การใช้มาตรวัดเชิงวัตถุทำนายเสถียรภาพของเมทธอด

นายธีรเดช แซ่ตั้น

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-17-6937-7

USING OBJECT-ORIENTED METRICS TO PREDICT METHOD STABILITY

Mr. Teeradej Ton

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 974-17-6937-7

หัวข้อวิทยานิพนธ์	การใช้มาตรวัดเชิงวัตถุทำนายเสถียรภาพของเมทธอด
โดย	นายธีรเดช แซ่ตั้น
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
(อาจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

…… กรรมการ
(อาจารย์ นครทิพย์ พร้อมพูล)

นายธีรเดช แซ่ตั้น : การใช้มาตรวัดเชิงวัตถุทำนายเสถียรภาพของเมทธอด. (USING OBJECT-ORIENTED METRICS TO PREDICT METHOD STABILITY) อ. ที่ปรึกษา: ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี, 118 หน้า. ISBN 974-17-6937-7.

วิทยานิพนธ์นี้นำเสนอโมเดลในการทำนายเสถียรภาพของเมทธอดโดยใช้มาตรวัดเชิงวัตถุมาช่วยทำนายโอกาสที่เมทธอดหนึ่ง ๆ จะได้รับผลกระทบจากการเปลี่ยนแปลงแอทริบิวต์หรือเมทธอดของโปรแกรม เพื่อใช้ประโยชน์ในการวางแผนรองรับการเปลี่ยนแปลงที่อาจจะเกิดขึ้นในอนาคต

งานวิจัยนี้ได้นำมาตรวัดที่ใช้สร้างโมเดลทำนายเสถียรภาพของเมทธอด 3 กลุ่มคือ กลุ่มมาตรวัดขนาดของเมทธอด กลุ่มมาตรวัดความซับซ้อนของเมทธอด และกลุ่มมาตรวัดการเข้าคู่ระหว่างเมทธอด ส่วนค่าเสถียรภาพที่นำมาใช้ในการสร้างโมเดล ได้มาจากการหาผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลงรหัสโปรแกรม โดยสนใจการเปลี่ยนแปลงที่เกิดขึ้นที่แอทริบิวต์และเมทธอด ที่เป็นการเปลี่ยนแปลงแบบสถิตย์และมีผลกระทบกับวากยสัมพันธ์ของโปรแกรม

นอกจากนี้ในวิทยานิพนธ์นี้ ยังได้ศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอดที่เป็นสมาชิกของคลาสนั้น เพื่อศึกษาถึงความเป็นไปได้ในการใช้ค่าเสถียรภาพของเมทธอดมาอธิบายเสถียรภาพของคลาส ทั้งนี้ได้เลือกค่าเสถียรภาพของเมทธอดมากที่สุด น้อยที่สุดและค่าเฉลี่ยมาทำการศึกษา ผลการศึกษาที่ระดับนัยสำคัญ 0.01 พบว่าเสถียรภาพเฉลี่ยของเมทธอดมีความสัมพันธ์กับเสถียรภาพของคลาสมากที่สุด รองลงมาคือเสถียรภาพน้อยที่สุดและเสถียรภาพมากที่สุด ตามลำดับ แสดงว่าค่าเฉลี่ยเสถียรภาพของเมทธอดสามารถเป็นเครื่องชี้วัดเสถียรภาพของคลาสได้ดีที่สุด

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

4570681421 : MAJOR SOFTWARE ENGINEERING

KEY WORD: METRICS / PREDICT / METHOD STABILITY / CLASS STABILITY / STATIC CHANGE / SYNTACTIC IMPACT

TEERADEJ TON : USING OBJECT-ORIENTED METRICS TO PREDICT METHOD STABILITY. THESIS ADVISOR: ASST. PROF. PORNSIRI MUENCHAISRI, Ph.D., 118 pp. ISBN 974-17-6937-7.

This thesis proposes a model for predicting method stability using object-oriented metrics. This model predicts possibility that a method will be affected from change at an attribute or a method of source code. Thus, suitable actions may be planned when a change occurs.

This research uses 3 metric sets including method size, method complexity and coupling between methods. The stability used to construct the model is gotten from change effect when source code had been changed. The considered changes are static changes and syntactic impacts which occur at an attribute or a method of source code.

This thesis also studies the relationship between stability of a class and stability of methods that are member of that class in order to study the possibility of using the method stability to explain the class stability by choosing maximum, minimum and mean of method stability to study. The result at 0.01 significant level shows that mean of method stability has highest correlation with class stability, and maximum of method stability has lowest correlation with class stability. Therefore, a mean of method stability should be used as an indicator of a class stability.

Department…. Computer Engineering…. Student's.....
Field of study…. Software Engineering…Advisor's.....
Academic year …2004……

กิตติกรรมประกาศ

ข้าพเจ้าใคร่ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ที่กรุณาแนะนำให้ความรู้ ให้คำปรึกษา ความช่วยเหลือต่าง ๆ ตลอดจนคอยดูแลการทำวิทยานิพนธ์ของข้าพเจ้าจนสำเร็จลุล่วงลงได้ด้วยดี

ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์ ซึ่งเป็นประธานกรรมการสอบวิทยานิพนธ์ อาจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์ และอาจารย์ นครทิพย์ พร้อมพูล ซึ่งเป็นกรรมการสอบวิทยานิพนธ์ ที่ได้สละเวลาและให้คำแนะนำต่าง ๆ อันเป็นประโยชน์อย่างยิ่งต่อการจัดทำวิทยานิพนธ์ฉบับนี้

ขอขอบคุณอาจารย์ทุกท่าน ที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า รวมถึงข้อชี้แนะต่างๆ ตลอดเวลาที่ข้าพเจ้าได้ศึกษาเล่าเรียนในระดับมหาบัณฑิต ณ สถาบันแห่งนี้

ขอขอบคุณเพื่อน ๆ พี่ ๆ และน้อง ๆ ทุกคนในห้องปฏิบัติการชั้น 19 และเพื่อนร่วมรุ่นของข้าพเจ้าทุกคนที่ช่วยแก้ปัญหาเบ็ดเตล็ดต่าง ๆ ในงานวิจัย

เหนือสิ่งอื่นใดข้าพเจ้าขอกราบขอบพระคุณของคุณพ่อคุณแม่ผู้ให้กำเนิด รวมถึงพี่น้องของข้าพเจ้าที่คอยเป็นกำลังใจ ห่วงใยข้าพเจ้าและสนับสนุนด้านการเงินแก่ข้าพเจ้าเสมอมา

สารบัญ

หน้า

สารบัญภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเปลี่ยนแปลงเป็นสิ่งที่หลีกเลี่ยงได้ยากในการพัฒนาซอฟต์แวร์ เมื่อมีการเปลี่ยนแปลงเกิดขึ้น ย่อมส่งผลกระทบให้ต้องมีการแก้ไขซอฟต์แวร์ในส่วนต่าง ๆ มาตรฐานคุณภาพซอฟต์แวร์ไอเอสโอ/ไออีซี 9126 (ISO/IEC 9126) [1] ได้กำหนดให้เสถียรภาพ (Stability) เป็นหนึ่งในหกคุณสมบัติในมาตรฐานชุดดังกล่าว โดยเสถียรภาพนั้นพิจารณาถึงความเสี่ยงที่ซอฟต์แวร์จะถูกกระทบโดยไม่ได้ตั้งใจจากการเปลี่ยนแปลงแก้ไขที่เกิดขึ้น ถ้ามีความถี่ของการแก้ไขสูงจะส่งผลให้เสถียรภาพของซอฟต์แวร์นั้นต่ำ แต่ในทางตรงกันข้าม ถ้ามีความถี่ของการแก้ไขต่ำเสถียรภาพของซอฟต์แวร์ก็จะสูง หากทราบว่าซอฟต์แวร์มีเสถียรภาพต่ำจะได้วางแผนรองรับกับการเปลี่ยนแปลงที่อาจจะเกิดขึ้นในอนาคต

ในการเปลี่ยนแปลงแต่ละครั้งจะประกอบด้วย 2 ส่วนคือ

1. ส่วนที่ต้องเปลี่ยนแปลง เป็นส่วนที่ต้องการเปลี่ยนแปลงแก้ไขจริง ๆ และเป็นส่วนที่ทำให้เกิดผลกระทบการเปลี่ยนแปลงเป็นแบบลูกคลื่น (Ripple effect) [2] หรือทำให้เกิดการแพร่กระจายของการเปลี่ยนแปลง (Propagation of change) [3] ไปยังส่วนอื่นๆ ขึ้นอยู่ว่ามีปฏิสัมพันธ์กับส่วนใดบ้าง

2. ส่วนที่ได้รับผลกระทบจากการเปลี่ยนแปลง จะเป็นส่วนที่จะต้องมีการปรับเปลี่ยนหรือแก้ไขพฤติกรรม อันเนื่องจากส่วนที่มีปฏิสัมพันธ์ด้วยนั้นเกิดการเปลี่ยนแปลงขึ้น เช่นตัวอย่างรหัสโปรแกรมในตารางที่ 1.1

จากตารางที่ 1.1 ถ้าต้องการลบแอทริบิวต์ b1 ในคลาส B การเปลี่ยนแปลงในครั้งนี้ประกอบด้วย

1. ส่วนที่ต้องเปลี่ยนแปลง คือ แอทริบิวต์ b1 ในคลาส B ซึ่งต้องการลบ

2. ส่วนที่ได้รับผลกระทบจากการเปลี่ยนแปลงคือ เมทธอด mA1 ในคลาส A และเมทธอด mB1 ในคลาส B ซึ่งทั้งสองเมทธอดนี้จะต้องถูกแก้ไขตามไปด้วย

ตารางที่ 1.1 รหัสโปรแกรมตัวอย่าง

public class A{
  public int a1;
  public void mA1(){
  B b = new B();
  a1= b.**b1**;
  }
}
	public class B{
	  public int **b1**;
	  public void mB1(){
	  if(**b1**==0)
	  System.out.println("0");
	  }
	}

ในซอฟต์แวร์เชิงวัตถุ เสถียรภาพของคลาส (Class stability) หมายถึง โอกาสหรือความเสี่ยงที่คลาสจะได้รับผลกระทบโดยไม่ได้ตั้งใจอันเนื่องมาจากการเปลี่ยนแปลงของคลาสอื่นในซอฟต์แวร์นั้น ได้ถูกนำมาใช้เพื่อหาความสามารถในการต้านทานการเปลี่ยนแปลงที่เกิดขึ้นกับคลาสหนึ่ง ๆ อันมีสาเหตุมาจากการแพร่กระจายการเปลี่ยนแปลงจากคลาสอื่น [2] ที่มีปฏิสัมพันธ์ด้วย

ในกรณีที่พบว่าคลาสมีเสถียรภาพต่ำ แล้วต้องการทราบว่าส่วนใดของคลาสที่มีผลทำให้เสถียรภาพต่ำจึงพิจารณาถึงองค์ประกอบของคลาส จะพบว่าคลาสประกอบด้วยแอทริบิวต์และเมทธอดโดยแอทริบิวต์เป็นส่วนที่ใช้เก็บค่าของวัตถุ จะได้รับผลกระทบจากการเปลี่ยนแปลงที่เป็นแบบพลวัต (Dynamic change) ส่วนเมทธอดจะเป็นส่วนที่ใช้แสดงพฤติกรรมของคลาสมีโอกาสจะได้รับผลกระทบจากการเปลี่ยนแปลงทั้งแบบสถิตย์ (Static change) และแบบพลวัต ดังนั้นเมทธอดจึงมีโอกาสได้รับผลกระทบจากการเปลี่ยนแปลงสูงกว่าแอทริบิวต์ เสถียรภาพของคลาสจึงน่าจะขึ้นอยู่กับว่าเมทธอดมีเสถียรภาพมากหรือน้อยเพียงใด หากทราบเสถียรภาพของเมทธอดก็อาจคาดคะเนถึงเมทธอดที่ทำให้คลาสมีเสถียรภาพต่ำได้

ในงานวิจัยนี้จึงสนใจการหาเสถียรภาพของเมทธอด แต่การคำนวณหาเสถียรภาพโดยตรงนั้นมีความซับซ้อน เนื่องจากจะต้องทดลองเปลี่ยนแปลงแก้ไขรหัสโปรแกรม แล้ววัดผลกระทบที่เกิดขึ้นเพื่อนำมาใช้ในการคำนวณ ดังนั้นในงานวิจัยนี้ จึงมีแนวคิดในการนำมาตรวัดเชิงวัตถุมาสร้างเป็นโมเดล เพื่อช่วยทำนายเสถียรภาพของเมทธอด โดยพิจารณาเฉพาะการเปลี่ยนแปลงที่เป็นแบบสถิตย์และมีผลกระทบกับวากยสัมพันธ์ของโปรแกรม (Syntactic impact) โดยตำแหน่งในการเปลี่ยนแปลงเกิดที่แอทริบิวต์หรือเมทธอดของโปรแกรม หลังจากนั้นจึงศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอดที่ได้จากการทำนาย ว่ามีความสัมพันธ์กันมากน้อยเพียงใด

1.2 วัตถุประสงค์

1. สร้างโมเดลทำนายเสถียรภาพของเมทธอดโดยใช้มาตรวัดเชิงวัตถุ
2. ศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด
3. ออกแบบและพัฒนาเครื่องมือจัดเก็บมาตรวัด เครื่องมือคำนวณเสถียรภาพและเครื่องมือทำนายเสถียรภาพของเมทธอด

1.3 ขอบเขตงานวิจัย

1. งานวิจัยนี้จะพิจารณาการเปลี่ยนแปลงที่เกิดขึ้นที่แอทริบิวต์และเมทธอด โดยเป็นการเปลี่ยนแปลงแบบสถิตย์และมีผลกระทบกับวากยสัมพันธ์ของโปรแกรมเท่านั้น
2. รหัสโปรแกรมที่นำมาใช้ในงานวิจัย จะต้องพัฒนามาจากภาษาจาวา และผ่านการคอมไพล์เรียบร้อยแล้ว โดยมีจำนวนคลาสตั้งแต่ 10 คลาส ถึง 50 คลาส
3. รหัสโปรแกรมที่นำมาใช้ในการหามาตรวัดและหาความสัมพันธ์ต่าง ๆ จะพิจารณาเฉพาะส่วนที่มีรหัสโปรแกรมเท่านั้น ไม่รวมคลาสที่เป็นไลบรารี่
4. ในการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของเมทธอดและเสถียรภาพของคลาส จะทำการศึกษาทั้งเสถียรภาพของเมทธอดที่มากที่สุด น้อยที่สุดและค่าเฉลี่ยในแต่ละคลาส

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. ศึกษาและค้นหามาตรวัดเชิงวัตถุ เพื่อนำมาใช้หาความสัมพันธ์และสร้างเป็นโมเดลในการทำนายเสถียรภาพของเมทธอด
2. พัฒนาเครื่องมือจัดเก็บมาตรวัดและเก็บค่ามาตรวัดจากรหัสโปรแกรม
3. พัฒนาเครื่องมือคำนวณเสถียรภาพของเมทธอดและคลาส และคำนวณเสถียรภาพของแต่ละเมทธอดและแต่ละคลาสในโปรแกรม ตามรูปแบบการเปลี่ยนแปลงที่เลือกไว้
4. หาความสัมพันธ์ระหว่างมาตรวัดและเสถียรภาพของเมทธอด เพื่อสร้างโมเดลในการทำนายเสถียรภาพของเมทธอด

5. ประเมินผลความถูกต้องของโมเดลทำนายเสถียรภาพของเมทธอด
6. พัฒนาเครื่องมือทำนายเสถียรภาพของเมทธอดตามโมเดลที่ได้สร้างขึ้น
7. ศึกษาความสัมพันธ์ระหว่างเสถียรภาพของเมทธอดกับเสถียรภาพของคลาส
8. วิเคราะห์สรุปผลการวิจัยและจัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โมเดลทำนายเสถียรภาพของเมทธอดของโปรแกรม
2. ได้เครื่องมือจัดเก็บมาตรวัด เครื่องมือคำนวณเสถียรภาพและเครื่องมือทำนายเสถียรภาพของเมทธอด
3. ทราบถึงความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การวัดซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Measurement) [4]

การวัดเป็นกระบวนการในการกำหนดตัวเลขหรือสัญลักษณ์ให้กับสิ่งที่สนใจ ซึ่งการวัดจะมีประโยชน์ทำให้สามารถควบคุมสิ่งที่สนใจได้ สำหรับการวัดซอฟต์แวร์จะทำให้สามารถควบคุมซอฟต์แวร์ให้มีคุณภาพตามต้องการได้ สำหรับการวัดซอฟต์แวร์ สามารถแบ่งได้เป็น 2 ประเภทคือ

2.1.1.1 การวัดโดยตรง เป็นการวัดคุณลักษณะภายในของตัวซอฟต์แวร์ เช่น การนับจำนวนบรรทัดของซอฟต์แวร์ เป็นต้น

2.1.1.2 การวัดโดยอ้อม เป็นการวัดที่ไม่ได้ทำกับคุณลักษณะภายใน แต่ได้จากการคำนวณคุณลักษณะภายในโดยตรง เช่น ความสามารถในการบำรุงรักษาซอฟต์แวร์ (Software maintainability) ซึ่งจะไม่สามารถวัดจากลักษณะภายในของซอฟต์แวร์ได้ จึงต้องอาศัยลักษณะภายในต่าง ๆ ของตัวซอฟต์แวร์เข้ามาช่วย เช่น ใช้จำนวนคลาส จำนวนเมทธอดของซอฟต์แวร์ มาคำนวณ เป็นต้น

2.1.2 มาตรวัดเชิงวัตถุ (Object-oriented metrics)

มาตรวัดเชิงวัตถุที่ใช้ในงานวิจัยนี้ ได้นำมาจาก [4][5][6] มีรายละเอียดดังนี้

2.1.2.1 มาตรวัดจำนวนพารามิเตอร์ (Parameter : Param) เป็นมาตรวัดขนาดเมทธอดโดยพิจารณาจากจำนวนพารามิเตอร์ที่ประกาศใช้ของเมทธอด หากเมทธอดมีจำนวนพารามิเตอร์มาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.2 มาตรวัดจำนวนตัวแปรแบบโลคอล (Local variable : LocalVar) เป็นมาตรวัดขนาดของเมทธอดโดยพิจารณาจากจำนวนตัวแปรที่ประกาศเป็นแบบโลคอล ซึ่งเป็นตัวแปรชั่วคราว คือมีการใช้งานเมทธอด ตัวแปรจะถูกสร้างขึ้นมา แต่เมื่อเสร็จสิ้นการทำงานในเมทธอดนั้นแล้ว ตัวแปรเหล่านี้จะถูกทำลาย หากเมทธอดมีจำนวนตัวแปรแบบโลคอลมาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.3 มาตรวัดจำนวนสเตทเมนต์ (Number of statement : NOS) เป็นมาตรวัดขนาดของโปรแกรม โดยสเตทเมนต์ยังแบ่งเป็นสเตทเมนต์ที่ประกาศค่าต่าง ๆ (Declarative statement) และสเตทเมนต์ที่ประมวลผล (Execultable statement) ซึ่งขึ้นอยู่กับข้อกำหนดในการนับสเตทเมนต์ว่านับสเตทเมนต์แบบใดบ้าง หากโปรแกรมมีจำนวนสเตทเมนต์มาก จะทำให้โปรแกรมมีขนาดใหญ่ ซึ่งเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.4 มาตรวัดจำนวนบรรทัดของรหัสโปรแกรม (Line of code : LOC) เป็นมาตรวัดขนาดของโปรแกรม สำหรับการคำนวณจำนวนบรรทัดมีด้วยกันหลายวิธี เช่น ผลรวมของจำนวนสเตทเมนต์ (State) จำนวนบรรทัดว่าง (Blank lines) จำนวนบรรทัดคอมเมนต์ (Comment lines) และจำนวนบรรทัดที่เป็นเครื่องหมายบล็อก (Block delimiters) เป็นต้น ซึ่งขึ้นอยู่กับข้อกำหนดในการนับจำนวนบรรทัดว่าจะนับค่าใดบ้าง หากโปรแกรมมีจำนวนบรรทัดมาก จะทำให้โปรแกรมมีขนาดใหญ่ ซึ่งเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.5 มาตรวัดไซโคลเมติกของแมคเคบ (Cyclomatic complexity : V(G)) เป็นมาตรวัดที่ใช้วัดความซับซ้อนของเส้นทางในการท่องในโปรแกรมหรือจำนวนเส้นทางอิสระในโปรแกรม ซึ่งพิจารณาจากจำนวนกิ่ง (Branch) ภายในโปรแกรม หากโปรแกรมมีจำนวนกิ่งมาก จะทำให้โปรแกรมมีความซับซ้อนมาก ซึ่งเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.6 มาตรวัดความซับซ้อนของเมทธอด (Method complexity : MCX) เป็นมาตรวัดที่ใช้วัดความซับซ้อนภายในเมทธอด โดยพิจารณาจากประเภทของข้อความที่มีการเรียกใช้ หากเมทธอดมีความซับซ้อนมาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.7 มาตรวัดการเข้าคู่กันระหว่างวัตถุ (Coupling between object : CBO) การเข้าคู่กัน หมายถึง การที่วัตถุหรือคลาสหนึ่งมีการเรียกใช้วัตถุหรือคลาสอื่น ๆ ซึ่งยังรวมถึงการเรียกใช้เมทธอดหรือแอทริบิวต์ในคลาสอื่น ๆ ด้วย ซึ่งขึ้นอยู่กับข้อกำหนดในการนับ หากมีการเข้าคู่กันระหว่างวัตถุมาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.8 มาตรวัดจำนวนการเรียกใช้เมทธอด (Number of method invocation : NMI) เป็นมาตรวัดการเข้าคู่ระหว่างวัตถุรูปแบบหนึ่ง โดยพิจารณาจากจำนวนเมทธอดที่ถูกเรียกในวัตถุหนึ่ง ๆ ซึ่งหากในวัตถุมีจำนวนการเรียกใช้เมทธอดมาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.2.9 มาตรวัดจำนวนการเรียกใช้แอทริบิวต์ (Number of attribute invocation : NAI) เป็นมาตรวัดการเข้าคู่ระหว่างวัตถุรูปแบบหนึ่ง โดยพิจารณาจากจำนวนแอทริบิวต์ที่ถูกเรียกในวัตถุหนึ่ง ๆ ซึ่งหากในวัตถุมีจำนวนการเรียกใช้แอทริบิวต์มาก จะเป็นผลให้การบำรุงรักษาหรือการทำความเข้าใจโปรแกรมทำได้ยาก

2.1.3 การเปลี่ยนแปลงในซอฟต์แวร์ [7]

การเปลี่ยนแปลงในซอฟต์แวร์ โดยทั่ว ๆ ไปสามารถแบ่งเป็นประเภทต่าง ๆ ตามการพิจารณา ได้ดังนี้

2.1.3.1 พิจารณาจากเวลาที่เกิดการเปลี่ยนแปลง สามารถแบ่งได้ 2 ประเภท คือ

1. การเปลี่ยนแปลงแบบสถิตย์ เป็นการเปลี่ยนแปลงที่เกิดขึ้นในขณะที่ไม่ได้ประมวลผลโปรแกรม คือ เปลี่ยนแปลงลงในรหัสโปรแกรม
2. การเปลี่ยนแปลงแบบพลวัต เป็นการเปลี่ยนแปลงที่เกิดขึ้นขณะประมวลผลโปรแกรม

2.1.3.2 พิจารณาจากผลกระทบที่เกิดจากการเปลี่ยนแปลง สามารถแบ่งได้ 2 ประเภท คือ

1. การเปลี่ยนแปลงที่มีผลกระทบกับวากยสัมพันธ์ เป็นการเปลี่ยนแปลงที่เมื่อเกิดขึ้นแล้วจะทำให้โปรแกรมผิดพลาดคอมไพล์ไม่ผ่าน
2. การเปลี่ยนแปลงที่มีผลกระทบกับความหมาย (Semantic impact) เป็นการเปลี่ยนแปลงที่เกิดขึ้นแล้ว โปรแกรมยังคงคอมไพล์ผ่าน แต่ทำให้การทำงานในแง่ของความหมายหรือพฤติกรรมของโปรแกรมเปลี่ยนแปลงไป

2.1.4 การเปลี่ยนแปลงและผลกระทบในซอฟต์แวร์เชิงวัตถุ [2][8][9]

การเปลี่ยนแปลงในซอฟต์แวร์เชิงวัตถุ สามารถแบ่งเป็นประเภทต่าง ๆ ตามจุดที่เกิดการเปลี่ยนแปลงได้ดังนี้

2.1.4.1 การเปลี่ยนแปลงที่ความสัมพันธ์ระหว่างคลาส

1. การเพิ่มการสืบทอดคุณสมบัติ ทำให้คลาสที่เพิ่มการสืบทอดนั้นกลายเป็นคลาสลูกและสืบทอดคุณสมบัติจากคลาสพ่อมา การเพิ่มการสืบทอดคุณสมบัติอาจเกิดผลกระทบกับคลาสลูกหรือไม่ก็ได้ ทั้งนี้ขึ้นอยู่กับคุณสมบัติที่สืบทอดมานั้น มีข้อขัดแย้งกันหรือไม่

2. การลบการสืบทอดคุณสมบัติ ทำให้คลาสลูกกลายเป็นคลาสอิสระ ดังนั้นการเปลี่ยนแปลงแบบนี้อาจจะเกิดผลกระทบหรือไม่ก็ได้ เนื่องจากคลาสลูกอาจไม่ได้เรียกใช้คุณสมบัติที่สืบทอดมา แต่หากมีการเรียกใช้คุณสมบัติสืบทอดมาก็จะได้รับผลกระทบ เนื่องจากไม่สามารถเรียกใช้คุณสมบัติที่สืบทอดมาได้

2.1.4.2 การเปลี่ยนแปลงที่คลาส

การเปลี่ยนแปลงที่เกิดที่คลาส จะพิจารณาเฉพาะการเพิ่มคลาส ลบคลาสและการแก้ไขส่วนโครงสร้างภายนอกของคลาสเท่านั้น ส่วนการแก้ไขภายในคลาสนั้นจะแยกพิจารณาเป็นการเปลี่ยนแปลงที่เกิดกับแอทริบิวต์และเมทธอด ในข้อ 2.1.4.3 และ 2.1.4.4 สำหรับการเปลี่ยนแปลงที่คลาสสามารถแสดงได้ดัง รูปที่ 2.1

รูปที่ 2.1 รูปแบบการเปลี่ยนแปลงที่คลาส

1. เพิ่มคลาส เป็นการเปลี่ยนแปลงที่ไม่มีผลกระทบกับคลาสอื่น เนื่องจากคลาสที่เพิ่มเข้าไปใหม่นี้ยังไม่ถูกอ้างอิงจากคลาสใดๆ

2. ลบคลาส เป็นการเปลี่ยนแปลงที่ทำให้คลาสอื่น ๆ ที่เรียกใช้คลาสที่ถูกลบทั้งหมดถูกกระทบ เพราะไม่สามารถเรียกใช้แอทริบิวต์หรือเมทธอดในคลาสที่ถูกลบไปได้

3. เปลี่ยนแปลงโครงสร้าง การเปลี่ยนแปลงในรูปแบบนี้ เป็นการเปลี่ยนแปลงโครงสร้างภายนอกของคลาส ประกอบด้วยการเปลี่ยนแปลงชื่อและเปลี่ยนแปลงขอบเขต

- เปลี่ยนแปลงชื่อ มีลักษณะเช่นเดียวกับการลบคลาส คือจะไม่ปรากฏคลาสเดิมอีกต่อไป ดังนั้นจึงเหมือนลบคลาสดังกล่าวไป
- เปลี่ยนแปลงขอบเขต มี 2 แบบคือ จาก public ไปเป็น protected และจาก protected ไปเป็น public ซึ่งผลกระทบจะเกิดขึ้นในกรณีที่เปลี่ยนแปลงจาก public ไปเป็น protected เท่านั้นส่วนกรณีที่เปลี่ยนแปลงจาก protected ไป

เป็น public นั้นจะไม่มีผลกระทบ สำหรับขอบเขตแบบ private นั้น คลาสจะไม่สามารถประกาศใช้งานได้

2.1.4.3 การเปลี่ยนแปลงที่เมทธอด

การเปลี่ยนแปลงที่เมทธอด สามารถแสดงได้ดังรูปที่ 2.2

รูปที่ 2.2 รูปแบบการเปลี่ยนแปลงที่เมทธอด

1. เปลี่ยนแปลงลายเซ็นต์ (Signature change) ลายเซ็นต์ [10] ประกอบด้วยชื่อเมทธอด จำนวนพารามิเตอร์และประเภทของพารามิเตอร์ ตัวอย่างการเปลี่ยนแปลงเช่น การเพิ่มหรือลดพารามิเตอร์ การเปลี่ยนชื่อเมทธอด เป็นต้น ซึ่งจะส่งผลกระทบไปยังเมทธอดหรือคลาสที่มีการอ้างอิงเมทธอดนี้

2. เปลี่ยนแปลงสัจพจน์ (Axiom change) การเปลี่ยนแปลงพรีคอนดิชันโพสคอนดิชันหรือค่าความจริง ซึ่งอาจจะเป็นการเปลี่ยนแปลงพฤติกรรมหรือความหมายของเมทธอด ซึ่งอาจจะมีผลกระทบกับคลาสที่อ้างอิงเมทธอดนี้หรืออาจไม่มีผลกระทบก็ได้

3. เปลี่ยนแปลงการอิมพลีเมนต์ (Implementation change) การเปลี่ยนแปลงแบบนี้จะกระทบกับรายละเอียดของการอิมพลีเมนต์ แต่ไม่กระทบกับอินเทอร์เฟสหรือส่วนที่ใช้ติดต่อกับเมทธอด ซึ่งการเปลี่ยนแปลงนี้อาจจะมีผลกระทบกับคลาสที่อ้างอิงเมทธอดนี้หรืออาจจะไม่มีผลกระทบก็ได้

4. เปลี่ยนแปลงขอบเขต (Scope change) เป็นการเปลี่ยนแปลงขอบเขตการเข้าถึงหรือการอ้างอิงเมทธอด สามารถเปลี่ยนได้ 6 กรณี คือ

- เปลี่ยนจาก public เป็น private ทุกคลาสที่เรียกใช้เมทธอดนี้จะได้รับผลกระทบ
- เปลี่ยนจาก public เป็น protected คลาสที่เรียกใช้ทั้งหมดจะถูกกระทบ ยกเว้นคลาสลูกจะไม่ถูกกระทบ
- เปลี่ยนจาก protected เป็น private คลาสลูกจะถูกกระทบ
- เปลี่ยนจาก protected เป็น public คลาสต่าง ๆ จะไม่ได้รับผลกระทบ
- เปลี่ยนจาก private เป็น public คลาสต่าง ๆ จะไม่ได้รับผลกระทบ
- เปลี่ยนจาก private เป็น protected คลาสต่าง ๆ จะไม่ได้รับผลกระทบ

5. ลบเมทธอด ทุกคลาสที่เรียกใช้เมทธอดนี้จะได้รับผลกระทบ

6. เพิ่มเมทธอด เมื่อมีการเพิ่มเมทธอด จะยังไม่มีการถูกเรียกใช้งาน แต่คลาสลูกจะรู้ว่าจะมีเมทธอดใหม่เพิ่มขึ้น จึงไม่มีผลกระทบกับคลาสใด ๆ

7. เปลี่ยนแปลงชนิดของค่าส่งกลับ จะทำให้เมทธอดที่เรียกใช้ได้รับผลกระทบ

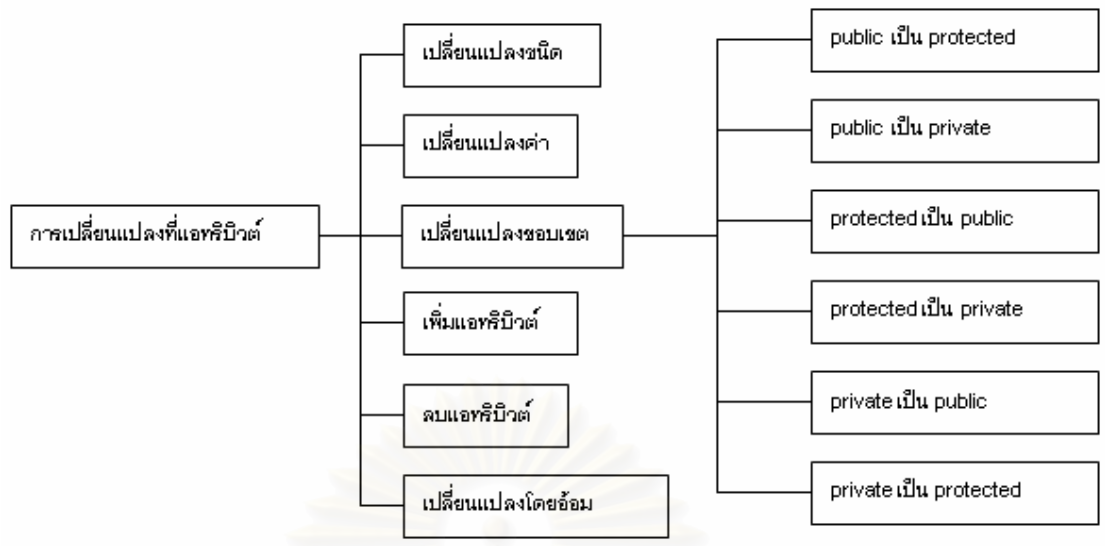
8. เปลี่ยนแปลงโดยอ้อม เกิดจากเมทธอดเรียกใช้แอทริบิวต์หรือเมทธอดอื่นที่ได้รับผลกระทบจากการเปลี่ยนแปลงเมทธอด จึงอาจเกิดผลกระทบขึ้นได้

2.1.4.4 การเปลี่ยนแปลงที่แอทริบิวต์

การเปลี่ยนแปลงที่แอทริบิวต์ เป็นการเปลี่ยนแปลงที่เกิดขึ้นกับตัวแปรที่ใช้ในการเก็บข้อมูลของวัตถุ การเปลี่ยนแปลงที่แอทริบิวต์สามารถแสดงได้ดังรูปที่ 2.3

1. เปลี่ยนแปลงชนิด (Type change) จะทำให้คลาสที่อ้างอิงถึงแอทริบิวต์นี้ได้รับผลกระทบ

2. เปลี่ยนแปลงค่า (Value change) การเปลี่ยนแปลงค่าของแอทริบิวต์อาจจะทำให้เกิดผลกระทบกับคลาสอื่นหรือไม่เกิดผลกระทบก็ได้ ขึ้นอยู่กับว่าค่าที่เปลี่ยนนั้นทำให้สถานะของวัตถุเปลี่ยนแปลงหรือไม่ หากเปลี่ยนอาจจะทำให้เส้นทาง (Path) ในการทำงานเปลี่ยนแปลงไป ซึ่งจะส่งผลกระทบกับคลาสนั้นได้



รูปที่ 2.3 รูปแบบการเปลี่ยนแปลงที่แอทริบิวต์

3. เปลี่ยนแปลงขอบเขตของแอทริบิวต์ จะเป็นเช่นเดียวกับ เปลี่ยนแปลงขอบเขตของเมทธอด สามารถเปลี่ยนได้ 6 กรณีเช่นกันดังแสดงในการเปลี่ยนแปลงขอบเขตของเมทธอด

4. เพิ่มแอทริบิวต์ จะยังไม่มีการอ้างอิงหรือเข้าถึงแอทริบิวต์นั้น แต่คลาสลูกจะรับรู้ว่ามีแอทริบิวต์ใหม่เกิดขึ้น ซึ่งยังไม่มีผลกระทบกับคลาสใดเนื่องจากยังไม่มีการเรียกใช้งาน

5. ลบแอทริบิวต์ จะมีความคล้ายคลึงกับการลบเมทธอดเช่นกัน ซึ่งผลกระทบจะมีมากหรือน้อยขึ้นอยู่กับขอบเขตของแอทริบิวต์นั้นๆ

6. เปลี่ยนแปลงโดยอ้อม เกิดจากแอทริบิวต์เรียกใช้แอทริบิวต์หรือเมทธอดอื่นที่ได้รับผลกระทบจากการเปลี่ยนแปลงแอทริบิวต์ จึงอาจเกิดผลกระทบขึ้นหรือไม่ก็ได้

2.1.5 **การวิเคราะห์สหสัมพันธ์อย่างง่าย** (Simple correlation analysis) [11][12][13]

การวิเคราะห์สหสัมพันธ์อย่างง่าย เป็นวิธีการวิเคราะห์เพื่อศึกษาว่า ตัวแปร 2 ตัวมีความสัมพันธ์กันมากน้อยเพียงใด ซึ่งวิธีในการวิเคราะห์สหสัมพันธ์มีหลายวิธี ขึ้นอยู่กับชนิดข้อมูลของตัวแปรทั้งสองตัวที่ต้องการวิเคราะห์ ถ้าข้อมูลของตัวแปรเป็นข้อมูลแบบอัตราส่วน (Ratio scale) จะใช้การวิเคราะห์แบบเพียร์สัน ในที่นี้จะกล่าวถึงวิธีนี้เท่านั้น

การวิเคราะห์สหสัมพันธ์แบบเพียร์สัน เป็นการหาความสัมพันธ์ระหว่างตัวแปรเชิงปริมาณสองตัวที่มีความสัมพันธ์กันในรูปแบบเชิงเส้น (Linear relationship) สำหรับค่าหรือปริมาณของความสัมพันธ์ระหว่างตัวแปรทั้งสองตัวนั้น จะเรียกว่า สัมประสิทธิ์สหสัมพันธ์ (Correlation coefficient) โดยมีสมการดังนี้

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

โดย

r คือ สัมประสิทธิ์สหสัมพันธ์

x_i คือ ตัวแปรตัวที่ 1 หรือตัวแปรอิสระ ลำดับที่ i เมื่อ $i = 1,2,\ldots,n$

y_i คือ ตัวแปรตัวที่ 2 หรือตัวแปรตาม ลำดับที่ i เมื่อ $i = 1,2,\ldots,n$

n คือ จำนวนข้อมูลทั้งหมด

ค่า r ที่ได้จากการคำนวณจะมีค่าตั้งแต่ -1 ถึง 1 ความหมายของค่า r มีดังนี้

1. ค่า r เป็นลบ แสดงว่า x และ y มีความสัมพันธ์ในทิศทางตรงกันข้าม คือ ถ้าค่า x เพิ่มขึ้น ค่า y จะลดลง แต่ถ้าหากค่า x ลดลง ค่า y จะเพิ่มขึ้น

2. ค่า r เป็นบวก แสดงว่า x และ y มีความสัมพันธ์ในทิศทางเดียวกัน คือ ถ้าค่า x เพิ่มขึ้นหรือลดลง ค่า y ก็จะเพิ่มขึ้นหรือลดลงตามไปตามด้วย

3. ถ้าค่า r มีค่าเข้าใกล้ 1 หมายถึง x และ y มีความสัมพันธ์กันมาก โดยมีความสัมพันธ์ในทิศทางเดียวกัน

4. ถ้าค่า r มีค่าเข้าใกล้ -1 หมายถึง x และ y มีความสัมพันธ์กันมาก แต่มีความสัมพันธ์กันในทิศทางตรงกันข้าม

5. ถ้า r = 0 แสดงว่า x และ y ไม่มีความสัมพันธ์กัน

6. ถ้า r เข้าใกล้ 0 แสดงว่า x และ y มีความสัมพันธ์กันน้อย

2.1.6 **การวิเคราะห์ความถดถอย** (Regression analysis) [11][12][13]

การวิเคราะห์ความถดถอย[13] เป็นวิธีที่ใช้หาสมการพีชคณิตเพื่อนำมาเป็นตัวกำหนดความสัมพันธ์ระหว่างปัจจัยที่ต้องการศึกษา ซึ่งต้องเป็นตัวแปรที่ต่อเนื่องกับปัจจัยที่เกี่ยวข้องหรือปัจจัยที่มีความสัมพันธ์กับปัจจัยที่ต้องการศึกษา ทั้งนี้เพื่อนำมาใช้ในการคาดคะเนหรือประมาณค่ากับตัวแปรที่สนใจ เมื่อทราบค่าปัจจัยที่เกี่ยวข้อง หรือเป็นการหากฎเกณฑ์ที่สามารถใช้ในการพยากรณ์หรือคาดคะเน เกี่ยวกับตัวแปรที่ต้องการศึกษาโดยอาศัยความรู้เกี่ยวกับค่าของตัวแปรที่เกี่ยวข้อง ในกรณีต้องการหาความสัมพันธ์ระหว่างตัวแปรที่ต้องการศึกษาหลาย ๆ ตัว (ตัวแปรอิสระ) กับตัวแปรที่สนใจ (ตัวแปรตาม) จะเรียกว่า การวิเคราะห์ความถดถอยเชิงซ้อน รูปแบบสมการความถดถอยเชิงซ้อน คือ

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_i X_i + \ldots + \beta_n X_n + e$$

ค่าประมาณของ Y คือ

$$\hat{Y} = a + b_1 x_1 + b_2 x_2 + \ldots + b_i x_i + \ldots + b_n x_n$$

โดย Y คือตัวแปรตาม

$\hat{Y}$ คือตัวแปรตามที่ได้จากการประมาณ

β_i คือสัมประสิทธิ์ความถดถอยเชิงส่วน (Partial regression coefficient) เมื่อ $i = 1,2,\ldots,n$

b_i คือค่าประมาณของสัมประสิทธิ์ความถดถอยเชิงส่วน เมื่อ $i = 1,2,\ldots,n$

a คือค่าคงที่

x_i คือตัวแปรอิสระของสมการความถดถอย เมื่อ $i = 1,2,\ldots,n$

e คือความผิดพลาด

การวิเคราะห์ความถดถอยมีขั้นตอน ดังนี้

1. นำข้อมูล Y กับ X_i มาทำการประมาณค่าสัมประสิทธิ์ความถดถอย คือ การหาค่า b_i โดยใช้วิธีกำลังสองน้อยที่สุด ซึ่งจะได้สมการถดถอย

2. ทดสอบสัมประสิทธิ์ความถดถอยของสมการที่ได้ เป็นการทดสอบว่าตัวแปรอิสระ X_i กับตัวแปรตาม Y มีความสัมพันธ์กันจริงหรือไม่ โดยใช้การวิเคราะห์ความแปรปรวนจำแนกแบบทางเดียวเป็นตัวทดสอบ เพื่อทดสอบว่า $\beta_1 = \beta_2 = \ldots = \beta_n = 0$ หรือไม่ หากพบว่ามีค่า β_i อย่างน้อยหนึ่งค่าที่ไม่เท่ากับ 0 จะถือว่าสมการที่ได้นั้น X_i กับ Y มีความสัมพันธ์กันจริง

3. เมื่อทราบแล้วว่า X_i กับ Y มีความสัมพันธ์กันจริง จะต้องทดสอบว่า มี β_i ใดบ้างที่เป็น 0 ซึ่งถ้ามี β_i ใดที่เป็น 0 ค่า X_i ตัวนั้นจะถูกตัดทิ้งเนื่องจากไม่มีความสัมพันธ์กับค่า Y เมื่อทดสอบเรียบร้อยแล้วจะได้สมการที่ใช้ในการประมาณค่า Y

4. เมื่อได้สมการประมาณค่า Y แล้วจึงหาค่าสัมประสิทธ์สหสัมพันธ์เชิงซ้อน (Multiple Coefficient of Correlation : r) โดยการถอดรากที่สองของค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อน (Mutliple Coefficient of Determination : R^2 หรือ r^2) เพื่อจะได้ทราบว่าสมการที่ได้ตัวแปรอิสระ X_i และตัวแปรตาม Y มีความสัมพันธ์กันมากน้อยเพียงใด ถ้า r มีค่าเข้าใกล้ 1 แสดงว่าตัวแปรอิสระ X_i กับตัวแปรตาม Y มีความสัมพันธ์กันมาก แต่ถ้า r มีค่าเข้าใกล้ 0 แสดงว่า X_i กับ Y มีความสัมพันธ์กันน้อย ส่วน r^2 จะเป็นสัดส่วนหรือเปอร์เซ็นต์ที่ตัวแปรอิสระ X_i สามารถอธิบายความเปลี่ยนแปลงของตัวแปรตาม Y ได้

เนื่องจากการวิเคราะห์ความถดถอยจะศึกษาถึงความสัมพันธ์ระหว่างตัวแปรอิสระและตัวแปรตาม ดังนั้นการเลือกตัวแปรอิสระเข้าในสมการถดถอยจึงมีความสำคัญมาก ถ้าหากเลือกตัวแปรอิสระเข้าสมการได้ดี จะทำให้ได้สมการถดถอยที่ดีด้วย เทคนิคในการเลือกตัวแปรอิสระที่เหมาะสมสำหรับเข้าสมการความถดถอย มีดังนี้

1. สมการความถดถอยทั้งหมดที่เป็นไปได้ (All possible regression) วิธีนี้จะสร้างสมการความถดถอยจากตัวแปรอิสระทุกทางที่เป็นไปได้ เช่น ถ้ามีตัวแปรอิสระ 3 ตัว จะสร้างสมการที่เป็นไปได้ทั้งหมด 8 สมการ จากนั้นจะเลือกสมการที่มีค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อนสูงที่สุด (r^2 ใกล้ 1 มากที่สุด) และมีค่าความคลาดเคลื่อนมาตรฐานน้อยที่สุด

2. การกำจัดแบบย้อนกลับ (Backward Elimination) จะสร้างสมการถดถอยที่มีตัวแปรอิสระทั้งหมดในสมการ แล้วจะพิจารณาตัดตัวแปรอิสระที่ไม่มีความสัมพันธ์กับ Y ออกไปทีละตัวจนกระทั่งไม่มีตัวแปรอิสระตัวใดถูกตัดออกไป สมการถดถอยที่ประกอบด้วยตัวแปรอิสระที่เหลืออยู่จึงเป็นสมการความถดถอยที่เหมาะสม เนื่องจากตัวแปรอิสระทุกตัวในสมการความถดถอยที่เหมาะสมเป็นตัวแปรที่มีความสัมพันธ์กับตัวแปรตามอย่างมีนัยสำคัญ

3. การเลือกไปข้างหน้า (Forward Selection) จะตรงข้ามกับแบบการกำจัดแบบย้อนกลับ คือจะเลือกตัวแปรอิสระที่มีความสัมพันธ์กับตัวแปรตามเข้าสมการครั้งละตัว โดยเลือกตัวแปรอิสระที่มีความสัมพันธ์กับตัวแปรตามมากที่สุด เข้าสมการเป็นอันดับแรก ทำไปจนกว่าจะ

ไม่มีตัวแปรอิสระตัวใดมีความสัมพันธ์กับ Y อย่างมีนัยสำคัญ ก็จะได้สมการความถดถอยที่เหมาะสม

4. ความถดถอยแบบสเตปไวส์ (Stepwise Regression) เป็นวิธีที่เลือกตัวแปรอิสระเข้าสมการถดถอยโดยใช้หลักเกณฑ์ทั้งวิธีการกำจัดแบบย้อนกลับและการเลือกไปข้างหน้ารวมกัน ซึ่งมี 2 ขั้นตอน คือ

ขั้นตอนที่ 1 ใช้หลักเกณฑ์ของการเลือกไปข้างหน้า โดยการเลือกตัวแปรอิสระ 1 ตัวที่มีความสัมพันธ์กับตัวแปรตามอย่างมีนัยสำคัญมากที่สุดเข้าสมการ

ขั้นตอนที่ 2 ใช้ทั้งหลักเกณฑ์ของการเลือกไปข้างหน้าและการกำจัดแบบย้อนกลับ โดยเลือกตัวแปรอิสระที่มีความสัมพันธ์กับตัวแปรตามที่มีนัยสำคัญมากที่สุดตัวถัดไปเข้าสมการ แล้วทดสอบสมมติฐานเพื่อหาว่ามีค่าสัมประสิทธิ์ของตัวแปรอิสระตัวใดที่มีค่าเป็น 0 ถ้ามีจึงตัดตัวแปรอิสระตัวนั้นออกจากสมการ

หลังจากนั้นจึงทำซ้ำขั้นตอนที่ 2 โดยเลือกตัวแปรอิสระที่เหลือ เข้าสมการ ขณะเดียวกันก็อาจจะมีการตัดตัวแปรอิสระที่อยู่ในสมการออก ทำจนกระทั่งไม่มีตัวแปรอิสระใดที่ควรนำเข้าสมการและไม่มีตัวแปรอิสระใดที่ควรตัดออกจากสมการจึงหยุด ก็จะได้สมการถดถอยที่เหมาะสม

ซึ่งวิธีการเลือกตัวแปรอิสระวิธีนี้ สามารถป้องกันการเกิดปัญหา ความสัมพันธ์ระหว่างตัวแปรอิสระ (Multicollinearity) เนื่องจากถ้าตัวแปรอิสระมีความสัมพันธ์กันแล้วจะทำให้สมการถดถอยที่ได้มีความคลาดเคลื่อน ซึ่งในงานวิจัยนี้ได้เลือกการสร้างสมการถดถอยแบบสเตปไวส์มาใช้ในการสร้างโมเดลทำนายเสถียรภาพของเมทธอด

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 การสำรวจมาตรวัดสำหรับเสถียรภาพเชิงตรรกะของการออกแบบเชิงวัตถุ (Investigation of Metrics for Object-Oriented Design Logical Stability) [2] โดย Mahmoud O.Elish และ David Rine

ในงานวิจัยนี้ได้ทดสอบมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer ซึ่งประกอบด้วย WMC (Weighted methods per class) DIT (Depth of Inheritance tree) CBO (Coupling between object class) RFC (Response for class) LCOM (Lack of cohesion in methods) และ NOC (Number of children) ว่าเป็นเครื่องบ่งชี้ถึงเสถียรภาพด้านตรรกะ (Logical stability) ของคลาสในโปรแกรมเชิงวัตถุ ได้ดีหรือไม่ โดยมีการตั้งสมมติฐานในการทดลองสอง

ประการคือ มาตรวัด WMC DIT CBO RFC และ LCOM มีความสัมพันธ์แบบผกผันกับเสถียรภาพของคลาสและสมมติฐานที่สองคือ NOC ไม่มีความสัมพันธ์กับเสถียรภาพของคลาส โดยใช้โปรแกรมที่พัฒนาด้วยภาษาจาวาเป็นหน่วยตัวอย่างในการทดลอง สำหรับขั้นตอนในการทดลองมี 3 ขั้นตอนคือ

1. เก็บค่ามาตรวัดของ Chidamber และ Kemerer ทั้ง 6 มาตรวัดจากคลาสของแต่ละโปรแกรม

2. แก้ไขโปรแกรมครั้งละหนึ่งอย่างเท่านั้น โดยการแก้ไขในแต่ละครั้งจะแก้ไขกับโปรแกรมต้นฉบับ สำหรับตำแหน่งการแก้ไขจะแก้ที่แอทริบิวต์และเมทธอดของคลาส แล้วเก็บคลาสที่มีผลกระทบจากการแก้ไขในแต่ละครั้ง เมื่อแก้ไขครบทุกคลาสของโปรแกรมแล้ว จึงคำนวณหาเสถียรภาพด้านตรรกะของคลาส

3. วิเคราะห์หาความสัมพันธ์ระหว่างค่ามาตรวัดทั้ง 6 มาตรวัดและเสถียรภาพด้านตรรกะของคลาสในโปรแกรมโดยใช้การวิเคราะห์สหสัมพันธ์ที่ความเชื่อมั่น 95% แล้วทำการสรุปผล

จากผลการทดลองของงานวิจัยนี้สอดคล้องกับสมมติฐานที่ได้ตั้งไว้คือ WMC DIT CBO RFC LCOM มีความสัมพันธ์แบบผกผันกับเสถียรภาพของคลาส กล่าวคือหากมาตรวัดมีค่ามาก เสถียรภาพของคลาสจะมีค่าน้อย และ NOC ไม่มีความสัมพันธ์กับเสถียรภาพของคลาส ในงานวิจัยนี้ได้ใช้มาตรวัดมาวัดเสถียรภาพของคลาสเปรียบเทียบกับเสถียรภาพของคลาสที่ได้จากการเปลี่ยนแปลงรหัสโปรแกรมโดยตรง แต่ไม่ได้พิจารณาถึงเสถียรภาพของเมทธอด ดังนั้นผู้ทำวิทยานิพนธ์จึงมีแนวคิดในการหามาตรวัดเชิงวัตถุมาวัดเสถียรภาพของเมทธอด โดยอาศัยรูปแบบการเปลี่ยนแปลงที่จากงานวิจัยชิ้นนี้มาประยุกต์ใช้

2.2.2 อัลกอริทึมในการวิเคราะห์ผลกระทบของการเปลี่ยนแปลงต่อซอฟต์แวร์เชิงวัตถุ (Algorithmic Analysis of the Impact of Changes to Object-Oriented Software) [9] โดย Li Li และ A. Jefferson Offutt

งานวิจัยนี้ได้ทำการรวบรวมรูปแบบของการเปลี่ยนแปลงที่เกิดขึ้นในซอฟต์แวร์เชิงวัตถุ โดยแบ่งเป็นการเปลี่ยนแปลงที่เกิดกับแอทริบิวต์ การเปลี่ยนแปลงที่เกิดขึ้นกับเมทธอดและการเปลี่ยนแปลงที่เกิดขึ้นกับคลาส นอกจากนี้ยังได้ทำการแบ่งลักษณะของการเปลี่ยนแปลงตามคลาสที่ถูกกระทบจากการเปลี่ยนแปลงเป็น 5 แบบ คือ

1. กระทบทั้งหมด (Contaminate_all) เป็นการเปลี่ยนแปลงที่มีผลกระทบกับแอทริบิวต์และเมทธอดในทุก ๆ คลาสที่เกี่ยวข้องกับคลาสที่เกิดการเปลี่ยนแปลง

2. กระทบปัจจุบัน (Contaminate_current) เป็นการเปลี่ยนแปลงที่มีผลกระทบกับสมาชิกข้อมูลและเมทธอดในคลาสที่เกิดการเปลี่ยนแปลงนั้น

3. กระทบลูก (Contaminate_children) เป็นการเปลี่ยนแปลงที่มีผลกับคลาสที่สืบทอดคุณสมบัติมาจากคลาสที่เกิดการเปลี่ยนแปลง

4. กระทบไคลเอนต์ (Contaminate_client) เป็นการเปลี่ยนแปลงที่มีผลกับคลาสที่เรียกใช้คลาสที่เกิดการเปลี่ยนแปลง

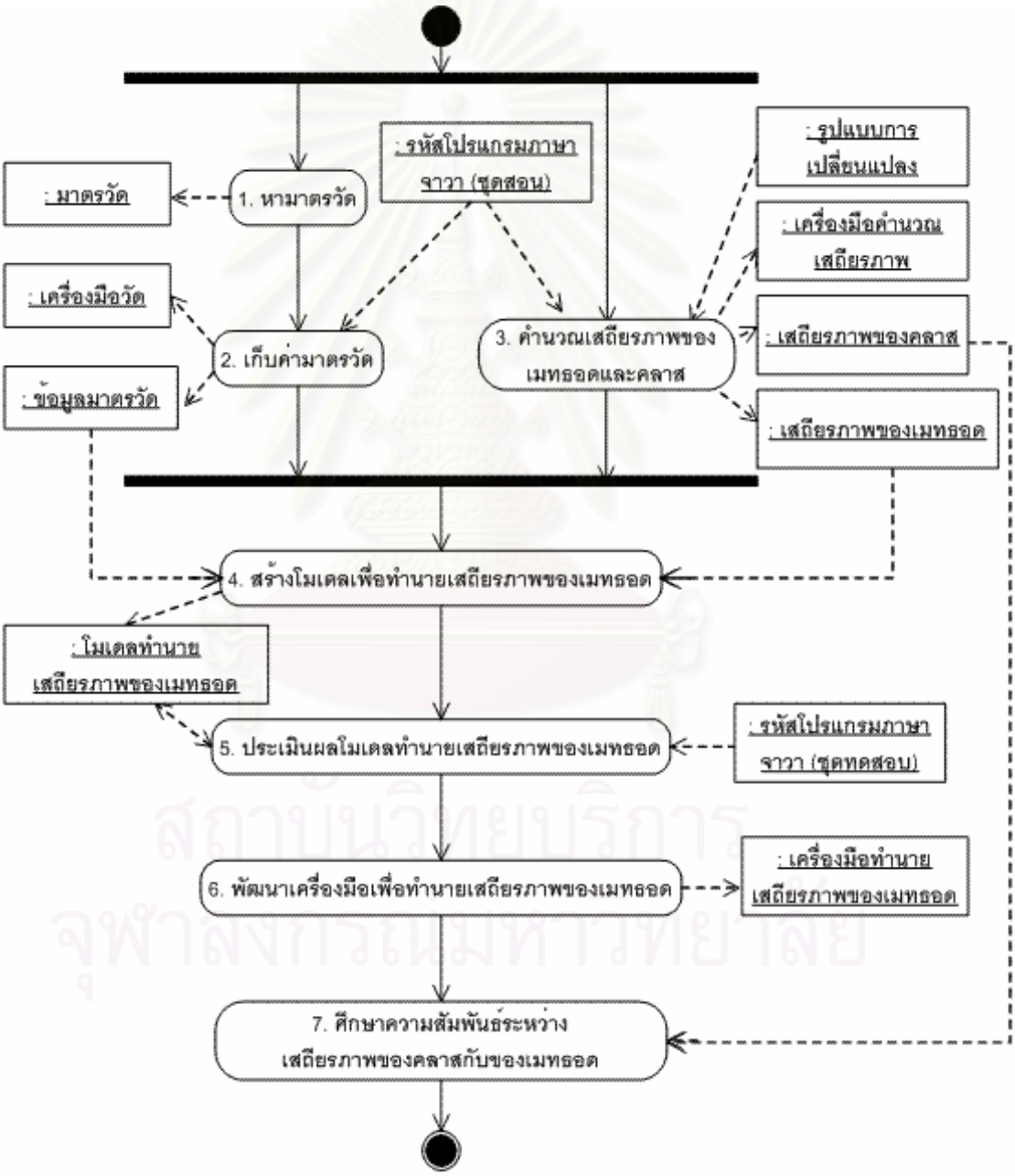
5. ไม่กระทบ (Contaminate_none) เป็นการเปลี่ยนแปลงที่ไม่มีผลกระทบกับคลาสใด ๆ ทั้งสิ้น

จากงานวิจัยชิ้นนี้ ผู้ทำวิทยานิพนธ์ได้นำรูปแบบของการเปลี่ยนแปลงที่เกิดขึ้นกับซอฟต์แวร์เชิงวัตถุมาใช้ เพื่อหาเสถียรภาพของเมทธอด

บทที่ 3

ขั้นตอนการสร้างโมเดลทำนายเสถียรภาพของเมทธอด

ในบทนี้ นำเสนอขั้นตอนการสร้างโมเดลทำนายเสถียรภาพของเมทธอด ซึ่งแสดงเป็นแผนภาพกิจกรรมได้ดังรูปที่ 3.1



รูปที่ 3.1 ขั้นตอนการวิจัย

จากรูปที่ 3.1 มีทั้งหมด 7 ขั้นตอน เริ่มจากการหามาตรวัดเกี่ยวกับเมทธอด เพื่อนำมาใช้สร้างโมเดล ซึ่งมีรายละเอียดในหัวข้อ 3.1 เมื่อได้มาตรวัดแล้วจึงเก็บค่ามาตรวัดจากรหัสโปรแกรม โดยมีรายละเอียดในหัวข้อ 3.2 สำหรับรหัสโปรแกรมที่ใช้ในขั้นตอนที่ 2 จะนำมาคำนวณเสถียรภาพของเมทธอดและคลาส ซึ่งมีรายละเอียดในหัวข้อ 3.3 เพื่อนำไปใช้ในขั้นตอนที่ 4 และขั้นตอนที่ 7 สำหรับขั้นตอนที่ 4 อยู่ในหัวข้อ 3.4 จะเป็นการสร้างโมเดลทำนายเสถียรภาพของเมทธอดโดยใช้ข้อมูลในขั้นตอนที่ 2 และขั้นตอนที่ 3 มาสร้าง เมื่อได้โมเดลทำนายเสถียรภาพแล้วจึงประเมินความถูกต้องของโมเดล ซึ่งมีรายละเอียดในหัวข้อ 3.5 หลังจากผ่านขั้นตอนที่ 5 แล้วจึงสร้างเครื่องมือเพื่อใช้ในการทำนายเสถียรภาพของเมทธอด ซึ่งมีรายละเอียดในหัวข้อ 3.6 ส่วนขั้นตอนสุดท้าย เป็นการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด ซึ่งมีรายละเอียดในหัวข้อ 3.7

3.1 การเลือกมาตรวัด

มาตรวัดเชิงวัตถุที่นำมาใช้ในงานวิจัยนี้เป็นมาตรวัดที่ใช้วัดกับเมทธอด ซึ่งเป็นมาตรวัดพื้นฐานที่เป็นที่รู้จักโดยทั่วไป โดยแบ่งมาตรวัดเป็น 3 กลุ่ม คือ

1. มาตรวัดขนาดของเมทธอด ประกอบด้วย
   - มาตรวัดจำนวนพารามิเตอร์(Param) นับจำนวนพารามิเตอร์ของแต่ละเมทธอด
   - มาตรวัดจำนวนตัวแปรแบบโลคอล (LocalVar) นับจำนวนตัวแปรที่ประกาศใช้งานในเมทธอด
   - มาตรวัดจำนวนสเตทเมนต์ (NOS) นับจำนวน สเตทเมนต์ในเมทธอด
   - มาตรวัดจำนวนบรรทัดของรหัสโปรแกรม (LOC) จะนับจำนวนบรรทัดของรหัสโปรแกรมของเมทธอด โดยจะไม่นับบรรทัดว่าง บรรทัดที่เป็นคอมเมนต์และบรรทัดที่เป็นบล็อก โดย LOC = NOS+LocalVar+การประกาศเมทธอด

2. มาตรวัดความซับซ้อนของเมทธอด ประกอบด้วย
   - มาตรวัดไซโคลเมติกของแมคเคบ (V(G)) นับจากจำนวนเงื่อนไขบวกหนึ่ง
   - มาตรวัดความซับซ้อนของเมทธอด(MCX) วัดจากจำนวนและชนิดของข้อความที่มีในเมทธอด ซึ่งมีสมการดังนี้

$$MCX = \sum_{i=1}^{6} (w_i * | m_i |)$$

โดย

MCX คือ ความซับซ้อนของเมทธอด

w_i คือ ค่าถ่วงน้ำหนักของข้อความชนิดที่ i เมื่อ $i=1,2,..,6$

$|m_i|$ คือ จำนวนข้อความชนิดที่ i เมื่อ $i=1,2,..,6$

ชนิดของข้อความและค่าถ่วงน้ำหนักของข้อความแต่ละชนิด แสดงได้ดังตารางที่ 3.1

ตารางที่ 3.1 ค่าถ่วงน้ำหนักของข้อความแต่ละประเภท [6]

ชนิดข้อความ	ค่าถ่วงน้ำหนัก
ยูนารีเอ็กเพรสชัน (Unary expressions)	1.0
ไบนารีเอ็กเพรสชัน (Binary expressions)	2.0
แอสซายน์เมนต์ (Assignments)	0.5
การส่งข้อความ (Messages send)	3.0
พารามิเตอร์ (Parameters)	0.3
ตัวแปรชั่วคราว (Temporary variables)	0.5

3. มาตรวัดการเข้าคู่ระหว่างเมทธอด ประกอบด้วย

- มาตรวัดการเข้าคู่กันระหว่างวัตถุ (CBO) คำนวณจากจำนวนคลาสที่ถูกใช้ในเมทธอด ซึ่งต้องเป็นคลาสที่ไม่ซ้ำกัน โดยจะพิจารณาทั้งในส่วนที่เป็นตัวแปรพารามิเตอร์ ตัวแปรแบบโลคอล ตัวแปรยกเว้น (exception variable) และตัวแปรอินสแตนท์
- มาตรวัดจำนวนการเรียกเมทธอด (NMI) คำนวณจากจำนวนเมทธอดที่ถูกเรียกใช้งานในเมทธอดนั้น โดยเมทธอดที่มีการเรียกใช้งานซ้ำจะนับเพียงครั้งเดียว สำหรับมาตรวัด NMI จะแบ่งเป็น มาตรวัดย่อยตามขอบเขตในการเข้าถึงเมทธอด [10] ที่เรียกใช้งานเป็น จำนวนการเรียกใช้เมทธอดแบบ public (NMIpub) จำนวนการเรียกใช้เมทธอดแบบ protected (NMIpro) จำนวนการเรียกใช้เมทธอดแบบ default (NMIdef) และการเรียกใช้เมทธอดแบบ private (NMIpri)

- มาตรวัดจำนวนการเรียกแอทริบิวต์ (NAI) คำนวณจากจำนวนแอทริบิวต์ที่ถูกเรียกใช้งานในเมทธอดนั้น ซึ่งแอทริบิวต์ที่มีการเรียกใช้งานซ้ำ จะนับเพียงครั้งเดียว สำหรับมาตรวัด NAI แบ่งเป็นมาตรวัดย่อยตามขอบเขตการเข้าถึงแอทริบิวต์ เช่นเดียวกับใน NMI คือ จำนวนการเรียกใช้แอทริบิวต์แบบ public (NAIpub) จำนวนการเรียกใช้แอทริบิวต์แบบ protected (NAIpro) จำนวนการเรียกใช้แอทริบิวต์แบบ default (NAIdef) และการเรียกใช้แอทริบิวต์แบบ private (NAIpri)

ตัวอย่างการนับมาตรวัด NAI และ NMI จาก ตารางที่ 3.2 เป็นตัวอย่างรหัสโปรแกรมซึ่งประกอบด้วย 4 คลาส คือ คลาส Main คลาส CX คลาส CY และคลาส CZ ในที่นี้ขอยกตัวอย่างการนับมาตรวัด NAI และ NMI ของเมทธอด test1 ของคลาส main เท่านั้น สำหรับเมทธอด test1 มีการเรียกใช้เมสเสจ x.y.mY1().mZ2(m1, x.getString( )) เพียงเมสเสจเดียว แต่ในเมสเสจนี้ประกอบด้วยเมสเสจย่อยหลายเมสเสจ ซึ่งการนับ NAI และ NMI จะนับเมสเสจย่อยต่าง ๆ เหล่านี้ด้วย ดังนั้นเมทธอด test1 จึงมีค่ามาตรวัดดังนี้

NAIpub = 2 คือ

- แอทริบิวต์ m1 ในคลาส Main
- แอทริบิวต์ y ในคลาส CX

NAIdef = 1 คือ

- แอทริบิวต์ x ในคลาส Main

NMIpub = 2 คือ

- เมทธอด mY1 ในคลาส CY
- เมทธอด mZ2 ในคลาส CZ

NMIpro = 1 คือ

- เมทธอด getString ในคลาส CX

3.2 การเก็บค่ามาตรวัด

หลังจากทำการเลือกมาตรวัดที่นำมาใช้งานเสร็จเรียบร้อยแล้ว จึงจัดการเก็บค่ามาตรวัดต่าง ๆ จากรหัสโปรแกรมซึ่งเป็นชุดที่ใช้ในการสร้างโมเดล สำหรับการเก็บค่ามาตรวัดนี้ได้พัฒนา

เครื่องมือช่วยในการเก็บมาตรวัดเพื่อสามารถเก็บมาตรวัดจากรหัสโปรแกรมได้โดยอัตโนมัติ โดยนำเครื่องมือ MTOOP2 [6] มาปรับปรุงและเพิ่มเติมมาตรวัด รายละเอียดในการพัฒนาเครื่องมือเก็บค่ามาตรวัด ได้อธิบายในบทที่ 5

ตารางที่ 3.2 ตัวอย่างรหัสโปรแกรมในการนับมาตรวัด NAI และ NMI

```
class CY  {
        CZ z ;
        public CY(){
                z = new CZ();
        }
        public CZ mY1(){
                return z;
        }
}
```

```
class CZ  {
        public void mZ2(int x,String str) {
                .
                .
                .
        }
}
```

```
public class Main  {
        public int m1;
        CX  x;
        public static void main(String[] args)     {
                Main m = new Main();
        }
        Main( ) {
                x = new CX();
                test1();
        }
        private void test1(){
                x.y.mY1( ).mZ2( m1, x.getString( ) );
        }
}
```

```
class CX  {
        public CY  y;
        public CX( ){
                y = new CY();
        }
        protected String getString ( ){
                return “str”;
        }
}
```

3.3 คำนวณเสถียรภาพของเมทธอดและคลาส

ในขั้นตอนนี้จะทำการปรับเปลี่ยนรหัสโปรแกรมที่เป็นชุดที่ใช้ในการสร้างโมเดล โดยรูปแบบของการเปลี่ยนแปลงและผลกระทบจากการเปลี่ยนแปลง จะมีรายละเอียดในหัวข้อ 3.3.1 สำหรับวิธีการคำนวณเสถียรภาพของเมทธอดและคลาส ได้อธิบายรายละเอียดในหัวข้อ 3.3.2 นอกจากนี้ในงานวิจัยนี้ได้สร้างเครื่องมือเพื่อคำนวณเสถียรภาพ ซึ่งมีรายละเอียดอยู่ในบทที่ 5

3.3.1 รูปแบบการเปลี่ยนแปลงและผลกระทบจากการเปลี่ยนแปลง

ในงานวิจัยนี้ จะพิจารณาการเปลี่ยนแปลงที่เกิดขึ้นที่แอทริบิวต์และเมทธอด โดยจะต้องเป็นการเปลี่ยนแปลงที่เป็นแบบสถิตย์และมีผลกระทบกับวากยสัมพันธ์ของโปรแกรม ซึ่งรูปแบบการเปลี่ยนแปลงและผลกระทบที่เกิดขึ้นแสดงดังตารางที่ 3.3 และ ตารางที่ 3.4 โดยได้มีการเพิ่มการเข้าถึงแอทริบิวต์และเมทธอดแบบ default เข้าไปเพื่อให้สอดคล้องกับลักษณะภาษาจาวาที่มีการเข้าถึงได้ 4 แบบ [10] คือ public protected private และไม่กำหนดขอบเขตการเข้าถึง (default) และเพื่อความสะดวกในการอ้างอิงถึงคลาสที่ได้รับผลกระทบจากการเปลี่ยนแปลง จึงกำหนดประเภทคลาสที่ถูกกระทบโดยปรับปรุงจาก [9] ดังนี้

1. คลาสทั้งหมด คือ ทุก ๆ คลาสที่มีความสัมพันธ์กับคลาสที่เกิดการเปลี่ยนแปลงจะได้รับผลกระทบ ซึ่งได้แก่ คลาสไคลเอนต์ คลาสลูกและคลาสปัจจุบัน
2. คลาสไคลเอนต์ คือ คลาสที่มีความสัมพันธ์กับคลาสที่เกิดการเปลี่ยนแปลง คือมีการเรียกใช้แอทริบิวต์หรือเมทธอดของคลาสที่เปลี่ยนแปลง ซึ่งจะรวมทั้งคลาสไคลเอนต์1 และคลาสไคลเอนต์2 แต่ไม่รวม คลาสลูกและคลาสปัจจุบัน
3. คลาสไคลเอนต์1 คือ คลาสที่มีความสัมพันธ์กับคลาสที่เกิดการเปลี่ยนแปลงและอยู่ในแพ็กเกจเดียวกับคลาสที่เปลี่ยนแปลง
4. คลาสไคลเอนต์2 คือ คลาสที่มีความสัมพันธ์กับคลาสที่เกิดการเปลี่ยนแปลงแต่อยู่คนละแพ็กเกจกับคลาสที่เปลี่ยนแปลง
5. คลาสลูก คือ คลาสที่มีการสืบทอดคุณสมบัติจากคลาสที่เกิดการเปลี่ยนแปลง โดยจะรวมคลาสลูก1 และคลาสลูก2
6. คลาสลูก1 คือ คลาสที่มีการสืบทอดคุณสมบัติจากคลาสที่เกิดการเปลี่ยนแปลงและอยู่ในแพ็กเกจเดียวกัน
7. คลาสลูก2 คือ คลาสที่มีการสืบทอดคุณสมบัติจากคลาสที่เกิดการเปลี่ยนแปลงแต่อยู่คนละแพ็กเกจกับคลาสที่สืบทอดนั้น
8. คลาสปัจจุบัน คือ คลาสที่เกิดการเปลี่ยนแปลง

ตารางที่ 3.3 รูปแบบของการเปลี่ยนแปลงแอทริบิวต์และผลกระทบจากการเปลี่ยนแปลง

ลักษณะการเปลี่ยนแปลงแอทริบิวต์	**ลักษณะเพิ่มเติม**	**ประเภทคลาสที่ถูกกระทบ**
เปลี่ยนชนิด	แอทริบิวต์เป็นแบบ public	คลาสทั้งหมด
	แอทริบิวต์เป็นแบบ protected	คลาสลูก คลาสไคลเอนต์1 คลาสปัจจุบัน
	แอทริบิวต์เป็นแบบ default	คลาสลูก1 คลาสไคลเอนต์1 คลาสปัจจุบัน
	แอทริบิวต์เป็นแบบ private	คลาสปัจจุบัน
การลบแอทริบิวต์	แอทริบิวต์เป็นแบบ public	คลาสทั้งหมด
	แอทริบิวต์เป็นแบบ protected	คลาสลูก คลาสไคลเอนต์1 คลาสปัจจุบัน
	แอทริบิวต์เป็นแบบ default	คลาสลูก1 คลาสไคลเอนต์1 คลาสปัจจุบัน
	แอทริบิวต์เป็นแบบ private	คลาสปัจจุบัน
เปลี่ยนขอบเขต	เปลี่ยนจาก public เป็น protected	คลาสไคลเอนต์2
	เปลี่ยนจาก public เป็น default	คลาสลูก2 คลาสไคลเอนต์2
	เปลี่ยนจาก public เป็น private	คลาสไคลเอนต์ คลาสลูก
	เปลี่ยนจาก protected เป็น default	คลาสลูก2
	เปลี่ยนจาก protected เป็น private	คลาสลูก คลาสไคลเอนต์1
	เปลี่ยนจาก default เป็น private	คลาสลูก1 คลาสไคลเอนต์1

ตารางที่ 3.4 รูปแบบของการเปลี่ยนแปลงเมทธอดและผลกระทบจากการเปลี่ยนแปลง

ลักษณะการเปลี่ยนแปลงเมทธอด	**ลักษณะเพิ่มเติม**	**ประเภทคลาสที่ถูกกระทบ**
เปลี่ยนลายเซ็นต์	เมทธอดเป็นแบบ public	คลาสทั้งหมด
	เมทธอดเป็นแบบ protected	คลาสลูก คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ default	คลาสลูก1 คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ private	คลาสปัจจุบัน
เปลี่ยนขอบเขต	เปลี่ยนจาก public เป็น protected	คลาสไคลเอนต์2
	เปลี่ยนจาก public เป็น default	คลาสไคลเอนต์2 คลาสลูก2
	เปลี่ยนจาก public เป็น private	คลาสไคลเอนต์ คลาสลูก
	เปลี่ยนจาก protected เป็น default	คลาสลูก2
	เปลี่ยนจาก protected เป็น private	คลาสลูก คลาสไคลเอนต์1
	เปลี่ยนจาก default เป็น private	คลาสลูก1 คลาสไคลเอนต์1
การลบเมทธอด	เมทธอดเป็นแบบ public	คลาสทั้งหมด
	เมทธอดเป็นแบบ protected	คลาสลูก คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ default	คลาสลูก1 คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ private	คลาสปัจจุบัน

ตารางที่ 3.4 รูปแบบของการเปลี่ยนแปลงเมทธอดและผลกระทบจากการเปลี่ยนแปลง (ต่อ)

ลักษณะการเปลี่ยนแปลงเมทธอด	ลักษณะเพิ่มเติม	ประเภทคลาสที่ถูกกระทบ
เปลี่ยนชนิดของค่าส่งกลับ	เมทธอดเป็นแบบ public	คลาสทั้งหมด
	เมทธอดเป็นแบบ protected	คลาสลูก คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ default	คลาสลูก1 คลาสไคลเอนต์1 คลาสปัจจุบัน
	เมทธอดเป็นแบบ private	คลาสปัจจุบัน

3.3.2 การคำนวณเสถียรภาพ

เมื่อเปลี่ยนแปลงรหัสโปรแกรมที่แอทริบิวต์และเมทธอดกับทุกคลาสของโปรแกรม ตามรูปแบบในหัวข้อ 3.3.1 โดยแก้ไขทุกแอทริบิวต์ที่ประกาศเป็นโกลบอลและทุกเมทธอดในคลาส โดยเปลี่ยนแปลงครั้งละหนึ่งอย่างเท่านั้น แล้วเก็บผลกระทบที่เกิดขึ้นกับแต่ละเมทธอดและแต่ละคลาสของโปรแกรม หลังจากนั้นจึงนำจำนวนผลกระทบที่ได้ มาทำการคำนวณเสถียรภาพของเมทธอดและเสถียรภาพของคลาส โดยมีสมการดังนี้

$$Stability(C_i) = 1-\left(\frac{NCH(C_i)}{TCH}\right) \quad [2]$$

$$Stability(M_i) = 1-\left(\frac{NCH(M_i)}{TCH}\right)$$

โดย $Stability(C_i)$ คือ ค่าเสถียรภาพของคลาส C_i ซึ่ง $0 \le Stability(C_i) \le 1$ เมื่อ $i=1,2,..,n$

$Stability(M_i)$ คือ ค่าเสถียรภาพของเมทธอด M_i ซึ่ง $0 \le Stability(M_i) \le 1$ เมื่อ $i=1,2,..,n$

$NCH(C_i)$ คือ จำนวนครั้งที่คลาส C_i ถูกกระทบจากการเปลี่ยนแปลง เมื่อ $i=1,2,..,n$

$NCH(M_i)$ คือ จำนวนครั้งที่เมทธอด M_i ถูกกระทบจากการเปลี่ยนแปลง เมื่อ $i=1,2,..,n$

TCH คือ จำนวนครั้งที่ทำการเปลี่ยนแปลงทั้งหมด

สำหรับสมการคำนวณเสถียรภาพของเมทธอด ในวิทยานิพนธ์นี้ได้ประยุกต์มาจากสมการคำนวณเสถียรภาพของคลาสจาก[2] ตัวอย่างการคำนวณเสถียรภาพ แสดงได้ดังตารางที่ 3.5 และตารางที่ 3.6

ตารางที่ 3.5 ตัวอย่างรหัสโปรแกรมที่ต้องการคำนวณเสถียรภาพ

```
public class A{
   public int a1;
  public int mA1(){
      B b = new B();
      a1=  b.b1;
      return a1;   }
}
```

```
public class B{
   public int b1;
   public int mB1(){
      b1 = 10;
      return b1;
   }
}
```

```
public class C{
   public int c1;
   public int mC1(){
      B b = new B();
      c1 = b.mB1();
      return c1;   }
}
```

ตารางที่ 3.6 ตัวอย่างการคำนวณเสถียรภาพ

ครั้งที่	การเปลี่ยนแปลง	mA1()	mB1()	mC1()	class A	class B	class C
1	ลบแอทริบิวต์ a1	X					
2	ลบแอทริบิวต์ b1	X	X		X		
3	ลบแอทริบิวต์ c1			X			
4	ลบเมทธอด mA1						
5	ลบเมทธอด mB1			X			X
6	ลบเมทธอด mC1						
	รวม	2	1	2	1	0	1
	เสถียรภาพ	1-(2/6) = 0.67	1-(1/6) = 0.83	1-(2/6) = 0.67	1-(1/6) = 0.83	1-(0/6) = 1.0	1-(1/6) = 0.83

จากตารางที่ 3.5 เป็นตัวอย่างรหัสโปรแกรมที่ต้องการคำนวณเสถียรภาพ ส่วนตารางที่ 3.6 จะเป็นตัวอย่างในการคำนวณเสถียรภาพของเมทธอดและของคลาส โดยในตารางนี้ได้ยกตัวอย่างเฉพาะการลบแอทริบิวต์และการลบเมทธอดเท่านั้น ส่วนตัวอย่างการเปลี่ยนแปลงทั้งหมดอยู่ใน ภาคผนวก ฉ จากตารางที่ 3.6 เมื่อทำการลบแอทริบิวต์ b1 ในคลาส B จะส่งผลกระทบไปยังเมทธอด mA1 เมทธอด mb1 และคลาส A ส่วนคลาส B เป็นคลาสที่เกิดการเปลี่ยนแปลงจึงไม่นับผลกระทบที่เกิดขึ้น เมื่อทำการเปลี่ยนแปลงจนหมดทุกกรณีกับทุกคลาสของโปรแกรมตามตารางที่ 3.3 และ ตารางที่ 3.4 แล้วจึงคำนวณเสถียรภาพของเมทธอดและคลาส เช่น เมทธอด mA1 ถูกกระทบ 2 ครั้ง ฉะนั้น เสถียรภาพของเมทธอด mA1 คือ 0.67 ส่วนคลาส A ถูกกระทบ 1 ครั้ง ดังนั้นเสถียรภาพของคลาส A คือ 0.83

สำหรับสมการในการคำนวณเสถียรภาพของเมทธอดและคลาสนี้ นำไปใช้ในเครื่องมือการคำนวณเสถียรภาพในบทที่ 5

3.4 การสร้างโมเดลเพื่อทำนายเสถียรภาพของเมทธอด

เมื่อผ่านขั้นตอน การเก็บค่ามาตรวัดในข้อ 3.2 และการเก็บค่าเสถียรภาพในข้อ 3.3 แล้วจะได้ค่ามาตรวัดและค่าเสถียรภาพของแต่ละเมทธอดของรหัสโปรแกรมชุดที่ใช้ในการสร้างโมเดล หลังจากนั้นจึงนำข้อมูลทั้ง 2 ส่วนที่ได้มานั้น มาสร้างโมเดลโดยใช้วิธีการวิเคราะห์ความถดถอยแบบสเตปไวส์ ซึ่งใช้โปรแกรมการวิเคราะห์ทางสถิติเอสพีเอสเอสสำหรับวินโดว์มาช่วย สำหรับโมเดลในการทำนายเสถียรภาพของเมทธอด อยู่ในรูปสมการเชิงเส้น

$$\hat{Y} = a + b_1x_1 + b_2x_2 + \ldots + b_ix_i + \ldots + b_nx_n$$

โดย

$\hat{Y}$ คือ ค่าเสถียรภาพของเมทธอด

b_i คือค่าประมาณของสัมประสิทธิ์ของมาตรวัดแต่ละตัว เมื่อ $i = 1,2,..,n$

a คือค่าคงที่

x_i คือค่ามาตรวัด เมื่อ $i = 1,2,..,n$

สำหรับการสร้างโมเดลทำนายเสถียรภาพของเมทธอด จะอยู่ในบทที่ 4

3.5 การประเมินผลโมเดลทำนายเสถียรภาพของเมทธอด

ในการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด ได้ใช้รหัสโปรแกรมชุดที่สองซึ่งแยกออกจากชุดที่หนึ่ง มาใช้เป็นชุดทดสอบโมเดล สำหรับวิธีการทดสอบใช้วิธีการทำนายที่ระดับแอล (Prediction at level l : (PRED(l))) [14] ซึ่งมีสมการดังนี้

$$PRED(l) = \frac{k}{n} \qquad [14]$$

โดย

$PRED(l)$ คือ การทำนายที่ระดับแอล

l คือ ช่วงความคลาดเคลื่อน

k คือ จำนวนหน่วยตัวอย่างที่มีค่า $MRE \leq l$

n คือ จำนวนหน่วยตัวอย่างทั้งหมด

สำหรับการหา MRE สามารถแสดงได้ดังสมการ

$$MRE = |RE| = \left|\frac{E - \hat{E}}{E}\right| \qquad [14]$$

โดย

MRE คือ ขนาดความผิดพลาดสัมพันธ์ (Magnitude relative error)

RE คือ ความผิดพลาดสัมพันธ์ (Relative error)

E คือ ค่าเสถียรภาพของเมทธอด

$\hat{E}$ คือ ค่าทำนายเสถียรภาพของเมทธอด

ตัวอย่างความหมายของวิธีการทำนายที่ระดับ แอล เช่น ถ้า PRED(0.25) = 0.83 หมายความว่า 83% ของค่าที่ทำนายได้ มีความคลาดเคลื่อนไม่เกิน 25% ของค่าจริง คืออยู่ในช่วง -25% ถึง 25% ของค่าจริง [14]

สำหรับการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด จะอยู่ในบทที่ 4

3.6 การพัฒนาเครื่องมือทำนายเสถียรภาพของเมทธอด

หลังจากสร้างโมเดลและทำการทดสอบโมเดลทำนายเสถียรภาพของเมทธอดเรียบร้อยแล้ว จึงพัฒนาเครื่องมือทำนายเสถียรภาพของเมทธอด โดยการนำเครื่องมือเก็บค่ามาตรวัดในข้อ 3.2 มาทำการเพิ่มส่วนการทำนายเสถียรภาพของเมทธอด คือ นำค่าของมาตรวัดและค่าสัมประสิทธิ์ของแต่ละมาตรวัดที่ปรากฏอยู่ในโมเดลทำนายเสถียรภาพของเมทธอดมาคำนวณ ซึ่งการใช้งานเครื่องมือแสดงในภาคผนวก ง

3.7 การศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด

ในขั้นตอนสุดท้ายของงานวิจัยนี้ ได้ศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสที่ได้จากหัวข้อ 3.3 และค่าทำนายเสถียรภาพของเมทธอดที่เป็นสมาชิกของคลาสนั้น ว่ามีความสัมพันธ์กันหรือไม่ โดยใช้การวิเคราะห์สหสัมพันธ์ สำหรับเสถียรภาพของเมทธอดที่คัดเลือกมาทดสอบนั้น มีดังนี้

- ค่าเสถียรภาพมากที่สุดของเมทธอดในแต่ละคลาส
- ค่าเสถียรภาพน้อยที่สุดของเมทธอดในแต่ละคลาส
- ค่าเสถียรภาพเฉลี่ยของเมทธอดในแต่ละคลาส

สำหรับการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด จะอยู่ในบทที่ 4

บทที่ 4

การสร้างและทดสอบโมเดลทำนายเสถียรภาพของเมทธอด

ในบทนี้จะกล่าวถึง รหัสโปรแกรมที่ใช้สร้างโมเดล การสร้างโมเดลทำนายเสถียรภาพของเมทธอดและโมเดลที่ได้ การทดสอบโมเดลทำนายเสถียรภาพของเมทธอด รวมถึงการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของเมทธอดและเสถียรภาพของคลาส ซึ่งมีรายละเอียดดังนี้

4.1 รหัสโปรแกรมที่ใช้ในงานวิจัย

สำหรับรหัสโปรแกรมภาษาจาวาที่ใช้ในงานวิจัยได้นำมาจาก [16][17][18] โดยแบ่งรหัสโปรแกรมออกเป็น 2 ชุด ดังแสดงในตารางที่ 4.1 ประกอบด้วย

- ชุดที่หนึ่ง ใช้สำหรับการสร้างโมเดลทำนายเสถียรภาพของเมทธอด ซึ่งมีทั้งหมด 10 โปรแกรม 216 คลาส 1451 เมทธอด ได้แก่ โปรแกรมที่ 1 ถึงโปรแกรมที่ 10

- ชุดที่สอง ใช้สำหรับการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด ซึ่งมีทั้งหมด 4 โปรแกรม 59 คลาส 317 เมทธอด ได้แก่ โปรแกรมที่ 11 ถึงโปรแกรมที่ 14

โดยรหัสโปรแกรมที่นำมาใช้ต้องมีจำนวนคลาสตั้งแต่ 10 คลาสถึง 50 คลาสและรหัสโปรแกรมทั้งหมดต้องผ่านการคอมไพล์แล้ว เพื่อให้แน่ใจว่ารหัสโปรแกรมมีความถูกต้องตามหลักภาษาจาวา

4.2 การสร้างโมเดลทำนายเสถียรภาพของเมทธอด

ในหัวข้อนี้จะกล่าวถึงการสร้างโมเดลทำนายเสถียรภาพของเมทธอด โดยจะอธิบายข้อมูลที่ใช้ในการสร้างโมเดล รายละเอียดและผลการสร้างโมเดลทำนายเสถียรภาพของเมทธอด ซึ่งมีรายละเอียดดังนี้

4.2.1 ข้อมูลที่ใช้ในการโมเดล

ข้อมูลที่ใช้สร้างโมเดลทำนายเสถียรภาพของเมทธอด ประกอบด้วย

1. ค่ามาตรวัดของเมทธอด โดยนำรหัสโปรแกรมชุดสร้างโมเดล มาเก็บค่ามาตรวัดโดยใช้เครื่องมือเก็บค่ามาตรวัดที่ได้จากบทที่ 5
2. ค่าคำนวณเสถียรภาพของเมทธอด โดยนำรหัสโปรแกรมชุดสร้างโมเดลมาคำนวณ โดยใช้เครื่องมือคำนวณเสถียรภาพที่ได้จากบทที่ 5

สำหรับข้อมูลค่ามาตรวัดและค่าเสถียรภาพของเมทธอด ที่นำไปใช้สร้างโมเดลทำนายเสถียรภาพของเมทธอด มีลักษณะดังตารางที่ 4.2 แต่เนื่องจากมีหน่วยทดลองจำนวนมาก จึงยกตัวอย่างหน่วยทดลองเพียงบางส่วนเท่านั้น

4.2.2 โมเดลทำนายเสถียรภาพของเมทธอด

หลังจากได้จัดเตรียมข้อมูลเรียบร้อยแล้ว จึงนำค่ามาตรวัดและค่าเสถียรภาพของเมทธอดที่เก็บได้นั้น มาสร้างโมเดลในการทำนายเสถียรภาพของเมทธอด โดยใช้วิธีการสร้างสมการถดถอย โดยในงานวิจัยนี้ได้นำโปรแกรมเอสพีเอสเอสสำหรับวินโดวส์เข้ามาช่วยในการสร้าง โดยกำหนดให้ค่ามาตรวัดเป็นตัวแปรอิสระและค่าเสถียรภาพของเมทธอดเป็นตัวแปรตาม จากตารางที่ 4.3 แสดงความสัมพันธ์ระหว่างค่ามาตรวัดต่าง ๆ กับเสถียรภาพของเมทธอด รวมถึงความสัมพันธ์ระหว่างค่ามาตรวัดด้วย โดยในสดมภ์ที่สองจะแสดงค่าความสัมพันธ์ระหว่างเสถียรภาพของเมทธอดกับค่ามาตรวัดต่าง ๆ ซึ่งจะพบว่าค่ามาตรวัด NMIpub มีความสัมพันธ์กับเสถียรภาพของเมทธอดมากที่สุด คือ -0.606

สำหรับการสร้างสมการถดถอย แบบสเตปไวส์ จะเลือกค่ามาตรวัด (ตัวแปรอิสระ) มาทดลองสร้างโมเดลครั้งละหนึ่งมาตรวัด โดยจะเลือกมาตรวัดที่มีความสัมพันธ์กับเสถียรภาพของเมทธอดมากที่สุด ซึ่งได้แก่ NMIpub นำมาใส่ในโมเดล ตามตารางที่ 4.4 และโมเดลที่ได้คือ โมเดลที่ 1 ในตารางที่ 4.6 โดยมีค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อนเท่ากับ 0.367

หลังจากนั้นจึงเลือกมาตรวัด ที่มีความสัมพันธ์กับเสถียรภาพของเมทธอดมากที่สุดจากมาตรวัดที่เหลืออยู่ ซึ่งแสดงในแถวของโมเดลที่ 1 ในสดมภ์สัมประสิทธิ์สหสัมพันธ์เชิงส่วน (Partial correlation) ของตารางที่ 4.5 คือ MCX มีค่าเท่ากับ -0.411 มาใส่ในโมเดลและทำการสร้างโมเดลที่ 2 ตามตารางที่ 4.4 และตารางที่ 4.6 ตามลำดับโดยมีค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อนมีค่าเท่ากับ 0.474

ลำดับถัดมาจึงเลือกมาตรวัดที่มีความสัมพันธ์กับเสถียรภาพของเมทธอดที่มากที่สุดจากมาตรวัดที่เหลืออยู่ ที่แสดงในแถวของโมเดลที่ 2 ใน ตารางที่ 4.5 มาทำการสร้างโมเดลต่อไป ทำ

การเลือกมาตรวัดที่เหลืออยู่ที่มีความสัมพันธ์กับเสถียรภาพของเมทธอดมากที่สุด มาทดลองสร้างโมเดลไปเรื่อย ๆ จนไม่มีมาตรวัดใดที่สามารถนำมาสร้างโมเดลโดยมีนัยสำคัญได้

เมื่อไม่สามารถสร้างโมเดลใดได้อีก จึงเลือกโมเดลที่มีค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อนที่มีค่ามากที่สุด มาใช้เป็นโมเดลในการทำนาย ซึ่งได้แก่ โมเดลที่ 7 ในตารางที่ 4.6 ซึ่งประกอบด้วยมาตรวัด 7 ตัวคือ NMIpub MCX NAIpro NAIpub NAIpro NAIdef และ Param โดยมีค่าสัมประสิทธิ์การตัดสินใจเชิงซ้อนเท่ากับ 0.617 หรือ 61.7% ส่วนสัมประสิทธิ์ของมาตรวัดแต่ละตัวแสดงในแถวโมเดลที่ 7 ใน ตารางที่ 4.7 ดังนั้นโมเดลทำนายเสถียรภาพของเมทธอดที่ได้มีรูปแบบสมการดังนี้

เสถียรภาพของเมทธอด = 1 -0.00496*NMIpub -0.0000407*MCX -0.00345*NAIpro
-0.00241*NAIpub -0.00141*NAIpri -0.00097*NAIdef +0.0006317*Param

จากสมการคำนวณเสถียรภาพของเมทธอดในหัวข้อ 3.3.2 เสถียรภาพของเมทธอดจะมีค่าอยู่ระหว่าง 0 ถึง 1 ดังนั้นหากค่าทำนายที่ได้จากโมเดลมีค่าน้อยกว่า 0 จะกำหนดให้มีค่าเท่ากับ 0 หรือในทางตรงข้ามหากค่าทำนายที่ได้จากโมเดลมีค่ามากกว่า 1 จะกำหนดให้มีค่าเท่ากับ 1

จากโมเดลที่ 7 ในตารางที่ 4.7 มาตรวัดที่มีค่าในสดมภ์ความสัมพันธ์เชิงส่วน (Partial correlations) มาก จะมีความสัมพันธ์กับเสถียรภาพของเมทธอดมาก ซึ่งสามารถเรียงลำดับมาตรวัดที่มีผลกับเสถียรภาพของเมทธอดจากมากไปน้อย ได้ดังนี้

1. มาตรวัด NMIpub มีความสัมพันธ์เชิงส่วนเท่ากับ -0.561
2. มาตรวัด NAIpro มีความสัมพันธ์เชิงส่วนเท่ากับ -0.445
3. มาตรวัด NAIpub มีความสัมพันธ์เชิงส่วนเท่ากับ -0.321
4. มาตรวัด NAIpri มีความสัมพันธ์เชิงส่วนเท่ากับ -0.244
5. มาตรวัด MCXมีความสัมพันธ์เชิงส่วนเท่ากับ -0.121
6. มาตรวัด NAIdef มีความสัมพันธ์เชิงส่วนเท่ากับ -0.117
7. มาตรวัด Param มีความสัมพันธ์เชิงส่วนเท่ากับ 0.077

ตารางที่ 4.1 รายชื่อรหัสโปรแกรมที่ใช้งานวิจัย

โปรแกรมที่	ชื่อโปรแกรม	จำนวนคลาส	จำนวนเมทธอด
1	Bomberman	24	133
2	Breakout	40	250
3	ChatProg	15	66
4	ChatterBoxClient	30	177
5	ChatterBoxServer	28	164
6	DBase	28	366
7	Download_Accelerator	11	46
8	EightPuzzle	10	84
9	EmailFinder	11	32
10	FourInA_Row	19	133
11	TicTacToe	10	42
12	TrivialFTP	11	28
13	TT4D	19	138
14	Whiteboard	19	109
	รวม	275	1768

ตารางที่ 4.2 ตัวอย่างค่ามาตรวัดและค่าเสถียรภาพของเมทธอดที่เก็บได้

ProjName	ClassName	MethodName	Param	LOC	NOS	VG	Local Var	CBO	MCX	NAI pub	NAI pro	NAI def	NAI pri	NMI pub	NMI pro	NMI def	NMI pri	MethodStability
EightPuzzle	ICallBack	solverMachineStarted ( )	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	1
EightPuzzle	PuzzleSolver	PuzzleSolver (Vector, int, int)	3	9	8	1	0	4	13.4	0	0	1	5	3	0	0	0	0.94747084
EightPuzzle	PuzzleSolver	setPaused (boolean)	1	2	1	1	0	1	0.8	0	0	0	1	0	0	0	0	0.99610895
EightPuzzle	PuzzleSolver	isSolved (PuzzleTree)	1	10	5	4	4	3	15.3	0	0	0	2	1	0	0	0	0.98054475
EightPuzzle	PuzzleSolver	run ( )	0	9	7	2	1	4	16.5	0	0	0	1	0	0	0	4	0.97859925
EightPuzzle	PuzzleSolver	solve ( )	0	67	58	28	8	4	154	1	0	0	4	16	0	0	2	0.78793776
EightPuzzle	PuzzleSolver	notifyMachineStarted ( )	0	6	3	2	2	2	17	0	0	1	0	1	0	0	0	0.9902724
EightPuzzle	PuzzleSolver	notifyMachineError (String)	1	6	3	2	2	3	17.3	0	0	1	0	1	0	0	0	0.9902724
EightPuzzle	PuzzleSolver	notifyMachineDo (String)	1	6	3	2	2	3	17.3	0	0	1	0	1	0	0	0	0.9902724
EightPuzzle	PuzzleStack	PuzzleStack ( )	0	2	1	1	0	1	0.5	0	0	0	1	0	0	0	0	0.99610895
EightPuzzle	PuzzleStack	add (PuzzleTree, boolean)	2	16	11	6	4	4	43.6	0	0	0	1	2	0	0	0	0.9824903
EightPuzzle	PuzzleStack	getOptimalHeuristicIndex ( )	0	13	9	5	3	3	35	0	0	0	1	3	0	0	0	0.9747082
EightPuzzle	PuzzleStack	contains (PuzzleTree)	1	2	1	1	0	3	3.3	0	0	0	1	0	0	0	0	0.99610895
EightPuzzle	PuzzleStack	remove (int)	1	2	1	1	0	3	3.3	0	0	0	1	0	0	0	0	0.99610895
EightPuzzle	PuzzleStack	add (int, PuzzleTree)	2	2	1	1	0	3	3.6	0	0	0	1	0	0	0	0	0.99610895

ตารางที่ 4.3 ค่าความสัมพันธ์ระหว่างมาตรวัดและเสถียรภาพของเมทธอด

Correlations

		MethodStability	PARAM	LOC	NOS	VG	LOCALVAR	CBO	MCX	NAIPUB	NAIPRO	NAIDEF	NAIPRI	NMIPUB	NMIPRO	NMIDEF	NMIPRI
MethodStability	Pearson Correlation	1	.009	-.507**	-.486**	-.381**	-.408**	-.428**	-.547**	-.453**	-.306**	-.122**	-.343**	-.606**	-.125**	-.019	-.166**
	Sig. (2-tailed)	.	.728	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.461	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
PARAM	Pearson Correlation	.009	1	.100**	.094**	.072**	.092**	.289**	.109**	.017	-.069**	-.045	.011	.103**	.004	.043	.018
	Sig. (2-tailed)	.728	.	.000	.000	.006	.000	.000	.000	.508	.008	.088	.671	.000	.878	.102	.496
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
LOC	Pearson Correlation	-.507**	.100**	1	.989**	.900**	.644**	.515**	.811**	.428**	.141**	.257**	.347**	.435**	.050	.072**	.394**
	Sig. (2-tailed)	.000	.000	.	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.056	.006	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NOS	Pearson Correlation	-.486**	.094**	.989**	1	.910**	.526**	.454**	.762**	.400**	.157**	.263**	.340**	.408**	.050	.055*	.412**
	Sig. (2-tailed)	.000	.000	.000	.	.000	.000	.000	.000	.000	.000	.000	.000	.000	.055	.036	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
VG	Pearson Correlation	-.381**	.072**	.900**	.910**	1	.474**	.344**	.700**	.358**	.017	.180**	.338**	.332**	.022	.052*	.418**
	Sig. (2-tailed)	.000	.006	.000	.000	.	.000	.000	.000	.000	.507	.000	.000	.000	.397	.048	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
LOCALVAR	Pearson Correlation	-.408**	.092**	.644**	.526**	.474**	1	.623**	.736**	.398**	.003	.124**	.242**	.396**	.028	.131**	.139**
	Sig. (2-tailed)	.000	.000	.000	.000	.000	.	.000	.000	.000	.901	.000	.000	.000	.280	.000	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
CBO	Pearson Correlation	-.428**	.289**	.515**	.454**	.344**	.623**	1	.554**	.306**	.213**	.173**	.375**	.370**	.113**	.079**	.127**
	Sig. (2-tailed)	.000	.000	.000	.000	.000	.000	.	.000	.000	.000	.000	.000	.000	.000	.003	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
MCX	Pearson Correlation	-.547**	.109**	.811**	.762**	.700**	.736**	.554**	1	.535**	.114**	.134**	.437**	.412**	.042	.062*	.133**
	Sig. (2-tailed)	.000	.000	.000	.000	.000	.000	.000	.	.000	.000	.000	.000	.000	.108	.018	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NAIPUB	Pearson Correlation	-.453**	.017	.428**	.400**	.358**	.398**	.306**	.535**	1	-.017	.043	.228**	.237**	.015	.028	.031
	Sig. (2-tailed)	.000	.508	.000	.000	.000	.000	.000	.000	.	.526	.106	.000	.000	.565	.285	.234
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NAIPRO	Pearson Correlation	-.306**	-.069**	.141**	.157**	.017	.003	.213**	.114**	-.017	1	-.094**	-.121**	.007	.255**	-.037	-.006
	Sig. (2-tailed)	.000	.008	.000	.000	.507	.901	.000	.000	.526	.	.000	.000	.797	.000	.157	.831
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NAIDEF	Pearson Correlation	-.122**	-.045	.257**	.263**	.180**	.124**	.173**	.134**	.043	-.094**	1	.037	.095**	-.054*	.096**	.245**
	Sig. (2-tailed)	.000	.088	.000	.000	.000	.000	.000	.000	.106	.000	.	.162	.000	.041	.000	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NAIPRI	Pearson Correlation	-.343**	.011	.347**	.340**	.338**	.242**	.375**	.437**	.228**	-.121**	.037	1	.218**	-.084**	-.021	.114**
	Sig. (2-tailed)	.000	.671	.000	.000	.000	.000	.000	.000	.000	.000	.162	.	.000	.001	.419	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NMIPUB	Pearson Correlation	-.606**	.103**	.435**	.408**	.332**	.396**	.370**	.412**	.237**	.007	.095**	.218**	1	.098**	.082**	.223**
	Sig. (2-tailed)	.000	.000	.000	.000	.000	.000	.000	.000	.000	.797	.000	.000	.	.000	.002	.000
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NMIPRO	Pearson Correlation	-.125**	.004	.050	.050	.022	.028	.113**	.042	.015	.255**	-.054*	-.084**	.098**	1	-.023	-.036
	Sig. (2-tailed)	.000	.878	.056	.055	.397	.280	.000	.108	.565	.000	.041	.001	.000	.	.382	.172
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NMIDEF	Pearson Correlation	-.019	.043	.072**	.055*	.052*	.131**	.079**	.062*	.028	-.037	.096**	-.021	.082**	-.023	1	.034
	Sig. (2-tailed)	.461	.102	.006	.036	.048	.000	.003	.018	.285	.157	.000	.419	.002	.382	.	.192
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451
NMIPRI	Pearson Correlation	-.166**	.018	.394**	.412**	.418**	.139**	.127**	.133**	.031	-.006	.245**	.114**	.223**	-.036	.034	1
	Sig. (2-tailed)	.000	.496	.000	.000	.000	.000	.000	.000	.234	.831	.000	.000	.000	.172	.192	.
	N	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451	1451

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

ตารางที่ 4.4 มาตรวัดที่เลือกมาใช้หรือเอาออกจากโมเดล

Variables Entered/Removed [a]

Model	Variables Entered	Variables Removed	Method
1	NMIPUB	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
2	MCX	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
3	NAIPRO	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
4	NAIPUB	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
5	NAIPRI	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
6	NAIDEF	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).
7	PARAM	.	Stepwise (Criteria: Probability-of-F-to-enter <= .050, Probability-of-F-to-remove >= .100).

a. Dependent Variable: MethodStability

ตารางที่ 4.5 มาตรวัดที่ไม่ได้นำมาสร้างโมเดล

Excluded Variables [h]

Model		Beta In	t	Sig.	Partial Correlation	Collinearity Statistics		
						Tolerance	VIF	Minimum Tolerance
1	PARAM	.072[a]	3.457	.001	.090	.989	1.011	.989
	LOC	-.300[a]	-13.734	.000	-.339	.811	1.233	.811
	NOS	-.286[a]	-13.227	.000	-.328	.833	1.200	.833
	VG	-.201[a]	-9.364	.000	-.239	.890	1.124	.890
	LOCALVAR	-.199[a]	-8.999	.000	-.230	.843	1.186	.843
	CBO	-.236[a]	-10.908	.000	-.276	.863	1.159	.863
	MCX	-.358[a]	-17.140	.000	-.411	.830	1.204	.830
	NAIPUB	-.327[a]	-16.611	.000	-.400	.944	1.059	.944
	NAIPRO	-.302[a]	-15.648	.000	-.380	1.000	1.000	1.000
	NAIDEF	-.064[a]	-3.077	.002	-.081	.991	1.009	.991
	NAIPRI	-.222[a]	-10.765	.000	-.272	.953	1.050	.953
	NMIPRO	-.066[a]	-3.133	.002	-.082	.990	1.010	.990
	NMIDEF	.031[a]	1.467	.143	.039	.993	1.007	.993
	NMIPRI	-.032[a]	-1.508	.132	-.040	.950	1.052	.950
2	PARAM	.097[b]	5.086	.000	.133	.984	1.016	.826
	LOC	-.050[b]	-1.521	.128	-.040	.330	3.030	.330
	NOS	-.063[b]	-2.104	.036	-.055	.409	2.445	.408
	VG	.044[b]	1.658	.098	.044	.508	1.968	.474
	LOCALVAR	.083[b]	2.928	.003	.077	.448	2.231	.441
	CBO	-.089[b]	-3.855	.000	-.101	.669	1.494	.644
	NAIPUB	-.213[b]	-9.765	.000	-.249	.713	1.402	.628
	NAIPRO	-.267[b]	-14.910	.000	-.365	.985	1.015	.818
	NAIDEF	-.030[b]	-1.575	.116	-.041	.980	1.020	.821
	NAIPRI	-.108[b]	-5.116	.000	-.133	.807	1.238	.704
	NMIPRO	-.065[b]	-3.404	.001	-.089	.990	1.010	.824
	NMIDEF	.041[b]	2.142	.032	.056	.992	1.008	.827
	NMIPRI	-.017[b]	-.854	.393	-.022	.948	1.054	.802
3	PARAM	.076[c]	4.258	.000	.111	.978	1.023	.813
	LOC	-.007[c]	-.225	.822	-.006	.327	3.057	.327
	NOS	-.014[c]	-.513	.608	-.013	.403	2.479	.403
	VG	.013[c]	.514	.607	.014	.504	1.982	.464
	LOCALVAR	.038[c]	1.440	.150	.038	.442	2.261	.429
	CBO	-.028[c]	-1.256	.209	-.033	.644	1.553	.644
	NAIPUB	-.244[c]	-12.145	.000	-.304	.707	1.414	.613
	NAIDEF	-.060[c]	-3.352	.001	-.088	.968	1.033	.807
	NAIPRI	-.170[c]	-8.643	.000	-.222	.778	1.285	.679
	NMIPRO	.004[c]	.199	.843	.005	.925	1.081	.817
	NMIDEF	.030[c]	1.674	.094	.044	.991	1.010	.817
	NMIPRI	-.020[c]	-1.112	.266	-.029	.948	1.055	.800
4	PARAM	.064[d]	3.763	.000	.099	.975	1.026	.607
	LOC	-.010[d]	-.338	.735	-.009	.327	3.057	.296
	NOS	-.017[d]	-.626	.531	-.016	.403	2.479	.348
	VG	.002[d]	.089	.929	.002	.504	1.985	.383
	LOCALVAR	.037[d]	1.442	.149	.038	.442	2.261	.364
	CBO	-.020[d]	-.967	.334	-.025	.644	1.554	.515
	NAIDEF	-.070[d]	-4.099	.000	-.107	.966	1.035	.604
	NAIPRI	-.176[d]	-9.450	.000	-.241	.778	1.286	.527
	NMIPRO	.006[d]	.369	.712	.010	.925	1.081	.613
	NMIDEF	.028[d]	1.620	.105	.043	.990	1.010	.612
	NMIPRI	-.032[d]	-1.828	.068	-.048	.945	1.058	.610
5	PARAM	.055[e]	3.292	.001	.086	.971	1.030	.520
	LOC	-.011[e]	-.392	.695	-.010	.327	3.057	.274
	NOS	-.010[e]	-.376	.707	-.010	.403	2.481	.322
	VG	.009[e]	.389	.697	.010	.503	1.987	.351
	LOCALVAR	-.002[e]	-.083	.934	-.002	.430	2.325	.309
	CBO	.023[e]	1.091	.276	.029	.613	1.631	.475
	NAIDEF	-.078[e]	-4.712	.000	-.123	.964	1.038	.517
	NMIPRO	-.005[e]	-.318	.750	-.008	.920	1.087	.527
	NMIDEF	.017[e]	1.043	.297	.027	.986	1.014	.525
	NMIPRI	-.024[e]	-1.392	.164	-.037	.943	1.061	.526
6	PARAM	.049[f]	2.950	.003	.077	.965	1.037	.509
	LOC	.026[f]	.887	.375	.023	.304	3.289	.271
	NOS	.024[f]	.895	.371	.024	.375	2.667	.321
	VG	.021[f]	.918	.359	.024	.497	2.012	.350
	LOCALVAR	-.001[f]	-.034	.973	-.001	.430	2.325	.306
	CBO	.039[f]	1.848	.065	.049	.598	1.671	.472
	NMIPRO	-.009[f]	-.513	.608	-.014	.918	1.089	.517
	NMIDEF	.023[f]	1.427	.154	.038	.980	1.021	.516
	NMIPRI	-.006[f]	-.333	.739	-.009	.894	1.119	.517
7	LOC	.023[g]	.777	.437	.020	.304	3.293	.270
	NOS	.021[g]	.778	.436	.020	.374	2.671	.319
	VG	.022[g]	.945	.345	.025	.497	2.012	.346
	LOCALVAR	.000[g]	-.001	.999	.000	.430	2.325	.302
	CBO	.021[g]	.921	.357	.024	.532	1.880	.472
	NMIPRO	-.009[g]	-.538	.591	-.014	.918	1.089	.509
	NMIDEF	.022[g]	1.340	.181	.035	.979	1.022	.508
	NMIPRI	-.006[g]	-.359	.719	-.009	.894	1.119	.509

a. Predictors in the Model: (Constant), NMIPUB

b. Predictors in the Model: (Constant), NMIPUB, MCX

c. Predictors in the Model: (Constant), NMIPUB, MCX, NAIPRO

d. Predictors in the Model: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB

e. Predictors in the Model: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI

f. Predictors in the Model: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI, NAIDEF

g. Predictors in the Model: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI, NAIDEF, PARAM

h. Dependent Variable: MethodStability

ตารางที่ 4.6 โมเดลทำนายเสถียรภาพของเมทธอด

Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics				
					R Square Change	F Change	df1	df2	Sig. F Change
1	.606[a]	.368	.367	.0113294284	.368	843.081	1	1449	.000
2	.689[b]	.474	.474	.0103334526	.107	293.781	1	1448	.000
3	.738[c]	.544	.543	.0096241071	.070	222.316	1	1447	.000
4	.766[d]	.587	.585	.0091710137	.042	147.510	1	1446	.000
5	.781[e]	.611	.609	.0089031940	.024	89.304	1	1445	.000
6	.785[f]	.617	.615	.0088385707	.006	22.208	1	1444	.000
7	.787[g]	.619	.617	.0088150998	.002	8.700	1	1443	.003

a. Predictors: (Constant), NMIPUB

b. Predictors: (Constant), NMIPUB, MCX

c. Predictors: (Constant), NMIPUB, MCX, NAIPRO

d. Predictors: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB

e. Predictors: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI

f. Predictors: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI, NAIDEF

g. Predictors: (Constant), NMIPUB, MCX, NAIPRO, NAIPUB, NAIPRI, NAIDEF, PARAM

ตารางที่ 4.7 ค่าสัมประสิทธิ์ของมาตรวัด

Coefficients [a]

Model		Unstandardized Coefficients		Standardized Coefficients			95% Confidence Interval for B		Correlations			Collinearity Statistics	
		B	Std. Error	Beta	t	Sig.	Lower Bound	Upper Bound	Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	.997	.000		3031.888	.000	.996	.998					
	NMIPUB	-6.52E-03	.000	-.606	-29.036	.000	-.007	-.006	-.606	-.606	-.606	1.000	1.000
2	(Constant)	.998	.000		3204.465	.000	.998	.999					
	NMIPUB	-4.93E-03	.000	-.459	-21.951	.000	-.005	-.004	-.606	-.500	-.418	.830	1.204
	MCX	-1.39E-04	.000	-.358	-17.140	.000	.000	.000	-.547	-.411	-.327	.830	1.204
3	(Constant)	1.000	.000		3334.419	.000	.999	1.000					
	NMIPUB	-5.07E-03	.000	-.472	-24.207	.000	-.005	-.005	-.606	-.537	-.430	.829	1.207
	MCX	-1.25E-04	.000	-.323	-16.449	.000	.000	.000	-.547	-.397	-.292	.818	1.222
	NAIPRO	-2.87E-03	.000	-.267	-14.910	.000	-.003	-.002	-.306	-.365	-.265	.985	1.015
4	(Constant)	1.000	.000		3490.014	.000	.999	1.000					
	NMIPUB	-5.03E-03	.000	-.468	-25.186	.000	-.005	-.005	-.606	-.552	-.426	.829	1.207
	MCX	-7.41E-05	.000	-.191	-8.868	.000	.000	.000	-.547	-.227	-.150	.613	1.631
	NAIPRO	-3.07E-03	.000	-.286	-16.694	.000	-.003	-.003	-.306	-.402	-.282	.977	1.024
	NAIPUB	-2.36E-03	.000	-.244	-12.145	.000	-.003	-.002	-.453	-.304	-.205	.707	1.414
5	(Constant)	1.001	.000		3492.878	.000	1.000	1.001					
	NMIPUB	-4.96E-03	.000	-.461	-25.559	.000	-.005	-.005	-.606	-.558	-.420	.827	1.209
	MCX	-4.30E-05	.000	-.111	-4.920	.000	.000	.000	-.547	-.128	-.081	.527	1.898
	NAIPRO	-3.40E-03	.000	-.316	-18.687	.000	-.004	-.003	-.306	-.441	-.307	.941	1.063
	NAIPUB	-2.40E-03	.000	-.249	-12.758	.000	-.003	-.002	-.453	-.318	-.209	.707	1.415
	NAIPRI	-1.40E-03	.000	-.176	-9.450	.000	-.002	-.001	-.343	-.241	-.155	.778	1.286
6	(Constant)	1.001	.000		3406.692	.000	1.000	1.001					
	NMIPUB	-4.92E-03	.000	-.458	-25.522	.000	-.005	-.005	-.606	-.558	-.416	.826	1.211
	MCX	-3.75E-05	.000	-.097	-4.275	.000	.000	.000	-.547	-.112	-.070	.517	1.933
	NAIPRO	-3.51E-03	.000	-.326	-19.255	.000	-.004	-.003	-.306	-.452	-.314	.927	1.078
	NAIPUB	-2.45E-03	.000	-.253	-13.063	.000	-.003	-.002	-.453	-.325	-.213	.705	1.418
	NAIPRI	-1.43E-03	.000	-.180	-9.743	.000	-.002	-.001	-.343	-.248	-.159	.776	1.289
	NAIDEF	-1.02E-03	.000	-.078	-4.712	.000	-.001	-.001	-.122	-.123	-.077	.964	1.038
7	(Constant)	1.000	.000		2918.158	.000	1.000	1.001					
	NMIPUB	-4.96E-03	.000	-.461	-25.731	.000	-.005	-.005	-.606	-.561	-.418	.822	1.217
	MCX	-4.07E-05	.000	-.105	-4.623	.000	.000	.000	-.547	-.121	-.075	.509	1.964
	NAIPRO	-3.45E-03	.000	-.321	-18.892	.000	-.004	-.003	-.306	-.445	-.307	.917	1.090
	NAIPUB	-2.41E-03	.000	-.250	-12.884	.000	-.003	-.002	-.453	-.321	-.209	.702	1.424
	NAIPRI	-1.41E-03	.000	-.177	-9.553	.000	-.002	-.001	-.343	-.244	-.155	.772	1.295
	NAIDEF	-9.70E-04	.000	-.074	-4.477	.000	-.001	-.001	-.122	-.117	-.073	.958	1.044
	PARAM	6.317E-04	.000	.049	2.950	.003	.000	.001	.009	.077	.048	.965	1.037

a. Dependent Variable: MethodStability

ตารางที่ 4.8 ตัวอย่างข้อมูลในการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด

ProjName	ClassName	MethodName	Param	MCX	NAI pub	NAI pro	NAI def	NAI pri	NMI pub	MethodStability	Predict	MRE
EightPuzzle	PuzzleSolver	PuzzleSolver (Vector, int, int)	3	13.4	0	0	1	5	3	0.94747084	0.978481218	0.032729638
EightPuzzle	PuzzleSolver	setPaused (boolean)	1	0.8	0	0	0	1	0	0.99610895	0.999193116	0.003096214
EightPuzzle	PuzzleSolver	isSolved (PuzzleTree)	1	15.3	0	0	0	2	1	0.98054475	0.992240531	0.01192784
EightPuzzle	PuzzleSolver	run ( )	0	16.5	0	0	0	1	0	0.97859925	0.997921955	0.019745269
EightPuzzle	PuzzleSolver	solve ( )	0	154	1	0	0	4	16	0.78793776	0.90639558	0.150339057
EightPuzzle	PuzzleSolver	notifyMachineStarted ( )	0	17	0	0	1	0	1	0.9902724	0.99338149	0.003139631
EightPuzzle	PuzzleSolver	notifyMachineError (String)	1	17.3	0	0	1	0	1	0.9902724	0.994000971	0.003765197
EightPuzzle	PuzzleSolver	notifyMachineDo (String)	1	17.3	0	0	1	0	1	0.9902724	0.994000971	0.003765197
EightPuzzle	PuzzleStack	PuzzleStack ( )	0	0.5	0	0	0	1	0	0.99610895	0.998573635	0.002474313
EightPuzzle	PuzzleStack	add (PuzzleTree, boolean)	2	43.6	0	0	0	1	2	0.9824903	0.988169572	0.005780487
EightPuzzle	PuzzleStack	getOptimalHeuristicIndex ( )	0	35	0	0	0	1	3	0.9747082	0.98230045	0.007789254
EightPuzzle	PuzzleStack	contains (PuzzleTree)	1	3.3	0	0	0	1	0	0.99610895	0.999091291	0.002993991
EightPuzzle	PuzzleStack	remove (int)	1	3.3	0	0	0	1	0	0.99610895	0.999091291	0.002993991
EightPuzzle	PuzzleStack	add (int, PuzzleTree)	2	3.6	0	0	0	1	0	0.99610895	0.999710772	0.003615892
EightPuzzle	ICallBack	solverMachineStarted ( )	0	0	0	0	0	0	0	1	1	0

4.3 การทดสอบโมเดลทำนายเสถียรภาพของเมทธอด

ในหัวข้อนี้จะกล่าวถึงการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด โดยจะอธิบายข้อมูลที่ใช้ในการทดสอบ รายละเอียดและผลการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด ซึ่งมีรายละเอียดดังนี้

4.3.1 ข้อมูลที่ใช้ในการโมเดล

ข้อมูลที่ใช้ในการทดสอบโมเดล ประกอบด้วย

1. ค่ามาตรวัดของเมทธอด โดยนำรหัสโปรแกรมชุดทดสอบ มาเก็บค่ามาตรวัดโดยใช้เครื่องมือเก็บค่ามาตรวัดที่ได้จากบทที่ 5
2. ค่าคำนวณเสถียรภาพของเมทธอด โดยนำรหัสโปรแกรมชุดทดสอบ มาคำนวณโดยใช้เครื่องมือคำนวณเสถียรภาพที่ได้จากบทที่ 5
3. ค่าทำนายเสถียรภาพของเมทธอด โดยนำค่ามาตรวัดในข้อ 1 มาคูณกับสัมประสิทธิ์ของแต่ละมาตรวัดที่ปรากฏอยู่ใน โมเดลทำนายเสถียรภาพของเมทธอด ที่ได้จากหัวข้อ 4.2

สำหรับข้อมูลทดสอบโมเดลทำนายเสถียรภาพของเมทธอดบางส่วน แสดงได้ดังตารางที่ 4.8 ซึ่งในตารางแสดงเฉพาะมาตรวัดที่ปรากฏอยู่ในโมเดลทำนายเสถียรภาพเท่านั้น สำหรับค่าคำนวณเสถียรภาพของเมทธอดอยู่ในสดมภ์ Methodstability และค่าทำนายเสถียรภาพของเมทธอดอยู่ในสดมภ์ Predict

4.3.2 ผลการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด

หลังจากได้จัดเตรียมข้อมูลเรียบร้อยแล้ว จึงนำค่าคำนวณเสถียรภาพของเมทธอดและค่าทำนายเสถียรภาพของเมทธอด (ข้อมูลในข้อ 2 และ 3 ในหัวข้อ 4.3.1) มาเปรียบเทียบโดยใช้วิธีการทำนายที่ระดับแอล ซึ่งมีสมการดังแสดงในหัวข้อ 3.5

สำหรับวิธีการทำนายที่ระดับแอล จะต้องมีการคำนวณขนาดความผิดพลาดสัมพันธ์ ซึ่งสมการการคำนวณแสดงในหัวข้อ 3.5 ตัวอย่างขนาดความผิดพลาดสัมพันธ์แสดงในสดมภ์ MRE ของตารางที่ 4.8 ส่วนตัวอย่างการคำนวณแสดงได้ดังนี้

จากตารางที่ 4.8 เมทธอด PuzzleSolver(Vector, int, int) ในแถวที่ 1 มีขนาดความผิดพลาดสัมพันธ์ เท่ากับ

$$\text{MRE(PuzzleSolver(Vector, int ,int)) } = \left| \frac{\text{ค่าคำนวณเสถียรภาพ} - \text{ค่าทำนายเสถียรภาพ}}{\text{ค่าคำนวณเสถียรภาพ}} \right|$$

$$\text{MRE(PuzzleSolver(Vector, int ,int)) } = \left| \frac{0.9474 - 0.9784}{0.9474} \right|$$

$$= 0.0327$$

จากข้อมูลใน ตารางที่ 4.8 การทำนายที่ระดับแอล เมื่อ แอล = 0.01 คือ จำนวนเมทธอดที่มีค่าในสดมภ์ MRE น้อยกว่าหรือเท่ากับ 0.01 มีจำนวน 11 เมทธอด (แถวที่มีแถบดำ) จากทั้งหมด 15 เมทธอด แสดงวิธีการคำนวณได้ดังนี้

$$\text{PRED(0.01)} = \frac{\text{จำนวนเมทธอดที่มีขนาดความผิดพลาดสัมพันธ์ น้อยกว่า 0.01}}{\text{จำนวนเมทธอดทั้งหมด}}$$

$$= \frac{11}{15}$$

$$= 0.73$$

จากตัวอย่างสามารถแปลหมายความได้ว่า ใน 100 เมทธอด ค่าคำนวณและค่าทำนายเสถียรภาพของเมทธอดที่มีความแตกต่างกัน น้อยกว่า 0.01 มีทั้งสิ้น 73 เมทธอด

สำหรับผลการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด สามารถสรุปได้ดังตารางที่ 4.9

ตารางที่ 4.9 ผลการทดสอบโมเดลทำนายเสถียรภาพของเมทธอด

ระดับแอล (l)	จำนวนเมทธอดที่ค่า MRE น้อยกว่า แอล (k)	จำนวนเมทธอดทั้งหมด(n)	ค่าทำนาย (k/n)
0.10	317	317	1
0.05	304	317	0.958
0.01	265	317	0.835

จากตารางที่ 4.9 การทำนายที่ระดับ 0.01 จะพบว่า ความแตกต่างระหว่างค่าคำนวณเสถียรภาพเมทธอดกับค่าที่ได้จากการทำนายที่มีความแตกต่างกันน้อยกว่าหรือเท่ากับ 0.01 มีทั้งหมด 83.5% หมายความว่า ใน 100 เมทธอด ค่าคำนวณและค่าทำนายเสถียรภาพของเมทธอดที่มีความแตกต่างกัน น้อยกว่า 0.01 มีทั้งสิ้น 83 เมทธอด

4.4 การศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด

ในหัวข้อนี้กล่าวถึงการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด โดยจะอธิบายข้อมูลที่ใช้ในการศึกษา ขั้นตอนและผลการศึกษาความสัมพันธ์ ซึ่งมีรายละเอียดดังนี้

4.4.1 ข้อมูลที่ใช้ในการศึกษาความสัมพันธ์

ข้อมูลที่ใช้ในการศึกษาความสัมพันธ์ ประกอบด้วย

1. ค่าเสถียรภาพของคลาส โดยนำรหัสโปรแกรมทั้งหมด มาคำนวณโดยใช้เครื่องมือคำนวณเสถียรภาพที่ได้จากบทที่ 5
2. ค่าทำนายเสถียรภาพของเมทธอด ในแต่ละคลาส เลือกมา 3 ค่า คือ
   - ค่าเสถียรภาพของเมทธอดที่มากที่สุดในคลาส
   - ค่าเสถียรภาพเฉลี่ยของเมทธอดในคลาส
   - ค่าเสถียรภาพของเมทธอดที่น้อยที่สุดในคลาส

ตัวอย่างข้อมูลทั้ง 2 ส่วนแสดงได้ดังตารางที่ 4.10 สำหรับค่าเสถียรภาพของเมทธอดที่มากที่สุด (อยู่ในสดมภ์ Max methodstability) ค่าเสถียรภาพของเมทธอดที่น้อยที่สุด (อยู่ในสดมภ์ Min methodstability) และค่าเสถียรภาพเฉลี่ยของเมทธอด (อยู่ในสดมภ์ Average methodstability) ข้อมูลทั้ง 3 สดมภ์นี้ เกิดจากการนำค่าทำนายเสถียรภาพของเมทธอดในแต่ละคลาส มาหาค่ามากสุด น้อยสุดและค่าเฉลี่ย เช่น ในสดมภ์ Predict ของตารางที่ 4.8 ค่าทำนายเสถียรภาพของเมทธอดมากที่สุด น้อยที่สุด และค่าเฉลี่ย ในคลาส PuzzleSolver แสดงในแถวที่ 1 ของตารางที่ 4.10 คือ

ค่าเสถียรภาพของเมทธอดที่มากที่สุดในคลาส PuzzleSolver = 0.999

ค่าเสถียรภาพเฉลี่ยของเมทธอดในคลาส PuzzleSolver = 0.983

ค่าเสถียรภาพของเมทธอดที่น้อยที่สุดในคลาส PuzzleSolver = 0.906

ตารางที่ 4.10 ตัวอย่างข้อมูลที่ใช้ศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด

projName	ClassName	class Stability	Max MethodStability	Mean MethodStability	Min MethodStability
EightPuzzle	PuzzleSolver	0.75875485	0.999193116	0.983290756	0.90639558
EightPuzzle	PuzzleStack	0.9766537	0.999710772	0.995792682	0.98230045
EightPuzzle	ICallBack	1	1.000619481	1.000464611	1
EightPuzzle	EightPuzzle	0.9766537	0.999091291	0.991763297	0.972767596
EightPuzzle	EightPuzzleApp	0.98832685	1.000619481	0.996411908	0.992204335
EmailFinder	SearchThread	0.922043	1.000476926	0.982838798	0.96520067
EmailFinder	URLItem	1	1.000953852	1.000953852	1.000953852
EmailFinder	URLList	0.96774197	0.99890771	0.994867668	0.988077261
EmailFinder	WaitThread	0.9758065	1.000953852	0.991747714	0.982541575
TicTacToe	GameManager	0.78515625	0.9990299	0.985779718	0.976112315
TicTacToe	Bot	0.890625	0.997399734	0.988138806	0.975608715
TicTacToe	BotBoardPosition	0.90234375	0.996282962	0.988661451	0.982168341
TicTacToe	T_Runnable	1	0.999629016	0.999268363	0.99890771
TicTacToe	ActionListener	1	1.000456561	1.000456561	1.000456561
TicTacToe	MouseAdapter	0.9765625	0.998068331	0.998068331	0.998068331
Whiteboard	BoxTool	0.93240345	0.9990299	0.987869809	0.976816443
Whiteboard	ColorPaletteTool	0.9796137	1.000044847	0.996153594	0.985052191
Whiteboard	Command	0.99141634	0.987309812	0.979576793	0.97002706
Whiteboard	FontSelector	0.97639483	0.9990299	0.993927108	0.986246306
Whiteboard	FontSelectorDialog	0.9860515	0.992093342	0.987245974	0.982398606
Whiteboard	Helper	1	1.001549665	0.99942184	0.995631282
Whiteboard	LineTool	0.93562233	1.001200157	0.990285861	0.977013161
Whiteboard	LineWidthTool	0.9796137	0.9990299	0.994572828	0.984278321
Whiteboard	MainCanvas	0.97103006	0.997916948	0.990553787	0.974145441
Whiteboard	MainFrame	1	1.000619481	1.000395466	1.000171451
Whiteboard	OvalTool	0.93240345	0.9990299	0.987869809	0.976816443
Whiteboard	PaintMonitor	0.9796137	0.999629016	0.988914941	0.978200865

4.4.2 ผลการศึกษาความสัมพันธ์ระหว่างเสถียรภาพของคลาสและของเมทธอด

เมื่อได้ข้อมูลทั้ง 2 ส่วนในข้อ 4.4.1 แล้วจึงทำการวิเคราะห์สหสัมพันธ์ระหว่าง

- ค่าเสถียรภาพของเมทธอดที่มากที่สุดในคลาสกับค่าเสถียรภาพของคลาส
- ค่าเสถียรภาพเฉลี่ยของเมทธอดในคลาสกับค่าเสถียรภาพของคลาส
- ค่าเสถียรภาพของเมทธอดที่น้อยที่สุดในคลาสกับค่าเสถียรภาพของคลาส

โดยผลการวิเคราะห์สหสัมพันธ์แสดงดังตารางที่ 4.11 จากตารางจะพบว่า ค่าเสถียรภาพเฉลี่ยของเมทธอดมีความสัมพันธ์กับเสถียรภาพของคลาสมากที่สุด คือ 0.645 รองลงมาคือ ค่าเสถียรภาพของเมทธอดที่น้อยที่สุด มีค่าความสัมพันธ์เท่ากับ 0.627 ซึ่งทั้งสองค่ามีความสัมพันธ์ใกล้เคียงกัน ส่วนค่าเสถียรภาพของเมทธอดที่มากที่สุด มีความสัมพันธ์กับเสถียรภาพของคลาสน้อยที่สุด คือ เท่ากับ 0.375

ตารางที่ 4.11 ความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอด

Correlations

		classStability	PRE_MAX	PRE_MEAN	PRE_MIN
classStability	Pearson Correlation	1	.375**	.645**	.627**
	Sig. (2-tailed)	.	.000	.000	.000
	N	275	275	275	275
PRE_MAX	Pearson Correlation	.375**	1	.702**	.241**
	Sig. (2-tailed)	.000	.	.000	.000
	N	275	275	275	275
PRE_MEAN	Pearson Correlation	.645**	.702**	1	.787**
	Sig. (2-tailed)	.000	.000	.	.000
	N	275	275	275	275
PRE_MIN	Pearson Correlation	.627**	.241**	.787**	1
	Sig. (2-tailed)	.000	.000	.000	.
	N	275	275	275	275

**. Correlation is significant at the 0.01 level (2-tailed).

บทที่ 5

การออกแบบและพัฒนาเครื่องมือจัดเก็บมาตรวัดและเครื่องมือคำนวณเสถียรภาพ

สำหรับเครื่องมือในการเก็บค่ามาตรวัดได้นำเครื่องมือในการเก็บมาตรวัด MTOOP2 [6] มาทำการปรับปรุง โดยทำการปรับปรุงข้อกำหนดภาษาจาวาให้ทันสมัยขึ้นจากข้อกำหนดภาษาจาวารุ่น 1.2 เป็นข้อกำหนดภาษารุ่น 1.4 และทำการเพิ่มการนับมาตรวัดชุด NMI และ NAI สำหรับการปรับปรุงข้อกำหนดภาษาทำให้ต้องมีการสร้างพาร์สเซอร์ของภาษาจาวาใหม่ ซึ่งมีรายละเอียดในหัวข้อ 5.1 เครื่องมือเก็บค่ามาตรวัดมีรายละเอียดในหัวข้อ 5.2

สำหรับเครื่องมือคำนวณเสถียรภาพ ได้นำเครื่องมือจัดเก็บมาตรวัดที่ทำการปรับปรุงข้อกำหนดภาษาให้ทันสมัยขึ้นแล้วนั้น นำส่วนของพาร์เซอร์ และซินแท็กซ์ทรี มาเพิ่มเติมการหาผลกระทบจากการเปลี่ยนแปลงและคำนวณเสถียรภาพของเมทธอดและคลาส โดยรายละเอียดจะอยู่ในหัวข้อ 5.3

5.1 การสร้างพาร์สเซอร์ภาษาจาวา [5][6]

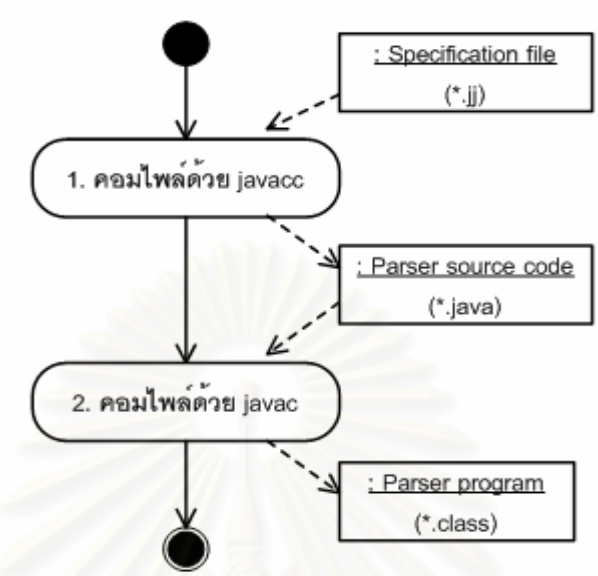
5.1.1 โปรแกรมจาวาซีซี (JavaCC)

ในการสร้างพาร์สเซอร์ได้นำโปรแกรมจาวาซีซี(JavaCC : Java Compiler Compiler) มาใช้ โดยโปรแกรมจาวาซีซีจะใช้ในการสร้างพาร์สเซอร์ภาษาจาวา ตามข้อกำหนดภาษาจาวา โดยขั้นตอนการใช้โปรแกรมจาวาซีซี แสดงดังรูปที่ 5.1 เริ่มต้นจากการนำไฟล์ข้อกำหนดภาษาจาวา (xxx.jj) มาคอมไพล์ผ่าน จาวาซีซี ซึ่งจะให้ผลลัพธ์เป็นโปรแกรมต้นฉบับของจาวาพาร์สเซอร์ (xxx.java) หลังจากนั้นจึงนำไฟล์โปรแกรมต้นฉบับมาคอมไฟล์ด้วยจาวาซี ซึ่งจะได้ จาวาคลาส (xxx.class) เพื่อนำไปใช้งานต่อไป

5.1.2 ข้อกำหนดภาษาจาวา

ข้อกำหนดภาษาจาวาที่นำมาใช้ในงานวิจัยนี้ เป็นรุ่น 1.4(java1.4.jj) ได้นำมาจาก [15] ซึ่งไฟล์นี้จะเป็นรายละเอียดเกี่ยวกับข้อกำหนดโทเคน (lexical specifications) และข้อกำหนด

ไวยากรณ์ (grammar specifications) ที่เขียนอยู่ในรูปแบบของบีเอ็นเอฟ (BNF : Backus Naur Form) โดยสามารถสรุปเป็นแผนภาพต้นไม้ของไวยากรณ์ภาษาจาวาได้ดังภาคผนวก ก



รูปที่ 5.1 ขั้นตอนการใช้โปรแกรมจาวาซีซี

5.1.3 การประยุกต์ใช้งานข้อกำหนดภาษาจาวา

สำหรับไฟล์ข้อกำหนดภาษาจาวา (java1.4.jj) ที่ได้นำมานี้ เมื่อผ่านทั้งสามขั้นตอนในรูปที่ 5.1 แล้วพาร์สเซอร์ที่ได้สามารถใช้ในการตรวจสอบรหัสโปรแกรมภาษาจาวา ว่ารหัสโปรแกรมที่นำมาตรวจสอบนั้น ถูกต้องตามไวยากรณ์ภาษาจาวาหรือไม่ แต่จะไม่สามารถเก็บข้อมูลต่าง ๆ ในรหัสโปรแกรมนั้นได้ ดังนั้นหากต้องการเก็บข้อมูลในรหัสโปรแกรมต้องมีการแก้ไขพาร์สเซอร์ โดยวิธีแก้ไขสามารถทำได้ 2 วิธี คือ

1. แก้ไขเพิ่มเติมลงในไฟล์ข้อกำหนดภาษาจาวา คือ แก้ไขลงในไฟล์ java1.4.jj ในขั้นตอนที่ 1 ของรูปที่ 5.1 โดยจะต้องเขียนในรูปของบีเอ็นเอฟ

2. แก้ไขเพิ่มเติมรหัสโปรแกรมภาษาจาวา ลงในรหัสโปรแกรมต้นแบบจาวาพาร์สเซอร์ ซึ่งจะอยู่ในขั้นตอนที่ 2 ของรูปที่ 5.1 คือหลังจากนำข้อกำหนดภาษาจาวา(java1.4.jj) มาคอมไพล์ด้วยโปรแกรมจาวาซีซี จะได้ไฟล์รหัสโปรแกรมต้นแบบจาวาพาร์สเซอร์ แล้วทำการแก้ไขไฟล์ JavaParser.java

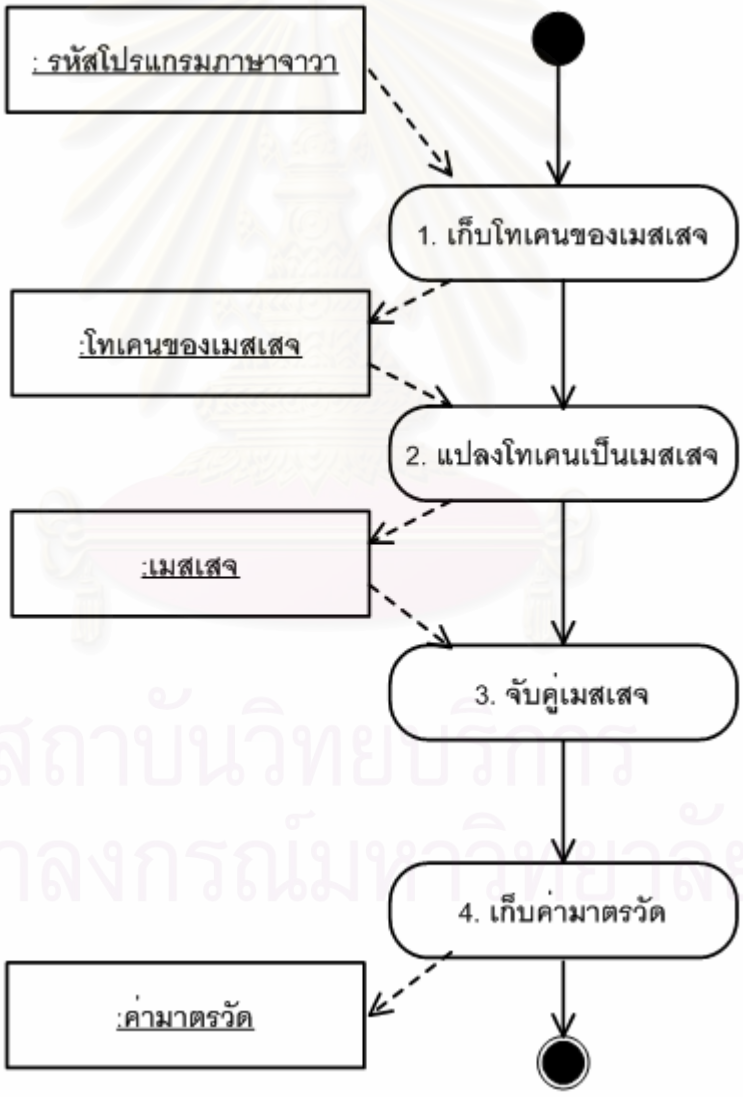
สำหรับตัวอย่างของการแก้ไขพาร์สเซอร์ทั้งสองแบบ สามารถศึกษาได้จาก [5][6] โดยงานวิจัยนี้เลือกการแก้ไขแบบที่สอง เนื่องจากเป็นการแก้ไขเพิ่มเติมรหัสโปรแกรมภาษาจาวา ซึ่งสะดวกมากกว่าแบบที่หนึ่ง

5.2 การออกแบบและพัฒนาเครื่องมือจัดเก็บมาตรวัด

ในการออกแบบและพัฒนาเครื่องมือจัดเก็บมาตรวัด จะกล่าวถึงเฉพาะส่วนที่เพิ่มเติมจาก MTOOP2 [6] คือ การเก็บค่ามาตรวัดชุด NMI และ NAI โดยอธิบายถึงขั้นตอนในการเก็บค่ามาตรวัดจากรหัสโปรแกรม แผนภาพคลาสและแผนภาพซีเควนซ์ของเครื่องมือ รวมถึงรายละเอียดของแต่ละแผนภาพ ซึ่งมีรายละเอียด ดังนี้

5.2.1 ขั้นตอนการเก็บค่ามาตรวัด

ขั้นตอนการเก็บค่ามาตรวัดชุด NMI และ NAI สามารถแสดงได้ดังในรูปที่ 5.2



รูปที่ 5.2 ขั้นตอนการเก็บค่ามาตรวัด NMI และ NAI

5.2.2.1 เก็บโทเคนของเมสเสจ

ในขั้นตอนแรกของการเก็บค่ามาตรวัด จะเก็บโทเคนที่เป็นการเรียกใช้เมสเสจ โดยทำการแก้ไขเพิ่มเติมรหัสโปรแกรมจาวา ลงในรหัสโปรแกรมต้นแบบจาวาพาร์สเซอร์โดยส่วนหลักจะปรากฏที่เมทธอด PrimaryPrefix และ PrimarySuffix ของไฟล์ JavaParser สำหรับโทเคนที่เก็บได้จะมีการกำหนดประเภทของโทเคนไว้ด้วย เพื่อนำไปใช้ในการขั้นตอนของการแปลงโทเคนไปเป็นเมสเสจในขั้นตอน 5.2.2.2 โดยการแก้ไขเพิ่มเติมรหัสโปรแกรมจาวา ค่าของโทเคนที่ถูกเก็บและประเภทของแต่ละโทเคนมีรายละเอียดในภาคผนวก ข

5.2.2.2 แปลงโทเคนเป็นเมสเสจ

หลังจากเก็บโทเคนจากทุกเมสเสจของทุก ๆ คลาสเรียบร้อยแล้ว จึงนำโทเคนที่เก็บได้แล้วนั้นมาแปลงเป็นเมสเสจ สำหรับเมสเสจที่ทำการแปลงจากโทเคนแล้ว จะอยู่ในรูปแบบดังนี้

C = P S*

P, S = M

M = N [A]

A = C [“,” C]*

N = Name

C (Complex message) คือ เมสเสจแบบซับซ้อน ซึ่งประกอบด้วยพรีฟิกส์และซับฟิกส์ โดยจะมีซับฟิกส์หรือไม่ก็ได้

P (Prefix) คือ พรีฟิกส์ เป็นส่วนหน้าของเมสเสจ ซึ่งมีรูปแบบเป็นเมสเสจ (M)

S (Suffix) คือ ซับฟิกส์ เป็นส่วนหลังของเมสเสจ ซึ่งมีรูปแบบเป็นเมสเสจ (M)

M (Message) คือ เมสเสจ ซึ่งประกอบด้วย ชื่อและอาร์กิวเมนต์ ในส่วนของอาร์กิวเมนต์จะมีหรือไม่ก็ได้ สำหรับ M นี้มีแบ่งประเภทได้ 3 แบบคือ

- แอทริบิวต์ จะเก็บชื่อแอทริบิวต์และจะไม่มีอาร์กิวเมนต์
- เมทธอด จะเก็บชื่อเมทธอด ส่วนอาร์กิวเมนต์จะมีหรือไม่ก็ได้

- อาร์เรย์ จะเก็บชื่อแอทริบิวต์ ส่วนอินเด็กซ์ของอาร์เรย์จะเก็บในรูปแบบเดียวกับอาร์กิวเมนต์ของเมทธอด ซึ่งจะมีหรือไม่ก็ได้

N (Name) คือ ชื่อแอทริบิวต์หรือชื่อเมทธอด

A (Argument) คือ อาร์กิวเมนต์ของเมทธอด โดยจะมีหรือไม่ก็ได้ ถ้ามีจะเป็นเมสเสจแบบซับซ้อน(C) อย่างน้อย 1 ตัว และหากมีอาร์กิวเมนต์มากกว่า 1 ตัว จะต้องมีเครื่องหมาย , คั่น

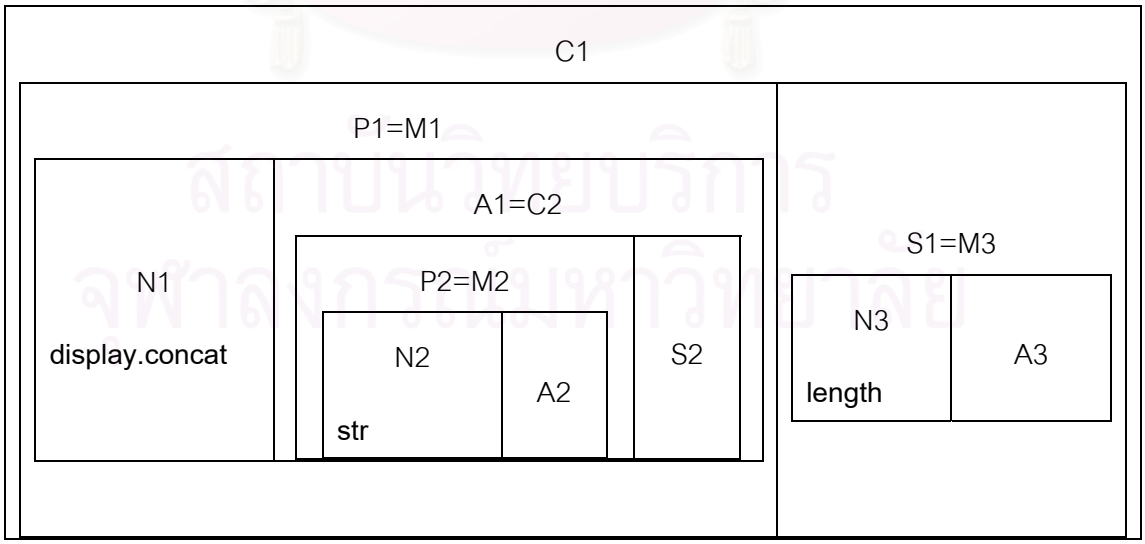
ตัวอย่างเช่น เมสเสจ display.concat(str).length() รูปแบบการเก็บเมสเสจสามารถแสดงได้ดังรูปที่ 5.3 โดย

ใน เมสเสจแบบซับซ้อน C1 จะประกอบด้วยพรีฟิกส์กับซับฟิกส์อย่างละตัวคือ P1 และ S1 โดยทั้ง P1 และ S1 ต่างก็เป็นเมสเสจคือ M1 และ M3 ตามลำดับ

เมสเสจ M1 เก็บชื่อเมทธอด N1คือ display.concat และมีอาร์กิวเมนต์หนึ่งตัวคือ A1 ซึ่งเป็นเมสเสจแบบซับซ้อน C2

เมสเสจแบบซับซ้อน C2 จะประกอบด้วยพรีฟิกส์หนึ่งตัว คือ P2 ไม่มีซับฟิกส์ ซึ่ง P2 เป็นเมสเสจคือ M2และเก็บชื่อของตัวแปร N2 คือ str โดยไม่มีอาร์กิวเมนต์ A2

ส่วนเมสเสจ M3 จะเก็บชื่อเมทธอด คือ length โดยไม่มีอาร์กิวเมนต์ A3



รูปที่ 5.3 ตัวอย่างเมสเสจที่ได้จากการแปลงจากโทเคนที่เก็บจากรหัสโปรแกรม

5.2.2.3 จับคู่เมสเสจ

หลังจากที่ทำการแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว จึงนำเมสเสจที่ได้มาจับคู่เพื่อหาว่าเมสเสจนั้น ๆ เป็นการเรียกใช้งานแอทริบิวต์หรือเมทธอดใด โดยในการจับคู่เมสเสจจะแบ่งเป็น 3 ประเภทตามชนิดของเมสเสจคือ เมสเสจที่เป็นแอทริบิวต์ เมสเสจที่เป็นอาร์เรย์ และเมสเสจที่เป็นเมทธอด

1. เมสเสจที่เป็นแอทริบิวต์

เมสเสจที่เป็นแอทริบิวต์ แบ่งได้ 2 ส่วนคือ

- โทเคนแรก จะเป็นชื่อคลาส ตัวแปรโลคอล ตัวแปรโกลบอลในคลาสนั้นหรือในคลาสที่สืบทอดมาก็ได้
- โทเคนถัด ๆ ไป จะเป็นตัวแปรโกลบอลในคลาสจากโทเคนก่อนหน้าหรือคลาสที่โทเคนก่อนหน้าสืบทอดมา

2. เมสเสจที่เป็นอาร์เรย์

เมสเสจที่เป็นอาร์เรย์ แบ่งได้ 2 ส่วนคือ

- ส่วนก่อนเครื่องหมาย [] จะเป็นเมสเสจที่เป็นแอทริบิวต์
- ส่วนในเครื่องหมาย [] จะเป็นเมสเสจ

3. เมสเสจที่เป็นเมทธอด

เมสเสจที่เป็นเมทธอด แบ่งได้ 3 ส่วน คือ

- ส่วนก่อนเครื่องหมาย () ก่อนโทเคนสุดท้าย จะเป็นเมสเสจที่เป็นแอทริบิวต์
- ส่วนก่อนเครื่องหมาย () โทเคนสุดท้าย จะเป็นชื่อเมทธอด
- ส่วนในเครื่องหมาย () จะเป็นอาร์กิวเมนต์ของเมทธอด ซึ่งจะเป็นเมสเสจ

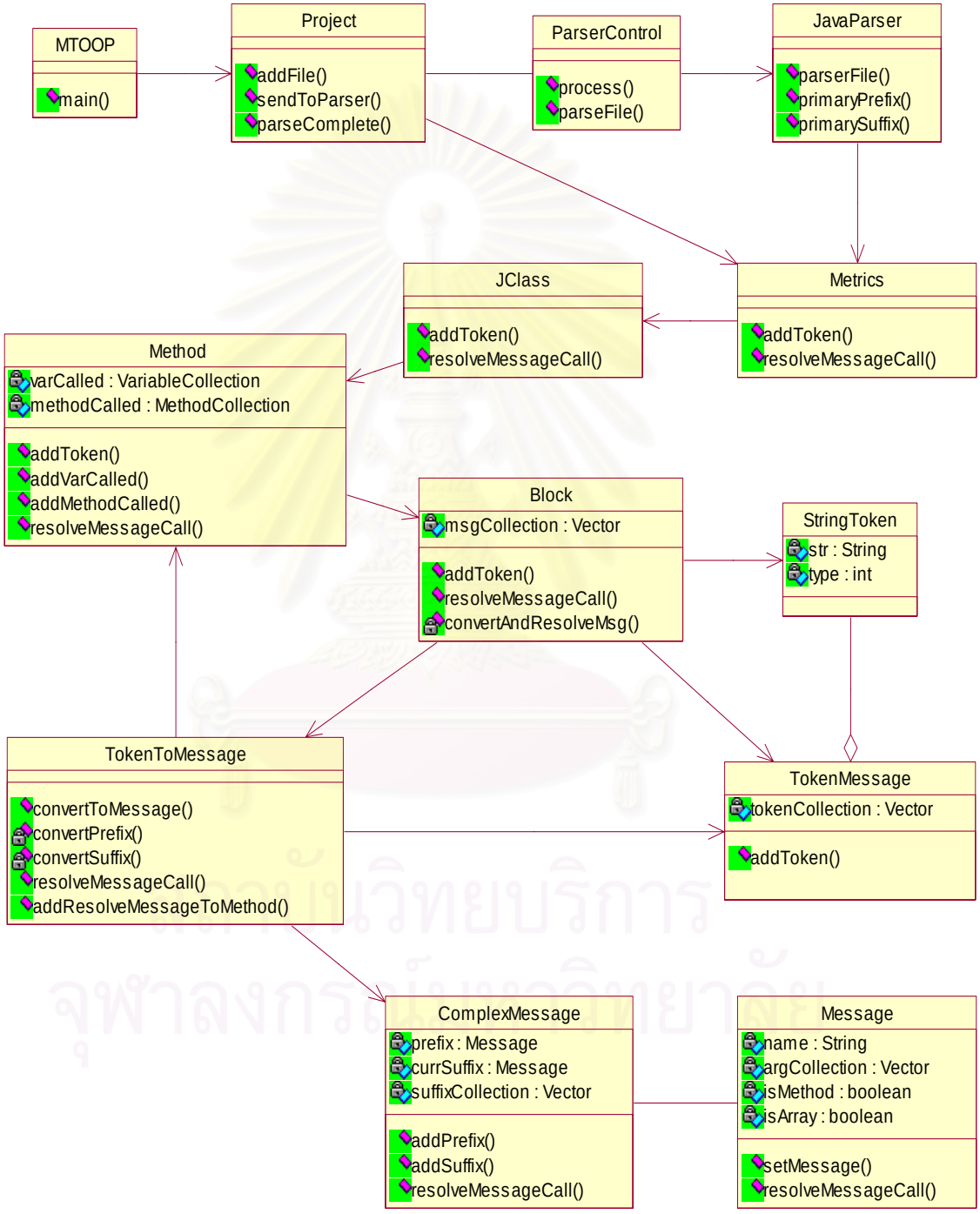
ทั้งนี้ ได้ทำการสรุปประเภทของเมสเสจทั้ง 3 แบบ เป็นแผนภาพต้นไม้พร้อมรายละเอียดในภาคผนวก ค

5.2.2.4 เก็บค่ามาตรวัด

หลังจากทำการจับคู่เมสเสจเรียบร้อยแล้ว จึงทำการนับค่ามาตรวัด NAI และ NMI ตามรายละเอียดในหัวข้อ3.1

5.2.2 แผนภาพคลาสของเครื่องมือจัดเก็บมาตรวัด

แผนภาพคลาสของเครื่องมือจัดเก็บมาตรวัด ในส่วนที่เกี่ยวข้องกับการเก็บมาตรวัดชุด NMI และ NAI แสดงได้ดังรูปที่ 5.4 และรายละเอียดของแต่ละคลาสแสดงได้ดังตารางที่ 5.1 ถึงตารางที่ 5.13



รูปที่ 5.4 แผนภาพคลาสของเครื่องมือเก็บมาตรวัด

ตารางที่ 5.1 รายละเอียดของคลาส MTOOP

ชื่อคลาส	MTOOP
คำอธิบาย	เป็นคลาสหลักของเครื่องมือ ซึ่งเป็นคลาสแรกที่ถูกเรียกใช้งาน
เมทธอด	
main	เป็นเมทธอดหลัก และเป็นเมทธอดที่ถูกเรียกใช้งานเป็นเมทธอดแรกของเครื่องมือ

ตารางที่ 5.2 รายละเอียดของคลาส Project

ชื่อคลาส	Project
คำอธิบาย	เป็นคลาสที่ควบคุมการทำงานของเครื่องมือ ในภาพรวม
เมทธอด	
addFile	เป็นเมทธอด ที่ทำหน้าที่เกี่ยวกับการเพิ่มไฟล์ที่ต้องการเก็บมาตรวัด
sendToParser	เป็นเมทธอดที่นำหน้าที่ส่งไฟล์ที่ต้องการเก็บมาตรวัดไปให้คลาส ParserControl
parseComplete	เป็นเมทธอดที่ถูกเรียกใช้เมื่อทำการเก็บโทเคนต่าง ๆ ในรหัสโปรแกรมเสร็จ แล้วจะทำงานในขั้นตอนอื่น ๆ ต่อไป

ตารางที่ 5.3 รายละเอียดของคลาส ParserControl

ชื่อคลาส	ParserControl
คำอธิบาย	เป็นคลาสที่ทำหน้าที่ติดต่อและควบคุม ไฟล์ที่จะส่งไปยังพาร์สเซอร์
เมทธอด	
process	เป็นเมทธอดที่จัดการกับไฟล์ก่อนที่จะส่งไปยังพาร์เซอร์
parseFile	เป็นเมทธอดที่ส่งไฟล์ไปยังพาร์สเซอร์

ตารางที่ 5.4 รายละเอียดของคลาส JavaParser

ชื่อคลาส	JavaParser
คำอธิบาย	เป็นคลาสที่ทำหน้าที่ท่องไปในรหัสโปรแกรม
เมทธอด	
parseFile	เป็นเมทธอดที่เริ่มต้นในการท่องในรหัสโปรแกรม
primaryPrefix	เป็นเมทธอดที่เก็บโทเคนในส่วนพรีฟิกส์ของเมสเสจจากรหัสโปรแกรม
primarySuffix	เป็นเมทธอดที่เก็บโทเคนในส่วนซับฟิกส์ของเมสเสจจากรหัสโปรแกรม

ตารางที่ 5.5 รายละเอียดของคลาส Metrics

ชื่อคลาส	Metrics
คำอธิบาย	เป็นคลาสที่ใช้จัดการกับโทเคนที่เก็บมาจากรหัสโปรแกรม
เมทธอด	
addToken	เป็นเมทธอดที่รับโทเคนของเมสเสจที่เก็บได้ ส่งไปยังคลาส JClass
resolveMessageCall	เป็นเมทธอดที่ส่งคำสั่งการแปลงโทเคนที่เก็บไว้เป็นเมสเสจ และการจับคู่เมสเสจไปยังคลาส JClass

ตารางที่ 5.6 รายละเอียดของคลาส JClass

ชื่อคลาส	JClass
คำอธิบาย	เป็นคลาสที่เก็บโครงสร้างของคลาส
เมทธอด	
addToken	เป็นเมทธอดที่รับโทเคนของเมสเสจที่เก็บได้ ส่งไปยังคลาส Method
resolveMessageCall	เป็นเมทธอดที่ส่งคำสั่งการแปลงโทเคนเป็นเมสเสจ และการจับคู่เมสเสจไปยังคลาส Method

ตารางที่ 5.7 รายละเอียดของคลาส Method

ชื่อคลาส	Method
คำอธิบาย	เป็นคลาสที่เก็บโครงสร้างของเมทธอด
แอทริบิวต์	
varCalled	เป็นแอทริบิวต์ที่เก็บแอทริบิวต์ที่ถูกเรียกใช้งานในเมทธอด
methodCalled	เป็นแอทริบิวต์ที่เก็บเมทธอดที่ถูกเรียกใช้งานในเมทธอด
เมทธอด	
addToken	เป็นเมทธอดที่รับโทเคนของเมสเสจที่เก็บได้ ส่งไปยังคลาส Block
addVarCalled	เป็นเมทธอดที่เพิ่มแอทริบิวต์ที่ถูกเรียกใช้งาน
addMethodCalled	เป็นเมทธอดที่เพิ่มเมทธอดที่ถูกเรียกใช้งาน
resolveMessageCall	เป็นเมทธอดที่ส่งคำสั่งการแปลงโทเคนเป็นเมสเสจ และการจับคู่เมสเสจไปยังคลาส Block

ตารางที่ 5.8 รายละเอียดของคลาส StringToken

ชื่อคลาส	StringToken
คำอธิบาย	เป็นคลาสที่เก็บโทเคนและประเภทของโทเคนที่เก็บได้จากรหัสโปรแกรม เพื่อนำไปแปลงเป็นเมสเสจต่อไป
แอทริบิวต์	
str	เป็นแอทริบิวต์ที่เก็บโทเคนที่ได้จากรหัสโปรแกรม
type	เป็นแอทริบิวต์ที่เก็บประเภทของโทเคนที่เก็บได้

ตารางที่ 5.9 รายละเอียดของคลาส Block

ชื่อคลาส	Block
คำอธิบาย	เป็นคลาสที่เก็บโครงสร้างของบล็อค
แอทริบิวต์	
msgCollection	เป็นแอทริบิวต์ที่เก็บโทเคนของทุกเมสเสจที่มีการเรียกใช้ในบล็อค
เมทธอด	
addToken	เป็นเมทธอดที่นำโทเคนที่เก็บมา สร้างเป็นออบเจ็คของ StringToken เพื่อเก็บโทเคนของแต่ละเมสเสจ
resolveMessageCall	เป็นเมทธอดที่ส่งคำสั่งการแปลงโทเคนเป็นเมสเสจ และการจับคู่เมสเสจไปยังเมทธอด convertAndResolveMsg
convertAndResolveMsg	เป็นเมทธอดที่ทำการแปลงโทเคนที่เก็บไว้เป็นเมสเสจและทำการจับคู่เมสเสจ

ตารางที่ 5.10 รายละเอียดของคลาส TokenToMessage

ชื่อคลาส	TokenToMessage
คำอธิบาย	เป็นคลาสที่นำโทเคนของเมสเสจที่เก็บได้ มาทำการแปลงเป็นเมสเสจและทำการจับคู่เมสเสจ
เมทธอด	
convertToMessage	เป็นเมทธอดที่ทำการแปลงโทเคนของเมสเสจที่เก็บไว้ ให้เป็นเมสเสจ
convertPrefix	เป็นเมทธอดที่ทำการแปลงโทเคนของเมสเสจในส่วนที่เป็นพรีฟิกส์
convertSuffix	เป็นเมทธอดที่ทำการแปลงโทเคนของเมสเสจในส่วนที่เป็นซับฟิกส์
resolveMessgaeCall	เป็นเมทธอดที่ใช้ในการจับคู่เมสเสจหลังจากแปลงโทเคนมาเป็นเมสเสจเรียบร้อยแล้ว
addResolveMessage ToMethod	เป็นเมทธอดที่ทำการเพิ่มแอทริบิวต์และเมทธอดที่ถูกเรียกใช้ในเมทธอด

ตารางที่ 5.11 รายละเอียดของคลาส TokenMessage

ชื่อคลาส	TokenMessage
คำอธิบาย	เป็นคลาสที่เก็บโทเคนของเมสเสจ ก่อนที่จะนำไปแปลงเป็นเมสเสจต่อไป
แอทริบิวต์	
tokenCollection	เป็นแอทริบิวต์ที่เก็บโทเคนของเมสเสจ
เมทธอด	
addToken	เป็นเมทธอดที่ทำการเก็บออบเจ็คของ StringToken ไว้

ตารางที่ 5.12 รายละเอียดของคลาส ComplexMessage

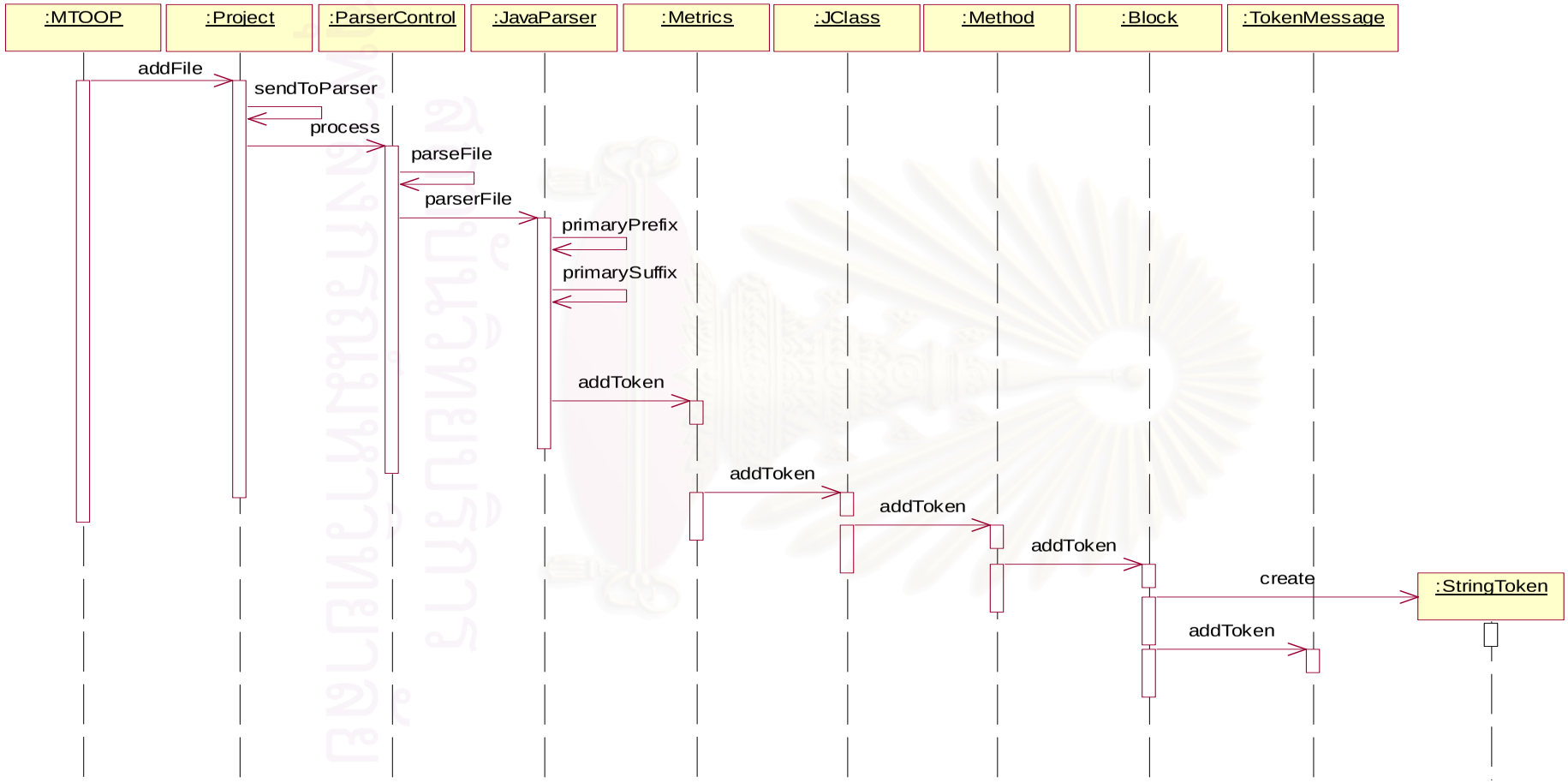
ชื่อคลาส	ComplexMessage
คำอธิบาย	เป็นคลาสที่เก็บเมสเสจหลังจากแปลงจากโทเคนแล้ว โดยเมสเสจที่เก็บจะเป็นเมสเสจแบบซับซ้อน ซึ่งจะนำไปใช้ในการจับคู่เมสเสจต่อไป
แอทริบิวต์	
prefix	เป็นแอทริบิวต์ที่เก็บเมสเสจในส่วนที่เป็นพรีฟิกส์หลังจากแปลงโทเคนแล้ว
currSuffix	เป็นแอทริบิวต์ที่เก็บเมสเสจในส่วนที่เป็นซับฟิกส์ของเมสเสจ ในขณะที่กำลังแปลงโทเคนไปเป็นเมสเสจ
suffixCollection	เป็นแอทริบิวต์ที่เก็บซับฟิกส์ของเมสเสจ
เมทธอด	
addPrefix	เป็นเมทธอดที่เพิ่มส่วนที่เป็นพรีฟิกส์ของเมสเสจ
addSuffix	เป็นเมทธอดที่เพิ่มส่วนที่เป็นซับฟิกส์ของเมสเสจ
resolveMessageCall	เป็นเมทธอดที่จับคู่เมสเสจหลังจากที่แปลงโทเคนเป็น เมสเสจเรียบร้อยแล้ว

ตารางที่ 5.13 รายละเอียดของคลาส Message

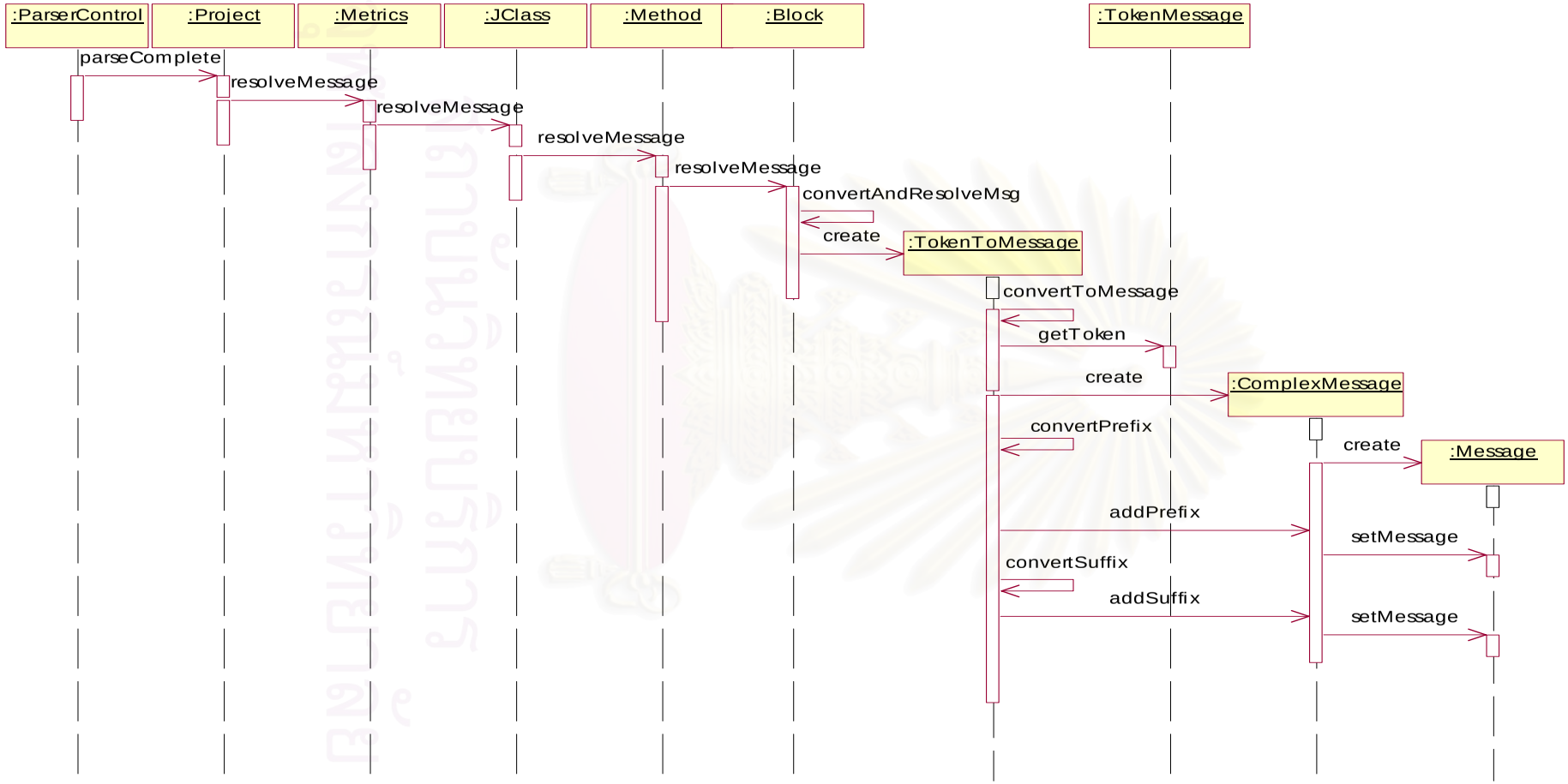
ชื่อคลาส	Message
คำอธิบาย	เป็นคลาสที่เก็บเมสเสจที่ทำการแปลงจากโทเคนแล้ว ซึ่งจะนำไปใช้ในการจับคู่เมสเสจต่อไป
แอทริบิวต์	
name	เป็นแอทริบิวต์ที่เก็บชื่อของแอทริบิวต์หรือชื่อของเมทธอด
argCollection	เป็นแอทริบิวต์ที่เก็บอาร์กิวเมนต์ของเมสเสจที่เป็นเมทธอด หรือเก็บอินเด็กซ์ของอาร์เรย์
isMethod	เป็นแอทริบิวต์ที่แสดงว่าเมสเสจที่เก็บนี้เป็นเมทธอดหรือไม่ - ถ้าเป็นจริง เมสเสจที่เก็บจะเป็นเมทธอด - ถ้าเป็นเท็จ เมสเสจที่เก็บอาจเป็นแอทริบิวต์หรืออาร์เรย์
isArray	เป็นแอทริบิวต์ที่แสดงว่าเมสเสจที่เก็บนี้เป็นอาร์เรย์หรือไม่ - ถ้าเป็นจริง เมสเสจที่เก็บจะเป็นอาร์เรย์ - ถ้าเป็นเท็จ เมสเสจที่เก็บอาจเป็นแอทริบิวต์หรือเมทธอด
เมทธอด	
setMessage	เป็นเมทธอดที่เก็บชื่อของแอทริบิวต์หรือชื่อของเมทธอด โดยกำหนดค่าให้กับแอทริบิวต์ name
resolveMessageCall	เป็นเมทธอดที่จับคู่เมสเสจ หลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว

5.2.3 แผนภาพซีเควนซ์ของเครื่องมือจัดเก็บมาตรวัด

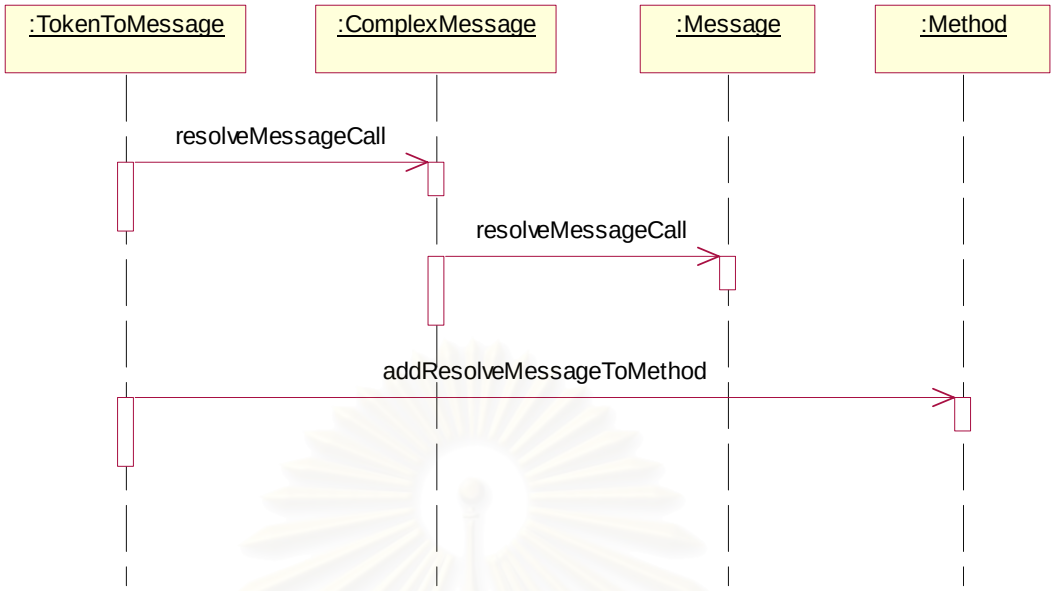
ในหัวข้อนี้แสดงแผนภาพซีเควนซ์ของเครื่องมือจัดเก็บมาตรวัด ขั้นตอนการเก็บโทเคนของเมสเสจ ในรูปที่ 5.5 ขั้นตอนการนำโทเคนของเมสเสจที่เก็บได้ มาแปลงเป็นเมสเสจ ในรูปที่ 5.6 และขั้นตอนของการจับคู่เมสเสจ ในรูปที่ 5.7



รูปที่ 5.5 แผนภาพซีเควนซ์แสดงการจัดเก็บโทเคนของเมสเสจ



รูปที่ 5.6 แผนภาพซีเควนซ์แสดงการแปลงโทเคนเป็นเมสเสจ



รูปที่ 5.7 แผนภาพซีเควนซ์แสดงการจับคู่เมสเสจ

จากรูปที่ 5.5 เป็นแผนภาพซีเควนซ์การเก็บโทเคนของเมสเสจ โดยการทำงานจะเริ่มจากคลาส MTOOP ซึ่งเป็นคลาสหลัก จะเรียกเมทธอด addFile ในคลาส Project ซึ่งคลาส Project นี้จะเป็นคลาสที่ควบคุมการทำงานของโปรแกรม หลังจากนั้นจึงเรียกเมทธอด sendToParser ในคลาสตัวเองเพื่อเตรียมไฟล์ที่จะทำการเก็บโทเคน หลังจากนั้นจึงเรียกเมทธอด process ในคลาส ParserControl ซึ่งคลาสนี้จะทำหน้าที่ควบคุมการทำงานของพาร์เซอร์ โดยเมทธอด process จะเรียกเมทธอด parseFile ในคลาสตัวเองโดยจะส่งไฟล์รหัสโปรแกรมที่ต้องการเก็บโทเคนไปทีละไฟล์ หลังจากนั้นคลาส ParserControl จะทำการเรียกเมทธอด parseFile ในคลาส JavaParser ซึ่งเป็นคลาสพาร์เซอร์ หลังจากนั้นเมทธอด parseFile จะทำการเรียกเมทธอด primaryPrefix และ primarySuffix ในคลาส JavaParser โดยการจะเรียกเมทธอดใดนั้นจะขึ้นอยู่กับ โทเคนที่ต้องการเก็บในขณะนั้นเป็นเมสเสจในส่วนของพรีฟิกส์หรือซับฟิกส์ เมื่อเจอโทเคนที่ต้องการเก็บแล้วจะเรียกเมทธอด addToken ในคลาส Metrics ซึ่งจะเรียกเมทธอด addToken ของคลาส JClass Method และ คลาส Block ต่อ ๆ กันไปตามลำดับ เนื่องจาก โทเคนของเมสเสจจะถูกเก็บไว้ในโครงสร้างที่เป็นบล็อก ที่อยู่ในคลาส Block เมื่อเมทธอด addToken ของคลาส Block ถูกเรียกใช้งาน จะสร้างวัตถุของคลาส StringToken แล้วนำโทเคนที่ได้นั้น มาเก็บไว้ในคลาส TokenMessage เพื่อไว้ทำการแปลงเป็นเมสเสจในภายหลัง

รูปที่ 5.6 เป็นแผนภาพซีเควนซ์การแปลงโทเคนที่เก็บได้จากรูปที่ 5.5 มาเป็นเมสเสจ โดยคลาส ParserControl จะเรียกเมทธอด parseComplete ที่คลาส Project เมื่อทำการเก็บโทเคน

ของเมสเสจในทุก ๆ คลาส เรียบร้อยแล้ว หลังจากนั้นจึงเรียกใช้เมทธอด resolveMessage ในคลาส Metrics JClass Method และ Block ตามลำดับ เนื่องจากโทเคนที่เก็บมาได้นั้น อยู่ที่คลาส Block หลังจากนั้นคลาส Block จะเรียกเมทธอด convertAndResolveMsg ในคลาสตัวเอง เพื่อทำการแปลงโทเคนเป็นเมสเสจและจับคู่เมสเสจ สำหรับการแปลงโทเคนเป็นเมสเสจ คลาส Block จะสร้างวัตถุ TokenToMessage และส่งโทเคนของเมสเสจที่เก็บในคลาส TokenMessage ไปทำการแปลงเป็นเมสเสจต่อไป ในขณะที่สร้างวัตถุของ TokenToMessage คลาส TokenToMessgae ก็จะเรียกเมทธอด convertToMessage เพื่อทำการแปลงโทเคนที่ได้รับมาจากการเรียกเมทธอด getToken ในคลาส TokenMessage ไปเป็นเมสเสจ ในการแปลงนั้นจะสร้างวัตถุ ComplexMessage และวัตถุ ComplexMessage จะสร้างวัตถุ Messgae ต่อเนื่องไป เมื่อสร้างวัตถุของทั้งสองคลาสเสร็จแล้วจึงเริ่มการแปลงโทเคน โดยนำโทเคนส่วนพรีฟิกส์มาแปลงก่อน ผ่านการเรียกใช้เมทธอด ConvertPrefix ซึ่งจะเรียกใช้เมทธอด addPrefix ในคลาส ComplexMessage และเมทธอด addPrefix จะเรียกเมทธอด setMessage ในคลาส Message ต่อไป เมื่อแปลงในส่วนพรีฟิกส์เสร็จแล้วคลาส TokenToMessage ก็จะเรียกเมทธอด ConvertSuffix เพื่อทำการแปลงในส่วนซับฟิกส์ของเมสเสจต่อไป ด้วยการเรียกเมทธอด addSuffix ในคลาส ComplexMessage และเมทธอด setMessage ในคลาส Message ตามลำดับ

รูปที่ 5.7 เป็นแผนภาพซีเควนซ์ ในการจับคู่เมสเสจ หลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว คลาส TokenToMessage จะเรียกเมทธอด resolveMessageCall ในคลาส ComplexMessage และเรียกเมทธอด resolveMessageCall ในคลาส Message เพื่อทำการจับคู่เมสเสจ หลังจากจับคู่เมสเสจเรียบร้อยแล้ว คลาส TokenToMessage ก็จะเรียกเมทธอด addResolveMessageToMethod ในคลาส Method เพื่อเก็บเมสเสจ (แอทริบิวต์และเมทธอด) ที่ถูกเรียกใช้งานในเมทธอดนั้น ๆ

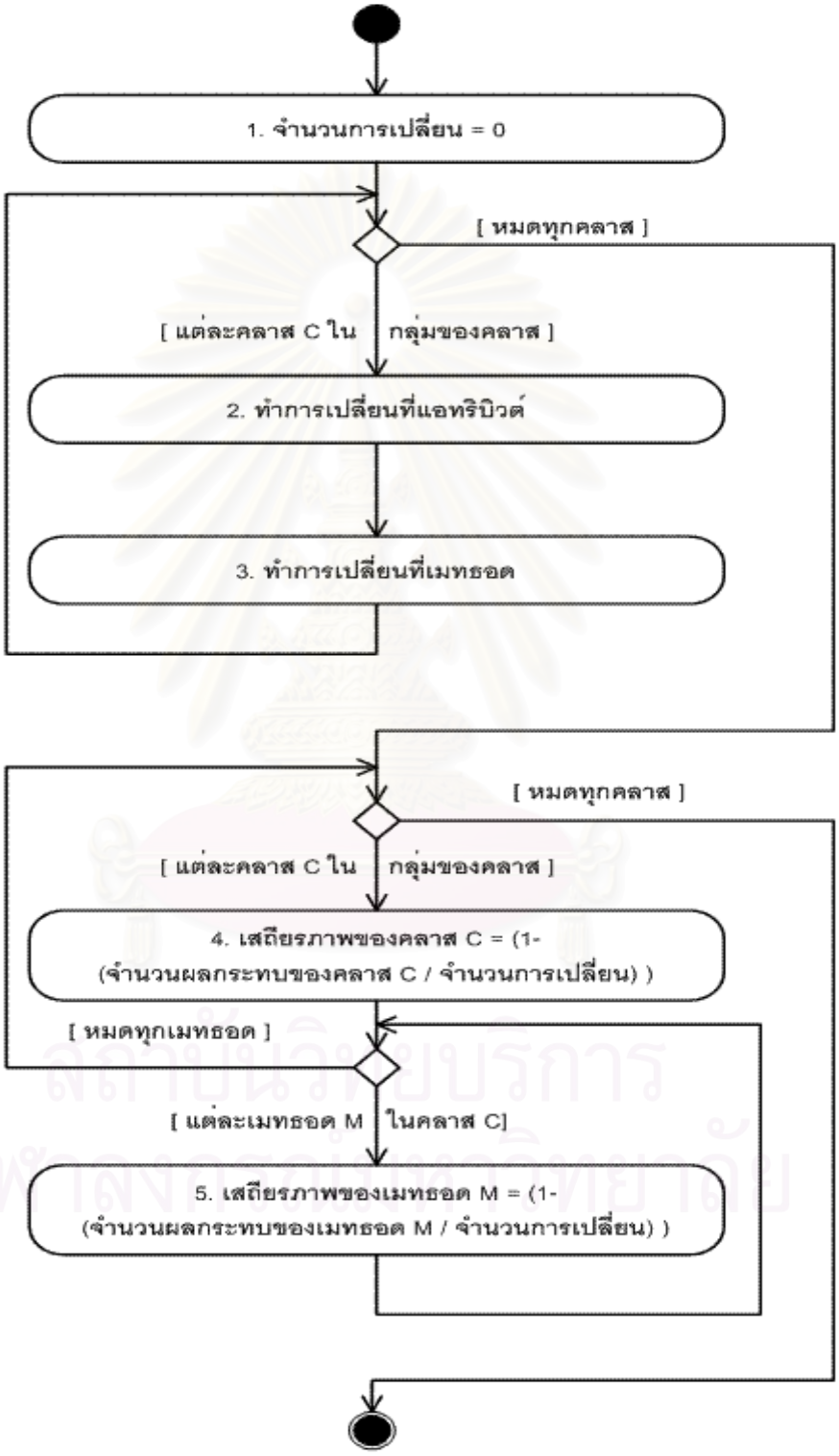
ส่วนการนับมาตรวัด NAI NMI ทำได้โดยการนับขนาดข้อมูลของแอทริบิวต์และเมทธอด ที่เก็บอยู่ในแอทริบิวต์ varCalled และ methodCalled ตามลำดับ โดยแยกตามขอบเขตการเข้าถึงเป็น public protected default และ private

5.3 การออกแบบและพัฒนาเครื่องมือคำนวณเสถียรภาพ

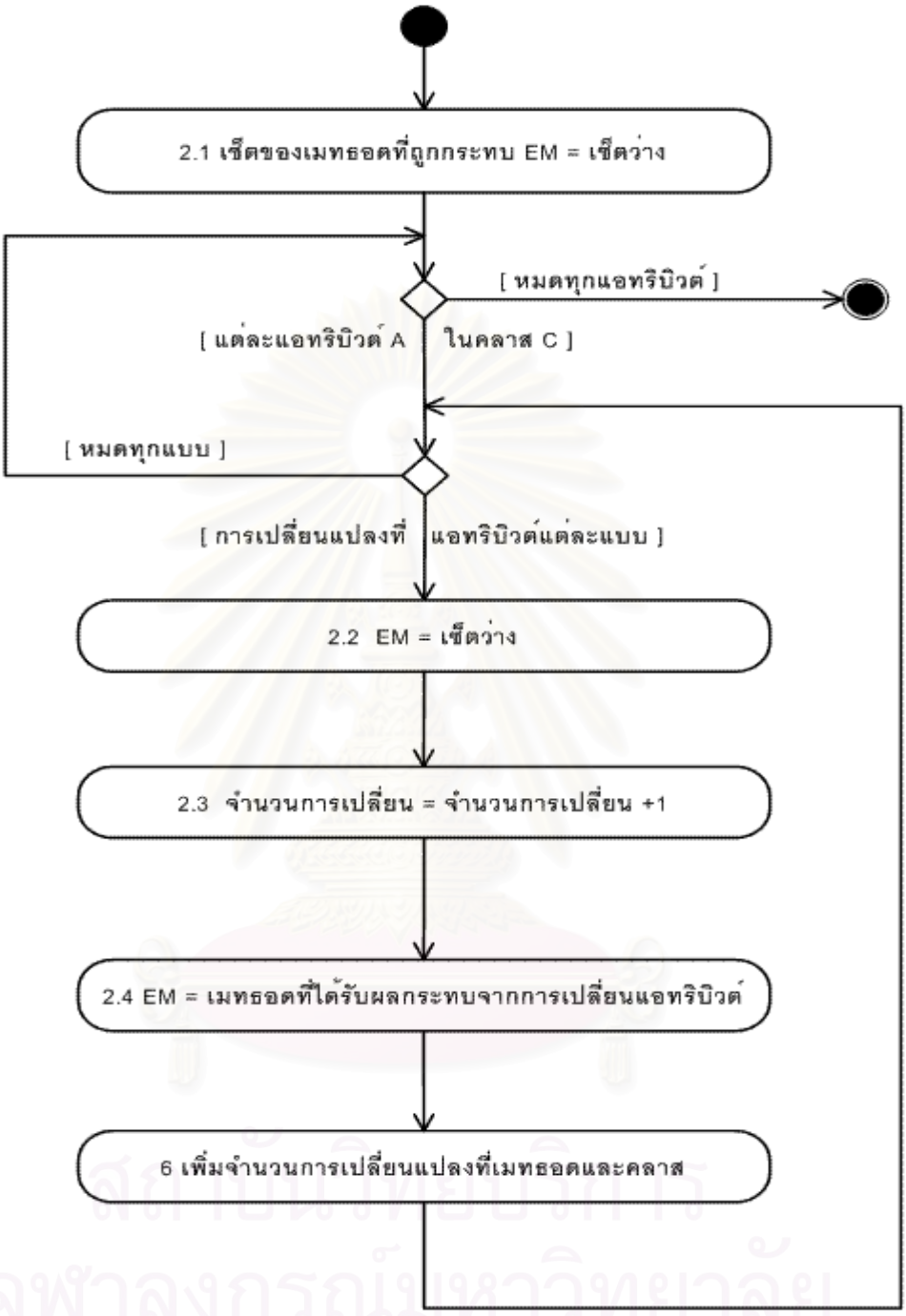
ในหัวข้อนี้ จะอธิบายถึงการออกแบบพัฒนาเครื่องมือคำนวณ โดยมีรายละเอียด ดังนี้

5.3.1 ขั้นตอนการคำนวณเสถียรภาพ

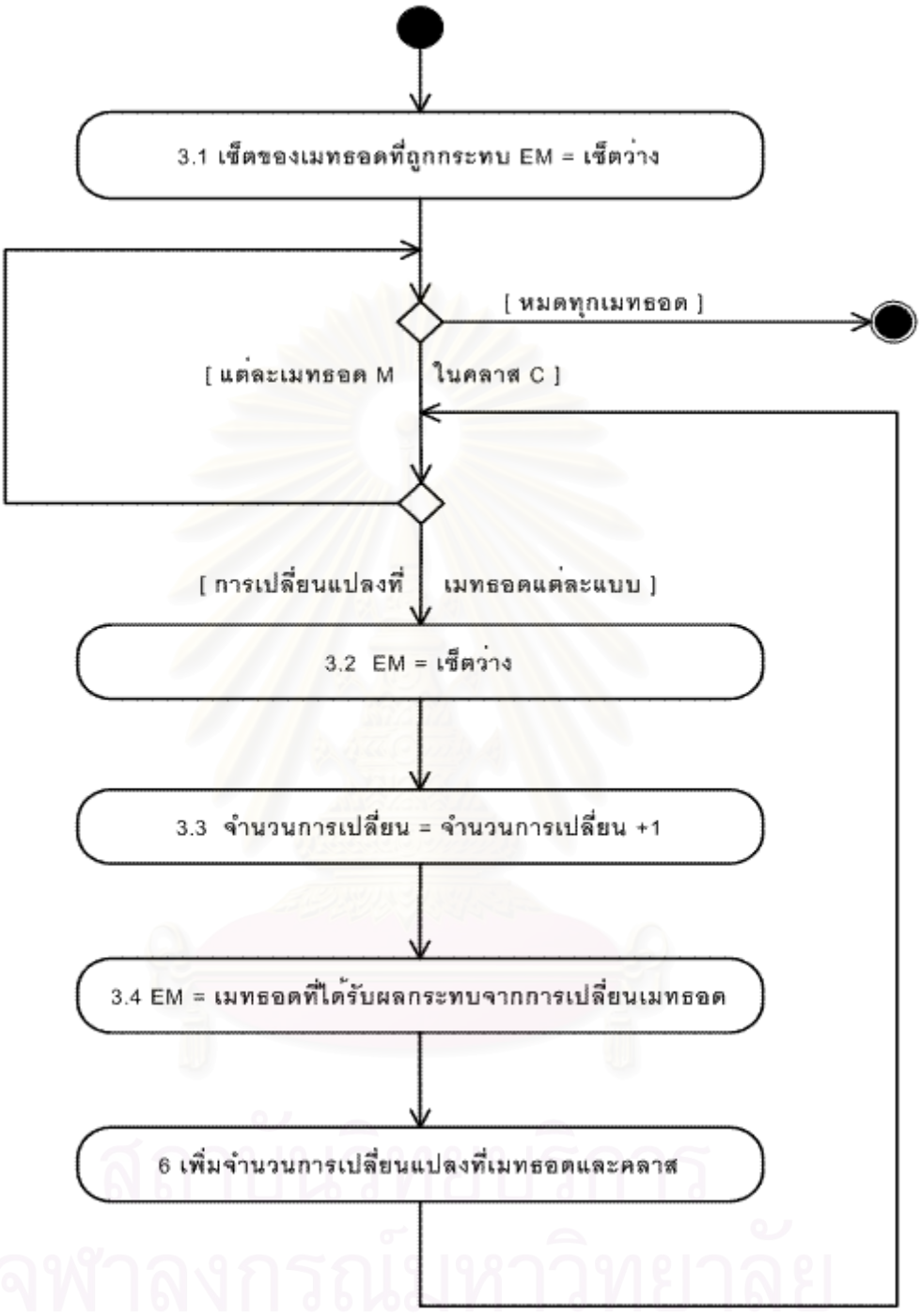
สำหรับขั้นตอนการคำนวณเสถียรภาพของเมทธอดและของคลาส ได้นำอัลกอริทึมในการคำนวณจาก [2] มาทำการปรับปรุง แสดงได้ดังรูปที่ 5.8 ถึง รูปที่ 5.11



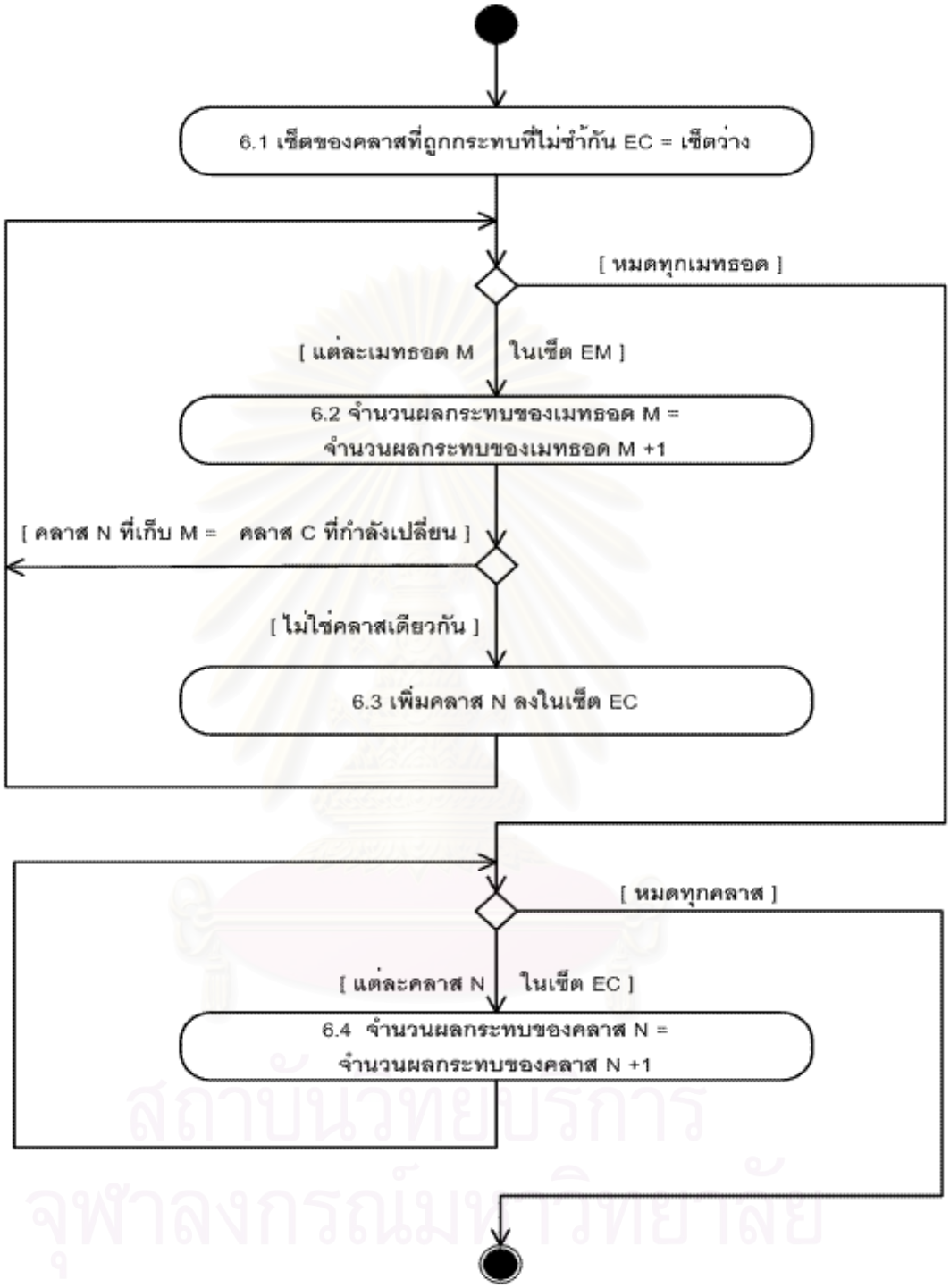
รูปที่ 5.8 แผนภาพกิจกรรมแสดงการคำนวณเสถียรภาพ



รูปที่ 5.9 แผนภาพกิจกรรมย่อยแสดงกิจกรรม “ทำการเปลี่ยนแปลงที่แอทริบิวต์” ในรูปที่ 5.8



รูปที่ 5.10 แผนภาพกิจกรรมย่อยแสดงกิจกรรม “ทำการเปลี่ยนแปลงที่เมทธอด” ในรูปที่ 5.8

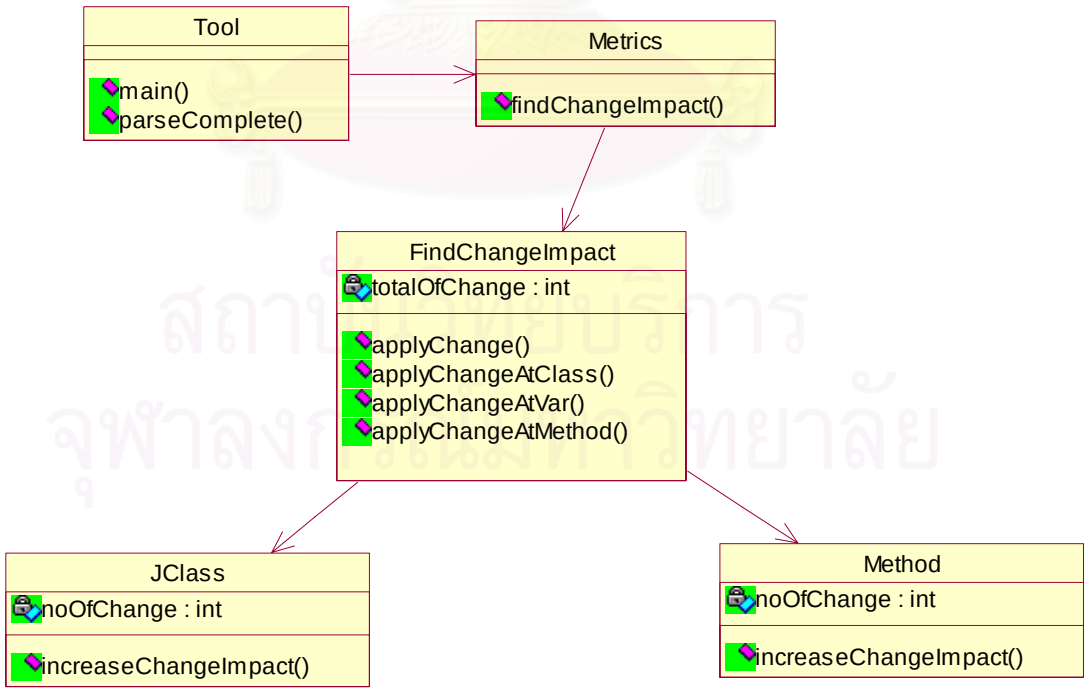


รูปที่ 5.11 แผนภาพกิจกรรมย่อยแสดงกิจกรรม “เพิ่มจำนวนการเปลี่ยนแปลงที่เมทธอดและคลาส” ใน รูปที่ 5.9 และ รูปที่ 5.10

โดยการเปลี่ยนแปลงจะทำกับทุกแอทริบิวต์และเมทธอดในแต่ละคลาส เมื่อเปลี่ยนแปลงจนครบทุกคลาสของรหัสโปรแกรมแล้ว จึงคำนวณเสถียรภาพของคลาสและเมทธอด ดังรูปที่ 5.8 ส่วนรูปที่ 5.9 จะเป็นการเปลี่ยนแปลงแอทริบิวต์ในแต่ละคลาส โดยนำแอทริบิวต์แต่ละตัวมาเปลี่ยนแปลงตามรูปแบบใน ตารางที่ 3.3 แล้วจึงเก็บเมทธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงในแต่ละครั้ง มาเพิ่มผลกระทบที่เกิดขึ้นกับแต่ละเมทธอด จากนั้นจึงหาผลกระทบที่เกิดขึ้นกับคลาส ดังใน รูปที่ 5.11 โดยจะไม่นับผลกระทบที่เกิดขึ้นกับคลาสที่กำลังเปลี่ยนแปลงอยู่ในขณะนั้น ส่วนรูปที่ 5.10 จะเป็นการเปลี่ยนแปลงเมทธอดในแต่ละคลาส ซึ่งจะนำแต่ละเมทธอดมาเปลี่ยนแปลงตามรูปแบบตารางที่ 3.4 แล้วเก็บเมทธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงในแต่ละครั้ง มาเพิ่มผลกระทบที่เกิดขึ้นกับแต่ละเมทธอด จากนั้นจึงหาผลกระทบที่เกิดขึ้นกับคลาส ดังใน รูปที่ 5.11 เช่นเดียวกับการเปลี่ยนแปลงที่แอทริบิวต์

5.3.2 แผนภาพคลาสของเครื่องมือคำนวณเสถียรภาพ

รูปที่ 5.12 เป็นแผนภาพคลาสของเครื่องมือสำหรับคำนวณเสถียรภาพ ซึ่งแสดงเฉพาะส่วนทำการเปลี่ยนแปลงและหาผลกระทบของการเปลี่ยนแปลงเท่านั้น โดยมีรายละเอียดของแต่ละคลาส แสดงได้ดังตารางที่ 5.14 ถึงตารางที่ 5.18



รูปที่ 5.12 แผนภาพคลาสของเครื่องมือสำหรับคำนวณเสถียรภาพ

ตารางที่ 5.14 รายละเอียดของคลาส Tool

ชื่อคลาส	Tool
คำอธิบาย	เป็นคลาสหลักของเครื่องมือ
เมทธอด	
main	เป็นเมทธอดหลัก และเป็นเมทธอดที่ถูกเรียกใช้งานเป็นเมทธอดแรกของเครื่องมือ
parseComplete	เป็นเมทธอดที่ถูกเรียกใช้งานหลังจากเก็บข้อมูลจากรหัสโปรแกรมแล้ว เพื่อเริ่มขั้นตอนการเปลี่ยนรหัสโปรแกรมและหาผลกระทบจากการเปลี่ยนแปลง

ตารางที่ 5.15 รายละเอียดของคลาส FindChangeImpact

ชื่อคลาส	FindChangeImpact
คำอธิบาย	เป็นคลาสที่เปลี่ยนแปลงและหาผลกระทบของการเปลี่ยนแปลง
แอทริบิวต์	
totalOfChange	เป็นแอทริบิวต์ที่เก็บจำนวนการเปลี่ยนแปลงทั้งหมด
เมทธอด	
applyChange	เป็นเมทธอดที่ควบคุมการเปลี่ยนแปลงให้ทำกับทุกคลาสในรหัสโปรแกรม
applyChangeAtClass	เป็นเมทธอดที่ควบคุมการเปลี่ยนแปลงในแต่ละคลาสของรหัสโปรแกรม
applyChangeAtVar	เป็นเมทธอดที่ทำหน้าที่เปลี่ยนแปลงแอทริบิวต์และหาผลกระทบที่เกิดจากการเปลี่ยนแปลงนั้น
applyChangeAtMethod	เป็นเมทธอดที่ทำหน้าที่เปลี่ยนแปลงเมทธอดและหาผลกระทบที่เกิดจากการเปลี่ยนแปลงนั้น

ตารางที่ 5.16 รายละเอียดของคลาส Metrics

ชื่อคลาส	Metrics
คำอธิบาย	เป็นคลาสที่ใช้จัดการกับข้อมูลต่าง ๆ ที่เก็บมาจากรหัสโปรแกรมและส่งคำสั่งในการเปลี่ยนแปลง
เมทธอด	
findChangeImpact	เป็นเมทธอดที่ส่งคำสั่งการเปลี่ยนแปลงและหาผลกระทบจากการเปลี่ยนแปลงไปยังคลาส FindChangeImpact

ตารางที่ 5.17 รายละเอียดของคลาส JClass

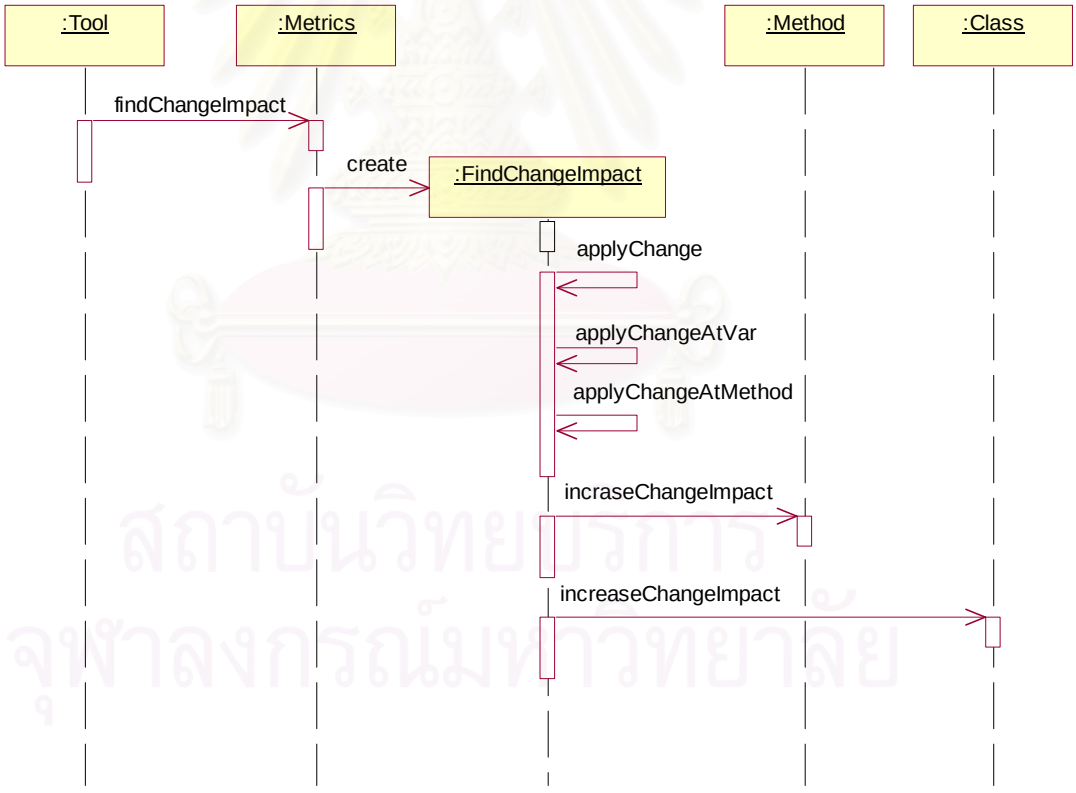
ชื่อคลาส	JClass
คำอธิบาย	เป็นคลาสที่เก็บโครงสร้างของคลาส
แอทริบิวต์	
noOfChange	เป็นแอทริบิวต์ที่เก็บจำนวนผลกระทบที่เกิดขึ้นกับคลาส
เมทธอด	
incraseChangeImpact	เป็นเมทธอดทำหน้าที่เพิ่มจำนวนผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลง

ตารางที่ 5.18 รายละเอียดของคลาส Method

ชื่อคลาส	Method
คำอธิบาย	เป็นคลาสที่เก็บโครงสร้างของเมทธอด
แอทริบิวต์	
noOfChange	เป็นแอทริบิวต์ที่เก็บจำนวนผลกระทบที่เกิดขึ้นกับเมทธอด
เมทธอด	
incraseChangeImpact	เป็นเมทธอดที่หน้าที่เพิ่มจำนวนผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลง

5.3.3 แผนภาพซีเควนซ์ของเครื่องมือคำนวณเสถียรภาพ

จากรูปที่ 5.13 เป็นแผนภาพซีเควนซ์แสดงการหาผลกระทบของการเปลี่ยนแปลง โดยเริ่มจากคลาส Tool ซึ่งเป็นคลาสหลักเรียกเมทธอด findChangeImpact ที่คลาส Metrics โดยคลาส Metrics จะสร้างวัตถุคลาส FindChangeImpact โดยจะส่งคลาสของรหัสโปรแกรมที่มีอยู่ทั้งหมดไปให้ เพื่อทำการเปลี่ยนแปลงและหาผลกระทบจากการเปลี่ยนแปลง หลังจากสร้างวัตถุของคลาส FindChangeImpact แล้วเมทธอด applyChange จะถูกเรียกใช้งานเพื่อเตรียมเปลี่ยนแปลงคลาสต่าง ๆ ซึ่งได้รับมาจากคลาส Metrics หลังจากนั้นเรียกเมทธอด applyChangeAtVar เพื่อทำการเปลี่ยนแปลงที่แอทริบิวต์และหาผลกระทบจากการเปลี่ยนแปลงที่เกิดขึ้น และทำการเรียกเมทธอด applyChangeAtMethod เพื่อเปลี่ยนแปลงที่เมทธอดและหาผลกระทบจากการเปลี่ยนแปลงที่เกิดขึ้น จากนั้นจึงเรียกเมทธอด increaseChangeImpact ในคลาส Method และคลาส JClass เพื่อทำการเพิ่มจำนวนผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลงในแต่ละครั้ง



รูปที่ 5.13 แผนภาพซีเควนซ์ของเครื่องมือคำนวณเสถียรภาพ

บทที่ 6

สรุปผลการวิจัย

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอโมเดลในการทำนายเสถียรภาพของเมทธอด เมื่อมีการเปลี่ยนแปลงที่แอทริบิวต์และเมทธอดของรหัสโปรแกรม โดยนำมาตรวัดเชิงวัตถุที่เกี่ยวกับเมทธอดมาใช้ในการสร้างโมเดล มาตรวัดที่นำมาใช้ในงานวิจัยมีทั้งหมด 15 มาตรวัด แบ่งเป็น 3 กลุ่ม คือ มาตรวัดเกี่ยวกับขนาดของเมทธอด ประกอบด้วย Param LocalVar NOS และ LOC มาตรวัดเกี่ยวกับความซับซ้อนของ เมทธอด ประกอบด้วย V(G) และ MCX และมาตรวัดการเข้าคู่ระหว่างเมทธอด ประกอบด้วย CBO NAIpub NAIpro NAIdef NAIpri NMIpub NMIpro NMIdef และ NMIpri สำหรับการเก็บค่ามาตรวัดนั้น ได้พัฒนาเครื่องมือโดยนำเครื่องมือเก็บค่ามาตรวัด MTOOP2 มาปรับปรุงเพิ่มเติมเพื่อให้สามารถเก็บค่ามาตรวัดได้อย่างอัตโนมัติ

สำหรับการเปลี่ยนแปลงที่ใช้ในการหาเสถียรภาพของเมทธอดนั้น พิจารณาเฉพาะการเปลี่ยนแปลงที่เกิดขึ้นที่แอทริบิวต์และเมทธอดของแต่ละคลาส โดยมีเงื่อนไขว่า การเปลี่ยนแปลงนั้น จะต้องเป็นการเปลี่ยนแปลงที่เกิดขึ้นแบบสถิตย์และมีผลกระทบกับวากยสัมพันธ์เท่านั้น ซึ่งในการหาผลกระทบของการเปลี่ยนแปลงได้ทำการพัฒนาเครื่องมือ โดยนำเครื่องมือจัดเก็บมาตรวัดในส่วนของพาร์เซอร์และซินแท็กซ์ทรี มาปรับปรุงเพิ่มเติม เพื่อให้สามารถหาผลกระทบจากการเปลี่ยนแปลงได้อย่างอัตโนมัติ

ในการสร้างโมเดลทำนายเสถียรภาพของเมทธอดนั้น ได้ใช้วิธีการสร้างสมการถดถอย โดยรหัสโปรแกรมที่นำมาใช้ในการทดลองนั้น มีทั้งหมด 14 โปรแกรม แบ่งเป็น 2 ชุด ชุดแรกเป็นชุดที่ใช้สำหรับการสร้างโมเดล มี 10 โปรแกรม 216 คลาส 1451 เมทธอด และชุดที่สอง เป็นชุดที่ใช้ในการทดสอบความถูกต้องของโมเดล มี 4 โปรแกรม 59 คลาส 317 เมทธอด

จากการทดลองพบว่า มาตรวัดเชิงวัตถุที่มีความสัมพันธ์กับเสถียรภาพของเมทธอด มีทั้งสิ้น 7 มาตรวัด คือ NMIpub MCX NAIpro NAIpub NAIpri NAIdef และ Param โดยสัมประสิทธิ์การตัดสินใจเชิงซ้อนเท่ากับ 61.7% และเมื่อนำรหัสโปรแกรมในชุดทดสอบมาทำการประเมินผลความถูกต้อง ด้วยวิธีการทำนายที่ระดับ 0.01 พบว่ามีความถูกต้อง 83.5%

หลังจากได้โมเดลทำนายเสถียรภาพของเมทธอดแล้ว จึงได้ทำการพัฒนาเครื่องมือในการทำนายเสถียรภาพของเมทธอดขึ้น โดยนำเครื่องมือจัดเก็บมาตรวัดมาทำการพัฒนาเพิ่มเติมและ

ในขั้นตอนสุดท้ายของงานวิจัยนี้ ได้ทำการศึกษา ความสัมพันธ์ระหว่างเสถียรภาพของคลาสและเสถียรภาพของเมทธอดที่เลือกมาเป็นตัวแทนของคลาส โดยเลือกทั้งค่าเสถียรภาพมากสุดของเมทธอด ค่าเสถียรภาพน้อยสุด และค่าเสถียรภาพเฉลี่ย มาทำการทดสอบโดยใช้วิธีการวิเคราะห์สหสัมพันธ์ จากผลการทดสอบ พบว่า ค่าเสถียรภาพเฉลี่ยของเมทธอด และค่าเสถียรภาพน้อยสุดของเมทธอดมีความสัมพันธ์กับเสถียรภาพของคลาสใกล้เคียงกัน คือ 64.5% และ 62.7% ตามลำดับ ส่วนเสถียรภาพมากสุดของเมทธอดมีความสัมพันธ์กับเสถียรภาพของคลาสน้อยที่สุด คือ 37.5%

6.2 ข้อสังเกต

ข้อสังเกตในงานวิจัยนี้ มีดังนี้

1. ขนาดของรหัสโปรแกรม เนื่องจากรหัสโปรแกรมที่นำมาใช้ในการทดลอง ได้กำหนดว่าจะต้องมีคลาสตั้งแต่ 10 คลาสถึง 50 คลาส ดังนั้นถ้าหากมีการนำโมเดลการทำนายเสถียรภาพของเมทธอด ไปทำนายกับรหัสโปรแกรมที่มีคลาสมากกว่า 50 คลาส อาจทำให้ผลการทำนายคลาดเคลื่อนได้
2. รูปแบบการเขียนรหัสโปรแกรม เนื่องจากรหัสโปรแกรมที่ใช้ในการวิจัยนี้ ได้นำมาจากเว็บไซต์ต่าง ๆ ซึ่งแต่ละโปรแกรมอาจจะมีรูปแบบการเขียนโปรแกรมแตกต่างกันไปจึงอาจมีผลทำให้ค่าทำนายเสถียรภาพของแต่ละโปรแกรมมีความแตกต่างกัน
3. โดเมนของรหัสโปรแกรม เนื่องจากในงานวิจัยนี้ได้กำหนดรหัสโปรแกรมที่นำมาใช้ที่ขนาดของโปรแกรม แต่ไม่ได้คำนึงถึงโดเมนของรหัสโปรแกรม ซึ่งอาจจะทำให้ผลของการทำนายของรหัสโปรแกรมที่มีโดเมนที่ต่างกันได้ผลที่ไม่เหมือนกัน

6.3 แนวทางการวิจัยในอนาคต

แนวทางในการทำวิจัยเพิ่มเติมในอนาคต ได้แก่

1. เพิ่มรูปแบบของการเปลี่ยนแปลง เนื่องจากในงานวิจัยนี้ พิจารณาเฉพาะการเปลี่ยนแปลงที่เป็นแบบสถิตย์และมีผลกับวากยสัมพันธ์เท่านั้น
2. วิจัยกับรหัสโปรแกรมที่มีจำนวนคลาสมากกว่า 50 คลาส
3. เพิ่มมาตรวัดเชิงวัตถุอื่น ๆ เพื่อใช้ในการทำนายเสถียรภาพให้มีความถูกต้องยิ่งขึ้น

รายการอ้างอิง

[1] ISO/IEC 9126-1:2000, Information technology - Software product quality – Part1: Quality model, Internationl Organization for Standardization.

[2] Elish, M., O. and Rine, D. Investigation of Metrics for Object-Oriented Design Logical Stablility. Proceedings of the Seventh European Conference of Software Maintenance and Reengineering(CSMR'03), 2003.

[3] Rajlich, V. A Model for Change propagation based on Graph Rewriting. Proceedings of IEEE International Conference on Software Maintenance, 1997.

[4] Fenton, N. E., and Pfleeger, S., L. Software Metrics A Rigorous and Practical Approach. 2nd ed. PWS Publishing Company, 1997.

[5] สมหวัง แซ่ตั้ง. การออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุ. ปริญญาวิทยาศาสตรมหาบัณฑิต, ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2543.

[6] วัฒนชัย รอดกำเนิด. การออกแบบและพัฒนาเครื่องมือวัดปัจจัยของความซับซ้อนของโปรแกรมเชิงวัตถุภาษาจาวา. ปริญญาวิศวกรรมศาสตรบัณฑิต, ภาควิชาวิศวกรรมศาสตร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2544.

[7] Li, L. Change Impact Analysis of Object-Oriented Software. Doctoral dissertation, Information Technology, George Mason University, 1998.

[8] Kung, D., et al. Change Impact Identification Object Oriented Software Maintenance. Proceedings of the International Conference on Software Maintenance, 1994.

[9] Li, L., and Offutt, A. J. Algorithmic Analysis of the Impact of Changes to Object-Oriented Software. Proceedings of the 1996 International Conference on Software Maintenance, Nov. 1996

[10] Gosling, J., Joy, B., and Steele G. The Java™ Language Specification. 2nd ed. California : Addison-Wesley, 2000.

[11] กัลยา วานิชย์บัญชา. การวิเคราะห์ตัวแปรหลายตัวด้วย SPSS for Windows. กรุงเทพมหานคร : โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.

[12] กัลยา วานิชย์บัญชา. การวิเคราะห์สถิติ : สถิติสำหรับการบริหารและวิจัย. พิมพ์ครั้งที่6 . กรุงเทพมหานคร : ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย, 2539.

[13] สมจิต วัฒนาชยากูล. สถิติวิเคราะห์เบื้องต้น. กรุงเทพมหานคร : ภาควิชาคณิตศาสตร์และสถิติ คณะวิทยาศาตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์, 2545.

[14] Conte, S.D., Dunsmore, H.E., and Shen, V.Y. Software engineering metrics and models. Benjamin/Cummings Publishing Company, 1986.

[15] Gini, A. Java Gramma [Online]. Available from : http://www.cobase.cs.ucla.edu/pub/javacc [2002, May 05]

[16] Java source code. Available from : http://www.planet-source-code.com

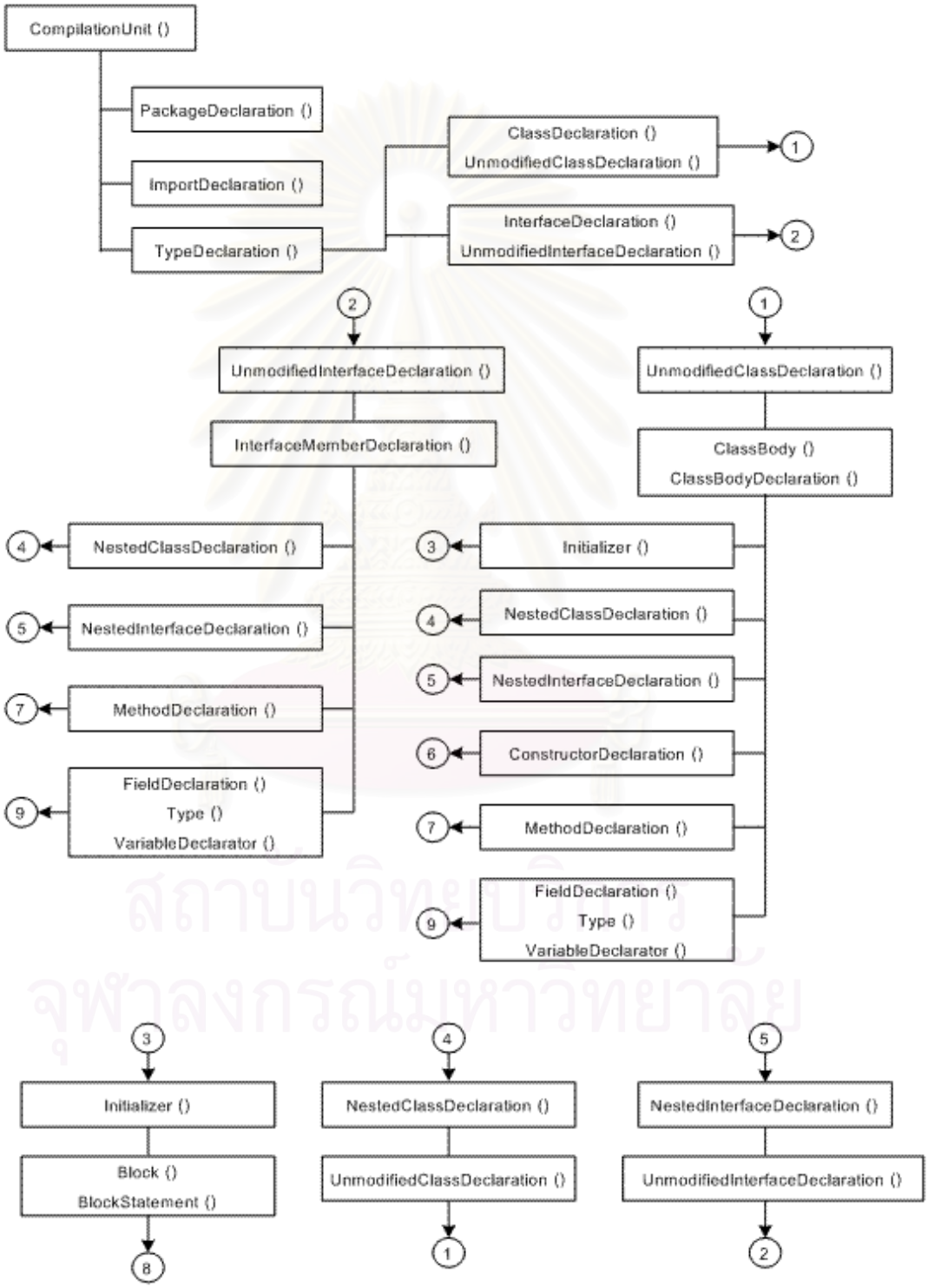
[17] Java source code. Available from : http://www.programmersheaven.com

[18] Java source code. Available from : http://sourceforge.net

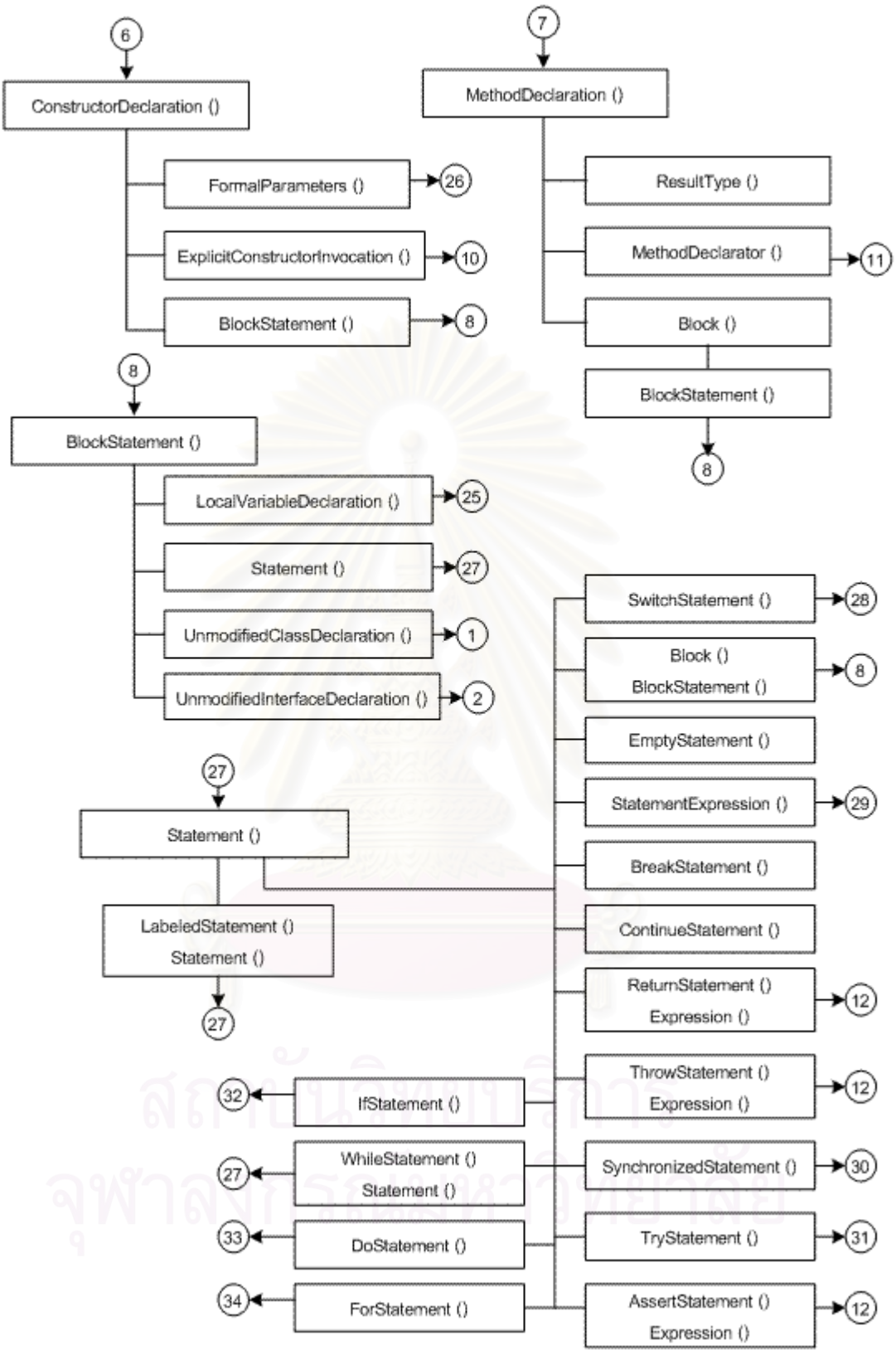
ภาคผนวก

ภาคผนวก ก

แผนภาพต้นไม้

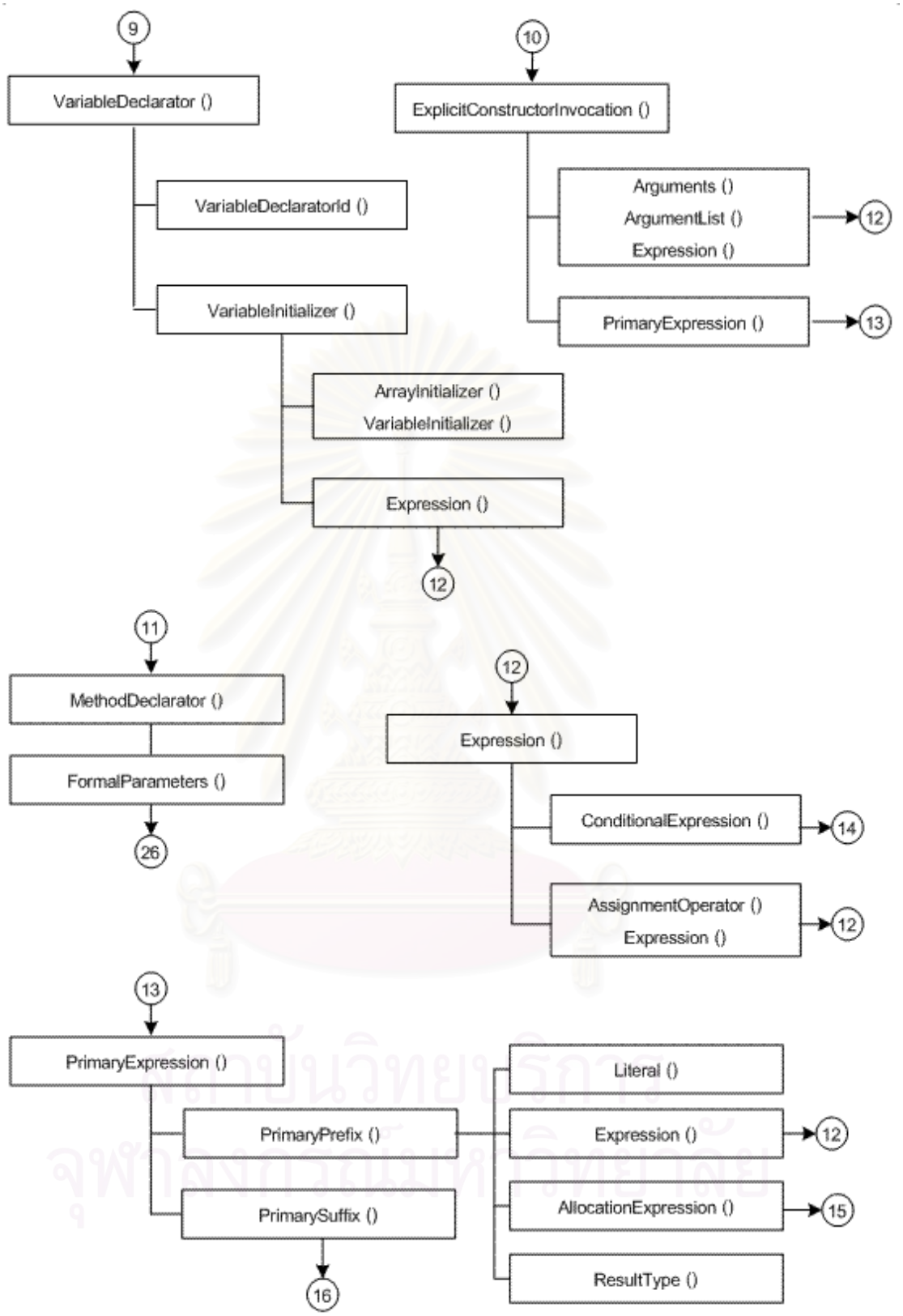


รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรีของคลาสจาวาพาร์เซอร์



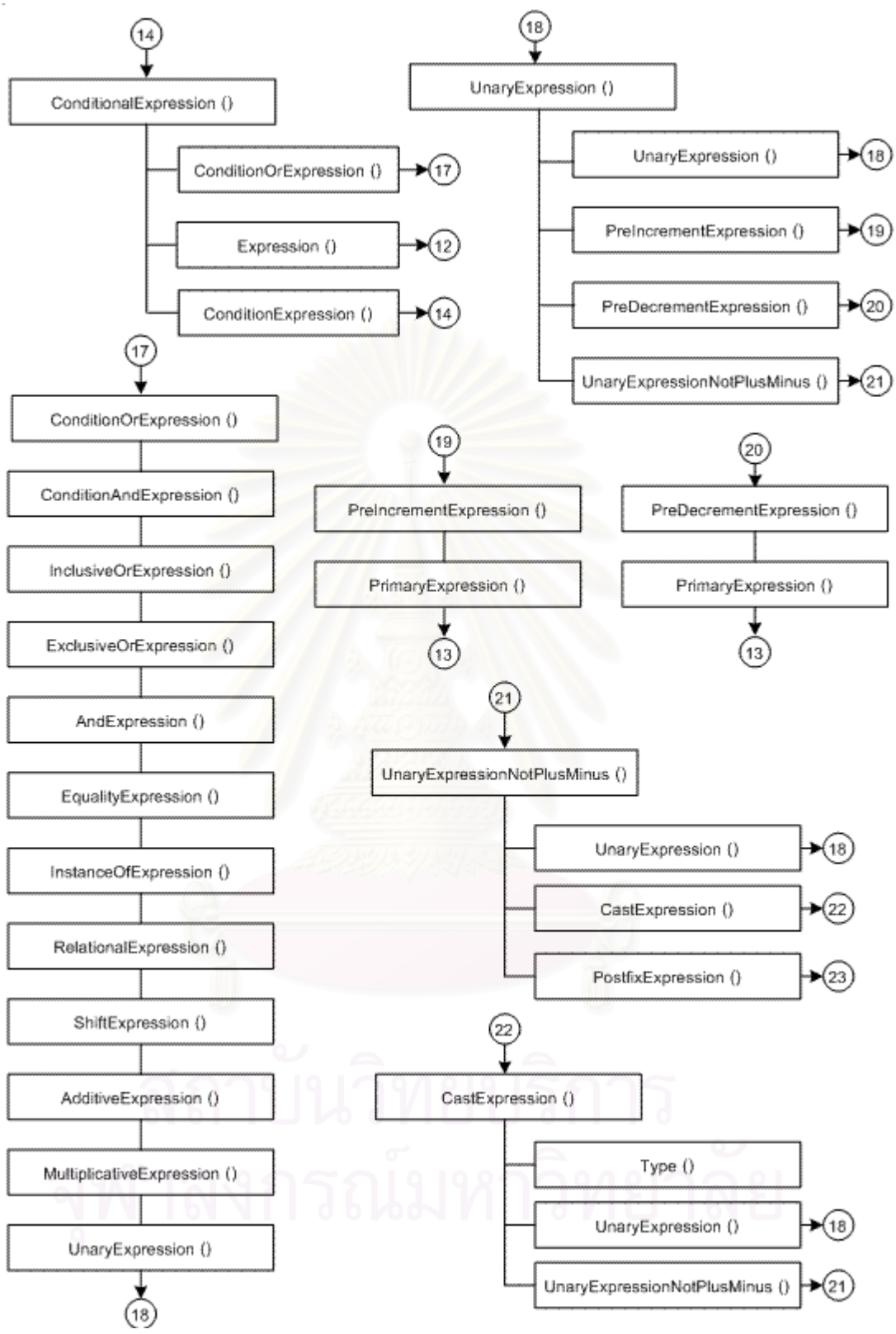
รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)



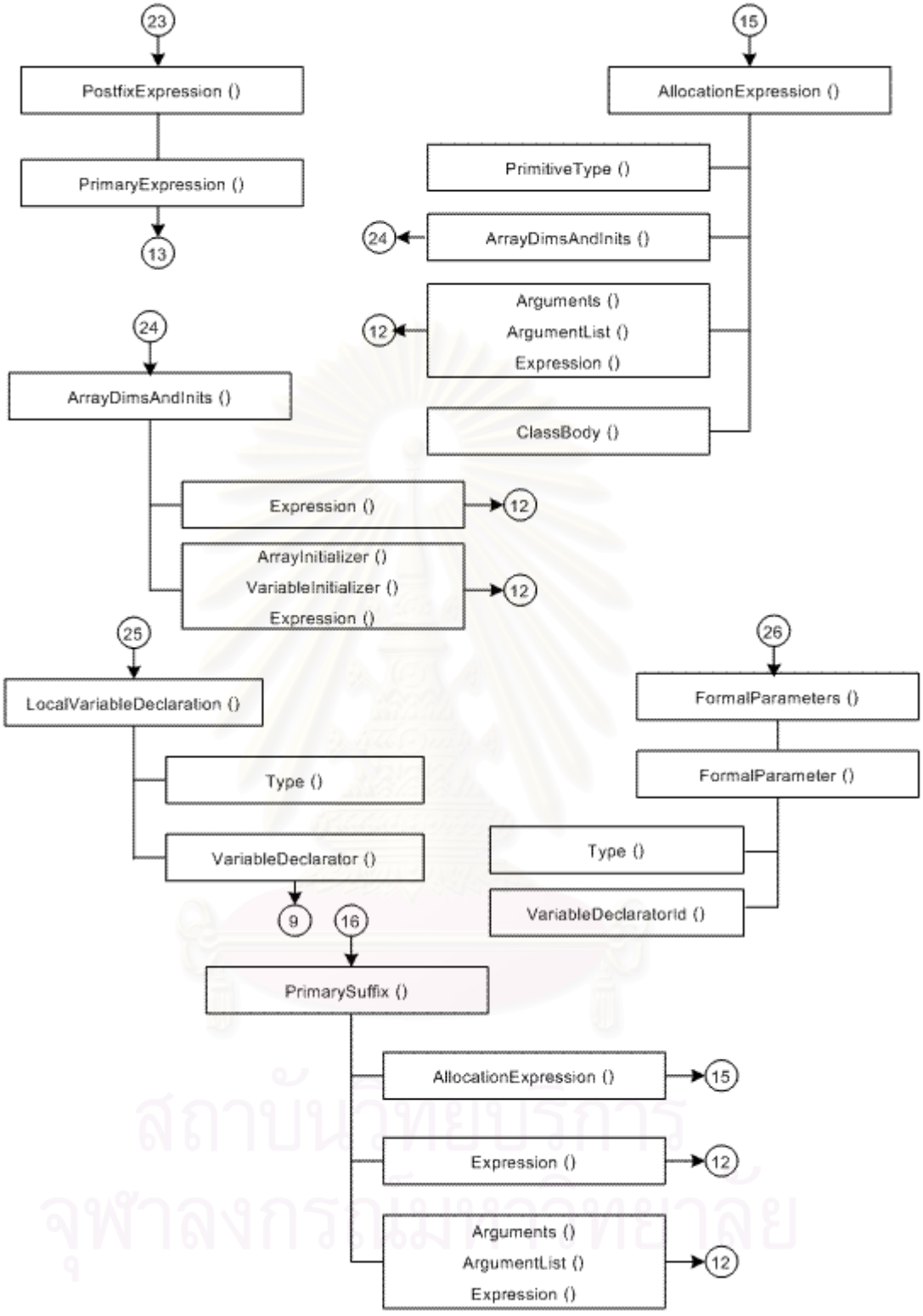
รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)



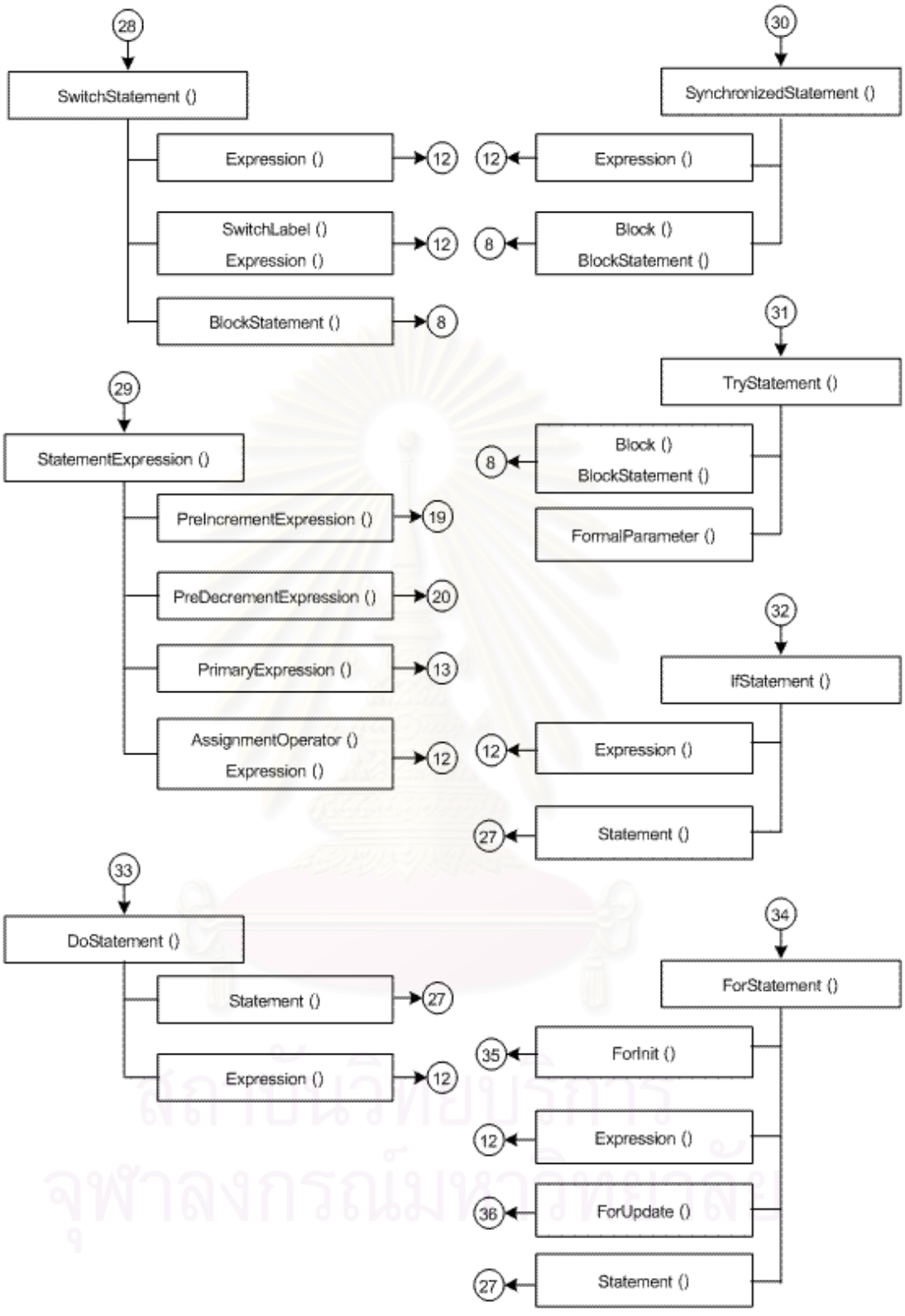
รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)



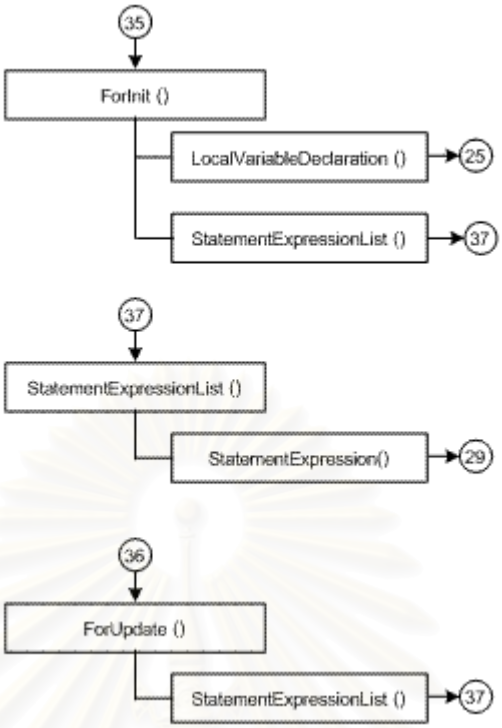
รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)



รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)



รูปที่ ก-1 แผนภาพต้นไม้แสดงโหนดต่าง ๆ ที่ได้จากการสร้างซินแท็กซ์ทรี

ของคลาสจาวาพาร์เซอร์(ต่อ)

ภาคผนวก ข

การแก้ไขรหัสโปรแกรมต้นแบบจาวาพาร์เซอร์

ในส่วนนี้จะกล่าวถึงการแก้ไขเพิ่มเติมรหัสโปรแกรมต้นแบบจาวาพาร์เซอร์ในคลาส JavaParser เพื่อทำการเก็บโทเคนของเมสเสจ แสดงได้ดังตาราง

ตารางที่ ข-1 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด PrimaryPrefix

static final public void PrimaryPrefix() throws ParseException { switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {	

case FALSE:
case NULL:
case TRUE:
case INTEGER_LITERAL:
case FLOATING_POINT_LITERAL:
case CHARACTER_LITERAL:
case STRING_LITERAL:
String theType = getLiteralType();
msgType = MessageConstant.P_LITERAL;
metrics.addToken(theType,msgType);
break;
	โทเคนประเภท 111
	เก็บโทเคนที่เป็น ลิเทอรัล

case THIS:
jj_consume_token(THIS);
metrics.addSelfCall();
msgType = MessageConstant.P_THIS;
metrics.addToken("this",msgType);
break;
	โทเคนประเภท 121
	เก็บโทเคน this

ตารางที่ ข-1 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด PrimaryPrefix (ต่อ)

case SUPER:
jj_consume_token(SUPER);
jj_consume_token(DOT);
Token t = jj_consume_token(IDENTIFIER);
msgType = MessageConstant.P_SUPER_IDENTIFIER;
metrics.addToken("super."+t.toString(),msgType);
break;
	โทเคนประเภท 131
	เก็บโทเคน super

case LPAREN:
jj_consume_token(LPAREN);
msgType = MessageConstant.P_LPAREN;
metrics.addToken("(",msgType);
Expression();
	โทเคนประเภท 141
	เก็บโทเคน (

jj_consume_token(RPAREN);
msgType = MessageConstant.P_RPAREN;
metrics.addToken(")",msgType);
break;
	โทเคนประเภท 142
	เก็บโทเคน)

case NEW: msgType = MessageConstant.P_NEW; AllocationExpression(); break;	โทเคนประเภท 151 เก็บโทเคนที่ตามหลัง new ในเมทธอด AllocateExpression
default: jj_la1[78] = jj_gen;	

if (jj_2_20(2147483647)) {
String resultType = getResultType();
jj_consume_token(DOT);
jj_consume_token(CLASS);
msgType = MessageConstant.P_RESULTTYPE;
metrics.addToken(resultType+".Class",msgType);
	โทเคนประเภท 161
	เก็บโทเคน ชื่อ.Class

ตารางที่ ข-1 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด PrimaryPrefix (ต่อ)

} else {
switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
case IDENTIFIER:
String theName = getName();
msgType = MessageConstant.P_IDENTIFIER;
metrics.addToken(theName,msgType);
break;
	โทเคนประเภท 171
	เก็บโทเคน ชื่อ

default:
jj_la1[79] = jj_gen;
jj_consume_token(-1);
throw new ParseException();
}
}
}
}	

ตารางที่ ข-2 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด PrimarySuffix

static final public void PrimarySuffix() throws ParseException {	

if (jj_2_21(2)) {
jj_consume_token(DOT);
jj_consume_token(THIS);
msgType = MessageConstant.S_THIS;
metrics.addToken("this",msgType);
	โทเคนประเภท 211
	เก็บโทเคน this

} else if (jj_2_22(2)) {
jj_consume_token(DOT);
jj_consume_token(SUPER);
msgType = MessageConstant.S_SUPER;
metrics.addToken("super",msgType);
	โทเคนประเภท 261
	เก็บโทเคน super

ตารางที่ ข-2 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด PrimarySuffix (ต่อ)

} else if (jj_2_23(2)) {
jj_consume_token(DOT);
msgType = MessageConstant.S_ALLOCATE;
AllocationExpression();
	โทเคนประเภท 221
	เก็บโทเคนหลัง new
	ในเมทธอด AllocateExpression

} else {
switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
case LBRACKET:
jj_consume_token(LBRACKET);
msgType = MessageConstant.S_LBRACKET;
metrics.addToken("[",msgType);
	โทเคนประเภท 231
	เก็บโทเคน [

Expression();
jj_consume_token(RBRACKET);
msgType = MessageConstant.S_RBRACKET;
metrics.addToken("]",msgType);
break;
	โทเคนประเภท 232
	เก็บโทเคน]

case DOT:
jj_consume_token(DOT);
Token t = jj_consume_token(IDENTIFIER);
metrics.addMessageSend();
metrics.addMessageSend(t.toString());
msgType = MessageConstant.S_IDENTIFIER;
metrics.addToken(t.toString(),msgType);
break;
	โทเคนประเภท 241
	เก็บโทเคน ชื่อ หลังจากโทเคน
	ก่อน

. . . } } }	

ตารางที่ ข-3 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด Argument

static final public void Arguments() throws ParseException {	

jj_consume_token(LPAREN);
msgType = MessageConstant.S_START_ARG;
metrics.addToken("(",msgType);
	โทเคนประเภท 251
	เก็บโทเคน (

switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {
case BOOLEAN:
.
.
.
case MINUS:
ArgumentList();
}	

jj_consume_token(RPAREN);
msgType = MessageConstant.S_END_ARG;
metrics.addToken(")",msgType);
}
	โทเคนประเภท 252
	เก็บโทเคน)

ตารางที่ ข-4 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด ArgumentList

static final public void ArgumentList() throws ParseException { . . .	

jj_consume_token(COMMA);
msgType = MessageConstant.S_COMMA;
metrics.addToken(",",msgType);
Expression();
}
}
	โทเคนประเภท 253
	เก็บโทเคน ,

ตารางที่ ข-5 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด ArgumentList

static final public void AllocationExpression() throws ParseException { . . .	

case NEW:
jj_consume_token(NEW);
typeName = getName();
metrics.addToken(typeName,msgType);
	เก็บโทเคนหลัง new

. . . } } }	

สำหรับประเภทและค่าของแต่ละโทเคนที่เก็บ มีรายละเอียด ดังนี้

111 จะเป็นโทเคนที่เก็บลิเทอรัล (Literal) คือ โทเคนที่มีประเภทข้อมูลเป็นแอบสแตรท (abstract data type) เป็นสายอักขระ (String) หรือค่านัล (null)

เช่น 10 จะเก็บเป็น int

“str” จะเก็บเป็น String

null จะเก็บเป็น null

121 จะเป็นโทเคนที่เก็บคำสงวน this

เช่น thi.x จะเก็บ this

131 จะเป็นโทเคนคำสงวน super ที่ตามด้วยชื่อตัวแปรหรือชื่อเมทธอด

เช่น super.x จะเก็บ super.x

ตารางที่ ข-6 การแก้ไขต้นแบบจาวาพาร์เซอร์ในเมทธอด CastExpression

static final public void CastExpression() throws ParseException { if (jj_2_18(2147483647)) {	

jj_consume_token(LPAREN);
String theType = getType();
jj_consume_token(RPAREN);
metrics.addUnaryExpression();
msgType = MessageConstant.P_CAST;
metrics.addToken(theType,msgType);
	โทเคนประเภท 143
	เก็บโทเคนการเปลี่ยนชนิดของข้อมูล

UnaryExpression();
} else {
switch ((jj_ntk==-1)?jj_ntk():jj_ntk) {	

case LPAREN:
jj_consume_token(LPAREN);
String theType =getType();
jj_consume_token(RPAREN);
msgType = MessageConstant.P_CAST;
metrics.addToken(theType,msgType);
	โทเคนประเภท 143
	เก็บโทเคนการเปลี่ยนชนิดของข้อมูล

UnaryExpressionNotPlusMinus();
break;
default:
jj_la1[77] = jj_gen;
jj_consume_token(-1);
throw new ParseException();
}
}
}	

141 จะเป็นโทเคนเครื่องหมาย (ซึ่งจะเก็บเมื่อเจอเครื่องหมายวงเล็บที่ใช้สำหรับจัดกลุ่มการทำงาน

เช่น (x+y) จะเก็บ (

142 จะเป็นโทเคนเครื่องหมาย) จะใช้ในกรณีเช่นเดียวกับประเภท 141

เช่น (x+y) จะเก็บ)

143 จะเป็นโทเคนที่เก็บชนิดข้อมูลที่ต้องการแปลงชนิดของข้อมูล

เช่น (String)x; จะเก็บ String

151 จะเป็นโทเคนที่เก็บชนิดของข้อมูลที่ตามหลังคำสงวน new

เช่น new String(); จะเก็บ String

161 จะเป็นโทเคนที่เก็บชนิดของข้อมูลที่ต้องการออบเจคของชนิดข้อมูลนั้น

เช่น String.class จะเก็บ String.class

171 จะเป็นโทเคนที่เก็บชื่อตัวแปรหรือชื่อเมทธอดของเมสเสจ

เช่น x.y จะเก็บ x.y

211 จะเป็นโทเคนที่เก็บคำสงวน this ที่ตามหลังโทเคนอื่น(.this)

เช่น x.this จะเก็บ this

221 จะเป็นโทเคนที่เก็บชนิดของข้อมูลที่ตามหลังคำสงวน new ที่คำสงวน new นั้นมาหลังโทเคนอื่น (.new)

เช่น a.new B() จะเก็บ B

231 จะเป็นโทเคนที่เก็บเครื่องหมาย [

เช่น a[i] จะเก็บ [

232 จะเป็นโทเคนที่เก็บเครื่องหมาย]

เช่น a[i] จะเก็บ]

241 จะเป็นโทเคนที่เก็บชื่อตัวแปรหรือชื่อเมทธอดที่ตามหลังโทเคนอื่น

เช่น x.y().z จะเก็บ z

251 จะเป็นโทเคนที่เก็บเครื่องหมาย (ที่ใช้สำหรับเรียกเมทธอด

เช่น x.y() จะเก็บ (

252 จะเป็นโทเคนที่เก็บเครื่องหมาย) ที่ใช้สำหรับเรียกเมทธอด

เช่น x.y() จะเก็บ)

253 จะเป็นโทเคนที่เก็บเครื่องหมาย , ที่ใช้สำหรับเป็นตัวแบ่งอาร์กิวเมนต์

เช่น x.y(a1,a2); จะเก็บ ,

261 จะเป็นโคเทนที่เก็บคำสงวน super ที่ตามหลังโทเคนอื่น

เช่น x.super จะเก็บ super

ภาคผนวก ค

ประเภทของเมสเสจ

เมสเสจที่เรียกใช้สามารถแบ่งได้เป็น 3 ประเภทคือ เมสเสจที่เป็นแอทริบิวต์ เมสเสจที่เป็นอาร์เรย์ และเมสเสจที่เป็นเมทธอด ซึ่งแสดงได้เป็นแผนภาพต้นไม้ในรูป ค-1 และมีรายละเอียดการเรียกใช้เมสเสจ ดังนี้

ค.1 เมสเสจที่เป็นแอทริบิวต์

เมสเสจที่เป็นแอทริบิวต์ สามารถเรียกต่อไปได้เรื่อย ๆ โดยจะมีเครื่องหมาย . เป็นตัวแบ่งโทเคนในการเรียกแอทริบิวต์แต่ละตัว สำหรับการเรียกใช้เมสเสจที่เป็นแอทริบิวต์สามารถแบ่งเป็น 2 ส่วนคือ

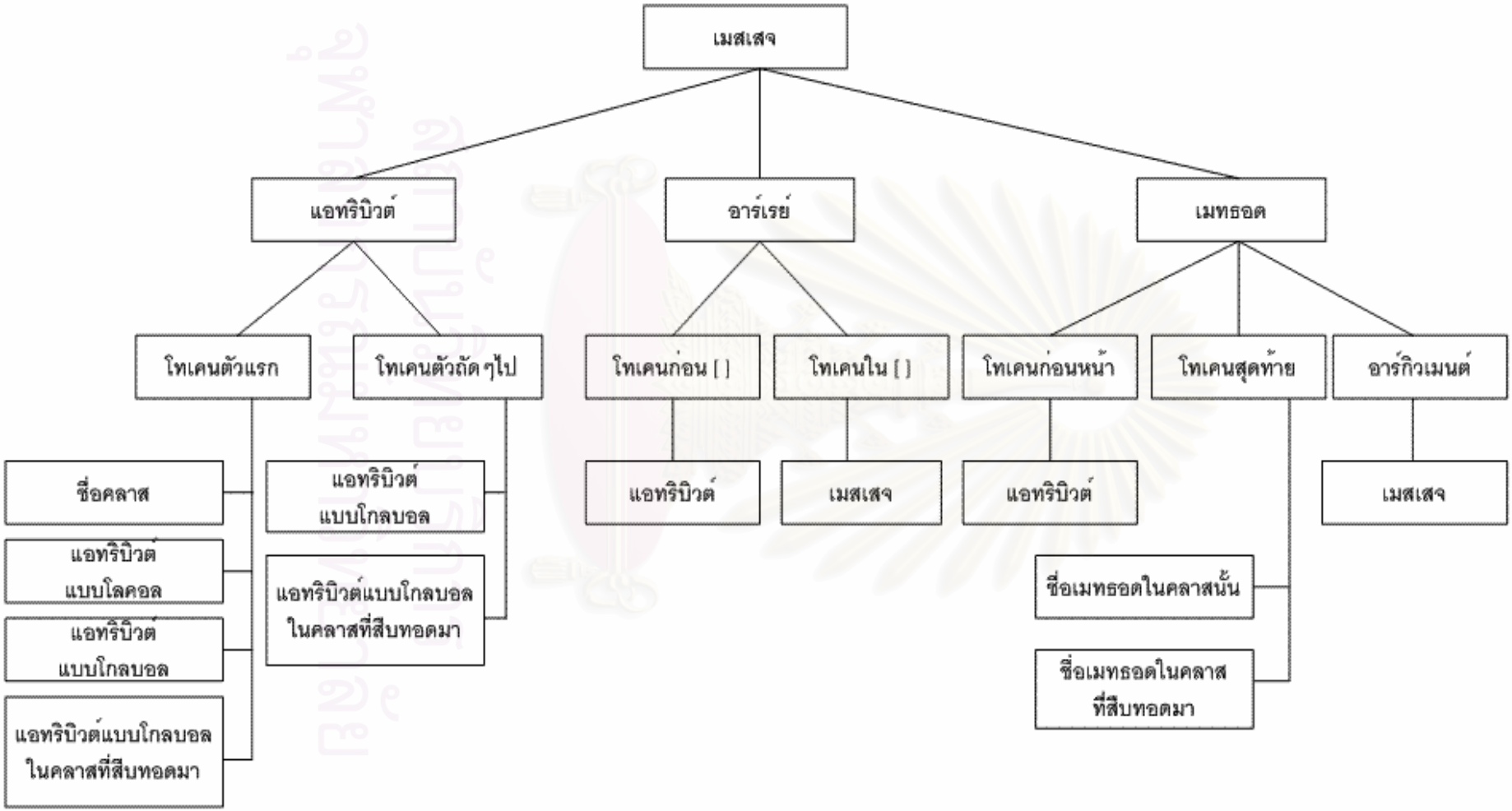
ค.1.1 ส่วนโทเคนแรก ในโทเคนแรกสามารถเป็นได้ 4 กรณีคือ

- ชื่อคลาส เช่น การเรียกใช้ B ในเมสเสจ B.b1 ในตารางที่ ค-1 จะเป็นการเรียกใช้คลาส B โดยตรง

ตารางที่ ค-1 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นชื่อคลาส

```
class A {
   public int a1;
   public A(){
      a1 = B.b1;
   }
}
```

```
class B {
    public static int b1;
}
```



รูป ค-1 แผนภาพต้นไม้การเรียกใช้เมสเสจ

- แอทริบิวต์ที่เป็นแบบโลคอล เป็นแอทริบิวต์ที่ประกาศอยู่ในเมทธอดที่เรียกใช้งาน เช่น การเรียกใช้ b ในเมสเสจ b.b1 ในตารางที่ ค-2 ซึ่งเป็นแอทริบิวต์ที่ประกาศใช้ภายในเมทธอด A()

ตารางที่ ค-2 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นแอทริบิวต์แบบโลคอล

```
class A {
   public int a1;
   public A(){
       B b = new B();
       a1 = b.b1;
   }
}
```

```
class B {
    public int b1;
}
```

- แอทริบิวต์ที่เป็นแบบโกลบอล เป็นแอทริบิวต์ที่ประกาศเป็นโกลบอลของคลาสที่เรียกใช้งาน เช่น การเรียกใช้งาน b ในเมสเสจ b.b1 ในตารางที่ ค-3 จะเรียกใช้แอทริบิวต์ b ที่ประกาศเป็นโกลบอลในคลาส A

ตารางที่ ค-3 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นแอทริบิวต์แบบโกลบอล

```
class A {
   public int a1;
   B b ;
    public A(){
       b =  new B();
       a1 = b.b1;
   }
}
```

```
class B {
    public int b1;
}
```

- แอทริบิวต์ที่เป็นแบบโกลบอลในคลาสที่มีการสืบทอด จะเป็นการเรียกใช้แอทริบิวต์ที่ประกาศเป็นแบบโกลบอลในคลาส ซึ่งเป็นคลาสที่ทำการสืบทอดมา เช่น การเรียกใช้แอทริบิวต์ b1 ในตารางที่ ค-4 ซึ่งจะเป็นการเรียกใช้แอทริบิวต์ b1 ในคลาส B ซึ่งคลาส A สืบทอดมา

ตารางที่ ค-4 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นแอทริบิวต์แบบโกลบอลที่มีการสืบทอด

```
class A extends B{
   public int a1;
   public A(){
      a1 = b1;
   }
}
```

```
class B {
    public int b1;
}
```

ค.1.2 ส่วนโทเคนถัด ๆ ไป สำหรับการเรียกใช้แอทริบิวต์โทเคนต่อ ๆ มา สามารถเป็นได้ 2 กรณีคือ

- แอทริบิวต์ที่เป็นแบบโกลบอล เป็นการเรียกใช้แอทริบิวต์ที่ประกาศเป็นโกลบอลที่สามารถเข้าถึงได้จากโทเคนหรือแอทริบิวต์ตัวก่อน เช่น การเรียกใช้งาน b1 ในเมสเสจ b.b1 ในตารางที่ ค-5 จะเป็นการเรียกใช้งานแอทริบิวต์ b1 โดยเรียกผ่านแอทริบิวต์ b ในคลาส A

ตารางที่ ค-5 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นแอทริบิวต์แบบโกลบอลที่เข้าถึงจากโทเคนก่อนหน้า

```
class A {
  public int a1;
  B b;
   public A(){
     b =  new B();
     a1 = b.b1;
   }
}
```

```
class B {
    public int b1;
}
```

- แอทริบิวต์ที่เป็นแบบโกลบอลในคลาสที่สืบทอดมา จะเป็นการเรียกใช้แอทริบิวต์ที่ประกาศเป็นแบบโกลบอลในคลาสที่สืบทอดมา ซึ่งสามารถเข้าถึงได้จากโทเคนหรือแอทริบิวต์ตัวก่อน เช่น การเรียกใช้แอทริบิวต์ c1 ในเมสเสจ b.c1 ในตารางที่ ค-6 จะเป็นการเรียกใช้แอทริบิวต์ c1 ในคลาส C โดยการเรียกผ่านแอทริบิวต์ b ในคลาส A เนื่องจากคลาส B ทำการสืบทอดคุณสมบัติมาจากคลาส C

ตารางที่ ค-6 ตัวอย่างการเรียกใช้แอทริบิวต์ที่เป็นแอทริบิวต์แบบโกลบอลในคลาสที่สืบทอดมา ที่เข้าถึงได้จากโทเคนก่อนหน้า

```
class A{
   public int a1;
  B b;
   public A(){
       b = new B();
       a1 = b.c1;
   }
}
```

```
class B extends C{
    public int b1;
}
```

```
class C{
  public int c1;
}
```

ค.2 เมสเสจที่เป็นอาร์เรย์

เมสเสจที่เป็นอาร์เรย์ สามารถแบ่งเป็น 2 ส่วนคือ

ค.2.1 ส่วนหน้าเครื่องหมาย [ ] ซึ่งก็คือ แอทริบิวต์ เช่น การเรียกใช้ b ในเมสเสจ b[getIndex()] ในตารางที่ ค-7 ซึ่งจะเป็นการเรียกใช้แอทริบิวต์ b ในคลาส A

ค.2.2 ส่วนในเครื่องหมาย [ ] ซึ่งก็คือ เมสเสจ เช่น การเรียกใช้ getIndex() ในเมสเสจ b[getIndex()] ในตารางที่ ค-8

ตารางที่ ค-7 ตัวอย่างการเรียกใช้เมสเสจที่เป็นอาร์เรย์ในส่วนก่อนเครื่องหมาย [ ]

```
class A {
  public int index;
  B[] b;
  B b1;
   public A(){
       b  = new B[1];
       b1 = b[getIndex()];
   }
   public int getIndex(){
       return index;
  }
}
```

```
class B {
}
```

ตารางที่ ค-8 ตัวอย่างการเรียกใช้เมสเสจที่เป็นอาร์เรย์ส่วนในเครื่องหมาย [ ]

```
class A {
 public int index;
  B[] b;
  B b1;
   public A(){
       b  = new B[1];
       b1 = b[getIndex()];
   }
   public int getIndex(){
       return index;
  }
}
```

```
class B {
}
```

ค.3 เมสเสจที่เป็นเมทธอด

การเรียกใช้เมสเสจที่เป็นเมทธอด สามารถแบ่งได้เป็น 3 ส่วนคือ

ค.3.1 ส่วนหน้าเครื่องหมาย ( ) ทั้งหมด ยกเว้นโทเคนสุดท้าย ซึ่งก็คือ แอทริบิวต์ โดยอาจมีหรือไม่ก็ได้ เช่น การเรียกใช้ b.c ในเมสเสจ b.c.getInt() ในตารางที่ ค-9

ตารางที่ ค-9 ตัวอย่างการเรียกใช้เมทธอดในส่วนที่เป็นการเรียกใช้แอทริบิวต์

```
class A{
   public int a1;
  B b;
   public A(){
      b = new B();
      b.c.setInt(a1);
   }
}
```

```
class B {
    public int b1;
    public C c;
    public B(){
        c = new C();
    }
}
```

```
class C{
  public int c1;
  public void setInt(int i){
       c1= i;
   }
}
```

ค.3.2 ส่วนหน้าเครื่องหมาย ( ) โทเคนสุดท้าย ซึ่งจะเป็นชื่อของเมทธอด โดยเมทธอดที่เรียกใช้นั้นจะอยู่ที่คลาสใดขึ้นอยู่กับว่ามีโทเคนในข้อ ค.3.1 หรือไม่

- ถ้าไม่มี จะเป็นเมทธอดที่อยู่ในคลาสที่เรียกใช้งานหรือคลาสที่สืบทอดมา เช่น การเรียกใช้งานเมทธอด getInt() ในตารางที่ ค-10 จะเป็นเมทธอดภายในคลาสที่เรียกใช้งาน โดยเมทธอด getInt() จะเป็นเมทธอดที่อยู่ในคลาส A ส่วนตารางที่ ค-11 จะเป็นเมทธอดในคลาสที่สืบทอดมา โดยเมทธอด getInt() เป็นเมทธอดที่อยู่ในคลาส B แต่ A สามารถเรียกใช้งานได้ เนื่องจากคลาส A ทำการสืบทอดคุณสมบัติมาจากคลาส B

ตารางที่ ค-10 ตัวอย่างการเรียกใช้เมทธอดในส่วนที่เป็นชื่อเมทธอด ที่เป็นเมทธอดภายในคลาสที่เรียกใช้งาน

```
class A {
   public int a1;
    public A(){
        a1 = getInt();
    }
    public int getInt(){
         return 0;
    }
}
```

```
class B {
     public int b1;
}
```

ตารางที่ ค-11 ตัวอย่างการเรียกใช้เมทธอดในส่วนที่เป็นชื่อเมทธอด ในคลาสที่ทำการสืบทอด

```
class A extends B{
   public int a1;
    public A(){
        a1 = getInt();
    }
}
```

```
class B {
     public int b1;
     public int getInt(){
          return 0;
     }
}
```

- ถ้ามี จะเป็นเมทธอดในคลาสที่เข้าถึงได้จากโทเคนก่อนหน้า ซึ่งอาจจะเป็นเมทธอดในคลาสนั้นหรือเมทธอดในที่คลาสดังกล่าวสืบทอดมาก็ได้ เช่นการเรียกใช้เมทธอด getInt() ในเมสเสจ b.getInt() ในตารางที่ ค-12 จะเป็นเมทธอดในคลาสจากโทเคนก่อนหน้า ซึ่งจะเป็นเมทธอดที่อยู่ในคลาส B โดยเรียกใช้ผ่านแอทริบิวต์ b ส่วนในตารางที่ ค-13 จะเป็นเมทธอดที่มีการสืบทอดมา ซึ่งเมทธอด getInt() จะอยู่ในคลาส C แต่คลาส A สามารถเรียกใช้ผ่านแอทริบิวต์ b ได้ เนื่องจากคลาส B ทำการสืบทอดคุณสมบัติมาจากคลาส C

ตารางที่ ค-12 ตัวอย่างการเรียกใช้เมทธอดในส่วนที่เป็นชื่อเมทธอด ที่เป็นเมทธอดภายในคลาสที่เข้าถึงได้จากโทเคนก่อนหน้า

```
class A {
  public int a1;
  B b ;
   public A(){
       b = new B();
       a1 = b.getInt();
   }
   public int getInt(){
        return 0;
   }
}
```

```
class B extends C{
    public int b1;
    public int getInt(){
        Return 0;
    }
}
```

```
class C {
}
```

ตารางที่ ค-13 ตัวอย่างการเรียกใช้เมทธอดในส่วนที่เป็นชื่อเมทธอด ซึ่งคลาสที่เข้าถึงได้จากโทเคนก่อนหน้าสืบทอดมา

```
class A {
  public int a1;
  B b ;
   public A(){
       b = new B();
       a1 = b.getInt();
   }
   public int getInt(){
        return 0;
   }
}
```

```
class B extends C{
    public int b1;
}
```

```
class C {
    public int getInt(){
        return 0;
    }
}
```

ค.3.3 ส่วนอาร์กิวเมนต์ คือ ส่วนในเครื่องหมาย ( ) ซึ่งจะเป็นเมสเสจ เช่น การเรียกใช้ a1 ในเมสเสจ b.c.setInt(a1) ในตาราง ค-14 ซึ่งจะเป็นเมสเสจแบบแอทริบิวต์ คือแอทริบิวต์ a1 ในคลาส A หรือ การเรียกใช้ b.getInt() ในเมสเสจ b.c.setInt(b.getInt()) ใน ซึ่งจะเป็นเมสเสจแบบเมทธอด โดยเมทธอด getInt() เป็นเมทธอดในคลาส B สามารถเรียกใช้งานโดยผ่านแอทริบิวต์ b ที่ประกาศอยู่ในคลาส A

ตารางที่ ค-14 ตัวอย่างการเรียกใช้เมทธอดที่มีอาร์กิวเมนต์เป็นเมสเสจแบบแอทริบิวต์

```
class A{
   public int a1;
  B b;
   public A(){
      b = new B();
       b.c.setInt(a1);
   }
}
```

```
class B {
    public int b1;
    public C c;
    public B(){
        c = new C();
    }
}
```

```
class C{
  public int c1;
  public void setInt(int i){
       c1= i;
   }
}
```

ตารางที่ ค-15 ตัวอย่างการเรียกใช้เมทธอดที่มีอาร์กิวเมนต์เป็นเมสเสจแบบเมทธอด

```
class A{
   public int a1;
  B b;
   public A(){
      b = new B();
       b.c.setInt( b.getInt() );
   }
}
```

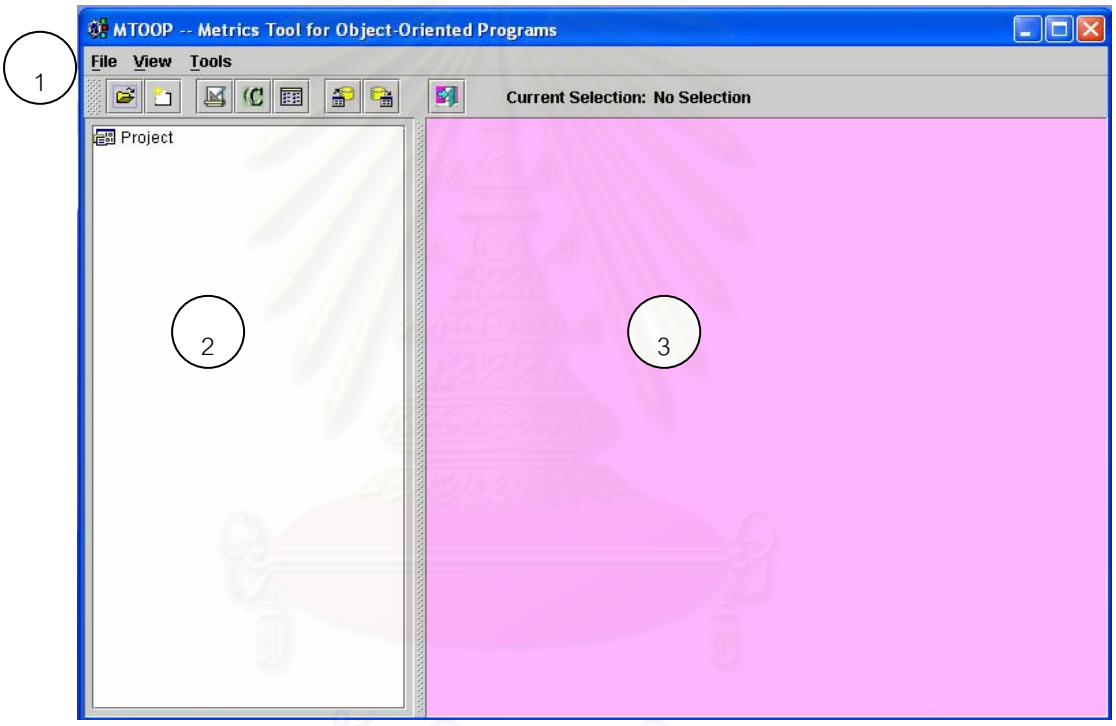
```
class B {
    public int b1;
    public C c;
    public B(){
        c = new C();
    }
    public int getInt(){
          return b1;
    }
}
```

```
class C{
  public int c1;
  public void setInt(int i){
       c1= i;
   }
}
```

ภาคผนวก ง

คู่มือการใช้งานโปรแกรม MTOOP รุ่นที่ 3

การใช้งานระบบ MTOOP ซึ่งจะแบ่งออกเป็น 3 ส่วนคือ การอ่านโปรแกรมต้นฉบับ การดูค่ามาตรวัดต่าง ๆ การเก็บและเรียกดูค่าตัววัดจากฐานข้อมูล แต่ก่อนที่จะอธิบายในรายละเอียดของแต่ละส่วน จะอธิบายหน้าจอหลักของโปรแกรมก่อน ซึ่งแสดงดังรูปที่ ง-1



รูปที่ ง-1 หน้าจอหลักของโปรแกรม MTOOP

ในหน้าจอหลักประกอบด้วย 3 ส่วนใหญ่

1. ส่วนของเมนู ประกอบด้วย 3 เมนู คือ

   - เมนู File ประกอบด้วย 3 เมนูย่อย คือ

     Add File หรือปุ่ม  ใช้ในการเพิ่มรหัสโปรแกรมที่ต้องการเก็บค่ามาตรวัด

     New Project หรือปุ่ม  ใช้ในการสร้างโปรเจคใหม่

     Exit หรือปุ่ม  ใช้ในการออกจากโปรแกรม

- เมนู View ประกอบด้วย 3 เมนูย่อย คือ

View Metrics หรือปุ่ม  ใช้ในการดูค่ามาตรวัดต่าง ๆ ซึ่งจะแสดงมาตรวัดของโปรเจค แพ็กเกจ คลาส เมทธอดและตัวแปร ในรูปแบบของแท็บ

View Complexity-Factor Metrics หรือปุ่ม  จะแสดงค่าความซับซ้อนของโปรเจค แพ็กเกจ คลาสและเมทธอด ในรูปแบบของแท็บ

View Tables หรือปุ่ม  จะแสดงตารางแสดงมาตรวัดของโปรเจค แพ็กเกจ คลาสและเมทธอด

- เมนู Tool ประกอบด้วย 2 เมนูย่อย คือ

Save to DB หรือปุ่ม  จะทำการบันทึกมาตรวัดลงในฐานข้อมูล

Load from DB หรือปุ่ม  จะทำการอ่านมาตรวัดที่เก็บอยู่ในฐานข้อมูลขึ้นมา

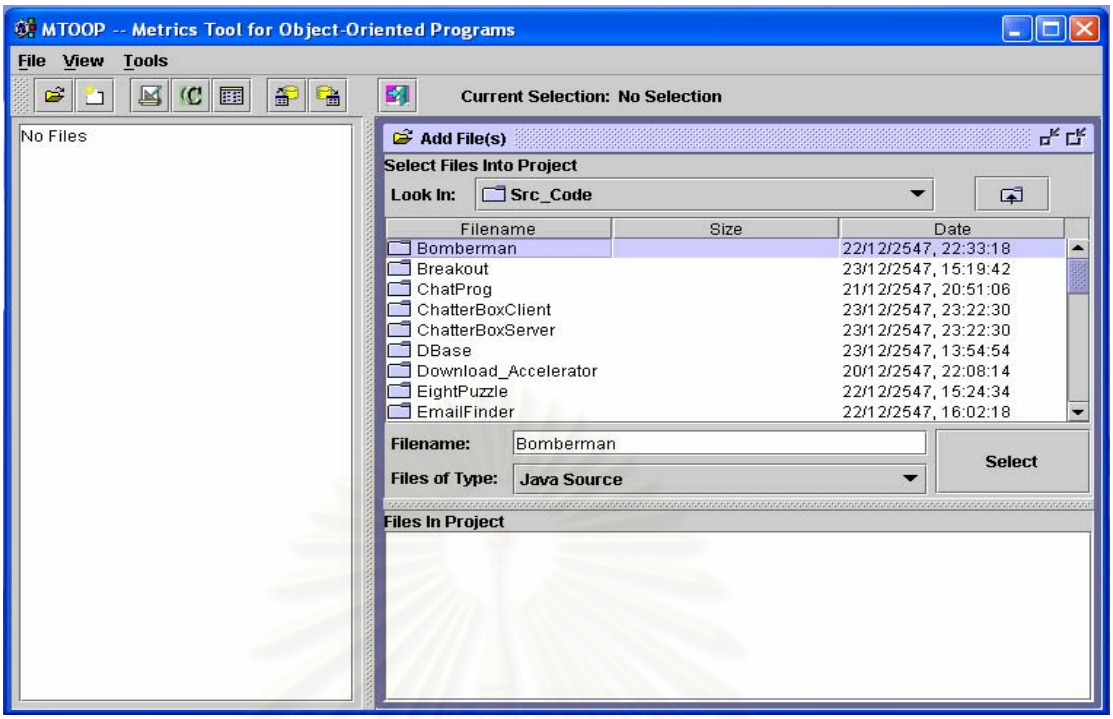
2. ส่วนของโครงสร้างต้นไม้แสดงโปรเจคของรหัสโปรแกรม ซึ่งจะเป็นแผนภาพต้นไม้แสดงส่วนประกอบของโปรเจคว่าประกอบด้วยแพ็กเจก คลาส เมทธอดและแอทริบิวต์อะไรบ้าง

3. ส่วนแสดงผลการทำงาน จะเป็นส่วนที่ใช้แสดงค่ามาตรวัดต่าง ๆ

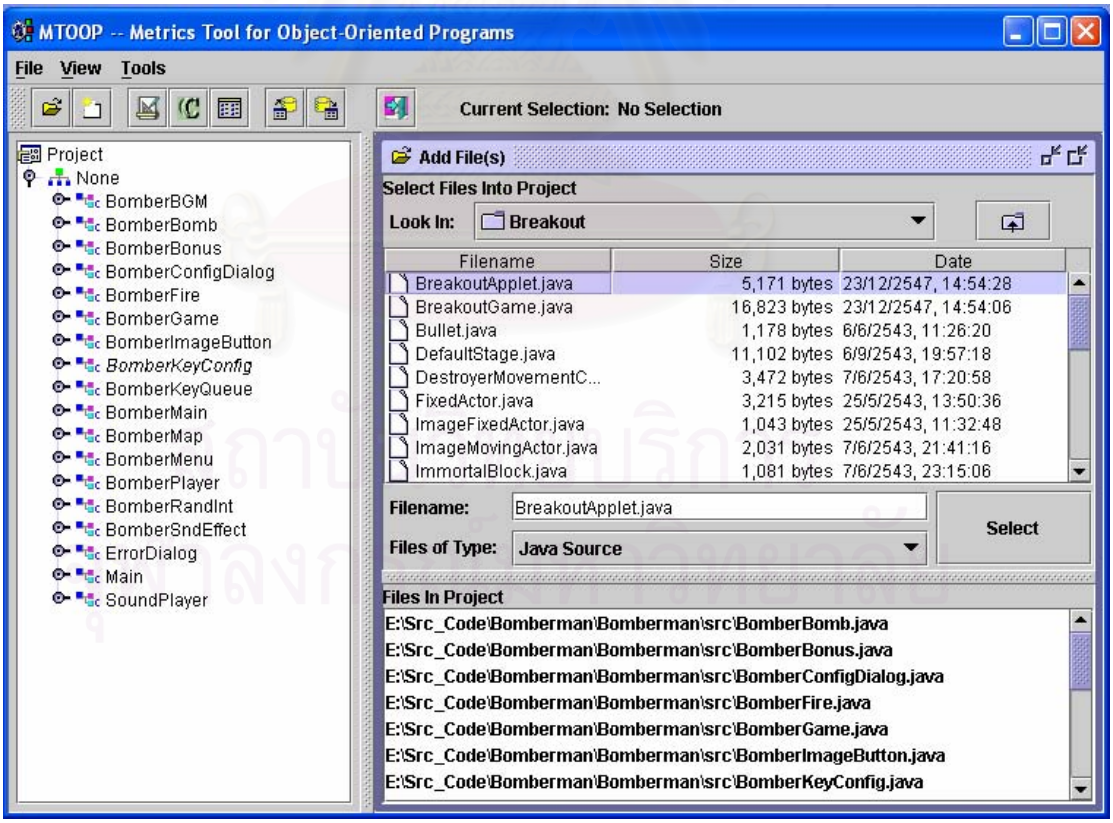
ง.1 การอ่านรหัสโปรแกรม

เมื่อผู้ใช้งานต้องการนำเข้าไฟล์รหัสโปรแกรมที่ต้องการเก็บค่ามาตรวัด ก็ทำได้โดยการเลือกเมนู  (Add File) แล้วเลือกไฟล์รหัสโปรแกรมที่ต้องการโดยสามารถเลือกทีละไฟล์หรือเลือกทั้งไดเรคทอรีก็ได้ ดังรูปที่ ง-2

หลังจากผู้ใช้ได้เลือกรหัสโปรแกรมเข้าสู่โปรแกรมแล้ว โปรแกรมจะแสดงชื่อของโปรเจค แพ็กเกจ คลาส เมทธอด และตัวแปรในรูปแบบต้นไม้ (Tree) ดังรูปที่ ง-3 ทางด้านซ้ายมือ เพื่อให้ผู้ใช้สามารถเลือกดูค่าตัววัดต่าง ๆ ตามที่ผู้ใช้ต้องการ หรือถ้าผู้ใช้ต้องการเลือกไฟล์รหัสโปรแกรมเพิ่มเติมเข้าสู่โปรเจค ผู้ใช้ก็สามารถใช้เมนู  (Add File) ได้ หรือถ้าหากต้องการสร้างโปรเจคใหม่ก็เลือกเมนู  (New Project) หรือหากต้องการออกจากโปรแกรมก็เลือกเมนู  (Exit)



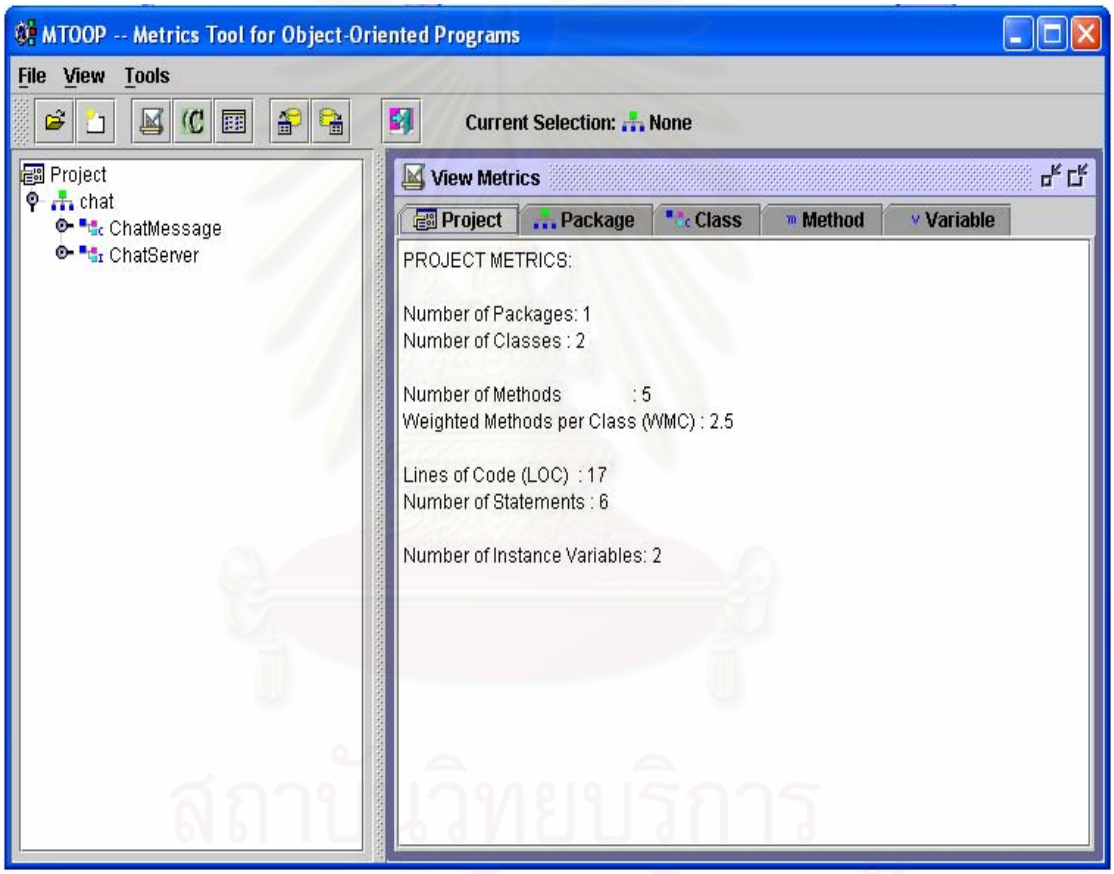
รูปที่ ง-2 การเพิ่มไฟล์รหัสโปรแกรมเข้าในโปรแกรม MTOOP



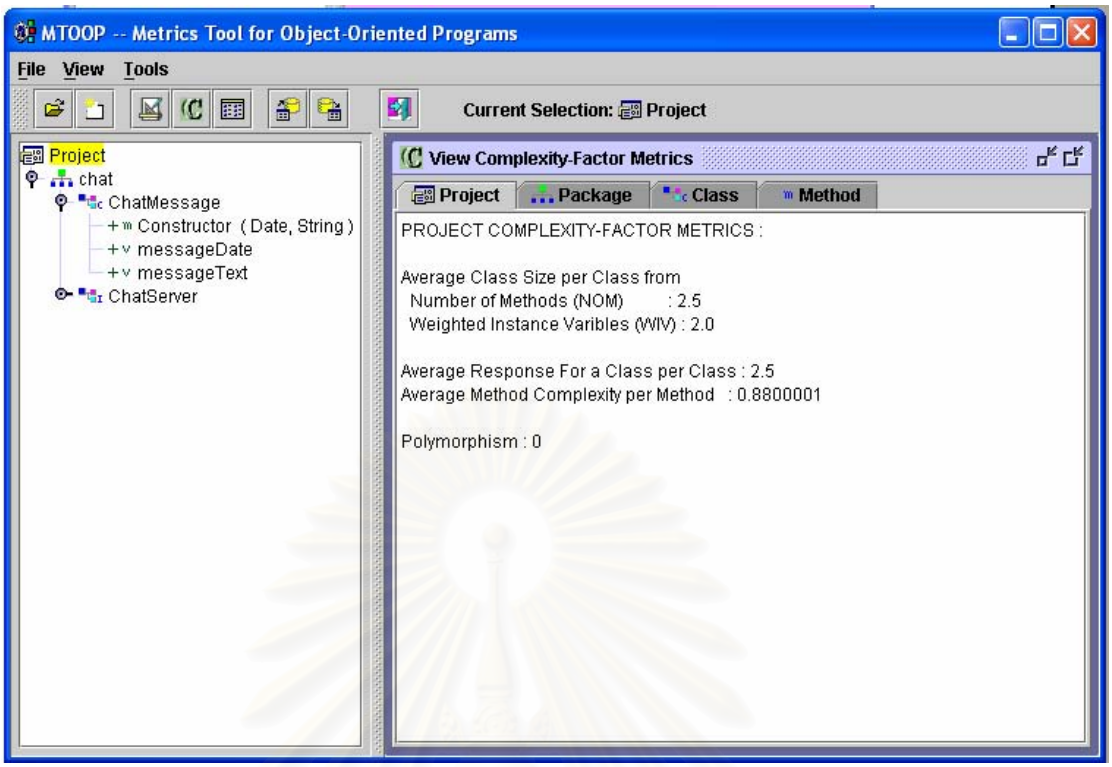
รูปที่ ง-3 ต้นไม้และโหนดต่าง ๆ ที่ได้จากไฟล์รหัสโปรแกรมที่เลือก

ง.2 การเรียกดูค่ามาตรวัด

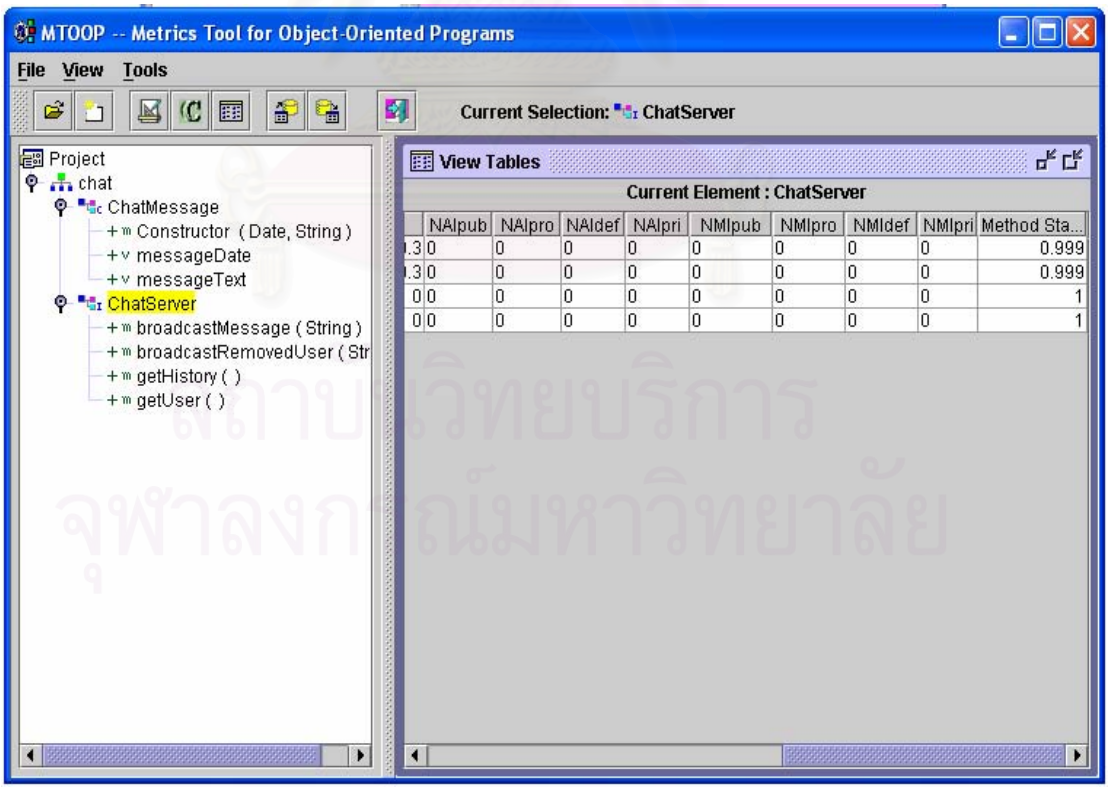
เมื่อผู้ใช้ต้องการดูค่ามาตรวัด ก็เลือกเมนู (View Metrics) จะแสดงมาตรวัดของโปรเจค แพ็กเกจ คลาส เมทธอดและตัวแปร ในรูปแบบของแท็บ ดังรูปที่ ง-4 แต่ถ้าต้องการดูค่ามาตรวัดความซับซ้อนของรหัสโปรแกรม ก็เลือกเมนู (View Complexity-Factor Metrics) จะแสดงความซับซ้อนของโปรเจค แพ็กเกจ คลาสและเมทธอด ในรูปแบบของแท็บ ดังรูปที่ ง-5 หากต้องการดูมาตรวัดต่าง ๆ ของรหัสโปรแกรม ก็เลือกเมนู (View Tables) ซึ่งจะแสดงมาตรวัดของโปรเจค แพ็กเกจ คลาสและเมทธอด ในรูปของตาราง ดังรูปที่ ง-6



รูปที่ ง-4 มาตรวัดของรหัสโปรแกรมเมื่อเลือกเมนู View Metrics



รูปที่ ง-5 มาตรวัดความซับซ้อนของรหัสโปรแกรมเมื่อเลือกเมนู View Complexity-Factor Metrics

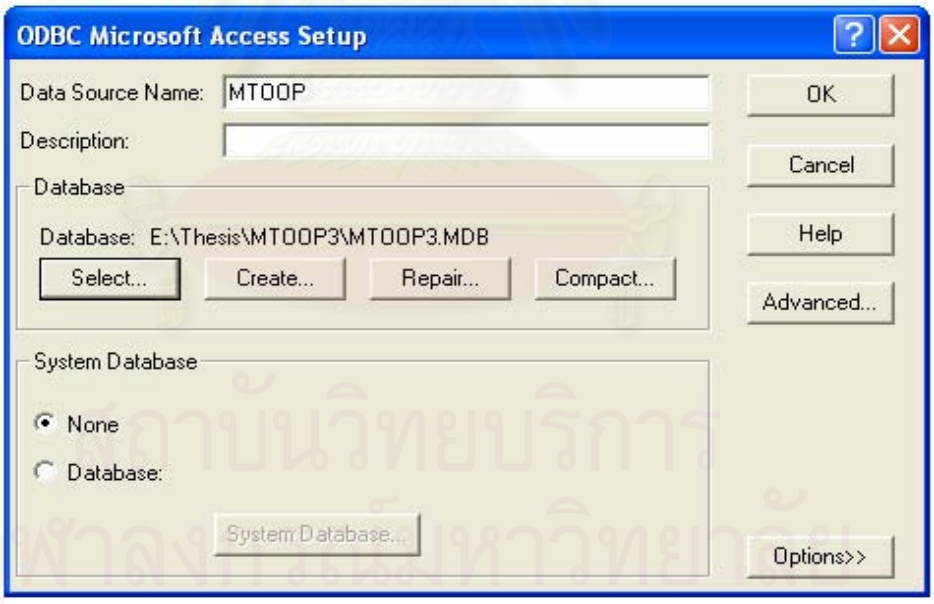


รูปที่ ง-6 ค่ามาตรวัดของรหัสโปรแกรมเมื่อเลือกเมนู View Tables

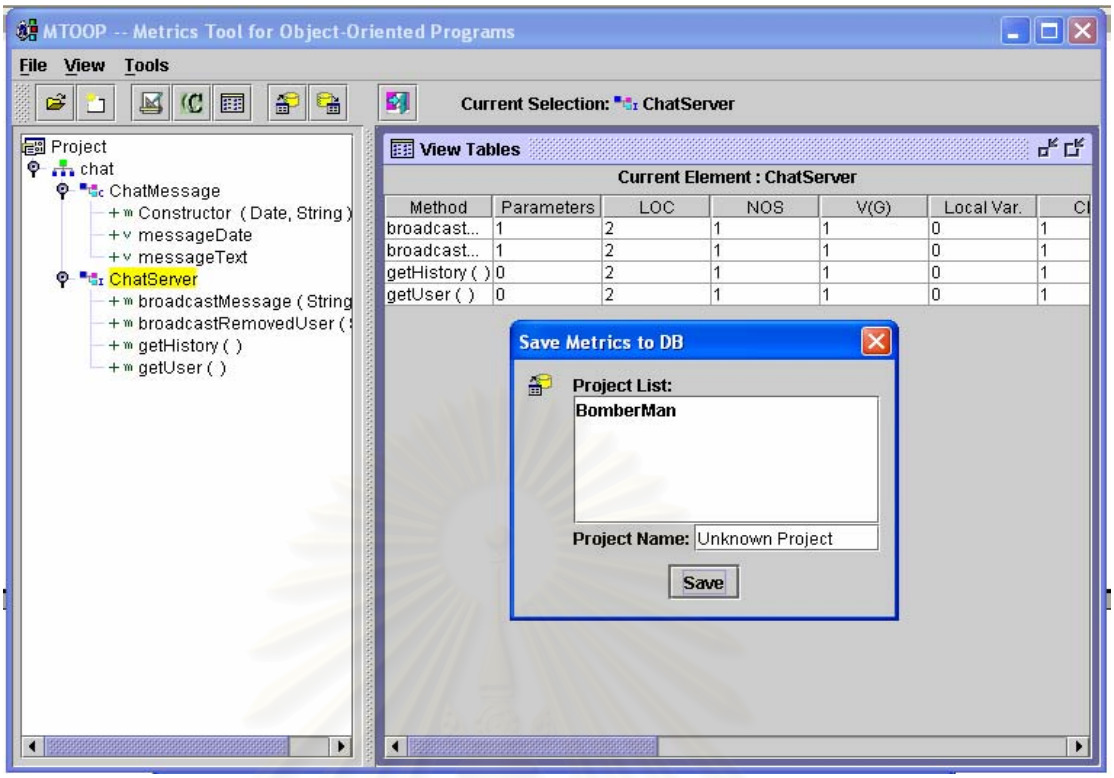
ง.3 การเก็บและการเรียกดูมาตรวัดจากฐานข้อมูล

การเก็บค่าตัววัดของโปรเจคลงฐานข้อมูล เนื่องจากเครื่องมือวัดนี้จะทำการบันทึกข้อมูลลงฐานข้อมูลด้วยการผ่านเจดีบีซี (JDBC – Java Database Connectivity) ซึ่งต้องมีการกำหนดช่องทางติดต่อเพื่อใช้รับส่งข้อมูลด้วยการใช้โอดีบีซี (ODBC – Open Database Connectivity) โดยใช้ไดรเวอร์ของไมโครซอฟต์แอคเซส (Microsoft Access Driver) และกำหนดดาต้าซอร์สเนม (Data Source Name) ชื่อ “MTOOP” และไฟล์ดาต้าเบสชื่อ “mtoop.mdb” ในเครื่องมือจัดการโอดีบีซี (ODBC Data Source Administrator) ผู้ใช้จึงจะสามารถเก็บบันทึกหรือร้องขอข้อมูลได้ ดังรูปที่ ง.7

ผู้ใช้สามารถเก็บค่ามาตรวัดที่เก็บจากรหัสโปรแกรมไว้เรียกดูภายหลังได้ โดยเมื่อผู้ใช้เลือกเมนู  (Save to DB) โปรแกรมจะแสดงไดอะล็อกบ็อกซ์ (Dialog Box) เพื่อให้เลือกชื่อโปรเจคที่ได้มีการบันทึกลงฐานข้อมูลไปแล้ว หรือให้กรอกชื่อโปรเจคที่ต้องการบันทึกโดยระบบได้กำหนดชื่อโปรเจคไว้เป็นค่าเริ่มต้นคือ Unknown Project และมีปุ่มบันทึกข้อมูล (Save) เพื่อทำการบันทึกข้อมูลลงฐานข้อมูล ดังรูปที่ ง-8

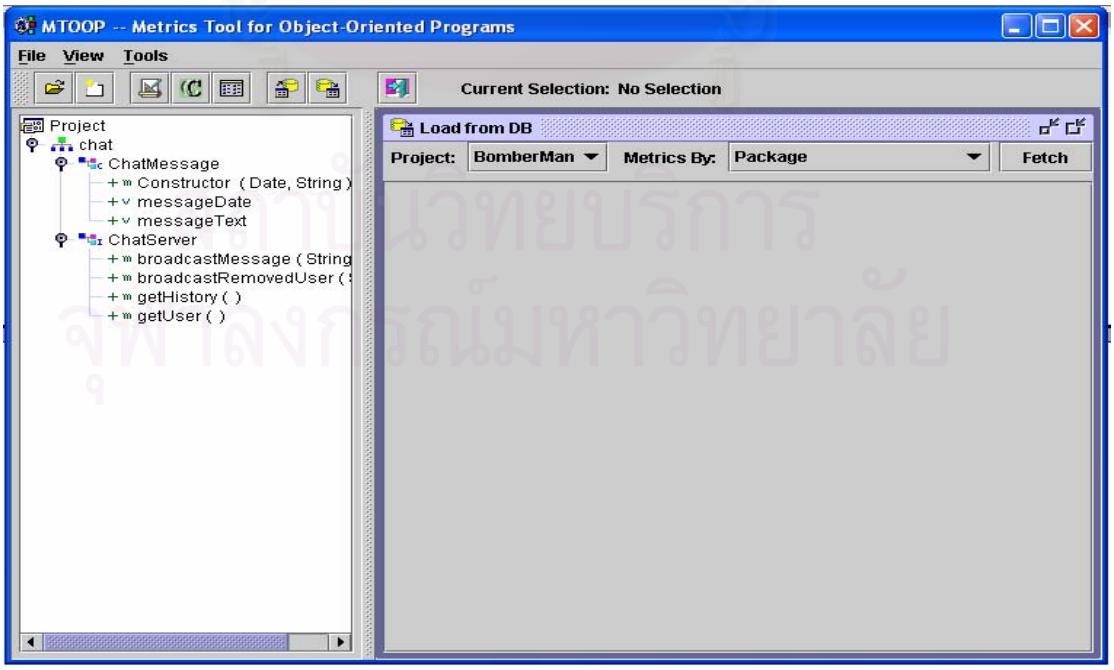


รูปที่ ง-7 การกำหนดค่าฐานข้อมูลเพื่อใช้ในการเก็บมาตรวัด

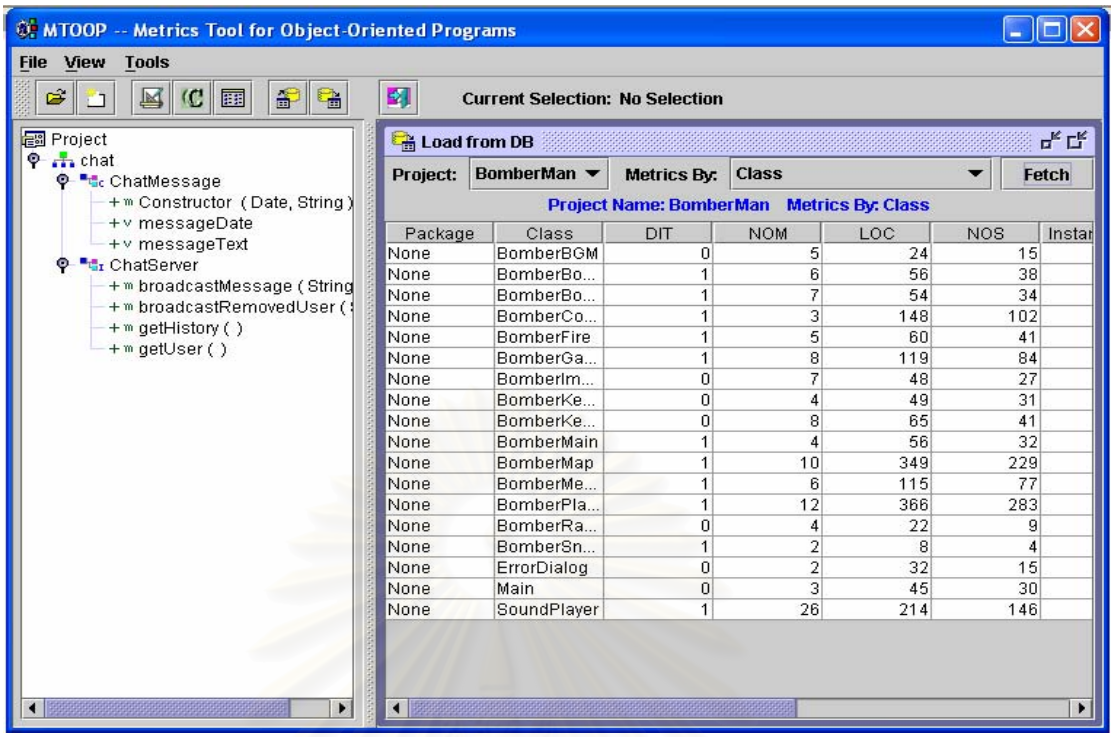


รูปที่ ง-8 ตัวอย่างการใช้เรียกใช้เมนู Save to DB

นอกจากนี้ ผู้ใช้ยังสามารถเรียกดูมาตรวัดที่เก็บไว้ในฐานข้อมูลในภายหลังได้ โดยเลือกเมนู  (Load from DB) จะแสดงรายชื่อโปรเจคที่เก็บไว้ในฐานข้อมูล โดยสามารถเลือกได้ว่าต้องการดูมาตรวัดของแพ็กเกจ คลาสหรือเมทธอด ดังรูปที่ ง-9 เมื่อกดปุ่ม Fetch โปรแกรมก็แสดงผลมาตรวัดในโปรเจคที่เลือก ดังรูปที่ ง-10



รูปที่ ง-9 ตัวอย่างการใช้เรียกใช้เมนู Load from DB

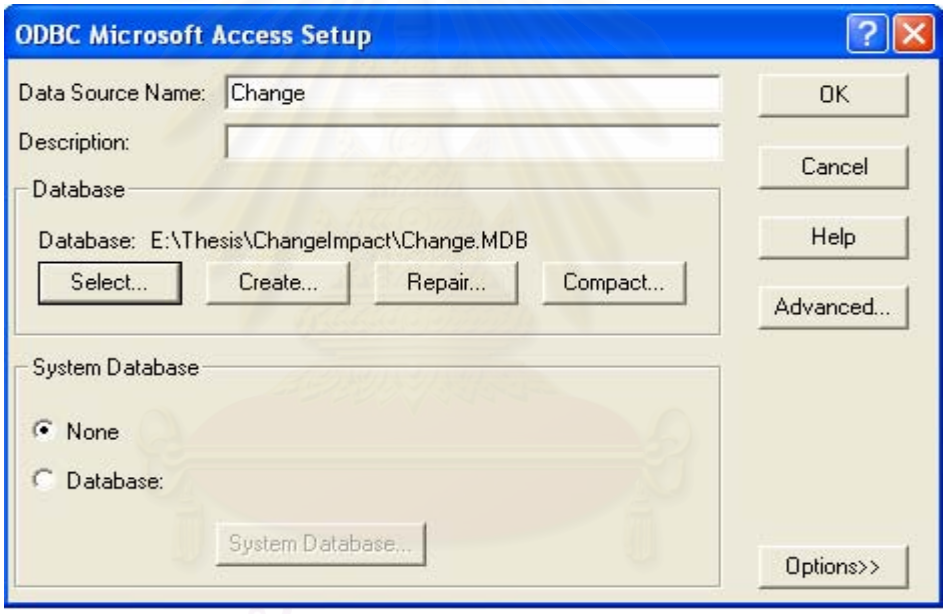


รูปที่ ง-10 ตัวอย่างผลการใช้เรียกใช้เมนู Load from DB

ภาคผนวก จ

คู่มือการใช้งานโปรแกรม ChangeImpact

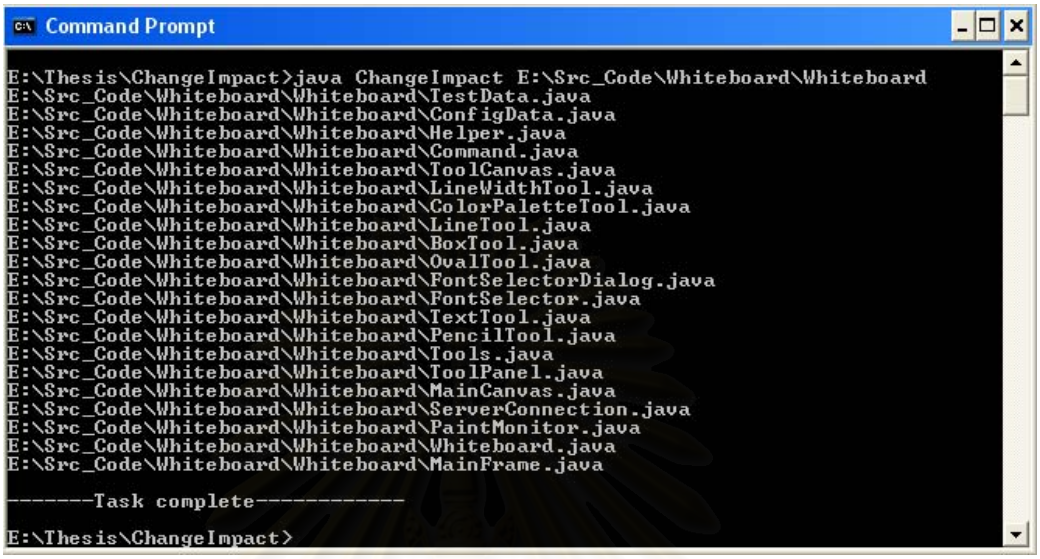
ก่อนการใช้งานโปรแกรม ChangeImpact ต้องมีการกำหนดค่าฐานข้อมูลก่อน เนื่องจากข้อมูลเสถียรภาพที่ได้จากเครื่องมือจะทำการบันทึกลงฐานข้อมูลผ่านเจดีบีซี ซึ่งต้องมีการกำหนดช่องทางติดต่อเพื่อใช้รับส่งข้อมูลด้วยการใช้โอดีบีซี โดยใช้ ไดรเวอร์ของไมโครซอฟต์แอคเซสและกำหนดดาต้าซอร์สเนม ชื่อ “Change” และไฟล์ดาต้าเบสชื่อ “change.mdb” ในเครื่องมือจัดการโอดีบีซี ผู้ใช้จึงจะสามารถเก็บบันทึกหรือเรียกใช้ข้อมูลได้ ตัวอย่างการตั้งค่าฐานข้อมูลแสดงได้ดังรูปที่ จ-1



รูปที่ จ-1 การกำหนดค่าฐานข้อมูลเพื่อใช้ในการเก็บข้อมูลเสถียรภาพ

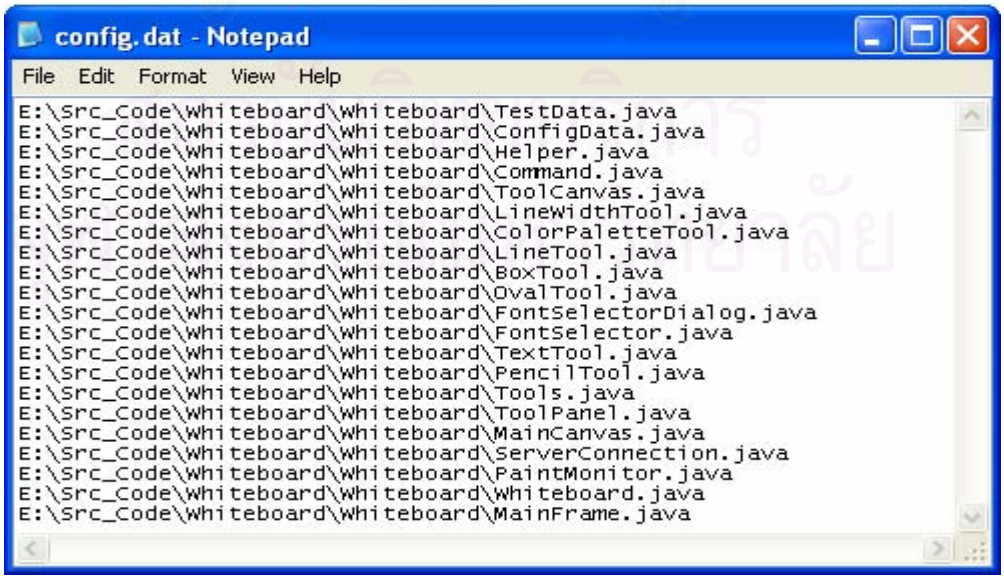
การใช้งานโปรแกรม ChangeImpact เป็นการใช้งานผ่านบรรทัดคำสั่ง (Command line) ในหน้าจอดอส (DOS) โดยโปรแกรมจะเปลี่ยนแปลงรหัสโปรแกรมแล้วหาผลกระทบที่เกิดขึ้น เมื่อเปลี่ยนแปลงรหัสโปรแกรมเรียบร้อยแล้ว จึงคำนวณเป็นเสถียรภาพของคลาสและเสถียรภาพของเมทธอด หลังจากนั้นจึงบันทึกเสถียรภาพที่ได้ลงฐานข้อมูล สำหรับรหัสโปรแกรมที่ต้องการให้โปรแกรม ChangeImpact คำนวณเสถียรภาพสามารถกำหนดได้ 2 วิธี คือ

1. กำหนดเป็นอาร์กิวเมนต์ สำหรับวิธีนี้จะรับชื่อไดเรคทอรี่เป็นอาร์กิวเมนต์ในขณะสั่งให้โปรแกรมทำงาน ซึ่งโปรแกรมจะคำนวณเสถียรภาพของรหัสโปรแกรมทุกไฟล์ที่มีอยู่ในไดเรคทอรี่นั้น รวมทั้งไดเรคทอรี่ย่อยด้วย ตัวอย่างการเรียกใช้โปรแกรมแสดงได้ดังรูปที่ จ-2

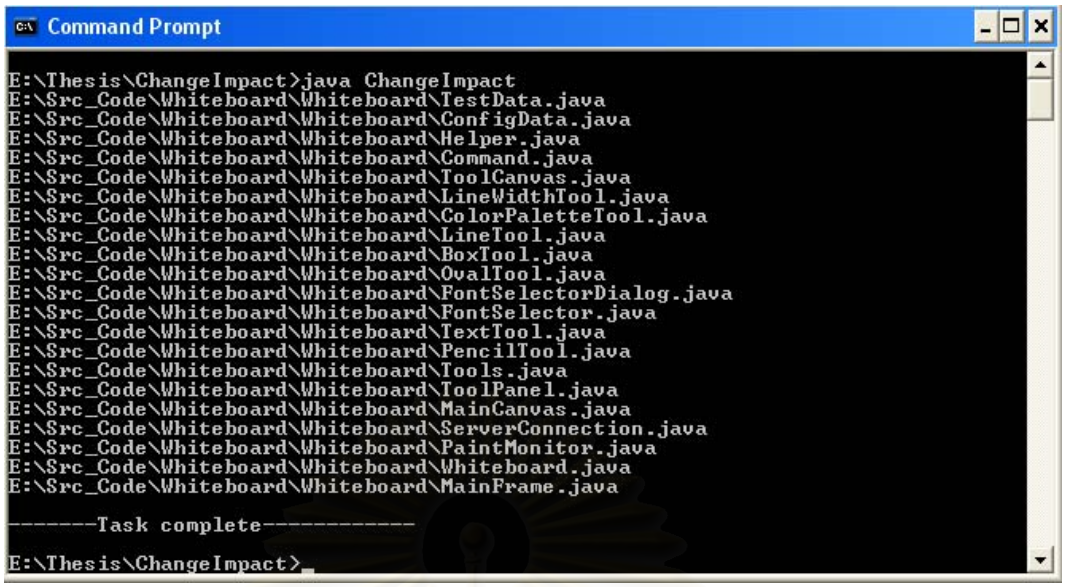


รูปที่ จ-2 การเรียกใช้โปรแกรม ChangeImpact แบบที่ 1 (มีอาร์กิวเมนต์)

2. กำหนดเป็นไฟล์ สำหรับวิธีนี้สามารถกำหนดไฟล์รหัสโปรแกรมที่ต้องการคำนวณเสถียรภาพได้ โดยใส่ชื่อไฟล์ที่ต้องการคำนวณลงในไฟล์ config.dat ส่วนการเรียกใช้งานโปรแกรม ChangeImpact สามารถเรียกโปรแกรมได้เลย ไม่ต้องกำหนดอาร์กิวเมนต์เหมือนกับวิธีที่ 1 โดยโปรแกรม ChangeImpact จะอ่านข้อมูลในไฟล์ config.dat โดยอัตโนมัติ ตัวอย่างข้อมูลในไฟล์ config.dat แสดงได้ดังรูปที่ จ-3 ส่วนรูปที่ จ-4 เป็นตัวอย่างการเรียกใช้โปรแกรม

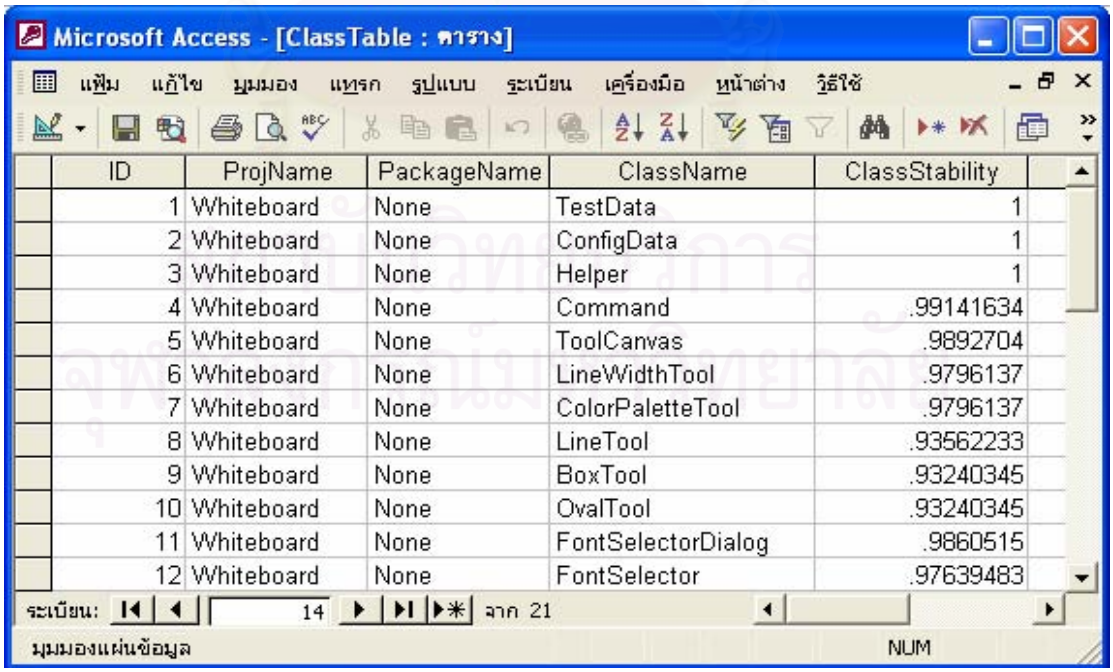


รูปที่ จ-3 ตัวอย่างข้อมูลในไฟล์ config.dat



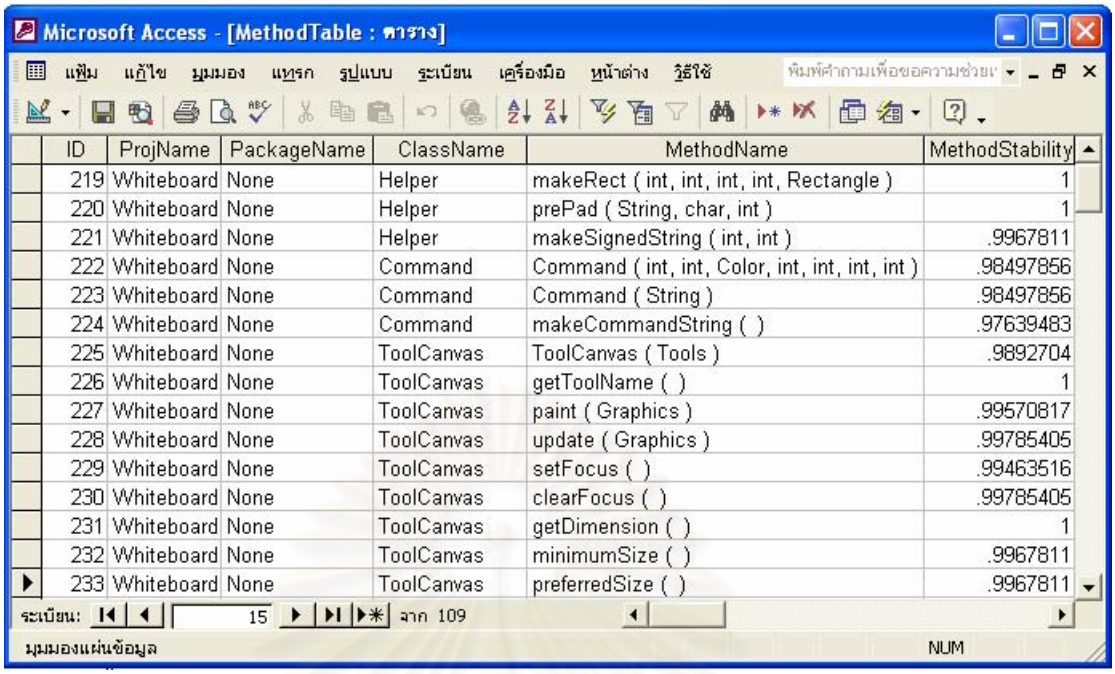
รูปที่ จ-4 การเรียกใช้โปรแกรม ChangeImpact แบบที่ 2 (ไม่มีอาร์กิวเมนต์)

สำหรับข้อมูลเสถียรภาพที่ได้จากโปรแกรม ChangeImpact ประกอบด้วยเสถียรภาพของคลาส ในตาราง ClassTable และเสถียรภาพของเมทธอด ในตาราง MethodTable ในไฟล์ change.mdb ตัวอย่างเสถียรภาพของคลาสแสดงได้ดังรูปที่ จ-5 ส่วนตัวอย่างเสถียรภาพของเมทธอดแสดงได้ดังรูปที่ จ-6



ID	ProjName	PackageName	ClassName	ClassStability
1	Whiteboard	None	TestData	1
2	Whiteboard	None	ConfigData	1
3	Whiteboard	None	Helper	1
4	Whiteboard	None	Command	.99141634
5	Whiteboard	None	ToolCanvas	.9892704
6	Whiteboard	None	LineWidthTool	.9796137
7	Whiteboard	None	ColorPaletteTool	.9796137
8	Whiteboard	None	LineTool	.93562233
9	Whiteboard	None	BoxTool	.93240345
10	Whiteboard	None	OvalTool	.93240345
11	Whiteboard	None	FontSelectorDialog	.9860515
12	Whiteboard	None	FontSelector	.97639483

รูปที่ จ-5 ตัวอย่างเสถียรภาพของคลาสที่ได้จากโปรแกรม ChangeImpact



รูปที่ จ-6 ตัวอย่างเสถียรภาพของเมทธอดที่ได้จากโปรแกรม ChangeImpact

ภาคผนวก ฉ

ตัวอย่างการคำนวณเสถียรภาพของเมทธอดและเสถียรภาพของคลาส

ในหัวข้อนี้เป็นการยกตัวอย่างในการคำนวณเสถียรภาพของเมทธอดและของคลาส ตามรูปแบบการเปลี่ยนแปลงแอทริบิวต์และเมทธอด ตามตารางที่ 3.3 และ ตารางที่ 3.4 ตามลำดับ โดยในตาราง ฉ-1 เป็นตัวอย่างรหัสโปรแกรมที่ต้องการคำนวณเสถียรภาพ ส่วนการคำนวณเสถียรภาพแสดงได้ดังตารางที่ ฉ-2

ตารางที่ ฉ-1 ตัวอย่างรหัสโปรแกรมที่ต้องการคำนวณเสถียรภาพ

```
public class A{
   public int a1;
  public int mA1(){
      B b = new B();
      a1=  b.b1;
      return a1;
  }
}
```

```
public class B{
   public int b1;
   public int mB1(){
      b1 = 10;
      return b1;
  }
}
```

```
public class C{
   public int c1;
  public int mC1(){
      B b = new B();
      c1 = b.mB1();
      return c1;
  }
}
```

ตารางที่ ฉ-2 ตัวอย่างการคำนวณเสถียรภาพ

ครั้งที่	การเปลี่ยนแปลง	mA1()	mB1()	mC1()	class A	class B	class C
1	เปลี่ยนชนิด a1	X					
2	ลบแอทริบิวต์ a1	X					
3	ขอบเขต a1 (pub -> pro)						
4	ขอบเขต a1 (pub -> def)						
5	ขอบเขต a1 (pub -> pri)						

ตารางที่ ฉ-2 ตัวอย่างการคำนวณเสถียรภาพ (ต่อ)

6	เปลี่ยนชนิด b1	X	X		X		
7	ลบแอทริบิวต์ b1	X	X		X		
8	ขอบเขต b1 (pub -> pro)						
9	ขอบเขต b1 (pub -> def)						
10	ขอบเขต b1 (pub -> pri)	X			X		
11	เปลี่ยนชนิด c1			X			
12	ลบแอทริบิวต์ c1			X			
13	ขอบเขต c1 (pub -> pro)						
14	ขอบเขต c1 (pub -> def)						
15	ขอบเขต c1 (pub -> pri)						
16	เปลี่ยนลายเซ็นต์ mA1()						
17	ขอบเขต mA1() (pub -> pro)						
18	ขอบเขต mA1() (pub -> def)						
19	ขอบเขต mA1() (pub -> pri)						
20	ลบเมทธอด mA1()						
21	เปลี่ยนชนิดค่าส่งกลับของ เมทธอด mA1()						
22	เปลี่ยนลายเซ็นต์ mB1()			X			X
23	ขอบเขต mB1() (pub -> pro)						
24	ขอบเขต mB1() (pub -> def)						
25	ขอบเขต mB1() (pub -> pri)			X			X
26	ลบเมทธอด mB1()			X			X

ตารางที่ ฉ-2 ตัวอย่างการคำนวณเสถียรภาพ (ต่อ)

28	เปลี่ยนชนิดค่าส่งกลับของเมทธอด mB1()			X			X
27	เปลี่ยนลายเซ็นต์ mC1()						
29	ขอบเขต mC1() (pub -> pro)						
30	ขอบเขต mC1() (pub -> def)						
31	ขอบเขต mC1() (pub -> pri)						
32	ลบเมทธอด mC1()						
33	เปลี่ยนชนิดค่าส่งกลับของเมทธอด mC1()						
	รวม	5	2	6	3	0	4
	เสถียรภาพ	1-(5/33) = 0.84	1-(2/33) = 0.93	1-(6/33) = 0.81	1-(3/33) = 0.90	1-(0/33) = 1.0	1-(4/33) = 0.87

ภาคผนวก ช

มาตรวัดเชิงวัตถุสำหรับเมทธอด

ในหัวข้อนี้จะกล่าวถึงมาตรวัดเชิงวัตถุสำหรับเมทธอดที่ใช้ในงานวิจัยนี้ ซึ่งประกอบด้วย

ช.1 มาตรวัดขนาดของเมทธอด

ช.1.1 มาตรวัดจำนวนพารามิเตอร์ (Parameter: Param) คือจำนวนพารามิเตอร์ของเมทธอด

ช.1.2 มาตรวัดจำนวนตัวแปรแบบโลคอล (Local variable: LocalVar) คือจำนวนตัวแปรที่ประกาศใช้ในเมทธอด (ตัวแปรแบบโลคอล)

ช.1.3 มาตรวัดจำนวนสเตทเมนต์ (Number Of Statement: NOS) คือจำนวนสเตทเมนต์ในเมทธอด

ช.1.4 มาตรวัดจำนวนบรรทัดของรหัสโปรแกรม (Line Of Code: LOC) คือจำนวนบรรทัดของรหัสโปรแกรมของเมทธอด

ช.2 มาตรวัดความซับซ้อนของเมทธอด

ช.2.1 มาตรวัดไซโคลเมติกของแมคเคบ (Cyclomatic complexity: V(G)) คือจำนวนเงื่อนไขบวกหนึ่ง

ช.2.2 มาตรวัดความซับซ้อนของเมทธอด (Method CompleXity: MCX) คือจำนวนข้อความแต่ละชนิดคูณกับค่าถ่วงน้ำหนัก

ช.3 มาตรวัดการเข้าคู่ระหว่างเมทธอด

ช.3.1 มาตรวัดการเข้าคู่กันระหว่างวัตถุ (Coupling Between Object: CBO) คือจำนวนคลาสที่ถูกใช้ในเมทธอด โดยเป็นคลาสที่ไม่ซ้ำ

ช.3.2 มาตรวัดจำนวนการเรียกใช้เมทธอดแบบ public (Number of public Method Invocation :NMIpub) คือจำนวนการเรียกใช้เมทธอดแบบ public ในเมทธอด

ช.3.3 มาตรวัดจำนวนการเรียกใช้เมทธอดแบบ protected (Number of protected Method Invocation :NMIpro) คือจำนวนการเรียกใช้เมทธอดแบบ protected ในเมทธอด

ช.3.4 มาตรวัดจำนวนการเรียกใช้เมทธอดแบบ default (Number of default Method Invocation :NMIdef) คือจำนวนการเรียกใช้เมทธอดแบบ defaultในเมทธอด

ช.3.5 มาตรวัดจำนวนการเรียกใช้เมทธอดแบบ private (Number of private Method Invocation :NMIpri) คือจำนวนการเรียกใช้เมทธอดแบบ private ในเมทธอด

ช.3.6 มาตรวัดจำนวนการเรียกใช้แอทริบิวต์แบบ public (Number of public Attribute Invocation :NAIpub) คือจำนวนการเรียกใช้แอทริบิวต์แบบ public ในเมทธอด

ช.3.7 มาตรวัดจำนวนการเรียกใช้แอทริบิวต์แบบ protected (Number of protected Attribute Invocation :NAIpro) คือจำนวนการเรียกใช้แอทริบิวต์แบบ protected ในเมทธอด

ช.3.8 มาตรวัดจำนวนการเรียกใช้แอทริบิวต์แบบ default (Number of default Attribute Invocation :NAIdef) คือจำนวนการเรียกใช้แอทริบิวต์แบบ default ในเมทธอด

ช.3.9 มาตรวัดจำนวนการเรียกใช้แอทริบิวต์แบบ private (Number of private Attribute Invocation :NAIpri) คือจำนวนการเรียกใช้แอทริบิวต์แบบ private ในเมทธอด

ประวัติผู้เขียนวิทยานิพนธ์

นายธีรเดช แซ่ตั้น เกิดเมื่อวันที่ 26 มกราคม พ.ศ. 2524 ที่จังหวัดฉะเชิงเทรา สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ประยุกต์ คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ในปีการศึกษา 2544 และได้เข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ณ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2545