การออกแบบและพัฒนาวิธีการค้นหาข้อบกพร่องของโมเดลการออกแบบซอฟต์แวร์ด้วยแผนภาพกราฟและแผนภาพต้นไม้

นางสาวณัฏฐา เยาวรัตนประเสริฐ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

DESIGN AND DEVELOPMENT OF AN APPROACH FOR DETECTING FLAWS IN SOFTWARE DESIGN MODEL USING GRAPH DIAGRAM AND TREE DIAGRAM

Miss Nattha Yaowarattanaprasert

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2013

หัวข้อวิทยานิพนธ์	การออกแบบและพัฒนาวิธีการค้นหาข้อบกพร่องของโมเดลการออกแบบซอฟต์แวร์ด้วยแผนภาพกราฟและแผนภาพต้นไม้
โดย	นางสาวณัฏฐา เยาวรัตนประเสริฐ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ นครทิพย์ พร้อมพูล)

.....กรรมการภายนอกมหาวิทยาลัย

(ผู้ช่วยศาสตราจารย์ ดร. มฑุปายาส ทองมาก)

ณัฏฐา เยาวรัตนประเสริฐ : การออกแบบและพัฒนาวิธีการค้นหาข้อบกพร่องของโมเดลการออกแบบซอฟต์แวร์ด้วยแผนภาพกราฟและแผนภาพต้นไม้. (DESIGN AND DEVELOPMENT OF AN APPROACH FOR DETECTING FLAWS IN SOFTWARE DESIGN MODEL USING GRAPH DIAGRAM AND TREE DIAGRAM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.พรศิริ หมื่นไชยศรี, 101 หน้า.

ความถูกต้องของการออกแบบซอฟต์แวร์ถือได้ว่ามีความสำคัญต่อกระบวนการในการพัฒนาเป็นอย่างมาก เอกสารโมเดลการออกแบบจะเป็นตัวกำหนดแนวทางให้กับกระบวนการต่อๆไปในการพัฒนา ข้อบกพร่องที่เกิดในโมเดลการออกแบบซอฟต์แวร์จะทำให้คุณภาพของซอฟต์แวร์ลดลง ข้อบกพร่องดังกล่าวเกิดได้จากการที่รหัสต้นฉบับมีโครงสร้างคลาสที่ไม่ดี มีโครงสร้างการสืบทอดที่ไม่ดี เกิดการซ้ำกันในหลายๆ ส่วนของซอฟต์แวร์ทำให้ซอฟต์แวร์มีขนาดใหญ่เกินความจำเป็น และมีการขึ้นต่อกันสูง นอกจากจะทำให้คุณภาพของซอฟต์แวร์ลดลงแล้ว ยังเป็นผลให้การทำความเข้าใจในเวลาบำรุงรักษาทำได้ยากขึ้นอีกด้วย

จากผลกระทบดังกล่าวข้างต้นนั้นเอง งานวิจัยนี้ได้นำเสนอวิธีการ และเครื่องมือในการค้นหาข้อบกพร่องในโมเดลการออกแบบซอฟต์แวร์ ประกอบไปด้วยข้อกำหนดของแผนภาพแสดงโมเดลการออกแบบ และแม่แบบของข้อบกพร่อง ครอบคลุมแม่แบบของข้อบกพร่องทั้งสิบประเภทด้วยกัน

โมเดลการออกแบบที่ประกอบไปด้วยแผนภาพคลาส ซึ่งจะให้ข้อมูลโครงสร้างของคลาสและส่วนประกอบของคลาส และแผนภาพซีเควนซ์ ซึ่งจะให้ข้อมูลความสัมพันธ์ระหว่างคลาส จะถูกแปลงด้วยข้อกำหนดของแผนภาพแสดงโมเดลการออกแบบ ในการค้นหาข้อบกพร่องจะนำแม่แบบของข้อบกพร่องมาเปรียบเทียบกับแผนภาพ

งานวิจัยนี้มีการประเมินผล ด้วยการพัฒนาเครื่องมือขึ้นตามวิธีการที่ได้นำเสนอ และนำไปทดสอบกับโมเดลการออกแบบซอฟต์แวร์ขนาดเล็ก 3 โมเดล ซึ่งให้ผลออกมาว่าสามารถนำไปใช้ค้นหาข้อบกพร่องได้ตามแม่แบบ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5371411621 : MAJOR SOFTWARE ENGINEERING

KEYWORDS : DETECTION APPROACH / DESIGN FLAW PATTERNS / PATTERN MATCHING

NATTHA YAOWARATTANAPRASERT : DESIGN AND DEVELOPMENT OF AN APPROACH FOR DETECTING FLAWS IN SOFTWARE DESIGN MODEL USING GRAPH DIAGRAM AND TREE DIAGRAM. ADVISOR : ASSOC. PROF. PORNSIRI MEUNCHAISRI, Ph.D., 101 pp.

Quality of software is very significant because the design gives the direction of how to implement the whole software. Design flaws have a direct impact to the quality of software. Flaws may occur from having a bad class structure, having a bad inheritance, or having duplications. Flaws may affect software to be oversized with high coupling.

From the impact description above, this research introduces a method to detect flaws in design model. Using Graph and Tree diagram, the proposed approach composes of Representation model definitions and Flaw patterns, covering 10 flaws.

The input design data, class diagram and sequence diagrams, is transformed using the given definitions. The class diagram gives information about class structure and its inheritance, while Sequence diagram gives information about classes' interaction. In detecting flaws, flaw patterns are used to check against the representation model.

For approach evaluation, a tool is implemented according to the proposed approach. It is used to detect flaws in 3 small size software design models. The results show that the proposed approach is able to detect flaws according to the flaw patterns.

Department : Computer Engineering Student's Signature .....

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

Academic Year : 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจาก รศ.ดร. พรศิริ หมื่นไชยศรี ในฐานะที่ปรึกษาวิทยานิพนธ์หลัก ขอกราบขอบพระคุณอาจารย์ที่ได้ให้คำปรึกษาวิทยานิพนธ์ในทุกๆด้าน กราบขอบพระคุณสำหรับคำแนะนำ และข้อเสนอแนะต่างๆตลอดช่วงระยะเวลาการทำวิจัย

ขอกราบขอบพระคุณ รศ.ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา ประธานกรรมการ ผู้ช่วยศาสตราจารย์ นครทิพย์ พร้อมพูล และ ดร.มฑุปายาส ทองมาก คณะกรรมการ ที่ได้สละเวลาในการเข้าร่วมสอบ และให้คำแนะนำวิทยานิพนธ์นี้เป็นอย่างดียิ่ง

และขอขอบคุณเพื่อนๆ และครอบครัว ที่เป็นกำลังใจ และแรงสนับสนุน

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

หน้า

สารบัญรูป

หน้า

หน้า

หน้า

หน้า

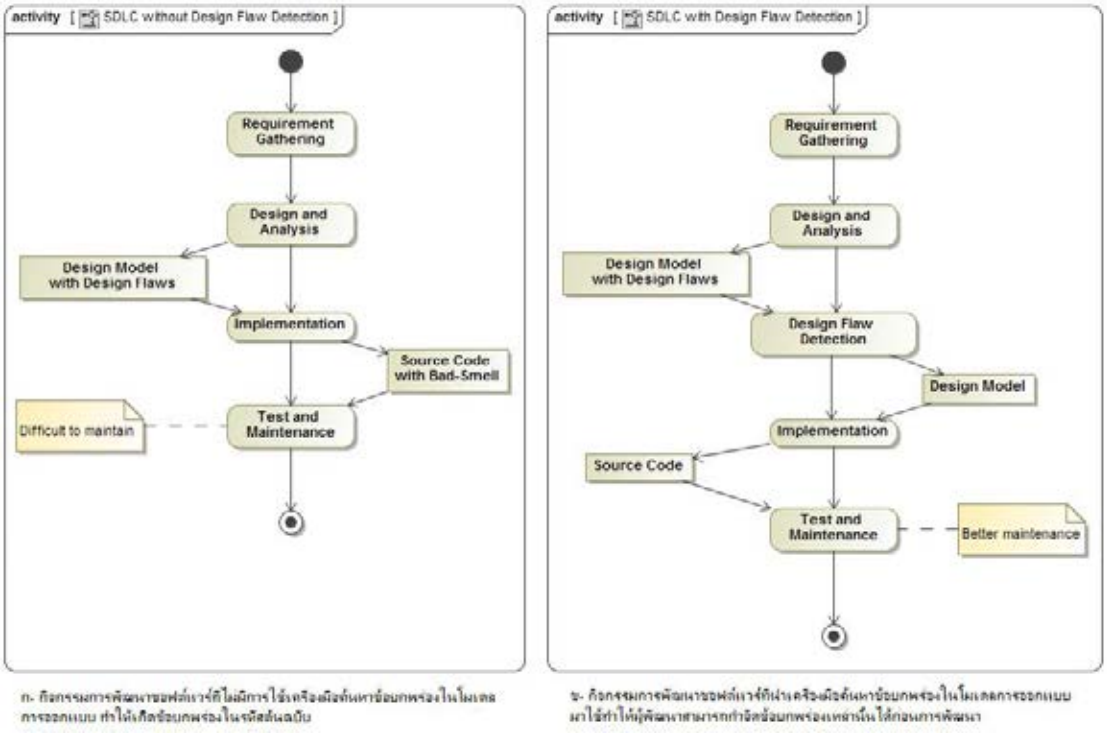
หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

คุณภาพของการออกแบบซอฟต์แวร์ถือได้ว่ามีความสำคัญมาก เพราะเอกสารโมเดลการออกแบบ จะเป็นตัวกำหนดแนวทางให้กับกระบวนการต่อๆไปในการพัฒนา จากรูปที่ 1.1 จะเห็นว่าหลังจากทำการเก็บความต้องการในการพัฒนาซอฟต์แวร์แล้ว เอกสารโมเดลการออกแบบจะถูกจัดทำขึ้นมา เอกสารดังกล่าวจะถูกนำไปพัฒนาในกระบวนการต่อๆไป ภายในเอกสารประกอบไปด้วยรายละเอียดการออกแบบ แผนภาพคลาส (Class Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) และคำอธิบาย



รูปที่ 1.1 เปรียบเทียบแผนภาพกิจกรรมเมื่อใช้และไม่ใช้เครื่องมือช่วยในการค้นหาข้อบกพร่องในโมเดลการออกแบบ

รูปที่ 1.1ก แสดงให้เห็นว่าการนำแม่แบบที่มีข้อบกพร่องไปพัฒนา จะทำให้เกิดเป็นข้อบกพร่องขึ้นในรหัสต้นฉบับต่อไป ข้อบกพร่องเหล่านั้นจะทำให้คุณภาพของซอฟต์แวร์ลดลง [8] เกิดได้จากการที่รหัสต้นฉบับมีโครงสร้างคลาสที่ไม่ดี มีโครงสร้างการสืบทอดที่ไม่ดี เกิดการซ้ำ

กันในหลายๆ ส่วนของซอฟต์แวร์ทำให้ซอฟต์แวร์มีขนาดใหญ่เกินความจำเป็น และมีการขึ้นต่อกันสูง ข้อบกพร่องดังกล่าวเป็นผลให้คุณภาพของซอฟต์แวร์ลดลง และการทำความเข้าใจในเวลาบำรุงรักษาทำได้ยากขึ้น จากผลกระทบต่างๆ เหล่านี้เอง ทำให้มีงานวิจัยที่มีจุดประสงค์ในการค้นหาข้อบกพร่องเหล่านั้นเกิดขึ้นมา รูปที่ 1.1ข แสดงกระบวนการพัฒนาซอฟต์แวร์ที่เพิ่มขั้นตอนของการค้นหาข้อบกพร่องในโมเดลการออกแบบ ซึ่งจะทำให้ผู้ออกแบบสามารถตรวจพบ และกำจัดข้อบกพร่องออกไปได้ก่อนที่จะนำโมเดลการออกแบบไปพัฒนา

การค้นหาข้อบกพร่องเริ่มมาจากการกำหนดรูปแบบของข้อบกพร่อง [8] และอาศัยความสามารถของผู้เชี่ยวชาญ นำรูปแบบเหล่านั้นไปค้นหาเปรียบเทียบในรหัสต้นฉบับ วิธีการดังนี้จะให้ผลที่ไม่แน่นอนและใช้เวลามาก โดยเฉพาะกับซอฟต์แวร์ในปัจจุบันที่มีขนาดที่ใหญ่ ด้วยเหตุนี้เองจึงได้มีผู้นำเสนอวิธีการในการค้นหาข้อบกพร่องแบบอัตโนมัติขึ้นมา งานวิจัยที่ใช้ความรู้ด้านมาตรวัดซอฟต์แวร์ในการค้นหาข้อบกพร่อง ได้แก่ งานวิจัยของ Munro [1] Marinescu [2] Pienlert [3] Salehie, Li, Tahvildari [12] Guo, Seamon, Zazworka, Shull [15] และ Sahraoui, Miceli, Godin [11] งานวิจัยที่นำภาษาโปรล็อก (Prolog) และกฎการอนุมานมาประยุกต์ใช้ ได้แก่ งานวิจัยของ Kanti Yeesoon [4] และ งานวิจัยของ Hassaine, Khomh, Gueheneuc และ Hamel [13] ได้นำวิธีการอุปมามาใช้ โดยจะเป็นการอุปมาข้อบกพร่องเป็นเสมือนเชื้อโรค และอุปมาการค้นหาข้อบกพร่องเสมือนกับการทำงานของภูมิคุ้มกันเชื้อโรคในการค้นหาและกำจัดเชื้อโรคออกจากร่างกายมนุษย์

นอกจากงานวิจัยในการค้นหาข้อบกพร่องในรหัสต้นฉบับแล้วนั้น งานวิจัยอีกส่วนได้หันมาให้ความสำคัญในการค้นหาข้อบกพร่องในการออกแบบซอฟต์แวร์ ซึ่งมีวัตถุประสงค์ที่จะสามารถลดข้อผิดพลาดในขั้นตอนการออกแบบซึ่งเป็นขั้นตอนต้นๆ ในกระบวนการพัฒนาซอฟต์แวร์ Moha, Gueheneuc, Leduc และ Le Meur [6] ได้นำเสนอวิธีการสร้างตรรกะในการตรวจจับข้อบกพร่องแบบอัตโนมัติในการออกแบบ ซึ่งต่อมาพวกเขาได้ทำการวิจัยต่อแล้วนำเสนอเป็นกระบวนการที่เรียกว่า DÉCOR [7] ซึ่งประกอบไปด้วยการกำหนดนิยามและวิธีการค้นหาข้อบกพร่องในรหัสต้นฉบับจากนิยาม ในขณะที่ Maneerat [5] ได้นำเสนอวิธีการค้นหาข้อบกพร่องในการออกแบบด้วยวิธีการเรียนรู้ของเครื่อง 7 วิธีด้วยกัน โดยใช้การทดสอบโดยการหานัยสำคัญทางสถิติ (Statistic Significant) ในการเปรียบเทียบประสิทธิภาพของแต่ละวิธี

จะเห็นได้จากงานวิจัยที่ผ่านมาว่าการนำมาตรวัดซอฟต์แวร์มาใช้นั้นเป็นรากฐานของงานวิจัยที่เกี่ยวกับการค้นหาข้อบกพร่องทั้งในรหัสต้นฉบับ และโมเดลการออกแบบ งานวิจัยที่นำเสนอนี้ได้นำมาตรวัดซอฟต์แวร์มาใช้ร่วมกับแผนภาพต้นไม้และแผนภาพกราฟ เพื่อค้นหาข้อบกพร่องที่เกี่ยวข้องกับโครงสร้างและความสัมพันธ์กันในโมเดลการออกแบบซอฟต์แวร์ ข้อบกพร่องเกี่ยวกับโครงสร้างที่ค้นหาได้ ได้แก่ Parallel Inheritance Hierarchy, Data Class, Large

Class, Lazy Class และ Long Parameter List และข้อบกพร่องเกี่ยวกับความสัมพันธ์ในการเรียกระหว่างคลาส ได้แก่ Refused Bequest, Feature Envy, Middle Man, Message Chain และ Inappropriate Intimacy ซึ่งประเภทของข้อบกพร่องที่วิธีการนี้ค้นหาได้จะแตกต่างไปจากวิธีการเรียนรู้ของเครื่อง แผนภาพต้นไม้จะถูกนำมาใช้ในการเก็บข้อมูลโครงสร้างการสืบทอดและโครงสร้างของสมาชิกภายในคลาสส่วนแผนภาพกราฟจะถูกนำมาใช้ในการเก็บข้อมูลความสัมพันธ์ในการเรียกระหว่างคลาส

1.2 วัตถุประสงค์

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอวิธีการและเครื่องมือในการค้นหาข้อบกพร่องในโมเดลการออกแบบซอฟต์แวร์ ประกอบไปด้วยแผนภาพแสดงโมเดลการออกแบบ 1) แผนภาพต้นไม้การสืบทอด 2) แผนภาพต้นไม้คุณลักษณะ 3) แผนภาพกราฟตรรกะ และแม่แบบของข้อบกพร่องสิบประเภทด้วยกัน

1.3 ขอบเขตของงานวิจัย

1. งานวิจัยนี้นำแผนภาพกราฟ และแผนภาพต้นไม้มาใช้ในการวิเคราะห์เพื่อค้นหาข้อบกพร่องในโมเดลการออกแบบ โดยไม่มีการตรวจสอบความถูกต้องของวากยสัมพันธ์ (Syntax) และ อรรถศาสตร์ (Semantic) ของแผนภาพ โดยแผนภาพที่นำมาใช้จะต้องมีความถูกต้องทางวากยสัมพันธ์อยู่แล้ว
2. วิธีการที่นำเสนอครอบคลุมการค้นหาข้อบกพร่องทั้งสิ้น 10 ประเภทด้วยกัน ได้แก่
    1. ข้อบกพร่องประเภท Refused Bequest
    2. ข้อบกพร่องประเภท Feature Envy
    3. ข้อบกพร่องประเภท Middle Man
    4. ข้อบกพร่องประเภท Message Chain
    5. ข้อบกพร่องประเภท Inappropriate Intimacy
    6. ข้อบกพร่องประเภท Lazy Class
    7. ข้อบกพร่องประเภท Data Class
    8. ข้อบกพร่องประเภท Long Parameter List
    9. ข้อบกพร่องประเภท Large Class
    10. ข้อบกพร่องประเภท Parallel Inheritance Hierarchy

3. งานวิจัยนี้ใช้ข้อมูลการออกแบบจากแผนภาพคลาส และแผนภาพซีเควนซ์ ในขั้นตอนของการออกแบบรายละเอียด (Detailed Design) อ้างอิงตามมาตรฐานยูเอ็มแอล 2.2 สามารถค้นหาข้อบกพร่องที่เกี่ยวข้องกับโครงสร้างการสืบทอด โครงสร้างคุณลักษณะของคลาส การเรียกเมท็อด โดยไม่ครอบคลุมการเรียกคุณลักษณะภายในคลาส และไม่ครอบคลุมถึงการทำซ้ำในแผนภาพซีเควนซ์ที่ไม่ได้ระบุจำนวนเป็นตัวเลข
4. พัฒนาเครื่องมือด้วยภาษาจาวา และทดสอบบนระบบปฏิบัติการวินโดวส์ (Windows)
5. ในการประเมินความสามารถของวิธีการและเครื่องมือ จะทำกรณีศึกษากับโมเดลการออกแบบซอฟต์แวร์ จำนวน 3 ระบบด้วยกัน โดยในแต่ละโมเดลการออกแบบ จะมีจำนวนคลาสตั้งแต่ 5 คลาสขึ้นไปและประกอบไปด้วยแผนภาพคลาสและแผนภาพซีเควนซ์

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. ศึกษาทฤษฎีพื้นฐานของข้อบกพร่องในโมเดลการออกแบบ
2. วิเคราะห์คุณสมบัติที่จะนำมาใช้ในการค้นหาข้อบกพร่องแต่ละประเภท และนำคุณสมบัติที่ได้มาออกแบบวิธีการในการค้นหาข้อบกพร่องจากแผนภาพกราฟและแผนภาพต้นไม้
3. พัฒนาเครื่องมือตามวิธีการที่ได้นำเสนอ
4. ประเมินความถูกต้องวิธีการและเครื่องมือ ด้วยการทำกรณีศึกษากับโมเดลการออกแบบซอฟต์แวร์ขนาดเล็ก
5. วิเคราะห์ผลการทดลอง
6. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำวิธีการ และเครื่องมือที่นำเสนอไปใช้ในการค้นหาข้อบกพร่องในโมเดลการออกแบบ เพื่อเป็นแนวทางให้ผู้พัฒนานำไปปรับปรุงการออกแบบ และกำจัดข้อบกพร่องดังกล่าวก่อนนำการออกแบบนั้นไปพัฒนาต่อ

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

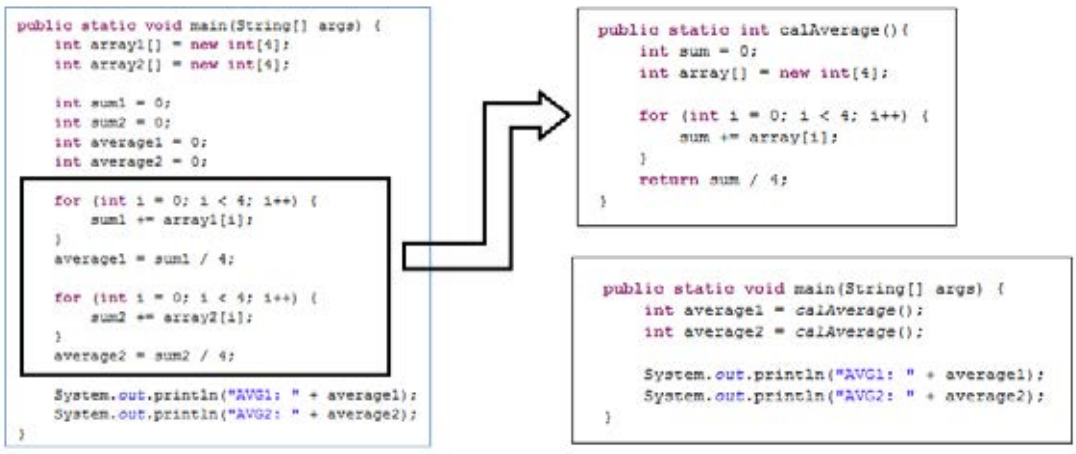
งานวิจัยนี้ได้นำทฤษฎีการคำนวณที่เกี่ยวข้องกับแผนภาพกราฟ และแผนภาพต้นไม้มาช่วยในการค้นหาข้อบกพร่องในโมเดลการออกแบบ โดยมีทฤษฎีที่เกี่ยวข้อง ดังนี้

2.1.1 ข้อบกพร่องในโมเดลการออกแบบ [8]

ข้อบกพร่อง ได้แก่ ลักษณะของการออกแบบซอฟต์แวร์ที่ไม่ดี โดยเกิดได้จากการที่รหัสต้นฉบับมีโครงสร้างคลาสที่ไม่ดี มีโครงสร้างการสืบทอดที่ไม่ดี เกิดการซ้ำกันในหลายๆ ส่วนของซอฟต์แวร์ทำให้มีขนาดใหญ่เกินความจำเป็น และมีการขึ้นต่อกันสูง อันจะทำให้เกิดผลกระทบต่อคุณภาพ และการบำรุงรักษาซอฟต์แวร์ ข้อบกพร่องแบ่งออกได้เป็น 22 ประเภท ดังนี้

1. Duplicate Code

Duplicate Code เป็นข้อบกพร่องที่เกิดจากการมีลักษณะโครงสร้างของซอฟต์แวร์ หรือส่วนของใดๆก็ตามที่ซ้ำกันตั้งแต่สองส่วนขึ้นไป ดังแสดงในรูปที่ 2.1 ซึ่งจะเห็นได้ว่าในการคำนวณหาค่าเฉลี่ยของตัวเลขนั้น มีส่วนของการซ้ำกันของรหัสต้นฉบับอยู่ ควรแก้ไขด้วยการย้ายส่วนที่ทำหน้าทีคำนวณออกไปเป็นเมท็อดแทน ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 ข้อบกพร่องประเภท Duplicate Code

2. Long Method

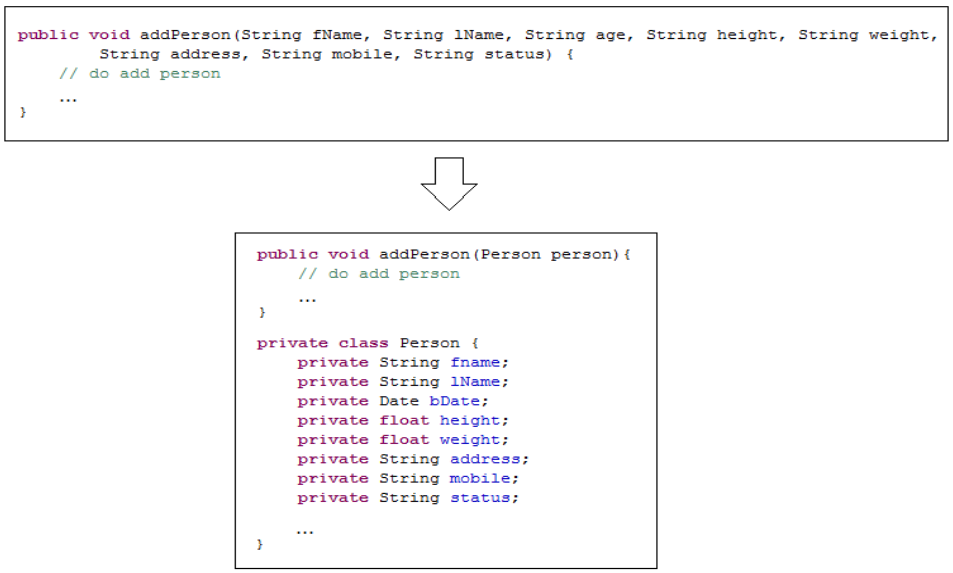
ข้อบกพร่องประเภท Long Method เกิดในลักษณะของเมท็อดที่มีขนาดใหญ่ หรือมีจำนวนบรรทัดมาก ทำให้การทำความเข้าใจเมท็อดทำได้ยาก ควรแก้ไขโดยการย้ายส่วนของรหัสต้นฉบับที่ทำหน้าที่อย่างเดียวกันออกเป็นอีกเมท็อดหนึ่งแทน

3. Large Class

ข้อบกพร่องประเภท Large Class เกิดในลักษณะที่มีคลาสที่มีขนาดใหญ่ ประกอบไปด้วยจำนวนตัวแปร อินแสตนซ์ (Instance Variable) จำนวนเมท็อด หรือจำนวนบรรทัด (Line of Code) เป็นจำนวนมาก ทำให้การทำความเข้าใจทำได้ยาก และอาจทำให้เกิดการซ้ำกันของรหัสต้นฉบับได้อีกด้วย

4. Long Parameter List

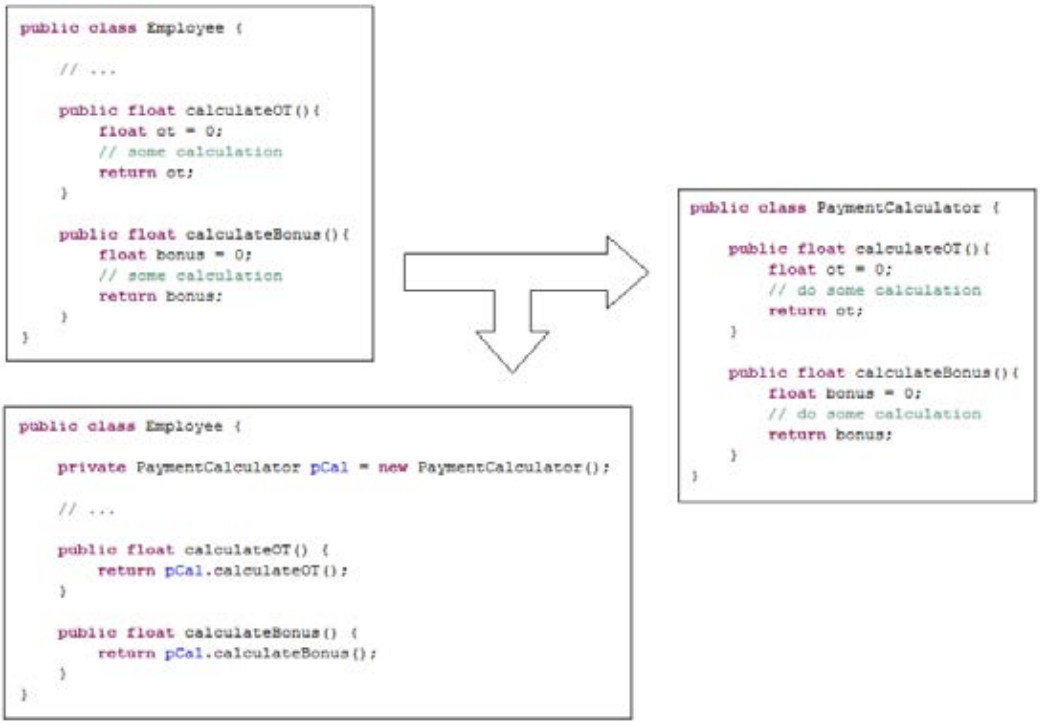
ข้อบกพร่องประเภท Long Parameter List เกิดจากการผ่านพารามิเตอร์เป็นจำนวนมากเข้าสู่เมท็อด ดังแสดงในรูปที่ 2.2 ซึ่งถือว่าไม่ถูกต้องตามหลักการการออกแบบโปรแกรมแบบอ็อบเจ็ค ที่สามารถผ่านค่าพารามิเตอร์ในรูปของอ็อบเจ็ค และใช้การเรียกค่าผ่านอ็อบเจ็คนั้นๆ แทนได้ ซึ่งจะดีกว่าการผ่านค่าพารามิเตอร์เป็นจำนวนมากเข้าไป อันจะทำให้ยากต่อการทำความเข้าใจ และการปรับเปลี่ยนค่าที่ส่งในภายหลัง



รูปที่ 2.2 ข้อบกพร่องประเภท Long Parameter Lists

5. Divergent Change

ข้อบกพร่องประเภท Divergent Change เกิดกับการออกแบบให้คลาสใดๆ ต้องทำหน้าที่หลายๆอย่าง ทำให้เมื่อต้องทำการแก้ไขโปรแกรม จะมีโอกาสเกิดผลกระทบกับคลาสนั้นๆได้มาก ตัวอย่างข้อบกพร่องแสดงในรูปที่ 2.3 จะเห็นได้ว่าคลาส Employee นั้นประกอบไปด้วยเมท็อดที่เกี่ยวข้องกับข้อมูลพนักงาน และการคำนวณค่าล่วงเวลา และเงินโบนัส โดยหากเกิดการเปลี่ยนแปลงสูตรในการคำนวณในภายหลังจะส่งผลกระทบให้ต้องทำการแก้ไขคลาส Employee ควรทำการแก้ไขโดยการย้ายการคำนวณออกเป็นคลาสใหม่



รูปที่ 2.3 ข้อบกพร่องประเภท Divergent Change

6. Shotgun Surgery

ข้อบกพร่องประเภท Shotgun Surgery เกิดกับการออกแบบที่เมื่อมีการเปลี่ยนแปลงใดๆ เกิดขึ้น จะส่งผลกระทบให้ต้องมีการแก้ไขโปรแกรมในหลายๆส่วน ดังแสดงในรูปที่ 2.4 จะเห็นว่ามีการตั้งค่าการเชื่อมต่อฐานข้อมูลไว้ภายในคลาส Person ซึ่งหากออกแบบให้มีการฝังค่าไว้ที่แต่ละคลาส เมื่อต้องการทำการเปลี่ยนแปลงค่าการเชื่อมต่อฐานข้อมูล ทำให้ต้องแก้การตั้งค่าแต่ละที่ ควรย้ายออกเป็นคลาสแยก แล้วเรียกจากคลาสนั้นแทน

```
public class Person {

    public static String CONNECTION_STRING = "jdbc:mysql://localhost:3306/organization";
    public static String USERNAME = "username";
    public static String PASSWORD = "password";
    private Connection con;

    public void makeConnection() throws ClassNotFoundException, SQLException {
        if (con == null || con.isClosed()) {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection(CONNECTION_STRING, USERNAME, PASSWORD);
        }
    }
}
```

รูปที่ 2.4 ข้อบกพร่องประเภท Shotgun Surgery

7. Feature Envy

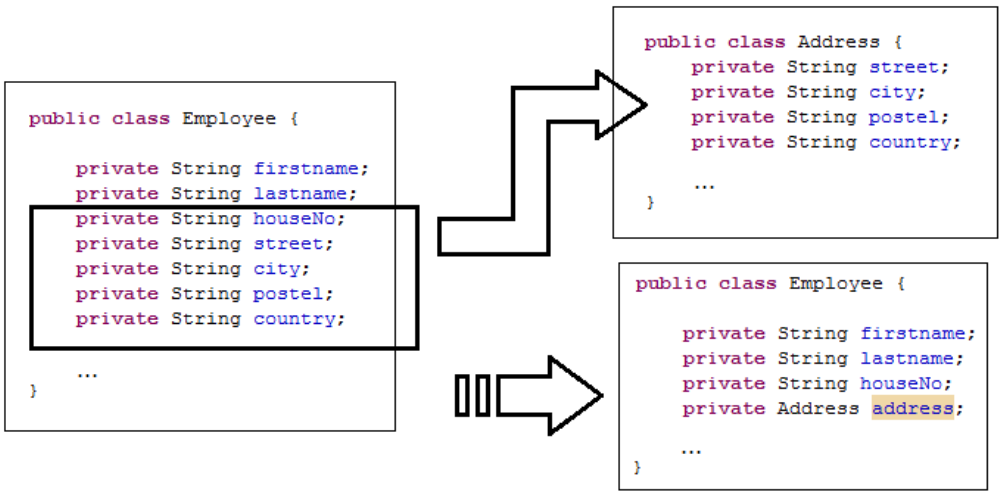
ข้อบกพร่องประเภท Feature Envy นั้นเกิดขึ้นกับเมท็อดที่ได้รับความสนใจ และมีการเรียกใช้จากคลาสอื่น มากกว่าจากคลาสที่ตัวเองนั้นปรากฏอยู่ สำหรับเมท็อดประเภทนี้นั้นอาจเกิดจากข้อบกพร่องในการออกแบบ ซึ่งควรปรากฏอยู่ที่คลาสที่มีการเรียกใช้งานมาก ผู้ออกแบบควรทำการพิจารณาความสำคัญของเมท็อดใหม่

8. Data Clumps

Data Clumps เป็นข้อบกพร่องที่จะพบกลุ่มของตัวแปรที่เกาะกันอยู่หลายๆที่ ซึ่งกลุ่มของตัวแปรดังกล่าวควรจะได้รับการสร้างเป็นคลาสใหม่ขึ้นมา และถูกจัดอยู่รวมกันตามแนวทางการออกแบบอ็อบเจ็ค รูปที่ 2.5 แสดงข้อบกพร่องประเภท Data Clumps จะเห็นว่าในคลาส Employee จะมีกลุ่มของข้อมูลเกี่ยวกับที่อยู่ ซึ่งควรจัดกลุ่มข้อมูลดังกล่าว แล้วสร้างเป็นคลาสแยกสำหรับกลุ่มข้อมูลดังกล่าว

9. Primitive Obsession

มีการใช้ข้อมูลที่มีชนิดเป็นข้อมูลพื้นฐานเป็นจำนวนมาก ซึ่งควรมาเป็นการจัดกลุ่มข้อมูลเหล่านั้นให้เป็นคลาส และใช้ชนิดข้อมูลที่เป็นคลาสแทน



รูปที่ 2.5 ข้อบกพร่องประเภท Data Clumps

10. Switch Statement

สำหรับข้อบกพร่องประเภท Switch Statement นั้น จะเป็นปัญหาในด้านของการซ้ำกันของรหัสต้นฉบับ กล่าวคือ เมื่อมีการใช้ Switch Statement สำหรับข้อมูลชุดเดียวกันหลายๆที่ จะทำให้เกิดการซ้ำกันขึ้น และเมื่อต้องการแก้ไข เปลี่ยนแปลง หรือเพิ่มเติมเงื่อนไขลงไป ผู้พัฒนาจะต้องตามแก้หลายที่ ซึ่งอาจเกิดข้อผิดพลาด แก้ไขไม่ครบได้ ควรทำการออกแบบให้มีคลาสใดคลาสหนึ่งทำหน้าที่ในการควบคุมการสลับเงื่อนไข ตามหลักการของ การพ้องรูป (Polymorphism) ของภาษาตามหลักการเขียนแบบอ็อบเจ็ค

11. Parallel Inheritance Hierarchy

Parallel Inheritance Hierarchy เกิดจากการขึ้นต่อกันของคลาสในการสืบทอด กล่าวคือ สำหรับโครงสร้างการสืบทอดคู่ใดๆ เมื่อต้องมีการสร้างคลาสลูกในโครงสร้างการสืบทอดหนึ่งแล้ว จำเป็นจะต้องทำการสร้างคลาสลูกในอีกโครงสร้างการสืบทอดหนึ่งขึ้นมาด้วย ซึ่งสามารถสังเกตได้จากคำขึ้นต้นคลาสของทั้งสองโครงสร้างจะเหมือน หรือคล้ายคลึงกันนั่นเอง

12. Lazy Class

เนื่องจากคลาสที่ปรากฏอยู่ทุกคลาสนั้นต้องเสียค่าใช้จ่ายในการบำรุงรักษา ข้อบกพร่องประเภท Lazy Class เกิดจากการที่มีคลาสใดๆก็ตามที่มีหน้าที่การทำงานไม่มากปรากฏอยู่ใน

โปรแกรม ผู้ออกแบบควรพิจารณาในการที่จะยุบ หรือรวมคลาสเล็กๆเหล่านั้นเสีย เพื่อลดค่าใช้จ่ายในการดูแลรักษา

13. Speculative Generality

ข้อบกพร่องประเภท Speculative Generality เกิดเมื่อมีการออกแบบเผื่อ ส่วนของการออกแบบที่เกิดขึ้นนั้นไม่มีการใช้งานจริงในเวอร์ชั่นปัจจุบัน ซึ่งทำให้โปรแกรมมีความซับซ้อนมากกว่าที่ควรจะเป็น ควรแก้ไขด้วยการนำส่วนที่ไม่มีการใช้งานออกไปก่อน

14. Temporary Field

Temporary Field เกิดเมื่อมีการใช้งานตัวแปรที่เก็บค่าชั่วคราว ซึ่งทำให้การทำความเข้าใจในภายหลังทำได้ยากขึ้น และต้องใช้เวลานานในการทำความเข้าใจ ถึงความต้องการในการมีอยู่ของตัวแปรชั่วคราวต่างๆ

15. Message Chain

รูปที่ 2.6 แสดงข้อบกพร่องประเภท Message Chain ซึ่งจะเกิดกับการเรียกค่าจากเมท็อด โดยเมื่อมีการเรียกเมท็อดใดๆแล้ว เมท็อดนั้นมีการเรียกต่อไปยังเมท็อดอื่นในอีกคลาสหนึ่งต่อกันเป็นลูกโซ่ ในกรณีนี้ ผู้พัฒนาควรพิจารณาถึงความต้องการในการเรียกเมท็อด และเพื่อทำการย้ายเมท็อดมายังตำแหน่งที่จะประมวลผลค่าที่ต้องการได้โดยตรงมากกว่า

```
public static void main(String[] args) {
    A a = new A();
    a.getB().getC().getValue();
}

private static class A{
    public B getB(){
        return new B();
    }
}

private static class B{
    public C getC(){
        return new C();
    }
}

private static class C{
    public Object getValue(){
        return new Object();
    }
}
```

รูปที่ 2.6 ข้อบกพร่องประเภท Message Chain

16. Middle Man

ข้อบกพร่องประเภท Middle Man เกิดขึ้นกับการเรียกใช้งานเมท็อดผ่านตัวแปรอินแสตนซ์ของคลาสซึ่งเป็นตัวกลางต่อกันเป็นลูกโซ่ไปเรื่อยๆ ดังแสดงในรูปที่ 2.7 ซึ่งจะคล้ายกับกรณีของ Message Chain ซึ่งผู้พัฒนาควรพิจารณาถึงความต้องการที่แท้จริง เพื่อเปลี่ยนมาเรียกตัวแปรอินแสตนซ์ของคลาสที่ทำหน้าที่ประมวลผลโดยตรงแทน

```
public static void main(String[] args) {
    A a = new A();
    a.getValue();
}

private static class A {
    public Object getValue() {
        B b = new B();
        return b.getValue();
    }
}

private static class B {
    public Object getValue() {
        C c = new C();
        return c.getValue();
    }
}

private static class C {
    public Object getValue() {
        return new Object();
    }
}
```

รูปที่ 2.7 ข้อบกพร่องประเภท Middle Man

17. Inappropriate Intimacy

ข้อบกพร่องประเภท Inappropriate Intimacy เกิดขึ้นเมื่อมีการเข้าถึงค่าที่เป็น private ของคลาสอื่น ซึ่งผิดต่อการออกแบบซอฟต์แวร์แบบอ็อบเจ็ค

18. Alternative Class with Different Interfaces

ข้อบกพร่องประเภท Alternative Class with Different Interfaces เกิดเมื่อมีเมท็อดที่ทำหน้าที่เดียวกัน แต่มีชื่อและการรับพารามิเตอร์ที่แตกต่างกัน ซึ่งทำให้ยากต่อการทำความเข้าใจ ควรทำการแก้ไขด้วยการเปลี่ยนชื่อของเมท็อดให้เหมือนกัน

19. Incomplete Library Class

การที่มีการใช้งานคลาสไลบรารี่ (Class Library) ทำให้การทำความเข้าใจคลาสเพื่อแก้ไขหรือนำกลับมาใช้ใหม่นั้นไม่สามารถทำได้

20. Data Class

Data Class เกิดที่คลาสที่มีเพียงคุณลักษณะ (Attribute) และ getter หรือ setter เพียงเท่านั้น โดยไม่มีหน้าที่การทำงานอื่นๆ ซึ่งคลาสลักษณะนี้จะทำหน้าที่เป็นคลาสสำหรับเก็บข้อมูล และจะถูกควบคุมโดยคลาสอื่น ควรทำการแก้ไขด้วยการพิจารณาเพื่อย้ายเมท็อดที่เป็นพฤติกรรมของคลาสไว้ภายในคลาสนั้นๆ

21. Refused Bequest

Refused Bequest เกิดกับโครงสร้างการสืบทอด ที่มีเมท็อดที่ไม่มีความจำเป็นต้องใช้งานได้รับสืบทอดลงมายังคลาสลูก ซึ่งผู้พัฒนาควรพิจารณาโครงสร้างการสืบทอดใหม่ และสามารถแก้ไขได้โดยการเพิ่มคลาสเข้าไปในโครงสร้างเดิมที่ระดับความลึกที่เกิดข้อบกพร่องขึ้น แล้วย้ายเมท็อดดังกล่าวไปยังคลาสที่เพิ่มเข้ามา เพื่อให้เมท็อดดังกล่าวได้สืบทอดไปยังคลาสลูกอื่นๆ ที่มีความต้องการใช้งานเมท็อดนั้น

22. Comment

สำหรับการเขียนคำอธิบายในรหัสต้นฉบับ (Comment) ที่จัดว่าเป็นข้อบกพร่องนั้นเนื่องมาจากว่า รหัสต้นฉบับในส่วนที่ได้รับการใส่คำอธิบายกำกับไว้โดยทั่วไปแล้วแล้วจะส่วนที่มีการพัฒนาไว้ค่อนข้างซับซ้อน จำเป็นต้องมีคำอธิบายกำกับเพื่อช่วยในการทำความเข้าใจในการบำรุงรักษา ซึ่งผู้พัฒนาควรทำการพิจารณาแก้ไขเพื่อลดความซับซ้อน หรือขจัดส่วนที่อาจทำให้เกิดความสับสนได้ โดยไม่ต้องใช้คำอธิบาย

2.1.2 การจัดกลุ่มข้อบกพร่องในโมเดลการออกแบบ [10]

ได้มีผู้ทำการวิจัย โดยการนำข้อบกพร่องทั้ง 22 ประเภทมาจัดกลุ่ม ซึ่งสามารถแบ่งออกได้เป็น 7 กลุ่มด้วยกัน ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 ตารางแสดงข้อบกพร่อง และกลุ่มของข้อบกพร่อง

ประเภท	คำอธิบาย	ข้อบกพร่อง
Bloaters	กลุ่มของข้อบกพร่องที่ประกอบไปด้วยชุดของคำสั่งที่ใหญ่ ทำให้การจัดการเป็นไปอย่างไม่มีประสิทธิภาพ	• Long Method • Large Class • Primitive Obsession • Long Parameter List • Data Clumps
Object-Orientation Abusers	กลุ่มของข้อบกพร่องที่นำการออกแบบเชิงวัตถุมาใช้อย่างไม่เต็มความสามารถ	• Switch Statement • Temporary Field • Refused Bequest • Alternative Class with Different Interfaces • Parallel Inheritance Hierarchy
Change Preventers	กลุ่มของข้อบกพร่องที่ปรากฏโครงสร้างของซอฟต์แวร์ที่แก้ไข เปลี่ยนแปลง หรือบำรุงรักษาได้ยาก	• Divergent Change • Shotgun Surgery
Dispensables	กลุ่มของข้อบกพร่องที่ปรากฏชุดคำสั่งที่ไม่ได้ทำงานอย่างเต็มประสิทธิภาพ ซึ่งควรได้รับการปรับปรุง หรือนำออกไป	• Lazy Class • Data Class • Duplicate Code • Speculative Generality
Encapsulators	กลุ่มของข้อบกพร่องที่เกี่ยวข้องกับกระบวนการในการรับส่งข้อมูลกันระหว่างส่วนต่างๆ ของซอฟต์แวร์ หรือการทำ Encapsulation	• Message Chains • Middle Man
Couplers	กลุ่มของข้อบกพร่องที่ทำให้เกิด Coupling สูง	• Feature Envy • Inappropriate Intimacy

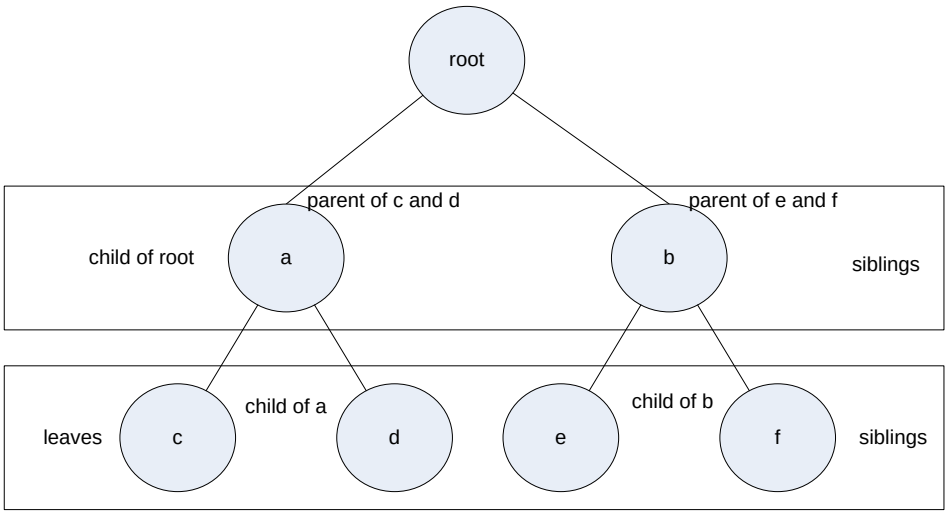
ตารางที่ 2.1 ตารางแสดงข้อบกพร่อง และกลุ่มของข้อบกพร่อง (ต่อ)

ประเภท	คำอธิบาย	ข้อบกพร่อง
Others	กลุ่มของข้อบกพร่องที่นอกเหนือไปจากที่ได้กล่าวมาแล้ว	• Incomplete Library Class • Comment

2.1.3 แผนภาพต้นไม้ (Tree Diagram) [9]

แผนภาพต้นไม้เป็นแผนภาพการจัดเก็บข้อมูลที่มีลักษณะเป็นโครงสร้างโดยไม่มีการวนซ้ำ ข้อมูลแต่ละหน่วยจะถูกจัดเก็บไว้ที่โหนด (node) ซึ่งจะมีความสัมพันธ์ในลักษณะของพ่อกับลูก (parent-child relationship) กับโหนดในระดับที่อยู่ติดกัน แผนภาพต้นไม้จะเริ่มจากรากของต้นไม้ ซึ่งเรียกว่าราก และจะมีส่วนประกอบแสดงในรูปที่ 2 ดังนี้

- ราก (Root) : คือข้อมูลที่เป็นรากของต้นไม้
- โหนด (Node) : ข้อมูลที่จัดเก็บอยู่ในแผนภาพต้นไม้มีความสัมพันธ์ในลักษณะของพ่อกับลูกกับโหนดที่อยู่ในระดับที่ติดกัน
- โหนดพ่อแม่ (Parent Node) : โหนดที่อยู่ในระดับบนที่ติดกัน
- โหนดลูก (Child Node) : โหนดที่อยู่ในระดับล่างที่ติดกัน
- โหนดพี่น้อง (Siblings Node) : โหนดที่อยู่ในระดับเดียวกัน และมี parent เป็นโหนดเดียวกัน
- โหนดใบ (Leaf Node) : โหนดนอกสุดที่ไม่มีโหนดลูก



รูปที่ 2.8 แผนภาพต้นไม้

2.1.3.1 การท่องแผนภาพต้นไม้ (Tree Traversal)

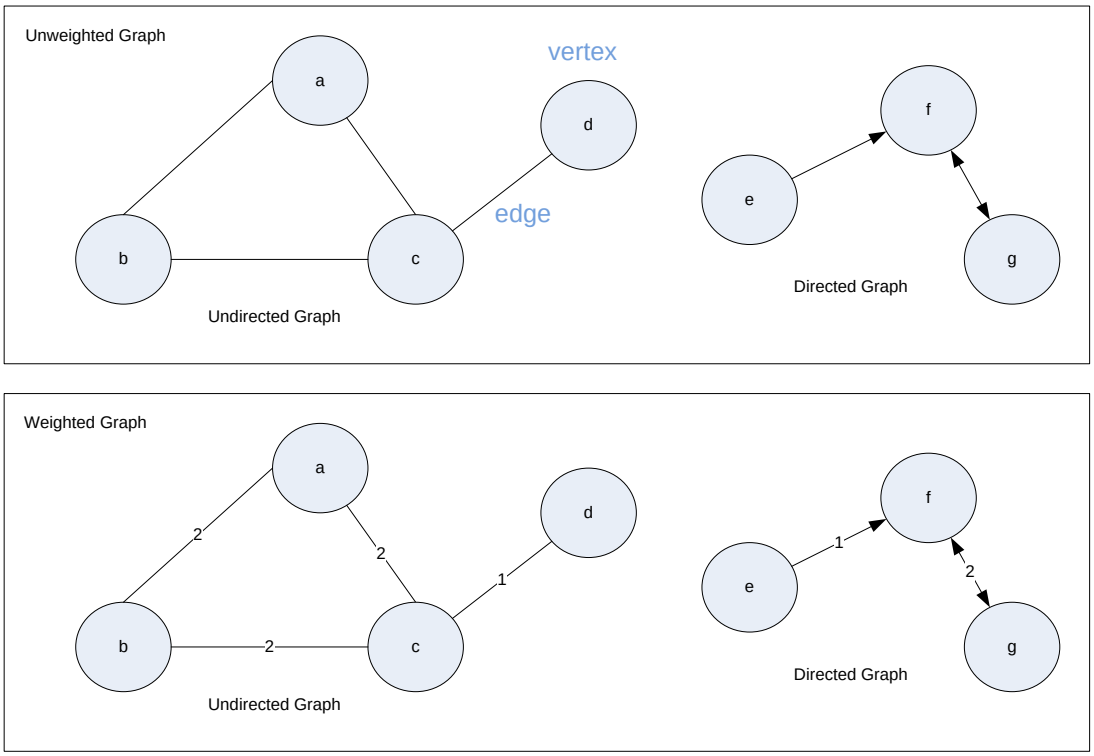
การท่องผ่านไปตามโหนดในแผนภาพต้นไม้ซึ่งจะสามารถแบ่งการลำดับการผ่าน ได้ดังนี้

- การท่องแบบก่อนลำดับ (Preorder Traversal) : การท่องโดยมีลำดับจากราก แล้วท่องไปยังโหนดลูกที่อยู่ทางซ้าย และขวา ตามลำดับ
- การท่องแบบตามลำดับ (Inorder Traversal) : การท่องโดยให้ความสำคัญกับโหนดลูกที่อยู่ทางซ้ายก่อน แล้วท่องไปที่ราก และลูกที่อยู่ทางขวา ตามลำดับ
- การท่องแบบหลังลำดับ (Postorder Traversal) : การท่องโดยให้ความสำคัญกับโหนดลูกที่อยู่ทางซ้ายก่อน แล้วท่องไปยังโหนดลูกที่อยู่ทางขวา และราก ตามลำดับ

2.1.4 แผนภาพกราฟ (Graph Diagram) [9]

แผนภาพกราฟ ดังแสดงในรูปที่ 2.9 ประกอบไปด้วยเซตของเวอร์เท็กซ์ (vertex) และเส้นแสดงความสัมพันธ์ระหว่างสองเวอร์เท็กซ์ ใดๆ เรียกว่าเอดจ์ (edge) โดยจะมีความแตกต่างจากแผนภาพต้นไม้ที่สามารถมีการวนซ้ำ หรือการมีความสัมพันธ์กลับเข้าหาตัวมันเองได้ เส้นแสดงความสัมพันธ์จะแบ่งออกตามลักษณะข้อมูลที่จัดเก็บได้เป็น

- ทิศทางความสัมพันธ์
  - กราฟแบบมีทิศทาง (Directed Graph)
  - กราฟแบบไม่มีทิศทาง (Undirected Graph)
- น้ำหนักความสัมพันธ์
  - *กราฟแบบมีน้ำหนัก (Weighted* Graph)
  - *กราฟแบบไม่มีน้ำหนัก (Unweighted* Graph)



รูปที่ 2.9 แผนภาพกราฟ

2.2 งานวิจัยที่เกี่ยวข้อง

มีความพยายามในการทำงานวิจัยเกี่ยวกับการค้นหาข้อบกพร่องในโมเดลการออกแบบและรหัสต้นฉบับของซอฟต์แวร์อยู่มากมาย โดยสามารถแบ่งออกได้หลักๆ เป็นการค้นหาข้อบกพร่องในรหัสต้นฉบับ และการทำนายข้อบกพร่องในโมเดลการออกแบบ

การค้นหาข้อบกพร่องในรหัสต้นฉบับนั้นนับได้ว่าเป็นวิธีการที่ทำได้ช้า เนื่องจากต้องรอให้มีการพัฒนาซอฟต์แวร์ก่อน จึงจะทำการค้นหาได้ และหากพบข้อบกพร่อง การทำการแก้ไขก็จะใช้ความพยายามมาก เนื่องจากในบางครั้งต้องย้อนกลับไปทำการวิเคราะห์และออกแบบใหม่อีกครั้ง และการแก้ไขก็อาจทำให้เกิดผลกระทบต่อส่วนอื่นได้

วิธีการในการค้นหาข้อบกพร่องในรหัสต้นฉบับนั้นเริ่มมีวัตถุประสงค์เพื่อการทำรีแฟคเทอริง (Refactoring) ซึ่งเป็นการปรับโครงสร้าง เพื่อปรับปรุงคุณภาพของรหัสต้นฉบับ ทำให้การบำรุงรักษาซอฟต์แวร์นั้นทำได้อย่างมีประสิทธิภาพมากขึ้น ในการทำรีแฟคทอริงนั้นเริ่มมาจากการให้ผู้เชี่ยวชาญตรวจสอบรหัสต้นฉบับเพื่อหารูปแบบของข้อบกพร่องตามที่ Fowler และ Beck [8] ได้นำเสนอไว้ด้วยกันทั้งหมด 22 รูปแบบ ดังที่ได้กล่าวถึงมาแล้ว โดยวิธีการนี้จะใช้เวลามาก และ

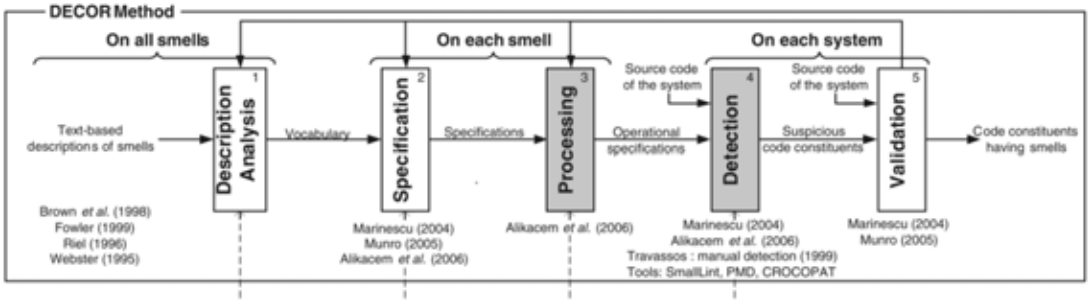
ให้ผลลัพธ์ไม่แน่นอน ขึ้นอยู่กับวิจารณญาณ และความเชี่ยวชาญของผู้ทำการตรวจสอบ โดยเฉพาะกับซอฟต์แวร์ในปัจจุบันที่มีแนวโน้มว่าจะมีขนาดใหญ่ขึ้นเรื่อยๆ และเพื่อให้วิธีการค้นหาข้อบกพร่องสามารถทำได้อย่างมีประสิทธิภาพมากขึ้น จึงมีผู้วิจัยได้ทำการศึกษาวิธีการในการพัฒนากระบวนการในการค้นหาให้เป็นมาตรฐานเพื่อสามารถนำไปพัฒนาเป็นเครื่องมือในการตรวจจับแบบอัตโนมัติ อันจะมีความรวดเร็ว และแม่นยำกว่า วิธีการที่นำมาใช้ประกอบไปด้วยวิธีการทางมาตรวัดซอฟต์แวร์ [1] [2] [3] [11] [12] [15] การค้นหาข้อบกพร่องในรหัสต้นฉบับด้วยโปรล็อก [4] และการค้นหาข้อบกพร่องด้วยวิธีการอุปมา [13]

การค้นหาข้อบกพร่องในโมเดลการออกแบบ มีขึ้นเพื่อให้ผู้มีหน้าที่ในการออกแบบซอฟต์แวร์สามารถทำการตรวจสอบข้อบกพร่องที่มีอยู่ได้ในขั้นตอนต้นๆ เนื่องจากไม่จำเป็นต้องรอจนมีการพัฒนาซอฟต์แวร์ งานวิจัยที่นำเสนอวิธีการในการค้นหาข้อบกพร่องในโมเดลการออกแบบ มีรายละเอียด ดังนี้

2.2.1 งานวิจัย “DECOR: A Method for the Specification and Detection of Code and Design Smells” [7]

งานวิจัยนี้ได้นำเสนอวิธีการสร้างตรรกะในการค้นหาข้อบกพร่องในการออกแบบแบบอัตโนมัติ งานวิจัยนี้แบ่งออกเป็น 2 ส่วนด้วยกัน ได้แก่ 1) การนำเสนอข้อกำหนดของข้อบกพร่องและกระบวนการในการตรวจจับข้อบกพร่อง เรียกว่า Detection and Correction (DECOR) และ 2) การนำกระบวนการ DECOR ที่ได้นำเสนอไปเบื้องต้นมาทำการปรับปรุง เพื่อให้กระบวนการมีความเป็นมาตรฐาน และสามารถนำไปใช้งานได้แบบอัตโนมัติมากขึ้น เรียกว่า Detection Export (DETEX)

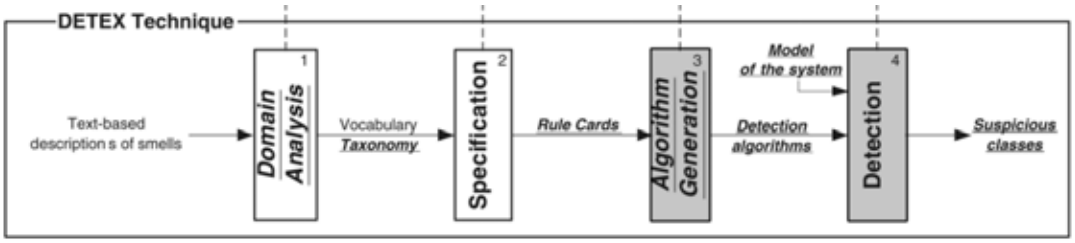
DECORE กระบวนการค้นหาข้อบกพร่อง DECORE ประกอบไปด้วย 5 ขั้นตอนด้วยกัน ดังแสดงในรูปที่ 2.10 ประกอบไปด้วย



รูปที่ 2.10 กระบวนการในการค้นหาข้อบกพร่องด้วยวิธีการ DECOR

- **ขั้นตอนที่ 1:** การวิเคราะห์นิยามของข้อบกพร่องจากงานวิจัยอื่นๆ ที่อยู่ในรูปของข้อความ เพื่อหาคุณลักษณะ หรือแนวคิดสำคัญของข้อบกพร่องนั้นๆ
- **ขั้นตอนที่ 2:** การนำคุณลักษณะและแนวคิดที่ได้จากขั้นตอนที่ 1 มาสร้างเป็นข้อกำหนดของข้อบกพร่องขึ้นมา
- **ขั้นตอนที่ 3:** การนำข้อกำหนดจากขั้นตอนที่ 2 มาสร้างอัลกอริทึมในการตรวจหาข้อบกพร่อง
- **ขั้นตอนที่ 4:** การนำข้อกำหนดมาใช้ในการค้นหาข้อบกพร่องภายในระบบ จากการค้นหาข้อบกพร่อง จะได้รายการของส่วนของซอฟต์แวร์ที่ต้องสงสัยว่าจะเป็นข้อบกพร่องออกมา
- **ขั้นตอนที่ 5:** ในขั้นตอนสุดท้าย จะเป็นการนำเอารายการข้อบกพร่องที่ได้จากการค้นหามาวิเคราะห์หาค่าความจริง

DETEX ส่วนที่สองของงานวิจัยได้นำกระบวนการค้นหาข้อบกพร่อง DÉCOR มาทำการปรับปรุง ในส่วนนี้ประกอบไปด้วย 4 ขั้นตอนด้วยกัน ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 กระบวนการในการค้นหาข้อบกพร่องด้วยวิธีการ DETEX

- **ขั้นตอนที่ 1:** การวิเคราะห์นิยามของข้อบกพร่องจากงานวิจัยอื่นๆ ที่อยู่ในรูปของข้อความ โดยมีการคำนึงถึงโดเมนด้วยเพื่อหาคุณลักษณะ แนวคิดสำคัญ รวมไปถึงหมวดหมู่ของข้อบกพร่องนั้นๆ

- **ขั้นตอนที่ 2:** การนำคุณลักษณะ แนวคิด และหมวดหมู่ที่ได้จากขั้นตอนที่ 1 มาสร้างเป็นข้อกำหนดของข้อบกพร่องขึ้นมา โดยจะอยู่ในรูปของ Domain Specific Language (DSL) เรียกว่าเป็น rule card ของข้อบกพร่อง
- **ขั้นตอนที่ 3:** การนำข้อกำหนดจากขั้นตอนที่ 2 มาสร้างอัลกอริทึมในการตรวจหาข้อบกพร่อง โดยงานวิจัยได้นำเสนอ ภาษาที่ใช้ในการอธิบายข้อกำหนดของข้อบกพร่อง เรียกว่า Smell Definition Language (SMELLDL) โดย SMELLDL จะถูกสร้างขึ้นมาจาก DSL หรือ rule card ของแต่ละข้อบกพร่องที่ได้มาจากขั้นตอนที่ 2 และเพื่อให้สามารถสร้างอัลกอริทึมในการตรวจจับได้แบบอัตโนมัติจากข้อกำหนด งานวิจัยได้นำเสนอเฟรมเวิร์คที่ชื่อว่า SMELLFW ขึ้นมาเพื่อใช้ในการค้นหาข้อบกพร่อง
- **ขั้นตอนที่ 4:** การนำข้อกำหนดมาใช้ในการค้นหาข้อบกพร่องของโมเดลการออกแบบ โดยโมเดลการออกแบบที่ใช้ในงานวิจัย จะได้มาด้วยวิธีการ Reverse Engineering จากรหัสต้นฉบับที่ได้ถูกพัฒนาแล้ว

2.2.2 งานวิจัย “Comparing Design and Code Metrics for Software Quality Prediction” [14]

วัตถุประสงค์ของงานวิจัยนี้ได้แก่การเปรียบเทียบประสิทธิภาพของโมเดลในการทำนายข้อบกพร่องด้วยวิธีการเรียนรู้ของเครื่อง โดยทำการเปรียบเทียบโมเดลที่ใช้มาตรวัดในระดับการออกแบบ โมเดลที่ใช้มาตรวัดในระดับการพัฒนา และโมเดลที่ใช้ทั้งสองมาตรวัด งานวิจัยนี้ได้ทำการวิเคราะห์ชุดข้อมูลจาก NASA Metrics Data Program ทั้ง 13 ชุดข้อมูล ซึ่งประกอบไปด้วยข้อมูลทั้งในระดับการออกแบบ และพัฒนา

ผลการทำวิจัยสรุปได้ว่าการทำนายด้วยโมเดลที่ใช้มาตรวัดทั้งสองระดับประกอบกันให้ผลที่ดีที่สุด และรองลงมาคือโมเดลที่ใช้มาตรวัดที่ระดับรหัสต้นฉบับ แต่ถึงแม้ว่าการทดสอบด้วยมาตรวัดแต่ละระดับจะให้ผลออกมาไม่เท่ากัน มาตรวัดแต่ละแบบก็สามารถนำมาใช้ได้ในขั้นตอนที่แตกต่างกันออกไป

2.2.3 งานวิจัย “Bad-smell Prediction from Software Design Model Using Machine Learning Techniques” [5]

งานวิจัยนี้ได้นำวิธีการค้นหาข้อบกพร่องในโมเดลการออกแบบด้วยการเรียนรู้ของเครื่อง 7 วิธีมาทำการเปรียบเทียบประสิทธิภาพ ภายใต้แนวความคิดที่ว่า การค้นหาข้อบกพร่องในโมเดลการออกแบบ สามารถทำได้ในขั้นตอนแรกๆของการพัฒนาซอฟต์แวร์ ซึ่งจะเป็นประโยชน์กว่าการค้นหาข้อบกพร่องในรหัสต้นฉบับ

งานวิจัยครอบคลุมการค้นหาข้อบกพร่อง 7 ประเภท ตามข้อกำหนด ประกอบไปด้วย Lazy Class, Feature Envy, Middle Man, Message Chains, Long Method, Long Parameter Lists, และ Switch Statement โดยใช้วิธีการเรียนรู้ของเครื่อง 7 วิธีด้วยกัน ได้แก่ Random Forest (RF), Naïve Bayes (NB), Logistic regression(LGI), IB1, IBk, VFI, และ J48 โดยชุดของข้อมูลที่ใช้ในการพิจารณาค้นหาข้อบกพร่องได้มาจากแผนภาพคลาส ซึ่งถึงได้ว่าเป็นหัวใจสำคัญของการออกแบบแบบอ็อบเจ็ค ประกอบไปด้วยข้อมูลการออกแบบของแต่ละคลาส แอททริบิวต์ และความสัมพันธ์ระหว่างคลาส ข้อมูลแผนภาพคลาสจะถูกนำมาสกัดเพื่อให้ได้ค่าของมาตรวัดแต่ละตัวด้วย MagicDraw 9.5 SP1.1 งานวิจัยสามารถแบ่งขั้นตอนออกได้ ดังนี้

ขั้นตอนที่ 1: การรวบรวมชุดข้อมูลโมเดลการออกแบบที่ประกอบไปด้วยข้อบกพร่องทั้ง 7 ประเภทตามที่ได้กล่าวไปแล้วข้างต้น

ขั้นตอนที่ 2: การใช้เครื่องมือ MagicDraw 9.5 SP1.1 มาสกัดเพื่อหาค่ามาตรวัดที่จะนำมาใช้ในการค้นหาข้อบกพร่องด้วยวิธีการเรียนรู้ของเครื่อง

ขั้นตอนที่ 3: การนำค่ามาตรวัดที่ได้จากขั้นตอนที่ 2 มาทำการค้นหาข้อบกพร่องด้วยวิธีการเรียนรู้ของเครื่อง

ขั้นตอนที่ 4: การนำผลการค้นหาข้อบกพร่องมาเปรียบเทียบกับค่าจริง เพื่อหาประสิทธิภาพในการค้นหาข้อบกพร่องแต่ละประเภทและนำผลการค้นหาด้วยวิธีการเรียนรู้ของเครื่องแต่ละวิธีมาเปรียบเทียบประสิทธิภาพ

งานวิจัยนี้สรุปผลว่าไม่มีวิธีการใดที่เหมาะสมในการค้นหาข้อบกพร่องทั้ง 7 ชนิด แต่ละวิธีการต่างก็ให้ผลการค้นหาข้อบกพร่องแต่ละประเภทได้ดีแตกต่างกันออกไป ตารางที่ 2.3 แสดงผลการคัดเลือกวิธีการที่ดีที่สุดในการค้นหาข้อบกพร่องแต่ละประเภท แบ่งตามระเบียบวิธีการวิจัย

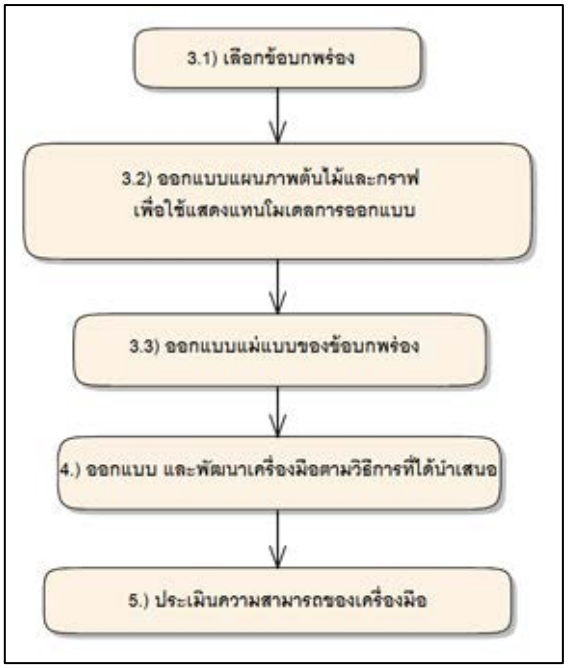
ตารางที่ 2.2 ตารางแสดงผลการคัดเลือกวิธีที่ดีที่สุดในการค้นหาข้อบกพร่องแต่ละประเภท แบ่งตามระเบียบวิธีการวิจัย

	Prediction accuracy	**Specificity**	**Sensitivity**	**PPV**
Feature Envy	RF	VFI	RF	RF
Long Method	IBk	NB	IB1 and IBk	IBk
Long Parameter Lists	J48	NB	LGI and RF	RF
Lazy Class	J48	VFI	J48	J48
Message Chains	LG1, IB1, IBk and RF	LGI, IB1 and IBk	LGI, IB1, IBk and RF	LGI, IB1 and IBk
Middle Man	IB1 and IBk	LGI, IB1 and IBk	J48	LGI, IB1 and IBk
Switch Statement	IBk	NB and VFI	J48	J48

บทที่ 3
วิธีค้นหาข้อบกพร่องในโมเดลการออกแบบซอฟต์แวร์

ในบทที่ 3 จะกล่าวถึงแนวคิดในการออกแบบวิธีการในการค้นหาข้อบกพร่องในโมเดลการออกแบบซอฟต์แวร์ และรายละเอียดของวิธีการ โดยได้แบ่งขั้นตอนออกเป็น 5 ขั้นตอนด้วยกัน ได้แก่ การเลือกข้อบกพร่อง การออกแบบแผนภาพแสดงโมเดลการออกแบบ การออกแบบแม่แบบของข้อบกพร่องเพื่อใช้ในกระบวนการค้นหา การออกแบบและพัฒนาเครื่องมือตามวิธีการที่ได้นำเสนอ และประเมินความสามารถของเครื่องมือและวิธีการ ดังแสดงในแผนภาพกิจกรรมตามรูปที่ 3.1

เริ่มจากการออกแบบแผนภาพแสดงโมเดลการออกแบบ งานวิจัยนี้ได้นำแผนภาพกราฟและแผนภาพต้นไม้มาใช้ แผนภาพทั้งสองนั้นมีโครงสร้างการเก็บข้อมูลคล้ายคลึงกับการเก็บข้อมูลโมเดลการออกแบบ แผนภาพต้นไม้จะสามารถนำมาใช้ในการจัดเก็บโครงสร้างการสืบทอดและโครงสร้างของคลาส ในขณะที่แผนภาพกราฟจะถูกนำมาใช้ในการจัดเก็บความสัมพันธ์ระหว่างคลาส แผนภาพแสดงโมเดลการออกแบบประกอบไปด้วย แผนภาพต้นไม้การสืบทอด แผนภาพต้นไม้คุณลักษณะ และแผนภาพกราฟตรรกะ



รูปที่ 3.1 แผนภาพกิจกรรม แสดงภาพรวมของการทำงานวิจัย

ในการค้นหาข้อบกพร่องจะมีการสร้างแม่แบบของข้อบกพร่อง เพื่อนำมาพิจารณาเปรียบเทียบกับแผนภาพแสดงโมเดลการออกแบบ แม่แบบของข้อบกพร่องสร้างขึ้นมาจากการนำข้อกำหนดข้อบกพร่องมาพิจารณา เพื่อหาคุณลักษณะที่จะใช้ในการตัดสินใจ และนำมาสร้างเป็นแม่แบบสำหรับนำมาตรวจสอบกับแผนภาพแสดงโมเดลการออกแบบ แม่แบบจะใช้ข้อกำหนดเดียวกับแผนภาพแสดงโมเดลการออกแบบ

3.1 เลือกข้อบกพร่อง

การค้นหาข้อบกพร่องในโมเดลการออกแบบจะมีข้อจำกัดทางด้านข้อมูลที่มีปรากฏอยู่ในขั้นตอนการออกแบบ ประกอบกับงานวิจัยนี้มีข้อจำกัดทางด้านการค้นหาด้วยการใช้แม่แบบของข้อบกพร่องด้วย จากข้อจำกัดของงานวิจัยทำให้ไม่สามารถสร้างแม่แบบสำหรับข้อบกพร่องบางประเภทได้ จากข้อกำหนดของแผนภาพที่นำเสนอ ข้อจำกัดดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 ตารางแสดงข้อจำกัดในการเลือกข้อบกพร่อง

ข้อจำกัดทางด้านข้อมูลการออกแบบ	
ข้อบกพร่อง	**ข้อจำกัดในการค้นหา**
Long Method	เมท็อดที่มีขนาดใหญ่ หรือมีหน้าที่ความรับผิดชอบมาก ซึ่งการพัฒนาภายในเมท็อดไม่สามารถทราบได้ในขั้นตอนการออกแบบ
Data Clumps	การใช้ชุดของข้อมูลเพื่อเก็บค่าในหลายๆส่วนของซอฟต์แวร์ ซึ่งชุดข้อมูลดังกล่าวสามารถรวมกลุ่มกันสร้างเป็นคลาสขึ้นมาแทนได้ ซึ่งข้อมูลในส่วนนี้ไม่ทราบได้ในขั้นตอนการออกแบบ
Primitive Obsession	การใช้ตัวแปร Primitive ในรหัสต้นฉบับ ซึ่งไม่สามารถทราบได้ในขั้นตอนการออกแบบ
Switch Statement	การใช้ Switch Statement ในรหัสต้นฉบับ ซึ่งไม่สามารถทราบได้ในขั้นตอนการออกแบบ
Speculative Generality	การเพิ่มรหัสต้นฉบับเข้าไปในโปรแกรม เพื่อให้มีการทำงานเฉพาะอย่างโดยไม่มีการวิเคราะห์ หรือออกแบบก่อน ซึ่งหากยังไม่มีการพัฒนาจะไม่สามารถทราบข้อมูลในส่วนนี้ได้
Temporary Field	การใช้ตัวแปร เพื่อเก็บค่าชั่วคราวในรหัสต้นฉบับ ซึ่งไม่สามารถทราบได้ในขั้นตอนการออกแบบ

ตารางที่ 3.1 ตารางแสดงข้อจำกัดในการเลือกข้อบกพร่อง (ต่อ)

ข้อบกพร่อง	ข้อจำกัดในการสร้างแม่แบบ
Incomplete Library Class	การเรียกไปยังไลบรารี่ภายนอก ซึ่งไม่สามารถทราบได้ในขั้นตอนการออกแบบ
Comment	การใส่คอมเมนท์ไว้ในรหัสต้นฉบับในส่วนที่เป็นข้อบกพร่อง ซึ่งไม่สามารถทราบได้ในขั้นตอนการออกแบบ
ข้อจำกัดทางในการสร้างแม่แบบ	
ข้อบกพร่อง	**ข้อจำกัดในการสร้างแม่แบบ**
Duplicate Code	การเกิดการซ้ำกันในหลายๆส่วนของซอฟต์แวร์ เป็นการเปรียบเทียบกันระหว่างสองโครงสร้างใดๆ ทำให้ไม่สามารถนำมาสร้างเป็นแม่แบบได้
Divergent Change	คลาสที่เมื่อเกิดการเปลี่ยนแปลงขึ้น จะส่งผลกระทบกับหลายๆส่วนของโปรแกรม วิเคราะห์ได้จากการขึ้นต่อกันของคลาส ซึ่งไม่สามารถนำมาสร้างเป็นแม่แบบจากนิยามของแผนภาพที่นำเสนอในงานวิจัยนี้ได้
Shotgun Surgery	คลาสที่ได้รับผลกระทบจากการเปลี่ยนแปลงส่วนอื่นๆของโปรแกรมซึ่งจะดูจากการขึ้นต่อกันของคลาส ซึ่งไม่สามารถนำมาสร้างเป็นแม่แบบจากนิยามของแผนภาพที่นำเสนอในงานวิจัยนี้ได้
Alternative Class With Different Interface	พิจารณาจากความคล้ายคลึงหรือเหมือนกันทางด้านการทำงานของคลาส เป็นการเปรียบเทียบกันระหว่างสองโครงสร้างใดๆ ทำให้ไม่สามารถนำมาสร้างเป็นแม่แบบได้

จากข้อจำกัดข้างต้น ทำให้งานวิจัยนี้ครอบคลุมการค้นหาข้อบกพร่อง 10 ประเภท จากทั้งหมด 22 ประเภท ตารางที่ 3.2 แสดงคุณลักษณะที่จะนำมาใช้ในการพิจารณาข้อบกพร่องในงานวิจัยนี้ทั้ง 10 ประเภท โดยรายละเอียดการวิเคราะห์จะอธิบายเพิ่มเติมในหัวข้อที่ 3.2.2 การออกแบบแม่แบบของข้อบกพร่องต่อไป

ตารางที่ 3.2 ตารางสรุปแนวทางเบื้องต้นในการค้นหาข้อบกพร่อง 10 ประเภทในงานวิจัย

ข้อบกพร่อง	แนวทางในการค้นหา
Refused Bequest	พิจารณาจากการสืบทอดคุณลักษณะของคลาส และความสัมพันธ์การเรียกใช้งานเมท็อด

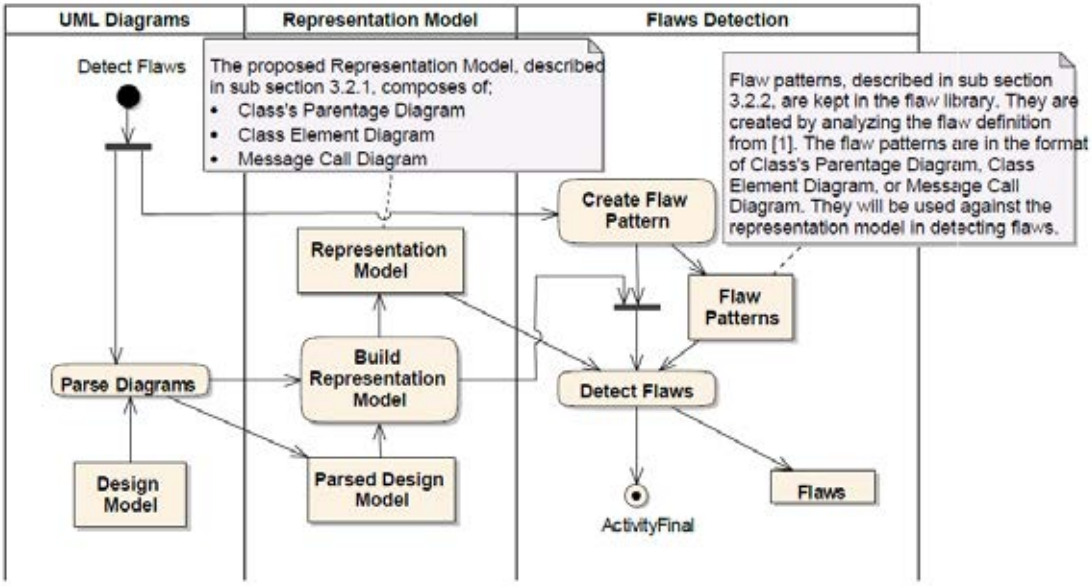
ตารางที่ 3.2 ตารางสรุปแนวทางเบื้องต้นในการค้นหาข้อบกพร่อง 10 ประเภทในงานวิจัย (ต่อ)

ข้อบกพร่อง	แนวทางในการค้นหา
Feature Envy	พิจารณาจากจำนวนครั้งที่เมท็อดถูกเรียกจากภายนอก เทียบกับจำนวนครั้งที่มีการเรียกภายในคลาส
Middle Man	พิจารณาจากลำดับการเรียกเมท็อด โดยจะมีลักษณะการเรียกเมท็อดผ่านตัวกลาง
Message Chain	พิจารณาจากลำดับการเรียกเมท็อดโดยจะมีลักษณะการเรียกต่อกันเป็นลูกโซ่
Inappropriate Intimacy	พิจารณาจากการออกแบบคุณลักษณะของคลาส ประกอบกับความสัมพันธ์การเรียกเมท็อดจากภายนอก
Lazy Class	ในงานวิจัยนี้จะพิจารณาว่าคลาสมีการทำงานที่ไม่เพียงพอเมื่อไม่พบการทำงานหลักในคลาส
Data Class	พิจารณาจากการออกแบบคุณลักษณะภายในคลาส โดยจะถือว่าเป็นข้อบกพร่องเมื่อพบเพียงแอคเซสเมท็อด (Access Method) เท่านั้น
Long Parameter Lists	พิจารณาจากการออกแบบคลาส ในส่วนของจำนวนพารามิเตอร์ที่ผ่านเข้าสู่เมท็อด
Large Class	พิจารณาจากการออกแบบคลาสในส่วนของจำนวนเมท็อดที่มีภายในคลาส
Parallel Inheritance Hierarchy	พิจารณาจากโครงสร้างการสืบทอดคลาส และความสัมพันธ์ของโหนดในสองโครงสร้างการสืบทอดใดๆ

3.2 ออกแบบวิธีการในการค้นหาข้อบกพร่อง

งานวิจัยนี้นำเสนอวิธีการในการค้นหาข้อบกพร่องด้วยการใช้แม่แบบของข้อบกพร่องนำมาเทียบกับแผนภาพแสดงโมเดลการออกแบบ ข้อกำหนดของแผนภาพแสดงโมเดลการออกแบบ และการสร้างข้อบกพร่องจะได้อธิบายในหัวข้อต่อไป รูปที่ 3.2 แสดงภาพรวมของวิธีการ โดยจะเริ่มจากข้อมูลนำเข้า ได้แก่ แผนภาพคลาส และแผนภาพซีเควนซ์ แผนภาพจะถูกนำมาอ่านเพื่อเก็บข้อมูลการออกแบบ และสร้างออกมาตามข้อกำหนดเป็นแผนภาพแสดงโมเดลการออกแบบ นิยามของข้อบกพร่องแต่ละประเภทจะถูกนำมาวิเคราะห์ เพื่อหาคุณลักษณะ และสร้างออกมาเป็นแม่แบบของข้อบกพร่อง

ในการค้นหาข้อบกพร่อง แม่แบบจะถูกนำมาตรวจสอบกับแผนภาพแสดงโมเดลการออกแบบ เพื่อตรวจหาส่วนของการออกแบบที่เป็นข้อบกพร่อง



รูปที่ 3.2 แผนภาพกิจกรรมแสดงภาพรวมการทางานในการค้นหาข้อบกพร่องในโมเดลการออกแบบ

3.2.1 ออกแบบแผนภาพที่นำมาใช้ในการแสดงโมเดลการออกแบบซอฟต์แวร์

หัวข้อนี้อธิบายถึงการออกแบบข้อกำหนดของแผนภาพที่นำมาใช้ในการแสดงโมเดลการออกแบบ และแม่แบบในการค้นหาข้อบกพร่อง ในการออกแบบแผนภาพแสดงโมเดลการออกแบบงานวิจัยนี้ได้ทำการวิเคราะห์นิยามของข้อบกพร่อง และสรุปคุณลักษณะที่จะนำมาใช้ในกระบวนการค้นหาได้ 3 กลุ่ม ดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 ตารางแสดงประเภทของคุณลักษณะที่นำมาใช้ในการค้นหาข้อบกพร่องแต่ละประเภท

ประเภทของคุณลักษณะ	ข้อบกพร่องที่ใช้ค้นหา
การสืบทอดคลาส	Refused Bequest Parallel Inheritance Hierarchy
การออกแบบโครงสร้างภายในของคลาส	Inappropriate Intimacy Lazy Class Data Class Long Parameter List Large Class

ตารางที่ 3.3 ตารางแสดงประเภทของคุณลักษณะที่นำมาใช้ในการค้นหาข้อบกพร่องแต่ละประเภท (ต่อ)

ประเภทของคุณลักษณะ	ข้อบกพร่องที่ใช้ค้นหา
ความสัมพันธ์ระหว่างคลาส	Refused Bequest Feature Envy Middle Man Message Chain Inappropriate Intimacy

ข้อมูลที่ต้องการในการนำมาใช้พิจารณาข้อบกพร่องทั้ง 3 กลุ่มนั้น จะมีโครงสร้างการเก็บข้อมูลคล้ายคลึงกับโครงสร้างของแผนภาพกราฟ และแผนภาพต้นไม้ กล่าวคือ แผนภาพต้นไม้สามารถนำมาใช้ในการจัดเก็บโครงสร้างแบบเป็นลำดับชั้นของข้อมูลการสืบทอด และข้อมูลโครงสร้างของคลาสได้ และความสัมพันธ์ระหว่างเวอร์เท็กซ์ในแผนภาพกราฟ ก็สามารถนำมาใช้ในการจัดเก็บความสัมพันธ์ระหว่างคลาสได้ เมื่อกำหนดให้แต่ละเวอร์เท็กซ์แสดงแต่ละคลาสใดๆ ในการออกแบบ

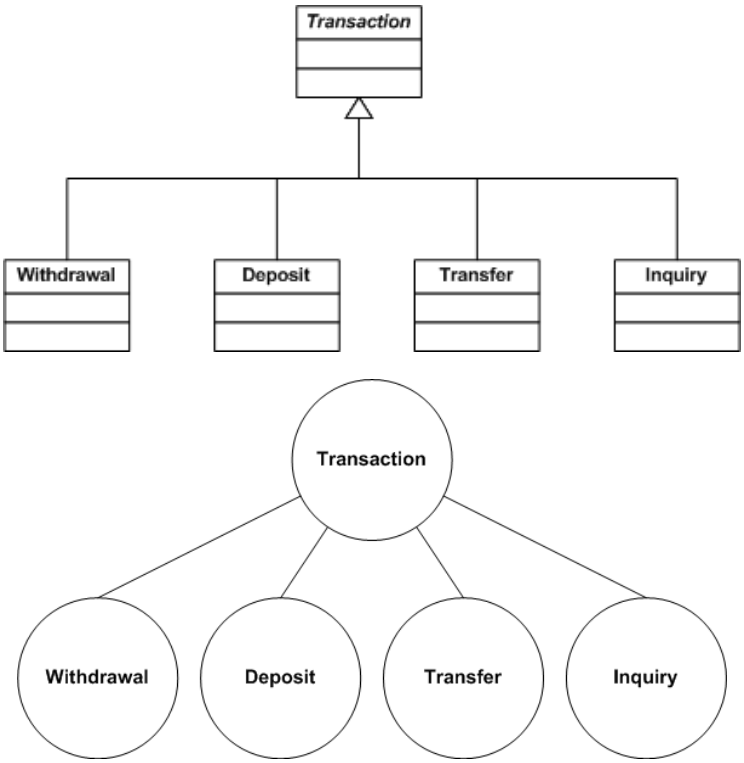
งานวิจัยนี้จึงได้นำแผนภาพกราฟ และแผนภาพต้นไม้มาใช้ในการออกแบบแผนภาพแสดงโมเดลการออกแบบ โดยได้นำเสนอ 3 แผนภาพ ด้วยกัน ประกอบไปด้วย 1) แผนภาพต้นไม้การสืบทอด เก็บข้อมูลการสืบทอดของคลาส 2) แผนภาพต้นไม้คุณลักษณะ เก็บข้อมูลโครงสร้างของคลาส และ 3) แผนภาพกราฟตรรกะ เก็บข้อมูลความสัมพันธ์ระหว่างคลาส นิยามและ การออกแบบแผนภาพทั้ง 3 มีรายละเอียด ดังนี้

แผนภาพต้นไม้การสืบทอด (Class Inheritance Diagram)

ข้อกำหนดที่ 1*: แผนภาพต้นไม้การสืบทอด คือแผนภาพต้นไม้ที่ประกอบไปด้วยเซตของ Class Node (CN) ที่แสดงเป็นลำดับขั้น โดย CN เป็นคลาสที่มีปรากฏอยู่ในแผนภาพคลาส*

แผนภาพต้นไม้การสืบทอดจัดเก็บข้อมูลการสืบทอดของคลาสแบบเป็นลำดับชั้น ข้อมูลโมเดลการออกแบบดังกล่าวได้มาจากการอ่านแผนภาพคลาส รูปที่ 3.3 แสดงแผนภาพต้นไม้การสืบทอด ที่ได้แปลงมาจากการสืบทอดคลาส Transaction คลาสจากแผนภาพคลาส จะแสดงด้วยหนึ่งโหนดในแผนภาพต้นไม้ รากของต้นไม้แสดงต้นตระกูลของการสืบทอด ในขณะที่คลาสที่สืบทอด

จะถูกนำมาใส่เป็นโหนดลูกต่อกันไปตามลำดับ แผนภาพต้นไม้การสืบทอดจะสร้างขึ้นสำหรับทุกๆ การสืบทอดที่มีปรากฏอยู่



รูปที่ 3.3 แผนภาพต้นไม้การสืบทอด

แผนภาพต้นไม้คุณลักษณะ (Class Characteristic Diagram)

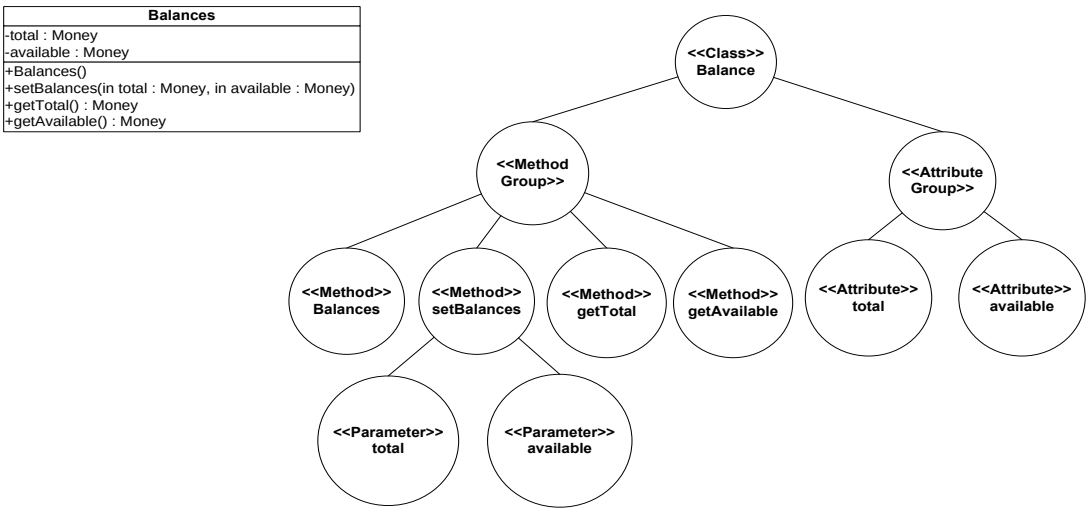
ข้อกำหนดที่ 2*: แผนภาพต้นไม้คุณลักษณะ คือแผนภาพต้นไม้ที่ประกอบไปด้วยเซตของโหนดแสดงคุณลักษณะ หรือ Properties Node (PN) แสดงเป็นลำดับขั้น เมื่อ*

- PN คือ เซตของโหนดที่แสดงคุณลักษณะของคลาส และ
  $PN = CN \cup MgN \cup MN \cup PrN \cup AgN \cup AN$
- CN คือ โหนดที่แสดงคลาส
- MgN และ AgN คือโหนดที่แสดงกลุ่มของเมท็อด และกลุ่มของแอททริบิวต์ตามลำดับ
- MN คือ เซตของโหนดที่แสดงแต่ละเมท็อด อยู่ภายใต้กลุ่มของเมท็อด
- PrN คือ เซตของโหนดที่แสดงแต่ละพารามิเตอร์ อยู่ภายใต้โหนดของเมท็อด
- AN คือ เซตของโหนดที่แสดงแต่ละแอททริบิวต์ อยู่ภายใต้กลุ่มของแอททริบิวต์

แผนภาพต้นไม้คุณลักษณะ แสดงคุณลักษณะของแต่ละคลาส แต่ละโหนดในแผนภาพแสดงถึงคุณลักษณะที่แตกต่างกันออกไป เริ่มจากโหนดรากที่แสดงถึงคลาส โครงสร้างของแผนภาพตามรูปที่ 3.4 คำอธิบายประเภทของโหนดแสดงอยู่ในตารางที่ 3.4

ตารางที่ 3.4 ตารางแสดงคำอธิบายโหนดในแผนภาพต้นไม้คุณลักษณะ

Tree Node	**Definition**
Class level (Depth = 0)	
Class Node (CN)	โหนดรูทที่แสดงคลาส
Class member's category level (Depth = 1)	
Method Group (MgN)	โหนดแสดงกลุ่มของเมท็อด
Attribute Group (AgN)	โหนดแสดงกลุ่มของแอททริบิวต์
Class member level (Depth = 2)	
Method Node (MN)	โหนดแสดงเมท็อด แต่ละเมท็อดจะแสดงด้วยหนึ่งโหนด
Attribute Node (AN)	โหนดแสดงแอททริบิวต์ของคลาส แต่ละแอททริบิวต์จะแสดงด้วยหนึ่งโหนด
Method's parameter level (Depth =3)	
Parameter Node (PrN)	โหนดแสดงพารามิเตอร์ที่ส่งเข้าสู่เมท็อด แต่ละพารามิเตอร์จะแสดงด้วยหนึ่งโหนด



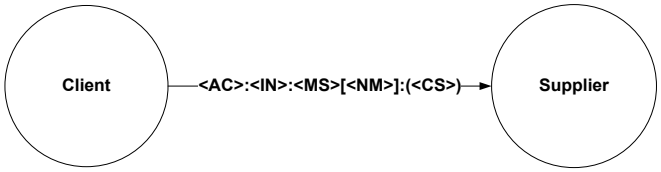
รูปที่ 3.4 แผนภาพต้นไม้คุณลักษณะ

แผนภาพกราฟตรรกะ (Logic Graph Diagram)

ข้อกำหนดที่ 3*: แผนภาพกราฟตรรกะ คือแผนภาพกราฟผสม 3 สิ่งอันดับ (3-tuple) <CN, M, F> แบบมีทิศทาง เมื่อ*

- CN คือ เวอร์เท็กซ์ที่แสดงแต่ละคลาส $\{CN_1, CN_2, \ldots, CN_n\}$, n คือจำนวนคลาสที่มีปรากฏอยู่ในโมเดลการออกแบบ
- *M แสดงการเรียกระหว่างคลาส* $\{M_1, M_2, \ldots, M_n\}$, *n คือจำนวนของการเรียกแสดงด้วยเส้นความสัมพันธ์ระหว่างเวอร์เท็กซ์*
- *F เก็บข้อมูลการเรียกระหว่างคลาส ประกอบไปด้วย 5 คุณลักษณะ ได้แก่ ระดับการเข้าถึงเมท็อด การสืบทอด Signature จำนวนครั้งที่ถูกเรียก และลำดับการเรียก*

แผนภาพกราฟตรรกะใช้คุณสมบัติของความสัมพันธ์ระหว่างเวอร์เท็กซ์ในการแสดงความสัมพันธ์ระหว่างคลาส การทำงานของโปรแกรมจะถูกเก็บอยู่ในลักษณะของการเรียกใช้งานเมท็อดในคลาสนั้นๆ มีทิศทางจากเวอร์เท็กซ์ของคลาสที่เรียกใช้ ไปยังเวอร์เท็กซ์ของคลาสที่เป็นเจ้าของเมท็อด ป้ายของเส้นแสดงความสัมพันธ์แสดงถึงเมท็อด ระดับการเข้าถึง การสืบทอด จำนวนครั้งที่ถูกเรียก และลำดับของการเรียกตามแผนภาพซีเควนซ์ มีรูปแบบดังแสดงในรูปที่ 3.5 ตารางที่ 3.5 แสดงคำอธิบายแผ่นป้ายแสดงความสัมพันธ์ในแผนภาพกราฟตรรกะ



รูปที่ 3.5 แผนภาพกราฟตรรกะ

ตารางที่ 3.5 ตารางแสดงคำอธิบายแผ่นป้ายแสดงความสัมพันธ์ในแผนภาพกราฟตรรกะ

ส่วนประกอบ	คำอธิบาย	ค่าที่เป็นไปได้
Accessibility (AC)	ระดับการเข้าถึงของเมท็อด	**Private**: ในกรณีที่เมท็อดมีค่าการเข้าถึงเป็น Private **Protected**: ในกรณีที่เมท็อดมีค่าการเข้าถึงเป็น Protected **Public**: ในกรณีที่เมท็อดมีค่าการเข้าถึงเป็น Public **?**: ในกรณีที่ไม่พบว่ามีการระบุค่าการเข้าถึงไว้ในการออกแบบ
Inheritance (IN)	การสืบทอด	**Inherited**: ในกรณีที่เมท็อดได้รับสืบทอดมา **-**: ในกรณีที่เมท็อดที่ไม่ได้รับสืบทอดมา
Method Signature (MS)	ชื่อเมท็อด และพารามิเตอร์	ชื่อเมท็อด และพารามิเตอร์
Number of Message called (NM)	จำนวนครั้งที่ถูกเรียก	จำนวนครั้งที่ถูกเรียก โดยจะนับรวมทั้งโมเดลการออกแบบ
Called Sequence (CS)	ลำดับการเรียกเมท็อด	ลำดับการเรียกเมท็อด ตามแผนภาพซีเควนซ์

3.2.2 ออกแบบแม่แบบของข้อบกพร่อง

ในการออกแบบแม่แบบของข้อบกพร่อง ข้อกำหนดข้อบกพร่องแต่ละประเภทจะถูกนำมาวิเคราะห์ เพื่อหาคุณลักษณะที่จะนามาใช้ในการตัดสินใจ คุณลักษณะที่ได้มานั้นจะถูกนำไปสร้าง

เป็นแม่แบบ และนำมาใช้เปรียบเทียบกับโมเดลการออกแบบที่ถูกแปลงเป็นแผนภาพกราฟและแผนภาพต้นไม้ด้วยข้อกำหนดจากหัวข้อที่ 3 โดยจะมีการใช้สัญลักษณ์เพิ่มเติม ดังนี้

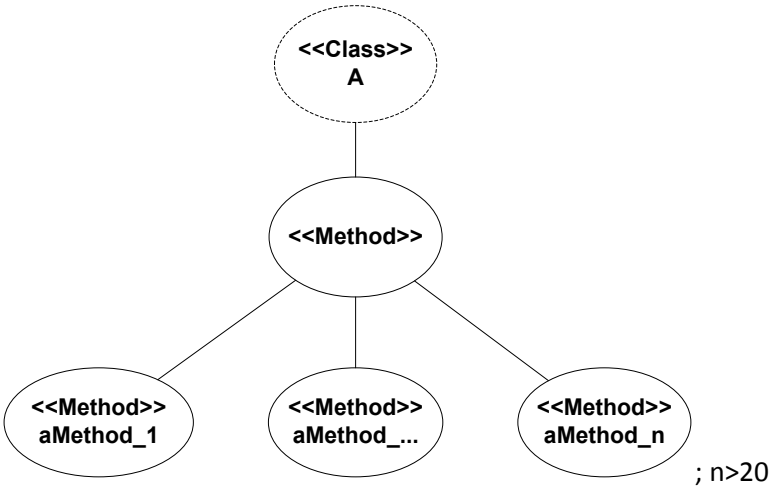
- *: แสดงถึงค่าใดๆ ที่สามารถเป็นไปได้สำหรับคุณลักษณะนั้นๆ ในการค้นหาจะไม่สนใจค่าของคุณลักษณะที่แทนค่าด้วย *
- **#CEATE#:** แสดงถึงการสร้างวัตถุในแผนภาพซีเควนซ์
- **#RETURN#**: แสดงถึงการส่งค่ากลับในแผนภาพซีเควนซ์

Large Class

นิยาม: คลาสที่มีเมท็อด และคุณลักษณะ เป็นจำนวนมาก [8]

แนวทางในการพิจารณา: พิจารณาจากจำนวนเมท็อด และคุณลักษณะภายในคลาส โดยจะเปรียบเทียบจำนวนกับค่าที่กำหนดไว้ โดยค่าที่อ้างอิงจากงานวิจัยที่เกี่ยวข้องคือ จำนวนเมท็อดมากกว่า 20 ในคลาสทั่วไป และมากกว่า 40 ในคลาสที่เกี่ยวกับส่วนติดต่อผู้ใช้งาน จำนวนอินแสตนซ์มากกว่า 3 ในคลาสทั่วไป และมากกว่า 9 ในคลาสที่เกี่ยวกับส่วนติดต่อผู้ใช้งาน [3]

แม่แบบ: แม่แบบสำหรับค้นหาข้อบกพร่องประเภท Large Class ที่มีจำนวนเมท็อดมากกว่า 20 ดังแสดงตามรูปที่ 3.6



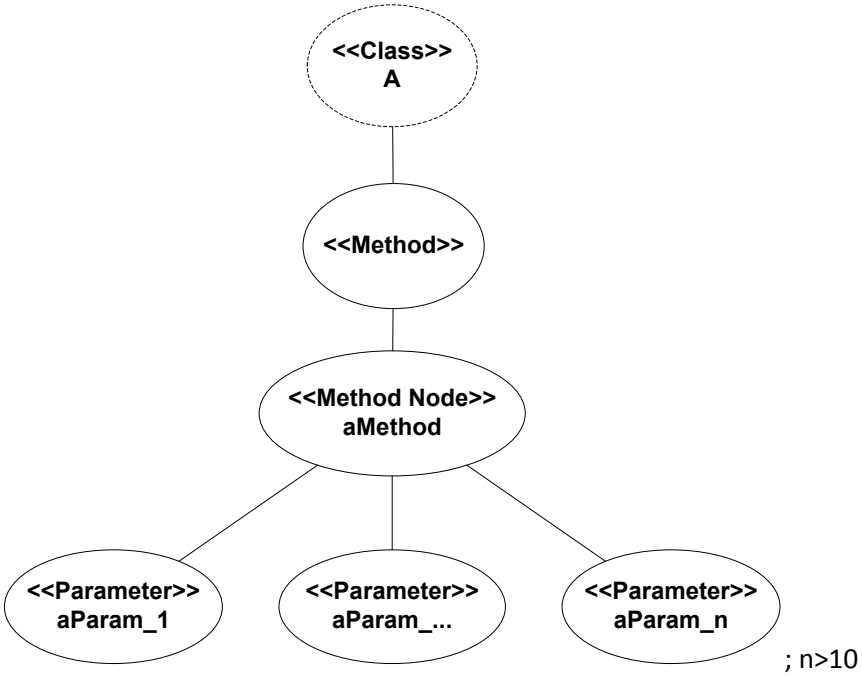
รูปที่ 3.6 แม่แบบในการค้นหาข้อบกพร่องประเภท Large Class

Long Parameter List

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อมีการผ่านพารามิเตอร์เข้าสู่เมท็อดเป็นจำนวนมาก [8]

แนวทางในการพิจารณา: นับจากจำนวนพารามิเตอร์ที่ผ่านเข้าสู่เมท็อด ซึ่งมากกว่าจำนวนที่กำหนด ไว้ โดยจะกำหนดจากปัจจัยหลายๆ อย่าง เช่น การทำงานของเมท็อด หรือโดเมนที่ซอฟต์แวร์เกี่ยวข้องด้วย ในงานวิจัยนี้ได้ใช้ค่าจำนวนพารามิเตอร์ที่ 10 อ้างอิงจากค่าปริยายของโปรแกรม CheckStyle และ PMD ซึ่งเป็นโปรแกรมตรวจรูปแบบรหัสต้นฉบับภาษาจาวา

แม่แบบ: แม่แบบตามรูปที่ 3.7



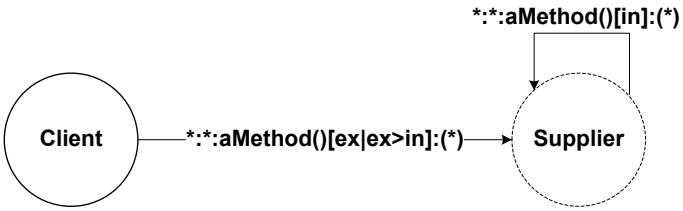
รูปที่ 3.7 แม่แบบในการค้นหาข้อบกพร่องประเภท Long Parameter Lists

Feature Envy

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อมีจำนวนครั้งการเรียกเมท็อดของคลาสหนึ่งจากภายนอกมากกว่าเมื่อเปรียบเทียบกับการเรียกใช้จากภายในคลาสนั้นๆ เอง [8]

แนวทางในการพิจารณา: พิจารณาจากความสัมพันธ์การเรียกใช้งานเมท็อด โดยจะเปรียบเทียบ จำนวนครั้งในการเรียกใช้งานจากภายนอก กับจำนวนครั้งที่เรียกจากภายในคลาสนั้นๆ

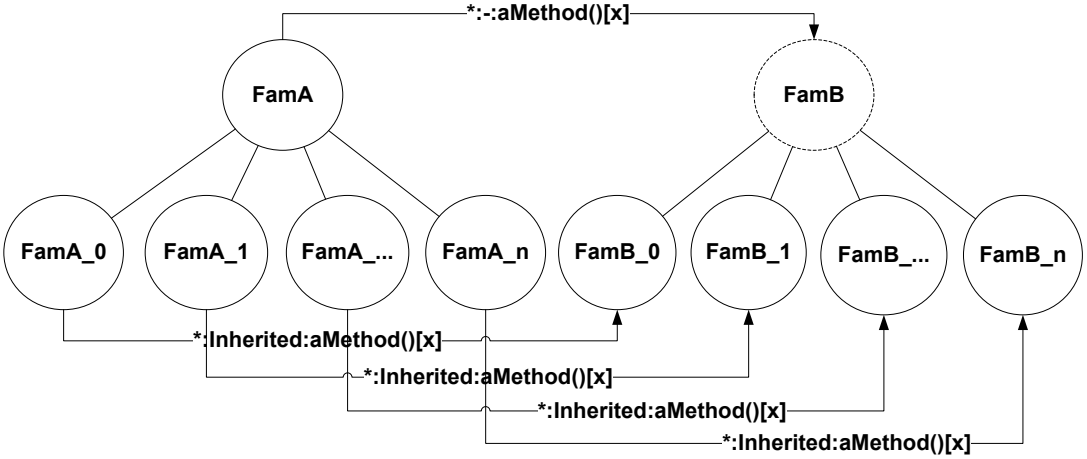
แม่แบบ: การเรียกใช้งานเมท็อดแสดงด้วยเส้นแสดงความสัมพันธ์ในแผนภาพกราฟตรรกะ สามารถสร้างออกมาเป็นแม่แบบได้ตามรูปที่ 3.8



รูปที่ 3.8 แม่แบบในการค้นหาข้อบกพร่องประเภท Feature Envy

Parallel Inheritance Hierarchy

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อมีการสร้างคลาสลูกในโครงสร้างการสืบทอดหนึ่ง แล้วมี ผลให้ต้องสร้างคลาสลูกในอีกโครงสร้างการสืบทอดหนึ่งด้วย [8] ซึ่งจะทำให้เกิดความสัมพันธ์การเรียกใช้เมท็อด เป็นในลักษณะการจับคู่กันระหว่างโครงสร้างการสืบทอดทั้งสอง ตามรูปที่ 3.9



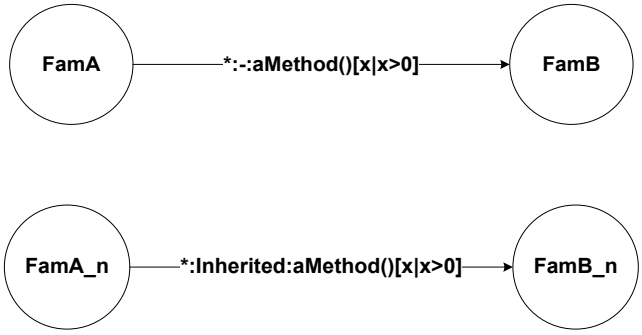
รูปที่ 3.9 แม่แบบในการค้นหาข้อบกพร่องประเภท Parallel Inheritance Hierarchy

แนวทางในการพิจารณา: ความสัมพันธ์การเรียกใช้งานเมท็อดที่มีลักษณะการจับคู่กันระหว่างสองโครงสร้างการสืบทอด โดยหากพบความสัมพันธ์การเรียกใช้งานที่รากของการสืบทอดไปยังโหนดรากของ โครงสร้างการสืบทอดใดๆ จะทำการตรวจสอบลูกในระดับที่อยู่ติดกันของสองโหนดนั้นๆ ว่ามีการเรียกใช้งานเมท์ อดเดียวกันในลักษณะของการจับคู่หรือไม่

แม่แบบ: แม่แบบในรูปใช้ในการตรวจสอบความสัมพันธ์ของโหนดราก ซึ่งหากตรวจพบจะใช้แม่แบบสำหรับการตรวจสอบความสัมพันธ์ของโหนดลูกตามรูป

ขั้นตอนการค้นหา: ในการค้นหาข้อบกพร่องในโมเดลการออกแบบประเภท Parallel Inheritance จะ เริ่มจากการค้นหาโครงสร้างที่น่าสงสัยว่าจะเกิดข้อบกพร่องในโมเดลการออกแบบ

โดยโมเดลที่น่าสงสัยจะมี ลักษณะของโครงสร้างการสืบทอดที่คล้ายคลึงกันในสองแผนภาพต้นไม้การสืบทอดใดๆ ดังแสดงตามรูปที่ 3.9 โดย จะทำการเปรียบเทียบทุกแผนภาพต้นไม้การสืบทอด ทำการตรวจสอบการเรียกใช้งานคุณลักษณะจากแผนภาพ กราฟด้วย แม่แบบตามรูปที่ 3.10



รูปที่ 3.10 แม่แบบในการตรวจสอบความสัมพันธ์ในแผนภาพกราฟตรรกะ ของการค้นหาข้อบกพร่องประเภท Parallel Inheritance Hierarchy

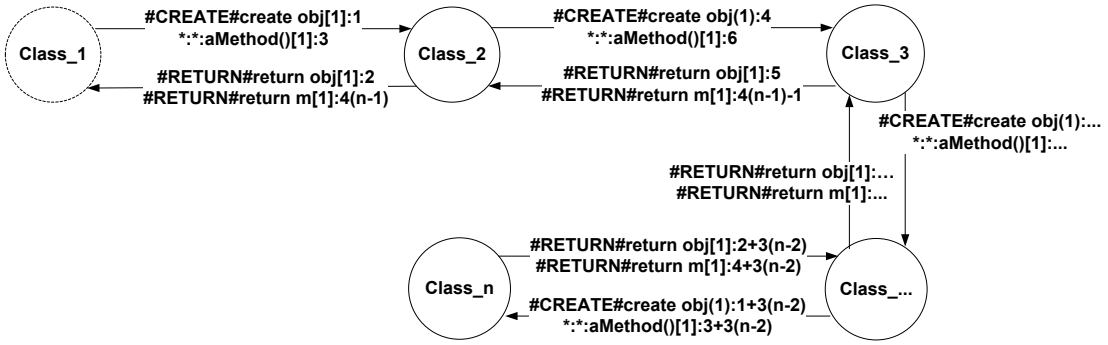
Message Chain

นิยาม: การที่มีการเรียกเมท็อดของคลาสหนึ่ง ผ่านอินสแตนท์ของอีกคลาสหนึ่งต่อกันไปเรื่อยๆ ทำให้ เกิดการขึ้นต่อกันสูง ควรเปลี่ยนมาเรียกจากอินสแตนท์ของคลาสนั้นโดยตรงแทน [8]

แนวทางในการพิจารณา: พิจารณาจากลำดับการเรียกเมท็อด โดยจะมีลักษณะการเรียกจากอินสแตนท์ของคลาสอื่นต่อกันไปเรื่อยๆ

แม่แบบ: แม่แบบ ดังแสดงในรูปที่ 3.11

ขั้นตอนการค้นหา: ในการค้นหาข้อบกพร่องประเภท Message Chain จะท่องไปตามโหนดใน แผนภาพกราฟตรรกะ หากพบความสัมพันธ์การสร้างอินสแตนท์ ตามด้วยการเรียกใช้เมท็อดจากอินสแตนท์นั้นจะ ดูลำดับการทำงานต่อไปเรื่อยๆเพื่อพิจารณาว่าเป็นการเรียกต่อไปยังอินสแตนท์ของคลาสอื่นอีกหรือไม่จนถึงลำดับที่ทำการส่งค่าคืนมายังคลาสต้นทาง



รูปที่ 3.11 แม่แบบในการค้นหาข้อบกพร่องประเภท Message Chain

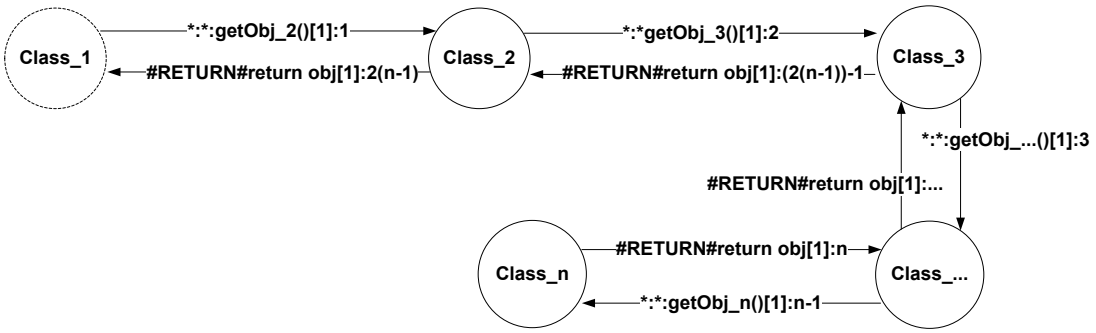
Middle Man

นิยาม: การที่มีคลาสที่ทำหน้าที่เป็นตัวกลางการติดต่อระหว่างอ็อบเจกต์ [8]

แนวทางในการพิจารณา: พิจารณาจากลำดับการเรียกเมท็อด โดยจะมีลักษณะการเรียกต่อกันไป เรื่อยๆ โดยไม่มีการทำงานหลักอื่น

แม่แบบ: แม่แบบ ดังแสดงในรูปที่ 3.12

ขั้นตอนการค้นหา: ในการค้นหาข้อบกพร่องประเภท Message Chain จะท่องไปตามโหนดใน แผนภาพกราฟตรรกะ หากพบความสัมพันธ์การเรียกใช้เมท็อดเพื่อเข้าถึงข้อมูล จะดูลำดับการทำงานต่อไปเรื่อยๆ เพื่อพิจารณาว่าเป็นเมท็อดในการเข้าถึงข้อมูลไปยังคลาสอื่นอีกหรือไม่ จนถึงลำดับที่ทำการส่งค่าคืนมายังคลาสต้นทาง



รูปที่ 3.12 แม่แบบในการค้นหาข้อบกพร่องประเภท Middle Man

Refused Bequest

นิยาม: ข้อบกพร่องในโมเดลการออกแบบ โดยการที่คลาสลูกได้รับการสืบทอดคุณลักษณะที่ไม่จำเป็นมาด้วย [8]

แนวทางในการพิจารณา: พิจารณาจากการเรียกใช้งานเมท็อดของคลาส โดยจะถือว่าเป็นข้อบกพร่อง เมื่อพบว่าไม่ได้มีการเรียกใช้งานเมท็อดที่ได้รับสืบทอดมา แสดงด้วยเส้นแสดงความสัมพันธ์ในแผนภาพ กราฟตรรกะ

แม่แบบ: การเรียกใช้งานเมท็อดแสดงด้วยเส้นแสดงความสัมพันธ์ในแผนภาพกราฟตรรกะ สามารถสร้างออกมาเป็นแม่แบบได้ตามรูปที่ 3.13

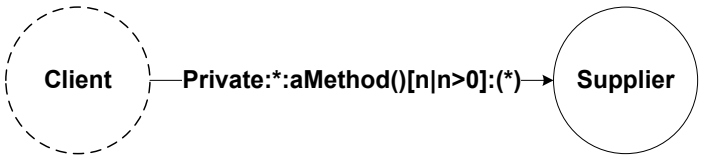
รูปที่ 3.13 แม่แบบในการค้นหาข้อบกพร่องประเภท Refused Bequest

Inappropriate Intimacy

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อพบว่ามีการพยายามเรียกใช้คุณลักษณะที่เป็นไพรเวทของ คลาสอื่น [8]

แนวทางในการพิจารณา: สามารถดูได้จากความสัมพันธ์การเรียกใช้งานกันระหว่างคลาส โดยจะถือ ว่าเป็นข้อบกพร่องเมื่อพบความสัมพันธ์การเรียกใช้งานคลาสที่เป็นไพรเวทจากภายนอก

แม่แบบ: การเรียกใช้งานแสดงด้วยเส้นแสดงความสัมพันธ์ในแผนภาพกราฟตรรกะ สามารถสร้าง ออกมาเป็นแม่แบบได้ตามรูปที่ 3.14



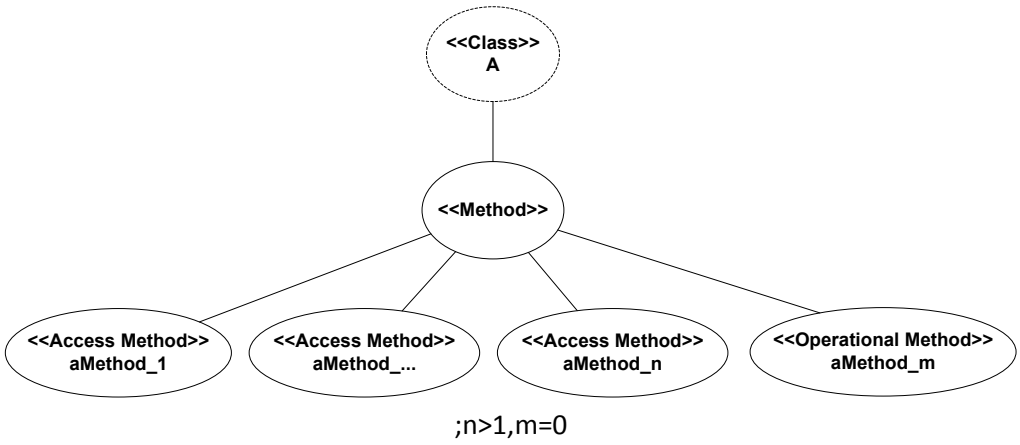
รูปที่ 3.14 แม่แบบในการค้นหาข้อบกพร่องประเภท Inappropriate Intimacy

Data Class

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อคลาสที่มีเพียงแอคเซสเมท็อดเท่านั้นโดยไม่มีฟังก์ชันการ ทำงานหลักอื่น[8]

แนวทางในการพิจารณา: พิจารณาจากการที่เมท็อดที่อยู่ภายในคลาส โดยจะถือว่าเป็นข้อบกพร่อง เมื่อพบเพียงแอคเซสเมท็อด (<<Access Method Node>>) เท่านั้น

แม่แบบ: แม่แบบตามรูปที่ 3.15



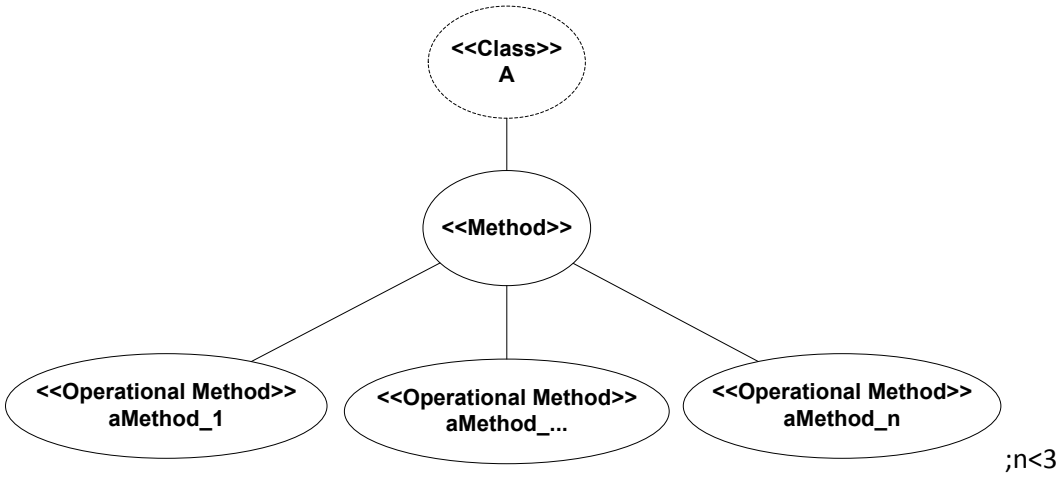
รูปที่ 3.15 แม่แบบในการค้นหาข้อบกพร่องประเภท Data Class

Lazy Class

นิยาม: ข้อบกพร่องในโมเดลการออกแบบเมื่อคลาสมีฟังก์ชันการทำงานที่ไม่มาก[8]

แนวทางในการพิจารณา: พิจารณาจากจำนวนเมท็อดที่ปรากฏอยู่ในคลาส โดยจะดูเฉพาะเมท็อดเป็นการทำงาน (<<Operational Method>>) โดยในงานวิจัยนี้ได้กำหนดว่าหากเมท็อดมีจำนวนน้อยกว่า 3 จะถือว่าเป็นข้อบกพร่อง

แม่แบบ: แม่แบบตามรูปที่ 3.16



รูปที่ 3.16 แม่แบบในการค้นหาข้อบกพร่องประเภท Lazy Class

3.3 การค้นหาข้อบกพร่อง

หลังจากทำการแปลงโมเดลการออกแบบเป็นแผนภาพตามที่ได้นำเสนอไปแล้วนั้น การค้นหาข้อบกพร่อง สามารถทำได้ด้วยการนำแม่แบบที่ได้จากหัวข้อ 3.2.2 มาเปรียบเทียบกับแผนภาพที่แสดงโมเดลการออกแบบ ในการสร้างแม่แบบของข้อบกพร่องจะใช้ข้อกำหนดเดียวกับการสร้างแผนภาพแสดงโมเดลการออกแบบ ซึ่งจะอยู่ในรูปของ แผนภาพต้นไม้การสืบทอด แผนภาพต้นไม้คุณลักษณะ หรือ แผนภาพกราฟตรรกะ โดยขึ้นอยู่กับประเภทของข้อบกพร่อง

ในการเปรียบเทียบจะท่องไปตามโหนดในแผนภาพแสดงโมเดลการออกแบบ เพื่อนำแม่แบบเปรียบเทียบกับแต่ละโหนด หรือเวอร์เท็กซ์ เพื่อตรวจสอบคุณลักษณะ โดยจะถือว่าตรวจพบข้อบกพร่อง เมื่อคุณลักษณะของโหนด หรือเวอร์เทกซ์นั้นตรงกับแม่แบบ

บทที่ 4
ออกแบบและพัฒนาเครื่องมือช่วยในการค้นหาข้อบกพร่อง

ในบทนี้จะแสดงรายละเอียดการออกแบบเครื่องมือช่วยในการค้นหาข้อบกพร่องตามวิธีการที่ได้นำเสนอด้วยแผนภาพยูเอ็มแอล ประกอบด้วย แผนภาพยูสเคสความสามารถของระบบ การออกแบบแพ็กเก็จ และโครงสร้างของคลาส และแผนภาพซีเควนซ์แสดงกระบวนการค้นหาข้อบกพร่อง

4.1 สภาพแวดล้อมของการพัฒนาเครื่องมือ

เครื่องมือสำหรับทดสอบในงานวิจัยนี้จะถูกพัฒนาขึ้นมาด้วยภาษาจาวา (Java) เวอร์ชัน 6 โดยใช้ Eclipse IDE for Java Developers ช่วยในการพัฒนา เครื่องมือที่พัฒนาแล้ว จะสามารถใช้งานได้บนเครื่องที่ลง Java Runtime Environment (JRE) โดยมีความต้องการที่เวอร์ชัน 6 หรือสูงกว่า สภาพแวดล้อมของการพัฒนาสรุปได้ดังตารางที่ 4.1

ตารางที่ 4.1 ตารางสรุปสภาพแวดล้อมของการพัฒนาเครื่องมือ

หัวข้อ	คำอธิบาย
ภาษาที่ใช้พัฒนา	Java เวอร์ชัน 6
ความต้องการด้านซอฟต์แวร์	Java Platform, Standard Edition (Java SE) เวอร์ชัน 6
เครื่องมือที่ช่วยในการพัฒนา	Eclipse IDE for Java Developers

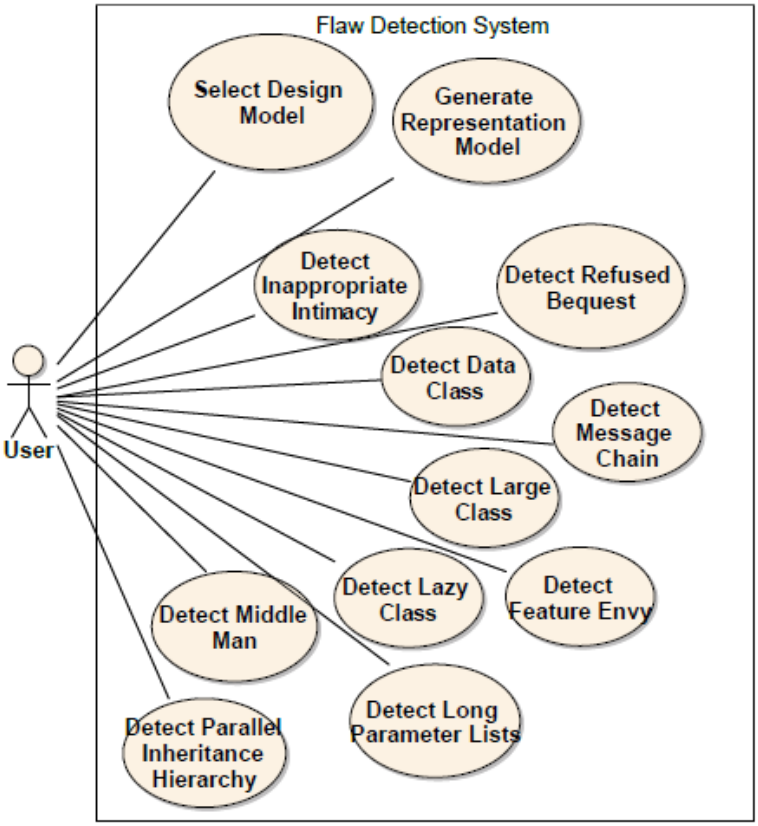
4.2 แผนภาพยูสเคส

แผนภาพยูสเคสของระบบค้นหาข้อบกพร่องในโมเดลการออกแบบ ตามรูปที่ 1 แสดงความต้องการของระบบ ในมุมมองของผู้ใช้งาน มีรายละเอียดตามรูปที่ 4.1 ผู้ใช้งานสามารถใช้งานระบบค้นหาข้อบกพร่องได้ ดังนี้

- ผู้ใช้สามารถเลือกไฟล์โมเดลการออกแบบซอฟต์แวร์ที่ต้องการค้นหาข้อบกพร่องได้
- ผู้ใช้สามารถสั่งให้โปรแกรมสร้างแผนภาพโมเดลการออกแบบซอฟต์แวร์ได

- ผู้ใช้สามารถสั่งให้โปรแกรมค้นหาข้อบกพร่องในโมเดลที่เลือกได้ โดยจะประกอบไปด้วย
    - ข้อบกพร่องประเภท Large Class
    - ข้อบกพร่องประเภท Long Parameter Lists
    - ข้อบกพร่องประเภท Feature Envy
    - ข้อบกพร่องประเภท Parallel Inheritance Hierarchy
    - ข้อบกพร่องประเภท Message Chain
    - ข้อบกพร่องประเภท Middle Man
    - ข้อบกพร่องประเภท Refused Bequest
    - ข้อบกพร่องประเภท Inappropriate Intimacy
    - ข้อบกพร่องประเภท Data Class
    - ข้อบกพร่องประเภท Lazy Class

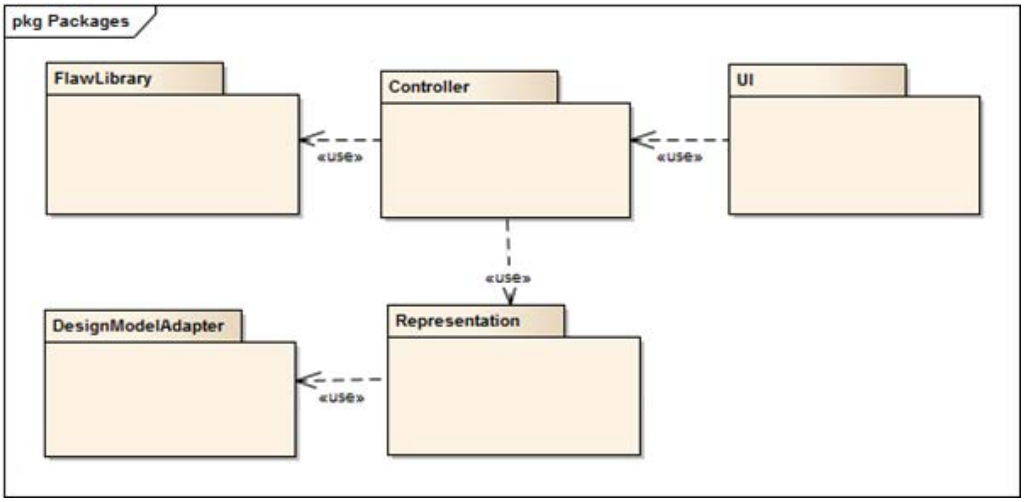
ในการค้นหาข้อบกพร่อง ระบบจะทำการอ่านแผนภาพยูเอ็มแอล ที่ผู้ใช้งานเลือกเข้าสู่ระบบ ข้อมูลโมเดลการออกแบบจะถูกอ่าน และแปลงออกมาในรูปของแผนภาพกราฟ และแผนภาพต้นไม้ที่ใช้แสดงแทนโมเดลการออกแบบซอฟต์แวร์ ข้อมูลคุณลักษณะต่างๆจะถูกจัดเก็บไว้สำหรับการค้นหาข้อบกพร่อง แผนภาพที่ถูกแปลงมานั้นจะถูกนำมาตรวจเช็คกับแม่แบบที่พัฒนาออกมาตามที่ได้ออกแบบไว้



รูปที่ 4.1 แผนภาพยูสเคสของเครื่องมือช่วยในการค้นหาข้อบกพร่อง

4.3 การออกแบบแพ็กเก็จ และโครงสร้างคลาส

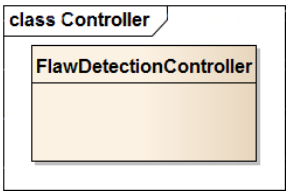
ในการพัฒนาเครื่องมือสำหรับงานวิจัยนี้ สามารถแบ่งแพ็กเก็จออกได้ดังแสดงตามรูปที่ 4.2 ประกอบไปด้วย แพ็กเก็จส่วนติดต่อกับผู้ใช้งาน (UI) แพ็กเก็จส่วนควบคุมการค้นหาข้อบกพร่อง (Controller) แพ็กเก็จส่วนโมเดลจัดเก็บแผนภาพกราฟ และแผนภาพต้นไม้ (Representation) แพ็กเก็จส่วนเชื่อมต่อกับโมเดลการออกแบบซอฟต์แวร์ (DesignModelAdapter) และ แพ็กเก็จส่วนจัดเก็บแม่แบบที่ใช้ในการค้นหาข้อบกพร่อง (FlawLibrary)



รูปที่ 4.2 แผนภาพคลาสแสดงความสัมพันธ์ระหว่างแพ็กเกจต่างๆของระบบ

4.3.1 แพ็กเก็จส่วนควบคุมการค้นหาข้อบกพร่อง (Controller)

แพ็กเก็จส่วนควบคุมการค้นหาข้อบกพร่อง ประกอบไปด้วยคลาส FlawDetectionController ทำหน้าที่ในการควบคุมกระบวนการค้นหาข้อบกพร่อง ดังแสดงในรูปที่ 4.3 โดยเมื่อผู้ใช้งานทำการสั่งการให้ระบบค้นหาข้อบกพร่อง และได้เลือกไฟล์โมเดลการออกแบบแล้ว ส่วนควบคุมจะทำหน้าที่ติดต่อไปยังส่วนรับผิดชอบในการอ่านแผนภาพยูเอ็มแอล และสร้างโมเดลแสดงแทนโมเดลการออกแบบ และนำแผนภาพนั้นส่งไปยังส่วนค้นหาข้อบกพร่อง เพื่อทำการค้นหาข้อบกพร่องด้วยการตรวจสอบกับแม่แบบข้อบกพร่องแต่ละประเภท

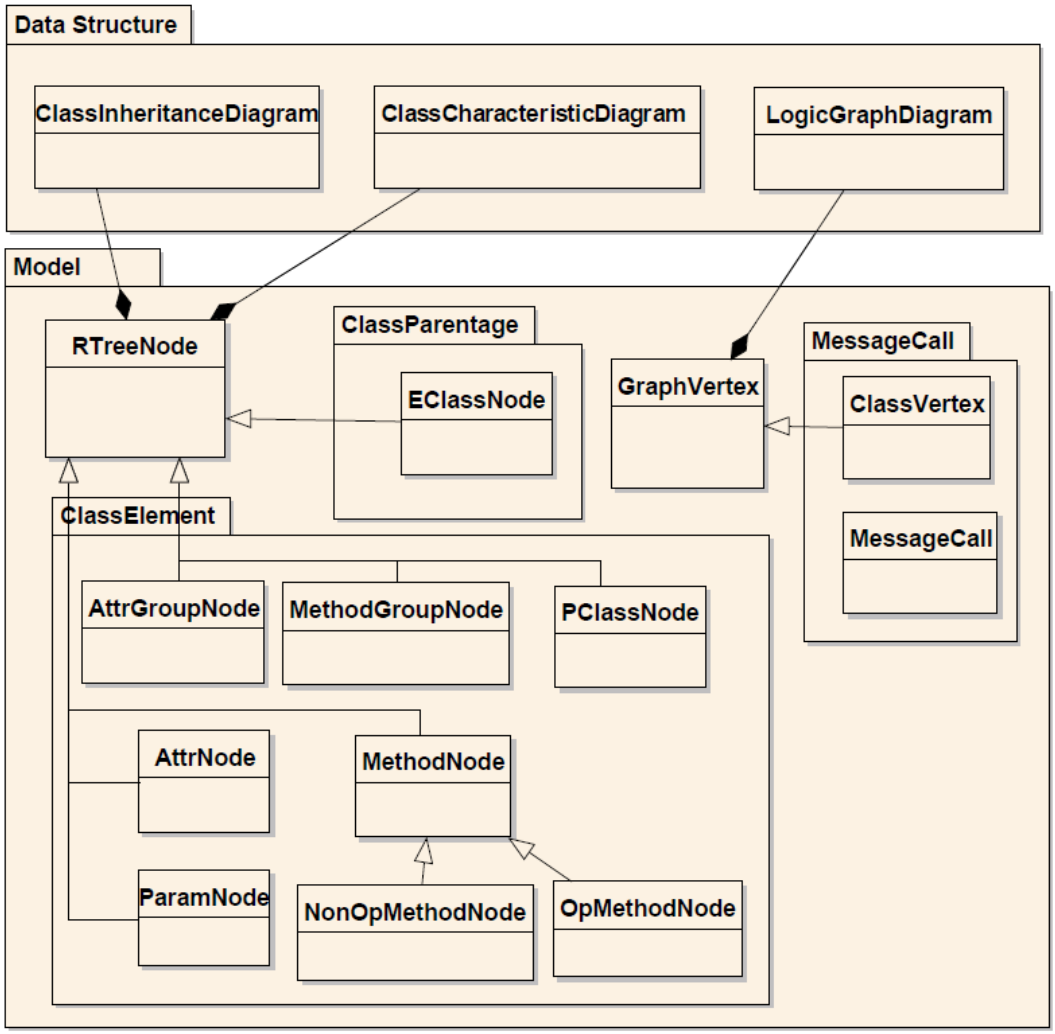


รูปที่ 4.3 แผนภาพคลาสในภาพรวมของส่วนควบคุมการค้นหาข้อบกพร่อง

4.3.2 แพ็กเก็จส่วนโมเดลจัดเก็บแผนภาพกราฟ และแผนภาพต้นไม้ (Representation)

แพ็กเก็จส่วนโมเดลจัดเก็บแผนภาพกราฟ และแผนภาพต้นไม้ ทำหน้าที่ในการจัดเก็บข้อมูลการออกแบบซอฟต์แวร์ ในรูปของแผนภาพกราฟ และแผนภาพต้นไม้ที่แปลงเสร็จแล้ว ดังแสดงในรูปที่ 4.4 แพ็กเก็จส่วนนี้จะประกอบไปด้วย โครงสร้างแผนภาพทั้งสามแผนภาพ ได้แก่ ClassInheritanceDiagram ClassCharacteristicDiagram และ LogicGraphDiagram และส่วนประกอบของแต่ละแผนภาพ ได้แก่ โหนดประเภทต่างๆในแผนภาพต้นไม้ สืบทอดมาจากคลาส RTreeNode และ เวอร์เท็กซ์ในแผนภาพกราฟ สืบทอดจากคลาส GraphVertex ตารางที่ 4.2

แสดงคำอธิบายส่วนประกอบของคลาสแสดงโหนดต่างๆในแผนภาพแสดงแทนโมเดลการออกแบบ



รูปที่ 4.4 แผนภาพคลาสในภาพรวมของส่วนโมเดลจัดเก็บแผนภาพกราฟ และแผนภาพต้นไม้

ตารางที่ 4.2 คำอธิบายส่วนประกอบของคลาสแสดงโหนดต่างๆในแผนภาพแสดงแทนโมเดลการออกแบบ

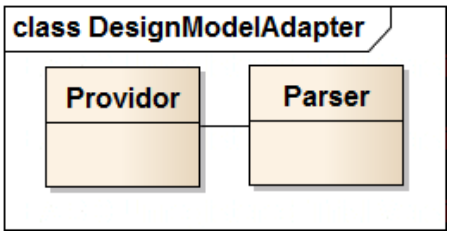
แผนภาพ	ส่วนประกอบ	คำอธิบาย
แผนภาพต้นไม้การสืบทอด	EClassNode	คลาสแสดงแทนโหนดของคลาสในแผนภาพ

ตารางที่ 4.2 คำอธิบายส่วนประกอบของคลาสแสดงโหนดต่างๆในแผนภาพแสดงแทนโมเดลการออกแบบ (ต่อ)

แผนภาพ	ส่วนประกอบ	คำอธิบาย
แผนภาพต้นไม้คุณลักษณะ	PClassNode	คลาสแสดงแทนโหนดของคลาสในแผนภาพ
	AttrGroupNode	คลาสแสดงแทนโหนดของกลุ่มของแอททริบิวต์ในแผนภาพ
	MethodGroupNode	คลาสแสดงแทนโหนดของกลุ่มของเมท็อดในแผนภาพ
	MethodNode	คลาสแสดงแทนโหนดของเมท็อดในแผนภาพ
	AttrNode	คลาสแสดงแทนโหนดของแอททริบิวต์ในแผนภาพ
	ParamNode	คลาสแสดงแทนโหนดของพารามิเตอร์ในแผนภาพ
แผนภาพกราฟตรรกะ	ClassVertex	คลาสของเวอร์เท็กซ์ของคลาสในแผนภาพ
	MessageCall	คลาสของความสัมพันธ์ในแผนภาพ

4.3.3 แพ็กเก็จส่วนเชื่อมต่อกับโมเดลการออกแบบซอฟต์แวร์ (DesignModelAdapter)

แพ็กเก็จส่วนเชื่อมต่อกับโมเดลการออกแบบซอฟต์แวร์ จะเป็นส่วนที่ติดต่อระหว่างโมเดลการออกแบบที่อยู่ในรูปของแผนภาพยูเอ็มแอล และระบบค้นหาข้อบกพร่อง ตามรูปที่ 4.5 ประกอบไปด้วยคลาส Parser ทำหน้าที่ในการอ่านแผนภาพยูเอ็มแอล และคลาส Providor ทำหน้าที่ในการจัดการข้อมูลการออกแบบ เพื่อให้ส่วนสร้างแผนภาพเรียกใช้ เพื่อนำไปสร้างเป็นแผนภาพแสดงแทนโมเดลการออกแบบต่อไป

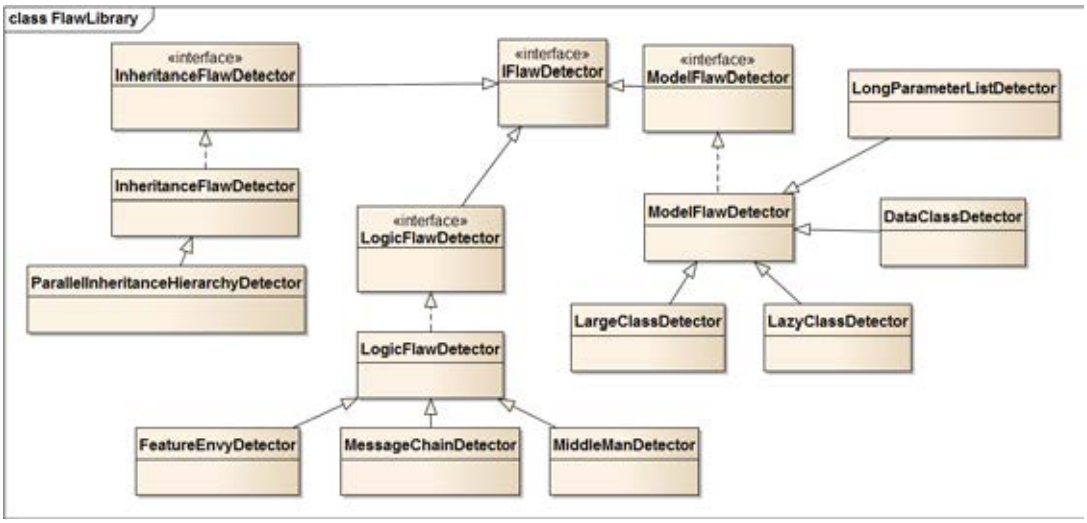


รูปที่ 4.5 แผนภาพคลาสในภาพรวมของส่วนเชื่อมต่อกับโมเดลการออกแบบซอฟต์แวร์

4.3.4 แพ็กเก็จส่วนจัดเก็บแม่แบบที่ใช้ในการค้นหาข้อบกพร่อง (FlawLibrary)

แม่แบบของข้อบกพร่องที่นำเสนอในงานวิจัยนี้ จะถูกสร้างออกมาเป็นโปรแกรมต้นฉบับภาษาจาวา และจัดเก็บอยู่ในแพ็กเก็จนี้ ตามรูปที่ 4.6 ในการออกแบบจะใช้หลักการของอินเตอร์เฟสโดยคลาสสำหรับค้นหาข้อบกพร่องแต่ละอัน จะจัดเป็นประเภท IFlawDetector แบ่งประเภทออกตามแผนภาพที่ค้นหา ได้เป็น

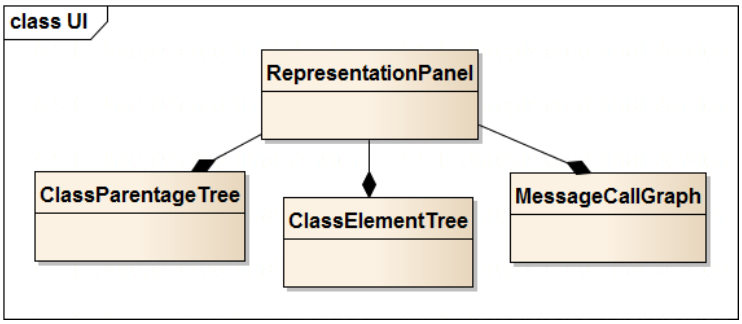
- InheritanceFlawDetector รับผิดชอบกระบวนการในการค้นหาข้อบกพร่องที่เกี่ยวกับการสืบทอดจากแผนภาพต้นไม้การสืบทอดประกอบไปด้วย
    - ParallelInheritanceHierarchyDetector
- LogicFlawDetector รับผิดชอบกระบวนการในการค้นหาข้อบกพร่องที่เกี่ยวกับโครงสร้างของคลาสจากแผนภาพต้นไม้คุณลักษณะ ประกอบไปด้วย
    - FeatureEnvyDetector
    - MiddleManDetector
    - MessageChainDetector
- LogicFlawDetector รับผิดชอบกระบวนการในการค้นหาข้อบกพร่องที่เกี่ยวกับการทำงานจากแผนภาพกราฟตรรกะ ประกอบไปด้วย
    - LazyClassDetector
    - DataClassDetector
    - LongParameterListDetector
    - LargeClassDetector



รูปที่ 4.6 แผนภาพคลาสในภาพรวมของส่วนจัดเก็บแม่แบบที่ใช้ในการค้นหาข้อบกพร่อง

4.3.5 แพ็กเก็จส่วนติดต่อกับผู้ใช้ (UI)

แพ็กเก็จส่วนติดต่อกับผู้ใช้งานจะทำหน้าที่รับคำสั่งจากผู้ใช้ และแสดงผลแผนภาพกราฟและแผนภาพต้นไม้ที่แสดงแทนโมเดลการออกแบบ และแสดงผลการค้นหาข้อบกพร่อง ดังแสดงในรูปที่ 4.7

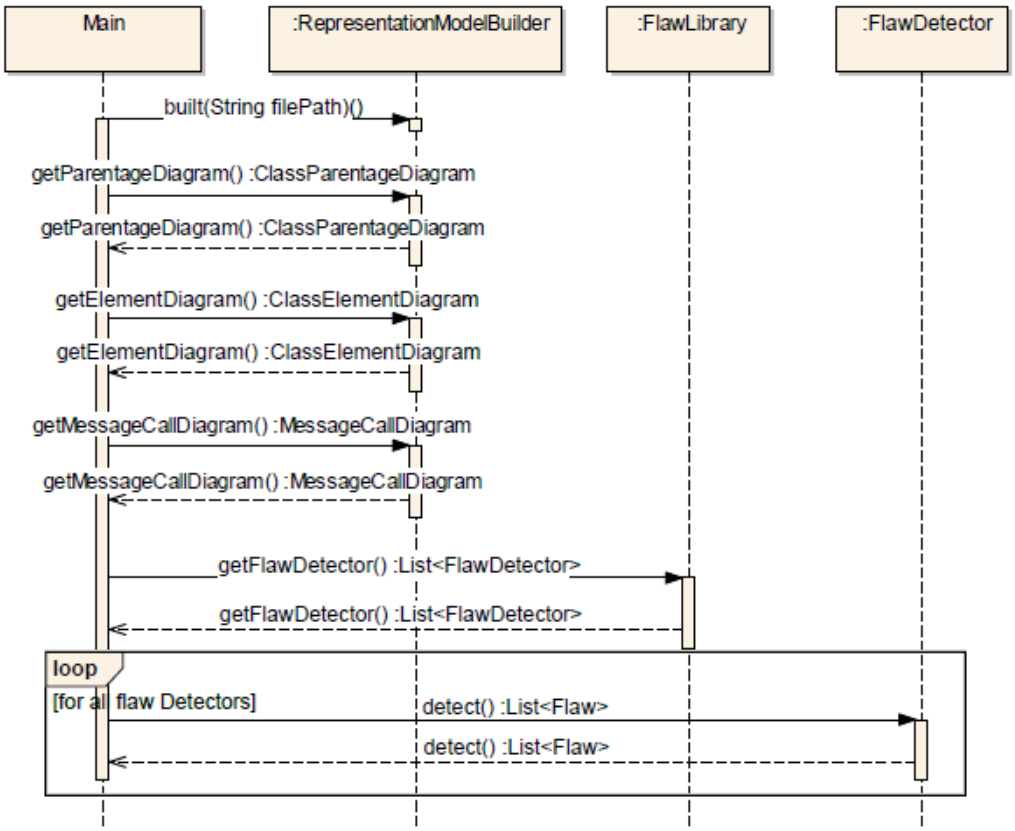


รูปที่ 4.7 แผนภาพคลาสในภาพรวมของส่วนติดต่อกับผู้ใช้

4.4 แผนภาพซีเควนซ์

4.4.1 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่อง

แผนภาพซีเควนซ์ของกระบวนการ การค้นหาข้อบกพร่อง ดังรูปที่ 4.8 เมื่อผู้ใช้งานได้ทำการเลือกไฟล์โมเดลการออกแบบที่ต้องการแล้ว ระบบจะทำการสร้างแผนภาพที่ใช้แสดงแทนโมเดลการออกแบบ แม่แบบสำหรับตรวจหาข้อบกพร่องแต่ละประเภทจะถูกจัดเก็บไว้ภายใน FlawLibrary แม่แบบแต่ละตัวจะถูกนำมาใช้ตรวจสอบกับแผนภาพ เพื่อค้นหาข้อบกพร่อง



รูปที่ 4.8 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่อง

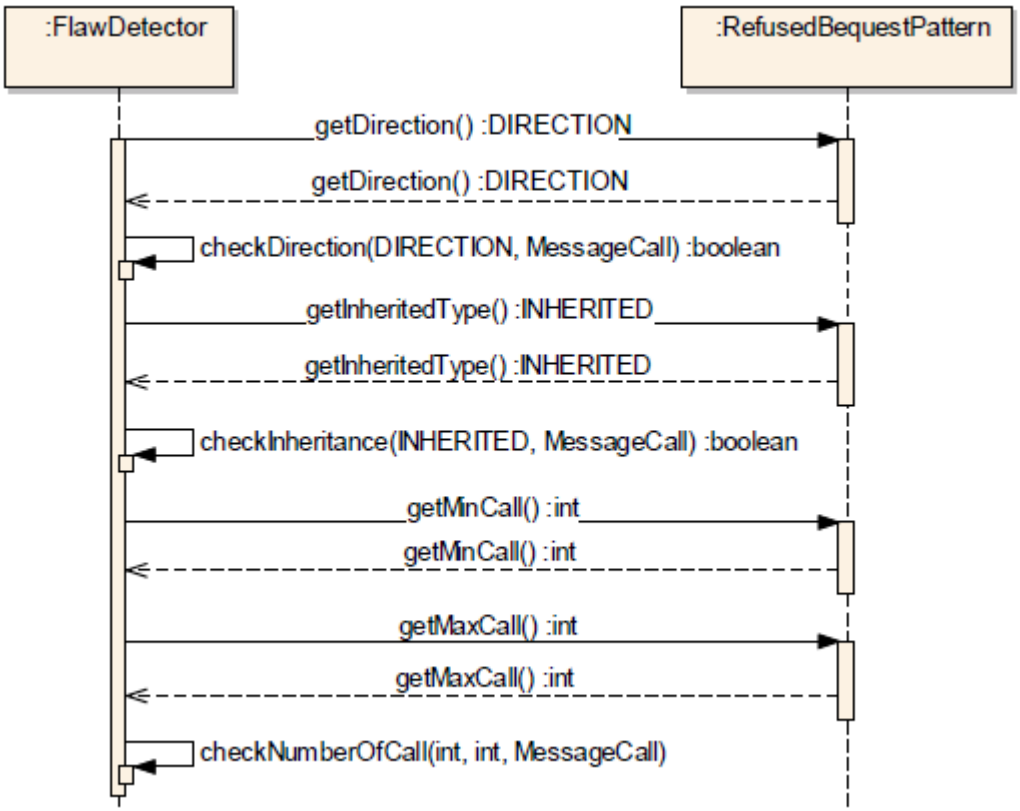
4.4.2 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Refused Bequest

แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Refused Bequest ดังรูปที่ 4.9 มีขั้นตอนการตรวจสอบ ดังนี้

(1) เริ่มจากการตรวจสอบทิศทางการเรียกใช้งานเมท็อดโดยจะกำหนดค่าแม่แบบไว้ให้ดูที่ความสัมพันธ์ที่มีทิศทางเข้าสู่โหนดที่กำลังตรวจสอบ

(2) ตรวจสอบประเภทเมท็อด ว่าเป็นเมท็อดที่ได้รับสืบทอดมาหรือไม่

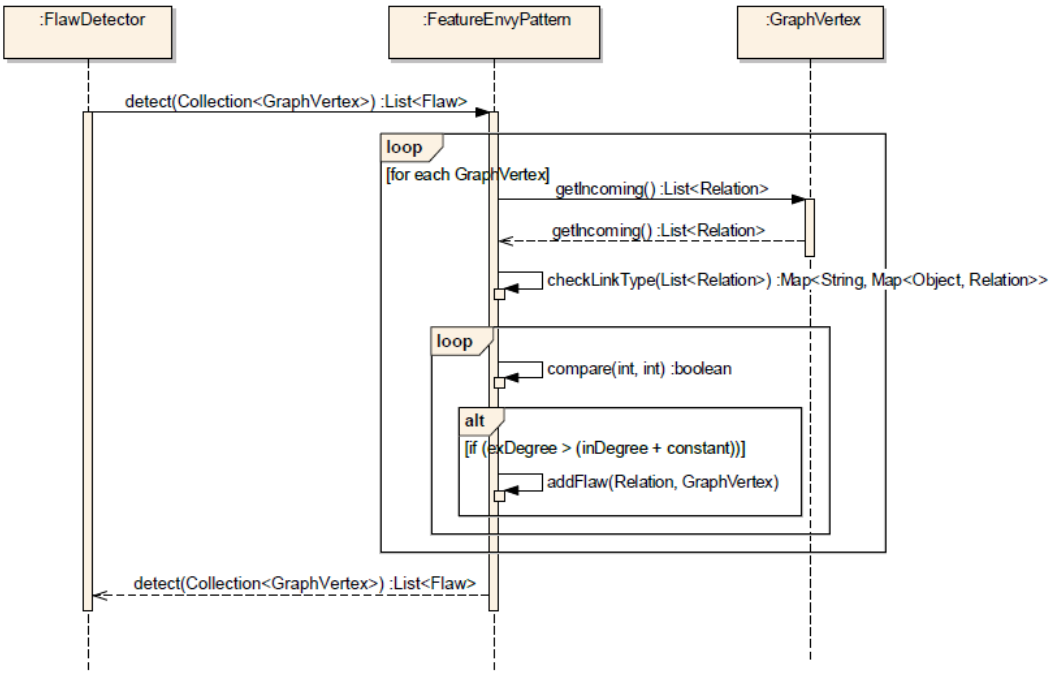
(3) ตรวจสอบการเรียกใช้งานเมท็อดดังกล่าว หากพบว่าเมท็อดที่ได้รับสืบทอดมาไม่มีการใช้งาน จะถือว่าเป็นข้อบกพร่อง



รูปที่ 4.9 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Refused Bequest

4.4.3 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Feature Envy

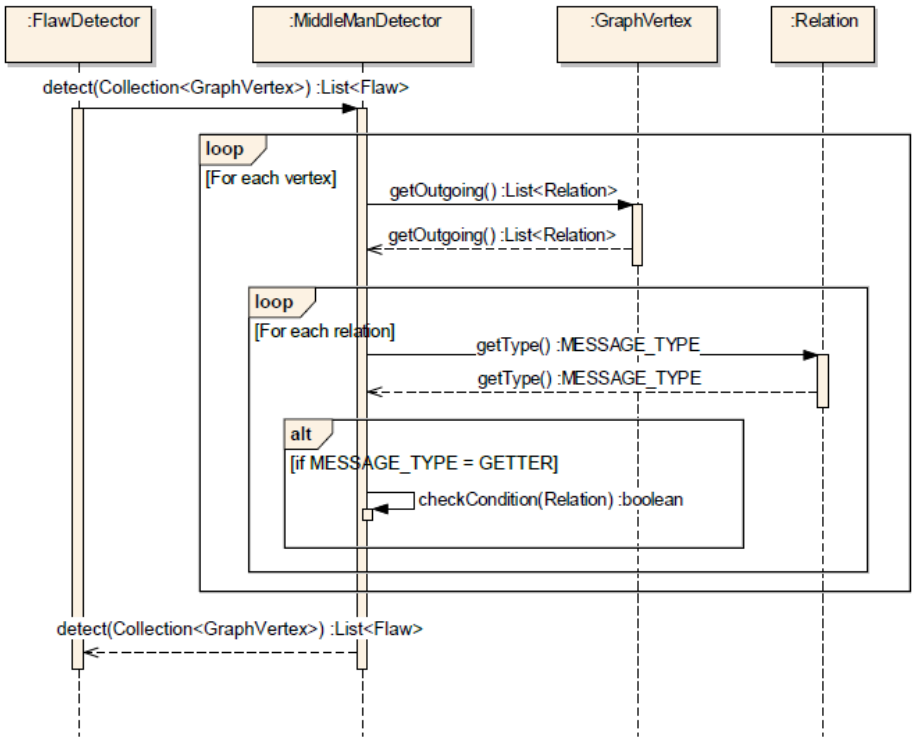
แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Feature Envy ดังแสดงในรูปที่ 4.10 ในแต่ละเวอร์เท็กซ์ในแผนภาพกราฟตรรกะ จะทำการตรวจสอบจำนวนครั้งการเรียกใช้งานแต่เมท็อด โดยเปรียบเทียบจำนวนครั้งการเรียกใช้งานจากภายในกับภายนอก หากพบว่ามีอัตราการเรียกใช้งานเมท็อดใดๆ จากภายนอกมากกว่าการเรียกจากภายในจะถือว่าเป็นข้อบกพร่องประเภท Feature Envy



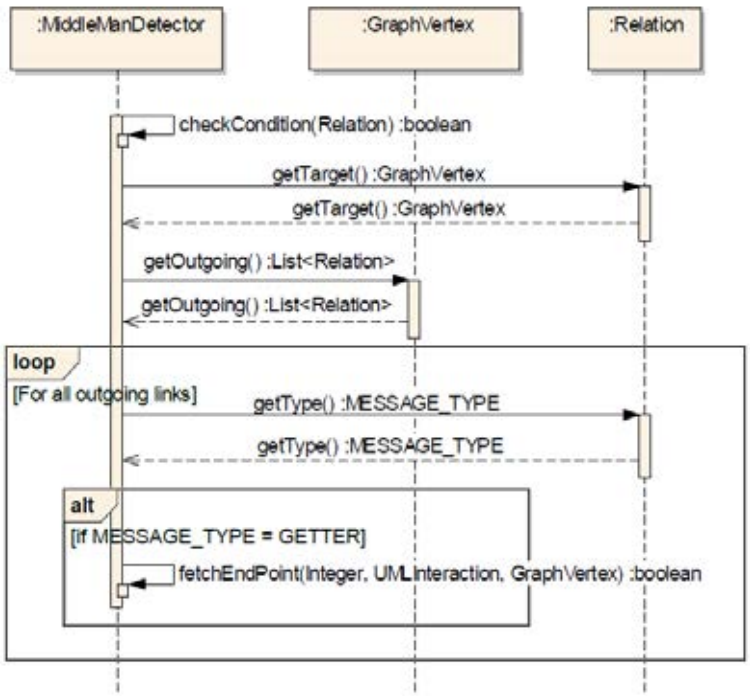
รูปที่ 4.10 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Feature Envy

4.4.4 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Middle Man

ในการค้นหาข้อบกพร่องประเภท Middle Man จะทำการตรวจสอบลำดับความสัมพันธ์การเรียกเมท็อดระหว่างคลาส โดยจะใช้ความสามารถในการเขียนโปรแกรมแบบการเรียกตัวเอง (Recursion) มาใช้ในการพัฒนา จากแม่แบบของข้อบกพร่องที่ได้ออกแบบไว้ การตรวจสอบจะท่องไปตามโหนดในแผนภาพกราฟตรรกะ ดูความสัมพันธ์ (Outgoing Relation) ตามรูปที่ 4.11 หากพบว่ามีการเรียกเมท็อดเพื่อคำนวณ จะทำการตรวจสอบกับโหนดปลายทาง ว่าได้ทำการเรียกต่อไปยังเมท็อดอื่นอีกในลักษณะลูกโซ่หรือไม่จนไปถึงเมท็อดสุดท้ายที่ทำการส่งค่ากลับ ดังแสดงตามรูปที่ 4.12



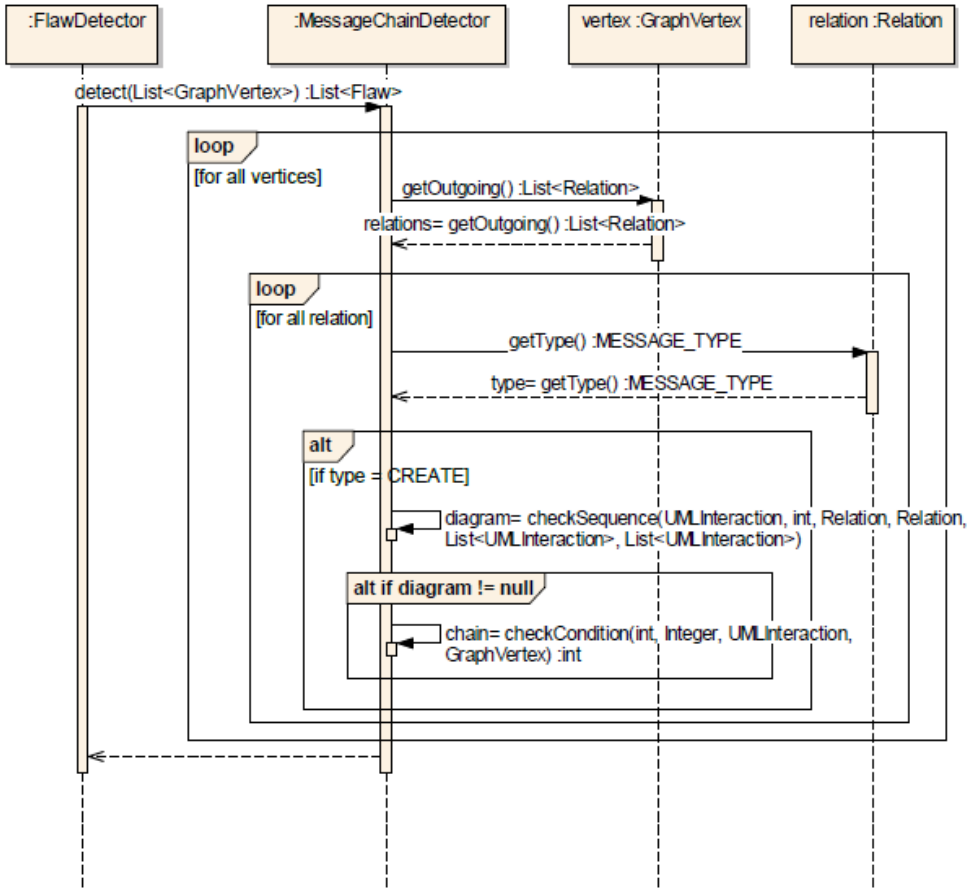
รูปที่ 4.11 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Middle Man ส่วนตรวจสอบเงื่อนไขเบื้องต้น



รูปที่ 4.12 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Middle Man ส่วนการเรียกตัวเองเพื่อตรวจสอบเงื่อนไขของข้อบกพร่อง

4.4.5 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Message Chain

ในการค้นหาข้อบกพร่องประเภท Message Chain จะทำการตรวจการเรียกเมื่อดเป็นลำดับตามแม่แบบ โดยในการพัฒนาจะใช้ความสามารถในการเขียนโปรแกรมแบบการเรียกตัวเอง (Recursion) การตรวจสอบจะท่องไปตามเวอร์เท็กซ์ในแผนภาพกราฟตรรกะ ในแต่ละเวอร์เท็กซ์จะตรวจสอบความสัมพันธ์ที่มีทิศทางออก (Outgoing Relation) หลังจากนั้นจึงไล่ไปยังเวอร์เท็กซ์ปลายทาง เพื่อตรวจสอบว่ามีการเรียกไปยังเวอร์เท็กซ์อื่นต่ออีกหรือไม่ ตามแผนภาพซีเควนซ์ในรูปที่ 4.13 หากตรวจพบความสัมพันธ์การเรียกในลักษณะที่เป็นลูกโซ่จะถือว่าเป็นข้อบกพร่องประเภท Message Chain



รูปที่ 4.13 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Message Chain ส่วนตรวจสอบเงื่อนไขความสัมพันธ์การสร้าง Object เบื้องต้น

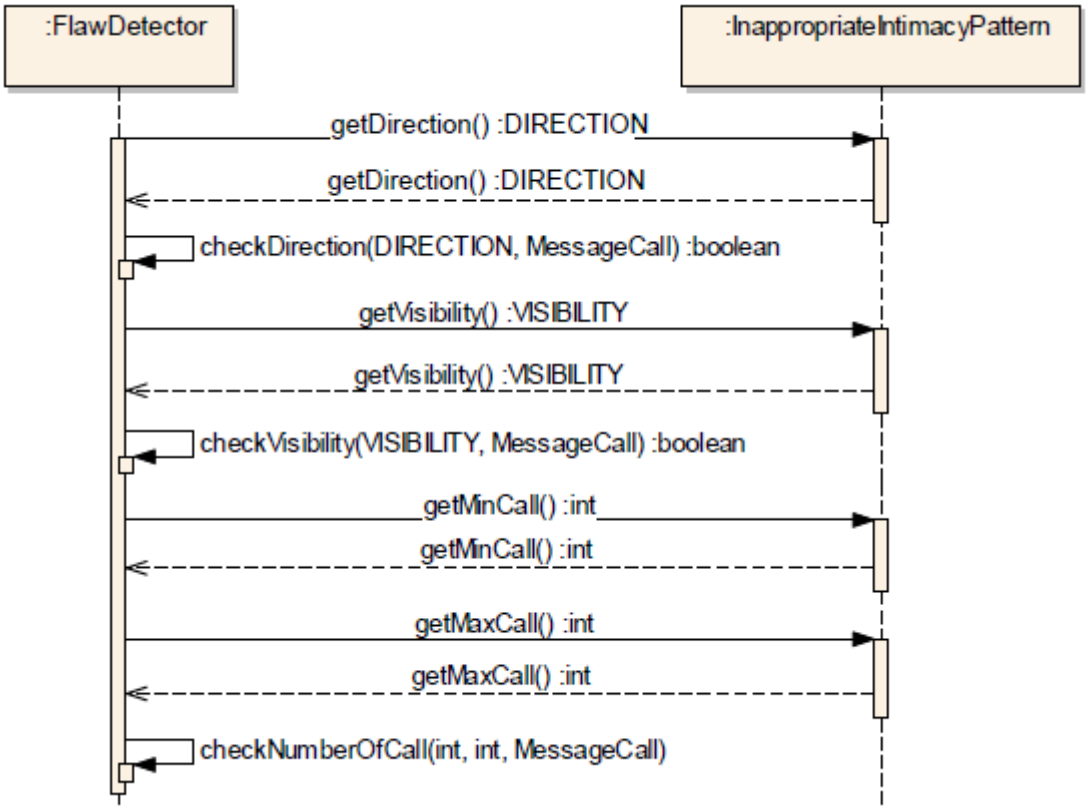
4.4.6 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Inappropriate Intimacy

แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Inappropriate Intimacy ดังรูปที่ 4.14 มีขั้นตอนการตรวจสอบ ดังนี้

(1) เริ่มจากการตรวจสอบทิศทางการเรียกใช้งานเมท็อดโดยจะกำหนดค่าแม่แบบไว้ให้ดูที่ความสัมพันธ์ที่มีทิศทางเข้าสู่โหนดที่กำลังตรวจสอบ

(2) ตรวจสอบค่าการเข้าถึงของเมท็อด โดยกำหนดแม่แบบให้ดูเมท็อดที่มีค่าการเข้าถึงเป็น Private

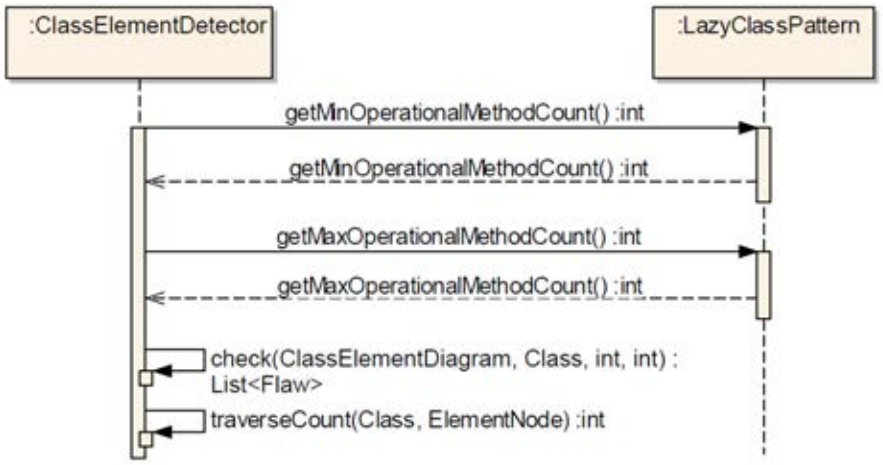
(3) ตรวจสอบการเรียกใช้งานเมท็อดดังกล่าว หากพบว่าเมท็อดมีการเรียกจากภายนอก จะถือว่าเป็นข้อบกพร่อง



รูปที่ 4.14 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Inappropriate Intimacy

4.4.7 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Lazy Class

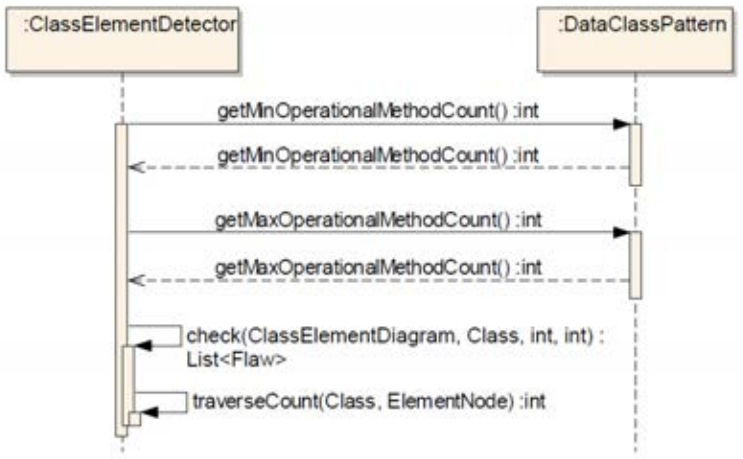
แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Lazy Class ดังรูปที่ 4.15 มีขั้นตอนการตรวจสอบ เริ่มจากการดูการกำหนดค่าจำนวนเมท็อดที่มากที่สุด และน้อยที่สุดที่แต่ละคลาสควรจะมีจากแม่แบบ และนำค่าดังกล่าว ตรวจสอบกับแต่ละโหนดในแผนภาพต้นไม้คุณลักษณะ



รูปที่ 4.15 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Lazy Class

4.4.8 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Data Class

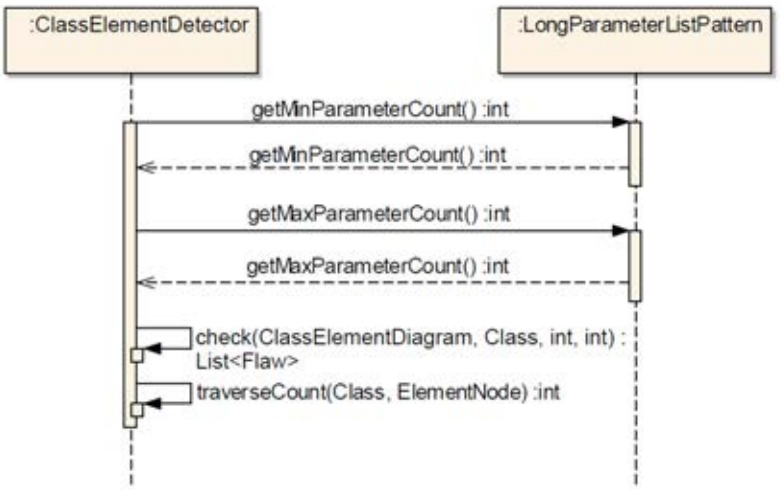
แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Data Class ดังรูปที่ 4.16 มีขั้นตอนเริ่มจากการดูการกำหนดค่าจำนวนเมท็อดที่มีการทำงาน (Operational Method) มากที่สุด และน้อยที่สุดที่แต่ละคลาสควรจะมีจากแม่แบบ และนำค่าดังกล่าว ตรวจสอบกับแต่ละโหนดในแผนภาพต้นไม้คุณลักษณะ



รูปที่ 4.16 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Data Class

4.4.9 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Long Parameter Lists

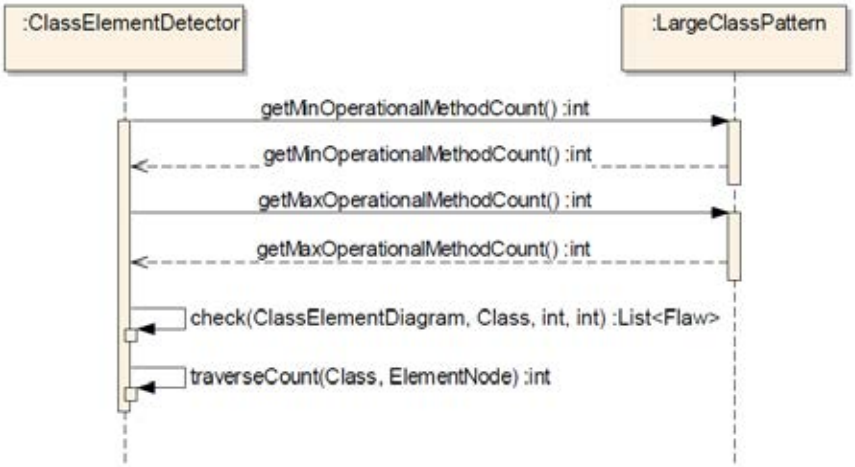
แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Long Parameter Lists ดังรูปที่ 4.17 มีขั้นตอนเริ่มจากการดูการกำหนดค่าจำนวนพารามิเตอร์ที่มากที่สุด และน้อยที่สุดที่แต่ละเมท็อดควรจะมีจากแม่แบบ และนำค่าดังกล่าว ตรวจสอบกับแต่ละโหนดของเมท็อดในแผนภาพต้นไม้คุณลักษณะ



รูปที่ 4.17 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Long Parameter Lists

4.4.10 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Large Class

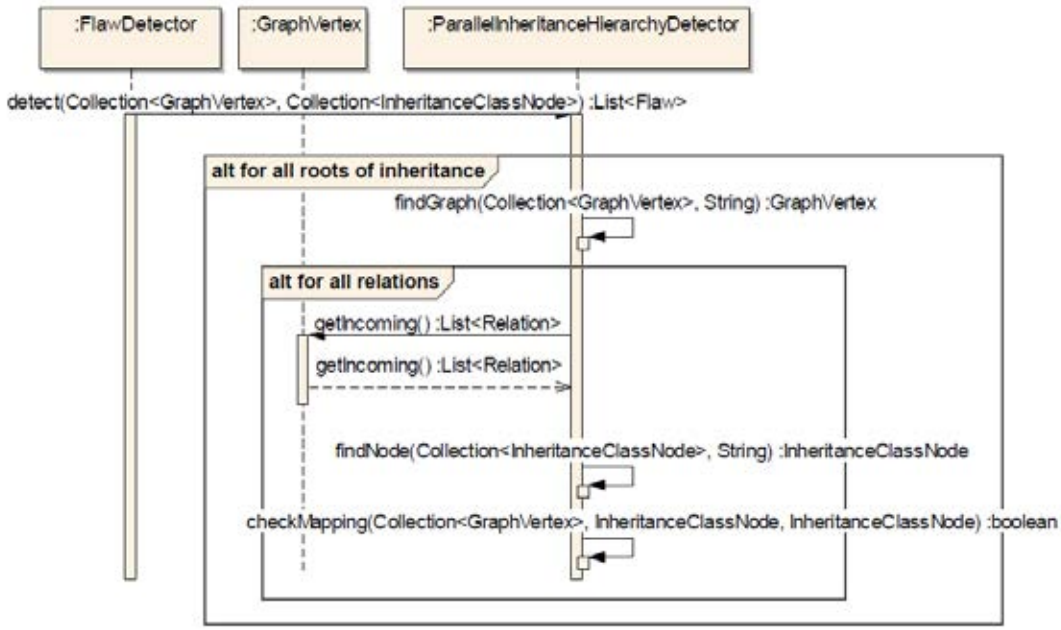
แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Large Class ดังรูปที่ 4.18 มีขั้นตอนเริ่มจากการดูการกำหนดค่าจำนวนเมท็อดที่มากที่สุด ที่แต่ละเมท็อดควรจะมีจากแม่แบบ และนำค่าดังกล่าว ตรวจสอบกับแต่ละโหนดของคลาสในแผนภาพต้นไม้คุณลักษณะ



รูปที่ 4.18 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Large Class

4.4.11 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Parallel Inheritance Hierarchy

ในการค้นหาข้อบกพร่องประเภท Parallel Inheritance Hierarchy จะมีขั้นตอนเริ่มจากการตรวจสอบความสัมพันธ์ที่โหนดรากของทุกๆการสืบทอดในแผนภาพต้นไม้การสืบทอด โดยในการตรวจสอบจะดูความสัมพันธ์กับแผนภาพกราฟตรรกะประกอบกัน เมื่อพบว่ามีความสัมพันธ์ไปยังรากของการสืบทอดใดๆ จะทำการตรวจสอบความสัมพันธ์ในโหนดลูกของการสืบทอดต่อ หากพบความสัมพันธ์ในลักษณะจับคู่กันของสองโครสร้างการสืบทอด จะถือว่าเป็นข้อบกพร่อง ตามแผนภาพซีเควนซ์ในรูปที่ 4.19



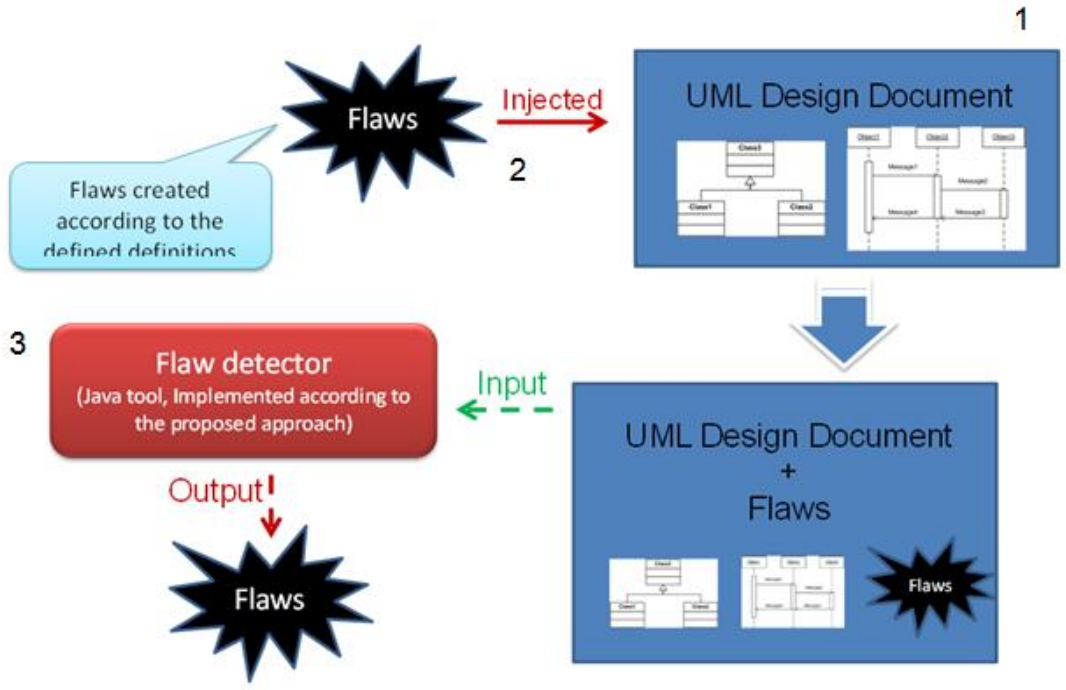
รูปที่ 4.19 แผนภาพซีเควนซ์ของการค้นหาข้อบกพร่องประเภท Parallel Inheritance Hierarchy

บทที่ 5

การทดสอบความสามารถของเครื่องมือ

ผู้วิจัยได้ทำการทดสอบความสามารถของเครื่องมือ ที่พัฒนามาจากวิธีการในการค้นหาข้อบกพร่อง โดยนำเครื่องมือที่พัฒนาตามวิธีการที่นำเสนอมาทำกรณีศึกษากับโมเดลการออกแบบซอฟต์แวร์ของระบบขนาดเล็ก ด้วยวิธีการจำลองจุดเสีย (Fault Injection) โดยจะทำการวิเคราะห์โมเดลการออกแบบ และจำลองข้อบกพร่องประเภทต่างๆในโมเดลการออกแบบ หลังจากนั้นจึงนำเครื่องมือที่พัฒนาตามวิธีการที่ได้นำเสนอมาทำการตรวจสอบกับโมเดลการออกแบบ

กระบวนการทดสอบความถูกต้องของเครื่องมือ แสดงตามรูปที่ 5.1 โดยแบ่งออกเป็น 3 ขั้นตอน ดังนี้ 1) การศึกษาและวิเคราะห์โมเดลการออกแบบที่จะนำมาทดสอบ 2) การจำลองจุดเสียหรือข้อบกพร่อง ในโมเดลการออกแบบ ในขั้นตอนนี้จะได้ผลลัพธ์เป็นโมเดลการออกแบบที่มีข้อบกพร่อง 3) นำโมเดลการออกแบบที่มีข้อบกพร่องไปทดสอบด้วยเครื่องมือที่พัฒนาขึ้นมา จะได้ผลลัพธ์เป็นรายการข้อบกพร่องที่เครื่องมือสามารถตรวจหาเจอ จากนั้นจึงนำผลการค้นหามาเปรียบเทียบกับค่าจริง



รูปที่ 5.1 แผนภาพแสดงกระบวนการทดสอบ

โมเดลการออกแบบระบบที่นำมาใช้ในการทดสอบเป็นโมเดลการออกแบบซอฟต์แวร์ขนาดเล็ก ประกอบไปด้วยแผนภาพยูเอ็มแอล ที่เป็นแผนภาพคลาส แสดงโครงสร้างคลาส ส่วนประกอบของแต่ละคลาส และการสืบทอดคลาส และแผนภาพซีเควนซ์ แสดงความสัมพันธ์และการทำงาน โมเดลที่นำมาใช้ในการทดสอบประกอบไปด้วย 1) โมเดลการออกแบบระบบถอนเงิน และฝากเงินอัตโนมัติ (Automated Teller Machine) 2) โมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ (Address Book) 3) โมเดลการออกแบบระบบควบคุมการทำงานของลิฟต์ (Elevator Control System) ตารางที่ 5.1 สรุปโมเดลการออกแบบระบบทั้ง 3 ระบบ

ตารางที่ 5.1 ตารางสรุปโมเดลการออกแบบระบบทดสอบทั้ง 3 ระบบ

โมเดลการออกแบบ	**Classes**	**Methods**	**Inheritance Group**	**Class Diagrams**	**Sequence Diagrams**
ระบบฝาก และถอนเงินอัตโนมัติ	22	74	1	1	4
ระบบบันทึกข้อมูลผู้ติดต่อ	9	31	-	1	10
ระบบควบคุมการทำงานของลิฟต์	19	54	2	1	11

5.1 การเตรียมข้อบกพร่องในโมเดลการออกแบบระบบสำหรับการทดสอบ

ในส่วนนี้จะอธิบายถึงการการออกแบบ และจำลองข้อบกพร่องในโมเดลการออกแบบระบบที่จะนำมาทดสอบ โดยเริ่มจากการศึกษาข้อกำหนดความต้องการของระบบ และทำการวิเคราะห์เพื่อจำลองข้อบกพร่องในโมเดลการออกแบบ

5.1.1 การเตรียมข้อบกพร่องในโมเดลการออกแบบระบบที่ 1

โมเดลการออกแบบซอฟต์แวร์ที่นำมาใช้ในกรณีศึกษาที่ 1 ได้มาจากตัวอย่างการวิเคราะห์และออกแบบระบบถอนเงิน และฝากเงินอัตโนมัติ หรือเรียกว่า ATM ซึ่งเป็นตัวอย่างในการออกแบบโปรแกรมเชิงวัตถุ โมเดลการออกแบบดังกล่าว เป็นโมเดลการออกแบบระบบขนาดเล็ก ประกอบไปด้วยข้อมูลการออกแบบเชิงลึกของแผนภาพคลาส และแผนภาพซีเควนซ์ ซึ่งถือเป็นความต้องการเบื้องต้นสำหรับวิธีการที่นำเสนอ

5.1.1.1 การศึกษาข้อกำหนดโมเดลการออกแบบระบบที่ 1

โมเดลการออกแบบระบบ ATM เป็นโมเดลการออกแบบซอฟต์แวร์เพื่อควบคุมการทำงานของเครื่อง ATM ประกอบไปด้วย ตัวอ่านแถบแม่เหล็กของบัตร ATM และ ส่วนที่ทำหน้าที่ติดต่อกับผู้ใช้งาน ได้แก่ Keyboard และ จอแสดงผล ช่องสำหรับใส่ และรับเงิน เครื่องพิมพ์สำหรับพิมพ์ใบเสร็จ และปุ่มสำหรับเริ่ม และหยุดการทำงานของเครื่อง ATM จะมีการทำการติดต่อกับระบบของธนาคาร (ส่วนของการติดต่อธนาคารไม่ได้รวมอยู่ในโมเดลการออกแบบ)

ATM จะให้บริการผู้ใช้งานทีละหนึ่งคน เริ่มจากการใส่บัตร ATM และกรอกรหัสผ่าน ข้อมูลทั้งสองส่วนจะถูกส่งไปยังธนาคารเพื่อยืนยันตัวตน หลังจากผ่านการยืนยันตัวตนแล้ว ผู้ใช้งานจึงจะสามารถทำรายการต่อได้ ระบบจะทำการเก็บบัตร ATM ไว้ตลอดการทำรายการ จนผู้ใช้งานยืนยันว่าไม่มีความประสงค์จะทำรายการใดๆต่อแล้ว

ATM มีความสามารถในการให้บริการแก่ผู้ใช้งาน ดังนี้

1. ผู้ใช้งานสามารถถอนเงินจากบัญชีที่ผูกอยู่กับบัตร ATM ได้ในจำนวนเงินที่เป็นจำนวนเท่าของ 20.00 ดอลลาร์โดยต้องได้รับการอนุมัติจากธนาคาร
2. ผู้ใช้งานสามารถทำการโอนเงินระหว่างบัญชีที่ผูกอยู่กับบัตร ATM ได้
3. ผู้ใช้งานสามารถตรวจสอบสถานะของบัญชีที่ผูกกอยู่กับบัตร ATM ได้
4. ผู้ใช้งานจะสามารถยกเลิกการทำรายการได้ด้วยการกดปุ่ม Cancel

เครื่อง ATM จะทำการติดต่อธนาคาร ในการทำรายการแต่ละครั้ง เพื่อขอรับข้อมูล และทำการอนุมัติ หากธนาคารพบว่ารหัสที่ผู้ใช้งานกรอกผิดพลาด ผู้ใช้งานจะต้องใส่รหัสผ่านใหม่ และรหัสผ่านผิดพลาดมากเกินกว่า 3 ครั้ง เครื่องจะยึดบัตร ATM ไว้ ผู้ใช้งานจะต้องติดต่อธนาคารเพื่อรับบัตรคืนด้วยตัวเอง หลังจากผู้ใช้งานทำรายการเสร็จแล้ว เครื่องจะพิมพ์ใบเสร็จสำหรับแต่ละรายการ ภายในใบเสร็จจะแสดงข้อมูล วันที่ เวลา สถานที่ ประเภทของการทำรายการ เลขที่บัญชี จำนวนเงินที่ทำรายการ และจำนวนเงินคงเหลือในบัญชี

ระบบมีการเก็บบันทึกการงานของระบบไว้เพื่อนำมาใช้ในการตรวจสอบ และแก้ไขปัญหาที่อาจเกิดขึ้น โมเดลการออกแบบระบบ ATM ประกอบไปด้วยคลาสทั้งหมด 22 คลาส และแผนภาพซีเควนซ์ทั้งหมด 4 แผนภาพ แผนภาพคลาสจะให้ข้อมูลเกี่ยวกับการสืบทอดคลาส ข้อมูล

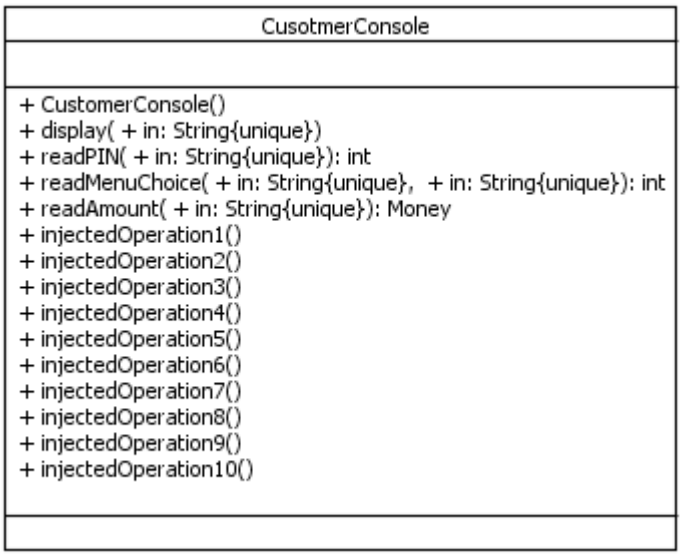
โครงสร้างของคลาส ในขณะที่แผนภาพซีเควนซ์ จะให้ข้อมูลเกี่ยวกับการออกแบบการทำงาน และการติดต่อกันระหว่างคลาส ตารางที่ 5.2 สรุปโมเดลการออกแบบระบบ ATM

ตารางที่ 5.2 ตารางสรุปโมเดลการออกแบบระบบ ATM

Classes	Methods	Inheritance Group	Class Diagrams	Sequence Diagrams
22	74	1	1	4

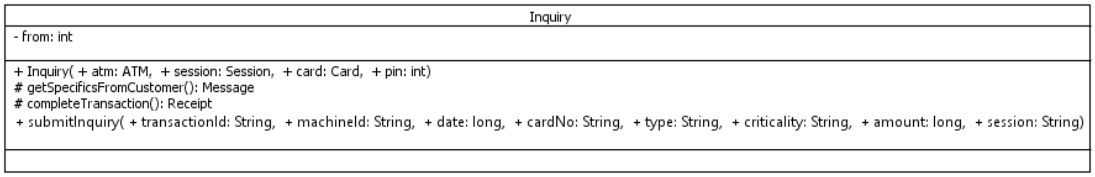
5.1.1.2 การจำลองข้อบกพร่องในโมเดลการออกแบบระบบที่ 1

ในส่วนนี้อธิบายรายละเอียดการจำลองข้อบกพร่องในโมเดลการออกแบบระบบที่ 1 การจำลองข้อบกพร่องประเภท Large Class ดังแสดงในรูปที่ 5.2 แสดงคลาส CustomerConsole โดยได้ทำการจำลองเมท็อดเพิ่มเข้าไป ทำให้คลาสดังกล่าวมีจำนวนเมท็อดรวมทั้งสิ้น 20 เมท็อด ซึ่งจัดว่าเป็นข้อบกพร่องประเภท Large Class



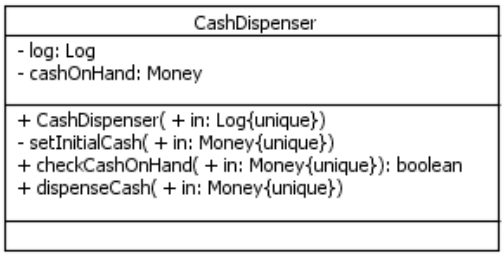
รูปที่ 5.2 การจำลองข้อบกพร่องประเภท Large Class คลาส CustomerConsole

ในการจำลองข้อบกพร่องประเภท Long Parameter Lists สำหรับโมเดลการออกแบบระบบ ATM ได้ทำการเพิ่มเมท็อด submitInquiry(…) เข้าไปที่คลาส Inquiry เพื่อทำการส่งคำร้องไปยังเครื่องแม่ข่าย ดังแสดงในรูปที่ 5.3 โดยเมท็อดนี้ถือว่าเป็นข้อบกพร่องประเภท Long Parameter Lists เนื่องจากมีจำนวนพารามิเตอร์ที่ส่งเข้าสู่เมท็อดจำนวน 11 ด้วยกัน

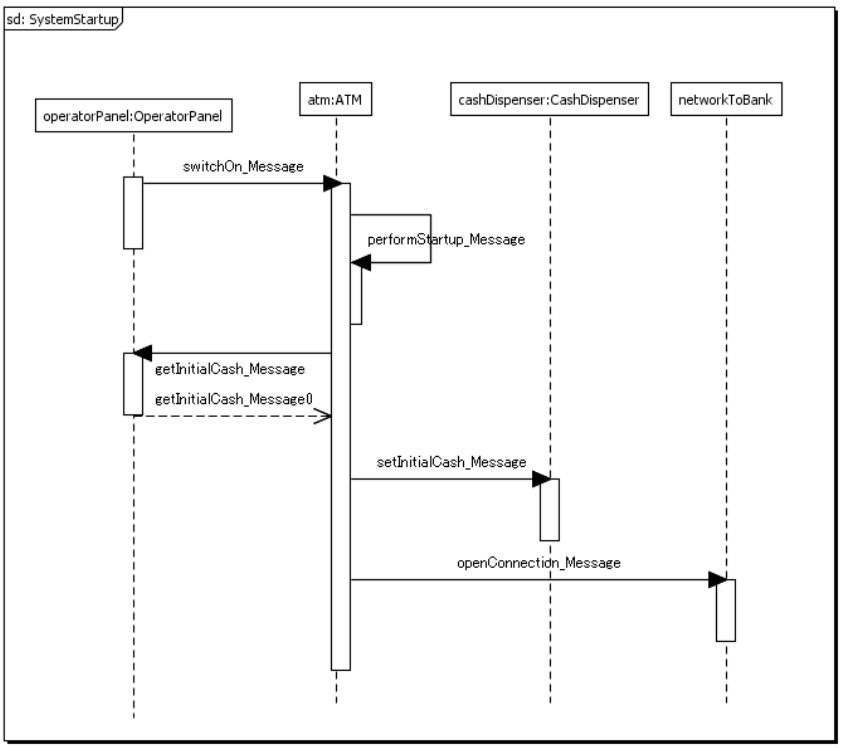


รูปที่ 5.3 การจำลองข้อบกพร่องประเภท Long Parameter Lists คลาส Inquiry

แผนภาพซีเควนซ์ในรูปที่ 5.5 แสดงการจำลองข้อบกพร่องประเภท Inappropriate Intimacy จากข้อกำหนดคลาส CashDispenser ในรูปที่ 5.4 จะเห็นได้ว่าเมท็อด setInitialCash(int Money) มีค่าการเข้าถึงได้เป็น private และจากแผนภาพซีเควนซ์แสดงให้เห็นว่า คลาส ATM ได้ทำการเรียกใช้งานเมท็อด setInitialCash(int Money) ที่ไม่สามารถเข้าถึงได้ ทำให้เกิดข้อบกพร่องประเภท Inappropriate Intimacy ขึ้น



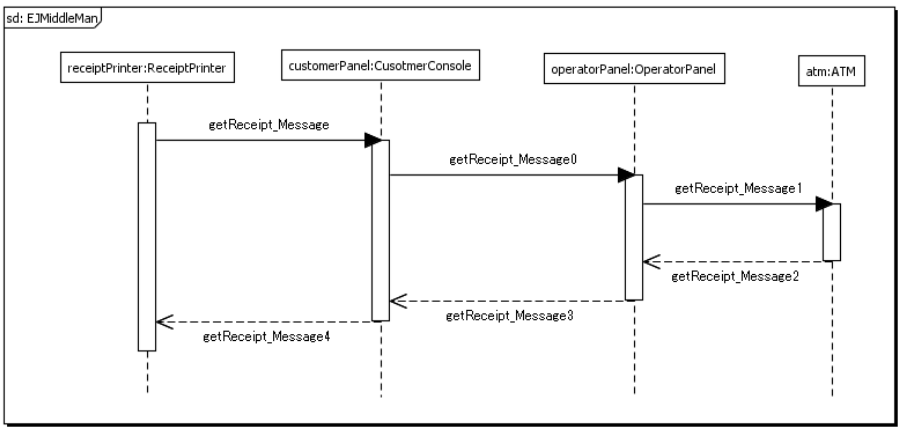
รูปที่ 5.4 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy คลาส CashDispenser



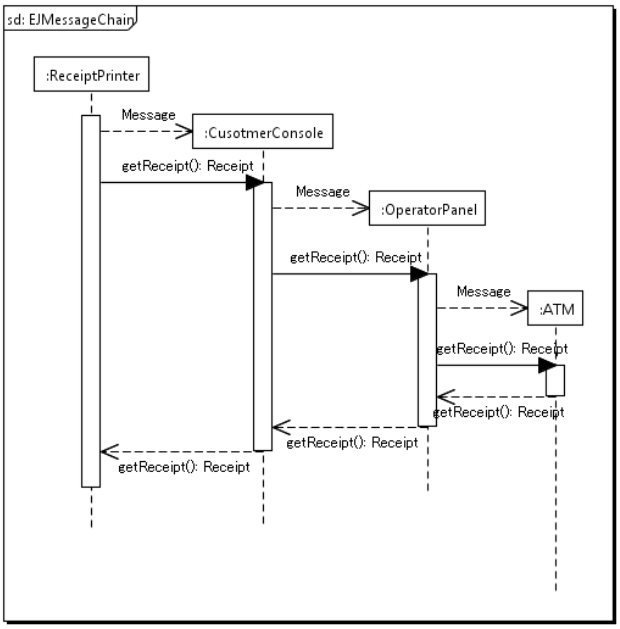
รูปที่ 5.5 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy แผนภาพซีเควนซ์ SystemStartup

รูปที่ 5.6 แสดงแผนภาพซีเควนซ์ของข้อบกพร่องประเภท Middle Man ในโมเดลการออกแบบที่ 1 โดยเป็นแผนภาพแสดงการเรียกดูใบเสร็จที่ได้จำลองเพิ่มเข้าไป เริ่มจากคลาส ReceiptPrinter มีการเรียกไปยังคลาส CustomerConsole และมีการเรียกไปยังคลาส OperatorPanel และ ATM ต่อตามลำดับ

รูปที่ 5.7 แสดงแผนภาพซีเควนซ์การออกแบบกระบวนการเรียกดูใบเสร็จอีกแบบหนึ่ง ซึ่งก็ยังถือเป็นข้อบกพร่องประเภท Message Chain อีกอยู่ดี เนื่องจากในการเรียกดูใบเสร็จ จากคลาส ReceiptPrinter มีการสร้างอ็อบเจ็ก และเรียกต่อกันเป็นลูกโซ่

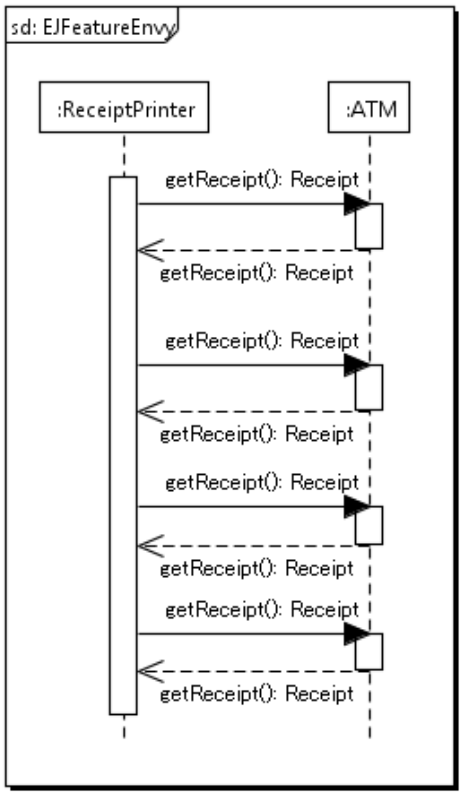


รูปที่ 5.6 แผนภาพซีเควนซ์การจำลองข้อบกพร่องประเภท Middle Man ในโมเดลการออกแบบที่ 1



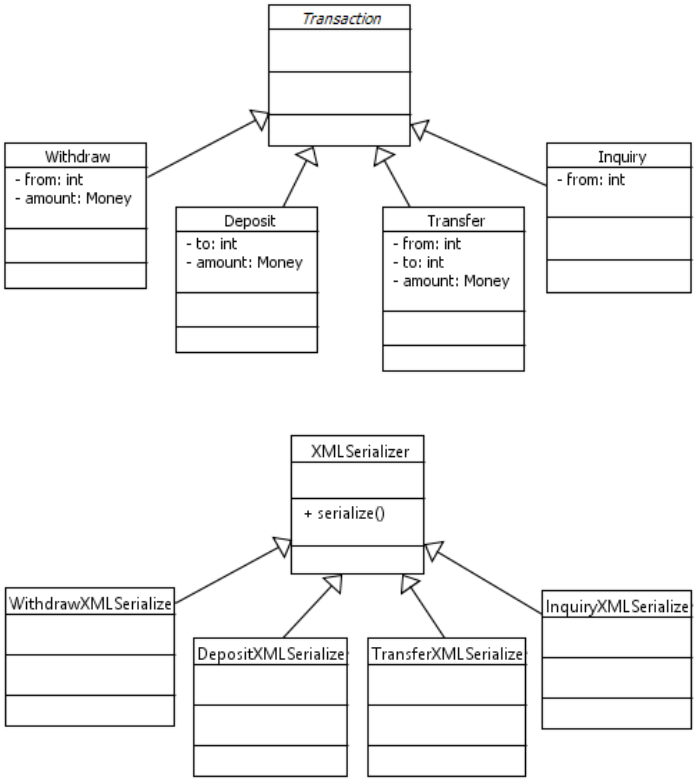
รูปที่ 5.7 แผนภาพซีเควนซ์การจำลองข้อบกพร่องประเภท Message Chain ในโมเดลการออกแบบที่ 1

แผนภาพซีเควนซ์ในรูปที่ 5.8 แสดงการจำลองข้อบกพร่องประเภท Feature Envy โดยเป็นการจำลองการเรียกเมท็อด getReceipt() จากภายนอก ทำให้มีการเรียกเมท็อดดังกล่าวจากภายนอกจำนวน 5 ครั้งด้วยกัน ในขณะที่เมท็อดไม่ได้ถูกเรียกจากภายในคลาสเองเลย จึงเกิดข้อบกพร่องขึ้น

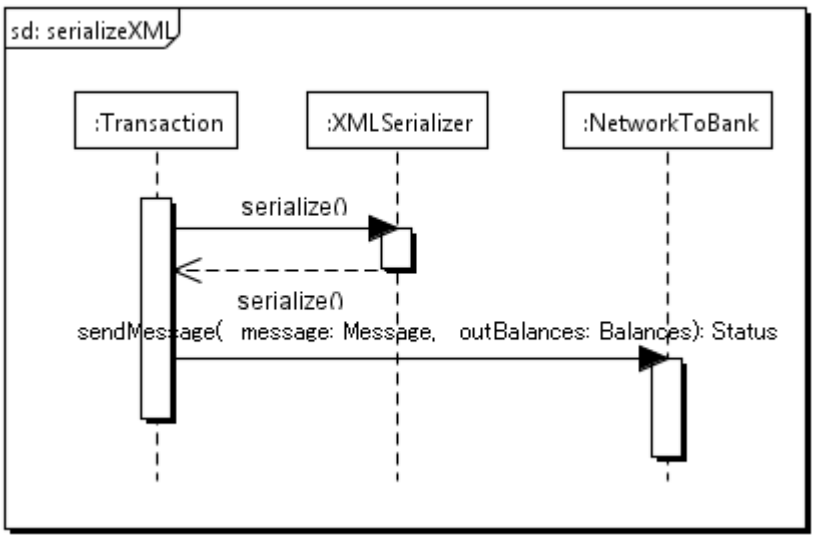


รูปที่ 5.8 แผนภาพซีเควนซ์การจำลองข้อบกพร่องประเภท Feature Envy ในโมเดลการออกแบบที่ 1

สำหรับข้อบกพร่องประเภท Parallel Inheritance Hierarchy ผู้วิจัยได้ทำการจำลองโครงสร้างการสืบทอดคลาส XMLSerializer เพิ่ม ตามรูปที่ 5.9 โดยคลาสดังกล่าวมีวัตถุประสงค์เพื่อซีเรียลไลซ์ (Serialize) เอกสารในรูปของเอ็กเอ็มแอล (XML) โดยมีความสัมพันธ์การเรียกเมท็อด serialize() ในลักษณะเป็นคู่อันดับกับโครงสร้างการสืบทอดคลาส Transaction เดิมที่มีอยู่แล้ว แผนภาพซีเควนซ์ในรูปที่ 5.10 แสดงตัวอย่างความสัมพันธ์ในระดับรากของการสืบทอด



รูปที่ 5.9 การจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy โครงสร้างการสืบทอดคลาส



รูปที่ 5.10 การจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy แผนภาพซีเควนซ์แสดงความสัมพันธ์ที่รากของการสืบทอด

ข้อบกพร่องในโมเดลการออกแบบระบบ ATM สามารถสรุปได้ตามตารางที่ 5.3 โดยโมเดลการออกแบบระบบ ATM จะมีข้อบกพร่องประเภท Data Class และ Lazy Class อยู่แล้ว และในโครงสร้างการสืบทอดคลาส Transaction นั้น จะมีเมท็อดที่มีการสืบทอด และไม่ได้มีการ

ออกแบบให้มีการใช้งานใดๆอยู่ ทำให้เกิดเป็นข้อบกพร่องประเภท Refused Bequest ที่คลาสลูกเหล่านั้นขึ้น

ตารางที่ 5.3 ตารางสรุปข้อบกพร่องในโมเดลการออกแบบระบบ ATM

เมท็อดที่เกิดข้อบกพร่อง	**คลาสที่เกิดข้อบกพร่อง**	**ประเภทของข้อบกพร่อง**
-	CustomerConsole	Large Class
-	AccountInformation	Long Parameter Lists
setInitialCash	CashDispenser	Inappropriate Intimacy
getReceipt	ReceiptPrinter	Middle Man
getReceipt	ATM	Feature Envy
checkBalance	Inquiry	Refused Bequest
checkBalance	Deposit	Refused Bequest
checkBalance	Transfer	Refused Bequest
checkBalance	Withdraw	Refused Bequest
-	XMLSerializer	Parallel Inheritance Hierarchy
CustomerConsole	ReceiptPrinter	Message Chain
-	AccountInformation	Data Class
-	InetAddress	Data Class
-	ReceiptPool	Data Class
-	Balance	Lazy Class
-	Message	Lazy Class
-	Receipt	Lazy Class
-	EnvelopeAccepter	Lazy Class
-	OperatorPanel	Lazy Class
-	ReceiptPrinter	Lazy Class
-	Session	Lazy Class
-	AccountInformation	Lazy Class

ตารางที่ 5.3 ตารางสรุปข้อบกพร่องในโมเดลการออกแบบระบบ ATM (ต่อ)

เมท็อดที่เกิดข้อบกพร่อง	คลาสที่เกิดข้อบกพร่อง	ประเภทของข้อบกพร่อง
-	Card	Lazy Class
-	InetAddress	Lazy Class
-	ReceiptPool	Lazy Class
-	XMLSerializer	Lazy Class
-	WithdrawXMLSerializer	Lazy Class
-	DespositXMLSerializer	Lazy Class
-	TransferXMLSerializer	Lazy Class
-	InquiryXMLSerializer	Lazy Class

5.1.2 การเตรียมข้อบกพร่องในโมเดลการออกแบบระบบที่ 2

โมเดลการออกแบบซอฟต์แวร์ที่นำมาใช้ในกรณีศึกษาที่ 2 ได้มาจากตัวอย่างการวิเคราะห์และออกแบบระบบบันทึกข้อมูลผู้ติดต่อ ซึ่งเป็นตัวอย่างในการออกแบบโปรแกรมเชิงวัตถุเช่นเดียวกับระบบ ATM ที่ใช้ในกรณีศึกษาที่ 1

5.1.2.1 การศึกษาข้อกำหนดโมเดลการออกแบบระบบที่ 2

ระบบบันทึกข้อมูลผู้ติดต่อเป็นระบบที่ออกแบบมาเพื่อจัดเก็บข้อมูลบุคคล ที่อยู่ และหมายเลขโทรศัพท์ ผู้ใช้งานสามารถที่จะทำการเพิ่มข้อมูลผู้ติดต่อ แก้ไขข้อมูลเดิม (ยกเว้นชื่อ) และลบข้อมูลที่ไม่ต้องการออก ในการเรียกดูข้อมูล ผู้ใช้งานสามารถเลือกจัดเรียงตามนามสกุล หรือหมายเลขไปรษณีย์ และสามารถสั่งพิมพ์ได้

ผู้ใช้งานสามารถบันทึกไฟล์ เปิดไฟล์ ข้อมูลผู้ติดต่อ และสามารถสร้างไฟล์ใหม่ได้ผ่านเมนู บันทึก เปิด สร้าง และสามารถปิดไฟล์ที่ไม่ต้องการใช้ได้ผ่านเมนู ปิด ผู้ใช้งานสามารถสั่ง บันทึกเป็น... เพื่อบันทึกไฟล์ไปยังที่อยู่ของไฟล์ใหม่ได้ ในการใช้งานระบบ ผู้ใช้งานสามารถใช้งานไฟล์ได้ทีละไฟล์เท่านั้น หากมีการสร้าง หรือเปิดไฟล์ใหม่ขึ้นมา ไฟล์ปัจจุบันจะถูกปิด ระบบจะเก็บการเปลี่ยนแปลงที่เกิดขึ้นหลังจากการบันทึกครั้งล่าสุดไว้ เพื่อให้ผู้ใช้งานมีสิทธิ์ในการบันทึกการแก้ไขก่อนที่จะทำการปิดไฟล์

โมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ ประกอบไปด้วยคลาสทั้งหมด 9 คลาส และแผนภาพซีเควนซ์ทั้งหมด 10 แผนภาพ แผนภาพคลาสจะให้ข้อมูลเกี่ยวกับการสืบทอดคลาส ข้อมูลโครงสร้างของคลาส ในขณะที่แผนภาพซีเควนซ์ จะให้ข้อมูลเกี่ยวกับการออกแบบการทำงาน และการติดต่อกันระหว่างคลาส ตารางที่ 5.4 สรุปโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ

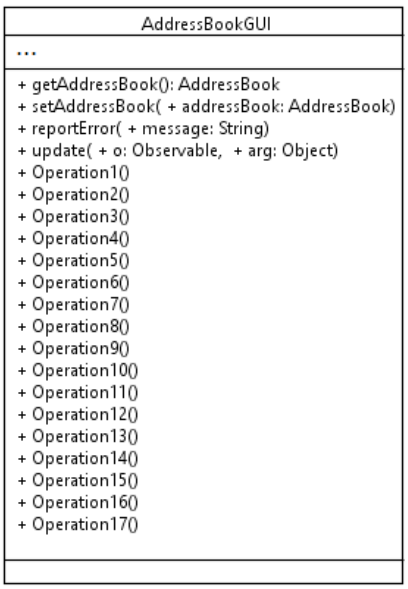
ตารางที่ 5.4 ตารางสรุปโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ

Classes	**Methods**	**Inheritance Group**	**Class Diagrams**	**Sequence Diagrams**
9	31	-	1	10

5.1.2.2 การจำลองข้อบกพร่องในโมเดลการออกแบบระบบที่ 2

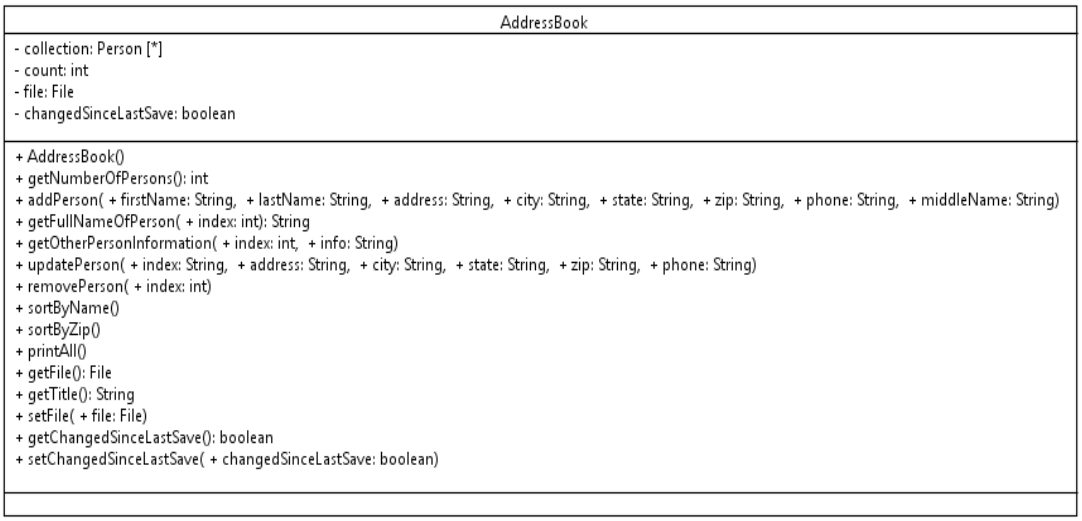
ในการทดสอบโมเดลการออกแบบที่ 1 ได้ทำการจำลองข้อบกพร่องในโมเดลการออกแบบดังนี้

ในการจำลองข้อบกพร่องประเภท Large Class ได้เพิ่มเมท็อดเข้าไปที่คลาส AddressBookGUI ดังรูปที่ 5.11 ทำให้คลาสดังกล่าวมีจำนวนเมท็อดรวมทั้งสิ้น 21 เมท็อดซึ่งถือเป็นข้อบกพร่องประเภท Large Class



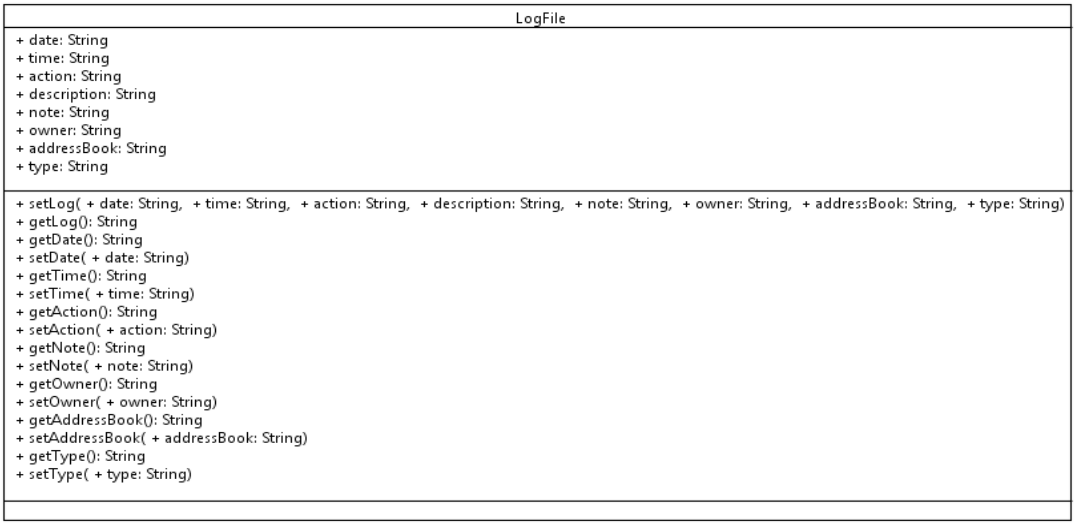
รูปที่ 5.11 การจำลองข้อบกพร่องประเภท Large Class คลาส AddressBookGUI

รูปที่ 5.12 แสดงคลาส AddressBook ในการทดสอบได้ทำการจำลองพารามิเตอร์ middleName เข้าไปในเมท็อด addPerson() หลังจากที่ได้ทำการเพิ่มพารามิเตอร์แล้ว ทำให้เมท็อดดังกล่าวมีจำนวนพารามิเตอร์รวมกัน 11 ซึ่งถือว่าเป็นข้อบกพร่องประเภท Long Parameter List ที่ได้กำหนดจำนวนพารามิเตอร์มากที่สุดไว้ที่ 7



รูปที่ 5.12 การจำลองข้อบกพร่องประเภท Long Parameter List คลาส AddressBook

รูปที่ 5.13 แสดงการจำลองข้อบกพร่องประเภท Data Class โดยตามนิยามแล้ว ข้อบกพร่องประเภท Data Class เกิดขึ้นกับคลาสที่มีเพียง Access Method เท่านั้น โดยไม่มี function การทำงานอื่นๆ สำหรับงานวิจัยนี้ได้จัดประเภทของเมท็อดเป็น Operational และ Non-Operation (Access Method) จากชื่อของเมท็อด โดยจะถือว่า Getter และ Setter เป็น Non-Operational Method เมื่อพบว่าชื่อของเมท็อดขึ้นต้องด้วย get- หรือ set- จากรูปในการทดสอบได้ทำการจำลองข้อบกพร่องประเภท Data Class โดยทำการเพิ่มคลาส LogFile เข้ามา ทำหน้าที่เป็นโมเดลในการเก็บ Log การทำงาน โดยจะเห็นว่าคลาส LogFile จะมีเพียง Access Method เท่านั้น ซึ่งถือว่าเป็นข้อบกพร่องประเภท Data Class และยังมีเมท็อด setLog() ที่ประกอบไปด้วยพารามิเตอร์จำนวน 8 ซึ่งถือว่าเป็นข้อบกพร่องประเภท Long Parameter List

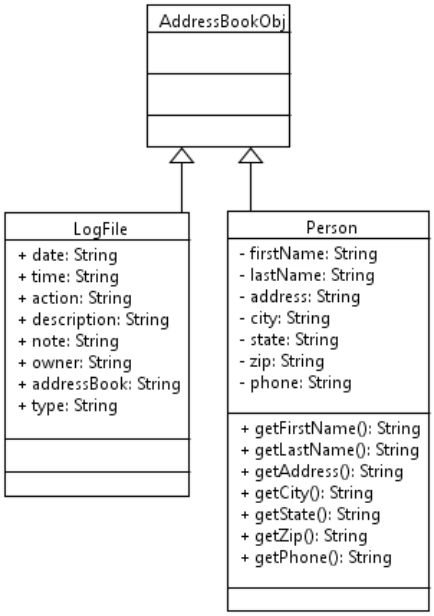


รูปที่ 5.13 การจำลองข้อบกพร่องประเภท Data Class คลาส LogFile

ในการจำลองข้อบกพร่องประเภท Refused Bequest นั้น ได้ทำการจำลองให้คลาส LogLibrary และ PersonLibrary สืบทอดเมท็อด registerLibrary() มาจากคลาส Library แต่เมท็อดที่สืบทอดมาดังกล่าวนั้นจะไม่มีการใช้งานใดๆ ทำให้เกิดเป็นข้อบกพร่องขึ้น ดังรูปที่ 5.14

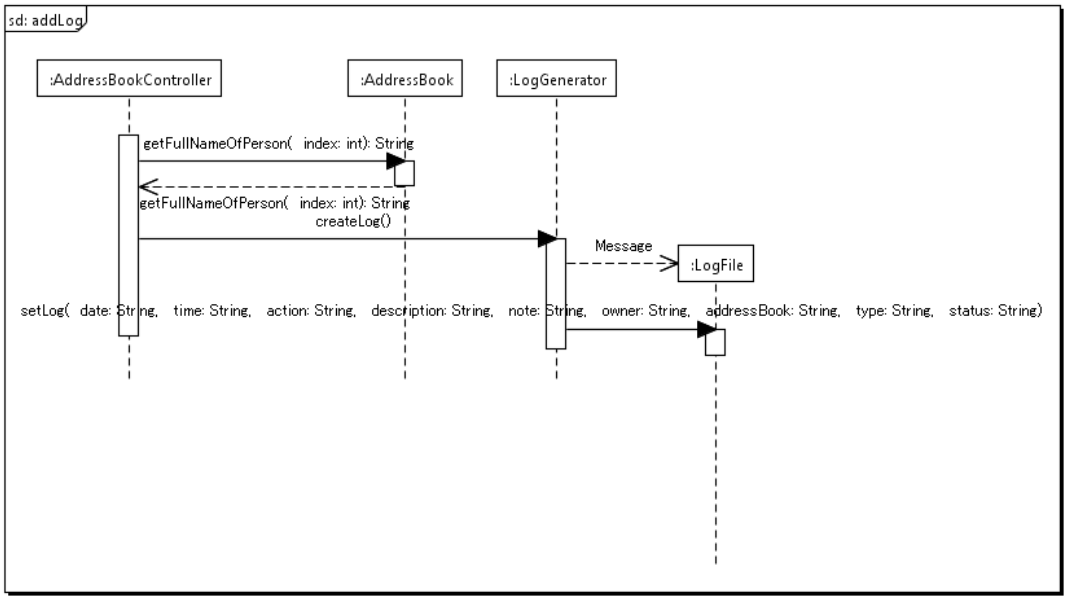
รูปที่ 5.14 การจำลองข้อบกพร่องประเภท Refused Bequest การสืบทอดคลาส Library

สำหรับโครงสร้างการสืบทอดคลาส Library นั้น ยังจำลองเป็นข้อบอกร่องประเภท Parallel Inheritance Hierarchy อีกด้วย โดยได้กำหนดให้คลาส LogFile และ Person ได้สืบทอดคลาส AddressBookObj ดังแสดงในรูปที่ 5.15 โดยทั้งสองโครงสร้างนี้ จะมีความสัมพันธ์การเรียกเมท็อด clearLibrary แบบจับคู่กัน

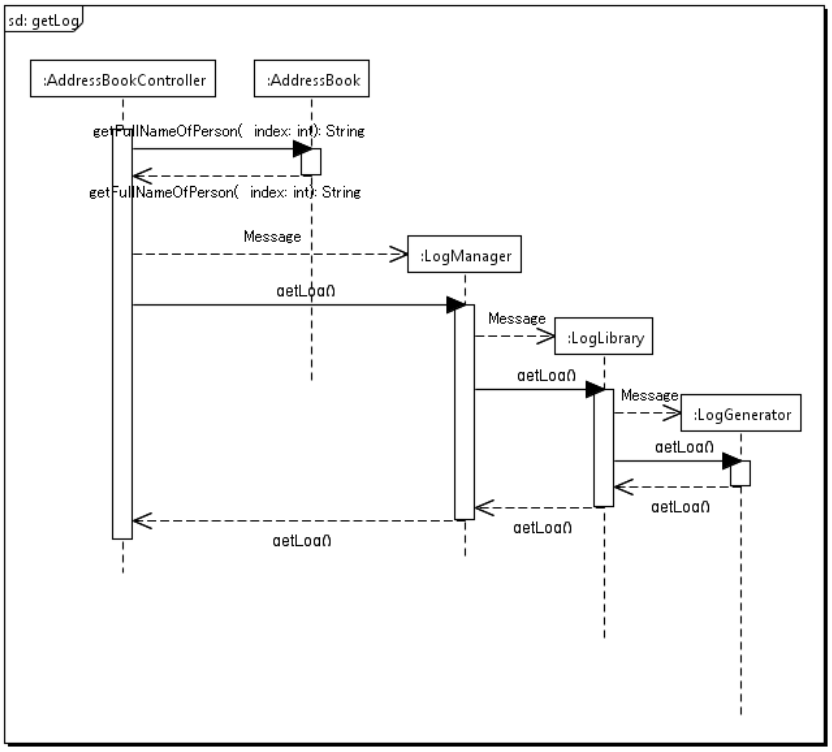


รูปที่ 5.15 การจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy โครงสร้างการสืบทอดคลาส AddressBookObj

สำหรับการทดสอบข้อบกพร่องประเภท Feature Envy ได้ทำการจำลองแผนภาพซีเควนซ์เพิ่มในส่วนของการบันทึก Log ตามรูปที่ 5.16 และ การเรียกดู Log ไฟล์ ตามรูปที่ 5.17 เข้าไป ทำให้ เมื่อรวมกับแผนภาพซีเควนซ์เดิมของระบบแล้ว เกิดข้อบกพร่องประเภท Feature Envy ที่การเรียกใช้งานเมท็อด getFullNameOfPerson() ที่คลาส AddressBook โดยจะมีการเรียกใช้งานเมท็อดจากคลาส AddressBookController เป็นจำนวน 6 ครั้งด้วยกัน ในขณะที่ไม่มีการเรียกใช้งานใดๆจากภายในคลาสเลย ซึ่งผู้ออกแบบระบบ ควรทำการวิเคราะห์การออกแบบใหม่ เพื่อพิจารณาย้ายเมท็อด getFullNameOfPerson() ไปยังคลาส AddressBookController



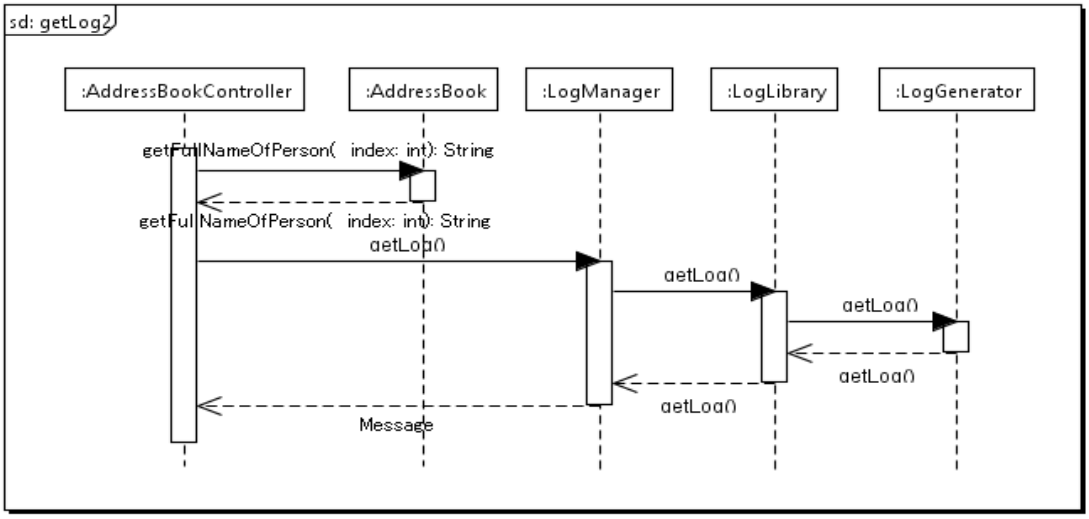
รูปที่ 5.16 การจำลองข้อบกพร่องประเภท Feature Envy แผนภาพซีเควนซ์ AddLog



รูปที่ 5.17 การจำลองข้อบกพร่องประเภท Feature Envy และ Message Chain แผนภาพซีเควนซ์ getLog

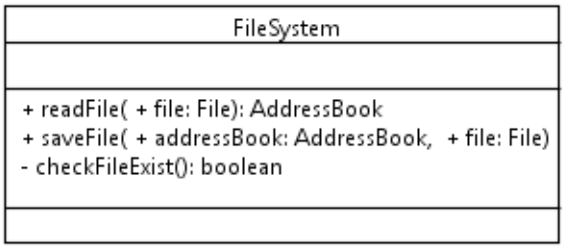
และนอกไปจากนั้น แผนภาพซีเควนซ์ในรูปที่ 5.17 ยังทำให้เกิดข้อบกพร่องประเภท Message Chain อีกด้วย เนื่องจากในขั้นตอนการเรียกดูบันทึกได้เกิดการเรียกต่อกันเป็นลูกโซ่ขึ้น ในการทดสอบนี้ได้จำลองแผนภาพซีเควนซ์กระบวนการในการดูบันทึกอีกแบบหนึ่งตามแผนภาพ

ซีเควนซ์ในรูปที่ 5.18 ซึ่งถือว่าเป็นข้อบกพร่องประเภท Middle Man เนื่องจากมีการเรียกผ่านตัวกลางเกิดขึ้น

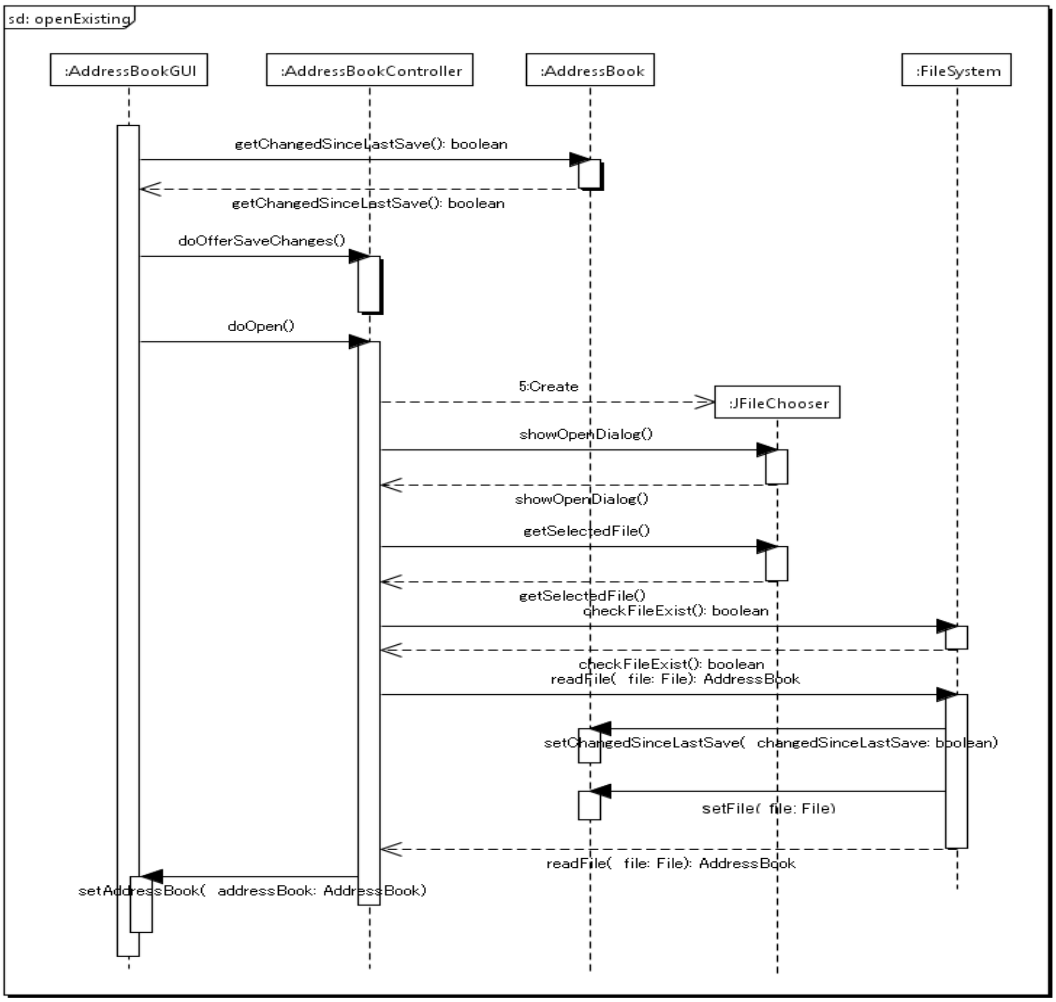


รูปที่ 5.18 การจำลองข้อบกพร่องประเภท Middle Man แผนภาพซีเควนซ์ getLog2

สำหรับการจำลองข้อบกพร่องประเภท Inappropriate Intimacy ได้ทำการเพิ่มเมท็อดเพื่อตรวจสอบว่าไฟล์ข้อมูลผู้ติดต่อที่ต้องการจะเปิดนั้นมีอยู่จริงหรือไม่ที่คลาส FileSystem โดยมีค่าการเข้าถึงเป็น Private ตามรูปที่ 5.19 โดยเมท็อดดังกล่าวมีการเรียกจากคลาส AddressBookController ตามแผนภาพซีเควนซ์ในรูปที่ 5.20 ทำให้เกิดข้อบกพร่องขึ้น



รูปที่ 5.19 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy คลาส FileSystem



รูปที่ 5.20 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy แผนภาพซีเควนซ์ openExisting

การจำลองข้อบกพร่องในโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ สามารถสรุปได้ตามตารางที่ 5.5 โดยในโมเดลการออกแบบจะมีข้อบกพร่องประเภท Lazy Class อยู่แล้ว

ตารางที่ 5.5 ตารางสรุปการจำลองข้อบกพร่องในโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ

เมท็อดที่เกิดข้อบกพร่อง	คลาสที่เกิดข้อบกพร่อง	ประเภทของข้อบกพร่อง
addPerson	AddressBook	Long Parameter List
setLog	LogFile	Long Parameter List
getAddressBook	AddressBookController	Feature Envy
getLog	AddressBookController	Message Chain
-	LogFile	Data Class
-	LogManager	Data Class

ตารางที่ 5.5 ตารางสรุปการจำลองข้อบกพร่องในโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ (ต่อ)

เมท็อดที่เกิดข้อบกพร่อง	คลาสที่เกิดข้อบกพร่อง	ประเภทของข้อบกพร่อง
-	AddressBookObj	Data Class
registerLibrary	LogLibrary	Refused Bequest
registerLibrary	PersonLibrary	Refused Bequest
checkFileExist	AddressBookController	Inappropriate Intimacy
-	AddressBookGUI	Large Class
getLog	AddressBookController	Middle Man
-	Library	Parallel Inheritance Hierarchy
-	Person	Lazy Class
-	AddressBookApplication	Lazy Class
-	MultiInputPane	Lazy Class
-	LogFile	Lazy Class
-	LogGenerator	Lazy Class
-	LogManager	Lazy Class
-	LogLibrary	Lazy Class
-	Library	Lazy Class
-	PersonLibrary	Lazy Class
-	AddressBookObj	Lazy Class

5.1.3 การเตรียมข้อบกพร่องในโมเดลการออกแบบระบบที่ 3

โมเดลการออกแบบซอฟต์แวร์ที่นำมาใช้ในกรณีศึกษาที่ 3 ได้มาจากโมเดลการระบบควบคุมลิฟต์ ซึ่งเป็นระบบ Embedded System ที่ออกแบบด้วยหลักการของ Object Oriented Analysis and Design เอกสารการออกแบบประกอบไปด้วยแผนภาพยูเอ็มแอล และคำอธิบายการออกแบบระบบ โดยจะทำงานเพื่อตอบสนองกับผู้ใช้งาน โดยจะมีหน้าที่ในการควบคุมการทำงานของลิฟต์ รวมไปถึงระบบไฟ และการจัดเก็บข้อมูลการใช้งานต่างๆ

5.1.3.1 การศึกษาข้อกำหนดโมเดลการออกแบบระบบที่ 3

โมเดลการออกแบบซอฟต์แวร์ที่นำมาใช้ในกรณีศึกษาที่ 3 ได้มาจากโมเดลการระบบควบคุมลิฟต์ ซึ่งเป็นระบบ Embedded System ที่ออกแบบด้วยหลักการของ Object Oriented Analysis and Design เอกสารการออกแบบประกอบไปด้วยแผนภาพยูเอ็มแอล และคำอธิบายการออกแบบระบบ โดยจะทำงานเพื่อตอบสนองกับผู้ใช้งาน โดยจะมีหน้าที่ในการควบคุมการทำงานของลิฟต์ รวมไปถึงระบบไฟ และการจัดเก็บข้อมูลการใช้งานต่างๆ

โมเดลการออกแบบระบบควบคุมการทำงานของลิฟต์ ประกอบไปด้วยคลาสที่เกี่ยวข้องทั้งหมด 19 คลาส และแผนภาพซีเควนซ์ทั้งหมด 11 แผนภาพด้วยกัน แผนภาพคลาสจะให้ข้อมูลเกี่ยวกับการสืบทอดคลาส ข้อมูลโครงสร้างของคลาส ในขณะที่แผนภาพซีเควนซ์ จะให้ข้อมูลเกี่ยวกับการออกแบบการทำงาน และการติดต่อกันระหว่างคลาส ตารางที่ 5.6 สรุปโมเดลการออกแบบระบบควบคุมการทำงานของลิฟต์

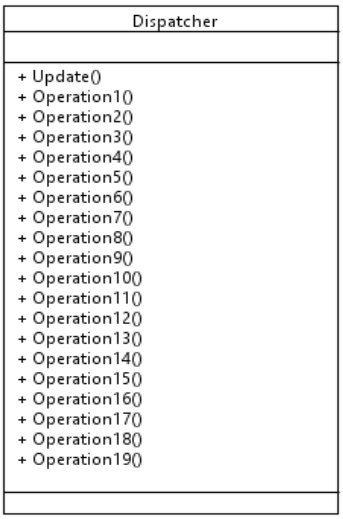
ตารางที่ 5.6 ตารางสรุปโมเดลการออกแบบระบบบันทึกข้อมูลผู้ติดต่อ

Classes	**Methods**	**Inheritance Group**	**Class Diagrams**	**Sequence Diagrams**
19	54	2	1	11

5.2.3.2 การจำลองข้อบกพร่องในโมเดลการออกแบบระบบที่ 3

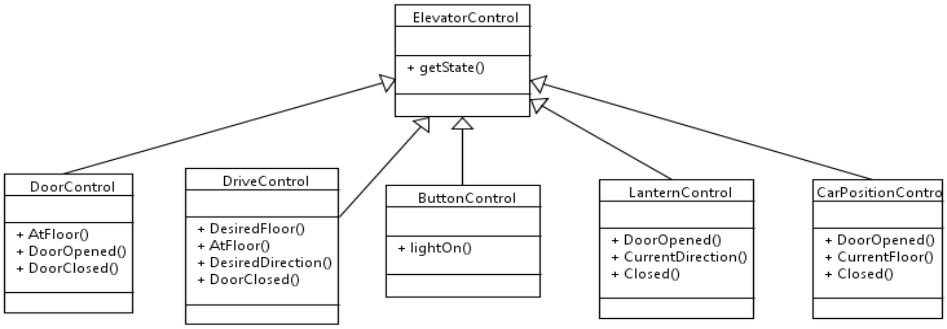
ในการทดสอบโมเดลการออกแบบที่ 3 ได้มีการจำลองข้อบกพร่องในโมเดลการออกแบบไว้ ดังนี้

รูปที่ 5.21 แสดงการจำลองข้อบกพร่องประเภท Large Class โดยการจำลองเมท็อดเพิ่มเข้าไปที่คลาส Dispatcher ทำให้คลาสดังกล่าวมีจำนวนเมท็อดรวม 20 ซึ่งถือว่าเป็นข้อบกพร่อง

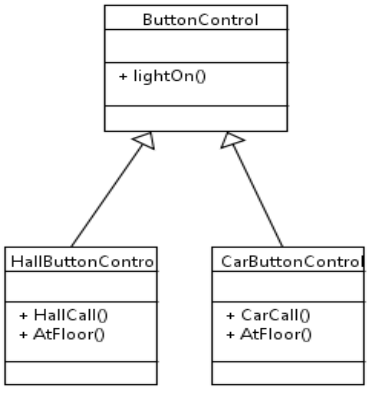


รูปที่ 5.21 การจำลองข้อบกพร่องประเภท Large Class คลาส Dispatcher

รูปที่ 5.22 และ รูปที่ 5.23 แสดงการจำลองข้อบกพร่องประเภท Refused Bequest โดยได้ทำการจำลองเมท็อด getState() เข้าไปที่คลาส ElevatorControl และเมท็อด lightOn() เข้าไปที่คลาส ButtonControl เมื่อเมท็อดดังกล่าวได้รับการสืบทอดไปยังคลาสลูกแล้ว ไม่เกิดการเรียกใช้งานใดๆ ทำให้เข้าข่ายข้อบกพร่องประเภท Refused Bequest

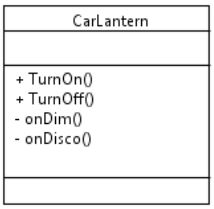


รูปที่ 5.22 การจำลองข้อบกพร่องประเภท Refused Bequest การสืบทอดคลาส ElevatorControl

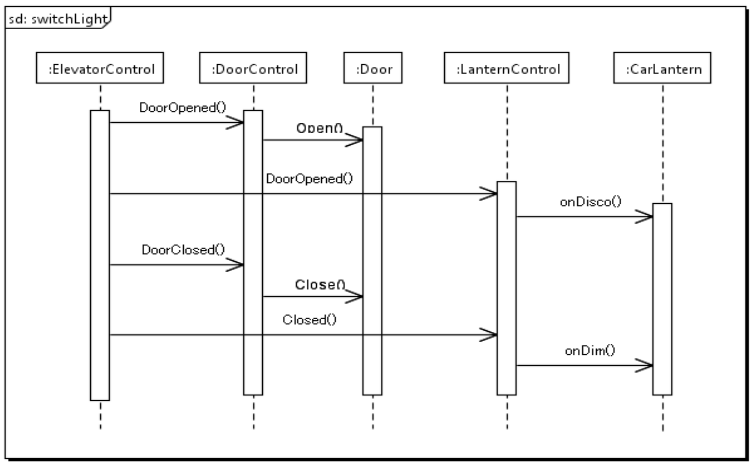


รูปที่ 5.23 การจำลองข้อบกพร่องประเภท Refused Bequest การสืบทอดคลาส ElevatorControl

รูปที่ 5.24 และ 5.25 แสดงการจำลองข้อบกพร่องประเภท Inappropriate Intimacy โดยได้ทำการเพิ่มเมท็อด onDim() และ onDisco() เข้าไปที่คลาส CarLantern เพื่อควบคุมรูปแบบการเปิดไฟ โดยมีประเภทการเข้าถึงเป็น private แผนภาพซีเควนซ์ในรูปที่ 5.24 แสดงการควบคุมไฟ เมื่อมีการเปิด และปิดประตูลิฟท์ โดยมีการเรียกเมท็อด onDim() และ onDisco() จากคลาส LanternControl ซึ่งถือเป็นข้อบกพร่องประเภท Refused Bequest เนื่องจากเมท็อดดังกล่าวมีค่าการเข้าถึงได้เป็น private ทำให้ไม่สามารถเข้าถึงได้จากภายนอกคลาส

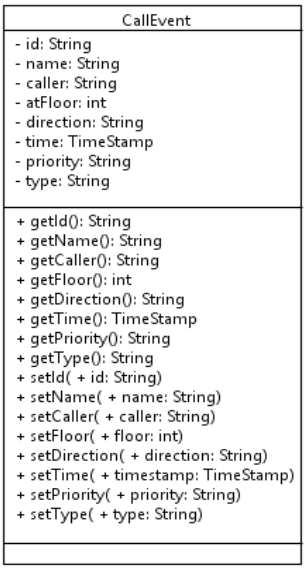


รูปที่ 5.24 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy คลาส CarLantern



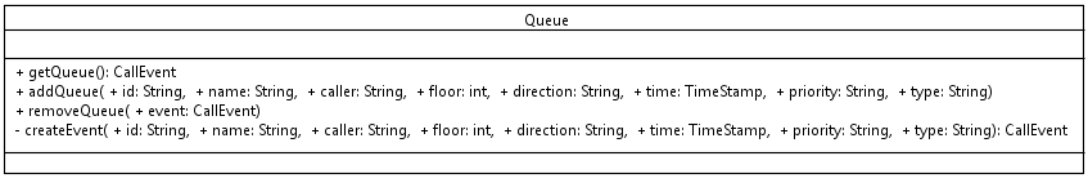
รูปที่ 5.25 การจำลองข้อบกพร่องประเภท Inappropriate Intimacy แผนภาพซีเควนซ์ switchLight

รูปที่ 5.26 แสดงคลาส CallEvent ที่ทำการจำลองขึ้นมา โดยคลาสดังกล่าวมีหน้าที่ในการเก็บข้อมูลการเรียกลิฟต์ โดยจะมีเพียง Access Method เท่านั้น ไม่มีฟังก์ชันการทำงานอื่น จัดว่าเป็นข้อบกพร่องประเภท Data Class



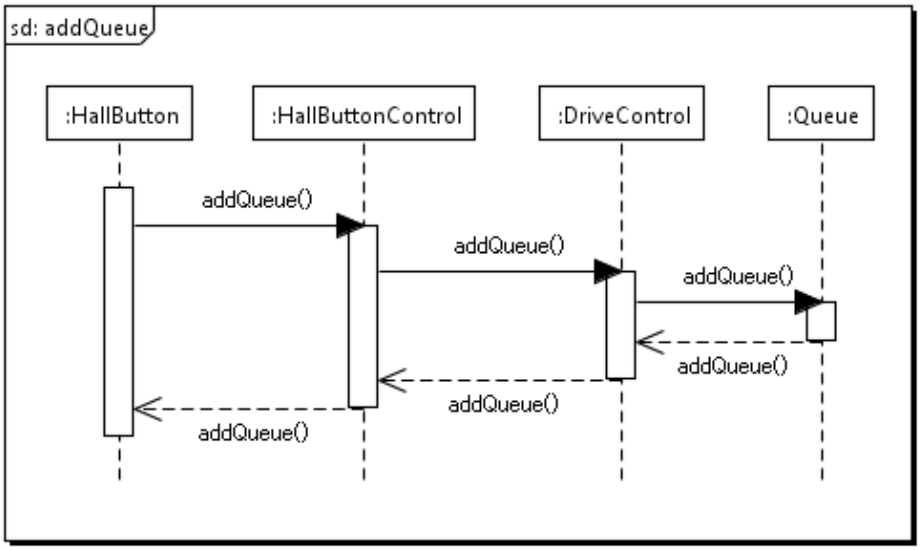
รูปที่ 5.26 การจำลองข้อบกพร่องประเภท Data Class คลาส CallEvent

รูปที่ 5.27 แสดงการจำลองคลาส Queue ขึ้นมา เพื่อจัดลำดับการเรียกของลิฟต์ เมื่อมีการเรียกลิฟต์เกิดขึ้นพร้อมกัน โดยการเรียกแต่ละครั้งจะถูกเก็บอยู่ในรูปของ CallEvent โดย Queue จะมีเมท็อดที่ทำหน้าที่ในการเพิ่ม ลบ และเรียกดูรายการ CallEvent ภายในคลาส Queue จะมีเมท็อด addQueue() และ createEvent() ที่รับพารามิเตอร์เป็นจำนวนที่เกินกว่าที่กำหนดไว้ ทำให้เกิดข้อบกพร่องประเภท Long Parameter List ขึ้น

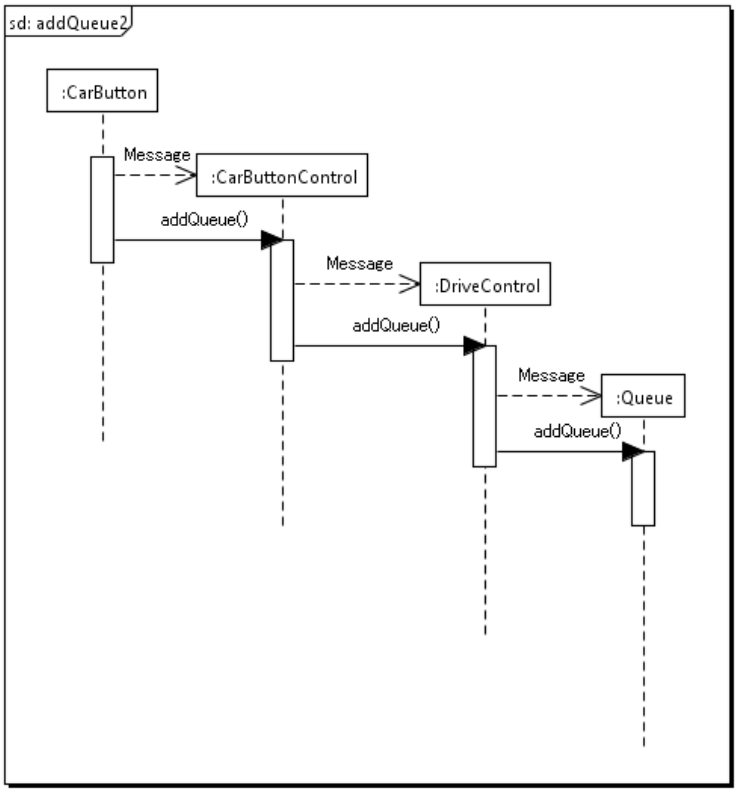


รูปที่ 5.27 การจำลองข้อบกพร่องประเภท Long Parameter List คลาส Queue

แผนภาพซีเควนซ์ในรูปที่ 5.28 และ 5.29 แสดงการเพิ่มเหตุการณ์เข้าคิวเมื่อมีการกดปุ่มเรียกลิฟต์ โดยแผนภาพซีเควนซ์ในรูปที่ 5.28 จะเป็นการจำลองข้อบกพร่องประเภท Middle Man เนื่องจากในกระบวนการเพิ่มคิวนั้นมีการเรียกเมท็อด addQueue() ผ่านตัวกลาง ในขณะที่แผนภาพซีเควนซ์ในรูปที่ 5.29 จะเป็นการจำลองข้อบกพร่องประเภท Message Chain เนื่องจากเกิดการเรียกเมท็อดต่อกันเป็นลูกโซ่นั่นเอง

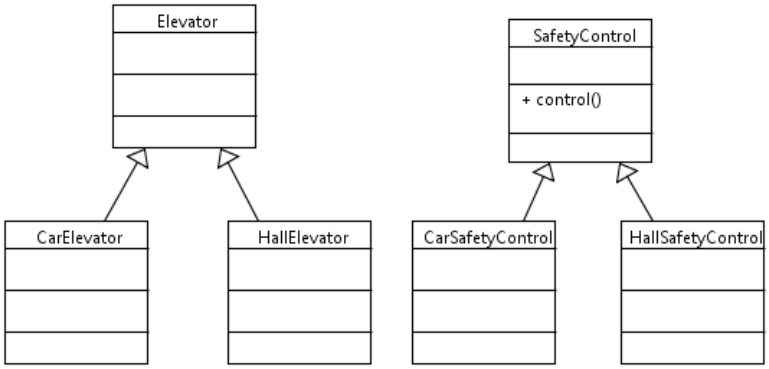


รูปที่ 5.28 การจำลองข้อบกพร่องประเภท Middle Man แผนภาพซีเควนซ์ addQueue

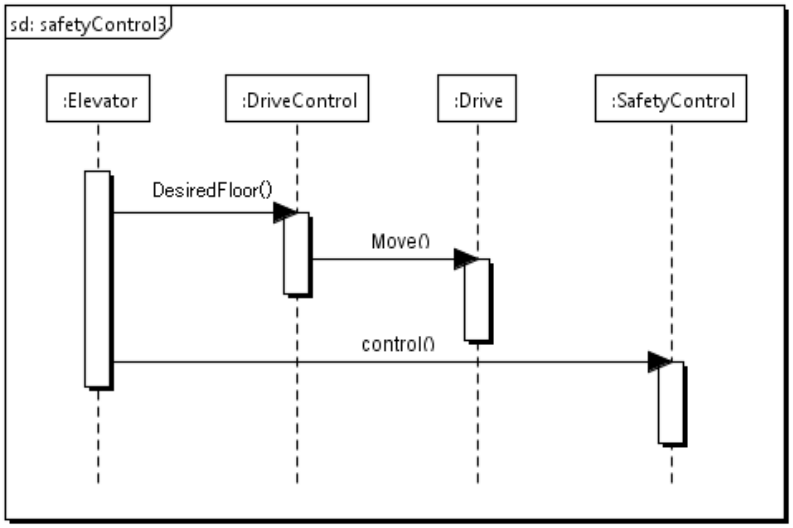


รูปที่ 5.29 การจำลองข้อบกพร่องประเภท Message Chain แผนภาพซีเควนซ์ addQueue2

ในการจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy ได้ทำการเพิ่มโครงสร้างการสืบทอดคลาส Elevator และ SafetyControl เข้าไปตามรูปที่ 5.30 โดยทั้งสองโครงสร้างนี้ จะมีความสัมพันธ์กันแบบจับคู่ ดังแสดงตัวอย่างความสัมพันธ์ที่รากของการสืบทอดตามรูปที่ 5.31 ซึ่งการมีความสัมพันธ์กันในลักษณะนี้จะถือว่าเป็นข้อบกพร่องประเภท Parallel Inheritance Hierarchy



รูปที่ 5.30 การจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy โครงสร้างการสืบทอดคลาส



รูปที่ 5.31 การจำลองข้อบกพร่องประเภท Parallel Inheritance Hierarchy แผนภาพซีเควนซ์แสดงความสัมพันธ์ที่รากของการสืบทอด

การจำลองข้อบกพร่องในโมเดลการออกแบบระบบควบคุมลิฟต์สามารถสรุปได้ตามตารางที่ 5.7 โดยโมเดลการออกแบบจะประกอบไปด้วยข้อบกพร่องประเภท Feature Envy ที่เมท็อดภายในคลาส Drive, DoorControl และ Door และข้อบกพร่องประเภท Lazy Class อยู่แล้ว

ตารางที่ 5.7 ตารางสรุปการจำลองข้อบกพร่องในโมเดลการออกแบบระบบควบคุมลิฟต์

เมท็อดที่เกิดข้อบกพร่อง	**คลาสที่เกิดข้อบกพร่อง**	**ประเภทของข้อบกพร่อง**
getState	ElevatorControl	Refused Bequest
lightOn	ButtonControl	Refused Bequest
onDisco	LanternContorl	Inappropriate Intimacy
onDim	LanternControl	Inappropriate Intimacy

ตารางที่ 5.7 ตารางสรุปการจำลองข้อบกพร่องในโมเดลการออกแบบระบบควบคุมลิฟต์ (ต่อ)

เมท็อดที่เกิดข้อบกพร่อง	คลาสที่เกิดข้อบกพร่อง	ประเภทของข้อบกพร่อง
-	CallEvent	Data Class
-	ElevatorControl	Data Class
-	Elevator	Data Class
-	CarElevator	Data Class
-	HallElevator	Data Class
addQueue	Queue	Long Parameter List
createEvent	Queue	Long Parameter List
-	Dispatcher	Large Class
addQueue	HallButton	Middle Man
Move	Drive	Feature Envy
AtFloor	DoorControl	Feature Envy
Open	Door	Feature Envy
CarButtonControl	CarButton	Message Chain
-	SafetyControl	Parallel Inheritance Hierarchy
-	EmergencyBrake	Lazy Class
-	ElevatorControl	Lazy Class
-	ButtonControl	Lazy Class
-	DoorMotor	Lazy Class
-	Lantern	Lazy Class
-	CarPositionIndicator	Lazy Class
-	CallEvent	Lazy Class
-	Elevator	Lazy Class
-	CarElevator	Lazy Class
-	HallElevator	Lazy Class

ตารางที่ 5.7 ตารางสรุปการจำลองข้อบกพร่องในโมเดลการออกแบบระบบควบคุมลิฟต์ (ต่อ)

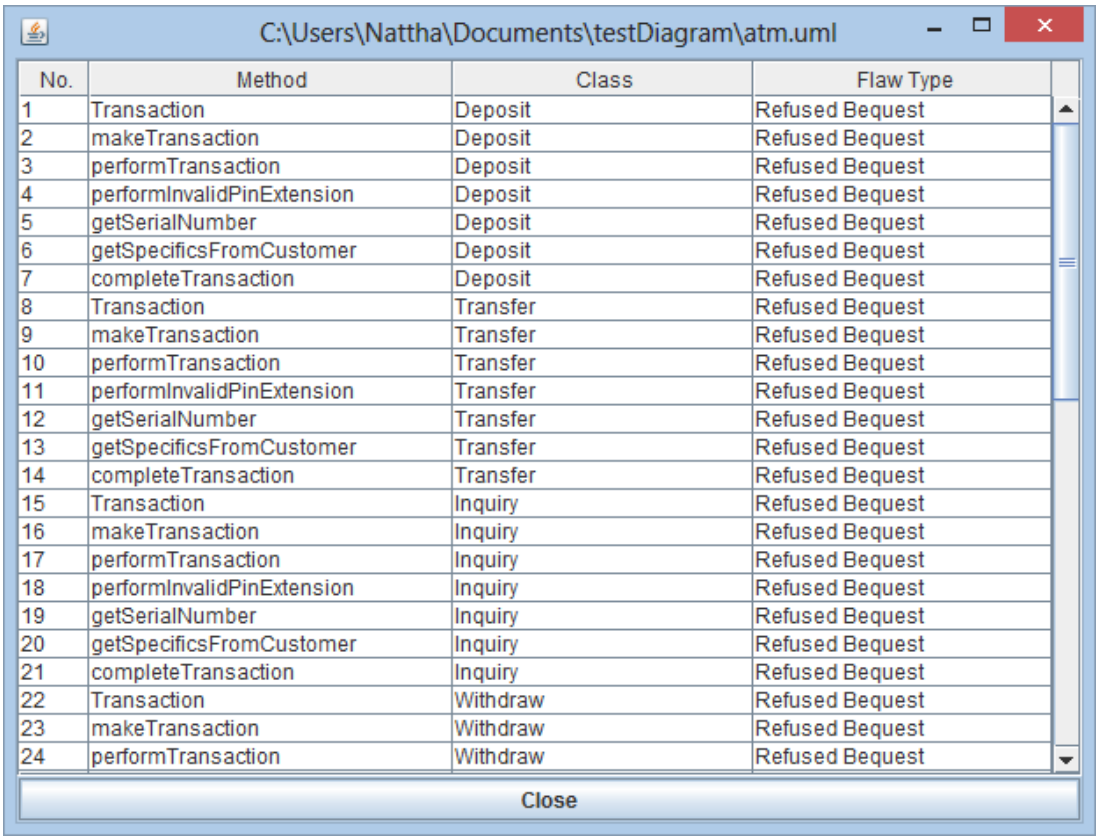
เมท็อดที่เกิดข้อบกพร่อง	คลาสที่เกิดข้อบกพร่อง	ประเภทของข้อบกพร่อง
-	SafetyControl	Lazy Class
-	CarSafetyControl	Lazy Class
-	HallSafetyControl	Lazy Class

5.2 ผลการค้นหาข้อบกพร่องโดยใช้เครื่องมือ

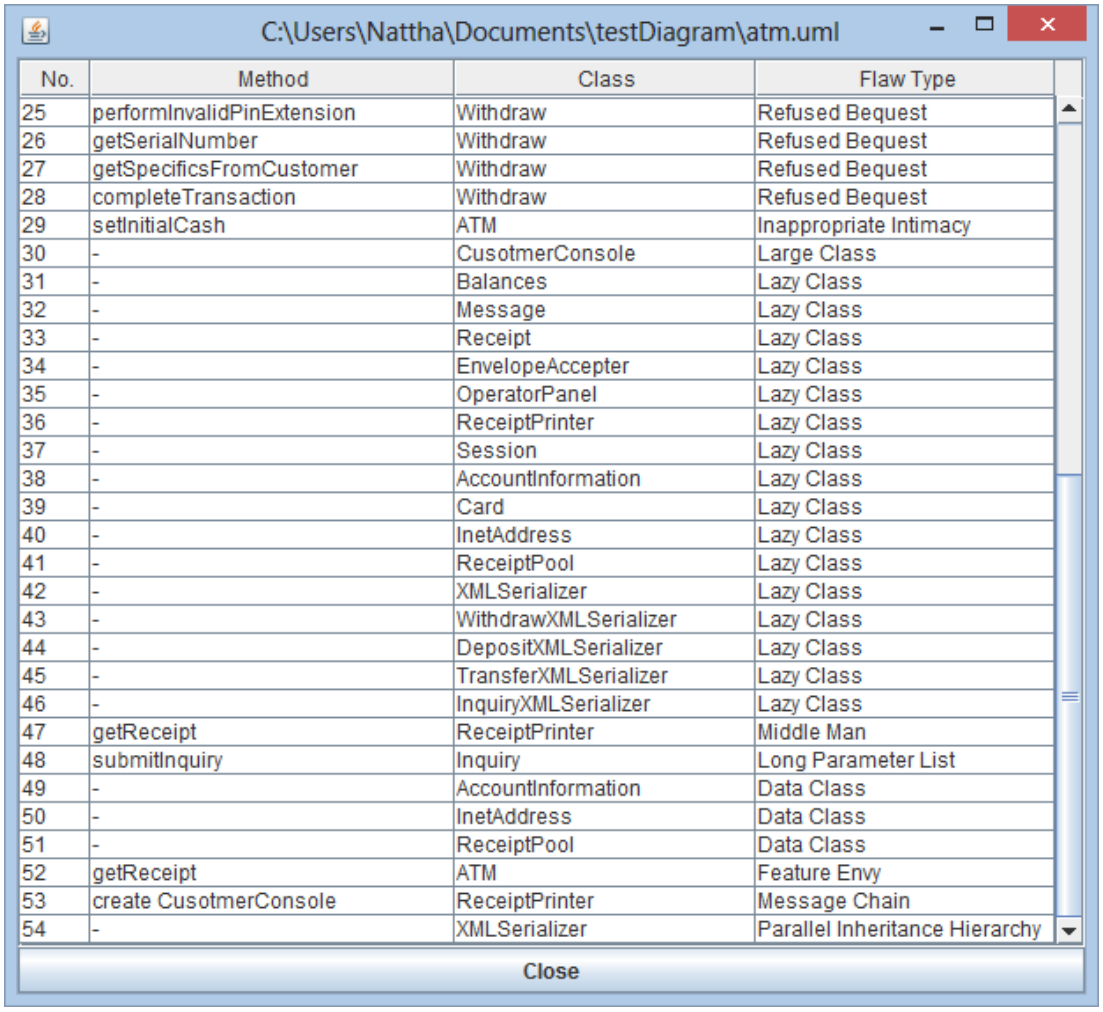
ในหัวข้อนี้แสดงผลการค้นหาข้อบกพร่องในโมเดลการออกแบบทั้ง 3 ระบบ โดยคู่มือการใช้งานเครื่องมือที่พัฒนาขึ้นมา สามารถดูได้ในภาคผนวก ก.

5.2.1 ผลการค้นหาข้อบกพร่องในโมเดลการออกแบบระบบที่ 1 ระบบ ATM

ผลการค้นหาดังแสดงในรูปที่ 5.32 และ 5.33 โดยระบบสามารถตรวจพบข้อบกพร่องทั้งที่มีอยู่เดิมในระบบ และที่จำลองเข้าไปในหัวข้อก่อนหน้าได้



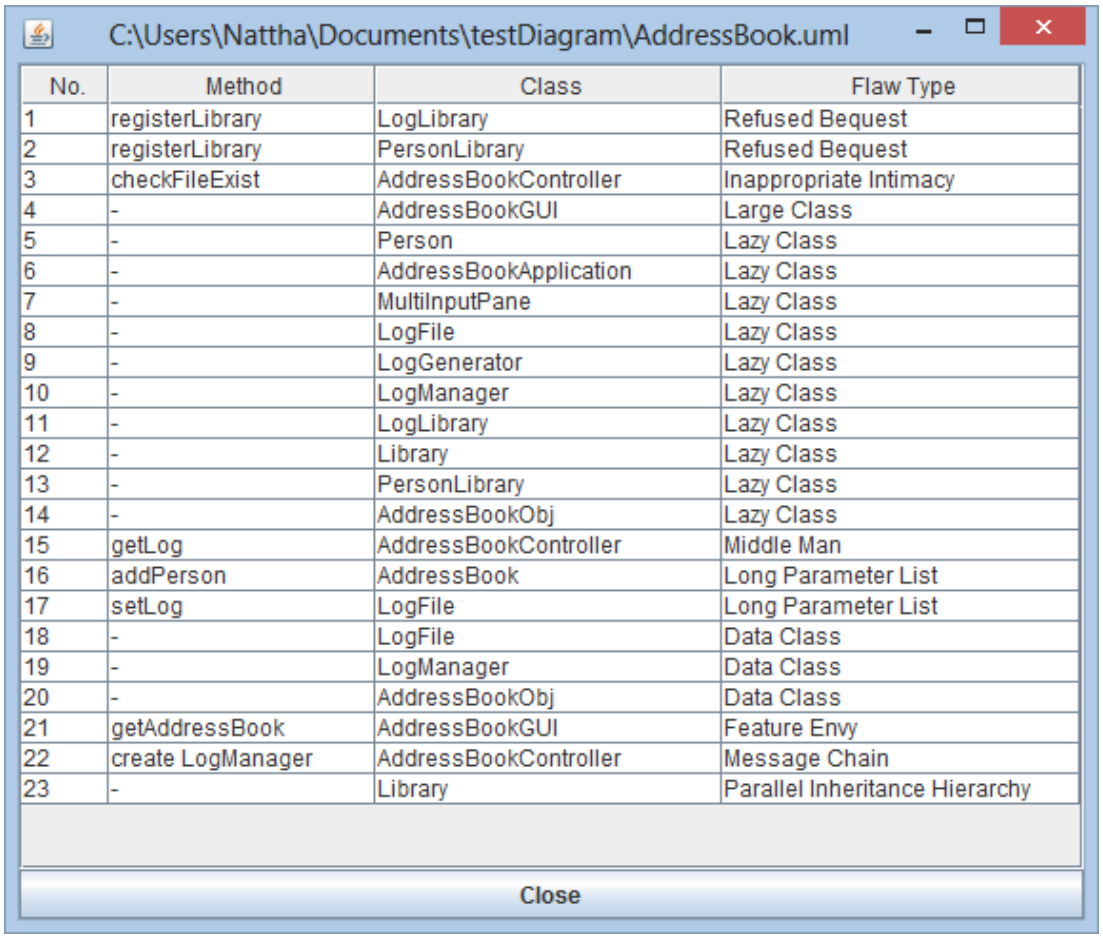
รูปที่ 5.32 ผลการค้นหาข้อบกพร่องในโมเดลการออกระบบแบบที่ 1 ระบบ ATM



รูปที่ 5.33 ผลการค้นหาข้อบกพร่องในโมเดลการออกระบบแบบที่ 1 ระบบ ATM (ต่อ)

5.2.2 ผลการค้นหาข้อบกพร่องในโมเดลการออกแบบระบบที่ 2 ระบบบันทึกข้อมูลผู้ติดต่อ

ผลการค้นหาดังแสดงในรูปที่ 5.34 โดยระบบสามารถตรวจพบข้อบกพร่องทั้งที่มีอยู่เดิมในระบบ และที่จำลองเข้าไปในหัวข้อก่อนหน้าได้

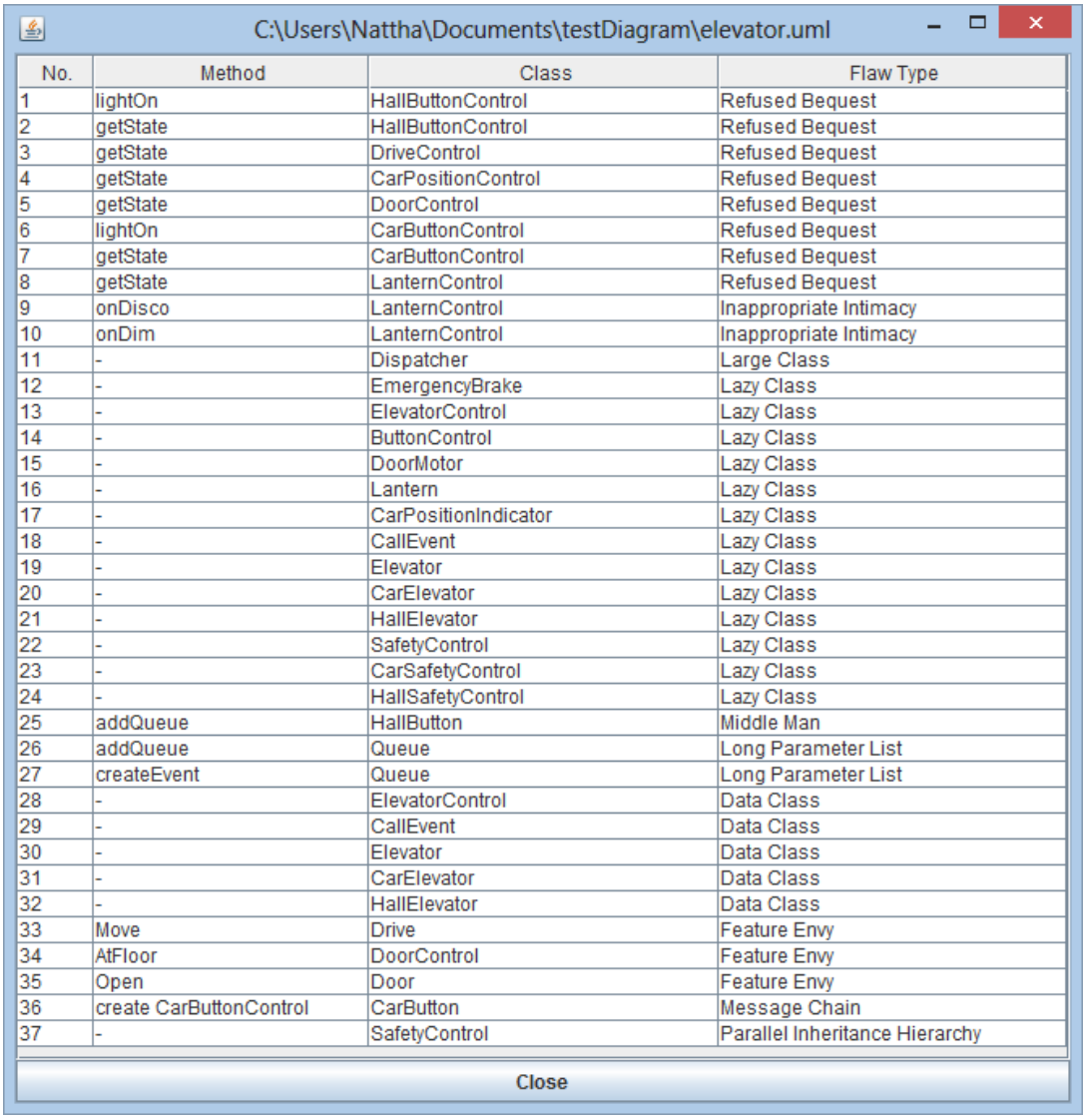


No.	Method	Class	Flaw Type
1	registerLibrary	LogLibrary	Refused Bequest
2	registerLibrary	PersonLibrary	Refused Bequest
3	checkFileExist	AddressBookController	Inappropriate Intimacy
4	-	AddressBookGUI	Large Class
5	-	Person	Lazy Class
6	-	AddressBookApplication	Lazy Class
7	-	MultiInputPane	Lazy Class
8	-	LogFile	Lazy Class
9	-	LogGenerator	Lazy Class
10	-	LogManager	Lazy Class
11	-	LogLibrary	Lazy Class
12	-	Library	Lazy Class
13	-	PersonLibrary	Lazy Class
14	-	AddressBookObj	Lazy Class
15	getLog	AddressBookController	Middle Man
16	addPerson	AddressBook	Long Parameter List
17	setLog	LogFile	Long Parameter List
18	-	LogFile	Data Class
19	-	LogManager	Data Class
20	-	AddressBookObj	Data Class
21	getAddressBook	AddressBookGUI	Feature Envy
22	create LogManager	AddressBookController	Message Chain
23	-	Library	Parallel Inheritance Hierarchy

รูปที่ 5.34 ผลการค้นหาข้อบกพร่องในโมเดลการออกระบบแบบที่ 2 ระบบ บันทึกข้อมูลผู้ติดต่อ

5.2.3 ผลการค้นหาข้อบกพร่องในโมเดลการออกแบบระบบที่ 3 ระบบควบคุมการทำงานของลิฟต์

ผลการค้นหาดังแสดงในรูปที่ 5.35 โดยระบบสามารถตรวจพบข้อบกพร่องทั้งที่มีอยู่เดิมในระบบ และที่จำลองเข้าไปในหัวข้อก่อนหน้าได้



No.	Method	Class	Flaw Type
1	lightOn	HallButtonControl	Refused Bequest
2	getState	HallButtonControl	Refused Bequest
3	getState	DriveControl	Refused Bequest
4	getState	CarPositionControl	Refused Bequest
5	getState	DoorControl	Refused Bequest
6	lightOn	CarButtonControl	Refused Bequest
7	getState	CarButtonControl	Refused Bequest
8	getState	LanternControl	Refused Bequest
9	onDisco	LanternControl	Inappropriate Intimacy
10	onDim	LanternControl	Inappropriate Intimacy
11	-	Dispatcher	Large Class
12	-	EmergencyBrake	Lazy Class
13	-	ElevatorControl	Lazy Class
14	-	ButtonControl	Lazy Class
15	-	DoorMotor	Lazy Class
16	-	Lantern	Lazy Class
17	-	CarPositionIndicator	Lazy Class
18	-	CallEvent	Lazy Class
19	-	Elevator	Lazy Class
20	-	CarElevator	Lazy Class
21	-	HallElevator	Lazy Class
22	-	SafetyControl	Lazy Class
23	-	CarSafetyControl	Lazy Class
24	-	HallSafetyControl	Lazy Class
25	addQueue	HallButton	Middle Man
26	addQueue	Queue	Long Parameter List
27	createEvent	Queue	Long Parameter List
28	-	ElevatorControl	Data Class
29	-	CallEvent	Data Class
30	-	Elevator	Data Class
31	-	CarElevator	Data Class
32	-	HallElevator	Data Class
33	Move	Drive	Feature Envy
34	AtFloor	DoorControl	Feature Envy
35	Open	Door	Feature Envy
36	create CarButtonControl	CarButton	Message Chain
37	-	SafetyControl	Parallel Inheritance Hierarchy

รูปที่ 5.35 ผลการค้นหาข้อบกพร่องในโมเดลการออกระบบแบบที่ 3 ระบบควบคุมการทำงานของลิฟต์

5.3 การเปรียบเทียบผลที่ได้ระหว่างการค้นหาด้วยเครื่องมือ กับค่าความจริง

การเปรียบเทียบผลการค้นหาข้อบกพร่องด้วยเครื่องมือกับค่าความจริงแบ่งตามโมเดลการออกแบบ ตามตารางที่ 5.8 ถึง ตารางที่ 5.10

ตารางที่ 5.8 ตารางเปรียบเทียบผลการค้นหาข้อบกพร่องด้วยเครื่องมือ กับค่าความจริงในโมเดลการออกแบบที่ 1

โมเดลการออกแบบที่ 1					
ลำดับ	ประเภทของข้อบกพร่อง	ส่วนที่เกิดข้อบกพร่อง		การค้นหา	
		เมท็อด	คลาส	ค่าความจริง	เครื่องมือ
1	Large Class	-	CustomerConsole	YES	YES
2	Long Parameter Lists	-	AccountInformation	YES	YES
3	Feature Envy	getReceipt	ATM	YES	YES
4	Parallel Inheritance Hierarchy	-	XMLSerializer	YES	YES
5	Message Chain	getReceipt	ReceiptPrinter	YES	YES
6	Middle Man	getReceipt	Receipt Printer	YES	YES
7	Refused Bequest	checkBalance	Inquiry	YES	YES
		checkBalance	Deposit	YES	YES
		checkBalance	Transfer	YES	YES
		checkBalance	Withdraw	YES	YES
8	Inappropriate Intimacy	setInitialCash	CashDispenser	YES	YES
9	Data Class	-	AccountInformation	YES	YES
		-	InetAddress	YES	YES
		-	ReceiptPool	YES	YES
10	Lazy Class	-	Balance	YES	YES
		-	Message	YES	YES
		-	Receipt	YES	YES
		-	EnvelopeAccepter	YES	YES
		-	OperatorPanel	YES	YES
		-	ReceiptPrinter	YES	YES

		-	Session	YES	YES
		-	AccountInformation	YES	YES
		-	Card	YES	YES
		-	InetAddress	YES	YES
		-	ReceiptPool	YES	YES
		-	XMLSerializer	YES	YES
		-	WithdrawXMLSerializer	YES	YES
		-	DespositXMLSerializer	YES	YES
		-	TransferXMLSerializer	YES	YES
		-	InquiryXMLSerializer	YES	YES

ตารางที่ 5.9 ตารางเปรียบเทียบผลการค้นหาร่องรอยที่ไม่ดีด้วยเครื่องมือกับค่าความจริงในโมเดลการออกแบบที่ 2

โมเดลการออกแบบที่ 2					
ลำดับ	ประเภทของข้อบกพร่อง	ส่วนที่เกิดข้อบกพร่อง		การค้นหา	
		เมท็อด	คลาส	ค่าความจริง	เครื่องมือ
1	Large Class	-	AddressBookGUI	YES	YES
2	Long Parameter List	addPerson	AddressBook	YES	YES
		setLog	LogFile	YES	YES
3	Feature Envy	getAddressBook	AddressBookController	YES	YES
4	Parallel Inheritance Hierarchy	-	Library	YES	YES
5	Message Chain	getLog	AddressBookController	YES	YES
6	Middle Man	getLog	AddressBookController	YES	YES
7	Refused Bequest	registerLibrary	LogLibrary	YES	YES
		registerLibrary	PersonLibrary	YES	YES
8	Inappropriate	checkFileExist	AddressBookController	YES	YES

	Intimacy				
9	Data Class	-	LogFile	YES	YES
		-	LogManager	YES	YES
		-	AddressBookObj	YES	YES
10	Lazy Class	-	Person	YES	YES
		-	AddressBookApplication	YES	YES
		-	MultiInputPane	YES	YES
		-	LogFile	YES	YES
		-	LogGenerator	YES	YES
		-	LogManager	YES	YES
		-	LogLibrary	YES	YES
		-	Library	YES	YES
		-	PersonLibrary	YES	YES
		-	AddressBookObj	YES	YES

ตารางที่ 5.10 ตารางเปรียบเทียบผลการค้นหาร่องรอยที่ไม่ดีด้วยเครื่องมือกับค่าความจริงในโมเดลการออกแบบที่ 3

โมเดลการออกแบบที่ 3					
ลำดับ	ประเภทของข้อบกพร่อง	ส่วนที่เกิดข้อบกพร่อง		การค้นหา	
		เมท็อด	คลาส	ค่าความจริง	เครื่องมือ
1	Large Class	-	Dispatcher	YES	YES
2	Long Parameter List	addQueue	Queue	YES	YES
		createEvent	Queue	YES	YES
3	Feature Envy	Move	Drive	YES	YES
		AtFloor	DoorControl	YES	YES
		Open	Door	YES	YES
4	Parallel Inheritance Hierarchy	-	SafetyControl	YES	YES
5	Message Chain	addQueue	CarButton	YES	YES
6	Middle Man	addQueue	HallButton	YES	YES

7	Refused Bequest	getState	ElevatorControl	YES	YES
		lightOn	ButtonControl	YES	YES
8	Inappropriate Intimacy	onDisco	LanternContorl	YES	YES
		onDim	LanternControl	YES	YES
9	Data Class	-	CallEvent	YES	YES
		-	ElevatorControl	YES	YES
		-	Elevator	YES	YES
		-	CarElevator	YES	YES
		-	HallElevator	YES	YES
10	Lazy Class	-	EmergencyBrake	YES	YES
		-	ElevatorControl	YES	YES
		-	ButtonControl	YES	YES
		-	DoorMotor	YES	YES
		-	Lantern	YES	YES
		-	CarPositionIndicator	YES	YES
		-	CallEvent	YES	YES
		-	Elevator	YES	YES
		-	CarElevator	YES	YES
		-	HallElevator	YES	YES
		-	SafetyControl	YES	YES
		-	CarSafetyControl	YES	YES
		-	HallSafetyControl	YES	YES

สามารถสรุปเป็นเปอร์เซ็นต์ ได้ดังรายละเอียดในตารางที่ 5.11

ตารางที่ 5.11 ตารางสรุปเปอร์เซ็นต์ในการทดสอบวิธีการในการค้นหาข้อบกพร่องในโมเดลการออกแบบ

ข้อบกพร่อง	โมเดลการออกแบบที่ 1	โมเดลการออกแบบที่ 2	โมเดลการออกแบบที่ 3
Large Class	100%	100%	100%
Long Parameter Lists	100%	100%	100%
Feature Envy	100%	100%	100%

ตารางที่ 5.11 ตารางสรุปเปอร์เซ็นต์ในการทดสอบวิธีการในการค้นหาข้อบกพร่องในโมเดลการออกแบบ (ต่อ)

ข้อบกพร่อง	โมเดลการออกแบบที่ 1	โมเดลการออกแบบที่ 2	โมเดลการออกแบบที่ 3
Data Class	100%	100%	100%
Lazy Class	100%	100%	100%
Parallel Inheritance Hierarchy	100%	100%	100%
Message Chain	100%	100%	100%
Middle Man	100%	100%	100%
Refused Bequest	100%	100%	100%
Inappropriate Intimacy	100%	100%	100%

บทที่ 6

บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

วิทยานิพนธ์ฉบับนี้ได้นำเสนอวิธีการในการค้นหาข้อบกพร่องในโมเดลการออกแบบ ด้วยแผนภาพกราฟ และแผนภาพต้นไม้ วิธีการที่นำเสนอประกอบไปด้วยข้อกำหนดแผนภาพแสดงโมเดลการออกแบบ และแม่แบบของข้อบกพร่อง โมเดลการออกแบบจะถูกแปลงด้วยข้อกำหนดของแผนภาพแสดงโมเดลการออกแบบ แม่แบบของข้อบกพร่องจะถูกนำมาใช้ตรวจสอบเปรียบเทียบกับแผนภาพเพื่อค้นหาข้อบกพร่อง

ในการออกแบบแผนภาพ นิยามของข้อบกพร่องจะถูกนำมาวิเคราะห์ เพื่อหาข้อมูลที่จำเป็นสำหรับการค้นหาข้อบกพร่อง โดยสรุปได้ 3 ประเภท ได้แก่ ข้อมูลการสืบทอดคลาส ข้อมูลการออกแบบคลาส และข้อมูลความสัมพันธ์ระหว่างคลาส งานวิจัยนี้จึงได้นำเสนอแผนภาพแสดงโมเดลการออกแบบทั้งสิ้น 3 แผนภาพด้วยกัน ประกอบไปด้วย 1) แผนภาพต้นไม้การสืบทอด เป็นแผนภาพต้นไม้แสดงการสืบทอดของคลาส 2) แผนภาพต้นไม้คุณลักษณะ เป็นแผนภาพต้นไม้แสดงส่วนประกอบคุณลักษณะของคลาส และ 3) แผนภาพกราฟตรรกะ เป็นแผนภาพกราฟ แสดงความสัมพันธ์ระหว่างคลาส โดยข้อมูลการออกแบบเหล่านี้จะมีปรากฏอยู่ในแผนภาพคลาส และแผนภาพซีเควนซ์ในขั้นตอนการออกแบบ

แม่แบบของข้อบกพร่องสร้างขึ้นมาจากการวิเคราะห์คุณลักษณะที่จะนำมาใช้ในการค้นหาข้อบกพร่องแต่ละประเภท และนำคุณลักษณะเหล่านั้นมาสร้างเป็นแม่แบบ แม่แบบจะถูกนำไปตรวจสอบกับแผนภาพแสดงโมเดลการออกแบบในกระบวนการค้นหา

ในการทดสอบความสามารถของวิธีการเครื่องมือได้ถูกพัฒนาขึ้นด้วยภาษาจาวาตามวิธีการที่นำเสนอ จากนั้นได้นำไปทำการทดสอบกับโมเดลการออกแบบซอฟต์แวร์ขนาดเล็ก จำนวน 3 ระบบด้วยกัน ประกอบไปด้วย 1) ระบบฝากและถอนเงินอัตโนมัติ 2) ระบบบันทึกข้อมูลผู้ติดต่อ และ 3) ระบบควบคุมการทำงานของลิฟต์ การเตรียมโมเดลทดสอบได้ใช้เทคนิคการจำลองข้อบกพร่อง ด้วยการวิเคราะห์โมเดลการออกแบบ ข้อบกพร่องได้ถูกจำลองสู่ส่วนต่างๆของโมเดลการออกแบบ จากนั้นจึงนำเครื่องมือมาทดสอบกับโมเดลที่ได้จำลองข้อบกพร่องไว้ ค่าที่ได้จากการ

ใช้เครื่องมือจะถูกนำมาเปรียบเทียบกับค่าจริงของข้อบกพร่องที่มีอยู่ในโมเดลทดสอบเพื่อประเมินประสิทธิภาพของวิธีการ

ผลการทดสอบพบว่าวิธีการนี้สามารถค้นหาข้อบกพร่องประเภท Large Class, Long Parameter List, Feature Envy, Parallel Inheritance Hierarchy, Message Chain, Middle Man, Refused Bequest, Inappropriate Intimacy, Data Class, และ Lazy Class ได้

6.2 ข้อเสนอแนะ

1. ควรมีการนำเครื่องมือที่พัฒนาทดสอบกับโมเดลการออกแบบซอฟต์แวร์ที่ใหญ่ขึ้น การทดสอบในงานวิจัยนี้ได้มีการดำเนินการกับโมเดลการออกแบบซอฟต์แวร์ขนาดเล็กเพื่อประเมินความสามารถของวิธีการในเบื้องต้นเท่านั้น
2. ควรมีการศึกษาและพัฒนาแม่แบบเพิ่มเติมเพื่อให้ครอบคลุมข้อบกพร่องอื่นๆ นอกเหนือไปจากที่ได้นำเสนอในงานวิจัยนี้ โดยเครื่องมือที่ออกแบบไว้ด้วยหลักการ Interface สามารถนำแม่แบบมาพัฒนา และนำมาบรรจุเพิ่มได้

6.3 ข้อจำกัดของงานวิจัย

1. วิธีการที่นำเสนอสามารถนำไปใช้ได้ในขั้นตอนการออกแบบซอฟต์แวร์ ซึ่งเป็นผลให้มีข้อจำกัดด้านข้อมูลที่นำมาวิเคราะห์ข้อบกพร่องประเภท Duplicate Code, Long Method, Divergent Change, Shotgun Surgery, Data Clumps, Primitive Obsession, Switch Statement, Speculative Generality, Temporary Field, Alternative Class with Difference Interfaces, Incomplete Library Class, และ Comment
2. งานวิจัยนี้ได้นำเสนอแม่แบบในการค้นหาข้อบกพร่อง ซึ่งรูปแบบของข้อบกพร่องจะต้องมีลักษณะการเกิดตรงกับแม่แบบ จึงจะสามารถค้นหาได้

รายการอ้างอิง

[1] Munro, M. Product Metrics for Automatic Identification of “Bad Smell" Design Problems in Java Source-Code. 11th IEEE International Software Metrics Symposium (2005): 15.

[2] Marinescu, R. Detecting Design Flaws via Metrics in Object-Oriented Systems. 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems (2001): 173.

[3] Pienlert, T. Bad-smell detection for refactoring using object-oriented software metrics, Master’s Thesis, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 2004

[4] Yeesoon, K. Design and implementation of a tool for detecting bad smells in Java program, Master’s Thesis, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 2008

[5] Maneerat, N. Bad-smell prediction from software design model using machine learning techniques, Master’s Thesis, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 2010

[6] Moha, N., Gueheneuc, Y., and Leduc, P. Automatic Generation of Detection Algorithms for Design Defects. 21st IEEE International Conference on Automated Software Engineering (2006): 297-300.

[7] Moha, N., Guéhéneuc, Y., and Duchien, L. Anne-Françoise Le Meur. DECOR: A Method for the Specification and Detection of Code and Design Smells. IEEE Transactions on Software Engineering (2010): 20-36.

[8] Fowler, M. Refactoring: Improving the Design of Existing Code. United States: Addison-Wesley Longman Publishing.

[9] Goodrich, M., and Tamassia, R. Algorithm Design: Foundations, Analysis, and Internet Examples. John Wiley & Sons, Inc.,: 2002.

[10] Mäntylä, M., Vanhanen, J., and Lassenius C. A Taxonomy and an Initial Empirical Study of Bad Smells in Code. 19th IEEE International Conference on Software Maintenance (2003): 381.

[11] Houari, A., Sahraoui, Miceli, T., and Godin, R. A Metric Based Technique for Design Flaws Detection and Correction. 14th IEEE International Conference on Automated Software Engineering (1999): 307.

[12] Salehie, M., Li, S., and Tahvildari, L. A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws. 14th IEEE International Conference on Program Comprehension (2006): 159-168.

[13] Hassaine, S., Khomh, F., Guéhéneuc, Y., and Hamel, S. IDS: An Immune-Inspired Approach for the Detection of Software Design Smells. 2010 Seventh International Conference on the Quality of Information and Communications Technology (2010): 343-348.

[14] Jiang, Y., Cukic, B., Menzies, T., and Bartlow, N. Comparing Design and Code Metrics for Software Quality Prediction. Proceedings of the PROMISE 2008 Workshop 2008.

[15] Guo, Y., Seaman, C., Zazworka, N., and Shull, F. Domain-specific tailoring of code smells: an empirical study. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2 (2010): 167-170.

ภาคผนวก

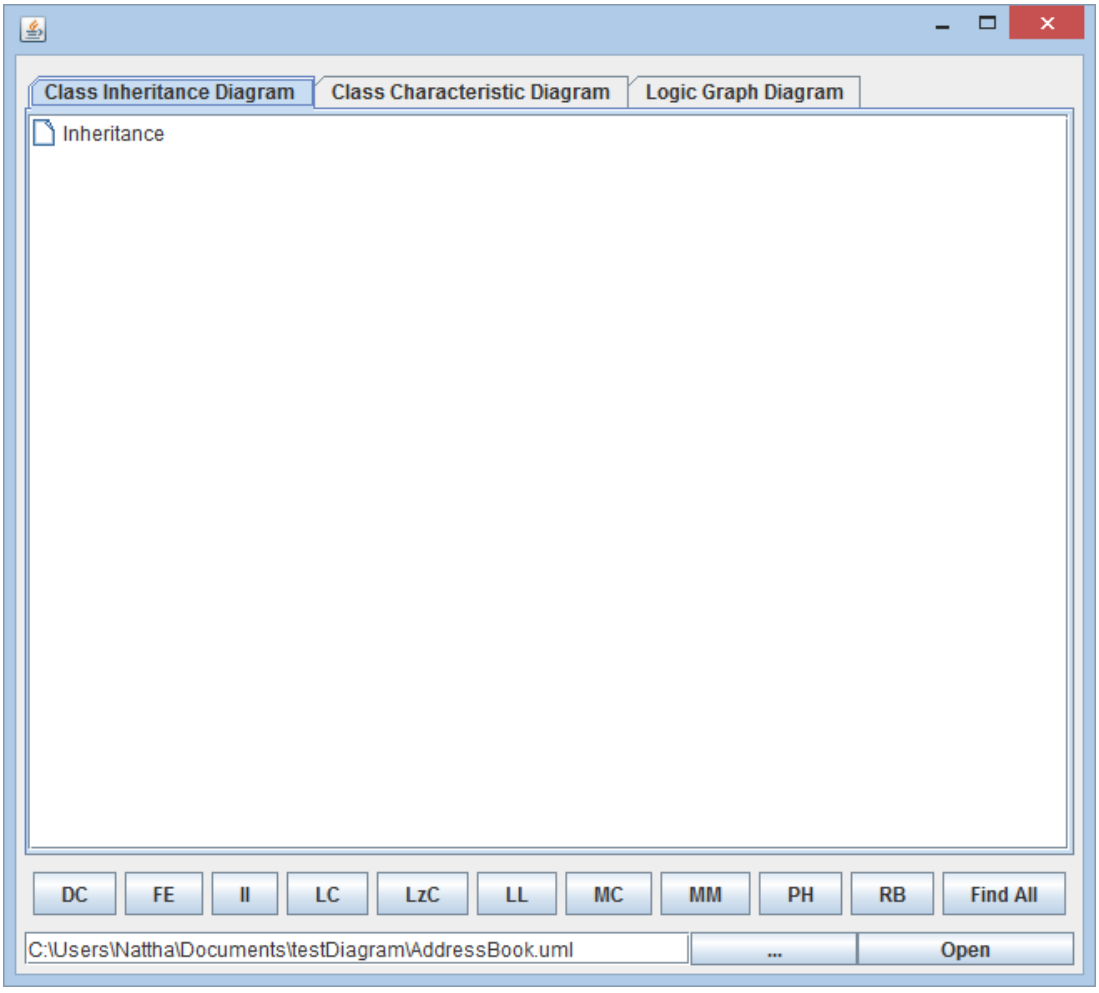
ภาคผนวก ก.

คู่มือการใช้งานเครื่องมือค้นหาข้อบกพร่องในโมเดลการออกแบบ

เครื่องมือในการค้นหาข้อบกพร่องในโมเดลการออกแบบได้พัฒนาขึ้นด้วยภาษาจาวาตามวิธีการที่นำเสนอในงานวิจัย โดยรายละเอียดการออกแบบเครื่องมือได้เขียนอธิบายไว้แล้วในบทที่ 3

หน้าต่างการทำงานหลักของโปรแกรม

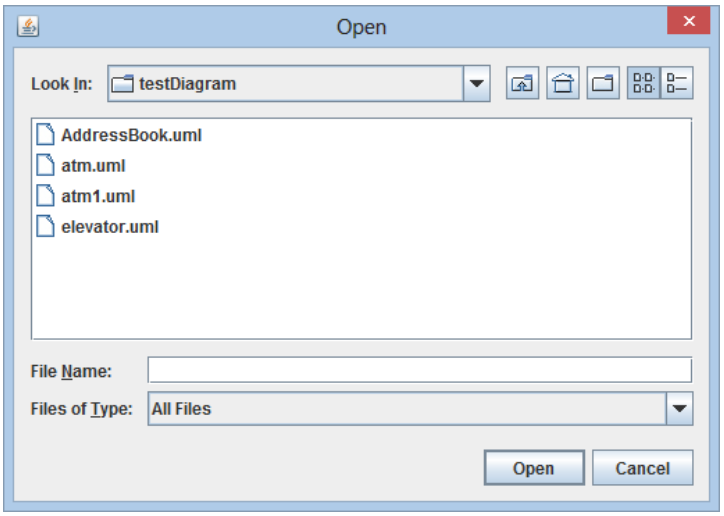
เมื่อเปิดโปรแกรมขึ้นมา ผู้ใช้งานจะเห็นหน้าต่างหลัก ตามรูปที่ ก.1



รูปที่ ก.1 หน้าต่างหลักของเครื่องมือค้นหาข้อบกพร่องในโมเดลการออกแบบ

หน้าต่างหลักประกอบไปด้วยส่วนแสดงผลการสร้างโมเดลแสดงแทนโมเดลการออกแบบที่ และเมนูสำหรับเลือกไฟล์การออกแบบซอฟต์แวร์ ผู้ใช้งานสามารถเลือกไฟล์โมเดลการ

ออกแบบที่ต้องการได้ด้วยการใส่ที่อยู่ไฟล์ลงไป หรือกดที่ปุ่ม '...' เพื่อเปิดหน้าต่างเลือกไฟล์ ตามรูปที่ ก.2

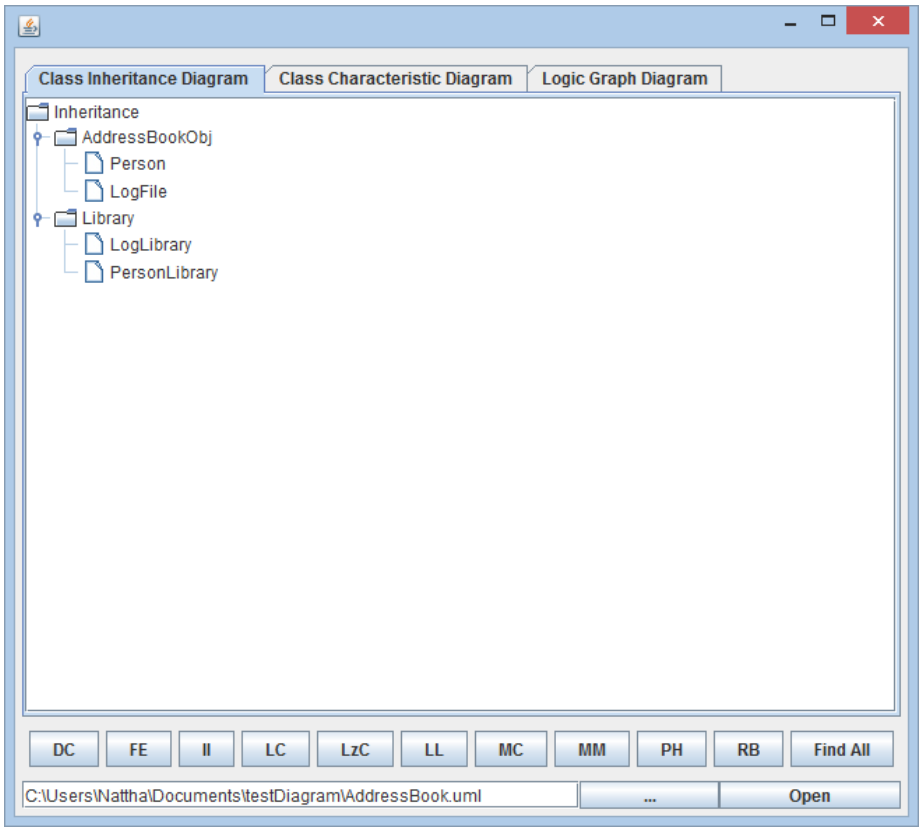


รูปที่ ก.2 หน้าต่างเลือกไฟล์โมเดลการออกแบบ

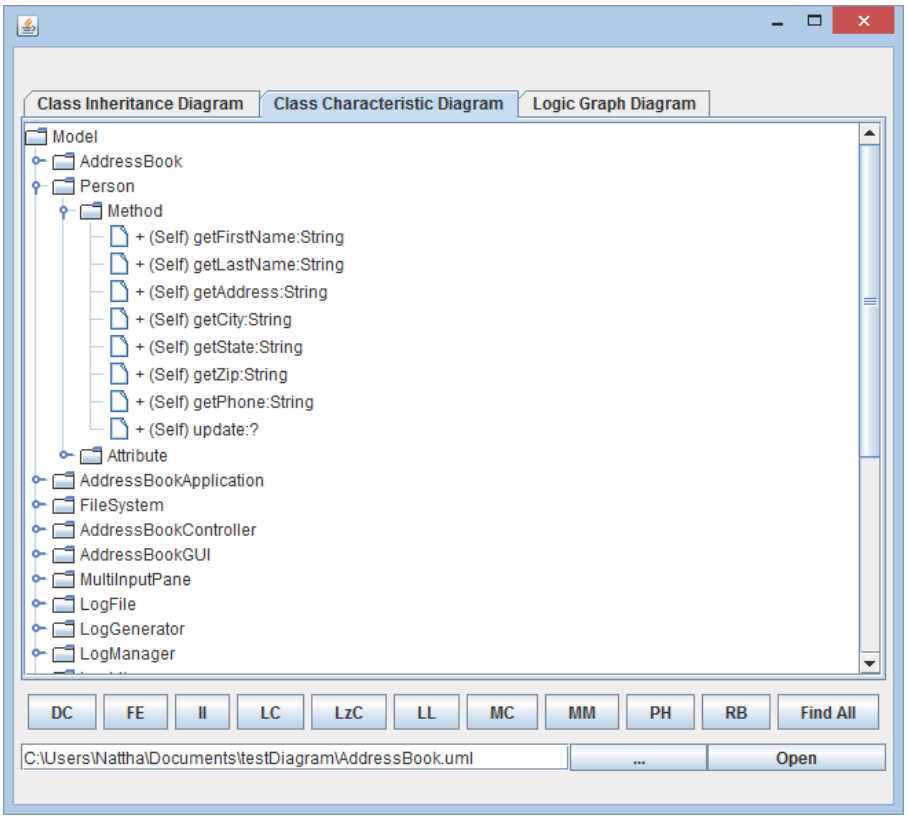
หลังจากที่ผู้ใช้งานเลือกไฟล์การออกแบบซอฟต์แวร์ที่ต้องการแล้ว โปรแกรมจะทำงาน 2 อย่าง คือ การอ่านแผนภาพยูเอ็มแอล แล้วสร้างแผนภาพแสดงแทนโมเดลการออกแบบ และการทำการค้นหาข้อบกพร่อง

หน้าต่างแสดงแผนภาพแสดงแทนโมเดลการออกแบบ

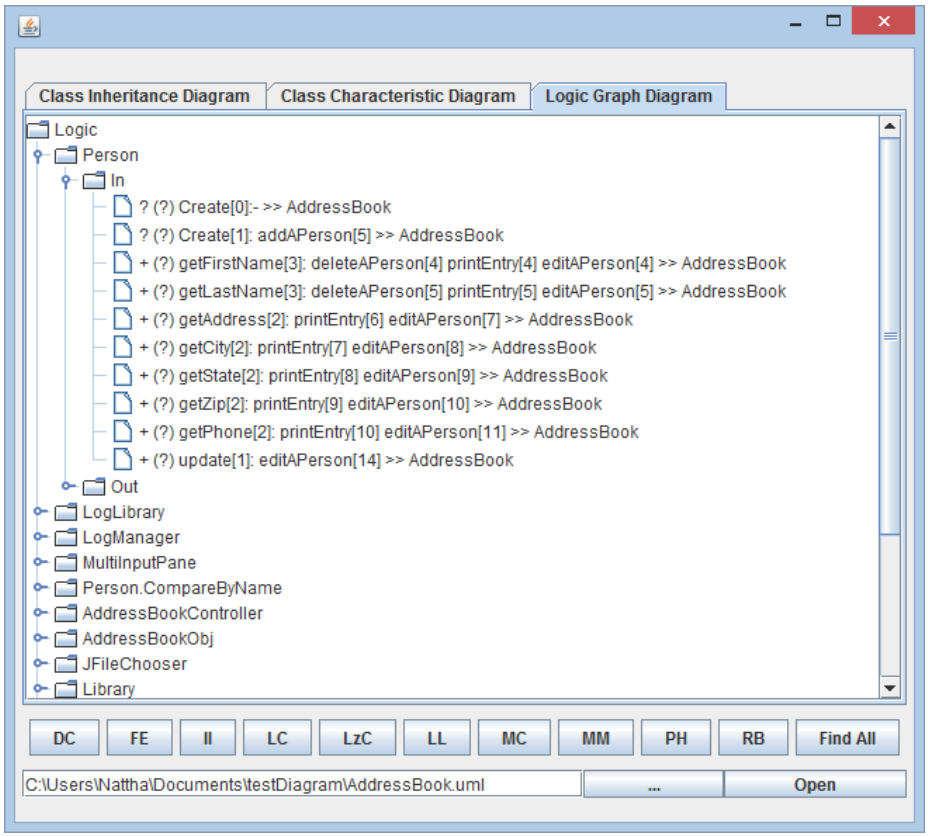
ที่หน้าต่างหลัก จะมีส่วนแสดงผลแผนภาพแสดงแทนโมเดลการออกแบบ ประกอบไปด้วย 3 แท็บด้วยกัน ประกอบด้วยแผนภาพต้นไม้การสืบทอด แผนภาพต้นไม้คุณลักษณะ และแผนภาพกราฟตรรกะ ผลการสร้างแผนภาพแสดงแทนโมเดลการออกแบบระบบ ATM ที่ใช้ในการทดสอบดังแสดงในรูปที่ ก.3-5



รูปที่ ก.3 หน้าต่างแสดงผลแผนภาพต้นไม้การสืบทอด



รูปที่ ก.4 หน้าต่างแสดงผลแผนภาพต้นไม้คุณลักษณะ



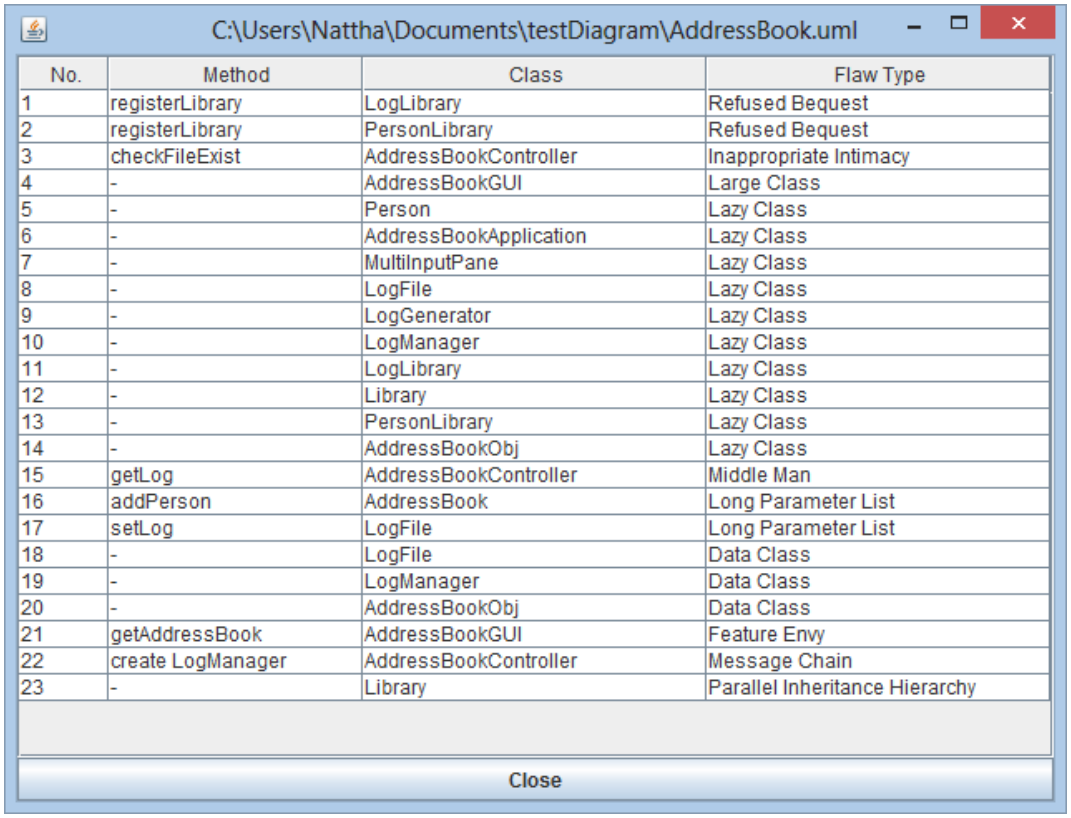
รูปที่ ก.5 หน้าต่างแสดงผลแผนภาพกราฟตรรกะ

หน้าต่างแสดงผลการค้นหาข้อบกพร่อง

หน้าต่างแสดงผลการค้นหาข้อบกพร่องจะเปิดขึ้นมาหลักจากที่ผู้ใช้งานได้ทำการเลือกเปิดไฟล์การออกแบบซอฟต์แวร์แล้ว ดังแสดงในรูปที่ ก.6 หน้าต่างแสดงผลการค้นหาข้อบกพร่องประกอบไปด้วยคอลัมน์ต่างๆ แสดงรายละเอียดของผลการค้นหา รายละเอียดของแต่ละคอลัมน์ดังแสดงในตารางที่ ก.1

ตารางที่ ก.1 ตารางแสดงคำอธิบายคอลัมน์แสดงผลการค้นหาข้อบกพร่อง

ชื่อคอลัมน์	คำอธิบาย
No.	ลำดับของข้อบกพร่อง
Primary Culprit	ส่วนที่เกิดข้อบกพร่องหลัก
Secondary Culprit	ส่วนขยายสถานที่เกิดข้อบกพร่อง เช่น ในกรณีที่ข้อบกพร่องหลักเกิดที่เมท็อด ส่วนขยายจะระบุว่าเป็นเมท็อดที่อยู่ในคลาสใด
Flaw Type	ประเภทของข้อบกพร่อง



No.	Method	Class	Flaw Type
1	registerLibrary	LogLibrary	Refused Bequest
2	registerLibrary	PersonLibrary	Refused Bequest
3	checkFileExist	AddressBookController	Inappropriate Intimacy
4	-	AddressBookGUI	Large Class
5	-	Person	Lazy Class
6	-	AddressBookApplication	Lazy Class
7	-	MultiInputPane	Lazy Class
8	-	LogFile	Lazy Class
9	-	LogGenerator	Lazy Class
10	-	LogManager	Lazy Class
11	-	LogLibrary	Lazy Class
12	-	Library	Lazy Class
13	-	PersonLibrary	Lazy Class
14	-	AddressBookObj	Lazy Class
15	getLog	AddressBookController	Middle Man
16	addPerson	AddressBook	Long Parameter List
17	setLog	LogFile	Long Parameter List
18	-	LogFile	Data Class
19	-	LogManager	Data Class
20	-	AddressBookObj	Data Class
21	getAddressBook	AddressBookGUI	Feature Envy
22	create LogManager	AddressBookController	Message Chain
23	-	Library	Parallel Inheritance Hierarchy

รูปที่ ก.6 หน้าต่างแสดงผลการค้นหาข้อบกพร่อง

ภาคผนวก ข.

ผลงานตีพิมพ์

ผลงานวิจัยนี้ ได้รับคัดเลือกให้ถูกตีพิมพ์ในงานสัมมนาวิชาการระหว่างประเทศ “International Symposium on Computer Science and Electrical Engineering (ISCSEE)” ซึ่งได้จัดขึ้นที่กรุงปักกิ่ง ประเทศจีน ระหว่างวันที่ 14-16 มิถุนายน 2556

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวณัฏฐา เยาวรัตนประเสริฐ เกิดเมื่อวันที่ 16 กันยายน พ.ศ. 2527 ที่จังหวัดสุโขทัย สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ในปีการศึกษา 2550 และได้เข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2553