ช่วงของค่ามาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดีของโปรแกรม

นางสาวพนิตา เมนะเนตร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-53-1169-3

THRESHOLD OF OBJECT-ORIENTED SOFTWARE METRICS FOR DETECTING BAD SMELLS CODE

Miss Panita Meananet

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2004

ISBN 974-53-1169-3

หัวข้อวิทยานิพนธ์	ช่วงของค่ามาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดีของโปรแกรม
โดย	นางสาวพนิตา เมนะเนตร
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

…….. อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

…… กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

…… กรรมการ
(อาจารย์นครทิพย์ พร้อมพูล)

พนิตา เมนะเนตร : ช่วงของค่ามาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดีของโปรแกรม. (THRESHOLD OF OBJECT-ORIENTED METRICS OF DETECTING BAD SMELLS CODE) อ. ที่ปรึกษา : ผศ. ดร.พรศิริ หมื่นไชยศรี 174 หน้า. ISBN 974-53-1169-3.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อหาช่วงของค่ามาตรวัดที่เกิด และไม่เกิดร่องรอยที่ไม่ดีของภาษาจาวา โดยเริ่มจากทำการสร้างโมเดลของมาตรวัดในการทำนายร่องรอยที่ไม่ดีทั้ง 8 แบบ คือ Data Class, Feature Envy, Large Class, Lazy Class, Long Method, Long Parameter List, Refused Bequest และ Switch Statement โดยใช้มาตรวัด 30 มาตรวัด ด้วยวิธีการวิเคราะห์ความถดถอยเพื่อหาความสัมพันธ์ระหว่างมาตรวัดกับร่องรอยที่ไม่ดี โดยใช้ชุดข้อมูลสอน 10 โปรแกรม และตรวจสอบความถูกต้องด้วยชุดข้อมูลทดสอบ 2 โปรแกรม หลังจากนั้นทำการออกแบบและทำการทดลองเพื่อเก็บข้อมูลในการหาช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีในโปรแกรมภาษาจาวา โดยใช้ชุดข้อมูลสอนสำหรับหาช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี แล้วตรวจสอบความถูกต้องของช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีด้วยชุดข้อมูลทดสอบ จากนั้นเปรียบเทียบการตรวจจับร่องรอยที่ไม่ดีด้วยมาตรวัดเดียวหรือหลายมาตรวัดที่ทำเงื่อนไข OR กัน ผลที่ได้คือใช้หลายมาตรวัดที่ทำเงื่อนไข OR กันสามารถตรวจจับร่องรอยที่ไม่ดีได้ดีกว่า

จากผลการตรวจสอบวิธีการใช้ช่วงของค่ามาตรวัด พบว่าสามารถใช้วิธีการใช้ช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน สำหรับร่องรอยที่ไม่ดีแบบ Data Class, Large Class, Lazy Class, Long Method, Long Parameter List และ Switch Statement แต่ไม่สามารถหาความสัมพันธ์ของมาตรวัดกับค่าร่องรอยที่ไม่ดีแบบ Feature Envy และ Refused Bequest ได้ เนื่องจากหาข้อมูลที่เกิดร่องรอยที่ไม่ดีทั้ง 2 แบบได้จำนวนน้อยไม่พอต่อทำการทดลอง นอกจากนี้ได้ออกแบบมาตรวัดซอฟต์แวร์ เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดีแบบ Data Class และ Refused Bequest รวมทั้งได้ออกแบบและพัฒนาเครื่องมือเพื่อการคำนวณมาตรวัดและตรวจจับการเกิดร่องรอยที่ไม่ดี

ภาควิชา..วิศวกรรมคอมพิวเตอร์.....ลายมือชื่อนิสิต.....
สาขาวิชา..วิศวกรรมซอฟต์แวร์.....ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2547

4670395021 : MAJOR SOFTWARE ENGINEERING

KEY WORD: OBJECT-ORIENTED SOFTWARE METRICS / REFACTORING / BAD-SMELLS

PANITA MEANANET : THRESHOLD OF OBJECT-ORIENTED METRICS OF DETECTING BAD SMELLS CODE.

THESIS ADVISOR : ASSIST. PROF. PORNSIRI MUENCHAISRI, Ph.D., 174 pp. ISBN 974-53-1169 -3.

The objective of this thesis is to find thresholds of metrics for detecting bad smell and non bad smell java code. First, bad smells predicting models and created for 8 types of bad smells, i.e. Data Class, Feature Envy, Large Class, Lazy Class, Long Method, Long Parameter List, Refused Bequest and Switch Statement. The models are created from 30 metrics that characterize these types of bad smells by using regression analysis on 10 programs of training data and are validated by 2 test data programs. After that, thresholds for using these metrics to detect bad smell and non bad smell java code are defined and validated by the previous 10 training data programs and 2 test data sets respectively. Usage of these metrics with their thresholds is then proposed in 2 ways: either use a single metric to detect bad smell or use several metrics which are ORed together. The suggestion is the latter way can detect bad smell more effectively.

Result of usage of these thresholds shows that the thresholds for Data Class, Large Class, Lazy Class, Long Method, Long Parameter List and Switch Statement can really be used to detect bad smells. On the contrary, the thresholds for Feature Envy and Refused Bequest cannot be defined because of insufficient data for experiment. Furthermore, this research designs object-oriented metrics for detecting 2 types of bad smells, Data Class and Refused Bequest. Finally, an automated tool for measuring these metrics and detecting bad smells for java code is developed.

Department..Computer Engineering……..Student's signature.....

Field of study..Software Engineering…….Advisor's signature.....

Academic year..2004..

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจาก ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ขอกราบขอบพระคุณอาจารย์ท่านที่ได้ให้คำแนะนำ และข้อเสนอแนะต่างๆ ตลอดเวลาระยะของการจัดทำวิทยานิพนธ์ของข้าพเจ้าอย่างดียิ่ง จนสำเร็จลุล่วงได้ด้วยดี

ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ เป็นประธานกรรมการ ผู้ช่วยศาตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา และอาจารย์นครทิพย์ พร้อมพูล เป็นกรรมการสอบวิทยานิพนธ์ ซึ่งได้สละเวลาและให้คำแนะนำต่างๆ ในการสอบวิทยานิพนธ์ของข้าพเจ้าอย่างดียิ่ง

ขอขอบคุณ คุณธิษณา เพียรเลิศ คุณเมทินี เขียวกันยะ และคุณธีรเดช แซ่ตั้น ที่ให้คำแนะนำ ปรึกษาและความช่วยเหลือในทุกๆ ด้าน สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา น้องชาย และเพื่อนๆ ทุกคนที่คอยเป็นกำลังใจและให้ความสนับสนุนมาโดยตลอด

สารบัญ

บทที่ หน้า

บทที่ หน้า

สารบัญตาราง

ตาราง หน้า

ตาราง หน้า

สารบัญภาพ

ภาพประกอบ หน้า

ภาพประกอบ หน้า

ภาพประกอบ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการบำรุงรักษาซอฟต์แวร์ (Software Maintainance) เป็นกระบวนการหนึ่งในการพัฒนาซอฟต์แวร์ที่มีความสำคัญ ซึ่งขั้นตอนในการบำรุงรักษาซอฟต์แวร์ประกอบด้วย การแก้ไขข้อผิดพลาดที่เกิดขึ้น (Corrective maintenance) การเพิ่มความสามารถในการบำรุงรักษาซอฟต์แวร์ให้มีประสิทธิภาพดีเยี่ยม (Perfective maintenance) การปรับเปลี่ยนซอฟต์แวร์ให้เข้ากับสภาพแวดล้อมใหม่ (Adaptive maintenance) และการป้องกันการเกิดข้อผิดพลาดในอนาคต (Preventive maintenance) หรือวิศวกรรมทำซ้ำซอฟต์แวร์ (Reengineering) ซึ่งการพัฒนาซอฟต์แวร์ที่มีการป้องกันการเกิดข้อผิดพลาดในอนาคตจะทำให้คุณภาพของซอฟต์แวร์สูงขึ้น และมีค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์ลดลง และวิธีหนึ่งของการแก้ไข คือ รีแฟคทอริง (Refactoring)

รีแฟคทอริงเป็นการเปลี่ยนแปลงโครงสร้างภายในซอฟต์แวร์ที่ไม่ส่งผลกระทบทำให้พฤติกรรมซอฟต์แวร์เปลี่ยนไป [1] ซึ่งในกระบวนการรีแฟคทอริง (Refactoring Process) มีขั้นตอนที่สำคัญคือ การตรวจจับร่องรอยที่ไม่ดี (Bad-smells detection) [2] ทำให้ทราบว่าเกิดร่องรอยที่ไม่ดีอะไร ซึ่งร่องรอยที่ไม่ดี (Bad Smells) คือ การออกแบบและพัฒนาโปรแกรมที่ไม่ดีทำให้ยากต่อการแก้ไข ปรับปรุง และ บำรุงรักษาซอฟต์แวร์ ดังนั้นการประยุกต์ใช้รีแฟคทอริงที่เหมาะสมจะทำให้ซอฟต์แวร์ทำงานได้อย่างมีประสิทธิภาพ และมีคุณภาพทางด้านต่างๆ ดีขึ้น เช่น การทำความเข้าใจ (Understandability) การบำรุงรักษาซอฟต์แวร์ (Maintainability) และการเปลี่ยนแปลง (Modifiability) เป็นต้น [3, 4] ตัวอย่างร่องรอยที่ไม่ดี ได้แก่ โครงสร้างโปรแกรมที่เหมือนกันมากกว่า 1 ที่ (Duplicated code) ทำให้การทำความเข้าใจยากและมีการทำงานที่ซ้ำซ้อน สามารถแก้ปัญหาโดยย้ายส่วนที่ซ้ำกันของโปรแกรมมาสร้างเป็นเมทธอดใหม่ซึ่งวิธีนี้เรียกว่าวิธี Extract Methods เป็นต้น และการตรวจจับร่องรอยที่ไม่ดีเหล่านี้สามารถทำให้ง่าย รวดเร็วและสะดวกได้นั้นควรมีเครื่องมือช่วยในการตรวจจับร่องรอยที่ไม่ดีเนื่องจากซอฟต์แวร์ในปัจจุบันมีขนาดใหญ่และมีความซับซ้อนมาก [2]

ในงานวิจัยนี้มีงานวิจัยสำคัญที่เกี่ยวข้องคืองานวิจัยของ ธิษณา เพียรเลิศ และ พรศิริ หมื่นไชยศรี [5] โดยงานวิจัยของ [5] นี้ได้ตรวจจับร่องรอยที่ไม่ดีจากโปรแกรม 6 แบบคือ Feature envy, Large class, Lazy class, Long method, Long parameter list และ Switch statement ด้วยมาตรวัดที่ออกแบบซึ่งใช้พิจารณาว่าเกิดร่องรอยที่ไม่ดีหรือไม่ พร้อมทั้งมีการแนะนำว่าควรใช้วิธีรีแฟคทอริงใดในการแก้ไขร่องรอยที่ไม่ดี ในการออกแบบมาตรวัดที่ช่วยในการ

ตรวจจับร่องรอยที่ไม่ดีมีเทคนิคการตรวจจับร่อยที่ไม่ดีของแต่ละแบบประกอบด้วย นิยามของร่องรอยที่ไม่ดี (Definition) กำหนดแรงจูงใจ (Motivation) กำหนดวิธีการวัด (Strategy) การออกแบบมาตรวัด (Metrics) ข้อกำหนดของค่าที่พิจารณา (Specification of Outliers) การประยุกต์ใช้รีแฟคทอริง (Application of Refactoring) และมีการประเมินความสามารถด้านการบำรุงรักษาซอฟต์แวร์ของการตรวจจับร่องรอยที่ไม่ดีโดยการเปรียบเทียบค่าของมาตรวัด การเกาะกันเป็นก้อน (Cohesion) และความซับซ้อน (Complexity) ของซอฟต์แวร์ ก่อนและหลังการทำรีแฟคทอริง

เนื่องจากในงานวิจัย [5] มีขั้นตอนข้อกำหนดของค่าที่พิจารณาในเทคนิคการตรวจจับร่องรอยที่ไม่ดีโดยกำหนดช่วงของค่าของมาตรวัด ถ้าค่าที่คำนวณได้มีค่าอยู่ในช่วงนี้อาจเป็นร่องรอยที่ไม่ดี แต่งานวิจัยนี้ได้นำเกณฑ์ช่วงค่ามาตรวัดของภาษาซีพลัสพลัส [6] สำหรับตรวจจับร่องรอยที่ไม่ดีมาใช้ตรวจจับร่องรอยที่ไม่ดีของโปรแกรมที่เขียนด้วยภาษาจาวา ดังนั้นวิทยานิพนธ์นี้ได้มุ่งเน้นการหาช่วงของค่ามาตรวัดซอฟแวร์เชิงวัตถุโดยใช้เกณฑ์ช่วงของค่ามาตรวัดของภาษาจาวาเพื่อช่วยในการตรวจจับร่องรอยที่ไม่ดีสำหรับโปรแกรมภาษาจาวา ซึ่งวิธีการหาช่วงของค่ามาตรวัดนั้นทำได้โดยเริ่มจากหาโมเดลในการทำนายร่องรอยที่ไม่ดีของมาตรวัดต่างๆ ด้วยการวิเคราะห์ความถดถอยระหว่างมาตรวัดต่างๆ และค่าร่องรอยที่ไม่ดี เพื่อให้ทราบความสัมพันธ์ระหว่างมาตรวัดต่างๆ กับร่องรอยที่ไม่ดี และใช้เลือกมาตรวัดที่สามารถตรวจจับร่องรอยที่ไม่ดีได้ หลังจากนั้นนำมาตรวัดต่างๆ มาหาช่วงของค่าที่เกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดีด้วยการพิจารณาจากค่าต่ำสุดและค่าสูงสุดกับกราฟ แล้วประเมินความน่าเชื่อของช่วงค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีถ้ามีเปอร์เซ็นต์ความถูกต้องมากสามารถใช้ช่วงของค่าของมาตรวัดนั้น แต่ถ้ามีเปอร์เซ็นต์ความถูกต้องน้อยต้องทำการปรับช่วงของค่าใหม่แล้วทำการประเมินความน่าเชื่อถือของช่วงค่ามาตรวัดอีกครั้งเพื่อตรวจสอบความถูกต้องของช่วงของค่ามาตรวัด พร้อมเสนอวิธีการนำช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน

นอกจากนี้เนื่องจาก มาตรวัดที่มีอยู่ในปัจจุบันยังค้นหาร่องรอยที่ไม่ดีได้เพียง 6 แบบ ดังนั้นผู้ทำวิทยานิพนธ์จึงออกแบบมาตรวัดซอฟต์แวร์เชิงวัตถุเพิ่มเติมเพื่อนำไปตรวจจับร่องรอยที่ไม่ดีแบบอื่นๆ รวมทั้งพัฒนาเครื่องมือในการคำนวณมาตรวัดและสำหรับตรวจจับร่องรอยที่ไม่ดีสำหรับโปรแกรมภาษาจาวาที่สามารถตรวจจับร่องรอยที่ไม่ดีได้หลายแบบมากขึ้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อหาช่วงของค่ามาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดี

1.2.2 เพื่อออกแบบมาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับรีแฟคทอริงในกระบวนการตรวจจับร่องรอยที่ไม่ดีเพิ่มเติม ได้แก่ Data Class และ Refused Bequest

1.2.3 เพื่อพัฒนาเครื่องมือสำหรับตรวจจับร่องรอยที่ไม่ดีสำหรับโปรแกรมภาษาจาวา โดยใช้ช่วงค่าของมาตรวัดที่ได้จากงานวิจัยนี้บอกว่าเกิดร่องรอยที่ไม่ดีหรือไม่ ประเภทใดและแบบไหน

1.3 ขอบเขตการวิจัย

1.3.1. เลือกร่องรอยที่ไม่ดีอย่างน้อย 2 แบบ (Data Class และ Refused Bequest) ที่นำมาออกแบบมาตรวัดซอฟต์แวร์เชิงวัตถุ

1.3.2. โปรแกรมต้นฉบับที่จะถูกนำมาตรวจจับร่องรอยที่ไม่ดีนั้น ต้องเป็นโปรแกรมภาษาจาวา และคอมไพล์ผ่าน

1.3.3. ซอฟต์แวร์ที่ใช้ในงานวิจัยนี้เขียนด้วยภาษาจาวาจำนวน 12 โปรแกรม แบ่งเป็นชุดข้อมูลสอน 10 โปรแกรมและชุดข้อมูลทดสอบ 2 โปรแกรม โดยมีขนาดของซอฟต์แวร์ต้องประกอบด้วยจำนวนคลาสอย่างน้อย 5 คลาส

1.3.4. พัฒนาเครื่องมือและใช้เครื่องมือบนระบบปฏิบัติการวินโดวส์ (Windows) ตั้งแต่รุ่น 98 ขึ้นไป

1.3.5. ใช้โปรแกรมเอสพีเอสเอสใช้ในการวิเคราะห์ทางสถิติ

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1. ศึกษาร่องรอยที่ไม่ดีจากงานวิจัยต่างๆ ในเรื่องปัญหา ผลกระทบที่เกิดจากร่องรอยที่ไม่ดี

1.4.2. ศึกษามาตรวัดซอฟต์แวร์เชิงวัตถุและวิธีการหาร่องรอยที่ไม่ดี

1.4.3. ออกแบบการหามาตรวัดซอฟต์แวร์เชิงวัตถุที่ใช้ในการหาร่องรอยที่ไม่ดี

1.4.4. พัฒนาเครื่องมือด้วยภาษาจาวา เพื่อคำนวณหาค่ามาตรวัดซอฟต์แวร์เชิงวัตถุที่ได้ออกแบบในขั้นตอน 1.4.2 และ 1.4.3

1.4.5. ออกแบบการทดลองเพื่อหามาตรวัดซอฟต์แวร์เชิงวัตถุที่มีความสัมพันธ์ต่อโมเดลการทำนายร่องรอยที่ไม่ดี

1.4.6. เก็บรวบรวมข้อมูลที่ได้จากการทดลอง

1.4.7. วิเคราะห์ข้อมูลและสรุปผลการทดลอง

1.4.8. ออกแบบการทดลองเพื่อหาช่วงค่าของแต่ละมาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับตรวจจับร่องรอยที่ไม่ดี

1.4.9. เก็บรวบรวมข้อมูลที่ได้จากการทดลอง

1.4.10. วิเคราะห์ข้อมูลและสรุปผลการทดลอง

1.4.11. จัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1. ได้ช่วงค่าของแต่ละมาตรวัดที่สามารถบอกว่าเกิดร่องรอยที่ไม่ดีหรือไม่ของซอฟต์แวร์

1.5.2. ได้มาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับรีแฟคทอริง ที่นำไปใช้ตรวจจับร่องรอยที่ไม่ดี

1.5.3. ได้เครื่องมือที่จัดทำขึ้นเพื่อตรวจจับร่องรอยที่ไม่ดีสำหรับโปรแกรมภาษาจาวา

1.5.4. สามารถนำมาตรวัดที่ได้ไปสนับสนุนการทำรีแฟคทอริงแบบอัตโนมัติต่อไป

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องในงานวิจัยช่วงของค่ามาตรวัดซอฟต์แวร์เชิงวัตถุ สำหรับตรวจจับร่องรอยที่ไม่ดีของโปรแกรมได้แก่ รีแฟคทอริง การวัดซอฟต์แวร์เชิงวัตถุ และสถิติสำหรับการวิจัย ซึ่งมีรายละเอียดดังต่อไปนี้

2.1.1 รีแฟคทอริง [1]

รีแฟคทอริง หมายถึง การเปลี่ยนแปลงโครงสร้างภายในซอฟต์แวร์ที่ไม่ส่งผลกระทบที่ทำให้พฤติกรรมของซอฟต์แวร์เปลี่ยนไป เพื่อทำให้ซอฟต์แวร์นั้นง่ายต่อการทำความเข้าใจ เพิ่มความสามารถในการบำรุงรักษาซอฟต์แวร์ ง่ายต่อการค้นพบข้อผิดพลาดที่ซ่อนอยู่ภายในโปรแกรม (Bugs) และช่วยให้โปรแกรมทำงานได้ดีขึ้น วิธีการในการรีแฟคทอริงประกอบด้วย 72 วิธีการ ซึ่งการเลือกประเภทวิธีการ และระบุโครงสร้างภายในที่ควรทำรีแฟคทอริงสามารถทำได้จากการพิจารณาโดยใช้วิจารณญาณส่วนตัว (Subjective perception) และพิจารณาจากร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี หมายถึง โมเดลการออกแบบซอฟต์แวร์หรือโปรแกรมที่ไม่ดี จึงทำให้ซอฟต์แวร์ทำงานได้ไม่มีประสิทธิภาพ เป็นผลให้ต้องทำการรีแฟคทอริง เพื่อให้ซอฟต์แวร์ทำงานได้มีประสิทธิภาพดีขึ้น ใน [1] มีร่องรอยที่ไม่ดี 22 แบบซึ่งแบ่งตามประเภทของร่องรอยที่ไม่ดีได้ 7 ประเภท [4] แสดงรายละเอียดได้ดังนี้

1. Bloaters

คือ บางส่วนในโปรแกรมเช่น คลาส หรือเมทธอด เป็นต้น สามารถเพิ่มเติมให้มีขนาดใหญ่ได้แต่ไม่สามารถจัดการได้อย่างมีประสิทธิภาพ ส่งผลให้ทำความเข้าใจ บำรุงรักษาและแก้ไขได้ยาก โดยร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Long Method คือ เมทธอดที่มีขนาดใหญ่ จำนวนพารามิเตอร์มากและตัวแปรชั่วคราวมาก ส่งผลให้ยากต่อการทำความเข้าใจและการบำรุงรักษา

b) Large Class คือ คลาสที่มีหน้าที่การทำงานมาก และตัวแปรอินสแตนท์จำนวนมาก อาจทำให้เกิดส่วนของโปรแกรมที่ซ้ำกันได้ ส่งผลให้โปรแกรมมีความซับซ้อนและทำให้การบำรุงรักษายากมากขึ้น

c) Primitive Obsession คือการเรียกใช้ข้อมูล ชนิดเรคคอร์ด (Record type) และ ชนิดพื้นฐาน (Primitive type) จำนวนมากส่งผลให้คลาสมีขนาดใหญ่ ซึ่งในการโปรแกรม

ทั่วไป มีข้อมูลอยู่ 2 ประเภทคือ ชนิดเรคคอร์ด ใช้สร้างข้อมูลเป็นกลุ่มที่มีความหมาย เช่น ตารางในฐานข้อมูล และชนิดพื้นฐาน โดยในภาษาจาวาข้อมูลชนิดพื้นฐานคือ ตัวเลข แต่ตัวอักษร (String) และวันที่ (Date) เป็นคลาส

d) Long Parameter List คือ การส่งผ่านทุกๆ ค่าที่เมทธอดต้องการใช้ไปยังเมทธอดนั้น โดยผ่านค่าพารามิเตอร์ การส่งผ่านพารามิเตอร์มากทำให้การทำความเข้าใจยากขึ้น เพราะทำให้เกิดความไม่สอดคล้องกันและใช้งานยาก

e) Data Clumps คือ การใช้กลุ่มข้อมูลเดียวกันในหลายๆ ที่ภายในโปรแกรม ส่งผลให้โปรแกรมมีความซับซ้อน โดยกลุ่มของข้อมูลประกอบด้วย คุณลักษณะภายในคลาส หรือพารามิเตอร์ในหลายๆ เมทธอด กลุ่มของข้อมูล

2. Object-Orientation Abusers

คือ การใช้การออกแบบเชิงวัตถุช่วยในการแก้ปัญหาแต่ใช้ไม่เต็มความสามารถ ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Switch Statements คือ การปรากฏของสวิทซ์สเตทเมนท์เหมือนกันหลายๆ ที่ในโปรแกรม ส่งผลให้เกิดความซับซ้อนในโปรแกรมจึงใช้โพลีมอร์ฟิซึมช่วยในการแก้ปัญหา

b) Temporary Field คือ การที่พบออบเจ็คที่มีตัวแปรที่ใช้สำหรับเก็บค่าผลลัพธ์ชั่วคราวในโปรแกรมจำนวนมาก ส่งผลให้เกิดความซับซ้อนในโปรแกรมและทำความเข้าใจยาก

c) Refused Bequest คือ คุณสมบัติหรือเมทธอดที่สืบทอดจากคลาสแม่ แต่คลาสลูกไม่ใช้ ส่งผลให้ความหมายของการสืบทอดผิด

d) Alternative Classes with different Interfaces คือ การที่มีเมทธอดที่ทำหน้าที่เดียวกันแต่มีส่วนซิกเนเจอร์ต่างกัน (Signature) ส่งผลให้ทำความเข้าใจยาก

e) Parallel Inheritance Hierarchies คือ เมื่อสร้างคลาสลูกของคลาสหนึ่งจะทำให้เป็นคลาสลูกอีกคลาสหนึ่งด้วย ส่งผลให้ความหมายของการสืบทอดผิด

3. Change Preventers

คือ โครงสร้างของโปรแกรมซอฟต์แวร์ที่สามารถแก้ไข หรือเปลี่ยนแปลงได้ยาก หรือมีการออกแบบที่ไม่ดี ส่งผลให้แก้ไขได้ยากต่อการเปลี่ยนแปลง ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Divergent Change คือ เมื่อมีการเปลี่ยนแปลงความต้องการซอฟต์แวร์ 1 อย่างแล้วต้องทำการแก้ไขได้หลายที่ในโปรแกรม เช่น ต้องทำการแก้ไข 3 เมทธอดเมื่อมีฐานข้อมูล

ใหม่ เป็นต้น ดังนั้น เมื่อมีการเพิ่มฟังก์ชันใหม่ การแก้ไขเพื่อรองรับการเปลี่ยนแปลงนี้ควรทำที่คลาสเดียว และมีการเขียนอธิบายการเปลี่ยนแปลงการทำงานของฟังก์ชันใหม่ในคลาสด้วย

b) Shotgun Surgery ตรงข้ามกับ Divergent Change คือ เมื่อมีการแก้ไขในโปรแกรม 1 ที่จะมีการเปลี่ยนแปลงความต้องการซอฟต์แวร์หลายอย่าง

4. Dispensables

คือ บางส่วนของโปรแกรมที่ไม่ได้ใช้งานควรเอาออกจากโปรแกรม เนื่องจากผู้พัฒนาต้องเสียเวลาพัฒนาและค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์ ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Lazy Class คือ คลาสที่ไม่ได้มีหน้าที่การทำงานที่สำคัญมาก ส่งผลให้ผู้พัฒนาต้องเสียเวลาพัฒนาและค่าใช้จ่ายในการบำรุงรักษาซอฟต์แวร์ ดังนั้นควรกำจัด Lazy Class ออก

b) Data Class คือ คลาสที่มีการทำงานหลักคือ การกำหนดค่าให้กับข้อมูล และภายในคลาสมีเพียงเอคเซสเซอร์เมทธอด (ได้แก่เมทธอดที่กำหนดคุณลักษณะและเมทธอดที่เรียกใช้คุณลักษณะ) เท่านั้น

c) Duplicated Code คือ โครงสร้างโปรแกรมที่เหมือนกันมากกว่า 1 ที่ ส่งผลให้เกิดความซับซ้อนของโปรแกรมและทำความเข้าใจยาก ดังนั้นถ้าสามารถหาวิธีการที่ทำให้โครงสร้างโปรแกรมที่ซ้ำกันนั้นหายไปได้ โปรแกรมจะมีประสิทธิภาพการทำงานมากขึ้น

d) Speculative Generality คือ มีการสร้างคลาสหรือเมทธอดทำหน้าที่เฉพาะกรณีพิเศษ ทำให้การทำความเข้าใจและการบำรุงรักษาซอฟต์แวร์ยากขึ้น เช่น มีการสร้างคลาสหรือเมทธอดขึ้นมาเป็นกรณีทดสอบ

5. Encapsulators

คือ การลดร่องรอยที่ไม่ดีแบบหนึ่งทำให้เกิดร่องรอยที่ไม่ดีแบบอื่น ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Message Chain คือ การเรียกใช้ข้อมูลของออบเจ็คอื่นเป็นทอดๆ และไม่ได้เรียกใช้ออบเจ็คนั้นโดยตรง

b) Middle Man คือ คลาสที่ทำหน้าที่เป็นเดลีเกท (Delegate) แต่ไม่ค่อยได้ใช้งานโดยสิ่งที่เป็นลักษณะสำคัญของออบเจ็คอย่างหนึ่งคือการห่อหุ้ม (Encapsulation) เป็นกลไกสำหรับกำหนดการมองเห็นสมาชิกภายในหน่วยที่สร้างขึ้น คือกำหนดให้สมาชิกแต่ละตัวถูกอ้างถึงจากภายนอกได้หรือไม่ ซึ่งจะมาพร้อมกับเดลีเกชัน (Delegation) เช่นผู้จัดการถามเลขาถึงเวลา

ว่าง เลขาจะส่งข้อความไปยังบันทึกประจำวัน เพื่อตรวจสอบข้อมูล ในกรณีนี้ Middle Man หมายถึงเลขาเอง

6. Couplers

คือ ร่องรอยที่ไม่ดีที่มีการ Coupling สูงเป็นการขัดแย้ง การออกแบบในเชิงวัตถุ ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Feature Envy คือ การที่มีเมทธอด หรือคุณลักษณะไปเรียกใช้หรือถูกใช้โดยคลาสอื่นมากกว่าคลาสที่เป็นเจ้าของ ส่งผลให้ขาดคุณสมบัติของการห่อหุ้มซึ่งเป็นการขัดแย้งกับการออกแบบเชิงวัตถุ

b) Inappropriate Intimacy คือ การที่พบว่ามีการพยายามเรียกใช้ส่วนไพรเวท (Private) ระหว่างคลาส

7. Others

คือ ร่องรอยที่ไม่ดีที่ไม่ได้อยู่ในประเภทที่กล่าวมาแล้ว ร่องรอยที่ไม่ดีที่อยู่ในประเภทนี้ ได้แก่

a) Incomplete Library Class คือ การที่มีคลาสไลบารี่ (Library Class) ที่ไม่สมบูรณ์ทำให้การทำความเข้าใจคลาส เพื่อนำกลับมาใช้ใหม่ทำได้ยากขึ้น

b) Comments คือ การเขียนคำอธิบาย ซึ่งควรเขียนคำอธิบายการทำงานในแต่ละส่วน เพื่อทำให้การทำความเข้าใจง่ายขึ้น

ยกตัวอย่างเช่น Dispensables เป็นร่องรอยที่ไม่ดีประเภทหนึ่งที่แสดงถึงส่วนที่ไม่จำเป็นที่อยู่ในโปรแกรมควรจะเอาออกเช่น Data Class เป็นคลาสที่มีการทำงานหลักคือการกำหนดค่าให้กับข้อมูล และภายในคลาสมีเพียงเอคเซสเซอร์เมทธอดเท่านั้น ก่อให้เกิดปัญหาเช่น เมทธอดที่กำหนดคุณลักษณะ ซึ่งถูกเรียกใช้ภายในคลาสเพียงแค่ครั้งเดียวเท่านั้นและไม่ถูกเรียกใช้จากคลาสอื่น ทำให้เกิดเมทธอดที่ไม่จำเป็นในคลาส ส่งผลให้การทำความเข้าใจยากและเกิดความซับซ้อนโปรแกรมมากขึ้น ดังนั้นควรแก้ไขโดยใช้วิธีรีแฟคทอริง วิธี Remove setting method ซึ่งเป็นการลบเมทธอดที่กำหนดคุณลักษณะออกไป ทำให้ลดความซับซ้อนของโปรแกรมและเพิ่มความเข้าใจให้มากขึ้น โดยตารางที่ 2.1 แสดงร่องรอยที่ไม่ดีและการแก้ไขได้ด้วยวิธีรีแฟคทอริง

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Bloaters	Long Method	Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
		Replace Temp with Query	การแทนที่ตัวแปรชั่วคราวด้วยการเรียกใช้เมทธอด
		Introduce Parameter Object	การแทนที่กลุ่มพารามิเตอร์ด้วยออบเจ็ค
		Preserve Whole Object	การส่งออบเจ็คแทนการส่งผ่านค่าพารามิเตอร์ทั้งหมดไปยังการเรียกใช้เมทธอด
		Replace Method with Method Object	การสร้างคลาสใหม่ขึ้นมาทำหน้าที่แทนเมทธอดเดิม
		Decompose Conditional	การย้ายโปรแกรมบางส่วนที่อยู่หลังเงื่อนไขไปสร้างเป็นเมทธอดใหม่
	Large Class	Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Bloaters	Large Class	Extract Subclass	สร้างคลาสลูกสำหรับบางส่วนของคุณสมบัติของคลาสเดิม
		Extract Interface	การย้ายบางส่วนไปสร้างเป็นอินเตอร์เฟส
		Duplicate Observed Data	การคัดลอกข้อมูลจากออบเจ็คโดเมนและสร้างออบเซิร์บเวอร์เพื่อซิงโครไนส์ (Synchronize) ข้อมูลทั้ง 2 โดเมน
	Primitive Obsession	Replace Data Value with Object	การแทนที่ค่าของข้อมูลด้วยออบเจ็ค
		Replace Type Code with Class	การแทนที่ประเภทของโปรแกรมด้วยคลาส
		Replace Type Code with Subclass	การแทนที่ประเภทของโปรแกรมด้วยคลาสลูก
		Replace Type Code with State/Strategy	การแทนที่ประเภทของโปรแกรมด้วยสเตทออบเจ็ค
		Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Bloaters	Primitive Obsession	Introduce Parameter Object	การแทนที่กลุ่มพารามิเตอร์ด้วยออบเจ็ค
		Replace Array with Object	การแทนที่อะเรย์ด้วยออบเจ็ค
	Long Parameter List	Replace Parameter with Method	การแทนที่พารามิเตอร์ด้วยเมทธอด
		Preserve Whole Object	การส่งออบเจ็คแทนการส่งผ่านค่าพารามิเตอร์ทั้งหมดไปยังการเรียกใช้เมทธอด
		Introduce Parameter Object	การแทนที่กลุ่มพารามิเตอร์ด้วยออบเจ็ค
	Data Clumps	Introduce Parameter Object	การแทนที่กลุ่มพารามิเตอร์ด้วยออบเจ็ค
		Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
		Preserve Whole Object	การส่งออบเจ็คแทนการส่งผ่านค่าพารามิเตอร์ทั้งหมดไปยังการเรียกใช้เมทธอด

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Object-Orientation Abusers	Switch Statements	Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
		Move Method	การย้ายเมทธอด
		Replace Type Code with Subclasses	การแทนที่ประเภทของโปรแกรมด้วยคลาสลูก
		Replace Type Code with State/Strategy	การแทนที่ประเภทของโปรแกรมด้วยสเตทออบเจ็ค
		Replace Conditional with Polymorphism	การแทนที่โปรแกรมส่วนที่เป็นเงื่อนไขด้วยการสร้างเมทธอดโอเวอร์ไรด์ในคลาสลูก
		Replace Parameter with Explicit Methods	การสร้างเมทธอดใหม่สำหรับโปรแกรมที่ต่างกันของพารามิเตอร์แต่ละตัว
		Introduce Null Object	การแทนที่ค่าว่างด้วยออบเจ็คว่าง
	Temporary Field	Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
		Introduce Null Object	การแทนที่ค่าว่างด้วยออบเจ็คว่าง

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Object-Orientation Abusers	Refused Bequest	Push Down Method	ย้ายเมทธอดของคลาสแม่ไปยังคลาสลูก
		Push Down Field	ย้ายแอทริบิวต์ของคลาสแม่ไปยังคลาสลูก
		Replace Inheritance with Delegation	การแทนที่การสืบทอดด้วยเดลีเกชัน
	Alternative Classes with different Interfaces	Rename Method	การเปลี่ยนชื่อเมทธอด
		Move Method	การย้ายเมทธอด
		Extract Superclass	สร้างคลาสแม่โดยย้ายคุณสมบัติของคลาสลูกไปยังคลาสแม่
	Parallel Inheritance Hierarchies	Move Method	การย้ายเมทธอด
		Move Field	การย้ายคุณลักษณะ
Change Preventers	Divergent Change	Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
	Shotgun Surgery	Move Method	การย้ายเมทธอด
		Move Field	การย้ายคุณลักษณะ
		Inline Class	การย้ายคุณสมบัติทั้งหมดไปยังคลาสอื่น

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Dispensables	Large Class	Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
		Extract Subclass	สร้างคลาสลูกสำหรับบางส่วนของคุณสมบัติของคลาสเดิม
		Extract Interface	การย้ายบางส่วนไปสร้างเป็นอินเตอร์เฟส
		Duplicate Observed Data	การคัดลอกข้อมูลจากออบเจ็คโดเมนและสร้างออบเซิร์บเวอร์เพื่อซิงโครไนส์ข้อมูลทั้ง 2 โดเมน
	Data Class	Encapsulate Field	การกำหนดเอคเซสเซอร์เมทธอดให้เป็นไพรเวท
		Encapsulate Collection	การกำหนดคอลเลคชันด้วยเมทธอด Add และ Remove
		Remove Setting Method	การลบเมทธอดที่กำหนดคุณลักษณะ
		Move Method	การย้ายเมทธอด
		Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
		Hide Method	กำหนดเมทธอดให้เป็นประเภทไพรเวท

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
	Duplicated Code	Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
		Pull Up Field	การย้ายคุณลักษณะของลูกไปยังคลาสแม่
		Form Template Method	สร้างเมทธอดที่จัดการแม่แบบของคุณลักษณะ
		Substitute Algorithm	แทนที่ส่วนของเมทธอดด้วยอัลกอริธึมใหม่
		Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
	Speculative Generality	Collapse Hierarchy	การรวมคลาสแม่และคลาสลูกเข้าด้วยกัน
		Inline Class	การย้ายคุณสมบัติทั้งหมดไปยังคลาสอื่น
		Remove Parameter	การลบพารามิเตอร์
		Rename Method	การเปลี่ยนชื่อเมทธอด
	Message Chains	Hide Delegate	กำหนดเดลิเกทให้เป็นไพรเวท
		Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Dispensables	Message Chains	Move Method	การย้ายเมทธอด
	Middle Man	Remove Middle Man	การเรียกเดลิเกทโดยตรง
		Inline Method	การย้ายคุณสมบัติทั้งหมดไปยังเมทธอดอื่น
		Replace Delegation with Inheritance	แทนที่การเดลิเกทด้วยการสืบทอด
Couplers	Feature Envy	Move Method	การย้ายเมทธอด
		Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
	Inappropriate Intimacy	Move Method	การย้ายเมทธอด
		Move Field	การย้ายคุณลักษณะ
		Change Bi-directional Association to Unidirectional	เปลี่ยนความสัมพันธ์ทางเดียวไปเป็น 2 ทาง
		Extract Class	การสร้างคลาสใหม่และย้ายคุณลักษณะและเมทธอดจากคลาสเดิมไปยังคลาสใหม่
		Hide Delegate	กำหนดเดลิเกทให้เป็นไพรเวท

ตารางที่ 2.1 การแบ่งประเภทร่องรอยที่ไม่ดีและวิธีรีแฟคทอริงสำหรับร่องรอยที่ไม่ดี (ต่อ)

ประเภทร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี	วิธีรีแฟคทอริง	คำอธิบายรีแฟคทอริง
Couplers	Inappropriate Intimacy	Replace Inheritance with Delegation	การแทนที่การสืบทอดด้วยการเดลิเกท
Others	Incomplete Library Class	Move Method	การย้ายเมทธอด
		Introduce Foreign Method	การสร้างเมทธอดในคลาสไคลเอนท์ด้วยอินสเตนท์ของคลาสเซิร์ฟเวอร์
		Introduce Local Extension	(ใช้เฉพาะใน C++)
	Comments	Extract Method	ย้ายบางส่วนของโปรแกรมมาสร้างเป็นเมทธอดใหม่
		Rename Method	การเปลี่ยนชื่อเมทธอด
		Introduce Assertion	สร้างเมทธอดตรวจสอบก่อนนำตัวแปรไปใช้

2.1.2 **การวัดซอฟต์แวร์เชิงวัตถุ** (Object-oriented software measurement)

การวัดเป็นหัวใจของทุกระบบซึ่งเป็นกระบวนการกำหนดค่าตัวเลขหรือสัญลักษณ์ให้กับคุณลักษณะ (Attributes) ของเอนทิตี (Entities) หรือสิ่งที่อยู่ในโลกของความเป็นจริงที่เราสนใจ เพื่อบรรยายเอนทิตีนั้นให้อยู่ในรูปของกฎที่นิยามไว้ชัดเจน

เอนทิตีคือวัตถุ (ได้แก่บุคคลหรือห้อง เป็นต้น) หรือเหตุการณ์ (ได้แก่การเดินทางหรือการทดสอบในช่วงโครงงานซอฟต์แวร์ เป็นต้น) ในโลกของความเป็นจริง การกำหนดลักษณะของเอนทิตีเป็นสิ่งสำคัญในการเห็นถึงความแตกต่างระหว่างเอนทิตีหนึ่งกับเอนทิตีอื่น

คุณลักษณะคือจุดเด่นหรือคุณสมบัติของเอนทิตี ตัวอย่างคุณลักษณะเช่นพื้นที่หรือสี (ของห้อง) ค่าใช้จ่าย (ของการเดินทาง) หรือช่วงเวลา (ของเฟสการทดสอบ)

คุณลักษณะของสิ่งที่สนใจสามารถแบ่งได้เป็น 2 ประเภท คือ

1. คุณลักษณะภายใน (Internal Attribute) เป็นคุณลักษณะที่สามารถหาค่าได้โดยตรงจากสิ่งที่สนใจ
2. คุณลักษณะภายนอก (External Attribute) เกิดจากการคำนวณจากคุณลักษณะภายในที่เกี่ยวข้องกับคุณลักษณะภายนอก

ประเภทของการวัดซอฟต์แวร์มี 2 ประเภท คือ

1. การวัดทางตรง (Direct measurement) เป็นการวัดเฉพาะคุณลักษณะภายในของสิ่งที่เราสนใจโดยไม่นำคุณลักษณะ หรือ เอนทิตีอื่นมาเกี่ยวข้อง เช่น การวัดความยาวของซอร์สโค้ดโปรแกรมสามารถวัดได้จากการนับจำนวนบรรทัดทั้งหมดของโปรแกรม (Line of code - LOC) เป็นต้น
2. การวัดทางอ้อม (Indirect measurement) เป็นวิธีการที่ใช้วัดคุณลักษณะภายนอก เช่น คุณภาพของซอฟต์แวร์ (Software Quality) ในด้านต่างๆ เช่น การวัดความสามารถในการใช้งาน (Usability) ความสามารถในการทดสอบ (Testability) ความสามารถในการบำรุงรักษา (Maintainability) เป็นต้น แต่การวัดคุณลักษณะภายนอกนั้นไม่สามารถหาค่าได้โดยตรง จึงต้องอาศัยการวัดทางตรงมาใช้ในการคำนวณหาค่า เช่น ความสามารถในการนำกลับมาใช้ใหม่ เป็นต้น

การวัดทั้งสองวิธีจำเป็นต้องใช้มาตรวัด เพื่อวัดคุณลักษณะของสิ่งที่เราสนใจให้ทำการวัดได้โดยง่าย มาตรวัดสามารถแบ่งได้เป็น 2 ประเภท คือ

1.มาตรวัดซอฟต์แวร์แบบดั้งเดิม (Traditional Metrics) เช่น มาตรวัดจำนวนบรรทัด (Lines of code - LOC) มาตรวัดไซโคลเมติกของแมคเคบ (Cyclomatic complexity - V(G))

2.มาตรวัดซอฟต์แวร์เชิงวัตถุ (Object Oriented Software Metrics) ของ Chidamber and Kemerer เช่น มาตรวัดจำนวนเมทธอดต่อคลาส(Weighted methods per class - WMC) มาตรวัดความรับผิดชอบของคลาส (Response for a class - RFC) มาตรวัดจำนวนคลาสลูก (Number of children - NOC) เป็นต้น

นอกจากนี้ยังมีมาตรวัดที่ได้จากงานวิจัยต่างๆ เช่นงานวิจัย [3] [7] และ [8] ซึ่งแสดงรายละเอียด ดังนี้

มาตรวัด Lorenz and Kidd [3]

มาตรวัดขนาดคลาส

1.มาตรวัด Number of Public Instance Methods in a Class (PIM) คือ จำนวนเมทธอดที่ประกาศเป็น public ภายในคลาสนั้น ซึ่งการประกาศเมทธอดเป็น public นั้น จะทำให้คลาสอื่นสามารถเข้ามาใช้งานเมทธอดนั้นได้

2.มาตรวัด Number of Instance Methods in a Class (NIM) คือ จำนวนเมทธอดที่ประกาศภายในคลาสนั้น ซึ่งการประกาศเมทธอดเป็น public, protected และ private

3.มาตรวัด Number of Instance Variable in a Class (NIV) คือ จำนวนตัวแปรที่ประกาศภายในคลาสนั้น

4.มาตรวัด Number of Class Methods in a Class (NCM) คือ จำนวนเมทธอดที่เป็นคลาสที่ประกาศภายในคลาสนั้น

5.มาตรวัด Number of Class Variables in a Class (NVC) คือ จำนวนตัวแปรที่เป็นคลาสที่ประกาศอยู่ในคลาสนั้น

มาตรวัดการสืบทอดคลาส

1.มาตรวัด Number of Methods Overridden (NMO) คือ จำนวนเมทธอดที่ทำการ Override โดยคลาสลูก

2.มาตรวัด Number of Methods Inherited (NMI) คือ จำนวนเมทธอดที่ถูก Inherit โดยคลาสลูก

3.มาตรวัด Number of Methods Added (NMA) คือ จำนวนเมทธอดที่สร้างขึ้นมาใหม่ภายในคลาสลูก

4.มาตรวัด Specialization Index (SIX) คือ มาตรวัดที่เกิดจากการคำนวณจำนวนเมทธอดที่ถูก Override คูณด้วยจำนวน Hierarchy หารด้วยจำนวนเมทธอดทั้งหมด

5.มาตรวัด Average Parameters Per Method (APPM) คือ อัตราส่วนระหว่างจำนวนพารามิเตอร์ในทุกๆ เมทธอดกับจำนวนเมทธอดทั้งหมด

มาตรวัด Brito e Abreu and Melo [3]

1.มาตรวัด Method Hide Factor (MHF) คือ อัตราส่วนระหว่างผลรวมของเมทธอดที่ประกาศเป็น Private กับจำนวนเมทธอดทั้งหมดในระบบ

2.มาตรวัด Attribute Hiding Factor (AHF) คือ อัตราส่วนระหว่างผลรวมของแอทริบิวต์ที่เป็นไพรเวทกับผลรวมของแอทริบิวต์ทั้งหมดในระบบ

3.มาตรวัด Method Inheritance Factor (MIF) คือ อัตราส่วนระหว่างผลรวมของเมทธอดที่ Inherit มาทั้งหมดในทุกๆ คลาสกับจำนวนเมทธอดทั้งหมดในทุกๆ คลาส

4.มาตรวัด Attribute Inheritance Factor (AIF) คือ อัตราส่วนระหว่างผลรวมของแอทริบิวต์ที่ Inherit มาทั้งหมดในทุกๆ คลาสกับจำนวนแอทริบิวต์ทั้งหมดในทุกๆ คลาส

5.มาตรวัด Polymorphism Factor (PF) คือ อัตราส่วนระหว่างจำนวนการพ้องรูปที่เป็นไปไม่ได้ทั้งหมดกับจำนวนการพ้องรูปของคลาสใดๆ ที่มีค่ามากที่สุด

มาตรวัดที่ได้จากงานวิจัย [7] เป็นมาตรวัดที่ตรวจจับร่องรอยที่ไม่ดีแบบ Data Class และ God Class

1.มาตรวัด Weight of a class (WOC) คือ จำนวนของเมทธอดที่ไม่ใช่เอคเซสเซอร์เมทธอดในคลาสหารด้วยจำนวนสมาชิกของอินเตอร์เฟสทั้งหมด โดยไม่รวมสมาชิกที่สืบทอดมา (Inherit)

2.มาตรวัด Number Of Public Attribute (NOPA) คือ จำนวนคุณลักษณะที่ไม่ได้สืบทอดมา แต่มีการประกาศในอินเตอร์เฟสคลาส

3.มาตรวัด Number Of Accessor Method (NOAM) คือ จำนวนเอคเซสเซอร์ที่ไม่ได้ถูกสืบทอดมา แต่มีการประกาศอยู่ในอินเตอร์เฟสของคลาส

4.มาตรวัด Access of Foreign Data (AOFD) คือ จำนวนคลาสภายนอกที่ยอมให้คลาสนั้นเข้าถึงข้อมูล ทั้งเข้าถึงโดยตรงและผ่านเอคเซสเซอร์เมทธอด โดยไม่รวมถึงอินเนอร์คลาส (Inner class) และคลาสบรรพบุรุษ (Super class)

5.มาตรวัด Weight Method Count (WMC) คือ ผลรวมของการคำนวณค่าความซับซ้อนของมาตรวัดไซโคลเมติกของแมคเคบของทุกเมทธอดภายในแต่ละคลาส ในกรณีไม่คำนึงถึงค่าความซับซ้อนแล้ว WMC จะคำนวณได้จากค่าจำนวนของเมทธอดในคลาส

6.มาตรวัด Tight Class Cohesion (TCC) คือ จำนวนความสัมพันธ์ของเมทธอดที่สื่อสารกันโดยตรง

มาตรวัด Chidamber and Kemerer [8] เป็นมาตรวัดความสามารถในการบำรุงรักษาซอฟต์แวร์

1.มาตรวัดจำนวนเมทธอดต่อคลาส (weighted methods per class - WMC) คือ ผลรวมของการคำนวณค่าความซับซ้อนของมาตรวัดไซโคลเมติกของแมคเคบทุกเมทธอดภายในแต่ละคลาส

2.มาตรวัดความรับผิดชอบต่อคลาส (Response for a class - RFC) คือ จำนวนเมทธอดที่สามารถตอบสนองกับการเรียกใช้งานที่ได้รับมาจากคลาสอื่น หรือจากเมทธอดบางตัวภายในคลาส

3.มาตรวัดระดับความลึกของการสืบทอดคุณสมบัติของคลาส (Depth of inheritance hierarchy - DIT) คือจำนวนของระดับชั้น (Level) การสืบทอดคุณสมบัติของแต่ละคลาสที่พิจารณา

4.มาตรวัดจำนวนคลาสลูก (Number of Children - NOC) คือ การนับจำนวนคลาสลูกทั้งหมดที่ทำการสืบทอดมาจากคลาสแม่ ซึ่งเป็นคลาสที่พิจารณาอยู่ในขณะนั้น

5.มาตรวัดการเข้าคู่กันระหว่างวัตถุ (Coupling between objects - CBO) คือ การแสดงความสัมพันธ์ระหว่างวัตถุ เมื่อมีการเรียกใช้ตัวแปรหรือเมทธอดระหว่างกัน

6.มาตรวัดระดับของการขาดการเกาะกันเป็นก้อนของเมทธอดภายในคลาส (Lack of cohesion of methods - LCOM) การเกาะกันเป็นก้อน (Cohesion) ของคลาส คือ การที่เมทธอดนั้น มีความสัมพันธ์กับเมทธอดอื่นอย่างไร การที่มีค่าของการเกาะกันเป็นก้อนสูง (การขาดการเกาะกันเป็นก้อนต่ำ (Low lack of cohesion)) แสดงว่าคลาสนั้นมีการออกแบบที่ดี

2.1.3 **สถิติสำหรับการวิจัย** (Statistics for Research) [9 -11]

สถิติหมายถึงศาสตร์ที่ว่าด้วยการเก็บรวบรวมข้อมูล การนำเสนอข้อมูลและการวิเคราะห์ข้อมูล และวิธีการทางสถิติได้นำมาใช้ในการวิเคราะห์ข้อมูลดิบเหล่านั้น โดยใช้วิธีต่างๆ เช่น การแจกแจงความถี่ การหาร้อยละ ตลอดจนถึงช่วยให้ทราบเกี่ยวกับคุณลักษณะต่างๆ ของข้อมูล เช่น การวัดแนวโน้มเข้าสู่ศูนย์กลาง การวัดการกระจาย ข้อมูล และสามารถนำเสนอรายงานการวิจัย เช่น การจัดทำตาราง การสร้างกราฟเส้น และแผนภูมิรูปภาพ

งานวิจัยนี้ได้นำวิธีการทางสถิติมาช่วยในการทำงานวิจัยและวิเคราะห์ข้อมูล ซึ่งวิธีการทางสถิติที่นำมาใช้ประกอบด้วย การหาค่าเฉลี่ย (Mean) การหาค่าส่วนเบี่ยงเบนมาตรฐาน (Standard Deviation) การวิเคราะห์ความสัมพันธ์ (Correlation Analysis) การวิเคราะห์ความถดถอย (Regression analysis) การประมาณค่าแบบช่วง (Interval Estimation) ระดับความเชื่อมั่น (Level of Confidence) และ การประมาณค่าเฉลี่ยประชากรแบบช่วง โดยมีรายละเอียดดังนี้

2.1.3.1. **การหาค่าเฉลี่ย**

การหาค่าเฉลี่ย เป็นวิธีการวัดแนวโน้มสู่ส่วนกลาง (Central Tendency) วิธีหนึ่งซึ่งเป็นการคำนวณค่ากลางของข้อมูลว่าอยู่ที่ใด โดยจะใช้ค่าเฉลี่ยเป็นตัวแทนของข้อมูลที่นำมาคำนวณ สูตรในการคำนวณหาค่าเฉลี่ย คือ

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + ... + x_n}{n}$$

โดยที่ $\bar{x}$ คือ ค่าเฉลี่ยของข้อมูลชุดใดๆ

$x_1, x_2, ..., x_n$ คือ ค่าของข้อมูลในชุดที่ 1, 2, … , n

n คือ จำนวนชุดข้อมูลทั้งหมด

2.1.3.2 **การหาค่าส่วนเบี่ยงเบนมาตรฐาน**

การหาค่าส่วนเบี่ยงเบนมาตรฐานใช้สำหรับการพิจารณาหรือสรุปถึงลักษณะของข้อมูลโดยใช้ค่ากลางหรือค่าเฉลี่ยเพียงอย่างเดียว อาจทำให้ไม่ทราบถึงลักษณะของข้อมูลได้ชัดเจน เนื่องจากอาจมีข้อมูลที่มีค่ากลางเท่ากัน แต่ลักษณะของข้อมูลแตกต่างกัน นั่นคือมีการกระจายข้อมูลไม่เหมือนกัน ดังนั้นค่าส่วนเบี่ยงเบนมาตรฐาน เป็นวิธีการวัดการกระจายวิธีหนึ่งที่นิยมใช้ ซึ่งมีสูตรในการคำนวณ คือ

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

โดยที่ SD คือ ค่าเบี่ยงเบนมาตรฐาน

$\bar{x}$ คือ ค่าเฉลี่ยของข้อมูลทั้งหมด

x_i คือ ค่าข้อมูลในชุดที่ i เมื่อ $1 \leq i \leq n$

n คือ จำนวนชุดข้อมูลทั้งหมด

2.1.3.3 **การวิเคราะห์ความสัมพันธ์**

สำหรับวิธีการวิเคราะห์ความสัมพันธ์มีหลายวิธีขึ้นอยู่กับลักษณะของข้อมูล เช่น การหาความสัมพันธ์ของข้อมูลจะใช้สถิติไคสแควร์สำหรับข้อมูลนามบัญญัติ การวิเคราะห์ความสัมพันธ์จัดอันดับแบบสเปียร์แมน (Spearman rank correlation) สำหรับข้อมูลอันดับ และการวิเคราะห์ความสัมพันธ์แบบเพียร์สัน (Pearson correlation) สำหรับข้อมูลอันตรภาค เนื่องจากงานวิจัยนี้ข้อมูลมีลักษณะแบบอันตรภาค จึงใช้วิธีการวิเคราะห์ความสัมพันธ์แบบเพียร์สัน ดังนั้นจะกล่าวถึงการวิเคราะห์เพียงแบบเดียวเท่านั้น

การวิเคราะห์ความสัมพันธ์แบบเพียร์สันเป็นการหาความสัมพันธ์ระหว่าง ตัวแปรเชิงปริมาณ 2 ตัว โดยมีข้อตกลงเบื้องต้นว่าตัวแปรทั้งสองมีการแจกแจงแบบปกติ และมีความสัมพันธ์กันในแบบเชิงเส้น (Linear relationship) สำหรับสถิติที่ใช้วัดความสัมพันธ์ระหว่างตัวแปรอิสระ (x) และตัวแปรตาม (y) ว่ามากหรือน้อยนั้นจะเรียกว่า สัมประสิทธิ์สหสัมพันธ์ (Correlation Coefficient) มีสูตรการคำนวณดังนี้

$$r = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}\right)\left(\sum_{i=1}^{n} y_i^2 - \frac{(\sum_{i=1}^{n} y_i)^2}{n}\right)}}$$

โดยที่ r คือ สัมประสิทธิ์สหสัมพันธ์

x_i คือ ตัวแปรตัวที่ 1 หรือตัวแปรอิสระ (Independent Variable) ในชุดที่ i เมื่อ $1 \leq i \leq n$

y_i คือ ตัวแปรตัวที่ 2 หรือตัวแปรตาม (Dependent variable) ในชุดที่ i เมื่อ $1 \leq i \leq n$

n คือ จำนวนชุดข้อมูลทั้งหมด

ค่า r ที่ได้จากการคำนวณจะมีค่าตั้งแต่ -1 ถึง 1 ความหมายของค่า r คือ

1. ค่า r เป็นลบ แสดงว่า x และ y มีความสัมพันธ์ในทิศทางตรงข้าม คือ ถ้าค่าของ x เพิ่มค่าของ y จะลด แต่ถ้าค่าของ x ลดค่าของ y จะเพิ่ม

2. ค่า r เป็นบวกแสดงว่า x และ y มีความสัมพันธ์ในทิศทางเดียวกัน คือ ถ้าค่าของ x เพิ่มค่าของ y จะเพิ่มด้วย แต่ถ้าค่าของ x ลดค่าของ y จะลดด้วย

3. ถ้า r มีค่าเข้าใกล้ 1 หมายถึง x และ y สัมพันธ์ในทิศทางเดียวกันและมีความสัมพันธ์กันมาก

4. ถ้า r มีค่าเข้าใกล้ -1 หมายถึง x และ y สัมพันธ์ในทิศทางตรงกันข้ามและมีความสัมพันธ์กันมาก

5. ถ้า r = 0 แสดงว่า x และ y ไม่มีความสัมพันธ์กัน

6. ถ้า r เข้าใกล้ 0 แสดงว่า x และ y มีความสัมพันธ์กันน้อย

2.1.3.4 การวิเคราะห์ความถดถอย

เป็นการศึกษาถึงความสัมพันธ์ของตัวแปรตั้งแต่ 2 ตัวขึ้นไป เพื่อประมาณหรือพยากรณ์ค่าของตัวแปรหนึ่งจากตัวแปรอื่นที่มีความสัมพันธ์กัน กับตัวแปรที่ต้องการประมาณนั้น วัตถุประสงค์ของการวิเคราะห์ความถดถอย เพื่อศึกษาถึงความสัมพันธ์ระหว่างตัวแปร และใช้ความสัมพันธ์ที่วิเคราะห์ได้มาประมาณค่าหรือพยากรณ์ค่าตัวแปรตาม (y) ในอนาคตเมื่อกำหนดค่าตัวแปรอิสระ (x) ซึ่งความสัมพันธ์ระหว่างตัวแปร 2 ตัว จะแสดงอยู่ในรูปสมการเชิงเส้นดังนี้

$$y = \beta_0 + \beta_1 x + e$$

โดยที่ y คือ ตัวแปรตาม

x คือ ตัวแปรอิสระ

β_0 คือ ส่วนตัดแกน y หรือค่าของ y เมื่อ x มีค่าเป็นศูนย์

β_1 คือ ความชัน (Slope) ของเส้นตรงแสดงถึงอัตราการเปลี่ยนแปลงของ y เมื่อ x เปลี่ยนไป 1 หน่วย

e คือ ความคลาดเคลื่อนอย่างสุ่ม (Random Error)

2.1.3.5 การประมาณค่าแบบช่วง

เป็นการประมาณค่าพารามิเตอร์ของประชากรว่าอยู่ในช่วงใดช่วงหนึ่ง โดยใช้ข้อมูลตัวอย่าง และช่วงของการประมาณค่า จะบอกถึงค่าที่ต่ำสุดและค่าที่สูงสุดของพารามิเตอร์ในการประมาณค่า ค่าต่ำสุด (L) และค่าสูงสุด (U) ของช่วงนี้จะขึ้นอยู่กับระดับความเชื่อมั่นในการประมาณค่าด้วย หรือกล่าวได้ว่าช่วงของค่าประมาณจะแคบหรือกว้างขึ้นอยู่กับระดับความเชื่อมั่น และการกระจายของลักษณะของประชากรที่สนใจศึกษา ถ้าระดับความเชื่อมั่นสูง และลักษณะที่สนใจศึกษามีการกระจายมาก ช่วงของค่าประมาณจะกว้าง (L ต่ำ และ U สูง) แต่ถ้าระดับความเชื่อมั่นต่ำ และการกระจายของลักษณะที่สนใจศึกษาน้อย ช่วงของค่าประมาณจะแคบ (L และ U มีค่าใกล้เคียงกัน)

2.1.3.6 ระดับความเชื่อมั่น

ระดับความเชื่อมั่น หมายถึง โอกาสที่พารามิเตอร์ของประชากรจะอยู่ในช่วงของค่าที่ประมาณได้ เช่น $P(l < \mu < u) = 0.95$ หมายถึง โอกาสที่ค่า μ จะมีค่าอยู่ในช่วง l และ u เป็น 0.95 หรือ 95% และโอกาสที่ค่า μ จะมีค่าน้อยกว่า l หรือมีค่ามากกว่า u เป็น 0.05 หรือ 5% หรือกล่าวได้ว่า ในการสุ่มตัวอย่างขนาด n จากประชากร 100 ครั้ง (ครั้งละ n หน่วย) ค่าเฉลี่ยประชากร μ จะมีค่าอยู่ในช่วง L และ U 95 ครั้ง จะมีเพียง 5 ครั้งที่ค่า μ จะไม่อยู่ในช่วง l และ U ($\mu \leq l$ หรือ $\mu \geq u$) จึงกล่าวได้ว่า μ มีค่าในช่วง l และ u ด้วยระดับความเชื่อมั่น 95%

ในกรณีทั่วไป ระดับความเชื่อมั่นเป็น $(1 - \alpha)$ 100% หมายความว่า โอกาสที่ค่าประมาณจะผิดพลาดเท่ากับ $\alpha(100)\%$ ถ้าให้ระดับความเชื่อมั่น $(1 - \alpha)$ มีค่ามากจะทำให้โอกาสของความผิดพลาดในการประมาณค่า (α) มีค่าน้อย (โอกาสของความผิดพลาดเรียกกันอีกชื่อหนึ่งว่า ระดับนัยสำคัญ = α)

$$P(l < \mu < u) = 1 - \alpha$$

โดยที่ l คือ ขีดกำจัดความเชื่อมั่นล่าง (Lower Confidence Limit)

u คือ ขีดกำจัดความเชื่อมั่นบน (Upper Confidence Limit)

โดยทั่วๆ ไปนิยมใช้ระดับความเชื่อมั่น $(1 - \alpha)$ เป็น 0.90 0.95 และ 0.99 ในงานวิจัยนี้ใช้ระดับความเชื่อมั่น 0.95

2.1.3.7 การประมาณค่าเฉลี่ยประชากรแบบช่วง

เมื่อสุ่มตัวอย่างขนาด n จากประชากรที่มีค่าเฉลี่ย μ ค่าแปรปรวน σ^2/n ถ้าตัวอย่างมีขนาดใหญ่พอแล้ว ค่าเฉลี่ย $\bar{x}$ จะมีการแจกแจงเข้าสู่การแจกแจงแบบปกติ

จากทฤษฎีลิมิตสู่ส่วนกลางจะได้ว่า เมื่อสุ่มตัวอย่างขนาดใหญ่ ($n \geq 30$) จากประชากรที่มีการแจกแจงใดๆ (แบบสมมาตร เบ้ขวาและเบ้ซ้าย) ก็ตามค่าเฉลี่ยตัวอย่าง $\bar{x}$ จะมี

การแจกแจงเข้าสู่การแจกแจงแบบปกติด้วยค่าเฉลี่ย μ และสามารถหาค่าแปรปรวนได้ ค่าประมาณแบบช่วงของ μ ที่ระดับความเชื่อมั่น (1 - α)100% คือ

$$\bar{x} - Z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}} < \mu < \bar{x} + Z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}$$

โดยที่ $\bar{x}$ คือ ค่าเฉลี่ยของข้อมูลทั้งหมด

$Z_{1-\alpha/2}$ คือ $\frac{\bar{x}-\mu}{\sigma/\sqrt{n}}$ หรือใช้ค่า Z จากตารางการแจกแจงแบบปกติมาตรฐาน

$\frac{\sigma}{\sqrt{n}}$ คือ ค่าเบี่ยงเบนมาตรฐาน

μ คือ ค่าเฉลี่ยประชากร

2.2 งานวิจัยที่เกี่ยวข้อง

จากการศึกษาเบื้องต้นพบว่ามีงานวิจัยที่เกี่ยวข้องจำนวนทั้งสิ้น 3 งานวิจัย ซึ่งเกี่ยวกับการตรวจจับร่องรอยที่ไม่ดีแต่ละประเภทและแต่ละแบบต่างๆ โดยใช้มาตรวัดซอฟต์แวร์เชิงวัตถุ โดยมีรายละเอียดดังนี้

2.2.1 Bad-Smell Detection Using Object-Oriented Software Metrics [5]

โดย ธิษณา เพียรเลิศ และ พรศิริ หมื่นไชยศรี

งานวิจัยนี้ได้ออกแบบมาตรวัดซอฟต์แวร์เชิงวัตถุในการตรวจจับร่องรอยที่ไม่ดีประเภท Bloaters, Object-Orientation Abusers, Dispensables และ Couplers แสดงรายละเอียดในตารางที่ 2.2

ตารางที่ 2.2 ประเภทร่องรอยที่ไม่ดีที่ตรวจจับในงานวิจัยนี้

ประเภทของร่องรอยที่ไม่ดี	ร่องรอยที่ไม่ดี
Bloaters	Long method, Large Class และ Long parameter list
Object-Orientation Abusers	Switch statement
Dispensables	Lazy class
Couplers	Feature envy

พร้อมทั้งแนะนำการประยุกต์ใช้วิธีรีแฟคทอริงในการแก้ร่องรอยที่ไม่ดี ดังแสดงในตารางที่ 2.3

ตารางที่ 2.3 มาตรวัดที่ใช้ในการตรวจจับร่องรอยที่ไม่ดี 6 แบบและรีแฟคทอริงที่มาประยุกต์ใช้

ร่องรอยที่ไม่ดี	มาตรวัดที่ใช้ในการตรวจจับ	รีแฟคทอริง
Feature envy	NCDM, NCDA, NCRA	move method, extract method, move attribute
Large class	NIM, NIV, TCC	extract class, extract subclass, extract interface, duplicate observed data
Lazy class	NIM, NIV, DIT	inline class, collapse hierarchy
Long method	NOS, NOP, NOT	extract method, replace method with method object, replace temp with query, introduce parameter object, preserve whole object, decompose conditional
Long parameter list	NOP	Replace parameters with method, preserve whole object
Switch statement	NOSS	move method, replace type code with subclass, replace type code with state/strategy, replace conditional with polymorphism, replace parameter with explicit methods,Introduce null object

สำหรับเทคนิคในการตรวจจับร่องรอยที่ไม่ดีแต่ละแบบในงานวิจัยนี้ประกอบด้วย 6 ขั้นตอนย่อย ได้แก่

2.2.1.1 นิยามของร่องรอยที่ไม่ดี อธิบายคุณลักษณะของร่องรอยที่ไม่ดี

2.2.1.2 กำหนดแรงจูงใจ วิเคราะห์ถึงผลกระทบที่เกิดจากร่องรอยที่ไม่ดีกับคุณลักษณะของคุณภาพภายนอกที่มีผลต่อซอฟต์แวร์เช่น ความสามารถในการบำรุงรักษาระบบ ความสามารถในการเข้าใจระบบ เป็นต้น และรวมถึงอธิบายโครงสร้างในการออกแบบ เช่น คลาส และเมทธอด เกิดผลกระทบจากร่องรอยที่ไม่ดีอย่างไร

2.2.1.3 กำหนดวิธีการวัด อธิบายถึงวิธีการที่จะใช้หาร่องรอยที่ไม่ดีเหล่านั้น ซึ่งวิธีการนี้จะขึ้นกับลักษณะของร่องรอยที่ไม่ดีที่ได้อธิบายไว้ในขั้นตอนก่อนหน้านี้

2.2.1.4 การออกแบบมาตรวัด เลือกมาตรวัดที่จะใช้สำหรับตรวจจับร่องรอยที่ไม่ดีที่สนับสนุนการกำหนดวิธีการวัด โดยทำได้ 2 วิธี คือ การทำการปรับปรุงมาตรวัดเดิมที่มีอยู่แล้วและออกแบบมาตรวัดใหม่

2.2.1.5 ข้อกำหนดของค่าที่พิจารณา เป็นการกำหนดช่วงของค่ามาตรวัด โดยถ้าผลลัพธ์ที่คำนวณได้มีค่าอยู่ในช่วงนี้อาจเป็นร่องรอยที่ไม่ดีได้

2.2.1.6 การประยุกต์ใช้รีแฟคทอริง แนะนำการเลือกวิธีรีแฟคทอริงสำหรับแก้ปัญหาร่องรอยที่ไม่ดีตามที่ได้เลือกในขั้นตอนที่ 2.2.1.1 และอธิบายโดยการยกตัวอย่างให้เห็น

หลังจากนั้นทำการประเมินความสามารถของการทำรีแฟคทอริงโดยเปรียบเทียบค่าของมาตรวัดก่อนและหลังทางด้านความสามารถในการบำรุงรักษาซอฟต์แวร์คือ มาตรวัดการเกาะกันเป็นก้อนและความซับซ้อนของซอฟต์แวร์ ได้แก่มาตรวัด LCOM และมาตรวัด Average Method Complexity (Avg. MCX) ซึ่งค่ามาตรวัดการเกาะกันเป็นก้อนสูงและค่าความซับซ้อนต่ำจะแสดงว่ามีความสามารถของการบำรุงรักษาซอฟต์แวร์ที่ดี โดยทั้งนี้ผู้ทำวิทยานิพนธ์นี้ได้สร้างเครื่องมือในการตรวจจับร่องรอยที่ไม่ดีของภาษาจาวาด้วยมาตรวัดที่ได้ออกแบบของงานวิจัยนี้และนำเครื่องมือมาประยุกต์ใช้กับโปรแกรมตัวอย่าง เพื่อแสดงให้เห็นว่าค่ามาตรวัด LCOM และ Avg. MCX ลดลงหลังจากทำรีแฟคทอริงแล้ว ช่วยทำให้การบำรุงรักษาซอฟต์แวร์ให้ดีขึ้น

จากงานวิจัย [5] ยังขาดมาตรวัดซอฟแวร์เชิงวัตถุที่ตรวจจับร่องรอยที่ไม่ดีอีกหลายแบบ ดังนั้นผู้ทำวิทยานิพนธ์มีความเห็นว่าควรออกแบบมาตรวัดซอฟแวร์เชิงวัตถุเพิ่มเติมเพื่อให้สามารถตรวจจับร่องรอยที่ไม่ดีได้มากขึ้น

2.2.2 Detecting Design Flaws via Metrics in Object-Oriented System [7]

โดย Radu Marinescu

งานวิจัยนี้ได้นำเสนอ มาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีในระบบซอฟต์แวร์เชิงวัตถุและขั้นตอนการออกแบบวิธีการสำหรับการตรวจจับร่องรอยที่ไม่ดีบนซอร์สโค้ด 5 ขั้นตอน ซึ่งมีรายละเอียดดังนี้ โดย 3 ขั้นตอนแรก คือ กำหนดแรงจูงใจ กำหนดวิธีการวัด การออกแบบมาตรวัด จะอธิบายถึงนิยามของเทคนิคการตรวจจับ 2 ขั้นตอนหลัง คือ คำนวณค่ามาตรวัด (Measurement) และการตรวจสอบผลที่พบ (Finding) จะอธิบายถึงการนำเทคนิคนี้ไปประยุกต์ใช้กับกรณีศึกษา

โดยผู้วิจัยนำวิธีนี้มาประยุกต์ใช้กับร่องรอยที่ไม่ดี 2 แบบ คือ Data-Classes ซึ่งเป็นคลาสที่กำหนดค่าให้กับข้อมูลซึ่งภายในคลาสมีเพียงเอคเซสเซอร์เมทธอด คือ เมทธอดที่ใช้

สำหรับอ่านค่าสมาชิกข้อมูลในคลาสเท่านั้น ร่องรอยที่ไม่ดีนี้ทำให้เกิดผลกระทบต่อคุณลักษณะทางคุณภาพ ดังนั้นในการบำรุงรักษาระบบ ถ้ามีการเปลี่ยนแปลงใน Data-Class จะทำให้กระทบกับคลาสอื่นที่เรียกใช้ข้อมูลใน Data Class นั้น ในการทดสอบระบบ ถ้ามีคลาสอื่นๆ มาเรียกใช้ข้อมูลใน Data-Class เหมือนกัน จะเกิดโค้ดที่ซ้ำซ้อนขึ้น ทำให้เพิ่มภาระในการทดสอบระบบมากขึ้น และในการทำความเข้าใจคลาสที่เรียกใช้ Data-Class เพราะฟังก์ชันการทำงานไม่ได้อยู่ในคลาสนั้น แต่มาอยู่ที่ Data-Class แทน ร่องรอยที่ไม่ดีอีกแบบ คือ God-Classes (หรือ Large Class [1]) เป็นคลาสที่มีการทำงานหลักหลายหน้าที่ ทำให้มีผลกระทบต่อคุณลักษณะทางคุณภาพใน การนำกลับมาใช้ใหม่ และในการทำความเข้าใจคลาส เพราะความรับผิดชอบของ God-Class มีหลากหลาย มีผลให้การทำความเข้าใจคลาสนั้นเข้าใจได้ยากขึ้น

งานวิจัยนี้ได้สร้างเครื่องมือสำหรับตรวจจับร่องรอยที่ไม่ดีซึ่งประกอบด้วย 2 ส่วน คือ TableGen ใช้สำหรับเก็บข้อมูลการออกแบบที่ต้องการสร้างโปรแกรมด้วยภาษาซีพลัสพลัส เช่น คลาส เมทธอด ความสัมพันธ์ระหว่างคลาส การเรียกใช้เมทธอด เป็นต้น นำข้อมูลเหล่านั้นมาเก็บในรูปแบบของตาราง และส่วนการนำมาตรวัดมาคำนวณโดยใช้เครื่องมือฐานข้อมูลเชิงสัมพันธ์ (Relational Database Engine) ซึ่งในงานวิจัยนี้ได้ใช้ฐานข้อมูลออราเคิล (Oracle) เพื่อสร้างชุดคำสั่งเอสคิวแอล (SQL) สำหรับคำนวณมาตรวัดที่เลือกใช้ โดยดึงข้อมูลจากตารางเหล่านั้น

ผลลัพธ์ที่ได้จากงานวิจัยนี้คือ มาตรวัดสำหรับการตรวจจับร่องรอยที่ไม่ดี 2 วิธีคือ Data-Classes และ God-Classes ได้แก่ Weight of a class (WOC), Number Of Public Attribute (NOPA), Number Of Accessor Method (NOAM) ซึ่งสามารถนำมาตรวัดนี้ไปใช้กับกรณีศึกษาอื่นๆ เพื่อตรวจสอบความน่าเชื่อถือของมาตรวัดนี้ และสามารถนำขั้นตอนการออกแบบวิธีการสำหรับการตรวจจับร่องรอยที่ไม่ดีนั้นไปประยุกต์ เพื่อการตรวจจับร่องรอยที่ไม่ดีอื่นๆ แต่ไม่สามารถประยุกต์ใช้กับร่องรอยที่ไม่ดี 2 แบบ คือ ร่องรอยที่ไม่ดีที่ไม่สามารถวัดได้ เช่น โครงสร้างโปรแกรมที่เหมือนกันมากกว่า 1 ที่ และร่องรอยที่ไม่ดีที่ไม่สามารถพบได้ที่โปรแกรมต้นฉบับ เช่น ร่องรอยที่ไม่ดีในกระบวนการทดสอบของระบบ ซึ่งหาได้จากรายงานข้อผิดพลาดที่พบโดยคำนวณจากจำนวนการตรวจสอบข้อผิดพลาดต่อคลาส

ผู้จัดทำวิทยานิพนธ์ได้นำมาตรวัดสำหรับการตรวจจับร่องรอยที่ไม่ดีแบบ Data-Classes ได้แก่ NOPA และ NOAM มาทำการหาช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีแบบ Data-Classes เพื่อให้การตรวจจับร่องรอยที่ไม่ดีแบบนี้สามารถทำได้ง่ายและรวดเร็วขึ้น

2.2.3 Taxonomy and an Initial Empirical Study of Bad Smells in Code [4]

โดย Mika Mantyla, Jari Vanhanen, Casper Lessenius

งานวิจัยนี้เสนอความสัมพันธ์ระหว่างร่องรอยที่ไม่ดีต่างๆ โดยใช้การทดลอง และแบ่งประเภทร่องรอยที่ไม่ดีซึ่งเป็นการนำร่องรอยที่ไม่ดีแบบต่างๆ ทั้ง 22 แบบของ [15] ที่ส่งผลกระทบที่คล้ายๆ กันมาอยู่ในกลุ่มเดียวกันซึ่งประโยชน์ของการแบ่งประเภทร่องรอยที่ไม่ดีจะช่วยให้สามารถเข้าใจความสัมพันธ์ระหว่างร่องรอยที่ไม่ดีต่างๆ โดยประเภทร่องรอยที่ไม่ดี (Smell taxonomy) แบ่งเป็น 7 ประเภท ได้แก่ Bloaters, Object-Orientation A busers, Change Preventers, Dispensables, Encapsulators, Couplers และ Others

ผลลัพธ์ที่ได้จากงานวิจัยนี้ คือ การแบ่งประเภทร่องรอยที่ไม่ดีออกเป็น 7 ประเภท ที่ช่วยในการจดจำและ เข้าใจได้ง่าย ซึ่งหาความสัมพันธ์ของร่องรอยที่ไม่ดีต่างๆ ที่มีอยู่จริงในการทดลองโดยใช้วิจารณญาณส่วนตัวในการประเมิน คือ ให้ผู้เชี่ยวชาญให้คะแนนการตรวจจับร่องรอยที่ไม่ดีว่ายากหรือง่าย และเสนอแนะวิธีในการตรวจจับร่องรอยที่ไม่ดีในแต่ละแบบ ทั้งนี้ผลลัพธ์จากการทดลองได้อธิบายความสัมพันธ์ต่างๆ ของร่องรอยที่ไม่ดีในแต่ละแบบและแต่ละประเภท เพื่อเป็นประโยชน์ในการประยุกต์ในการวิเคราะห์และตรวจจับร่องรอยที่ไม่ดีต่างๆ ได้ง่ายและรวดเร็วขึ้น

บทที่ 3

การออกแบบมาตรวัดและการออกแบบการหาช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดี

งานวิจัยนี้ แบ่งขั้นตอนในการดำเนินงานวิจัยออกเป็น 5 ส่วน สามารถแสดงได้ด้วยแผนภาพแอคทิวิตีดังรูปที่ 3.1 ซึ่งมีขั้นตอนของการวิจัยเริ่มจากขั้นตอนที่ 3.1 การออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีเพิ่มเติมจากงานวิจัย [5] ขั้นตอนที่ 3.2 แสดงขั้นตอนพัฒนาเครื่องมือคำนวณมาตรวัดต่อจาก [5] ขั้นตอนที่ 3.3 แสดงขั้นตอนของการออกแบบการทดลอง ขั้นตอนที่ 3.4 แสดงขั้นตอนของการออกแบบการสร้างโมเดลการทำนายร่องรอยที่ไม่ดี และขั้นตอนที่ 3.5 แสดงขั้นตอนของการหาและตรวจสอบช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีและสรุปผล รายละเอียดของแต่ละขั้นตอนแสดงในหัวข้อที่ 3.1 3.2 3.3 3.4 และ 3.5 ตามลำดับ

3.1 ขั้นตอนการออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีนอกเหนือจากงานวิจัย [5]

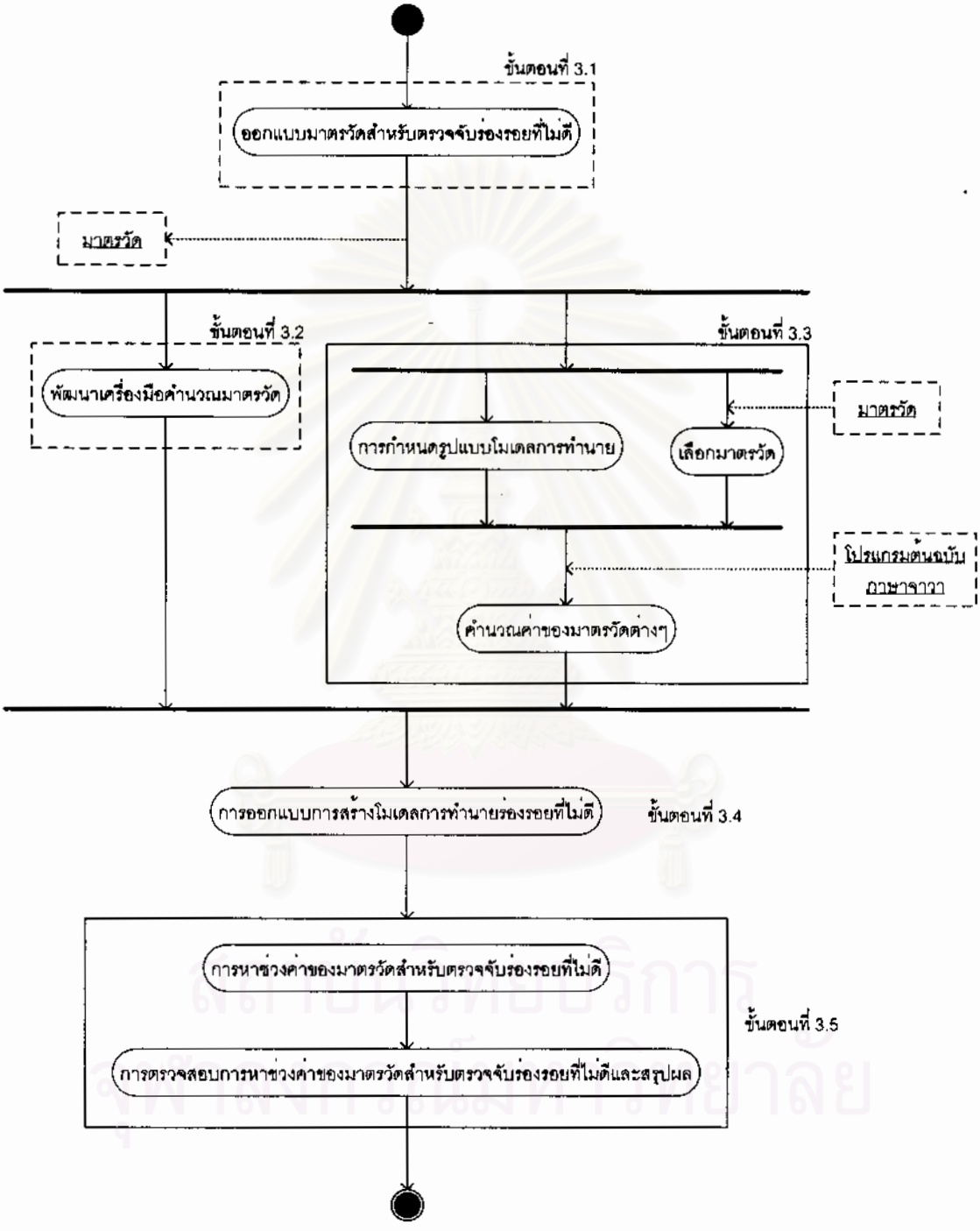
การออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีแบ่งเป็น 6 ขั้นตอนย่อยได้แก่ นิยามของร่องรอยที่ไม่ดี กำหนดแรงจูงใจ กำหนดวิธีการวัด การออกแบบมาตรวัด ข้อกำหนดของค่าที่พิจารณา และการประยุกต์ใช้รีแฟคทอริง ซึ่งมีรายละเอียดตามงานวิจัยที่เกี่ยวข้องหัวข้อ 2.2.1

โดยงานวิจัยนี้ได้เลือกร่องรอยที่ไม่ดี 2 แบบ คือ Data Class และ Refused Bequest มาทำการออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีทั้ง 2 นี้ ซึ่งแสดงรายละเอียดในบทที่ 4

3.2.ขั้นตอนการออกแบบและพัฒนาเครื่องมือคำนวณมาตรวัด

ขั้นตอนนี้เป็นการพัฒนาเครื่องมือด้วยภาษาจาวาสำหรับ การคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดีเพิ่มเติมจากของ [5] อีก 2 แบบและใช้ช่วงค่าของมาตรวัดในงานวิจัยนี้ช่วยการตรวจจับร่องรอยที่ไม่ดี ซึ่งเครื่องมือนี้จะเริ่มทำงานโดยรับโปรแกรมต้นฉบับ ซึ่งเป็นโปรแกรมภาษาจาวาเข้ามา และพาร์เซอร์จะสร้างซินแท็กทรีขึ้น จากนั้นโปรแกรมจะคำนวณค่ามาตรวัดตามที่ได้ออกแบบไว้ เพื่อตรวจจับร่องรอยที่ไม่ดีแต่ละประเภทโดยท่องไปบนซินแท็กทรีเก็บค่า เมื่อได้ผลลัพธ์จากการตรวจจับ (Detection Strategy) แล้วโปรแกรมจะดึงข้อมูลเฉพาะค่าที่อยู่ในข้อกำหนดมาแสดงในรูปแบบของตารางโดยแบ่งตามร่องรอยที่ไม่ดีแต่ละแบบ ซึ่ง

รายละเอียดเกี่ยวกับขั้นตอนการพัฒนาเครื่องมือสำหรับการคำนวณมาตรวัด และตรวจจับร่องรอยที่ไม่ดีทั้ง 8 แบบอยู่ในบทที่ 6



รูปที่ 3.1 ขั้นตอนการวิจัย

3.3 ขั้นตอนการออกแบบการทดลอง

การออกแบบการทดลองนั้นจะเป็นการวางแผนการทำงาน ซึ่งจะมีรายละเอียดเกี่ยวกับการกำหนดรูปแบบโมเดลการทำนาย การเลือกมาตรวัด การเลือกซอร์สโค้ดภาษาจาวา และคำนวณค่าของมาตรวัดต่างๆ

3.3.1 การกำหนดรูปแบบโมเดลการทำนาย

งานวิจัยนี้มีวัตถุประสงค์เพื่อสร้างโมเดลการทำนายร่องรอยที่ไม่ดีและ เพื่อแสดงความสัมพันธ์ของมาตรวัดต่างๆ ในโมเดลกับร่องรอยที่ไม่ดีแต่ละแบบ ซึ่งโมเดลการทำนายร่องรอยที่ไม่ดีมีรูปแบบตามสมการคือ

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

โดยที่ y คือ ค่าทำนายการเกิดร่องรอยที่ไม่ดี

x คือ มาตรวัดที่สนใจ

β_0 คือ ค่าประมาณของส่วนตัดแกน y หรือค่าของ y เมื่อ x มีค่าเป็นศูนย์

β_1 คือ ค่าประมาณค่าสัมประสิทธิ์ของค่ามาตรวัด

3.3.2 การเลือกมาตรวัด

ขั้นตอนนี้จะเลือกมาตรวัดจากขั้นตอนที่ 3.1 และงานวิจัยอื่นๆ เช่น [2, 5, 7-13] ที่สามารถตรวจจับร่องรอยที่ไม่ดีของแต่ละแบบ ซึ่งมาตรวัดที่ใช้ในงานวิจัยนี้ ประกอบด้วยมาตรวัดทั้งหมด 32 มาตรวัด ดังแสดงในตารางที่ 3.1

จากตารางที่ 3.1 มาตรวัดที่มีเครื่องหมายดอกจันทร์ (*) กำกับ คือ มาตรวัดที่งานวิจัยนี้นำเสนอ ซึ่งได้จากการปรับปรุงมาตรวัดที่มีอยู่เดิมให้เป็นมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดี และมาตรวัดที่ไม่มีเครื่องหมายดอกจันทร์กำกับคือมาตรวัดที่มีนักวิจัยท่านอื่นนำเสนอไว้แล้ว [2, 5, 7-13]

ตารางที่ 3.1 แสดงมาตรวัดที่ใช้ในงานวิจัยจำนวน 32 มาตรวัด

ค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ	สำหรับ	มาตรวัด	ชื่อมาตรวัด
Data Class	คลาส	NOPA	Number of Public Attribute
		NOAM	Number of Accessor Method
	เมทธอด	NSM *	Number Of Called Setting Method
		NSOC *	Number of called Setting Method Other Class
		NGM *	Number Of Called Getting Method
		NGOC *	Number of called Getting Method Other Class
	คุณลักษณะ	NCA *	Number of Called Attributes
		NAOC *	Number of Called Attributes Other Class
Feature Envy	เมทธอด	NCM	Number of Called Methods
		NCDA	Number of Called Attributes
		Total	NCM + NCDA
	คุณลักษณะ	NCRA	Number of Caller Attributes
Large Class	คลาส	NIM	Number of Instance Methods in a class
		NIV	Number of Instance Variables in a Class
		TCC	Tight Class Cohesion
		NLOC	Number of statement
		AMC	Average Method Complexity per Method
Lazy Class	คลาส	NIM	Number of Instance Methods in a class
		NIV	Number of Instance Variables in a Class
		DIT	Depth of Inheritance Hierarchy
		AMC	Average Method Complexity per Method
Long Method	เมทธอด	NOS	Number of Statements in Method
		NOP	Number of Parameters in Method
		NOT	Number of Temporary variables in Method
		MCX	Method complexity

ตารางที่ 3.1 แสดงมาตรวัดที่ใช้ในงานวิจัยจำนวน 31 มาตรวัด (ต่อ)

ค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ	สำหรับ	มาตรวัด	ชื่อมาตรวัด
Long Parameter List	เมทธอด	NOP	Number of Parameters in Method
Refused Bequest	เมทธอด	NOCM *	Number of Called Methods
		NSMP *	Number of Subclasses Use Method from Parent Class
		SNOCM*	Sum of NOCM
	คุณลักษณะ	NOCA *	Number of Called Attribute
		SNOCA*	Sum of NOCA
		NSAP *	Number of Subclasses Use Attribute from Parent Class
Switch Statement	เมทธอด	NOSS	Number of Switch statements in Method

3.3.3 การเลือกซอร์สโค้ดที่ใช้ในการทดลองที่เป็นภาษาจาวา

ขั้นตอนนี้จะเป็นขั้นตอนหาโปรแกรมต้นฉบับ ที่ใช้ในการทดลองที่เป็นภาษาจาวา แล้วให้ผู้เชี่ยวชาญในที่นี้ คือ ผู้ทำวิทยานิพนธ์เป็นผู้กำหนดว่าโปรแกรมต้นฉบับนั้นเกิดร่องรอยที่ไม่ดีหรือไม่ โดยดูจากเกณฑ์มาตรวัดของภาษาซีพลัสพลัสสำหรับตรวจจับร่องรอยที่ไม่ดีแบบต่างๆ [5, 8] ซึ่งแสดงดังตารางที่ 3.2 ซึ่งใช้เงื่อนไขแบบ OR คือ ถ้าค่ามาตรวัดอย่างน้อยหนึ่งตัวอยู่ในข้อกำหนดค่าพิจารณาแสดงว่า คลาส เมทธอด หรือ คุณลักษณะ นั้นเกิดร่องรอยที่ไม่ดี

ถ้าเกิดร่องรอยที่ไม่ดีเป็นร่องรอยที่ไม่ดีประเภทใด ซึ่งจะกำหนดให้ค่า 1 คือเกิดร่องรอยที่ไม่ดีในโปรแกรมต้นฉบับและค่า 0 คือไม่เกิดร่องรอยที่ไม่ดีในโปรแกรมต้นฉบับ โดยที่โปรแกรมต้นฉบับภาษาจาวาแบ่งเป็น 2 ชุด คือ ชุดข้อมูลสอนสำหรับการสร้างโมเดลการทำนาย ซึ่งมีจำนวน 10 โปรแกรม 760 คลาส และชุดข้อมูลสำหรับทดสอบ เพื่อตรวจสอบความถูกต้องของโมเดล 2 โปรแกรม 200 คลาส

ตารางที่ 3.2 เกณฑ์มาตรวัดของภาษาซีพลัสพลัสสำหรับตรวจจับร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	สำหรับ	มาตรวัด	ข้อกำหนดค่าที่พิจารณา
Data Class	คลาส	NOPA	10 คลาสที่มี NOPA มากที่สุดแต่ไม่น้อยกว่า 5
		NOAM	10 คลาสที่มี NOAM มากที่สุดแต่ไม่น้อยกว่า 3
Feature Envy	เมทธอด	NCM,NCDA	NCM+NCDA (คลาสของตัวเอง) < NCM+NCDA (คลาสอื่น)
	คุณลักษณะ	NCRA	NCRA (คลาสของตัวเอง) < NCRA (คลาสอื่น)
Large Class	คลาส	NIM	NIM >20
		NIV	NIV >3
		TCC	เลือกคลาสที่มี TCC ต่ำที่สุด
Lazy Class	คลาส	NIM, NIV	ค่าน้อยที่สุดของ NIM+NIV
Long Method	เมทธอด	NOS	NOS > 7
Long Parameter List	เมทธอด	NOP	เลือกเมทธอดที่มี NOP มากที่สุด
Switch statement	เมทธอด	NOSS	NOSS >0

3.3.4 คำนวณค่าของมาตรวัดต่าง ๆ

ขั้นตอนนี้จะนำเครื่องมือจากขั้นตอนที่ 3.2 มาช่วยคำนวณค่ามาตรวัดเพื่อทำการเก็บข้อมูลของแต่ละมาตรวัด

3.4 ขั้นตอนการออกแบบการสร้างโมเดลการทำนายร่องรอยที่ไม่ดี

งานวิจัยนี้สร้างโมเดลการทำนายร่องรอยที่ไม่ดีของแต่ละแบบ โดยใช้ข้อมูลที่เป็นชุดข้อมูลสอน ที่คำนวณค่าที่เกิดร่องรอยที่ไม่ดี (Y) จากเกณฑ์มาตรวัดของภาษาซีพลัสพลัสสำหรับตรวจจับร่องรอยที่ไม่ดีในตารางที่ 3.2 กับมาตรวัด (x) ในตาราง 3.1 มาวิเคราะห์ทางสถิติด้วยวิธีการวิเคราะห์ความถดถอย ด้วยโปรแกรมเอสพีเอสเอสสำหรับวินโดว์ เพื่อสร้างโมเดลการทำนายร่องรอยที่ไม่ดี ซึ่งโมเดลการทำนายร่องรอยที่ไม่ดีมีรูปแบบตามสมการของขั้นตอนที่ 3.3.1 เนื่องจากค่าอาร์สแควร์ที่ปรับแล้ว (Adjusted R Square) เป็นค่าที่แสดงว่าตัวแปรอิสระ (x) อธิบายความแปรปรวนของตัวแปรตาม (y) ได้ดีเพียงใด ดังนั้นมาตรวัดที่มีค่าค่าอาร์สแควร์ที่ปรับแล้วสูงแสดงว่าสามารถอธิบายค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ได้ดี ค่า

มาตรวัดนั้นจะถูกเลือกไว้ เพื่อสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ส่วนมาตรวัดที่มีค่าอาร์สแควร์ที่ปรับแล้วน้อยกว่าจะถูกตัดทิ้ง

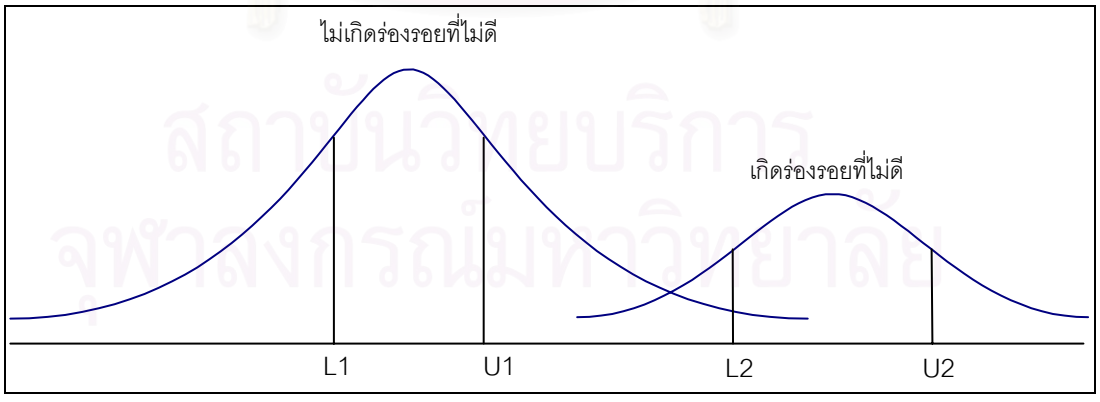
3.5.ขั้นตอนการหาและตรวจสอบช่วงค่าของมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีและสรุปผล

ขั้นตอนนี้มี 2 ขั้นตอนย่อยคือ ซึ่งมีรายละเอียดของแต่ละขั้นตอนย่อยดังนี้

3.5.1 ขั้นตอนการหาช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดี

ขั้นตอนนี้ จะนำค่ามาตรวัดโดยเลือกมาตรวัดเฉพาะที่อยู่ในโมเดลจากขั้นตอน 3.3.3 มาแสดงอยู่ในตารางแจกแจงความถี่ของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีเพื่อทำการแบ่งช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และ ตารางแจกแจงความถี่ของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีเพื่อทำการแบ่งช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี โดยการประมาณแบบช่วงของค่ามาตรวัดทั้งเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดีนั้นใช้ระดับความเชื่อมั่น 95% และมีสูตรการคำนวณตามทฤษฎีที่เกี่ยวข้องในหัวข้อ 2.1.3.7 ซึ่งกำหนด L เป็น ค่าต่ำสุด และ U เป็นค่าสูงสุด ดังนั้นช่วงของค่าคือ (L, U)

หลังจากนั้น นำข้อมูลในตารางแจกแจงความถี่มาแสดงในรูปของกราฟดังรูปที่ 3.2 โดยมีช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L2, U2) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, U1) แล้วพิจารณาว่ามีพื้นที่ใต้กราฟของช่วงของค่ามาตรวัดทั้งเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดีซ้อนทับกันหรือไม่ ถ้าไม่มีพื้นที่ใต้กราฟของช่วงของค่ามาตรวัดทั้งเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดีที่ซ้อนทับกันแล้วสามารถใช้ช่วงของค่ามาตรวัดนี้ได้

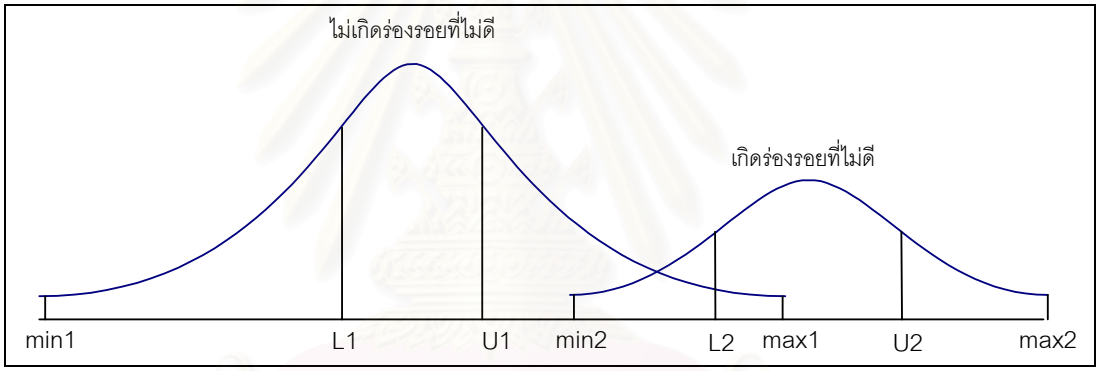


รูปที่ 3.2 กราฟแสดงช่วงของค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีและการไม่เกิดร่องรอยที่ไม่ดี

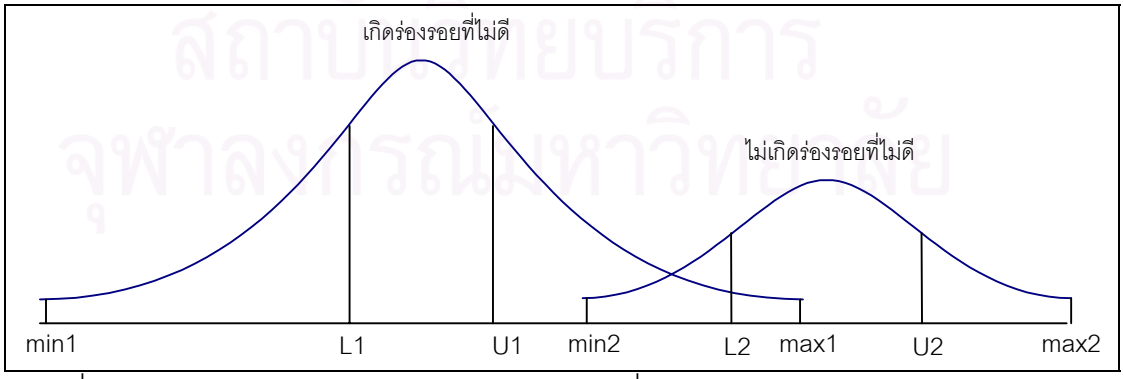
3.5.2 ขั้นตอนการประเมินค่าความน่าเชื่อถือของช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดี

ขั้นตอนนี้เริ่มจากทำการเปรียบเทียบค่าที่ได้จากช่วงของค่ามาตรวัดและค่าที่ได้จากผู้เชี่ยวชาญเป็นผู้กำหนดโปรแกรมที่ใช้สำหรับทำการประเมินความน่าเชื่อถือ แล้วหาเปอร์เซ็นต์ความผิดพลาด เปอร์เซ็นต์ความถูกต้องและเปอร์เซ็นต์ที่ไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่ พร้อมกับสรุปผลการทดลอง ถ้าเปอร์เซ็นต์ความถูกต้องมีค่าน้อยกว่าผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ที่ไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่ ควรปรับช่วงค่าของมาตรวัดใหม่ โดยนำตารางแจกแจงความถี่มาแสดงให้อยู่ในรูปของกราฟเพื่อประกอบการพิจารณาดังนี้

พิจารณากรณีที่ไม่มีส่วนที่ซ้อนทับกันระหว่างช่วงของค่ามาตรวัดเกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดไม่เกิดร่องรอยที่ไม่ดี



รูปที่ 3.3 กราฟแสดงค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีมากกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดี



รูปที่ 3.4 กราฟแสดงค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดี

กรณีที่ 1 ค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีมากกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดีดังรูปที่ 3.3 ควรเปลี่ยนเป็นช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L2, max2) และ ช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (min1, U1)

กรณีที่ 2 ค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดีดังรูปที่ 3.4 ควรเปลี่ยนเป็นช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (min1, U1) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L2, max2)

หลังจากปรับค่าของมาตรวัดใหม่แล้ว ทำการหาเปอร์เซ็นต์ความผิดพลาด เปอร์เซ็นต์ความถูกต้องและเปอร์เซ็นต์ที่ไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่ แล้วนำเปอร์เซ็นต์ความถูกต้องมาเปรียบเทียบกันระหว่างก่อนปรับค่าและหลังปรับค่า ถ้าเปอร์เซ็นต์ความถูกต้องของมาตรวัดก่อนปรับค่ามีค่ามากกว่าเปอร์เซ็นต์ความถูกต้องของมาตรวัดหลังปรับค่า ให้ใช้ช่วงของค่าของมาตรวัดก่อนปรับค่า แต่ถ้าเปอร์เซ็นต์ความถูกต้องของมาตรวัดหลังปรับค่าช่วงมาตรวัดมีค่ามากกว่าเปอร์เซ็นต์ความถูกต้องของมาตรวัดก่อนปรับค่า ให้ใช้ช่วงค่าของมาตรวัดหลังปรับค่า

บทที่ 4
การออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีเพิ่มเติม

ในบทนี้จะอธิบายการออกแบบมาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีแบบ Data Class และ Refused Bequest และทดสอบความน่าเชื่อถือของมาตรวัดร่องรอยที่ไม่ดี โดยมาตรวัดที่ออกแบบนี้เป็นมาตรวัดใหม่ทั้งหมด ซึ่งมีรายละเอียดดังนี้

4.1 ร่องรอยที่ไม่ดีแบบ Data Class

4.1.1 นิยามของร่องรอยที่ไม่ดี

Data Class เป็นคลาสที่มีการทำงานหลักทำการกำหนดค่าให้กับข้อมูล และภายในคลาสมีเพียงเอคเซสเซอร์เมทธอด (ได้แก่เมทธอดที่กำหนดคุณลักษณะและเมทธอดที่เรียกใช้คุณลักษณะ) เท่านั้น

4.1.2 กำหนดแรงจูงใจ

4.1.2.1 เมทธอดที่กำหนดคุณลักษณะในคลาส แต่ไม่ได้กำหนดคุณลักษณะขึ้นมาใหม่ ถูกเรียกใช้เพื่อกำหนดคุณลักษณะภายในคลาสเพียงครั้งเดียวและไม่มีคลาสภายนอกเรียกใช้ ทำให้เกิดความซ้ำซ้อนในการกำหนดคุณลักษณะและเสียค่าใช้จ่ายในการบำรุงรักษาเมทธอดที่ไม่จำเป็น

4.1.2.2 คุณลักษณะถูกเรียกใช้เพื่อกำหนดคุณลักษณะใหม่หลายครั้งทั้งคลาสตัวเองและคลาสอื่น ในการเรียกคุณลักษณะโดยตรงทำให้ค่าความสัมพันธ์ของวัตถุสูงและมีการห่อหุ้ม (Encapsulation) ต่ำ

4.1.2.3 เอคเซสเซอร์เมทธอด (เมทธอดที่กำหนดคุณลักษณะในคลาสและเมทธอดที่ดึงค่าคุณลักษณะในคลาส) ที่ไม่ถูกเรียกใช้จากคลาสอื่น ทำให้ค่าการห่อหุ้ม (Encapsulation) ต่ำ

4.1.3 กำหนดวิธีการวัด

4.1.3.1 โดยพิจารณาภายในเมทธอดว่า ถ้าผลรวมของจำนวนครั้งที่เมทธอดกำหนดคุณลักษณะที่สนใจภายในคลาสที่พิจารณาถูกเรียกใช้และ จำนวนครั้งที่เมทธอดกำหนดคุณลักษณะที่สนใจถูกเรียกใช้จากคลาสอื่นมีค่าเท่ากับ 0 คือไม่มีจำนวนเรียกใช้เมทธอดที่กำหนดคุณลักษณะ ควรลบเมทธอดที่กำหนดคุณลักษณะนั้นออก เพราะจะช่วยลดความซับซ้อนในการเรียกใช้ภายในคลาส และลดค่าใช้จ่ายการบำรุงรักษาเพราะเป็นส่วนของโปรแกรมที่ไม่จำเป็น

4.1.3.2 โดยพิจารณาจากคุณลักษณะที่สนใจว่าถูกเรียกใช้ภายในคลาสตัวเองและจากคลาสอื่นมาก ดังนั้นควรสร้างเอคเซสเซอร์เมทธอด เพื่อเพิ่มค่าความสัมพันธ์ของวัตถุต่ำและมีการห่อหุ้ม (Encapsulation) สูง

4.1.3.3 โดยพิจารณาจากเอคเซสเซอร์เมทธอดของคุณลักษณะที่พิจารณาว่าไม่ถูกเรียกใช้จากคลาสอื่นและเรียกใช้เฉพาะในคลาสของตัวเอง ดังนั้นควรกำหนดประเภทของเอคเซสเซอร์เมทธอดให้เป็น Private

4.1.4 การออกแบบมาตรวัด

4.1.4.1. มาตรวัด Number Of Called Setting Method (NSM)

คือ จำนวนครั้งที่เมทธอดกำหนดคุณลักษณะที่สนใจภายในคลาสที่พิจารณาถูกเรียกใช้และเมทธอดเป็นประเภท public

4.1.4.2. มาตรวัด Number of called Setting Method Other Class (NSOC)

คือ จำนวนครั้งที่เมทธอดกำหนดคุณลักษณะที่สนใจถูกเรียกใช้จากคลาสอื่น

4.1.4.3. มาตรวัด Number of Called Attributes(NCA)

คือ จำนวนครั้งที่คุณลักษณะถูกเรียกภายในคลาสที่คุณลักษณะอยู่

4.1.4.4. มาตรวัด Number of Called Attributes Other Class (NAOC)

คือ จำนวนครั้งที่คุณลักษณะถูกเรียกจากคลาสอื่น

4.1.4.5. มาตรวัด Number Of Called Getting Method (NGM)

คือ จำนวนครั้งที่เมทธอดรับค่าคุณลักษณะที่สนใจภายในคลาสที่พิจารณาถูกเรียกใช้และคุณลักษณะเป็นประเภท public

4.1.4.6. มาตรวัด Number of called Getting Method Other Class (NGOC)

คือ จำนวนครั้งที่เมทธอดรับค่ากำหนดคุณลักษณะที่สนใจถูกเรียกใช้จากคลาสอื่น

4.1.5 ข้อกำหนดของค่าที่พิจารณา

ข้อกำหนดของค่าที่พิจารณาได้จากการหาช่วงค่าของร่องรอยที่ไม่ดีซึ่งมีรายละเอียดในบทที่ 5

4.1.5.1 สำหรับ Remove setting method

NSM + NSOC = 0 หรือ 1

4.1.5.2 สำหรับ Encapsulate field

$24 \leq NCA \leq 310$ หรือ $6 \leq NAOC \leq 36$

4.1.5.3 สำหรับ Hide method

(NSM ≥ 1 และ NSOC = 0) หรือ (NGM ≥ 1 และ NGOC = 0)

4.1.6 การประยุกต์ใช้รีแฟคทอริง

วิธีการรีแฟคทอริงที่ใช้ปรับปรุง Data Class ในที่นี้มี 3 วิธีคือ Remove setting method, Encapsulate field และ Hide method

4.1.6.1. เมื่อพิจารณาภายในคลาสพบว่า NSM + NSOC = 0 จะประยุกต์ใช้วิธีรีแฟคทอริงคือ Remove setting method ซึ่งเป็นการลบเมทธอดที่กำหนดคุณลักษณะออก

4.1.6.2. เมื่อพิจารณาภายในคลาสพบว่า $24 \leq NCA \leq 310$ หรือ $6 \leq NAOC \leq 36$ จะประยุกต์ใช้วิธีรีแฟคทอริงคือ Encapsulate field ซึ่งเป็นการเพิ่มเอคเซสเซอร์เมทธอด

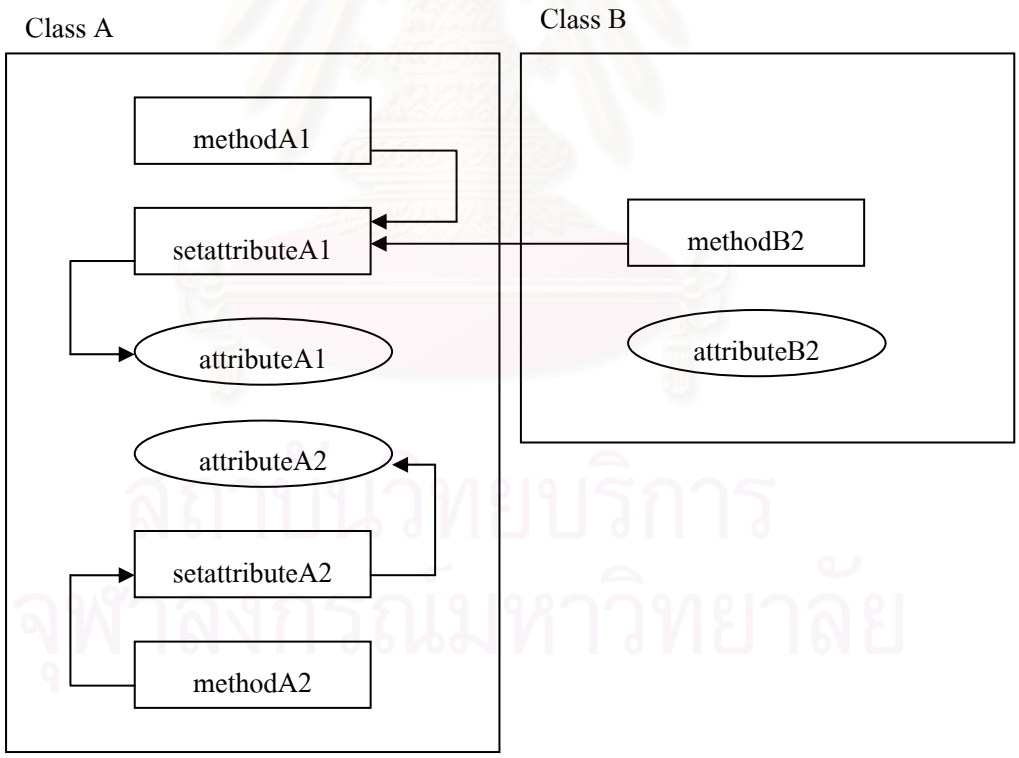
4.1.6.3. เมื่อพิจารณาภายในคลาสพบว่า (NSM ≥ 1 และ NSOC = 0) หรือ (NGM≥ 1 และ NGOC = 0) จะประยุกต์ใช้วิธีรีแฟคทอริงคือ Hide method ซึ่งเป็นการกำหนดประเภทของเมทธอดเป็น Private

ตัวอย่างการตรวจจับร่องรอยที่ไม่ดีแบบ Data Class และการประยุกต์ใช้รีแฟคทอริง ซึ่งแสดงดังตัวอย่างของโปรแกรมหนึ่งที่ประกอบด้วย คลาส A และคลาส B ดังนี้

```
Class A{
    Private String attributeA1;
    Private String attributeA2;
    methodA1(String arg_attributeA1){
        setattributeA1(arg_attributeA1);
    }
    methodA2(String arg_attributeA2){
        setattributeA2(arg_attributeA2);
    }
    void setattributeA1(String arg1){
        attributeA1 =  arg1;
    }
```

```
        void setattributeA2(String arg2){
            attributeA2 =  arg2;
        }
    }
    Class B{
        Private A _A;
        Private String attributeB1;
        methodB1(){
            _A.setattributeA1(attributeB1);
        }
    }
```

เพื่อการทำความเข้าใจได้ง่ายขึ้นจึงสร้างรูปที่ 4.1 เพื่อแสดงการเรียกใช้เมทธอดหรือคุณลักษณะภายในเมทธอดของคลาส A โดยคุณลักษณะที่สนใจคือ attributeA1 และ attributeA2



รูปที่ 4.1 แสดงการเรียกใช้เมทธอดหรือคุณลักษณะภายในเมทธอดก่อนการทำรีแฟคทอริง

พิจารณาจากรูปที่ 4.1 นำมาตรวัดที่ได้ออกแบบไว้มาคำนวณได้ผลดังตารางที่ 4.1 และ 4.2

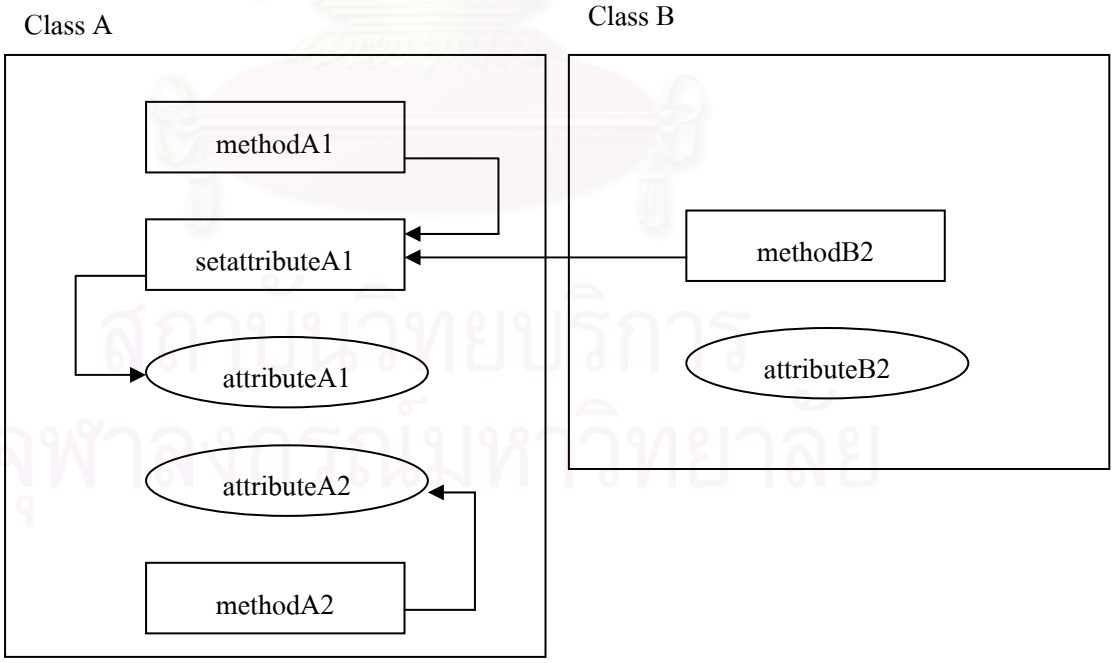
ตารางที่ 4.1 แสดงค่าของมาตรวัด NSM, NSOC, NGM, AGOC ของคลาส A และ คลาส B

Class	Method	NSM	NSOC	NGM	AGOC
A	methodA1	-	-	-	-
	methodA2	-	-	-	-
	setattributeA1	1	1	-	-
	setattributeA2	1	0	-	-
B	methodB1	-	-	-	-

ตารางที่ 4.2 แสดงค่าของมาตรวัด NCA, NAOC ของคลาส A และ คลาส B

Class	Attribute	NCA	NAOC
A	attributeA1	1	1
	attributeA2	1	0
B	attributeB1	0	0

จากผลที่ได้ในตารางที่ 4.1 จะเห็นว่า $(NSM + NSOC)_{(\ setattributeA2)}$ =1 ดังนั้นจึงควรลบเมทธอดที่กำหนดคุณลักษณะ attributeA2 ออก ซึ่งแสดงดังรูปที่ 4.2



รูปที่ 4.2 แสดงการเรียกใช้เมทธอดหรือคุณลักษณะภายในคลาสหลังการทำรีเฟคทอริง

จากรูปที่ 4.2 แสดงให้เห็นว่า ลบเมทธอดที่กำหนดคุณลักษณะ attributeA2 ออก ทำให้จำนวนการเรียกใช้ภายในคลาสลดลง ดังผลที่ได้หลังจากการทำรีเฟคทอริงแล้ว

attributeA2	NSM = 0	NSOCA=0	NSM + NSOC =0

ซึ่งจะได้โปรแกรมหลังการทำวิธี Remove setting method แล้วดังนี้

```
Class A{
        Private String attributeA1;
        Private String attributeA2;
        methodA1(String arg_attributeA1){
                setattributeA1(arg_attributeA1);
        }
        methodA2(String arg_attributeA2){
                 attributeA2 = arg_ attributeA2;
        }
        void setattributeA1(String arg){
                attributeA1 =  arg;
        }
}

Class B{
        Private A _A;
        Private String attributeB1;
        methodB1(){
             _A.setattributeA1(attributeB1);
        }
}
```

ข้อควรระวัง

ไม่ควรประยุกต์ใช้รีแฟคทอริงแบบ Remove setting method ในกรณีที่เมทธอดที่กำหนดคุณลักษณะนั้นไม่กำหนดเป็นค่าของตัวมันเอง หรือเมทธอดที่กำหนดคุณลักษณะมีการกำหนดค่าเพิ่มเติม ถ้าลบเมทธอดที่กำหนดคุณลักษณะจะส่งผลกระทบต่อออบเจ็คที่เรียกใช้คุณลักษณะนั้นเนื่องจากค่าของคุณลักษณะไม่ถูกต้อง แสดงตัวอย่างในคลาส Account ดังนี้

```
Class Account{
        Private String _id;
        Account(String id){
            setId(id);
        }
        void setId(String arg){
            _id =  “ZZ” + arg;
        }
}
```

หลังการรีแฟคทอริงแล้วจะได้เพียงคอนสเตคเตอร์เดียว การเปลี่ยนแปลงคอนสเตคเตอร์นี้ส่งผลให้ซับซ้อนมากขึ้นและไม่เป็นที่นิยมทำ ดังนั้นจึงใช้สร้างอีกเมทธอดขึ้นมาเพื่อเรียกใช้ ดังนี้

```
Class Account{
        Private final String _id;
        Account(String id){
            initializeId(id);
        }
        void initializeId(String arg){
            _id =  “ZZ” + arg;
        }
}
```

4.2 ร่องรอยที่ไม่ดีแบบ Refused Bequest

4.2.1 นิยามของร่องรอยที่ไม่ดี

Refused Bequest คือคุณสมบัติหรือเมทธอดที่สืบทอดจากคลาสแม่ แต่คลาสลูกไม่ใช้ ทำให้ความหมายของการสืบทอดผิด

4.2.2 กำหนดแรงจูงใจ

คลาสลูกส่วนใหญ่ไม่ได้ใช้เมทธอดหรือคุณลักษณะที่สืบทอดมาจากคลาสแม่

4.2.3 กำหนดวิธีการวัด

4.2.3.1 โดยพิจารณาจากเมทธอดที่สืบทอดจากคลาสแม่ในคลาสลูก ถ้ามีคลาสลูกเพียงตัวเดียวที่มีจำนวนเรียกใช้เมทธอดนั้นภายในคลาสลูก และจำนวนเรียกใช้เมทธอดจากคลาสอื่นมากกว่า 1 แล้ว ให้ลบเมทธอดนั้นจากคลาสแม่และเพิ่มเมทธอดนั้นในคลาสลูกแทน

4.2.4.1 โดยพิจารณาจากคุณลักษณะ ในคลาสลูกที่สืบทอดจากคลาสแม่ ถ้ามีคลาสลูกเพียงตัวเดียวที่มีจำนวนเรียกใช้คุณลักษณะนั้นภายในคลาสลูก และจำนวนเรียกใช้คุณลักษณะจากคลาสอื่นมากกว่า 1 ครั้งแล้ว ให้ลบคุณลักษณะนั้นจากคลาสแม่และเพิ่มเมทธอดนั้นในคลาสลูกแทน

4.2.4 การออกแบบมาตรวัด

4.2.4.1 มาตรวัด Number of Called Method (NOCM)

คือ NOCM = 1 เมื่อจำนวนเมทธอดที่เราสนใจที่สืบทอดจากคลาสแม่ในคลาสที่เราสนใจถูกเรียกใช้ภายในคลาสและถูกเรียกใช้จากคลาสอื่นๆ ที่มีค่ามากกว่าหรือเท่ากับ 1

NOCM = 0 เมื่อจำนวนเมทธอดที่เราสนใจที่สืบทอดจากคลาสแม่ในคลาสที่เราสนใจถูกเรียกใช้ภายในคลาสและถูกเรียกใช้จากคลาสอื่นๆ ที่มีค่าเท่ากับ 0

4.2.4.2 มาตรวัด Number of Subclass Use Method from Parent Class (NSMP)

คือ จำนวนคลาสลูกที่ใช้เมทธอดที่เราสนใจ

4.2.4.3 มาตรวัด Sum of NOCM (SNOCM)

คือ ผลรวมของ NOCM ของทุกคลาสที่เป็นเมทธอดที่เราสนใจ

4.2.4.4 มาตรวัด Number of Called Attribute (NOCA)

คือ NOCA = 1 เมื่อจำนวนคุณลักษณะที่เราสนใจที่สืบทอดจากคลาสแม่ในคลาสที่เราสนใจถูกเรียกใช้ภายในคลาสและถูกเรียกใช้จากคลาสอื่นๆ มีค่ามากกว่าหรือเท่ากับ 1

NOCA = 0 เมื่อจำนวนคุณลักษณะที่เราสนใจที่สืบทอดจากคลาสแม่ในคลาสที่เราสนใจถูกเรียกใช้ภายในคลาสและถูกเรียกใช้จากคลาสอื่นๆ มีค่าเท่ากับ 0

4.2.4.5 มาตรวัด Number of Subclass Use Attribute from Parent Class (NSAP)

นิยาม จำนวนคลาสลูกที่ใช้คุณลักษณะที่เราสนใจ

4.2.4.6 มาตรวัด Sum of NOCA (SNOCA)

นิยาม ผลรวมของ NOCA ของทุกคลาสที่เป็นคุณลักษณะที่เราสนใจ

4.2.5 ข้อกำหนดของค่าที่พิจารณา

วิธีการรีแฟคทอริงที่ใช้ปรับปรุง Refused Bequest ในที่นี้มี 2 วิธีคือ Push down method และ Push down field

4.2.5.1 สำหรับ Push down method

SNOCM = 0 หรือ 1 และ NSMP $\leq$ 2

4.2.5.2 สำหรับ Push down field

SNOCA = 0 หรือ 1 และ NSAP $\leq$ 2

4.2.6 การประยุกต์ใช้รีแฟคทอริง

4.2.6.1 เมื่อในโปรแกรมพบค่า SNOCM = 0 หรือ 1 ของคลาสลูกที่สนใจ และ NSMP $\leq$ 2 คือ มีคลาสลูกน้อยกว่า 2 คลาสที่มีการใช้เมทธอดที่สืบทอดจากคลาสแม่ จะประยุกต์ใช้วิธีรีแฟคทอริงคือ Push down method เป็นการลบเมทธอดนั้นจากคลาสแม่แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่มีการใช้เมทธอดนั้น

4.2.6.2 เมื่อในโปรแกรมพบค่า SNOCA = 0 หรือ 1 ของคลาสลูกที่สนใจ และ NSAP $\leq$ 2 คือ มีคลาสลูกน้อยกว่า 2 คลาสที่มีการใช้คุณลักษณะที่สืบทอดจากคลาสแม่ จะประยุกต์ใช้วิธีรีแฟคทอริงคือ Push down field เป็นการลบคุณลักษณะนั้นจากคลาสแม่แล้วเพิ่มคุณลักษณะนั้นในคลาสลูกที่มีการใช้คุณลักษณะนั้น

ตัวอย่างการตรวจจับร่องรอยที่ไม่ดีแบบ Refused Bequest และการประยุกต์ใช้รีแฟคทอริง แสดงดังรูปที่4.3 โปรแกรมนี้ประกอบด้วย คลาส B และคลาส A โดยคลาส A เป็นคลาสแม่ของคลาส A1 และ A2

ตารางที่ 4.3 ตาราง NOCM ของเมทธอดที่สนใจ

เมทธอดที่สนใจ	NOCM	
	ClassA1	ClassA2
methodA	1	0

ตารางที่ 4.4 ตาราง NOCA ของคุณลักษณะที่สนใจ

คุณลักษณะที่สนใจ	NOCM	
	ClassA1	ClassA2
attributeA1	1	0
attributeA2	1	1

จากตารางที่ 4.3 และ 4.4 แสดงให้เห็นว่า

$SNOCM_{(methodA)} = NOCM_{(methodA)\ (ClassA1)} + NOCM_{(methodA)\ (ClassA2)}$

$= 1+0$

$= 1$

$SNOCA_{(attributeA1)} = NOCA_{(attributeA1)\ (ClassA1)} + NOCA_{(attributeA1)\ (ClassA2)}$

$= 1+0$

$= 1$

$SNOCA_{(attributeA2)} = NOCA_{(attributeA2)\ (ClassA1)} + NOCA_{(attributeA2)\ (ClassA2)}$

$= 1+1$

$= 2$

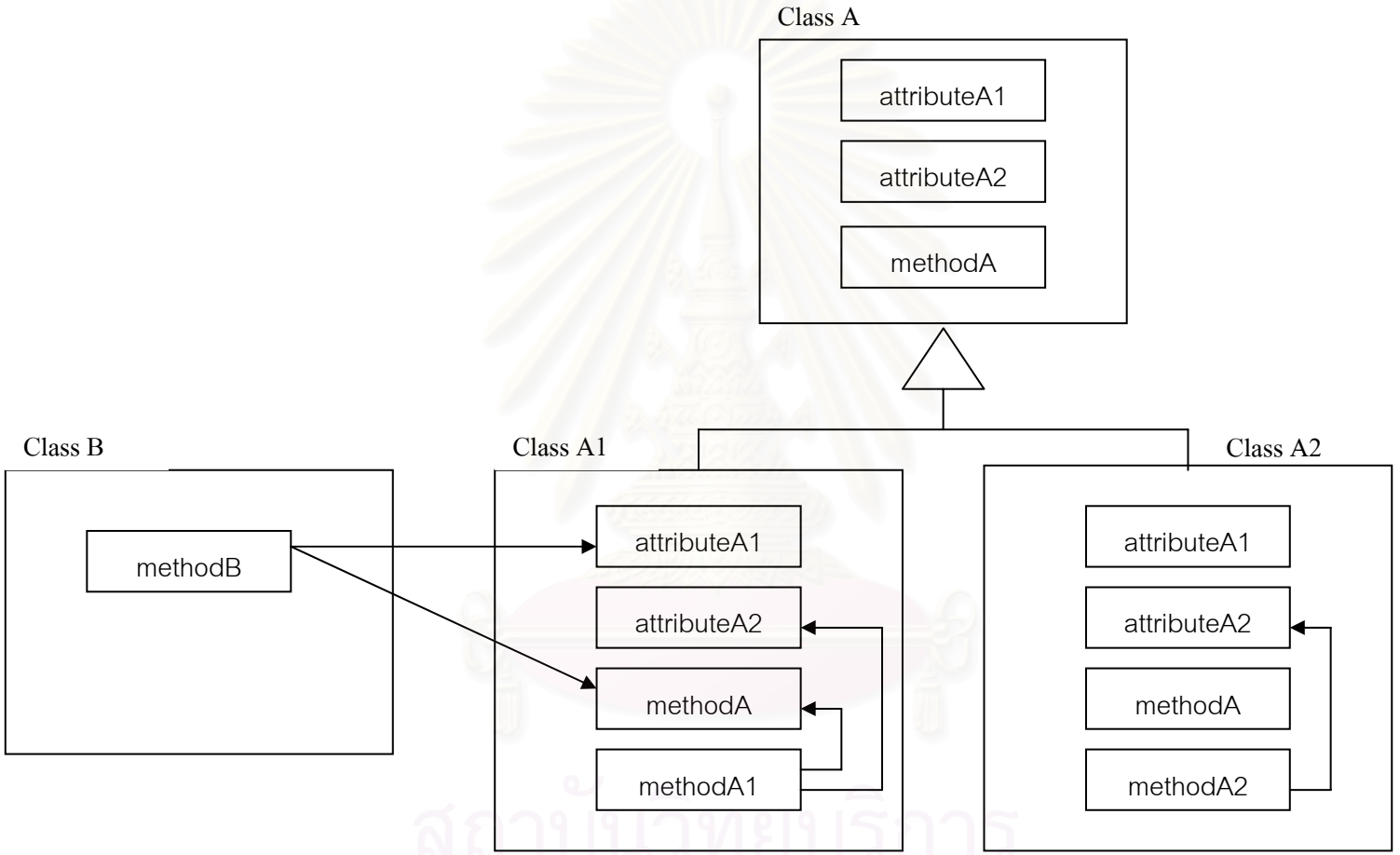
$NSMP_{(methodA)} = 1$

$NSAP_{(attributeA1)} = 1$

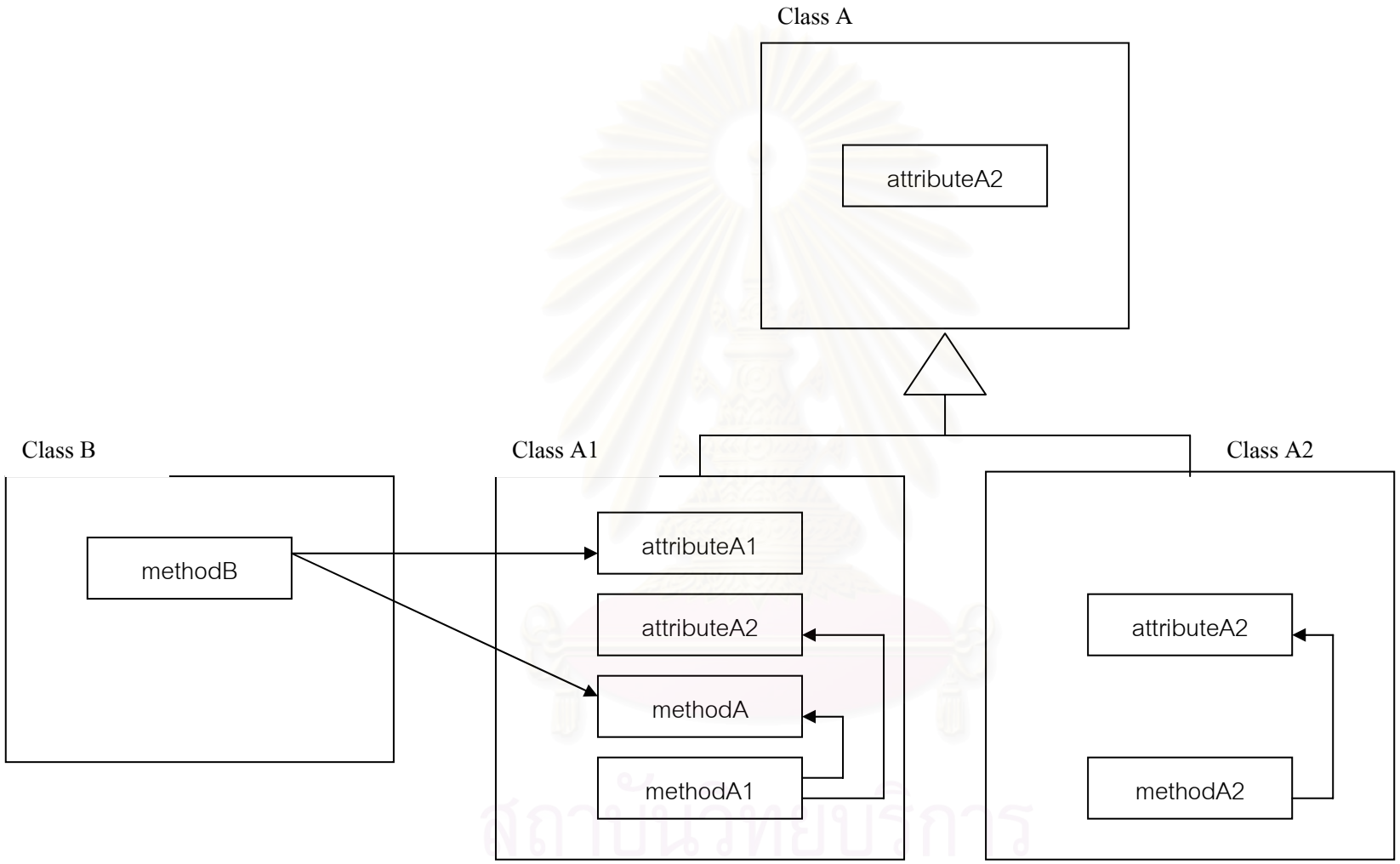
$NSAP_{(attributeA2)} = 2$

สรุปได้ว่าควรลบเมทธอด methodA และคุณลักษณะ attributeA ใน ClassA ซึ่งเป็นคลาสแม่ของ ClassA1 และ ClassA2 แล้วเพิ่มเมทธอด methodA และคุณลักษณะ attributeA ใน ClassA1 ซึ่งสามารถแสดงได้ดังรูปที่ 4.4

จากรูปที่ 4.4 แสดงให้เห็นว่าหลังจากลบเมทธอด methodA และคุณลักษณะ attributeA แล้วทำให้ SNOCM ของเมทธอด methodA และ SNOCA ของคุณลักษณะ attributeA ไม่มีค่า ทำให้ความหมายของการสืบทอดไม่ผิด



รูปที่ 4.3 แสดงการเรียกใช้เมทธอดและคุณลักษณะในคลาสลูกก่อนการทำรีแฟคทอริง



รูปที่ 4.4 แสดงการเรียกใช้เมทธอดและคุณลักษณะในคลาสลูกหลังการทำรีเฟคทอริง

4.3 ทดสอบความน่าเชื่อถือของมาตรวัดร่องรอยที่ไม่ดี

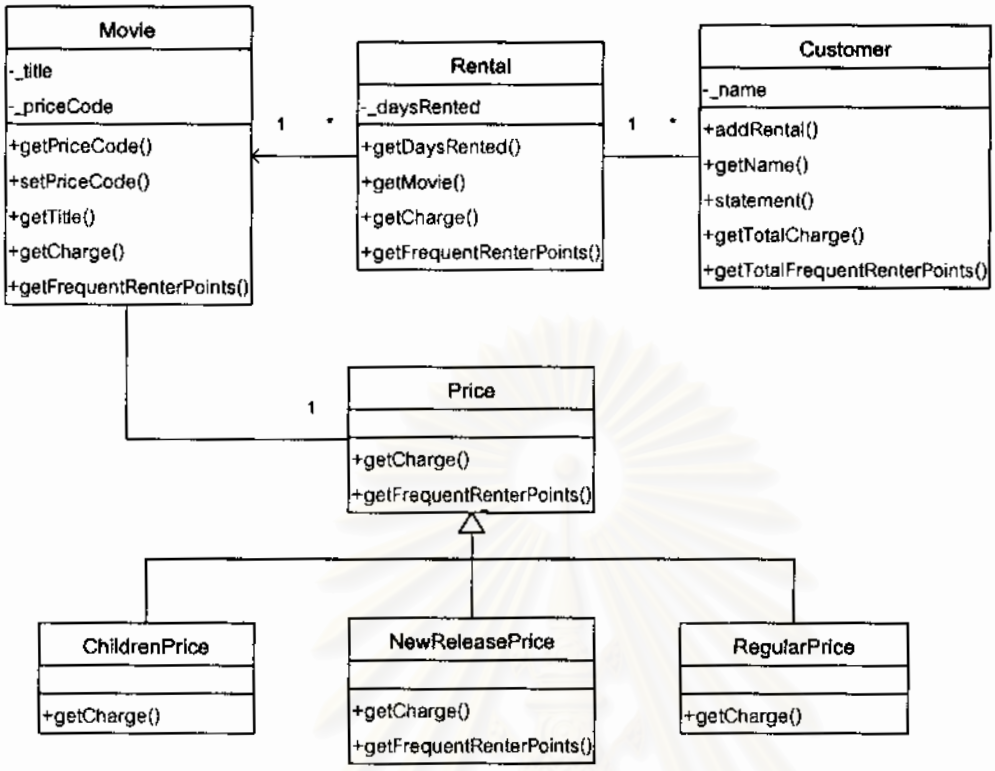
ผู้วิจัยได้ทำการทดสอบความน่าเชื่อถือของมาตรวัดร่องรอยที่ไม่ดีแบบ Data Class และ Refused Bequest โดยนำโปรแกรมต้นฉบับมาตรวจจับร่องรอยที่ไม่ดีโดยใช้เครื่องมือที่ช่วยในการคำนวณค่ามาตรวัดและตรวจจับร่องรอยที่ไม่ดีที่พัฒนาขึ้น จากนั้นจะเลือกวิธีการรีแฟคทอริงที่เหมาะสมตามหัวข้อ 4.1 และ 4.2 เพื่อแก้ไขร่องรอยที่ไม่ดีที่พบให้ดีขึ้น หลังจากได้โปรแกรมต้นฉบับที่ทำการรีแฟคทอริงแล้ว นำมาคำนวณค่ามาตรวัดอีกครั้ง เพื่อเปรียบเทียบค่ามาตรวัดก่อนและหลังการประยุกต์ใช้รีแฟคทอริง แบ่งเป็น 2 ขั้นตอน คือ ขั้นตอนการคำนวณค่ามาตรวัดร่องรอยที่ไม่ดีและขั้นตอนการประเมินความสามารถของมาตรวัดร่องรอยที่ไม่ดี ดังรายละเอียดต่อไปนี้

4.3.1 ขั้นตอนการคำนวณค่ามาตรวัดร่องรอยที่ไม่ดี

ในขั้นตอนนี้ใช้เครื่องมือที่พัฒนา มาช่วยคำนวณค่ามาตรวัดที่ออกแบบไว้เพื่อตรวจจับร่องรอยที่ไม่ดีจากโปรแกรมต้นฉบับจำนวน 2 โปรแกรม ซึ่งเครื่องมือจะตรวจสอบว่าผลของค่ามาตรวัดร่องรอยที่ไม่ดีอยู่ในข้อกำหนดของร่องรอยที่ดีแบบไหน แล้วใช้วิธีรีแฟคทอริงแก้ไขร่องรอยที่ดีนั้นตามข้อกำหนดของค่าที่พิจารณาในหัวข้อที่ 4.1 และ 4.2 จากนั้นผู้วิจัยจะเลือกวิธีการรีแฟคทอริงตามความเหมาะสมกับโปรแกรมที่นำมาทดสอบและนำโปรแกรมนั้นไปทำรีแฟคทอริงตามวิธีการที่เลือก

4.3.1.1 ผลการทดสอบโปรแกรมที่ 1

โปรแกรมที่ใช้ในการทดสอบโปรแกรมที่ 1 นี้ ได้มาจากตัวอย่างโปรแกรมในหนังสือ Refactoring: Improving the design of existing code เขียนโดย Martin Fowler [1] เป็นโปรแกรมเช่าภาพยนตร์มี 7 คลาส ได้แก่ Movie, Rental, Customer, Price, ChildrenPrice, NewRelesePrice และ RegularPrice โดย ChildrenPrice, NewRelesePrice และ RegularPrice เป็นคลาสลูกของ Price ซึ่งแผนภาพคลาสแสดงรายละเอียดในรูปที่ 4.5 และโปรแกรมต้นฉบับแสดงรายละเอียดในรูปที่ 4.6



รูปที่ 4.5 แสดงแผนภาพคลาสของโปรแกรม 1

```
class Price{
        double getCharge(int daysRented){
                double result = 0;
                return result;
        }
        int getfrequentRenterPoints(int daysRented){
                return 1;
        }
}

class RegularPrice extends Price{
        double getCharge(int daysRented){
                double result = 2;
                if(daysRented > 2)
                        result += (daysRented - 2)* 1.5;
                return result;
        }
}

class NewReleasePrice extends Price{
        double getCharge(int daysRented){
                return daysRented * 3;
        }
        int getfrequentRenterPoints(int daysRented){
                return (daysRented>1)? 2:1;
        }
}
```

```
class ChildrensPrice extends Price{
        double getCharge(int daysRented){
          double result = 1.5;
          if(daysRented > 3)
            result += (daysRented - 3)* 1.5;
          return result;
        }
}
```

รูปที่ 4.6 แสดงโปรแกรมต้นฉบับของโปรแกรม 1 ก่อนทำรีแฟคทอริง

```
class Movie             {
        public static final int CHILDREN = 2;
        public static final int REGULAR = 0;
        public static final int NEW_RELEASE = 1;
        private String _title;
        private int _priceCode;
        private Price _price;
        public Movie (String title, int priceCode){
          _title = title;
          _priceCode = priceCode;
        }
        public int getPriceCode(){
          return _priceCode;
        }
        public void setPriceCode(int arg){
          _priceCode = arg;
        }
        public String getTitle(){
          return _title;
        }
        public double getCharge(int daysRented){
                return _price.getCharge(daysRented);
        }
}
class Rental
{
        private Movie _movie;
        private int _daysRented;
        public Rental (Movie movie, int daysRented){
                _movie = movie;
                _daysRented = daysRented;
                                }
        public int getDaysRented(){
                return _daysRented;
        }
        public Movie getMovie(){
                return _movie;
         }
         double getCharge(){
                double result = 0;
                  switch (getMovie().getPriceCode())
                   {
                        case Movie.REGULAR:
                              result +=2;
                              if (getDaysRented()>2)
                                        result += (getDaysRented()-2) * 1.5;
                              break;
                        case Movie.NEW_RELEASE:
                              result += getDaysRented() * 3;
                              break;
                        case Movie.CHILDREN:
                              result += 1.5;
                              if (getDaysRented()>3)
                                        result += (getDaysRented()-3) * 1.5;
                              break;
                    }
              return result;
   }
   int getFrequentRenterPoints(){
        if((getMovie().getPriceCode()== Movie.NEW_RELEASE)&& getDaysRented()>1)
                return 2;
        else
                return 1;
   }
}
```

รูปที่ 4.6 แสดงโปรแกรมต้นฉบับของโปรแกรม 1 ก่อนทำรีแฟคทอริง (ต่อ)

```
class Rental
{
        private Movie _movie;
        private int _daysRented;
        public Rental (Movie movie, int daysRented)
        {
                _movie = movie;
                _daysRented = daysRented;
        }
        public int getDaysRented()
        {
                return _daysRented;
        }
        public Movie getMovie()
        {
                return _movie;
         }
         double getCharge()
        {
                double result = 0;
                  switch (getMovie().getPriceCode())
                   {
                        case Movie.REGULAR:
                             result +=2;
                             if (getDaysRented()>2)
                                        result += (getDaysRented()-2) * 1.5;
                             break;
                        case Movie.NEW_RELEASE:
                             result += getDaysRented() * 3;
                             break;
                        case Movie.CHILDREN:
                             result += 1.5;
                             if (getDaysRented()>3)
                                        result += (getDaysRented()-3) * 1.5;
                             break;
                   }
          return result;
         }
         int getFrequentRenterPoints()
        {
        if((getMovie().getPriceCode()== Movie.NEW_RELEASE)&& getDaysRented()>1)
                return 2;
        else
                return 1;
        }
}

class Customer
{
         private String _name;
         private Vector _rental = new Vector();
          private double totalAmount ;
         public Customer(String name)
        {
                _name = name;
         }
        public void addRental(Rental arg)

        {

                 _rental.addElement(arg);

        }
```

รูปที่ 4.6 แสดงโปรแกรม 1 ของโปรแกรมต้นฉบับก่อนทำรีแฟคทอริง (ต่อ)

```
    public String getName()
   {
          return _name;
    }
    public String Statement()
   {
                int frequentRenterPoints = 0;
                Enumeration rentals = _rental.elements();
                String result = "Rental Record for "+ getName() + "\n";
                while(rentals.hasMoreElements()){
                  double thisAmount = 0;
                  Rental each = (Rental) rentals.nextElement();
                  frequentRenterPoints += each.getFrequentRenterPoints();
                  result += "\t" + each.getMovie().getTitle()+"\t"+
                                  String.valueOf(each.getCharge())+"\n";
                }
          result += "Amount owed is "+String.valueOf(getTotalCharge())+"\n";
          result += "You earned"+String.valueOf(frequentRenterPoints)+"frequent renter pointers";
          return result;
   }
   public double getTotalCharge()
   {
          double result = 0;
          Enumeration rentals = _rental.elements();
          while(rentals.hasMoreElements()){
                Rental each = (Rental) rentals.nextElement();
                result += each.getCharge();
          }
          return result;
    }
  private setTotalAmount()
  {
          totalAmount  = 0;
  }
}
```

รูปที่ 4.6 แสดงโปรแกรม 1 ของโปรแกรมต้นฉบับก่อนทำรีแฟคทอริง (ต่อ)

เมื่อนำโปรแกรม 1 นี้มาตรวจจับร่องรอยที่ไม่ดีด้วยเครื่องมือที่พัฒนาขึ้น จะได้มาตรวัด 2 ประเภทคือ มาตรวัดสำหรับเมทธอด และคุณลักษณะ ของร่องรอยที่ไม่ดีแบบ Data Class และ Refused Bequest ดังตารางที่ 4.5, 4.6, 4.7 และ 4.8 ตามลำดับ ยกตัวอย่าง เช่น ในตารางที่ 4.5 คลาส Movie เมทธอด getCharge มีค่ามาตรวัด NSM=0, NSOC=0, NGM=1, NGOC = 0 และ NCOM ไม่มีค่าเพราะว่า เมทธอด getCharge เป็นเมทธอดที่ไม่ได้สืบทอดจากคลาสแม่ แต่เมทธอด getCharge ของคลาส ChildrensPrice มาตรวัด NCOM มีค่าเพราะว่า เมทธอด getCharge เป็นเมทธอดที่สืบทอดจากคลาสแม่คือ คลาส Price ส่วนมาตรวัด NSMP ของเมทธอด getCharge=3 คือจำนวนคลาสลูกที่มีเมทธอด getCharge สืบทอดมาจาก

คลาสแม่ และมาตรวัด SNOCM ของเมทธอด getCharge=0 คือผลรวมมาตรวัด NOCM ของคลาสลูกที่มีเมทธอด getCharge ที่สืบทอดจากคลาสแม่ ในตารางที่ 4.6 คลาส Movie คุณลักษณะ CHILDREN มีค่ามาตรวัด NCA=0, NAOC=1 และ NOCA ไม่มีค่าเพราะว่าคุณลักษณะ CHILDREN เป็นคุณลักษณะที่ไม่ได้สืบทอดจากคลาสแม่ มาตรวัด NSAP และ SNOCA ไม่มีค่าเพราะว่าไม่มีคุณลักษณะจากคลาสไหนที่สืบทอดจากคลาสแม่ และช่องที่แรงเงาของเมทธอดคือค่าของมาตรวัดที่อยู่ในช่วงของข้อกำหนดที่พิจารณาร่องรอยที่ไม่ดี

ตารางที่ 4.5 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Class

เมทธอด		มาตรวัด			
		NSM	NSOC	NGM	NGOC
คลาส Movie	getCharge	-	-	1	0
	getPriceCode	-	-	1	0
	getTitle	-	-	1	0
	setPriceCode	1	0	0	0
คลาส Rental	getCharge	-	-	6	2
	getDaysRented	-	-	1	0
	getFrequentRenterPoints	-	-	2	1
	getMovie	-	-	1	1
คลาส Customer	Statement	-	-	-	-
	addRental	-	-	-	-
	getName	-	-	1	0
	getTotalCharge	-	-	1	0
	setTotalAmount	0	0	-	-
คลาส Price	getCharge	-	-	0	0
	getfrequentRenterPoints	-	-	0	0
คลาส ChildrensPrice	getCharge	-	-	0	0
คลาส NewReleasePrice	getCharge	-	-	0	0
	getfrequentRenterPoints	-	-	0	0
คลาส RegularPrice	getCharge	-	-	0	0

- คือ ไม่มีค่าของมาตรวัดนั้น

ตารางที่ 4.6 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Refused Bequest

เมทธอด	มาตรวัด	
	NSMP	SNOCM
getCharge	3	0
getfrequentRenterPoints	1	0

ตารางที่ 4.7 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Data Class

คุณลักษณะ		มาตรวัด	
		NCA	NAOC
คลาส Movie	CHILDREN	0	1
	NEW_RELEASE	0	2
	REGULAR	0	1
	_price [Movie]	1	0
	_priceCode [Movie]	3	0
	_title [Movie]	2	0
คลาส Rental	_daysRented [Rental]	2	0
	_movie [Rental]	2	0
คลาส Customer	_name [Customer]	2	0
	_rental [Customer]	3	0

ตารางที่ 4.8 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Refused Bequest

คุณลักษณะ	มาตรวัด	
	NSAP	SNOCA
_title	-	0
_priceCode	-	0

จากผลของมาตรวัดดังตารางที่ 4.5, 4.6, 4.7 และ 4.8 พบว่าค่ามาตรวัดสำหรับเมทธอดอยู่ในช่วงของข้อกำหนดที่พิจารณาเป็นร่องรอยที่ไม่ดี 2 แบบ คือ

1. Data Class ได้แก่เมทธอด getCharge, getPriceCode, getTitle, setPriceCode ของคลาส Movie เมทธอด getDaysRented ของคลาส rental เมทธอด getName และ getTotalCharge ของคลาส Customer เพราะพบค่ามาตรวัด NGM =1 และ NGOC =0

หรือ NSM =1 และ NSOC =0 และเมทธอด setTotalAmount ของคลาส Customer มีค่ามาตรวัด NSM+NSOC =0 ซึ่งค่ามาตรวัดของเมทธอดเหล่านี้อยู่ในข้อกำหนดของค่าที่พิจารณาร่องรอยที่ไม่ดีแบบ Data Class ควรช้วิธีรีแฟคทอริง Hide method และ Remove setting method ตามลำดับ

2. Refused Bequest ได้แก่เมทธอด getfrequentRenterPoints ของคลาส Customer เพราะพบค่ามาตรวัด SNOCM = 0 และ NSMP = 1 ซึ่งอยู่ในข้อกำหนดของค่าที่พิจารณาร่องรอยที่ไม่ดีแบบ Refused Bequest ควรช้วิธีรีแฟคทอริง Push down method

แต่ผลของมาตรวัดของตารางที่ 4.7 และ 4.8 พบว่าค่ามาตรวัดสำหรับคุณลักษณะไม่อยู่ในช่วงของข้อกำหนดที่พิจารณา ดังนั้นจึงไม่มีคุณลักษณะของคลาสไหนที่เกิดร่องรอยที่ไม่ดี

ดังนั้นผู้วิจัยจึงได้เสนอวิธีการรีแฟคทอริง ให้เลือกประยุกต์ใช้กับโปรแกรมต้นฉบับ เพื่อปรับปรุงร่องรอยทั้ง 2 แบบที่พบแบ่งเป็น 3 กรณี

กรณีที่ 1 วิธี Hide method โดยกำหนดประเภทของเมทธอดเป็น Private ได้แก่เมทธอด getCharge, getPriceCode, getTitle, setPriceCode ของคลาส Movie เมทธอด getDaysRented ของคลาส Rental เมทธอด getName และ getTotalCharge ของคลาส Customer ดังแสดงในรูปที่ 4.7

```
class Movie          {                                    class Rental{
        …                                                          …
        private int getPriceCode(){                                private  int getDaysRented(){
          return _priceCode;                                               return _daysRented;
        }                                                          }
                private void setPriceCode(int arg){                …
          _priceCode = arg;                                }
        }
        private String getTitle(){
          return _title;
        }
        private  double getCharge(int daysRented){
                return _price.getCharge(daysRented);
        }
}

class Customer{
        …
    private  String getName(){ return _name; }
    private double getTotalCharge(){
        double result = 0;
        Enumeration rentals = _rental.elements();
        while(rentals.hasMoreElements()){
                Rental each = (Rental) rentals.nextElement();
                result += each.getCharge();
        }
        return result;
  }
}
```

รูปที่ 4.7 แสดงโปรแกรม 1 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 1

กรณีที่ 2 วิธี Remove setting method โดยการลบเมทธอดที่กำหนดคุณลักษณะออกคือเมทธอด setTotalAmount ของคลาส Customer และกำหนดค่าเริ่มต้นให้กับคุณลักษณะ totalAmount ดังแสดงในรูปที่ 4.8

```
class Customer
{
      private String _name;
      private Vector _rental = new Vector();
      private double totalAmount=0 ;
      public Customer(String name){
            _name = name;
      }
      public void addRental(Rental arg){ _rental.addElement(arg); }
      public String getName(){ return _name; }
      public String Statement(){
                        int frequentRenterPoints = 0;
                        Enumeration rentals = _rental.elements();
                        String result = "Rental Record for "+ getName() + "\n";
                        while(rentals.hasMoreElements()){
                          double thisAmount = 0;
                          Rental each = (Rental) rentals.nextElement();
                          frequentRenterPoints += each.getFrequentRenterPoints();
                          result += "\t" + each.getMovie().getTitle()+"\t"+
                                          String.valueOf(each.getCharge())+"\n";
                        }
            result += "Amount owed is "+String.valueOf(getTotalCharge())+"\n";
            result += "You earned"+String.valueOf(frequentRenterPoints)+"frequent renter pointers";
            return result;
   }
            …
}
```

รูปที่ 4.8 แสดงโปรแกรม 1 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 2

กรณีที่ 3 วิธี Push down method โดยการลบเมทธอด getfrequentRenterPoints จากคลาสแม่คือคลาส Price แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่งคือคลาส NewReleasePrice ดังแสดงในรูปที่ 4.9

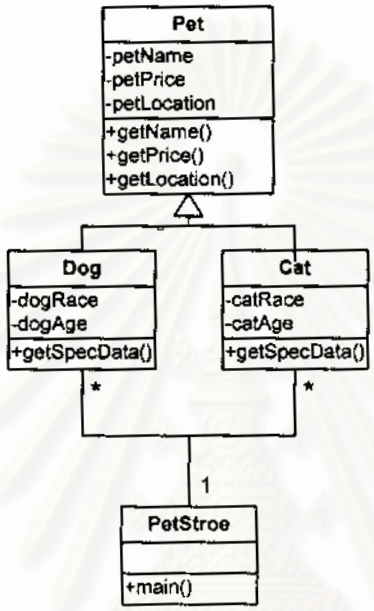
```
class Price{
            double getCharge(int daysRented){
                        double result = 0;
                        return result;
            }
}
```

```
class NewReleasePrice extends Price{
            double getCharge(int daysRented){
                        return daysRented * 3;
            }
            int getfrequentRenterPoints(int daysRented){
                        return (daysRented>1)? 2:1;
            }
}
```

รูปที่ 4.9 แสดงโปรแกรม 1 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 3

4.3.1.2 ผลการทดสอบโปรแกรมที่ 2

โปรแกรมที่ใช้ในการทดสอบโปรแกรมที่ 2 นี้ ได้มาจากตัวอย่างโปรแกรมในหนังสือ Java for Engineer เขียนโดย Martin Fowler [14] เป็นโปรแกรมเซ่าภาพยนตร์มี 4 คลาส ได้แก่ PetStore, Pet, Dog และ Cat โดย Dog และ Cat เป็นคลาสลูกของ Pet ซึ่งแผนภาพคลาสแสดงรายละเอียดในรูปที่ 4.10 และโปรแกรมต้นฉบับแสดงรายละเอียดในรูปที่ 4.11



รูปที่ 4.10 แสดงแผนภาพคลาสของโปรแกรม 2

```
abstract class Pet
{
    private String petName;
    private int petLocation;
    private double petPrice;

  public Pet(String name, int location, double price) {
    this.petName = name;
    this.petLocation = location;
            this.petPrice = price;
  }
  public void getName() {
    System.out.println ("Pet name: " + this.petName);
            return;
  }
  public void getLocation() {
    System.out.println("Pet location: " + this.petLocation);
            return;
  }
  public void getPrice() {
    System.out.println("Pet price: " + this.petPrice);
            return;
  }
   public abstract void getSpecData();
  }


  class Dog extends Pet
  {
   String dogRace;
   int dogAge;
  constructor
   public Dog(String name, int loc, double price,
String race)
   {
   super(name, loc, price);
   this.dogRace = race;
   }
   public void getSpecData() {
    System.out.println("Dog race :" +
this.dogRace);
            return;
   }
  }
```

รูปที่ 4.11 แสดง โปรแกรมต้นฉบับของโปรแกรม 2 ก่อนทำรีแฟคทอริง

```
class Cat extends Pet
{
  String catRace;
  int catAge;

  public Cat(String name, int loc, double price, String race) {
  super(name, loc, price);
  this.catRace = race;
  }

  public void getSpecData()  {
    System.out.println("Cat race: " + this.catRace);
              return;
  }
}


public class PetStore
{
    public static void main(String[] args)
    {
    Dog dog1 = new Dog("Fido",2, 12.95,"Spaniel");
    Dog dog2 = new Dog("Atila",3, 20.75,"Hound");

              // Create a Cat and a Bird object
              Cat cat1 = new Cat("Fifo", 3, 15.95, "Siamese");

              // Display pet data using super class and sub class
              // methods
              dog1.getName();     // Method in super class
              dog1.getPrice();
              dog2.getName();     // Methods in super class
              dog2.getLocation();
              dog2.getSpecData(); // Method in sub class
              cat1.getName();     // Methods in super class
              cat1.getLocation();
              cat1.getSpecData();
              System.out.println("Age Cat : "+cat1.catAge);
              if(cat1. catAge >0 && cat1. catAge <3)
                        System.out.println("Young cat");
              if(cat1.catAge ==3)
                        System.out.println("Teen cat");
              if(cat1. catAge >3 && cat1. catAge <7)
                        System.out.println("Old cat");
    }
}
```

รูปที่ 4.11 แสดง โปรแกรมต้นฉบับของโปรแกรม 2 ก่อนทำรีแฟคทอริง (ต่อ)

เมื่อนำโปรแกรม 2 นี้มาตรวจจับร่องรอยที่ไม่ดีด้วยเครื่องมือที่พัฒนาขึ้น จะได้มาตรวัด 2 ประเภทคือ มาตรวัดสำหรับเมทธอด และคุณลักษณะของร่องรอยที่ไม่ดีแบบ Data Class และ Refused Bequest ดังตารางที่ 4.9, 4.10, 4.11 และ 4.12 ตามลำดับ ยกตัวอย่างเช่น ในตารางที่ 4.11 คุณลักษณะ catAge ในคลาส Cat มาตรวัด NCA มีค่า 0 หมายถึงไม่มีการเรียกใช้คุณลักษณะ catAge ในคลาส Cat ซึ่งเป็นคลาสตัวเองเลย และมาตรวัด NAOC มีค่าเท่ากับ 6

หมายถึง มีการเรียกใช้คุณลักษณะ catAge จากคลาสคลาสอื่นๆ เป็นจำนวน 6 ครั้ง และตารางที่ 4.12 คุณลักษณะ petPrice เป็นคุณลักษณะที่สืบทอดจากคลาสแม่คือ คลาส Pet ซึ่งมีมาตรวัด NSAP = 1 และ SNOCA = 0 หมายถึง มีคลาสลูกเดียวที่เรียกใช้ คุณลักษณะ petPrice และไม่มีคลาสอื่นมาเรียกใช้ด้วย

ตารางที่ 4.9 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Class

เมทธอด		มาตรวัด			
		NSM	NSOC	NGM	NGOC
คลาส PetStore	main	-	-	-	-
คลาส Cat	getName	-	-	0	1
	getLocation	-	-	0	1
	getPrice	-	-	0	0
	getSpecData	-	-	0	1
คลาส Dog	getName	-	-	0	2
	getLocation	-	-	0	1
	getPrice	-	-	0	1
	getSpecData	-	-	0	1
คลาส Pet	getName	-	-	0	0
	getLocation	-	-	0	0
	getPrice	-	-	0	0
	getSpecData	-	-	0	0

- คือ ไม่มีค่าของมาตรวัดนั้น

ตารางที่ 4.10 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Refused Bequest

เมทธอด	มาตรวัด	
	NSMP	SNOCM
getName	2	2
getLocation	2	2
getPrice	1	1
getSpecData	2	2

ตารางที่ 4.11 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Data Class

คุณลักษณะ		มาตรวัด	
		NCA	NAOC
คลาส Pet	petName	0	0
	petLocation	0	0
	petPrice	0	0
คลาส Dog	petName	0	2
	petLocation	0	1
	petPrice	0	1
	dogRace	0	1
	dogAge	0	2
คลาส Cat	petName	0	1
	petLocation	0	1
	petPrice	0	0
	catRace	1	0
	catAge	0	6

ตารางที่ 4.12 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Refused Bequest

คุณลักษณะ	มาตรวัด	
	NSAP	SNOCA
petName	2	1
petLocation	2	1
petPrice	1	0

จากผลของมาตรวัดดังตารางที่ 4.9, 4.10, 4.11 และ 4.12 พบว่าค่ามาตรวัดสำหรับเมทธอดอยู่ในช่วงของข้อกำหนดที่พิจารณาคือช่องที่แรเงาในตารางเป็นร่องรอยที่ไม่ดี 2 แบบ คือ

1. Data Class ได้แก่ คุณลักษณะ catAge ของคลาส Cat เพราะว่าพบมาตรวัด NAOC = 6 ซึ่งค่ามาตรวัดนี้อยู่ในข้อกำหนดของค่าที่พิจารณาร่องรอยที่ไม่ดีแบบ Data Class ที่ควรใช้วิธีรีแฟคทอริง Encapsulate field ในการแก้ปัญหา

2. Refused Bequest ได้แก่ เมทธอด getPrice และคุณลักษณะ petPrice ของคลาส Dog เพราะว่าพบมาตรวัด NSMP =1 และ SNOCM =1 ของมาตรวัดสำหรับเมทธอด มาตรวัด NSAP =1 และ SNOCA =0 ของมาตรวัดสำหรับคุณลักษณะ ซึ่งค่ามาตรวัดนี้อยู่ในข้อกำหนดของค่าที่พิจารณาร่องรอยที่ไม่ดีแบบ Refused Bequest ที่ควรใช้วิธีรีแฟคทอริง Push down method และ Push down field ในการแก้ปัญหาตามลำดับ

ดังนั้นผู้วิจัยจึงได้เสนอวิธีการรีแฟคทอริง ให้เลือกประยุกต์ใช้กับโปรแกรมต้นฉบับ เพื่อปรับปรุงร่องรอยทั้ง 2 แบบที่พบแบ่งเป็น 3 กรณี

กรณีที่ 1 วิธี Encapsulate field คือการสร้างเอคเซสเซอร์เมทธอดให้กับคุณลักษณะ catAge ของคลาส Cat ดังแสดงในรูปที่ 4.12

```
class Cat extends Pet
{
  String catRace;
  int catAge;

  public Cat(String name, int loc, double price, String race) {
  super(name, loc, price);
  this.catRace = race;
  }
  public int getCatAge(){
            return catAge;
  }
  public void setCatAge(int arg){
            catAge = arg;
  }
  public void getSpecData()  {
    System.out.println("Cat race: " + this.catRace);
            return;
  }
}
```

รูปที่ 4.12 แสดงโปรแกรม 2 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 1

กรณีที่ 2 วิธี Push down method โดยการลบเมทธอด getPrice จากคลาสแม่คือคลาส Pet แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่งคือคลาส Dog ดังแสดงในรูปที่ 4.13

กรณีที่ 3 วิธี Push down field โดยการลบคุณลักษณะ petPrice จากคลาสแม่คือคลาส Pet แล้วเพิ่มคุณลักษณะนั้นในคลาสลูกที่ใช้งานคุณลักษณะนี้ซึ่งคือคลาส Dog ดังแสดงในรูปที่ 4.14

```
abstract class Pet
{
    private String petName;
    private int petLocation;
     private double petPrice;

  public Pet(String name, int location, double price)  {
    this.petName = name;
    this.petLocation = location;
              this.petPrice = price;
  }
  public void getName()  {
    System.out.println ("Pet name: " + this.petName);
              return;
  }
  public void getLocation()  {
    System.out.println("Pet location: " + this.petLocation);
              return;
  }
    public abstract void getSpecData();
  }
```

```
  class Dog extends Pet
  {
    String dogRace;
    int dogAge;

  constructor
    public Dog(String name, int loc, double price,
String race)
    {
    super(name, loc, price);
    this.dogRace = race;
    }
    public void getSpecData()  {
      System.out.println("Dog race :" +
this.dogRace);
              return;
    }
  public void getPrice()  {
    System.out.println("Pet price: " +
this.petPrice);
              return;
  }
```

รูปที่ 4.13 แสดงโปรแกรม 2 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 2

```
abstract class Pet
{
    private String petName;
    private int petLocation;


  public Pet(String name, int location, double price)  {
    this.petName = name;
    this.petLocation = location;
              this.petPrice = price;
  }
  public void getName()  {
    System.out.println ("Pet name: " + this.petName);
              return;
  }
  public void getLocation()  {
    System.out.println("Pet location: " + this.petLocation);
              return;
  }
  public void getPrice()  {
    System.out.println("Pet price: " + this.petPrice);
              return;
  }
```

```
    public abstract void getSpecData();
  }


  class Dog extends Pet
  {
    String dogRace;
    int dogAge;
  private double petPrice;

    public Dog(String name, int loc, double price,
String race)
    {
    super(name, loc, price);
    this.dogRace = race;
    }
    public void getSpecData()  {
      System.out.println("Dog race :" +
this.dogRace);
              return;
    }
  }
```

รูปที่ 4.14 แสดงโปรแกรม 2 หลังการประยุกต์ใช้รีแฟคทอริงในกรณีที่ 3

4.3.2 ขั้นตอนการประเมินความสามารถของมาตรวัดร่องรอยที่ไม่ดี

ขั้นตอนนี้จะประเมินความสามารถมาตรวัดร่องรอยที่ไม่ดี โดยประเมินจากมาตรวัดหลังจากการทำรีแฟคทอริงแล้วควรมีค่าของมาตรวัดที่ดีขึ้น

4.3.2.1 ผลการทดสอบโปรแกรมที่ 1 ทั้ง 3 กรณีมีรายละเอียดมาตรวัดดังนี้

ตารางที่ 4.13 ค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Class หลังประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 1 ของกรณีที่ 1

เมทธอด		มาตรวัด				
		NSM	NSOC	NGM	NGOC	NOCM
คลาส Movie	getCharge	-	-	-	-	-
	getPriceCode	-	-	-	-	-
	getTitle	-	-	-	-	-
	setPriceCode	-	-	0	0	-
คลาส Rental	getCharge	-	-	6	2	-
	getDaysRented	-	-	-	-	-
	getFrequentRenterPoints	-	-	2	1	-
	getMovie	-	-	1	1	-
คลาส Customer	Statement	-	-	-	-	-
	addRental	-	-	-	-	-
	getName	-	-	-	-	-
	getTotalCharge	-	-	-	-	-
	setTotalAmount	0	0	-	-	-
คลาส Price	getCharge	-	-	0	0	-
	getfrequentRenterPoints	-	-	0	0	-
คลาส ChildrensPrice	getCharge	-	-	0	0	0
คลาส NewReleasePrice	getCharge	-	-	0	0	0
	getfrequentRenterPoints	-	-	0	0	0
คลาส RegularPrice	getCharge	-	-	0	0	0

กรณีที่ 1 เนื่องจากมีการกำหนดประเภทของเมทธอดจาก Public เป็น Private ช่องที่แรงเงาในตารางที่ 4.5 ที่ไม่ใช่เมทธอด setTotalAmount ของคลาส Customer และตารางที่ 4.13 จะพบว่าหลังการทำรีแฟคทอริงแล้วค่าของมาตรวัด NSM, NSOC, NGM และ NGOC ไม่มีค่าคือไม่เกิดร่องรอยที่ไม่ดีแบบ Data Class ส่วนมาตรวัดอื่นเท่าเดิม

ตารางที่ 4.14 ค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Classหลังประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 1 ของกรณีที่ 2

เมทธอด		มาตรวัด			
		NSM	NSOC	NGM	NGOC
คลาส Movie	getCharge	-	-	-	-
	getPriceCode	-	-	-	-
	getTitle	-	-	-	-
	setPriceCode	-	-	0	0
คลาส Rental	getCharge	-	-	6	2
	getDaysRented	-	-	-	-
	getFrequentRenterPoints	-	-	2	1
	getMovie	-	-	1	1
คลาส Customer	Statement	-	-	-	-
	addRental	-	-	-	-
	getName	-	-	-	-
	getTotalCharge	-	-	-	-
คลาส Price	getCharge	-	-	0	0
	getfrequentRenterPoints	-	-	0	0
คลาส ChildrensPrice	getCharge	-	-	0	0
คลาส NewReleasePrice	getCharge	-	-	0	0
	getfrequentRenterPoints	-	-	0	0
คลาส RegularPrice	getCharge	-	-	0	0

กรณีที่ 2 เนื่องจากมีการลบเมทธอด setTotalAmount ของคลาส Customer ออกและกำหนดค่าเริ่มต้นให้กับคุณลักษณะ totalAmount จากตารางที่ 4.14 พบว่า

หลังการทำรีแฟคทอริงแล้ว เมทธอด setTotalAmount ของคลาส Customer หายไปดังนั้นจึงไม่เกิดร่องรอยที่ไม่ดีแบบ Data Class ส่วนมาตรวัดอื่นที่ค่าเท่าเดิม

ตารางที่ 4.15 ค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Refused Bequest หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 1 ของกรณีที่ 3

เมทธอด	มาตรวัด	
	NSMP	SNOCM
getCharge	3	0

กรณีที่ 3 เนื่องจากมีการลบเมทธอด getfrequentRenterPoints จากคลาสแม่ คือ คลาส Price แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่ง คือ คลาส NewReleasePrice จากตารางที่ 4.15 พบว่าหลังการทำรีแฟคทอริงแล้ว เมทธอด getfrequentRenterPoints ไม่มีจึงไม่มีค่ามาตรวัด ดังนั้นจึงไม่เกิดร่องรอยที่ไม่ดีแบบ Refused Bequest

ดังนั้นวิธีรีแฟคทอริงที่เสนอทั้ง 3 กรณี ควรประยุกต์ใช้เพื่อปรับปรุงร่องรอยที่ไม่ดีที่พบภายในโปรแกรมต้นฉบับโปรแกรมที่ 1

4.3.2.2 ผลการทดสอบโปรแกรมที่ 2 ทั้ง 3 กรณีมีรายละเอียดมาตรวัดดังนี้

ตารางที่ 4.16 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Class หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 2 ของกรณีที่ 1

เมทธอด		มาตรวัด			
		NSM	NSOC	NGM	NGOC
คลาส PetStore	main	-	-	-	-
คลาส Cat	getName	-	-	0	1
	getLocation	-	-	0	1
	getPrice	-	-	0	0
	getSpecData	-	-	0	1
	getCatAge	-	-	0	6
	setCatAge	0	0	-	-
คลาส Dog	getName	-	-	0	2
	getLocation	-	-	0	1
	getPrice	-	-	0	1
	getSpecData	-	-	0	1

ตารางที่ 4.16 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Data Class หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 2 ของกรณีที่ 1 (ต่อ)

เมทธอด		มาตรวัด			
		NSM	NSOC	NGM	NGOC
คลาส Pet	getName	-	-	0	0
	getLocation	-	-	0	0
	getPrice	-	-	0	0
	getSpecData	-	-	0	0

- คือ ไม่มีค่าของมาตรวัดนั้น

ตารางที่ 4.17 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Data Class หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 2 ของกรณีที่ 1

คุณลักษณะ		มาตรวัด	
		NCA	NAOC
คลาส Pet	petName	0	0
	petLocation	0	0
	petPrice	0	0
คลาส Dog	petName	0	2
	petLocation	0	1
	petPrice	0	1
	dogRace	0	1
	dogAge	0	2
คลาส Cat	petName	0	1
	petLocation	0	1
	petPrice	0	0
	catRace	1	0
	catAge	0	0

กรณีที่ 1 เนื่องจากมีการสร้างเมทธอดเอคเซสเซอร์ คือ เมทธอด getCatAge และ setCatAge ของคุณลักษณะ catAge ในคลาส Cat จากตารางที่ 4.16 และ 4.17 พบว่าหลังการทำรีแฟคทอริงแล้ว มีการใช้เมทธอดทั้งสองแทนการเรียกใช้คุณลักษณะ catAge โดยตรง

ตารางที่ 4.17 แสดงค่ามาตรวัดสำหรับเมทธอดของร่องรอยที่ไม่ดีแบบ Refused Bequest หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 2 ของกรณีที่ 2

เมทธอด	มาตรวัด	
	NSMP	SNOCM
getName	2	2
getLocation	2	2
getSpecData	2	2

กรณีที่ 2 เนื่องจากมีการลบเมทธอด getPrice จากคลาสแม่ คือ คลาส Pet แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่ง คือ คลาส Dog จากตารางที่ 4.18 พบว่าหลังการทำรีแฟคทอริงแล้ว เมทธอด getPrice ไม่มีจึงไม่มีค่ามาตรวัด ดังนั้นจึงไม่เกิดร่องรอยที่ไม่ดีแบบ Refused Bequest

ตารางที่ 4.18 แสดงค่ามาตรวัดสำหรับคุณลักษณะของร่องรอยที่ไม่ดีแบบ Refused Bequest หลังการประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรม 2 ของกรณีที่ 3

คุณลักษณะ	มาตรวัด	
	NSAP	SNOCA
petName	2	1
petLocation	2	1

กรณีที่ 3 เนื่องจากมีการลบคุณลักษณะ petPrice จากคลาสแม่ คือ คลาส Pet แล้วเพิ่มคุณลักษณะนั้นในคลาสลูกที่ใช้งานคุณลักษณะนี้ซึ่ง คือ คลาส Dog จากตารางที่ 4.19 พบว่าหลังการทำรีแฟคทอริงแล้ว คุณลักษณะ petPrice ไม่มีจึงไม่มีค่ามาตรวัด ดังนั้นจึงไม่เกิดร่องรอยที่ไม่ดีแบบ Refused Bequest

ดังนั้นวิธีรีแฟคทอริงที่เสนอทั้ง 3 กรณี ควรประยุกต์ใช้เพื่อปรับปรุงร่องรอยที่ไม่ดีที่พบภายในโปรแกรมต้นฉบับโปรแกรมที่ 2

การประเมินความสามารถมาตรวัดร่องรอยที่ไม่ดี นอกจากประเมินด้วยมาตรวัดตรวจจับร่องรอยที่ไม่ดีแล้ว ควรประเมินจากผลกระทบของซอฟต์แวร์หลังการทำรีแฟคทอริงด้วย เนื่องจากเป้าหมายของการทำรีแฟคทอริง คือ ช่วยเพิ่มความสามารถทางด้านการบำรุงรักษาซอฟต์แวร์ ดังนั้นสามารถประเมินผลกระทบของการทำรีแฟคทอริงได้จากการตรวจสอบว่าหลังจากการทำรีแฟคทอริงแล้ว จะเพิ่มความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งมีวิธีการตรวจสอบดังนี้ เริ่มจากเลือกมาตรวัดที่เหมาะสมทางด้านการบำรุงรักษา ในที่นี้เลือกมาตรวัดระดับของการขาดความสัมพันธ์ภายในคลาส (LCOM) และมาตรวัดการเข้าคู่กันระหว่าง

วัตถุ (CBO) คำนวณโดยเครื่องมือ MTOOP รุ่นที่ 2 หลังจากนั้นนำโปรแกรมต้นฉบับจากขั้นตอนที่ 4.3.1 มาคำนวณค่ามาตรวัด เพื่อเปรียบเทียบค่ามาตรวัดก่อน และหลังการประยุกต์ใช้รีแฟคทอริง ซึ่งแสดงผลดังตารางที่ 4.19 และ 4.20

ตารางที่ 4.19 การเปรียบเทียบค่ามาตรวัด LCOM และ CBO ก่อนและหลังประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรมต้นฉบับที่ 1

คลาส		มาตรวัด LCOM		มาตรวัด CBO	
		ก่อน	หลัง	ก่อน	หลัง
กรณีที่ 1	Movie	1.0	1.0	1	1
	Rental	0.75	0.75	3	3
	Customer	0.8	0.8	2	2
	Price	0.0	0.0	0	0
	ChildrensPrice	0.0	0.0	0	0
	NewReleasePrice	0.0	0.0	0	0
	RegularPrice	0.0	0.0	0	0
กรณีที่ 2	Movie	1.0	1.0	1	1
	Rental	0.75	0.75	3	3
	Customer	0.8	0.8	2	2
	Price	0.0	0.0	0	0
	ChildrensPrice	0.0	0.0	0	0
	NewReleasePrice	0.0	0.0	0	0
	RegularPrice	0.0	0.0	0	0
กรณีที่ 3	Movie	1.0	1.0	1	1
	Rental	0.75	0.75	3	3
	Customer	0.8	0.8	2	2
	Price	0.0	0.0	0	0
	ChildrensPrice	0.0	0.0	0	0
	NewReleasePrice	0.0	0.0	0	0
	RegularPrice	0.0	0.0	0	0

จากตารางที่ 4.19 เป็นผลของโปรแกรมต้นฉบับที่ 1 พบว่า

กรณีที่ 1 เนื่องจากมีการกำหนดประเภทของเมทธอดจาก Public เป็น Private เท่านั้น ดังนั้นค่า LCOM และ CBO ของทุกคลาสจึงไม่มีการเปลี่ยน

กรณีที่ 2 เนื่องจากมีการลบเมทธอด setTotalAmount ของคลาส Customer ออก ดังนั้นค่า LCOM และ CBO ของทุกคลาสจึงไม่มีการเปลี่ยน

กรณีที่ 3 เนื่องจากมีการลบเมทธอด getfrequentRenterPoints จากคลาสแม่ คือ คลาส Price แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่ง คือ คลาส NewReleasePrice ดังนั้นค่า LCOM และ CBO ของทุกคลาสจึงไม่มีการเปลี่ยน

ตารางที่ 4.20 การเปรียบเทียบค่ามาตรวัด LCOM และ CBO ก่อนและหลังประยุกต์ใช้วิธีรีแฟคทอริงของโปรแกรมต้นฉบับที่ 2

คลาส		มาตรวัด LCOM		มาตรวัด CBO	
		ก่อน	หลัง	ก่อน	หลัง
กรณีที่ 1	Pet	1.25	1.25	0	0
	Dog	2	2	0	0
	Cat	2	1	0	0
	PetStore	0	0	3	3
กรณีที่ 2	Pet	1.25	1.25	0	0
	Dog	2	2	0	0
	Cat	2	1.5	0	0
	PetStore	0	0	3	3
กรณีที่ 3	Pet	1.25	1.25	0	0
	Dog	2	2	0	0
	Cat	2	1.5	0	0
	PetStore	0	0	3	3

จากตารางที่ 4.20 เป็นผลของโปรแกรมต้นฉบับที่ 2 พบว่า

กรณีที่ 1 เนื่องจากมีการสร้างเอคเซสเซอร์เมทธอดให้กับ คุณลักษณะ catAge ของคลาส Cat ดังนั้นค่า CBO ของทุกคลาสจึงไม่มีการเปลี่ยน แต่ค่า ค่า LCOM ลดลง แสดงว่าภายในคลาสมีความสัมพันธ์ระหว่าง เมทธอด และ คุณลักษณะมาก ส่งผลให้โปรแกรมมีคุณภาพที่ดีขึ้น

กรณีที่ 2 เนื่องจากมีการลบเมทธอด getPrice จากคลาสแม่ คือ คลาส Pet แล้วเพิ่มเมทธอดนั้นในคลาสลูกที่ใช้งานเมทธอดนี้ซึ่ง คือ คลาส Dog ดังนั้นค่า LCOM และ CBO ของคลาส Dog, Pet และ Pet Store จึงไม่มีการเปลี่ยน แต่คลาส Cat มีค่า LCOM ลดลง เนื่องจากไม่มีเมทธอด getPrice แล้ว

กรณีที่ 3 เนื่องจากมีการลบคุณลักษณะ petPrice จากคลาสแม่ คือ คลาส Pet แล้วเพิ่มคุณลักษณะนั้นในคลาสลูกที่ใช้งานคุณลักษณะนี้ซึ่ง คือ คลาส Dog ดังนั้นค่า LCOM และ CBO ของคลาส Dog, Pet และ Pet Store จึงไม่มีการเปลี่ยน แต่คลาส Cat มีค่า LCOM ลดลงเนื่องจากไม่มีเมทธอด getPrice แล้ว

บทที่ 5

การหาช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีและการนำไปใช้งาน

ในบทนี้จะกล่าวถึงผลที่ได้จากการทดลองในบทที่ 3 ประกอบด้วย คำนวณค่ามาตรวัดทั้ง 10 โปรแกรม สร้างโมเดลการทำนายร่องรอยที่ไม่ดีของมาตรวัดต่างๆ หาช่วงค่าของมาตรวัดต่างๆ ประเมินค่าน่าเชื่อถือของช่วงค่าของมาตรวัดต่างๆ นำเสนอวิธีการนำช่วงค่าของมาตรวัดต่างๆสำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน และเปรียบเทียบช่วงค่าของโปรแกรมภาษาซีพลัสพลัสและจาวา ซึ่งมีรายละเอียดดังนี้

5.1 คำนวณค่ามาตรวัดทั้ง 10 โปรแกรม

ผลจากการให้ค่าร่องรอยที่ไม่ดีและการคำนวณค่ามาตรวัดของร่องรอยที่ไม่ดี 8 แบบ ซึ่งเป็นชุดข้อมูลสอน 10 โปรแกรมและ 2 โปรแกรมเป็นชุดข้อมูลทดสอบ ได้ผลข้อมูลแสดงในภาคผนวก ก

5.2 สร้างโมเดลการทำนายร่องรอยที่ไม่ดีของมาตรวัดต่าง ๆ

งานวิจัยนี้ได้ทดลองสร้างโมเดลการทำนายร่องรอยที่ไม่ดี โดยการนำชุดข้อมูลสอนมาทำการวิเคราะห์ทางสถิติด้วยโปรแกรม SPSS for Windows ผลลัพธ์ที่ได้จากการวิเคราะห์เป็นดังตารางที่ 5.1 และตารางที่ 5.2

จากตารางที่ 5.1 แสดงความสัมพันธ์ระหว่างมาตรวัดกับร่องรอยที่ไม่ดีทั้ง 8 แบบพบว่ามีมาตรวัดที่ถูกตัดออก 6 มาตรวัดคือ NSM NGM และ NGOC ของร่องที่ไม่ดีแบบ Data Class และมาตรวัดของร่องรอยที่ไม่ดีแบบ Lazy Class คือ DIT และ AMC เนื่องจากค่า R มีค่าน้อยกว่า 0.1 ซึ่งหมายความว่ามาตรวัดนั้นมีความสัมพันธ์กับร่องรอยที่ไม่ดีนั้นๆ น้อยมาก ดังนั้นไม่สามารถสร้างโมเดลทำนายร่องรอยที่ไม่ดีได้ และมาตรวัดของร่องรอยที่ไม่ดีแบบ Feature Envy, Refused Bequest ไม่สามารถเก็บข้อมูลที่เกิดร่องรอยที่ไม่ดีแบบนั้นได้ ดังนั้นจึงไม่สามารถหาสัมพันธ์ของมาตรวัดกับร่องรอยที่ไม่ดีเหล่านี้ได้

ค่าตัวเลขที่แสดงในตารางที่ 5.2 คือ ค่าสัมประสิทธิ์ของมาตรวัดแต่ละมาตรวัด (β_1) และค่าคงที่ (β_0) โดยคอลัมน์ร่องรอยที่ไม่ดีคือคอลัมน์ที่แสดงกลุ่มของมาตรวัดของร่องรอยที่ไม่ดีนั้นๆ คอลัมน์มาตรวัดคือมาตรวัดของร่องรอยที่ไม่ดีนั้นๆ คอลัมน์ Unstandardized Coefficients คือค่าสัมประสิทธิ์ของมาตรวัดและค่าคงที่ ตัวอย่างการสร้างโมเดลการทำนายร่องรอยที่ไม่ดีแบบ Data Class โดยใช้มาตรวัด NOPA สามารถเขียนโมเดลการทำนายร่องรอยที่ไม่ดีได้ดังนี้ ค่าร่องรอยที่ไม่ดี = 0.164 + 0.0257 NOPA และโมเดลการทำนายร่องรอยที่ไม่ดีของแต่ละมาตรวัดแสดงในตารางที่ 5.3

ตารางที่ 5.1 ผลลัพธ์แสดงความสัมพันธ์ของมาตรวัดต่อร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	มาตรวัด	R	R Square	Adjust R Square	Std. Error of the Estimate
Data Class	NOPA	0.291	0.085	0.084	0.3739
	NOAM	0.238	0.056	0.055	0.3797
	NSM	0.082	0.007	0.007	0.0330
	NSOC	0.695	0.484	0.484	0.0238
	NGM	0.002	0.000	0.000	0.0331
	NGOC	0.004	0.000	0.000	0.0331
	NSM+ NSOC	0.221	0.049	0.049	0.0323
	NCA	0.562	0.316	0.316	0.3157
	NAOC	0.136	0.018	0.018	0.3782
Feature Envy	NCM *	-	-	-	-
	NCDA*	-	-	-	-
	Total*	-	-	-	-
	NCRA*	-	-	-	-
Large Class	NIM	0.304	0.92	0.910	17.6387
	NIV	0.608	0.370	0.369	0.2775
	TCC	0.224	0.500	0.490	0.3408
	NLOC	0.106	0.110	0.100	0.3473
	AMC	0.141	0.020	0.019	0.3469
Lazy Class	NIM	0.407	0.165	0.164	0.2691
	NIV	0.539	0.290	0.289	0.2481
	DIT	0.019	0.000	-0.001	0.2947
	AMC	0.044	0.002	0.001	0.2944
Long Method	NOS	0.576	0.332	0.332	0.2180
	NOP	0.502	0.252	0.252	0.2307
	NOT	0.553	0.306	0.305	0.2222
	MCX	0.486	0.236	0.236	0.2263
Long Parameter List	NOP	0.674	0.455	0.455	0.1112
Refused Bequest	NOCM *	-	-	-	-
	NSMP *	-	-	-	-
	NOCA *	-	-	-	-
	NSAP *	-	-	-	-
Switch statement	NOSS	0.978	0.957	0.957	0.01841

ตารางที่ 5.2 แสดงค่าสัมประสิทธิ์ของมาตรวัดแต่ละตัว

ร่องรอยที่ไม่ดี	มาตรวัด		Unstandardized Coefficients	Std. Error	Standardized Coefficients	t	Sig.
			B		Beta		
Data Class	NOPA	β_0	0.1640	0.014		11.784	0.000
		β_1	0.0257	0.003	0.291	8.367	0.000
	NOAM	β_0	0.1730	0.014		12.405	0.000
		β_1	0.0032	0.000	0.238	6.721	0.000
	NSOC	β_0	1.0030	0.000		3325.124	0.000
		β_1	- 0.0793	0.001	- 0.695	- 77.387	0.000
	NSM+NSOC	β_0	1.0000	0.000		2458.339	0.000
		β_1	- 0.0051	0.000	- 0.221	- 18.132	0.000
	NCA	β_0	0.1160	0.007		17.114	0.000
		β_1	0.0117	0.000	0.562	32.798	0.000
	NAOC	β_0	0.1720	0.008		21.855	0.000
		β_1	0.0380	0.006	0.136	6.603	0.000
Large Class	NIM	β_0	0.0872	0.014		6.396	0.000
		β_1	0.0057	0.001	0.302	8.728	0.000
	NIV	β_0	0.0258	0.011		2.244	0.025
		β_1	0.0358	0.002	0.608	21.099	0.000
	TCC	β_0	0.1910	0.015		13.100	0.000
		β_1	- 0.0999	0.016	- 0.224	- 6.319	0.000
	NLOC	β_0	0.1340	0.013		10.395	0.000
		β_1	0.0021	0.000	0.106	2.942	0.003
	AMC	β_0	0.1100	0.015		7.350	0.000
		β_1	0.0050	0.001	0.141	3.937	0.000
Lazy Class	NIM	β_0	0.9670	0.011		87.947	0.000
		β_1	- 0.0065	0.001	- 0.407	-12.280	0.000
	NIV	β_0	0.9910	0.010		96.737	0.000
		β_1	- 0.0266	0.002	- 0.539	-17.640	0.000
Long Method	NOS	β_0	0.0179	0.005		3.758	0.000
		β_1	0.0177	0.001	0.576	34.623	0.000
	NOP	β_0	- 0.0187	0.006		- 3.242	0.001
		β_1	0.0920	0.003	0.502	28.509	0.000
	NOT	β_0	0.0249	0.005		5.193	0.000
		β_1	0.0588	0.002	0.553	32.602	0.000
	MCX	β_0	0.0350	0.005		7.165	0.000
		β_1	0.0059	0.000	0.486	26.820	0.000
Long Parameter List	NOP	β_0	- 0.0494	0.003		-17.763	0.000
		β_1	0.0697	0.002	0.674	44.861	0.000

ตารางที่ 5.2 แสดงค่าสัมประสิทธิ์ของมาตรวัดแต่ละตัว (ต่อ)

ร่องรอยที่ไม่ดี	มาตรวัด		Unstandardized Coefficients	Std. Error	Standardized Coefficients	t	Sig.
			B		Beta		
Switch Statement	NOSS	β_0	0.0034	0.000		0.908	0.364
		β_1	0.909	0.004	0.978	230.652	0.000

เนื่องจากการวิเคราะห์ความถดถอยเชิงซ้อนเป็นการหาความสัมพันธ์ของตัวแปรตั้งแต่ 2 ตัวขึ้นไปที่มีผลต่อตัวแปรตาม 1 ตัว ดังนั้นการหาช่วงของแต่ละตัวแปรจะมีความยากและซับซ้อนกว่าการวิเคราะห์ความถดถอยแบบธรรมดาซึ่งเป็นการหาความสัมพันธ์ของตัวแปร 1 ตัวและตัวแปรตาม 1 ตัว ดังนั้นงานวิจัยนี้จึงใช้การวิเคราะห์ความถดถอยแบบธรรมดา

ตารางที่ 5.3 โมเดลทำนายร่องรอยที่ไม่ดีของมาตรวัดในร่องรอยที่ไม่ดีทั้ง 8 แบบ

ร่องรอยที่ไม่ดี	มาตรวัด	โมเดลการทำนายร่องรอยที่ไม่ดี
Data Class	NOPA	ค่าร่องรอยที่ไม่ดี = 0.164 + 0.0257 NOPA
	NOAM	ค่าร่องรอยที่ไม่ดี = 0.173 + 0.0032 NOAM
	NSOC	ค่าร่องรอยที่ไม่ดี = 1.003 – 0.0793 NSOC
	NSM+NSOC	ค่าร่องรอยที่ไม่ดี = 1 – 0.0051 (NSM+NSOC)
	NCA	ค่าร่องรอยที่ไม่ดี = 0.116 + 0.0117 NCA
	NAOC	ค่าร่องรอยที่ไม่ดี = 0.172 + 0.038 NAOC
Large Class	NIM	ค่าร่องรอยที่ไม่ดี = 0.0872 + 0.0057 NIM
	NIV	ค่าร่องรอยที่ไม่ดี = 0.0258 + 0.0358 NIV
	TCC	ค่าร่องรอยที่ไม่ดี = 0.191 – 0.0999 TCC
	NLOC	ค่าร่องรอยที่ไม่ดี = 0.134 + 0.0021 NLOC
	AMC	ค่าร่องรอยที่ไม่ดี = 0.11 + 0.005 AMC
Lazy Class	NIM	ค่าร่องรอยที่ไม่ดี = 0.967 – 0.0065 NIM
	NIV	ค่าร่องรอยที่ไม่ดี = 0.991 – 0.0266 NIV
Long Method	NOS	ค่าร่องรอยที่ไม่ดี = 0.0179 + 0.0177 NOS
	NOP	ค่าร่องรอยที่ไม่ดี = 0.092 NOP – 0.0187
	NOT	ค่าร่องรอยที่ไม่ดี = 0.0249 + 0.0588 NOT
	MCX	ค่าร่องรอยที่ไม่ดี = 0.0350 + 0.0059 MCX
Long Parameter List	NOP	ค่าร่องรอยที่ไม่ดี = 0.0697 NOP – 0.494
Switch Statement	NOSS	ค่าร่องรอยที่ไม่ดี = 0.909 NOSS + 0.0034

5.3 หาช่วงค่าของมาตรวัดต่าง ๆ

สร้างตารางแจกแจงความถี่ของค่ามาตรวัดที่เกิดและไม่เกิดร่องรอยที่ไม่ดี ได้โดยใช้โปรแกรม SPSS เลือก Analyze เลือก Descriptive Statistics เลือก Crosstabs เลือก Columns เป็น ค่าเกิดร่องรอยที่ไม่ดี และ rows คือ ค่าของมาตรวัดที่ให้แสดง (รายละเอียดตารางแจกแจงความถี่ของมาตรวัดต่างๆอยู่ในภาคผนวก ข) หลังจากนั้นนำข้อมูลจากตารางแจกแจงความถี่มาคำนวณเพื่อให้ได้ค่าทางสถิติดังตารางที่ 5.4 ซึ่งมีรายละเอียดดังนี้

n คือ จำนวนคลาส min คือ ค่ามาตรวัดต่ำสุด

max คือ ค่ามาตรวัดสูงสุด $\bar{x}$ คือ ค่าเฉลี่ยของค่ามาตรวัด NIM ต่อหนึ่งคลาส

σ คือ ค่าส่วนเบี่ยงเบนมาตรฐาน ระดับความเชื่อมั่น 95% หรือ $\alpha = 0.05$

L คือ ค่าต่ำสุด U คือ ค่าสูงสุด

โดยที่ L และ U หาได้จากสูตร $\bar{x} - Z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}$ และ $\bar{x} + Z_{1-\alpha/2}\frac{\sigma}{\sqrt{n}}$ ตามลำดับ

ตารางที่ 5.4 ค่าทางสถิติต่างๆ

ร่องรอยที่ไม่ดี	มาตรวัด	เกิดร่องรอยที่ไม่ดี	n	min	max	$\bar{x}$	σ	$\alpha = 0.05$	
								L	U
Data Class	NOPA	เกิด	142	0	67	3.6056	9.61748	2.0101~2	5.2012~5
		ไม่เกิด	615	0	5	0.3106	0.87088	0.2416~0	0.3795~0
	NOAM	เกิด	142	0	771	18.8732	65.36215	8.0296~8	29.7168~30
		ไม่เกิด	615	0	5	1.1789	1.43271	1.0654~1	1.2923~1
	NSOC	เกิด	6388	0	2	0.0437	0.20667	0.0386~0	0.0487~0
		ไม่เกิด	7	0	8	6.1429	0.89974	5.3107~5	6.957~7
	NSM + NSOC	เกิด	6388	0	85	0.1752	1.38936	0.1411~0	0.2092~0
		ไม่เกิด	7	6	14	9.7143	2.69037	7.2261~7	12.2025~12
	NCA	เกิด	412	0	310	27.4417	36.06096	23.94~24	30.9341~31
		ไม่เกิด	1918	0	5	0.3978	1.10282	0.3484~0	0.4472~0
	NAOC	เกิด	412	0	36	0.5267	3.12073	0.2245~0	0.8289~1
		ไม่เกิด	1918	0	5	0.0422	0.35937	0.0261~0	0.0538~0
Large Class	NIM	เกิด	108	6	107	23.3796	18.42576	19.9 ~ 20	26.85 ~ 27
		ไม่เกิด	654	0	4056	7.3823	17.53944	6.014 ~ 6	8.7067 ~ 9
	NIV	เกิด	108	5	64	12.1111	8.76520	10.43~10	13.7831~14
		ไม่เกิด	652	0	75	1.7807	3.63792	1.5 ~ 2	2.06 ~ 2
	TCC	เกิด	108	0	0.67	0.0556	0.09253	0.038	0.0733
		ไม่เกิด	652	0	2	0.5564	0.82226	0.4931	0.6196
	NLOC	เกิด	108	0	1155	85.8687	155.4907	56	116
		ไม่เกิด	654	0	4056	31.0321	183.1067	17	45
	AMC	เกิด	109	0	40.36	9.9406	9.66615	8	11
		ไม่เกิด	654	0	128	5.9370	9.85563	5	7

ตารางที่ 5.4 ค่าทางสถิติต่างๆ (ต่อ)

ร่องรอยที่ไม่ดี	มาตรวัด	เกิดร่องรอยที่ไม่ดี	n	min	max	$\bar{x}$	σ	$\alpha = 0.05$	
								L	U
Lazy Class	NIM	เกิด	690	0	350	7.2304	1536651	6.05~ 6	8.4013~ 8
		ไม่เกิด	73	3	132	32.7945	25.99036	26.73~27	38.86~ 39
	NIV	เกิด	690	0	75	2.2261	3.92283	1.9329~2	2.5193~3
		ไม่เกิด	73	0	64	13.1370	10.93963	10.58~11	15.6894~16
Long Method	NOS	เกิด	186	1	245	20.6344	24.80989	17.05~17	24.2234~24
		ไม่เกิด	2229	0	10	1.8964	1.83021	1.8203~2	1.9724~2
	NOP	เกิด	186	0	11	3.5699	2.84709	3.158~3	3.9817~4
		ไม่เกิด	2229	0	5	0.83	1.02188	0.788~1	0.8724~1
	NOT	เกิด	186	0	52	5.6989	6.24679	4.795~5	6.6026~7
		ไม่เกิด	2229	0	10	0.4868	1.22966	0.436~0	0.5378~1
	MCX	เกิด	2159	1	555.9	43.4268	66.023	33.370	53.4833
		ไม่เกิด	168	0	73.5	3.4034	6.05350	3.1479	3.6589
Long Parameter List	NOP	เกิด	56	6	11	7.4107	1.1406	7.105~7	7.7162~8
		ไม่เกิด	2359	0	5	0.8898	1.0735	0.846~1	0.9331~1
Switch Statement	NOSS	เกิด	19	1	2	1.0526	0.22942	0.9421~1	1.1632~1
		ไม่เกิด	2415	0	0	0.0083	0.09510	0.0045~0	0.0121~0

~ คือการปัดเศษเนื่องจากมาตรวัดตัวนั้นเป็นเลขจำนวนเต็ม

หลังจากได้ค่าทางสถิติต่างๆ แล้ว คำนวณหาค่า L และ U เพื่อหาช่วงค่าซึ่งคือ (L, U) ดังรายละเอียดที่แสดงการหาช่วงค่าของมาตรวัดต่างๆ ดังนี้

5.3.1 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Data Class

5.3.1.1 มาตรวัด NOPA

ช่วงค่าของมาตรวัด NOPA ที่เกิดร่องรอยที่ไม่ดีคือ (2.0101, 5.2012) เนื่องจากค่ามาตรวัด NOPA เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (2, 5) แต่ข้อกำหนดของค่าที่พิจารณามาตรวัดนี้ต้องมากกว่า 5 ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (5, 5) และช่วงค่าของมาตรวัด NOPA ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.2416, 0.3795) เนื่องจากค่ามาตรวัด NOPA เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0)

5.3.1.2 มาตรวัด NOAM

ช่วงค่าของมาตรวัด NOAM ที่เกิดร่องรอยที่ไม่ดีคือ (8.0296, 29.7168) เนื่องจากค่ามาตรวัด NOAM เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัด

จึงเป็น (8, 30) และช่วงค่าของมาตรวัด NOAM ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (1.0654, 1.2923) เนื่องจากค่ามาตรวัด NOAM เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (1, 1)

5.3.1.3 มาตรวัด NSOC

ช่วงค่าของมาตรวัด NSOC ที่เกิดร่องรอยที่ไม่ดีคือ (0.0386, 0.0487) เนื่องจากค่ามาตรวัด NSOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0) และช่วงค่าของมาตรวัด NSOC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (5.3107, 6.957) เนื่องจากค่ามาตรวัด NSOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (5, 7)

5.3.1.4 มาตรวัด NSM+NSOC

ช่วงค่าของมาตรวัดที่เป็นผลรวมของ NSM และ NSOC ที่เกิดร่องรอยที่ไม่ดีคือ (0.1411, 0.2092) เนื่องจากค่ามาตรวัดที่เป็นผลรวมของ NSM และ NSOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0) และช่วงค่าของมาตรวัดที่เป็นผลรวมของ NSM และ NSOC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (7.2261, 12.2025) เนื่องจากค่ามาตรวัดที่เป็นผลรวมของ NSM และ NSOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (7, 12)

5.3.1.5 มาตรวัด NCA

ช่วงค่าของมาตรวัด NCA ที่เกิดร่องรอยที่ไม่ดีคือ (23.9494, 30.9341) เนื่องจากค่ามาตรวัด NCA เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (24, 31) และช่วงค่าของมาตรวัด NCA ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.3484, 0.4472) เนื่องจากค่ามาตรวัด NCA เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0)

5.3.1.6 มาตรวัด NAOC

ช่วงค่าของมาตรวัด NAOC ที่เกิดร่องรอยที่ไม่ดีคือ (0.2245, 0.8289) เนื่องจากค่ามาตรวัด NAOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 1) และช่วงค่าของมาตรวัด NAOC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.0261, 0.0538) เนื่องจากค่ามาตรวัด NAOC เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0) จากช่วงเกิดและไม่เกิดร่องรอยที่ไม่ดีมีส่วนที่ซ้อนทับกันอยู่ที่ค่ามาตรวัด NAOC = 0 ที่เดียว ดังนั้นจึงดูตารางแจกแจงความถี่ของมาตรวัด NAOC ดังตารางที่ 5.5 ช่วยในการพิจารณา

ตารางที่ 5.5 ตารางแจกแจงความถี่ของมาตรวัด NAOC

NAOC	Bad Smell		Total
	Bad	Not Bad	
0	395	1891	2286
1	0	1	1
3	0	25	25
5	0	1	1
6	8	0	8
7	1	0	1
12	2	0	2
18	4	0	4
30	1	0	1
36	1	0	1
Total	412	1918	2330

จากตารางที่ 5.5 คอลัมน์ NAOC คือ ค่าของมาตรวัด NAOC ส่วนคอลัมน์ Bad คือ จำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่เกิดร่องรอยที่ไม่ดี และคอลัมน์ Not Bad คือ จำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งผลรวมของจำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่เกิดร่องรอยที่ไม่ดี และจำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่ไม่เกิดร่องรอยที่ไม่ดีอยู่ในคอลัมน์ Total ซึ่งจะพิจารณาได้ว่า ช่วงค่าของมาตรวัด NAOC ที่เกิดร่องรอยที่ไม่ดีคือ (0, 1) ไม่สามารถใช้ได้พราะ 0 อยู่ในช่วงค่าของมาตรวัด NAOC ที่ไม่เกิดร่องรอยที่ไม่ดีแล้วและจากตารางที่ 5.5 ค่าของมาตรวัด NAOC = 1 ไม่มีจำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่เกิดร่องรอยที่ไม่ดี ดังนั้นช่วงค่าของมาตรวัด NAOC ที่เกิดร่องรอยที่ไม่ดีคือ (6, 36) ซึ่งมีจำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่เกิดร่องรอยที่ไม่ดีจริงๆ ส่วนช่วงค่าของมาตรวัด NAOC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 0) ใช้ได้พราะจำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่ไม่เกิดร่องรอยที่ไม่ดีมีค่า 1891 แต่จำนวนการถูกเรียกใช้คุณลักษณะจากคลาสอื่นที่เกิดร่องรอยที่ไม่ดีมีค่า 395 ซึ่งมีจำนวนมากกว่าถึง 4.8 เท่า

5.3.2 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Large Class

5.3.2.1 มาตรวัด NIM

ช่วงค่าของมาตรวัด NIM ที่เกิดร่องรอยที่ไม่ดีคือ (19.9, 26.85) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (20, 27) และช่วงค่าของ

มาตรวัด NIM ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (6.014, 8.7067) เนื่องจากค่ามาตรวัด NIM เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (6, 9)

5.3.2.2 มาตรวัด NIV

ช่วงค่าของมาตรวัด NIV ที่เกิดร่องรอยที่ไม่ดีคือ (10.4391, 13.7831) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (10, 14) และช่วงค่าของมาตรวัด NIV ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (1.5, 2.06) เนื่องจากค่ามาตรวัด NIV เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (2, 2)

5.3.2.3 มาตรวัด TCC

ช่วงค่าของมาตรวัด TCC ที่เกิดร่องรอยที่ไม่ดีคือ (0.038, 0.0733) และช่วงค่าของมาตรวัด TCC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.4931, 0.6196)

5.3.2.4 มาตรวัด NLOC

ช่วงค่าของมาตรวัด NLOC ที่เกิดร่องรอยที่ไม่ดีคือ (56, 116) และช่วงค่าของมาตรวัด NLOC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (17, 45)

5.3.2.5 มาตรวัด AMC

ช่วงค่าของมาตรวัด AMC ที่เกิดร่องรอยที่ไม่ดีคือ (8, 11) และช่วงค่าของมาตรวัด AMC ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (5, 7)

5.3.3 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Lazy Class

5.3.3.1 มาตรวัด NIM

ช่วงค่าของมาตรวัด NIM ที่เกิดร่องรอยที่ไม่ดีคือ (6.0595, 8.4013) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (6, 8) และช่วงค่าของมาตรวัด NIM ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (26.7305, 38.86) เนื่องจากค่ามาตรวัด NIM เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (27, 39)

5.3.3.2 มาตรวัด NIV

ช่วงค่าของมาตรวัด NIV ที่เกิดร่องรอยที่ไม่ดีคือ (1.9329, 2.5193) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (2, 3) และช่วงค่าของมาตรวัด NIV ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (10.5846, 15.6894) เนื่องจากค่ามาตรวัด NIV เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (11, 16)

5.3.4 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Long Method

5.3.4.1 มาตรวัด NOS

ช่วงค่าของมาตรวัด NOS ที่เกิดร่องรอยที่ไม่ดีคือ (17.0455, 24.2234) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (17, 24) และช่วงค่าของมาตรวัด NOS ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (1.8203, 1.9724) เนื่องจากค่ามาตรวัด NOS เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (2, 2)

5.3.4.2 มาตรวัด NOP

ช่วงค่าของมาตรวัด NOP ที่เกิดร่องรอยที่ไม่ดีคือ (3.1580, 3.9817) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (3, 4) และช่วงค่าของมาตรวัด NOP ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.7875, 0.8724) เนื่องจากค่ามาตรวัด NOP เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (1, 1)

5.3.2.3 มาตรวัด NOT

ช่วงค่าของมาตรวัด NOT ที่เกิดร่องรอยที่ไม่ดีคือ (4.7953, 6.6026) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (5, 7) และช่วงค่าของมาตรวัด NOT ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.4357, 0.5378) เนื่องจากค่ามาตรวัด NOT เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 1)

5.3.2.4 มาตรวัด MCX

ช่วงค่าของมาตรวัด MCX ที่เกิดร่องรอยที่ไม่ดีคือ (33.3703, 53.4833) และช่วงค่าของมาตรวัด MCX ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (3.1479, 3.6589)

5.3.5 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Long Parameter List

5.3.5.1 มาตรวัด NOP

ช่วงค่าของมาตรวัด NOP ที่เกิดร่องรอยที่ไม่ดีคือ (7.1053, 7.7162) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (7, 8) และช่วงค่าของมาตรวัด NOP ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.8464, 0.9331) เนื่องจากค่ามาตรวัด NOP เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (1, 1)

5.3.6 หาช่วงค่ามาตรวัดสำหรับร่องรอยที่ไม่ดีแบบ Switch Statement

5.3.6.1 มาตรวัด NOSS

ช่วงค่าของมาตรวัด NOSS ที่เกิดร่องรอยที่ไม่ดีคือ (0.9421,1.1632) ซึ่งควรเป็นเลขจำนวนเต็มจึงทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (1, 1) และช่วงค่าของมาตรวัด NOSS ที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0.0045, 0.0121) เนื่องจากค่ามาตรวัด NOSS เป็นเลขจำนวนเต็มจึงควรทำการปัดเศษ ดังนั้นช่วงค่าของมาตรวัดจึงเป็น (0, 0)

ตารางที่ 5.6 สรุปช่วงค่าของแต่ละมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของแต่ละมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีของร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	มาตรวัด	ช่วงค่าของมาตรวัดที่	
		เกิดร่องรอยที่ไม่ดี	ไม่เกิดร่องรอยที่ไม่ดี
Data Class	NOPA	(5,5)	(0,0)
	NOAM	(8,30)	(1,1)
	NSOC	(0,0)	(5,7)
	NSM+NSOC	(0,0)	(7,12)
	NCA	(24,31)	(0,0)
	NAOC	(6,36)	(0,0)
Large Class	NIM	(20,27)	(6,9)
	NIV	(10,14)	(2,2)
	TCC	(0.038, 0.0733)	(0.4931, 0.6196)
	NLOC	(56,116)	(17,45)
	AMC	(8,11)	(5,7)
Lazy Class	NIM	(6,8)	(27,39)
	NIV	(2,3)	(11,16)
Long Method	NOS	(17,24)	(2,2)
	NOP	(3,4)	(1,1)
	NOT	(5,7)	(0,1)
	MCX	(33.3703,53.4833)	(3.1479,3.6589)
Long Parameter List	NOP	(7,8)	(1,1)
Switch Statement	NOSS	(1,1)	(0,0)

6.4 ประเมินค่าน่าเชื่อถือของช่วงค่าของมาตรวัดต่าง ๆ

นำค่ามาตรวัดต่างๆ ของชุดข้อมูลทดสอบจำนวน 2 โปรแกรม มาเปรียบเทียบค่าที่ได้จากช่วงค่าที่เกิดร่องรอยที่ไม่ดีในตารางที่ 5.6 ทีละมาตรวัด ถ้าอยู่ในช่วงเกิดร่องรอยที่ไม่ดีให้ค่าเป็น 1 ถ้าอยู่ในช่วงไม่เกิดร่องรอยที่ไม่ดีให้ค่าเป็น 0 ถ้าที่ไม่อยู่ทั้ง 2 ช่วงแสดงว่าไม่สามารถบอกได้ นำค่าที่ได้นี้มาเปรียบเทียบกับค่าที่ได้จากผู้เชียวชาญเป็นผู้กำหนด ถ้าค่าที่ได้และค่าที่ได้จากผู้เชียวชาญเป็นผู้กำหนดเป็นค่าเดียวกันแสดงว่าค่าถูกต้อง แต่ถ้าค่าที่ได้และค่าที่ได้จากผู้เชียวชาญเป็นผู้กำหนดไม่เป็นค่าเดียวกันแสดงว่าเกิดความผิดพลาด และถ้าค่าที่ได้นั้นไม่สามารถบอกได้แสดงว่าช่วงนั้นไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่ หลังจากนั้นนับจำนวนความถูกต้อง ความผิดพลาดและไม่สามารถบอกได้ เพื่อตรวจสอบความถูกต้อง ความผิดพลาดและไม่สามารถบอกได้ ซึ่งดังแสดงในตารางที่ 5.7

ตารางที่ 5.7 แสดงความถูกต้อง ผิดพลาดและไม่สามารถบอกได้ของช่วงค่ามาตรวัดต่างๆ สำหรับทำนายค่ารองรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	มาตรวัด	ค่าร่องรอยที่ไม่ดี	จำนวนข้อมูลทดสอบ			รวม
			ถูกต้อง	ผิดพลาด	ไม่สามารถบอกได้	
Data Class	NOPA*	จำนวน	160	40	0	200
		เปอร์เซ็นต์	81	19	0	100
	NOAM	จำนวน	51	149	0	200
		เปอร์เซ็นต์	25.5	74.5	0	100
	NSOC*	จำนวน	1362	16	0	1378
		เปอร์เซ็นต์	98.8	1.2	0	100
	NSM+ NSOC*	จำนวน	1232	146	0	1378
		เปอร์เซ็นต์	89.4	10.6	0	100
	NCA	จำนวน	274	292	0	566
		เปอร์เซ็นต์	48.4	51.6	0	100
	NAOC*	จำนวน	455	329	0	784
		เปอร์เซ็นต์	58	42	0	100
Large Class	NIM	จำนวน	8	27	165	200
		เปอร์เซ็นต์	4	13.5	82.5	100

ตารางที่ 5.7 แสดงความถูกต้อง ผิดพลาดและไม่สามารถบอกได้ของช่วงค่ามาตรวัดต่างๆ สำหรับทำนายค่ารองรอยที่ไม่ดี (ต่อ)

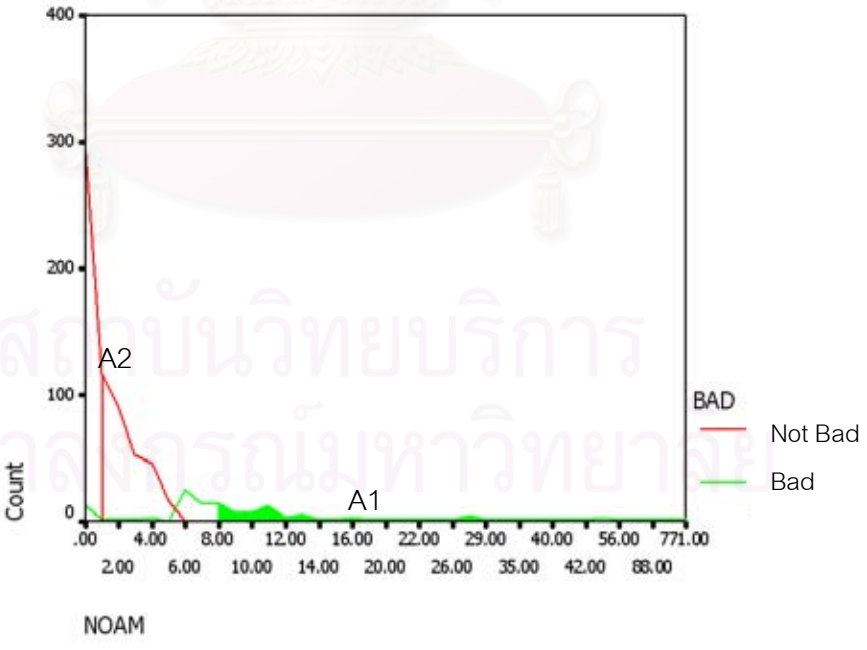
ร่องรอยที่ไม่ดี	มาตรวัด	ค่าร่องรอยที่ไม่ดี	จำนวนข้อมูลทดสอบ			รวม
			ถูกต้อง	ผิดพลาด	ไม่สามารถบอกได้	
Large Class	NIV	จำนวน	11	9	180	200
		เปอร์เซ็นต์	5.5	4.5	90	100
	TCC	จำนวน	6	0	194	200
		เปอร์เซ็นต์	3	0	97	100
	NLOC	จำนวน	25	18	157	200
		เปอร์เซ็นต์	12.5	9	78.5	100
	AMC	จำนวน	16	11	173	200
		เปอร์เซ็นต์	8	5.5	86.5	100
Lazy Class	NIM	จำนวน	16	184	0	200
		เปอร์เซ็นต์	8	92	0	100
	NIV	จำนวน	28	1	171	200
		เปอร์เซ็นต์	14	0.5	85.5	100
Long Method	NOS	จำนวน	33	167	0	200
		เปอร์เซ็นต์	16.5	83.5	0	100
	NOP	จำนวน	69	131	0	200
		เปอร์เซ็นต์	34.5	65.5	0	100
	NOT	จำนวน	144	56	0	200
		เปอร์เซ็นต์	72	28	0	100
	MCX	จำนวน	7	193	0	200
		เปอร์เซ็นต์	3.5	96.5	0	100
Long Parameter List	NOP	จำนวน	474	722	0	1196
		เปอร์เซ็นต์	39.63	60.37	0	100
Switch Statement	NOSS*	จำนวน	1405	5	0	1410
		เปอร์เซ็นต์	99.64	0.36	0	100

จากตารางที่ 5.7 มาตรวัดที่มี * กำกับมีเปอร์เซ็นต์ความถูกต้องมากกว่าผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ไม่สามารถบอกได้ดังนั้นจึงไม่ปรับช่วงค่ามาตรวัด แต่มาตรวัดอื่นที่ไม่มี * กำกับ มีเปอร์เซ็นต์ความถูกต้องน้อยกว่าผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ไม่สามารถบอกได้ดังนั้นจึงควรเปลี่ยนช่วงค่ามาตรวัดใหม่ดังนี้

6.4.1 ร่องรอยที่ไม่ดีแบบ Data Class

6.4.1.1 มาตรวัด NOAM

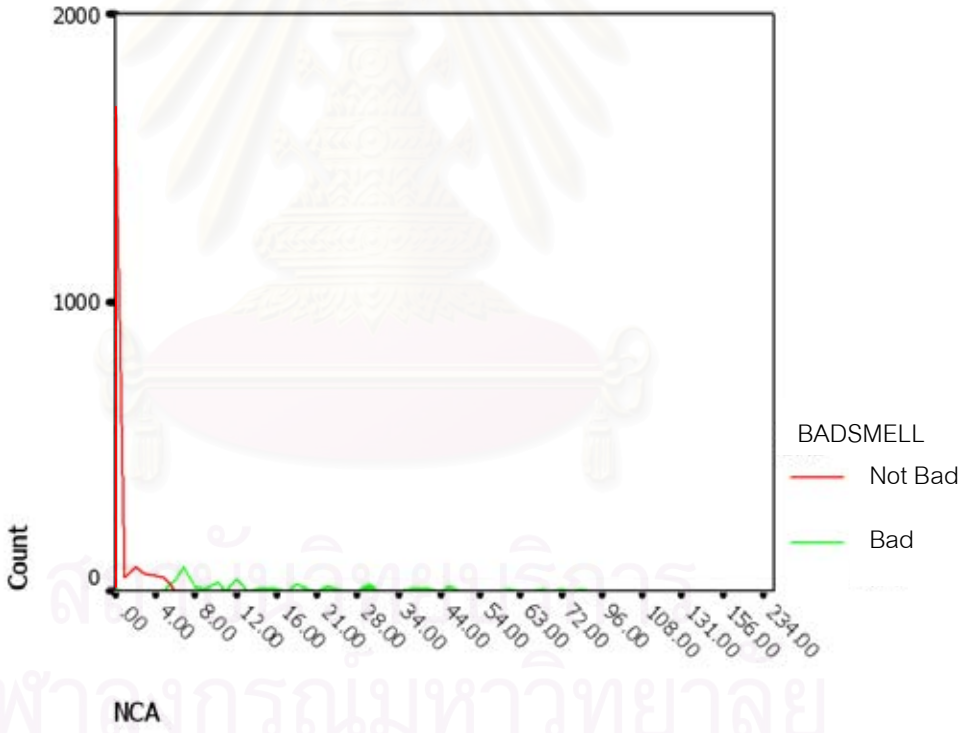
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (8,771) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0,1) จากรูปที่ 5.1 เป็นกราฟแสดงความถี่จำนวนเอคเซสเซอร์เมทธอดของแต่ละค่ามาตรวัด NOAM โดยแกนนอนคือ ค่ามาตรวัดของ NOAM และแกนตั้งคือ จำนวนเอคเซสเซอร์เมทธอด กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนจำนวนเอคเซสเซอร์เมทธอดของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนเอคเซสเซอร์เมทธอดของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด NOAM มากกว่า 8 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (8,∞)



รูปที่ 5.1 กราฟความถี่จำนวนเอคเซสเซอร์เมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.1.2 มาตรวัด NCA

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (24,310) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0,5) จากรูปที่ 5.2 เป็นกราฟแสดงความถี่จำนวนการเรียกใช้คุณลักษณะในคลาสที่คุณลักษณะอยู่ของแต่ละค่ามาตรวัด NCA โดยแกนนอนคือ ค่ามาตรวัดของ NCA และแกนตั้งคือ จำนวนการเรียกใช้คุณลักษณะในคลาสที่คุณลักษณะอยู่ กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนการเรียกใช้คุณลักษณะในคลาสที่คุณลักษณะอยู่ของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนการเรียกใช้คุณลักษณะในคลาสที่ คุณลักษณะอยู่ ของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด NCA มากกว่า 24 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (24,∞)

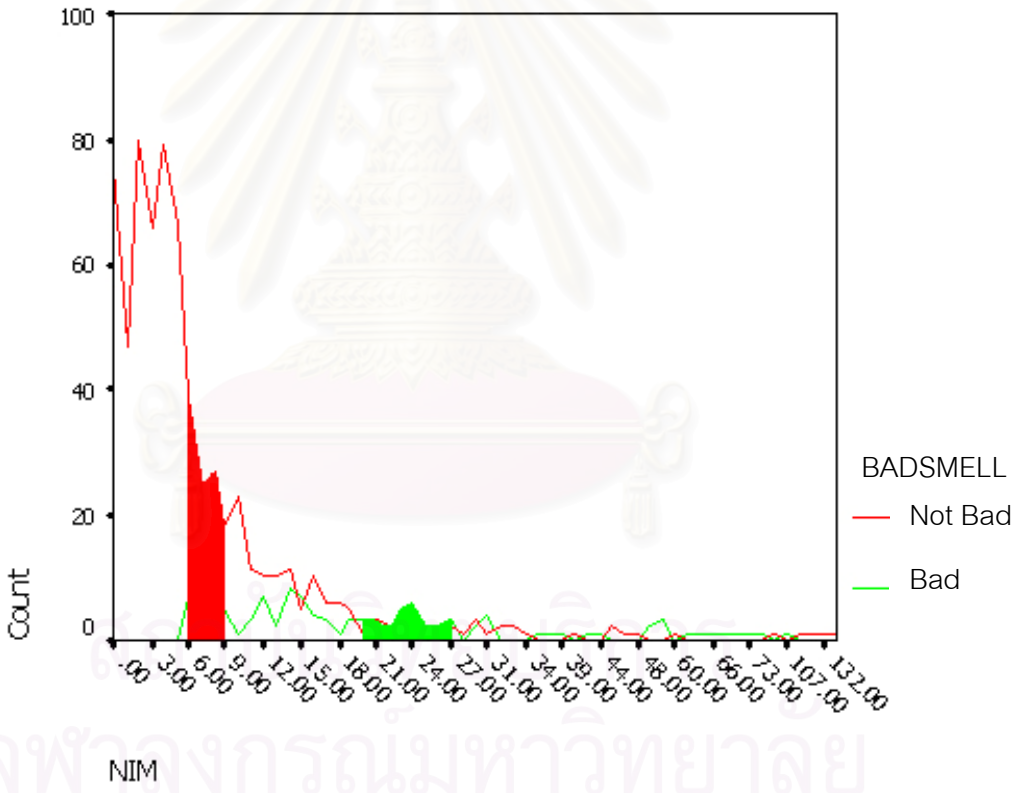


รูปที่ 5.2 กราฟความถี่จำนวนการเรียกใช้คุณลักษณะในคลาสที่คุณลักษณะอยู่ของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.2 ร่องรอยที่ไม่ดีแบบ Large Class

6.4.2.1 มาตรวัด NIM

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (20,350) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 9) จากรูปที่ 5.3 เป็นกราฟแสดงความถี่จำนวนคลาสของแต่ละค่ามาตรวัด NIM โดยแกนนอนคือ ค่ามาตรวัดของ NIM และแกนตั้งคือ จำนวนคลาส กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนคลาสของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนคลาสของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด NIM มากกว่า 20 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (20,∞)

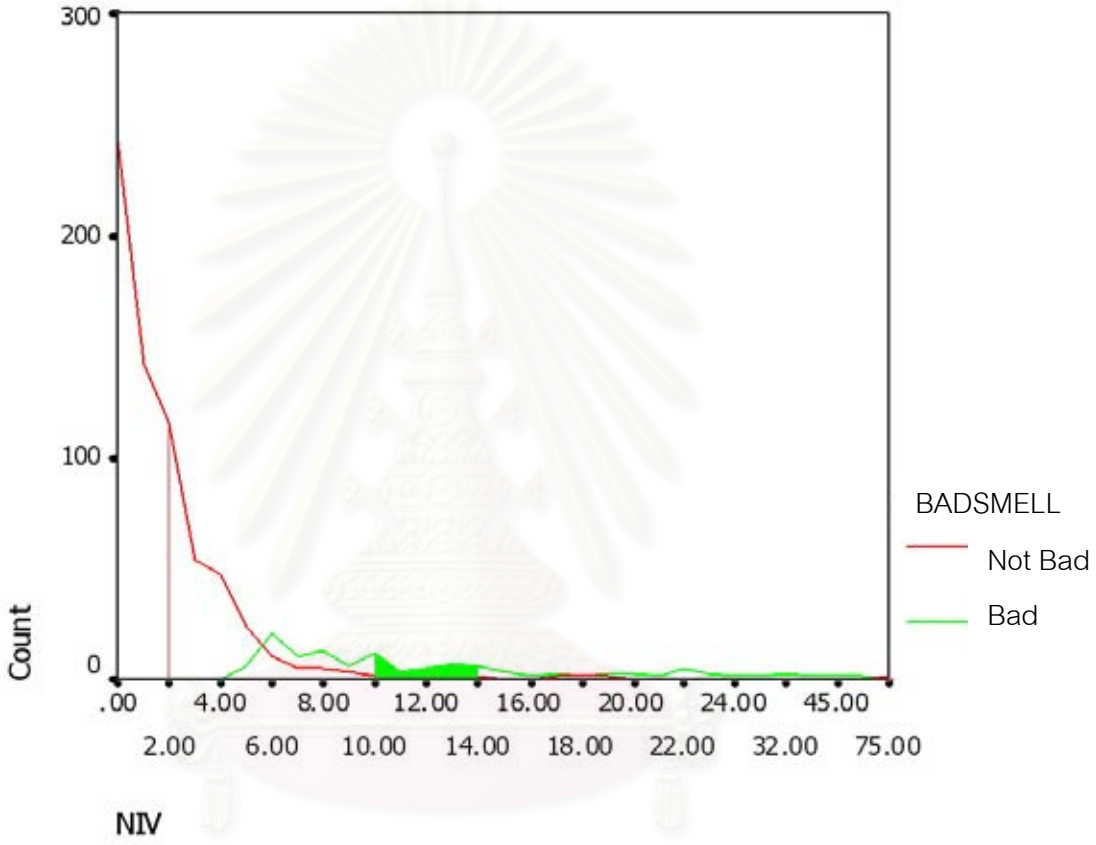


รูปที่ 5.3 กราฟความถี่จำนวนคลาสของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.2.2 มาตรวัด NIV

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (10, 64) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 2)

จากรูปที่ 5.4 เป็นกราฟแสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัด NIV โดยแกนนอนคือ ค่ามาตรวัดของ NIV และแกนตั้งคือ จำนวนคุณลักษณะ กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด NIV มากกว่า 10 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (10,∞)

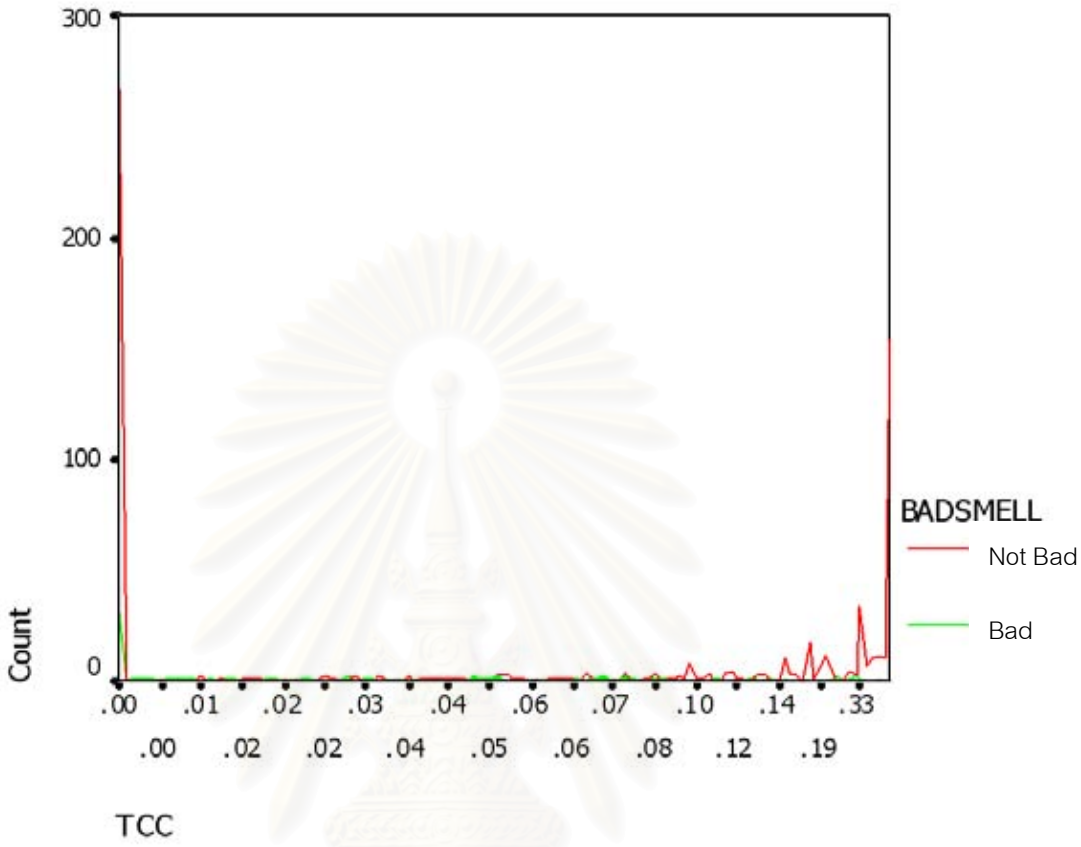


รูปที่ 5.4 กราฟความถี่จำนวนคุณลักษณะของค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.2.3 มาตรวัด TCC

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 2 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (0.038, 0.67) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 0.6196) จากรูปที่ 5.5 เป็นกราฟแสดงความถี่จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรงของแต่ละค่ามาตรวัด TCC โดยแกนนอนคือ ค่ามาตรวัดของ TCC และแกนตั้งคือ จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรง กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรงของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่

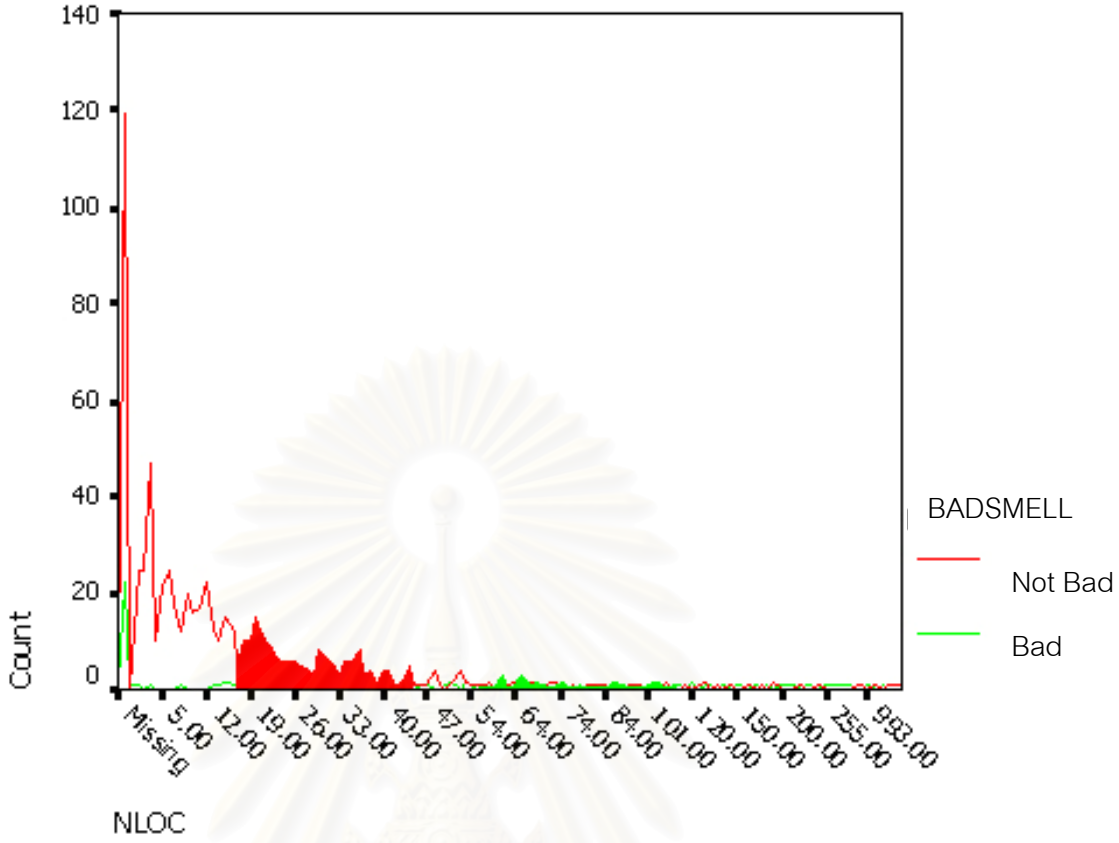
ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรงของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี



รูปที่ 5.5 กราฟความถี่จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรงของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

4.6.2.5 มาตรวัด NLOC

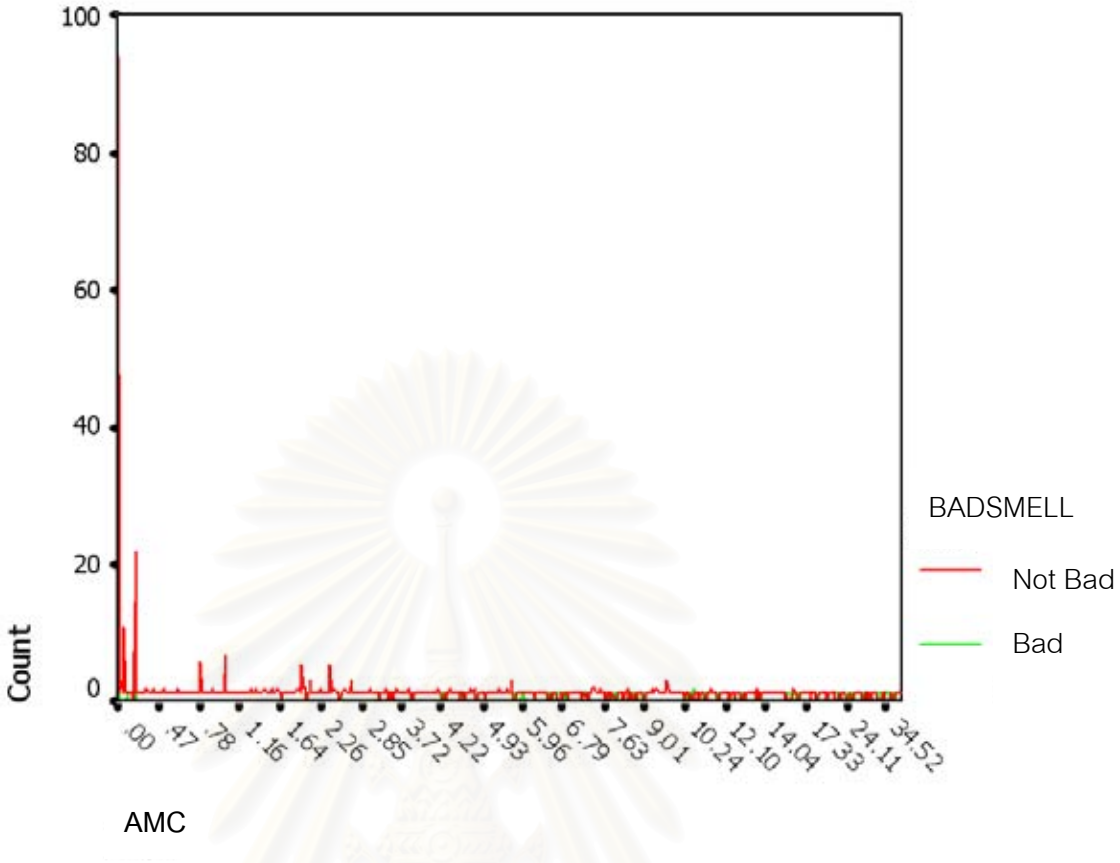
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (56, 1155) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 45) จากรูปที่ 5.6 เป็นกราฟแสดงความถี่จำนวนจำนวนสเตทเมนต์ในคลาสของแต่ละค่ามาตรวัด NLOC โดยแกนนอนคือ ค่ามาตรวัดของ NLOC และแกนตั้งคือ จำนวนสเตทเมนต์ในคลาส กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนสเตทเมนต์ในคลาสของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนสเตทเมนต์ในคลาสของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด NLOC มากกว่า 56 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (56,∞)



รูปที่ 5.6 กราฟความถี่จำนวนสเตทเมนต์ในคลาสของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

4.6.2.6 มาตรวัด AMC

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (8, 40.36) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 7) จากรูปที่ 5.7 เป็นกราฟแสดงความถี่จำนวนค่าเฉลี่ยความซับซ้อนต่อเมทธอดของแต่ละค่ามาตรวัด AMC โดยแกนนอนคือ ค่ามาตรวัดของ AMC และแกนตั้งคือ จำนวนค่าเฉลี่ยความซับซ้อนต่อเมทธอด กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนค่าเฉลี่ยความซับซ้อนต่อเมทธอด ของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนค่าเฉลี่ยความซับซ้อนต่อเมทธอดของ แต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือมาตรวัด AMC มากกว่า 8 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (8,∞)

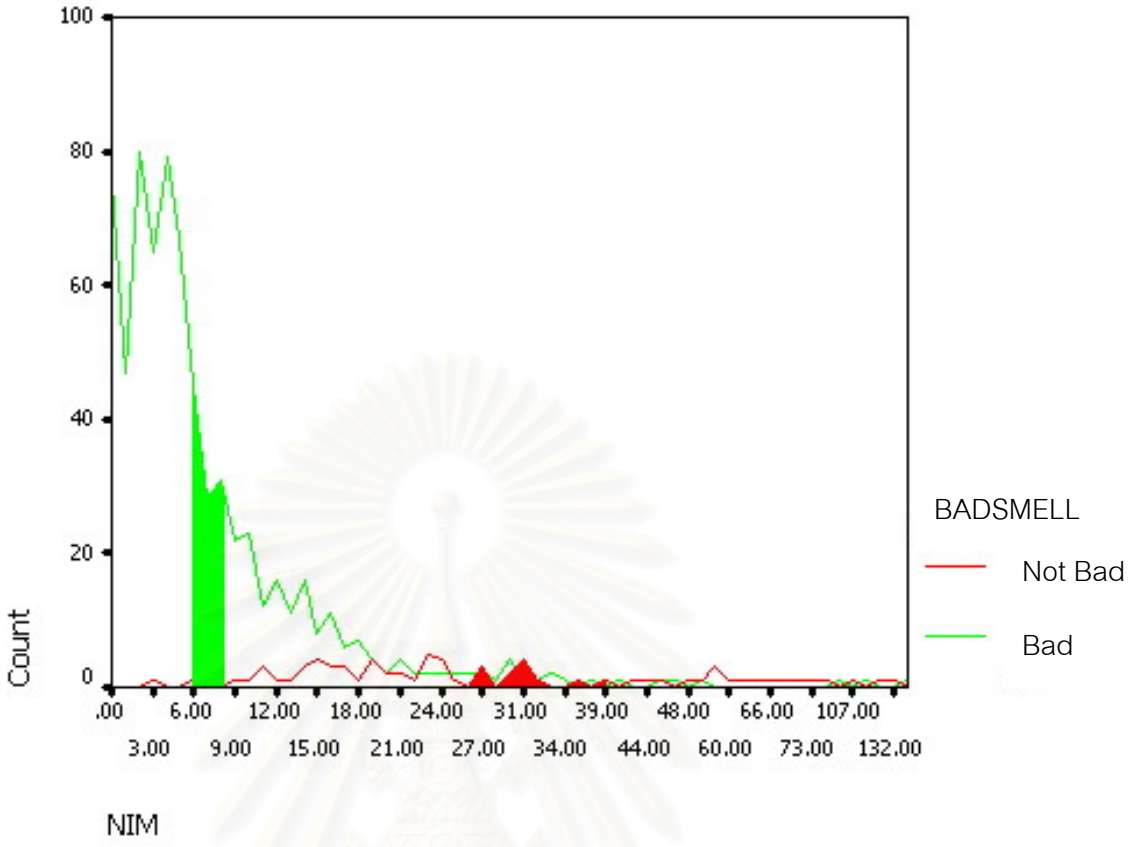


รูปที่ 5.7 กราฟความถี่จำนวนค่าเฉลี่ยความซับซ้อนต่อเมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.3 ร่องรอยที่ไม่ดีแบบ Lazy Class

6.4.3.1 มาตรวัด NIM

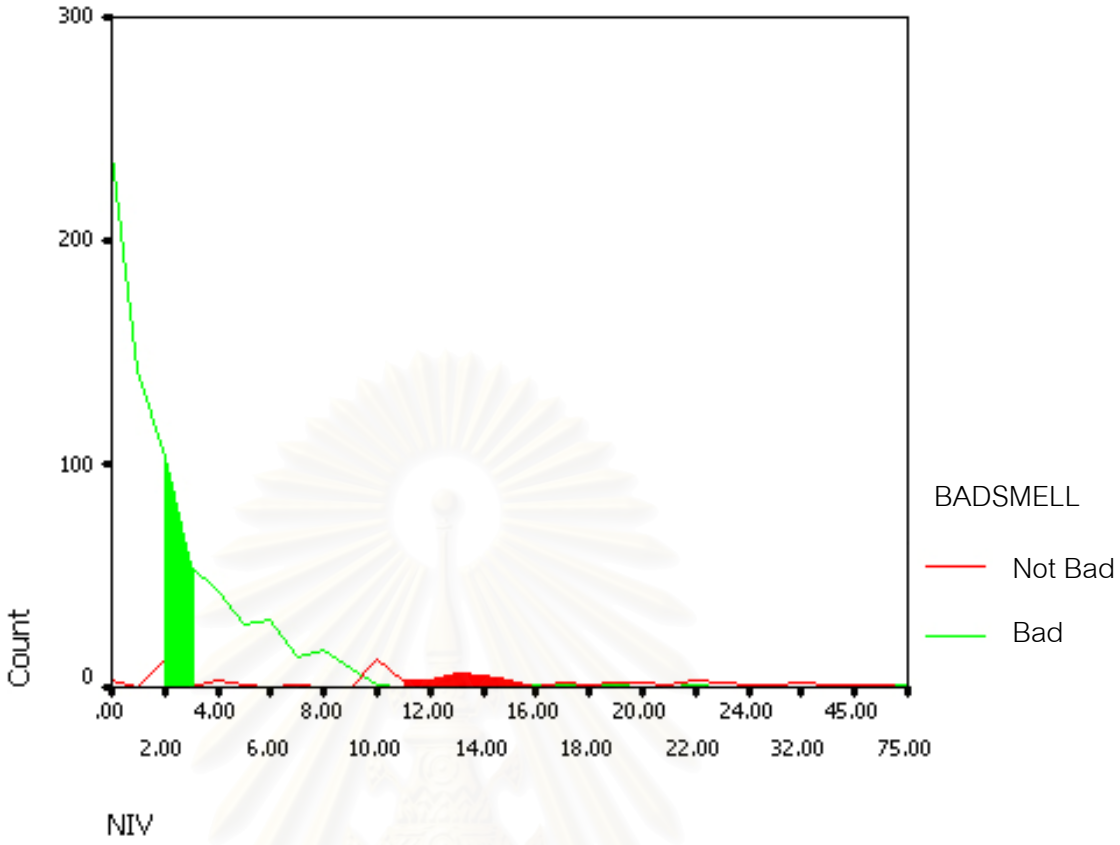
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 2 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (0, 8) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (27,132) จากรูปที่ 5.8 เป็นกราฟแสดงความถี่จำนวนเมทธอดในคลาสของแต่ละค่ามาตรวัด NIM โดยแกนนอนคือ ค่า มาตรวัดของ NIM และแกนตั้งคือ จำนวนเมทธอดในคลาส กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนเมทธอดในคลาสของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนเมทธอดในคลาสของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NIM มากกว่า 27 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (27,∞)



รูปที่ 5.8 กราฟความถี่จำนวนเมทธอดในคลาสของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.3.2 มาตรวัด NIV

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 2 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (0, 3) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (11, 64) จากรูปที่ 5.9 เป็นกราฟแสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัด NIV โดยแกนนอนคือ ค่ามาตรวัดของ NIV และแกนตั้งคือ จำนวนคุณลักษณะ กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนคุณลักษณะของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NIV มากกว่า 11 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (11,∞)

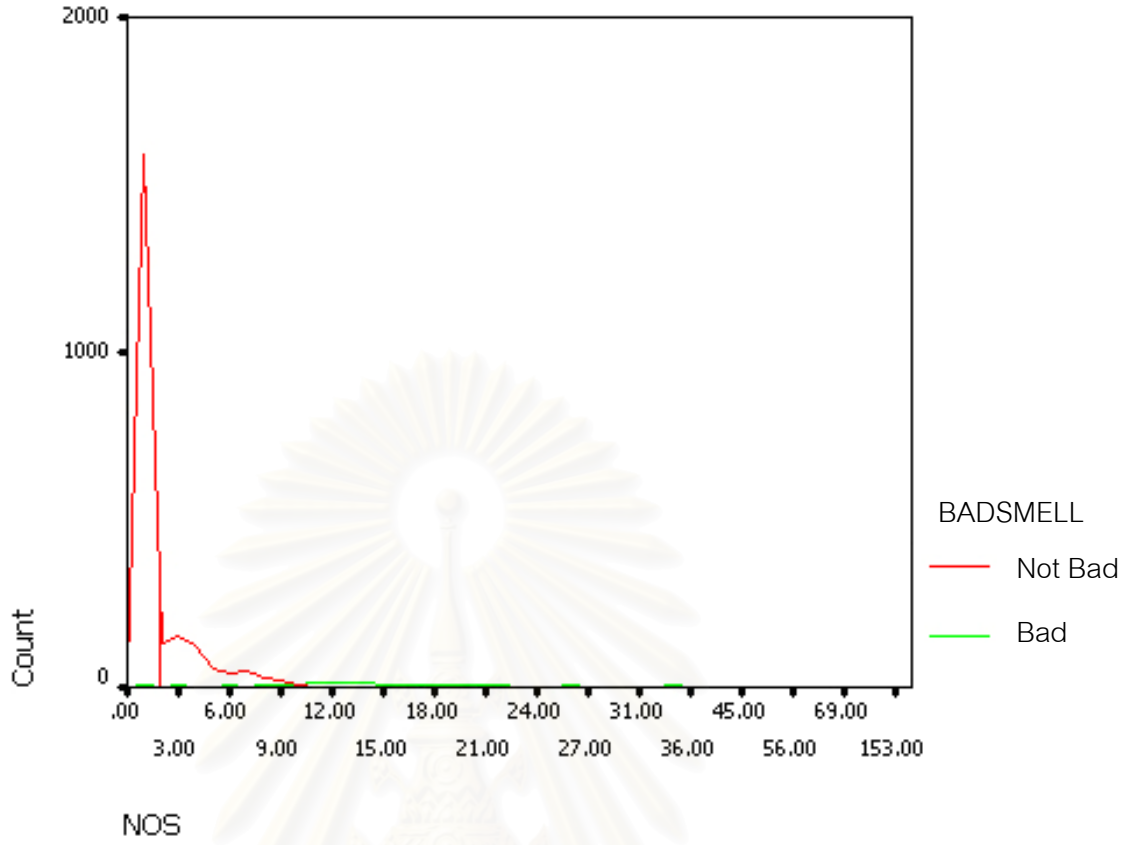


รูปที่ 5.9 กราฟความถี่จำนวนคุณลักษณะในคลาสของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.4 ร่องรอยที่ไม่ดีแบบ Long Method

6.4.4.1 มาตรวัด NOS

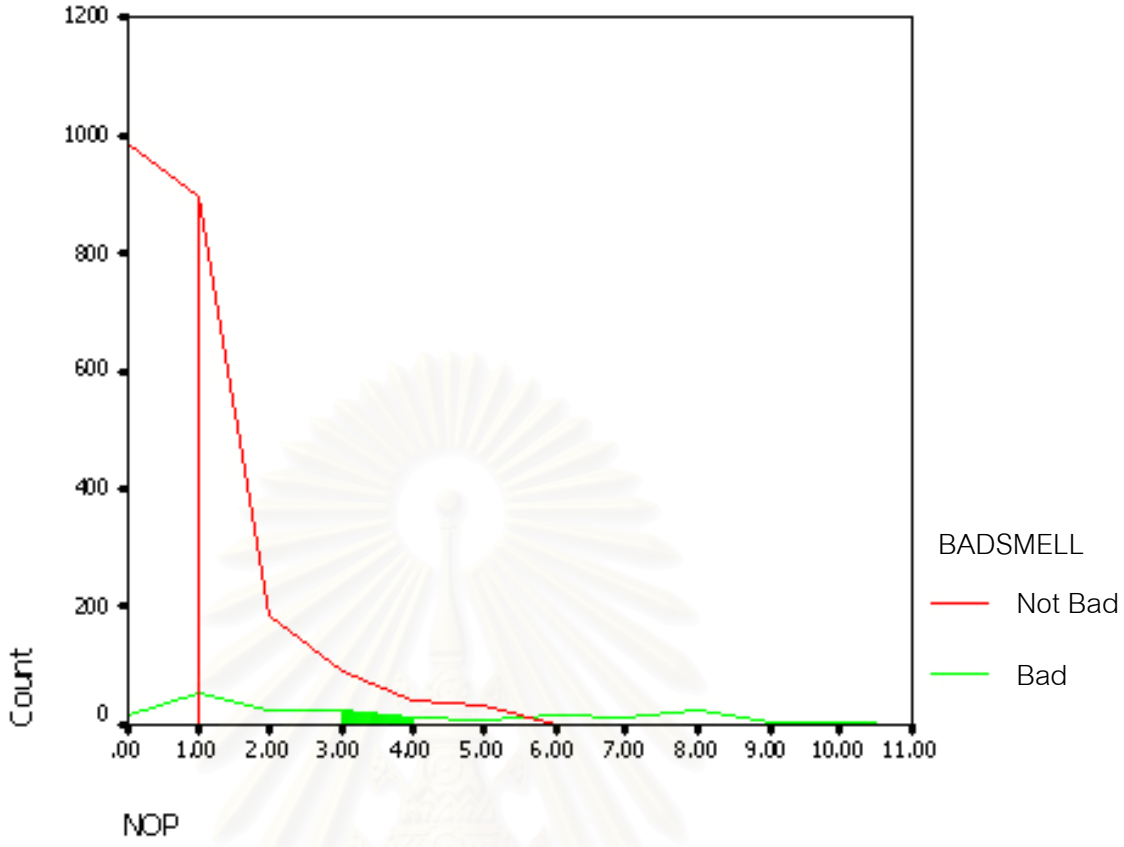
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (17,245) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 2) จากรูปที่ 5.10 เป็นกราฟแสดงความถี่จำนวนสเตทเมนต์ในเมทธอดของแต่ละค่ามาตรวัด NOS โดยแกนนอนคือ ค่ามาตรวัดของ NOS และแกนตั้งคือ จำนวนสเตทเมนต์ในเมทธอด กราฟที่มีค่า BADSMELL = Bad คือ กราฟที่แสดงความถี่จำนวนสเตทเมนต์ในเมทธอดของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนสเตทเมนต์ในเมทธอดของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NOS มากกว่า 17 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (17,∞)



รูปที่ 5.10 กราฟความถี่จำนวนสเตทเมนต์ในเมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.4.2 มาตรวัด NOP

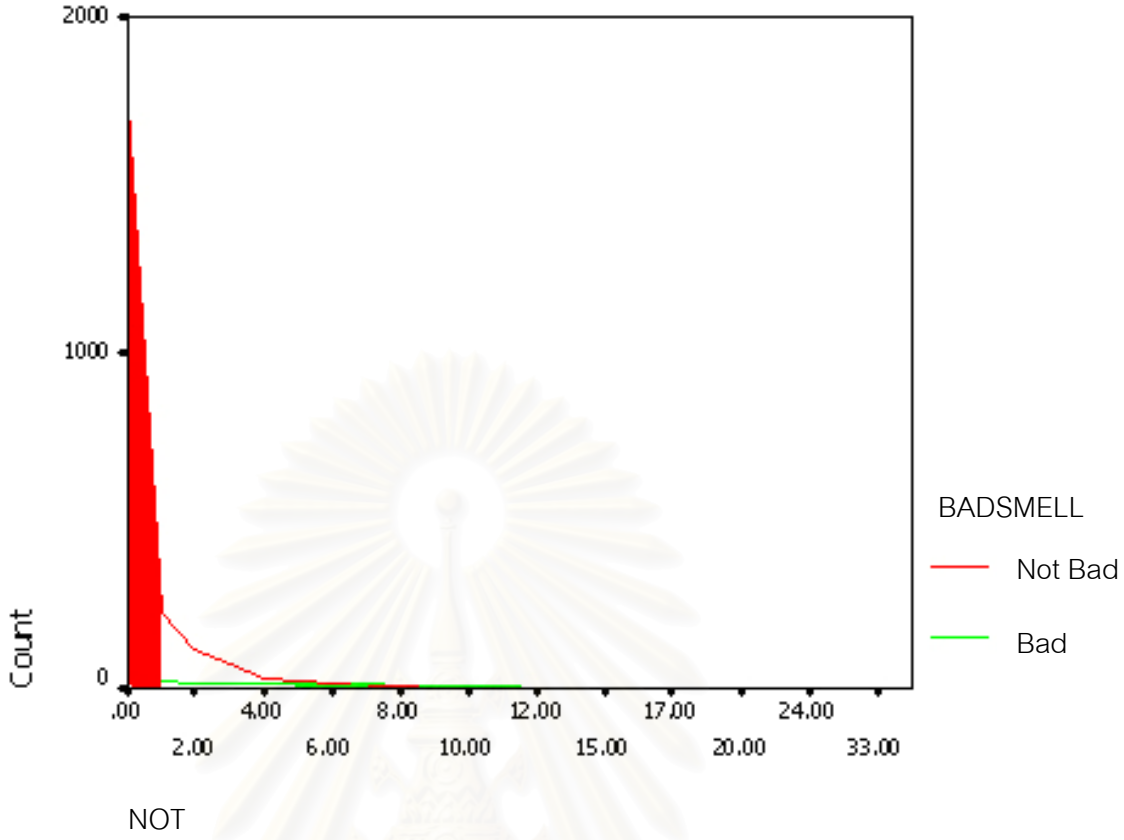
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (3, 11) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 1) จากรูปที่ 5.11 เป็นกราฟแสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัด NOP โดยแกนนอนคือ ค่า มาตรวัดของ NOP และแกนตั้งคือ จำนวนพารามิเตอร์ในเมทธอด กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NOP มากกว่า 11 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (11,∞)



รูปที่ 5.11 กราฟความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.4.3 มาตรวัด NOT

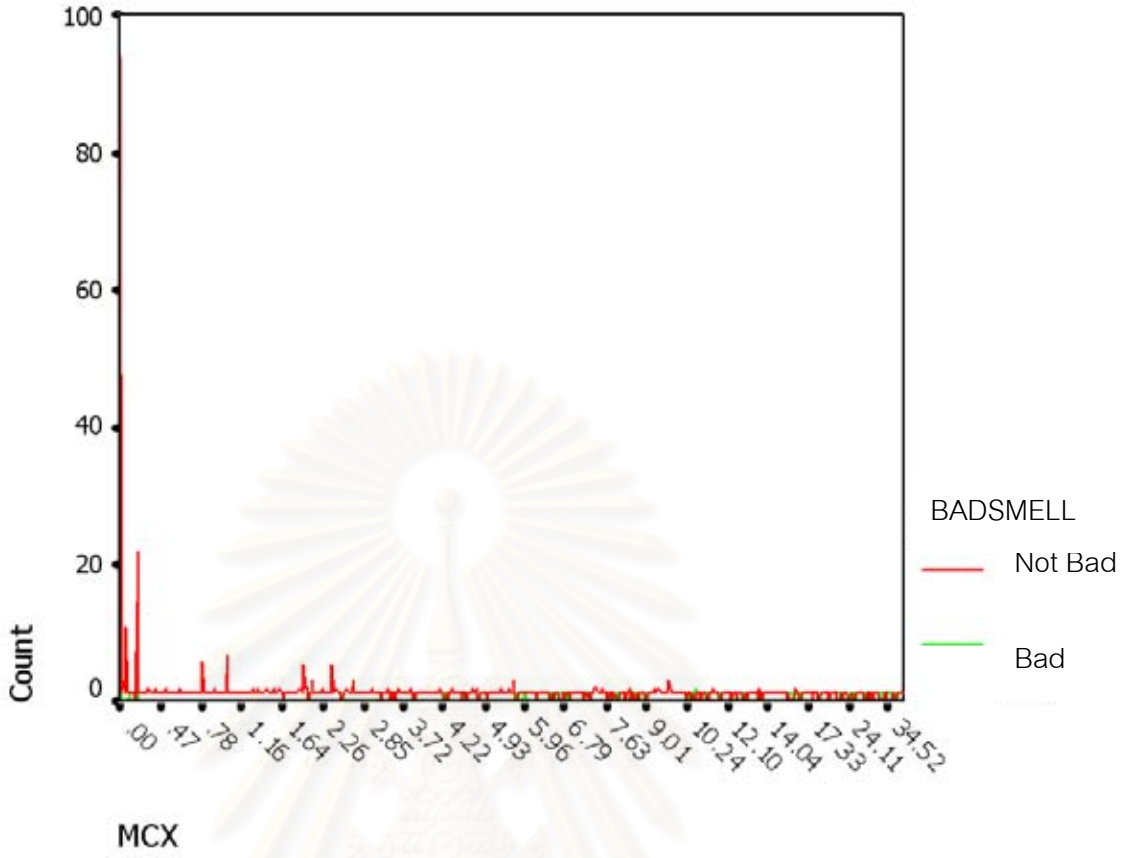
ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (5, 52) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 1) จากรูปที่ 5.12 เป็นกราฟแสดงความถี่จำนวนตัวแปรชั่วคราวของแต่ละค่ามาตรวัด NOT โดยแกนนอนคือ ค่ามาตรวัดของ NOT และแกนตั้งคือ จำนวนตัวแปรชั่วคราว กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนตัวแปรชั่วคราวของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนตัวแปรชั่วคราวของแต่ละค่ามาตรวัดที่ ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NOT มากกว่า 5 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (5,∞)



รูปที่ 5.12 กราฟความถี่จำนวนตัวแปรชั่วคราวของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.4.4 มาตรวัด MCX

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (33.3703, 555.9) สำหรับช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (0, 3.6589) จากรูปที่ 5.13 เป็นกราฟแสดงความถี่จำนวนค่าความซับซ้อนของเมทธอดของแต่ละค่ามาตรวัด MCX โดยแกนนอนคือ ค่ามาตรวัดของ MCX และแกนตั้งคือ จำนวนค่าความซับซ้อนของเมทธอด กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนค่าความซับซ้อนของเมทธอดของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนค่าความซับซ้อนของเมทธอดของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด MCX มากกว่า 33.3703 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (33.3703,∞)

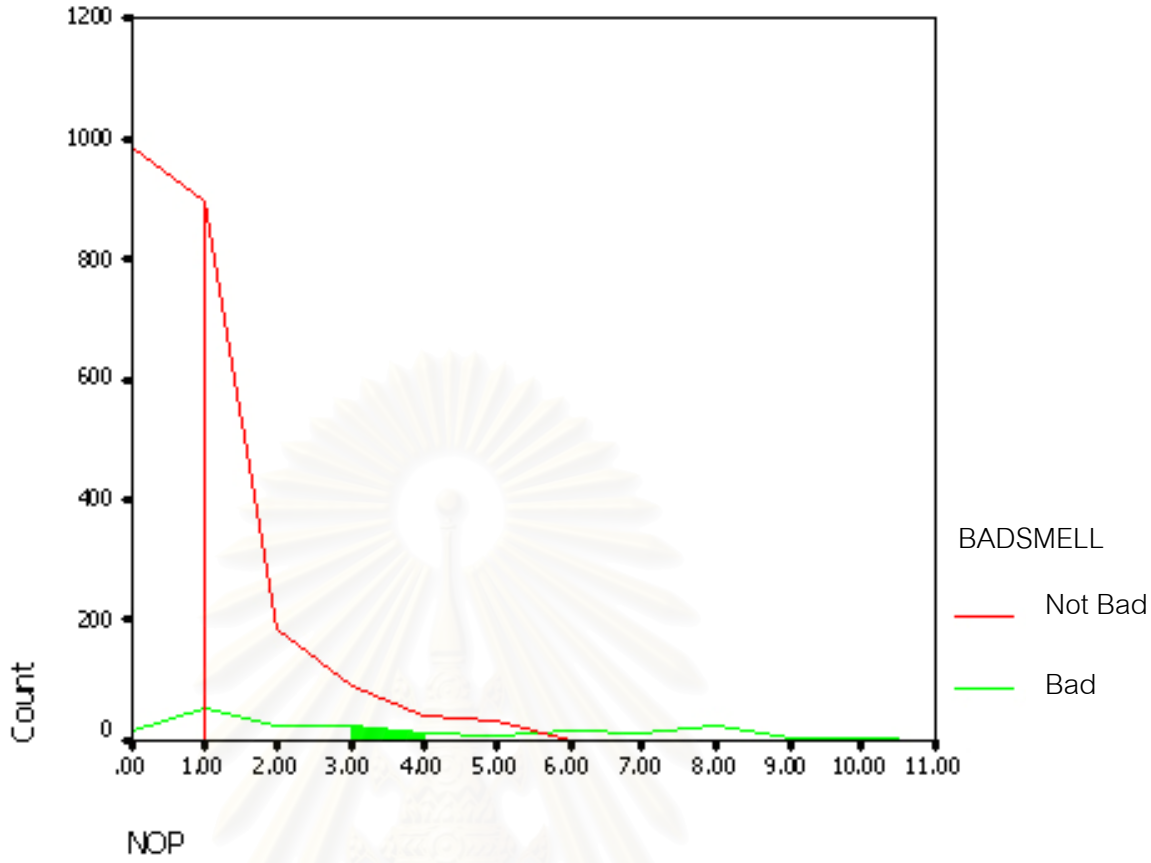


รูปที่ 5.13 กราฟความถี่จำนวนค่าความซับซ้อนของเมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

6.4.5 ร่องรอยที่ไม่ดีแบบ Long Parameter List

6.4.5.1 มาตรวัด NOP

ใช้การพิจารณาที่ไม่มีส่วนที่ซ้อนทับกันระหว่าง ช่วงค่าของมาตรวัดเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี ในกรณีที่ 1 ดังนั้นช่วงค่ามาตรวัดใหม่คือ (0, 1) สำหรับช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีและช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (7, 11) จากรูปที่ 5.14 เป็นกราฟแสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัด NOP โดยแกนนอนคือ ค่า มาตรวัดของ NOP และแกนตั้งคือ จำนวนพารามิเตอร์ในเมทธอด กราฟที่มีค่า BADSMELL = Bad คือกราฟที่แสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และกราฟที่มีค่า BADSMELL = Not Bad คือกราฟที่แสดงความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี ซึ่งกราฟนี้แสดงให้เห็นว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ มาตรวัด NOP มากกว่า 7 เป็นต้นไปดังนั้นช่วงค่ามาตรวัดใหม่คือ (7,∞)



รูปที่ 5.14 กราฟความถี่จำนวนพารามิเตอร์ในเมทธอดของแต่ละค่ามาตรวัดของการเกิดและไม่เกิดร่องรอยที่ไม่ดี

จากการปรับช่วงค่ามาตรวัดสามารถสรุปช่วงค่ามาตรวัดใหม่ ดังแสดงในตารางที่ 5.8 หลังจากนั้นทำการประเมินความน่าเชื่อถือกับข้อมูลทดสอบอีกครั้งโดยใช้ช่วงค่ามาตรวัดใหม่เพื่อตรวจสอบความถูกต้อง ความผิดพลาดและไม่สามารถบอกได้ ดังแสดงในตารางที่ 5.9

จากตารางที่ 5.9 ทุกมาตรวัดมีเปอร์เซ็นต์ความถูกต้องมากกว่าผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ที่ไม่สามารถบอกได้ และมีเปอร์เซ็นต์ความถูกต้องมากกว่าตารางที่ 5.7 เช่น ร่องรอยที่ไม่ดีแบบ Large Class มาตรวัด NIM มีเปอร์เซ็นต์ความถูกต้องก่อนปรับช่วงค่าคือ 4 เปอร์เซ็นต์ความผิดพลาดคือ 13.5 และเปอร์เซ็นต์ที่ไม่สามารถบอกได้ 82.5 ดังนั้นผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ที่ไม่สามารถบอกได้คือ 96 ซึ่งมากกว่าเปอร์เซ็นต์ความถูกต้อง แต่หลังจากปรับช่วงค่าแล้วมีเปอร์เซ็นต์ความถูกต้องสูงขึ้นเป็น 73.5 และผลรวมของเปอร์เซ็นต์ความผิดพลาดและเปอร์เซ็นต์ที่ไม่สามารถบอกได้คือ 26.5 ซึ่งน้อยกว่าเปอร์เซ็นต์ความถูกต้อง เป็นต้น ดังนั้นจึงควรใช้ช่วงค่าที่ไม่เกิดร่องรอยที่ไม่ดีและช่วงค่าที่เกิดร่องรอยที่ไม่ดีในตารางที่ 5.9

ตารางที่ 5.8 แสดงช่วงของค่ามาตรวัดที่เกิดไม่เกิดร่องรอยที่ไม่ดีใหม่

ร่องรอยที่ไม่ดี	มาตรวัด	ช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี	ช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี
Data Class	NOAM	(8,∞)	(0,1)
	NCA	(24,∞)	(0,5)
Large Class	NIM	(20, ∞)	(0,9)
	NIV	(10, ∞)	(0,2)
	TCC	(0.038,0.67)	(0,0.6196)
	NLOC	(56,∞)	(0,45)
	AMC	(8, ∞)	(0,7)
Lazy Class	NIM	(0,8)	(27, ∞)
	NIV	(0,3)	(11, ∞)
Long Method	NOS	(17, ∞)	(0,2)
	NOP	(3, ∞)	(0,1)
	NOT	(5, ∞)	(0,1)
	MCX	(33.3703, ∞)	(0,3.6589)
Long Parameter List	NOP	(7, ∞)	(0,1)

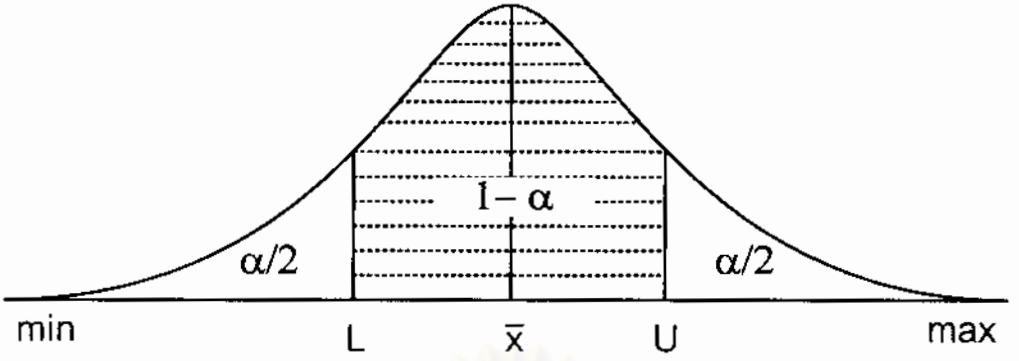
ตารางที่ 5.9 แสดงความถูกต้อง ผิดพลาดและไม่สามารถบอกได้ของช่วงค่ามาตรวัดหลังปรับค่าสำหรับทำนายค่าร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	มาตรวัด	ค่าร่องรอยที่ไม่ดี	จำนวนข้อมูลทดสอบ			รวม
			ถูกต้อง	ผิดพลาด	ไม่สามารถบอกได้	
Data Class	NOAM	จำนวน	131	69	0	200
		เปอร์เซ็นต์	65.5	34.5	0	100
	NCA	จำนวน	511	273	0	784
		เปอร์เซ็นต์	65.2	34.8	0	100
Large Class	NIM	จำนวน	147	33	20	200
		เปอร์เซ็นต์	73.5	16.5	10	100
	NIV	จำนวน	142	24	34	200
		เปอร์เซ็นต์	71	12	17	100

ตารางที่ 5.9 แสดงความถูกต้อง ผิดพลาดและไม่สามารถบอกได้ของช่วงค่ามาตรวัดหลังปรับค่าสำหรับทำนายค่าร่องรอยที่ไม่ดี (ต่อ)

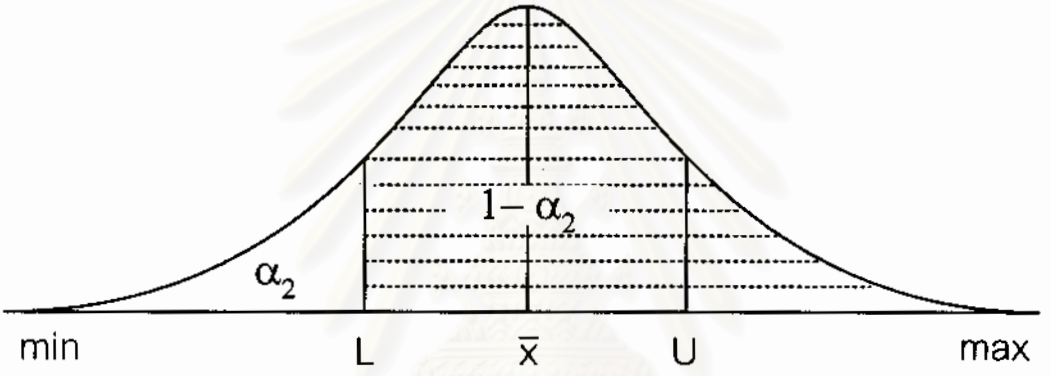
ร่องรอยที่ไม่ดี	มาตรวัด	ค่าร่องรอยที่ไม่ดี	จำนวนข้อมูลทดสอบ			รวม
			ถูกต้อง	ผิดพลาด	ไม่สามารถบอกได้	
Large Class	TCC	จำนวน	90	65	45	100
		เปอร์เซ็นต์	45	32.5	22.5	100
	NLOC	จำนวน	155	37	8	200
		เปอร์เซ็นต์	77.5	18.5	4	100
	AMC	จำนวน	122	67	11	200
		เปอร์เซ็นต์	61	33.5	5.5	100
Lazy Class	NIM	จำนวน	155	45	0	20
		เปอร์เซ็นต์	77.5	22.5	0	100
	NIV	จำนวน	170	2	28	200
		เปอร์เซ็นต์	85	1	14	100
Long Method	NOS	จำนวน	120	80	0	200
		เปอร์เซ็นต์	60	40	0	100
	NOP	จำนวน	114	86	0	200
		เปอร์เซ็นต์	57	43	0	100
	NOT	จำนวน	152	48	0	200
		เปอร์เซ็นต์	76	24	0	100
	MCX	จำนวน	124	76	0	200
		เปอร์เซ็นต์	62	38	0	100
Long Parameter List	NOP	จำนวน	898	298	0	1196
		เปอร์เซ็นต์	75.08	24.92	0	100

ช่วงค่าของมาตรวัดเดิมคือ (L, U) ซึ่ง L คือ ค่าต่ำสุดของช่วงค่ามาตรวัด และ U คือ ค่าสูงสุดของช่วงค่ามาตรวัด โดยใช้ความเชื่อมั่น 95 % หรือ 1- α = 95 ดังนั้น α = 0.05 และ $\alpha/2$ = 0.025 แสดงดังรูปที่ 5.15

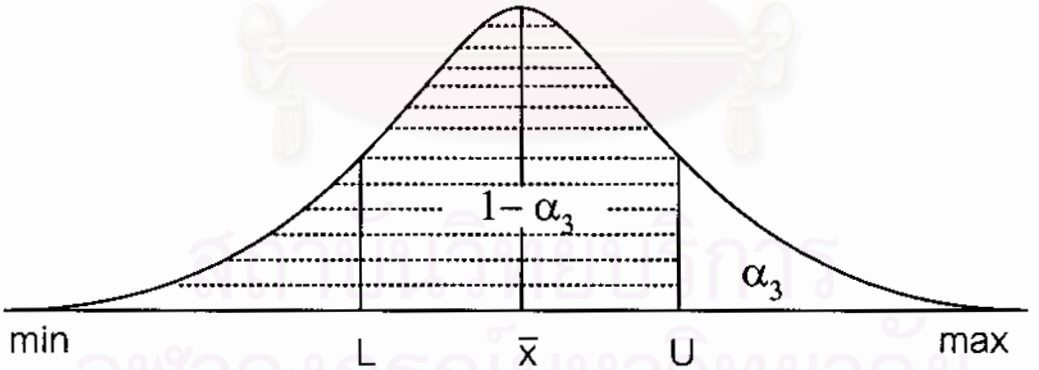


รูปที่ 5.15 แสดงช่วงค่าของมาตรวัดเดิมคือ (L, U)

แต่เนื่องจากมีการเปลี่ยนแปลงช่วงค่าของมาตรวัด จึงทำให้ระดับความเชื่อมั่นเปลี่ยนไปด้วย แสดงดังรูปที่ 5.16 หรือ 5.17



รูปที่ 5.16 แสดงช่วงค่าของมาตรวัดใหม่คือ (L, max)



รูปที่ 5.17 แสดงช่วงค่าของมาตรวัดใหม่คือ (min, U)

จากรูปที่ 5.16 และ 5.17 ค่า α_2 และ α_3 มีค่าเท่ากับ $\alpha/2$ คือ 0.025 ดังนั้นระดับความเชื่อมั่นใหม่ของช่วงค่า (L, max) และ (min, U) เท่ากับ 0.975 หรือ 97.5 %

ซึ่งแสดงระดับความเชื่อมั่นใหม่ของมาตรวัดต่างๆ ดังในตารางที่ 5.10

ตารางที่ 5.10 ค่าระดับความเชื่อมั่นใหม่

ร่องรอยที่ไม่ดี	มาตรวัด	เกิดร่องรอยที่ไม่ดี	n	$\bar{x}$	σ	L	U	ระดับความเชื่อมั่นใหม่ (%)
Data Class	NOPA	เกิด	142	3.6056	9.61748	5	5	95.00
		ไม่เกิด	615	0.3106	0.87088	0	0	95.00
	NOAM	เกิด	142	18.8732	65.36215	8	∞	97.50
		ไม่เกิด	615	1.1789	1.43271	0	1	97.50
	NSOC	เกิด	6388	0.0437	0.20667	0	0	95.00
		ไม่เกิด	7	6.1429	0.89974	5	7	97.50
	NSM + NSOC	เกิด	6388	0.1752	1.38936	0	0	95.00
		ไม่เกิด	7	9.7143	2.69037	7	12	95.00
	NCA	เกิด	412	27.4417	36.06096	24	∞	95.00
		ไม่เกิด	1918	0.3978	1.10282	0	5	95.00
	NAOC	เกิด	412	0.5267	3.12073	0	1	95.00
		ไม่เกิด	1918	0.0422	0.35937	0	0	95.00
Large Class	NIM	เกิด	108	23.3796	18.42576	20	∞	97.50
		ไม่เกิด	654	7.3823	17.53944	0	9	97.50
	NIV	เกิด	108	12.1111	8.76520	10	∞	97.50
		ไม่เกิด	652	1.7807	3.63792	0	2	97.50
	TCC	เกิด	108	0.0556	0.09253	0.038	0.27	97.50
		ไม่เกิด	652	0.5564	0.82226	0	0.6196	97.50
	NLOC	เกิด	108	85.8687	155.4907	56	∞	97.50
		ไม่เกิด	654	31.0321	183.1067	0	45	97.50
	AMC	เกิด	109	9.9406	9.66615	8	∞	97.50
		ไม่เกิด	654	5.9370	9.85563	0	7	97.50
Lazy Class	NIM	เกิด	690	7.2304	15.36651	0	8	97.50
		ไม่เกิด	73	32.7945	25.99036	27	∞	97.50
	NIV	เกิด	690	2.2261	3.92283	0	3	97.50
		ไม่เกิด	73	13.1370	10.93963	11	∞	97.50
Long Method	NOS	เกิด	186	20.6344	24.80989	17	∞	97.50
		ไม่เกิด	2229	1.8964	1.83021	0	2	97.50
	NOP	เกิด	186	3.5699	2.84709	3	∞	97.50
		ไม่เกิด	2229	0.83	1.02188	0	1	97.50
	NOT	เกิด	186	5.6989	6.24679	5	∞	97.50
		ไม่เกิด	2229	0.4868	1.22966	0	1	95.00
	MCX	เกิด	2159	43.4268	66.023	33.370	∞	97.50
		ไม่เกิด	168	3.4034	6.05350	3.1479	3.36589	95.00

ตารางที่ 5.10 ค่าระดับความเชื่อมั่นใหม่ (ต่อ)

ร่องรอยที่ไม่ดี	มาตรวัด	เกิดร่องรอยที่ไม่ดี	n	$\bar{x}$	σ	L	U	ระดับความเชื่อมั่นใหม่ (%)
Long Parameter List	NOP	เกิด	56	7.4107	1.1406	7	∞	97.50
		ไม่เกิด	2359	0.8898	1.0735	0	1	97.50
Switch Statement	NOSS	เกิด	19	1.0526	0.22942	1	1	97.50
		ไม่เกิด	2415	0.0083	0.09510	0	0	97.50

6.5 นำเสนอวิธีการนำช่วงค่าของมาตรวัดต่าง ๆ สำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน

ขั้นตอนการนำช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีและช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีไปใช้งานมี 2 วิธีดังนี้

6.5.1 การใช้เพียงมาตรวัดเดียวในการตรวจจับร่องรอยที่ไม่ดี

6.5.2 การใช้หลายมาตรวัดโดยใช้เงื่อนไขแบบ OR กันในการตรวจจับร่องรอยที่ไม่ดี ซึ่งมีวิธีการใช้งาน ดังนี้

6.5.2.1 คำนวณค่ามาตรวัดของร่องรอยที่ไม่ดีที่เลือก 8 แบบ ได้แก่ Data Class, Feature Envy, Large Class, Lazy Class, Long Method, Long Parameter List, Refused Bequest และ Switch Statement จากโปรแกรมต้นฉบับภาษาจาวา

6.5.2.2 เปรียบเทียบค่ามาตรวัดทั้งหมดของร่องรอยที่ไม่ดีที่เลือกไว้ว่า

ถ้ามีค่าของมาตรวัดอย่างน้อยหนึ่งตัว อยู่ในช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี สรุปว่าคลาสนั้นเกิดร่องรอยที่ไม่ดี

ถ้าไม่มีค่ามาตรวัดตัวไหน อยู่ในช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีให้เปรียบเทียบค่ามาตรวัดว่ามีมาตรวัดอย่างน้อยหนึ่งตัวที่ อยู่ในช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี สรุปว่าคลาสนั้นไม่เกิดร่องรอยที่ไม่ดี

ถ้าไม่มีค่ามาตรวัดตัวไหน อยู่ในช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและในช่วงค่าของมาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี สรุปว่าคลาสนั้นไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่

ช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีทั้ง 6 แบบแสดงตารางที่ 5.11

ตัวอย่าง การใช้หลายมาตรวัดโดยใช้เงื่อนไขแบบ OR กันในการตรวจจับร่องรอยที่ไม่ดีไปใช้งาน เริ่มจากเลือกร่องรอยที่ไม่ดีแบบ Large Class คำนวณมาตรวัด NIM, NIV, TCC, NLOC และ AMC ถ้ามีค่าของมาตรวัดใดมาตรวัดทั้ง 5 ตัวอยู่ในช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีสรุปว่าคลาสนั้นเกิดร่องรอยที่ไม่ดีแบบ Large Class

NLOC และ AMC ถ้ามีค่าของมาตรวัดใดมาตรวัดทั้ง 5 ตัวอยู่ในช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีสรุปว่าคลาสนั้นเกิดร่องรอยที่ไม่ดีแบบ Large Class

ตารางที่ 5.11 สรุปช่วงของค่ามาตรวัดที่เกิดและไม่เกิดร่องรอยที่ไม่ดี

ร่องรอยที่ไม่ดี	มาตรวัด	ช่วงค่าของมาตรวัดที่	
		เกิดร่องรอยที่ไม่ดี	ไม่เกิดร่องรอยที่ไม่ดี
Data Class	NOPA	(5,5)	(0,0)
	NOAM	(8,771)	(0,1)
	NSOC	(0,0)	(5,7)
	NSM+NSOC	(0,0)	(7,12)
	NCA	(24,310)	(0,0)
	NAOC	(6,36)	(0,0)
Large Class	NIM	(20,350)	(0,9)
	NIV	(10,64)	(0,2)
	TCC	(0.038, 0.67)	(0, 0.6196)
	NLOC	(56,1155)	(0,45)
	AMC	(8,40.36)	(0,7)
Lazy Class	NIM	(0,8)	(27,132)
	NIV	(0,3)	(11,64)
Long Method	NOS	(17,245)	(0,2)
	NOP	(3,11)	(0,1)
	NOT	(5,52)	(0,1)
	MCX	(33.3703,555.9)	(0,3.6589)
Long Parameter List	NOP	(7,11)	(0,1)
Switch statement	NOSS	(1,1)	(0,0)

5.6 ประเมินความถูกต้องของวิธีการการใช้หลายมาตรวัดโดยใช้เงื่อนไขแบบ OR กันในการตรวจจับร่องรอยที่ไม่ดีไปใช้งาน

นำชุดข้อมูลทดสอบมาเปรียบเทียบค่าที่ได้การใช้หลายมาตรวัดโดยใช้เงื่อนไขแบบ OR กันในการตรวจจับร่องรอยที่ไม่ดีไปใช้งานกับค่าที่ได้จากผู้เชี่ยวชาญเป็นผู้กำหนด เพื่อตรวจสอบความถูกต้อง ความผิดพลาดและไม่สามารถบอกได้ ดังแสดงในตารางที่ 5.13

ตารางที่ 5.13 แสดงความถูกต้อง ผิดพลาดและไม่สามารถบอกได้ของวิธีการนำช่วงค่าของมาตรวัดต่างๆ สำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน

ร่องรอยที่ไม่ดี	สำหรับ	ค่าร่องรอยที่ไม่ดี	จำนวนข้อมูลทดสอบ			รวม
			ถูกต้อง	ผิดพลาด	ไม่สามารถบอกได้	
Data Class	คลาส	จำนวน	179	21	0	200
		เปอร์เซ็นต์	89.5	10.5	0	100
	เมทธอด	จำนวน	1365	13	0	1378
		เปอร์เซ็นต์	99	1	0	100
	คุณลักษณะ	จำนวน	524	260	0	784
		เปอร์เซ็นต์	66.8	33.2	0	100
Large Class	คลาส	จำนวน	135	65	0	200
		เปอร์เซ็นต์	67.5	32.5	0	100
Lazy Class	คลาส	จำนวน	172	28	0	200
		เปอร์เซ็นต์	86	14	0	100
Long Method	เมทธอด	จำนวน	179	21	0	200
		เปอร์เซ็นต์	89.5	10.5	0	100
Long Parameter List	พารามิเตอร์	จำนวน	898	298	0	1196
		เปอร์เซ็นต์	75.1	24.9	0	100
Switch Statement	เมทธอด	จำนวน	1405	5	0	1410
		เปอร์เซ็นต์	99.64	0.36	0	100

จากตารางที่ 5.13 แสดงให้เห็นร่องรอยที่ไม่ดีแบบ Data Class, Large Class, Lazy Class, Long Method, Long Parameter List และ Switch Statement มีเปอร์เซ็นต์ความ

ความถูกต้องของวิธีการการใช้หลายมาตรวัดโดยใช้เงื่อนไขแบบ OR กันในการตรวจจับร่องรอยที่ไม่ดีนำไปใช้งานมากกว่าหรือเท่ากับการนำช่วงค่าเพียงมาตรวัดเดียวของร่องรอยที่ไม่ดีไปตรวจจับร่องรอยที่ไม่ดี เช่น ร่องรอยที่ไม่ดีแบบ Lazy Class เมื่อนำมาตรวัดเดียวไปตรวจจับร่องรอยที่ไม่ดี คือ มาตรวัด NIM มีเปอร์เซ็นต์ความถูกต้องเท่ากับ 77.5 หรือ มาตรวัด NIV มีเปอร์เซ็นต์ความถูกต้องเท่ากับ 85 แต่เมื่อนำมาตรวัดทั้งสองมาใช้เงื่อนไข OR กันมีเปอร์เซ็นต์ความถูกต้องเท่ากับ 86 ซึ่งมากกว่าการใช้เพียงมาตรวัดเดียว ดังนั้นสรุปได้ว่าการนำวิธีการนำช่วงค่าของมาตรวัดต่างๆ สำหรับตรวจจับร่องรอยที่ไม่ดีนำไปใช้งานได้

5.7 เปรียบเทียบช่วงค่าของโปรแกรมภาษาซีพลัสพลัสและจาวา

ช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีของภาษาจาวาคือ ช่วงค่าที่หาได้จากงานวิจัยนี้ และช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีของภาษาซีพลัสพลัสนำมาจาก [5] และ [6] สามารถเปรียบเทียบร่องรอยไม่ดี 6 แบบ คือ Data Class, Large Class, Lazy Class, Long Method, Long Parameter List และ Switch Statement ส่วนร่องรอยที่ไม่ดี 2 แบบที่เหลือ คือ Feature Envy และ Refuse Bequest นั้นไม่สามารถหาช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีของภาษาจาวาในงานวิจัยนี้ได้ เนื่องจากข้อมูลที่เกิดร่องรอยที่ไม่ดีแบบนี้มีจำนวนไม่พอต่อการทดลอง ซึ่งการเปรียบเทียบช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีระหว่างภาษาจาวา และซีพลัสพลัส แสดงได้ดังตารางที่ 5.14

ตารางที่ 5.13 การเปรียบเทียบช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีระหว่างภาษาจาวาและซีพลัสพลัส

ร่องรอยที่ไม่ดี	มาตรวัด	ช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี	
		สำหรับภาษาจาวา	สำหรับภาษาซีพลัสพลัส
Data Class	NOPA	(5,5)	10 คลาสที่มี NOPA มากที่สุดแต่ไม่น้อยกว่า 5
	NOAM	(8,771)	10 คลาสที่มี NOAM มากที่สุดแต่ไม่น้อยกว่า 3
	NSOC	(0,0)	-
	NSM+ NSOC	(0,0)	-
	NCA	(24,310)	-
	NAOC	(6,36)	-

Large Class	NIM	(20,350)	>20 สำหรับคลาสทั่วไป
	NIV	(10,64)	>3 สำหรับคลาสทั่วไป
	TCC	(0.038, 0.67)	เลือกคลาสที่มี TCC ต่ำที่สุด
	NLOC	(56,1155)	-
	AMC	(8,40.36)	-

ตารางที่ 5.13 การเปรียบเทียบช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีระหว่างภาษาจาวาและซีพลัสพลัส

ร่องรอยที่ไม่ดี	มาตรวัด	ช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี	
		สำหรับภาษาจาวา	สำหรับภาษาซีพลัสพลัส
Lazy Class	NIM	(0,8)	ค่าน้อยที่สุดของ NIM+NIV
	NIV	(0,3)	
Long Method	NOS	(17,245)	> 7
	NOP	(3,11)	a
	NOT	(5,52)	a
	MCX	(33.3703,555.9)	-
Long Parameter List	NOP	(7,11)	เลือกเมทธอดที่มี NOP มากที่สุด
Switch statement	NOSS	(1,1)	>0

- คือไม่มีช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดี

a คือ มาตรวัดที่ใช้การเลือกใช้วิธีรีแฟคทอริง

จากตารางที่ 5.14 พิจารณาได้ว่าช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีระหว่างภาษาจาวาและซีพลัสพลัสของร่องรอยที่ไม่ดีแบบต่างๆ มีทิศทางเดียวกัน ยกตัวอย่างเช่น ร่องรอยที่ไม่ดีแบบ Large Class มาตรวัด NIM มีช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีสำหรับภาษาจาวาคือ ตั้งแต่ 20 ถึง 350 และสำหรับภาษาซีพลัสพลัส คือ ค่าของมาตรวัดมากกว่า 20 สำหรับคลาสทั่วไป ซึ่งมีช่วงค่าของมาตรวัดสำหรับภาษาจาวาอยู่ในช่วงค่าของมาตรวัดสำหรับภาษาซีพลัสพลัส แต่ช่วงค่าของมาตรวัดสำหรับภาษาจาวามีความละเอียดกว่าช่วงค่าของมาตรวัดสำหรับภาษาซีพลัสพลัส

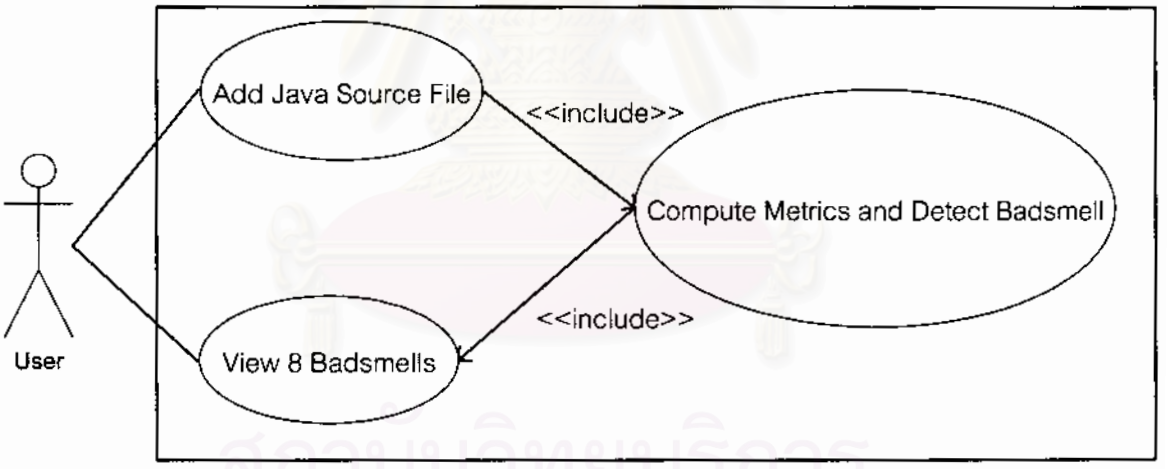
บทที่ 6
การออกแบบและพัฒนาเครื่องมือสำหรับการคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดี 8 แบบ

ในบทนี้จะอธิบายรายละเอียดเกี่ยวกับ ขั้นตอนการออกแบบและพัฒนาเครื่องมือและการตรวจสอบค่ามาตรวัดที่ได้จากเครื่องมือ ดังนี้

6.1 การออกแบบและพัฒนาเครื่องมือ

ผู้วิจัยแสดงการวิเคราะห์ และออกแบบเครื่องมือสำหรับคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดี โดยใช้ภาษายูเอ็มแอล (Unified Modeling Language - UML) ซึ่งแสดงภาพรวมของระบบ ได้แก่ แผนภาพยูสเคส (Use Case Diagram) แผนภาพคลาส (Class Diagram) และแผนภาพแสดงลำดับการทำงาน (Sequence Diagram)

6.1.1 แผนภาพยูสเคส

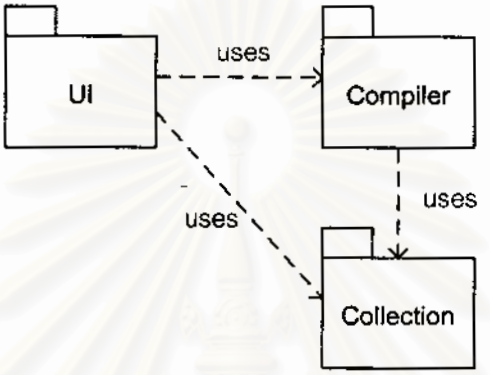


รูปที่ 6.1 แสดงแผนภาพยูเคสของเครื่องมือออกแบบเครื่องมือสำหรับคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดี

ผู้ใช้กำหนดโปรแกรมต้นฉบับภาษาจาวาที่ต้องการหาร่องรอยที่ไม่ดี หลังจากนั้นโปรแกรมจะคำนวณค่ามาตรวัดของร่องรอยที่ไม่ดีทั้ง 8 แบบพร้อมทั้งตรวจจับร่องรอยที่ไม่ดีด้วย โปรแกรมจะแสดงผลให้ผู้ใช้สามารถดูร่องรอยที่ไม่ดีทั้ง 8 แบบว่าเกิดที่คลาสไหน เมทธอดไหน และแอทริบิวต์ไหน โดยจะแสดงค่าของมาตรวัดวัดและค่าที่เกิดร่องรอยที่ไม่ดีในรูปของตารางซึ่งแยกตามร่องรอยที่ไม่ดีทั้ง 8 แบบ

6.1.2 แผนภาพคลาส

แผนภาพของระบบประกอบด้วยแผนภาพคลาส ซึ่งสามารถแบ่งออกเป็น 3 แพ็กเกจหลักคือแพ็กเกจส่วนติดต่อผู้ใช้ (UI) แพ็กเกจคอมไพล์เลอร์ (Compiler) และเพ็กเกจคำนวณค่ามาตรวัด (Collection) ซึ่งสามารถแสดงความสัมพันธ์ระหว่างแพ็กเกจต่างๆ ของระบบดังรูปที่ 6.2



รูปที่ 6.2 แผนภาพแสดงความสัมพันธ์ระหว่างแพ็กเกจต่างๆ ของระบบ

6.1.2.1 แพ็กเกจส่วนติดต่อผู้ใช้

แพ็กเกจส่วนติดต่อผู้ใช้ เป็นชุดของคลาสที่ทำหน้าที่ติดต่อกับผู้ใช้งานระบบแสดงดังรูปที่ 6.3 แพ็กเกจส่วนติดต่อผู้ใช้ ประกอบด้วยคลาสต่างๆ ได้แก่

1) คลาสเมนเฟรม (MainFrame) เป็นคลาสที่สร้างเมนูให้ผู้ใช้เลือกและเป็นเฟรมหลักที่ให้เฟรมอื่นๆ แสดงผลในเฟรมหลักนี้ คลาสเมนเฟรมประกอบด้วยคลาสต่างๆ ได้แก่ คลาสจาวาไฟล์ฟิลเตอร์ (JavaFileFilter) เป็นคลาสที่ใช้เลือกไฟล์ต้นฉบับเฉพาะไฟล์จาวา คลาสมายเรนเดอร์ (MyRender) คลาสโหนดอินโฟ (NodeInfo) และคลาสยูทิล (Utils) ทั้ง 3 คลาสเป็นคลาสที่แสดงแผนภาพต้นไม้เมื่อเลือกไฟล์ต้นฉบับจาวาแล้ว

2) DataClassTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Data Class

3) FeatureEnvyTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Feature Envy

4) LargeClassTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Large Class

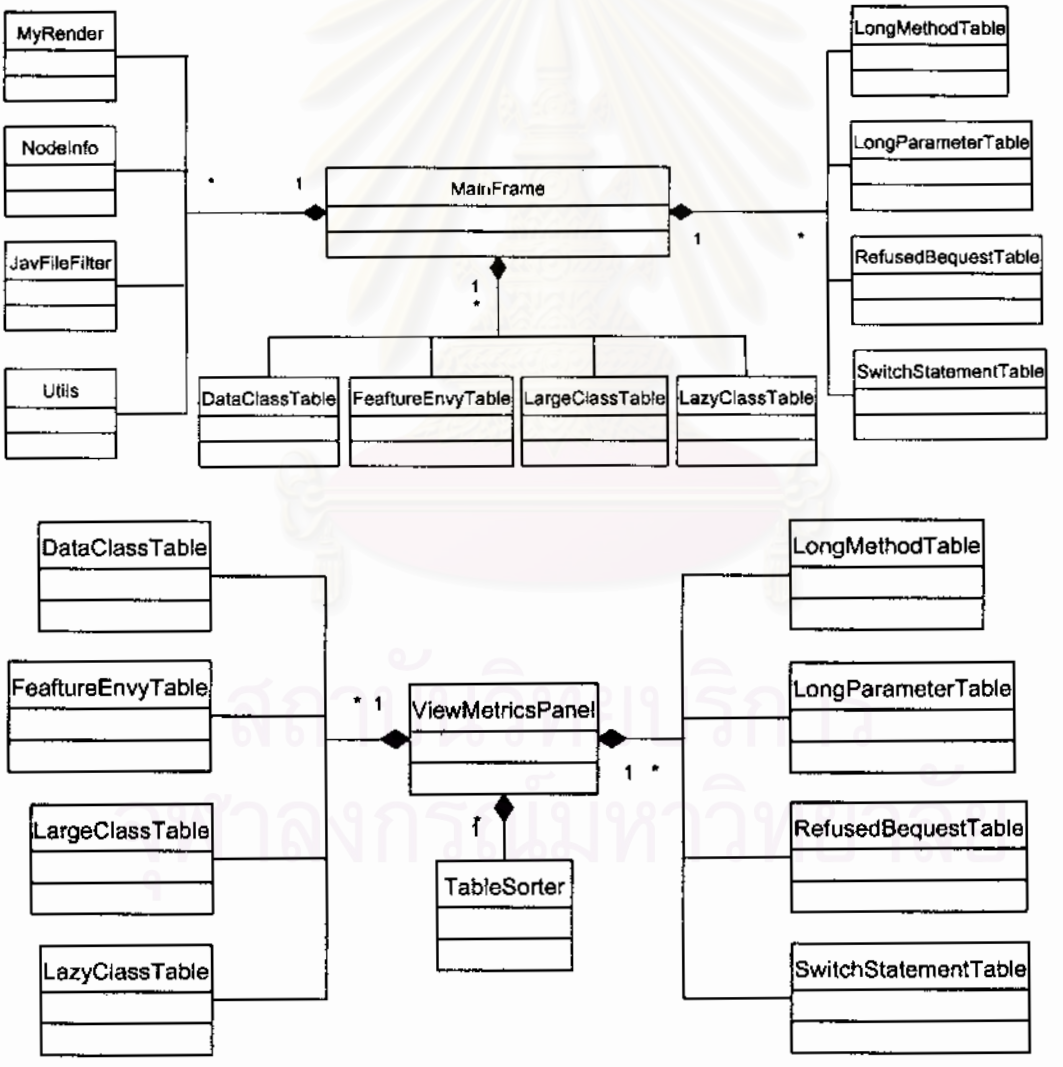
5) LazyClassTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Lazy Class

6) LongMethodTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบLong Method

7) LongParameterTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Long Parameter List

8) SwitchStatementTable เป็นคลาสที่แสดงค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีแบบ Switch Statement

และคลาสที่แสดงค่ามาตรวัดร่องรอยที่ไม่ดีประกอบด้วย ViewMetricsPanel สำหรับแสดงผลมาตรวัดต่างๆ ซึ่งคลาสต่างๆ ในแพ็กเกจนี้สืบทอดคุณสมบัติจากแพ็กเกจสวิง (swing)



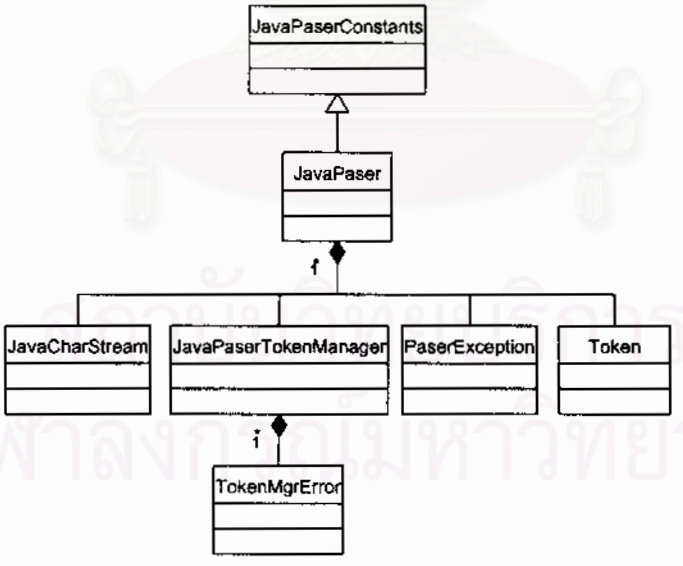
รูปที่ 6.3 แผนภาพคลาสในภาพรวมของแพ็กเกจส่วนติดต่อผู้ใช้

6.1.2.2 **แพ็กเกจคอมไพเลอร์**

แพ็กเกจคอมไพเลอร์ เป็นชุดของคลาสที่ทำหน้าที่นำโปรแกรมต้นฉบับมาผ่านขบวนการสร้างตัวแปลภาษาเพื่อสร้างซินแท็กซ์ทรี รูปที่ 6.4 แผนภาพคลาสของแพ็กเกจคอมไพเลอร์ ผู้วิจัยได้เลือกเครื่องมือช่วยสร้างตัวแปลภาษาที่ชื่อว่า จาวา คอมไพล์เลอร์ คอมไพล์เลอร์ (Java Compiler Compiler - JavaCC) โดยผลลัพธ์ที่ได้จากการประมวลผลด้วยเครื่องมือนี้จะได้ชุดของคลาสต่างๆ ดังนี้

1) คลาสจาวาพาร์เซอร์ (JavaPaser) จะเป็นคลาสหลักที่จะนำโปรแกรมต้นฉบับมาสร้างซินแท็กซ์ทรี และคลาสจาวาพร์เซอร์นี้สืบทอดอินเตอร์เฟสคลาสจาวาพาร์เซอร์คอนสแตนท์ส (JavaPaserConstants) ที่มีการประกาศค่าคงที่ต่างๆ และโทเคนต่างๆ (โทเคน คือ กลุ่มอักขระตามไวยากรณ์ของภาษาจาวา)

2) คลาสโทเคน (Token) คลาสชาร์สตรีม (JavaCharStream) และคลาสจาวาพาร์เซอร์โทเคนเมเนเจอร์ (JavaPaserTokenManager) ที่มีหน้าที่อ่านอักขระของโปรแกรมต้นฉบับและแบ่งให้เป็นโทเคน คลาสจาวาพาร์เซอร์ภายในจะประกอบไปด้วยตัวสร้างโหนดต่างๆ โดยมี CompilationUnit() เป็นโหนดราก (root node) โหนดกิ่ง (branch node) และโหนดใบ (leaf node) และโหนดต่างๆ เป็นส่วนสำคัญในการวิเคราะห์ว่าจะสามารถคำนวณค่ามาตรวัดแต่ละค่าได้จากโหนดใดบ้าง



รูปที่ 6.4 แผนภาพคลาสของแพ็กเกจคอมไพเลอร์

6.1.2.3 **แพ็กเกจคำนวณค่ามาตรวัด**

แพ็กเกจคำนวณค่ามาตรวัด เป็นชุดของคลาสหน่วยรู้จำ โดยมีคลาสมาตรวัด (Metrics) ทำหน้าที่ท่องไปบนซินแท็กซ์ทรี เพื่อเก็บข้อมูลต่างๆ จากโหนดต่างๆ และคำนวณค่ามาตรวัดต่างๆ ซึ่งแพ็กเกจนี้ประกอบด้วยคลาสได้แก่

1) คลาสมาตรวัด

คลาส Metrics เป็นคลาสที่ทำการคำนวณค่ามาตรวัดทั้ง 30 มาตรวัดจากไฟล์จาวาต้นฉบับ ซึ่งคลาส Metrics ประกอบไปด้วยคลาส JClass, JMessageSend, JMethod และ JVariable

2) คลาสรู้จำเกี่ยวกับคลาส (JClass)

คลาสรู้จำเกี่ยวกับคลาสเป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับคลาสจากไฟล์จาวาต้นฉบับเพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด

3) คลาสรู้จำเกี่ยวกับเมทธอด (JMethod)

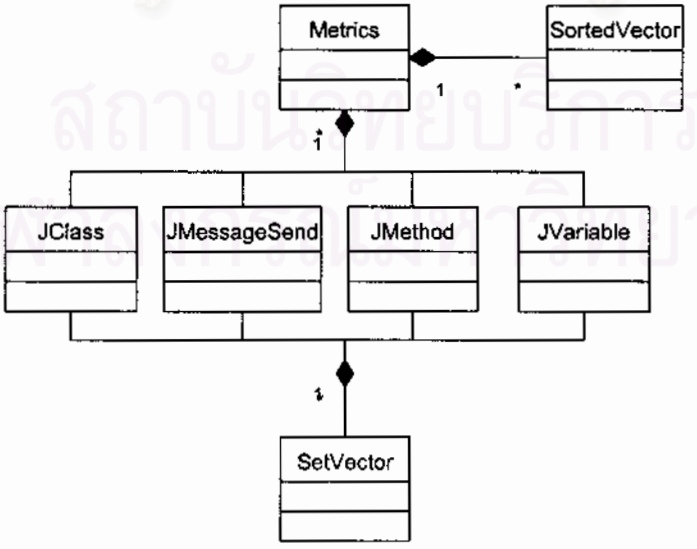
คลาสรู้จำเกี่ยวกับเมทธอดเป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับเมทธอดจากไฟล์จาวาต้นฉบับเพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด

4) คลาสรู้จำเกี่ยวกับตัวแปร (JVariable)

คลาสรู้จำเกี่ยวกับตัวแปรเป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับตัวแปรจากไฟล์จาวาต้นฉบับเพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด

5) คลาสรู้จำเกี่ยวกับแมสเซส (JMessageSend) ซึ่งจะถูกเก็บอยู่ในรูปของตัวแปรแบบเวคเตอร์ ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับแมสเซสจากไฟล์จาวาต้นฉบับเพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด

โดยมีคลาสที่สืบทอดคุณสมบัติมาจากคลาสเวคเตอร์ (java.util.Vector) เช่น คลาสซอร์ดเตดเวคเตอร์ (SortedVector) และคลาสเซตเวคเตอร์ (SetVector) โดยรูปที่ 6.5 แสดงให้เห็นถึงความสัมพันธ์ของคลาสต่างๆ ในชุดแพ็กเกจนี้



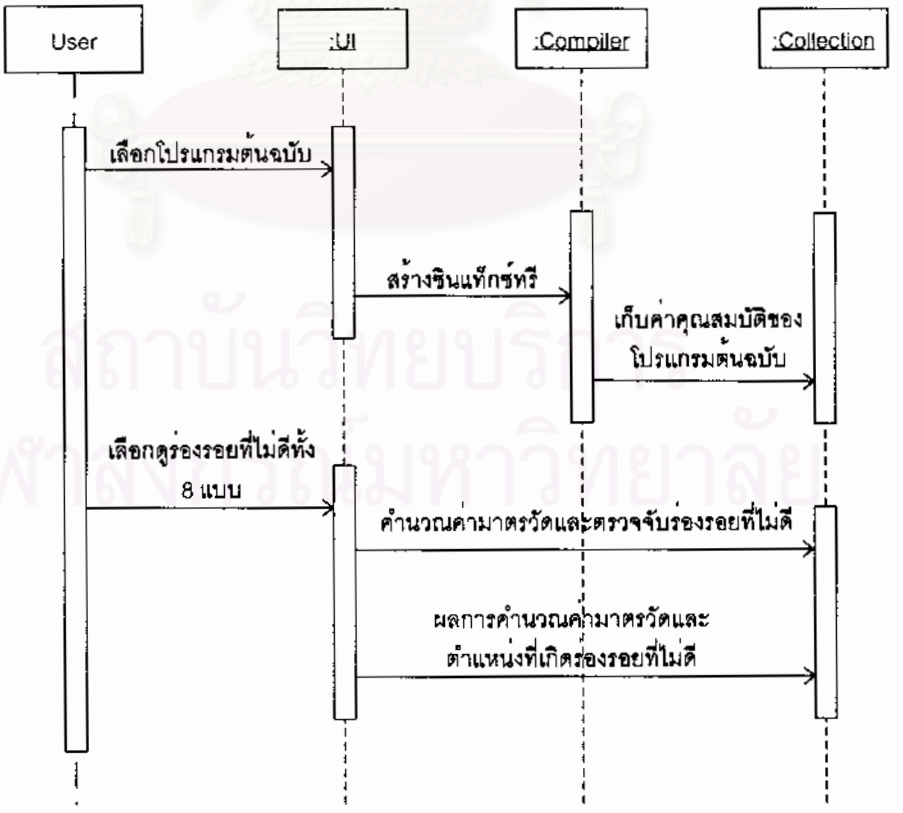
รูปที่ 6.5 แผนภาพคลาสของแพ็กเกจคำนวณค่าวัดวัด

6.1.3 แผนภาพแสดงลำดับการทำงาน

แผนภาพแสดงลำดับการทำงานของเครื่องมือคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดีมี 11 แผนภาพ คือ แผนภาพแสดงลำดับการทำงานรวมของการเรียกใช้แพ็กเกจ แผนภาพแสดงลำดับการทำงานกำหนดโปรแกรมต้นฉบับ แผนภาพแสดงลำดับการทำงานคำนวณค่ามาตรวัด แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดี 8 แบบ ได้แก่ Data Class, Feature Envy, Large Class, Lazy Class, Long Method, Long Parameter List, Refuse Bequest และ Switch Statement

6.1.3.1 แผนภาพแสดงลำดับการทำงานรวมของการเรียกใช้แพ็กเกจ

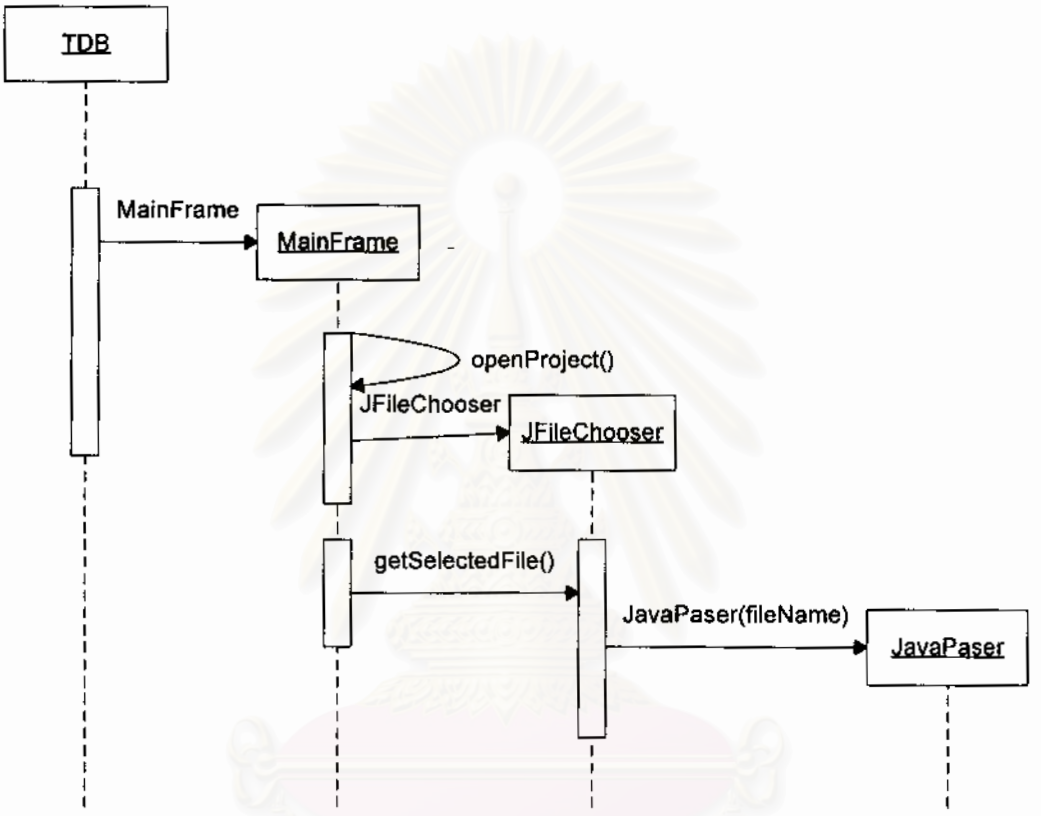
เมื่อผู้ใช้ต้องการดูค่ามาตรวัด ดังรูปที่ 6.6 โดยเริ่มจากผู้ใช้ทำการเลือกโปรแกรมต้นฉบับจากวัตถุ "UI" แล้วนำโปรแกรมต้นฉบับที่ผู้ใช้เลือกมาทำการสร้างซินแท็กซ์ทรีด้วยวัตถุ "Collection" จากนั้นผู้ใช้ทำการเลือกดูค่าตัววัด หลังจากที่ได้เลือกโปรแกรมต้นฉบับเข้าสู่ระบบ โดยระบบจะทำการจัดรูปแบบการแสดงผลด้วยวัตถุ "UI" แล้วทำการส่งคำร้องขอเพื่อทำการคำนวณค่ามาตรวัดจากวัตถุ "Collection" แล้วส่งผลของค่ามาตรวัดกลับมาแล้วแสดงผลและตรวจจับร่องรอยที่ไม่ดีด้วยวัตถุ"UI"



รูปที่ 6.6 แผนภาพแสดงลำดับการทำงานเมื่อผู้ใช้ต้องการดูร่องรอยที่ไม่ดี

6.1.3.2 แผนภาพแสดงลำดับการทำงานกำหนดโปรแกรมต้นฉบับ

แผนภาพแสดงลำดับการทำงานกำหนดโปรแกรมต้นฉบับดังรูปที่ 6.7 เริ่มจากเครื่องเริ่มสร้างเฟรมเพื่อให้ผู้ใช้เลือกโปรแกรมต้นฉบับภาษาจาวา แล้วนำโปรแกรมต้นฉบับที่ผู้ใช้เลือกมาทำการสร้างพาร์เซอร์ของภาษาจาวา เพื่อข้อข้อมูลในโปรแกรม



รูปที่ 6.7 แผนภาพแสดงลำดับการทำงานการกำหนดโปรแกรมต้นฉบับ

6.1.3.3 แผนภาพแสดงลำดับการทำงานคำนวณค่ามาตรวัด

แผนภาพแสดงลำดับการทำงานคำนวณค่ามาตรวัดแสดงดังรูปที่ 6.8 เริ่มจากอ่านโปรแกรมต้นฉบับมาสร้างซินแท็กซ์ทรี และเก็บข้อมูลที่สำคัญคือ แพกเกจ คลาส เมทธอด คุณลักษณะ และข้อความที่ส่งภายในเมทธอด โดยมาตรวัดทางตรงซึ่งได้จากการเก็บข้อมูลโดยตรงได้แก่ NIM, NIV และ NLOC เป็นต้น ส่วนมาตรวัดทางอ้อมคือมาตรวัดที่นำมาตรวัดทางตรงมาคำนวณ ซึ่งได้แก่ TCC และ AMC เป็นต้น

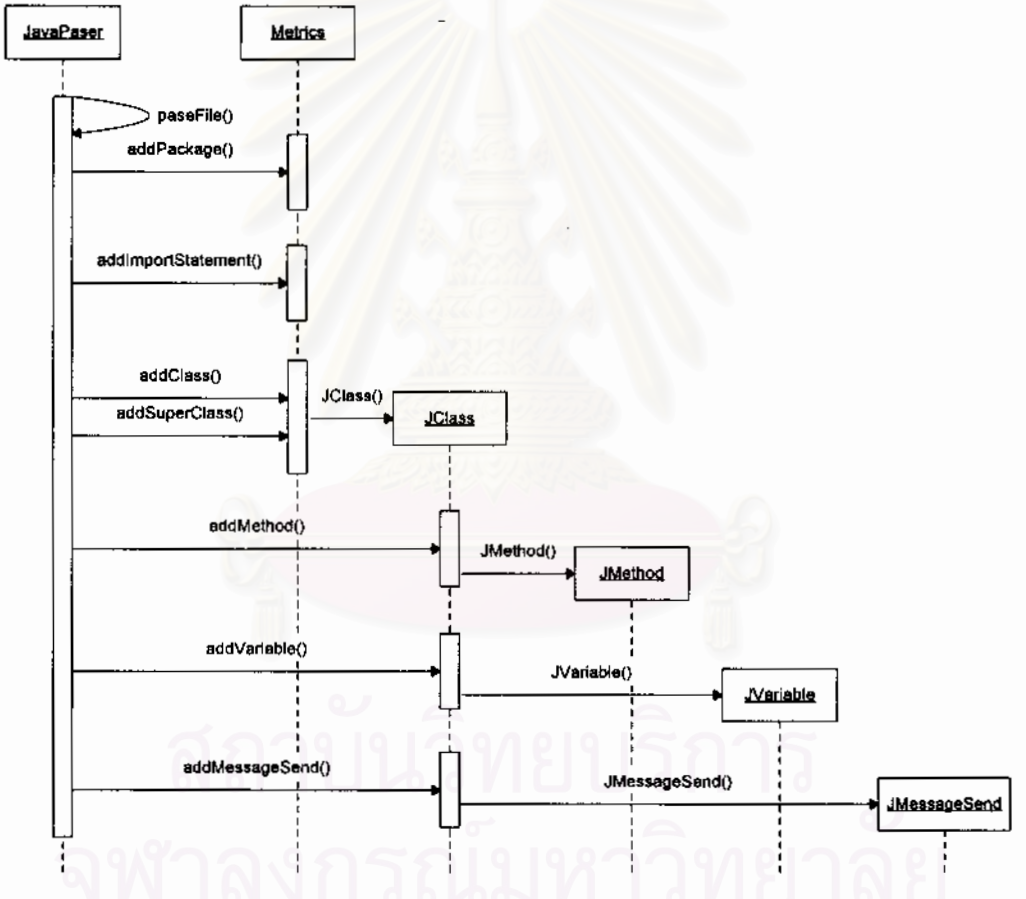
6.1.3.4 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Data Class

แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Data Class แสดงดังรูปที่ 6.9 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด

คือ NSM, NSOC, NCA, NAOC, NGM, NGOC, NOPAและ NOAM มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.5 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Feature Envy

แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Feature Envy แสดงดังรูปที่ 6.10 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NCM, NCDA และ NCRA มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา



รูปที่ 6.8 แผนภาพแสดงลำดับการทำงานการคำนวณค่ามาตรวัด

6.1.3.6 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Large Class

แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Large Class แสดงดังรูปที่ 6.11 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NIM, NIV, TCC, NLOC และ AMC มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.7 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Lazy Class

แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Lazy Class แสดงดังรูปที่ 6.12 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NIM, NIV, DIT และ AMC มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.8 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Long Method

แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Long Method แสดงดังรูปที่ 6.13 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NOS, NOP, NOT และ MCX มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.9 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Long Parameter List

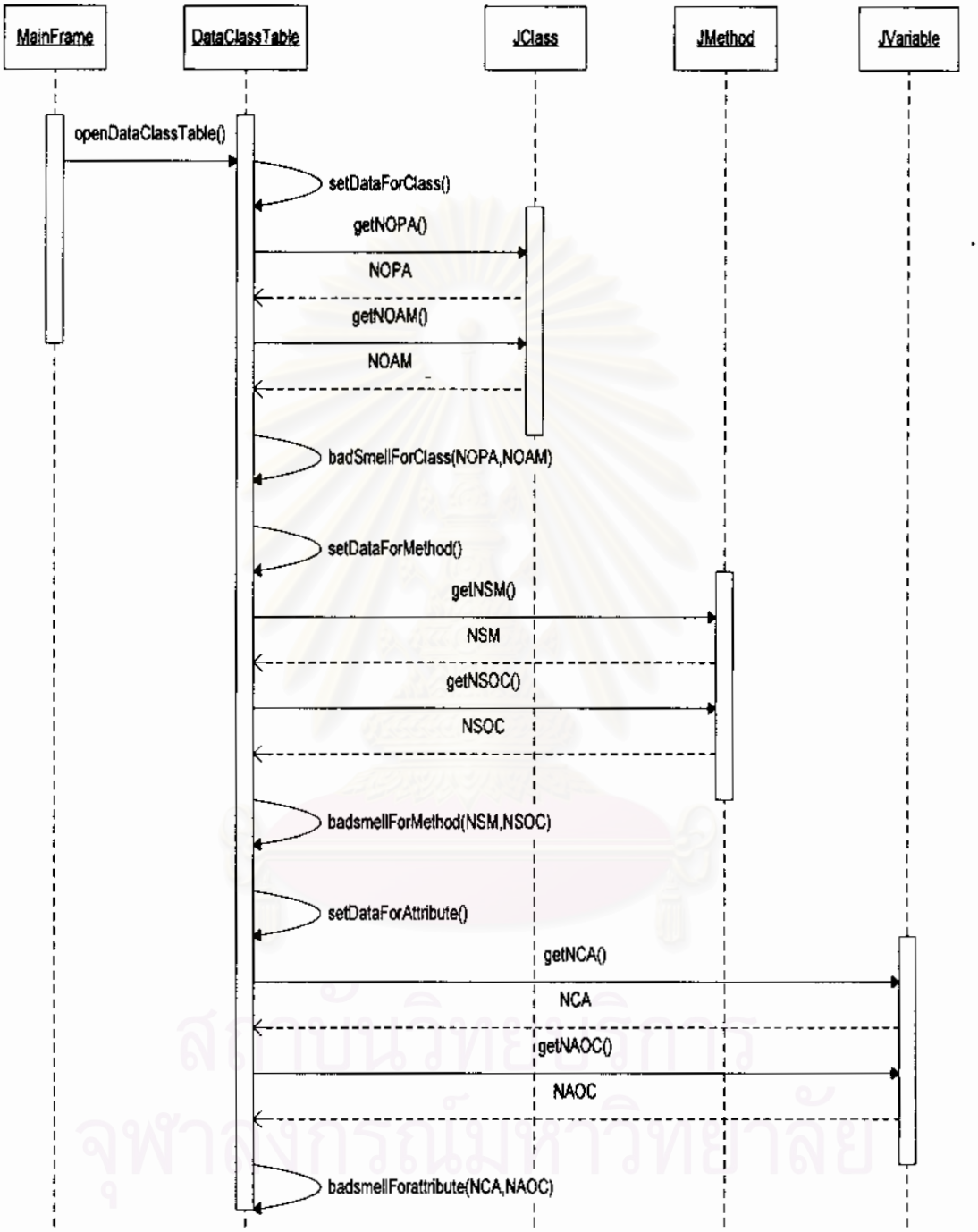
แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Long Parameter List แสดงดังรูปที่ 6.14 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NOP มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.10 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Refuse Bequest

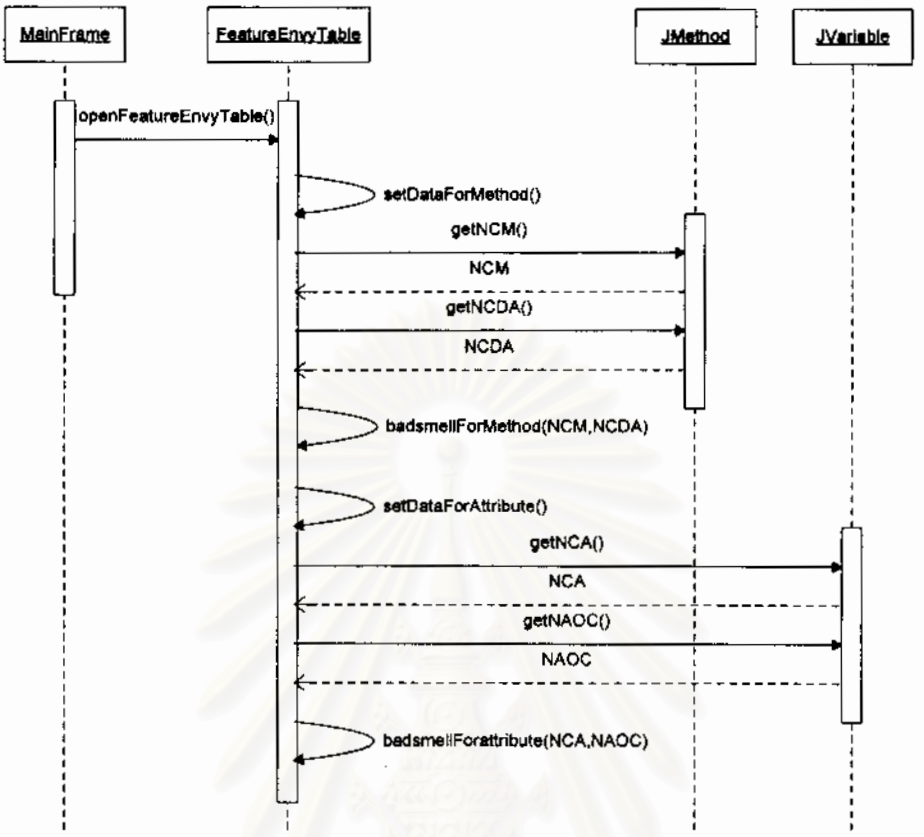
แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Refused Brquest แสดงดังรูปที่ 6.15 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NOCM, NSMP, NOCA และ NSAP มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา

6.1.3.11 แผนภาพแสดงลำดับการทำงานเรียกดูค่าของมาตรวัดและค่าร่องรอยที่ไม่ดีแบบ Switch Statement

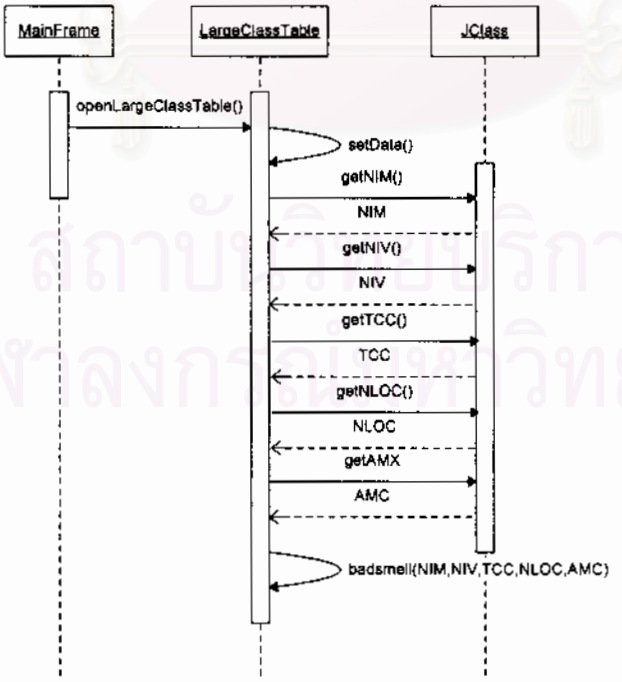
แผนภาพแสดงลำดับการทำงานเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Switch Statement แสดงดังรูปที่ 6.16 เริ่มจากสร้างเฟรมเพื่อแสดงผลของค่ามาตรวัดและดึงค่ามาตรวัด คือ NOSS มาคำนวณค่าร่องรอยที่ไม่ดีตามช่วงค่าของมาตรวัดที่เกิดร่องรอยที่ไม่ดีและตามข้อกำหนดที่พิจารณา



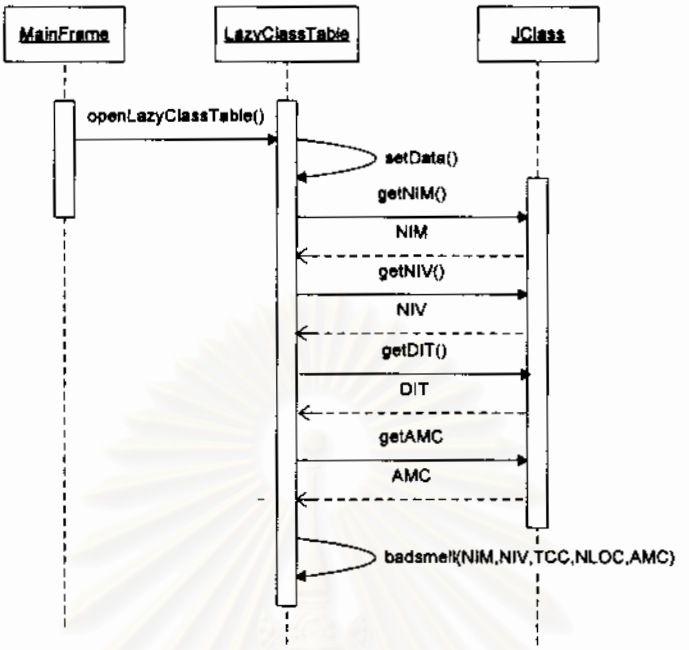
รูปที่ 6.9 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Data Class



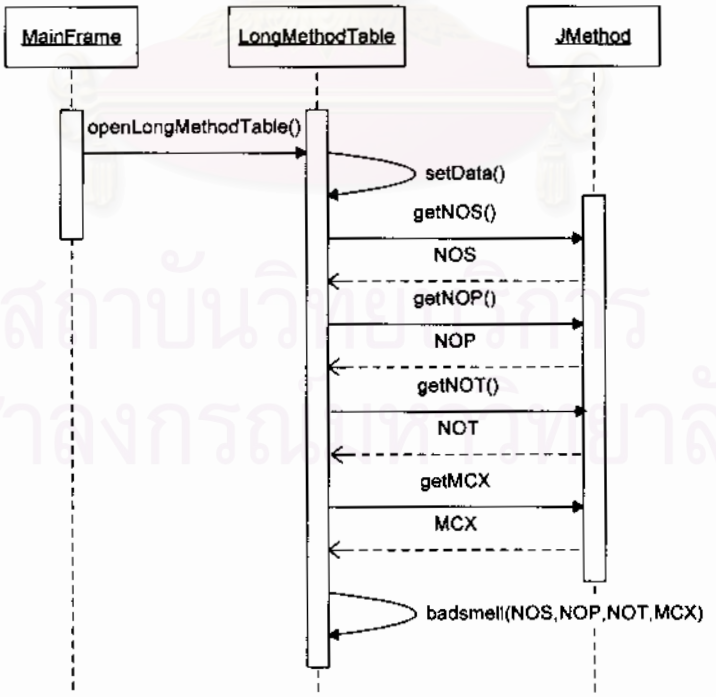
รูปที่ 6.10 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Feature Envy



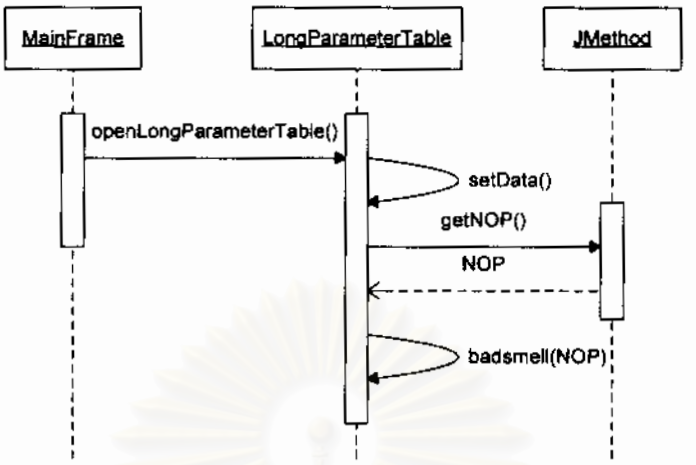
รูปที่ 6.11 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Large Class



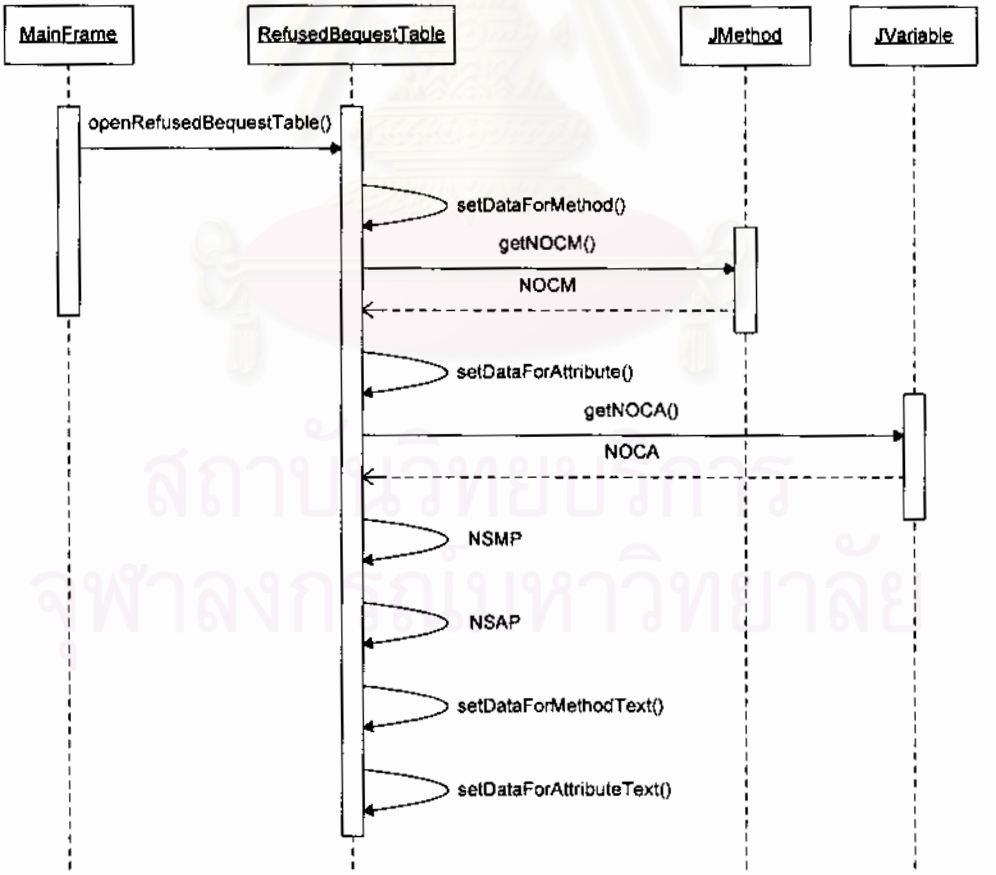
รูปที่ 6.12 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Lazy Class



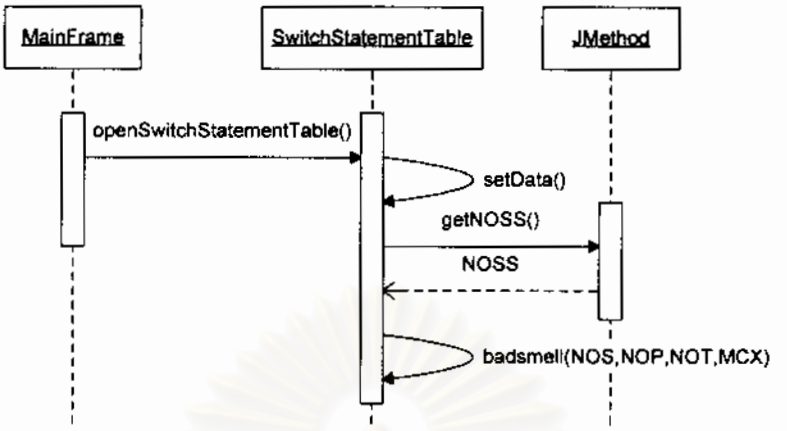
รูปที่ 6.13 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Long Method



รูปที่ 6.14 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Long Parameter List



รูปที่ 6.15 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Refused Bequest



รูปที่ 6.16 แผนภาพแสดงลำดับการทำงานการเรียกดูค่ามาตรวัด และค่าร่องรอยที่ไม่ดีแบบ Switch Statement

6.2 การตรวจสอบค่ามาตรวัดที่ได้จากเครื่องมือ

ผู้วิจัยได้ทำการตรวจสอบความถูกต้องของค่ามาตรวัดของร่องรอยที่ไม่ดี 8 แบบ ที่คำนวณได้จากเครื่องมือสำหรับการคำนวณมาตรวัด และตรวจจับร่องรอยที่ไม่ดีทั้ง 8 แบบ กับค่ามาตรวัดของร่องรอยที่ไม่ดี 8 แบบที่ได้จากวิธีการนับและการคำนวณด้วยมือ ด้วยการสุ่มเลือกคลาสจำนวน 5 คลาสของโปรแกรมชุดทดสอบ 2 โปรแกรม และทำการเปรียบเทียบค่าของมาตรวัดทุกมาตรวัดที่คำนวณได้จากทั้งสองวิธี ผลการเปรียบเทียบพบว่าเครื่องมือที่พัฒนาขึ้นสามารถคำนวณค่ามาตรวัดได้ถูกต้องตรงกับการคำนวณด้วยการนับและการคำนวณด้วยมือ

บทที่ 7

บทสรุปและข้อเสนอแนะ

7.1 บทสรุป

วิทยานิพนธ์นี้ได้มุ่งเน้นหาช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี โดยเริ่มจากทำการหาโมเดลของมาตรวัดในการทำนายร่องรอยที่ไม่ดีทั้ง 8 แบบ คือ Data Class, Feature Envy, Large Class, Lazy Class, Long Method, Long Parameter List, Refused Bequest และ Switch Statement โดยใช้มาตรวัด 30 มาตรวัด ด้วยวิธีการวิเคราะห์ความถดถอยเพื่อหาความสัมพันธ์ระหว่างมาตรวัดกับร่องรอยที่ไม่ดี โดยใช้ชุดข้อมูลสอนจำนวน 10 โปรแกรม และตรวจสอบความถูกต้องของโมเดลการทำนายด้วยชุดข้อมูลทดสอบจำนวน 2 โปรแกรม ผลการทดลองพบว่าสามารถหามาตรวัดที่มีความสัมพันธ์กับค่าร่องรอยที่ไม่ดี 6 แบบคือ Data Class, Large Class, Lazy Class, Long Method, Long Parameter List และ Switch Statement มีจำนวน 21 มาตรวัด และไม่สามารถหามาตรวัดที่มีความสัมพันธ์กับค่าร่องรอยที่ไม่ดีแบบ Feature Envy และ Refused Bequestได้ เนื่องจากหาข้อมูลที่เกิดร่องรอยที่ไม่ดีทั้ง 2 แบบได้จำนวนน้อยไม่พอต่อทำการทดลอง

หลังจากนั้นทำการออกแบบ และคำนวณมาตรวัดเพื่อเก็บข้อมูลในการหาช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี โดยใช้ชุดข้อมูลสอนสำหรับหาช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี และชุดข้อมูลทดสอบสำหรับตรวจสอบความถูกต้องของช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีและช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีจำนวน 2 โปรแกรม ผลการทดลองพบว่ามีเปอร์เซ็นต์ความถูกต้องของทุกมาตรวัดมากกว่าผลรวมของเปอร์เซ็นต์ความผิดพลาด และเปอร์เซ็นต์ที่ไม่สามารถบอกได้ จากนั้นจึงนำมาตรวัดที่ได้มาตรวจสอบวิธีการใช้ช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งาน พบว่าสามารถใช้วิธีการใช้ช่วงของค่ามาตรวัดสำหรับตรวจจับร่องรอยที่ไม่ดีไปใช้งานสำหรับร่องรอยที่ไม่ดีแบบ Data Class, Large Class, Lazy Class, Long Method, Long Parameter List และ Switch Statement เพราะว่ามีเปอร์เซ็นต์ความถูกต้องของวิธีการการนำช่วงค่าของมาตรวัดต่างๆ สำหรับตรวจจับร่องรอยที่ไม่ดีนำไปใช้งานมากกว่าหรือเท่ากับการนำช่วงค่าของมาตรวัดใดมาตรวัดหนึ่ง ของร่องรอยที่ไม่ดีไปตรวจจับร่องรอยที่ไม่ดี

นอกจากนี้ได้ออกแบบมาตรวัดซอฟต์แวร์เชิงวัตถุสำหรับรีแฟคทอริง ในการตรวจจับร่องรอยที่ไม่ดีเพิ่มเติมคือ มาตรวัด NSM, NSOC, NGM, NGOC, NCA, NAOC สำหรับร่องรอยที่ไม่ดีแบบ Data Class และมาตรวัด NOCM, NSMP, NOCA, NSAP สำหรับร่องรอยที่ไม่ดีแบบ Refused Bequest ซึ่งมีรายละเอียดดังบทที่ 4

ผู้ทำวิทยานิพนธ์ได้พัฒนาเครื่องมือเพื่อการคำนวณมาตรวัด และทำนายการเกิดร่องรอยที่ไม่ดีโดยใช้มาตรวัดต่างๆ พร้อมทั้งทำนายตำแหน่งการเกิดร่องรอยที่ไม่ดีแต่ทั้ง 6 แบบ โดยใช้ช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีสำหรับภาษาจาวา และทำนายการเกิดร่องรอยที่ไม่ดีอีก 2 แบบด้วยข้อกำหนดของค่าที่พิจารณา ซึ่งข้อมูลนำเข้าสำหรับเครื่องมือที่พัฒนาขึ้นคือโปรแกรมที่พัฒนาด้วยภาษาจาวา

7.2 ข้อจำกัดและข้อเสนอแนะ

1. ในการหาช่วงของค่าไม่คำนวณมาตรวัดของคลาสที่ เป็นส่วนต่อประสานผู้ใช้ (User Interface)

2. สามารถใช้วิธีการนำช่วงของค่ามาตรวัดหลายมาตรวัด โดยใช้เงื่อนไขแบบ OR กันตรวจจับร่องรอยที่ไม่ดีแบบ Data Class, Large Class, Lazy Class, Long Method และ Long Parameter List

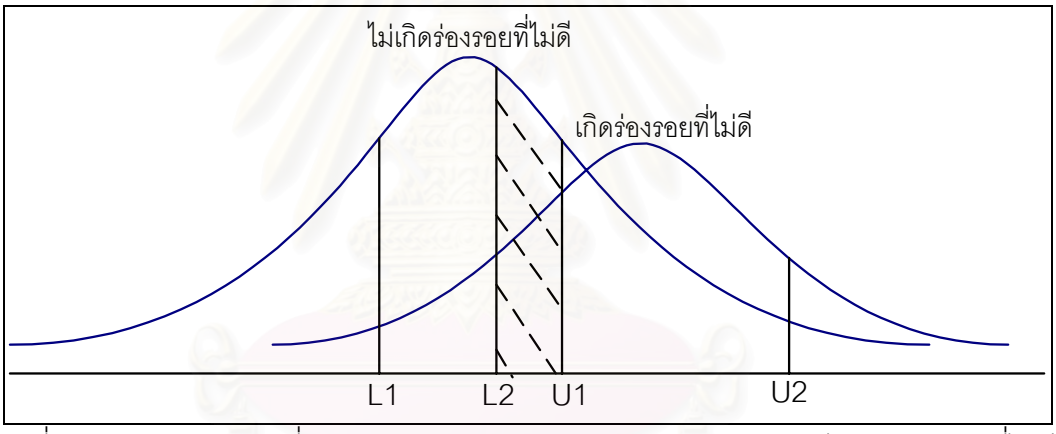
3. ไม่สามารถใช้วิธีการนำช่วงของค่ามาตรวัดหลายมาตรวัด โดยใช้เงื่อนไขแบบ OR กันตรวจจับร่องรอยที่ไม่ดีแบบ Feature Envy และ Refused Bequest

4. การประเมินความสามารถในการออกแบบมาตรวัดในงายวิจัยนี้ได้ ใช้มาตรวัดคุณภาพซอฟต์แวร์คือ LCOM และ CBO ซึ่งมาตรวัด CBO ก่อนและหลังการทำรีแฟคทอริงไม่เปลี่ยนแปลงเพราะ CBO อาจไม่มีผลกระทบต่อการใช้วิธีรีแฟคทอริง ดังนั้นควรใช้มาตรวัดคุณภาพอื่นมาประเมินความสามารถในการออกแบบมาตรวัด

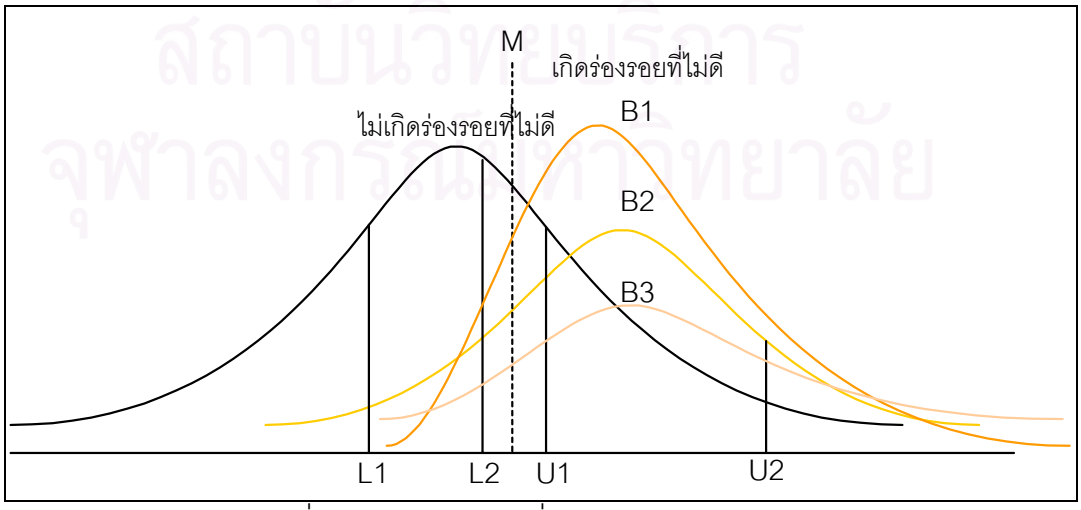
5. ช่วงของค่ามาตรวัดที่เกิดและไม่เกิดร่องรอยที่ไม่ดียังมี ช่วงที่ไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่ ดังนั้นถ้าสามารถทำให้มีช่วงที่ไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีหรือไม่น้อยลงกว่างานวิจัยนี้จะช่วยเพิ่มความสามารถในการตรวจจับร่องรอยที่ไม่ดีได้ เช่น การใช้วิธีหาจุดตัดของกราฟระหว่างช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีและช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี เป็นต้น

6. กำหนดช่วงให้มาตรวัดที่ใช้ในการตรวจจับร่องรอยที่ไม่ดีมี 8 แบบจากร่องรอยที่ไม่ดีทั้งหมด 22 แบบ ดังนั้นควรเพิ่มมาตรวัดในการตรวจจับร่องรอยที่ไม่ดี เพื่อเพิ่มความสามารถในการตรวจจับร่องรอยที่ไม่ดีได้หลายแบบมากขึ้น

7. ในการหาช่วงค่าของมาตรวัด ถ้ามีพื้นที่ใต้กราฟของช่วงของค่า มาตรวัดทั้งเกิดและไม่เกิดร่องรอยที่ไม่ดีที่ซ้อนทับกันดังรูปที่ 7.1 คือ (L2, U1) ค่ามาตรวัดที่อยู่ในช่วงนี้จะไม่สามารถบอกได้ว่าเกิดร่องรอยที่ไม่ดีและไม่เกิดร่องรอยที่ไม่ดี มีวิธีแก้ปัญหา คือ หาค่ากึ่งกลางระหว่าง (L2, U1) ดังรูปที่ 7.2 ค่ากึ่งกลางเรียกว่า M ซึ่งหาได้โดย (L2+U1)/2 ดังนั้นช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, M) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (M, U2) แต่วิธีนี้มีความน่าเชื่อถือน้อยดังตัวอย่างรูปที่ 7.2 กราฟเส้น B1 ช่วงระหว่าง L2 และ M จะเห็นว่าโอกาสที่เกิดร่องรอยที่ไม่ดีมากกว่าไม่เกิดร่องรอยที่ไม่ดีทั้งที่อยู่ในช่วงการไม่เกิดร่องรอยที่ไม่ดี และอีกตัวอย่างคือกราฟเส้น B3 ของรูปที่ 7.2 ช่วงระหว่าง M และ U1 จะเห็นว่าโอกาสที่ไม่เกิดร่องรอยที่ไม่ดีมากกว่าเกิดร่องรอยที่ไม่ดี ทั้งที่อยู่ในช่วงการเกิดร่องรอยที่ไม่ดี



รูปที่ 7.1 กราฟแสดงส่วนที่ซ้อนทับกันระหว่างช่วงของค่ามาตรวัดเกิดและไม่เกิดร่องรอยที่ไม่ดี



รูปที่ 7.2 กราฟแสดงค่ากึ่งกลางระหว่าง (L2, U1)

จากปัญหาดังกล่าวจึงใช้อีกวิธีคือวิธีเปรียบเทียบพื้นที่ใต้กราฟซึ่งพิจารณาดังนี้

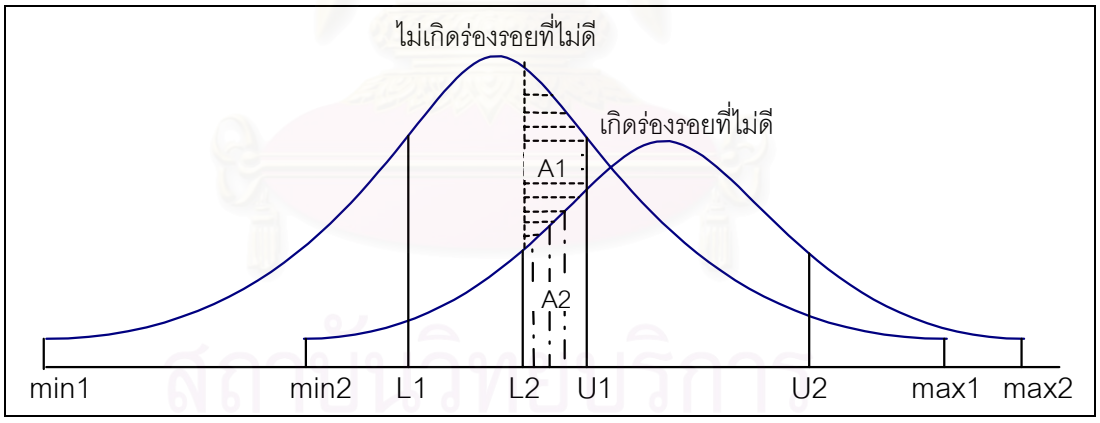
กรณีที่ 1 จากรูปที่ 7.3 ถ้าพื้นที่ใต้กราฟ A1 < A2 และค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดี ควรใช้ช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, L2) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L2, U2)

กรณีที่ 2 จากรูปที่ 7.3 ถ้าพื้นที่ใต้กราฟ A1 > A2 และค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดี ควรใช้ช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, U1) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (U1, U2)

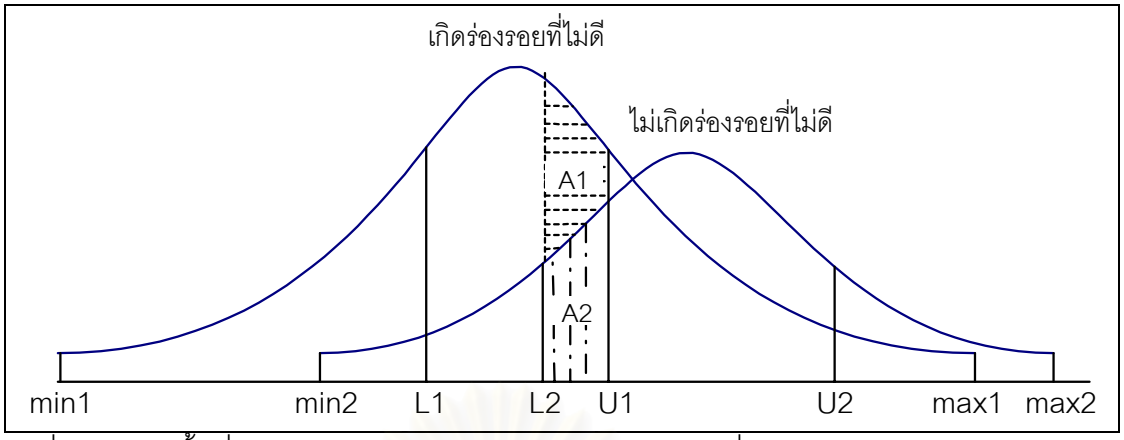
กรณีที่ 3 จากรูปที่ 7.4 ถ้าพื้นที่ใต้กราฟ A1 < A2 และค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดี ควรใช้ช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L1, L2) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L2, U2)

กรณีที่ 4 จากรูปที่ 7.4 ถ้าพื้นที่ใต้กราฟ A1 > A2 และค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดี ควรใช้ช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L1, U1) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (U1, U2)

พื้นที่ใต้กราฟ A1 และ A2 หาได้ในตาราง Z โดยหาค่า Z ที่ได้จากสูตร $\frac{\bar{x}-\mu}{\sigma/\sqrt{n}}$ จากนั้นเลือกค่าพื้นที่ใต้กราฟจากค่า Z ที่คำนวณได้



รูปที่ 7.3 แสดงพื้นที่ใต้กราฟโดยค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดี



รูปที่ 7.4 แสดงพื้นที่ใต้กราฟโดยค่ามาตรวัดของการเกิดร่องรอยที่ไม่ดีน้อยกว่าค่ามาตรวัดของการไม่เกิดร่องรอยที่ไม่ดี

7. ในขั้นตอนการประเมินมาตรวัด พิจารณากรณีที่มีส่วนที่ซ้อนทับกันระหว่างช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดี และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดี

กรณีที่ 1 จากรูปที่ 7.3 ถ้าช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, L2) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L2, U2) แล้วเปลี่ยนเป็นช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (min1, L2) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L2, max2)

กรณีที่ 2 จากรูปที่ 7.3 ถ้าช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L1, U1) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (U1, U2) แล้วเปลี่ยนเป็นช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (min1, U1) และช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (U1, max2)

กรณีที่ 3 จากรูปที่ 7.4 ถ้าช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L1, L2) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L2, U2) แล้วเปลี่ยนเป็นช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (min1, L2) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (L2, max2)

กรณีที่ 4 จากรูปที่ 7.4 ถ้าช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (L1, U1) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (U1, U2) แล้วเปลี่ยนเป็นช่วงของค่ามาตรวัดที่เกิดร่องรอยที่ไม่ดีคือ (min1, U1) และช่วงของค่ามาตรวัดที่ไม่เกิดร่องรอยที่ไม่ดีคือ (U1, max2)

รายการอ้างอิง

1. Martin Fowler. Refactoring: Improving the Design of Existing Code. Addison-Wesley,1999.
2. Yoshio Kataoka, Takeo Imai, Hiroki Andou and Tetsuji /fukaya. Quantitative Evaluation of Maintainability Enhancement by Refactoring. Proceedings of the International Conference on Software Maintenance (ICSM'02). 2002.
3. M.Genero, M. Piattini and C. Calero. Early Measures for UML class diagrams. L'Object, 6, 4(2000)
4. Mika Mantyla, Jari Vanhanen and Casper Lessenius. A Taxonomy and an Initial Empirical Study of Bad Smells in Code. Proceeding of the International Conference on Software Maintenance (ICSM'03) IEEE, 2003.
5. Thisana Pienlert and Pornsiri Muenchaisri. Bad-Smell Detection Using Object-Oriented Software Metrics. International Conference Computer Science, Software Engineering, Information Technology, e-Business and Application (CSITeA '04), December 27-29, 2004.
6. Mark Lorenz and Jeff Kidd. Object-Oriented Software Metrics. Prentice-Hall Inc., 1994.
7. Radu Marinescu. Detecting Design Flaws via Metrics in Object-Oriented System. Proceedings of the 39th International Conference and Exhibition on Technology of Object-Oriented Languages and System (TOOLS39), July 29 – August 03, 2001.
8. S. R. Chidamber and C.F. Kemerer. A Mertic Suit for Object-Oriented Design. IEEE Transactions on Software Engineering, (June 1994):476-493
9. กัลยา วินิชย์บัญชา. การวิเคราะห์ตัวแปรหลายตัวด้วย SPSS for Windows. กรุงเทพมหานคร: โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2544.
10. กัลยา วินิชย์บัญชา. การวิเคราะห์สถิติ. สถิติสำหรับการบริหารและวิจัย. กรุงเทพมหานคร: โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.
11. กัลยา วินิชย์บัญชา. การวิเคราะห์สถิติขั้นสูงด้วย ด้วย SPSS for Windows. กรุงเทพมหานคร : โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2546.

12. Frank Simon, Frank Steinbruckner, and Claus Lewerentz, Metrics Base Refactoring. Proceeding of 5th European Conference on Software Maintenance and Reengineering, IEEE Computer Society Press, (2001): 30-38.

13. Tom Mens, Member, IEEE and Tom Tourwe. A Survey of Software Refactoring. IEEE Transactions on Software Engineering, 30, 2 (Feb 2004).

14. Julio Sanchez and Maria P. Canton. Java Programming for Engineer. CRC PRESS, 2002.

ภาคผนวก

ภาคผนวก ก

ผลการคำนวณค่าร่องรอยที่ไม่ดีและค่ามาตรวัดของร่องรอยที่ไม่ดี 8 แบบ

ชุดข้อมูลสอน

1. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Data Class

สำหรับคลาส

Program No.	Class	Bad Smell	NOPA	NOAM
1. JCALC-alpha0-1	Jcalc	0	0	0
	JCalcBuffer	0	0	0
	JCalcList	0	1	0
	JCalcMenuBar	0	0	1
	JCalcStandardFrame	0	0	1

สำหรับเมทธอด

Program No.	Method	Bad Smell	NSM	NSOC	NGM	AGOC	NSM+ NSOC
1. JCALC-alpha0-1	Display [JCalcStandardFrame]	1	0	0	0	0	0
	actionPerformed [JCalcMenuBar]	1	0	0	0	0	0
	add [JCalcBuffer]	1	0	0	0	0	0
	add [JCalcBuffer]	1	0	0	0	0	0
	add0 [JCalcBuffer]	1	0	0	0	0	0
	add1 [JCalcBuffer]	1	0	0	0	0	0
	add2 [JCalcBuffer]	1	0	0	0	0	0
	add3 [JCalcBuffer]	1	0	0	0	0	0
	add4 [JCalcBuffer]	1	0	0	0	0	0
	add5 [JCalcBuffer]	1	0	0	0	0	0
	add6 [JCalcBuffer]	1	0	0	0	0	0
	add7 [JCalcBuffer]	1	0	0	0	0	0

สำหรับแอทริบิวต์

Program No.	attribute of Class	Bad Smell	NCA	NAOC	NCA+NAOC
1. JCALC-alpha0-1	OpList [JCalcList]	1	12	0	12
	buffer [JCalcBuffer]	1	13	0	13
	buffer [JCalcList]	1	14	0	14

2. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Feature Envy

สำหรับเมทธอด

project	method	bad	Total	toalself-totalother
2	init [Whiteboard]		4	2
	main [Whiteboard]		2	-2
	makeConnection [ServerConnection]		1	9
	makeCommandString [LineTool]		1	9
	readln [ServerConnection]		3	9
	write [ServerConnection]		2	-1

สำหรับแอทริบิวต์

project	Attribute a of Class c (a[c])	NCRA (Config Data)	NCRA (Server Connection)	NCRA (TestData)	NCRA (Whiteboard)
2	DEFAULT_HOST_NAME [ConfigData]	0	0	0	1
	DEFAULT_PORT_NUMBER [ConfigData]	0	0	0	1
	FRAME_INITIAL_HEIGHT [ConfigData]	0	0	0	2
	FRAME_INITIAL_WIDTH [ConfigData]	0	0	0	2
	LOCALMODE [TestData]	0	3	0	0

3. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Large Class

Program No.	Class	Bad Smell	NIM	NIV	TCC	Number of Statements	Average Method Complexity
1. JCALC-alpha0-1	JCalc	0	0	0	2	5	0
	JCalcBuffer	0	19	0	0.035088	70	4.021053
	JCalcList	0	11	0	0.145455	96	13.472727
	JCalcMenuBar	0	5	1	0.1	36	7.5
	JCalcStandardFrame	0	22	0	0	58	1.4318175
2. Whiteboard	BoxTool		9	5	0.071429	51	5.9555554
	ColorPaletteTool	0	10	4	0.166667	29	3.86
	Command	0	3	7	2	15	10.633334
	ConfigData	0	0	0	2	0	0
	FontSelector	0	6	4	0	25	3.9833336
	FontSelectorDialog	0	2	8	2	36	14.1
	Helper	0	0	0	0	11	0
	LineTool	0	10	4	0.055556	54	9.35
	LineWidthTool	0	8	3	0.190476	33	6.8999996
	MainCanvas	1	6	6	0.3	24	7.25
	MainFrame	0	2	0	2	3	1.3
	OvalTool	1	9	5	0.071429	0.82142854	5.9555554
	PaintMonitor	0	2	1	2	9	3.7
	PencilTool	0	6	4	0.1	20	4.2166667
	ServerConnection	0	3	7	0.333333	28	10.3
	TestData	0	0	0	2	0	0
	TextTool	0	5	3	0	27	5.4
	ToolCanvas	0	13	4	0.015152		1.9076922
	ToolPanel	0	1	1	2	10	0.8
	Tools	0	4	9	0.333333	31	1.7750001
	Whiteboard	1	6	8	0	30	4.533333

4. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Lazy Class

Program No.	Class	Bad Smell	NIM	NIV	DIT	AMC
1. JCALC-alpha0-1	JCalc	1	0	0	1	0
	JCalcBuffer	0	19	0	0	4.021053
	JCalcList	0	11	0	1	13.47273
	JCalcMenuBar	1	5	1	1	7.5
	JCalcStandardFrame	0	22	0	1	1.431818
2. Whiteboard	BoxTool	0	9	5	2	5.955555
	ColorPaletteTool	0	10	4	2	3.86
	Command	0	3	7	0	10.63333
	ConfigData	1	0	0	0	0
	FontSelector	0	6	4	2	3.983334
	FontSelectorDialog	1	2	8	1	14.1
	Helper	1	0	0	0	0
	LineTool	1	10	4	2	9.35
	LineWidthTool	1	8	3	2	6.9
	MainCanvas	1	6	6	1	7.25
	MainFrame	1	2	0	1	1.3
	OvalTool	1	9	5	2	5.955555
	PaintMonitor	1	2	1	1	3.7
	PencilTool	1	6	4	2	4.216667
	ServerConnection	1	3	7	0	10.3
	TestData	1	0	0	0	0
	TextTool	1	5	3	2	5.4
	ToolCanvas	0	13	4	1	1.907692
	ToolPanel	1	1	1	1	0.8

5. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Long Method

Program No.	Class	Method	Bad Smell	NOS	NOP	NOT	MCX
1. JCALC-alpha0-1	JCalc	main [JCalc]	0	4	1	4	5.3
	JCalcBuffer	add	0	4	1	0	5.3
		clear	0	4	0	1	3.5
		add0	0	1	0	0	3
		add1	0	1	0	0	3
		add2	0	1	0	0	3
		add3	0	1	0	0	3
		add4	0	1	0	0	3
		add5	0	1	0	0	3
		add6	0	1	0	0	3
		add7	0	1	0	0	3
		add8	0	1	0	0	3
		add9	0	1	0	0	3
		addDot	0	6	0	4	10.5
		del	0	5	0	1	7.5
		print	0	1	0	0	2
		toString	0	1	0	0	0
		isANumber	1	13	0	2	6
		main	1	18	1	1	2.8
	JCalcList	addToBuffer	0	4	1	0	2.3
		addToBuffer	0	4	1	0	2.3
		addBufferToList	0	2	0	0	0
		addStringToList	0	1	1	0	0.3
		addPlus	0	3	0	0	6
		addMinus	0	3	0	0	6
		addMultiply	0	3	0	0	6

6. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Long Parameter List

Program No.	Class	Method	Bad Smell	NOP
1. JCALC-alpha0-1	JCalc	main	0	1
	JCalcBuffer	add	0	1
		clear	0	0
		add0	0	0
		add1	0	0
		add2	0	0
		add3	0	0
		add4	0	0
		add5	0	0
		add6	0	0
		add7	0	0
		add8	0	0
		add9	0	0
		addDot	0	0
		del	0	0
		print	0	0
		toString	0	0
		isANumber	0	0
		main	0	1
	JCalcList	addToBuffer	0	1
		addToBuffer	0	1
		addBufferToList	0	0
		addStringToList	0	1
		addPlus	0	0
		addMinus	0	0
		addMultiply	0	0

7. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Refused Bequest

สำหรับเมทธอด

Program No.	BadSmell	Super Class	Method m	NOCM
1. JCALC-alpha0-1		JCalcBuffer	main [JCalcList]	0
4. jfreechart-0.9.21		ChartEntity	equals [CategoryItemEntity]	0
		ChartEntity	equals [LegendItemEntity]	0
		ChartEntity	clone [LegendItemEntity]	0
		ChartEntity	equals [XYItemEntity]	0
5. JGroups-2.2.7.src		ConnectionTable	getConnection [ConnectionTable1_4]	1
		ConnectionTable	stop [ConnectionTable1_4]	1
		ConnectionTable	run [ConnectionTable1_4]	1
		ConnectionTable	createServerSocket [ConnectionTable1_4]	1
		MessageDispatcher	castMessage [RpcDispatcher]	0
		MessageDispatcher	sendMessage [RpcDispatcher]	0
		MessageDispatcher	handle [RpcDispatcher]	1
		View	clone [MergeView]	1
		View	toString [MergeView]	1
		View	writeExternal [MergeView]	1
		View	readExternal [MergeView]	1

สำหรับแอทริบิวต์

Program No.	Super Class	Attribute a	BadSmell	NOCA
1. JCALC-alpha0-1	JCalcBuffer	buffer [JCalcList]		1
	JCalcBuffer	debug [JCalcList]		1

8. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Switch Statment

Program No.	Class		Bad Smell	NOSS
1. JCALC-alpha0-1	JCalc	main [JCalc]	0	0
	JCalcBuffer	add	0	0
		clear	0	0
		add0	0	0
		add1	0	0
		add2	0	0
		add3	0	0
		add4	0	0
		add5	0	0
		add6	0	0
		add7	0	0
		add8	0	0
		add9	0	0
		addDot	0	0
		del	0	0
		print	0	0
		toString	0	0
		isANumber	0	0
		main	0	0
	JCalcList	addToBuffer	0	0
		addToBuffer	0	0
		addBufferToList	0	0
		addStringToList	0	0
		addPlus	0	0
		addMinus	0	0
		addMultiply	0	0

ชุดข้อมูลทดสอบ

1. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Data Class

สำหรับคลาส

Program No.	Class	Bad Smell	NOPA	NOAM
1 jGraph	AbstractCellView	1	0	14
	AttributeMap	0	0	0
	BasicGraphDropTargetListener	0	0	1
	BasicGraphTransferable	0	0	0
	BasicGraphUI	0	0	0

สำหรับเมทธอด

Program No.	Method	Bad Smell	NSM	NSOC	NGM	AGOC	NSM+ NSOC
1 jGraph	acceptsSource [DefaultGraphModel]	1	0	0	0	0	0
	acceptsTarget [DefaultGraphModel]	1	0	0	0	0	0
	actionPerformed [BasicGraphDropTargetListener]	1	0	0	0	0	0
	actionPerformed [DefaultGraphCellEditor]	1	0	0	0	0	0
	addCellEditorListener [DefaultGraphCellEditor]	1	0	0	0	0	0
	addChanged [GraphLayoutCache]	1	0	0	0	0	0
	addConnections [ConnectionSet]	1	0	0	0	0	0
	addEdge [DefaultPort]	1	0	0	0	0	0
	addEntry [ParentMap]	1	0	0	0	0	0
	addGraphModelListener [DefaultGraphModel]	1	0	0	0	0	0
	addGraphSelectionListener [JGraph]	1	0	0	0	0	0
	addPoint [EdgeView]	1	0	0	0	0	0

สำหรับแอทริบิวต์

Program No.	attribute of Class	Bad Smell	NCA	NAOC	NCA+NAOC
1. JCALC-alpha0-1	OpList [JCalcList]	1	12	0	12
	buffer [JCalcBuffer]	1	13	0	13
	buffer [JCalcList]	1	14	0	14

2. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Large Class

Program No.	Class	Bad Smell	NIM	NIV	TCC	NLOC	AMC
1 jGraph	AbstractCellView	1	25	6	0.043	88	9.272
	AttributeMap	1	22	0	0	87	8.495
	BasicGraphDropTargetListener	1	15	0	0	41	4.426
	BasicGraphTransferable	0	0	0	1	0	0
	BasicGraphUI	0	0	1	0.009	0	0
	BasicMarqueeHandler	1	43	4	0.667	73	2.016
	Bezier	0	4	1	0.030	16	24.15
	ConnectionSet	1	13	2	0	30	3.384
	DefaultCellViewFactory	1	6	0	0.095	15	1.683
	DefaultEdge	1	10	1	0.267	17	2.63
	DefaultGraphCell	1	10	11	0.058	26	3.59
	DefaultGraphCellEditor	1	26	5	0.02	76	4.292
	DefaultGraphModel	1	43	1	0	165	8.193
	DefaultGraphSelectionModel	0	0	2	0.036	0	0
	DefaultPort	1	11	0	0	10	1.327
	DefaultRealEditor	1	9	1	0	10	1.055
	EdgeRenderer	0	0	12	0.046	0	0
	EdgeView	1	26	0	0.01	81	4.646

3. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Lazy Class

Program No.	Class	Bad Smell	NIM	NIV
1 jGraph	AbstractCellView	0	25	6
	AttributeMap	1	22	0
	BasicGraphDropTargetListener	1	15	0
	BasicGraphTransferable	1	0	0
	BasicGraphUI	1	43	1
	BasicMarqueeHandler	1	4	4
	Bezier	1	13	1
	ConnectionSet	1	6	2
	DefaultCellViewFactory	1	10	0
	DefaultEdge	1	10	2
	DefaultGraphCell	0	26	1
	DefaultGraphCellEditor	0	43	11
	DefaultGraphModel	1	0	5
	DefaultGraphSelectionModel	1	11	1
	DefaultPort	1	9	2
	DefaultRealEditor	0	26	0
	EdgeRenderer	0	43	1
	EdgeView	1	0	12
	GraphConstants	1	11	0
	DefaultRealEditor	1	9	7
	EdgeRenderer	1	0	17
	EdgeView	0	26	6
	GraphConstants	1	0	0
	GraphContext	0	14	0
	GraphLayoutCache	0	54	0

4. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Long Method

Program No.	Class	Method	Bad Smell	NOS	NOP	NOT	MCX
1 jGraph	AbstractCellView	getCell	0	1	0	0	0
		refresh	0	10	3	5	19.4
		update	0	3	0	0	0.5
		updateAllAttributes	0	4	0	0	4
		createLocalAttributeMap	0	1	0	0	0
		childUpdated	0	3	0	0	2.5
		getParentView	0	1	0	0	0
		getChildViews	0	2	0	1	0.5
		getDescendantViews	0	8	1	7	9.8
		removeFromParent	0	2	0	1	6.5
		isLeaf	0	1	0	0	0
		getAttributes	0	1	0	0	0
		getAllAttributes	0	1	0	0	0
		setAttributes	0	2	1	1	0.8
		getBounds	0	5	0	0	3
		getBounds	0	9	1	3	25.3
		setBounds	0	7	1	9	52.3
		updateGroupBounds	0	8	0	5	17.5
		includeInGroupBounds	1	11	1	6	53.3
	AttributeMap	createPoint	0	1	2	0	0.6
		createRect	0	1	0	0	0

5. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Long Parameter List

Program No.	Class	Method	Bad Smell	NOP
1 jGraph	AbstractCellView	getCell	0	0
		refresh	0	3
		update	0	0
		updateAllAttributes	0	0
		createLocalAttributeMap	0	0
		childUpdated	0	0
		getParentView	0	0
		getChildViews	0	0
		getDescendantViews	0	1
		removeFromParent	0	0
		isLeaf	0	0
		getAttributes	0	0
		getAllAttributes	0	0
		setAttributes	0	1
		getBounds	0	0
		getBounds	0	1
		setBounds	0	1
		updateGroupBounds	0	0
		includeInGroupBounds	0	1
	AttributeMap	createPoint	0	createPoint
		createPoint	0	createPoint
		createPoint	0	createPoint
		createRect	0	createRect
		createRect	0	createRect
		createRect	0	createRect
		createRect	0	createRect

6. ข้อมูลตัวอย่างระหว่างค่าร่องรอยที่ไม่ดี และค่ามาตรวัดของร่องรอยที่ไม่ดีแบบ Switch Statement

Program No.	Class	Method	Bad Smell	NCA
1 jGraph	AbstractCellView	getCell	0	0
		refresh	0	0
		update	0	0
		updateAllAttributes	0	0
		createLocalAttributeMap	0	0
		childUpdated	0	0
		getParentView	0	0
		getChildViews	0	0
		getDescendantViews	0	0
		removeFromParent	0	0
		isLeaf	0	0
		getAttributes	0	0
		getAllAttributes	0	0
		setAttributes	0	0
		getBounds	0	0
		getBounds	0	0
		setBounds	0	0
		updateGroupBounds	0	0
		includeInGroupBounds	0	0
	AttributeMap	createPoint	0	0
		createPoint	0	0
		createPoint	0	0
		createRect	0	0
		createRect	0	0
		createRect	0	0

ภาคผนวก ข

ตารางแจกแจงความถี่ของมาตรวัดที่เกิดและไม่เกิดร่องรอยที่ไม่ดี

1. ร่องรอยที่ไม่ดีแบบ Data Class

   มาตรวัด NOPA

NOPA	BadSmell		Total
	Not Bad	Bad	
0	528	97	625
1	29	2	31
2	27	8	35
3	18	3	21
4	11	1	12
5	2	3	5
6	0	3	3
7	0	4	4
8	0	2	2
9	0	3	3
10	0	2	2
11	0	1	1
12	0	1	1
13	0	3	3
14	0	1	1
15	0	1	1
17	0	1	1
22	0	1	1
…	…	…	…
67	0	1	1
Total	615	142	757

มาตรวัด NOAM

NOAM	BadSmell		Total
	Not Bad	Bad	
0	292	12	304
1	117	1	118
2	90	1	91
3	53	1	54
4	46	3	49
5	17		17
6	0	25	25
7	0	14	14
8	0	14	14
9	0	7	7
10	0	7	7
11	0	13	13
12	0	3	3
13	0	5	5
14	0	2	2
15	0	2	2
16	0	3	3
17	0	1	1
20	0	2	2
21	0	1	1
22	0	2	2
25	0	1	1
26	0	1	1
27	0	4	4
…	…	…	…
771	0	1	1
Total	615	142	757

มาตรวัด NSOC

NSOC	BadSmell		Total
	Not Bad	Bad	
0	0	6112	6112
1	0	273	273
2	0	3	3
5	1	0	1
6	5	0	5
8	1	0	1
Total	7	6388	6395

มาตรวัด NSM+NSOC

NSM+NSOC	BadSmell		Total
	Not Bad	Bad	
0	0	5723	5723
1	0	582	582
2	0	31	31
3	0	6	6
4	0	6	6
6	1	21	22
7	0	7	7
8	2		2
9	0	1	1
10	2	1	3
12	1	4	5
14	1	2	3
18	0	1	1
24	0	1	1
36	0	1	1
85	0	1	1
Total	7	6388	6395

มาตรวัด NCA

NCA	BadSmell		Total
	Not Bad	Bad	
0	1651	9	1660
1	43	0	43
2	80	0	80
3	56	0	56
4	48	0	48
5	40	0	40
6	0	34	34
7	0	84	84
8	0	14	14
9	0	7	7
10	0	30	30
11	0	2	2
12	0	35	35
13	0	1	1
14	0	7	7
15	0	10	10
16	0	4	4
17	0	1	1
18	0	18	18
20	0	6	6
21	0	3	3
24	0	17	17
25	0	4	4
27	0	3	3
…	…	…	…
Total	1918	412	2330

มาตรวัด NAOC

NAOC	BadSmell		Total
	Not Bad	Bad	
0	1891	395	2286
1	1	0	1
3	25	0	25
5	1	0	1
6	0	8	8
7	0	1	1
12	0	2	2
18	0	4	4
30	0	1	1
36	0	1	1
Total	1918	412	2330

2. ร่องรอยที่ไม่ดีแบบ Large Class

มาตรวัด NIM

NIM	BadSmell		Total
	Not Bad	Bad	
0	76	0	76
1	47	0	47
2	80	0	80
3	66	0	66
4	79	0	79
5	67	0	67
6	39	7	46
7	24	4	28
…	…	…	…
350	1		1
Total	652	108	760

มาตรวัด NIV

NIV	BadSmell		Total
	Not Bad	Bad	
0	243	0	243
1	142	0	142
2	114	0	114
3	54	0	54
4	46	0	46
5	24	5	29
6	10	20	30
7	4	11	15
8	4	13	17
9	3	5	8
10	1	12	13
11	0	3	3
12	1	4	5
13	1	6	7
14	1	5	6
15	0	3	3
16	0	1	1
17	1	2	3
18	1	1	2
19	1	2	3
20	0	2	2
21	0	1	1
22	0	3	3
23	0	2	2
…	…	…	…
75	1		1
Total	652	108	760

มาตรวัด TCC

TCC	BadSmell		Total
	Not Bad	Bad	
0	267	30	297
0.00285	1	0	1
0.00391	0	1	1
0.00584	0	1	1
0.00699	0	1	1
0.00925	0	1	1
0.00932	0	1	1
0.00944	1	0	1
0.001332	0	1	1
0.001357	0	1	1
0.001983	0	1	1
0.002115	0	1	1
0.002903	0	1	1
0.005848	0	1	1
0.006536	2	0	2
0.006536	0	1	1
0.006897	0	1	1
0.007905	1	0	1
0.008602	1	1	2
0.009524	0	1	1
0.009524	0	1	1
0.011853	1		1
0.013228	1	1	2
0.015146	1	0	1
…	…	…	…
2	154	0	154
Total	654	109	763

มาตรวัด NLOC

NLOC	BadSmell		Total
	Not Bad	Bad	
0	119	22	141
0.82	0	1	1
1	25	1	26
2	25	0	25
3	47	1	48
4	10	0	10
5	21	0	21
6	25	0	25
7	18	0	18
8	12	1	13
9	20	0	20
10	16	0	16
11	17	0	17
12	22	0	22
13	13	1	14
14	10	1	11
15	15	2	17
16	12	1	13
17	5	1	6
18	10	1	11
19	10	0	10
20	15	2	17
21	11	0	11
22	9	0	9
…	…	…	…
4056	1	0	1
Total	654	108	762

มาตรวัด AMC

AMC	BadSmell		Total
	Not Bad	Bad	
.00	94	23	117
.04	1	0	1
.10	3	0	3
.11	1	0	1
.15	11	0	11
.17	1	0	1
.20	1	1	2
.21	1	0	1
.24	1	0	1
.27	1	0	1
.28	0	1	1
.30	22	1	23
.32	1	0	1
.325	1	0	1
.33	1	0	1
.34	1	0	1
.36	1	0	1
.38	1	0	1
.40	1	0	1
.402	2	0	2
.403	1	0	1
.42	1	0	1
.43	1	0	1
.433	2	0	2
…	…	…	…
128	1	0	1
Total	654	109	763

3. ร่องรอยที่ไม่ดีแบบ Lazy Class

มาตรวัด NIM

NIM	BadSmell		Total
	Not Bad	Bad	
0	0	76	76
1	0	47	47
2	0	80	80
3	1	65	66
4	0	79	79
5	0	67	67
6	1	45	46
7	0	28	28
8	0	31	31
9	1	22	23
10	1	23	24
11	3	12	15
12	1	16	17
13	1	11	12
14	3	16	19
15	4	8	12
16	3	11	14
17	3	6	9
18	1	7	8
19	4	4	8
22	1	2	3
23	5	2	7
…	…	…	…
350	0	1	1
Total	73	690	763

มาตรวัด NIV

NIV	BadSmell		Total
	Not Bad	Bad	
0	3	240	243
1	0	142	142
2	13	103	116
3	0	54	54
4	3	43	46
5	1	28	29
6	0	30	30
7	1	14	15
8	0	17	17
9	0	8	8
10	12	1	13
11	3	0	3
12	3	2	5
13	6	1	7
14	5	1	6
15	3	0	3
16	0	1	1
17	2	1	3
18	1	1	2
19	2	1	3
20	2	0	2
21	1	0	1
22	3	1	4
…	…	…	…
75	0	1	1
Total	73	690	763

4. ร่องรอยที่ไม่ดีแบบ Long Method

มาตรวัด NOS

NOS	BadSmell		Total
	Not Bad	Bad	
0	18	0	18
1	1591	10	1601
2	134	1	135
3	151	4	155
4	124	1	125
5	58	2	60
6	45	4	49
7	47	0	47
8	28	4	32
9	24	4	28
10	9	6	15
11	0	17	17
12	0	14	14
13	0	15	15
14	0	12	12
15	0	6	6
16	0	5	5
17	0	5	5
18	0	10	10
19	0	5	5
20	0	7	7
21	0	7	7
…	…	…	…
245	0	1	1
Total	2229	186	2415

มาตรวัด NOP

NOP	BadSmell		Total
	Not Bad	Bad	
0	987	15	1002
1	897	50	947
2	185	23	208
3	89	22	111
4	39	13	52
5	32	7	39
6	0	15	15
7	0	12	12
8	0	24	24
9	0	2	2
10	0	2	2
11	0	1	1
Total	2229	186	2415

มาตรวัด NOT

NOT	BadSmell		Total
	Not Bad	Bad	
0	1750	25	1775
1	220	24	244
2	110	12	122
3	69	13	82
4	29	16	45
5	22	16	38
6	11	17	28
…	…	…	…
52.00	0	1	1
Total	2229	186	2415

มาตรวัด MCX

MCX	BadSmell		Total
	Not Bad	Bad	
.00	564	0	564
.30	409	3	412
.50	14	0	14
.60	64	0	64
.80	49	0	49
.90	33	0	33
1.00	15	0	15
1.20	19	0	19
1.30	13	0	13
1.50	14	0	14
1.60	7	0	7
1.80	5	0	5
1.80001	0	3	3
2.00	21	0	21
2.10	3	0	3
2.10001	0	1	1
2.30	39	0	39
2.40	1	6	7
2.50	66	0	66
2.60	2	0	2
2.70	1	0	1
2.80	18	1	19
2.90	2	0	2
…	…	…	…
555.90	0	1	1
Total	2159	168	2327

5. ร่องรอยที่ไม่ดีแบบ Long Parameter List

มาตรวัด NOP

NOP	BadSmell		Total
	Not Bad	Bad	
0	1002	0	1002
1	947	0	947
2	208	0	208
3	111	0	111
4	52	0	52
5	39	0	39
6	0	15	15
7	0	12	12
8	0	24	24
9	0	2	2
10	0	2	2
11	0	1	1
Total	2359	56	2415

6. ร่องรอยที่ไม่ดีแบบ Switch Statement

มาตรวัด NOSS

NOP	BadSmell		Total
	Not Bad	Bad	
0	2396	0	2396
1	0	18	18
2	0	1	1
Total	2396	19	2415

ภาคผนวก ค

อภิธานคำศัพท์มาตรวัด

ชื่อมาตรวัด	คำย่อ	คำอธิบาย
Number of Public Attribute	NOPA	จำนวนแอทริบิวต์ประเภท Public
Number of Accessor Method	NOAM	จำนวนเอคเซสเซอร์เมทธอด
Number of Called Methods	NCM	จำนวนการเรียกใช้เมทธอดของคลาส C รวมถึงคลาส C สืบทอดคุณสมบัติมาทั้งหมดภายในเมทธอด m
Number of Called Attributes	NCDA	จำนวนการเรียกใช้คุณสมบัติของคลาส C รวมถึงคลาส C สืบทอดคุณสมบัติมาทั้งหมดภายในเมทธอด m
Number of Caller Attributes	NCRA	จำนวนการเรียกใช้คุณลักษณะ x ภายในคลาส และคลาสที่สืบทอดคุณสมบัติมาทั้งหมด
Number of Instance Methods in a class	NIM	จำนวนเมทธอดทั้งหมดที่กำหนดภายในคลาสทั้ง Public, Private และ Protected
Number of Instance Variables in a Class	NIV	จำนวนตัวแปรทั้งหมดที่ประกาศภายในคลาสทั้ง Public, Private และ Protected
Tight Class Cohesion	TCC	จำนวนความสัมพันธ์ของเมทธอดที่สัมพันธ์กันทางตรง
Number of statement	NLOC	จำนวนสเตทเมนต์ของคลาส
Average Method Complexity per Method	AMC	ค่าเฉลี่ยของค่าความซับซ้อนของเมทธอดทั้งหมดในคลาส
Depth of Inheritance Hierarchy	DIT	ความลึกของการสืบทอดคุณสมบัติของคลาส ในกรณีที่มีการสืบทอดคุณลักษณะหลายคลาส DIT คือความลึกของการสืบทอดคุณสมบัติของคลาสที่มากที่สุดจากรูธโหนด (Root node)
Number of Statements in Method	NOS	จำนวนสเตทเมนต์ของเมทธอด
Number of Parameters in Method	NOP	จำนวนพารามิเตอร์ที่ปรากฏอยู่ในซิกเนเจอร์ (Signature) ของเมทธอด

ชื่อมาตรวัด	คำย่อ	คำอธิบาย
Number of Temporary variables in Method	NOT	จำนวนตัวแปรชั่วคราวที่ประกาศใช้ในเมทธอด
Method complexity	MCX	ค่าความซับซ้อนของเมทธอด
Number of Switch statements in Method	NOSS	จำนวนสวิทช์สเตทเมนท์ภายในเมทธอด

ภาคผนวก ง

การใช้งานเครื่องมือเพื่อการคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดี

ในภาคผนวกนี้จะอธิบายถึงการติดตั้งเครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี (Tool for Detecting BadSmells: TDB) และการใช้งานเครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี ซึ่งมีรายละเอียดดังต่อไปนี้

1. การติดตั้งเครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี

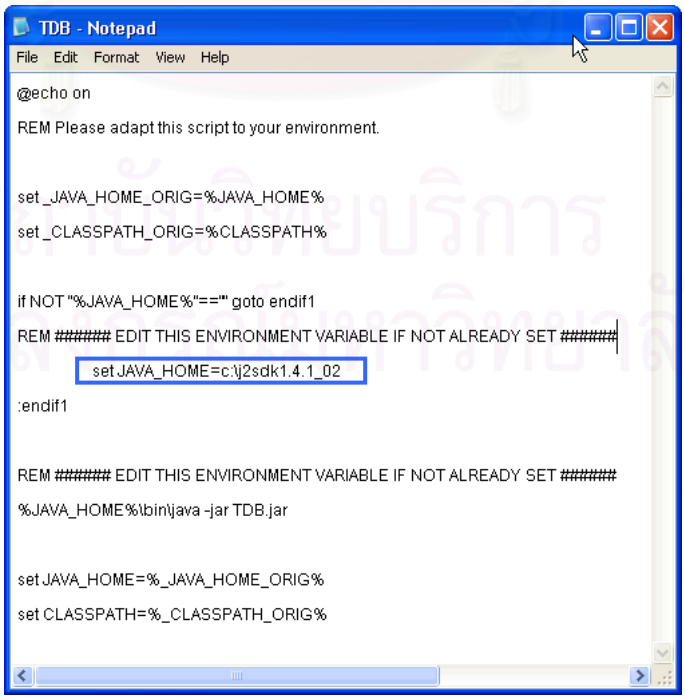
เครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี ประกอบด้วยไฟล์ข้อมูลหลัก 2 ไฟล์ คือ TDB.bat ซึ่งเป็นไฟล์ MS-DOS Batch File เพื่อใช้เรียกไฟล์ TDB.jar ซึ่งเป็นไฟล์ Execute Jar File ให้เริ่มต้นรันโปรแกรม ซึ่งทั้ง 2 ไฟล์อยู่ในแฟ้มข้อมูล TDB ขั้นตอนการติดตั้งเครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี มีดังนี้

คัดลอกไฟล์แฟ้มข้อมูล TDB ไว้ที่เครื่องคอมพิวเตอร์ซึ่งติดตั้งโปรแกรม Java™ 2 Standard Edition Runtime Environment (JRE) ไว้แล้ว แต่ถ้าเครื่องคอมพิวเตอร์ยังไม่ติดตั้งโปรแกรมดังกล่าวสามารถดาว์นโหลดได้ที่ http://java.sun.com

เปิดไฟล์ TDB.dat ด้วยโปรแกรม Notepad หรือโปรแกรม Text Editor อื่นๆ เพื่อแก้ไขค่าแปรให้เข้ากับสภาพแวดล้อมของเครื่องคอมพิวเตอร์ที่ใช้รันโปรแกรม

จากรูปที่ ง-1 ทำการแก้ไขค่าข้อมูลของตัวแปร JAVA_HOME ในบรรทัดที่ 7 ตามไดเรกทอรีที่ติดตั้งโปรแกรม JRE ไว้ ในที่นี้กำหนดค่าไว้เป็น c:\j2sdk1.4.1_02

ขั้นตอนสุดท้ายทำการบันทึกไฟล์ TDB.bat



รูปที่ ง-1 แสดงไฟล์ข้อมูล TDB.dat ที่เปิดด้วยโปรแกรม Notepad

2. **การใช้งานเครื่องมือเพื่อการคำนวณและตรวจจับร่องรอยที่ไม่ดี**

การเลือกไฟล์โปรแกรมต้นฉบับภาษาจาวา

เมื่อผู้ใช้งานเข้าสู่โปรแกรมจะแสดงหน้าจอโปรแกรมดังรูปที่ ง-2 จากนั้นผู้ใช้สามารถเลือกไฟล์โปรมต้นฉบับภาษาจาวาได้จากคลิกที่ปุ่ม  หรือเลือกเมนูเปิดไฟล์ (File -> Add Project) จะแสดงหน้าจอให้เลือกไฟล์ข้อมูลดังแสดงในรูปที่ ง-3

การเลือกไฟล์เดียว

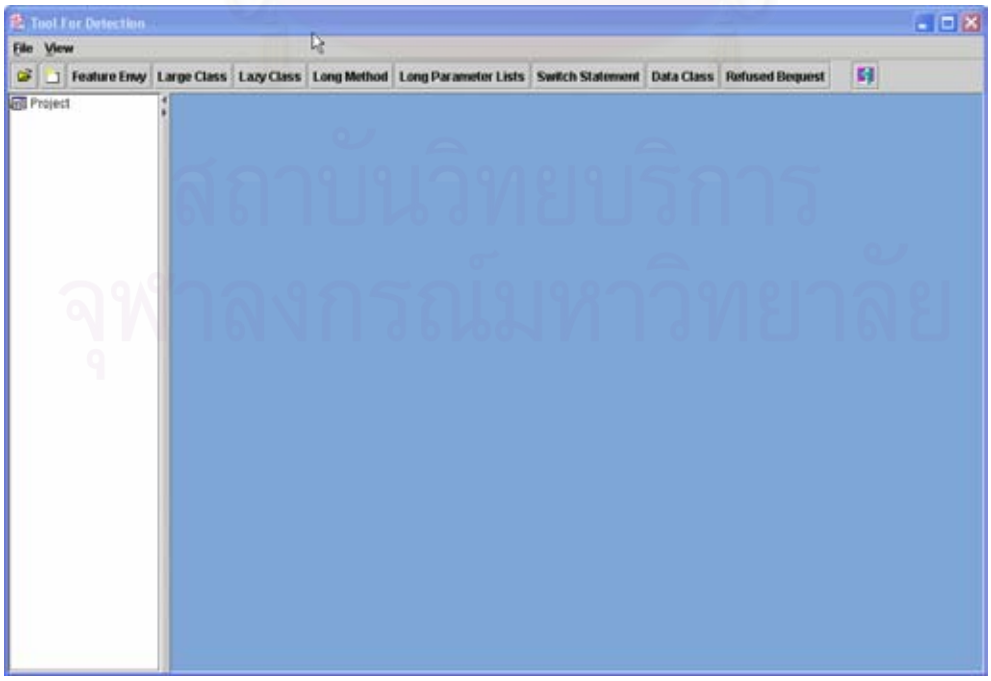
เมื่อผู้ใช้ต้องการเลือกไฟล์จาวาเพียงไฟล์เดียวสามารถทำได้โดยเลือกไฟล์จาวาต้องการแล้วกดปุ่ม Open ดังแสดงรูปที่ ง-4

การเลือกมากกว่า 1 ไฟล์

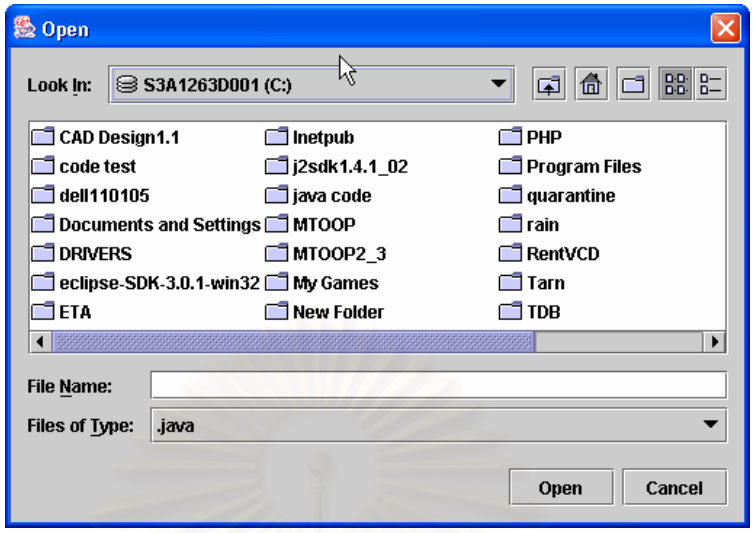
เมื่อผู้ใช้ต้องการเลือกไฟล์จาวามากกว่า 1 ไฟล์แบบติดกันสามารถทำได้โดยเลือกไฟล์จาวาเริ่มต้นที่ต้องการแล้วกดปุ่ม Shift ค้างไว้แล้วเลือกไฟล์สุดท้ายที่ต้องการแล้ว กดปุ่ม Open ดังแสดงรูปที่ ง-5 หรือ ผู้ใช้ต้องการเลือกไฟล์จาวามากกว่า 1 ไฟล์แบบไม่ติดกันสามารถทำได้โดยเลือกไฟล์จาวาที่ต้องการแล้วกดปุ่ม Ctrl ค้างไว้แล้วเลือกไฟล์อื่นๆ ที่ต้องการแล้วกดปุ่ม Open ดังแสดงรูปที่ ง-6

ยกเลิกการเลือกไฟล์

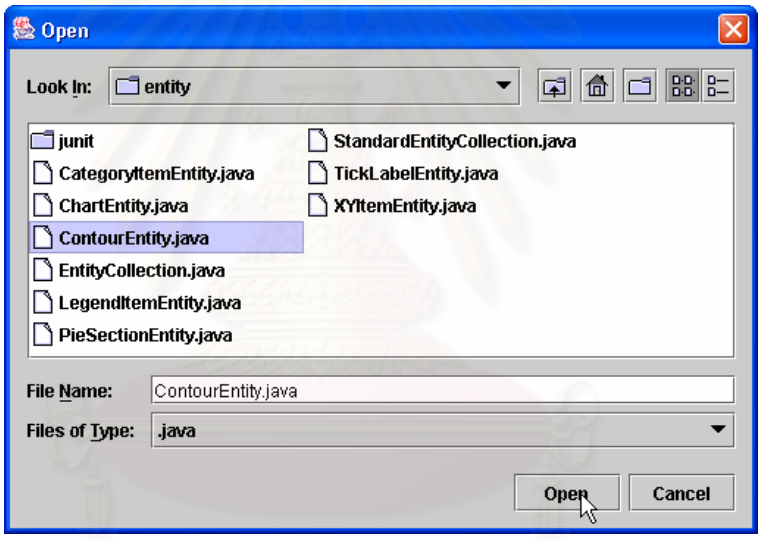
เมื่อผู้ใช้ต้องการยกเลิกการเลือกไฟล์กดปุ่ม Cancel ดังรูปที่ ง-7



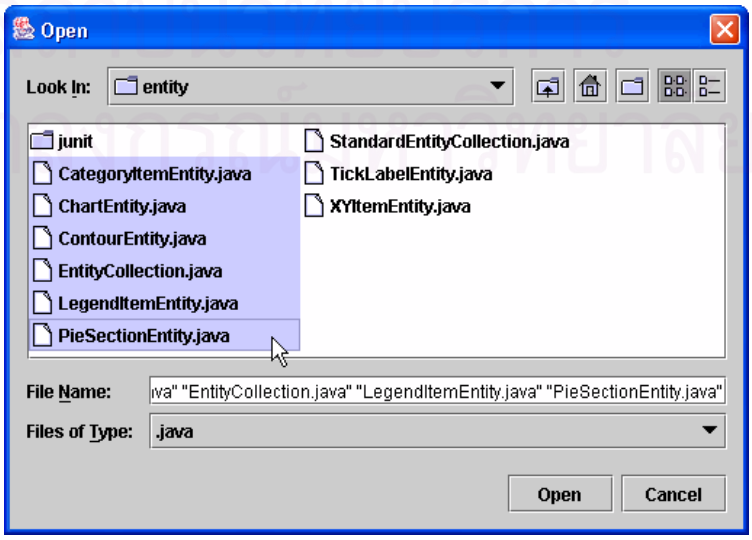
รูปที่ ง-2 แสดงหน้าจอเครื่องมือเพื่อคำนวณมาตรวัดและตรวจจับร่องรอยที่ไม่ดี



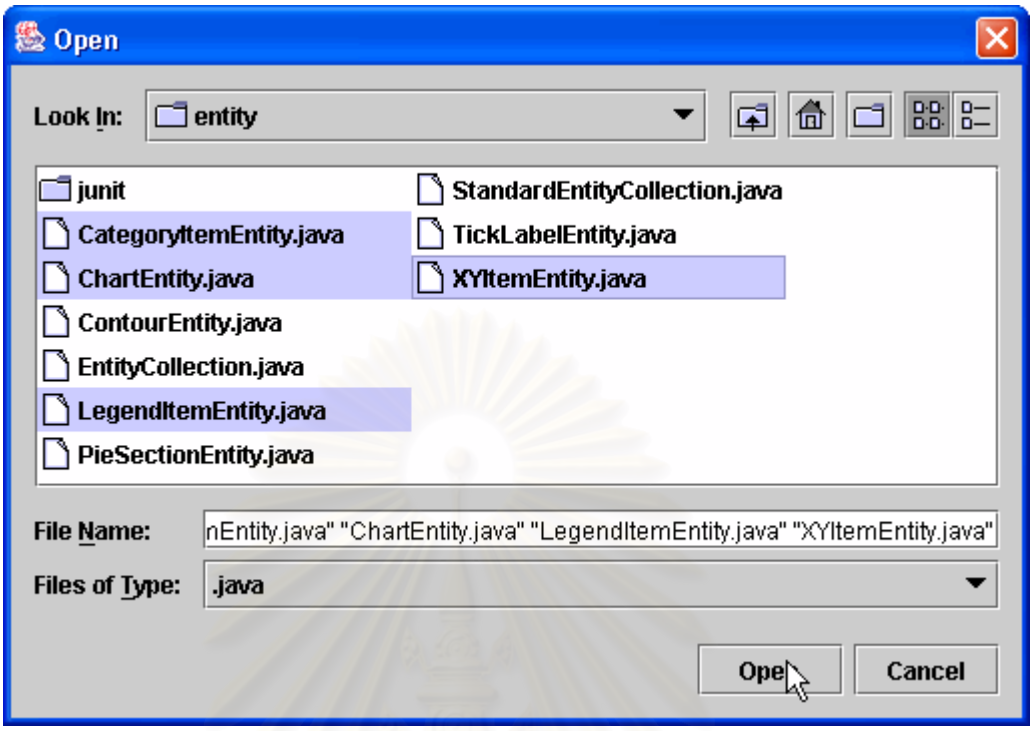
รูปที่ ง-3 แสดงหน้าจอเลือกไฟล์ .java



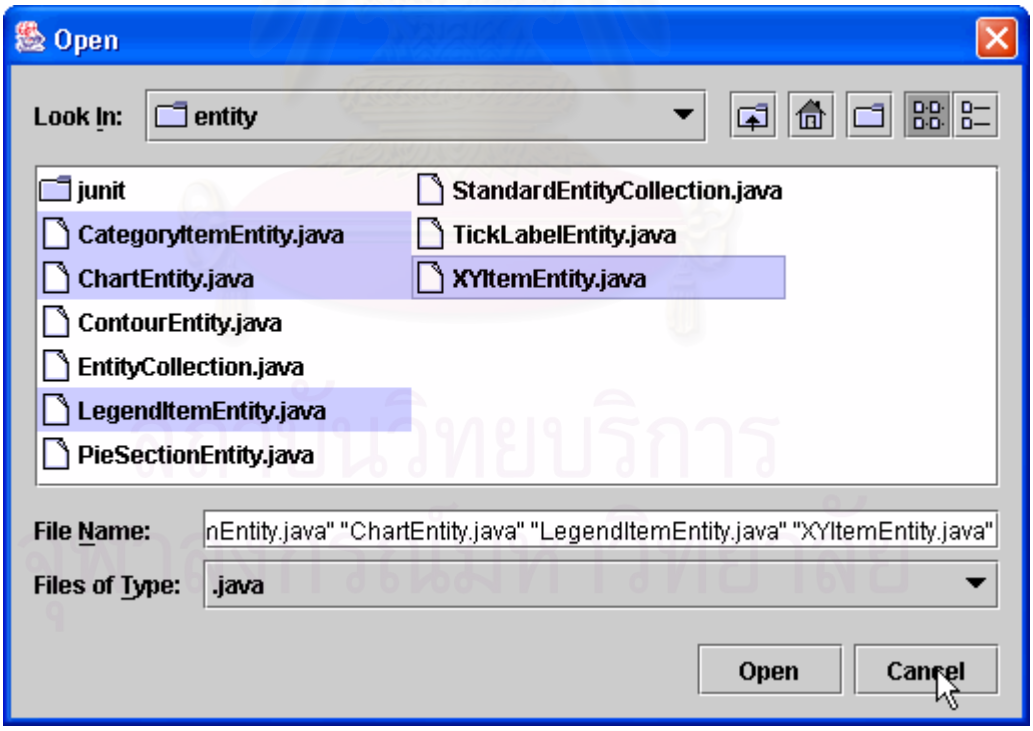
รูปที่ ง-4 แสดงการเลือกไฟล์จาวาเพียงไฟล์เดียว



รูปที่ ง-5 แสดงการเลือกไฟล์จาวาหลายไฟล์แบบติดกัน



รูปที่ ง-6 แสดงการเลือกไฟล์จาวาหลายไฟล์แบบไม่ติดกัน



รูปที่ ง-7 แสดงการยกเลิกการเลือกไฟล์จาวา

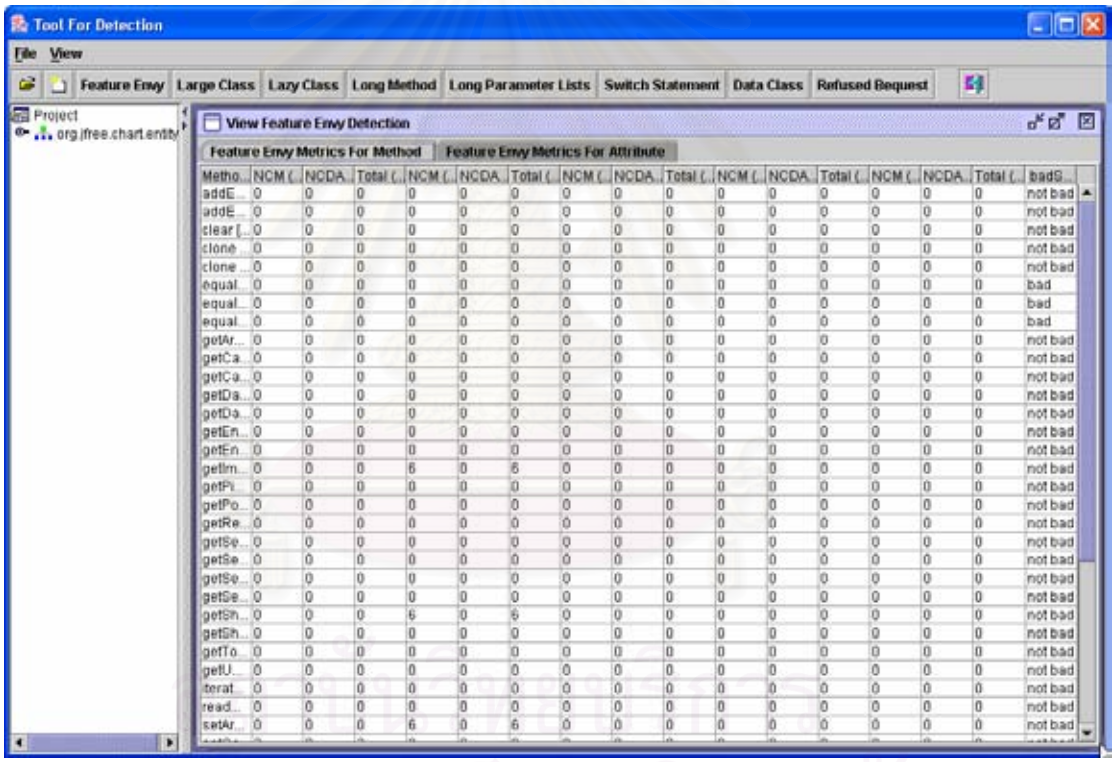
ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีทั้ง 8 แบบ

หลังจากผู้ใช้งานเลือกไฟล์จาวาแล้ว โปรแกรมจะคำนวณค่ามาตรวัดต่างๆ ของร่องรอยที่ไม่ดี เมื่อผู้ใช้งานต้องการดูค่ามาตรวัดและตำแหน่งที่เกิดร่องรอยที่ไม่ดีสามรถทำได้ดังนี้

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Feature Envy

ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Feature Envy (View -> Feature Envy) หรือกดปุ่ม 

แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Feature Envy ดังรูปที่ ง-8

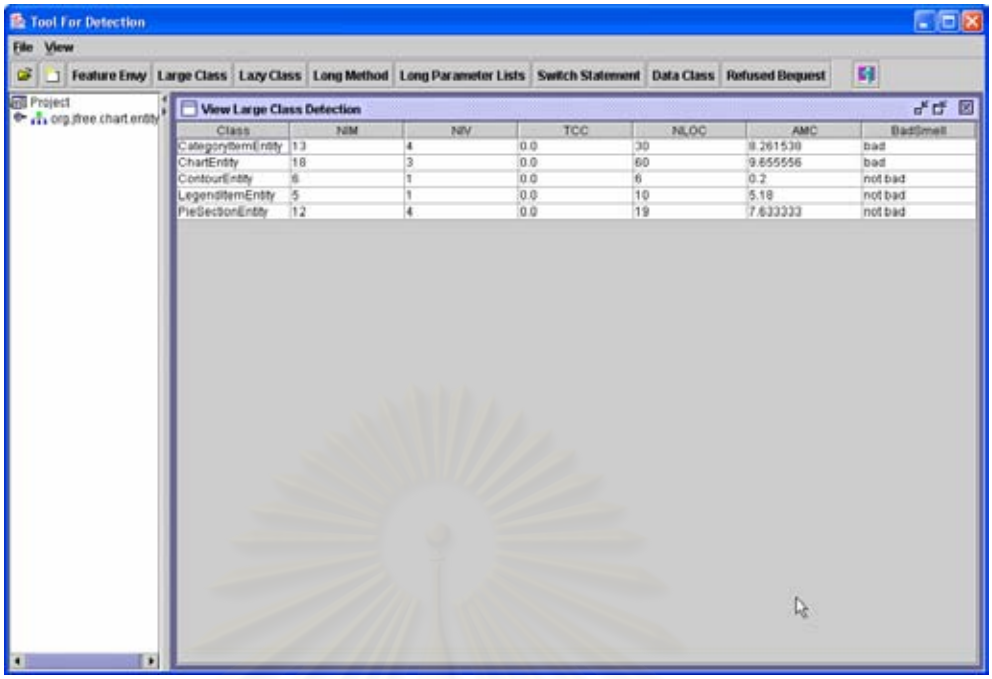


รูปที่ ง-8 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Feature Envy

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Large Class

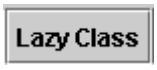
ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Large Class (View -> Large Class) หรือกดปุ่ม 

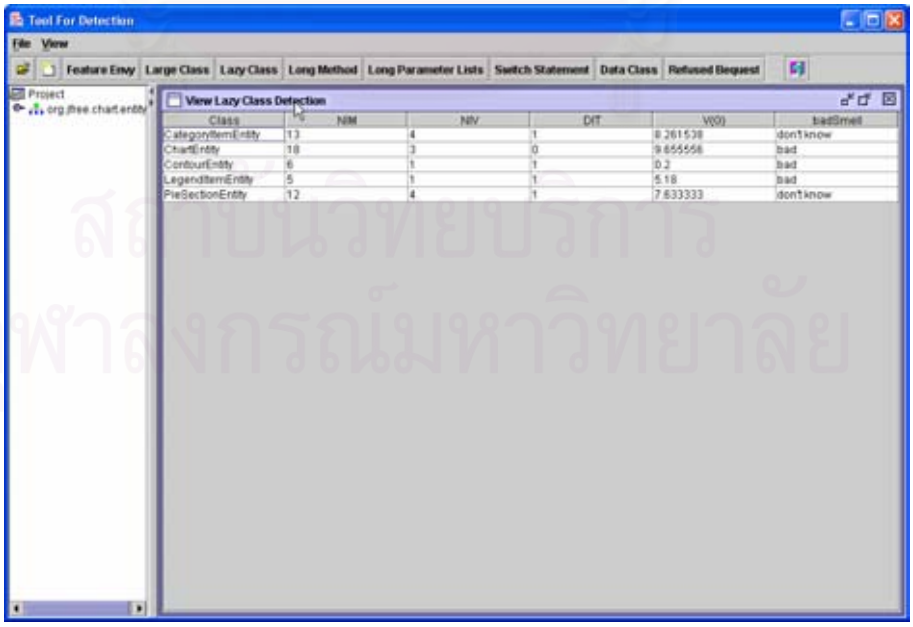
แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Large Class ดังรูปที่ ง-9



รูปที่ ง-9 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Large Class

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Lazy Class

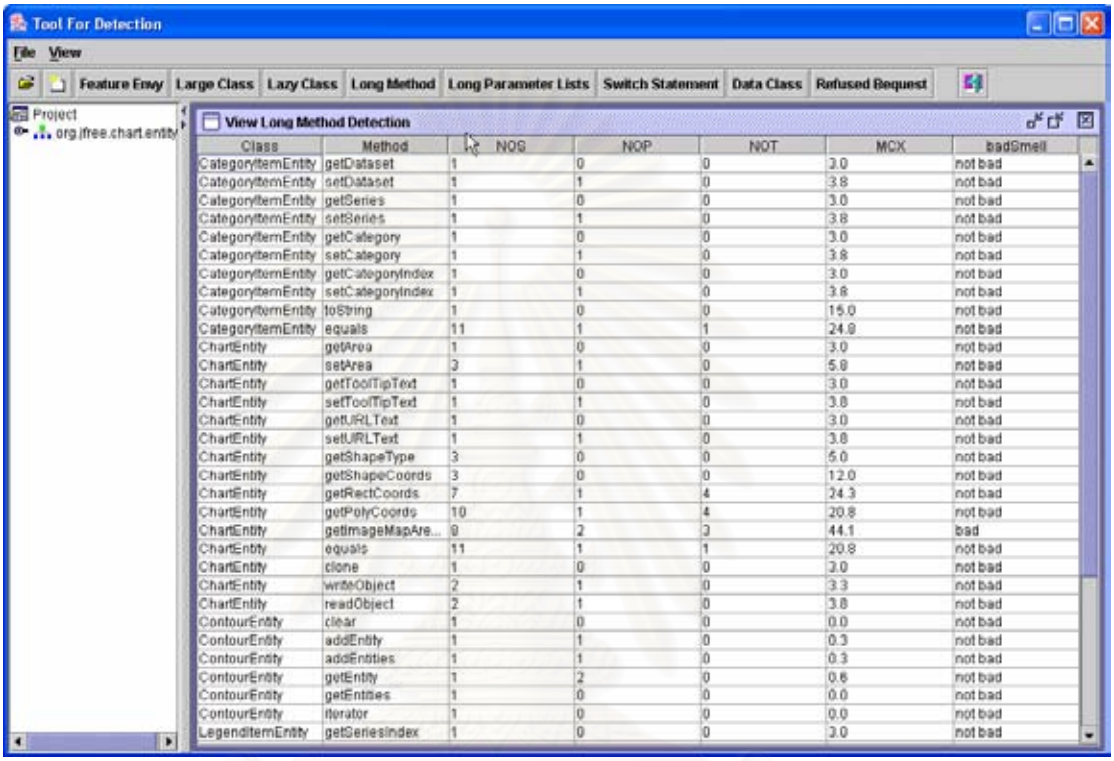
ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Lazy Class (View -> Lazy Class) หรือกดปุ่ม  แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Lazy Class ดังรูปที่ ง-10



รูปที่ ง-10 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Lazy Class

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Long Method

ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Long Method (View -> Long Method) หรือกดปุ่ม Long Method แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Long Method ดังรูปที่ ง-11

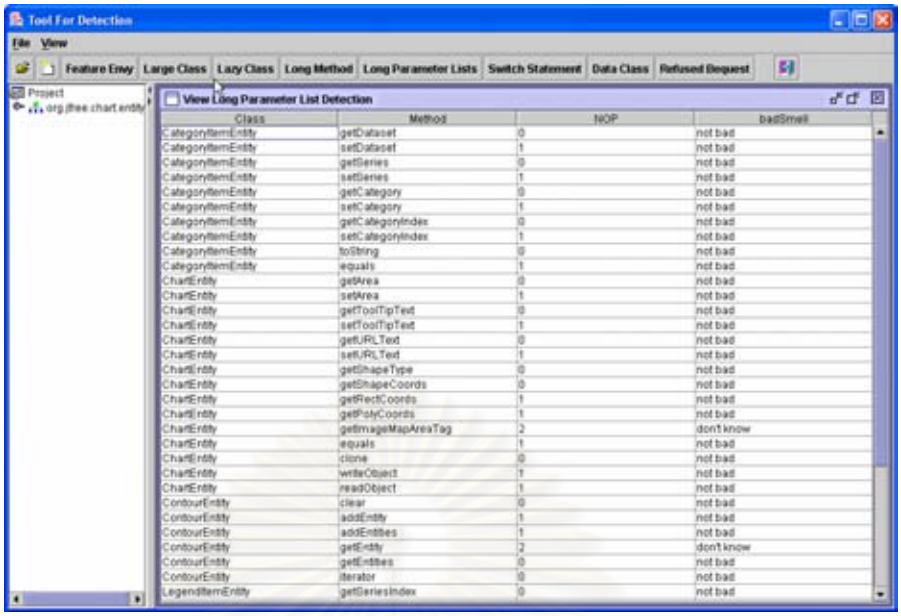


Class	Method	NOS	NOP	NOT	MCX	badSmell
CategoryItemEntity	getDataset	1	0	0	3.0	not bad
CategoryItemEntity	setDataset	1	1	0	3.8	not bad
CategoryItemEntity	getSeries	1	0	0	3.0	not bad
CategoryItemEntity	setSeries	1	1	0	3.8	not bad
CategoryItemEntity	getCategory	1	0	0	3.0	not bad
CategoryItemEntity	setCategory	1	1	0	3.8	not bad
CategoryItemEntity	getCategoryIndex	1	0	0	3.0	not bad
CategoryItemEntity	setCategoryIndex	1	1	0	3.8	not bad
CategoryItemEntity	toString	1	0	0	15.0	not bad
CategoryItemEntity	equals	11	1	1	24.9	not bad
ChartEntity	getArea	1	0	0	3.0	not bad
ChartEntity	setArea	3	1	0	5.0	not bad
ChartEntity	getToolTipText	1	0	0	3.0	not bad
ChartEntity	setToolTipText	1	1	0	3.8	not bad
ChartEntity	getURLText	1	0	0	3.0	not bad
ChartEntity	setURLText	1	1	0	3.8	not bad
ChartEntity	getShapeType	3	0	0	5.0	not bad
ChartEntity	getShapeCoords	3	0	0	12.0	not bad
ChartEntity	getRectCoords	7	1	4	24.3	not bad
ChartEntity	getPolyCoords	10	1	4	20.8	not bad
ChartEntity	getImageMapAre...	9	2	3	44.1	bad
ChartEntity	equals	11	1	1	20.8	not bad
ChartEntity	clone	1	0	0	3.0	not bad
ChartEntity	writeObject	2	1	0	3.3	not bad
ChartEntity	readObject	2	1	0	3.8	not bad
ContourEntity	clear	1	0	0	0.0	not bad
ContourEntity	addEntity	1	1	0	0.3	not bad
ContourEntity	addEntities	1	1	0	0.3	not bad
ContourEntity	getEntity	1	2	0	0.6	not bad
ContourEntity	getEntities	1	0	0	0.0	not bad
ContourEntity	iterator	1	0	0	0.0	not bad
LegendItemEntity	getSeriesIndex	1	0	0	3.0	not bad

รูปที่ ง-11 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Long Method

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Long Parameter List

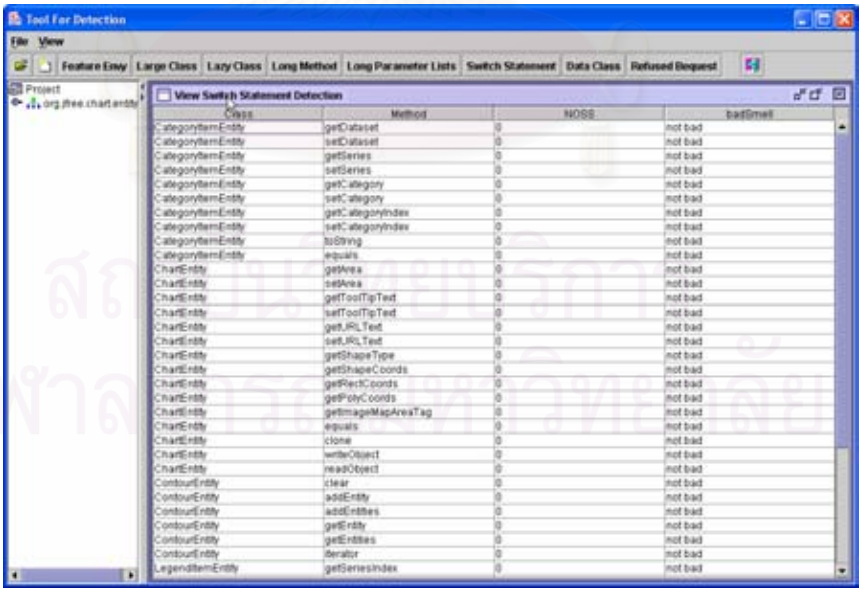
ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Long Parameter List (View -> Long Parameter List) หรือกดปุ่ม Long Parameter Lists แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดี Long Parameter List ดังรูปที่ ง-12



รูปที่ ง-12 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Long Parameter List

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Switch Statement

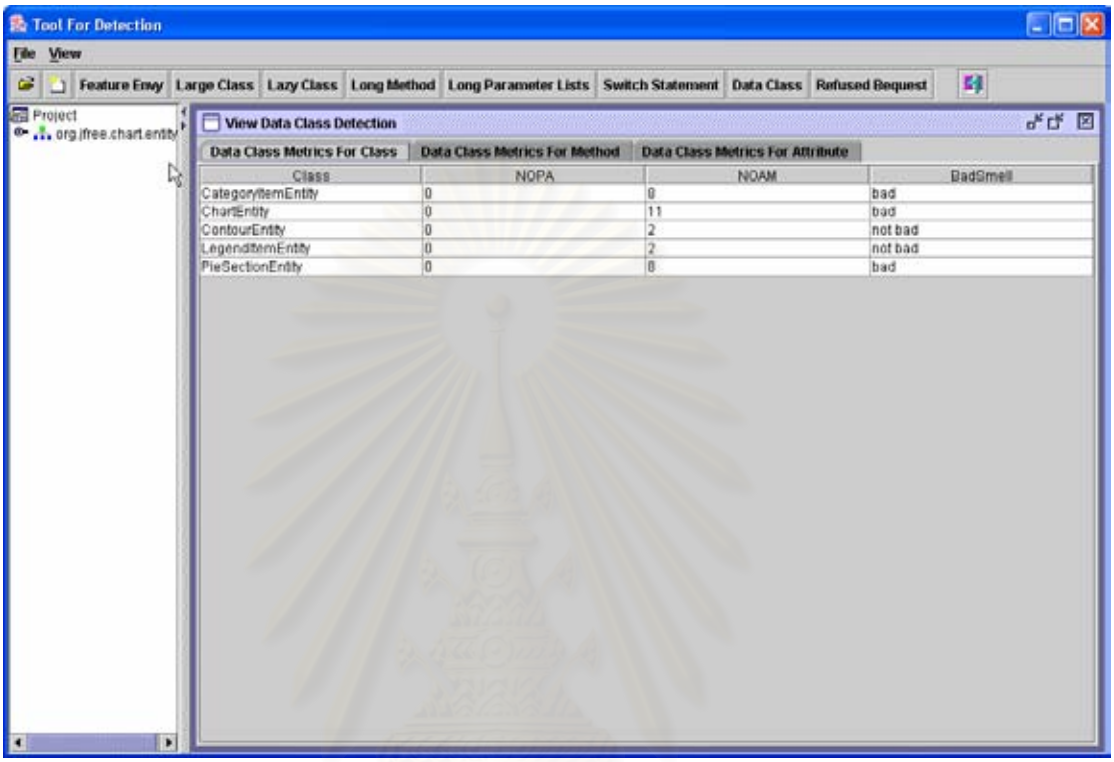
ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Switch Statement (View -> Switch Statement) หรือกดปุ่ม  แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดี Switch Statement ดังรูปที่ ง-13



รูปที่ ง-13 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Switch Statement

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Data Class

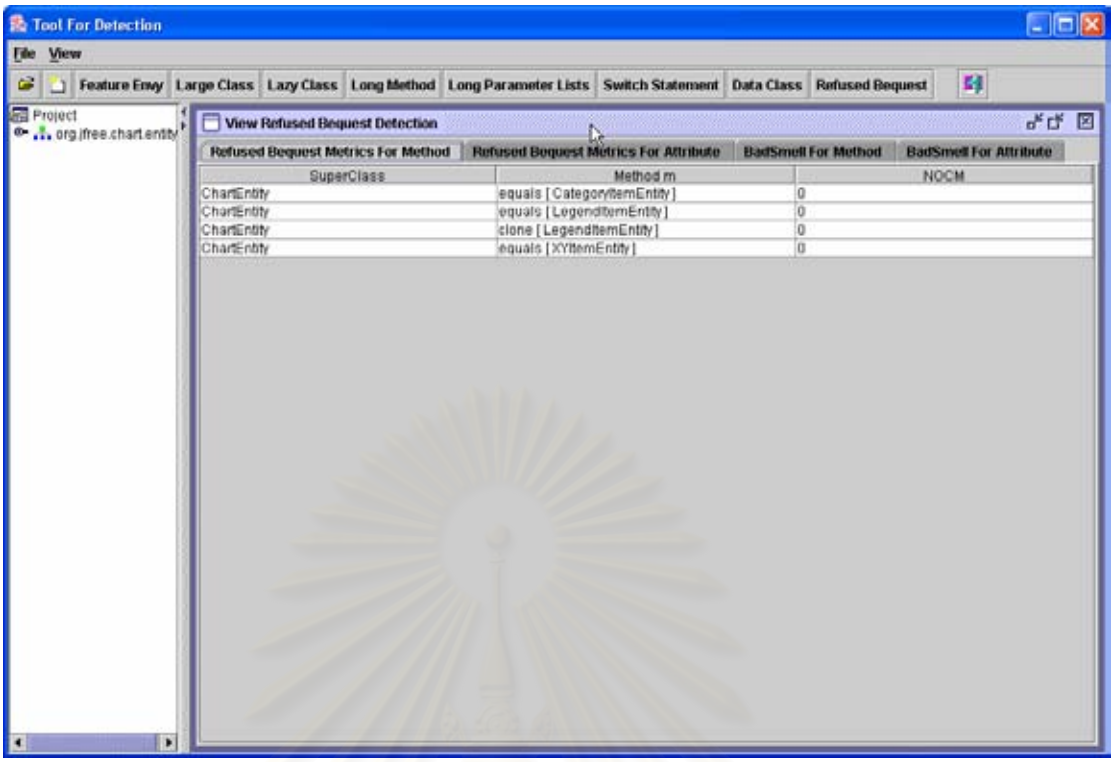
ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Data Class (View -> Data Class) หรือกดปุ่ม Data Class แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดี Data Class ดังรูปที่ ง-14



รูปที่ ง-14 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Data Class

ดูค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Refused Bequest

ผู้ใช้เลือกเมนูร่องรอยที่ไม่ดีแบบ Refused Bequest (View -> Refused Bequest) หรือกดปุ่ม Refused Bequest แล้วจะแสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดี Refused Bequest ดังรูปที่ ง-15



รูปที่ ง-15 แสดงค่ามาตรวัดและตำแหน่งที่เกิดของร่องรอยที่ไม่ดีแบบ Refused Bequest

การลบค่าสำหรับการเลือกโปรเจคใหม่

เมื่อผู้ใช้ต้องการการลบค่าสำหรับการเลือกโปรเจคใหม่ให้เลือกเมนูสร้างโปรเจคใหม่ (File -> New Project) หรือกดปุ่ม  โปรแกรมจะทำการลบค่ามาตรวัดทั้งหมด รวมทั้งลบค่าที่แสดงแผนภาพต้นไม้เพื่อเตรียมสำหรับการเลือกโปรเจคใหม่

การออกจากโปรแกรม

เมื่อผู้ใช้ต้องการออกจากโปรแกรมให้เลือกเมนูออกจากโปรแกรม (File -> Exit Project) หรือกดปุ่ม  โปรแกรมจะทำการปิดโปรแกรม

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวพนิตา เมนะเนตร เกิดเมื่อวันที่ 31 สิงหาคม พ.ศ. 2524 สำเร็จการศึกษาระดับปริญญาบัญฑิต หลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ประยุกต์ ภาควิชาวิทยาการคอมพิวเตอร์และสารสนเทศ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ เมื่อปีการศึกษา 2545 และเข้าศึกษาต่อระดับปริญญามหาบัญฑิต ปีการศึกษา 2546 หลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย