ที่เหมาะสม ซึ่งแสดงรายละเอียดการวัดไว้ในหัวข้อที่ 2.1.4 การวัดคุณภาพซอฟต์แวร์ (Software Quality Measurement) การจัดการให้ซอฟต์แวร์มีโครงสร้างเป็นไปตามที่หาความสัมพันธ์ทำโดยการย้ายคลาสไปยังระบบย่อยที่เหมาะสมโดยอธิบายใน หัวข้อที่ 2.1.5 มูฟรีแฟคทอริง (Move Refactoring) และหัวข้อที่ 2.1.6 ดีเพนเดนซีไฟน์เดอร์ (Dependency Finder)

2.1.1 มอดุลาไรเซชันซอฟต์แวร์

มอดุลาไรเซชันซอฟต์แวร์ (Software Modularization) [16] คือกระบวนการเพื่อจัดองค์ประกอบซอฟต์แวร์ให้มีลักษณะเป็นระบบย่อย โดยพิจารณาที่สถาปัตยกรรมของระบบซอฟต์แวร์ ซึ่งระบบย่อยหมายถึง ชุดองค์ประกอบ (คลาส) ของซอฟต์แวร์ที่สัมพันธ์กัน ซึ่งกระบวนการมอดุลาไรเซชันมีกระบวนการดังรูปที่ 2.1 กระบวนการมอดุลาไรเซชัน นั้นแบ่งเป็น 4 ส่วน คือ ส่วนที่ 1 เป็นการวิเคราะห์และจัดเตรียมข้อมูล ได้แก่ การนำเข้ารหัสต้นฉบับ การใช้เครื่องมือเพื่อวิเคราะห์รหัสต้นฉบับเพื่อการแปลงเป็นกราฟพึ่งพา ซึ่งจะเป็นผลลัพธ์ที่ได้จากส่วนแรก ส่วนที่ 2 วิธีการรีมอดุลาไรเซชัน คือวิธีการต่างๆเพื่อที่จะจัดกลุ่มองค์ประกอบของซอฟต์แวร์ให้เป็นส่วนย่อย ส่วนที่ 3 คือการแบ่งส่วนองค์ประกอบซึ่งเป็นผลลัพธ์จากการใช้วิธีการของขั้นตอนก่อนหน้า และส่วนสุดท้ายคือการใช้เครื่องมือเพื่อแสดงผล



รูปที่ 2.1 กระบวนการมอดุลาไรเซชัน

2.1.1.1 การจัดเตรียมข้อมูล หลังจากที่ทีมพัฒนาได้ออกแบบระบบเสร็จและมีฟังก์ชันการทำงานที่ตรงตามความต้องการ นำรหัสต้นฉบับ ของระบบมาเพื่อวิเคราะห์ความสัมพันธ์ จะใช้เทคนิคต่างๆ เพื่อแยกความสัมพันธ์ขององค์ประกอบในระบบโดยแยกให้เป็นส่วนแนวคิดหลักและความสัมพันธ์ย่อยโดยสร้างเป็นกราฟแสดงความสัมพันธ์ ซึ่งขึ้นอยู่กับการเลือกใช้เครื่องมือเพื่อจะนำเอาข้อมูลไปสร้างกราฟพึ่งพา หลังจากที่ได้วิเคราะห์ความสัมพันธ์ของรหัสต้นฉบับแล้ว กราฟพึ่งพานี้จะเป็นส่วนที่แสดงให้เห็นรูปแบบความสัมพันธ์ระหว่างองค์ประกอบในระบบ ซึ่งในขั้นตอนนี้จะทำให้เห็นว่าความสัมพันธ์ของแต่ละองค์ประกอบ มีความซับซ้อนทำให้ยากต่อการทำความเข้าใจ โดยกราฟที่สามารถแสดงความสัมพันธ์นั้นมีหลายประเภทสามารถเลือกเพื่อให้แสดงตามวัตถุประสงค์ได้ เช่น กราฟพึ่งพาแบบการเรียกใช้งาน (Call dependency graph) ที่จะแสดงความสัมพันธ์ในการเรียกใช้งานระหว่างองค์ประกอบในระบบ กราฟพึ่งพาแบบฟีเจอร์ (Feature dependency graph) ก็จะแสดงความสัมพันธ์ระหว่างฟีเจอร์ต่างๆ ในระบบ ส่วนนี้จะได้ข้อมูลส่งออกเป็นกราฟความสัมพันธ์ของคลาสในระบบย่อย

2.1.1.2 วิธีการมอดุลาไรเซชัน ขั้นตอนนี้มีเป้าหมายคือการแบ่งกลุ่มของกราฟด้วยความสัมพันธ์ให้เป็นกลุ่มย่อยที่มีความสัมพันธ์กันมากที่สุด วิธีการมอดุลาไรเซชัน จะต้องนำเข้ากราฟพึ่งพาและต้องดำเนินการเพื่อจัดสรรแบ่งกลุ่มให้กราฟเหล่านั้น ซึ่งวิธีการมอดุลาไรเซชันแบ่งได้เป็น 5 ประเภท คือ

1) วิธีการลำดับชั้น (Hierarchical methods) วิธีการนี้สามารถแบ่งออกได้เป็น 2 แนวทาง คือ จากบนลงล่าง หรือการแบ่งคลาสจากกลุ่มใหญ่เป็นกลุ่มที่เล็กลง และจากล่างขึ้นบนหรือการรวมกันของคลาสขึ้นมาเป็นกลุ่มที่มีความสัมพันธ์กัน

2) วิธีการที่ไม่ใช่ลำดับชั้น (Non-hierarchical methods) วิธีการนี้จะเป็นการดำเนินการแบ่งกลุ่มโดยจำกัดจำนวนกลุ่มไว้ล่วงหน้าและหาค่าเฉลี่ยกลุ่มและแบ่งกลุ่มตามค่าที่ใกล้เคียง คล้ายการแบ่งกลุ่มแบบ K-Mean

3) วิธีการเหมืองข้อมูล (Data-mining based methods) ใช้วิธีการเหมืองข้อมูลมาใช้เพื่อสร้างระบบย่อย ซึ่งจะแตกต่างจากวิธีอื่นที่สามารถใช้จัดการกับข้อมูลขนาดใหญ่ได้

4) วิธีการวิเคราะห์แนวคิด (Concept analysis methods) เป็นวิธีที่ใช้เพื่อวิเคราะห์ประเภทหรือชนิดข้อมูลที่คล้ายกัน ก่อนจะใช้อัลกอริทึมเพื่อจัดกลุ่มข้อมูลโดยจะแยกประเภทและตัวแปรเป็นมอดูล โดยวิธีการนี้ดำเนินการภายใต้หลักการทางคณิตศาสตร์ และมีข้อจำกัดคือไม่สามารถใช้กับข้อมูลที่มีขนาดใหญ่ได้

5) วิธีการค้นหา (Search-based methods) เป็นวิธีการแก้ปัญหาการจัดกลุ่มในการทำด้วยการใช้วิธีการค้นหาแบบฮิวริสติก โดยค้นหาค่าที่ใกล้เคียงมากที่สุดด้วยฟังก์ชันที่เป็นวัตถุประสงค์และพิจารณาถึงการเชื่อมต่อสูงสุดของคลาสในแต่ละมอดูล

2.1.1.3 การแบ่งส่วนองค์ประกอบ ผลลัพธ์ที่ได้ในขั้นตอนนี้มากจากการดำเนินการตามวิธีการแบ่งกลุ่มตามข้อที่ 2.1.1.2 ซึ่งกราฟที่มีการแบ่งกลุ่มและจะได้การจัดองค์ประกอบแบบใหม่โดยการแบ่งกลุ่มใหม่ด้วยความสัมพันธ์ที่เหมาะสมตามวิธีการที่เลือกใช้

2.1.1.4 เครื่องมือแสดงผล ขั้นตอนนี้จะแสดงให้เห็นว่าเมื่อทำการแบ่งกลุ่มใหม่ จะได้ระบบที่มีองค์ประกอบย่อยในลักษณะใด ซึ่งจะมีผลลัพธ์เป็นกราฟความสัมพันธ์ที่มีการจัดกลุ่มใหม่ให้เหมาะสมมากยิ่งขึ้น โดยในขั้นตอนนี้สามารถใช้เครื่องมือที่เหมาะสมกับความต้องการได้

กระบวนการมอดุลาไรเซชัน ที่ได้กล่าวไปนั้นข้อมูลนำเข้าที่จำเป็นของกระบวนการคือข้อมูลนำเข้าระบบนั้นจะต้องแสดงลักษณะพฤติกรรมและโครงสร้างของระบบซอฟต์แวร์ การรีมอดุลาไรเซชันซอฟต์แวร์ เป็นเทคนิคที่เป็นวิวัฒนาการมาจากมอดุลาไรเซชันซอฟต์แวร์ โดยจะทำการจัดองค์ประกอบซอต์ฟแวร์ใหม่อีกครั้ง โดยที่จะปรับปรุงให้การจัดกลุ่มขององค์ประกอบเดิมให้มีความเหมาะสมมากขึ้น

2.1.3 การค้นหาต้องห้าม

การค้นหาต้องห้าม (Tabu Search) ถูกเสนอโดย Fred Glover [14] เป็นการค้นหาแบบฮิวริสติกวิธีหนึ่ง ซึ่งจะใช้ความรู้มาช่วยทำให้การค้นหามีประสิทธิภาพมากยิ่งขึ้น โดยจะแนะนำกระบวนการค้นหาเพื่อเลือกสถานะใดๆ เพื่อทำการค้นหาต่อไปให้ได้คำตอบอย่างมีประสิทธิภาพ และการค้นหาต้องห้ามนี้จะทำเครื่องหมายไว้บนเส้นทางที่ไม่สนใจจะค้นหา การทำเครื่องหมายอยู่ในระดับตัวกระทำกระทำการหรือหน่วยย่อยของตัวกระทำให้อยู่ใน สถานะต้องห้าม (Tabu status) คือหน่วยย่อยนี้จะไม่ถูกนำมาใช้เพื่อสร้างเส้นทางค้นหา ซึ่งอาจจะเพราะเส้นทางนี้จะไม่นำไปสู่คำตอบที่ถูกต้องหรือเคยค้นหามาแล้ว ซึ่งเป็นการนำมาใช้เพื่อตัดสมาชิกบางตัวออกจากการค้นหา แนวคิดของการค้นหาต้องห้ามจะการพิจารณาถึงหน่วยความจำ หน่วยความจำแบบปรับตัว (Adaptive memory) เพื่อการค้นหาอย่างมีประสิทธิภาพ และพิจารณาการสำรวจแบบตอบสนอง (Responsive exploration) เนื่องจากแนวคิดที่ว่าบางครั้งเส้นทางที่ไม่ดีให้ข้อมูลมากกว่าเส้นทางที่ดี [17] เพื่อให้สามารถหาเส้นทางใหม่ที่มีประสิทธิภาพใหม่ที่ดีขึ้น โดยการค้นหาต้องห้ามนี้มีจุดเด่นที่มีความละเอียดและหลากหลายในการค้นหาคำตอบ เพราะมีการค้นหาเพิ่มเติมในบริเวณที่ต่างจากคำตอบที่ดีที่เคยพบและค้นหาเพิ่มในบริเวณใกล้เคียงคำตอบที่ดีด้วย [18] กำหนดจุดประสงค์ในการค้นหา การค้นหานั้นต้องมีการกำหนดปัญหาหรือวัตถุประสงค์ในการทำการค้นหาก่อน เพื่อให้การค้นหาดำเนินไปอย่างเหมาะสม พิจารณาจุดข้างเคียงหรือเพื่อนบ้านใกล้เคียง (Neighborhood) เพื่อค้นหาแนวทางต่างๆ ในการค้นหาที่เป็นไปได้ ซึ่งเป็นวิธีการที่จะสามารถเปลี่ยนการค้นหาไปจากสถานะปัจจุบันได้ในรอบการค้นหาต่อไป การกำหนดลักษณะเพื่อนบ้านนั้นมีความสำคัญสำหรับประสิทธิภาพการค้นหาต้องห้าม การกำหนดเพื่อนบ้านที่ไม่เหมาะสมอาจจะทำให้โอกาสที่จะพบคำตอบที่ดีคลาดเคลื่อนไปหรืออาจจะต้องใช้เวลาในการค้นหามากขึ้น รายการตัวเลือก (Candidate list) เป็นเซตย่อยของเพื่อนบ้าน โดยที่การใช้รายการตัวเลือกจะช่วยให้การค้นหา รายการต้องห้าม เป็นรายการที่เก็บข้อมูลในการค้นหาที่จะไม่ต้องทำการค้นหาในการค้นหารอบต่อไป [13] การค้นหาต้องห้าม จะค้นหาคำตอบที่เหมาะสมที่สุดของปัญหา โดยที่จะต้องกำหนดปัญหาที่จะทำการหาคำตอบ ขั้นตอนการค้นหาต้องห้ามโดยสรุปมีขั้นตอนดังนี้

ขั้นตอนที่ 1 การกำหนดข้อกำหนดเบื้องต้น โดยจะกำหนดจุดเริ่มต้น เพื่อนบ้านใกล้เคียง เงื่อนไขการค้นหา

ขั้นตอนที่ 2 การค้นหา (มีการวนรอบ) ค้นหาคำตอบหรือชุดคำตอบที่เป็นไปได้ โดยต้องการค้นหาคำตอบที่เป็นไปได้จากเงื่อนไขที่กำหนดจากเพื่อนบ้านใกล้เคียง ถ้าพบคำตอบที่ดีที่สุดแล้วจะบันทึกคำตอบนั้นแทนและปรับปรุงรายการต้องห้าม แต่ถ้าหากการค้นหายังไม่พบคำตอบที่ดีที่สุดจะทำการค้นหาจนกว่าจะครบรอบที่กำหนดและดำเนินการขั้นต่อไป

ขั้นตอนที่ 3 การหยุดการค้นหา จะหยุดเมื่อถ้าพบว่าเงื่อนไขที่กำหนดนั้นเป็นจริง หรือครบรอบการค้นหาแล้ว

ขั้นตอนที่ 4 ทำการค้นหาจนครบตามเงื่อนไขและปรับปรุงรายการต้องห้ามแล้ว กลับไปดำเนินการค้นหาอีกครั้ง

2.1.2.1 เพื่อนบ้านใกล้เคียง หรือโหนดใกล้เคียงที่มีความสัมพันธ์กัน ในการค้นหาต้องห้ามการค้นหาคำตอบเป็นการย้ายเพื่อค้นหาคำตอบที่ดีที่สุดของปัญหา การค้นหาจากเพื่อนบ้านหรือบริเวณใกล้เคยนั้นจะช่วยให้ได้คำตอบที่มีประสิทธิภาพมากที่สุด [19]

2.1.2.2 รายการต้องห้าม จะเกิดการย้ายทำให้โครงสร้างภายในของบริเวณค้นหาและเพื่อนบ้านใกล้เคียงเกิดการเปลี่ยนแปลง เพื่อเป็นการค้นหาคำตอบที่ดีที่สุด แต่เพื่อป้องกันไม่ให้เกิดการย้ายที่ซ้ำซ้อนจึงมีการเก็บค่าที่เคยย้ายแล้วไว้ในรายการต้องห้าม (Tabu List) ซึ่งรายการต้องห้ามนี้สามารถกำหนดขนาดเพื่อความเหมาะสมได้ หรือกำหนดระยะเวลาเพื่อให้ค่านั้นอยู่ในรายการต้องห้ามได้อีกด้วย ทั้งนี้ขึ้นอยู่กับแนวทางการแก้ปัญหา

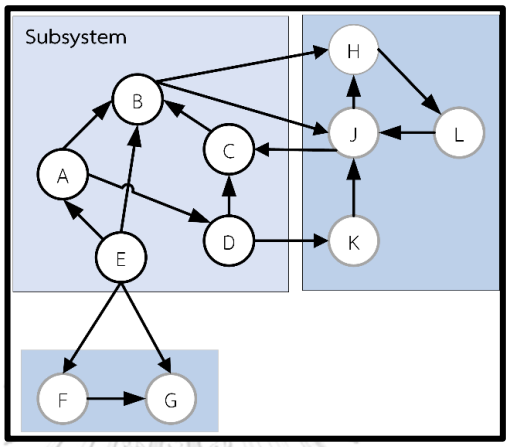
หลักการพื้นฐานของการค้นหาต้องห้ามนี้ คือการค้นหาคำตอบที่ดีที่สุดจากบริเวณใกล้เคียงหรือเพื่อนบ้าน โดยจะเกิดการย้ายเมื่อเกิดคำตอบที่เหมาะสมกว่าคำตอบเดิม และปล่อยผ่านคำตอบที่ไม่ได้ให้คำตอบที่ดีสำหรับการแก้ปัญหา การนำการค้นหาต้องห้ามมาใช้งานจึงมีหลากหลาย เพราะเป็นวิธีการแก้ปัญหาที่ยืดหยุ่น

2.1.3 กราฟพึ่งพา

กราฟพึ่งพา (Dependency Graph) [16] เป็นกราฟมีทิศทางที่นำมาใช้เพื่อแสดงความสัมพันธ์ของคลาสที่อยู่ในระบบย่อยและระบบเดียวกัน โดยที่กราฟพึ่งพานี้จะสร้างขึ้นจากความสัมพันธ์ของรหัสต้นฉบับ และจะแสดงลักษณะพฤติกรรมของระบบได้ในลักษณะที่เป็นกราฟ ที่ประกอบด้วยโหนด (Node) แทน คลาส และเส้นเชื่อม (Edges) แทน ความสัมพันธ์ต่างๆ ระหว่างคลาส กราฟพึ่งพามีหลายประเภท ซึ่งแต่ละประเภทแตกต่างกันไปตามจุดประสงค์การนำไปใช้ และประเภทข้อมูล ประเภทของกราฟพึ่งพา กราฟพึ่งพาแบบมอดูล (Module dependency graph) ซึ่งจะเป็นกราฟพึ่งพาที่จะแสดงการพึ่งพากันระหว่างคลาสในระบบหรือในมอดูล เช่น กราฟพึ่งพาแบบการเรียกใช้งาน ซึ่งเป็นกราฟที่จะแสดงความสัมพันธ์ระหว่างคลาสที่แสดงการความสัมพันธ์การเรียกใช้งานระหว่าง 2 คลาส แต่ละโหนดแทนคลาส เส้นเชื่อมแทนเมท็อดที่มีการเรียกใช้งานระหว่างคลาสในรหัสต้นฉบับ ซึ่งกราฟพึ่งพาแบบการเรียกใช้งานนั้นสามารถใช้ได้กับทั้งข้อมูลไดนามิก (Dynamic) และสแตติก (Static) ซึ่งจะศึกษาเฉพาะแบบสแตติกซึ่งเป็นการแสดงการทำงานที่จะเกิดขึ้นได้ในระบบซอฟต์แวร์

กราฟพึ่งพาของระบบย่อยสามารถจำแนกลักษณะความสัมพันธ์ของการพึ่งพาได้เป็น 4 แบบ [20] โดยเป็นการพิจารณาคลาสที่มีการพึ่งพากันในระบบ พึ่งพากันภายในระบบย่อย พึ่งพากันระหว่างระบบย่อย และการพึ่งพาที่เกี่ยวข้องกับระบบย่อยหนึ่งๆ จะสามารถแสดงเป็นตัวอย่างการพิจารณาได้ดังนี้

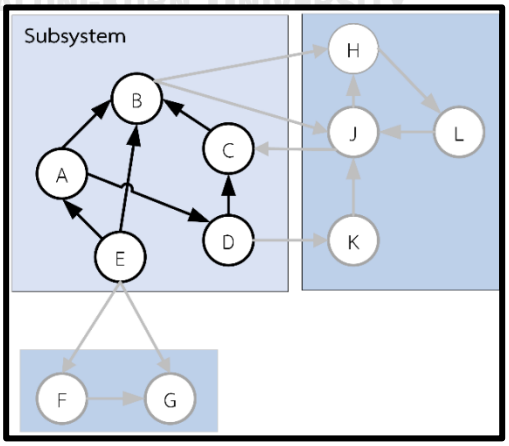
1) กราฟพึ่งพาของระบบย่อย แสดงการพึ่งพาของคลาสที่เกี่ยวข้องกันทั้งระบบ



รูปที่ 2.2 กราฟพึ่งพาของระบบย่อยภายในระบบ

จากรูปที่ 2.2 เป็นกราฟพึ่งพาของระบบย่อยภายในระบบ ซึ่งจะแสดงความสัมพันธ์ของคลาสที่อยู่ภายในระบบและทุกระบบย่อยภายในระบบ

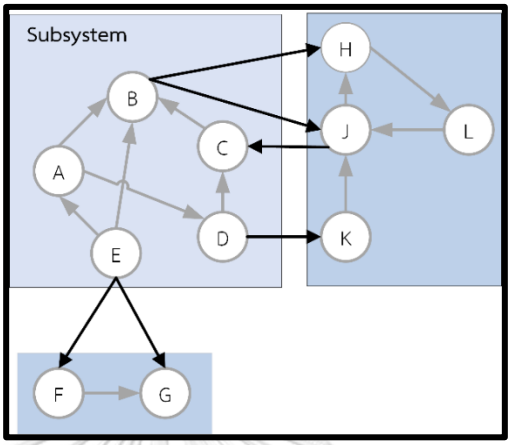
2) กราฟที่แสดงการพึ่งพาภายในระบบย่อย คือความสัมพันธ์ของคลาสภายในระบบย่อยเดียวกัน



รูปที่ 2.3 กราฟพึ่งพาภายในระบบย่อย

จากรูปที่ 2.3 เป็นกราฟพึ่งพาภายในระบบย่อยหนึ่ง ซึ่งจะแสดงความสัมพันธ์ระหว่างคลาสที่อยู่ภายในระบบย่อยนี้ที่มีต่อกันภายในระบบเท่านั้น

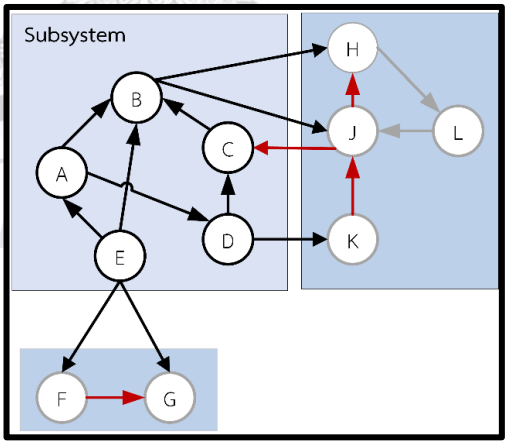
3) กราฟที่แสดงการพึ่งพาระหว่างคลาสในระบบย่อยกับระบบย่อยอื่น



รูปที่ 2.4 กราฟพึ่งพากับระบบย่อยภายนอก

จากรูปที่ 2.4 เป็นกราฟพึ่งพาระหว่างคลาสของระบบย่อยหนึ่งกับคลาสของระบบย่อยอื่นๆที่เกี่ยวข้องกัน ซึ่งเป็นการแสดงการพึ่งพาระบบภายนอก

4) กราฟแสดงการพึ่งพาที่เกี่ยวข้องทั้งหมดในระบบย่อย



รูปที่ 2.5 กราฟพึ่งพาที่มีความเกี่ยวข้องกับระบบย่อย

จากรูปที่ 2.5 เป็นกราฟพึ่งพาภายในระบบย่อยหนึ่งและการพึ่งพาระบบย่อยอื่นๆ ที่เกี่ยวข้อง ซึ่งจะแสดงความสัมพันธ์ระหว่างคลาสที่อยู่ภายในระบบย่อยนี้ด้วยกันและกับคลาสที่อยู่ในระบบย่อยอื่นๆ

ในปัจจุบันมีเครื่องมือสนับสนุนการวิเคราะห์และสร้างกราฟพึ่งพา โดยจะทำการวิเคราะห์รหัสต้นฉบับเพื่อสร้างกราฟพึ่งพา เป็นการจำลองลักษณะและพฤติกรรมของซอฟต์แวร์ ซึ่งเครื่องมือ

ในปัจจุบันมีทั้งเครื่องมือเชิงธุรกิจ และเครื่องมือจากโอเพนซอร์ส โดยมีลักษณะที่แตกต่างกันตามประเภทการใช้งานและภาษาของรหัสต้นฉบับ

2.1.4 การวัดคุณภาพซอฟต์แวร์

การวัดคุณภาพซอฟต์แวร์ และประเมินการทำรีมอดุลาไรเซชันของซอฟต์แวร์ เป็นเงื่อนไขที่ชี้ถึงคุณภาพของระบบซอฟต์แวร์โดยจะทำการประเมินทั้งก่อนและหลังการจัดองค์ประกอบเพื่อให้ชี้ให้เห็นถึงคุณภาพที่แตกต่างกันของการจัดองค์ประกอบ ซึ่งคือสภาพมอดุลาร์ของซอฟต์แวร์ (Software Modularity) การพิจารณาสภาพมอดุลาร์ของซอฟต์แวร์มาจากปัจจัยคุณภาพของซอฟต์แวร์ 2 ประการคือ ความสามารถนำกับมาใช้ใหม่ (Reusability) และ ความสามารถในการขยาย (Extendibility) สภาพมอดุลาร์จึงเป็นคุณลักษณะหนึ่งที่บ่งชี้ถึงคุณภาพซอฟต์แวร์ โดยที่สภาพมอดุลาร์คือการแบ่งซอฟต์แวร์เป็นส่วนย่อยๆ และทำให้ซอฟต์แวร์มีการเกาะกลุ่ม (Cohesion) สูงและเข้าคู่ (Coupling) ต่ำ และการทำรีมอดุลาไรเซชันมีเป้าหมายเพื่อปรับปรุงให้ซอฟต์แวร์มีสภาพมอดุลาร์ที่ดีขึ้น แนวคิดคือให้ระบบย่อยเดียวกันมีการเกาะกลุ่มกันสูงและระหว่างกลุ่มมีค่าการเข้าคู่กันต่ำ โดยที่มีตัววัดที่นำมาใช้จะต้องมีหลักการพื้นฐานดังที่กล่าวมา

2.1.4.1 ค่าสัมประสิทธิ์ซิลูเอท ถูกเสนอโดย Peter J. Rousseeuw ในปี 1986 [15] โดยมีวัตถุประสงค์เพื่อใช้สำหรับเทคนิคการจัดกลุ่มข้อมูล โดยพิจารณาการยึดเหนี่ยวภายในกลุ่มและความสามารถในการแยกกันระหว่างกลุ่ม ค่าค่าสัมประสิทธิ์ซิลูเอทเป็นเกณฑ์ภายใน (Internal criteria) สำหรับการประเมินคุณภาพของการทำมอดุลาไรเซชัน [16] โดยใช้คลาสและความสัมพันธ์ระหว่างกัน ซึ่งการวัดคุณภาพนี้เป็นการวัดความคล้ายในระบบย่อย โดยจะใช้ค่าสัมประสิทธิ์ซิลูเอทประเมินจากข้อมูลการพึ่งพาระหว่างคลาส ซึ่งมี 3 ส่วนดังนี้

1) จำนวนการเชื่อมต่อระหว่างคลาส i กับคลาสใดๆ ในกลุ่มเดียวกัน ให้เป็น: a_i

2) จำนวนการเชื่อมต่อระหว่างคลาส i กับคลาสใดๆ ระหว่างกลุ่ม ให้เป็น: b_i

3) ค่าสัมประสิทธิ์ซิลูเอท (S) สามารถคำนวณได้จากสมการที่ 1

$$S(i) = \frac{a(i) - b(i)}{Max\{a(i), b(i)\}} \qquad (1)$$

ช่วงของค่าสัมประสิทธิ์ คือ ช่วงตั้งแต่ -1 ถึง 1 โดยที่หากมีค่าใกล้ 1 แสดงว่าคลาสใดๆ มีความเหมาะสมกับกลุ่มมาก ซึ่งคือมีค่าการเกาะกลุ่มสูง แต่หากเข้าใกล้ -1 แสดงว่ามีความผิดพลาดคือมีค่าการเกาะกลุ่มต่ำ ทำให้มีความเป็นน่าจะเป็นสูงที่คลาสใดๆจะต้องทำการมูฟรีแฟคทอริง คำนวนค่า S(i) ของคลาสใดๆ ในแต่ละกลุ่มเพื่อหาค่าเฉลี่ย $\bar{S}$ เพื่อประเมินความเหมาะสมภายในกลุ่ม ซึ่งก็จะมีค่าอยู่ระหว่าง -1 ถึง 1 ตามตารางที่ 1

ตารางที่ 2.1 การประเมินค่าสัมประสิทธิ์ซิลูเอท

ค่าสัมประสิทธิ์ซิลูเอท (S)	ความหมาย
ช่วงระหว่าง 0.71-1	โครงสร้างของกลุ่มอยู่ในเกณฑ์ที่ดี
ช่วงระหว่าง 0.51-0.71	โครงสร้างของกลุ่มอยู่ในเกณฑ์ที่ยอมรับได้
ช่วงระหว่าง 0.26 – 0.50	ควรปรับปรุงการจัดกลุ่ม
น้อยกว่า 0.25	โครงสร้างของกลุ่มไม่มีความสัมพันธ์กัน

2.1.4.2 เทอร์โบเอ็มคิว (TurboMQ) [21] เป็นตัววัดสำหรับการวัดคุณภาพของการทำมอดุลาไรเซชัน (Modularization Quality) ชนิดหนึ่ง เป็นการวัดเพื่อประเมินคุณภาพที่เหมาะสมของการจัดองค์ประกอบจากการทำมอดุลาไรเซชันโดย เป็นการวัดการเกาะกลุ่ม (Cohesion) และการเข้าคู่ (Coupling) โดยจะวัดการเชื่อมต่อภายในกลุ่มเดียวกัน และวัดจำนวนการเชื่อมกับภายนอกกลุ่ม การคำนวณค่าเทอร์โบเอ็มคิวทำได้ดังนี้

$$Turbo\ MQ = \sum CF_i \ ; \qquad (4)$$

$$CF_i = \frac{2\mu_i}{2\mu_i + \Sigma\ (\varepsilon_{i,j} + \varepsilon_{j,i})} \qquad (5)$$

Where

CF_i คือค่าแฟคเตอร์ที่คำนวณได้จากแพ็กเกจที่ *i* ซึ่งค่าของแต่ละแพ็กเกจจะอยู่ในช่วง 0 ถึง 1

μ_i คือจำนวนความสัมพันธ์ภายในแพ็กเกจที่ *i*

$\varepsilon_{i,j}$ คือจำนวนความสัมพันธ์ระหว่าง2แพ็กเกจ แพ็กเกจ *I และแพ็กเกจ j ใดๆ*

ค่าคุณภาพของมอดุลาไรเซชัน คือผลรวมของ CF ของแต่ละกลุ่มรวมกัน ซึ่งการประเมินคุณภาพสามารถทำได้โดยการคำนวณค่า ผลต่างของคุณภาพก่อนและหลังการทำมอดุลาไรเซชัน

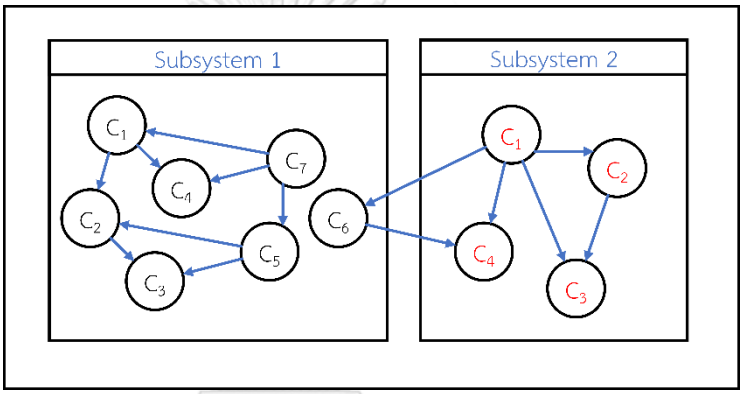
2.1.5 มูฟรีแฟคทอริง

การรีแฟคทอริงเป็นการปรับปรุงโครงสร้างของซอฟต์แวร์โดยที่ไม่ทำให้ลักษณะพฤติกรรมของซอฟต์แวร์เปลี่ยนแปลงไป โดยการปรับปรุงโครงสร้างที่มีอยู่ให้มีลักษณะที่ดีขึ้น ง่ายต่อความเข้าใจ และสามารถนำไปพัฒนาต่อได้สะดวกมากขึ้น การรีแฟคทอริงมีวิธีการมากถึง 72 วิธี โดยในวิทยานิพนธ์นี้นี้เลือกใช้วิธีการมูฟรีแฟคทอริง เพื่อย้ายคลาสไปยังระบบย่อยที่เหมาะสมหลังมีการค้นหาระบบย่อยใหม่แล้ว

การมูฟรีแฟคทอริง (Move Refactoring) [22] คือการปรับปรุงโครงสร้างของซอฟต์แวร์ โดยการย้ายคลาสจากระบบย่อยเดิมไปยังระบบย่อยอื่น เมื่อคลาสนั้นถูกเรียกใช้โดยคลาสในระบบย่อยอื่นมากกว่าหรือมีความสัมพันธ์กับคลาสในระบบย่อยอื่นมากกว่า จึงควรย้ายคลาสนั้นไปยังระบบย่อยอื่น

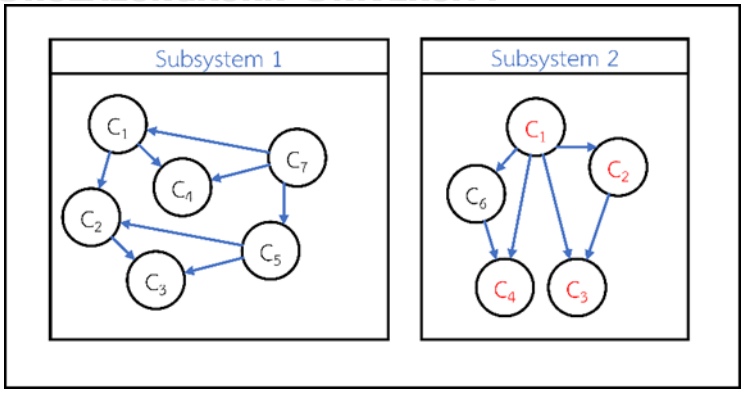
แล้วจะทำการลบคลาสที่อยู่ในระบบย่อยเดิมออก โดยการย้ายคลาสนี้คือ การย้ายโหนดของกราฟจากกราฟพึ่งพาโดยย้ายไปยังระบบย่อยอื่น สาเหตุการทำมูฟรีแฟคทอริง ได้แก่ การเกิดฟีเจอร์เอนวี (Feature Envy) คือการที่คลาสถูกสร้างในระบบซอฟต์แวร์ที่มีขนาดใหญ่เกินไปหรือไม่ถูกเรียกใช้ ซึ่งการย้ายไปยังระบบย่อยอื่นที่มีการเรียกใช้ หรือมีความสัมพันธ์มากกว่าจะทำให้ซอฟต์แวร์มีลักษณะที่ดีขึ้นและลดความซับซ้อนของซอฟต์แวร์ได้

ดังตัวอย่างในรูปที่ 2.6 เป็นตัวอย่างระบบย่อยเมื่อพิจารณาจากกราฟที่แสดงความสัมพันธ์แล้ว พบว่าต้องมีการรีแฟคทอริงเนื่องจากการที่มีคลาสที่ไม่ถูกเรียกใช้งานในระบบย่อยของตัวเอง แต่มีการเรียกใช้งานจากระบบย่อยอื่น



รูปที่ 2.6 ตัวอย่างระบบย่อยที่ต้องทำการรีแฟคทอริง

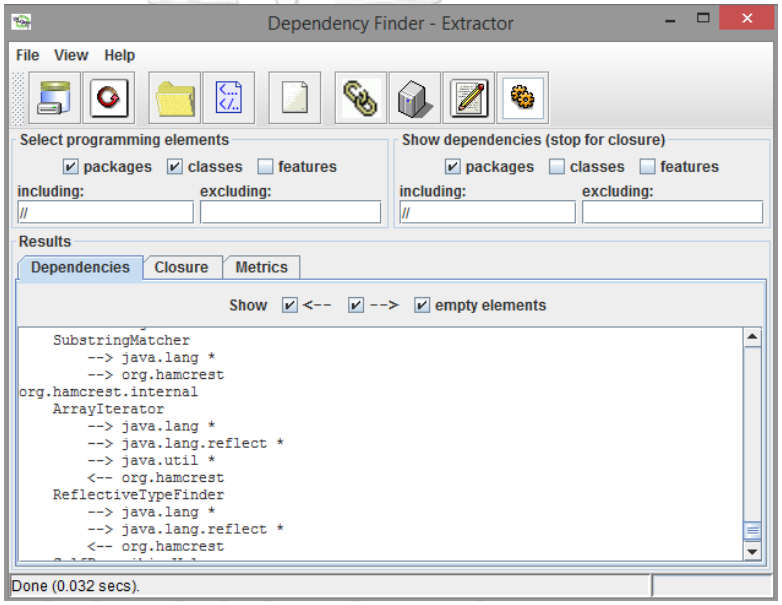
จากรูปที่ 2.6 นั้นจะพบว่า คลาสที่ C6 ควรต้องการรีแฟคทอริง เพื่อปรับปรุงให้ซอฟต์แวร์มีลักษณะที่ดีขึ้น ในที่นี้จะทำการมูฟรีแฟคทอริง ย้ายคลาส C6 จากระบบย่อย 1 เป็นระบบย่อย 2 ดังรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างระบบย่อยหลังการรีแฟคทอริง

2.1.6 เครื่องมือดีเพนเดนซีไฟน์เดอร์

ดีเพนเดนซี ไฟน์เดอร์ (Dependency Finder) [23] พัฒนาโดย Jean Tessier เป็นชุดเครื่องมือสำหรับการวิเคราะห์รหัสต้นฉบับภาษาจาวา โดยที่เครื่องมือนี้ถูกออกแบบมาเพื่อการแปลงข้อมูลจากรหัสต้นฉบับมาเป็นกราฟพึ่งพา และยังสามารถคำนวณเมตริกพื้นฐานด้วย รูปที่ 2.8 เป็นตัวอย่างเครื่องมือดีเพนเดนซี ไฟน์เดอร์ โดยจะสามารถเคราะห์กราฟพึ่งพาจากรหัสต้นฉบับได้อย่างมีประสิทธิภาพ โดยมีงานวิจัยก่อนหน้า [2, 24] ที่ได้ใช้เครื่องมือนี้สำหรับการวิเคราะห์เพื่อสร้างกราฟพึ่งพา โดยจะได้ไฟล์กราฟพึ่งพาในรูปแบบไฟล์นามสกุล .xml โดยจะได้กราฟพึ่งพาจากเครื่องมือที่มีรายละเอียดของแพ็กเกจในระบบของซอฟต์แวร์ คลาสที่อยู่ในแพ็กเกจแต่ละแพ็กเกจ ฟีเจอร์ทั้งหมดที่อยู่ในคลาส และความสัมพันธ์ทั้งในระดับแพ็กเกจและคลาส



รูปที่ 2.8 ตัวอย่างเครื่องมือดีเพนเดนซีไฟน์เดอร์

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการรีมอดุลาไรเซชันซอฟต์แวร์ แบ่งเป็น 2 กลุ่มคือ งานกลุ่มแรกเป็นการศึกษาวิธีการทำรีมอดุลาไรเซชัน เพื่อหาวิธีการที่มีประสิทธิภาพมากที่สุด ซึ่งได้แก่ การปรับปรุงเทคนิคการค้นหา การเสนอเทคนิคใหม่เพื่อการรีมอดุลาไรเซชัน และงานอีกกลุ่มคือการศึกษามาตรวัดที่มีอิทธิพลต่อการรีมอดุลาไรเซชัน

2.2.1 งานวิจัยที่เกี่ยวข้องกับการศึกษาวิธีการรีมอดุลาไรเซชันซอฟต์แวร์

1) งานวิจัย“Automatic Package Coupling and Cycle Minimization” [11]

งานวิจัยนี้ได้เสนอวิธีการทำมอดุลาไรเซชันโดยอัตโนมัติ โดยใช้หลักการลดเข้าคู่กันของคลาสระหว่างแพ็กเกจ (Package) และการลดขนาดวงการเชื่อมต่อของคลาส คณะผู้วิจัยได้เสนอวิธีแก้ปัญหาการจัดกลุ่มของซอฟต์แวร์ที่มีอยู่ ด้วยวิธีค้นหา เสนออัลกอริทึมเพื่อจัดหาแพ็กเกจที่เหมาะสมสำหรับคลาสในระบบ และเสนอตัวชี้วัดในการประเมินการทำมอดุลาไรเซชันของซอฟต์แวร์ เพื่อทำให้สามารถประเมินคุณภาพของการทำมอดุลาไรเซชันได้อย่างอัตโนมัติ โดยใช้วิธีในการดำเนินการแก้ไขโครงสร้างโดยใช้การมูฟรีแฟคทอริง เพื่อไม่ทำให้ลักษณะและพฤติกรรมของซอฟต์แวร์ไปแตกต่างจากเดิม คณะผู้จัดทำได้ดำเนินการโดยเลือกใช้การค้นหาแบบอบเหนียวจำลอง (Simulated annealing) โดยใช้เทคนิคการค้นหาเพื่อนบ้านใกล้เคียง (neighborhood search-based) เพื่อนำมาใช้ค้นหากลุ่มของคลาสที่เหมาะสมที่สุด และได้เสนอชุดของตัววัดคุณภาพ 2 ประเภท คือ คุณภาพของการทำมอดุลาไรเซชัน และคุณภาพของแพ็กเกจ

งานวิจัยนี้ได้ใช้ข้อมูลเพื่อทดลองในกระบวนการและวิธีการที่ได้เสนอ ด้วยการใช้ระบบซอฟต์แวร์จากโอเพนซอร์ส จำนวน 4 โครงการ คือ JEdit, ArgoUML, Jboss และ Azureus ซึ่งได้ผลการทดลองจากทั้งหมดพบว่า สามารถลดการเข้าคู่ระหว่าคลาสและลดขนาดวงการเชื่อมต่อได้ จากการศึกษางานวิจัยชิ้นนี้ได้ให้ความสำคัญกับการค้นหาตัววัดและการเสนออัลกอริทึมเพื่อใช้ประเมินคุณภาพการทำมอดุลาไรเซชันอย่างอัตโนมัติ

2) งานวิจัย “Class Modularization Using Indirect Relationships” [25]

งานวิจัยนี้ได้ศึกษาเกี่ยวข้องกับการทำมอดุลาไรเซชันในระดับคลาส นั่นคือการจัดกลุ่มคลาสโดยใช้ความสัมพันธ์ของคลาสเป็นตัวชี้วัด ซึ่งวิธีการที่นำเสนอนี้ได้มีการใช้ความสัมพันธ์ทั้งแบบทางตรงและทางอ้อม เนื่องจากมีแนวคิดว่า ความสัมพันธ์ทางอ้อมระหว่างคลาสที่มีต่อกันนั้นก็สามารถทำให้เกิดผลกระทบเมื่อคลาสใดคลาสหนึ่งมีการเปลี่ยนแปลง โดยคณะผู้วิจัยได้พิจารณาว่าความสัมพันธ์ทั้งทางตรงและทางอ้อมนั้นมีผลต่อการเปลี่ยนแปลงของซอฟต์แวร์ทั้งสิ้น จึงควรพิจารณาความสัมพันธ์ทางอ้อมของระบบด้วย กำหนดความสัมพันธ์ทั้งทางตรงและทางอ้อม รวมทั้งงานวิจัยนี้ได้เริ่มกระบวนการโดยให้แต่ละคลาสอยู่อย่างอิสระ คือยังไม่เป็นมอดุลาไรเซชัน ซึ่งการจัดโครงสร้างในงานวิจัยนี้ได้ทำการเลือกวิธีการทำมอดุลาไรเซชันโดยใช้อัลกอริทึม Hierarchical agglomerative ซึ่งจะดำเนินการรวมและแตกย่อยกลุ่มของคลาสให้มีรูปแบบที่เหมาะสม ได้ทดสอบกับซอฟต์แวร์จากโอเพนซอร์ส จำนวน 3 โครงการ คือ ArgoUML JHotDraw และJMeter

3) งานวิจัย “Automated Software Remodularization Based on Move Refactoring” [26]

งานวิจัยนี้ได้ศึกษาการรีมอดุลาไรเซชัน โดยได้เสนอวิธีการรีมอดุลาไรเซชันด้วยการมูฟรีแฟคทอริง ในระบบที่มีความซับซ้อน โดยใช้อัลกอริทึมสโทแคสติกอย่างง่าย (Simple Stochastic Algorithm) ในการทำมอดุลาไรเซชัน เพื่อหากลุ่มของมอดูลที่ให้ค่าการเข้าคู่กันระหว่างกลุ่มต่ำและค่าการเกาะกลุ่มสูงที่สุด โดยได้เสนออัลกอริทึมสโทแคสติกการมูฟรีแฟคทอริง และใช้ความน่าจะเป็นในการชี้ว่าคลาสใดควรย้ายไปแพ็กเกจอื่นเป็นการตัดสินใจ งานวิจัยนี้คณะผู้วิจัยมีการใช้ซอฟต์แวร์จากโอเพนซอร์สเพื่อนำมาทดสอบกระบวนการจำนวน 39 โครงการ โดยงานวิจัยนี้ได้สร้างเครื่องมือชื่อ SOMOMOTO ขึ้นเพื่อการแนะนำการมูฟรีแฟคทอริงได้อย่างอัตโนมัติ และได้มีการเปรียบเทียบการทดลองโดยการเทียบกับนักพัฒนา ซึ่งได้ผลคือการมูฟรีแฟคทอริงคลาสที่วิธีตามงานวิจัยนี้แนะนำนั้นหาได้มากกว่าที่นักพัฒนาหาได้ และเปรียบเทียบค่า Precision และ Recall

4) งานวิจัย “A novel approach for automatic remodularization of software systems using extended ant colony optimization algorithm” [12]

งานวิจัยนี้ได้ทำการเสนอวิธีการทำรีมอดุลาไรเซชันรูปแบบใหม่คือการนำเอาวิธีการหาคำตอบที่ดีที่สุด (Optimization) จึงมีการนำวิธีการอาณาจักรมดมาใช้เพื่อทำการรีมอดุลาไรเซชัน โดยที่งานวิจัยนี้ได้จัดทำเครื่องมือโดยใช้วิธีการอาณาจักรมดนี้ ซึ่งได้ทดลองใช้กับซอฟต์แวร์ จำนวน 7 ตัวอย่าง โดยใช้ความสัมพันธ์กราฟพึ่งพาและได้ใช้ตัววัดคุณภาพเทอร์โบเพื่อประเมินคุณภาพของการทำรีมอดุลาไรเซชัน ซึ่งได้ผลที่มีแนวโน้มที่ดีเมื่อเปรียบเทียบกับวิธีการอื่นๆ ได้แก่ Bunch-GA, Bunch-Hill Climbing ดังนั้นงานนี้จึงเป็นแนวทางที่ชี้ให้เห็นว่าการพัฒนาวิธีการค้นหาใหม่ๆ สามารถเพิ่มทางเลือกให้กับการรีมอดุลาไรเซชันซอฟต์แวร์ได้

2.2.2 งานวิจัยที่เกี่ยวข้องกับมาตรวัด

1) งานวิจัย “Modularization Metrics : Assessing Package Organization in Legacy Large Object-Oriented Software” [27]

งานวิจัยนี้ได้ศึกษาเกี่ยวข้องกับการหาตัววัดและปัจจัยที่มีผลต่อการจัดองค์ประกอบในระบบซอฟต์แวร์เชิงวัตถุขนาดใหญ่แบบเดิม ซึ่งก็คือหาตัววัดในการประเมินการทำมอดุลาไรเซชัน งานวิจัยนี้คณะผู้วิจัยได้เสนอชุดมาตรวัดโดยพิจารณาจากหลักแนวคิดเรื่องมอดุลาไรเซชัน ซึ่งงานวิจัยนี้ได้ดำเนินการภายใต้ 3 หลักการคือ การซ่อนข้อมูล (Information-Hiding) ความสามารถในการเปลี่ยนแปลงได้ (Changeability) และความสามารถในการนำกลับมาใช้ซ้ำ (Reusability) โดยที่การหามาตรวัดนี้พิจารณาจากการพึ่งพากันของซอฟต์แวร์เชิงวัตถุ คือการเรียกใช้งาน (Call) และการสืบ

ทอดคุณสมบัติ (Inheritance) ระหว่างคลาสในระบบ งานวิจัยนี้ได้เสนอมาตรวัด ได้แก่ มาตรวัดการเข้าคู่ (Coupling Metrics) และมาตรวัดการเกาะกลุ่ม (Cohesion Metrics) โดยมีรายละเอียดดังนี้

มาตรวัดการเข้าคู่ ประกอบด้วยตัววัด 2 ประเภทคือ

(1) Index of Inter-Package Interaction ได้เสนอตัววัดที่คล้ายกัน 2 ตัวคือ IIPU ซึ่งเป็นตัวชี้จำนวนของความสัมพันธ์ของการใช้ข้อมูลระหว่างคลาสกับแพ็กเกจ โดยมีค่าอยู่ระหว่าง 0 ถึง 1 โดยเมื่อค่าเป็น 1 คือมีการเชื่อมต่อระหว่างคลาสกับแพ็กเกจอื่นน้อย นั่นคือมีความสัมพันธ์กับคลาสในแพ็กเกจเดียวกัน และ IIPE โดยมีค่าอยู่ระหว่าง 0 ถึง 1 โดยเมื่อค่าเป็น 1 คือชี้ว่าคลาสที่มีความสัมพันธ์กับคลาสที่ได้รับสือทอดนั้นถูกจัดอยู่ในแพ็กเกจเดียวกัน นั่นคือเมื่อมีการเปลี่ยนใดเกิดขึ้นก็จะไม่กระทบกับแพ็กเกจอื่น

(2) Index of Package Changing Impact ได้เสนอตัววัดใหม่เพื่อจะวัดผลกระทบจากการเปลี่ยนแปลงระหว่างแพ็กเกจ โดยมีแนวคิดในการลดการพึงพากันระหว่างแพ็กเกจ โดยที่หากคลาสหรือแพ็กเกจมีการเพิ่งพากับคลาสหรือแพ็กเกจอื่นจำนวนมากจะทำให้เมื่อเกิดการเปลี่ยนแปลงต้องพิจารณาคลาสหรือแพ็กเกจที่เกี่ยวข้องจำนวนมากเช่นกัน ซึ่งจะเป็นผลเสียในการทำมอดุลาไรเซชันโดยที่ ตัววัดคือ IPCI จะมีค่าอยู่ระหว่าง 0 ถึง 1 โดยค่าที่คาดหวังคือ 1

มาตรวัดการเกาะกลุ่ม ประกอบด้วยมาตรวัด 2 ประเภทคือ

(1) Index of Package Goal Focus คณะผู้วิจัยมีแนวคิดในการพิจารณาจากหน้าที่ของแพ็กเกจโดยควรมีหน้าที่เดียวกันที่ให้กับไคล์เอนทั้งหมด โดยที่ะมีค่าอยู่ระหว่าง 0 ถึง 1 โดยค่าที่คาดหวังคือ 1

(2) Index of Package Services Cohesion คณะผู้วิจัยมีแนวคิดในการวัดความเกาะกลุ่มสำหรับการบริการแบบคอมโพสิทจากการคล้ายกันของการให้บริการ ทั้งหมดโดยที่ะมีค่าอยู่ระหว่าง 0 ถึง 1 โดยค่าที่คาดหวังคือ 1 โดยที่มีมาตรวัดสำหรับแพ็กเกจ และสำหรับมอดุลาไรเซชัน

- มาตรวัด Index of Inter-Package Interaction

$$IIPU(M) = 1 - \frac{UsesSum(P)}{UsesSum(C)}, IIPE(M) = 1 - \frac{ExtSum(P)}{ExtSum(C)}$$

- มาตรวัด Index of Package Changing Impact

$$IPCI(p) = 1 - \frac{Clients_p(P)}{1 - |P|} \quad , \quad IPCI(M) = \frac{\sum_{p_i \epsilon P} IPCI(p_i)}{|P|}$$

- มาตรวัด Index of Package Goal Focus

$$PF(p) = \frac{\sum_{p_i \epsilon Clients_p} Role(p,p_i)}{|Clients_p(p)|},\ PF(M) = \frac{\sum_{p_i \epsilon P} PF(p_i)}{|P|}$$

- มาตรวัด Index of Package Services Cohesion

$$IPSC(p) = \frac{\sum_{p_i \epsilon Clients_p(p)} CS_{Cohesion}(p,p_i)}{|Clients_p(p)|},$$

$$IPSC(M) = \frac{\sum_{p_i \epsilon P} IPSC(p_i)}{|P|}$$

หลังจากได้ทำการทดลองเพื่อหาตัววัดแล้วคณะผู้วิจัยยังได้นำตัววัดที่มีผู้เสนอก่อนหน้านี้มาเปรียบเทียบผลการทดลอง งานวิจัยนี้ได้นำเสนอตัววัดเพื่อการประเมินการจัดองค์ประกอบของแพ็กเกจ โดยชุดตัววัดการเข้าคู่กันและการเกาะกลุ่มกันของซอฟต์แวร์เชิงวัตถุ

2) งานวิจัย “Software Re-modularization based on Structural and Sematic Metrics” [28]

งานวิจัยนี้ได้เสนอเทคนิคการวัดเพื่อการรีมอดุลาไรเซชัน สามารถช่วยแนะนำการปรับปรุงโครงสร้างของแพ็กเกจโดยพยายามเพิ่มความสอดคล้องกันหรือการเกาะกลุ่มภายในมอดูล ซึ่งงานวิจัยนี้เป็นการศึกษาความสัมพันธ์ระหว่างคลาสในแพ็กเกจ โดยใช้พิจารณาความสัมพันธ์เชิงโครงสร้างและความหมาย เพื่อกำหนดห่วงโซ่ที่เก็บความสัมพันธ์ของคลาสที่มีระหว่างกัน การกำหนดห่วงโซ่ใหม่ที่เกิดขึ้นนั้นคือการสร้างแพ็กเกจใหม่ให้คลาสภายในระบบ โดยรวมแนวคิดการเพิ่งพากันระหว่าง 2 คลาส ให้เป็นสมการต่อไปนี้ โดยเป็นการหาการคัพปลิงระหว่าง 2 คลาส

ต่อไปนี้ โดยเป็นการหาการคัพปลิงระหว่าง 2 คลาส

$$CCBC(c_k, c_j) = \frac{\sum_{l=1}^{r} CCMC(m_{kl}, c_j)}{r}$$

เมื่อ *CCMC* คือ ค่าการเข้าคู่ระหว่างเมทธอดกับคลาส

โดยการสร้างห่วงโซ่ของคลาสสร้างโดย มี 3 ขั้นตอนคือ 1. การสร้างเมตริกซ์ความสัมพันธ์ระหว่างคลาส 2. การเลือกแพ็กเกจใหม่โดยจะพยายามไม่ให้เกิดแพ็กเกจที่ใหญ่เกิน 3. คำนวณมาตรวัดการเข้าคู่ ระหว่างห่วงโซ่แต่ละตัว โดยที่ใช้สมการต่อไปนี้เพื่อประเมินค่าการเข้าคู่

$$Coipling(Ch_i, Ch_j) = \frac{1}{N} \sum_{c_i \in Ch_i, c_j \in Ch_j} Coupling(c_i, c_i)$$

ซึ่งจากงานวิจัยนี้ผู้วิจัยสามารถนำวิธีการไปเพื่อปรับปรุงซอฟต์แวร์ทมี่มีคุณภาพของแพ็กเกจต่ำได้

3) งานวิจัย “MoJo: A Distance Metric for Software Clustering” [29]

งานวิจัยนี้ได้เสนอตัววัดใหม่ ที่มีชื่อว่า MOJO ใช้สำหรับวัดและประเมินวิธีการมอดุลาไรเซชัน โดยที่จะวัดระยะห่างหรือความแตกต่างระหว่าง 2 วิธีการจัดกลุ่ม ซึ่งเสนอวิธีการคำนวณโดยมีตัวดำเนินการที่พิจารณา 2 ประเภท ที่ได้กำหนดขึ้นมาคือ Mo มาจากคำว่า Moving นั่นคือการย้ายโหนดจากกลุ่มหนึ่งไปยังอีกกลุ่มหนึ่ง และอีกตัวหนึ่งคือ Jo มีที่มาจากคำว่า Joining คือการร่วมกันของสองมอดูล โดยตัวดำเนินการเหล่านี้จะใช้วิธีการนับโดยที่นับตัวดำเนินการโดยให้มีค่าน้อยที่สุดที่จะสามารถแปลงให้มอดูลหนึ่งคล้ายกับอีกมอดูล โดยได้เสนอโมเดลการคำนวนไว้ดังต่อไปนี้

$$MoJo(A, B) = min(mno(A, B), mno(B, A))$$

และมีการเพิ่มประสิทธิภาพในการคำนวนโดยใช้อัลกอลิทึมฮิวริสติกส์ในการคำนวน และได้เสนอตัววัดสำหรับวัดคุณภาพการแบ่งกลุ่มไว้ดังต่อไปนี้

$$Q(M) = (1 - \frac{MoJo(A, B)}{n}) \times 100\%$$

เมื่อ n คือ จำนวนโหนด, M คือ อัลลอริทึมที่ใช้จัดกลุ่ม

การศึกษางานวิจัยที่เกี่ยวข้องทำให้สามารสร้างกรอบแนวคิดวิธีการวิจัย ซึ่งจากที่ได้ศึกษาทบทวนงานวิจัยที่ได้กล่าวมา ทำให้ผู้วิจัยสามารถนำความรู้ที่เกี่ยวข้องเหล่านั้นมาศึกษาให้เกิดประประโยชน์แก่งานวิทยานิพนธ์ได้โดยสรุป คือ การรีมอดุลาไรเซชันซอฟต์แวร์เป็นวิธีการที่มีวิวัฒนาการมายาวนานมีผู้ศึกษาวิจัยมากมายที่ทำการศึกษาเพื่อพัฒนาปรับปรุงรูปแบบวิธีการ เพื่อให้สามารถทำการรีมอดุลาไรเซชันให้มีประสิทธิภาพสูงสุด การค้นหาและนำเสนอวิธีการใหม่ๆ เป็นส่วนหนึ่งของรูปแบบงานวิจัยที่ผ่านมา โดยมีการเน้นไปที่วิธีจัดกลุ่มที่เหมาะสมภายในระบบย่อยโดยที่ พิจารณา

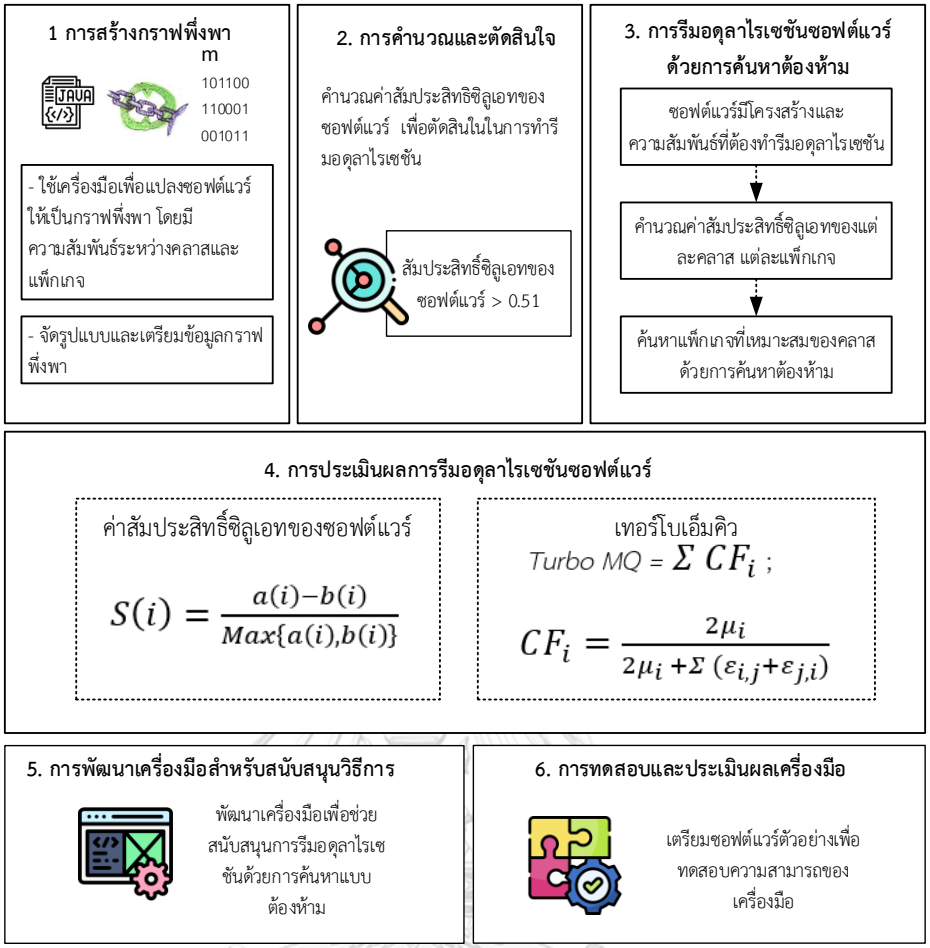
จากมาตรวัดหลัก 2 ตัวคือ มาตรวัดการเข้าคู่ (Coupling Metrics) และมาตรวัดการเกาะกลุ่ม (Cohesion Metrics) โดยมีเป้าหมายในการปรับปรุงให้ซอฟต์แวร์หลักจากรีมอดุลาไรเซชันแล้วมีค่ามาตรวัดทั้งสองตัวนี้มีค่าเหมาะสมที่สุด นอกจากการเสนอวิธีการจัดกลุ่มเพื่อหาวิธีการปรับปรุงตัววัดนี้แล้ว ยังมีงานวิจัยอีกส่วนหนึ่งที่พยายามเสนอวิธีในการวัดเพื่อให้เหมาะสมและครบถ้วนมากที่สุดภายใต้จุดประจุดคือการเพิ่มการเกาะกลุ่มและลดการเข้าคู่กัน ดังนั้นจากการศึกษางานวิจัยเหล่านี้ทำให้ผู้วิจัยได้เสนอวิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งเป็นวิธีการค้นหารูปแบบหนึ่งที่ใช้แก้ปัญหาการหาคำตอบที่เหมาะสมได้ดีเพราะมีความยืดหยุ่นในการใช้งาน และได้นำตัววัด คือ ค่าสัมประสิทธิ์ซิลูเอทมาใช้เพื่อวัดและประเมินผลการทำรีมอดุลาไรเซชัน เนื่องจากการใช้ค่าสัมประสิทธิ์ซิลูเอทนี้สามารถตีความหมายค่าผลลัพธ์ที่ได้จากการคำนวณได้ ทำให้สามารถช่วยให้พิจารณาตัดสินใจสำหรับการทำรีมอดุลาไรเซชันได้จากเกณฑ์เดียวกัน ดังนั้นวิทยานิพนธ์นี้จึงเสนอวิธีการรีมอดุลาไรเซชันนี้ เพื่อที่สามารถค้นหาและจัดองค์ประกอบซอฟต์แวร์ เพื่อปรับปรุงคุณภาพซอฟต์แวร์ให้ดีขึ้นได้

บทที่ 3
วิธีการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม

วิทยานิพนธ์นี้เสนอการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยวิธีการค้นหาต้องห้าม ระบบซอฟต์แวร์นั้นภายในจะมีองค์ประกอบที่ถูกจัดสรรเป็นแพ็กเกจให้การทำงานร่วมกัน ภายในแพ็กเกจจะประกอบด้วยคลาส ที่ทำหน้าที่ร่วมกัน ซึ่งเมื่อซอฟต์แวร์มีการใช้งานและได้ถูกปรับปรุงแก้ไข ทำให้ลักษณะองค์ประกอบภายในนั้นเปลี่ยนแปลงไป คลาสบางส่วนอาจจะอยู่ในแพ็กเกจที่ไม่เหมาะสม หรือซอฟต์แวร์นั้นไม่ได้มีออกแบบเพื่อจัดสรรองค์ประกอบที่ดี มีโครงสร้างที่ไม่มีความสัมพันธ์กัน ซึ่งจะส่งผลต่อสภาพมอดุลาร์ของซอฟต์แวร์ หากซอฟต์แวร์มีสภาพมอดุลาร์ต่ำคือหนึ่งในตัวบ่งชี้ถึงคุณภาพที่ไม่เหมาะสมของซอฟต์แวร์ จึงต้องมีการทำรีแฟคทอริงเพื่อให้ได้คุณภาพที่เหมาะสม ซึ่งแนวคิดในการจัดองค์ประกอบซอฟต์แวร์ใหม่ โดยจะจัดสรรให้ซอฟต์แวร์มีการจัดกลุ่มคลาสในแพ็กเกจอย่างเหมาะสม มีสภาพมอดุลาร์ที่ดีขึ้นโดยประเมินจากตัววัดเทอร์โบเอ็มคิว ค่าสัมประสิทธิ์ซิลูเอทอยู่ในเกณฑ์ที่สามารถยอมรับได้ซึ่งจะมีการวัดเพื่อเปรียบเทียบทั้งก่อนและหลัง วิธีรีมอดุลาไรเซชันซอฟต์แวร์โดยใช้การค้นหาต้องห้ามและใช้สัมประสิทธิ์ซิลูเอท ซึ่งจะช่วยประเมินและตัดสินใจในการทำรีมอดุลาไรเซชัน โดยที่การค้นหาต้องห้ามเป็นการค้นหาแบบฮิวริสติกรูปแบบหนึ่ง ที่สามารถออกแบบเพื่อการค้นหาคำตอบที่ดีที่สุด การค้นหาต้องห้ามจะมีลักษณะพิเศษคือจะมีการใช้หน่วยความจำในการจดจำเส้นทางที่ดีไว้ได้ นั่นคือรายการต้องห้าม รายการต้องห้ามมีไว้เพื่อบันทึกเส้นทางนั้นๆ ไว้และเพื่อที่จะไม่ต้องค้นบริเวณเดิมซ้ำอีก จึงเป็นความพิเศษของการค้นหาต้องห้าม ทำให้เกิดความหลากหลายในการค้นหา วิทยานิพนธ์นี้ใช้การค้นหาต้องห้ามในการรีมอดุลาไรเซชันเพื่อที่จะค้นหารูปแบบคำตอบที่ดีที่สุด

การค้นหาต้องห้ามสำหรับการรีมอดุลาไรเซชันเริ่มต้นค้นหาจากคลาสที่ไม่เหมาะสมกับแพ็กเกจปัจจุบันมากที่สุด เพื่อหาแพ็กเกจที่มีความเหมาะสมกว่าให้คลาสนั้น ซึ่งมีเป้าหมายเพื่อปรับปรุงให้ระบบซอฟต์แวร์มีการจัดองค์ประกอบที่ดีขึ้น โดยที่จะใช้คุณสมบัติพิเศษของการค้นหาต้องห้าม คือรายการต้องห้าม บันทึกเส้นทางการย้ายแพ็กเกจของคลาส เพื่อเป็นข้อมูลสำหรับการตัดสินใจเพื่อทำการปรับปรุงคุณภาพของซอฟต์แวร์ โดยจะพิจารณาจากค่าสัมประสิทธิซิลูเอท

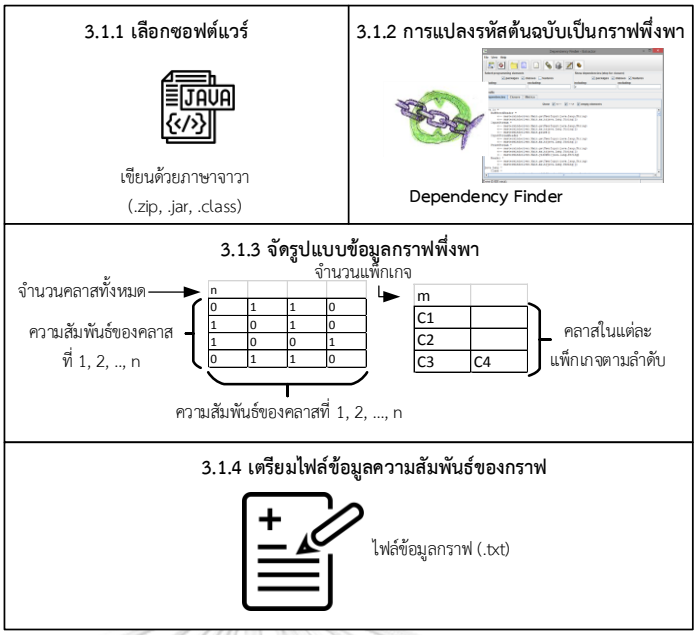
วิทยานิพนธ์นี้เสนอวิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามโดยแบ่งออกเป็น 6 ขั้นตอน ได้แก่ การสร้างกราฟพึ่งพา การคำนวณค่าสัมประสิทธิ์ซิลูเอทเพื่อตัดสินใจ การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม การประเมินผลการรีมอดุลาไรเซชัน การพัฒนาเครื่องมือสำหรับสนับสนุนวิธีการ และการประเมินผลเครื่องมือ ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 ภาพรวมของขั้นตอนในการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

3.1 การสร้างกราฟพึ่งพา

ขั้นตอนแรกของการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม คือการสร้างกราฟพึ่งพา โดยขั้นตอนนี้จะแบ่งออกเป็น 4 ส่วน ดังรูปที่ 3.2 เริ่มจากการเลือกซอฟต์แวร์สำหรับการทดสอบ ซึ่งเป็นซอฟต์แวร์ที่พัฒนาด้วยภาษาจาวาและผ่านการคอมไพลแล้วจะนำมาแปลงจากรหัสต้นฉบับมาเป็นกราฟพึ่งพาโดยใช้เครื่องมือภายนอก นั่นคือดีเพนเดนซีไฟน์เดอร์ ใช้ในส่วนการแปลงรหัสต้นฉบับเป็นกราฟพึ่งพา ส่วนจัดรูปแบบข้อมูลกราฟพึ่งพา และส่วนในการจัดเตรียมข้อมูลความสัมพันธ์ของกราฟเพื่อจะนำเอาข้อมูลกราฟนี้เตรียมเป็นข้อมูลสำหรับนำเข้าเครื่องมือ เพื่อทำการตัดสินใจในการรีมอดุลาไรเซชัน ซึ่งวิทยานิพนธ์นี้กำหนดกฎเกณฑ์เบื้องต้นสำหรับซอฟต์แวร์ที่จะนำมาใช้ในการทดสอบนี้ คือ ซอฟต์แวร์จะต้องสามารถแปลงเป็นกราฟพึ่งพาและสามารถแสดงข้อมูลเป็นลักษณะเมตริกซ์พึ่งพาได้



รูปที่ 3.2 การสร้างกราฟพึ่งพา

3.1.1 การเลือกซอฟต์แวร์

ส่วนนี้เป็นการเลือกซอฟต์แวร์ตัวอย่างสำหรับใช้ในการทดสอบ โดยรูปแบบซอฟต์แวร์จะต้องอยู่ในรูปแบบไฟล์นามสกุล .zip เนื่องจากความต้องการของเครื่องมือ เขียนด้วยภาษาจาวา และมีเงื่อนไขว่าจะต้องประกอบด้วย แพ็กเกจจำนวนตั้งแต่ 3 แพ็กเกจ ภายในมีคลาสที่มีความสัมพันธ์กันตั้งแต่ 3 คลาสขึ้นไป วิทยานิพนธ์นี้ได้เตรียมซอฟต์แวร์ตัวอย่างเพื่อใช้ในการทดสอบทั้งหมด 7 ตัวอย่าง โดยประกอบด้วยซอฟต์แวร์จากโอเพนซอร์สและจากการสร้างกรณีตัวอย่าง โดยมีรายละเอียดดังนี้

ตารางที่ 3.1 ซอฟต์แวร์ตัวอย่างที่ใช้ในการทดสอบและประเมินผล

ลำดับ	ตัวอย่าง	ขนาด	ที่มา
1.	JMeter	3 แพ็กเกจ 47 คลาส	https://github.com/apache/jmeter
2.	JUnit4	3 แพ็กเกจ 79 คลาส	https://github.com/junit-team/junit4
3.	Example 1	3 แพ็กเกจ 8 คลาส	กรณีทดสอบ 1
4.	Example 2	4 แพ็กเกจ 16 คลาส	กรณีทดสอบ 2
5.	Example 3	3 แพ็กเกจ 12 คลาส	กรณีทดสอบ 3
6.	Example 4	3 แพ็กเกจ 13 คลาส	กรณีทดสอบ 4
7.	Example 5	4 แพ็กเกจ 16 คลาส	กรณีทดสอบ 5

3.1.2 การแปลงรหัสต้นฉบับเป็นกราฟพึ่งพา

ส่วนการแปลงรหัสต้นฉบับเป็นกราฟพึ่งพา โดยวิทยานิพนธ์นี้ได้ใช้เครื่องมือภายนอกเพื่อแปลงกราฟพึ่งพา คือ ดีเพนเดนซี ไฟน์เดอร์ (Dependency Finder) พัฒนาโดย Jean Tessier เวอร์ชันที่นำมาใช้ คือ 1.2.1 โดยผลลัพธ์ที่ได้จากเครื่องมือ จะสามารถบันทึกเป็นไฟล์กราฟพึ่งพาที่มีลักษณะเป็นไฟล์นามสกุล .xml ได้ โดยมีลักษณะดังแสดงในรูปที่ 3.3

```
<package confirmed="yes">
    <name>org.gradle.cli</name>
    <class confirmed="yes">
        <name>org.gradle.cli.AbstractCommandLineConverter</name>
        <outbound type="class" confirmed="no">java.lang.Object</outbound>
        <outbound type="class" confirmed="yes">org.gradle.cli.CommandLineConverter</outbound>
        <inbound type="class" confirmed="yes">org.gradle.cli.AbstractPropertiesCommandLineConverter</inbound>
        <feature confirmed="yes">
            <name>org.gradle.cli.AbstractCommandLineConverter.AbstractCommandLineConverter()</name>
            <outbound type="feature" confirmed="no">java.lang.Object.Object()</outbound>
            <inbound type="feature" confirmed="yes">
            org.gradle.cli.AbstractPropertiesCommandLineConverter.AbstractPropertiesCommandLineConverter()
            </inbound>
        </feature>
        <feature confirmed="no">
            <name>org.gradle.cli.AbstractCommandLineConverter.configure(org.gradle.cli.CommandLineParser)</name>
            <inbound type="feature" confirmed="yes">org.gradle.cli.AbstractCommandLineConverter.convert
            (java.lang.Iterable, java.lang.Object)</inbound>
        </feature>
```

รูปที่ 3.3 ตัวอย่างข้อมูล.xml ที่ได้จากเครื่องมือดีเพนเดนซีไฟน์เดอร์

จากรูปที่ 3.3 เป็นส่วนหนึ่งของกราฟพึ่งพาของซอฟต์แวร์ที่นำมาทดสอบคือ JMeter โดยที่ซอฟต์แวร์นี้มีองค์ประกอบภายใน ประกอบด้วย 3 แพ็กเกจ 47 คลาส ซึ่งรูปจากข้อมูลกราฟพึ่งพา .xml ที่ปรากฏนี้เป็นเพียงส่วนหนึ่งของกราฟทั้งหมด จากรูปจะแสดงให้ส่วนของกราฟพึ่งพานี้ คือ ข้อมูลของแพ็กเกจชื่อ org.gradle.cli ภายในมีคลาสและฟีเจอร์ต่างๆที่มีความสัมพันธ์กัน และคลาสที่ปรากฏในรูปคือ คลาสที่ชื่อ AbstractCommandLineConverter โดยที่ข้อมูลของกราฟพึ่งพานี้จะแสดงความสัมพันธ์โดยจะแสดงข้อมูลเป็นลำดับความสัมพันธ์คือ แพ็กเกจ.คลาส จึงทำให้คลาสที่ปรากฏจะแสดงชื่อคลาสเป็น org.gradle.cli.AbstractCommandLineConverter และแสดงให้เห็นความสัมพันธ์กับคลาสอื่นๆคลาสทั้งในแพ็กเกจเดียวกันและต่างแพ็กเกจใด ซึ่งจากรูปคือคลาสนี้มีความสัมพันธ์กับคลาสอื่นๆอยู่ 2 คลาส คือคลาส org.gradle.cli.CommandLineConverter และคลาส org.gradle.cli.AbstractPropertiesCommandLineConverter ซึ่งอยู่ในแพ็กเกจเดียวกัน

3.1.3 การจัดรูปแบบข้อมูลกราฟพึ่งพา

ส่วนการจัดรูปแบบข้อมูลกราฟพึ่งพา หลังจากที่เครื่องมือเพื่อแปลงรหัสต้นฉบับเป็นกราฟพึ่งพาแล้ว ในส่วนนี้คือการจัดรูปแบบข้อมูล โดยนำเอาความสัมพันธ์ของกราฟพึ่งพาที่ได้จากเครื่องมือนั้นมาจัดรูปแบบให้อยู่ในรูปแบบที่กำหนด ซึ่งในวิทยานิพนธ์นี้จะใช้เพียงข้อมูลความสัมพันธ์ระหว่างคลาส (Class relationship) ภายในแพ็กเกจเดียวกันและระหว่างแพ็กเกจโดยที่ไม่ได้

พิจารณาทิศทาง นำไฟล์ที่ได้ขั้นตอนก่อนหน้านี้นำมาสกัดเอาข้อมูลโดยนับคลาสที่มีความสัมพันธ์อยู่ในแพ็กเกจ โดยที่ข้อมูลที่ต้องการนั้นแบ่งเป็น 2 ส่วนคือ ข้อมูลแพ็กเกจ และข้อมูลกราฟโดยที่ข้อมูลทั้งสองส่วนมีลักษณะดังต่อไปนี้

1. ข้อมูลแพ็กเกจ ข้อมูลแพ็กเกจสำหรับการสร้างกราฟพึ่งพาประกอบด้วย จำนวนแพ็กเกจในระบบ และคลาสที่อยู่ในแต่ละแพ็กเกจ ในรูปที่ 3.4 เป็นตัวอย่างสำหรับการจัดรูปแบบข้อมูลโดยที่ส่วนที่แรเงาเป็นเพียงส่วนในการอธิบายเท่านั้น รูปแบบข้อมูลแพ็กเกจเมื่อจัดเรียบร้อยแล้วจะได้เฉพาะส่วนสีขาวเท่านั้น จากรูปที่ 3.4 เป็นรูปแบบการจัดข้อมูลโดยที่ระบบมีจำนวน m แพ็กเกจ โดยในแพ็กเกจที่ 1 มีคลาสจำนวน 3 คลาสปรากฏอยู่ ซึ่งประกอบไปด้วย C1, C2 และ C3 แพ็กเกจที่ 2 มีคลาสอยู่ภายใน 2 คลาสนั่นคือ C4 และ C5 แพ็กเกจที่ 3 มีคลาสอยู่ภายใน 2 คลาสนั่นคือ C6 และ C7 ซึ่งการจัดรูปแบบข้อมูลแพ็กเกจนี้จะเป็นลำดับไปจนถึงลำดับสุดท้าย นั่นคือ แพ็กเกจที่ m มีคลาส 3 คลาสอยู่ภายใน นั่นคือ Cn-2, Cn-1 และ Cn การจัดเรียงคลาสจะไล่เรียงไปตามลำดับแพ็กเกจและคลาส โดยข้อมูลแพ็กเกจที่เตรียมได้ในขั้นตอนนี้จะได้เป็นข้อมูลเพ็กเกจ ที่บอกจำนวนแพ็กเกจ คลาส และจำนวนคลาสในแต่แพ็กเกจ

ลำดับแพ็กเกจที่	m
1	C1 C2 C3
2	C4 C5
3	C6 C7
...	...
m	Cn-2 Cn-1 Cn

รูปที่ 3.4 การจัดรูปแบบข้อมูลแพ็กเกจ

2. ข้อมูลกราฟของซอฟต์แวร์ ข้อมูลกราฟเป็นข้อมูลที่เป็นความสัมพันธ์ระหว่างคลาสในระบบซอฟต์แวร์ ซึ่งมีทั้งความสัมพันธ์ของคลาสในแพ็กเกจเดียวกันและต่างแพ็กเกจ โดยที่ข้อมูลส่วนนี้จะรวบรวมความสัมพันธ์ของคลาสเป็นเมตริกซ์ความสัมพันธ์ของคลาส โดยที่ข้อมูลกราฟสำหรับการสร้างกราฟพึ่งพาสำหรับวิทยานิพนธ์นี้จะประกอบด้วย จำนวนคลาสและความสัมพันธ์ระหว่างคลาส จากรูปที่ 3.3 ได้อธิบายส่วนของข้อมูลที่ได้จากเครื่องมือดีเพนเดนซีไฟน์เดอร์แล้ว ซึ่งจะพิจารณาความสัมพันธ์ที่อยู่ในรูปแบบกราฟพึ่งพา โดยจะเก็บข้อมูลให้เป็นความสัมพันธ์ของคลาสโดยเก็บในโปรแกรมไมโครซอฟต์เอ็กเซล นำข้อมูลจากตารางมาแปลงเป็นเมตริกซ์พึ่งพาโดยที่เมื่อเตรียมข้อมูลได้แล้ว นำข้อมูลทั้งสองตารางนั้นมาแปลงเป็นเมตริกซ์ความสัมพันธ์ โดยแต่ละข้อมูลห่างกันหนึ่งเว้นวรรค ข้อมูลกราฟที่ได้จะมีลักษณะดังต่อไปนี้

รูปที่ 3.5 การเก็บข้อมูลกราฟ จากรูประบบซอฟต์แวร์นี้มีคลาสทั้งหมด 5 คลาส ซึ่งจะเป็นเมตริกซ์การพึ่งพาของคลาส ขนาด 5x5 โดยที่คลาสจะเรียงข้อมูลตามลำดับทั้งหลักและแถว โดยเริ่มจาก C1, C2, C3, C4, C5 โดยที่ภายในจะแสดงความสัมพันธ์ของคลาสโดยที่ หากคลาสที่มีความสัมพันธ์กันจะมีข้อมูลเป็น 1 และไม่มีความสัมพันธ์จะเป็น 0

5

	C_1	C_2	C_3	C_4	C_5
C_1	0	1	1	0	1
C_2	1	0	1	0	0
C_3	1	1	0	0	1
C_4	0	0	0	0	1
C_5	1	0	1	1	0

รูปที่ 3.5 การจัดรูปแบบข้อมูลกราฟ

หากพิจารณาด้วยกราฟพึ่งพาจากรูปที่ 3.3 กราฟพึ่งพาที่แสดงด้วยไฟล์ .xml นั้น จะมีคลาสชื่อ AbstractCommandLineConverter กับคลาสชื่อ CommandLineConverter และคลาสชื่อ AbstractPropertiesCommandLineConverter นั้นมีความสัมพันธ์กันจะแทนข้อมูลในตารางด้วย 1 ส่วนกับคลาสอื่นๆที่ไม่ได้มีความสัมพันธ์กันคือ ไม่ได้ปรากฏในไฟล์ข้อมูลกราฟพึ่งพาในส่วนของคลาส AbstractCommandLineConverter ที่แสดงในข้อ 3.1.2 จะแทนความสัมพันธ์นั้นด้วย 0

3.1.4 การเตรียมไฟล์ข้อมูลความสัมพันธ์ของกราฟ

ส่วนการเตรียมไฟล์ข้อมูลความสัมพันธ์ของกราฟ หลังจากเตรียมข้อมูลกราฟพึ่งพาให้อยู่ในรูปแบบที่ต้องการ โดยประกอบด้วยสองส่วนคือ ข้อมูลแพ็กเกจและข้อมูลกราฟจากขั้นตอนก่อนหน้า จากนั้นนำข้อมูลที่ได้นั้นจัดรูปเตรียมให้อยู่ในรูปข้อมูลอักษร เป็นไฟล์นามสกุล .txt คือข้อมูลแพ็กเกจ (Package.txt) และ ข้อมูลความสัมพันธ์ของคลาสซึ่งคือ ข้อมูลกราฟ (Graph.txt) เป็นเมตริกซ์พึ่งพาระหว่างคลาสในระบบที่เตรียมได้จากขั้นตอนที่ผ่านมาทั้งสองข้อมูล เพื่อจะนำไปใช้ในการวิเคราะห์ผลและทำการค้นหาต่อไป

โดยรายละเอียดของการสร้างกราฟพึ่งพา รวมถึงการใช้เครื่องมือภายนอกเพื่อสร้างกราฟพึ่งพานี้ได้แสดงรายละเอียดไว้ในภาคผนวก ข.

3.2 การคำนวณและการตัดสินใจ

เมื่อได้โครงสร้างซอฟต์แวร์จากกราฟพึ่งพาแล้ว ขั้นตอนนี้จะพิจารณาความสัมพันธ์ของคลาสระหว่างคลาสในแพ็กเกจเดียวกันและความสัมพันธ์กับแพ็กเกจอื่น เพื่อประเมินสภาพมอดุลาร์ของซอฟต์แวร์เพื่อนำไปพิจาณาการทำรีมอดุลาไรเซชัน หากทำการรีมอดุลาร์ไรเซชัน ในขั้นตอนนี้ก็จะวัดความเหมาะสมของคลาสในแพ็กเกจ หากคลาสใดไม่เหมาะสมกับแพ็กเกจในปัจจุบันก็จะนำไปพิจารณาเพื่อหาแพ็กเกจที่ไม่เหมาะสมต่อไป โดยเพื่อประเมินสภาพมอดุลาร์จะทำการคำนวณซอฟต์แวร์จากความสัมพันธ์ภายในของระบบซอฟต์แวร์ โดยที่จะพิจารณาความสัมพันธ์ของคลาสโดยค่าสัมประสิทธิ์ซิลูเอท

3.2.1 การประเมินด้วยค่าสัมประสิทธิ์ซิลูเอท

ค่าสัมประสิทธิ์ซิลูเอทนี้จะถูกนำมาใช้ในวิทยานิพนธ์นี้เพื่อเป็นมาตรวัดในการตัดสินใจ ในการทำรีมอดุลาไรเซชันซอฟต์แวร์ ซึ่งจากตารางที่ 2.1 ซึ่ง Ayaz Isazadeh และคณะ ได้นำเสนอไว้ ซึ่งค่าสัมประสิทธิ์ซิลูเอท มีค่าอยู่ในช่วง -1 ถึง 1 ซึ่งค่าสัมประสิทธิ์ซิลูเอทสามารถตีความได้จากช่วงค่าผลลัพธ์ที่ชี้ถึงลักษณะความสัมพันธ์ในโครงสร้างของซอฟต์แวร์ได้ โดยแบ่งเป็น 4 ช่วงดังที่ได้กล่าวมาแล้ว วิทยานิพนธ์นี้จึงใช้เกณฑ์การตีความของค่าสัมประสิทธิ์ซิลูเอทจากการอ้างอิงตารางที่ 2.1 โดยได้พิจารณาที่ช่วงค่าที่เป็นจุดวิกฤตของค่าสัมประสิทธิ์ซิลูเอท คือ ค่าที่เป็นจุดที่ชี้ว่าระบบซอฟต์แวร์นี้ควรปรับปรุงโครงสร้างการจัดกลุ่มหรือไม่ ด้วยการตัดสินใจเพื่อทำการรีมอดุลาไรเซชันซอฟต์แวร์ที่ค่าดังกล่าว โดยตารางที่ 3.2 ได้แสดงความหมายและการตัดสินใจเพื่อรีมอดุลาไรเซชันซอฟต์แวร์ของวิทยานิพนธ์นี้โดยอ้างอิงการตีความจากตารางที่ 2.1 การประเมินซอฟต์แวร์ในขั้นตอนนี้ เพื่อให้สามารถระบุว่าแพ็กเกจใดควรจะต้องทำการปรับปรุงองค์ประกอบภายใน และเพื่อแสดงว่าคลาสใดในแพ็จเกจที่ไม่เหมาะสมกับแพ็กเกจในปัจจุบันและควรจะย้ายไปยังแพ็จเกจอื่น ดังนั้นวิทยานิพนธ์นี้จึงพิจารณาที่จะทำการรีมอดุลาไรเซชัน ด้วยค่าสัมประสิทธิ์ซิลูเอท ซึ่งจะคำนวณจากความสัมพันธ์ของคลาสทุก ๆ คลาส โดยพิจารณาจากความสัมพันธ์ระหว่างคลาสในแพ็กเกจเดียวกันและต่างแพ็กเกจ

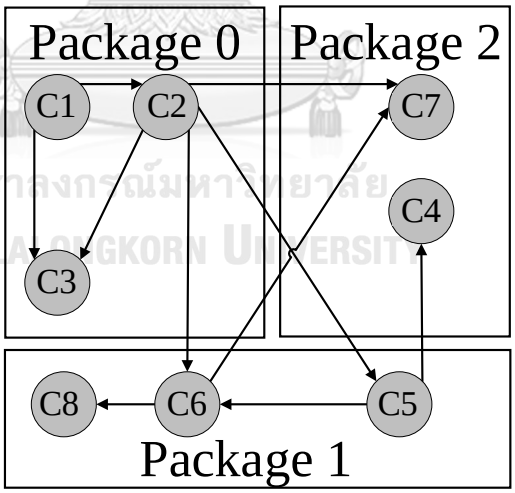
ตารางที่ 3.2 การประเมินค่าสัมประสิทธิ์ซิลูเอทของวิทยานิพนธ์

ค่าสัมประสิทธิ์ซิลูเอท	การตัดสินใจ
ค่าตั้งแต่ 0.51 ขึ้นไป	ระบบซอฟต์แวร์มีโครงสร้างที่มีคุณภาพที่ยอมรับได้
ค่าน้อยกว่า 0.51	ระบบซอฟต์แวร์ต้องทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

ซึ่งเมื่อได้กราฟพึ่งพาแล้ว สำหรับในขั้นตอนนี้มีจุดประสงค์คือ การตัดสินใจในการรีมอดุลาไรเซชัน และหลังจากที่ตัดสินใจได้แล้ว ขั้นตอนถัดไป คือการเลือกจุดเริ่มต้นในการค้นหาต้องห้าม โดยหาโหนดของกราฟเพื่อเริ่มต้นทำการค้นหา โดยมีวิธีการ คือคำนวณค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ทั้งระบบ เพื่อตัดสินใจในการรีมอดุลาไรเซชัน และหากค่าต่ำกว่าเกณฑ์ที่กำหนดไว้ให้เลือกโหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุดเพื่อเป็นจุดเริ่มต้นในการค้นหาต้องห้าม การคำนวณค่าสัมประสิทธิ์ซิลูเอท จะคำนวณจากคลาสและความสัมพันธ์ ซึ่งจะแทนด้วยโหนดและเส้นเชื่อม โดยแบ่งเป็น 3 กรณีดังต่อไปนี้

1) การคำนวณค่าสัมประสิทธิ์ซิลูเอทของคลาส จะคำนวณจากความสัมพันธ์ระหว่างคลาสทั้งในแพ็กเกจเดียวกันและต่างแพ็กเกจ ซึ่งสามารถคำนวณจากสมการที่ (1) ได้ โดยการคำนวณค่าสัมประสิทธิ์ซิลูเอทของคลาสจะคำนวณจากทุกๆ คลาสในซอฟต์แวร์

ต่อไปนี้จะแสดงตัวอย่างการคำนวณค่าสัมประสิทธิ์ซิลูเอทของคลาสจากระบบซอฟต์แวร์หนึ่ง ดังรูปที่ 3.6 เป็นระบบซอฟต์แวร์ ซึ่งประกอบด้วยสามแพ็กเกจ แปดคลาส โดยประกอบด้วยแพ็กเกจศูนย์ แพ็กเกจหนึ่ง และแพ็กเกจสอง ซึ่งแพ็กเกจศูนย์ มีสมาชิกในแพ็กเกจสามคลาส ได้แก่ C1 C2 และ C3 แพ็กเกจหนึ่ง มีสมาชิกสามคลาสคือ C5 C6 และ C8 และแพ็กเกจสอง มีคลาสในแพ็กเกจสองคลาส คือ C4 และ C7 โดยในตัวอย่างนี้จะยกตัวอย่างการคำนวณหาค่าสัมประสิทธิ์ซิลูเอทของคลาส C2 ซึ่งอยู่ในแพ็กเกจศูนย์



รูปที่ 3.6 ตัวอย่างซอฟต์แวร์ที่มี 3 แพ็กเกจ 8 คลาส

จากรูปที่ 3.6 เป็นตัวอย่างซอฟต์แวร์ที่มี 3 แพ็กเกจ 8 คลาส ซึ่งจากรูปที่ 3.6 นี้ จะแสดงตัวอย่างการคำนวณค่าสัมประสิทธิ์ซิลูเอทของคลาส C2 จากรูปจะพบว่า C2 มีความสัมพันธ์กับคลาสทั้งในแพ็กเกจเดียวกันและต่างแพ็กเกจ โดยที่ มีความสัมพันธ์กับคลาสในแพ็กเกจเดียวกัน 2

ความสัมพันธ์คือ ความสัมพันธ์กับ C1 และ C3 ความสัมพันธ์กับคลาสต่างแพ็กเกจ คือ C5 และ C6 ในแพ็กเกจหนึ่ง เป็น 2 ความสัมพันธ์ และ C7 ในแพ็กเกจสองอีก 1 ความสัมพันธ์ ทำให้รวมมีความสัมพันธ์ทั้งหมด 5 ความสัมพันธ์ ดังนั้นเมื่อพิจารณาสมการที่ 1 ค่า *a*(C2) จะได้ $\frac{2}{5}$ และค่า *b*(C2) จะได้ $\frac{2}{5}$ เนื่องจากความสัมพันธ์ที่มากที่สุดคือ ความสัมพันธ์ C2 กับคลาสในแพ็กเกจหนึ่ง เมื่อแทนค่าลงไปในสมการจะได้ดังต่อไปนี้

จากสมการที่ 1

$$S(i) = \frac{a(i)-b(i)}{Max\{a(i),b(i)\}} \qquad (1)$$

เมื่อ

$$a(i) = \frac{\text{จำนวนความสัมพันธ์ระหว่างคลาสในแพ็กเกจเดียวกัน}}{\text{จำนวนความสัมพันธ์ทั้งหมด}}$$

$$b(i) = \frac{\text{จำนวนความสัมพันธ์ระหว่างคลาสต่างแพ็กเกจที่มีมากที่สุด}}{\text{จำนวนความสัมพันธ์ทั้งหมด}}$$

จะได้ $a(C2) = \frac{2}{5}$

$b(C2) = \frac{2}{5}, \frac{1}{5}$

จากสมการที่ 1 จะได้

$$S(C2) = \frac{\frac{2}{5}-\frac{2}{5}}{max\{\frac{2}{5},\frac{2}{5}\}} = 0$$

เพราะฉะนั้น ค่าสัมประสิทธิ์ซิลูเอทของ C2 คือ 0

2) การคำนวณค่าสัมประสิทธิ์ซิลูเอทของแพ็กเกจ ค่าสัมประสิทธิ์ซิลูเอทของแพ็กเกจ *(S_j)* จะสามารถคำนวณได้จาก ค่าเฉลี่ยของค่าสัมประสิทธิ์ซิลูเอทของคลาสทั้งหมดในแพ็กเกจ โดยจะสามารถคำนวณได้โดยใช้สมการดังนี้

$$S_j = \frac{\sum S(i)}{n} \qquad (2)$$

โดยที่ *S(i)* คือ ค่าสัมประสิทธิ์ซิลูเอท ของคลาส *i*

j คือ แพ็กเกจที่ *j*

n คือ จำนวนคลาสทั้งหมดในแพ็กเกจ

3) การคำนวณค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ สัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ *(S)* จะสามารถคำนวณได้จากค่าเฉลี่ยของค่าสัมประสิทธิ์ซิลูเอทของแพ็กเกจทั้งหมดในระบบซอฟต์แวร์ โดยจะสามารถคำนวณได้โดยใช้สมการดังนี้

$$S = \frac{\sum S_j}{m} \qquad (3)$$

โดยที่ S_j คือ ค่าสัมประสิทธิ์ซิลูเอทของแพ็กเกจ *j*

m คือ จำนวนแพ็กเกจทั้งหมดในระบบซอฟต์แวร์

การคำนวณค่าสัมประสิทธิ์ซิลูเอท ในวิทยานิพนธ์นี้เป็นการตัดสินใจเพื่อจะทำการรีมอดุลาไรเซชันซอฟต์แวร์นั้น จะใช้ผลการคำนวณในกรณีที่ 3 ซึ่งซอฟต์แวร์ที่จะทำการรีมอดุลาไรเซชันคือซอฟต์แวร์ที่มีค่าสัมประสิทธิ์ซิลูเอทที่ต่ำกว่า 0.51 และคลาสใดๆ ที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำนั่นหมายถึง คลาสนั้นไม่เหมาะสมกับแพ็กเกจที่อยู่ในปัจจุบัน ควรจะย้ายไปยังแพ็กเกจอื่นๆ ที่มีความเหมาะสมมากกว่า ซึ่งเป็นผลจากการคำนวณและเปรียบเทียบกันกันจากกรณีที่ 1

3.3 รีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม

แนวคิดในการทำรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม จะปรับปรุงโครงสร้างหลังจากที่ได้คำนวณค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์แล้ว พบว่าโครงสร้างของซอฟต์แวร์นั้นมีความสัมพันธ์กันต่ำ เพื่อจะทำการปรับปรุงโครงสร้างของซอฟต์แวร์เพื่อให้มีสภาพมอดุลาร์ที่ดี จะกำหนดฟังก์ชันค้นหาโดยใช้ค่าสัมประสิทธิ์ซิลูเอท และใช้การค้นหาต้องห้ามเพื่อค้นหาคำตอบที่เหมาะสมที่สุดสำหรับการจัดสรรคลาสในแพ็กเกจเพื่อการรีมอดุลาไรเซชัน

3.3.1 การค้นหาต้องห้าม

วิทยานิพนธ์นี้ใช้การค้นหาต้องห้ามเป็นอัลกอริทึมในการค้นหา โดยที่โหนดและเส้นเชื่อมนั้นจะเป็นตัวแทนของคลาสและความสัมพันธ์ และมีตัวแปรที่เกี่ยวข้องดังต่อไปนี้

3.3.1.1 กำหนดตัวแปรที่เกี่ยวข้อง

1. S_0 คือ คำตอบของการค้นหาโหนดเริ่มต้น คือ โหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด

   S_c คือ คำตอบของการค้นหาในโหนดปัจจุบัน

S_N คือ คำตอบของการค้นหาจากโหนดเพื่อนบ้านใกล้เคียง

S_{Best} คือ คำตอบของการค้นหาที่ดีที่สุด ได้จากฟังก์ชันที่การค้นหาที่ 2

2. N(S) คือ โหนดเพื่อนบ้านหรือโหนดใกล้เคียง โดยกำหนดให้เป็นโหนดที่มีความสัมพันธ์กับโหนดที่ค้นหา

3. $S = \{ f_1(s) , f_2(s) \}$ คือ คำตอบของการค้นหา ประกอบด้วยฟังก์ชันในการค้นหา 2 ฟังก์ชัน

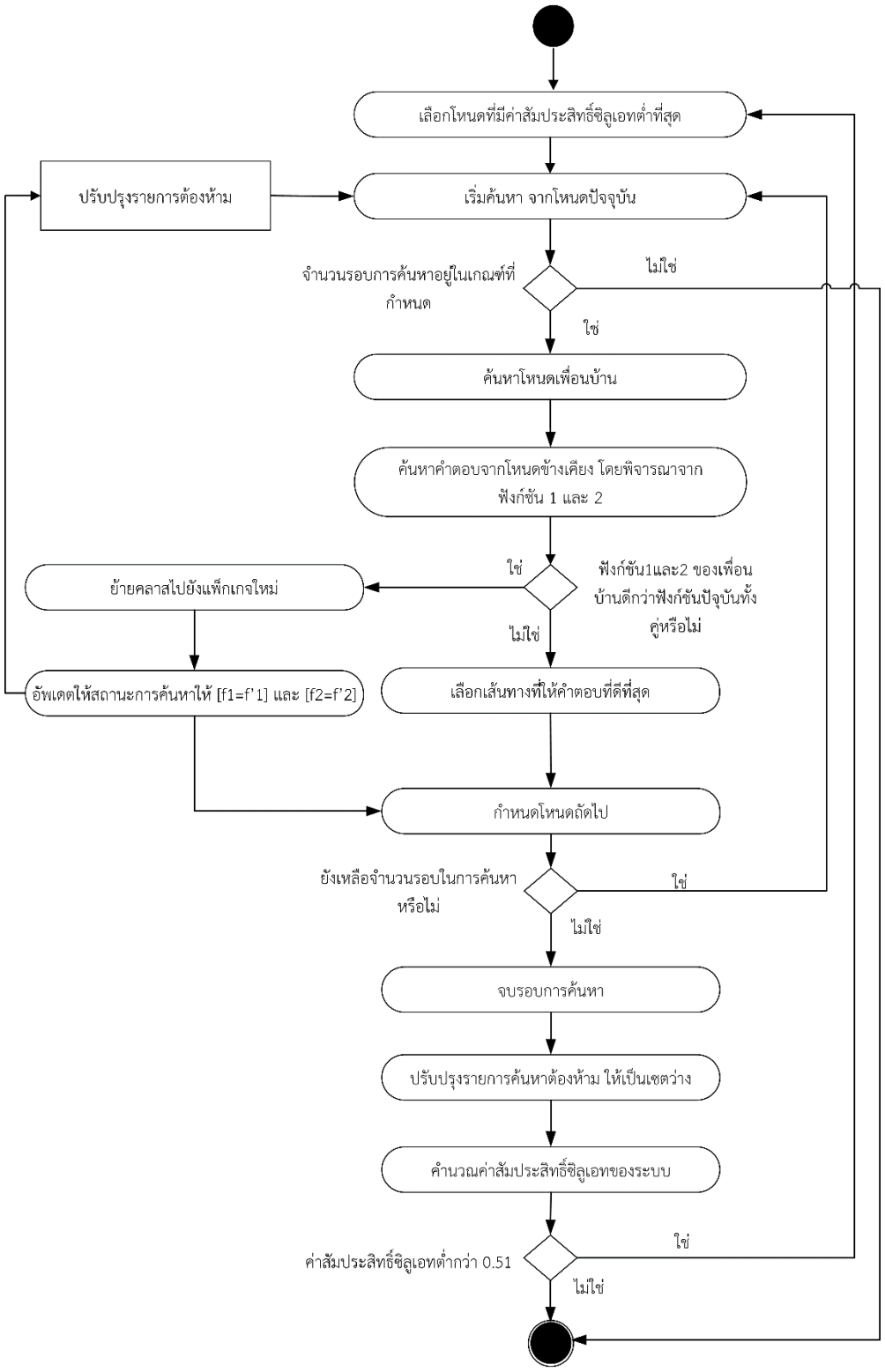
3.1 ฟังก์ชันที่ 1คือ $f_1 = \frac{a(i)-b(i)}{Max\{a(i),b(i)\}}$

3.2 ฟังก์ชันที่ 2 คือ $f_2 = \frac{\sum S_j}{m}$

4. I_{MAX} คือ จำนวนรอบที่มากที่สุดในการค้นหา

3.3.1.2 ขั้นตอนรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม สำหรับงานวิทยานิพนธ์นี้เป็นการนำวิธีการค้นหาต้องห้ามมาประยุกต์เพื่อใช้สำหรับการรีมอดุลาไรเซชัน โดยการค้นหาต้องห้ามนี้เป็นการค้นหาจากโหนดเพื่อนบ้าน คือโหนดที่มีความสัมพันธ์กับโหนดค้นหา เพื่อค้นหาคำตอบที่ดีที่สุดโดยมีการวนซ้ำและปรับปรุงการค้นหา หลังค้นหาจบครบในแต่ละรอบ ไปจนกว่าจะได้ค่าที่ดีที่สุด ซึ่งขั้นตอนเริ่มต้นจากการเลือกโหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด เพื่อเป็นจุดเริ่มต้นในการค้นหา และจะกำหนดให้โหนดนั้นเป็นโหนดการค้นหาปัจจุบันตรวจสอบว่า จำนวนรอบอยู่ในเกณฑ์ที่จะทำการค้นหาอยู่หรือไม่ หากอยู่ในเกณฑ์แล้วจะทำการค้นหาต่อไป โดยค้นหาโหนดเพื่อนบ้าน และค้นหาคำตอบจากโหนดเพื่อนบ้านโดยพิจารณาจากฟังก์ชันที่กำหนด จากนั้นเลือกคำตอบที่ดีที่สุดจากเพื่อนมาเปรียบเทียบการคำตอบของการค้นหาในปัจจุบัน ถ้าค่าของคำตอบเพื่อนบ้านที่ได้นั้นดีกว่าคำตอบในปัจจุบันทั้ง 2 ฟังก์ชั้น จะทำให้มีการย้ายไปอยู่แพ็กเกจเดียวกันกับเพื่อนบ้าน และเก็บโหนดนี้ไว้ในรายการต้องห้าม แต่หากไม่ได้มากกว่าเลือกโหนดนั้นเป็นโหนดการค้นหาถัดไป โดยไม่มีการเปลี่ยนแปลงคำตอบที่ดีที่สุด หลังจากเลือกโหนดถัดไปแล้วพิจาณาว่ายังเหลือรอบการค้นหาหรือไม่ หากยังมีก็ทำการค้นหาจนครบรอบ และทำการปรับปรุงการค้นหาและปรับปรุงให้รายการต้องห้ามว่าง และเริ่มต้นค้นหาใหม่อีกครั้งจนกว่าจะได้คำตอบที่ดีที่สุดและไม่มีการเปลี่ยนแปลง ซึ่งรูปที่ 3.7 จะแสดงภาพรวมลำดับขั้นตอนการทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม



รูปที่ 3.7 แผนภาพลำดับขั้นตอนในการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

จากรูปที่ 3.7 เป็นภาพรวมของขั้นตอนการทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม มีลำดับขั้นตอนดังนี้

1. เลือกโหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด หลังจากที่ตรวจสอบกว่าค่าสัมประสิทธิ์ซิลูเอทของระบบ มีค่าต่ำกว่า 0.51 จะเริ่มการค้นหาต้องห้าม จะพิจารณาโครงสร้างซอฟต์แวร์จากการคำนวณค่าสัมประสิทธิ์ซิลูเอทของระบบซอฟต์แวร์เพื่อตัดสินใจ โดยคำนวณทุกๆ คลาสในระบบซอฟต์แวร์ และเมื่อคำนวณแล้วจะพบว่าคลาสใดๆ มีความไม่เหมาะสมกับแพ็กเกจที่เป็นรูปแบบการจัดสรรในปัจจุบัน โดยจากรูปที่ 3.6 เป็นระบบซอฟต์แวร์ที่มี 3 แพ็กเกจ 8 คลาส ซึ่งสามารถคำนวณค่าสัมประสิทธิ์ซิลูเอทได้ดังตารางที่ 3.3 และ ตารางที่ 3.4

ตารางที่ 3.3 การคำนวณค่าสัมประสิทธิ์ของซอฟต์แวร์ก่อนการรีมอดุลาไรเซชัน

ClassName	C1	C2	C3	C5	C6	C8	C7	C4
Package 0	2	2	2	1	1	0	1	0
Package 1	0	2	0	1	2	1	1	1
Package 2	0	1	0	1	1	0	0	0
Total	2	5	2	3	4	1	2	1
Internal	2	2	2	1	2	1	0	0
External	0	3	0	2	2	0	2	1
ai	1.00	0.40	1.00	0.33	0.50	1.00	0.00	0.00
bi	0.00	0.40	0.00	0.33	0.25	0.00	0.50	1.00
Max{ai,bi}	1.00	0.40	1.00	0.33	0.50	1.00	0.50	1.00
Si	1.00	0.00	1.00	0.00	0.50	1.00	-1.00	-1.00
Sj	0.67			0.50			-1.00	
S	0.056							

ตารางที่ 3.4 สรุปผลการคำนวณค่าสัมประสิทธิ์ซิลูเอท

แพ็กเกจ	จำนวนคลาส	คลาส	ค่าซิลูเอทของแพ็กเกจ	ค่าซิลูเอทของซอฟต์แวร์
แพ็กเกจ 0	3	C1,C2,C3	0.67	<u>0.056</u>
แพ็กเกจ 1	3	C5,C6,C8	0.5	
แพ็กเกจ 2	2	C4,C7	-1	

จากตารางที่ 3.4 พบว่าค่าสัมประสิทธิ์ซิลูเอทของระบบซอฟต์แวร์มีค่าต่ำกว่าเกณฑ์ที่กำหนดนั่นคือ 0.51 ระบบซอฟต์แวร์นี้มีผลการคำนวณเพียง 0.056 จึงต้องทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม และโหนด C4 และ C7 มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุดและอยู่ในแพ็กเกจสอง

2. การค้นหา การค้นหาเริ่มต้นที่ โหนด i ที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด ซึ่งสำหรับการค้นหาต้องห้ามของงานวิทยานิพนธ์นี้เป็นการค้นหาต้องห้ามโดยพิจารณาที่โหนด และความสัมพันธ์กับโหนดเพื่อนบ้าน จึงกำหนดให้โหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุดนั้นเป็นโหนดเริ่มต้นการค้นหา นั่นคือ Ni เป็นโหนดปัจจุบัน จากซอฟต์แวร์รูปที่ 3.6 และตารางที่ 3.3 จะพบว่ามีโหนดที่มีค่าต่ำมากที่สุดถึงค่า -1 คือ C4 และ C7 จึงเริ่มต้นการค้นหาที่โหนด C4 หรือ C7

3. ตรวจสอบเงื่อนไข ณ โหนด Ni หรือโหนดปัจจุบันต้องตรวจสอบก่อนจะเริ่มต้นการค้นหาได้หรือไม่ โดยมี 2 เงื่อนไข

- หากค่าสัมประสิทธิ์ซิลูเอทของระบบซอฟต์แวร์ (S) น้อยกว่า 1 สามารถค้นหาได้
- หากการวนซ้ำ น้อยกว่าจำนวนโหนดจะสามารถค้นหาได้

เงื่อนไขการหยุดค้นหา คือ เมื่อผลการค้นหาอยู่ในเกณฑ์ที่กำหนด หรือ ผลการค้นหานั้นไม่มีการเปลี่ยนแปลง

4. การค้นหาและรวบรวมโหนดเพื่อนบ้าน การรวบรวมโหนดเพื่อนบ้าน โดยที่โหนดที่เป็นเพื่อนบ้านนั้นต้องไม่อยู่ในรายการต้องห้าม จากรูปที่ 3.6 เป็นตัวอย่างระบบซอฟต์แวร์ โดยรูปแสดงให้เห็นเพื่อนบ้านของโหนด C7 คือโหนดที่มีเส้นเชื่อมระหว่างโหนด C7 กับโหนดนั้นๆ ซึ่งจากภาพมีทั้งหมด 2 โหนด คือ โหนด C2 ในแพ็กเกจศูนย์ และ โหนด C6 ในแพ็กเกจหนึ่ง โดยที่แต่ละโหนดนั้นอยู่คนละแพ็กเกจ

5. การคำนวณ เพื่อเลือกเส้นทาง โดยมีตัวแปรที่เกี่ยวข้องดังต่อไปนี้

S_i คือ ค่าสัมประสิทธิ์ซิลูเอทของ โหนด i

S_{ij} คือ ค่าสัมประสิทธิ์ซิลูเอทของ โหนด i หลังจากย้ายไปแพ็กเกจของโหนด j

F_i คือ ค่าสัมประสิทธิ์ซิลูเอทของระบบในปัจจุบัน

F_{ij} คือ ค่าสัมประสิทธิ์ซิลูเอทของระบบหลังจากที่โหนด i ย้ายไปแพ็กเกจของโหนด j

- คำนวณคำตอบของการค้นหาโดยฟังก์ชัน 1 และ 2

  คือ การคำนวณค่า $S_{ij1}, S_{ij2}..., S_{ijn}$ และ $F_{ij1}, F_{ij2},..., F_{ijn}$
- เลือกโหนด j_n ที่ให้คำตอบ S_{jn}, F_{jn} ที่ดีที่สุด
- ถ้าไม่มีโหนดสำหรับการเดินแล้ว จะเลือกโหนดปัจจุบันเป็นคำตอบ

จากเงื่อนไขที่กล่าวมาจะทำให้สามารถคำนวณคำตอบของการค้นหาในขั้นตอนนี้ได้ดังนี้ จากรูปที่ 3.6 สามารถแสดงตัวอย่างการค้นหาต้องห้ามได้ดังตารางที่ 3.5

ตารางที่ 3.5 ตัวอย่างการเลือกเส้นทางของการค้นหาต้องห้าม

โหนดปัจจุบัน	**รายการโหนดเพื่อนบ้าน**	**ฟังก์ชันการค้นหาของโหนดเพื่อนบ้าน**	**ฟังก์ชันการค้นหาของโหนดปัจจุบัน**	**คำตอบที่ดีที่สุด**	**รายการต้องห้าม**
C4	-	-	S(C4) = -1 F(C4) = 0.056	F(C4) = 0.056	-
C4	C5	**S(45) = 1** **F(45) = 0.1389**	S(C4) = -1 F(C4) = 0.056	F(45) = 0.1389	C4
C5	C2	S(52) = -0.5 F(52) = -0.264	S(C5) = 0.5 F(C5) = 0.1389	F_{best} = 0.1389	C4
	C6	**S(56) = 0.5** **F(56) = 0.1389**			
C6	C2	S(62) = -0.5 F(62) = -0.208	S(C6) = 0.5 F(C6) = 0.1389	F_{best} = 0.1389	C4
	C7	S(67) = -0.5 F(67) = 0.139			
	C8	**S(68) = 0.5** **F(68) = 0.1389**			
C8	C6	**S(86) = 0.5** **F(86) = 0.1389**	S(C8) = 1 F(C8) = 0.1389	F_{best} = 0.1389	C4
C6	C2	S(62) = -0.5 F(62) = -0.208	S(C6) = 0.5 F(C6) = 0.1389	F_{best} = 0.1389	C4
	C7	**S(67) = -0.5** **F(67) = 0.139**			
C7	C2	**S(72) = 0** **F(72) = 0.604**	S(C7) = -1 F(C7) = 0.1389	F_{best} = 0.604	C4,C7
	C6	S(76) = 0 F(76) = 0.594			

ตารางที่ 3.5 ตัวอย่างการเลือกเส้นทางของการค้นหาต้องห้าม (ต่อ)

โหนดปัจจุบัน	รายการโหนดเพื่อนบ้าน	ฟังก์ชันการค้นหาของโหนดเพื่อนบ้าน	ฟังก์ชันการค้นหาของโหนดปัจจุบัน	คำตอบที่ดีที่สุด	รายการต้องห้าม
C2	C5	S(25) = 0 F(25) = 0.02	S(C2) = 0.33 F(C2) = 0.604	F_{best} = 0.604	C4,C7
	C6	**S(26) = 0** **F(26) = 0.02**			
C6	C2	**S(62) = 0** **F(62) = 0.092**	S(C6) = 0 F(C6) = 0.604	F_{best} = 0.604	C4,C7
	C7	S(67) = 0 F(67) = 0.092			
	C5	S(65) = 0 S(65)= 0.604			

จากผลลัพธ์จากตารางที่ 3.5 การเลือกเส้นทางของโหนดไม่ว่าโหนดใดก็ตามนั้น การย้ายทุกๆ ครั้งจะมีผลกระทบต่อภาพรวมของระบบนั่นคือค่าในฟังก์ชันที่ 2 จะมีการเปลี่ยนแปลง การค้นหานี้จึงเป็นการค้นหาทั้งในส่วนที่เป็นค่าที่ดีที่สุดในเชิงท้องถิ่นคือค่าของโหนด และค่าที่ดีที่สุดของระบบซอฟต์แวร์คือค่าสัมประสิทธิ์ซิลูเอทของทั้งระบบ โดยจากตารางด้านบนสามารถอธิบายการค้นหาในแต่ละรอบได้ดังต่อไปนี้

1. โหนด C4 มีทางเลือกในการย้ายอยู่ 1 ทางเลือกคือ โหนด C5 การที่ C4 ย้ายไปอยู่ในแพ็กเกจเดียวกันกับ C5 เป็นการเปลี่ยนแพ็กเกจทำให้ C4 มีค่า ความสัมพันธ์ภายในที่เปลี่ยนไป ทำให้ค่าฟังก์ชันที่1 เปลี่ยนจาก -1 เป็น 1 และค่าฟังก์ชันที่ 2 จาก 0.056 เป็น 0.1389 ซึ่งเป็ค่าที่ดีขึ้นทั้งสองฟังก์ชัน จึงทำให้เกิดการย้ายแพ็กเกจตามโหนด C5 และเก็บ C4 ที่ย้ายจากแพ็กเกจสอง ไปแพ็กเกจหนึ่งไว้ในรายการต้องห้าม และเปลี่ยนให้ฟังก์ชันการค้นหาปัจจุบันเป็นค่าฟังก์ชันการค้นหานี้ และโหนดการค้นหาต่อไปคือโหนด C5

2. โหนด C5 มีทางเลือกที่เลือกได้อยู่ 2 ทางเลือกคือ โหนด C2 และ C6 เมื่อเปรียบเทียบฟังก์ชันการค้นหาแล้วพบว่าทางเลือก C6 เป็นทางเลือกที่ดีที่สุด แต่ค่าฟังก์ชันการค้นหาไม่ได้ดีกว่าจึงทำให้เลือกโหนด C6 เป็นโหนดถัดไปและทำการค้นหาต่อไปโดยไม่มีการย้ายแพ็กเกจ

3. โหนด C6 มีทางเลือกที่เลือกได้อยู่ 3 ทางเลือกคือ โหนด C2 C7และ C8 เมื่อเปรียบเทียบฟังก์ชันการค้นหาแล้วพบว่าทางเลือก C8 เป็นทางเลือกที่ดีที่สุด แต่ค่าฟังก์ชันการค้นหาไม่ได้ดีกว่าจึงทำให้เลือกโหนด C8 เป็นโหนดถัดไปและทำการค้นหาต่อไปโดยไม่มีการย้ายแพ็กเกจ

4. โหนด C8 มีทางเลือกที่เลือกได้อยู่ 1 ทางเลือกคือ โหนด C6 เพียงโหนดเดียว และการเลือกโหนด C6 จะเป็นการเลือกซ้ำในแพ็กเกจเดียวกันจะทำให้การเลือกครั้งต่อไปจะไม่สามารถเลือกโหนดที่อยู่ในแพ็กเกจเดียวกันได้อีก ซึ่ง C6 จะเป็นโหนดถัดไปและทำการค้นหาต่อไปโดยไม่มีการย้ายแพ็กเกจ

5. โหนด C6 มีทางเลือกที่เลือกได้อยู่ 2 ทางเลือกคือ โหนด C2 และ C7 เมื่อเปรียบเทียบฟังก์ชันการค้นหาแล้วพบว่าทางเลือก C7 เป็นทางเลือกที่ดีที่สุด แต่ค่าฟังก์ชันการค้นหาไม่ได้ดีกว่าจึงทำให้เลือกโหนด C7 เป็นโหนดถัดไปและทำการค้นหาต่อไปโดยไม่มีการย้ายแพ็กเกจ

6. โหนด C7 มีทางเลือกที่เลือกได้อยู่ 2 ทางเลือกคือ โหนด C2 และ C6 เมื่อเปรียบเทียบฟังก์ชันการค้นหาแล้วพบว่าทางเลือก C2 เป็นทางเลือกที่ดีที่สุด และค่าฟังก์ชันการค้นหาดีกว่าฟังก์ชันการค้นหาในปัจจุบันจึงทำให้เลือกโหนด C2 เป็นโหนดถัดไปซึ่งทำให้ค่าฟังก์ชันที่1 เปลี่ยนจาก -1 เป็น 0.5 และค่าฟังก์ชันที่ 2 จาก 0.1389 เป็น 0.604 ซึ่งเป็ค่าที่ดีขึ้นทั้งสองฟังก์ชัน จึงทำให้เกิดการย้ายแพ็กเกจตามโหนด C2 และเก็บ C7 ที่ย้ายจากแพ็กเกจสอง ไปแพ็กเกจศูนย์ไว้ในรายการต้องห้าม และเปลี่ยนให้ฟังก์ชันการค้นหาปัจจุบันเป็นค่าฟังก์ชันการค้นหานี้ และโหนดการค้นหาต่อไปคือโหนด C2

7. โหนด C2 มีทางเลือกที่เลือกได้อยู่ 2 ทางเลือกคือ โหนด C5 และ C6 เมื่อเปรียบเทียบฟังก์ชันการค้นหาแล้วพบว่าทางเลือกทั้งสองทางเลือกมีค่าเท่ากันจึงเลือกฟังก์ชันใดก็ได้ ซึ่งเลือก C6 แต่ค่าฟังก์ชันการค้นหาไม่ได้ดีกว่าจึงทำให้เลือกโหนด C6 เป็นโหนดถัดไปและทำการค้นหาต่อไปโดยไม่มีการย้ายแพ็กเกจ

8. โหนด C6 มีทางเลือกที่เลือกได้อยู่ 1 ทางเลือกคือ โหนด C2 เพราะ C7 อยู่ในรายการต้องห้ามและ C5 อยู่ในแพ็กเกจเดียวกัน จึงเลือก C2 เป็นทางเลือกที่ดีที่สุด แต่ค่าฟังก์ชันการค้นหาไม่ได้ดีกว่าจึงทำให้เลือกโหนด C2 เป็นโหนดถัดไปและจบการค้นหาเพราะครบจำนวนรอบที่กำหนด

เมื่อค้นหาจนครบจำนวนรอบที่กำหนดแล้ว จะทำการปรับปรุงการค้นหา เพื่อจะทำการค้นหาซ้ำอีกครั้งโดยจะปรับปรุงรายการต้องห้ามให้ว่าง และเริ่มทำการค้นรอบใหม่อีกครั้ง โดยจะค้นหาต่อจากเดิม โดยมีกระบวนการค้นหาเหมือนกับการค้นหาครั้งแรกคือเริ่มต้นจากตัดสินใจว่าซอฟต์แวร์ต้องการทำรีมอดุลาไรเซชันหรือไม่ และหากต้องการจะทำการค้นหาโดยเริ่มต้นจากโหนดที่มีค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด ซึ่งจะทำการค้นหาซ้ำจนกว่าจะได้ผลลัพธ์ที่ดีที่สุด ตามเงื่อนไขที่กำหนด

3.4 การประเมินผลการรีมอดุลาไรเซชัน

การประเมินผลจากการเปรียบเทียบผลการประเมินจากสัมประสิทธิ์ซิลูเอทและค่าเทอร์โบเอ็มคิว โดยมีเป้าหมายเพื่อจะปรับปรุงสภาพมอดุลาร์ของซอฟต์แวร์ โดยที่ผลลัพธ์ที่คาดหวัง คือ หลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามแล้วมีค่ามากกว่า 0.51

3.4.1 การประเมินผลด้วยสัมประสิทธิ์ซิลูเอท

หลังการทำรีมอดุลาไรเซชันนั้นจะต้องมีผลลัพธ์ที่ดีกว่าก่อนทำการรีมอดุลาไรเซชัน การวัดคุณภาพของการรีมอดุลาไรเซชันนั้นจะทำ 2 ครั้ง เพื่อเป็นการเปรียบเทียบเทียบคุณภาพได้จาก 2 กรณีดังนี้

(1) การวัดคุณภาพก่อนการรีมอดุลาไรเซชัน เพื่อตรวจสอบคุณภาพของซอฟต์แวร์ในสถานการณ์ปัจจุบันเพื่อทำการรีมอดุลาไรเซชัน มีค่าน้อยกว่า 0.51

(2) การวัดคุณภาพหลังการรีมอดุลาไรเซชัน เพื่อตรวจสอบคุณภาพของซอฟต์แวร์หลังการเปลี่ยนแปลงการจัดองค์ประกอบภายในควรมีค่าตั้งแต่ 0.51 ขึ้นไป

3.4.1 การประเมินผลด้วยเทอร์โบเอ็มคิว

หลังการทำรีมอดุลาไรเซชันนั้นจะต้องมีผลลัพธ์ค่าเทอร์โบเอ็มคิวที่ดีกว่าก่อนทำการรีมอดุลาไรเซชัน คือมีค่าผลต่างที่เป็นบวก

ดังนั้นเมื่อนำไปเทียบค่าสัมประสิทธิ์ซิลูเอทในตารางที่ 3.2 แล้วโครงสร้างซอฟต์แวร์หลังทำการรีมอดุลาไรเซชันจะต้องดีกว่าก่อนทำ และจากค่าจากเทอร์โบเอ็มคิวหลังทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามจะต้องเพิ่มขึ้น

3.5 การพัฒนาเครื่องมือสำหรับสนับสนุนวิธีการ

การพัฒนาเครื่องมือเพื่อช่วยสำหรับการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามจะถูกพัฒนาด้วยภาษาจาวา เป็นโปรแกรมอรรถประโยชน์ (Utility program) เพื่อเป็นเครื่องมือช่วยสนับสนุนวิศวกรซอฟต์แวร์ใช้ตัดสิน สำหรับการดำเนินพิจารณาการรีมอดุลาไรเซชัน ซึ่งเครื่องมือจะแนะนำรูปแบบคำตอบที่เหมาะสมที่สุดและให้ผลคาดการณ์ค่าตัววัดหลังทำการรีมอดุลาไรเซชัน โดยจะนำเสนอรายละเอียดการออกแบบและพัฒนาเครื่องมือในบทถัดไป

3.6 การทดสอบและประเมินผลเครื่องมือ

ในวิทยานิพนธ์นี้เลือกตัวอย่างเพื่อนำมารีมอดุลาไรเซชัน 7 ตัวอย่าง ดังตารางที่ 3.1 ตัวอย่างระบบซอฟต์แวร์ที่นำมาใช้เพื่อประเมินผลเครื่องมือ การแสดงผลข้อมูล เมื่อเครื่องมือทำการค้นหาและคำนวณผลการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามแล้ว จะแสดงผลข้อมูลการทำงานของเครื่องมือในรูปแบบ ข้อความแสดงผล ซึ่งเป็นไปได้ 2 กรณี คือ

กรณีที่ 1 หากเครื่องมือพบว่ากราฟพึ่งพาที่นำเข้าข้อมูลไม่จำเป็นต้องทำรีมอดุลาไรเซชัน เนื่องจากค่าสัมประสิทธิ์ซิลูเอทที่คำนวณได้นั้นมากกว่าเกณฑ์ที่กำหนด จะแสดงข้อความ “This system has Silhouette greater than 0.51 and don't need remodularization”

กรณีที่ 2 หากเครื่องมือพบว่ากราฟพึ่งพาที่นำเข้าข้อมูลนี้ต้องมีการทำรีมอดุลาไรเซชัน จะทำการค้นหาคำตอบที่เหมาะสม จะแสดงข้อมูลที่ประกอบด้วย วิธีการมูฟคลาสว่าควรย้ายไปยังแพ็กเกจที่เหมาะสมจึงจะเหมาะสมกว่า สรุป จำนวน แพ็กเกจและสมาชิกในแต่ละแพ็กเกจหลังจากการีมอดุลาไรเซชัน และแสดงค่าสัมประสิทธิ์ซิลูเอททั้งก่อนและหลังการรีมอดุลาไรเซชัน

บทที่ 4
การออกแบบและพัฒนาเครื่องมือ

การออกแบบและพัฒนาเครื่องมือสำหรับสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งเครื่องมือนี้พัฒนาขึ้นเพื่อช่วยในการตัดสินใจในรีมอดุลาไรเซชันซอฟต์แวร์ โดยที่เครื่องมือจะดำเนินการวิเคราะห์และช่วยในการตัดสินใจรีมอดุลาไรเซชันซอฟต์แวร์ โดยเครื่องมือจะแบ่งการทำงานออกเป็นการตัดสินใจว่าซอฟต์แวร์นั้นๆ ควรจะทำรีมอดุลาไรเซชันหรือไม่ และ การค้นหา เพื่อเสนอแนวทางในการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่จะค้นหาวิธีการที่เหมาะสมที่สุดสำหรับการจัดองค์ประกอบภายในซอฟต์แวร์ ใช้การค้นหาต้องห้ามและตัววัดสัมประสิทธิ์ซิลูเอท และในบทนี้จะนำเสนอการออกแบบและการพัฒนาเครื่องมือประกอบด้วย 4 ส่วน คือ การอธิบายสภาพแวดล้อมในการพัฒนาเครื่องมือ ความต้องการที่เป็นหน้าที่หลักของเครื่องมือ โครงสร้างและการทำงานของเครื่องมือ และแบบจำลองการออกแบบและพัฒนาเครื่องมือ โดยรายละเอียดการออกแบบและการพัฒนาเครื่องมือมีรายละเอียดดังต่อไปนี้

4.1 สภาพแวดล้อมในการพัฒนาเครื่องมือ

สภาพแวดล้อมในการพัฒนาเครื่องมือสำหรับช่วยสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม มีรายละเอียดดังตารางที่ 4.1 ซึ่งแสดงรายละเอียดสภาพแวดล้อมด้านฮาร์ดแวร์ และตารางที่ 4.2 จะแสดงรายละเอียดสภาพแวดล้อมด้านซอฟต์แวร์

ตารางที่ 4.1 รายละเอียดสภาพแวดล้อมด้านฮาร์ดแวร์

สภาพแวดล้อมด้านฮาร์ดแวร์	รายละเอียด
คอมพิวเตอร์ส่วนบุคคล	หน่วยประมวลผล Intel Core i5-4210U
	หน่วยความจำ 8.00 กิกะไบต์

ตารางที่ 4.2 รายละเอียดสภาพแวดล้อมด้านซอฟต์แวร์

สภาพแวดล้อมด้านซอฟต์แวร์	รายละเอียด
ภาษาที่ใช้ในการพัฒนาเครื่องมือ	จาวา (Java)
เครื่องมือที่ใช้ในการพัฒนา	IntelliJ IDEA Community Edition 2020.2.1
ระบบปฏิบัติการ	ไมโครซอฟต์วินโดว์ 8 แบบ 64 บิต

4.2 ความต้องการที่เป็นหน้าที่หลักของเครื่องมือ

ในส่วนนี้จะอธิบายถึงความต้องการในการทำงานที่เป็นหน้าที่หลักของเครื่องมือ ซึ่งจากระเบียบวิธีวิจัยที่ได้เสนอในบทที่ผ่านมา ทำให้สามารถกำหนดความต้องการพื้นฐานของเครื่องมือได้ เพื่อให้เครื่องมือนี้มีหน้าที่และการทำงานเป็นไปตามวิธีการที่นำเสนอไว้ ความต้องการที่เป็นหน้าที่หลักจะประกอบด้วย 2 ฟังก์ชันการทำงานดังตารางที่ 4.3

ตารางที่ 4.3 ความต้องการที่เป็นหน้าที่หลัก

ความต้องการที่เป็นหน้าที่หลัก	รายละเอียด
1.การนำเข้าข้อมูล	1. ผู้ใช้สามารถนำเข้าข้อมูลกราฟ Graph.txt ได้
	2. ผู้ใช้สามารถนำเข้าข้อมูลแพ็กเกจ Package.txt ได้
2.การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	1. เครื่องมือสามารถคำนวณค่าสัมประสิทธิ์ซิลูเอทจากข้อมูลนำเข้าได้
	2. เครื่องมือจะต้องสามารถคำนวณค่าสัมประสิทธิ์ซิลูเอทก่อนและหลังรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามได้
	3. เครื่องมือจะต้องสามารถตัดสินใจได้ว่าซอฟต์แวร์ควรทำรีมอดุลาไรเซชันหรือไม่ได้
	4. เครื่องมือจะต้องสามารถเสนอวิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามได้
	5. กรณีที่รีมอดุลาไรเซชัน เครื่องมือจะต้องแสดงผลได้ว่าคลาสใดควรถูกย้ายและย้ายไปแพ็กเกจใดได้

4.3 แบบจำลองการออกแบบและพัฒนาเครื่องมือ

การออกแบบและพัฒนาเครื่องมือที่ใช้สนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม การพัฒนาและออกแบบเครื่องมือนี้ ส่วนต่างๆ จะถูกพัฒนาขึ้นโดยใช้เครื่องมืออินเทลลิเจเป็นเครื่องมือหลัก โดยระบบซอฟต์แวร์ที่ได้จะอยู่ในรูปแบบโปรแกรมอรรถประโยชน์ เพื่อช่วยสนับสนุนในการทำรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม ซึ่งจะสามารถอธิบายการออกแบบเครื่องมือได้ โดยแบบจำลองต่างๆ ดังต่อไปนี้

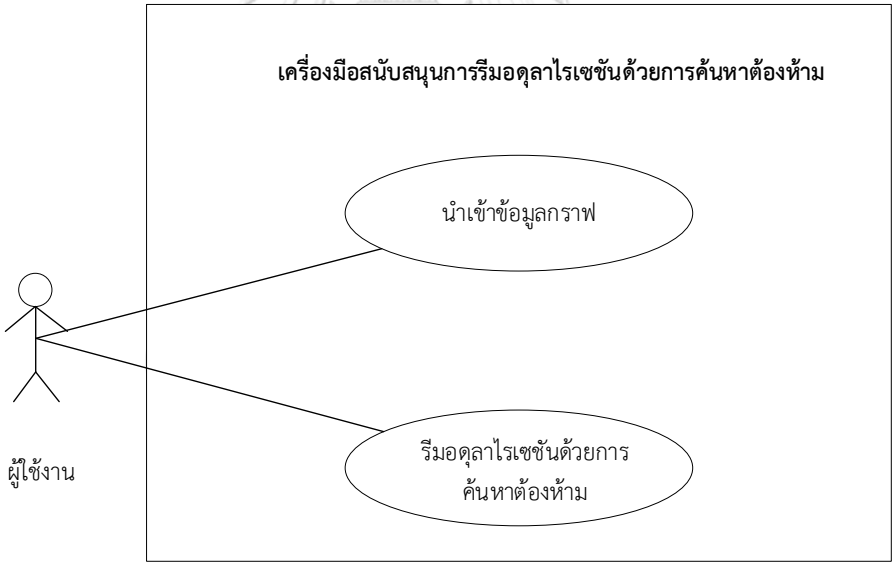
4.3.1 แบบจำลองเชิงหน้าที่

4.3.1.1 แผนภาพยูสเคส ในวิทยานิพนธ์นี้ผู้วิจัยได้ทำการพัฒนาเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ตามที่วิทยานิพนธ์นี้ได้เสนอระเบียบวิธีการไปแล้ว โดยที่จะมีหลักการทำงานและความสามารถของเครื่องมือสนับสนุนนี้ ซึ่งจะสามารถอธิบายรายละเอียดได้ดังนี้

1. ผู้ใช้งานสามารถนำเข้าข้อมูลกราฟ ซึ่งจะประกอบด้วยข้อมูลความสัมพันธ์ของกราฟและข้อมูลแพ็กเกจ เพื่อที่จะนำไปเป็นข้อมูลเพื่อวิเคราะห์ คำนวณค่าสัมประสิทธิ์ซิลูเอท และการค้นหาต่อไป

2. ผู้ใช้งานสามารถแสดงการรีมอดุลาไรเซชัน โดยที่ข้อมูลจะแสดงผลการตัดสินใจเพื่อทำรีมอดุลาไรเซชันหรือไม่ และหากทำก็จะแสดงผลลัพธ์วิธีการทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

จากความสามารถและการทำงานของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม สามารถแสดงเป็นแผนภาพยูสเคสได้ดังรูปที่ 4.1



รูปที่ 4.1 แผนภาพยูสเคสของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

จากรูปที่ 4.1 แผนภาพยูสเคสเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม สามารถอธิบายความสัมพันธ์ของยูสเคสที่เกิดขึ้นภายในการทำงานของระบบได้ดังนี้

คำอธิบายยูสเคส

ตารางที่ 4.4 ยูสเคสการนำเข้าข้อมูลกราฟ

Use Case:	นำเข้าข้อมูลกราฟ
Actor:	ผู้ใช้งาน
Purpose:	แสดงลำดับเหตุการณ์ในการนำเข้าข้อมูลกราฟ
Description:	เมื่อผู้ใช้งานเปิดการใช้งานเครื่องมือ และมีซอฟต์แวร์ที่ต้องการตรวจสอบ ผู้ใช้งานนำเข้าข้อมูลซอฟต์แวร์ทีละรายการคือข้อมูลกราฟ และข้อมูลซอฟต์แวร์ เครื่องมือจะมีข้อมูลกราฟและข้อมูลแพ็กเกจ
Basic Flow:	1. ผู้ใช้งานนำเข้าไฟล์ข้อมูลกราฟของซอฟต์แวร์ที่ต้องการ 2. ผู้ใช้งานนำเข้าไฟล์ข้อมูลแพ็กเกจของซอฟต์แวร์ที่ต้องการ
Alternative Flow:	-
PreCondition:	มีข้อมูลกราฟและแพ็กเกจของซอฟต์แวร์ที่ต้องการของแต่ละซอฟต์แวร์ที่มีลักษณะเป็นตัวอักษร(.txt)
PostCondition:	ผู้ใช้งานเลือกทำขั้นตอนต่อไป

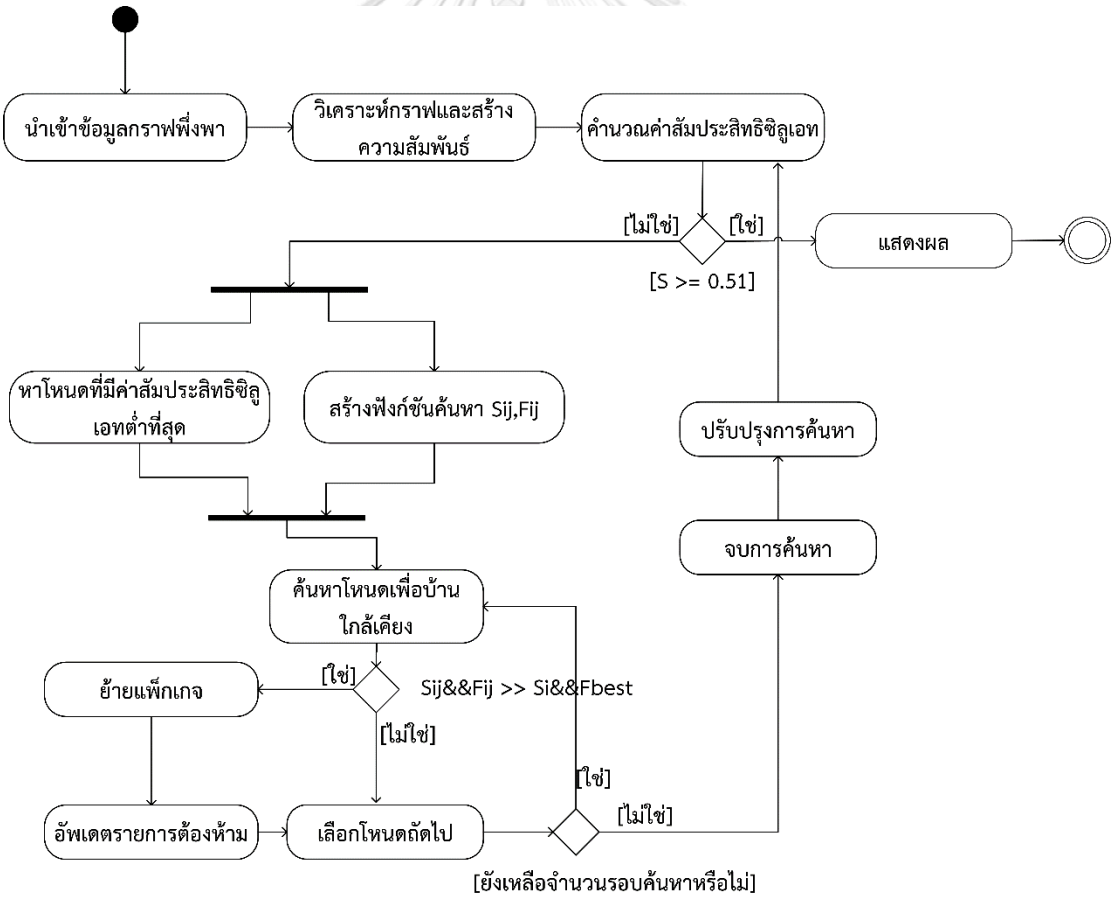
ตารางที่ 4.5 ยูสเคสรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

Use Case:	รีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม
Actor:	ผู้ใช้งาน
Purpose:	แสดงลำดับเหตุการณ์ในการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม
Description:	เมื่อผู้ใช้งานนำเข้าข้อมูลซอฟต์แวร์ที่ต้องการตรวจสอบการรีมอดุลาไรเซชัน และได้สั่งรันโปรแกรมเพื่อรีมอดุลาไรเซชันซอฟต์แวร์ เมื่อเสร็จสิ้น ผู้ใช้จะได้รับผลการรีมอดุลาไรเซชัน
Basic Flow:	1. ผู้ใช้งานสั่งรันเพื่อแสดงผล 2. ระบบสร้างกราฟจากความสัมพันธ์จากชุดข้อมูลที่นำเข้า 3. ระบบคำนวณค่าสัมประสิทธิ์ซิลูเอทจากข้อมูลที่นำเข้า 4. ระบบตัดสินใจทำรีมอดุลาไรเซชันซอฟต์แวร์ 5. ระบบแสดงผล
Alternative Flow:	-
PreCondition:	ผู้ใช้นำเข้าข้อมูลกราฟและข้อมูลแพ็กเกจของซอฟต์แวร์แล้ว
PostCondition:	-

ตารางที่ 4.4 เป็นยูสเคสที่แสดงการทำงานของส่วนนำเข้าข้อมูลกราฟ ซึ่งผู้ใช้งานจำเป็นต้องนำเข้าข้อมูลกราฟให้ครบและสอดคล้องกันทั้ง 2 ส่วนคือกราฟและแพ็กเกจ ในรูปแบบไฟล์ตัวอักษร .txt เพื่อนำไปวิเคราะห์และคำนวณค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ต่อไป

ตารางที่ 4.5 เป็นยูสเคสรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งจะเกิดขึ้นเมื่อผู้ใช้งานนำเข้าชุดข้อมูลเรียบร้อยแล้ว ระบบจะทำการวิเคราะห์ชุดข้อมูลที่ผู้ใช้งานนำเข้า และจะทำการคำนวณค่าสัมประสิทธิ์ซิลูเอทเพื่อตัดสินในการทำรีมอดุลาไรเซชัน และหากอยู่ในเกณฑ์ที่กำหนดก็จะทำการรีมอดุลาไรเซชันตามที่ได้เสนอวิธีการในวิทยานิพนธ์ เมื่อระบบทำการประมวลผลตามขั้นตอนแล้วจะแสดงผลการดำเนินการและจบกระบวนการ

4.3.1.1 แผนภาพกิจกรรม แผนภาพกิจกรรมแสดงขั้นตอนการทำงานของเครื่องมือ โดยเครื่องสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้ามมีลำดับและขั้นตอนดังรูปที่ 4.2



รูปที่ 4.2 แผนภาพกิจกรรมของเครื่องมือ

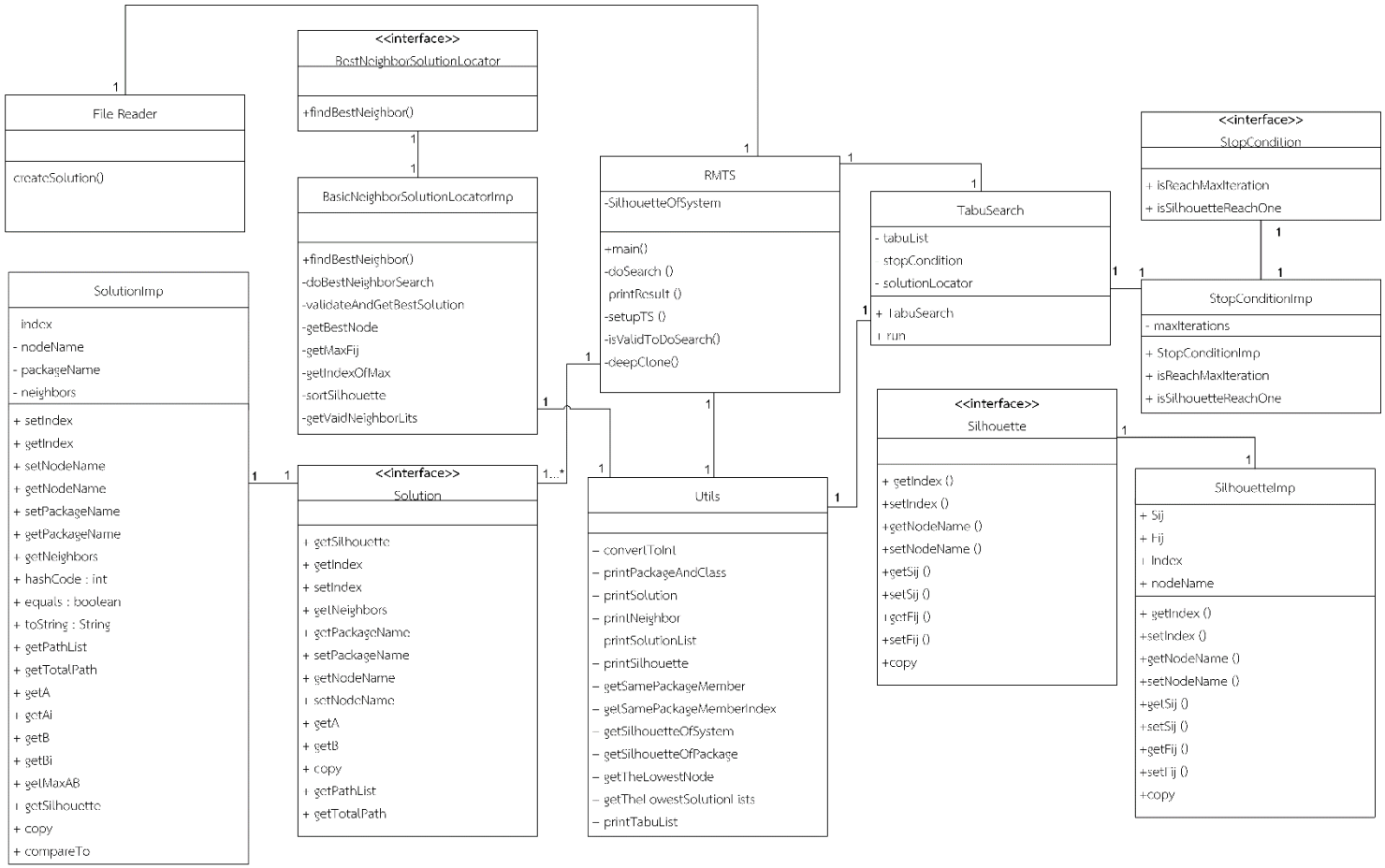
แผนภาพกิจกรรมรูปที่ 4.2 เป็นแผนภาพกิจกรรมของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งจะอธิบายถึงลำดับขั้นตอนการทำงานของเครื่องมือซึ่งมีรายละเอียดคือ เริ่มต้นเริ่มจากการนำเข้าข้อมูลกราฟพึ่งพาของซอฟต์แวร์ที่ต้องการตรวจสอบเพื่อการรีมอดุลาไรเซชัน ประกอบด้วย ข้อมูลกราฟและข้อมูลแพ็กเกจ จากนั้นวิเคราะห์ข้อมูลกราฟเพื่อสร้างความสัมพันธ์ของกราฟในเครื่องมือ ทำการคำนวณค่าสัมประสิทธิ์ซิลูเอท เพื่อตัดสินใจในการทำรีมอดุลาไรเซชันเมื่อตัดสินใจ กรณีที่ตัดสินใจทำรีมอดุลาไรเซชันจะเริ่มต้นการค้นหาต้องห้าม ซึ่งเมื่อสิ้นสุดการค้นหาต้องห้ามแล้วจะได้การจัดสรรคลาสใหม่และค่าสัมประสิทธิ์ซิลูเอทใหม่ และจะเริ่มทำการค้นหาอีกครั้งโดยปรับปรุงการค้นหาโดยปรับปรุงรายการต้องห้าม ให้ว่างและเข้าสู่กระบวนการค้นหาอีกครั้งเพื่อค้นหาให้ได้คำตอบที่ดีที่สุด เมื่อได้คำตอบบที่ดีที่สุดแล้วจะหยุดการค้นหาและแสดงผล และกิจกรรมแสดงผลของเครื่องมือ โดยจะแสดงเป็นข้อความประกอบด้วย แสดงคลาสที่ควรย้ายและแพ็กเกจที่เป็นเป้าหมาย แสดงจำนวนคลาสในแต่ละแพ็กเกจ แสดงค่าสัมประสิทธิ์ซิลูเอททั้งก่อนและหลังการค้นหา เช่นเดียวกันกับกรณีที่ไม่ต้องทำรีมอดุลาไรเซชัน จะแสดงผลการวิเคราะห์จากเครื่องมือเป็นข้อความ โดยอธิบายว่าซอฟต์แวร์นั้นมีค่าสัมประสิทธิ์ซิลูเอทตั้งแต่ 0.51 จึงไม่ต้องรีมอดุลาไรเซชัน

4.3.2 แบบจำลองเชิงโครงสร้าง

4.3.2.1 แผนภาพคลาส แผนภาพคลาสการทำงานของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งแสดงภาพรวมของการทำงานของเครื่องมือ ดังรูปที่ 4.3 ซึ่งจะแสดงความสัมพันธ์ในการทำงานระหว่างคลาสต่างๆ ภายในระบบของเครื่องมือที่พัฒนา

จากรูปที่ 4.3 คลาสหลักภายในระบบมีหน้าที่และการทำงานดังต่อไปนี้

- คลาส RMTS เป็นคลาสหลักในการทำรีมอดุลาไรเซชันโดยจะตรวจสอบว่าจะต้องทำการค้นหาหรือไม่ และหากเงื่อนไขถูกต้องจะทำการค้นหา และทำแสดงผล
  แอททริบิว คือ SilhouetteOfSystem เก็บค่าสัมประสิทธิ์ซิลูเอทของระบบที่เป็นเกณฑ์ตัดสินใจในการค้นหา
  เมธอดที่สำคัญ คือ doSearch, isValidToDoSearch ใช้เพื่อการค้นหาและตัดสินใจในการค้นหาโดยเปรียบเทียบจากเกณฑ์ค่าสัมประสิทธิ์ซิลูเอทของระบบกับค่าเกณฑ์ตัดสินใจ (SilhouetteofSystem) และ printResult ใช้สำหรับแสดงผล
- คลาส TabuSearch เป็นคลาสหลักที่มีหน้าที่ในการค้นหาต้องห้าม
  แอททริบิว คือ tabuList, stopCondition, solutionLocator ซึ่งจะเก็บข้อมูลรายการต้องห้าม (Tabu list) เงื่อนไขในการหยุดค้นหา และโหนดที่ทำการค้นหา
  เมธอดที่สำคัญ คือ TabuSearch, run เพื่อใช้ในการค้นหาต้องห้าม



รูปที่ 4.3 แผนภาพคลาสของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

จากรูปที่ 4.3 คลาสภายในระบบมีหน้าที่และการทำงานดังต่อไปนี้

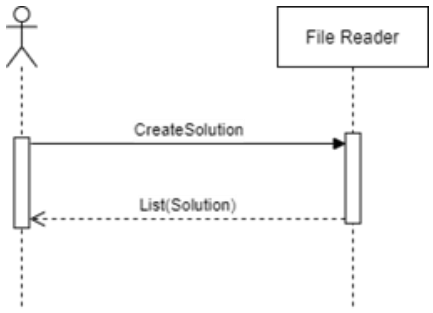
- คลาส BasicNeighborSolutionLocatorImp เป็นคลาสที่มีหน้าในการคำนวนและเปรียบเทียบค่าฟังก์ชันการค้นหา
  เมธอดที่สำคัญ คือ findBestNeighbor, doBestNeighborSearch, validateAndGetBestSolution, getBestNode, getMaxFij, getIndexOfMax, sortSilhouette, getVaidNeighborLits ซึ่งช่วยจัดการข้อมูลต่างๆสำหรับการคำนวณค่าฟังก์ชันการค้นหา และคำนวณ เปรียบเทียบค่า Sij และ Fij เพื่อเลือกโหนดที่ดีที่สุด
- คลาส FileReader เป็นคลาสที่ทำหน้าที่สร้างกราฟ จากความสัมพันธ์ของ ข้อมูลกราฟและข้อมูลแพ็กเกจที่นำเข้าทีละโหนด และเก็บอยู่ในรูปแบบลิสต์ของโหนด
  เมธอดที่สำคัญ คือ createSolution() ใช้สำหรับสร้างกราฟจากความสัมพันธ์ของกราฟพึ่งพาที่นำเข้า
- คลาส SolutionImp เป็นคลาสสำหรับจัดเก็บข้อมูลโหนดและลำดับของโหนด (index ใช้เพื่อเก็บลำดับของโหนดเพื่อให้ง่ายสำหรับการใช้งาน)
  แอททริบิว คือ index, nodeName, packageName, neighbors ซึ่งเป็นข้อมูลของโหนด ได้แก่ ชื่อ แพ็กเกจ เพื่อนบ้าน ค่า A ค่า B และค่าซิลูเอทของโหนดเอง โดยหนึ่งกราฟประกอบด้วยลิสต์ของโหนด
  เมธอดที่สำคัญ คือ getA, getAi, getB, getBi, getMaxAB, getSilhouette เมธอดที่เกี่ยวข้องกับตัวแปรในการคำนวณค่าซิลูเอท นอกจากนี้ยังมีส่วนที่เกี่ยวข้องกับโหนด และแพ็กเกจทั้งหมด setIndex, getIndex, setNodeName, setPackageName, getPackageName, getNeighbors

- คลาส Utils เป็นคลาสที่มีเมธอดสำหรับทำหน้าที่ช่วยคำนวณและจัดการข้อมูลค่าสัมประสิทธิ์ซิลูเอทประเภทต่างๆ ทั้งโหนดและระบบ
  เมธอดที่สำคัญ จะเกี่ยวข้องกับการคำนวณค่าสัมประสิทธิ์ซิลูเอทและค้นหาโหนดที่ให้ค่าสัมประสิทธิ์ซิลูเอทต่ำที่สุด เพื่อทำการหาจุดเริ่มต้น เช่น getSamePackageMember, getSamePackageMemberIndex, getSilhouetteOfSystem, getSilhouetteOfPackage, getTheLowestNode, getTheLowestSolutionLists และ เกี่ยวข้องกับแสดงผล เช่น printPackageAndClass, printSolution, printNeighbor, printSolutionList

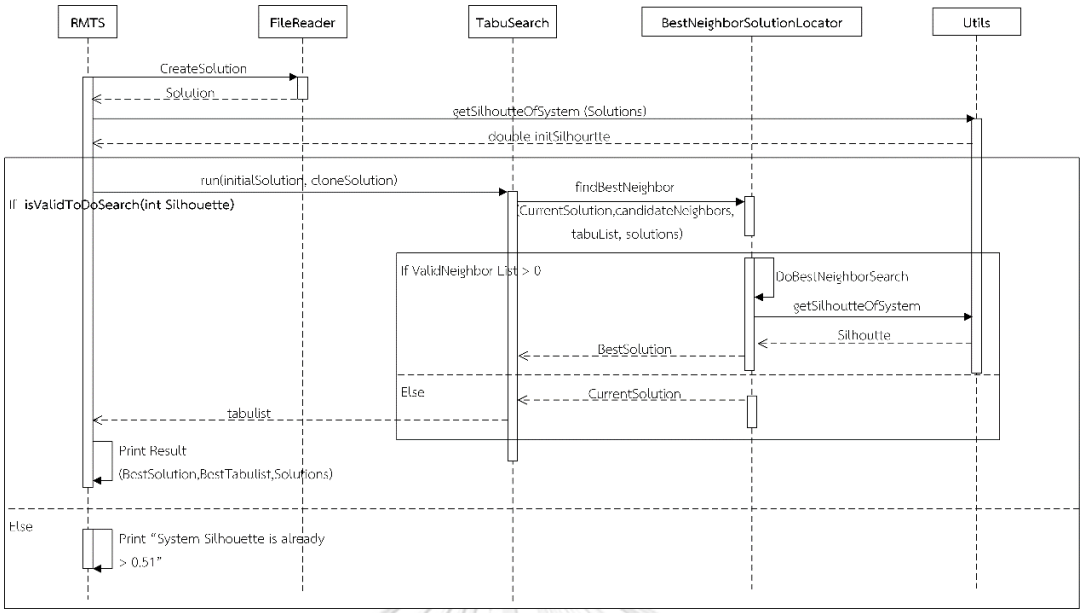
- คลาส SilhouetteImp เป็นคลาสสำหรับอ้างอิง ค่าสัมประสิทธิ์ซิลูเอทและโหนดก่อนและหลังการย้ายแพ็กเกจ และฟังก์ชันการค้นหา ซึ่งประกอบด้วย ฟังก์ชันการค้นหาทั้ง 2 ฟังก์ชัน ชื่อโหนด และ ลำดับของโหนด
  เมธอดที่สำคัญ จะเกี่ยวข้องกับการเก็บค่าการคำนวณฟังก์ชันการค้นหาและค่าสัมประสิทธิ์ซิลูเอทของโหนดและของระบบ เช่น getSij, setSij, getFij, setFij getIndex, setIndex, getNodeName เป็นต้น
- คลาส StopCondition เป็นคลาสที่ใช้เพื่อกำหนดจำนวนรอบการค้นหา จะเกี่ยวข้องกับจำนวนโหนด, ค่ามากที่สุดของสัมประสิทธิ์ซิลูเอท โดยที่เงื่อนไขจำนวนรอบน้อยกว่าหรือเท่ากับจำนวนโหนด และค่าสัมประสิทธิ์ซิลูเอทของระบบน้อยกว่าหรือเท่ากับ 0.51 ถ้าเข้าเงื่อนไขก็จะทำการค้นหาต่อไป
  แอททริบิว คือ maxIterations เก็บจำนวนรอบที่มากที่สุดที่ค้นได้
  เมธอดที่สำคัญจะเกี่ยวข้องเป็นเงื่อนไขสำหรับหยุดการค้นหา เช่น StopConditionImp, isReachMaxIteration, isSilhouetteReachOne

4.3.3 แบบจำลองเชิงพฤติกรรม

4.3.3.1 แผนภาพลำดับ แผนภาพลำดับเป็นแผนภาพที่แสดงให้เห็นถึงการติดต่อกันระหว่างฟังก์ชันการทำงานและการส่งผ่านข้อความระหว่างฟังก์ชัน รูปที่ 4.4 และ 4.5 แผนภาพลำดับการนำเข้ากราฟพึ่งพาและการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม



รูปที่ 4.4 แผนภาพลำดับการนำเข้ากราฟพึ่งพา



รูปที่ 4.5 แผนภาพลำดับการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

บทที่ 5
การทดสอบและประเมินผลความสามารถของเครื่องมือ

ในบทนี้จะทำการทดสอบและประเมินผลความสามารถของเครื่องมือ โดยที่ผู้วิจัยได้นำเสนอเครื่องมือที่ช่วยสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยวิธีค้นหาแบบต้องห้าม โดยแสดงรายละเอียดการใช้งานในภาคผนวก ก. ดังนั้นเพื่อเป็นการทดสอบและประเมินความสามารถของเครื่องมือในการรีมอดุลาไรเซชัน โดยวิทยานิพนธ์นี้มีแนวทางในการประเมินผลความารถของเครื่องมือ 2 ขั้นตอนคือ ประเมินความสามารถของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยวิธีค้นหาแบบต้องห้าม และประเมินคุณภาพการรีมอดุลาไรเซชันด้วยวิธีค้นหาแบบต้องห้าม โดยมีรายละเอียดดังต่อไปนี้

5.1 การประเมินความสามารถของเครื่องมือ

การประเมินความสามารถของเครื่องมือสนับสนุนนี้ เริ่มจากการเตรียมข้อมูลซอฟต์แวร์เพื่อเป็นชุดข้อมูลในการทดสอบความสามารถในการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เตรียมข้อมูลให้อยู่ในรูปแบบความสัมพันธ์ของกราฟพึ่งพา ตามที่ระเบียบวิธีวิจัยได้เสนอไว้แล้ว จากนั้นจะทำการนำเข้าเครื่องมือเพื่อตรวจสอบการรีมอดุลาไรเซชัน และหากมีการดำเนินการพิจารณาให้ทำรีมอดุลาไรเซชันซอฟต์แวร์ ผลลัพธ์ที่ได้จากการแนะนำของเครื่องมือ หากมีการมูฟตามคำแนะนำของเครื่องมือแล้วความสัมพันธ์จากกราฟที่ได้จะมีผลที่ดีขึ้น เป็นไปตามข้อกำหนดหรือไม่

5.1.1 รายละเอียดตัวอย่างที่ใช้ในการทดสอบ

เพื่อทดสอบและประเมินผลความสามารถของเครื่องมือ จึงได้ทดสอบด้วยซอฟต์แวร์จำนวน 7 ตัวอย่าง เขียนด้วยภาษาจาวา มีขนาดแตกต่างกัน โดยแบ่งเป็น 2 ประเภท คือ ซอฟต์แวร์จากโอเพนซอร์สจำนวน 2 ตัวอย่าง และเพื่อศึกษาความสามารถในการทำงานของเครื่องมือผู้วิจัยจึงสร้างกรณีศึกษาขึ้นมาอีก 5 ตัวอย่าง เพื่อศึกษาและทำการทดสอบการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม โดยที่ซอฟต์แวร์ที่นำมาใช้ในการทดสอบจะต้องสามารถแปลงเป็นกราฟพึ่งพาจากเครื่องมือดีเพนเดนซีไฟน์เดอร์ และสามารถแปลงเป็นข้อมูลกราฟพึ่งพาได้ มีข้อมูลกราฟอยู่ในลักษณะเมตริทซ์พึ่งพา มีข้อมูลแพ็กเกจและข้อมูลกราฟ เป็นไฟล์นามสกุล .txt ที่ครบถ้วน ซึ่งตารางที่ 5.1 คือรายละเอียดซอฟต์แวร์ที่นำมาทดสอบ

ตารางที่ 5.1 รายละเอียดซอฟต์แวร์ที่นำมาใช้ทดสอบ

ลำดับ	ซอฟต์แวร์	ขนาด	รายละเอียด
1.	JMeter	3 แพ็กเกจ 47 คลาส	แพ็กเกจที่ 1 มี 24 คลาส แพ็กเกจที่ 2 มี 20 คลาส แพ็กเกจที่ 3 มี 3 คลาส
2.	JUnit4	5 แพ็กเกจ 79 คลาส	แพ็กเกจที่ 1 มี 11 คลาส แพ็กเกจที่ 2 มี 23 คลาส แพ็กเกจที่ 3 มี 21 คลาส แพ็กเกจที่ 4 มี 20 คลาส แพ็กเกจที่ 5 มี 4 คลาส
3.	Example 1	3 แพ็กเกจ 8 คลาส	แพ็กเกจที่ 1 มี 3 คลาส แพ็กเกจที่ 2 มี 3 คลาส แพ็กเกจที่ 3 มี 2 คลาส
4.	Example 2	4 แพ็กเกจ 16 คลาส	แพ็กเกจที่ 1 มี 8 คลาส แพ็กเกจที่ 2 มี 2 คลาส แพ็กเกจที่ 3 มี 3 คลาส แพ็กเกจที่ 4 มี 3 คลาส
5.	Example 3	3 แพ็กเกจ 12 คลาส	แพ็กเกจที่ 1 มี 3 คลาส แพ็กเกจที่ 2 มี 5 คลาส แพ็กเกจที่ 3 มี 4 คลาส
6.	Example 4	3 แพ็กเกจ 13 คลาส	แพ็กเกจที่ 1 มี 4 คลาส แพ็กเกจที่ 2 มี 5 คลาส แพ็กเกจที่ 3 มี 4 คลาส
7.	Example 5	4 แพ็กเกจ 16 คลาส	แพ็กเกจที่ 1 มี 8 คลาส แพ็กเกจที่ 2 มี 2 คลาส แพ็กเกจที่ 3 มี 3 คลาส แพ็กเกจที่ 4 มี 3 คลาส

หลังจากได้ซอฟต์แวร์เพื่อใช้ในการทดสอบแล้ว และเตรียมข้อมูลกราฟพึ่งพาเพื่อนำเข้าระบบสำหรับการทดสอบการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยเครื่องมือสนับสนุนที่พัฒนาขึ้น โดยในขั้นตอน

แรกเครื่องมือจะวัดค่าสัมประสิทธิ์ซิลูเอทของกราฟที่นำเข้าเครื่องมือ ซึ่งจะสามารถวิเคราะห์ค่าสัมประสิทธิ์ซิลูเอทก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม เพื่อตัดสินใจทำการรีมอดุลาไรเซชันหรือไม่ ซึ่งหากมีการทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามจะมีการค้นหาและแนะนำรูปแบบที่เหมาะสม และแนะนำว่าควรย้ายคลาสใดไปอยู่แพ็กเกจใดจึงจะเหมาะสมมากขึ้น ซึ่งมีผลตามตารางที่ 5.2

ตารางที่ 5.2 ผลการวิเคราะห์ของเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

ลำดับ	ซอฟต์แวร์	จำนวนแพ็กเกจ		จำนวนคลาสที่มูฟคลาสรีแฟคทอริง	ค่าสัมประสิทธิ์ซิลูเอท	
		ก่อน	หลัง		ก่อน	หลัง
1.	JMeter	3	-	0	> 0.51	-
2.	JUnit4	5	5	2	0.289	0.436
		5	5	3	0.436	0.538
3.	Example 1	3	2	2	0.056	0.604
4.	Example 2	4	4	2	0.325	0.512
5.	Example 3	3	3	2	0.161	0.542
6.	Example 4	3	3	1	0.453	0.667
7.	Example 5	4	4	1	0.257	0.275
		4	3	2	0.275	0.716

จากตารางที่ 5.2 ผลการทดสอบเครื่องมือโดยใช้ซอฟต์แวร์ตัวอย่างจำนวน 7 ตัวอย่างมาทำการทดสอบพบว่า มีซอฟต์แวร์ตัวอย่างจำนวน 1 ตัวอย่างที่มีค่าสัมประสิทธิ์ซิลูเอทไม่อยู่ในเกณฑ์ทำการรีมอดุลาไรเซชันนั่นคือ ซอฟต์แวร์ลำดับที่ 1 ซอฟต์แวร์ JMeter ซึ่งจากการวิเคราะห์ของเครื่องมือชี้ว่าซอฟต์แวร์นี้มีค่าสัมประสิทธิ์ซิลูเอทมีค่ามากกว่า 0.51 จึงไม่เข้าสู่กระบวนการค้นหา และนอกจากนี้ซอฟต์แวร์ตัวอย่างอีก 6 ตัวอย่างมีผลการรีมอดุลาไรเซชันที่เครื่องมือสามารถทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์เพิ่มมากขึ้นได้ โดยมีการแนะนำให้มูฟคลาส และเสนอวิธีการจัดสรรคลาสในแพ็กเกจ เพื่อปรับปรุงค่าสัมประสิทธิ์ซิลูเอทให้ดีขึ้นได้ โดยมีรายละเอียดดังต่อไปนี้

ซอฟต์แวร์ลำดับที่ 2 ซอฟต์แวร์ Junit 4 ได้ทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามโดยที่เกิดการปรับปรุงการจัดรูปแบบจำนวน 2 รอบ เพื่อค้นหารูปแบบคำตอบที่ดีที่สุด โดยรอบการค้นหารอบแรกได้เกิดการมูฟคลาส 2 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทเกิดการเปลี่ยนแปลง โดยเพิ่มขึ้นจาก 0.289 เป็น 0.436 และเมื่อปรับปรุงเงื่อนไขการค้นหาและค้นซ้ำอีกครั้ง ได้รูปแบบการจัด

องค์ประกอบใหม่โดยมีการมูฟคลาสจำนวน 3 คลาส ทำให้ผลค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์เพิ่มมากขึ้นเป็น 0.538 และเมื่อปรับปรุงการค้นหาและค้นหาซ้ำเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลูเอทเกินกว่าเกณฑ์ที่กำหนดไว้ที่ 0.51 แล้ว

ซอฟต์แวร์ลำดับที่ 3 ซอฟต์แวร์ Example 1 ได้ทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เกิดการปรับปรุงการจัดรูปแบบโดยมีการมูฟคลาสจำนวน 2 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ปรับปรุงจาก 0.056 เป็น 0.604 และเมื่อปรับปรุงการค้นหาและนำไปค้นหาซ้ำเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลูเอทมากกว่าเกณฑ์ที่กำหนดไว้แล้ว

ซอฟต์แวร์ลำดับที่ 4 ซอฟต์แวร์ Example 2 ได้ทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เกิดการปรับปรุงการจัดรูปแบบโดยมีการมูฟคลาสจำนวน 2 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ปรับปรุงจาก 0.325 เป็น 0.512 และเมื่อปรับปรุงการค้นหาและนำไปค้นหาซ้ำเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลูเอทมากกว่าเกณฑ์ที่กำหนดไว้แล้ว

ซอฟต์แวร์ลำดับที่ 5 ซอฟต์แวร์ Example 3 ได้ทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เกิดการปรับปรุงการจัดรูปแบบโดยมีการมูฟคลาสจำนวน 2 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ปรับปรุงจาก 0.161 เป็น 0.542 และเมื่อปรับปรุงการค้นหาและนำไปค้นหาซ้ำเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลูเอทมากกว่าเกณฑ์ที่กำหนดไว้แล้ว

ซอฟต์แวร์ลำดับที่ 6 ซอฟต์แวร์ Example 4 ได้ทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เกิดการปรับปรุงการจัดรูปแบบโดยมีการมูฟคลาสจำนวน 1 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ปรับปรุงจาก 0.453 เป็น 0.667 และเมื่อปรับปรุงการค้นหาและนำไปค้นหาซ้ำเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลูเอทมากกว่าเกณฑ์ที่กำหนดไว้แล้ว

ซอฟต์แวร์ลำดับที่ 7 ซอฟต์แวร์ Example 5 ได้ทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยที่เกิดการปรับปรุงการจัดรูปแบบโดยมีการมูฟคลาสจำนวน 1 คลาส ทำให้ค่าสัมประสิทธิ์ซิลูเอทของซอฟต์แวร์ปรับปรุงจาก 0.257 เป็น 0.275 และเมื่อปรับปรุงการค้นหาและนำไปค้นหาซ้ำจะทำให้ค่าสัมประสิทธิ์ซิลูเอทของระบบจะปรับปรุงขึ้นไปอีกจาก 0.275 เป็น 0.716 และเมื่อปรับปรุงการค้นหาและค้นซ้ำอีกครั้งเครื่องมือจะหยุดการค้นหาเนื่องจากค่าสัมประสิทธิ์ซิลู-เอทเกินกว่าเกณฑ์ที่กำหนดไว้แล้ว

จากการประเมินด้วยซอฟต์แวร์ตัวอย่าง เครื่องมือสามารถวิเคราะห์ผลซอฟต์แวร์ได้ทั้ง 2 กรณีคือ กรณีที่ซอฟต์แวร์อยู่ในเกณฑ์ดีแล้วซึ่งไม่ต้องการทำรีมอดุลาไรเซชัน และกรณีที่ซอฟต์แวร์ยัง

อยู่ในเกณฑ์ที่ต่ำกว่าค่าที่กำหนดไว้มีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามและนำเสนอรูปแบบการจัดองค์ประกอบที่ทำให้ค่าสัมประสิทธิ์ซิลูเอทดีขึ้นได้ ซึ่งในวิทยานิพนธ์นี้จะแสดงรายละเอียดรูปแบบการจัดองค์ประกอบของซอฟต์แวร์ก่อนและหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามนี้และการปรับปรุงการค้นหาไว้ใน ภาคผนวก ค.

5.2 การประเมินคุณภาพการรีมอดุลาไรเซชันด้วยวิธีค้นหาต้องห้าม

เมื่อเครื่องมือสามารถรีมอดุลาไรเซชัน โดยทำให้ซอฟต์แวร์มีค่าสัมประสิทธิ์ซิลูเอทที่ดีขึ้นได้ เพื่อทดสอบประสิทธิภาพของการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ซึ่งจากงานวิจัยก่อนหน้า [12] มีการใช้ตัววัดเทอร์โบเอ็มคิวเพื่อประเมินผลคุณภาพของการมอดุลาไรเซชัน ซึ่งเป็นการประเมินคุณภาพที่เหมาะสมของการจัดองค์ประกอบ โดยมีพื้นฐานมาตรวัดการเกาะกลุ่มและการเข้าคู่กัน จึงใช้ตัววัดนี้ในการประเมินคุณภาพการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามโดย โดยประเมินผลก่อนและหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม โดยตัวย่อในตารางที่ 5.3 มีความหมายดังนี้

- ก่อน หมายถึง ก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม
- หลัง หมายถึง หลังที่มีการมูฟคลาสตามคำแนะนำของเครื่องมือตามวิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามแล้ว
- ผลต่าง หมายถึง ค่าผลลัพธ์ของค่าเทอร์โบเอ็มคิวก่อนและหลัง

ตารางที่ 5.3 การประเมินโดยใช้เทอร์โบเอ็มคิว

ซอฟต์แวร์	TurboMQ		
	ก่อน	หลัง	ผลต่าง
JUnit4	3.153	3.454	0.301
Example 1	1.167	1.24	0.073
Example 2	2.30	2.389	0.089
Example 3	1.587	2.017	0.43
Example 4	2.051	2.341	0.29
Example 5	2.252	2.404	0.152

หลังจากการประเมินด้วยตัววัดคุณภาพเทอร์โบเอ็มคิว ตารางที่ 5.3 แสดงการเปรียบเทียบผลลัพธ์จากการใช้ตัววัดเพื่อประเมินว่าการใช้วิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามนี้สามารถช่วยแนะนำวิธีการเพื่อปรับปรุงให้ซอฟต์แวร์มีคุณภาพที่ดีมากขึ้นได้ โดยทุกๆซอฟต์แวร์ใน

การทดสอบเครื่องมือนี้มีค่าเทอร์โบเอ็มคิวที่ดีขึ้นทุกซอฟต์แวร์ ซึ่งทำให้ผลที่ได้รับจากตัววัดนี้สามารถยืนยันได้ว่าหลังการรีมอดุลาไรเซชันด้วยวิธีนี้แล้วสามารถปรับปรุงสภาพมอดุลาร์ และคุณภาพของซอฟต์แวร์ให้ดีขึ้นได้

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 บทสรุปงานวิจัย

วิทยานิพนธ์นี้เสนอวิธีการรีมอดุลาไรเซชันด้วยการค้นต้องห้าม และพัฒนาเครื่องมือสำหรับช่วยสนับสนุนการรีมอดุลาไรเซชันตามวิธีการที่นำเสนอนี้ การทำรีมอดุลาไรเซชันเป็นวิธีการปรับปรุงคุณภาพซอฟต์แวร์รูปแบบหนึ่งโดยเป็นการปรับปรุงโครงสร้างของซอฟต์แวร์ ด้วยการจัดสรรคลาสให้อยู่ในแพ็กเกจที่เหมาะสม ดังนั้นวิทยานิพนธ์นี้จึงเสนอวิธีการค้นหาต้องห้ามเพื่อช่วยในการค้นหาแพ็กเกจที่เหมาะสมมากขึ้นให้กับคลาส โดยใช้ค่าสัมประสิทธิ์ซิลูเอทสำหรับวิเคราะห์และตัดสินใจในการทำการรีมอดุลาไรเซชัน โดยมีเป้าหมายเพื่อปรับปรุงให้ซอฟต์แวร์มีสภาพมอดุลาร์และคุณภาพที่ดีขึ้น จากวิธีการที่นำเสนอนี้ได้พัฒนาเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามด้วยภาษาจาวา โดยที่เครื่องมือสามารถวิเคราะห์และประเมินผลการรีมอดุลาไรเซชันซอฟต์แวร์ได้ทั้งสองกรณี คือกรณีที่ซอฟต์แวร์ไม่อยู่ในเกณฑ์ที่จำเป็นต้องทำรีมอดุลาไรเซชัน และกรณีที่ซอฟต์แวร์อยู่ในเกณฑ์ที่ต้องทำ ซึ่งในส่วนนี้จะเกิดการค้นหาต้องห้ามขึ้นเพื่อค้นหารูปแบบการจัดองค์ประกอบที่เหมาะสมโดยจะแนะนำวิธีการมูฟคลาสรีแฟคทอริงเพื่อให้ซอฟต์แวร์มีการจัดองค์ประกอบที่ดีขึ้น เพื่อจะเป็นเครื่องมือสำหรับช่วยสนับสนุนวิศวกรซอฟต์แวร์ในการตัดสินใจ สำหรับบำรุงรักษาหรือปรับปรุงคุณภาพของซอฟต์แวร์

วิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ประกอบด้วยหกขั้นตอนคือ การสร้างกราฟพึ่งพา การคำนวณค่าสัมประสิทธิ์ซิลูเอทเพื่อตัดสินใจ การรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม การประเมินผลการรีมอดุลาไรเซชัน การพัฒนาเครื่องมือ และการประเมินผลเครื่องมือ ซึ่งการสร้างกราฟพึ่งพานั้นเริ่มจากการแปลงรหัสต้นฉบับของซอฟต์แวร์ที่เลือกมาให้เป็นกราฟพึ่งพาด้วยเครื่องมือดีเพนเดนซีไฟน์เดอร์ จากนั้นแปลงข้อมูลจากกราฟพึ่งพาให้เป็นเมตริกซ์ความสัมพันธ์ระหว่างคลาสเพื่อเตรียมเป็นข้อมูลเพื่อใช้ในการวิเคราะห์ต่อไป การคำนวณค่าสัมประสิทธิ์ซิลูเอทสำหรับการตัดสินใจการรีมอดุลาไรเซชัน โดยเกณฑ์การตัดสินใจของงานนี้อ้างอิงจากช่วงเกณฑ์ตามตารางของ [16] ซึ่งคือช่วงที่ตัดสินใจในการทำรีมอดุลาไรเซชันคือช่วงที่โครงสร้างอยู่ในเกณฑ์ที่ยอมรับได้และช่วงที่ต้องมีการปรับปรุง ซึ่งหากต้องมีการปรับปรุงวิทยานิพนธ์นี้ได้ประยุกต์ใช้วิธีการค้นหาต้องห้ามสำหรับการรีมอดุลาไรเซชัน ซึ่งการค้นหาต้องห้ามได้นำมาใช้ในการค้นหานี้ เป็นการใช้วิธีการค้นหาต้องห้ามโดย ที่การค้นหาจะพิจารณาโหนดเพื่อนบ้านที่เป็นโหนดที่มีการเชื่อมต่อกับโหนดค้นหาปัจจุบันเท่านั้น และรายการต้องห้ามของงานนี้จะเก็บโหนดที่จะต้องย้ายจากแพ็กเกจปัจจุบันไปยังแพ็กเกจอื่น โดยเก็บชื่อโหนด และแพ็กเกจใหม่ที่จะย้ายไป และเมื่อทำจนครบจำนวนโหนดตามที่

กำหนดไว้แล้วจะมีการค้นหาซ้ำอีก ด้วยการปรับปรุงการค้นหาเดิมไปจนกว่าจะได้คำตอบที่ดีที่สุด การประเมินผลการรีมอดุลาไรเซชัน มีการใช้ตัววัดคุณภาพสองตัวสำหรับการวัดและประเมินผล ตัวแรกคือค่าสัมประสิทธิ์ซิลูเอท เนื่องจากคุณสมบัติของตัววัดนี้คือมีช่วงคำตอบที่แน่นอนและสามารถตีความได้ตามตารางที่ 2.1 ซึ่งช่วงการตีความที่แน่นอนทำให้สามารถทำให้ผู้ใช้เกณฑ์นี้ใช้ได้เหมือนกันโดยไม่ขึ้นกับทักษะและประสบการณ์ จึงมีความเหมาะสมที่นำมาใช้ ตัววัดตัวที่สองคือเทอร์โบเอ็มคิว เป็นตัววัดที่มีงานวิจัยในอดีตนำมาประเมินผลทั้งในวิธีการและเครื่องมือ ซึ่งตัววัดนี้ใช้ได้ในเชิงเปรียบเทียบคุณภาพกับงานอื่นและการเปรียบเทียบผลการศึกษาโดยเทียบก่อนและหลังการปรับปรุง ดังนั้นวิทยานิพนธ์นี้จึงมีการใช้ตัววัดทั้งสองตัวนี้เพื่อยืนยันผลการประเมินซึ่งได้ผลสอดคล้องกันคือค่าจากตัววัดทั้งสองเพิ่มขึ้นหลังทำการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม

การประเมินผลเครื่องมือสนับสนุนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ได้ทดสอบกับซอฟต์แวร์จำนวนเจ็ดตัวอย่าง พบว่าเครื่องมือสามารถนำแนะรูปแบบการจัดสรรคลาสในแพ็กเกจที่เหมาะสมได้ โดยสามารถแนะนำวิธีการมูฟคลาสที่ทำให้ซอฟต์แวร์มีโครงสร้างที่ดีขึ้นจนผ่านเกณฑ์การประเมินที่กำหนดไว้ และจากการประเมินด้วยตัววัดสัมประสิทธิ์ซิลูเอทและเทอร์โบเอ็มคิว ซอฟต์แวร์หลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามนี้ มีผลลัพธ์ของค่าตัววัดทั้งสองดีขึ้นเมื่อเปรียบเทียบกับก่อนทำรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามทุกตัวอย่าง จึงสรุปได้ว่าเครื่องมือนี้สามารถปรับปรุงให้ซอฟต์แวร์มีคุณภาพที่ดีตามตัววัดในจุดประสงค์ได้ คือ สามารถค้นหาคำตอบและสามารถปรับปรุงคุณภาพซอฟต์แวร์ให้ดีขึ้นได้

6.2 ข้อจำกัดของงานวิจัย

1. ซอฟต์แวร์ที่นำมาทดสอบจะต้องสามารถจัดรูปแบบ แปลงความสัมพันธ์ให้อยู่ในรูปแบบกราฟพึ่งพาและสามารถจัดเตรียมข้อมูลกราฟให้อยู่ในรูปแบบตามข้อกำหนดในระเบียบวิธีการวิจัยที่กำหนดไว้

2. ความสัมพันธ์ของแพ็กเกจและคลาสภายในซอฟต์แวร์ตัวอย่าง จะพิจารณาความสัมพันธ์ตามเครื่องมือดีเพนเดนซีไฟน์เดอร์ และพิจารณาความสัมพันธ์ของกราฟพึ่งพาสำหรับขั้นตอนการจัดรูปแบบข้อมูลกราฟพึ่งพา จากการอ้างถึงคลาสนั้นในข้อมูลกราฟพึ่งพาที่ได้จากเครื่องมือดังกล่าว

3. ความสัมพันธ์ของกราฟพึ่งพานี้จะพิจารณาความสัมพันธ์ระหว่างคลาส ทั้งในแพ็กเกจเดียวกันและต่างแพ็กเกจ โดยไม่พิจารณาทิศทางและน้ำหนัก

4. ซอฟต์แวร์ที่สามารถนำมารีมอดุลาไรเซชันด้วยเครื่องมือนี้จะต้องมีค่าสัมประสิทธิ์ซิลูเอทตามเกณฑ์ที่กำหนดเท่านั้น

5. การค้นหาต้องห้ามของงานนี้ พิจารณาค้นหาจากโหนดเพื่อนบ้านที่มีความสัมพันธ์กับโหนดค้นหาปัจจุบันเท่านั้น

6.3 แนวทางในการพัฒนางานวิจัยต่อ

1. ศึกษาวิธีการรีมอดุลาไรเซชันด้วยวิธีการค้นหาในรูปแบบอื่นๆ
2. ปรับปรุงเงื่อนไขการค้นหาให้สามารถค้นหาได้อย่างรวดเร็วและครบถ้วนมากขึ้น
3. พัฒนาเครื่องมือให้มีส่วนต่อประสานกับผู้ใช้ที่สะดวกมากขึ้น
4. ศึกษาผลกระทบของรีแฟคทอริงที่มีโอกาสเกิดได้ ขึ้นหลังจากการรีมอดุลาไรเซชัน

บรรณานุกรม

1. English, M., J. Buckley, and J. Collins, *Investigating software modularity using class and module level metrics*, in *Software Quality Assurance*. 2016, Elsevier. p. 177-200.
2. Mathew Hall, M.A.K., Neil Walkinshaw,Phil McMinn, *Establishing the Source Code Disruption Caused by Automated Remodularisation Tools.* IEEE International Conference on Software Maintenance and Evolution, 2014.
3. Mathew Hall , N.W., and Phil McMinn, *Effectively Incorporating Expert Knowledge in Automated Software Remodularisation.* IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 2018. 44.
4. A. Alkhalid, M.A., S.A. Mahmoud, *Software refactoring at the package level using clustering techniques.* IET Software 2010. 5(3).
5. Mitchell, B.S., *A Heuristic Approach to Solving the Software Clustering Problem.* Proceedings of the International Conference on Software Maintenance, 2003.
6. Rathee, A. and J.K. Chhabra. *Software Remodularization by Estimating Structural and Conceptual Relations Among Classes and Using Hierarchical Clustering*. 2017. Singapore %@ 978-981-10-5780-9: Springer Singapore.
7. Santos, G., M.T. Valente, and N. Anquetil, *Remodularization Analysis using Semantic Clustering*, in *IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. 2014, IEEE: Antwerp, Belgium.
8. Arie van, D. and K. Tobias, *Identifying objects using cluster and concept analysis %@* 1581130740 %*U* https://doi.org/10.1145/302405.302629, in *Proceedings of the 21st international conference on Software engineering*. 1999, Association for Computing Machinery: Los Angeles, California, USA. p. 246–255.
9. Brian, S.M. and M. Spiros, *On the Automatic Modularization of Software Systems Using the Bunch Tool.* IEEE Trans. Softw. Eng., 2006. 32 %**@** 0098-5589 %**U** **https://doi.org/**10.1109/**TSE.**2006.31(3): p. 193–208.
10. Olaf Seng, M.B., Matthias Biehl and Gert Pache, *Search-based Improvement of*

Subsystem Decompositions. 2005: p. 1045-1051.

11. Hani Abdeen*, S.D., Houari Sahraouiy and Ilham Alloui, *Automatic Package Coupling and Cycle Minimization*, in *WCRE*. 2009.
12. Bright Gee Varghese R, K.R., Jeno Lovesum, *A novel approach for automatic remodularization of software systems using extended ant colony optimization algorithm.* Information and Software Technology 2019. 114 p. 107–120.
13. Glover, F., *Tabu search fundamentals and uses.* 1995.
14. Glover, F. and M. Laguna, *Tabu Search*, in *Handbook of Combinatorial Optimization: Volume*1–3, D.-Z. Du and P.M. Pardalos, Editors. 1998, Springer US: Boston, MA %@ 978-1-4613-0303-9. p. 2093-2229.
15. Rousseeuw, P.J., *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.* Journal of computational and applied mathematics, 1987. 20: p. 53-65 %@ 0377-0427.
16. Elgedawy, A.I.H.I.I., *Source Code Modularization*. 2017, Switzerland: Springer International Publishing.
17. กิจศิริกุล, บ., ปัญญาประดิษฐ์ (*Artificial Intelligence)*. 2548. p. 7-38.
18. ประสิทธิ์จุตระกูล, ส., การออกแบบและวิเคราะห์อัลกอริทึม. 2553, กรุงเทพ: ภาควิชาคอมพิวเตอร์. จุฬาลงกรณ์มหาวิทยาลัย
19. สวราชย์, ป., การใช้ฮิวริสติกส์แบบทาบูเพื่อแก้ปัญหาเกี่ยวกับเทคโนโลยีกลุ่มที่มีทางเลือกแผนกระบวนการผลิตหลายแบบ, in วิศวกรรมศาสตรมหาบัณฑิต. 2541, จุฬาลงกรณ์มหาวิทยาลัย. p. 9-15.
20. Thomas Zimmermann, N.N., *Predicting Subsystem Failures using Dependency Graph Complexities*, in *The* 18*th IEEE International Symposium on Software Reliability (ISSRE '*07). 2007, IEEE: Trollhattan, Sweden.
21. Brian Scott Mitchell, Z.A.S.M., *A heuristic search approach to solving the software clustering problem %@* 0493526064. 2002, Drexel University.
22. *Refactoring: improving the design of existing code %@* 0201485672. 1999: Addison-Wesley Longman Publishing Co., Inc.
23. Tessier, J., *Dependency Finder*. 2001.
24. Hayden, M. and T. Ewan, *Towards Assessing Modularity %@* 0769529674 %*U*

https://doi.org/10.1109/ACOM.2007.10, in *Proceedings of the First International Workshop on Assessment of Contemporary Modularization Techniques*. 2007, IEEE Computer Society. p. 3.

25. Junha Lee, J.P., and Sooyong Park, *Class Modularization Using Indirect Relationships.* International Conference on Engineering of Complex Computer Systems, 2017.
26. Marcelo Serrano Zanetti, C.J.T., Ingo Scholtes and Frank Schweitzer, *Automated Software Remodularization Based on Move Refactoring.* MODULARITY '14, 2014.
27. Hani Abdeen, S.D., Houari Sahraoui, *Modularization Metrics : Assessing Package Organization in Legacy Large Object-Oriented Software, .* 18th Working Conference on Reverse Engineering, 2011.
28. Bavota, G., et al., *Software Re-Modularization Based on Structural and Semantic Metrics*, in 2010 17*th Working Conference on Reverse Engineering*. 2010, IEEE: Beverly, MA, USA.
29. Vassilios, T. and R.C. Holt, *MoJo: A Distance Metric for Software Clusterings %@* 0769503039, in *Proceedings of the Sixth Working Conference on Reverse Engineering*. 1999, IEEE Computer Society. p. 187.

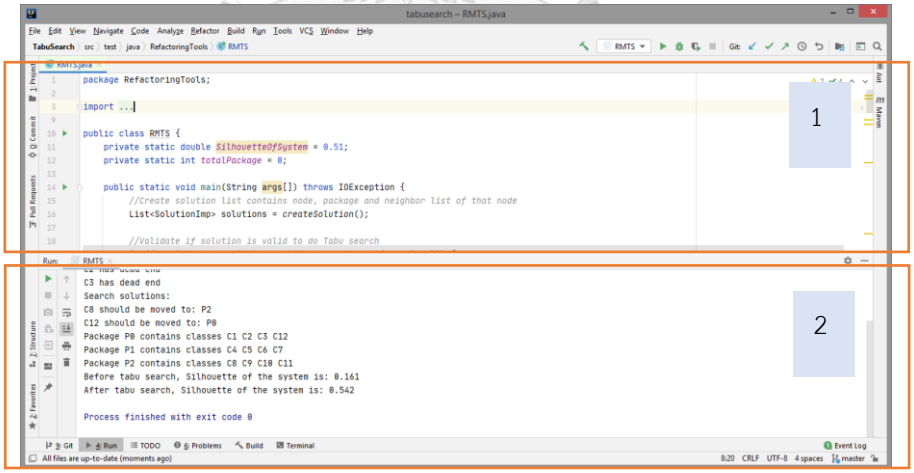
ภาคผนวก

ภาคผนวก ก.
การใช้งานเครื่องมือสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม

สำหรับภาคผนวกจะอธิบายวิธีการใช้เครื่องมือสนับสนุนการรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม ซึ่งเครื่องมือจะสามารถตรวจสอบซอฟต์แวร์ว่าควรมีการปรับปรุงคุณภาพซอฟต์แวร์โดยการรีมอดุลาไรเซชันหรือไม่ และหากต้องทำรีมอดุลาไรเซชันซอฟต์แวร์ จะใช้วิธีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม และเครื่องมือจะสามารถแนะนำให้ผู้ใช้งานทราบว่าจากการค้นหาและวิเคราะห์ผลแล้วควรทำการมูฟรีแฟคทอริงซอฟต์แวร์ที่คลาสใดจึงจะเหมาะสม โดยที่การใช้งานโดยมีรายละเอียดดังต่อไปนี้

ก.1 การใช้งานเครื่องมือ RMTS ด้วย IntelliJ IDEA

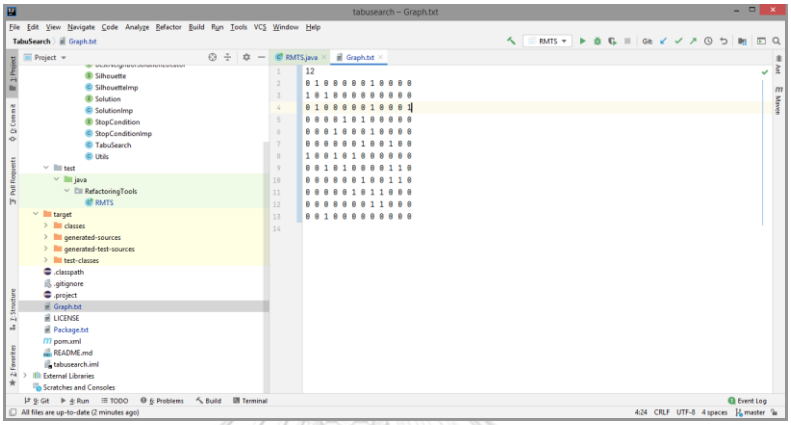
เริ่มต้นใช้งานโดยเป็นโปรเจค RMTS ซึ่งเป็นเครื่องมือที่พัฒนาขึ้นผ่าน เครื่องมือนี้ เมื่อผู้ใช้งานเปิดเครื่องมือและนำเข้าข้อมูลกราฟพึ่งพา ซึ่งประกอบด้วย Graph.txt และ Package.txt เสร็จเรียบร้อยแล้ว เมื่อสั่งรันหน้าจอจะแสดงผลโดยมีส่วนที่ 2 ปรากฏขึ้นมาพร้อมกับแสดงผลการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม ดังรูปที่ ก. 1



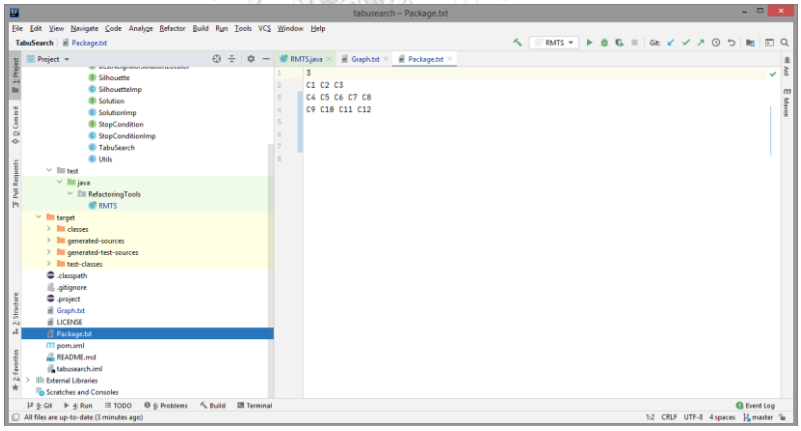
รูปที่ ก.1 หน้าจอแสดงผล

ก.1.1 การนำเข้าข้อมูลกราฟและแพ็กเกจ

เมื่อเตรียมข้อมูลสำหรับใช้กับเครื่องมือ ซึ่งประกอบด้วยข้อมูลกราฟและข้อมูลแพ็กเกจ จะต้องนำเข้าข้อมูล โดยข้อมูลกราฟจะต้องนำเข้าเป็นไฟล์อักษร Graph.txt และข้อมูลแพ็กเกจก็เช่นเดียวกันนำเข้าเป็น Package.txt ดังรูปที่ ก.2 และ ก.3

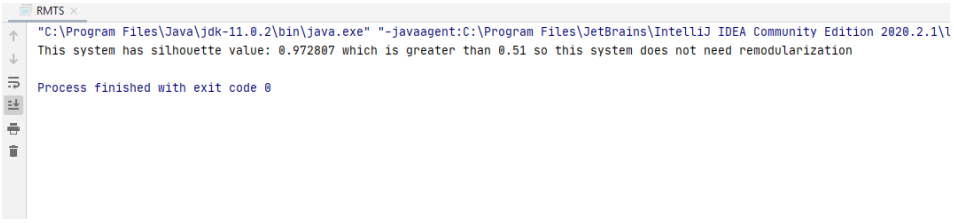


รูปที่ ก.2 ตัวอย่างการนำเข้าข้อมูลกราฟ



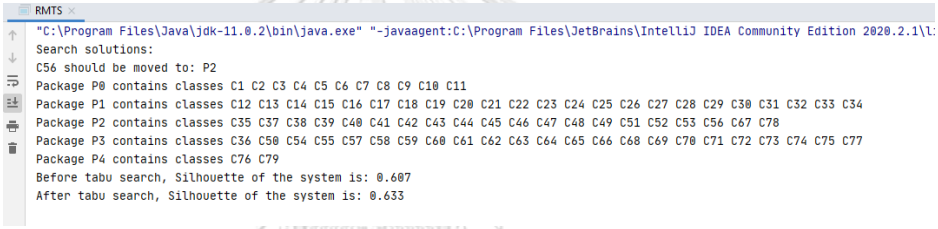
รูปที่ ก.3 ตัวอย่างการนำเข้าข้อมูลแพ็กเกจ

ตัวอย่างผลการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม กรณีที่ซอฟต์แวร์มีค่าสัมประสิทธิ์ซิลูเอทมากกว่า 0.51



รูปที่ ก.4 ตัวอย่างผลการรีมอดุลาไรเซชัน กรณีที่ค่าสัมประสิทธิ์ซิลูเอทมากกว่า 0.51

ตัวอย่างผลการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม กรณีที่ซอฟต์แวร์มีการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม จะมีการแสดงผลโดย แสดงคลาสที่ที่ควรจะมูฟรีแฟคทอริง แสดงรูปแบบการจัดสรรคลาสในแต่ละแพ็กเกจ และค่าสัมประสิทธิ์ซิลูเอทก่อนและหลังการรีมอดุลาไรเซชัน



รูปที่ ก.5 ตัวอย่างผลการรีมอดุลาไรเซชัน กรณีที่ค่าสัมประสิทธิ์ซิลูเอทน้อย 0.51

ภาคผนวก ข.
การสร้างกราฟพึ่งพา

เนื่องจากการใช้เครื่องมือสนับสนุนงานวิจัยนี้ต้องใช้กราฟพึ่งพาโดยมีรูปแบบตามที่กำหนดไว้ และมีการใช้เครื่องมือจากภายนอกเพื่อช่วยในการได้มาซึ่งข้อมูล ภาคผนวก ข. จึงเป็นหัวข้อที่จะอธิบายวิธีการได้มาซึ่งข้อมูลกราฟพึ่งพา ที่เป็นทั้งข้อมูลกราฟ และข้อมูลแพ็กเกจ

ข.1 การแปลงซอฟต์แวร์ตัวอย่างเป็นกราฟพึ่งพา

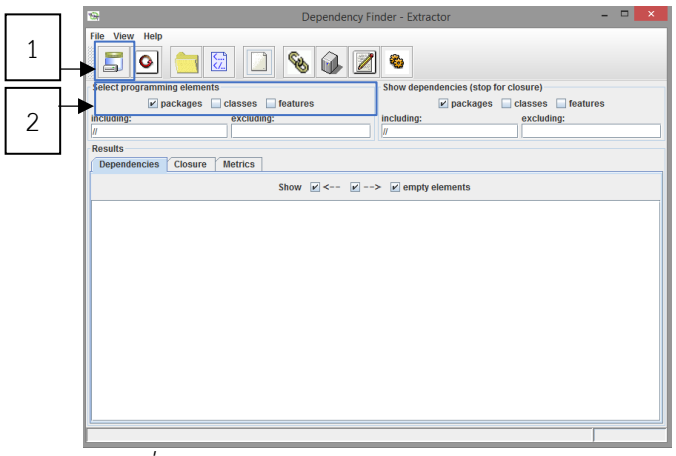
ในขั้นตอนนี้ถือเป็นการเตรียมข้อมูลสำหรับการใช้เครื่องมือ ซึ่งเป็นการใช้เครื่องมือภายนอกมาช่วยสนับสนุนในการเตรียมข้อมูล โดยการแปลงซอฟต์แวร์ตัวอย่างเป็นกราฟพึ่งพามีขั้นตอนดังต่อไปนี้

ข.1.1 การใช้เครื่องมือ DependencyFinder

1. ดาวน์โหลดและติดตั้งเครื่องมือจาก https://depfind.sourceforge.io/
2. ติดตั้งเครื่องมือตามคำแนะนำของผู้พัฒนา
3. เมื่อติดตั้งเสร็จเรียบร้อยแล้วเริ่มการใช้งานโดยเรียกใช้งานเครื่องมือผ่าน command Prompt โดยเรียกผ่านการระบุที่อยู่ไฟล์ และเรียก DependencyFinder ดังรูปที่ ข.1

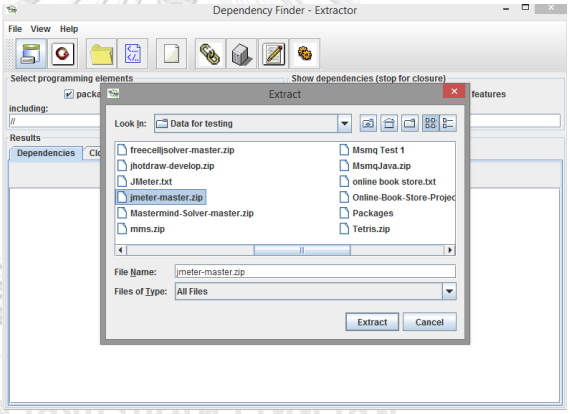
รูปที่ ข.1 การเรียกใช้งานโปรแกรมดีเพนเดนซีไฟน์เดอร์

4. เครื่องมือจะปรากฏขึ้นโดยมีลักษณะดังต่อไปนี้ ดังรูปที่ ข.2 ซึ่งสามารถนำเข้าข้อมูลโดยเลือกที่ ไอคอน Extract หมายเลข 1 และ หมายเลข 2 เป็นส่วนสำหรับเลือกการแสดงส่วนของกราฟที่ต้องการ



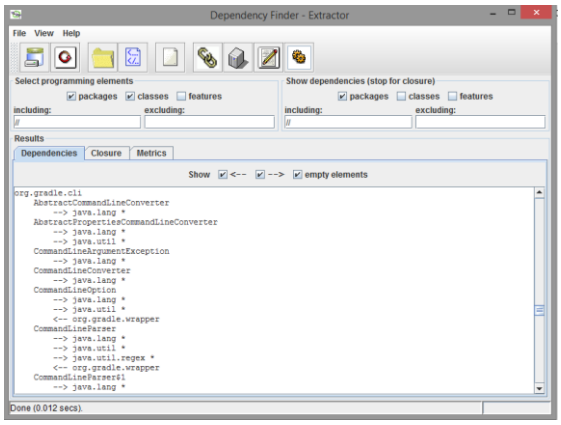
รูปที่ ข.2 หน้าจอโปรแกรมดีเพนเดนซีไฟน์เดอร์

5. รูปที่ ข.3 ตัวอย่าง**การ**นำเข้าไฟล์ซอฟต์แวร์ นามสกุล .zip ในเครื่องมือ DependencyFinder



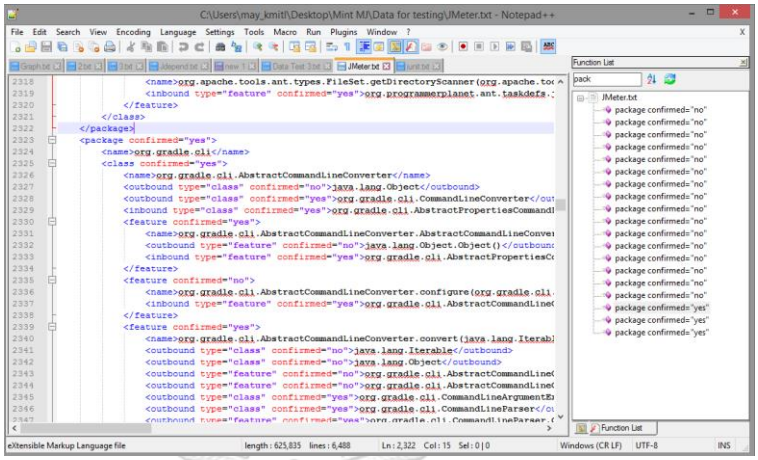
รูปที่ ข.3 การนำเข้าซอฟต์แวร์

6. กราฟพึ่งพาที่ได้จะมีลักษณะเป็นไฟล์ .xml หลังการเลือกบันทึกจากเครื่องมือดีเพนเดนซีไฟน์เดอร์



รูปที่ ข.4 กราฟพึ่งพาที่ได้จากเครื่องมือก่อนนำออก

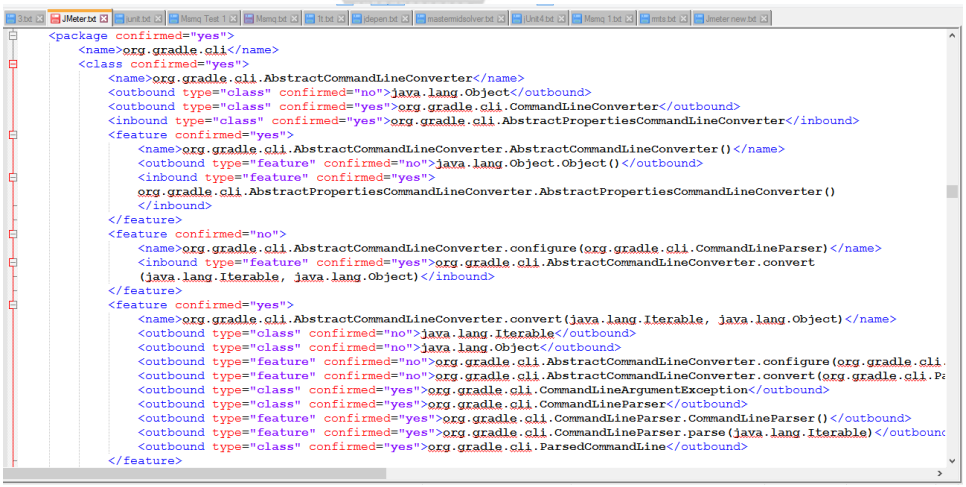
7. นำข้อมูลกราฟที่ได้จากเครื่องมือเป็นไฟล์นามสกุล .xml ซึ่งจะมีรูปแบบดังรูปที่ ข.5 มาเพื่อจัดรูปแบบให้เป็นตามข้อกำหนด



รูปที่ ข.5 ไฟล์ตัวอย่างกราฟพึ่งพาที่ได้จากเครื่องมือ นำมาแปลงเป็นรูปแบบข้อมูลที่ต้องการ

8. เมื่อได้ไฟล์มาแล้วนำมาสกัดเอาข้อมูลโดยนับคลาสที่มีความสัมพันธ์อยู่ในแพ็กเกจ โดยมีวิธีการนับข้อมูลดังต่อไปนี้

1.) เปิดไฟล์ข้อมูลกราฟ .xml ในโปรแกรม Notepad++ ข้อมูลกราฟพึ่งพามีลักษณะตามรูปที่ ข.6



```
<package confirmed="yes">
    <name>org.gradle.cli</name>
    <class confirmed="yes">
        <name>org.gradle.cli.AbstractCommandLineConverter</name>
        <outbound type="class" confirmed="no">java.lang.Object</outbound>
        <outbound type="class" confirmed="yes">org.gradle.cli.CommandLineConverter</outbound>
        <inbound type="class" confirmed="yes">org.gradle.cli.AbstractPropertiesCommandLineConverter</inbound>
        <feature confirmed="yes">
            <name>org.gradle.cli.AbstractCommandLineConverter.AbstractCommandLineConverter()</name>
            <outbound type="feature" confirmed="no">java.lang.Object.Object()</outbound>
            <inbound type="feature" confirmed="yes">
            org.gradle.cli.AbstractPropertiesCommandLineConverter.AbstractPropertiesCommandLineConverter()
            </inbound>
        </feature>
        <feature confirmed="no">
            <name>org.gradle.cli.AbstractCommandLineConverter.configure(org.gradle.cli.CommandLineParser)</name>
            <inbound type="feature" confirmed="yes">org.gradle.cli.AbstractCommandLineConverter.convert
            (java.lang.Iterable, java.lang.Object)</inbound>
        </feature>
        <feature confirmed="yes">
            <name>org.gradle.cli.AbstractCommandLineConverter.convert(java.lang.Iterable, java.lang.Object)</name>
            <outbound type="class" confirmed="no">java.lang.Iterable</outbound>
            <outbound type="class" confirmed="no">java.lang.Object</outbound>
            <outbound type="feature" confirmed="no">org.gradle.cli.AbstractCommandLineConverter.configure(org.gradle.cli.
            <outbound type="feature" confirmed="no">org.gradle.cli.AbstractCommandLineConverter.convert(org.gradle.cli.Pa
            <outbound type="class" confirmed="yes">org.gradle.cli.CommandLineArgumentException</outbound>
            <outbound type="class" confirmed="yes">org.gradle.cli.CommandLineParser</outbound>
            <outbound type="feature" confirmed="yes">org.gradle.cli.CommandLineParser.CommandLineParser()</outbound>
            <outbound type="feature" confirmed="yes">org.gradle.cli.CommandLineParser.parse(java.lang.Iterable)</outbound
            <outbound type="class" confirmed="yes">org.gradle.cli.ParsedCommandLine</outbound>
        </feature>
```

รูปที่ ข.6 ข้อมูลกราฟพึ่งพา .xml

2.) เรียกดูส่วนที่เป็นข้อมูลแพ็กเกจที่เครื่องมือดีเพนเดนซีไฟน์เดอร์นับว่าเป็นแพ็กเกจ ดังรูป ข.7 จะแสดงแพ็กเกจทั้งหมด และคลาสภายในที่อยู่ในแพ็กเกจ

```
<package confirmed="yes">
    <name>org.gradle.cli</name>
    <class confirmed="yes">
        <name>org.gradle.cli.AbstractCommandLineConverter</name>
        <outbound type="class" confirmed="no">java.lang.Object</outbound>
        <outbound type="class" confirmed="yes">org.gradle.cli.CommandLineConverter</outbound>
        <inbound type="class" confirmed="yes">org.gradle.cli.AbstractPropertiesCommandLineConverter</inbound>
```

รูปที่ ข.7 ข้อมูลกราฟพึ่งพาส่วนที่เป็นข้อมูลในแพ็กเกจ

3.) พิจารณาความสัมพันธ์ของคลาสในแพ็กเกจ และนับจำนวนคลาสที่มีความเกี่ยวข้องกัน

4.) สร้างตารางเมตริกซ์ความสัมพันธ์คลาสพึ่งพา

5.) สร้างตารางแสดงข้อมูลแพ็กเกจและจำนวนคลาสในแต่ละแพ็กเกจ

A	B	C	D	E	F	G	H	I	J	K	L	M
1	AbstractCommandLineConverter	0	1	1	0	0	1	0	0	0	0	0
2	AbstractPropertiesCommandLineConvert	1	0	1	1	1	1	0	0	0	0	0
3	CommandLineArgumentException	1	1	0	1	0	1	0	0	0	1	0
4	CommandLineConverter	1	0	0	0	0	1	0	0	0	0	0
5	CommandLineOption	0	1	0	0	0	1	0	1	0	1	0
6	CommandLineParser	1	1	1	1	1	0	1	1	1	1	0
7	CommandLineParser$1	0	0	0	0	0	1	0	1	1	1	1
8	CommandLineParser$AfterFirstSubCon	0	0	0	0	0	1	1	0	0	0	0
9	CommandLineParser$AfterOptions	0	0	0	0	0	1	1	0	0	0	0
10	CommandLineParser$BeforeFirstSubCc	0	0	0	0	0	1	1	0	0	0	0
11	CommandLineParser$CaseInsensitiveS	0	0	0	0	0	0	1	0	0	0	0
12	CommandLineParser$KnownOptionPar	0	0	1	0	1	0	1	1	0	1	0
13	CommandLineParser$MissingOptionAr	0	0	0	0	0	0	1	0	0	0	0
14	CommandLineParser$OptionAwarePar	0	0	0	0	0	1	1	1	1	1	0
15	CommandLineParser$OptionComparat	0	0	0	0	1	1	1	0	0	0	1
16	CommandLineParser$OptionParserStat	0	0	0	0	0	1	1	1	1	1	0
17	CommandLineParser$OptionString	0	0	0	0	0	0	1	1	0	1	0
18	CommandLineParser$OptionStringCom	0	0	0	0	0	1	1	0	0	0	1
19	CommandLineParser$ParserState	0	0	0	0	0	1	1	0	1	1	0
20	CommandLineParser$UnknownOption	0	0	0	0	0	0	0	1	1	1	0
21	ParsedCommandLine	1	1	0	1	1	1	0	1	1	1	0

รูปที่ ข.8 ตัวอย่างไฟล์ในการเก็บข้อมูลความสัมพันธ์ของคลาส

8. ตัวอย่างข้อมูลที่ได้ ประกอบด้วยข้อมูล 2 ประเภทคือข้อมูลกราฟและข้อมูลแพ็กเกจ ดังรูปที่ ข.9 และ ข.10

```
47
0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

รูปที่ ข.9 รูปแบบข้อมูล Graph.txt

```
3
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24
C25 C26 C27 C28 C29 C30 C31 C32 C33 C34 C35 C36 C37 C38 C39 C40 C41 C42 C43 C44
C45 C46 C47
```

รูปที่ ข.10 รูปแบบข้อมูล Package.txt

ภาคผนวก ค.
ผลการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามโดยละเอียด

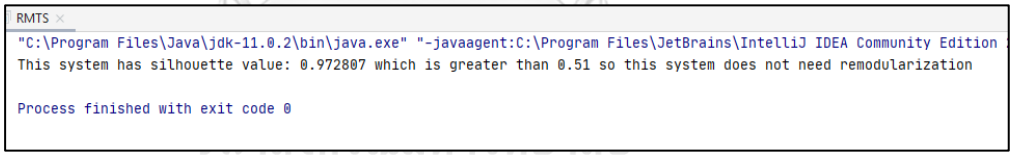
การรีมอดุลาไรเซชันซอฟต์แวร์ด้วยการค้นหาต้องห้าม ในการประเมินผลเครื่องมือนี้ได้ใช้ซอฟต์แวร์เพื่อทำการสอบและประเมินผลจำนวน 7 ตัวอย่าง โดยที่แต่ละตัวอย่าง ซอฟต์แวร์มีรายละเอียดดังต่อไปนี้

ค.1 ซอฟต์แวร์ JMeter

ตารางที่ ค.1 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ JMeter

ชื่อ	JMeter
ขนาด	3 แพ็กเกจ 47 คลาส
ผลจากการวิเคราะห์โดยเครื่องมือ	มีค่าสัมประสิทธิ์ซิลูเอทดีกว่า 0.51
สัมประสิทธิ์ซิลูเอท	0.972

ผลที่ได้จากเครื่องมือเป็นดังรูปที่ ค.1 โดยจะมีค่าสัมประสิทธิ์ปัจจุบันและแสดงข้อความว่าไม่จำเป็นต้องทำการรีมอดุลาไรเซชัน



รูปที่ ค.1 ผลลัพธ์ที่ได้จากเครื่องมือ

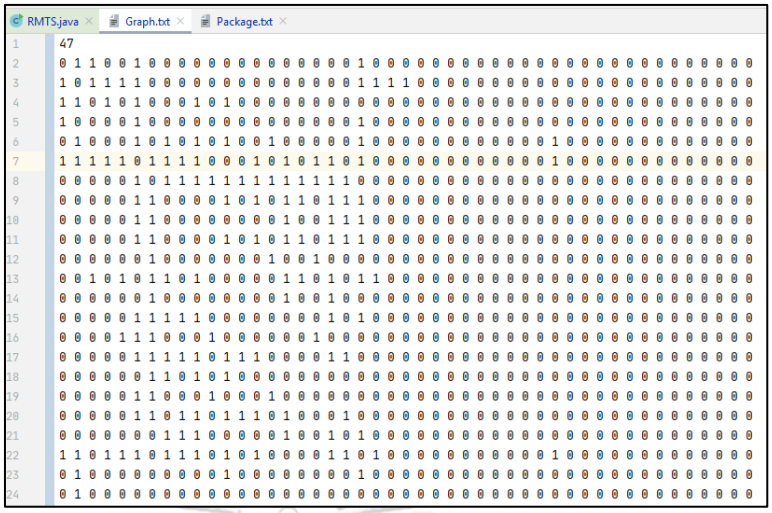
ข้อมูลแพ็กเกจ : 3 แพ็กเกจ 47 คลาส

แพ็กเกจที่ 1 มี 24 คลาส : C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24

แพ็กเกจที่ 2 มี 20 คลาส: C25 C26 C27 C28 C29 C30 C31 C32 C33 C34 C35 C36 C37 C38 C39 C40 C41 C42 C43 C44

แพ็กเกจที่ 3 มี 4 คลาส: C45 C46 C47

ข้อมูลกราฟของซอฟต์แวร์ JMeter เป็นดังรูปที่ ค.2



รูปที่ ค.2 ตัวอย่างข้อมูลกราฟของซอฟต์แวร์ JMeter

รายละเอียดของกราฟพึ่งพาของซอฟต์แวร์ตัวอย่าง JMeter ตาราง ค.2 จะแสดงรายละเอียดคลาส คือชื่อคลาสและแพ็กเกจ

ตารางที่ ค.2 รายละเอียดคลาสของกราฟพึ่งพา JMeter

ลำดับ	ชื่อคลาส	Node No.	Package No.
1	AbstractCommandLineConverter	C1	P1
2	AbstractPropertiesCommandLineConverter	C2	P1
3	CommandLineArgumentException	C3	P1
4	CommandLineConverter	C4	P1
5	CommandLineOption	C5	P1
6	CommandLineParser	C6	P1
7	CommandLineParser$1	C7	P1
8	CommandLineParser$AfterFirstSubCommand	C8	P1
9	CommandLineParser$AfterOptions	C9	P1
10	CommandLineParser$BeforeFirstSubCommand	C10	P1
11	CommandLineParser$CaseInsensitiveStringComparator	C11	P1
12	CommandLineParser$KnownOptionParserState	C12	P1
13	CommandLineParser$MissingOptionArgState	C13	P1
14	CommandLineParser$OptionAwareParserState	C14	P1
15	CommandLineParser$OptionComparator	C15	P1
16	CommandLineParser$OptionParserState	C16	P1
17	CommandLineParser$OptionString	C17	P1

ตารางที่ ค.2 รายละเอียดคลาสของกราฟพึ่งพา JMeter (ต่อ)

ลำดับ	ชื่อคลาส	Node No.	Package No.
18	CommandLineParser$OptionStringComparator	C18	P1
19	CommandLineParser$ParserState	C19	P1
20	CommandLineParser$UnknownOptionParserState	C20	P1
21	ParsedCommandLine	C21	P1
22	ParsedCommandLineOption	C22	P1
23	ProjectPropertiesCommandLineConverter	C23	P1
24	SystemPropertiesCommandLineConverter	C24	P1
25	BootstrapMainStarter	C25	P2
26	BootstrapMainStarter$1	C26	P2
27	Download	C27	P2
28	Download$1	C28	P2
29	Download$DefaultDownloadProgressListener	C29	P2
30	Download$ProxyAuthenticator	C30	P2
31	DownloadProgressListener	C31	P2
32	ExclusiveFileAccessManager	C32	P2
33	GradleUserHomeLookup	C33	P2
34	GradleWrapperMain	C34	P2
35	IDownload	C35	P2
36	Install	C36	P2
37	Install$1	C37	P2
38	Install$InstallCheck	C38	P2
39	Logger	C39	P2
40	PathAssembler	C40	P2
41	PathAssembler$LocalDistribution	C41	P2
42	SystemPropertiesHandler	C42	P2
43	WrapperConfiguration	C43	P2
44	WrapperExecutor	C44	P2
45	Arg	C45	P3
46	JMeterTask	C46	P3
47	Property	C47	P3

ค.2 ซอฟต์แวร์ JUint4

ตารางที่ ค.3 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ JUint 4 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตารางที่ ค.3 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ JUnit4

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	
ขนาด	5 แพ็กเกจ 79 คลาส
แพ็กเกจที่ 1	11 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
แพ็กเกจที่ 2	23 คลาส ; C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30 C31 C32 C33 C34
แพ็กเกจที่ 3	21 คลาส ; C35 C36 C37 C38 C39 C40 C41 C42 C43 C44 C45 C46 C47 C48 C49 C50 C51 C52 C53 C54 C55
แพ็กเกจที่ 4	20 คลาส ; C56 C57 C58 C59 C60 C61 C62 C63 C64 C65 C66 C67 C68 C69 C70 C71 C72 C73 C74 C75
แพ็กเกจที่ 5	4 คลาส ; C76 C77 C78 C79
ค่าสัมประสิทธิ์ซิลูเอท	0.289
ค่าเทอร์โบเอ็มคิว	3.153
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 1
ขนาด	5 แพ็กเกจ 79 คลาส
แพ็กเกจที่ 1	11 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
แพ็กเกจที่ 2	23 คลาส ; C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30 C31 C32 C33 C34
แพ็กเกจที่ 3	21 คลาส ; C35 C37 C38 C39 C40 C41 C42 C43 C44 C45 C46 C47 C48 C49 C50 C51 C52 C53 C54 C55 C67
แพ็กเกจที่ 4	20 คลาส ; C36 C56 C57 C58 C59 C60 C61 C62 C63 C64 C65 C66 C68 C69 C70 C71 C72 C73 C74 C75
แพ็กเกจที่ 5	4 คลาส ; C76 C77 C78 C79
ค่าสัมประสิทธิ์ซิลูเอท	0.436

ตารางที่ ค.3 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ JUnit4 (ต่อ)

ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 2
ขนาด	5 แพ็กเกจ 79 คลาส
แพ็กเกจที่ 1	11 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
แพ็กเกจที่ 2	23 คลาส ; C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30 C31 C32 C33 C34
แพ็กเกจที่ 3	20 คลาส ; C35 C37 C38 C39 C40 C41 C42 C43 C44 C45 C46 C47 C48 C49 C50 C51 C52 C53 C67 C78
แพ็กเกจที่ 4	22 คลาส ; C36 C54 C55 C56 C57 C58 C59 C60 C61 C62 C63 C64 C65 C66 C68 C69 C70 C71 C72 C73 C74 C75
แพ็กเกจที่ 5	3 คลาส ; C76 C77 C79
ค่าสัมประสิทธิ์ซิลูเอท	0.538
ค่าเทอร์โบเอ็มคิว	3.454

รายละเอียดของกราฟพึ่งพาของซอฟต์แวร์ตัวอย่าง JUnit4 ในตาราง ค.4 จะแสดงรายละเอียดคลาส คือชื่อคลาสและแพ็กเกจ

ตาราง ค.4 รายละเอียดคลาสของกราฟพึ่งพา

ลำดับ	ชื่อคลาส	โหนด	ลำดับ	ชื่อคลาส	โหนด
1.	BootstrapMainStarter	**C1**	**11.**	WrapperExecutor	**C11**
2.	DefaultDownloader	**C2**	**12.**	AbstractCommandLineConverter	**C12**
3.	DefaultDownloader$SystemPropertiesProxyAuthenticator	**C3**	**13.**	AbstractPropertiesCommandLineConverter	**C13**
4.	Downloader	**C4**	**14.**	CommandLineArgumentException	**C14**
5.	Installer	**C5**	**15.**	CommandLineConverter	**C15**
6.	MavenWrapperMain	**C6**	**16.**	CommandLineOption	**C16**
7.	PathAssembler	**C7**	**17.**	CommandLineParser	**C17**
8.	PathAssembler$LocalDistribution	**C8**	**18.**	CommandLineParser$AfterFirstSubCommand	**C18**
9.	SystemPropertiesHandler	**C9**	**19.**	CommandLineParser$AfterOptions	**C19**
10.	WrapperConfiguration	**C10**	**20.**	CommandLineParser$BeforeFirstSubCommand	**C20**

ตาราง ค.4 รายละเอียดคลาสของกราฟพึ่งพา (ต่อ)

ลำดับ	ชื่อคลาส	โหนด	ลำดับ	ชื่อคลาส	โหนด
21.	CommandLineParser$CaseInsensitiveString Comparator	**C21**	**51.**	MatcherAssert	**C51**
22.	CommandLineParser$KnownOptionParserS tate	**C22**	**52.**	SelfDescribing	**C52**
23.	CommandLineParser$MissingOptionArgState	**C23**	**53.**	StringDescription	**C53**
24.	CommandLineParser$OptionAwareParserSt ate	**C24**	**54.**	TypeSafeDiagnosingMatcher	**C54**
25.	CommandLineParser$OptionComparator	**C25**	**55.**	TypeSafeMatcher	**C55**
26.	CommandLineParser$OptionParserState	**C26**	**56.**	AllOf	**C56**
27.	CommandLineParser$OptionString	**C27**	**57.**	AnyOf	**C57**
28.	CommandLineParser$OptionStringCompar ator	**C28**	**58.**	CombinableMatcher	**C58**
29.	CommandLineParser$ParserState	**C29**	**59.**	CombinableMatcher$CombinableB othMatcher	**C59**
30.	CommandLineParser$UnknownOptionPars erState	**C30**	**60.**	CombinableMatcher$CombinableEit herMatcher	**C60**
31.	ParsedCommandLine	**C31**	**61.**	DescribedAs	**C61**
32.	ParsedCommandLineOption	**C32**	**62.**	Every	**C62**
33.	ProjectPropertiesCommandLineConverter	**C33**	**63.**	Is	**C63**
34.	SystemPropertiesCommandLineConverter	**C34**	**64.**	IsAnything	**C64**
35.	BaseDescription	**C35**	**65.**	IsCollectionContaining	**C65**
36.	BaseMatcher	**C36**	**66.**	IsEqual	**C66**
37.	Condition	**C37**	**67.**	IsInstanceOf	**C67**
38.	Condition$1	**C38**	**68.**	IsNot	**C68**
39.	Condition$Matched	**C39**	**69.**	IsNull	**C69**
40.	**Condition$NotMatched**	**C40**	**70.**	IsSame	**C70**
41.	Condition$Step	**C41**	**71.**	ShortcutCombination	**C71**
42.	CoreMatchers	**C42**	**72.**	StringContains	**C72**
43.	CustomMatcher	**C43**	**73.**	StringEndsWith	**C73**
44.	CustomTypeSafeMatcher	**C44**	**74.**	StringStartsWith	**C74**
45.	Description	**C45**	**75.**	SubstringMatcher	**C75**
46.	Description$NullDescription	**C46**	**76.**	ArrayIterator	**C76**
47.	DiagnosingMatcher	**C47**	**77.**	ReflectiveTypeFinder	**C77**
48.	Factory	**C48**	**78.**	SelfDescribingValue	**C78**
49.	FeatureMatcher	**C49**	**79.**	SelfDescribingValueIterator	**C79**
50.	Matcher	**C50**			

ค.3 ซอฟต์แวร์ Example 1

ตารางที่ ค.5 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ Example 1 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตาราง ค.5 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 1

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม		
ขนาด	3 แพ็กเกจ 8 คลาส	
แพ็กเกจที่ 1	3 คลาส ; C1 C2 C3	
แพ็กเกจที่ 2	3 คลาส ; C5 C6 C8	
แพ็กเกจที่ 3	2 คลาส ; C4 C7	
ค่าสัมประสิทธิ์ซิลูเอท	0.056	
ค่าเทอร์โบเอ็มคิว	1.167	
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม		การค้นหารอบที่ 1
ขนาด	2 แพ็กเกจ 8 คลาส	
แพ็กเกจที่ 1	4 คลาส ; C1 C2 C3 C7	
แพ็กเกจที่ 2	4 คลาส ; C4 C5 C6 C8	
ค่าสัมประสิทธิ์ซิลูเอท	0.604	
ค่าเทอร์โบเอ็มคิว	1.24	

ค.4 ซอฟต์แวร์ Example 2

ตารางที่ ค.6 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ Example 2 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตาราง ค.6 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 2

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	
ขนาด	4 แพ็กเกจ 16 คลาส
แพ็กเกจที่ 1	8 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8
แพ็กเกจที่ 2	2 คลาส ; C9 C10
แพ็กเกจที่ 3	3 คลาส ; C11 C12 C13
แพ็กเกจที่ 4	3 คลาส ; C14 C15 C16
ค่าสัมประสิทธิ์ซิลูเอท	0.325

ตาราง ค.6 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 2 (ต่อ)

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	
ค่าเทอร์โบเอ็มคิว	2.30
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 1
ขนาด	4 แพ็กเกจ 16 คลาส
แพ็กเกจที่ 1	10 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C15 C16
แพ็กเกจที่ 2	2 คลาส ; C9 C10
แพ็กเกจที่ 3	3 คลาส ; C11 C12 C13
แพ็กเกจที่ 4	1 คลาส ; C14
ค่าสัมประสิทธิ์ซิลูเอท	0.512
ค่าเทอร์โบเอ็มคิว	2.389

ค.5 ซอฟต์แวร์ Example 3

ตารางที่ ค.7 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ Example 3 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตาราง ค.7 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 3

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	
ขนาด	3 แพ็กเกจ 12 คลาส
แพ็กเกจที่ 1	3 คลาส ; C1 C2 C3
แพ็กเกจที่ 2	5 คลาส ; C4 C5 C6 C7 C8
แพ็กเกจที่ 3	4 คลาส ; C9 C10 C11 C12
ค่าสัมประสิทธิ์ซิลูเอท	0.161
ค่าเทอร์โบเอ็มคิว	1.587
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 1
ขนาด	3 แพ็กเกจ 12 คลาส
แพ็กเกจที่ 1	3 คลาส ; C1 C2 C3
แพ็กเกจที่ 2	5 คลาส ; C4 C5 C6 C7 C8
แพ็กเกจที่ 3	4 คลาส ; C9 C10 C11 C12
ค่าสัมประสิทธิ์ซิลูเอท	0.542

ตาราง ค.7 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 3 (ต่อ)

ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 1
ค่าเทอร์โบเอ็มคิว	2.017

ค.6 ซอฟต์แวร์ Example 4

ตารางที่ ค.8 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ Example 4 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตาราง ค.8 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 4

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	
ขนาด	3 แพ็กเกจ 13 คลาส
แพ็กเกจที่ 1	3 คลาส ; C1 C2 C3
แพ็กเกจที่ 2	5 คลาส ; C4 C5 C6 C7 C8
แพ็กเกจที่ 3	4 คลาส ; C9 C10 C11 C12
ค่าสัมประสิทธิ์ซิลูเอท	0.453
ค่าเทอร์โบเอ็มคิว	2.051
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม	การค้นหารอบที่ 1
ขนาด	3 แพ็กเกจ 13 คลาส
แพ็กเกจที่ 1	5 คลาส ; C1 C2 C3 C4 C12
แพ็กเกจที่ 2	5 คลาส ; C5 C6 C7 C8 C6
แพ็กเกจที่ 3	3 คลาส ; C10 C11 C13
ค่าสัมประสิทธิ์ซิลูเอท	0.667
ค่าเทอร์โบเอ็มคิว	2.341

ค.7 ซอฟต์แวร์ Example 5

ตารางที่ ค.9 จะอธิบายรายละเอียดการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้ามของซอฟต์แวร์ Example 5 ทั้งก่อนและหลังการรีมอดุลาไรเซชัน และการปรับปรุงการค้นหา

ตาราง ค.9 ข้อมูลการรีมอดุลาไรเซชันซอฟต์แวร์ Example 5

ข้อมูลก่อนการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม		
ขนาด	4 แพ็กเกจ 16 คลาส	
แพ็กเกจที่ 1	8 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8	
แพ็กเกจที่ 2	2 คลาส ; C9 C10	
แพ็กเกจที่ 3	3 คลาส ; C11 C12 C13	
แพ็กเกจที่ 4	3 คลาส ; C14 C15 C16	
ค่าสัมประสิทธิ์ซิลูเอท	0.257	
ค่าเทอร์โบเอ็มคิว	2.252	
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม		การค้นหารอบที่ 1
ขนาด	4 แพ็กเกจ 16 คลาส	
แพ็กเกจที่ 1	9 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C14	
แพ็กเกจที่ 2	2 คลาส ; C9 C10	
แพ็กเกจที่ 3	3 คลาส ; C11 C12 C13	
แพ็กเกจที่ 4	2 คลาส ; C15 C16	
ค่าสัมประสิทธิ์ซิลูเอท	0.275	
ข้อมูลหลังการรีมอดุลาไรเซชันด้วยการค้นหาต้องห้าม		การค้นหารอบที่ 2
ขนาด	3 แพ็กเกจ 16 คลาส	
แพ็กเกจที่ 1	11 คลาส ; C1 C2 C3 C4 C5 C6 C7 C8 C14 C15 C16	
แพ็กเกจที่ 2	2 คลาส ; C9 C10	
แพ็กเกจที่ 3	3 คลาส ; C11 C12 C13	
ค่าสัมประสิทธิ์ซิลูเอท	0.716	
ค่าเทอร์โบเอ็มคิว	2.404	

ประวัติผู้เขียน

ชื่อ-สกุล	พจนารถ จันทน์วัฒนวงษ์
วัน เดือน ปี เกิด	9 พฤศจิกายน 2536
สถานที่เกิด	จ. อุดรธานี
วุฒิการศึกษา	ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ที่อยู่ปัจจุบัน	47/1 หมู่ 5 ตำบลศรีสุทโธ อำเภอบ้านดุง จังหวัดอุดรธานี 41190