ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์ โดยใช้มาตรวัดเชิงวัตถุ

นายศิรธันย์ ศุภธนะรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

A RELATIONSHIP BETWEEN REFACTORING AND SOFTWARE QUALITY USING OBJECT-ORIENTED METRICS

Mr. Sirathan Suppatanarat

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Business Software Development

Department of Statistics
Faculty of Commerce and Accountancy
Chulalongkorn University

Academic Year 2006

หัวข้อวิทยานิพนธ์	ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์โดยใช้มาตรวัดเชิงวัตถุ
โดย	นายศิรธันย์ ศุภธนะรัตน์
สาขาวิชา	การพัฒนาซอฟต์แวร์ด้านธุรกิจ
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะพาณิชยศาสตร์และการบัญชี
(รองศาสตราจารย์ ดร. ดนุชา คุณพนิชกิจ)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ถาวร อานุภาพไตรรงค์)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์)

.....กรรมการ
(อาจารย์ ดร. วัชรา จันทาทับ)

ศิรธันย์ ศุภธนะรัตน์ : ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์ โดยใช้มาตรวัดเชิงวัตถุ. (A RELATIONSHIP BETWEEN REFACTORING AND SOFTWARE QUALITY USING OBJECT-ORIENTED METRICS) อ. ที่ปรึกษา : ผศ.ดร. อัษฎาพร ทรัพย์สมบูรณ์, 211 หน้า.

วัตถุประสงค์ของการวิจัยนี้คือ เพื่อ (1) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี (2) เปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี และ (3) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน ซึ่งคุณภาพซอฟต์แวร์คือ ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability) โดยใช้มาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995)

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง ซึ่งหน่วยทดลองที่นำมาใช้เพื่อตอบวัตถุประสงค์ของงานวิจัยนี้คือ ผลงานการโปรแกรมของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) โดยผู้วิจัยได้กำหนดให้ผลงานที่นำมาเป็นหน่วยทดลองต้องเป็นโปรแกรมที่พัฒนาด้วยภาษาซีชาร์ป (C#) ที่เป็นการประยุกต์เชิงธุรกิจและต้องมีจำนวนคลาสภายในตั้งแต่ 5 คลาสขึ้นไป ซึ่งมีหน่วยทดลองที่สามารถนำมาใช้ในการวิจัยได้ทั้งหมด 32 หน่วยทดลอง

ผลการวิจัยพบว่า การนำกระบวนการรีแฟคทอริงมาใช้ในซอร์สโค้ดที่ถูกพัฒนาโดยวิธีการเชิงวัตถุ ทำให้คุณภาพซอฟต์แวร์ดีขึ้นไม่ว่าจะทำกระบวนการรีแฟคทอริง 1 วิธีหรือ 2 วิธี แต่การสลับลำดับการนำกระบวนการรีแฟคทอริงไปใช้ไม่ทำให้คุณภาพซอฟต์แวร์แตกต่างกัน

ภาควิชา.....สถิติ..... ลายมือชื่อนิสิต.....ศิรธันย์ ศุภธนะรัตน์.....

สาขาวิชา...การพัฒนาซอฟต์แวร์ด้านธุรกิจ... ลายมือชื่ออาจารย์ที่ปรึกษา...อัษฎาพร ทรัพย์สมบูรณ์...

ปีการศึกษา.....2549.....

4782208526 : MAJOR BUSINESS SOFTWARE DEVELOPMENT

KEY WORD: REFACTORING / SOFTWARE QUALITY / OBJECT-ORIENTED METRICS

SIRATHAN SUPPATANARAT : A RELATIONSHIP BETWEEN REFACTORING AND SOFTWARE QUALITY USING OBJECT-ORIENTED METRICS. THESIS ADVISOR : ASST. PROF. ASSADAPORN SAPSOMBOON, 211 pp.

The objective of this research is (1) to compare the software quality of source code before refactoring and source code after refactoring; (2) to compare the software quality of source code under one refactoring method and source code under two refactoring methods; (3) to compare the software quality of source code under two refactoring methods and source code under the same two refactoring methods in different order. The software quality in focus is maintainability, reusability and understandability through 6 Chidamber and Kemerer's metrics (1994), 1 Lorenz and Kidd's metric (1994), 6 MOOD metrics by Abreu (1996) and 1 CommentPercentage metric (Rosenberg and Hyatt, 1995).

This research is an experimental research. Subjects in the study are the assigned projects to undergraduate students in Object-Oriented Programming Foundation class. The subjects are all business applications with at least 5 classes and developed by C# language. There were 32 sample units.

The results indicate that one or two refactoring methods used can improve software quality. However, the order of applying refactoring methods shows no effect to the software quality.

Department.....Statistics..... Student's Signature.....

Field of Study ...Business Software Development... Advisor's Signature.....

Academic Year .....2006.....

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. อัษฎาพร ทรัพย์สมบูรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้การชี้แนะแนวทางต่างๆ ให้กับผู้วิจัยจนสำเร็จเป็นวิทยานิพนธ์ฉบับนี้ และขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. ถาวร อานุภาพไตรรงค์ ประธานกรรมการวิทยานิพนธ์ อาจารย์ ดร. วัชรา จันทาทับ กรรมการวิทยานิพนธ์ ที่ช่วยชี้แนะสิ่งต่างๆ และอาจารย์ทุกท่านที่ให้ความรู้และอบรมสิ่งต่างๆ ให้กับผู้วิจัย และขอขอบคุณโครงงานจากหน่วยทดลองในวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ทุกท่านที่ช่วยสนับสนุนการทดลองของผู้วิจัย

ที่สำคัญที่สุดขอขอบพระคุณคุณพ่อ คุณแม่ที่ให้การสนับสนุนในด้านการศึกษาตลอดมา สุดท้ายขอขอบคุณเพื่อนๆ ทุกคนที่ช่วยเหลือ และนางสาวอุมาพร นิลเอวะที่คอยช่วยเหลือและให้คำปรึกษาที่ดีตลอดมา

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

หน้า

สารบัญภาพ

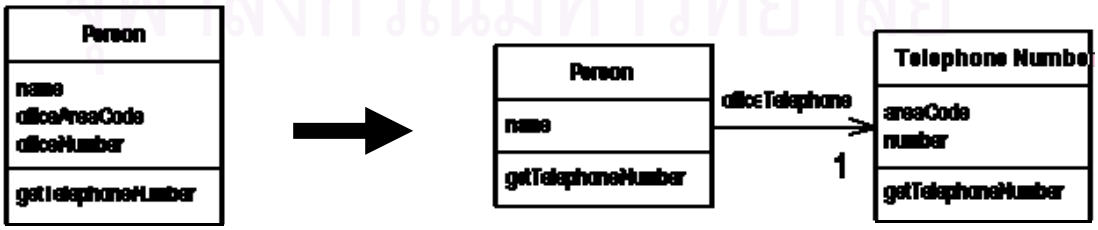
หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเปลี่ยนแปลงของซอฟต์แวร์ (Software Change) เกิดขึ้นได้เสมอหลังจากการส่งมอบซอฟต์แวร์ (Software) ไปแล้ว ทำให้กระบวนการพัฒนาซอฟต์แวร์ (Software Development) จำเป็นต้องมีการตอบสนองการเปลี่ยนแปลงดังกล่าว ทั้งนี้การแก้ไขหรือเปลี่ยนแปลงตามความต้องการ (Requirement) ใหม่ๆ กับระบบ นำไปสู่กระบวนการที่เรียกว่า “การบำรุงรักษาซอฟต์แวร์ (Software Maintenance)” การเปลี่ยนแปลงที่เกิดขึ้นไม่ว่าจะเป็นการเพิ่ม ลด และแก้ไขความต้องการนั้น สิ่งที่เป็นผลตามมาก็คือ การทำให้ความยาวของซอร์สโค้ด (Source Code) เพิ่มมากขึ้น ซอร์สโค้ดมีความซับซ้อน (Complexity) มากขึ้นและซอฟต์แวร์มีการเบี่ยงเบนออกจากการออกแบบดั้งเดิม (Sommerville, 2001) ดังนั้นแล้วการรับมือกับปัญหาที่เกิดขึ้นดังกล่าวเป็นสิ่งสำคัญอย่างยิ่ง จึงทำให้เกิดวิธีการที่ต้องการลดความซับซ้อนของซอฟต์แวร์และปรับปรุงคุณภาพ (Quality) โครงสร้างภายในของซอฟต์แวร์ขึ้น เรียกว่า “กระบวนการรีแฟคทอริง (Refactoring)”

กระบวนการรีแฟคทอริงคือกระบวนการปรับปรุงซอร์สโค้ดที่มีอยู่ให้ดีขึ้น โดยการเปลี่ยนแปลงเฉพาะโครงสร้างภายใน ซึ่งการเปลี่ยนแปลงจะต้องไม่ส่งผลกระทบกับโครงสร้างภายนอก (Fowler, 1999) หรือกระบวนการรีแฟคทอริงเป็นกระบวนการที่เปลี่ยนแปลงซอร์สโค้ดในรูปแบบเชิงวัตถุ (Object-Oriented) โดยที่ไม่ส่งผลกระทบให้เกิดการเปลี่ยนแปลงกับพฤติกรรมภายนอก (External Behavior) ของซอร์สโค้ดแต่มุ่งเน้นการปรับปรุงโครงสร้างภายในซอร์สโค้ดเท่านั้น (Opdyke, 1992) ตัวอย่างเช่น การทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรคคลาส (Extract Class) ซึ่งเป็นวิธีการแยกคลาส (Class) ออกเป็น 2 คลาส โดยการแยกคลาสดังกล่าวจะกระทำกับคลาสที่มีหน้าที่การทำงานมากเกินไปก่อให้เกิดความยุ่งยากและสับสนในตัวคลาส ซึ่งการแยกคลาสออกจะทำให้คลาสที่ได้มีความเหมาะสมมากขึ้นและง่ายต่อการนำไปใช้



รูปที่ 1-1 แสดงซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงด้วยวิธีเอ็กแทรคคลาส

การทำกระบวนการรีแฟคทอริงช่วยให้การพัฒนาซอฟต์แวร์ประหยัดเวลา (Time) แรงงาน (Effort) อีกทั้งยังเพิ่มความสามารถการนำกลับมาใช้ใหม่ (Reusability) ทำซอร์สโค้ดที่ได้สามารถนำไปใช้กับงานที่เกี่ยวข้องได้โดยปราศจากการเพิ่มหรือแก้ไขส่วนประกอบและขอบเขตของซอร์สโค้ดที่เป็นอยู่ (Neill และ Gill, 2003) และความสามารถการพัฒนาต่อ (Extendability) โดยการกระบวนการรีแฟคทอริงจะไม่ส่งผลกระทบต่อส่วนต่อประสาน (Interface) ทำให้ซอฟต์แวร์ที่ได้จะสามารถใช้งานร่วมกับส่วนต่อประสานของซอฟต์แวร์เดิมที่เคยเชื่อมต่ออยู่ได้โดยไม่เกิดปัญหา (Mancl, 2001) มีงานวิจัยหลายงานวิจัยได้นำกระบวนการรีแฟคทอริงมาวิจัย เช่น การระบุว่าจุดใดของในซอร์สโค้ดควรจัดการด้วยกระบวนการรีแฟคทอริงหรือเรียกว่าการระบุ “ร่องรอยไม่ดีในซอร์สโค้ดนั้น” (Bad smells in source code) Beck (1999) ได้กล่าวว่า “ร่องรอยไม่ดีในซอร์สโค้ด” คือโครงสร้างภายในของซอร์สโค้ดที่ไม่เหมาะสมและส่งผลไม่ดีต่อประสิทธิภาพของซอฟต์แวร์ ควรจัดการด้วยกระบวนการรีแฟคทอริง ซึ่งต่อมา Fowler (1999) ได้นำมาตีพิมพ์ในหนังสือกระบวนการรีแฟคทอริง ทำให้มีงานวิจัยที่กล่าวถึงการวิจัยเกี่ยวกับเครื่องมือที่ช่วยในการตรวจหาร่องรอยไม่ดีในซอร์สโค้ด เช่น Kataoka (2001) ได้สร้างเครื่องมือชื่อ ไดคอน (Daikon Tool) เพื่อระบุตำแหน่งที่ควรทำกระบวนการรีแฟคทอริงได้อย่างอัตโนมัติ (Automate) ทั้งนี้ยังมีงานวิจัยที่กล่าวเชื่อมโยงถึงร่องรอยไม่ดีในซอร์สโค้ดอีกคือ Tourwé และ Mens (2003) ได้วิจัยเกี่ยวกับการจัดการตรวจหาตำแหน่งที่ต้องการทำกระบวนการรีแฟคทอริงแบบกึ่งอัตโนมัติและนำเสนอว่าควรใช้กระบวนการรีแฟคทอริงวิธีใดจัดการกับซอร์สโค้ดที่เกิดปัญหานั้น ยิ่งไปกว่านั้น Dudziak และ Wloka (2002) ได้นำเสนอเครื่องมือที่ช่วยในการตัดสินใจว่าจุดอ่อนในซอร์สโค้ดอยู่ตำแหน่งใด โดยใช้หลักการจากกระบวนการรีแฟคทอริงแล้ววิเคราะห์หาส่วนที่เกิดปัญหา นอกจากนั้น Emden และ Moonen (2002) ได้รวบรวมวิธีในการตรวจหาร่องรอยไม่ดีในซอร์สโค้ดแล้วสร้างเป็นเครื่องมือที่สามารถทำให้เห็นภาพได้ (Visualization) เพื่อความสะดวกในการมองปัญหาที่เกิดขึ้นในโครงสร้างของซอร์สโค้ดโดยใช้ภาษาจาวา (Java) เป็นภาษาในการพัฒนา

ยังมีงานวิจัยที่ยืนยันว่ากระบวนการรีแฟคทอริงที่นำมาปฏิบัตินั้น จะไม่ทำให้พฤติกรรม (Behavior) ของซอฟต์แวร์เปลี่ยนแปลงไป เช่น การให้คำจำกัดความของการรักษาพฤติกรรมของซอฟต์แวร์ไว้โดย Opdyke (1992) ให้ความหมายว่าค่าอินพุต (Input) และค่าเอาท์พุต (Output) จะต้องเหมือนกันทั้งก่อนและหลังทำกระบวนการรีแฟคทอริง โดยแนวทางที่นำเรื่องการรักษาพฤติกรรมของซอฟต์แวร์ไปปฏิบัติคือการนำไปสร้างข้อบังคับในการทดสอบ โดยนำมาใช้ในการสร้างกรณีการทดสอบซึ่งใช้ตรวจสอบผลลัพธ์ของซอฟต์แวร์ก่อนและหลังใช้กระบวนการรีแฟคทอริง (Pipka, 2002) อีกทั้งยังมีงานวิจัยที่กล่าวถึงการพิสูจน์คุณสมบัติข้อนี้ของกระบวนการรีแฟคทอริง ได้แก่ งานวิจัยของ Proeitti และ Pettorossi (1991) ได้พิสูจน์ว่าภาษาโปรลอค (Prolog)

เป็นภาษาทางคอมพิวเตอร์หนึ่งที่สามารถพิสูจน์ได้ว่ากระบวนการรีแฟคทอริงไม่ทำให้พฤติกรรมของซอฟต์แวร์เปลี่ยนแปลงไป และงานวิจัยของ Tokuda และ Batory (1999) ได้พิสูจน์ว่าภาษาที่ซับซ้อนอย่างซีพลัสพลัส (C + +) สามารถนำกระบวนการรีแฟคทอริงมาพัฒนาเป็นเครื่องมือที่ใช้ช่วยพัฒนาซอฟต์แวร์ได้ แต่ยังมีข้อจำกัดในส่วนของคอมไพเลอร์ (Compiler) อยู่

อย่างไรก็ตามนอกจากงานวิจัยที่กล่าวข้างต้นแล้วยังมีงานวิจัยที่ชี้ให้เห็นถึงประเด็นคุณภาพของซอฟต์แวร์กับการทำกระบวนการรีแฟคทอริงอีกด้วย เช่น งานวิจัยของ Demeyer (2002) ได้วิจัยหาผลกระทบของกระบวนการรีแฟคทอริงที่มีต่อความสามารถการมีภาวะพหุสัณฐาน (Polymorphism) โดยได้สรุปว่าซอฟต์แวร์จะมีประสิทธิภาพเพิ่มมากขึ้นหลังจากทำกระบวนการรีแฟคทอริง เนื่องจากประสิทธิภาพที่เกิดจากคอมไพเลอร์ในปัจจุบันสามารถเข้ากันได้ดีกับวิธีของภาวะพหุสัณฐานดังกล่าว อีกทั้งได้มีงานวิจัยที่กล่าวถึงการวัดหรือการประเมิน ผลกระทบจากการทำกระบวนการรีแฟคทอริงด้วยการวัดและมาตรวัดซอฟต์แวร์ (Software Measurement and Metrics) ซึ่งการวัดซอฟต์แวร์เป็นเครื่องมือสำหรับวัดผลคุณภาพทางซอฟต์แวร์ได้เป็นอย่างดี สำหรับการวัดคุณภาพซอฟต์แวร์หลักการที่นำมาใช้คือการใช้มาตรวัด (Metric) เพื่อเป็นตัวชี้วัดถึงคุณภาพของซอฟต์แวร์แต่ละด้านทั้งนี้สามารถวัดได้โดยตรง (Direct) จากคุณภาพภายใน (Internal Quality Attribute) เช่น คุณภาพภายในของซอฟต์แวร์เชิงวัตถุคือ การเชื่อมต่อ (Coupling) และการทำงานร่วมกัน (Cohesion) แล้วนำค่าที่ได้จากการวัดดังกล่าวไปประเมินเพื่อนำไปบ่งบอกถึงคุณภาพภายนอก (External Quality Attribute) เช่น ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการพัฒนาต่อ (Extendability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability)และความสามารถในการทำความเข้าใจ (Understandability) เป็นต้น โดย Kataoka และคณะ (2002) ได้นำเสนอมาตรวัดคุณภาพภายในเรื่องการเชื่อมต่อไว้และได้ประเมินผลกระทบของกระบวนการรีแฟคทอริงกับความสามารถในการบำรุงรักษาของซอฟต์แวร์ไว้ หรืองานวิจัยของ Tahvildari และ Kotogiannis (2002) ได้เปลี่ยนข้อมูลที่สามารถวัดได้ไปเป็นคุณภาพซอฟต์แวร์ด้านต่างๆ ซึ่งถูกเรียกว่า “ซอฟต์ – โกล กราฟ” (Soft – goal graphs) โดยซอฟต์ – โกล กราฟนี้ได้อธิบายถึงโครงสร้างที่เชื่อมต่อความสัมพันธ์ของคุณภาพความสามารถในการบำรุงรักษาโดยเชื่อมโยงจากคุณภาพซอฟต์แวร์ภายในในด้านต่างๆ แล้วแยกกระบวนการรีแฟคทอริงออกเป็นหมวดหมู่แล้วเชื่อมโยงความเป็นไปได้ที่กระทบต่อคุณภาพภายในต่างๆ เข้ากับซอฟต์ – โกล กราฟ ต่อมา Tahvildari และ Kotogiannis (2003) ได้ใช้มาตรวัดเชิงวัตถุ (Object-Oriented Metrics) เป็นตัวชี้วัดเพื่อวัดคุณภาพของซอฟต์แวร์ที่ถูกปรับปรุงขึ้นโดยใช้กระบวนการรีแฟคทอริงที่จัดหมวดหมู่ไว้

จะเห็นได้ว่ากระบวนการรีแฟคทอริงได้ถูกนำมาวิจัยในแง่มุมหรือประเด็นต่างๆ มากมาย ทั้งนี้ยังคงมีประเด็นที่สำคัญและเกี่ยวข้องกับกระบวนการรีแฟคทอริง โดยการพัฒนาซอฟต์แวร์ในปัจจุบันไม่สามารถปฏิเสธเรื่องคุณภาพของซอฟต์แวร์ได้อีกต่อไป เนื่องจากคุณภาพซอฟต์แวร์เป็นสิ่งสำคัญเพราะสามารถช่วยลดค่าใช้จ่ายของการบำรุงรักษา (Maintenance) การทดสอบ (Testing) และการนำซอฟต์แวร์กลับไปใช้ใหม่ (Reuse) ทั้งหมดนี้เป็นผลจากการพัฒนาซอฟต์แวร์ที่มีคุณภาพ (Khosravi และ Guéhéneuc, 2004) ทำให้มีงานวิจัยที่กล่าวบ่งชี้ถึงผลกระทบของกระบวนการรีแฟคทอริงที่มีต่อคุณภาพของซอฟต์แวร์ ซึ่งโดยปกติแล้วการวัดคุณภาพซอฟต์แวร์เชิงวัตถุสามารถทำได้โดยใช้มาตรวัดเชิงวัตถุ (Object-Oriented Metrics) ซึ่งมีความแตกต่างจากมาตรวัดทั่วไป เนื่องจากมาตรวัดซอฟต์แวร์เชิงวัตถุจำเป็นต้องออกแบบมาตรวัดให้เข้ากับคุณสมบัติของซอฟต์แวร์เชิงวัตถุ (Lorenz และ Kidd, 1994) งานวิจัยมากมายได้นำเสนอมาตรวัดสำหรับซอฟต์แวร์เชิงวัตถุ ซึ่งมาตรวัดเหล่านี้สามารถนำมาใช้วัดคุณภาพผลิตผลของซอฟต์แวร์เชิงวัตถุได้แก่ งานวิจัยของ Dagpinar และ Jahnke (2003) ได้นำมาตรวัดเชิงวัตถุมาใช้เพื่อทำนายความสามารถการบำรุงรักษาของซอฟต์แวร์ โดยผลการทดลองของงานวิจัยได้ชี้ให้เห็นว่ามาตรวัดขนาด (Size Metric) และมาตรวัดการเชื่อมต่อ (Coupling Metric) สามารถทำนายความสามารถการบำรุงรักษาแบบมีนัยสำคัญ อีกทั้งยังมีงานวิจัยที่รวบรวมแล้วสรุปมาตรวัดเชิงวัตถุเป็นมาตรฐานและนำเสนอแนวทางปฏิบัติไว้ได้แก่ งานวิจัยของ Herrison Counsell และ Nithi (1997) ได้นำเสนอมาตรวัดเชิงวัตถุไว้ 3 กลุ่มได้แก่ มาตรวัดเชิงวัตถุของ ของ Chidamber และ Kemerer (1994) มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) และมาตรวัดเชิงวัตถุของ Abreu (1996) โดยงานวิจัยดังกล่าวได้สรุปปัญหาที่เกี่ยวข้องกันจากมาตรวัดทั้ง 3 กลุ่มนี้ และได้นำเสนอแนวทางการนำมาตรวัดเชิงวัตถุทั้ง 3 กลุ่มไปใช้ อีกทั้งได้มีงานวิจัยที่กล่าวสนับสนุนถึงการนำมาตรวัดเชิงวัตถุไปปฏิบัติใช้ในอุตสาหกรรมซอฟต์แวร์ ทั้งนี้ได้บอกถึงประโยชน์ที่ช่วยให้สามารถบอกปริมาณของงานเพื่อใช้ประมาณแรงงาน (Effort) ในอนาคต ช่วยตรวจสอบนโยบายในการพัฒนาซอฟต์แวร์ ช่วยจัดลำดับความสำคัญของการตรวจสอบและช่วยประเมินคุณภาพของซอฟต์แวร์โดยรวม (Schroeder, 1999)

ด้วยเหตุนี้ทางผู้วิจัยได้สนใจที่จะนำประเด็นของกระบวนการรีแฟคทอริงและคุณภาพซอฟต์แวร์มาศึกษา เนื่องจากกระบวนการรีแฟคทอริงเป็นกระบวนการที่ช่วยปรับปรุงโครงสร้างภายในซอฟต์แวร์เชิงวัตถุ ดังนั้นการทำกระบวนการรีแฟคทอริงแต่ละวิธีย่อมส่งผลกระทบต่อคุณภาพซอฟต์แวร์ ดังเช่น การส่งผลต่อคุณภาพภายในของซอฟต์แวร์เชิงวัตถุ มีงานวิจัยพบว่าการทำกระบวนการรีแฟคทอริงได้ช่วยปรับปรุงคุณลักษณะการเชื่อมต่อและการทำงานร่วมกันให้ดีขึ้น (Bois, Demeyer และ Verelst, 2004) อีกทั้งกระบวนการรีแฟคทอริงยังส่งผลต่อคุณภาพภายนอกของซอฟต์แวร์เชิงวัตถุอีกด้วย เช่น การนำกระบวนการรีแฟคทอริงมาใช้เพื่อช่วยเพิ่มความสามารถ

ในการนำกลับมาใช้ใหม่ (Reusability) ของซอร์สโค้ดในอนาคต (Neill และ Gill, 2003) ดังนั้นจึงมีความเป็นไปได้ที่ว่ากระบวนการรีแฟคทอริงแต่ละวิธีจะส่งผลต่อระดับคุณภาพซอฟต์แวร์ในระดับที่แตกต่างกันและมีความเป็นไปได้ว่ากระบวนการรีแฟคทอริงที่นำมาใช้ร่วมกันอาจส่งผลต่อคุณภาพของซอฟต์แวร์ที่แตกต่างกันไป โดยได้มีงานวิจัยที่บ่งชี้แล้วว่ากระบวนการรีแฟคทอริงบางคู่ เช่น กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (Move Method) กับ กระบวนการรีแฟคทอริงวิธีเอนแคบซูเลชันแวริเอเบิล (Encapsulation Variable) ฯลฯ ที่ทำงานร่วมกันแล้วเกิดความขัดแย้งกัน ซึ่งงานวิจัยนี้ได้นำเสนอลำดับการทำกระบวนรีแฟคทอริงที่เหมาะสมในแต่ละคู่ไว้ (Mens และคณะ , 2004) แต่งานวิจัยดังกล่าวไม่ได้กล่าวบ่งชี้ในเชิงคุณภาพซอฟต์แวร์เพียงแต่ชี้ให้เห็นถึงลำดับในการทำกระบวนการรีแฟคทอริงแต่ละวิธีที่มีผลขัดแย้งกันหรือไม่เมื่อนำมาทำงานร่วมกันเป็นคู่

ดังนั้นงานวิจัยนี้จึงสนใจที่จะนำกระบวนการรีแฟคทอริงในแต่ละวิธีมาหาคุณภาพซอฟต์แวร์โดยใช้มาตรวัดคุณภาพซอฟต์แวร์เชิงวัตถุ เพื่อบอกคุณภาพซอฟต์แวร์ที่เปลี่ยนแปลงไปจากเดิมและการนำกระบวนการรีแฟคทอริงมาวิเคราะห์หาความสัมพันธ์ว่าการทำงานร่วมกันในแต่ละคู่ของกระบวนการรีแฟคทอริงนั้นส่งผลต่อคุณภาพซอฟต์แวร์ในเชิงบวกหรือเชิงลบต่อกัน รวมทั้งการนำกระบวนการรีแฟคทอริงมาหาว่าลำดับในการทำกระบวนการรีแฟคทอริงนั้นส่งผลต่อคุณภาพซอฟต์แวร์หรือไม่ โดยเลือกกระบวนการรีแฟคทอริงจากกลุ่มของวิธีที่ถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ในปัจจุบันและเลือกมาตรวัดวัดคุณภาพซอฟต์แวร์เชิงวัตถุจากกลุ่มของมาตรวัดเชิงวัตถุที่เผยแพร่ในปัจจุบัน แล้วนำผลที่คำนวณได้แต่ละมาตรวัดเชิงวัตถุมาวิเคราะห์เพื่อบอกคุณภาพซอฟต์แวร์ที่เปลี่ยนไป

1.2 วัตถุประสงค์งานวิจัย

1.2.1 เพื่อเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงและหลังทำกระบวนการรีแฟคทอริงแต่ละวิธี

1.2.2 เพื่อเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี

1.2.3 เพื่อเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน

1.3 ขอบเขตงานวิจัย

1.3.1 กระบวนการรีแฟคทอริงแต่ละวิธีที่นำมาใช้ในงานวิจัยอยู่ภายใต้เครื่องมือที่เรียกว่า “รีชาร์ปเปอร์ 2.0 (ReSharper 2.0)”

1.3.2 มาตรวัดคุณภาพซอฟต์แวร์เชิงวัตถุที่นำมาใช้วัดในงานวิจัยประกอบด้วยมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) มาตรวัดเชิงวัตถุของ Abreu (1996) และมาตรวัดเชิงวัตถุอื่นๆ

1.3.3 ซอร์สโค้ดที่นำมาใช้ในงานวิจัยจำกัดเพียงงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิต ที่เรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) โดยระบบจากงานที่ได้รับมอบหมายจะเป็นระบบที่มีความหลากหลายในเชิงธุรกิจและจะต้องมีจำนวนคลาสภายในระบบตั้งแต่ 5 คลาสขึ้นไป

1.3.4 เครื่องมือที่พัฒนาสามารถทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ที่ใช้ระบบปฏิบัติการวินโดวส์ (Microsoft Windows)

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษากระบวนการรีแฟคทอริง

1.4.2 ศึกษามาตรวัดคุณภาพสำหรับซอฟต์แวร์ทั่วไปและสำหรับซอฟต์แวร์เชิงวัตถุ

1.4.3 ศึกษาวิธีการ เทคนิคและเครื่องมือในรูปแบบต่างๆ ที่นำกระบวนการรีแฟคทอริงไปใช้

1.4.4 ศึกษาวิธีการ เทคนิคและเครื่องมือในรูปแบบต่างๆ สำหรับมาตรวัดคุณภาพที่ใช้ในการทดสอบ

1.4.5 ออกแบบและพัฒนาเครื่องมือด้วยภาษาซีชาร์ป (C #) เพื่อวิเคราะห์และคำนวณค่ามาตรวัดซอฟต์แวร์เชิงวัตถุ

1.4.6 ออกแบบการทดลองเพื่อตรวจสอบการวิเคราะห์ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริง

1.4.7 ทำการทดลองเครื่องมือที่พัฒนาขึ้นกับหน่วยทดลอง

1.4.8 เก็บรวบรวมข้อมูลที่ได้จากการทดลอง

1.4.9 วิเคราะห์ข้อมูลและสรุปผลการทดลอง

1.4.10 จัดทำเอกสารรายงานผลการทดลองและเอกสารงานวิจัยต่างๆ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถบอกคุณภาพซอฟต์แวร์ที่เปลี่ยนแปลงไปจากเดิมของกระบวนการรีแฟคทอริงในแต่ละวิธีได้

1.5.2 สามารถบอกคุณภาพซอฟต์แวร์ที่เปลี่ยนแปลงไปจากเดิมของกระบวนการรีแฟคทอริงในลำดับที่แตกต่างได้

1.5.3 สามารถนำผลสรุปที่ได้ไปสร้างแบบแผน (Pattern) ในการปรับปรุงซอฟต์แวร์ด้วยการใช้กระบวนการรีแฟคทอริงโดยไม่ให้เกิดความขัดแย้งกันด้านคุณภาพ

1.5.4 ได้มาตรวัดการคำนวณและการวิเคราะห์ที่เหมาะสมต่อกระบวนการรีแฟคทอริง

1.5.5 สามารถนำผลการทดลองที่ได้มาเลือกกระบวนการรีแฟคทอริงที่เหมาะสมสำหรับนำไปสอนกลุ่มตัวอย่างซึ่งเป็นนักศึกษาชั้นปีที่ 2 ได้

1.6 นิยามคำศัพท์

1.6.1 การเชิงวัตถุ (Object-Oriented) คือสิ่งที่เป็นอยู่ของวัตถุประกอบไปด้วย 2 สิ่งต่อไปนี้ คือ คุณลักษณะ(Attributes) และพฤติกรรม (Behavior) โดยวัตถุจะประกอบไปด้วยคุณสมบัติของซอฟต์แวร์เชิงวัตถุดังนี้ (1) ความสามารถในการปกปิดหรือซ่อนเร้น (Encapsulation) (2) ความสามารถการสืบทอด (Inheritance) (3) ความสามารถภาวะพหุสัณฐาน (Polymorphism) และ (4) ความสามารถในการรวมกันขององค์ประกอบ (Composition)

1.6.2 การบำรุงรักษาซอฟต์แวร์ (Software Maintenance) คือกระบวนการเปลี่ยนแปลงซอฟต์แวร์ที่ผิดพลาดให้ถูกต้องตรงตามความต้องการ ภายหลังจากส่งมอบซอฟต์แวร์ไปแล้ว

1.6.3 การวัดและมาตรวัดซอฟต์แวร์ (Software Measurement and Metrics) คือเงื่อนไขที่ใช้ในการตรวจสอบหรือวัดผลซอฟต์แวร์ โดยใช้มาตรวัดเป็นตัวชี้วัดคุณภาพซอฟต์แวร์

1.6.4 ความต้องการ (Requirement) คือความต้องการของผู้ใช้ไม่ว่าจะเป็นความต้องการด้านฟังก์ชันของระบบ (Functional Requirement)และความต้องการที่ไม่ใช่ฟังก์ชันของระบบ (Non-Functional Requirement)

1.6.5 การเชื่อมต่อ (Coupling) คือปริมาณของจำนวนการเชื่อมต่อกันระหว่างวัตถุ (Object) แต่ละวัตถุ

1.6.6 การทำงานร่วมกัน (Cohesion) คือระดับของความเป็นเอกภาพของวัตถุ โดยบ่งบอกถึงปริมาณหน้าที่การทำงานของแต่ละวัตถุ

1.6.7 ความซับซ้อน (Complexity) คือความซับซ้อน โดยในที่นี้อ้างอิงถึงความซับซ้อนของซอฟต์แวร์

1.6.8 คุณลักษณะภายในของคุณภาพ (Internal Quality Attribute) คือคุณภาพที่สามารถวัดได้โดยตรงจากซอฟต์แวร์ เช่น ขนาด (Size) ความซับซ้อน (Complexity) และจำนวนคำอธิบาย (Number of Comments)

1.6.9 คุณลักษณะภายนอกของคุณภาพ (External Quality Attribute) คือคุณภาพในแต่ละด้านของซอฟต์แวร์ เช่น ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการขยายต่อ (Expandability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability)

1.6.10 ซอร์สโค้ด (Source Code) คือรหัสคอมพิวเตอร์ซึ่งเปลี่ยนเป็นภาษาทางเครื่องคอมพิวเตอร์ก่อนทำงานบนเครื่องคอมพิวเตอร์

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 บทนำ

ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์ โดยใช้มาตรวัดเชิงวัตถุ มีทฤษฎีที่เกี่ยวข้องในงานวิจัยได้แก่ กระบวนการรีแฟคทอริง (Refactoring) คุณภาพซอฟต์แวร์ (Software Quality) และมาตรวัดเชิงวัตถุ (Object-Oriented Metrics) จากนั้นจะเป็นงานวิจัยที่เกี่ยวข้องกับกระบวนการรีแฟคทอริง คุณภาพซอฟต์แวร์และมาตรวัดเชิงวัตถุ

2.2 กระบวนการรีแฟคทอริง (Refactoring)

กระบวนการรีแฟคทอริง (Refactoring) คือกระบวนการปรับปรุงโครงสร้างของซอฟต์แวร์ โดยการปรับปรุงจะทำเฉพาะโครงสร้างภายในแต่คงไว้ซึ่งโครงสร้างหรือพฤติกรรม (Behavior) ภายนอก ทั้งนี้การปรับปรุงด้วยกระบวนการรีแฟคทอริงนี้จะกระทำกับซอร์สโค้ด (Source Code) ที่มีอยู่แล้ว ดังนั้นหากการเปลี่ยนแปลงแล้วมีผลกระทบต่อปัจจัยภายนอกอื่น ซึ่งปัจจัยภายนอกนี้เองได้ติดต่อเรียกใช้โครงสร้างภายในโดยผ่านทางส่วนต่อประสาน (Interface) ที่กำหนดไว้ ถ้าหากช่องทางการติดต่อ มีการเปลี่ยนแปลงก็จะส่งผลกระทบต่อการทำงานของปัจจัยภายนอกอื่นๆ ด้วย กระบวนการรีแฟคทอริงช่วยปรับปรุงโครงสร้างซอร์สโค้ดเสียใหม่ เพื่อให้ซอร์สโค้ดเหล่านี้สามารถถูกบำรุงรักษาและพัฒนาต่อได้ง่าย เนื่องจากความต้องการทางด้านธุรกิจเปลี่ยนแปลงไปแบบไม่หยุดยั้ง หน้าที่ของนักพัฒนาซอฟต์แวร์ที่ดีคือ การทำให้ซอฟต์แวร์สามารถปรับเปลี่ยนโดยง่ายด้วยเช่นกัน (Fowler, 1999)

ประเด็นสำคัญของกระบวนการรีแฟคทอริงประกอบไปด้วย (1) การทำให้ซอร์สโค้ดมีขนาดเล็กลงและกระชับมากขึ้น เพื่อให้ผู้พัฒนาเข้าใจซอร์สโค้ดได้ง่ายขึ้นและง่ายต่อการนำไปพัฒนาต่อ (2) เมื่อทำกระบวนการรีแฟคทอริงจะไม่มีผลกระทบต่อปัจจัยภายนอกที่เรียกใช้งานอยู่ นั่นคือการปรับปรุงซอร์สโค้ดด้วยกระบวนการรีแฟคทอริงนี้จะไม่ทำให้โครงสร้างภายนอกเปลี่ยนแปลง

Beck (1999) ได้กล่าวเปรียบเทียบกระบวนการรีแฟคทอริงไว้ว่า “เมื่อนักพัฒนาได้เรียนรู้กระบวนการรีแฟคทอริง นักพัฒนาเปรียบเสมือนสวมหมวก 2 ใบในขณะที่พัฒนาซอฟต์แวร์” หมวก 2 ใบดังกล่าวมีความหมายดังนี้ (1) นักพัฒนาต้องพยายามเพิ่มฟังก์ชัน (Function) อยู่เสมอ เมื่อมีการเพิ่มฟังก์ชันเข้าไปในซอร์สโค้ดนักพัฒนาจะพุ่งความสนใจในส่วนของฟังก์ชันใหม่ เพื่อเพิ่มมูลค่าให้กับซอฟต์แวร์ที่มีต่อผู้ใช้ โดยอาจขาดความใส่ใจในความซับซ้อนที่เกิดขึ้น ความกระชับของ ซอร์สโค้ดและคุณภาพของซอฟต์แวร์ที่สูญเสียไป (2) นักพัฒนาต้องทำกระบวนการรีแฟคทอริงหรือพยายามปรับปรุงโครงสร้างซอฟต์แวร์ให้มีคุณภาพที่ดี เมื่อมีการเพิ่มฟังก์ชัน นักพัฒนาต้องพุ่งความสนใจไปที่การปรับปรุงคุณภาพซอฟต์แวร์ให้ดีที่สุด

ทั้ง 2 ข้อข้างต้น เป็นการแบ่งบทบาทอย่างชัดเจน ซึ่ง นักพัฒนาจะต้องแสดงบทบาททั้งคู่ เมื่อต้องการเพิ่มฟังก์ชันก็พยายามใช้ความคิดเพื่อเพิ่มฟังก์ชันที่มีประสิทธิภาพให้มากที่สุด เช่นเดียวกันก็ต้องตระหนักถึงว่าการเพิ่มฟังก์ชันนั้นกระทบกับซอร์สโค้ดที่มีอยู่เดิม แต่จำเป็นต้องมีการปรับปรุง การทำกระบวนการรีแฟคทอริงเป็นการเลือกทำ แต่การเลือกที่จะทำกระบวนการรีแฟคทอริงเมื่อใดนั้น Roberts (1999) ได้ให้กฎไว้ 3 ข้อ (The Rule of Three) ดังนี้

1. ทำกระบวนการรีแฟคทอริงเมื่อเพิ่มหรือแก้ไขฟังก์ชัน

โดยส่วนมากแล้วเมื่อซอฟต์แวร์จำเป็นต้องเพิ่มหรือแก้ไขฟังก์ชันให้กับซอฟต์แวร์ ช่วงเวลานี้จึงเหมาะสมกับการนำกระบวนการรีแฟคทอริงเข้าไปใช้เพราะจะช่วยให้นักพัฒนาเข้าใจในซอร์สโค้ดมากขึ้น เมื่อมีความต้องการที่จะเพิ่มหรือแก้ไขฟังก์ชันใดๆ ให้กับซอฟต์แวร์ นักพัฒนาจำเป็นต้องทำความเข้าใจกับซอร์สโค้ดที่ได้มาซึ่งจะเป็นซอร์สโค้ดที่ตนเองพัฒนาหรือไม่ก็ตาม ดังนั้นเมื่อซอร์สโค้ดที่ได้มามีความซับซ้อน ยากต่อการทำความเข้าใจ การทำกระบวนการรีแฟคทอริงจะช่วยปรับปรุงซอร์สโค้ดให้มีคุณภาพที่ดีขึ้น และเมื่อต้องการเพิ่มหรือแก้ไขฟังก์ชันในอนาคตก็สามารถทำได้รวดเร็วและง่ายยิ่งขึ้น

2. ทำกระบวนการรีแฟคทอริงเมื่อต้องการแก้ข้อผิดพลาดหรือบัก (Bug)

ในการแก้ไขข้อผิดพลาดหรือบักนั้น นักพัฒนาจำเป็นต้องเข้าใจซอร์สโค้ดที่แก้ไขเป็นอย่างดีเพื่อที่จะสามารถแก้ไขข้อผิดพลาดได้ถูกต้อง ดังนั้นแล้วการนำกระบวนการรีแฟคทอริงมาใช้ในขั้นตอนนี้จึงทำให้สามารถช่วยให้นักพัฒนาสามารถทำความเข้าใจกับซอร์สโค้ดได้มากขึ้นและทำให้แก้ไขข้อผิดพลาดได้รวดเร็วและถูกต้อง

3. ทำกระบวนการรีแฟคทอริงเสมือนการตรวจสอบซอร์สโค้ด (Code Review)

ในบางองค์กรที่พัฒนาซอฟต์แวร์มีขั้นตอนของการตรวจสอบซอร์สโค้ด การตรวจสอบซอร์สโค้ดเปรียบเสมือนการช่วยกระจายความรู้ให้กับทีมที่พัฒนา โดยการให้ทีมที่พัฒนาช่วยกันระดมความคิดเพื่อปรับปรุงซอร์สโค้ด ทั้งนี้การตรวจสอบซอร์สโค้ดทำเพื่อให้นักพัฒนาที่มีประสบการณ์มากได้กระจายความรู้ไปสู่นักพัฒนาที่มีประสบการณ์น้อยและยังช่วยให้เข้าใจซอฟต์แวร์ที่พัฒนาเพิ่มมากขึ้นด้วย การนำกระบวนการรีแฟคทอริงมาใช้สามารถทำควบคู่เสมือนเป็นการตรวจสอบซอร์สโค้ดไปในตัว ทั้งนี้กระบวนการรีแฟคทอริงช่วยให้การทำความเข้าใจซอร์สโค้ดนั้นง่ายขึ้น อ่านง่ายขึ้นและทำให้มองภาพการพัฒนาซอร์สโค้ดที่อ่านได้ง่ายมากขึ้น

กระบวนการรีแฟคทอริงของ Fowler มีทั้งหมด 6 ประเภท 72 วิธี (Fowler, 1999) โดยสามารถอธิบายได้ดังนี้

A. คอมโพสชิงเมท็อด (Composing Methods) คือปัญหาส่วนใหญ่มักเกิดมาจากเมท็อด (Method) เนื่องจาก ยาวเกินไป เนื้อหามากเกินไปหรือมีความซับซ้อนเกินไป อาจทำให้เกิดความสับสนและความผิดพลาดได้ โดยกระบวนการรีแฟคทอริงในประเภทคอมโพสชิงเมท็อด มีดังนี้

1. เอ็กแทรกเมท็อด (Extract Method) คือกระบวนการที่สร้างเมท็อดขึ้นใหม่แล้วนำซอร์สโค้ดเข้ามาแยกไว้ที่เมท็อดใหม่นี้ โดยจะนำซอร์สโค้ดที่มองแล้วว่ารวมกันอยู่มากเกินไปหรือมองแล้วว่าซอร์สโค้ดต้องการคำอธิบายมากขึ้นเพื่อเสริมความเข้าใจ

2. อินไลน์เมท็อด (Inline Method) คือการยุบรวมเมท็อดที่มีเนื้อหาซอร์สโค้ดในเมท็อดสั้น ไม่จำเป็นต้องสร้างขึ้นเป็นเมท็อด แต่ให้ยุบรวมเมท็อดไว้ในเมท็อดที่เรียกใช้เมท็อดนี้

3. อินไลน์เทมป์ (Inline Temp) คือการยุบรวมตัวแปรชั่วคราว (Temporary Variable) ที่ถูกสร้างขึ้นเพื่อใช้เพียงครั้งหรือ 2 ครั้ง ไม่จำเป็นต้องสร้างรวมตัวแปรชั่วคราวขึ้น แต่ให้ยุบรวมไว้กับนิพจน์ (Expression) ที่นำไปใช้

4. รีเพลสเทมป์วิธคิวรี (Replace Temp with Query) คือการนำส่วนซอร์สโค้ดที่มีเนื้อหาการประมวลผลแล้วนำผลลัพธ์จัดเก็บไว้ในรวมตัวแปรชั่วคราวแทนที่ด้วยเมท็อดที่สร้างขึ้นใหม่แล้วให้ซอร์สโค้ดส่วนที่ต้องการใช้เรียกใช้ผ่านเมท็อดแทน

5. อินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล (Introduce Explaining Variable) คือการนำส่วนนิพจน์ที่มีเนื้อหาการประมวลผลแล้วได้ผลลัพธ์ มาสร้างรวมตัวแปรชั่วคราวแล้วจัดเก็บผลลัพธ์ก่อนนำไปใช้ เพื่ออธิบายความหมายและบอกจุดประสงค์ของนิพจน์นั้นได้อย่างชัดเจน

6. สปลิทเทมโพรารีแวริเอเบิล (Split Temporary Variable) คือการแยกตัวแปรชั่วคราวตัวใดที่ถูกกำหนดค่ามากกว่า 1 ครั้ง แต่ตัวแปรชั่วคราวไม่ได้ถูกใช้ในนิพจน์การวนซ้ำ (Loop) โดยแยกออกเป็นหลายตัวแปรชั่วคราวเพื่อป้องกันความสับสน

7. รีมูฟแอสไซเมนต์ทูพารามิเตอร์ (Remove Assignment to Parameter) คือการแทนที่พารามิเตอร์ (Parameter) ที่ส่งไปเรียกใช้ในเมท็อดด้วยตัวแปรชั่วคราวที่สร้างขึ้นในเมท็อดเพื่อป้องกันผลกระทบกับซอร์สโค้ดส่วนอื่นๆ ที่นำพารามิเตอร์นี้ไปใช้

8. รีเพลสเมท็อดวิธเมท็อดอ็อบเจกต์ (Replace Method with Method Object) คือการแทนที่เมท็อดที่มีเนื้อหายาวเกินไป แต่ไม่สามารถใช้วิธีเอ็กแทรกเมท็อดได้ เนื่องจากในเมท็อดมีการใช้ตัวแปรเฉพาะที่ (Local Variable) โดยสามารถแทนที่ได้โดยเปลี่ยนจากเมท็อดไปเป็นรูปแบบของวัตถุ(Object)

9. ซับสติติวต์อัลกอริทึม (Substitute Algorithm) คือการจัดการให้โครงสร้างซอร์สโค้ดกระชับและอ่านเข้าใจได้ง่าย โดยการเปลี่ยนรูปแบบการเขียนซอร์สโค้ดด้วยรูปแบบการเขียนใหม่ๆ

กระบวนการรีแฟคทอริงในประเภทคอมโพสซิงเมท็อดถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **เอ็กแทรกเมท็อด**

สาเหตุ คือการมีซอร์สโค้ดที่มีเนื้อหามากเกินไปและซับซ้อนเกินไป หรือมีเนื้อหาที่ไม่ปะติดปะต่อกัน โดยมองเห็นว่าสามารถแบ่งแยกออกได้

วิธีการแก้ไข โดยนำซอร์สโค้ดที่แบ่งแยกออกเป็นส่วนๆ นั้น ให้จัดรวมกลุ่มเป็นวิธีการโดยตั้งชื่อให้สื่อถึงจุดประสงค์ของเมท็อด

ตัวอย่าง

```
void printOwing() {
        printBanner();

        //print details
        System.out.println ("name:         " + _name);
        System.out.println ("amount       " + amount);
}
```

รูปที่ 2-1 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
void printOwing() {
        printBanner();
        printDetails(amount);
}

void printDetails (double amount) {
        System.out.println ("name:         " + _name);
        System.out.println ("amount       " + amount);
}
```

รูปที่ 2-2 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีเอ็กแทรกเมท็อด

- **รีเพลสเทมป์วิธคิวรี**

สาเหตุ คือการมีซอร์สโค้ดที่มีเนื้อหาการประมวลผลแล้วได้ผลลัพธ์เก็บไว้ในตัวแปรชั่วคราว

วิธีการแก้ไข โดยเปลี่ยนจากตัวแปรชั่วคราวเป็นเมท็อดแล้วให้ซอร์สโค้ดที่ต้องการใช้เรียกใช้ผ่านเมท็อดแทน

ตัวอย่าง

```
double basePrice = _quantity * _itemPrice;
        if (basePrice > 1000)
                return basePrice * 0.95;
        else
                return basePrice * 0.98;
```

รูปที่ 2-3 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
if (basePrice() > 1000)
        return basePrice() * 0.95;
else
        return basePrice() * 0.98;
double basePrice() {
        return _quantity * _itemPrice;
}
```

รูปที่ 2-4 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีรีเพลสเทมป์วิธคิวรี

- **อินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล**

สาเหตุ คือมีนิพจน์ที่ยุ่งยาก ซับซ้อนมากเกินไปและทำให้การอ่านซอร์สโค้ดทำความเข้าใจได้ยาก

วิธีการแก้ไข โดยดึงนิพจน์ที่ซับซ้อนมาสร้างเป็นตัวแปรชั่วคราวเพื่อทำให้นิพจน์เดิมสามารถอ่านได้ง่ายขึ้น

ตัวอย่าง

```
if ( (platform.toUpperCase().indexOf("MAC") > -1) &&
        (browser.toUpperCase().indexOf("IE") > -1) &&
                wasInitialized() && resize > 0 )
{
        // do something
}
```

รูปที่ 2-5 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
final boolean isMacOs = platform.toUpperCase().indexOf("MAC") > -1;
final boolean isIEBrowser = browser.toUpperCase().indexOf("IE")  > -1;
final boolean wasResized  = resize > 0;
if (isMacOs && isIEBrowser && wasInitialized() && wasResized)
{
                    // do something
}
```

รูปที่ 2-6 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล

B. มูฟวิงฟีเจอร์บีทวีนอ็อบเจกต์ (Moving Features Between Object) คือการทำความเข้าใจกับการตัดสินใจในการออกแบบวัตถุในครั้งแรกอาจไม่ถูกต้อง ทำให้จำเป็นต้องมีการเคลื่อนย้ายเปลี่ยนแปลงวัตถุ โดยกระบวนการรีแฟคทอริงในประเภทมูฟวิงฟีเจอร์บีทวีนอ็อบเจกต์สามารถช่วยแก้ปัญหาเหล่านี้ได้ มีวิธีดังนี้

1. มูฟเมท็อด (Move Method) คือการเคลื่อนย้ายเมท็อดไปยังอีกคลาสหนึ่งที่มีปริมาณการใช้งานเมท็อดนี้มากกว่าคลาสเดิมที่เมท็อดนั้นอยู่

2. มูฟฟิลด์ (Move Field) คือการเคลื่อนย้ายเขตข้อมูล (Field) ไปยังอีกคลาสหนึ่งที่มีปริมาณการใช้งานเขตข้อมูลนี้มากกว่าคลาสเดิมที่เขตข้อมูลนั้นอยู่

3. เอ็กแทรกคลาส (Extract Class) คือการแบ่งแยกคลาสจากคลาสหนึ่งออกเป็นหลายคลาส เนื่องจากคลาสมีหน้าที่การทำงานมากเกินไปหรือมองเห็นได้ว่าควรที่ทำงานแยกเป็น 2 คลาส

4. อินไลน์คลาส (Inline Class) คือการเคลื่อนย้ายเขตข้อมูลและเมท็อดจากคลาสหนึ่งไปยังอีกคลาสหนึ่ง เนื่องจากคลาสนั้นไม่ได้ถูกเรียกใช้งานมากเท่าที่ควร การยุบรวมกันเป็นคลาสเดียวกันทำให้เรียกใช้งานได้สะดวกกว่า

5. ไฮด์ดีลิเกท (Hide Delegate) คือการสร้างเมท็อดที่ซ่อนการดีลิเกท (Delegate) ไว้เพื่อให้คลาสลูกข่าย (Client Class) ทำการเรียกดีลิเกทคลาสผ่านทางเมท็อดที่สร้างขึ้น เพื่อให้คลาสยังคงคุณสมบัติการห่อหุ้ม (Encapsulation) เอาไว้

6. รีมูฟมิดเดิลแมน (Remove Middle Man) คือการยุบเมท็อดที่สร้างขึ้นเพื่อใช้งานการดีลิเกท เนื่องจากเมท็อดมีการใช้งานมากเกินไป ทำให้ไม่สะดวกต่อการเรียกใช้จึงควรยุบแล้วให้เรียกใช้ดีลิเกทคลาสผ่านโดยตรง

7. อินโทรดิวซ์ฟอเรนจ์เมท็อด (Introduce Foreign Method) คือการสร้างเมท็อดขึ้นในคลาสลูกข่ายแล้วเพิ่มกรณีตัวอย่าง (Instance) ลงในเมท็อดที่สร้างขึ้น เนื่องจากคลาสแม่ข่าย (Server Class) ไม่สามารถเพิ่มเติมหรือแก้ไขคลาสได้

8. อินโทรดิวซ์โลคอลเอ็กเทนชัน (Introduce Local Extension) คือการสร้างคลาสขึ้นใหม่ในคลาสลูกข่าย โดยมีเมท็อดเพิ่มเติมในคลาสใหม่นี้เนื่องจากคลาสแม่ข่ายไม่สามารถเพิ่มเติมหรือแก้ไขคลาสได้

กระบวนการรีแฟคทอริงในประเภทมูฟวิงฟีเจอร์บีทวีนอ็อบเจกต์ถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **มูฟเมท็อด**

สาเหตุ คือการที่เมท็อดหนึ่งๆ ถูกใช้โดยคลาสอื่นมากกว่าที่จะถูกใช้จากคลาสตัวเองที่เป็นผู้ประกาศเมท็อดไว้

วิธีการแก้ไข คือการสร้างเมท็อดใหม่ภายในคลาสที่เรียกใช้เมท็อดนี้มากกว่า โดยเมท็อดที่สร้างใหม่นี้มีลักษณะเหมือนกับเมท็อดเดิมทุกประการ

ตัวอย่าง

```
Class Class_A{
public:
        static void methodA1 ( )
        {       attributeA1 = 0;
        }
}
```

```
Class Class_B{
public:
        static void methodB1 ( )
        {       class_A::attributeA1 = 0;
                class_A::methodA1 ( ); }
}
```

รูปที่ 2-7 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
Class Class_A{
public:
}
```

```
Class Class_B{
public:
        static void methodB1 ( )
        {       attributeA1 = 0;
                methodA1 ( ); }
        static void methodA1 ( )
        {       attributeA1 = 0;
                methodA1 ( ); }
}
```

รูปที่ 2-8 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีมูฟเมท็อด

- **มูฟฟิลด์**

สาเหตุ คือการที่เขตข้อมูลหนึ่งๆ ถูกใช้โดยคลาสอื่นมากกว่าที่จะถูกใช้จากคลาสตัวเองที่เป็นผู้ประกาศเขตข้อมูลไว้

วิธีการแก้ไข คือการสร้างเขตข้อมูลใหม่ภายในคลาสที่เรียกใช้เขตข้อมูลนี้มากกว่า โดย ฟิลด์ที่สร้างใหม่นี้มีลักษณะเหมือนกับเขตข้อมูลเดิมทุกประการ

ตัวอย่าง

```
Class Class_A{
public int _low;
public int _high;
        static void methodA1 ( )
        {       _low = 0;
        }
}
```

```
Class Class_B{
public int _medium;
public:
        static void methodB1 ( )
        {       _medium = 5;
                class_A::_low = 0;
                class_A:: _high = 10; }
}
```

รูปที่ 2-9 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
Class Class_A{
        static void methodA1 ( )
        {       classB::_low = 0;
        }
}
```

```
Class Class_B{
public int _medium;
public int _low;
public int _high;
public:
        static void methodB1 ( )
        {       _medium = 5;
                _low = 0;
                _high = 10; }
}
```

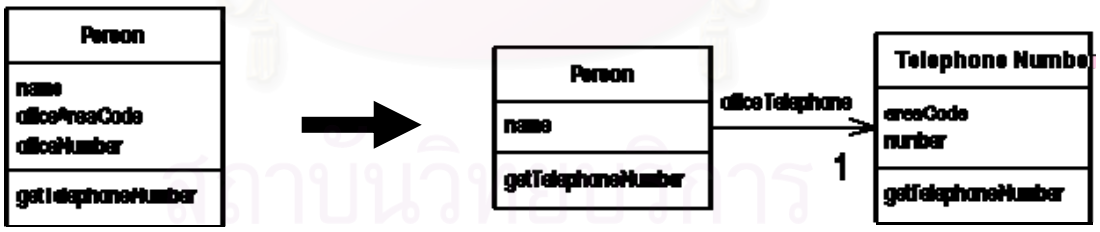
รูปที่ 2-10 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีมูฟฟิลด์

- **เอ็กแทรกคลาส**

สาเหตุ คือคลาสหนึ่งๆ มีหน้าที่รับผิดชอบมากเกินไป ซึ่งควรแยกออกเป็น 2 คลาส

วิธีการแก้ไข คือการสร้างคลาสใหม่แล้วเคลื่อนย้ายเขตข้อมูลและเมท็อดที่เกี่ยวข้องกับคลาสใหม่ทั้งหมด

ตัวอย่าง



รูปที่ 2-11 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีเอ็กแทรกคลาส

C. ออแกไนซิงดาต้า (Organizing Data) คือการจัดการกับข้อมูลที่อยู่ภายในคลาส โดยกระบวนการรีแฟคทอริงในประเภทออแกไนซ์ซิงดาต้า มีดังนี้

1. เซลฟ์เอนแคปซูเลทฟิลด์ (Self Encapsulate Field) คือการเปลี่ยนการจัดการข้อมูลที่สามารถเข้าถึงได้โดยตรง ให้เปลี่ยนเป็นการเข้าถึงโดยอ้อมโดยใช้วิธีการเกตติง (Getting) และเซตติง (Setting)

2. รีเพลสดาตาแวลู วิธอ็อบเจกต์ (Replace Data Value with Object) คือการสร้างวัตถุขึ้นเพื่อรองรับการเปลี่ยนแปลงที่เกิดกับข้อมูลหรือเมท็อดโดยใช้ความสัมพันธ์การประกอบด้วย (Composition) เชื่อมต่อกับคลาสเดิม

3. เชนจ์แวลูทูรีเฟอเรนซ์ (Change Value to Reference) คือการเปลี่ยนแปลงความสัมพันธ์ที่เชื่อมต่อกันระหว่างคลาส 2 คลาส โดยเปลี่ยนแปลงจากความสัมพันธ์การประกอบด้วยเป็นความสัมพันธ์การอ้างอิง (Reference) ซึ่งสามารถให้ทุกคลาสสามารถอ้างอิงได้ เนื่องจากมีความต้องการเรียกใช้คลาสเป็นจำนวนมาก

4. เชนจ์รีเฟอเรนซ์ทูแวลู (Change Reference to Value) คือการเปลี่ยนแปลงความสัมพันธ์ที่เชื่อมต่อกันระหว่างคลาส 2 คลาส โดยเปลี่ยนแปลงจากความสัมพันธ์การอ้างอิงเป็นความสัมพันธ์การประกอบด้วย เนื่องจากมีความต้องการเรียกใช้คลาสเป็นจำนวนน้อย

5. รีเพลสอาเรย์วิธอ็อบเจกต์ (Replace Array with Object) คือการเปลี่ยนนิพจน์ที่เป็นอาเรย์ (Array) แล้วมิติมีเนื้อหาที่แตกต่างกัน เช่น อาเรย์ 2 มิติ แล้วแต่ละมิติมีเนื้อหาแตกต่างกัน โดยแทนที่อาเรย์ด้วยวัตถุแล้วเปลี่ยนมิติแต่ละตัวให้เป็นเขตข้อมูลหนึ่งในวัตถุเพื่อสามารถระบุความหมายที่ชัดเจนได้

6. ดุพลิเคทอ็อบเซิฟดาตา (Duplicate Observed Data) คือการแยกส่วนการติดต่อข้อมูลกับตัวข้อมูลออกจากกันเนื่องจากการมีรูปแบบการติดต่อหลายแบบทำให้ซอร์สโค้ดซ้ำซ้อนอ่านแล้วทำความเข้าใจยาก โดยการสร้างส่วนของข้อมูลที่เหมือนกันเพื่อแทนที่การติดต่อนั้นขึ้นมาแล้วแยกส่วนออกจากกัน

7. เชนจ์ยูนิไดเรกชันนอลแอสโซซิเอชันทูไบไดเรกชันนอล (Change Unidirectional Association to Bidirectional) คือการเปลี่ยนแปลงความสัมพันธ์ของการเชื่อมต่อระหว่างคลาส 2 คลาสจากความสัมพันธ์แบบทางเดียวให้เป็นความสัมพันธ์ 2 ทาง เนื่องจากคลาสทั้ง 2 จำเป็นต้องเรียกใช้ซึ่งกันและกันบ่อยครั้ง

8. เชนจ์ไบไดเรกชันนอลแอสโซซิเอชันทูยูนิไดเรกชันนอล (Change Bidirectional Association to Unidirectional) คือการเปลี่ยนแปลงความสัมพันธ์ของการเชื่อมต่อระหว่างคลาส 2 คลาสจากความสัมพันธ์แบบ 2 ทางให้เป็นความสัมพันธ์ทางเดียว ซึ่งเพียงพอต่อการใช้งาน

9. รีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ (Replace Magic Number with Symbolic Constant) คือการเปลี่ยนค่าคงที่ (Constant) ต่างๆ ในระบบเป็นตัวแปร (Variable) ที่สร้างขึ้น โดยตัวแปรนี้จะสื่อความหมายที่ชัดเจนขึ้น

10. เอนแคปซูเลทฟิลด์ (Encapsulate Field) คือการเปลี่ยนคลาสที่มีเขตข้อมูลที่สามารถเข้าถึงได้โดยตรง แล้วเปลี่ยนให้สามารถเข้าถึงได้โดยอ้อมโดยใช้วิธีเกตติงและเซตติง

11. เอนแคปซูเลทคอลเลกชัน (Encapsulate Collection) คือการเปลี่ยนเมท็อดที่มีการคืนที่เป็นกลุ่มข้อมูล โดยการเปลี่ยนแปลงจะจัดการส่วนของข้อมูลที่คืนมาเป็นข้อมูลที่ไม่สามารถแก้ไขได้ แล้วเพิ่มเมท็อดการเพิ่มและลบข้อมูลเข้าไป

12. รีเพลสเรกคอร์ดวิธ ดาตาคลาส (Replace Record with Data Class) คือการเปลี่ยนที่ข้อมูลชุดที่มีลักษณะเป็นโครงสร้างหลายเขตข้อมูลเป็นวัตถุ

13. รีเพลสไทป์โค้ดวิธ คลาส (Replace Type Code with Class) คือการแทนที่ชนิดของข้อมูลที่มีหลายแบบหรือหลายประเภทด้วยการสร้างเป็นคลาสใหม่

14. รีเพลสไทป์โค้ดวิธซับคลาส (Replace Type Code with Subclasses) คือการแทนที่ชนิดของข้อมูลที่ไม่เปลี่ยนแปลง แต่มีผลกระทบต่อพฤติกรรมของคลาสด้วยการสร้างเป็นคลาสลูก (Subclass) ใหม่

15. รีเพลสไทป์โค้ดวิธสเตต/สเตรตีจี (Replace Type Code with State/strategy) คือการแทนที่ชนิดของข้อมูลโดยที่ชนิดของข้อมูลนี้ส่งผลกระทบต่อพฤติกรรมของคลาส แต่ใช้วิธีการแทนที่ด้วยคลาสลูกไม่ได้ทำให้ต้องใช้วิธีการแทนชนิดของข้อมูลด้วยสถานะวัตถุ (State Object) ที่สร้างใหม่

16. รีเพลสซับคลาสวิธฟิลด์ (Replace Subclass with Fields) คือการเปลี่ยนแปลงจากคลาสลูกจำนวนมากที่มีเพียงเมท็อดของเกตติงและเซตติงเป็นลักษณะประจำ (Attribute) ในคลาสทำให้คลาสลูกนั้นไม่จำเป็นอีกต่อไป

กระบวนการรีแฟคทอริงในประเภทออแกไนซิงดาต้าถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **เอนแคปซูเลทฟิลด์**
  สาเหตุ คือมีข้อมูลที่สามารถเข้าถึงได้โดยตรง
  วิธีการแก้ไข คือจัดการข้อมูลนั้นให้มีการเข้าถึงโดยใช้วิธีเกตติงและเซตติง
  ตัวอย่าง

```
public String _name
```

รูปที่ 2-12 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
private String _name;
public String getName() {return _name;}
public void setName(String arg) {_name = arg;}
```

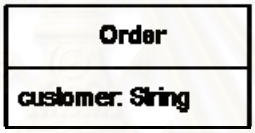
รูปที่ 2-13 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีเอนแคปซูเลทฟิลด์

- **รีเพลสดาตาแวลู วิธอ็อบเจกต์**

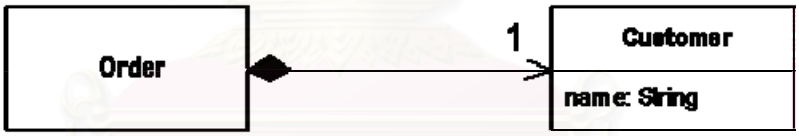
  สาเหตุ คือเขตข้อมูลเดิมมีข้อมูลหรือพฤติกรรมเพิ่มเติม

  วิธีการแก้ไข คือเปลี่ยนลักษณะประจำนั้นให้เป็นคลาส

  ตัวอย่าง



รูปที่ 2-14 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง



รูปที่ 2-15 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีรีเพลสดาตาแวลูวิธอ็อบเจกต์

- **รีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์**

  สาเหตุ คือมีค่าคงที่ในระบบที่ถูกเรียกใช้บ่อยครั้ง

  วิธีการแก้ไข คือเปลี่ยนค่าคงที่นั้นให้เป็นตัวแปรให้เรียกใช้

  ตัวอย่าง

```
double potentialEnergy(double mass, double height) {
        return mass * height * 9.81;
}
```

รูปที่ 2-16 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
double potentialEnergy(double mass, double height) {
        return mass * GRAVITATIONAL_CONSTANT * height;
        }
static final double GRAVITATIONAL_CONSTANT = 9.81;
```

รูปที่ 2-17 แสดงซอร์สโค้ดที่เป็นผลจากวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์

D. ซิมพลิไฟอิงคอนดิชันนอลเอ็กเพรสชัน (Simplifying Conditional Expressions) คือการจัดการเงื่อนไขของซอร์สโค้ดซึ่งสิ่งที่จัดการได้ยาก เนื่องจากการแก้ไขทำให้กระทบต่อซอร์สโค้ดส่วนอื่น ดังนั้นการนำกระบวนการรีแฟคทอริงเข้ามาช่วยสามารถช่วยจัดการได้ง่ายขึ้น โดยกระบวนการรีแฟคทอริงสำหรับประเภทซิมพลิไฟอิงคอนดิชันนอลเอ็กเพรสชัน มีดังนี้

1. ดีคอมโพสคอนดิชันนอล (Decompose Conditional)คือการยุบซอร์สโค้ดที่มีเงื่อนไขยุ่งยากซับซ้อนให้เป็นเมท็อดแล้วเรียกใช้ผ่านเมท็อด

2. คอนโซลิเดทคอนดิชันนอลเอ็กเพลสชัน (Consolidate Conditional Expressions) คือการแทนที่ซอร์สโค้ดเงื่อนไขที่เป็นลำดับของการทดสอบเงื่อนไขแล้วตอบสนองด้วยผลลัพธ์เดียวกัน ให้แยกซอร์สโค้ดเงื่อนไขนั้นให้เป็นเมท็อดแล้วเรียกใช้จากเมท็อดนี้

3. คอนโซลิเดทดุพลิเคทคอนดิชันนอลแฟรกเมนต์ (Consolidate Duplicate Conditional Fragment) คือการดึงซอร์สโค้ดที่ทำงานเหมือนกันในทุกเงื่อนไขให้ย้ายออกไปนอกเงื่อนไขเพื่อไม่ให้เกิดการทำงานที่ซ้ำซ้อนและทำให้ซอร์สโค้ดมีความชัดเจน

4. รีมูฟคอนโทรลแฟลก (Remove Control Flag) คือการเปลี่ยนตัวแปรที่ใช้เพื่อควบคุมการทำงานของเงื่อนไข โดยเปลี่ยนไปใช้คำสั่งหยุด (Break) และคำสั่งกลับคืน (Return) แทน

5. รีเพลสเนสเตดคอนดิชันนอลวิธการ์ดคลอส (Replace Nested Conditional with Guard Clauses) คือการเปลี่ยนซอร์สโค้ดที่เป็นเงื่อนไขที่ไม่ชัดเจน โดยใช้วิธีการกรองเงื่อนไขสำหรับกรณีพิเศษทั้งหมดก่อน

6. รีเพลสคอนดิชันนอลวิธพอลิมอฟิซึม (Replace Conditional with Polymorphism) คือการเปลี่ยนซอร์สโค้ดที่เป็นเงื่อนไขรูปแบบสวิตช์เคส (Switch Case) ให้เป็นรูปแบบของภาวะพหุสัณฐาน (Polymorphism) โดยให้เงื่อนไขแต่ละข้อเปลี่ยนเป็นคลาสลูก แล้วทำโอเวอร์ไรดิง (Overriding) มาที่คลาสแม่ (Superclass)

7. อินโทรดิวซ์นัลอ็อบเจกต์ (Introduce Null Object) คือการเปลี่ยนค่าว่าง (Null) ให้เป็นวัตถุว่าง (Null Object) เนื่องจากมีนิพจน์ที่สร้างเงื่อนไขเกี่ยวกับค่าว่างของวัตถุ

8. อินโทรดิวซ์แอซเซิสชัน (Introduce Assertion) คือการแก้ไขให้ซอร์สโค้ดมีการตรวจสอบสิทธิ์ก่อนเข้าใช้งานเมท็อด

กระบวนการรีแฟคทอริงในประเภทซิมพลิไฟอิงคอนดิชันนอลเอ็กเพรสชันถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **ดีคอมโพสคอนดิชันนอล**

  สาเหตุ คือการมีเงื่อนไขที่ยุ่งยากซับซ้อน

  วิธีการแก้ไข คือแยกเมท็อดจากเงื่อนไขดังกล่าวแล้วแยกเป็นส่วนๆ

  ตัวอย่าง

```
if (date.before (SUMMER_START) || date.after(SUMMER_END))
        charge = quantity * _winterRate + _winterServiceCharge;
else charge = quantity * _summerRate;
```

รูปที่ 2-18 แสดงตัวอย่างของซอร์สโค้ดที่ควรทำกระบวนการรีแฟคทอริง

```
if (notSummer(date))
        charge = winterCharge(quantity);
else charge = summerCharge (quantity);
```

รูปที่ 2-19 แสดงซอร์สโค้ดที่เป็นผลจากวิธีดีคอมโพสคอนดิชันนอล

E. เมกกิงเมท็อดคอลซิมเพลอร์ (Making Method Calls Simpler) คือการจัดการการเชื่อมต่อของคลาสซึ่งคลาสประกอบไปด้วยการเชื่อมต่อจากคลาสอื่นๆ อยู่แล้ว โดยการเชื่อมต่อที่เข้าใจได้ง่ายเป็นหัวใจในการพัฒนาซอฟต์แวร์เชิงวัตถุที่ดี โดยกระบวนการรีแฟคทอริงสำหรับประเภทเมกกิงเมท็อดคอลซิมเพลอร์ มีดังนี้

1. **รีเนมเมท็อด (Rename Method)** คือการเปลี่ยนแปลงชื่อของเมท็อดให้สื่อความหมายที่ความเข้าใจอย่างชัดเจน

2. **แอดพารามิเตอร์ (Add Parameter)** คือการเพิ่มพารามิเตอร์ให้กับคลาสที่ส่งข้อมูลมาให้ เนื่องจากเมท็อดต้องการใช้ข้อมูลมากกว่าข้อมูลที่ได้รับมาจากผู้เรียก

3. **รีมูฟพารามิเตอร์ (Remove Parameter)** คือการลบพารามิเตอร์ออกจากคลาส เนื่องจากพารามิเตอร์ไม่ได้ถูกใช้ในเมท็อดนั้น

4. **เซเพอเรทคิวรีฟรอมโมดิไฟเออร์ (Separate Query from Modifier)** คือการแยกเมท็อดซึ่งทำงาน 2 หน้าที่ออกเป็น 2 เมท็อด เนื่องจากเมท็อดนี้เป็นเมท็อดที่ส่งคืนค่า แต่กลับมีการเปลี่ยนแปลงค่าที่ส่งคืน ดังนั้นการแยกเป็น 2 เมท็อดจึงเหมาะสมกว่า

5. **พารามิเตอร์ไรซ์เมท็อด (Parameterize Method)** คือการเปลี่ยนเมท็อดหลายๆ เมท็อดที่มีลักษณะการทำงานเหมือนกันแต่ต่างกันที่ค่าคงที่ที่นำไปใช้ ดังนั้นจึงสามารถสร้างเมท็อดที่ใช้พารามิเตอร์สำหรับค่าคงที่ที่แตกต่างกันออกไป

6. **รีเพลสพารามิเตอร์วิธเอ็กพลิซิทเมท็อด (Replace Parameter with Explicit Method)** คือการสร้างเมท็อดที่แยกสำหรับแต่ละค่าพารามิเตอร์ เนื่องจากเมท็อดเดิมเป็นซอร์สโค้ดที่ทำงานแตกต่างกันจากค่าที่รับมา

7. **พรีเซิฟโฮลอ็อบเจกต์ (Preserve Whole Object)** คือการเปลี่ยนจากการส่งค่าพารามิเตอร์หลายค่าเป็นการส่งทั้งคลาสนั้นไปเนื่องจากวิธีเดิมจะส่งค่าพารามิเตอร์หลายค่า แต่สามารถเปลี่ยนมาส่งทั้งค่าทั้งหมดได้โดยส่งทั้งคลาสแทน

8. **รีเพลสพารามิเตอร์วิธเมท็อด (Replace Parameter with Method)** คือการแทนที่พารามิเตอร์ด้วยเมท็อด เนื่องจากพารามิเตอร์ถูกสร้างขึ้นเพื่อเรียกใช้เมท็อดอีกที การเรียกใช้จากเมท็อดโดยตรงจะเหมาะสมกว่า

9. **อินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (Introduce Parameter Object)** คือการเปลี่ยนกลุ่มของพารามิเตอร์ในเมท็อดเป็นวัตถุ

10. **รีมูฟเซตติงเมท็อด (Remove Setting Method)** คือการลบเมท็อดที่ไม่เคยแก้ไขและใช้งานออกจากคลาส

11. **ไฮด์เมท็อด (Hide Method)** คือการซ่อนเมท็อดที่ไม่ได้ถูกใช้ในคลาสอื่นๆ เลย

12. **รีเพลสคอนสตรักเตอร์วิธแฟกทอรีเมท็อด (Replace Constructor with Factory Method)** คือการแทนที่ตัวสร้าง (Constructor) ด้วยแฟคทอรีเมท็อด (Factory Method)

13. **เอนแคปซูเลทดาวน์แคป (Encapsulate Downcast)** คือการเคลื่อนย้ายเมท็อดการดาวน์แคป (Downcast) ไปไว้ในเมท็อด เนื่องจากเมท็อดส่งคืนค่าวัตถุที่ถูกดาวน์แคปโดยผู้เรียกเอง

14. รีเพลสเออเรอโค้ดวิธเอ็กเซปชัน (Replace Error Code with Exception) คือการใช้ความผิดปรกติ (Exception) เพื่อบอกความผิดพลาดต่างๆ

15. รีเพลสเอ็กเซปชันวิธเทส (Replace Exception with Test) คือการเปลี่ยนจากการใช้ความผิดปรกติไปใช้เงื่อนไขในการตรวจสอบผู้เรียกก่อน

กระบวนการรีแฟคทอริงในประเภทเมกกิงเมท็อดคอลซิมเพลอร์ถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **รีเนมเมท็อด**

  สาเหตุ คือชื่อของเมท็อดไม่แสดงวัตถุประสงค์ที่ชัดเจน

  วิธีการแก้ไข คือเปลี่ยนชื่อเมท็อดเป็นชื่อที่สื่อความหมายที่ชัดเจน

  ตัวอย่าง

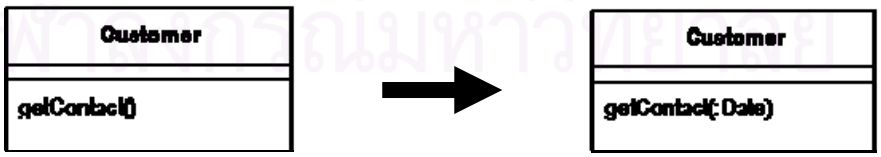
รูปที่ 2-20 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีรีเนมเมท็อด

- **แอดพารามิเตอร์**

  สาเหตุ คือเมท็อดต้องการพารามิเตอร์เพิ่มเติม

  วิธีการแก้ไข คือเพิ่มพารามิเตอร์กับวัตถุที่ส่งไป

  ตัวอย่าง



รูปที่ 2-21 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีแอดพารามิเตอร์

- **รีมูฟพารามิเตอร์**

  สาเหตุ คือเมท็อดไม่ต้องการใช้พารามิเตอร์นั่นอีก

  วิธีการแก้ไข คือลบพารามิเตอร์กับวัตถุที่ส่งไป

  ตัวอย่าง

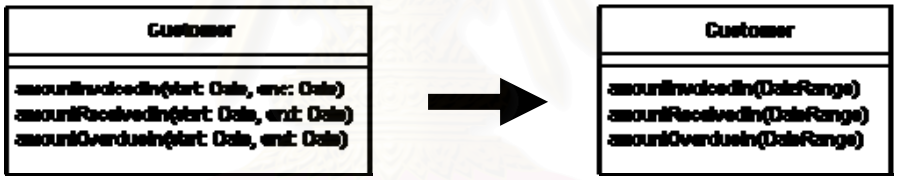
รูปที่ 2-22 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีรีมูฟพารามิเตอร์

- **อินโทรดิวซ์พารามิเตอร์อ็อบเจกต์**

  สาเหตุ คือกลุ่มของพารามิเตอร์เป็นกลุ่มที่ต้องทำงานร่วมกันตลอด เช่น วัน/เดือน/ปี

  วิธีการแก้ไข คือเปลี่ยนพารามิเตอร์เป็นวัตถุ

  ตัวอย่าง



รูปที่ 2-23 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีอินโทรดิวซ์พารามิเตอร์อ็อบเจกต์

F. ดีลลิงวิธเจนเนอรัลไลเซชัน (Dealing with Generalization) คือการจัดการกับลำดับชั้นของการสืบทอด (Inheritance) โดยกระบวนการรีแฟคทอริงสำหรับประเภทดีลลิงวิธเจนเนอรัลไลเซชัน มีดังนี้

1. **พูลอัพฟิลด์ (Pull Up Field)** คือการย้ายเขตข้อมูลที่เหมือนกันไปไว้ในคลาสแม่
2. **พูลอัพเมท็อด (Pull Up Method)** คือการย้ายเมท็อดที่เหมือนกันไปไว้ในคลาสแม่
3. **พูลอัพคอนสตรักเตอร์บอดี (Pull Up Constructor Body)** คือการสร้างตัวสร้างที่คลาสแม่แล้วให้เมท็อดเรียกจากคลาสลูก เนื่องจากในคลาสลูกมีซอร์สโค้ดเหมือนกันกับคลาสแม่
4. **พุชดาวน์เมท็อด (Push Down Method)** คือการย้ายเมท็อดไปไว้ที่คลาสลูก เนื่องจากเมท็อดมีความเกี่ยวข้องกับคลาสลูกนั้นเพียงคลาสเดียว
5. **พุชดาวน์ฟิลด์ (Push Down Field)** คือการย้ายเขตข้อมูลไปไว้ที่คลาสลูก เนื่องจากเขตข้อมูลมีความเกี่ยวข้องกับคลาสลูกนั้นเพียงคลาสเดียว

6. เอ็กแทรกซับคลาส (Extract Subclass) คือการสร้างคลาสลูกเพื่อรองรับลักษณะที่ใช้เฉพาะในคลาสนั้นเท่านั้น

7. เอ็กแทรกซุปเปอร์คลาส (Extract Superclass) คือการสร้างคลาสแม่ขึ้นเพื่อนำคลาส 2 คลาส ที่มีลักษณะประจำที่เหมือนกัน แล้วนำลักษณะประจำที่เหมือนกันไปไว้ในคลาสแม่แล้วให้ 2 คลาส นั้นเป็นคลาสลูก

8. เอ็กแทรกอินเตอร์เฟซ (Extract Interface) คือการแยกเมท็อดที่เป็นส่วนของการติดต่อออกคลาสที่เป็นส่วนของการติดต่อที่ชัดเจน

9. คอลแลปส์ไฮราคี (Collapse Hierarchy) คือการรวมคลาสแม่และคลาสลูกเข้าด้วยกัน เนื่องจาก 2 คลาส นั้นไม่แตกต่างกัน

10. ฟอร์มเทมเพลตเมท็อด (Form Template Method) คือการนำเมท็อดที่ทำงานเหมือนกันแล้วดึงไปไว้ในคลาสแม่ โดยสามารถสร้างเมท็อดต้นแบบไว้ที่คลาสแม่ได้

11. รีเพลสอินเฮอริแทนซ์ดีลีเกชัน (Replace Inheritance with Delegation) คือการสร้างคลาสแม่ขึ้นแล้วปรับเมท็อดเอาเข้าไปแทนด้วยคลาสแม่แล้วลบคลาสลูกออก เนื่องจากคลาสลูกถูกใช้เป็นส่วนหนึ่งของการติดต่อของคลาสแม่หรือไม่ต้องการให้สืบทอดข้อมูล

12. รีเพลสดีลีเกชันวิธอินเฮอริแทนซ์ (Replace Delegation with Inheritance) คือการสร้างคลาสที่ใช้เป็นคลาสตัวแทน เนื่องจากการใช้การดีลีเกจในการติดต่อหลายครั้งทำให้ยุ่งยาก

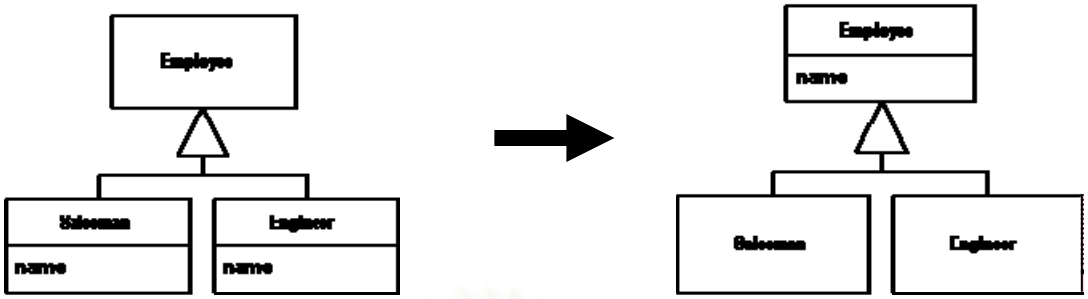
กระบวนการรีแฟคทอริงในประเภทดีลลิงวิธเจนเนอรัลไลเซชันถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์ เช่น ไมโครซอฟต์วิชวลสตูดิโอ 2005 (Microsoft Visual Studio 2005) อีคลิบซ์ (Eclipse) รีแฟคเทอร์ไอที (RefactorIT) ซีชาร์ปรีแฟคทอรี (C#Refactory) และ รีชาร์ปเปอร์ (ReSharper) ฯลฯ ซึ่งในแต่ละเครื่องมือได้นำหลักการของกระบวนการรีแฟคทอริงไปใช้ แต่นำไปใช้ในชื่อที่แตกต่างกันในบางวิธี จึงขอยกตัวอย่างอย่างละเอียด เช่น

- **พูลอัพฟิลด์**

  สาเหตุ คือมีเขตข้อมูลที่เหมือนกันในคลาสลูก

  วิธีการแก้ไข คือย้ายเขตข้อมูลไปไว้ที่คลาสแม่

  ตัวอย่าง



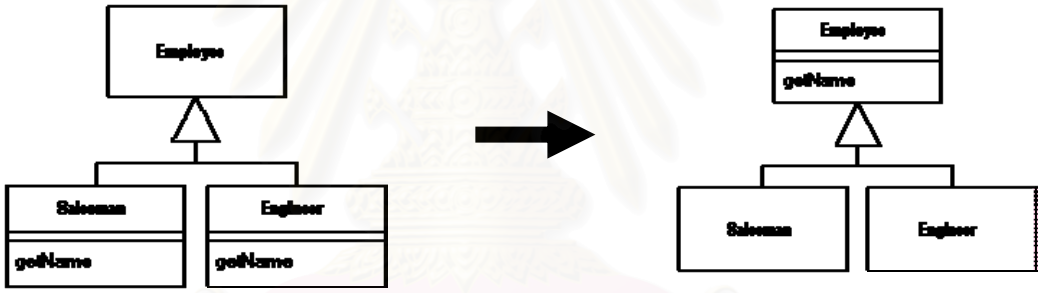
รูปที่ 2-24 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีพูลอัพฟิลด์

- **พูลอัพเมท็อด**

  สาเหตุ คือมีเมท็อดที่เหมือนกันในคลาสลูก

  วิธีการแก้ไข คือย้ายเมท็อดไปไว้ที่คลาสแม่

  ตัวอย่าง



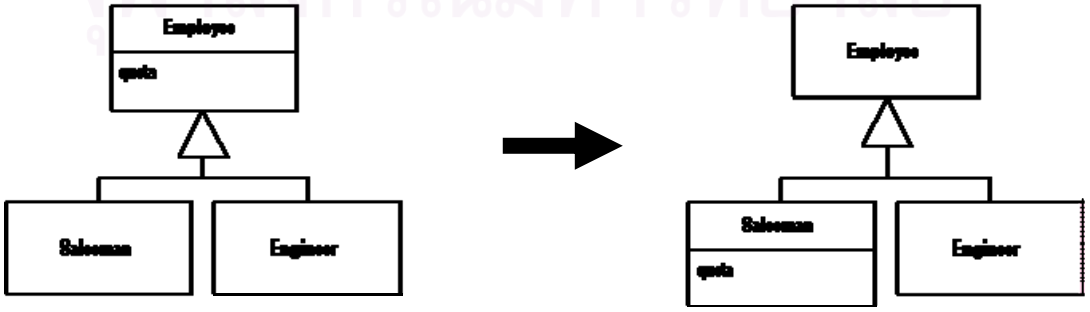
รูปที่ 2-25 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีพูลอัพเมท็อด

- **พุชดาวน์ฟิลด์**

  สาเหตุ คือมีเขตข้อมูลที่ใช้เพียงคลาสลูกเดียว

  วิธีการแก้ไข คือย้ายเขตข้อมูลไปไว้ที่คลาสลูกนั้น

  ตัวอย่าง



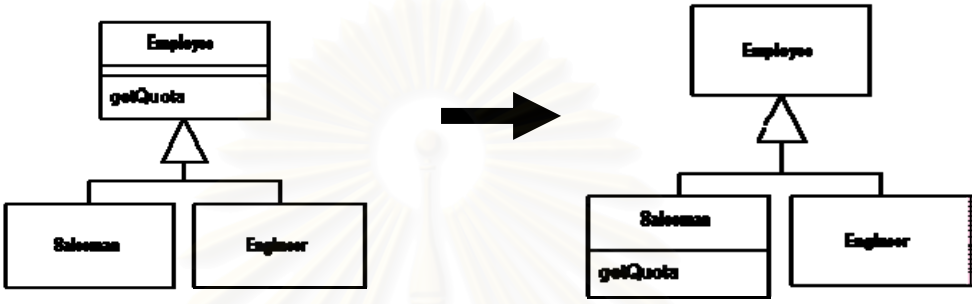
รูปที่ 2-26 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีพุชดาวน์ฟิลด์

- **พุชดาวน์เมท็อด**

  สาเหตุ คือมีเมท็อดที่ใช้เพียงคลาสลูกเดียว

  วิธีการแก้ไข คือย้ายเมท็อดไปไว้ที่คลาสลูกนั้น

  ตัวอย่าง



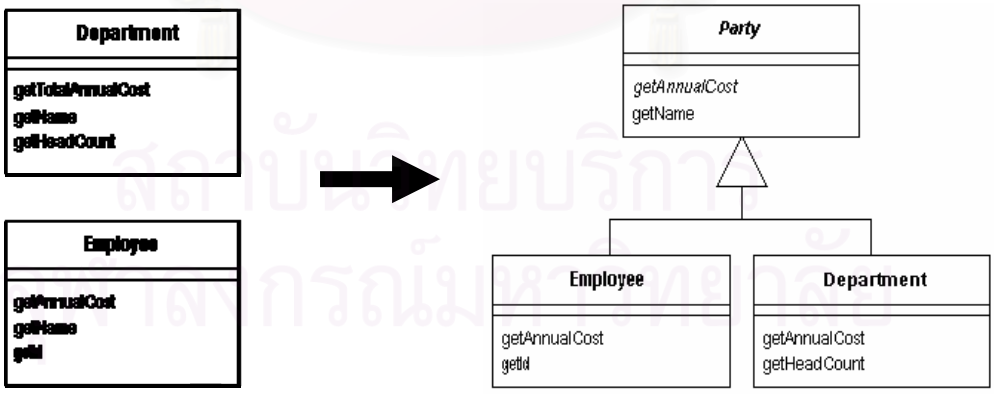
รูปที่ 2-27 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีพุชดาวน์เมท็อด

- **เอ็กแทรกซุปเปอร์คลาส**

  สาเหตุ คือมีคลาส 2 คลาส ที่มีลักษณะเหมือนกัน

  วิธีการแก้ไข คือสร้างคลาสแม่ขึ้นแล้วนำลักษณะประจำที่เหมือนกันไปไว้ในคลาสแม่ โดยให้ 2 คลาส ที่เหมือนกันนั้นเป็นคลาสลูก

  ตัวอย่าง



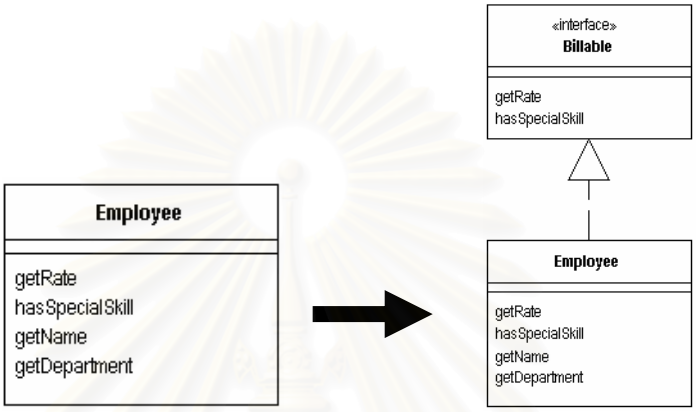
รูปที่ 2-28 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีเอ็กแทรกซุปเปอร์คลาส

- **เอ็กแทรกอินเตอร์เฟซ**

  สาเหตุ คือคลาสที่มีการติดต่อที่เหมือนกัน

  วิธีการแก้ไข คือแยกส่วนของส่วนต่อประสานออกเป็นส่วนของการติดต่อที่ชัดเจน

  ตัวอย่าง



รูปที่ 2-29 แสดงตัวอย่างของซอร์สโค้ดที่เป็นผลจากวิธีเอ็กแทรกอินเตอร์เฟซ

จากกระบวนการรีแฟคทอริงของ Fowler (1999) สามารถระบุวิธีในการสังเกตซอร์สโค้ดที่เป็นปัญหาหรือ **"ร่องรอยไม่ดีในซอร์สโค้ด"** ไว้ดังนี้

1. **ดุพลิเคทโค้ด (Duplicated Code)** คือเลือกซอร์สโค้ดที่เกิดการซ้ำซ้อนกันมากกว่า 1 จุด แล้วจัดการให้เป็นเมท็อดที่สามารถเรียกใช้งานได้ โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - การมีนิพจน์ (Expression) เหมือนกันใน 2 เมท็อด ภายในคลาสเดียว ให้แยกนิพจน์นั้นด้วยกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1)
   - การมีนิพจน์เหมือนกันใน 2 ซับคลาส ให้แยกนิพจน์นั้นด้วยกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) ออกจากทั้ง 2 ซับคลาส ก่อนจากนั้นจึงใช้กระบวนการ รีแฟคทอริงวิธีพูลอัพเมท็อด (F1) กับเมท็อดที่แยกออกมา
   - การมีนิพจน์เหมือนกันใน 2 ซับคลาส โดยโครงสร้างซอร์สโค้ดเหมือนกัน แต่รายละเอียดภายในซอร์สโค้ดไม่เหมือนกัน ให้แยกนิพจน์ด้วยกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) จากนั้นใช้กระบวนการรีแฟคทอริงวิธีฟอร์มเทมเพลตเมท็อด (F10) กับเมท็อดที่แยกออกมา
   - การมีนิพจน์เหมือนกันใน 2 ซับคลาส โดยเมท็อดทำหน้าที่เหมือนกันแต่มีความแตกต่างกันที่ขั้นตอนวิธีการทำงาน ให้ใช้กระบวนการรีแฟคทอริงวิธีซับสติติวต์อัลกอริทึม (A9) กับเมท็อดนั้น

- การมีนิพจน์เหมือนกันใน 2 คลาส ที่ขั้นตอนวิธีการทำงานไม่มีความเกี่ยวข้องกัน ให้ใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3)

2. ลองเมท็อด (Long Method) คือเลือกซอร์สโค้ดในเมท็อดที่มีความยาวของซอร์สโค้ดมากเกินไป จัดการให้เป็นเมท็อดที่เหมาะสมและเข้าใจได้ง่าย โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- สำหรับเมท็อดที่มีพารามิเตอร์และตัวแปรชั่วคราวจำนวนมาก ให้ใช้กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี (A4) เพื่อขจัดตัวแปรชั่วคราวจากนั้นจึงใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (E9) และวิธีพรีเซิฟโฮลอ็อบเจกต์ (E7) ในการลดจำนวนพารามิเตอร์
- ถ้าเมท็อดนั้นยังคงมีพารามิเตอร์และตัวแปรชั่วคราวอยู่เป็นจำนวนมาก ให้ใช้กระบวนการรีแฟคทอริงวิธีรีเพลสเมท็อดวิธเมท็อดอ็อบเจกต์ (A8)
- สำหรับขั้นตอนการทำงานที่มีเงื่อนไขและลูปที่สามารถแบ่งแยกได้ ให้ใช้กระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอล (D1)

3. ลาจคลาส (Large Class) คือเลือกซอร์สโค้ดในคลาสที่มีขนาดใหญ่ หรือมีจำนวนตัวแปรเป็นจำนวนมาก โดยสามารถจัดการให้คลาสมีหน้าที่การทำงานที่เหมาะสม โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) หรือวิธีเอ็กแทรกซับคลาส (F6) สำหรับคลาสที่มีขนาดใหญ่ โดยอาจเลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกอินเตอร์เฟซ (F8) สำหรับความต้องการที่มีความแตกต่างกัน

4. ลองพารามิเตอร์ลิสต์ (Long Parameter List) คือเลือกเมท็อดที่มีการส่งผ่านพารามิเตอร์จำนวนมาก จัดการให้พารามิเตอร์อยู่ในรูปแบบที่เหมาะสมและเรียกใช้งานได้ง่าย โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- สำหรับพารามิเตอร์ที่เป็นส่วนหนึ่งในการเรียกใช้เมท็อดให้เปลี่ยนพารามิเตอร์ให้กลายเป็นเมท็อด โดยเลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสพารามิเตอร์วิธเมท็อด (E8)
- สำหรับพารามิเตอร์ที่มีการแบ่งแยกของข้อมูลเป็นกลุ่มๆ แล้วแทนที่ด้วยลักษณะแบบอ็อบเจกต์ โดยใช้กระบวนการรีแฟคทอริงวิธีพรีเซิฟโฮลอ็อบเจกต์ (E7)
- ถ้าพารามิเตอร์มีความหลากหลายของข้อมูล ให้เลือกใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (E9)

5. **ไดเวอเจนต์เชนจ์ (Divergent Change)** คือเลือกคลาสที่ทำหน้าที่หรือรองรับการเปลี่ยนแปลงหลายๆ อย่างภายในคลาส โดยให้แยกคลาสออกจากกัน เพื่อทำหน้าที่แต่ละอย่างให้เหมาะสม โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) กับการเปลี่ยนแปลงที่กระทบกับคลาส 1 คลาส

6. **ชอตกันเซอเจอรี (Shotgun Surgery)** คือฟังก์ชันใดๆ เมื่อเกิดการแก้ไขแล้วมีผลต้องแก้ไขกับคลาสหลายๆ คลาส โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - เลือกใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) และวิธีมูฟฟิลด์ (B2) สำหรับการจัดการกับซอร์สโค้ดที่กระทบไปยังคลาสใหม่
   - เลือกใช้กระบวนการรีแฟคทอริงวิธีอินไลน์คลาส (B4) สำหรับการเปลี่ยนแปลงซอร์สโค้ดที่ส่งกระทบกับเงื่อนไขหลายๆ จุด

7. **ฟีเจอเอ็นวี (Feature Envy)** คือเมท็อดในคลาสใดๆ ที่เกิดการเรียกใช้ทรัพยากรจากคลาสอื่นมากกว่าคลาสที่เป็นอยู่ โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - เลือกใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) ซึ่งอาจจะต้องการใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) ก่อนที่จะใช้วิธีมูฟเมท็อด (B1)

8. **ดาตาคลัมพ์ (Data Clumps)** คือการจัดการกับข้อมูลที่เป็นกลุ่ม เช่น ข้อมูลที่อยู่ประกอบไปด้วย ถนน ตำบล อำเภอ และจังหวัด โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) สำหรับรวบรวมข้อมูลเข้ากันเป็นกลุ่ม แล้วต่อด้วยกระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (E9) หรือพรีเซิฟโฮลอ็อบเจกต์ (E7)

9. **พริมิทีฟออบเซสชัน (Primitive Obsession)** คือข้อมูลหรือชนิดของข้อมูลบางชนิดจำเป็นต้องสร้างเป็นคลาสเฉพาะ เพื่อให้การพัฒนาซอฟต์แวร์สามารถทำได้ง่ายขึ้น โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

   - เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสดาตาแวลูวิธอ็อบเจกต์ (C2) สำหรับข้อมูลส่วนบุคคล
   - เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสไทป์โค้ดวิธคลาส (C13) สำหรับชนิดของข้อมูล
   - เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสไทป์โค้ดวิธซับคลาส (C14) หรือรีเพลสไทป์โค้ดวิธสเตจ/สตราทิจี้ (C15) สำหรับข้อมูลที่ไม่กระทบกับคลาส

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) สำหรับกลุ่มของข้อมูลที่เป็นฟิลด์ข้อมูลและต้องใช้ประกอบกันตลอดเวลา
- เลือกใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (E9) ถ้ากลุ่มของข้อมูลอยู่ในรูปแบบพารามิเตอร์ หรืออาเรย์ ให้ใช้กระบวนการ รีแฟคทอริงวิธีรีเพลสอาเรย์วิธอ็อบเจกต์ (C5)

10. สวิตช์สเตตเมนต์ (Switch Statements) คือเลือกเงื่อนไขรูปแบบสวิตช์เคสที่ก่อให้เกิดปัญหาความซ้ำซ้อนของซอร์สโค้ด จึงควรเปลี่ยนให้อยู่ในรูปแบบของภาวะพหุสัณฐาน (Polymorphism) โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) เพื่อแยกกลุ่มของเงื่อนไขรูปแบบสวิตช์เคส จากนั้นจึงใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) เพื่อให้กลุ่มเงื่อนไขเข้าไปอยู่ในคลาสในรูปแบบของภาวะพหุสัณฐาน
- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสไทป์โค้ดวิธซับคลาส (C14) หรือวิธีรีเพลสไทป์โค้ดวิธสเตต/สเตรตีจี (C15) เมื่อต้องการจัดการให้โครงสร้างเป็นรูปแบบของความสามารถการสืบทอด หรือสามารถใช้กระบวนการรีแฟคทอริงวิธีรีเพลสคอนดิชันนอลวิธพอลิมอฟฟีซึม (D6)
- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสพารามิเตอร์วิธเอ็กพลิซิทเมท็อด (E6) สำหรับกรณีที่ผลกระทบที่เกิดจะเกี่ยวข้องกับเมท็อดเพียง 1 เมท็อด
- เลือกใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์นัลอ็อบเจกต์ (D7) ในกรณีที่เงื่อนไขเป็นค่าว่าง (Null)

11. พาราแลลอินเฮอริแทนซ์ไฮราคี (Parallel Inheritance Hierarchies) คือการสร้างคลาสลูกขึ้นในคลาสใดๆ ย่อมทำให้เกิดคลาสลูกอีกคลาสหนึ่งด้วย โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) และวิธีมูฟฟิลด์ (B2) สำหรับการรวมระดับชั้นให้เป็นอันเดียว

12. เลซีคลาส (Lazy Class) คือคลาสที่ไม่ได้มีฟังก์ชันการทำงานสำคัญมาทำกระบวนการรีแฟคทอริง โดยใช้กฎเกณฑ์ดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีคอลแลปส์ไฮราคี (F9) สำหรับซับคลาสที่ไม่ได้ทำหน้าที่อะไรมากมายหรือเลือกใช้วิธีอินไลน์คลาส (B4)

13. สเปกคิวเลทีฟเจเนรอลิตี (Speculative Generality) คือคลาสหรือเมท็อดที่สร้างเพื่อกรณีเฉพาะหรือรองรับฟังก์ชันการทำงานในอนาคตเกินความจำเป็น มาทำกระบวนการรีแฟคทอริงโดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีคอลแลปส์ไฮราคี (F9) สำหรับคลาสที่ไม่ค่อยได้ทำหน้าที่อะไร
- เลือกใช้กระบวนการรีแฟคทอริงวิธีอินไลน์คลาส (B4) สำหรับซอร์สโค้ดที่ไม่ถูกใช้งานแล้วแทนด้วยคลาส

14. เทมโพราลีฟิลด์ (Temporary Field) คือเลือกตัวแปรที่ถูกสร้างขึ้นเพื่อใช้สำหรับเก็บข้อมูลชั่วคราวมาทำกระบวนการรีแฟคทอริง โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) เพื่อสร้างคลาสมารองรับตัวแปรแล้วนำซอร์สโค้ดที่พิจารณามาใส่ไว้ จากนั้นอาจใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์นัลอ็อบเจกต์ (D7) เพื่อกำจัดเงื่อนไขในซอร์สโค้ดเดิมที่ตัวแปรอยู่ในรูปแบบของค่าว่าง

15. เมสเซสเชน (Message Chains) คือการเรียกใช้ทรัพยากรจากคลาสกันเป็นทอดๆ โดยไม่ได้เรียกใช้โดยตรงมาทำกระบวนการรีแฟคทอริง โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีไฮด์ดีลิเกท (B5) สำหรับการย้ายคลาส หรือเลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) และตามด้วยวิธีมูฟเมท็อด (B1)

16. มิดเดิลแมน (Middle Man) คือคลาสที่ทำหน้าที่เป็นคลาสตัวกลางสำหรับติดต่อระหว่างคลาส โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีมูฟมิดเดิลแมน (B6) สำหรับกำจัดคลาสที่ไม่ต้องการหรือเลือกใช้กระบวนการรีแฟคทอริงวิธีอินไลน์เมท็อด (A2) สำหรับเมท็อดที่ไม่มาก และถ้าเป็นกรณีการแก้ไขที่เกี่ยวข้องกับคลาส ให้ใช้กระบวนการรีแฟคทอริงวิธีรีเพลสดีลีเกชันวิธอินเฮอริแทนซ์ (F12)

17. อินแอพพรอพิเอทอินทิเมซี (Inappropriate Intimacy) คือการเรียกใช้ทรัพยากรของคลาสใดๆ โดยที่ทรัพยากรนั้นเป็นส่วนที่อยู่ในรูปแบบส่วนบุคคล (Private)โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) และวิธีมูฟฟิลด์ (B2) สำหรับการแยกคลาสที่ใกล้ชิดกันออกเป็นส่วนๆ และลดการเรียกใช้ข้อมูลที่ง่ายแต่ไม่ถูกหลักการ

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเชนจ์ยูนิไดเรกชันนอลแอสโซซิเอชันทูไบไดเรกชันนอล (C8) สำหรับจัดการการเรียกใช้ข้อมูล
- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส (B3) หรือวิธีไฮด์ดีลิเกท (B5) ถ้าเป็นกรณีคล้ายกับกรณีเมสเซสเชน

18. อัลเทอร์เนทีฟคลาสวิธดิฟเฟอเรนต์อินเตอร์เฟซ (Alternative Classes with Different Interfaces) คือการมีเมท็อดที่มีการทำงานเหมือนกันแต่มีซิกเนเจอร์ (Signature) ต่างกัน โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด (E1) สำหรับเมท็อดที่มีหน้าที่การทำงานเหมือนกันแต่มีความแตกต่างกันในเรื่องลักษณะลายเซ็น แต่บางครั้งจำเป็นต้องใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) สำหรับกรณีที่ย้ายเมท็อดไปยังคลาสหรือกรณีที่จำเป็นต้องให้เกิดความซ้ำซ้อนให้ใช้กระบวนการรีแฟคทอริงวิธีเอ็กซ์แทรคซุปเปอร์คลาส (F7)

19. อินคอมพลีทไลบรารีคลาส (Incomplete Library Class) คือคลาสที่สร้างเพื่อเป็นคลาสไลบราลี (Library) แต่คลาสนั้นไม่สมบูรณ์

- เลือกใช้กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์ฟอเรนจ์เมท็อด (B7) หรือวิธีอินโทรดิวซ์โลคอลเอ็กเทนชัน (B8)

20. ดาตาคลาส (Data Class) คือเลือกคลาสที่มีฟิลด์ข้อมูลเฉพาะเกตติงเมท็อดและเซตติงเมท็อดมาทำกระบวนการรีแฟคทอริง โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอนแคปซูเลทฟิลด์ (C10) สำหรับคลาสที่ทำหน้าที่เป็นที่พักข้อมูลโดยมีเฉพาะฟิลด์ข้อมูลภายในคลาสนั้น หรือใช้กระบวนการรีแฟคทอริงวิธีเอนแคปซูเลทคอลเลกชัน (C11) สำหรับฟิลด์ข้อมูลสะสมและถ้าไม่เป็นทั้ง 2 กรณีแรกให้ใช้วิธีรีมูฟเซตติงเมท็อด (E10)
- เลือกใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด (B1) สำหรับคลาสที่ทำหน้าที่เฉพาะเกตติงเมท็อดและเซตติงเมท็อด โดยย้ายจากคลาสที่เรียกใช้คลาสข้อมูลนั้นไปยังคลาสข้อมูลนั้นเลย ถ้าในกรณีที่ไม่สามารถย้ายได้ให้เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) จากนั้นให้ใช้กระบวนการ รีแฟคทอริงวิธีไฮด์เมท็อด (E11)

21. รีฟิวส์รีเควสท์ (Refused Request) คือคุณสมบัติของคลาสแม่ที่ถ่ายทอดมายังคลาสลูกแต่คลาสลูกไม่จำเป็นต้องใช้ โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีพุชดาวน์เมท็อด (F4) และวิธีพุชดาวน์ฟิลด์ (F5) สำหรับจัดการกับข้อมูลในคลาสที่ไม่จำเป็นต้องใช้ในคลาสนั้น

22. คอมเมนต์ (Comment) คือเมท็อดที่มีการอธิบายซอร์สโค้ดที่ไม่กระจ่าง โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด (A1) กับซอร์สโค้ดที่ต้องการอธิบายขั้นตอนการทำงานด้วยตัวซอร์สโค้ดเอง
- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี (A4) กับตัวเลขที่เป็นค่าที่ต้องใช้งานบ่อยและเมื่อแก้ไขค่าจะกระทบกับซอร์สโค้ดหลายจุด
- เลือกใช้วิธีรีเนมเมท็อด (E1) สำหรับเมท็อดที่มีชื่อไม่ชัดเจน

23. ไทป์เอ็มเบดเดดอินเนม (Type Embedded in Name) คือเลือกเมท็อดที่มีชื่อซ้ำซ้อนกันแต่มีขั้นตอนการทำงานไม่เหมือนกันในต่างคลาส โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด (E1) สำหรับเมท็อดที่มีชื่อซ้ำซ้อนกัน เพื่อให้สามารถทำความเข้าใจได้ง่าย

24. อันคอมมิวนิเคทีฟเนม (Uncommunicative Name) คือเลือกเมท็อดที่มีชื่อไม่เหมาะสมตามเจตนาหรือวัตถุประสงค์ โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด (E1) สำหรับเมท็อดที่มีชื่อไม่เหมาะสมตามเจตนาหรือวัตถุประสงค์

25. อินคอนซิสเทนซ์เนม (Inconsistence Name) คือเลือกเมท็อดที่มีชื่อไม่เป็นหลักเกณฑ์ที่แน่นอน โดยใช้กฎเกณฑ์ในการเลือกปฏิบัติดังนี้

- เลือกใช้กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด (E1) สำหรับเมท็อดที่มีชื่อไม่เป็นหลักเกณฑ์ที่แน่นอน

จากกระบวนการรีแฟคทอริงทั้ง 72 วิธีจาก 6 ประเภท งานวิจัยนี้ได้เลือกกระบวนการรีแฟคทอริงที่ถูกนำไปใช้ในเครื่องมือพัฒนาซอฟต์แวร์หรือเครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริงโดยเฉพาะ ดังนี้

1. เอ็กแทรกเมท็อด (A1)
2. รีเพลสเทมป์วิธคิวรี (A4)
3. อินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล (A5)
4. มูฟเมท็อด (B1)
5. มูฟฟิลด์ (B2)
6. เอ็กแทรกคลาส (B3)

7. เอนแคปซูเลทฟิลด์ (C10)
8. รีเพลสดาตาแวลู วิธอ็อบเจกต์ (C2)
9. รีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ (C9)
10. ดีคอมโพสคอนดิชันนอล (D1)
11. รีเนมเมท็อด (E1)
12. แอดพารามิเตอร์ (E2)
13. รีมูฟพารามิเตอร์ (E3)
14. อินโทรดิวซ์พารามิเตอร์อ็อบเจกต์ (E9)
15. พูลอัพฟิลด์ (F1)
16. พูลอัพเมท็อด (F2)
17. พุชดาวน์ฟิลด์ (F5)
18. พุชดาวน์เมท็อด (F4)
19. เอ็กแทรกซุปเปอร์คลาส (F7)
20. เอ็กแทรกอินเตอร์เฟซ (F8)

จากกระบวนการรีแฟคทอริงทั้ง 20 วิธี ที่เลือกใช้จะถูกนำมาใช้ด้วยเครื่องมือรีชาร์ปเปอร์ 2.0 (ReSharper 2.0) ซึ่งเป็นซอฟต์แวร์ที่ถูกเพิ่ม (Add - in) ให้กับเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 (Visual Studio .NET 2005) โดยจำกัดเพียงการพัฒนาซอฟต์แวร์ด้วยภาษาซีชาร์ป (C#) เครื่องมือรีชาร์ปเปอร์ 2.0 (ReSharper 2.0) ถูกสร้างขึ้นโดยบริษัทเจ็ทเบรน (JetBrains) โดยมีลักษณะหน้าที่การทำงานดังนี้

1. ซินแทคและเออเรอไฮไลท์ทิง (Syntax & Error Highlighting) เครื่องมือรีชาร์ปเปอร์สามารถวิเคราะห์ซอร์สโค้ดที่ผิดพลาดได้และจะชี้จุดที่พบการผิดพลาดได้อย่างอัตโนมัติ
2. เออเรอควิกฟิคซ์ (Error Quick-Fixes) เครื่องมือรีชาร์ปเปอร์สามารถจัดการกับความผิดพลาดของซอร์สโค้ดได้อย่างรวดเร็ว โดยการสร้างคำตอบที่ถูกต้องให้กับความผิดพลาดของซอร์สโค้ดที่เกิดขึ้น
3. รีแฟคทอริง (Refactoring) เครื่องมือรีชาร์ปเปอร์สามารถทำกระบวนการรีแฟคทอริงได้ตามวิธีดังนี้
   - รีเนมซิมโบลวิธเรฟเฟอเรนซ์คอเรคชัน (Rename Symbols with Reference Correction)
   - มูฟไทป์วิธเรฟเฟอเรนซ์คอเรคชัน (Move Types with Reference Correction)
   - เอ็กแทรกไทป์ทูอะนิวไฟล์(Extract Type to a New File)

- เชนจ์เมท็อดซิคเนเชอร์ (Change Method Signature)
- เอ็กแทรกเมท็อด (Extract Method)
- เอ็กแทรกคลาสฟอร์มพารามิเตอร์ (Extract Class from Parameters)
- อินโทรดิวซ์แวริเอเบิล (Introduce Variable)
- อินโทรดิวซ์พารามิเตอร์ (Introduce Parameter)
- อินโทรดิวซ์ฟิลด์ (Introduce Field)
- อินไลน์แวริเอเบิล (Inline Variable)
- คอนเวิร์ทเมท็อดทูพรอพเพอที (Convert Method to Property)
- คอนเวิร์ทพรอพเพอทีทูเมท็อด (Convert Property to Method)
- เอ็กแทรกอินเตอร์เฟซ (Extract Interface)
- เอ็กแทรกซุปเปอร์คลาส (Extract Superclass)
- ก็อปปีไทป์ (Copy Type)
- เอ็นแคปซูเลทฟิลด์ (Encapsula Field)
- คอนเวิร์ทอินเตอร์เฟซทูแอปสแทรกคลาส (Convert Interface to Abstract Class)
- คอนเวิร์ทแอปสแทรกคลาสทูอินเตอร์เฟซ (Convert Abstract Class to Interface)
- พูลเมมเบอร์อัพ (Pull Member Up)
- พูลเมมเบอร์ดาวน์ (Push Member Down)
- รีเพลสคอนสตรัคเตอร์วิธแฟคทอรีเมท็อด (Replace Contructor with Factory Method)
- เมคเมท็อดสแทททิค (Make Method Static)
- เมคเมท็อดนอนสแทททิค (Make Method Non – Static)
- ยูสเบสไทป์แวร์พอสสิเบิล (Use Base Type where Possible)
- มูฟสแทททิคเมมเบอร์ (Move Static Members)
- เซฟดีลีท (Safe Delete)

4. เนฟวิเกชัน (Navigation) เครื่องมือรีชาร์ปเปอร์สามารถเข้าถึงข้อมูลต่างๆ จากการทำงานของรีชาร์ปเปอร์ในส่วนนี้

5. ค้นหา (Search) เครื่องมือรีชาร์ปเปอร์สามารถค้นหาจำนวนการใช้ของเนมสเปซ (Namespaces) ไทป์เมท็อด (Types Methods) ฟิลด์ (Fields) หรือโลคอลแวริเอเบิล (Local Variables) ในซอร์สโค้ดได้

6. ไลฟ์เทมเพลส (Live Templates) เครื่องมือรีชาร์ปเปอร์สามารถสร้างแบบของซอร์สโค้ดที่มีโครงสร้างง่ายๆ และเหมาะสมกับการใช้งาน

7. โค้ดเจเนอเรชัน (Code Generation) เครื่องมือรีชาร์ปเปอร์สามารถสร้างซอร์สโค้ดอย่างง่ายให้ได้ ตัวอย่างเช่น ซอร์สโค้ดเกี่ยวกับเงื่อนไข อิฟ .. เอล (If .. Else) วาย (While) ฟอร์ (For) และดู .. วาย (Do .. While)

8. โค้ดคอมพลีชัน (Code Completion) เครื่องมือรีชาร์ปเปอร์สามารถเติมคำให้กับการเขียนซอร์สโค้ดในขณะปฏิบัติงาน ทำให้สามารถเขียนซอร์สโค้ดได้รวดเร็วและถูกต้องมากขึ้น

9. โค้ดแอสซิสแทนซ์ (Code Assistance) เครื่องมือรีชาร์ปเปอร์ช่วยให้การพัฒนาซอร์สโค้ดทำให้สะดวกสบายมากขึ้นโดยมี การอธิบายรายละเอียดของข้อมูล แหล่งที่มาและการจับคู่วงเล็บให้เห็น

10. โค้ดฟอร์แมททิง (Code Formatting) เครื่องมือรีชาร์ปเปอร์สามารถจัดการกับรูปแบบของซอร์สโค้ดได้ตามใจผู้พัฒนา

11. อัลเทอโปรดักทิวิทีฟีเจอร์ (Other Productivity Features) นอกจากนี้แล้วเครื่องมือรีชาร์ปเปอร์ยังสามารถแสดงจุดเด่นในซอร์สโค้ดเกี่ยวกับจำนวนการใช้ด้วยสีที่แตกต่างกัน การจัดการกับคำอธิบายซอร์สโค้ดและการยืดหดซอร์สโค้ดที่ต้องการเลือกด้วยการลากเมาส์ (Mouse)

โดยเครื่องมือรีชาร์ปเปอร์ต้องการความต้องการของระบบที่สำคัญคือ เครื่องคอมพิวเตอร์จะต้องมีซอฟต์แวร์เครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 และไมโครซอฟต์ดอทเน็ตเฟรมเวิคเอสดีเคเวอร์ชัน 2.0 (Microsoft .NET Framwork SDK v2.0)

2.3 คุณภาพซอฟต์แวร์ (Software Quality)

เป็นที่ยอมรับแล้วว่าคุณภาพซอฟต์แวร์คือสิ่งสำคัญในกระบวนการพัฒนาซอฟต์แวร์เพราะการมีคุณภาพที่สูงสามารถช่วยลดค่าใช้จ่าย (Cost) ในการทดสอบซอฟต์แวร์การบำรุงรักษา (Maintenance Test) และการนำซอฟต์แวร์กลับมาใช้ใหม่ (Software reusing) (Khosravi และ Guéhéneuc, 2004) แต่ทว่าคุณภาพซอฟต์แวร์มีความหมายและมุมมองที่แตกต่างกันไปสำหรับลูกค้า ผู้ใช้งานในแผนกต่างๆ นักพัฒนา นักทดสอบระบบ ผู้ดูแลรักษาระบบและกลุ่มคนที่คอยสนับสนุนในด้านต่างๆ ทำให้การมองคำว่าคุณภาพซอฟต์แวร์ถูกให้ความสำคัญไปในแต่ละด้านของกลุ่มคนแต่ละประเภท จึงทำให้หลายสถาบันและองค์กรได้ให้คำจำกัดความของคำว่าคุณภาพ

ซอฟต์แวร์และกลุ่มของคุณลักษณะของคุณภาพ (Quality Characteristic หรือ Quality Attributes หรือ Quality Factors) ไว้ดังนี้

- Kitchenham (1989) ได้อธิบายว่า “คุณภาพซอฟต์แวร์เป็นสิ่งที่ยากที่จะกำหนดหรือนิยามลงไป แต่เป็นไปได้ที่จะวัดและทำให้ง่ายต่อการจดจำได้”
- Kan (2000) ได้อธิบายว่า “คุณภาพซอฟต์แวร์ไม่ใช่มุมมองเพียงด้านเดียวแต่ทว่าเป็นมุมมองหลายๆ ด้าน โดยมุมมองของคุณภาพนั้นคือคุณลักษณะของคุณภาพของแต่ละด้าน”

นอกจากนี้ยังมีบางองค์กรได้พยายามพัฒนามาตรฐานการให้นิยามกับคำว่าคุณภาพซอฟต์แวร์ ตัวอย่างเช่น

- ไอเอสโอ 9126 (ISO 9126) (2001)ได้อธิบายว่า “คุณลักษณะของคุณภาพซอฟต์แวร์เป็นกลุ่มของคุณลักษณะในผลิตภัณฑ์ซอฟต์แวร์ ซึ่งใช้อธิบายและประเมินค่าคุณภาพในซอฟต์แวร์ได้”
- เจอมันอินดัสทรีสแทนดาร์ดดีไอเอ็น 55350 พาร์ท 11 (German Industry Standard DIN 55350 Part 11) ได้อธิบายว่า “คุณภาพซอฟต์แวร์ประกอบด้วยทุกๆ คุณลักษณะและลักษณะเฉพาะสำคัญ (Significant Features)ของผลิตภัณฑ์หรือกิจกรรม ที่ซึ่งเกี่ยวข้องกับความพึงพอใจและความต้องการของผู้ใช้”
- เอเอ็นเอสไอสแทนดาร์ด (ANSI Standard) (ANSI/ASQC A3/1978) ได้อธิบายว่า “คุณภาพซอฟต์แวร์คือลักษณะเฉพาะและคุณลักษณะทั้งหมดของผลิตภัณฑ์หรือบริการที่ส่งผลกระทบต่อความสามารถหรือความพึงพอใจในผลิตภัณฑ์หรือบริการนั้น”
- ไออีอีอีสแทนดาร์ด (IEEE Standard) (IEEE Std 729-1983) ได้อธิบายคุณภาพซอฟต์แวร์ไว้ 4 ความหมายคือ (1) “ลักษณะเฉพาะและคุณลักษณะทั้งหมดของผลิตภัณฑ์ ที่ส่งผลกระทบต่อความสามารถหรือความพึงพอใจในผลิตภัณฑ์นั้น” (2) “ระดับของซอฟต์แวร์ซึ่งมองได้จากลักษณะประจำที่รวมอยู่ในซอฟต์แวร์” (3) “ระดับของการคาดหวังของผู้ใช้ (User)หรือลูกค้า (Customer) ในซอฟต์แวร์ที่มาบรรจบกันได้” (4) “การประกอบกันขึ้นของคุณลักษณะของซอฟต์แวร์ที่กำหนดระดับความพอใจจากลูกค้ากับซอฟต์แวร์ที่มาบรรจบกันได้”

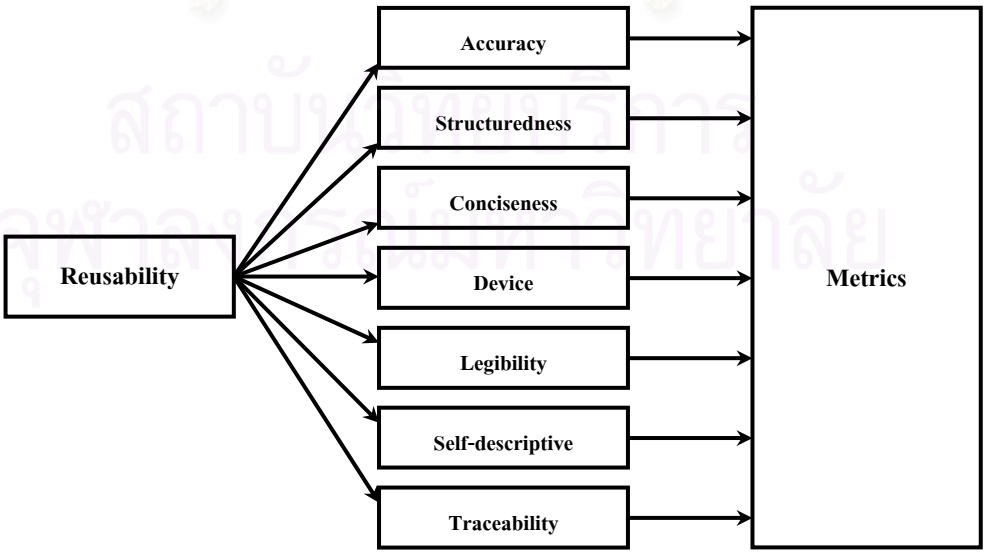
จากคำจำกัดความคุณภาพซอฟต์แวร์ที่กล่าวมาทั้งหมดนี้ทำให้เห็นถึงมุมมองหลายมุมมอง แต่จะสังเกตได้ว่าในคำอธิบายส่วนใหญ่จะกล่าวอ้างถึงคำว่าคุณลักษณะของคุณภาพเนื่องจากการประเมินคุณภาพจำเป็นต้องคำนึงถึงสิ่งที่จะมาวัดค่าคุณภาพ แต่ละด้านและเชื่อมโยงความสามารถ

ในการวัดต่างๆ หรือมาตรวัดให้เข้ากับคุณลักษณะในแต่ละด้านของคุณภาพซอฟต์แวร์โดยคุณภาพที่ถูกนำมาใช้วัดสามารถพิจารณาออกเป็น 2 กลุ่ม (Fenton, 1997) คือ

- คุณลักษณะภายในของคุณภาพ (Internal Quality Characteristics) หมายถึงลักษณะภายในของขั้นตอนการพัฒนาซอฟต์แวร์หรือผลิตภัณฑ์ซอฟต์แวร์ที่สามารถวัดได้โดยตรง เช่น ขนาดของซอร์สโค้ด ความซับซ้อนของซอร์สโค้ดและเวลาในการพัฒนาซอฟต์แวร์ ฯลฯ
- คุณลักษณะภายนอกของคุณภาพ (External Quality Characteristics) หมายถึงลักษณะภายนอกของขั้นตอนการพัฒนาซอฟต์แวร์หรือผลิตภัณฑ์ซอฟต์แวร์ที่สามารถวัดได้โดยอ้อม เช่น ความน่าเชื่อถือ ความสามารถในการบำรุงรักษาและความสามารถในการนำกลับมาใช้ใหม่ ฯลฯ

เมื่อพิจารณาคุณลักษณะภายในของคุณภาพจะเห็นได้ว่า คุณลักษณะภายในของคุณภาพมีผลกระทบหรือส่งผลต่อคุณภาพไม่ทางก็ใดทางหนึ่ง โดยการการวัดและวิเคราะห์คุณลักษณะภายในของคุณภาพได้บ่งบอกถึงระดับคุณลักษณะภายนอกของคุณภาพโดยการวัดในลักษณะนี้มีประโยชน์ที่แตกต่างดังนี้ (1) การใช้คุณลักษณะภายในของคุณภาพเหมาะสมสำหรับการวัดในช่วงแรกเริ่มของการพัฒนาซอฟต์แวร์ (2) การใช้คุณลักษณะภายนอกของคุณภาพเหมาะสมสำหรับการวัดเมื่อผลิตภัณฑ์ (Product) หรือซอฟต์แวร์เสร็จสมบูรณ์ (หรือใกล้เสร็จสมบูรณ์)

รูปที่ 2-30 แสดงแบบจำลองของ McCall (1977)



รูปที่ 2-31 แสดงตัวอย่างแบบจำลองของ McCall (1977) ด้านความสามารถการนำกลับมาใช้ใหม่

จากคุณสมบัติของคุณภาพที่แบ่งออกเป็น 2 กลุ่มนี้ (คุณลักษณะภายในของคุณภาพและคุณลักษณะภายนอกของคุณภาพ) ทำให้มีแนวคิดสร้างแบบจำลองขึ้น คุณลักษณะภายนอกของคุณภาพแต่ละตัวสามารถประกอบไปด้วยคุณลักษณะย่อยหลายตัว เพื่อนำมาใช้ประเมินและสามารถนำไปใช้วัดได้ **แบบจำลองของ McCall** (McCall, 1977) ได้นำเสนอรูปแบบความสัมพันธ์ระหว่างคุณลักษณะของคุณภาพกับมาตรวัดเอาไว้ โดยกำหนดลำดับชั้นออกเป็น (1) คุณลักษณะ (2) เกณฑ์ (Criteria) (3) มาตรวัด ดังจะเห็นได้จากภาพที่ 2-26 แสดงแบบจำลองของ McCall (1977) และภาพที่ 2-27 แสดงตัวอย่างความสัมพันธ์ของความสามารถการนำกลับมาใช้ใหม่จะเห็นได้ว่า

แบบจำลองของ McCall (1977) ซึ่งชี้ให้เห็นว่าถึงการนำคุณลักษณะซึ่งเป็นคุณภาพที่ไม่สามารถวัดได้โดยตรง โดยการสร้างความสัมพันธ์กับเกณฑ์ที่สามารถวัดได้โดยออกแบบมาตรวัดมาใช้วัดเกณฑ์ดังกล่าว ทำให้สามารถบ่งบอกถึงระดับความสามารถในแต่ละด้าน คุณลักษณะภายนอกของคุณภาพ

นอกจากนี้ McCall (1977) ได้นำเสนอคุณลักษณะของคุณภาพไว้ 11 ชนิด ดังนี้

1. **ความถูกต้อง (Correctness)** คือเป็นตัวชี้บอกเกี่ยวกับความสมบูรณ์ของข้อกำหนดซอฟต์แวร์ (Software Specification) ในระหว่างขั้นตอนวิเคราะห์ความต้องการของลูกค้าในกระบวนการพัฒนาซอฟต์แวร์
2. **ความเชื่อถือได้ (Reliability)** คือขอบเขตของซอฟต์แวร์ที่มีความแม่นยำในการทำงานของฟังก์ชัน
3. **ประสิทธิภาพ (Efficiency)** คือจำนวนรวมของทรัพยากรที่ใช้ในการทำงานฟังก์ชันหนึ่งของซอฟต์แวร์
4. **บูรณภาพ (Integrity)** คือขอบเขตการควบคุมการเข้าถึงซอฟต์แวร์หรือข้อมูลโดยผู้ที่ไม่ได้รับอนุญาต (Unauthorized Person)
5. **ความสามารถการใช้งานง่าย (Usability)** คือความสามารถที่ให้ผู้ใช้งานเรียนรู้ ปฏิบัติ เตรียมอินพุตและทำความเข้าใจผลลัพธ์ ของซอฟต์แวร์
6. **ความสามารถการบำรุงรักษา (Maintainability)** คือความพยายามในการติดตั้งและบำรุงรักษาข้อผิดพลาดในการดำเนินการของซอฟต์แวร์
7. **ความสามารถการยืดหยุ่นได้(Flexibility)** คือความยืดหยุ่นในการเปลี่ยนแปลงการดำเนินการของซอฟต์แวร์
8. **ความสามารถการทดสอบ(Testability)** คือความสามารถในการทดสอบระบบเพื่อรับประกันการทำงานของฟังก์ชัน

9. ความสามารถการถ่ายโอน (Portability) คือความสามารถในการย้ายซอฟต์แวร์ไปใช้งานในหลายโครงสร้าง (Platform) เช่น จากโครงแบบฮาร์ดแวร์ (Hardware Configuration) หนึ่งไปยังอีกโครงแบบหนึ่ง

10. ความสามารถการนำกลับมาใช้ใหม่ (Reusability) คือความสามารถของซอฟต์แวร์ที่สามารถนำไปใช้ในซอฟต์แวร์อื่นๆได้

11. ความสามารถการทำงานร่วมกัน (Interoperability) คือความพยายามในการเชื่อมต่อระบบหนึ่งไปยังอีกระบบหนึ่ง

โดยเกณฑ์ที่ McCall (1977) ได้กำหนดให้เข้ากับคุณลักษณะของคุณภาพทั้ง 11 ชนิด ประกอบได้ด้วยเกณฑ์ 22 ชนิด ดังนี้

1. การตรวจสอบ (Auditability) คือความสามารถการตรวจสอบซอฟต์แวร์ให้สอดคล้องกับมาตรฐานได้อย่างสะดวก

2. ความแม่นยำ (Accuracy) คือความสามารถของการคำนวณได้อย่างแม่นยำ

3. มาตรฐานการสื่อสาร (Communication commonality) คือระดับความเป็นมาตรฐานของช่องทางการเชื่อมต่อ (Interface Protocol) และอัตราการส่งและรับข้อมูล (Bandwidth) ที่ถูกใช้

4. ความสมบูรณ์ (Completeness) คือระดับความสมบูรณ์ของซอฟต์แวร์ที่ตรงตามความต้องการผู้ใช้

5. ความซับซ้อน (Complexity) คือระดับความซับซ้อนของซอฟต์แวร์

6. ความกระชับ (Conciseness) คือความกระชับของซอร์สไลท์ออฟโค้ด (Source Line of Code) ในซอฟต์แวร์

7. ความสอดคล้องกัน (Consistency) คือความสอดคล้องกันระหว่างเอกสารการออกแบบและซอฟต์แวร์ที่พัฒนา รวมถึงความสอดคล้องที่เกิดขึ้นตลอดทั้งกระบวนการพัฒนาซอฟต์แวร์

8. มาตรฐานข้อมูล (Data commonality) คือการใช้มาตรฐานของโครงสร้างข้อมูลและชนิดของข้อมูลตลอดทั้งกระบวนการพัฒนาซอฟต์แวร์

9. มาตรการรองรับข้อผิดพลาด(Error tolerance) คือการรองรับความเสียหายที่เกิดขึ้นเมื่อซอฟต์แวร์เกิดความผิดพลาดขึ้น

10. ประสิทธิภาพขณะทำงาน (Execution efficiency) คือประสิทธิภาพขณะซอฟต์แวร์ทำงาน (Run - Time)

11. ความสามารถการขยายต่อ (Expandability คือระดับความสามารถการขยายโครงสร้างซอฟต์แวร์ข้อมูลในซอฟต์แวร์หรือรูปแบบการออกแบบขั้นตอนซอฟต์แวร์

12. ความเป็นสากล (Generality) คือความสามารถของซอฟต์แวร์ว่าสามารถรองรับการใช้งานที่กว้างเพียงใด

13. มาตรฐานการรองรับฮาร์ดแวร์ (Hardware independence) คือระดับความเป็นอิสระของซอฟต์แวร์จากฮาร์ดแวร์

14. การใช้เครื่องมือ (Instrumentation) คือระดับความสามารถของการดูแลหรือเฝ้าติดตามซอฟต์แวร์ในขณะทำงานและสามารถกำหนดความผิดพลาดที่เกิดขึ้นได้

15. สภาพเป็นส่วนจำเพาะ (Modularity) คือความสามารถของซอฟต์แวร์ที่เป็นลักษณะของส่วนประกอบที่ทำงานแยกอิสระกัน

16. การดำเนินการ (Operability) คือความสามารถในการทำงานของซอฟต์แวร์

17. ความมั่นคง (Security) คือความสามารถในการควบคุมกลไกของการควบคุมหรือป้องกันซอฟต์แวร์และข้อมูลของซอฟต์แวร์

18. ความเข้าใจเอกสารตัวเอง (Self – documentation) คือการที่ซอร์สโค้ดสามารถทำความเข้าใจได้เหมือนเอกสารหรือคู่มือของซอฟต์แวร์

19. ความง่าย (Simplicity) คือระดับความสามารถในการทำความเข้าใจซอฟต์แวร์ได้

20. ความเป็นอิสระต่อระบบซอฟต์แวร์ (Software system independence) คือระดับความสามารถของซอฟต์แวร์ที่มีความเป็นอิสระต่อสภาพแวดล้อมของซอฟต์แวร์

21. ความสามารถการติดตาม (Traceability) คือความสามารถการบอกหรือการแกะรอยซอฟต์แวร์จากแบบหรือส่วนประกอบของซอฟต์แวร์ เพื่อสามารถบอกได้ถึงความต้องการของผู้ใช้

22. ความสามารถการฝึกฝน (Training) คือระดับความสามารถช่วยเหลือให้ผู้ใช้ใหม่ๆ เข้าใจการใช้งานซอฟต์แวร์

ความสัมพันธ์ระหว่างคุณลักษณะของคุณภาพกับเกณฑ์เหล่านี้สามารถแสดงในตารางที่ 2-1 โดยแสดงลักษณะการขึ้นต่อกันได้ดังนี้

Quality Factor / Quality Metric	Correctness	Reliability	Efficiency	Integrity	Maintainability	Flexibility	Testability	Portability	Reusability	Interoperability	Usability
Auditability				×			×				
Accuracy		×								×	
Communication commonality											
Completeness	×										
Complexity		×				×	×				
Concision			×		×	×					
Consistency	×	×			×	×					
Data commonality										×	
Error tolerance		×									
Execution efficiency			×								

ตารางที่ 2-1 แสดงตารางความสัมพันธ์ระหว่างคุณลักษณะของคุณภาพและมาตรวัดคุณภาพของ McCall (1977)

Expandability						✕					
Generality						✕		✕	✕	✕	
Hardware Independence								✕	✕		
Instrumentation				✕	✕		✕				
Modularity		✕			✕	✕	✕	✕	✕	✕	
Operability			✕								✕
Security				✕							
Self – documentation					✕	✕	✕	✕	✕		
Simplicity		✕			✕	✕	✕				
System Independence								✕	✕		
Traceability	✕										
Training											✕

ตารางที่ 2-1 แสดงตารางความสัมพันธ์ระหว่างคุณลักษณะของคุณภาพและมาตรวัดคุณภาพของ McCall (1977) (ต่อ)

2.4 มาตรวัดเชิงวัตถุ (Object-Oriented Metrics)

มาตรวัดซอฟต์แวร์ (Software Metric) คือกลุ่มของข้อกำหนดหรือเงื่อนไขที่ใช้ตรวจสอบซอฟต์แวร์ที่สร้างขึ้น โดยมาตรวัด (Metric) เป็นส่วนหนึ่งของการวัด (Measurement) โดยถูกมองเหมือนเป็นตัวชี้วัดคุณภาพ (Quality Indicator) ของระบบซอฟต์แวร์กล่าวคือมาตรวัดเป็นเครื่องสะท้อนคุณภาพซอฟต์แวร์ตลอดจนเป็นแนวทางที่ใช้ในการออกแบบ เพื่อให้ได้ระบบซอฟต์แวร์ที่ทำงานอย่างมีประสิทธิภาพในทุกๆ ส่วนของระบบ (Fenton, 1997) เมื่อต้องการให้ระบบซอฟต์แวร์ทำงานได้อย่างมีประสิทธิภาพจำเป็นต้องตรวจสอบซอฟต์แวร์หรือคอยดูแลปัญหาอยู่ตลอด ทั้งการตรวจสอบหรือการคอยดูแลจะต้องใช้มาตรวัดซึ่งเป็นตัวชี้วัดคุณภาพหรือสุขภาพของซอฟต์แวร์ได้เป็นอย่างดี จะทำให้สามารถควบคุมคุณภาพซอฟต์แวร์ได้อย่างมีประสิทธิภาพ โดย DeMarco (1982) ได้กล่าวว่า “จะไม่สามารถควบคุมสิ่งใดได้ถ้าสิ่งนั้นไม่สามารถวัดได้” จากจุดนี้เองสื่อให้เห็นว่าถ้าต้องการจะทราบถึงคุณภาพของสิ่งใด สิ่งนั้นจำเป็นต้องวัดได้ จำเป็นต้องมีมาตรวัดซึ่งสามารถเข้าไปวัดสิ่งนั้นได้อย่างมีประสิทธิภาพ โดยมาตรวัดซอฟต์แวร์ถูกนำไปใช้วัดซอฟต์แวร์ 3 ประเภท (Fenton, 1997) (1) ผลิตภัณฑ์ซอฟต์แวร์ (Software Product) (2) กระบวนการพัฒนาซอฟต์แวร์ (Software Process) (3) ผู้พัฒนาซอฟต์แวร์ (Software People) และถ้ามาตรวัดซอฟต์แวร์ถูกใช้อย่างเหมาะสมมาตรวัดซอฟต์แวร์สามารถช่วยเพิ่มความสามารถดังนี้ (1) สามารถกำหนดหรือวัดระดับความสำเร็จและความล้มเหลวสำหรับผลิตภัณฑ์ กระบวนการพัฒนาและผู้พัฒนาได้ (2) สามารถบอกระดับสิ่งที่ดีขึ้นและสิ่งที่ยังต้องการการปรับปรุงในผลิตภัณฑ์ กระบวนการพัฒนาและผู้พัฒนาได้ (3) ช่วยให้ผู้จัดการโครงการ (Project manager) ตัดสินใจในการทำงานได้ดีขึ้น (4) ช่วยวิเคราะห์หาแนวโน้มข้อผิดพลาดในระบบได้

การใช้มาตรวัดเพื่อวัดสิ่งต่างๆ นั้นเป็นพื้นฐานของวิศวกรรมทั่วไปไม่เว้นกระทั่งวิศวกรรมซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Engineering) โดยมาตรวัดในวิศวกรรมซอฟต์แวร์ (Software Engineering) ช่วยให้นักพัฒนาซอฟต์แวร์ (Software Developer) ความเข้าใจถึงประสิทธิภาพของกระบวนการพัฒนาซอฟต์แวร์หรือผลิตภัณฑ์ซอฟต์แวร์ และช่วยให้เข้าใจถึงวัตถุประสงค์การประเมินค่า (Evaluation) โดยสามารถแบ่งวัตถุประสงค์พื้นฐานของอ็อบเจกต์-โอเรนเตด ซอฟต์แวร์ เอนจิเนียร์ริง (Object-Oriented Software Engineering) (Pressman, 2001) ได้ดังนี้ (1) ทำให้เข้าใจคุณภาพของซอฟต์แวร์ได้ดีขึ้น (2) ประเมินประสิทธิภาพกระบวนการพัฒนาซอฟต์แวร์ได้ (3) ช่วยปรับปรุงคุณภาพงานของแต่ละขั้นตอนในกระบวนการพัฒนาซอฟต์แวร์ โดยวัตถุประสงค์ในแต่ละข้อล้วนมีความสำคัญ แต่สำหรับวิศวกรรมซอฟต์แวร์คุณภาพของผลิตภัณฑ์หรือคุณภาพของซอฟต์แวร์นั้นมีความสำคัญสูงสุด แต่สิ่งที่ได้มาซึ่งคุณภาพของซอฟต์แวร์คือการมี

กระบวนการพัฒนา ซอฟต์แวร์ที่มีคุณภาพ แต่การวัดคุณภาพของระบบเชิงวัตถุมีข้อแตกต่างจากการวัดคุณภาพซอฟต์แวร์โดยทั่วไป

โดย Berard (1993) ได้กำหนด 5 คุณลักษณะที่เป็นลักษณะเฉพาะของมาตรวัดสำหรับระบบเชิงวัตถุเป็นข้อแตกต่างของมาตรวัดสำหรับวิศวกรรมซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Engineering) ซึ่งประกอบไปด้วย

1. **โลคอลไลเซชัน (Localization)** คือคุณลักษณะของซอฟต์แวร์ที่แสดงถึงชนิดของข้อมูลในโปรแกรม เช่น คุณสมบัติการห่อหุ้มทั้งข้อมูล (Data) และกระบวนการพัฒนาในขอบเขตของคลาส

2. **เอ็นแคบซูเลชัน (Encapsulation)** คือการห่อหุ้มกลุ่มของข้อมูล สำหรับคุณสมบัติ Encapsulation ในซอฟต์แวร์เชิงวัตถุจะอธิบายในรูปแบบของคลาสซึ่งประกอบไปด้วยลักษณะประจำกับเมท็อดที่เป็นข้อมูลอยู่ภายใน

3. **อินฟอร์เมชันไฮดิง (Information Hiding)** คือการซ่อนข้อมูลและการดำเนินการของระบบ โดยเปิดเผยให้เฉพาะส่วนประกอบ (Component) ที่จำเป็นต้องใช้เท่านั้นเพราะการออกแบบระบบเชิงวัตถุที่ดีจำเป็นต้องคงคุณสมบัติข้อนี้ไว้

4. **อินเฮอริแทนซ์ (Inheritance)** คือกลไกที่ว่าด้วยการเปิดให้วัตถุหนึ่งสามารถถ่ายทอดคุณสมบัติของระบบเชิงวัตถุไปยังวัตถุอื่นๆ ได้ โดยการสืบทอด (Inheritance) จะทำอยู่ในรูปแบบการสืบทอดของคลาสเป็นระดับๆ ไป

5. **แอบสแทรกชัน (Abstraction)** คือกลไกที่ว่าด้วยการสร้างแม่แบบที่มีเฉพาะรายละเอียดข้อมูลที่จำเป็นต่อระบบหรือส่วนประกอบนั้นเท่านั้น

การวัดเชิงวัตถุด้วยมาตรวัดเชิงวัตถุนั้นได้มีแนวคิดที่นำคุณลักษณะที่ได้กล่าวข้างต้นมาพัฒนาเป็นมาตรฐานและได้ถูกนำไปใช้ในงานวิจัยและในเครื่องมือ (Tools) ที่ใช้พัฒนาซอฟต์แวร์ โดยจะขอยกตัวอย่างดังนี้

1. **เดอะซีเคเมทริกสูท (The CK Metric Suite)**

มาตรวัดเชิงวัตถุที่ถูกนำไปใช้อย่างกว้างขวางคือมาตรวัดของ Chidamber และ Kemerer (1994) จะวัดค่าจากคุณสมบัติของคลาส เช่น การสืบทอด การเชื่อมต่อ และการทำงานร่วมกัน เป็นต้น ซึ่งประกอบไปด้วย 6 มาตรวัด คือ

1.1. เวทเทดเมท็อดเปอคลาส (Weighted Methods per Class) (WMC): เป็นมาตรวัดที่วัดจากความซับซ้อนของแต่ละคลาสโดยสามารถวัดได้ดังนี้

$$WMC = \sum_{i=1}^{n} c_i$$

โดย c_i คือค่าความซับซ้อนสำหรับคลาส

n คือจำนวนของ Method ในคลาสนั้น

1.2. เดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree) (DIT): เป็นมาตรวัดที่วัดจากจำนวนของระดับชั้นความลึกมากที่สุดที่เกิดความสัมพันธ์การสืบทอดของคลาสที่อยู่ในระบบ

1.3. นัมเบอร์ออฟชิลเดรน (Number of Children) (NOC): เป็นมาตรวัดที่วัดจากจำนวนของคลาสลูกของคลาสนั้น

1.4. คัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes) (CBO): เป็นมาตรวัดที่วัดจากจำนวนการเชื่อมต่อกันระหว่างคลาสหนึ่งกับอีกคลาสหนึ่งภายในระบบ โดยจะนับจากจำนวนการเรียกใช้เมท็อดหรือการใช้ตัวแปรของอีกคลาสหนึ่ง โดยไม่นับคลาสลูก โดยหาได้จาก

$$CBO = \frac{\sum Arcs}{n}$$

โดย อาร์ค (Arcs) คือจำนวนของความสัมพันธ์ที่มีของอ็อบเจกต์เดี่ยว (Single Object) ใดๆ ในระบบกับอ็อบเจกต์อื่นๆ

n คือจำนวนอ็อบเจกต์ทั้งหมด

1.5. เรสปอนซ์ฟอร์อะคลาส (Response for a Class) (RFC): เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดของคลาสแม่ที่สามารถเข้าถึงหรือใช้งานได้จากคลาสลูก

1.6. แลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method) (LCOM): เป็นมาตรวัดที่วัดจากจำนวนเมท็อดแต่ละคู่ในคลาสเดียวกันว่ามีการเรียกใช้ลักษณะประจำร่วมกันหรือไม่ ดังนี้

LCOM = |P| - |Q| เมื่อ |P| > |Q| และเมื่อ |P| <= |Q| ให้ LCOM = 0

โดยที่ |P| คือจำนวนผลของการอินเทอเซกชัน (Intersection) ของเมท็อดกันแล้วได้เท่ากับเซตว่าง (Null Set)

|Q| คือจำนวนผลของการอินเทอเซกชัน (Intersection) ของเมท็อดกันแล้วได้ไม่เท่ากับเซตว่าง

ตัวอย่างเช่น กำหนดให้

ลักษณะประจำ: *1, 2, 3, 4*

เมท็อด: *A, B, C, D*

เมท็อด *A* เรียกใช้ลักษณะประจำ*1* และ *4*

เมท็อด *B* ไม่มีการเรียใช้งานลักษณะประจำใดๆ

เมท็อด *C* เรียกใช้ลักษณะประจำ *2* และ *4*

เมท็อด *D* เรียกใช้ลักษณะประจำ *3*

ดังนั้นจะได้ว่า

A อินเทอเซกชัน *B* = เซตว่าง

A อินเทอเซกชัน *C* = *4*

A อินเทอเซกชัน *D* = เซตว่าง

B อินเทอเซกชัน *C* = เซตว่าง

B อินเทอเซกชัน *D* = เซตว่าง

C อินเทอเซกชัน *D* = เซตว่าง

ดังนั้นจาก LCOM = |P| - |Q|

|P| = 5

|Q| = 1

LCOM = 5-1 = 4

นอกจากนี้มาตรวัดแลคออฟโคฮีชันอินเมท็อดยังได้ถูกนำไปพัฒนาต่อในอีกรูปแบบ โดย Li และ Henry (1993) ได้แนะนำว่าแลคออฟโคฮีชันอินเมท็อดจะมีค่าเท่ากับจำนวนกลุ่ม (Set) ของเมท็อดที่ไม่มีการใช้ลักษณะประจำรวมกันอยู่เลย จากตัวอย่างเดียวกัน สามารถคำนวณค่าแลคออฟโคฮีชันอินเมท็อดได้ดังนี้

ลักษณะประจำ: *1, 2, 3, 4*

เมท็อด: *A, B, C, D*

เมท็อด *A* เรียกใช้ลักษณะประจำ *1* และ *4*

เมท็อด *B* ไม่มีการเรียกใช้งานลักษณะประจำใดๆ

เมท็อด *C* เรียกใช้ลักษณะประจำ *2* และ *4*

เมท็อด *D* เรียกใช้ลักษณะประจำ *3*

ดังนั้นจะได้ว่า

เซตของเมท็อดที่ไม่เกี่ยวข้องกันหรือไม่ได้ใช้ลักษณะประจำร่วมกันมีดังนี้

{A,C}, {B}, {D}

ดังนั้น LCOM = 3

สรุปคือไม่ว่าจะใช้วิธีของ CK (1994) หรือ Li และ Henry (1993) การทำให้การทำงานร่วมกันมีค่าต่ำๆ จะบ่งบอกถึงหน้าที่ที่ไม่เกี่ยวข้องกันอยู่มาก ผลคือทำให้ยากในการทำความเข้าใจและนำไปใช้งาน

2. มาตรวัดที่นำเสนอโดยลอเรนซ์และคิดด์ (Metric Proposed by Lorenz and Kidd)

มาตรวัดเชิงวัตถุของลอเรนซ์และคิดด์ (1994) ได้ถูกออกแบบให้วัดค่าโดยตรงจากตัวคลาส เช่น การนับจำนวนเมท็อดการนับจำนวนเมท็อดที่ถูกสืบทอดโดยคลาสลูก เป็นต้น ซึ่งประกอบไปด้วยมาตรวัด ดังนี้

2.1 นัมเบอร์ออฟพับบลิคเมท็อด (Number of Public Methods) (PM) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดที่เป็นสาธารณะ (Public) ทั้งหมดในคลาสนั้น

2.2 นัมเบอร์ออฟเมท็อด (Number of Methods) (NM) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดในคลาสนั้น โดยนับรวมทั้งเมท็อดที่เป็นสาธารณะ ส่วนบุคคล (Private) และป้องกัน (Protected)

2.3 นัมเบอร์ออฟพับบลิคแวริเอเบิลเปอคลาส (Number of Public Variables per Class) (NPV) เป็นมาตรวัดที่วัดจากจำนวนของตัวแปรที่เป็นสาธารณะทั้งหมดในคลาสนั้น

2.4 นัมเบอร์ออฟแวริเอเบิลเปอคลาส (Number of Variables per Class) (NV) เป็นมาตรวัดที่วัดจากจำนวนของตัวแปรทั้งหมดในคลาสนั้น โดยนับรวมทั้งสาธารณะ ส่วนบุคคลและป้องกัน

2.5 นัมเบอร์ออฟเมท็อดอินเฮอริทเททบายอะซับคลาส (Number of Methods Inherited by a subclass) (NMI) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดที่ถูกสืบทอดโดยคลาสลูก

2.6 นัมเบอร์ออฟเมท็อดโอเวอร์ไรเดนบายอะซับคลาส (Number of Methods Overridden by a subclass) (NMO) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดที่อยู่ในคลาสลูกแล้วถูกให้ชื่อเดียวกับคลาสแม่หรือเรียกว่าโอเวอร์ไรเดน (Overridden)

2.7 นัมเบอร์ออฟเมท็อดแอดเดดบายอะซับคลาส (Number of Methods Added by a subclass) (NMA) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดที่ถูกเพิ่มขึ้นในคลาสลูก

2.8 เอเวอร์เรจเมท็อดไซซ์ (Average Method Size) (AMS) เป็นมาตรวัดที่วัดจากขนาดในคลาส โดยคำนวณจากจำนวนบรรทัดในซอร์สโค้ดที่ไม่นับรวมคำอธิบาย (Comment) และส่วนที่เว้นว่างไว้ แล้วนำมาหารด้วยจำนวนเมท็อดทั้งหมดในคลาส

2.9 นัมเบอร์ออฟไทม์อะคลาสอีสรียูส (Number of times a Class is Reused) (NCR) เป็นมาตรวัดที่วัดจากจำนวนครั้งของคลาสที่เรียกใช้

2.10 นัมเบอร์ออฟเฟรนออฟอะคลาส (Number of Friends of a class) (NF) เป็นมาตรวัดที่วัดจากจำนวนของคลาสเพื่อน (Friend Class) ในแต่ละคลาส โดยคลาสเพื่อนเป็นคลาสที่ถูกจัดให้สามารถเรียกใช้เมท็อดหรือตัวแปร ที่เป็นส่วนบุคคลและป้องกันได้

3. เดอะเอ็มโอโอดีเมทริกสูท (The MOOD Metric Suite)

มาตรวัดเชิงวัตถุโดย Abreu (1996) ประกอบไปด้วยมาตรวัด ดังนี้

3.1 เมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor) (MHF) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดของทุกๆ คลาสในระบบว่ามีเมท็อดที่เป็นส่วนบุคคลและป้องกัน ทั้งนี้ไม่นับรวมเมท็อดที่สืบทอดมา

$$MHF = \frac{\sum_{i=1}^{TC}\sum^{M_d(C_i)}\left(1-V\left(M_{mi}\right)\right)}{\sum_{i=1}^{TC} M_d\left(C_i\right)}$$

โดยที่

$$V\left(M_{mi}\right) = \frac{\sum_{j=1}^{TC} is\_visible\left(M_{mi}, C_j\right)}{TC}$$

$$is\_visible\left(M_{mi}, C_j\right) = \begin{cases} 1 \\ 0 \end{cases}$$ เป็น 1 เมื่อมี C_j ที่สามารถเรียกใช้ M_{mi} ได้

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

M คือ เมท็อดในแต่ละคลาส

3.2 แอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor) (AHF) เป็นมาตรวัดที่วัดจากจำนวนของลักษณะประจำทั้งหมดของทุกๆ คลาสในระบบว่ามีลักษณะประจำที่เป็นส่วนบุคคลและป้องกัน ทั้งนี้ไม่นับรวมลักษณะประจำที่สืบทอดมา

$$AHF = \frac{\sum_{i=1}^{TC}\sum^{A_d(C_i)}\left(1-V\left(A_{mi}\right)\right)}{\sum_{i=1}^{TC} A_d\left(C_i\right)}$$

โดยที่

$$V(A_{mi}) = \frac{\sum_{j=1}^{TC} is\_visible(A_{mi}, C_j)}{TC}$$

$$is\_visible(A_{mi}, C_j) = \begin{cases} 1 \\ 0 \end{cases}$$ เป็น 1 เมื่อมี C_j ที่สามารถเรียกใช้ A_{mi} ได้

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

A คือ ลักษณะประจำในแต่ละคลาส

3.3 เมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor) (MIF) เป็นมาตรวัดที่วัดจากจำนวนของเมท็อดทั้งหมดของทุกๆ คลาสในระบบว่ามีเมท็อดใดที่เป็นเมท็อดที่ถูกสืบทอดมา

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

โดยที่

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

M คือ เมท็อดในแต่ละคลาส

3.4 แอตทริบิวต์อินเฮอริแทนแฟคเทอร์ (Attribute Inheritance Factor) (AIF) เป็นมาตรวัดที่วัดจากจำนวนของลักษณะประจำทั้งหมดของทุกๆ คลาสในระบบว่ามีลักษณะประจำใดที่เป็นลักษณะประจำที่ถูกสืบทอดมา

$$AIF = \frac{\sum_{i=1}^{TC} A_i(C_i)}{\sum_{i=1}^{TC} A_a(C_i)}$$

โดยที่

$$A_a(C_i) = A_d(C_i) + A_i(C_i)$$

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

A คือ ลักษณะประจำในแต่ละคลาส

3.5 โพลีมอร์ฟิซึมแฟคเทอร์ (Polymorphism Factor) (POF) เป็นมาตรวัดที่วัดจากจำนวนเมท็อดของทุกๆ คลาสในระบบว่ามีเมท็อดที่เป็นเมท็อดแบบโอเวอร์ไรเดน

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

โดยที่

$$M_d(C_i) = M_n(C_i) + M_o(C_i)$$

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

M คือ เมท็อดในแต่ละคลาส

3.6 คัพพลิงแฟคเทอร์ (Coupling Factor) (COF) เป็นมาตรวัดที่วัดจากจำนวนของการเชื่อมต่อของทุกๆ คลาส

$$COF = \frac{\sum_{i=1}^{TC} \left[\sum_{j=1}^{TC} is\_client(C_i, C_j) \right]}{TC^2 - TC}$$

โดยที่

$TC^2 - TC$ คือจำนวนการเชื่อมต่อในระบบทั้งหมด

$$is\_client(C_c, C_s) = \begin{cases} 1 & \text{เป็น 1 เมื่อมี } C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 & \end{cases}$$

โดยกำหนดให้

TC คือจำนวนรวมของคลาสทั้งหมด

C คือ คลาสในระบบ

C_s คือ คลาสแม่ข่าย

C_c คือ คลาสลูกข่าย

นอกจากมาตรวัดเหล่านี้แล้วยังมีมาตรวัดที่มีความสัมพันธ์กับคุณสมบัติโดยตรงของซอฟต์แวร์เชิงวัตถุ เช่น ความซับซ้อน การเชื่อมต่อและการทำงานร่วมกัน ดังนี้

4. ไซโครเมทริกคอมเพลกซิตี้ (McCabe's Cyclomatic Complexity Measure)

การวัดความซับซ้อนของโปรแกรมของ McCabe (1989) จะวัดจากจำนวนไซโครเมทริก (Cyclomatic) ซึ่งเกิดจากขั้นตอนการทำงานของโปรแกรม โดยสามารถคำนวณได้ดังนี้

$$v(F) = e - n + 2$$

โดยที่

e คือ อาร์ค (Arcs) หรือขั้นตอนของโปรแกรม เช่น เงื่อนไขอีฟ (if) สามารถแบ่งได้ออกเป็น 2 อาร์ค

n คือจำนวนปม (Node) ในโปรแกรมนั้น

ทั้งนี้การนำมาตรวัดการวัดค่าไซโครเมทริกคอมเพลกซิตี้ (Cyclomatic Complexity) ของ McCabe (1987) มาใช้กับซอฟต์แวร์เชิงวัตถุในปัจจุบันสามารถทำได้ โดยการนำไปวัดค่าความซับซ้อนของขั้นตอนวิธี (Algorithm) ในแต่ละเมท็อด (Rosenberg และ Hyatt, 1995)

5. คอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage)

เป็นมาตรวัดที่สะท้อนจำนวนสัดส่วนของคำอธิบาย (Comment) ทั้งหมดที่ถูกเขียนไว้ในซอร์สโค้ดซึ่งจะบอกได้ถึงความช่วยเหลือที่มีให้กับนักพัฒนาซอฟต์แวร์หรือผู้ที่เข้ามาการทบทวนซอร์สโค้ดเพื่อตรวจสอบและในกรณีที่เป็นการบำรุงรักษา ปรับปรุงแก้ไขซอร์สโค้ดตาม ความต้องการของลูกค้าที่เปลี่ยนแปลงไป โดยเปอร์เซ็นต์ของคำอธิบายที่เกิดขึ้นจะแปรผันตรงกับความสามารถในการทำความเข้าใจการทำงานของเมท็อดนั้นๆ (Rosenberg และ Hyatt, 1995) โดยมีวิธีคำนวณดังนี้

$$Comment(\%) = \frac{Comment_t}{SLOC_t - SLOC_b}$$

โดยที่

$Comment_t$ คือจำนวนคำอธิบายทั้งหมด

$SLOC_t$ คือจำนวนบรรทัดของซอร์สโค้ดทั้งหมด

$SLOC_b$ คือจำนวนบรรทัดของซอร์สโค้ดที่เป็นบรรทัดว่างทั้งหมด

จากมาตรวัดเชิงวัตถุทั้งหมด 24 มาตรวัด สำหรับงานวิจัยนี้ผู้วิจัยเลือกมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995) เนื่องจากมาตรวัดเชิงวัตถุอีก 10 มาตรวัด มีการสื่อความหมายไปในทางเดียวกันกับมาตรวัดอื่นๆ ที่กล่าวมาแล้ว (Harrison, R. และคณะ, 1997) สำหรับงานวิจัยนี้สามารถวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์คือ ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability) โดยอ้างอิงตามงานวิจัยของ Quenel และ Lövdahl (2004) ซึ่งได้กล่าวถึงผลการวิเคราะห์มาตรวัดเชิงวัตถุดังกล่าวกับคุณภาพซอฟต์แวร์ไว้ โดยสามารถแจกแจงตามแบบจำลองของ McCall (1977) โดย ได้ดังนี้

มาตรวัดเชิงวัตถุ / คุณภาพซอฟต์แวร์	Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
Maintainability	-	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓
Reusability	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-	✓
Understandability	✓	✓	-	-	-	-	-	✓	-	-	-	-	-	-

ตารางที่ 2-2 แสดงตารางแบบจำลองความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์ที่เลือกใช้ในงานวิจัยนี้ (Quenel และ Lövdahl (2004))

หมายเหตุ ✓ หมายถึงมาตรวัดเชิงวัตถุมีความสัมพันธ์กับคุณภาพซอฟต์แวร์

\- หมายถึงมาตรวัดเชิงวัตถุไม่มีความสัมพันธ์กับคุณภาพซอฟต์แวร์

2.5 งานวิจัยที่เกี่ยวข้อง

ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์ โดยใช้มาตรวัดเชิงวัตถุ มีงานวิจัยที่เกี่ยวข้องดังนี้

Tourwé และ Mens (2003) ได้จัดกลุ่มสำรวจงานวิจัยที่มีอยู่ในปัจจุบันที่เกี่ยวข้องกับกระบวนการรีแฟคทอริงโดยจัดกลุ่มเป็นกลุ่มๆ ดังนี้ (1) กลุ่มงานวิจัยด้านทฤษฎีทางซอฟต์แวร์ต่างๆ ที่เกี่ยวข้องกับกระบวนการรีแฟคทอริง เช่น การใช้โปรแกรมสไลด์ซิง (Program slicing) เป็นการนำวิธีการจากโปรแกรมสไลด์ซิงมาใช้ยืนยันว่าการทำกระบวนการรีแฟคทอริงจะคงไว้ซึ่งพฤติกรรมที่วัตถุนั้นแสดงไว้ (Lanubile และ Visaggio, 1997) หรือการนำเทคนิคกราฟทรานฟอร์มเมชัน (Graph transformations) มาใช้สนับสนุนการทำกระบวนการรีแฟคทอริง โดยแนะนำการออกแบบที่ไม่ดีในโปรแกรม (Program) (Mens และคณะ, 2002) (2) กลุ่มงานวิจัยด้านเทคนิค (Techniques) ในซอฟต์แวร์ที่นำมาใช้ในกระบวนการรีแฟคทอริง เช่น การนำซอฟต์แวร์วิชวลไลเซชัน (Software Visualization) มาช่วยแสดงผลส่วนของโปรแกรม (Program) ที่ควรทำกระบวนการรีแฟคทอริง (Lanza และ Ducasse, 2002) (3) กลุ่มงานวิจัยด้านภาษา (Languages) ที่สนับสนุนการทำกระบวนการรีแฟคทอริง เช่น ภาษาฟอร์แทรน (Fortran) (Bodin และ Sage, 1994) ภาษาซีพลัสพลัส (C+ +) (Opdyke, 1999) และภาษาจาวา (Java) (Fowler, 1999) ฯลฯ นอกจากนี้ยังกล่าวถึงภาษาการออกแบบซอฟต์แวร์เชิงวัตถุ เช่น ยูเอ็มแอล (UML) (4) กลุ่มงานวิจัยด้านเครื่องมือที่สนับสนุนการทำกระบวนการรีแฟคทอริง ได้แก่ รีแฟคทอริงบราวเซอร์ (Refactoring Browser) (Roberts และคณะ, 1997) เจแฟคเทอร์ (jFactor) (Instantiations, 2002) และเอ็กรีแฟคทอรี (XRefactory) (XRef-Tech, 2002) เป็นต้น และเครื่องมือที่ใช้พัฒนาซอฟต์แวร์แล้วรวมกระบวนการรีแฟคทอริงไว้ให้เลือกใช้ ได้แก่ สมอลทอล์ทวิชวลเวิคเวอร์ชัน 7 (Smalltalk VisualWorks v7) (CinCom, 2002) อีคริปซ์เวอร์ชัน 2 (Eclipse v2) (eclipse.org, 2002) ทูเกตเธอร์คอนโทรลเซ็นเตอร์เวอร์ชัน 6 (Together ControlCenter v6) (TogetherSoft, 2002) อินเทลไลเจไอดีอีเอเวอร์ชัน 3 (IntelliJ IDEA v3) (IntelliJ, 2002) และบอร์แลนเจบิวเดอร์เวอร์ชัน 7 (Borland JBuilder v7) (Borland, 2002) Tourwé และ Mens (2003) ยังได้กล่าวถึงงานวิจัยในอนาคตที่ควรตระหนักแล้วนำมาวิจัย ตัวอย่างเช่น วิเคราะห์ความสัมพันธ์ระหว่างกระบวนการรีแฟคทอริงการเปรียบเทียบความเหมาะสมของเครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริงการนำกระบวนการรีแฟคทอริงไปประยุกต์ใช้ในภาษาการออกแบบซอฟต์แวร์เชิงวัตถุและผลกระทบที่เกิดจากกระบวนการรีแฟคทอริงกับคุณภาพซอฟต์แวร์

Quenel และ Lövdahl (2004) ได้วิจัยมาตรวัดที่นำมาใช้ในการพัฒนาซอฟต์แวร์เชิงวัตถุ โดยกล่าวถึงผลกระทบต่อคุณภาพซอฟต์แวร์ เช่น ประสิทธิภาพของซอฟต์แวร์ (Efficiency) ความ

ซับซ้อนของซอฟต์แวร์ (Complexity) ความสามารถในการทำความเข้าใจ (Understandability) ความสามารถของการนำมาใช้ใหม่ (Reusability) ความสามารถการทดสอบ (Testability) และความสามารถในการบำรุงรักษา (Maintainability) โดยงานวิจัยนี้ได้สำรวจและนำเสนอมาตรวัดโดยแบ่งเป็น 2 กลุ่มดังนี้ (1) มาตรวัดทั่วไป (Traditional Metrics) เช่น มาตรวัดความซับซ้อนของขั้นตอนวิธี (Algorithm) (Cyclomatic Complexity) มาตรวัดขนาด (Size) และคอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) (2) มาตรวัดเชิงวัตถุ (Object-Oriented Metrics) เช่น มาตรวัดของ Chidamber และ Kemerer (1994) และมาตรวัดของ Abreu (1996)

Simon และคณะ (2001) ได้กล่าวถึงมาตรวัดที่นำมาวัดกระบวนการรีแฟคทอริง โดยได้เลือกกระบวนการรีแฟคทอริงมา 4 วิธีคือ มูฟเมท็อด มูฟฟิลด์ เอ็กแทรกคลาสและอินไลน์คลาส โดยนำเทคนิคซอฟต์แวร์วิชวลไลเซชันมาช่วยในการวิเคราะห์เมท็อดหรือลักษณะประจำ โดยวัดจากความห่างหรือระยะทางของแต่เมท็อดหรือลักษณะประจำว่ามีความห่างออกจากกลุ่มหรือไม่ เช่น คลาส A มีเมท็อด A1 และ A2 และมีลักษณะประจำ A1 และ A2 ส่วนคลาส B มีเมท็อด B1 และมีลักษณะประจำ B1 และ B2 เมื่อทำการทดลองแล้วสามารถสังเกตระยะห่างจากเทคนิคซอฟต์แวร์วิชวลไลเซชันได้ว่าเมท็อด A1 มีความห่างออกจากกลุ่มของคลาส A แล้วไปอยู่ในกลุ่มของคลาส B ทำให้วิเคราะห์ได้ว่าควรใช้กระบวนการรีแฟคทอริงวิธีมูฟเมท็อดจากคลาส A ไปยังคลาส B โดยค่าที่ใช้คำนวณระยะห่างในกราฟฟิกมาจากการคำนวณมาตรวัดเชิงวัตถุ เช่น มาตรวัดของ Chidamber และ Kemerer (1994)

Bois และ Mens (2003) นำเสนอผลกระทบจากการทำกระบวนการรีแฟคทอริงต่อมาตรวัดเชิงวัตถุอย่างไร โดยใช้เทคนิคการนำซอร์สโค้ดมาวิเคราะห์ด้วยวิธีแอบสแทรกซินแทคทรีเรพพลิเซนเทชัน (Abstract syntax tree representation) (AST) โดยได้เลือกอธิบายผลกระทบของกระบวนการรีแฟคทอริงกับโครงสร้าง Program ไว้ดังนี้ เอ็กแทรกเมท็อด เอนแคปซูเลทฟิลด์ พูลอัพเมท็อดซับคลาส (Pull Up Method Subclass) และพูลอัพเมท็อดซุปเปอร์คลาส (Pull Up Method Superclass) และเลือกมาตรวัดโดยแบ่งเป็นประเภทได้แก่ จำนวนเมท็อดความซับซ้อนของโปรแกรม จำนวนคลาสลูก จำนวนการเชื่อมต่อระหว่างวัตถุและจำนวนการทำงานร่วมกันระหว่างเมท็อด ซึ่งงานวิจัยนี้ได้วิเคราะห์ผลกระทบกระบวนการรีแฟคทอริงด้วยระดับ 3 ระดับคือ นิล (nil) (0) โพสิทีฟ (positive) (+) และ เนคกะทีฟ (negative) (-) โดยได้สรุปผลกระทบของกระบวนการรีแฟคทอริงทั้ง 4 วิธีและสรุปความสัมพันธ์ของมาตรวัดคุณภาพซอฟต์แวร์กับกระบวนการรีแฟคทอริงไว้

Demeyer และคณะ (2004) ได้นำเสนอผลที่ดีขึ้นของการทำกระบวนการรีแฟคทอริงในซอร์สโค้ดในด้านการเชื่อมต่อและการทำงานร่วมกัน โดยได้แบ่งแยกมุมมองการวิเคราะห์

คุณสมบัติด้านการเชื่อมต่อและการทำงานร่วมกันไว้ดังนี้ การทำงานร่วมกัน (Non-Normalized และ Normalized) และการเชื่อมต่อ (Import Coupling General Coupling Export Coupling และ Aggregated Import Coupling) โดยได้ทดลองกับกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด วิธีมูฟเมท็อด วิธีรีเพลสเมท็อดเวลูวิธเมท็อดอ็อบเจกต์ (Replace Method Value with Method Object) วิธีรีเพลสดาต้าแวลูซ์วิธอ็อบเจกต์และวิธีเอ็กแทรกคลาส แล้วสรุปผลกระทบในเชิงอิมพรูฟเมนต์ (Improvement) (+) ดีเทอเรียเรชัน (Deterioration) (-) และนิวทรัลอิมแพ็ค (Neutral Impact) (0) กับมุมมองของการเชื่อมต่อและการทำงานร่วมกันแต่ละด้าน โดยได้สรุปว่า กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด วิธีรีเพลสเมท็อดเวลูวิธเมท็อดอ็อบเจกต์ วิธีรีเพลสดาต้าแวลูซ์วิธอ็อบเจกต์และวิธีเอ็กแทรกคลาสมีส่วนช่วยปรับปรุงซอร์สโค้ดให้ดีขึ้น แต่วิธีเอ็กแทรกเมท็อดจะส่งผลที่ไม่ดีสำหรับคุณสมบัติในด้านการทำงานร่วมกัน

Mens และคณะ (2004) ได้ศึกษากระบวนการรีแฟคทอริงที่ส่งผลขัดแย้งกัน โดยวิเคราะห์แต่ละคู่ของกระบวนการรีแฟคทอริงซึ่งใช้วิธีการจับคู่จากกระบวนการรีแฟคทอริง ดังนี้ เอนแคบซูเลชันแวริเอเบิล (Encapsulation Variable) มูฟเมท็อด (Move Method) มูฟแวริเอเบิล (Move Variable) พูลอัพเมท็อด (Pull Up Method) พลูอัพแวริเอเบิล(Pull Up Variable) ครีเอทซุปเปอร์คลาส (Create Superclass) รีเนมเมท็อด (Rename Method) รีเนมแวริเอเบิล (Rename Variable) และรีเนมคลาส (Rename Class) ได้วิเคราะห์ผลการขัดแย้งกันไว้ดังนี้ การวิเคราะห์โดยใช้การจับคู่แล้วดูความผิดพลาดที่เกิดขึ้นเมื่อทำกระบวนการรีแฟคทอริงในแต่ละคู่ โดยใช้เทคนิคกราฟทรานฟอร์มเมชันและการวิเคราะห์แบบคริทิคอลแพร์ (Critical pair analysis) โดยใช้วิเคราะห์บนพื้นฐานของเครื่องมือเอจีจีกราฟทรานฟอร์มเมชัน (Graph transformation tool AGG) ซึ่งสามารถวิเคราะห์ปัญหาที่ขัดแย้งในแต่ละคู่ได้แก่ การใช้วิธีมูฟชันแวริเอเบิลกับพลูอัพแวริเอเบิลเกิดความขัดแย้งกันเนื่องจากถ้าตัวแปรที่ต้องการทำเป็นตัวแปรเดียวกัน จะทำให้ไม่สามารถเคลื่อนย้ายได้เนื่องจากกระบวนการรีแฟคทอริงทั้ง 2 วิธีเป็นการเคลื่อนย้ายตัวแปร เป็นต้น

บทที่ 3
ระเบียบวิธีวิจัย

3.1 บทนำ

แนวทางการวิจัยประกอบด้วย แนวทางของการทำวิจัย แผนแบบการทดลอง (Experimental Design) การทดสอบสมมติฐาน การเตรียมเครื่องมือที่ใช้ในการทดลอง ขั้นตอนการเก็บรวบรวมข้อมูล (Data gathering execution) ประเด็นของความเชื่อถือได้ (Reliability) และความถูกต้อง (Validity) และกรอบการวิเคราะห์ข้อมูล (Data analysis framework) ดังรายละเอียดต่อไปนี้

3.2 แผนแบบการทดลอง (Experimental Design)

วัตถุประสงค์ของงานวิจัยนี้จัดทำขึ้นเพื่อ (1) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงและหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี (2) เปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี และ (3) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน ว่าแตกต่างกันอย่างไร

จากวัตถุประสงค์ข้างต้น งานวิจัยนี้ใช้แผนแบบการทดลองแบบวันกรุ๊ปพรีเทสโพสเทสดีไซน์ (One Group Pretest - Posttest Design) (Babbie, 2001) ซึ่งเป็นแผนแบบการทดลองที่เหมาะสมกับการทดลองที่ต้องการวัดค่าตัวแปรตามของหน่วยทดลองก่อนถูกกระตุ้นเทียบกับค่าของตัวแปรตามของหน่วยทดลองหลังจากถูกกระตุ้นด้วยตัวแปรต้นว่ามีค่าแตกต่างกันอย่างไร ซึ่งมีรายละเอียดปัจจัยที่เกี่ยวข้องดังนี้

3.3.1 ตัวแปรต้น (Independent Variable) งานวิจัยนี้มีตัวแปรต้น 2 ตัว คือกระบวนการรีแฟคทอริงอย่างน้อย 10 วิธีและลำดับของกระบวนการรีแฟคทอริงในแต่ละคู่

3.3.2 ตัวแปรตาม (Dependent Variable) ค่าที่ต้องการวัดการเปลี่ยนแปลงคือคุณภาพซอฟต์แวร์ด้านความสามารถในการบำรุงรักษา (Maintainability) ด้านความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และด้านความสามารถในการทำความเข้าใจ (Understandability) ซึ่งคุณภาพเหล่านี้วัดได้จากมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995) โดยผู้วิจัยจะปรับเปลี่ยนค่า (Manipulate) ของตัวแปรต้นให้กับหน่วยทดลองเพื่อพิจารณาผลอันเกิดจากการกำหนดค่าตัวแปรต้นที่แตกต่างกัน

3.3.3 ตัวแปรควบคุม (Control Variable) ในงานวิจัยเพื่อให้ผลการทดลองนั้นเกิดจากการปรับเปลี่ยนค่าของตัวแปรต้นอย่างแท้จริง โดยผู้วิจัยต้องควบคุมปัจจัยอื่นๆ ที่อาจส่งผลกระทบกับการทดลองให้มีความคงที่และเหมือนกันมากที่สุด ดังต่อไปนี้

3.3.3.1 หน่วยทดลองซึ่งเป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งเป็นวิชาแรกที่เกี่ยวกับซอฟต์แวร์เชิงวัตถุ (Object – Oriented Software) โดยผู้วิจัยได้กำหนดให้ระบบจากงานที่ได้รับมอบหมายต้องเป็นระบบที่มีความหลากหลายในเชิงธุรกิจ แต่ต้องมีจำนวนคลาสภายในระบบตั้งแต่ 5 คลาสขึ้นไป โดยตัวอย่างของหน่วยทดลองได้แก่ ระบบจองตั๋วโรงภาพยนตร์ ระบบจองห้องพักโรงแรมและระบบร้านหนังสือ เป็นต้น และระบบที่ได้จากหน่วยทดลองต้องถูกพัฒนาด้วยภาษาซีชาร์ป (C #) เนื่องจากผู้วิจัยไม่ต้องการให้เกิดปัจจัยทางด้านความแตกต่างของระบบที่ได้รับจากหน่วยทดลองและทางด้านภาษาของโปรแกรมที่พัฒนา ทั้งนี้เพื่อให้ผลการทดลองที่มีผลมาจากการเปลี่ยนแปลงของตัวแปรต้นที่ผู้วิจัยกำหนดเท่านั้น

3.3.3.2 เครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริง ซึ่งผู้วิจัยเลือกใช้เครื่องมือที่เรียกว่ารีชาร์ปเปอร์ 2.0 (ReSharper 2.0) ทำกระบวนการรีแฟคทอริง โดยรีชาร์ปเปอร์ 2.0 เป็นเครื่องมือที่เพิ่มเข้าไปในเครื่องมือพัฒนาซอฟต์แวร์ที่เรียกว่าวิชวลสตูดิโอดอทเน็ต 2005 เนื่องจากผู้วิจัยไม่ต้องการให้มีผลกระทบที่เกิดจากการทำกระบวนการรีแฟคทอริงจึงนำเครื่องมือในที่มีใช้ในปัจจุบันมาใช้ ซึ่งมีความแม่นยำและเป็นมาตรฐาน

3.3.3.3 เครื่องมือที่ใช้วัดค่ามาตรวัดเชิงวัตถุที่กล่าวมาในบทที่ 2 จะถูกนำไปพัฒนาเป็นซอฟต์แวร์ที่ใช้ในการวัดค่ามาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995)

3.3 การทดสอบสมมติฐาน

จากวัตถุประสงค์ของงานวิจัยนี้ ผู้วิจัยต้องการ (1) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงและหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี (2) เปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี และ (3) เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน โดยนำข้อมูลจากซอร์สโค้ดผู้วิจัยสามารถตั้งสมมติฐานสำหรับงานวิจัยได้ดังนี้

กำหนดให้ M_0 คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริง

M_x คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงแต่ละวิธี (x) จาก 20 วิธี

$M_{x\to y}$ คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงวิธีแรก (x) จาก 20 วิธี แล้วตามด้วยวิธีที่สอง (y) จาก 20 วิธี โดยไม่ซ้ำกัน

1. เปรียบเทียบคุณภาพ M_0 และ M_x

   H_0: M_0 และ M_x มีค่าไม่แตกต่างกัน

   H_1: M_0 และ M_x มีค่าแตกต่างกัน

2. เปรียบเทียบคุณภาพ M_x และ $M_{x\to y}$

   H_0: M_x และ $M_{x\to y}$ มีค่าไม่แตกต่างกัน

   H_1: M_x และ $M_{x\to y}$ มีค่าแตกต่างกัน

3. เปรียบเทียบคุณภาพ $M_{x\to y}$ และ $M_{y\to x}$

   H_0: $M_{x\to y}$ และ $M_{y\to x}$ มีค่าไม่แตกต่างกัน

   H_1: $M_{x\to y}$ และ $M_{y\to x}$ มีค่าแตกต่างกัน

3.4 แนวทางการทำวิจัย

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เนื่องจากงานวิจัยนี้ต้องการทราบว่าเมื่อกำหนดตัวแปรที่สนใจให้เปลี่ยนค่า แล้วส่งผลกระทบที่มีต่อตัวแปรตามอย่างไร (ศิริวรรณ เสรีรัตน์ และคณะ, 2541; Babbie, 2001) ซึ่งในงานวิจัยนี้ตัวแปรที่สนใจคือกระบวนการรีแฟคทอริงที่นำมาช่วยปรับปรุงโครงสร้างของซอร์สโค้ดที่มีอยู่ กล่าวคืองานวิจัยนี้ต้องการวัดความเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ระหว่างซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี และต้องการวัดความเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ระหว่างซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี นอกจากนี้ยังให้ความสนใจกับลำดับของทำกระบวนการรีแฟคทอริงแต่ละคู่ กล่าวคืองานวิจัยนี้ต้องการวัดความเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ระหว่างซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน โดยตัวแปรอื่นๆ เช่น เครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริงเครื่องมือที่ใช้วัดคุณภาพซอฟต์แวร์และซอร์สโค้ดที่นำมาใช้ในการทดลองต้องถูกพัฒนาด้วยภาษาเดียวกันและหน่วยทดลองที่พัฒนาต้องมีลักษณะเหมือนกัน ทั้งนี้เพื่อให้ผล

การทดลองที่เกิดขึ้นนั้นเป็นผลมาจากการทำกระบวนการรีแฟคทอริงและลำดับในการทำกระบวนการ รีแฟคทอริงอย่างแท้จริง

3.5 การเตรียมเครื่องมือที่ใช้ในการทดลอง

เครื่องมือที่ใช้ในการทดลองนี้คือ ซอฟต์แวร์มาตรวัดเชิงวัตถุที่สามารถวัดค่ามาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995) ซึ่งมีลักษณะการทำงานเบื้องต้นดังนี้

1. เครื่องมือมาตรวัดเชิงวัตถุจะเรียกซอร์สโค้ดเข้ามา เพื่อคำนวณแต่ละมาตรวัดเชิงวัตถุ
2. เครื่องมือจะคำนวณค่าตามมาตรวัดเชิงวัตถุจากซอร์สโค้ด แล้วสร้างตารางแสดงค่าที่คำนวณได้ของแต่ละมาตรวัดเชิงวัตถุและบันทึกข้อมูลลงฐานข้อมูล

โดยผู้วิจัยมีขั้นตอนการออกแบบเครื่องมือมาตรวัดเชิงวัตถุที่ใช้ในการทดลองดังต่อไปนี้

3.5.1 เอกสารความต้องการซอฟต์แวร์ (Software Requirement Specification) เป็นเอกสารแสดงการทำงานของซอฟต์แวร์ที่ต้องการพัฒนาขึ้น สามารถแบ่งเป็นข้อกำหนดของความต้องการซอฟต์แวร์ (Software Requirement Specification) ได้ดังนี้

ฟังก์ชันนอลรีไควเมนต์ (Functional Requirement)

- ซอฟต์แวร์สามารถคำนวณค่ามาตรวัดเชิงวัตถุได้ โดยใช้มาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995)
- ซอฟต์แวร์เรียกข้อมูลเข้าจากซอร์สโค้ดซีชาร์ป ข้อมูลเข้าของซอฟต์แวร์นี้เป็นซอร์สโค้ดที่พัฒนาด้วยภาษาซีชาร์ป (C #)
- ซอฟต์แวร์สามารถแสดงผลการคำนวณออกทางหน้าจอคอมพิวเตอร์

นอนฟังก์ชันนอลรีไควเมนต์ (Non - Functional Requirement)

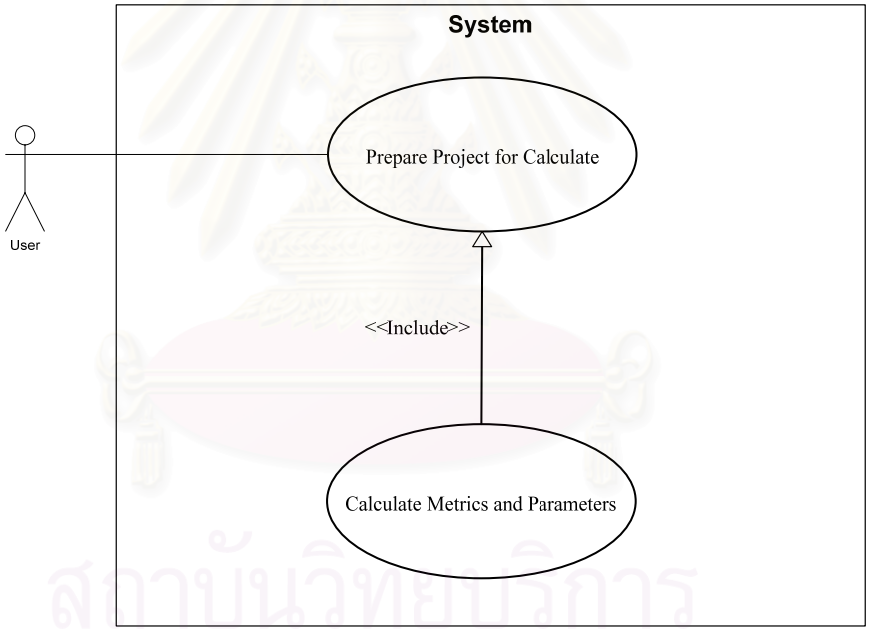
- ซอฟต์แวร์สามารถคำนวณค่ามาตรวัดเชิงวตถุได้อย่างรวดเร็ว
- ซอฟต์แวร์สามารถทำความเข้าใจได้ง่ายและง่ายต่อการใช้งาน
- ซอฟต์แวร์สามารถคำนวณค่ามาตรวัดเชิงวัตถุได้อย่างถูกต้องแม่นยำ

และมีข้อกำหนดของระบบที่จะนำไปใช้ (System Requirement) ดังนี้

- ระบบต้องติดตั้งระบบปฏิบัติการไมโครซอฟต์วินโดวส์เอ็กซ์พี (Windows XP) ขึ้นไป

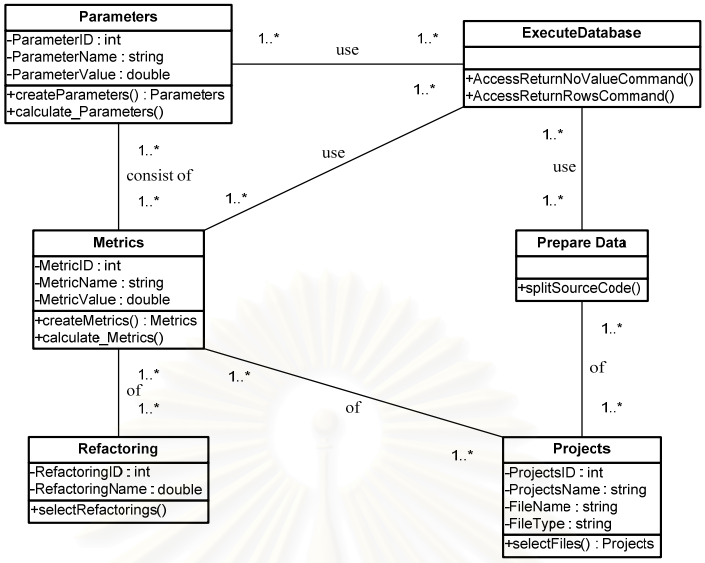
- ระบบต้องติดตั้งวิชวลสตูดิโอดอทเน็ต 2005 และไมโครซอฟต์ดอทเน็ตเฟรมเวิคเอสดีเคเวอร์ชัน 2.0 ขึ้นไป
- ซอฟต์แวร์การวัดคำนวณค่ามาตรวัดเชิงวัตถุถูกติดตั้งเพิ่ม (Add-in) เข้าไปกับเครื่องมือพัฒนาซอฟต์แวร์วิชวลสตูดิโอดอทเน็ต 2005 โดยจำกัดเพียงภาษาซีชาร์ป (C #)

3.5.2 แผนภาพยูสเคส (Use Case Diagram) เป็นเอกสารที่แสดงการโต้ตอบของข้อมูล เมื่อผู้วิจัยจัดทำเอกสารความต้องการซอฟต์แวร์เสร็จเรียบร้อยแล้วจึงได้จัดทำแผนภาพยูสเคส ซึ่งประกอบไปด้วย 1) ยูสเคสการจัดเตรียมข้อมูลก่อนการคำนวณค่ามาตรวัดเชิงวัตถุ (Prepare Project for Calculate) และ 2) ยูสเคสการคำนวณค่ามาตรวัดเชิงวัตถุและค่าพารามิเตอร์ (Calculate Metrics and Parameters) ดังรูปที่ 3-1 และสามารถดูเอกสารคำอธิบายยูสเคสได้ที่ภาคผนวก ก



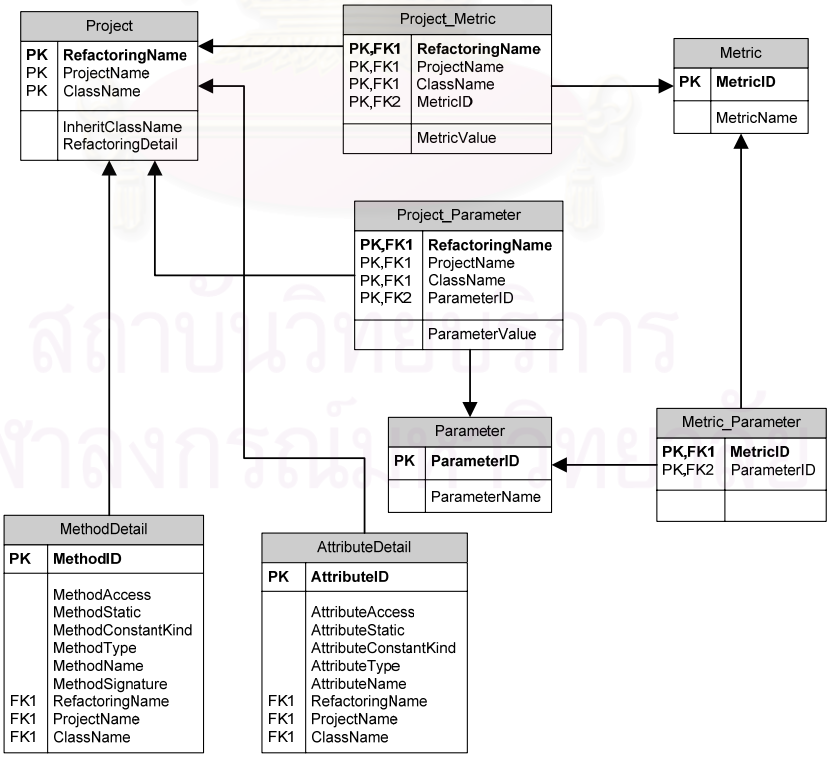
รูปที่ 3-1 แสดงแผนภาพยูสเคสของเครื่องมือที่พัฒนาขึ้น

3.5.3 แผนภาพคลาส (Class Diagram) แผนภาพคลาสจะเป็นเอกสารที่แสดงคลาสในซอฟต์แวร์ที่จะพัฒนาขึ้น ซึ่งจัดทำขึ้นจากการวิเคราะห์ยูสเคส ดังรูปที่ 3-2 และสามารถดูเอกสารคำอธิบายคลาสได้ที่ภาคผนวก ก



รูปที่ 3-2 แสดงแผนภาพคลาสของเครื่องมือที่พัฒนาขึ้น

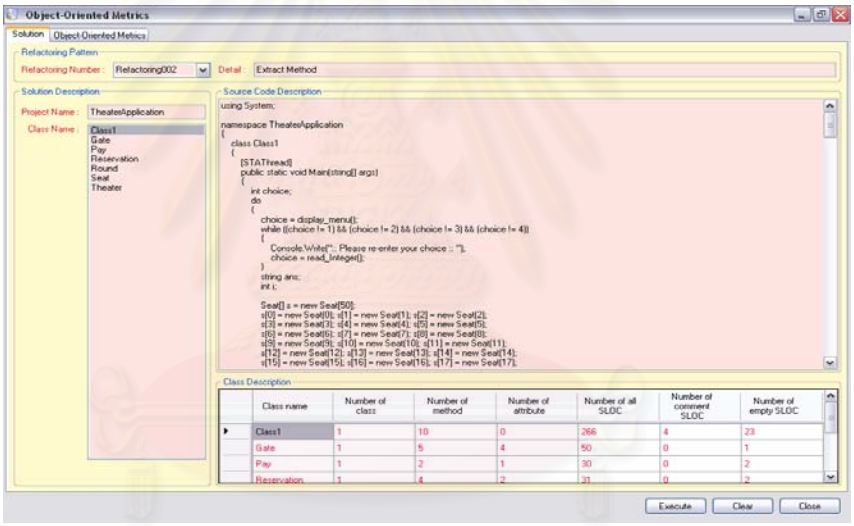
3.5.4 แผนภาพอีอาร์ (ER Diagram) แผนภาพอีอาร์เป็นเอกสารที่แสดงความสัมพันธ์ระหว่างข้อมูลที่จัดเก็บในฐานข้อมูล สามารถแสดงได้ดังรูปที่ 3-3 และสามารถดูเอกสารพจนานุกรมข้อมูลได้ที่ภาคผนวก ก



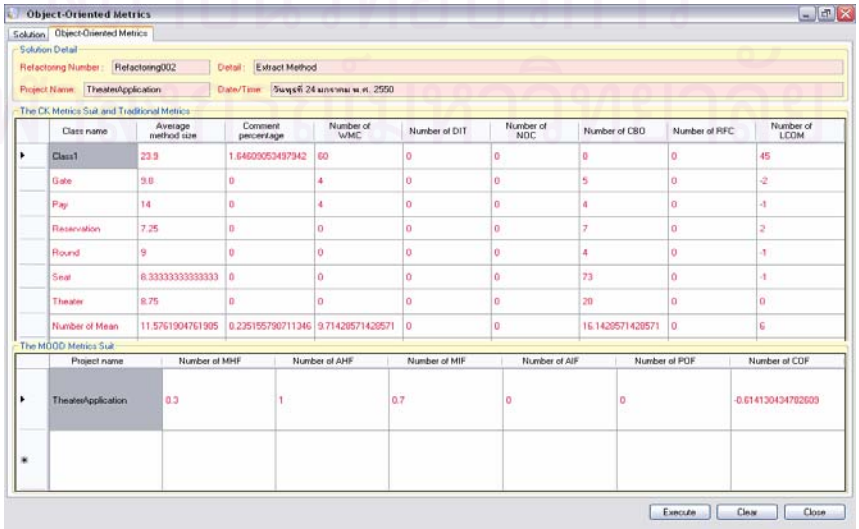
รูปที่ 3-3 แสดงแผนภาพอีอาร์ของเครื่องมือที่พัฒนาขึ้น

3.5.5 การพัฒนาเครื่องมือ

จากเอกสารในขั้นตอนการออกแบบ (สามารถดูเอกสารอย่างละเอียดได้ที่ภาคผนวก ก) สามารถนำไปพัฒนาเป็นซอฟต์แวร์มาตรวัดเชิงวัตถุ ซึ่งมีลักษณะเป็นโปรแกรมประยุกต์แบบวินโดวส์ (Windows Application) ที่เพิ่มเข้าไปให้กับเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 (Visual Studio .NET 2005) โดยเครื่องมือที่ทางผู้วิจัยพัฒนาขึ้นสามารถคำนวณค่ามาตรวัดเชิงวัตถุกับซอร์สโค้ดที่เปิดขึ้นใช้งานกับเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 นี้ได้ สำหรับเครื่องมือหรือซอฟต์แวร์มาตรวัดเชิงวัตถุนี้ได้ถูกออกแบบให้มีการใช้งานเพื่องานวิจัยนี้โดยเฉพาะ ดังนั้นฟังก์ชันการทำงานอาจจะไม่ครบถ้วนสมบูรณ์เหมือนกับซอฟต์แวร์ที่มีอยู่ในตลาด เช่น ฟังก์ชันข้อมูลการช่วยเหลือ ฟังก์ชันของแถบแสดงสถานะหรือฟังก์ชันของแถบเครื่องมือ เป็นต้น โดยผู้วิจัยได้ออกแบบให้มีลักษณะหน้าจอการทำงานดังรูปที่ 3-4 และรูปที่ 3-5



รูปที่ 3-4 แสดงหน้าจอการทำงานหน้าที่ 1 ของเครื่องมือที่พัฒนาขึ้น



รูปที่ 3-5 แสดงหน้าจอการทำงานหน้าที่ 2 ของเครื่องมือที่พัฒนาขึ้น

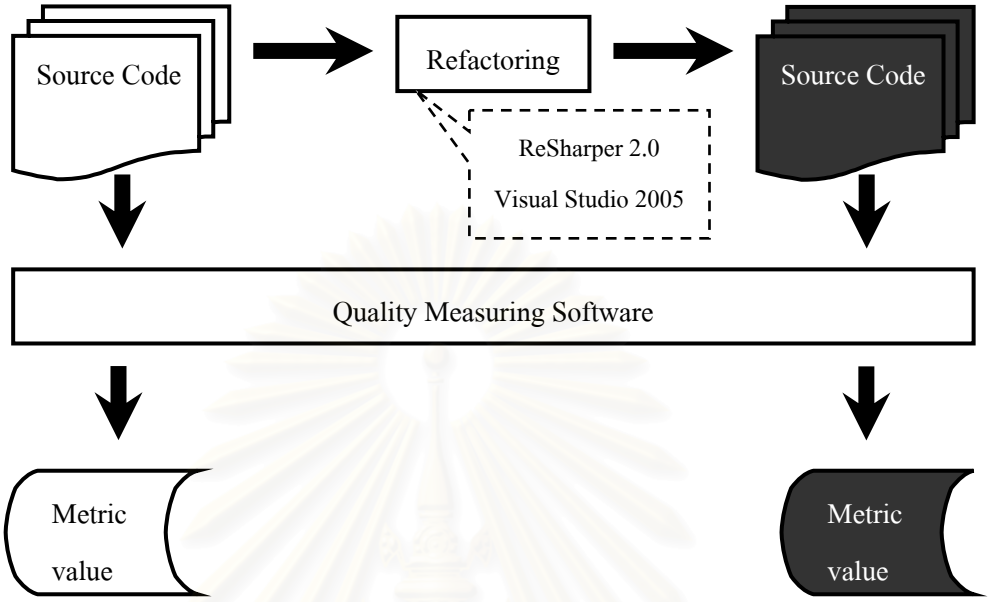
จากหน้าจอการทำงานที่เห็นข้างต้นแสดงลักษณะการทำงานของเครื่องมือที่ใช้วัดมาตรวัดเชิงวัตถุนี้ โดยหน้าจอที่ 1 แสดงถึงการเริ่มต้นการทำงานโดยการดึงซอร์สโค้ดจากหน่วยทดลองเข้ามาคำนวณค่าพื้นฐานของโปรแกรม เช่น จำนวนคลาส จำนวนเมท็อด หรือจำนวนบรรทัดทั้งหมด เป็นต้น จากนั้นเมื่อได้ค่าพื้นฐานของโปรแกรมเรียบร้อยแล้ว เครื่องมือจะนำค่าดังกล่าวไปคำนวณค่ามาตรวัดเชิงวัตถุทั้ง 14 ค่า ซึ่งแสดงไว้ในหน้าจอที่ 2 ของโปรแกรม โดยจะถูกแสดงในรูปแบบของตาราง และสำหรับมาตรวัดเชิงวัตถุที่มีผลลัพธ์เป็นค่ามาตรวัดของแต่ละคลาส ทางผู้วิจัยจะนำไปคำนวณเป็นค่าเฉลี่ย เพื่อให้สะดวกกับการนำไปใช้ในงานวิจัยต่อไป

3.5.6 การทดสอบเครื่องมือ

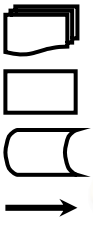
ผู้วิจัยจัดการทดสอบเครื่องมือที่ใช้ในการทดลอง เพื่อลดความผิดพลาดของซอฟต์แวร์ที่พัฒนาขึ้น โดยผู้วิจัยเริ่มทดสอบซอฟต์แวร์ตั้งแต่ขั้นตอนของการพัฒนาเป็นโมดูลย่อยๆ ซึ่งการทดสอบนี้เรียกว่า “การทดสอบแต่ละหน่วย (Unit Testing)” การทดสอบนี้จะช่วยลดความผิดพลาดที่เกิดขึ้นในแต่ละโมดูลออกไป สำหรับการทดสอบแต่ละหน่วย ผู้วิจัยใช้กรณีทดสอบอย่างง่าย เช่น การทดสอบความถูกต้องในการคำนวณค่าของมาตรวัด โดยผู้วิจัยจะสร้างซอร์สโค้ดตัวอย่างซึ่งมีคุณสมบัติต่างๆ ตามที่ผู้วิจัยกำหนดไว้ แล้วทดลองให้โมดูลนั้นทำงานเพื่อตรวจสอบความถูกต้องในการคำนวณค่าของมาตรวัด หรือการทดสอบการแสดงผล โดยผู้วิจัยจะป้อนข้อมูลที่ผู้วิจัยกำหนดไว้จากนั้นทดลองให้โมดูลแสดงผลการทำงาน แล้วจึงตรวจสอบความถูกต้องในการแสดงผล เป็นต้น เมื่อผู้วิจัยพัฒนาโมดูลย่อยแต่ละโมดูลเรียบร้อยแล้ว ผู้วิจัยจึงทดสอบการทำงานร่วมกันของแต่ละโมดูล ซึ่งเรียกการทดสอบนี้ว่า “การทดสอบรวม (Integration Testing)” เป็นการทดสอบเพื่อหาความผิดพลาดในการนำแต่ละโมดูลมาทำงานร่วมกัน โดยผู้วิจัยเริ่มทดสอบตั้งแต่การเรียกใช้ข้อมูลจากฐานข้อมูล การนำข้อมูลไปคำนวณค่ามาตรวัดเชิงวัตถุ การแสดงผลข้อมูลและการจัดเก็บข้อมูลลงฐานข้อมูล สุดท้ายผู้วิจัยได้ทดสอบการทำงานทั้งระบบ ซึ่งเรียกการทดสอบนี้ว่า “การทดสอบระบบ (System Testing)” เป็นการทดสอบความถูกต้องของระบบ โดยผู้วิจัยทดสอบโดยการนำระบบที่ผู้วิจัยกำหนดขึ้นเพื่อใช้สำหรับทดสอบเข้ามาใช้งานซอฟต์แวร์นี้ ซึ่งผู้วิจัยได้กำหนดผลลัพธ์ของการใช้งานไว้เรียบร้อยแล้ว เมื่อทดสอบเสร็จเรียบร้อยแล้วผู้วิจัยจะนำผลที่ได้จากการทดสอบเปรียบเทียบกับผลลัพธ์ที่กำหนดไว้แล้วว่ามีผลที่ถูกต้องตรงกันหรือไม่ ทั้งนี้ผู้วิจัยได้ให้ความสำคัญกับผลของค่ามาตรวัดเชิงวัตถุที่จำเป็นต้องคำนวณค่าได้ถูกต้อง

3.6 ขั้นตอนการเก็บรวบรวมข้อมูล (Data gathering execution)

ในงานวิจัยนี้ผู้วิจัยพัฒนาเครื่องมือที่ใช้วัดคุณภาพซอฟต์แวร์เป็นเครื่องมือที่นำมาเก็บรวบรวมข้อมูล โดยนำมาตรวัดเชิงวัตถุที่เลือกใช้มาสร้างเป็นเครื่องมือที่นำมาใช้วัดซอร์สโค้ดจากหน่วยทดลองซึ่งมีลักษณะการทำงานเบื้องต้น โดยมีขั้นตอนการเก็บรวบรวมข้อมูลดังรูปที่ 3-6



รูปที่ 3-6 แสดงขั้นตอนการเก็บรวบรวมข้อมูล



1. นำกลุ่มข้อมูลซอร์สโค้ดจากหน่วยทดลองมาวัดด้วยเครื่องมือวัดมาตรวัดเชิงวัตถุแล้วบันทึกข้อมูลที่คำนวณได้เก็บไว้

2. นำกลุ่มข้อมูลซอร์สโค้ดจากหน่วยทดลองมาตรวจหาร่องรอยไม่ดีว่าซอร์สโค้ดนั้นควรทำกระบวนการรีแฟคทอริงวิธีใดบ้าง

3. นำกลุ่มข้อมูลซอร์สโค้ดจากหน่วยทดลองผ่านกระบวนการรีแฟคทอริงแต่ละวิธีตามที่ได้กำหนดไว้ แล้วนำกลุ่มข้อมูลซอร์สโค้ดจัดเก็บแยกส่วนกันไว้

4. นำกลุ่มข้อมูลซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงมาวัดด้วยเครื่องมือมาตรวัดเชิงวัตถุแล้วบันทึกข้อมูลที่คำนวณได้เก็บไว้

5. จากนั้นจึงสรุปและวิเคราะห์ผลตามหลักวิชาสถิติตามข้อมูลที่ได้จากการวัดก่อนทำกระบวนการรีแฟคทอริงและหลังทำกระบวนการรีแฟคทอริง

3.7 ประเด็นของความเชื่อถือได้ (Reliability) และความถูกต้อง (Validity)

การตอบวัตถุประสงค์ให้ถูกต้องและน่าเชื่อถือจำเป็นต้องควบคุมปัจจัยที่เกี่ยวข้องอันได้แก่ การเลือกหน่วยทดลอง เครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริง เครื่องมือที่ใช้วัดคุณภาพซอฟต์แวร์ วิธีการเก็บข้อมูลและสภาพแวดล้อมของการทดลอง

เนื่องจากงานวิจัยนี้ต้องการศึกษาถึงผลกระทบของตัวแปรต้น 2 ตัวคือ ปัจจัยจากการทำกระบวนการรีแฟคทอริงและลำดับในการทำกระบวนการรีแฟคทอริงว่าส่งผลต่อคุณภาพซอฟต์แวร์ที่เปลี่ยนแปลงไปหรือไม่ จึงจำเป็นต้องควบคุมปัจจัยในด้านต่างๆ ให้มีความเหมือนกันหรือมีความคงที่ภายใต้สภาวะเดียวกัน เพื่อผลการทดลองที่ออกมาเกิดจากตัวแปรต้นที่ต่างกันเท่านั้น โดยต้องควบคุมปัจจัยดังต่อไปนี้

3.7.1 การเลือกหน่วยทดลอง งานวิจัยนี้เป็นการเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงที่ต่างกัน ดังนั้นหน่วยทดลองที่ผู้วิจัยต้องการนำมาใช้ในการทดลองคือ ซอร์สโค้ดของทุกระบบที่พัฒนาแบบเชิงวัตถุ (Object-Oriented) ซึ่งในทางปฏิบัติแล้วไม่สามารถทำได้ และเพื่อให้คุณสมบัติของหน่วยทดลองที่จะมาเป็นตัวแทนของประชากรในการทดลอง มีความใกล้เคียงกับประชากรที่ผู้วิจัยต้องการมากที่สุด ผู้วิจัยจึงเลือกตัวแทนของประชากรเป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งเป็นระบบที่มีความหลากหลายในเชิงธุรกิจ แต่ต้องมีจำนวนคลาสภายในระบบตั้งแต่ 5 คลาสขึ้นไป และระบบถูกพัฒนาด้วยภาษาซีชาร์ป (C#) ตัวอย่างของหน่วยทดลองได้แก่ ระบบจองตั๋วโรงภาพยนตร์ ระบบจองห้องพักโรงแรมและระบบร้านหนังสือ เป็นต้น

เนื่องจากระบบที่ถูกพัฒนาจากนิสิตระดับปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ทำให้หน่วยทดลองเป็นผู้มีคุณสมบัติพร้อมที่จะสามารถพัฒนาซอฟต์แวร์เชิงวัตถุได้ การที่ผู้วิจัยควบคุมการเก็บข้อมูลตามที่ได้กล่าวมานี้ ช่วยให้งานวิจัยมีความเที่ยงตรงภายใน (Internal Validity) สูง กล่าวคือสามารถสรุปผลอ้างอิงไปหากลุ่มประชากรเป้าหมายได้อย่างถูกต้อง แต่จะมีค่าความเที่ยงตรงภายนอก (External Validity) ต่ำ กล่าวคือผลที่ได้จากงานวิจัยไม่สามารถใช้อธิบายได้โดยทั่วไป (Generalization)

3.7.2 เครื่องมือที่ใช้ในการทดลอง ประกอบไปด้วย 2 ส่วน คือ

1. เครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริง ผู้วิจัยเลือกใช้เครื่องมือรีชาร์ปเปอร์ 2.0 (ReSharper 2.0) ทำกระบวนการ รีแฟคทอริง ซึ่งเครื่องมือนี้ถูกพัฒนาจากทฤษฎีของ Fowler (1999) โดยผู้วิจัยได้ติดต่อกับทางผู้ผลิตซอฟต์แวร์รีชาร์ปเปอร์ 2.0 เพื่อขอลงทะเบียนในการนำเครื่องมือนี้มาใช้ในการศึกษา โดยกำหนดระยะเวลาการนำมาใช้ (Shareware) ของเครื่องมือรีชาร์ปเปอร์ 2.0

เอาไว้ สำหรับเครื่องมือรีชาร์ปเปอร์ 2.0 ถูกพัฒนาโดยองค์กรผลิตซอฟต์แวร์ ทำให้มีความน่าเชื่อถือในการนำไปใช้ในการทดลองเนื่องจากทางผู้ผลิตซอฟต์แวร์ย่อมมีการทดสอบเครื่องมือเป็นที่เรียบร้อยแล้ว

2. เครื่องมือที่ใช้วัดมาตรวัดเชิงวัตถุ ผู้วิจัยเลือกใช้มาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995) ซึ่งเป็นที่ยอมรับในงานวิจัยต่างๆ มาเป็นพื้นฐานที่ใช้ในการพัฒนาเครื่องมือ ดังนั้นจึงมีความน่าเชื่อถือในการนำไปใช้ในการพัฒนาเป็นเครื่องมือที่นำมาใช้วัด นอกจากนี้ผู้วิจัยได้ทดสอบเครื่องมือที่ใช้ในการทดลอง เพื่อตรวจสอบว่าเครื่องมือที่ใช้วัดมาตรวัดเชิงวัตถุที่พัฒนาขึ้นนั้นมีการทำงานที่ถูกต้อง

3.7.3 วิธีการเก็บข้อมูล ผู้วิจัยกำหนดให้หน่วยทดลองระบบที่มีความหลากหลายในเชิงธุรกิจ แต่ต้องมีจำนวนคลาสภายในระบบตั้งแต่ 5 คลาสขึ้นไป ดังนั้นการเลือกซอร์สโค้ดไปผ่านกระบวนการรีแฟคทอริง ผู้วิจัยจึงเลือกเฉพาะซอร์สโค้ดที่เกี่ยวข้องกับฟังก์ชันของระบบเท่านั้น โดยไม่สนใจซอร์สโค้ดที่เป็นส่วนของการติดต่อประสานผู้ใช้ (User Interface) เนื่องจากผู้วิจัยต้องการให้ผลการทดลองมีผลเกี่ยวข้องกับฟังก์ชันการทำงานของแต่ละโปรแกรมเท่านั้น สำหรับการเลือกซอร์สโค้ดมาผ่านกระบวนการรีแฟคทอริงผู้วิจัยได้พิจารณาตามกฎเกณฑ์ในการพิจารณาร่องรอยไม่ดีในซอร์สโค้ดเป็น 25 ประเภท (Fowler, 1999) โดยในแต่ละประเภทจะระบุหลักในการพิจารณาซอร์สโค้ด และกระบวนการรีแฟคทอริงที่ใช้แก้ไขโครงสร้างไม่ดีเอาไว้ ผู้วิจัยได้ดำเนินการทำกระบวนการรีแฟคทอริงกับหน่วยทดลองตามกฎเกณฑ์ในการพิจารณาร่องรอยไม่ดี ซึ่งผู้วิจัยได้บันทึกรายละเอียดการทำกระบวนการรีแฟคทอริงไว้ดังรูปที่ 3-7 ผู้วิจัยได้นำหน่วยทดลองมาผ่านกระบวนการรีแฟคทอริงทั้งหมดแล้วบันทึกรายละเอียดเก็บไว้ จากนั้นจึงนำบันทึกดังกล่าวไปใช้ในการทดลองกับหน่วยทดลองในรูปแบบของแต่ละสมมติฐานที่ผู้วิจัยกำหนดไว้ สำหรับการเลือกกฎเกณฑ์ในการพิจารณาร่องรอยไม่ดีไปใช้กับหน่วยทดลอง ผู้วิจัยได้ใช้การพิจารณาจากทางผู้วิจัยคนเดียวในการพิจารณาการทำกระบวนการรีแฟคทอริงกับหน่วยทดลอง

Project Name		
Bad Smells	**Refactoring (step)**	**Description**
Class Name and Source Line of Code (Before and After)		
1.		
2.		

รูปที่ 3-7 แสดงเอกสารการบันทึกผลการทำกระบวนการรีแฟคทอริง

Project Name		
Bad Smells	**Refactoring (step)**	**Description**
Class Name = Class1, OldSLOC = 201, NewSLOC = 188		
3. Uncommunicative Name	Rename Method (1)	เปลี่ยนชื่อ Variable จาก g เป็น gate
4. Long Method	Decompose Conditional (1)	แยก (input_Keyboard=="y")||(input_Keyboard=="Y") ไปเป็น Method แล้วเรียกใช้งานโดยการป้อน Parameter check_answeryes(input_Keyboard)

รูปที่ 3-8 แสดงเอกสารตัวอย่างการบันทึกผลการทำกระบวนการรีแฟคทอริง

3.8 กรอบการวิเคราะห์ข้อมูล (Data analysis framework)

งานวิจัยนี้เก็บข้อมูลจากเครื่องมือที่ใช้วัดคุณภาพซอฟต์แวร์ทั้งก่อนและหลังทำกระบวนการรีแฟคทอริง หลังจากนั้นผู้วิจัยจึงนำมาวิเคราะห์เพื่อตอบวัตถุประสงค์ที่ได้ตั้งไว้ ซึ่งแบ่งสมมติฐานออกเป็น 2 กลุ่มจากสมมติฐานทั้งหมด โดยได้กำหนดกรอบการวิเคราะห์ข้อมูลไว้ดังนี้

1. เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงแต่ละวิธี โดยนำข้อมูลจากซอร์สโค้ดงานที่ได้รับมอบหมายของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งสามารถวิเคราะห์สมมติฐานกลุ่มนี้ได้ดังนี้

หลังจากได้ข้อมูลสำหรับวิเคราะห์สมมติฐานแล้ว ข้อมูลที่ใช้ทดสอบจะถูกแบ่งเป็น 2 กลุ่มคือ (1) ข้อมูลก่อนทำกระบวนการรีแฟคทอริง (2) ข้อมูลหลังทำกระบวนการรีแฟคทอริง

	ก่อนทำกระบวนการรีแฟคทอริง	หลังทำกระบวนการรีแฟคทอริง
ค่ามาตรวัดเชิงวัตถุ	M_0	M_x

ตารางที่ 3-1 แสดงตารางข้อมูลสำหรับวิเคราะห์สมมติฐานที่ 1

โดยที่ M_0 คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริง

M_x คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงวิธีที่ x

จากทฤษฎีของกระบวนการรีแฟคทอริง ที่กล่าวไว้ว่ากระบวนการรีแฟคทอริงเป็นกระบวนการที่ช่วยปรับปรุงซอร์สโค้ดที่มีอยู่ (Fowler, 1999) ทำให้ผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่ได้หลังจากผ่านกระบวนการรีแฟคทอริงจะมีคุณภาพซอฟต์แวร์ดีกว่าซอร์สโค้ดที่ไม่ผ่านกระบวนการรีแฟคทอริง ดังนั้นจาก M ซึ่งเป็นค่าที่คำนวณได้จากมาตรวัดเชิงวัตถุในแต่ละเงื่อนไขของแต่ละหน่วยทดลอง จะได้เป็นกลุ่มข้อมูลจำนวนมากซึ่งสามารถนำมาวิเคราะห์ผลต่อไป เมื่อได้ค่า M จากการทดลองแล้วจะต้องนำมาตรวจสอบการแจกแจงของข้อมูลที่ได้ว่าเป็นการแจกแจงปกติหรือไม่ (Normal Distribution) ถ้าเป็นการแจกแจงแบบปกติจะใช้การทดสอบแบบใช้พารามิเตอร์ (Parametric Test) ดังนี้

เนื่องจากงานวิจัยนี้เป็นการทดสอบผลต่างระหว่างค่าเฉลี่ยของมาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันหรือกรณีที่เป็นข้อมูลจับคู่ (Matched Pair t - Test) โดยรูปแบบของสมมติฐานในงานวิจัยนี้เป็นแบบการทดสอบสมมติฐานว่าแตกต่างกันจากค่าก่อนทำทรีทเมนต์ (Treatment) หรือไม่ (อำนาจ วังจีน และ พรรณี บุญสุยา, 2548) โดยแยกการทดสอบสมมติฐานเป็น 2 กรณี ดังนี้

1. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุมีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \leq \mu_1$$
$$H_1: \mu_2 > \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริง
μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง

สำหรับมาตรวัดเชิงวัตถุที่มีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นประกอบไปด้วย 1) คอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) 2) เมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) 3) แอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) 4) เมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) และ 5) โพลีมอร์ฟิซึมแฟคเทอร์ (Polymorphism Factor: POF) (Quenel และ Lövdahl, 2004)

หลังจากทดสอบผลต่างระหว่างค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันแล้วสำหรับกรณีที่ 1 ถ้าค่า Sig. (Significance) ที่คำนวณได้น้อยกว่า 0.05 และค่าสถิติ t มากกว่า 0 จึงจะสามารถปฏิเสธ H_0 โดยที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $v = n - 1$ ซึ่ง n คือจำนวนหน่วยทดลอง

2. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุมีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \geq \mu_1$$
$$H_1: \mu_2 < \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริง
μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง

สำหรับมาตรวัดเชิงวัตถุที่มีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นประกอบไปด้วย 1) เวทเทดเมท็อดเปอคลาส (Weighted Methods per Class: WMC) 2) เดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree: DIT) 3) นัมเบอร์ออฟชิลเดรน (Number of Children: NOC) 4) คัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) 5) เรสปอนซ์ฟอร์อะคลาส (Response for a Class: RFC) 6) แลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) 7) เอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) 8) แอตทริบิวต์อินเฮอริแทนแฟคเทอร์ (Attribute Inheritance Factor: AIF) และ 9) คัพพลิงแฟคเทอร์ (Coupling Factor: COF) (Quenel และ Lövdahl, 2004)

หลังจากทดสอบผลต่างระหว่างค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันแล้วสำหรับกรณีที่ 2 ถ้าค่า Sig. (Significance) ที่คำนวณได้น้อยกว่า 0.05 และค่าสถิติ t น้อยกว่า 0 จึงจะสามารถปฏิเสธ H_0 โดยที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $v = n - 1$ ซึ่ง n คือจำนวนหน่วยทดลอง

ทั้ง 2 กรณีสามารถคำนวณค่าสถิติ t ได้ดังนี้

$$t = \frac{\bar{d} - \mu_D}{S_d / \sqrt{n}}$$

โดยที่ $\bar{d}$ คือค่าเฉลี่ยของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงและที่ไม่ผ่านกระบวนการรีแฟคทอริง

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

d_i คือผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงและที่ไม่ผ่านกระบวนการรีแฟคทอริง ($M_{x_i} - M_0$)

μ_D คือค่าเฉลี่ยของผลต่างของมาตรวัดเชิงวัตถุของประชากรที่ผ่านกระบวนการรีแฟคทอริงและที่ไม่ผ่านกระบวนการรีแฟคทอริง ในที่นี้คือ 0

S_d คือค่าเบี่ยงเบนมาตรฐานของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงและที่ไม่ผ่านกระบวนการรีแฟคทอริง (d)

$$S_d = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} d^2 - n\bar{d}^2\right)}$$

n คือจำนวนหน่วยทดลอง

แต่ถ้าตรวจสอบแล้วว่าการแจกแจงของข้อมูลที่ได้ไม่เป็นการแจกแจงปกติจะใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Nonparametric Test) ต่อไป

2. เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี แล้วโดยนำข้อมูลจากซอร์สโค้ดงานที่ได้รับมอบหมายของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งสามารถวิเคราะห์สมมติฐานกลุ่มนี้ได้ดังนี้

หลังจากได้ข้อมูลสำหรับวิเคราะห์สมมติฐานแล้ว ข้อมูลที่ใช้ทดสอบจะถูกแบ่งเป็น 2 กลุ่มคือ (1) ข้อมูลที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี (2) ข้อมูลที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี

	ผ่านกระบวนการรีแฟคทอริง 1 วิธี	ผ่านกระบวนการรีแฟคทอริง 2 วิธี
ค่ามาตรวัดเชิงวัตถุ	M_x	$M_{x\to y}$

ตารางที่ 3-2 แสดงตารางข้อมูลสำหรับวิเคราะห์สมมติฐานที่ 2

โดยที่ M_x คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี

$M_{x\to y}$ คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี

จากทฤษฎีของกระบวนการรีแฟคทอริง ที่กล่าวไว้ว่ากระบวนการรีแฟคทอริงเป็นกระบวนการที่ช่วยปรับปรุงซอร์สโค้ดที่มีอยู่ (Fowler, 1999) ทำให้ผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่

ได้หลังจากผ่านกระบวนการรีแฟคทอริง 2 วิธี ย่อมจะมีคุณภาพซอฟต์แวร์ดีกว่าซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี ดังนั้นจาก M ซึ่งเป็นค่าที่คำนวณได้จากมาตรวัดเชิงวัตถุในแต่ละเงื่อนไขของแต่ละหน่วยทดลอง จะได้เป็นกลุ่มข้อมูลจำนวนมากซึ่งสามารถนำมาวิเคราะห์ผลต่อไป เมื่อได้ค่า M จากการทดลองแล้วจะต้องนำมาตรวจสอบการแจกแจงของข้อมูลที่ได้ว่าเป็นการแจกแจงปกติหรือไม่ (Normal Distribution) ถ้าเป็นการแจกแจงแบบปกติจะใช้การทดสอบแบบใช้พารามิเตอร์ (Parametric Test) ดังนี้

เนื่องจากงานวิจัยนี้เป็นการทดสอบผลต่างระหว่างค่าเฉลี่ยของมาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันหรือกรณีที่เป็นข้อมูลจับคู่ (Matched Pair t - Test) โดยรูปแบบของสมมติฐานในงานวิจัยนี้เป็นแบบการทดสอบสมมติฐานว่าแตกต่างกันจากค่าก่อนทำทรีทเมนต์ (Treatment) หรือไม่ (อำนาจ วังจีน และ พรรณี บุญสุยา, 2548) โดยแยกการทดสอบสมมติฐานเป็น 2 กรณี ดังนี้

1. การทดสอบสมมติฐานกรณีที่เมื่อมาตรวัดเชิงวัตถุมีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \leq \mu_1$$
$$H_1: \mu_2 > \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี
μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 2 วิธี

สำหรับมาตรวัดเชิงวัตถุที่มีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นประกอบไปด้วย 1) คอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) 2) เมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) 3) แอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) 4) เมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) และ 5) โพลีมอร์ฟีซึมแฟคเทอร์ (Polymorphism Factor: POF) (Quenel และ Lövdahl, 2004)

หลังจากทดสอบผลต่างระหว่างค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันแล้วสำหรับกรณีที่ 1 ถ้าค่า Sig. (Significance) ที่คำนวณได้น้อยกว่า 0.05 และค่าสถิติ t มากกว่า 0 จึงจะสามารถปฏิเสธ H_0 โดยที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $v = n - 1$ ซึ่ง n คือจำนวนหน่วยทดลอง

2. การทดสอบสมมติฐานกรณีที่เมื่อมาตรวัดเชิงวัตถุมีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \geq \mu_1$$
$$H_1: \mu_2 < \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี
μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 2 วิธี

สำหรับมาตรวัดเชิงวัตถุที่มีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นประกอบไปด้วย 1) เวทเทดเมท็อดเปอคลาส (Weighted Methods per Class: WMC) 2) เดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree: DIT) 3) นัมเบอร์ออฟชิลเดรน (Number of Children: NOC) 4) คัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) 5) เรสปอนซ์ฟอร์อะคลาส (Response for a Class: RFC) 6) แลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) 7) เอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) 8) แอตทริบิวต์อินเฮอริแทนแฟคเทอร์ (Attribute Inheritance Factor: AIF) และ 9) คัพพลิงแฟคเทอร์ (Coupling Factor: COF) (Quenel และ Lövdahl, 2004)

หลังจากทดสอบผลต่างระหว่างค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันแล้วสำหรับกรณีที่ 2 ถ้าค่า Sig. (Significance) ที่คำนวณได้น้อยกว่า 0.05 และค่าสถิติ t น้อยกว่า 0 จึงจะสามารถปฏิเสธ H_0 โดยที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $v = n - 1$ ซึ่ง n คือจำนวนหน่วยทดลอง

ทั้ง 2 กรณีสามารถคำนวณค่าสถิติ t ได้ดังนี้

$$t = \frac{\bar{d} - \mu_D}{S_d / \sqrt{n}}$$

โดยที่ $\bar{d}$ ค่าเฉลี่ยของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี และที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

d_i คือผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี และที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี ($M_{(x\to y)_i} - M_{x_i}$)

μ_D คือค่าเฉลี่ยของผลต่างของมาตรวัดเชิงวัตถุของประชากรที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี และที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี ในที่นี้คือ 0

S_d คือค่าเบี่ยงเบนมาตรฐานของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี และที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี (d)

$$S_d = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} d^2 - n\bar{d}^2\right)}$$

n คือจำนวนหน่วยทดลอง

แต่ถ้าตรวจสอบแล้วว่าการแจกแจงของข้อมูลที่ได้ไม่เป็นการแจกแจงปกติจะใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Nonparametric Test) ต่อไป

3. เปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน โดยนำข้อมูลจากซอร์สโค้ดงานที่ได้รับมอบหมายของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งสามารถวิเคราะห์สมมติฐานกลุ่มนี้ได้ดังนี้

หลังจากได้ข้อมูลสำหรับวิเคราะห์สมมติฐานแล้ว ข้อมูลที่ใช้ทดสอบจะถูกแบ่งเป็น 2 กลุ่มคือ (1) ข้อมูลที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y (2) ข้อมูลที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x

	ผ่านกระบวนการรีแฟคทอริง วิธีที่ x ตามด้วยวิธีที่ y	ผ่านกระบวนการรีแฟคทอริง วิธีที่ y ตามด้วยวิธีที่ x
ค่ามาตรวัดเชิงวัตถุ	$M_{x\to y}$	$M_{y\to x}$

ตารางที่ 3-3 แสดงตารางข้อมูลสำหรับวิเคราะห์สมมติฐานที่ 3

โดยที่ $M_{x\to y}$ คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y

$M_{y \to x}$ คือค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x

จากทฤษฎีของกระบวนการรีแฟคทอริง ที่กล่าวไว้ว่ากระบวนการรีแฟคทอริงเป็นกระบวนการที่ช่วยปรับปรุงซอร์สโค้ดที่มีอยู่ (Fowler, 1999) และจากงานวิจัยที่พบว่าการทำกระบวนการรีแฟคทอริงร่วมกัน 2 วิธี ทำให้เกิดความขัดแย้งกัน (Mens, 2004) ทำให้ผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี ที่เหมือนกันแต่ทำ 2 วิธีนั้นในลำดับที่ต่างกันย่อมจะมีคุณภาพซอฟต์แวร์ที่แตกต่างกันด้วย ดังนั้นจาก M ซึ่งเป็นค่าที่คำนวณได้จากมาตรวัดเชิงวัตถุในแต่ละเงื่อนไขของแต่ละหน่วยทดลอง จะได้เป็นกลุ่มข้อมูลจำนวนมากซึ่งสามารถนำมาวิเคราะห์ผลต่อไป เมื่อได้ค่า M จากการทดลองแล้วจะต้องนำมาตรวจสอบการแจกแจงของข้อมูลที่ได้ว่าเป็นการแจกแจงปกติหรือไม่ (Normal Distribution) ถ้าเป็นการแจกแจงแบบปกติจะใช้การทดสอบแบบใช้พารามิเตอร์ (Parametric Test) ดังนี้

เนื่องจากงานวิจัยนี้เป็นการทดสอบผลต่างระหว่างค่าเฉลี่ยของมาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันหรือกรณีที่เป็นข้อมูลจับคู่ (Matched Pair t - Test) โดยรูปแบบของสมมติฐานในงานวิจัยนี้เป็นแบบการทดสอบสมมติฐานว่าแตกต่างกันจากค่าก่อนทำทรีทเมนต์ (Treatment) หรือไม่ (อำนาจ วังจีน และ พรรณี บุญสุยา, 2548) ดังนี้

$$H_0: \mu_2 = \mu_1$$
$$H_1: \mu_2 \neq \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y

μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x

หลังจากทดสอบผลต่างระหว่างค่าเฉลี่ยของมาตรวัดเชิงวัตถุจากหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกันแล้ว ถ้าค่า Sig. (Significance) ที่คำนวณได้น้อยกว่า 0.05 จึงจะสามารถปฏิเสธ H_0 โดยที่องศาอิสระ (Degree of Freedom) มีค่าเท่ากับ $v = n - 1$ ซึ่ง n คือจำนวนหน่วยทดลอง

โดยสามารถคำนวณค่าสถิติ t ได้ดังนี้

$$t = \frac{\bar{d} - \mu_D}{S_d / \sqrt{n}}$$

โดยที่ $\bar{d}$ คือค่าเฉลี่ยของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y และที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

d_i คือผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y และที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x ($M_{(y\to x)_i} - M_{(x\to y)_i}$)

μ_D คือค่าเฉลี่ยของผลต่างของมาตรวัดเชิงวัตถุของประชากรที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y และที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x ในที่นี้คือ 0

S_d คือค่าเบี่ยงเบนมาตรฐานของผลต่างระหว่างค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y และที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x (d)

$$S_d = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} d^2 - n\bar{d}^2\right)}$$

n คือจำนวนหน่วยทดลอง

แต่ถ้าตรวจสอบแล้วว่าการแจกแจงของข้อมูลที่ได้ไม่เป็นการแจกแจงปกติจะใช้การทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Nonparametric Test) ต่อไป

จากนั้นจึงนำค่าสถิติ t ที่ได้ไปวิเคราะห์ความสัมพันธ์ของผลต่างระหว่างค่าเฉลี่ยของหน่วยทดลอง 2 กลุ่มที่ไม่เป็นอิสระต่อกัน

บทที่ 4
ผลการวิเคราะห์ข้อมูล

4.1 บทนำ

บทนี้จะกล่าวถึงผลการวิเคราะห์ข้อมูลที่ได้จากการทดลอง เพื่อนำมาตอบวัตถุประสงค์ของงานวิจัยที่กล่าวมาข้างต้น ซึ่งได้แก่ การเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี การเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธีกับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี และการเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน ซึ่งจะประกอบด้วยการวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive statistic) การตรวจสอบการแจกแจงของข้อมูล ผลการทดสอบสมมติฐานในลักษณะของสถิติเชิงอนุมาน (Inferential statistics) และผลการวิเคราะห์ข้อมูลเพิ่มเติม

4.2 ผลการวิเคราะห์ข้อมูล

หลังจากผู้วิจัยได้กำหนดแผนแบบการทดลอง (Experimental Design) และจัดทำเครื่องมือที่ใช้ในการทดลองซึ่งได้แก่ หน่วยทดลองที่ผ่านและไม่ผ่านกระบวนการรีแฟคทอริงเรียบร้อยแล้ว และซอฟต์แวร์มาตรวัดเชิงวัตถุ ผู้วิจัยได้ดำเนินการทดลองตามแผนแบบการทดลอง กล่าวคือเก็บข้อมูลจากหน่วยทดลองซึ่งเป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งสามารถนำมาใช้ในการศึกษาได้ มีทั้งหมด 32 หน่วยทดลอง ซึ่งประกอบด้วยหน่วยทดลองจากโครงงานที่จัดทำโดยนิสิตระดับปริญญาบัณฑิตที่เรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ในปีการศึกษา 2547 และ 2548 โดยมีข้อกำหนดว่าจะต้องมีจำนวนคลาสอย่างน้อย 5 คลาส สำหรับซอร์สโค้ดจากหน่วยทดลองสามารถแบ่งลักษณะของซอร์สโค้ดได้เป็น 2 ประเภทคือ โปรแกรมประยุกต์แบบจอเฝ้าคุม (Console Application) และโปรแกรมประยุกต์แบบวินโดวส์ (Windows Application) โดยผู้วิจัยเลือกทำกระบวนการรีแฟคทอริงเฉพาะซอร์สโค้ดที่เกี่ยวข้องกับฟังก์ชันของระบบเท่านั้น จะไม่ทำกระบวนการรีแฟคทอริงส่วนของคลาสที่เป็นส่วนติดต่อประสานผู้ใช้ (User Interface) เพราะซอร์สโค้ดในส่วนติดต่อประสานผู้ใช้นี้เป็นซอร์สโค้ดส่วนที่เครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 สร้างให้อยู่แล้ว ไม่ได้มีส่วนที่เกี่ยวข้องกับการพัฒนาซอร์สโค้ดจากหน่วยทดลอง

ในการทดลองนั้นหน่วยทดลองจะถูกนำมาผ่านกระบวนการรีแฟคทอริงในรูปแบบต่างๆ โดยผู้วิจัยเก็บรวบรวมข้อมูลและนำข้อมูลที่ได้มาวิเคราะห์ เพื่อตอบวัตถุประสงค์ที่กำหนดไว้ใน

ข้างต้น จากนั้นนำไปวิเคราะห์ผลประกอบด้วยการวิเคราะห์ข้อมูลขั้นต้น (Descriptive Statistics) และการวิเคราะห์ข้อมูลเชิงอนุมาน (Inferential Statistics) เพื่อนำไปตอบวัตถุประสงค์ต่อไป

4.2.1 การวิเคราะห์ข้อมูลในลักษณะสถิติเชิงพรรณนา (Descriptive Statistics) งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) เพื่อเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดดังที่กล่าวแล้วข้างต้น โดยมีหน่วยทดลองทั้งหมด 32 หน่วยทดลอง ซึ่งสามารถแบ่งออกเป็น 2 ปีการศึกษาซึ่งประกอบไปด้วยปีการศึกษา 2547 จำนวน 19 หน่วยทดลองและปีการศึกษา 2548 จำนวน 13 หน่วยทดลอง โดยหน่วยทดลองทั้งหมดได้รับกระบวนการรีแฟคทอริงที่ต่างกัน สามารถแจกแจงจำนวนหน่วยทดลองได้ดังนี้

	หน่วยทดลองปี 2547	**หน่วยทดลองปี 2548**	**รวม**
จำนวนหน่วยทดลอง	19	13	32
คิดเป็นเปอร์เซ็นต์ (%)	59.37	40.63	100

ตารางที่ 4-1 แสดงตารางแจกแจงจำนวนหน่วยทดลองที่นำมาผ่านกระบวนการรีแฟคทอริง

จากแบบแผนการทดลองที่กำหนดให้หน่วยทดลองผ่านการพิจารณากฎเกณฑ์ร่องรอยไม่ดีในซอร์สโค้ดจึงพบว่าพบว่าร่องรอยไม่ดีในหน่วยทดลองทั้งหมด 6 ข้อ คือ 1) อันคอมมิวนิเคทีฟเนม 2) ลองเมท็อด 3) ดุพลิเคทโค้ด 4) ลาจคลาส 5) คอมเมนต์ และ 6) ฟีเจอเอ็นวี ตามลำดับความถี่ที่พบ ดังตารางที่ 4-2

Bad Smells / **Project No.**	**Uncommunicative Name**	**Long Method**	**Large Class**	**Duplicate Code**	**Comments**	**Feature Envy**
1	56	2	-	-	-	-
2	84	1	-	-	1	3
3	128	4	-	-	1	-
4	83	1	-	3	-	-
5	75	-	-	3	-	-
6	13	-	-	-	2	-

ตารางที่ 4-2 แสดงตารางจำนวนการพบร่องรอยไม่ดีในหน่วยทดลอง

Bad Smells / Project No.	Uncommunicative Name	Long Method	Large Class	Duplicate Code	Comments	Feature Envy
7	70	5	10	15	9	-
8	10	2	5	-	8	-
9	18	-	-	-	-	-
10	58	3	-	3	14	-
11	79	1	5	3	2	-
12	102	14	27	3	16	-
13	28	6	8	5	2	-
14	74	18	1	1	2	-
15	64	11	3	-	5	-
16	66	46	-	3	1	-
17	87	-	20	-	7	-
18	42	1	-	-	-	-
19	71	9	3	5	-	-
20	19	7	6	3	3	-
21	58	5	5	1	6	-
22	74	24	14	2	1	-
23	58	6	3	2	2	-
24	50	4	3	3	3	-
25	50	5	1	3	-	-
26	52	15	13	3	1	-
27	55	2	4	2	-	-
28	70	12	5	2	1	-
29	62	9	3	3	3	-
30	62	7	4	2	7	-
31	46	25	4	27	-	-
32	36	10	9	2	-	-
Total	1900	255	156	99	97	3

ตารางที่ 4-2 แสดงตารางจำนวนการพบร่องรอยไม่ดีในหน่วยทดลอง (ต่อ)

โดยร่องรอยไม่ดีในซอร์สโค้ดดังกล่าวถูกเสนอให้ทำกระบวนการรีแฟคทอริงทั้งหมด 10 วิธีจาก 20 วิธีที่กำหนดไว้คือ 1) รีเนมเมท็อด 2) เอ็กแทรกเมท็อด 3) ดีคอมโพสคอนดิชันนอล 4) มูฟเมท็อด 5) รีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ 6) มูฟฟิลด์ 7) อินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล 8) เอ็กแทรกคลาส 9) รีเพลสเทมป์วิธคิวรี และ 10) พูลอัพเมท็อด ตามลำดับความถี่ที่พบ ดังตารางที่ 4-3

No	Bad Smells	Refactoring	
1	Uncommunicative Name	Rename (ประเภทที่ 5)	1900
2	Long Method	Introduce Explaining Variable (ประเภทที่ 1)	38
		Decompose Conditional (ประเภทที่ 4)	204
		Replace Temp with Query (ประเภทที่ 1)	13
3	Duplicate Code	Extract Method (ประเภทที่ 1)	96
		Pull Up Method (ประเภทที่ 6)	3
4	Large Class	Extract Method (ประเภทที่ 1)	156
		Extract Class (ประเภทที่ 2)	22
		Move Method (ประเภทที่ 2)	156
		Move Field (ประเภทที่ 2)	91
5	Comments	Replace Magic Number with Symbolic Constant (ประเภทที่ 3)	97
6	Feature Envy	Move Method (ประเภทที่ 2)	3

ตารางที่ 4-3 แสดงตารางจำนวนการทำกระบวนการรีแฟคทอริงในแต่ละร่องรอยไม่ดี

จากตารางที่ 4-3 จะเห็นได้ว่ากระบวนการรีแฟคทอริงที่ทำกับหน่วยทดลองได้มาจาก 6 ประเภทของกระบวนการรีแฟคทอริง (Fowler, 1999) โดยประเภทที่ 1 คอมโพสซิงเมท็อดมีกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด วิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล และวิธีรีเพลสเทมป์วิธคิวรี ประเภทที่ 2 มูฟวิงฟีเจอร์บีทวีนอ็อบเจกต์มีกระบวนการรีแฟคทอริงวิธีมูฟเมท็อด วิธีมูฟฟิลด์และวิธีเอ็กแทรกคลาส ประเภทที่ 3 ออแกไนซ์ซิงดาต้ามีกระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ ประเภทที่ 4 ซิมพลิไฟอิงคอนดิชันนอลเอ็กเพรสชันมีกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอล ประเภทที่ 5 เมกกิงเมท็อดคอลซิมเพลอร์มี

กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด และประเภทที่ 6 ดีลลิงวิธเจนเนอรัลไลเซชันมีกระบวนการรีแฟคทอริงวิธีพูลอัพเมท็อด

จากกระบวนการรีแฟคทอริงที่พบในหน่วยทดลองทั้งหมด 10 วิธีนั้นสามารถนำไปสร้างรูปแบบการทำกระบวนการรีแฟคทอริงเพื่อตอบวัตถุประสงค์งานวิจัยได้ทั้งหมด 101 รูปแบบ (รายชื่อกระบวนการรีแฟคทอริงในแต่ละรูปแบบแสดงไว้ในภาคผนวก ข) โดยในแต่ละรูปแบบสามารถประยุกต์กับหน่วยทดลองในจำนวนที่แตกต่างกันไป

		1. Extract Method	2. Replace Temp with Query	3. Introduce Explaining Variable	4. Move Method	5. Move Field	6. Extract Class	7. Replace Magic Number with Symbolic Constant	8. Decompose Conditional	9. Rename Method	10. Pull Up Method
1. Extract Method	จำนวนหน่วยทดลอง	26	9	13	22	20	22	19	21	26	1
	คิดเป็นเปอร์เซ็นต์ (%)	81.25	28.12	40.62	68.75	62.50	68.75	59.37	65.62	81.25	3.12
2. Replace Temp with Query	จำนวนหน่วยทดลอง	9	12	4	8	8	8	7	7	12	0
	คิดเป็นเปอร์เซ็นต์ (%)	28.12	37.50	12.50	25	25	25	21.87	21.87	37.50	0
3. Introduce Explaining Variable	จำนวนหน่วยทดลอง	13	4	15	11	10	11	12	11	15	1
	คิดเป็นเปอร์เซ็นต์ (%)	40.62	12.50	46.87	34.37	31.25	34.37	37.50	34.37	46.87	3.12
4. Move Method	จำนวนหน่วยทดลอง	22	8	11	22	20	22	17	19	22	1
	คิดเป็นเปอร์เซ็นต์ (%)	68.75	25	34.37	68.75	62.50	68.75	53.12	59.37	68.75	3.12
5. Move Field	จำนวนหน่วยทดลอง	20	8	10	20	20	20	15	17	20	1
	คิดเป็นเปอร์เซ็นต์ (%)	62.50	25	31.25	62.50	62.50	62.50	46.87	53.12	62.50	3.12

ตารางที่ 4-4 แสดงตารางแจกแจงจำนวนหน่วยทดลองที่ได้รับการทำกระบวนการรีแฟคทอริงในแต่ละรูปแบบ

		1. Extract Method	2. Replace Temp with Query	3. Introduce Explaining Variable	4. Move Method	5. Move Field	6. Extract Class	7. Replace Magic Number with Symbolic Constant	8. Decompose Conditional	9. Rename Method	10. Pull Up Method
6. Extract Class	จำนวนหน่วยทดลอง	22	8	11	22	20	22	17	19	22	1
	คิดเป็นเปอร์เซ็นต์ (%)	68.75	25	34.37	68.75	62.50	68.75	53.12	59.37	68.75	3.12
7. Replace Magic Number with Symbolic Constant	จำนวนหน่วยทดลอง	19	7	12	17	15	17	22	18	22	1
	คิดเป็นเปอร์เซ็นต์ (%)	59.37	21.87	37.50	53.12	46.87	53.12	68.75	56.25	68.75	3.12
8. Decompose Conditional	จำนวนหน่วยทดลอง	21	7	11	19	17	19	18	22	22	1
	คิดเป็นเปอร์เซ็นต์ (%)	65.62	21.87	34.37	59.37	53.12	59.37	56.25	68.75	68.75	3.12
9. Rename Method	จำนวนหน่วยทดลอง	26	12	15	22	20	22	22	22	32	1
	คิดเป็นเปอร์เซ็นต์ (%)	81.25	37.50	46.87	68.75	62.50	68.75	68.75	68.75	100	3.12
10. Pull Up Method	จำนวนหน่วยทดลอง	1	0	1	1	1	1	1	1	1	1
	คิดเป็นเปอร์เซ็นต์ (%)	3.12	0	3.12	3.12	3.12	3.12	3.12	3.12	3.12	3.12

ตารางที่ 4-4 แสดงตารางแจกแจงจำนวนหน่วยทดลองที่ได้รับการทำกระบวนการรีแฟคทอริงในแต่ละรูปแบบ (ต่อ)

หมายเหตุ ช่องสีดำ หมายถึงหน่วยทดลองที่ถูกทำกระบวนการรีแฟคทอริงเพียง 1 วิธี ตามวัตถุประสงค์ข้อที่ 1 เช่น แถวที่ 1 คอลัมน์ที่ 1 คือหน่วยทดลองที่ถูกทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด
วิธีการแปลความหมายตารางให้อ่านเริ่มจากด้านซ้ายไปบรรจบกับด้านบน เช่น แถวที่ 1 คอลัมน์ที่ 2 สามารถแปลความหมายได้ว่ารูปแบบหน่วยทดลองที่ถูกทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี โดยมีหน่วยทดลองที่ถูกทำด้วยกระบวนการรีแฟคทอริงทั้ง 2 วิธีนี้ 9 หน่วยทดลอง คิดเป็น 28.12 เปอร์เซ็นต์

จากตารางที่ 4-4 จะเห็นได้ว่ากระบวนการรีแฟคทอริงวิธีพูลอัพเมท็อดเกิดขึ้นกับหน่วยทดลองเพียง 1 หน่วยทดลอง ทางสถิติแล้วไม่สามารถนำไปวิเคราะห์ผลทางสถิติได้ ทำให้ผู้วิจัยต้องตัดกระบวนการรีแฟคทอริงวิธีพูลอัพเมท็อดออกไป จึงทำให้เหลือกระบวนการรีแฟคทอริงสำหรับเปรียบเทียบค่ามาตรวัดเชิงวัตถุเพียง 9 วิธี

4.2.2 การตรวจสอบการแจกแจงของข้อมูล ผู้วิจัยตรวจสอบการแจกแจกของข้อมูลว่าเป็นการแจกแจงปกติหรือไม่ ถ้าข้อมูลมีการการแจกแจงแบบปกติผู้วิจัยจะทดสอบสมมติฐานแบบอิงพารามิเตอร์ (Parametric Test) แต่ถ้าข้อมูลไม่ได้มีการแจกแจงแบบปกติ ผู้วิจัยจะทดสอบสมมติฐานแบบไม่อิงกับพารามิเตอร์ (Non Parametric Test) (กัลยา วาณิชย์บัญชา, 2544) โดยงานวิจัยนี้ผู้วิจัยสนใจตัวแปรคือ คุณภาพซอฟต์แวร์ในที่นี้จะเป็นมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995)

ผู้วิจัยต้องตรวจสอบการแจกแจงข้อมูลว่ามีการแจกแจงแบบปกติหรือไม่ โดยการกำหนดสมมติฐานของการทดสอบได้ดังนี้

1. H_0: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง มีการแจกแจงแบบปกติ

   H_1: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ไม่ได้มีการแจกแจงแบบปกติ

2. H_0: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี มีการแจกแจงแบบปกติ

   H_1: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี หลังผ่านกระบวนการรีแฟคทอริง 2 วิธี ไม่ได้มีการแจกแจงแบบปกติ

3. H_0: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y กับหลังผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x มีการแจกแจงแบบปกติ

   H_1: ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y กับหลังผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x ไม่ได้มีการแจกแจงแบบปกติ

ในการตรวจสอบการแจกแจงของข้อมูลเชิงปริมาณว่าเป็นแบบปกติโดยใช้สถิติทดสอบนั้น มีสถิติทดสอบที่ใช้คือ Kolmogorov-Smirnov สำหรับหน่วยทดลองมากกว่า 50 หน่วย และ Shapiro-Wilk สำหรับหน่วยทดลองน้อยกว่า 50 หน่วย (กัลยา วานิชย์บัญชา, 2548) สำหรับงานวิจัยนี้ตัวอย่างในแต่ละกลุ่มน้อยกว่า 50 หน่วย ดังนั้นงานวิจัยนี้จึงใช้เทคนิค Shapiro-Wilk ในการตรวจสอบการแจกแจงข้อมูล โดยจะเลือกทดสอบสมมติฐานแบบอิงพารามิเตอร์ (Parametric Test) เมื่อค่า Sig. (Significance) ของการทดสอบน้อยกว่าระดับนัยสำคัญที่กำหนด โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05 ดังนั้นการทดสอบสมมติฐานแบบอิงพารามิเตอร์ใช้วิธีการทดสอบสมมติฐานเกี่ยวกับผลต่างระหว่างค่าเฉลี่ย 2 ประชากรแบบจับคู่ (Paired t-test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการหาค่าผลต่างระหว่างค่ามาตรวัดเชิงวัตถุของหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงที่ต่างกัน (กัลยา วานิชย์บัญชา, 2548) ในทางตรงกันข้ามจะเลือกทดสอบสมมติฐานแบบไม่อิงพารามิเตอร์ (Nonparametric Test) เมื่อค่า Sig. ของการทดสอบมากกว่าหรือเท่ากับระดับนัยสำคัญที่กำหนด โดยงานวิจัยนี้กำหนดระดับนัยสำคัญเท่ากับ 0.05 ดังนั้นการทดสอบสมมติฐานแบบอิงพารามิเตอร์ทางผู้วิจัยเลือกใช้วิธีการทดสอบสมมติฐานเกี่ยวกับการแจกแจงของข้อมูล 2 ชุดที่สัมพันธ์กัน (Two-Related-Samples Tests) โดยเลือกใช้วิธีการทดสอบประเภทวิลคอกสัน ไซน์-แรงค์ เทส (Wilcoxon signed-rank test) เนื่องจากการทดสอบสมมติฐานนี้เป็นการหาค่าผลต่างระหว่างค่ามาตรวัดเชิงวัตถุของหน่วยทดลองเดียวกันที่ผ่านกระบวนการรีแฟคทอริงที่ต่างกัน (กัลยา วานิชย์บัญชา, 2548)

จากตารางที่แสดงไว้ในภาคผนวก ข แสดงค่าการแจกแจงวิธีการทดสอบทางสถิติสำหรับของค่ามาตรวัดเชิงวัตถุทั้ง 14 ค่า ตามผลการทดสอบการแจกแจงข้อมูลว่าเป็นแบบปกติหรือไม่ ตามสมมติฐานข้างต้นคือ ตามสมมติฐานข้อที่ 1 มี 9 รูปแบบ ตามสมมติฐานข้อที่ 2 มี 72 รูปแบบ และตามสมมติฐานข้อที่ 3 มี 36 รูปแบบ สามารถแสดงผลการสรุปการทดสอบการแจกแจงข้อมูลว่าเป็นแบบปกติของสมมติฐานทั้ง 3 ข้อ ได้ดังนี้

1. สมมติฐานข้อที่ 1 ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ว่าเป็นการแจกแจงแบบปกติหรือไม่

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
1	Extract Method	10	2	2
2	Replace Temp with Query	4	4	6
3	Introduce Explaining Variable	8	0	6
4	Move Method	10	4	0
5	Move Field	8	6	0
6	Extract Class	11	3	0
7	Replace Magic Number with Symbolic Constant	7	0	7
8	Decompose Conditional	7	0	7
9	Rename Method	11	0	3

ตารางที่ 4-5 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง

จากตารางที่ 4-5 จะเห็นได้ว่ากระบวนการรีแฟคทอริงแต่ละรูปแบบตามสมมติฐานข้อที่ 1 จะมีค่ามาตรวัดเชิงวัตถุที่ต้องทดสอบด้วยการทดสอบสมมติฐานแบบอิงพารามิเตอร์และแบบไม่อิงพารามิเตอร์ และสำหรับค่ามาตรวัดเชิงวัตถุที่ไม่สามารถวิเคราะห์ผลการแจกแจงข้อมูลว่าเป็นปกติได้ เนื่องจากค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองที่ผ่านกระบวนการรีแฟคทอริงตามรูปแบบ

ในสมมติฐานข้อที่ 1 นี้ไม่มีความแตกต่างระหว่างค่ามาตรวัดเชิงวัตถุของหน่วยทดลองเดียวกัน (สามารถดูรายละเอียดผลการแจกแจงข้อมูลว่าเป็นปกติของสมมติฐานข้อที่ 1 ได้ที่ภาคผนวก ค)

2. สมมติฐานข้อที่ 2 ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x กับหลังผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y ว่าเป็นการแจกแจงแบบปกติหรือไม่

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
1	(Extract Method + Replace Temp with Query) – Extract Method	4	4	6
2	(Extract Method + Introduce Explaining Variable) - Extract Method	8	0	6
3	(Extract Method + Move Method) - Extract Method	9	5	0
4	(Extract Method + Move Field) - Extract Method	7	6	1
5	(Extract Method + Extract Class) - Extract Method	7	4	3
6	(Extract Method + Replace Magic Number with Symbolic Constant) - Extract Method	7	0	7
7	(Extract Method + Decompose Condition) - Extract Method	8	2	4
8	(Extract Method + Rename Method) - Extract Method	11	0	3
9	(Replace Temp with Query + Extract Method) - Replace Temp with Query	5	6	3
10	(Replace Temp with Query + Introduce Explaining Variable) - Replace Temp with Query	1	2	11

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
11	(Replace Temp with Query + Move Method) - Replace Temp with Query	4	9	1
12	(Replace Temp with Query + Move Field) - Replace Temp with Query	4	8	2
13	(Replace Temp with Query + Extract Class) - Replace Temp with Query	4	6	4
14	(Replace Temp with Query + Replace Magic Number with Symbolic Constant) - Replace Temp with Query	3	4	7
15	(Replace Temp with Query + Decompose - Conditional) Replace Temp with Query	2	8	4
16	(Replace Temp with Query + Rename Method) - Replace Temp with Query	14	0	0
17	(Introduce Explaining Variable + Extract Method) - Introduce Explaining Variable	6	4	4
18	(Introduce Explaining Variable + Replace Temp with Query) - Introduce Explaining Variable	2	5	7
19	(Introduce Explaining Variable + Move Method) - Introduce Explaining Variable	5	7	2
20	(Introduce Explaining Variable + Move Field) - Introduce Explaining Variable	4	8	2
21	(Introduce Explaining Variable + Extract Class) - Introduce Explaining Variable	10	3	1
22	(Introduce Explaining Variable + Replace Magic Number with Symbolic Constant) - Introduce Explaining Variable	11	1	2

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
23	(Introduce Explaining Variable + Decompose Conditional) - Introduce Explaining Variable	6	4	4
24	(Introduce Explaining Variable + Rename Method) - Introduce Explaining Variable	7	1	6
25	(Move Method + Extract Method) - Move Method	8	2	4
26	(Move Method + Replace Temp with Query) - Move Method	6	4	4
27	(Move Method + Introduce Explaining Variable) - Move Method	8	2	4
28	(Move Method + Move Field) - Move Method	3	0	11
29	(Move Method + Extract Class) - Move Method	4	0	10
30	(Move Method + Replace Magic Number with Symbolic Constant) - Move Method	6	1	7
31	(Move Method + Decompose Conditional) - Move Method	5	5	4
32	(Move Method + Rename Method) - Move Method	14	0	0
33	(Move Field + Extract Method) - Move Field	8	3	3
34	(Move Field + Replace Temp with Query) - Move Field	3	5	6
35	(Move Field + Introduce Explaining Variable) - Move Field	6	4	4
36	(Move Field + Move Method) - Move Field	10	0	4
37	(Move Field + Extract Class) - Move Field	5	0	9

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
38	(Move Field + Replace Magic Number with Symbolic Constant) - Move Field	6	1	7
39	(Move Field + Decompose Conditional) - Move Field	6	4	4
40	(Move Field + Rename Method) - Move Field	10	0	4
41	(Extract Class + Extract Method) - Extract Class	9	2	3
42	(Extract Class + Replace Temp with Query) - Extract Class	2	5	7
43	(Extract Class + Introduce Explaining Variable) - Extract Class	13	0	1
44	(Extract Class + Move Method) - Extract Class	9	1	4
45	(Extract Class + Move Field) - Extract Class	9	1	4
46	(Extract Class + Replace Magic Number with Symbolic Constant) - Extract Class	6	1	7
47	(Extract Class + Decompose Conditional) - Extract Class	8	2	4
48	(Extract Class + Rename Method) - Extract Class	11	0	3
49	(Replace Magic Number with Symbolic Constant + Extract Method) - Replace Magic Number with Symbolic Constant	9	2	3
50	(Replace Magic Number with Symbolic Constant + Replace Temp with Query) - Replace Magic Number with Symbolic Constant	2	5	7
51	(Replace Magic Number with Symbolic Constant + Introduce Explaining Variable) - Replace Magic Number with Symbolic Constant	12	0	2

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
52	(Replace Magic Number with Symbolic Constant + Move Method) - Replace Magic Number with Symbolic Constant	7	6	1
53	(Replace Magic Number with Symbolic Constant + Move Field) - Replace Magic Number with Symbolic Constant	6	7	1
54	(Replace Magic Number with Symbolic Constant + Extract Class) - Replace Magic Number with Symbolic Constant	8	3	3
55	(Replace Magic Number with Symbolic Constant + Decompose Conditional) - Replace Magic Number with Symbolic Constant	9	3	2
56	(Replace Magic Number with Symbolic Constant + Rename Method) - Replace Magic Number with Symbolic Constant	8	0	6
57	(Decompose Conditional + Extract Method) - Decompose Conditional	9	2	3
58	(Decompose Conditional + Replace Temp with Query) - Decompose Conditional	3	6	5
59	(Decompose Conditional + Introduce Explaining Variable) - Decompose Conditional	6	4	4
60	(Decompose Conditional + Move Method) - Decompose Conditional	7	6	1
61	(Decompose Conditional + Move Field) - Decompose Conditional	4	9	1

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
62	(Decompose Conditional + Extract Class) - Decompose Conditional	7	4	3
63	(Decompose Conditional + Replace Magic Number with Symbolic Constant) - Decompose Conditional	11	1	2
64	(Decompose Conditional + Rename Method) - Decompose Conditional	10	0	4
65	(Rename Method + Extract Method) - Rename Method	8	2	4
66	(Rename Method + Replace Temp with Query) - Rename Method	8	4	2
67	(Rename Method + Introduce Explaining Variable) - Rename Method	5	1	8
68	(Rename Method + Move Method) -Rename Method	11	3	0
69	(Rename Method + Move Field) - Rename Method	7	6	1
70	(Rename Method + Extract Class) - Rename Method	11	1	2
71	(Rename Method + Replace Magic Number with Symbolic Constant) - Rename Method	8	0	6
72	(Rename Method + Decompose Conditional) - Rename Method	8	2	4

ตารางที่ 4-6 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี (ต่อ)

จากตารางที่ 4-6 จะเห็นได้ว่ากระบวนการรีแฟคทอริงแต่ละรูปแบบตามสมมติฐานข้อที่ 2 จะมีค่ามาตรวัดเชิงวัตถุที่ต้องทดสอบด้วยการทดสอบสมมติฐานแบบอิงพารามิเตอร์และแบบไม่อิงพารามิเตอร์ และสำหรับค่ามาตรวัดเชิงวัตถุที่ไม่สามารถวิเคราะห์ผลการแจกแจงข้อมูลว่าเป็นปกติได้ เนื่องจากค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองที่ผ่านกระบวนการรีแฟคทอริงตามรูปแบบในสมมติฐานข้อที่ 2 นี้ไม่มีความแตกต่างเลยระหว่างค่ามาตรวัดเชิงวัตถุของหน่วยทดลองเดียวกัน (สามารถดูรายละเอียดผลการแจกแจงข้อมูลว่าเป็นปกติของสมมติฐานข้อที่ 2 ได้ที่ภาคผนวก ข)

3. สมมติฐานข้อที่ 3 ความแตกต่างของค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y กับหลังผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x ว่าเป็นการแจกแจงแบบปกติหรือไม่

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
1	(Extract Method + Replace Temp with Query) - (Replace Temp with Query + Extract Method)	1	0	13
2	(Extract Method + Introduce Explaining Variable) - (Introduce Explaining Variable + Extract Method)	1	0	13
3	(Extract Method + Move Method) - (Move Method + Extract Method)	1	0	13
4	(Extract Method + Move Field) - (Move Field + Extract Method)	1	0	13
5	(Extract Method + Extract Class) - (Extract Class + Extract Method)	1	0	13
6	(Extract Method + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Extract Method)	1	0	13
7	(Extract Method + Decompose Conditional) - (Decompose Conditional + Extract Method)	1	0	13

ตารางที่ 4-7 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีสลับลำดับกัน

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
8	(Extract Method + Rename Method) - (Rename Method + Extract Method)	1	0	13
9	(Replace Temp with Query + Introduce Explaining Variable) - (Introduce Explaining Variable + Replace Temp with Query)	0	0	14
10	(Replace Temp with Query + Move Method) - (Move Method + Replace Temp with Query)	0	0	14
11	(Replace Temp with Query + Move Field) - (Move Field + Replace Temp with Query)	0	0	14
12	(Replace Temp with Query + Extract Class) - (Extract Class + Replace Temp with Query)	0	0	14
13	(Replace Temp with Query + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Replace Temp with Query)	0	0	14
14	(Replace Temp with Query + Decompose Conditional) - (Decompose Conditional + Replace Temp with Query)	0	0	14
15	(Replace Temp with Query + Rename Method) - (Rename Method + Replace Temp with Query)	0	0	14
16	(Introduce Explaining Variable + Move Method) - (Move Method + Introduce Explaining Variable)	0	0	14
17	(Introduce Explaining Variable + Move Field) - (Move Field + Introduce Explaining Variable)	0	0	14

ตารางที่ 4-7 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีสลับลำดับกัน (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
18	(Introduce Explaining Variable + Extract Class) - (Extract Class + Introduce Explaining Variable)	0	0	14
19	(Introduce Explaining Variable + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Introduce Explaining Variable)	0	0	14
20	(Introduce Explaining Variable + Decompose Conditional) - (Decompose Conditional + Introduce Explaining Variable)	0	0	14
21	(Introduce Explaining Variable + Rename Method) - (Rename Method + Introduce Explaining Variable)	0	0	14
22	(Move Method + Move Field) - (Move Field + Move Method)	0	0	14
23	(Move Method + Extract Class) - (Extract Class + Move Method)	0	0	14
24	(Move Method + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Move Method)	0	0	14
25	(Move Method + Decompose Conditional) - (Decompose Conditional + Move Method)	0	0	14
26	(Move Method + Rename Method) - (Rename Method + Move Method)	0	0	14
27	(Move Field + Extract Class) - (Extract Class + Move Field)	0	0	14

ตารางที่ 4-7 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีสลับลำดับกัน (ต่อ)

ลำดับ	กระบวนการรีแฟคทอริง	อิงพารามิเตอร์	ไม่อิงพารามิเตอร์	วิเคราะห์ผลไม่ได้
28	(Move Field + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Move Field)	0	0	14
29	(Move Field + Decompose Conditional) - (Decompose Conditional + Move Field)	0	0	14
30	(Move Field + Rename Method) - (Rename Method + Move Field)	0	0	14
31	(Extract Class + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Extract Class)	0	0	14
32	(Extract Class + Decompose Condition) - (Decompose Condition + Extract Class)	0	0	14
33	(Extract Class + Rename Method) - (Rename Method + Extract Class)	0	0	14
34	(Replace Magic Number with Symbolic Constant + Decompose Conditional) - (Decompose Conditional + Replace Magic Number with Symbolic Constant)	0	0	14
35	(Replace Magic Number with Symbolic Constant + Rename Method) - (Rename Method + Replace Magic Number with Symbolic Constant)	0	0	14
36	(Decompose Conditional + Rename Method) - (Rename Method + Decompose Conditional)	0	0	14

ตารางที่ 4-7 แสดงตารางจำนวนวิธีการทดสอบทางสถิติสำหรับค่ามาตรวัดเชิงวัตถุ 14 ค่า ตามผลการทดลองว่าค่ามาตรวัดเชิงวัตถุมีการแจกแจงเป็นแบบปกติหรือไม่ สำหรับการเปรียบเทียบค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีสลับลำดับกัน (ต่อ)

จากตารางที่ 4-7 จะเห็นได้ว่ากระบวนการรีแฟคทอริงตามสมมติฐานข้อที่ 3 จะสามารถทดสอบค่ามาตรวัดเชิงวัตถุได้เพียง 1 มาตรวัด สำหรับการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ แบบอิงพารามิเตอร์ ส่วนค่ามาตรวัดเชิงวัตถุสำหรับกระบวนการรีแฟคทอริงตั้งแต่ลำดับที่ 9-36 ที่เหลือไม่สามารถวิเคราะห์ผลการแจกแจงข้อมูลว่าเป็นปกติได้ เนื่องจากค่ามาตรวัดเชิงวัตถุจากหน่วยทดลองที่ผ่านกระบวนการรีแฟคทอริง 2 วิธีทำสลับลำดับกันตามสมมติฐานข้อที่ 3 นี้ไม่มีความแตกต่างในค่ามาตรวัดเชิงวัตถุของหน่วยทดลองเดียวกัน (สามารถดูรายละเอียดผลการแจกแจงข้อมูลว่าเป็นปกติของสมมติฐานข้อที่ 3 ได้ที่ภาคผนวก ค ค่ามาตรวัดเชิงวัตถุที่สามารถทดสอบได้ตามสมมติฐานที่ 3 นี้ คือมาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size))

4.3 การเปรียบเทียบค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง

ผู้วิจัยต้องการเปรียบเทียบค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ซึ่งการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุนั้นแบ่งเป็น 2 กรณีคือ 1) มาตรวัดเชิงวัตถุที่มีการแจกแจงแบบปกติ 2) มาตรวัดเชิงวัตถุที่ไม่ได้มีการแจกแจงแบบปกติ สำหรับกรณีที่ 1 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับผลต่างระหว่างค่าเฉลี่ย 2 ประชากรแบบจับคู่ (Paired t-test) และสำหรับกรณีที่ 2 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับการแจกแจงของข้อมูล 2 ชุดที่สัมพันธ์กัน (Two-Related-Samples Tests) โดยทั้ง 2 กรณีจะใช้โปรแกรม SPSS ในการทดสอบ

การเปรียบเทียบค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง โดยได้แยกการทดสอบสมมติฐานเป็น 2 กรณี ไว้ดังนี้

1. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุเพิ่มขึ้น แล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \leq \mu_1$$
$$H_1: \mu_2 > \mu_1$$

2. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุลดลง แล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \geq \mu_1$$
$$H_1: \mu_2 < \mu_1$$

โดย μ_1 หมายถึงค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงและ μ_2 หมายถึงค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง เนื่องจากผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่ได้หลังจากผ่านการทำกระบวนการรีแฟคทอริงจะมีคุณภาพซอฟต์แวร์ดีกว่าซอร์สโค้ดที่ไม่ผ่านการทำกระบวนการรีแฟคทอริง โดยการทดสอบค่ามาตรวัดเชิงวัตถุทั้ง 2 กรณี จะแสดงไว้ในตารางที่ 4-8

หากผลการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุเป็นแบบปกติ สำหรับสมมติฐานกรณีที่ 1 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $t > 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548) และสำหรับสมมติฐานกรณีที่ 2 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $t < 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548)

หากผลการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุไม่เป็นแบบปกติ สำหรับสมมติฐานกรณีที่ 1 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Asymp.Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $Z > 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548) และสำหรับสมมติฐานกรณีที่ 2 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Asymp.Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $Z < 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548)

ซึ่งจากตารางที่แสดงไว้ในภาคผนวก แสดงให้เห็นว่าค่ามาตรวัดเชิงวัตถุทั้ง 14 ค่า สามารถยอมรับ H_0 และปฏิเสธ H_0 โดยสามารถสรุปได้ดังตารางต่อไปนี้

โดยกำหนดให้ เครื่องหมายขีดคั่น (-) หมายถึงผลต่างของค่ามาตรวัดเชิงวัตถุมีค่าไม่แตกต่างกัน ทำให้ไม่สามารถวิเคราะห์ผลทางสถิติได้

เครื่องหมายดอกจัน (*) หมายถึงผลของค่ามาตรวัดเชิงวัตถุที่ปฏิเสธ H_0

เครื่องหมายที (T) หมายถึงผลของค่ามาตรวัดเชิงวัตถุที่ลดลงสำหรับการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

สดมภ์สีขาว คือสดมภ์มาตรวัดเชิงวัตถุที่ค่าน้อยหมายถึงคุณภาพซอฟต์แวร์ดี

สดมภ์สีเทา คือสดมภ์มาตรวัดเชิงวัตถุที่ค่ามากหมายถึงคุณภาพซอฟต์แวร์ดี

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	.000[T]	.401	-	-	0.0055*	0.1585	.000*	.000*	.000*	0.1585	.000[T]	0.129	0.1585	0.0025*
	t, Z	-3.490[T]	-.840	-	-	-2.533*	-1.000	-4.266*	-4.372*	8.435*	-1.000	-7.622[T]	-1.069	-1.000	-2.796*
2. Replace Temp with Query	Sig.	0.005[T]	-	-	-	0.1725	-	0.0375*	0.0045*	.000*	-	.000[T]	0.5	-	0.2525
	t, Z	-3.101[T]	-	-	-	-.943	-	-1.779*	-3.141*	6.253*	-	-5.517[T]	.000	-	-.667
3. Introduce Explaining Variable	Sig.	0.0025[T]	0.09	-	-	0.1585	-	0.2325	0.001*	-	0.3275	-	0.3275	-	0.1585
	t, Z	-2.803[T]	-1.342	-	-	-1.000	-	-.730	-3.059*	-	-.447	-	-.447	-	-1.000
4. Move Method	Sig.	.000[T]	.000*	.000*	.000*	0.2085	.000*	.000*	.000*	0.09	0.0545	0.0025[T]	0.343	0.1425	0.002*
	t, Z	-4.082[T]	-4.107*	-7.233*	-6.621*	-.828	-3.724*	-4.106*	-3.523*	-1.342	-1.604	-2.800[T]	-.405	-1.069	-2.890*
5. Move Field	Sig.	0.0005[T]	.000*	.000*	.000*	.000*	.000*	0.0035[T]	0.0005*	0.09	0.034[T]	0.0055[T]	0.2325	0.3275	0.0005*
	t, Z	-3.783[T]	-4.015*	-8.207*	-7.339*	-4.464*	-4.741*	3.032[T]	-3.389*	-1.342	-1.826[T]	-2.533[T]	-.730	-.447	-3.320*
6. Extract Class	Sig.	.000[T]	.000*	.000*	.000*	.000*	.000*	0.007*	.000*	0.1585	0.0545	0.3275	0.2965	0.3275	.000*
	t, Z	-4.903[T]	-4.107*	-7.651*	-6.935*	-4.107*	-3.724*	-2.451*	-4.107*	-1.000	-1.604	-.447	-.535	-.447	-3.490*
7. Replace Magic Number with Symbolic Constant	Sig.	0.004[T]	-	-	-	0.088	-	0.2965	.000*	-	0.019[T]	-	0.002*	-	0.0455*
	t, Z	-2.668[T]	-	-	-	-1.352	-	-.535	-3.823*	-	-2.073[T]	-	-2.919*	-	-1.690*
8. Decompose Conditional	Sig.	0.0025[T]	-	-	-	0.0615	-	0.343	.000*	-	0.011[T]	-	0.001*	-	0.025*
	t, Z	-2.840[T]	-	-	-	-1.540	-	-.405	-4.107*	-	-2.293[T]	-	-3.103*	-	-1.960*
9. Rename Method	Sig.	0.2325	0.5	-	-	0.123	0.1585	0.0805	0.072	-	0.1425	0.09	0.0865	0.1585	0.4155
	t, Z	-.730	.000	-	-	-1.160	-1.000	-1.400	-1.461	-	-1.069	-1.342	-1.363	-1.000	-.213

ตารางที่ 4-8 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของสมมติฐานที่ 1

102

จากตารางที่ 4-8 เป็นตารางที่แสดงค่า Sig. (1-tailed) และค่าสถิติ t หรือ Z ของมาตรวัดเชิงวัตถุ เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของสมมติฐานที่ 1 จะเห็นได้ว่ากระบวนการรีแฟคทอริงโดยรวมไม่ได้ทำให้ค่ามาตรวัดเชิงวัตถุทุกตัวดีขึ้น ซึ่งในกระบวนการรีแฟคทอริงในแต่ละวิธีจะมีส่วนทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นแตกต่างกัน แต่สำหรับการทดสอบสมมติฐานที่ 1 นี้มีมาตรวัดเชิงวัตถุ 4 ชนิด ที่ไม่มีค่าดีขึ้นคือ มาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) มาตรวัดแอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) มาตรวัดเมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) และมาตรวัดโพลีมอร์ฟีซึมแฟคเทอร์ (Polymorphism Factor: POF)

สำหรับกระบวนการรีแฟคทอริงในแต่ละวิธี จะเห็นได้ว่าการเปรียบเทียบของหน่วยทดลองตามสมมติฐานที่ 1 ค่ามาตรวัดเชิงวัตถุจะดีขึ้นกับกระบวนการรีแฟคทอริงแต่ละวิธีไม่เท่ากัน แต่โดยรวมแล้วจะมีการดีขึ้นของค่ามาตรวัดเชิงวัตถุ ยกเว้นกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด ซึ่งผู้วิจัยเห็นว่ากระบวนการรีแฟคทอริงที่ทำให้ค่ามาตรวัดเชิงวัตถุมีค่าดีขึ้นอย่างมีนัยสำคัญในสมมติฐานที่ 1 (ซึ่งจะพบได้อีกในสมมติฐานที่ 2 ผู้วิจัยจะกล่าวต่อไป) คือ

- กระบวนการรีแฟคทอริงเกือบทุกวิธีมีผลทำให้มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) มีค่าดีขึ้น เนื่องจากกระบวนการรีแฟคทอริงมีส่วนช่วยให้เมท็อดในแต่ละคลาสมีซอร์สโค้ดที่สั้นและกระชับมากขึ้น โดยเฉพาะอย่างยิ่งกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี และกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอล จะเป็นกระบวนการรีแฟคทอริงที่เกี่ยวข้องกับเมท็อดในแต่ละคลาสโดยตรงจึงมีผลให้ค่ามาตรวัดชนิดอื่นๆ มีค่าดีขึ้นด้วยคือ มาตรวัดแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) มาตรวัดเมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) และมาตรวัดคัพพลิงแฟคเทอร์ (Coupling Factor: COF)
- กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์และกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส มีผลทำให้มาตรวัดเวทเทดเมท็อดเปอคลาส (Weighted Methods per Class: WMC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดเดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree: DIT) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดนัมเบอร์ออฟชิลเดรน (Number of Children: NOC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดคัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) มี 2 วิธีมีค่าดีขึ้น โดยยกเว้นกระบวนการรีแฟคทอริงวิธีมูฟเมท็อด มาตรวัดเรสปอนซ์ฟอร์อะคลาส (Response for a Class: RFC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) มี 2 วิธีมีค่าดีขึ้น โดยยกเว้นกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) ทั้ง 3 วิธีมีค่า

ดีขึ้นและมาตรวัดคัพพลิงแฟคเทอร์ (Coupling Factor: COF) ทั้ง 3 วิธีมีค่าดีขึ้น ผู้วิจัยเห็นว่าการที่ผลการทดลองมีค่าที่ใกล้เคียงกันเนื่องจากหลักการในการทำกระบวนการรีแฟคทอริงทั้ง 3 วิธีนี้ ซึ่งอยู่ในกระบวนการรีแฟคทอริงประเภทเดียวกันจำเป็นต้องมีการเพิ่มคลาสขึ้นในซอร์สโค้ด ซึ่งทำให้โครงสร้างของระบบเปลี่ยนแปลงไปมากกว่ากระบวนการรีแฟคทอริงวิธีอื่นๆ ดังนั้นค่ามาตรวัดเชิงวัตถุที่ดีขึ้นก็จะมีส่วนที่สอดคล้องกันและกระทบกับมาตรวัดเชิงวัตถุหลายชนิดดังกล่าว

- กระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนต์ และกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอล มีผลทำให้มาตรวัดแอตทริบิวต์อินเฮอริแทนแฟคเทอร์ (Attribute Inheritance Factor: AIF) มีค่าดีขึ้น เนื่องจากกระบวนการรีแฟคทอริง 2 วิธีนี้มีส่วนที่เกี่ยวข้องกับแอตทริบิวต์ในแต่ละคลาสคือ การจัดการให้แอตทริบิวต์ในแต่ละคลาสถูกเรียกใช้งานได้เหมาะสมมากขึ้น

4.4 การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี เปรียบเทียบกับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี

ผู้วิจัยต้องการเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี เปรียบเทียบกับการทำกระบวนการรีแฟคทอริง 2 วิธี ซึ่งการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุนั้นแบ่งเป็น 2 กรณีคือ 1) มาตรวัดเชิงวัตถุที่มีการแจกแจงแบบปกติ 2) มาตรวัดเชิงวัตถุที่ไม่ได้มีการแจกแจงแบบปกติ สำหรับกรณีที่ 1 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับผลต่างระหว่างค่าเฉลี่ย 2 ประชากรแบบจับคู่ (Paired t-test) และสำหรับกรณีที่ 2 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับการแจกแจงของข้อมูล 2 ชุดที่สัมพันธ์กัน (Two-Related-Samples Tests) โดยทั้ง 2 กรณีจะใช้โปรแกรม SPSS ในการทดสอบ

การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริง 1 วิธี เปรียบเทียบกับการทำกระบวนการรีแฟคทอริง 2 วิธี โดยได้แยกการทดสอบสมมติฐานเป็น 2 กรณี ไว้ดังนี้

1. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุเพิ่มขึ้น แล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \leq \mu_1$$

$$H_1: \mu_2 > \mu_1$$

2. การทดสอบสมมติฐานกรณีที่เมื่อค่ามาตรวัดเชิงวัตถุลดลง แล้วส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้น ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \geq \mu_1$$

$$H_1: \mu_2 < \mu_1$$

โดย μ_1 หมายถึงค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x และ μ_2 หมายถึงค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y เนื่องจากผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่ได้หลังจากผ่านการทำกระบวนการรีแฟคทอริง 2 วิธี ย่อมจะมีคุณภาพซอฟต์แวร์ดีกว่าซอร์สโค้ดที่ผ่านการทำกระบวนการรีแฟคทอริง 1 วิธี โดยการทดสอบค่ามาตรวัดเชิงวัตถุทั้ง 2 กรณี จะแสดงไว้ในตารางที่ 4-9 ถึงตารางที่ 4-17

หากผลการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุเป็นแบบปกติ สำหรับสมมติฐานกรณีที่ 1 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $t > 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548) และสำหรับสมมติฐานกรณีที่ 2 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $t < 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548)

หากผลการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุไม่เป็นแบบปกติ สำหรับสมมติฐานกรณีที่ 1 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Asymp.Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $Z > 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548) และสำหรับสมมติฐานกรณีที่ 2 จะปฏิเสธสมมติฐานเมื่อค่า $\frac{\text{Asymp.Sig.(2-tailed)}}{2} <$ ค่าระดับนัยสำคัญ และค่าสถิติ $Z < 0$ โดยที่เงื่อนไขทั้งสองข้อนี้ต้องเป็นจริงทั้งคู่จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548)

ซึ่งจากตารางที่แสดงไว้ในภาคผนวก แสดงให้เห็นว่าค่ามาตรวัดเชิงวัตถุทั้ง 14 ค่า สามารถยอมรับ H_0 และปฏิเสธ H_0 โดยสามารถสรุปได้ดังตารางต่อไปนี้

โดยกำหนดให้ เครื่องหมายขีดคั่น (-) หมายถึงผลต่างของค่ามาตรวัดเชิงวัตถุมีค่าไม่แตกต่างกัน ทำให้ไม่สามารถวิเคราะห์ผลทางสถิติได้

เครื่องหมายดอกจัน (*) หมายถึงผลของค่ามาตรวัดเชิงวัตถุที่ปฏิเสธ H_0

เครื่องหมายที (T) หมายถึงผลของค่ามาตรวัดเชิงวัตถุที่ลดลงสำหรับการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

สดมภ์สีขาว คือสดมภ์มาตรวัดเชิงวัตถุที่ค่าน้อยหมายถึงคุณภาพซอฟต์แวร์ดี

สดมภ์สีเทา คือสดมภ์มาตรวัดเชิงวัตถุที่ค่ามากหมายถึงคุณภาพซอฟต์แวร์ดี

1. กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Replace Temp with Query	Sig.	0.0085 T	-	-	-	0.1585	-	0.2875	0.0115*	0.005*	-	0.01 T	0.09	-	0.1585
	t, Z	-2.993 T	-	-	-	-1.000	-	-.560	-2.803*	5.257*	-	-4.378 T	-1.342	-	-1.000
2. Introduce Explaining Variable	Sig.	0.0185 T	0.1585	-	-	0.1585	-	0.04*	0.0015*	-	0.1425	-	0.1115	-	0.1585
	t, Z	-2.090 T	-1.000	-	-	-1.000	-	-1.753*	-2.934*	-	-1.069	-	-1.219	-	-1.000
3. Move Method	Sig.	0.0005 T	.000*	.000*	.000*	.000*	.000*	.000*	.000*	.000 T	0.3275	.000*	0.5	0.1585	.000*
	t, Z	-3.737 T	-4.107*	-7.651*	-6.935*	-3.997*	-3.724*	-4.107*	-4.074*	-7.978 T	-.447	7.981*	.000	-1.000	-4.107*
4. Move Field	Sig.	0.0015 T	.000*	.000*	.000*	.000*	.000*	.000*	.000*	.000 T	0.1425	.000*	0.072	-	.000*
	t, Z	-3.344 T	-3.920*	-8.207*	-7.339*	-3.808*	-4.741*	-3.920*	-3.808*	-6.663 T	-1.069	6.651*	-1.461	-	-3.920*
5. Extract Class	Sig.	.000 T	.000*	.000*	.000*	.000*	.000*	.000*	.000*	-	0.1425	-	0.072	-	.000*
	t, Z	-4.886 T	-4.107*	-7.651*	-6.935*	-4.107*	-3.724*	-4.108*	-12.676*	-	-1.069	-	-1.461	-	-4.107*
6. Replace Magic Number with Symbolic Constant	Sig.	0.0005 T	-	-	-	0.014*	-	0.1365	.000*	-	0.006 T	-	0.0005*	-	0.014*
	t, Z	-3.296 T	-	-	-	-2.201*	-	-1.095	-3.501*	-	-2.521 T	-	-3.180*	-	-2.201*
7. Decompose Conditional	Sig.	0.0005 T	0.091	-	-	0.278	-	.000*	.000*	-	0.1425	.000 T	0.1425	-	0.158
	t, Z	-3.300 T	-1.334	-	-	-.588	-	-3.632*	-4.015*	-	-1.069	-7.522 T	-1.069	-	-1.002
8. Rename Method	Sig.	0.064 T	0.5	-	-	0.136	-	0.104	0.006*	0.0545	0.5	0.034 T	0.4585	0.1585	0.4625
	t, Z	-1.521 T	.000	-	-	-1.099	-	-1.260	-2.521*	-1.604	.000	-1.826 T	-.105	-1.000	-.094

ตารางที่ 4-9 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

2. กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.006 [T]	0.4375	-	-	0.2325	-	0.0105 [T]	0.0095*	.000*	0.1585	0.0005 [T]	0.1585	0.1585	0.3555
	t, Z	-2.521 [T]	-.162	-	-	-.730	-	2.869 [T]	-2.927*	5.875*	-1.000	-5.176 [T]	-1.000	-1.000	.384
2. Introduce Explaining Variable	Sig.	0.1895	-	-	-	-	-	0.1585	0.0465*	-	-	-	-	-	-
	t, Z	-1.031	-	-	-	-	-	-1.000	2.428*	-	-	-	-	-	-
3. Move Method	Sig.	0.0085 [T]	0.002*	.000*	0.0005*	0.4325	0.0055*	0.2965	0.006*	0.2345	0.1585	0.1425	0.09	-	0.006*
	t, Z	-3.113 [T]	-4.248*	-6.235*	-5.821*	-.176	-3.434*	.560	-2.521*	-.766	-1.000	-1.158	-1.342	-	-2.521*
4. Move Field	Sig.	0.0095 [T]	0.002*	.000*	0.0005*	0.0165*	0.0055*	0.346	0.006*	0.009 [T]	-	0.2285	0.1585	-	0.006*
	t, Z	-3.053 [T]	-4.248*	-6.235*	-5.821*	-2.659*	-3.434*	-.413	-2.521*	-2.366 [T]	-	-.787	-1.000	-	-2.521*
5. Extract Class	Sig.	0.0055 [T]	0.002*	.000*	0.0005*	0.012*	0.0055*	0.0465*	0.006*	-	-	-	0.1585	-	0.006*
	t, Z	-3.413 [T]	-4.231*	-6.235*	-5.821*	-2.857*	-3.434*	-1.680*	-2.521*	-	-	-	-1.000	-	-2.521*
6. Replace Magic Number with Symbolic Constant	Sig.	0.498	-	-	-	0.0515	-	0.1585	0.008 [T]	-	0.0545	-	0.069	-	0.0545
	t, Z	-.005	-	-	-	1.919	-	-1.000	3.315 [T]	-	-1.604	-	-1.711	-	-1.604
7. Decompose Conditional	Sig.	0.005 [T]	0.0415*	-	-	0.166	-	0.3515	0.029*	0.001*	0.1585	0.0025 [T]	0.09	-	0.1215
	t, Z	-3.738 [T]	-2.081*	-	-	1.056	-	.400	-2.335*	5.204*	-1.000	-4.236 [T]	-1.342	-	1.296
8. Rename Method	Sig.	0.3275	0.1585	0.1585	0.1585	0.5	0.1585	0.2325	0.5	0.1585	0.1585	0.1425	0.2325	0.1585	0.3575
	t, Z	-.447	-1.000	-1.000	-1.000	.000	-1.000	-.730	.000	-1.000	-1.000	-1.069	-.730	-1.000	-.365

ตารางที่ 4-10 แสดงค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

3. กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิลกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.065	0.069	-	-	0.0615	-	0.005*	0.001*	.000*	0.3275	.000 T	0.2965	-	0.21
	t, Z	-1.627	-1.483	-	-	-1.540	-	-2.589*	-3.059*	5.622*	-.447	-5.225 T	-.535	-	.835
2. Replace Temp with Query	Sig.	0.0925	-	-	-	0.1585	-	0.3815	0.0505	0.0505	-	0.0475 T	-	-	0.1585
	t, Z	-1.715	-	-	-	-1.000	-	-.331	-2.344	2.341	-	-2.410 T	-	-	-1.000
3. Move Method	Sig.	0.0135 T	0.0025*	0.0015*	0.0015*	0.1665	0.01*	0.0455*	0.0015*	-	0.3275	0.0055 T	0.2965	-	0.0015*
	t, Z	-2.597 T	-3.598*	-3.995*	-3.943*	-1.018	-2.770*	-1.689*	-2.934*	-	-.447	-2.547 T	-.535	-	-3.962*
4. Move Field	Sig.	0.0315 T	0.004*	0.0015*	0.0015*	0.0975	0.01*	0.064	0.0015*	-	0.3275	0.006 T	0.3275	-	0.004*
	t, Z	-2.088 T	-3.303*	-3.995*	-3.943*	-1.390	-2.770*	1.661	-2.934*	-	-.447	-2.521 T	-.447	-	-3.314*
5. Extract Class	Sig.	0.1205	0.083	0.006*	0.0465*	0.025*	0.006*	0.026*	0.003*	-	0.09	0.09	0.0545	0.1585	0.025*
	t, Z	-1.172	-1.493	-2.527*	-1.680*	-1.956*	-2.521*	-2.208*	-3.443*	-	-1.342	-1.342	-1.604	-1.000	-1.956*
6. Replace Magic Number with Symbolic	Sig.	0.2575	0.09	0.1585	0.1585	0.4465	0.1585	0.3575	0.1045	-	0.3275	0.1585	0.193	-	0.343
	t, Z	-.625	-1.342	-1.000	-1.000	-.135	-1.000	-.365	-1.256	-	-.459	-1.000	-.866	-	-.405
7. Decompose Conditional	Sig.	0.0025 T	0.1035	-	-	0.0255*	-	0.2015	0.0015*	0.001	0.1585	0.0005 T	0.1585	-	0.058
	t, Z	-2.803 T	-1.261	-	-	-1.955*	-	.874	-2.934*	4.165	-1.000	-4.828 T	-1.000	-	-1.719
8. Rename Method	Sig.	0.3275	0.09	-	-	0.1395	-	0.2325	0.1425	-	0.3275	-	0.2965	-	0.2875
	t, Z	-.447	-1.342	-	-	-1.140	-	-.730	-1.069	-	-.447	-	-.535	-	-.561

108

ตารางที่ 4-11 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิลกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

4. กระบวนการรีแฟคทอริงวิธีมูฟเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.0035	0.034*	-	-	0.2225	-	0.0025*	.000*	.000*	0.5	.000 [T]	2.5	-	0.1425
	t, Z	-2.688	-	-	-	-.764	-	-2.799*	-4.136*	6.138*	.000	-6.050 [T]	-.674	-	-1.070
2. Replace Temp with Query	Sig.	0.2405	0.1585	-	-	0.3275	-	0.4655	0.3895	0.02*	0.1585	0.027 [T]	0.1585	-	0.3275
	t, Z	-7.44	-1.000	-	-	-.447	-	.089	-2.80	2.509*	-1.000	-2.314 [T]	-1.000	-	-.447
3. Introduce Explaining Variable	Sig.	0.02 [T]	0.3275	-	-	0.1915	-	0.5	0.0065*	0.1585	0.3275	0.09	0.3275	-	0.1365
	t, Z	-2.359	-.447	-	-	.913	-	.000	-2.490*	-1.000	-.447	-1.342	-.447	-	-1.095
4. Move Field	Sig.	-	-	-	-	-	-	0.1585	-	-	0.1585	-	0.09	-	-
	t, Z	-	-	-	-	-	-	-1.000	-	-	-1.000	-	-1.342	-	-
5. Extract Class	Sig.	-	-	-	-	-	-	0.3275	0.1585	-	0.1425	-	0.072	-	-
	t, Z	-	-	-	-	-	-	-.447	-1.000	-	-1.069	-	-1.461	-	-
6. Replace Magic Number with Symbolic Constant	Sig.	0.0005	-	-	-	0.0215*	-	0.3275	.000 [T]	-	0.009 [T]	-	0.001*	-	0.0215*
	t, Z	-3.180	-	-	-	-2.023*	-	-.447	5.714 [T]	-	-2.366 [T]	-	-3.059*	-	-2.023*
7. Decompose Conditional	Sig.	0.006 [T]	0.111	-	-	0.2675	-	0.001*	.000*	.000*	0.09	.000 [T]	0.09	-	0.3795
	t, Z	-2.800	-1.266	-	-	-.621	-	-3.099*	-3.823*	7.902*	-1.342	-7.147 [T]	-1.342	-	-.311
8. Rename Method	Sig.	0.3765	0.294	0.1585	0.1585	0.1435	0.1585	0.4585	0.266	0.3275	0.1425	0.4465	0.034*	0.1585	0.3265
	t, Z	-3.14	-.542	-1.000	-1.000	-1.065	-1.000	-.105	-.625	-.447	-1.069	-.135	-1.826*	-1.000	-.450

ตารางที่ 4-12 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

109

5. กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์กับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.1175	0.3765	-	-	0.0195*	-	0.0005*	.000*	.000*	0.09	.000 T	0.09	0.1585	0.0165*
	t, Z	-1.224	-.314	-	-	-2.063*	-	-3.324*	-4.015*	9.018*	-1.342	-8.102 T	-1.342	-1.000	-2.132*
2. Replace Temp with Query	Sig.	0.144	0.1585	-	-	0.1425	-	0.0635	0.1175	0.022*	-	0.041 T	-	-	0.1425
	t, Z	-1.151	-1.000	-	-	-1.069	-	-1.732	1.300	2.452*	-	-2.030 T	-	-	-1.069
3. Introduce Explaining Variable	Sig.	0.1825	0.5	-	-	0.25	-	0.4785	0.369	0.1585	0.0545	0.47	0.0545	-	0.3575
	t, Z	-.955	.000	-	-	-.674	-	.055	-.346	-1.000	-1.604	.077	-1.604	-	-.365
4. Move Method	Sig.	0.388	0.3275	-	-	0.0025*	-	0.01*	0.0005*	0.1585	0.0545	0.47	0.0545	-	0.0005*
	t, Z	-.284	-.447	-	-	-2.786*	-	-2.329*	-3.464*	-1.000	-1.604	-3.920	-1.604	-	-3.210*
5. Extract Class	Sig.	0.1585	-	-	-	-	-	0.1425	0.3275	-	0.0545	-	0.0545	-	-
	t, Z	-1.000	-	-	-	-	-	-1.069	-.447	-	-1.604	-	-1.604	-	-
6. Replace Magic Number with Symbolic	Sig.	0.0015 T	-	-	-	0.0215*	-	0.2325	.000 T	-	0.014 T	-	0.003*	-	0.0215*
	t, Z	-2.934 T	-	-	-	-2.023*	-	-.730	5.180 T	-	-2.201 T	-	-2.756*	-	-2.023*
7. Decompose Conditional	Sig.	0.0165	0.0295*	-	-	0.276	-	0.0045*	.000*	.000*	0.034 T	.000 T	0.034*	-	0.37
	t, Z	-2.330 T	-1.886*	-	-	-594	-	-2.628*	-3.621*	8.460*	-1.826 T	-6.510 T	-1.826*	-	-.338
8. Rename Method	Sig.	0.048 T	0.393	-	-	0.098	-	0.026*	0.098	0.2965	0.3575	0.1205	0.3575	-	0.418
	t, Z	-1.664 T	-.271	-	-	-1.293	-	-1.947*	-1.293	-.535	-.365	-1.172	-.365	-	-.207

110

ตารางที่ 4-13 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์กับกระบวนการรีแฟคทอริงวิธีอื่นๆ

6. กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาสกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.0005 T	0.3675	0.5	0.5	0.0005*	0.5	.000*	.000*	.000*	0.3275	.000 T	0.3275	0.1585	0.0005*
	t, Z	-3.285 T	-.338	.000	.000	-3.297*	.000	-3.912*	-4.107*	10.834*	-.447	-9.658 T	-.447	-1.000	-3.296*
2. Replace Temp with Query	Sig.	0.024 T	-	-	-	0.1585	-	0.485	0.0105*	0.0005*	-	0.002 T	-	-	0.1585
	t, Z	-2.399 T	-	-	-	-1.000	-	.039	-2.961*	5.118*	-	-4.250 T	-	-	-1.000
3. Introduce Explaining Variable	Sig.	0.055	0.3275	0.1585	0.1585	0.1585	0.1585	0.09	0.0025*	-	0.09	0.1585	0.09	0.1585	0.1585
	t, Z	-1.599	-.447	-1.000	-1.000	-1.000	-1.000	-1.342	-2.805*	-	-1.342	-1.000	-1.342	-1.000	-1.000
4. Move Method	Sig.	.000 T	0.3575	-	-	.000*	-	.000*	0.0025*	0.3275	0.1425	0.0105 T	0.1425	-	.000*
	t, Z	-3.920 T	-365	-	-	-4.107*	-	-3.528*	-2.841*	-.447	-1.069	-2.486 T	-1.069	-	-4.107*
5. Move Field	Sig.	.000 T	0.0545	-	-	.000*	-	0.001*	0.012*	0.3275	0.1425	0.021 T	0.2965	-	.000*
	t, Z	-3.724 T	-1.604	-	-	-3.921*	-	-3.114*	-2.259*	-.447	-1.069	-2.176 T	-.535	-	-3.920*
6. Replace Magic Number with Symbolic	Sig.	0.0005 T	-	-	-	0.0215*	-	0.1585	.000 T	-	0.009 T	-	0.001*	-	0.0215*
	t, Z	-3.180 T	-	-	-	-2.023*	-	-1.000	5.408 T	-	-2.366 T	-	-3.059*	-	-2.023*
7. Decompose Conditional	Sig.	0.0005 T	0.13	-	-	0.4375	-	0.0005*	.000*	.000*	0.0545	.000 T	0.1425	-	0.255
	t, Z	-3.258 T	-1.125	-	-	-.157	-	-3.332*	-3.724*	7.922*	-1.604	-7.114 T	-1.609	-	-.659
8. Rename Method	Sig.	-	0.5	0.1585	0.1585	0.1655	0.1585	0.3275	0.5	-	0.09	0.1585	0.0545	-	0.365
	t, Z	-	.000	-1.000	-1.000	-.973	-1.000	-.447	.000	-	-1.342	-1.000	-1.604	-	-.345

ตารางที่ 4-14 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาสกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

7. กระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์กับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.0155 T	2.5	-	-	0.0005*	-	0.0005*	.000*	.000*	0.3575	.000 T	0.1725	0.1585	0.0005*
	t, Z	-2.154 T	-.674	-	-	-3.296*	-	-3.463*	-3.920*	7.344*	-.365	-6.798 T	-.944	-1.000	-3.297*
2. Replace Temp with Query	Sig.	0.051	-	-	-	0.1585	-	0.433	0.0235*	0.0005*	-	0.0015 T	-	-	0.1585
	t, Z	-1.932	-	-	-	-1.000	-	-.176	-2.489*	6.766*	-	-4.657 T	-	-	-1.000
3. Introduce Explaining Variable	Sig.	0.2575	0.2325	0.1585	0.1585	0.09	0.1585	0.3765	0.143	-	0.1425	0.1585	0.1125	-	0.09
	t, Z	-.652	-.730	-1.000	-1.000	-1.342	-1.000	-.314	-1.067	-	-1.069	-1.000	-1.214	-	-1.342
4. Move Method	Sig.	0.0045 T	.000*	.000*	.000*	0.3785	0.0005*	0.002*	0.001*	0.1585	0.1425	0.0535	0.1425	-	.000*
	t, Z	-2.975 T	-3.621*	-	-	-.315	-3.940*	-2.912*	-3.053*	-1.000	-1.069	-1.705	-1.069	-	-3.622*
5. Move Field	Sig.	0.009 T	.000*	.000*	.000*	0.0005*	0.0005*	0.027 T	0.002*	0.1585	0.1425	0.0885	0.5	-	.000*
	t, Z	-2.643 T	-3.516*	-	-	-4.315*	-4.057*	2.091 T	-2.844*	-1.000	-1.069	-1.418	.000	-	-3.516*
6. Extract Class	Sig.	0.0005 T	.000*	.000*	.000*	.000*	0.0005*	0.0015*	.000*	-	0.09	-	0.09	-	.000*
	t, Z	-3.408 T	-3.621*	-	-	-3.622*	-3.180*	-3.006*	-3.621*	-	-1.342	-	-1.342	-	-3.622*
7. Decompose Conditional	Sig.	0.0035 T	0.069	-	-	0.472	0.1585	0.004*	.000*	.000*	0.3575	.000 T	0.25	0.1585	0.4485
	t, Z	-2.689 T	-1.483	-	-	-.071	-1.000	-2.651*	-3.724*	6.946*	-.365	-4.891 T	-.674	-1.000	-.129
8. Rename Method	Sig.	0.3675	0.2325	-	-	0.074	-	0.3575	0.2205	-	0.343	-	0.242	-	0.2505
	t, Z	-.338	-.730	-	-	-1.448	-	-.365	-.770	-	-.405	-	-.700	-	-.672

ตารางที่ 4-15 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์กับกระบวนการรีแฟคทอริงวิธีอื่นๆ

112

8. กระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	0.002 T	0.0805	-	-	0.0025*	-	.000*	.000*	.000*	0.3275	.000 T	0.1425	0.1585	0.001*
	t, Z	-2.897 T	-1.400	-	-	-2.795*	-	-3.621*	-4.015*	6.684*	-.447	-6.403 T	-1.069	-1.000	-3.108*
2. Replace Temp with Query	Sig.	0.021 T	-	-	-	0.1165	-	0.3915	0.009*	0.0015*	0.1585	0.001 T	0.09	-	0.356
	t, Z	-2.578 T	-	-	-	-1.325	-	.288	-2.366*	4.751*	-1.000	-5.087 T	-1.342	-	-.387
3. Introduce Explaining Variable	Sig.	0.094	0.3205	-	-	0.3575	-	0.2325	0.396	0.3905	0.1585	0.5	0.1585	-	0.2325
	t, Z	-1.412	.481	-	-	-.365	-	-.730	.270	-.286	-1.000	.000	-1.000	-	-.730
4. Move Method	Sig.	0.0005 T	.000*	.000*	.000*	0.0385*	0.0005*	0.014*	0.0025*	0.002 T	0.09	0.447	0.0545	-	.000*
	t, Z	-3.741 T	-3.823*	-6.969*	-6.326*	-1.771*	-3.408*	-2.200*	-3.243*	-3.263 T	-1.342	-.135	-1.604	-	-3.823*
5. Move Field	Sig.	0.002 T	.000*	.000*	.000*	.000*	0.0005*	0.2695	0.015*	0.014 T	0.09	0.3885	0.0545	-	.000*
	t, Z	-3.401 T	-5.743*	-8.408*	-7.338*	-3.527*	-4.139*	.628	-2.377*	-2.421 T	-1.342	.288	-1.604	-	-3.621*
6. Extract Class	Sig.	.000 T	.000*	.000*	.000*	.000*	0.0005*	0.0005*	.000*	-	0.0545	-	0.072	-	.000*
	t, Z	-4.354 T	-3.823*	-6.969*	-6.326*	-3.823*	-3.408*	-3.462*	-	-	-1.604	-	-1.461	-	-3.823*
7. Replace Magic Number with Symbolic Constant	Sig.	0.07	0.072	-	-	0.4975	0.1585	0.4585	0.062	3	0.033 T	0.2315	0.008*	0.1585	0.3605
	t, Z	-1.476	-1.461	-	-	.057	-1.000	-.105	-1.538	-.524	-1.836 T	-.734	-2.417*	-1.000	-.357
8. Rename Method	Sig.	0.2875	0.1315	-	-	0.0445*	-	0.4765	0.4375	0.4065	0.2965	0.4765	0.3575	-	0.198
	t, Z	-.561	-1.120	-	-	-1.699*	-	-.059	-.157	-.237	-.535	-.059	-.365	-	-.849

113

ตารางที่ 4-16 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

9. กระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

		Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
1. Extract Method	Sig.	.000 T	0.3765	-	-	0.001*	-	.000*	.000*	.000*	0.3275	.000 T	0.3275	-	0.002*
	t, Z	-3.920 T	-.314	-	-	-3.101*	-	-3.632*	-4.015*	8.706*	-.447	-8.349 T	-.447	-	-2.865*
2. Replace Temp with Query	Sig.	0.018 T	-	0.1585	0.1585	0.3275	0.1585	0.025 T	0.0025*	.000*	0.1585	0.001 T	0.09	-	0.3275
	t, Z	-2.393 T	-	-1.000	-1.000	-.447	-1.000	2.199 T	-3.482*	8.214*	-1.000	-3.059 T	-1.342	-	-.447
3. Introduce Explaining Variable	Sig.	0.0025 T	0.09	-	-	-	-	0.2965	0.003 T	0.5	-	-	0.1425	-	-
	t, Z	-2.803 T	-1.342	-	-	-	-	-.535	3.433 T	.000	-	-	-1.069	-	-
4. Move Method	Sig.	0.0005 T	0.0005*	.000*	.000*	0.2535	.000*	.000*	0.002*	0.3275	0.1425	0.002 T	0.0545	0.1585	.000*
	t, Z	-3.893 T	-3.393*	-7.659*	-3.729*	-.676	-3.724*	-3.620*	-2.906*	-.447	-1.069	-2.857 T	-1.604	-1.000	-4.107*
5. Move Field	Sig.	0.0005 T	.000*	.000*	.000*	.000*	.000*	0.002*	0.0005*	0.5	0.3275	0.016 T	0.3275	-	.000*
	t, Z	-3.850 T	-4.015*	-8.207*	-7.339*	-4.993*	-4.741*	-2.885*	-3.354*	.000	-.447	-2.303 T	-.447	-	-4.015*
6. Extract Class	Sig.	.000 T	.000*	.000*	.000*	.000*	.000*	.000*	.000*	-	0.09	0.1585	0.0545	-	.000*
	t, Z	-4.916 T	-4.107*	-3.736*	-3.727*	-4.107*	-3.724*	-3.165*	-4.107*	-	-1.342	-1.000	-1.604	-	-4.107*
7. Replace Magic Number with Symbolic Constant	Sig.	0.017 T	0.1585	-	-	0.1185	-	0.4195	0.0005*	-	0.0235 T	-	0.0035*	-	0.1185
	t, Z	-2.120 T	-1.000	-	-	-1.183	-	-.135	-3.393*	-	-1.988 T	-	-2.689*	-	-1.183
8. Decompose Conditional	Sig.	0.0035 T	0.0175*	-	-	0.192	-	.000*	.000*	.000*	0.2965	.000 T	0.5	-	0.204
	t, Z	-2.688 T	-2.106*	-	-	-.871	-	-3.734*	-4.107*	7.913*	-.535	-6.931 T	.000	-	-.828

ตารางที่ 4-17 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

จากตารางที่ 4-9 ถึงตารางที่ 4-17 เป็นตารางที่แสดงค่า Sig. (1-tailed) และค่าสถิติ t หรือ Z ของมาตรวัดเชิงวัตถุ เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของสมมติฐานที่ 2 จะเห็นได้ว่า กระบวนการรีแฟคทอริงในสมมติฐานที่ 2 นี้โดยรวมทำให้ค่ามาตรวัดเชิงวัตถุทุกตัวดีขึ้น ซึ่งในกระบวนการรีแฟคทอริงในแต่ละวิธีจะมีส่วนทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นแตกต่างกัน ดังที่อธิบายในสมมติฐานที่ 1 ว่าการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ กระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส และกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชัน มีผลทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นอย่างมีนัยสำคัญและสอดคล้องกัน เมื่อมองผลการทดสอบในสมมติฐานที่ 1 และสมมติฐานที่ 2 ประกอบกัน ผู้วิจัยสังเกตว่ากระบวนการรีแฟคทอริงบางวิธีจะทำให้ค่ามาตรวัดเชิงวัตถุบางชนิดมีค่าดีขึ้นที่สอดคล้องกันทั้ง 10 ตาราง (ดูตารางที่ 4-8 ถึงตารางที่ 4-17 ประกอบ) โดยสามารถวิเคราะห์ได้ดังนี้

- กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดทำให้มาตรวัดแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) และมาตรวัดเมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) มีค่าดีขึ้น เนื่องจากผู้วิจัยสังเกตว่าการดีขึ้นของค่ามาตรวัดทั้ง 3 ชนิดดังกล่าว จะมีผลจากกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดไม่ว่าจะในสมมติฐานที่ 1 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดเพียงวิธีเดียว หรือในสมมติฐานที่ 2 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับวิธีอื่นๆ ก็จะทำให้ผลของค่ามาตรวัดทั้ง 3 ชนิดนี้ มีค่าที่ดีขึ้น โดยผู้วิจัยเห็นว่ามีเหตุผลมาจากกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดจะมีผลทำให้คลาสที่มีความเป็นเอกภาพมากขึ้น นั่นหมายถึงทำให้คลาสมีหน้าที่การทำงานที่ชัดเจนขึ้น ส่งเสริมการทำงานรูปแบบเชิงวัตถุเนื่องจากซอร์สโค้ดจากหน่วยทดลองยังมีการพัฒนาในรูปแบบภาษาเชิงกระบวนคำสั่ง (Procedural language) อยู่ และยังสนับสนุนคุณสมบัติการห่อหุ้ม (Encapsulation) กับข้อมูลภายในระบบอีกด้วย

- กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีทำให้มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) และมาตรวัดเมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) มีค่าดีขึ้น เนื่องจากผู้วิจัยสังเกตว่าการดีขึ้นของค่ามาตรวัดทั้ง 2 ชนิดดังกล่าว จะมีผลจากกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีไม่ว่าจะในสมมติฐานที่ 1 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีเพียงวิธีเดียว หรือในสมมติฐานที่ 2 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีกับวิธีอื่นๆ ก็จะทำให้ผลของค่ามาตรวัดทั้ง 2 ชนิดนี้ มีค่าที่ดีขึ้น โดยผู้วิจัยเห็นว่ามีเหตุผลมาจากกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีจะมีผล

ทำให้เมท็อดในแต่ละคลาสมีซอร์สโค้ดที่กระชับมากขึ้นและทำให้ระบบมีคุณสมบัติการห่อหุ้ม (Encapsulation) ที่ดีขึ้นด้วย

- กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ และกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาสทำให้มาตรวัดเวทเทดเมท็อดเปอคลาส (Weighted Methods per Class: WMC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดเดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree: DIT) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดนัมเบอร์ออฟชิลเดรน (Number of Children: NOC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดคัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) มี 2 วิธีที่มีค่าดีขึ้น โดยยกเว้นกระบวนการรีแฟคทอริงวิธีมูฟเมท็อด มาตรวัดเรสปอนซ์ฟอร์อะคลาส (Response for a Class: RFC) ทั้ง 3 วิธีมีค่าดีขึ้น มาตรวัดแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) มีเพียง 2 วิธีที่มีค่าดีขึ้น โดยยกเว้นกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) ทั้ง 3 วิธีมีค่าดีขึ้น และมาตรวัดคัพพลิงแฟคเทอร์ (Coupling Factor: COF) ทั้ง 3 วิธีมีค่าดีขึ้น เนื่องจากผู้วิจัยสังเกตว่าการดีขึ้นของค่ามาตรวัดทั้ง 8 ชนิดดังกล่าวมีเหตุผลมาจากหลักการในการทำกระบวนการรีแฟคทอริงทั้ง 3 วิธีนี้ จำเป็นต้องมีการเพิ่มคลาสขึ้นในซอร์สโค้ด ดังนั้นค่ามาตรวัดเชิงวัตถุที่ดีขึ้นก็จะมีส่วนที่สอดคล้องกัน โดยผู้วิจัยเห็นว่าการทำกระบวนการรีแฟคทอริงทั้ง 3 วิธี ซึ่งอยู่ในกระบวนการรีแฟคทอริงประเภทเดียวกันมีผลทำให้คุณสมบัติการสืบทอด (Inheritance) ของคลาสดีขึ้น และทำให้เมท็อดและแอตทริบิวต์มีหน้าที่การทำงานที่ตรงกับคลาสมากขึ้น ซึ่งจะเป็นผลทำให้การเชื่อมต่อ (Coupling) ระหว่างคลาสมีผลที่ดีขึ้นด้วย

- กระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลทำให้มาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) มีค่าดีขึ้น เนื่องจากผู้วิจัยสังเกตว่าการดีขึ้นของมาตรวัดทั้ง 2 ชนิดดังกล่าว จะมีผลจากกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลไม่ว่าจะในสมมติฐานที่ 1 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลเพียงวิธีเดียว หรือในสมมติฐานที่ 2 ซึ่งเป็นการทำกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลกับวิธีอื่นๆ ก็จะทำให้ผลของค่ามาตรวัดทั้ง 2 ชนิดนี้ มีค่าที่ดีขึ้น โดยผู้วิจัยเห็นว่ามีเหตุผลมาจากกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันจะมีผลทำให้ซอร์สโค้ดในแต่ละเมท็อดมีความกระชับมากยิ่งขึ้น

นอกจากการเพิ่มขึ้นของค่ามาตรวัดเชิงวัตถุที่สอดคล้องกันแล้วผู้วิจัยสังเกตเห็นอีกมุมมองหนึ่งว่า

- เมื่อเกิดการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดพบว่ากระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดมีส่วนทำให้ค่ามาตรวัดคัพพลิงบี

ทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) ดีขึ้น โดยมาตรวัดเชิงวัตถุนี้จะไม่ดีขึ้นเมื่อทำกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดกับกระบวนการรีแฟคทอริงวิธีอื่นๆ

- เมื่อเกิดการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์พบว่ากระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดมีส่วนทำให้ค่ามาตรแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) ดีขึ้น โดยมาตรวัดเชิงวัตถุนี้จะไม่ดีขึ้นเมื่อทำกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์กับกระบวนการรีแฟคทอริงวิธีอื่นๆ

จะเห็นได้ว่าการทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดเพิ่มเข้าไป มีส่วนช่วยเพิ่มให้ซอร์สโค้ดมีการเชื่อมต่อ (Coupling) ระหว่างคลาสมีผลที่ดีขึ้น คุณสมบัติการทำงานร่วมกัน (Cohesion) กับคลาสอื่นๆ น้อยลง นั่นหมายความว่าแต่ละคลาสมีความชัดเจนในการทำงานที่ดีขึ้นและสามารถทำงานได้เองโดยไม่จำเป็นต้องได้รับการช่วยเหลือจากเมท็อดของคลาสอื่นๆ

สำหรับกระบวนการรีแฟคทอริงวิธีที่เหลือ ผู้วิจัยไม่สามารถกล่าวสรุปการเพิ่มขึ้นของค่ามาตรวัดเชิงวัตถุต่างๆ ได้ เนื่องจากผลการทดลองไม่ได้แสดงให้เห็นการเพิ่มขึ้นของค่ามาตรวัดเชิงวัตถุในทางที่สอดคล้องกันใน 2 สมมติฐานนี้

4.5 การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x

ผู้วิจัยต้องการเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y เปรียบเทียบกับการทำกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x ซึ่งการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุนั้นแบ่งเป็น 2 กรณีคือ 1) มาตรวัดเชิงวัตถุที่มีการแจกแจงแบบปกติ 2) มาตรวัดเชิงวัตถุที่ไม่ได้มีการแจกแจงแบบปกติ สำหรับกรณีที่ 1 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับผลต่างระหว่างค่าเฉลี่ย 2 ประชากรแบบจับคู่ (Paired t-test) และสำหรับกรณีที่ 2 ผู้วิจัยเลือกการทดสอบสมมติฐานเกี่ยวกับการแจกแจงของข้อมูล 2 ชุดที่สัมพันธ์กัน (Two-Related-Samples Tests) โดยทั้ง 2 กรณีจะใช้โปรแกรม SPSS ในการทดสอบ

การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y เปรียบเทียบกับการทำกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x มีหน่วยทดลองที่ผ่านกระบวนการรีแฟคทอริงวิธีวิธีเอ็กแทรกเมท็อดสลับลำดับกับกระบวนการรีแฟคทอริงวิธีอื่นๆ โดยมีเพียงมาตรวัดเอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) ที่มีค่าสำหรับการทดสอบสมมติฐานได้ ส่วนมาตรวัดอื่นๆ ที่เหลือไม่สามารถทดสอบสมมติฐานได้เนื่องจากผลต่างของค่ามาตรวัดเชิงวัตถุมีค่าไม่แตกต่างกัน ซึ่งมีการทดสอบสมมติฐานไว้ดังนี้

$$H_0: \mu_2 = \mu_1$$

$$H_1: \mu_2 \neq \mu_1$$

โดย μ_1 หมายถึงค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x และวิธีที่ y และ μ_2 หมายถึงค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ y และวิธีที่ x เนื่องจากผู้วิจัยได้คาดหวังว่าซอร์สโค้ดที่ผ่านการทำกระบวนการรีแฟคทอริง 2 วิธี ที่เหมือนกันแต่ทำ 2 วิธีนั้นในลำดับที่ต่างกันอาจจะมีคุณภาพซอฟต์แวร์ที่แตกต่างกันด้วย โดยการทดสอบค่ามาตรวัดเชิงวัตถุจะแสดงไว้ในตารางที่ 4-18

ซึ่งผลการตรวจสอบการแจกแจงปกติของค่ามาตรวัดเชิงวัตถุเป็นแบบปกติ (ดูตารางที่ 4-7 ประกอบ) สำหรับสมมติฐานกรณีที่ 3 จะปฏิเสธสมมติฐานเมื่อค่า Sig. (2 - tailed) < ค่าระดับนัยสำคัญ จึงจะสามารถปฏิเสธ H_0 (กัลยา วานิชย์บัญชา, 2548)

เนื่องจากการทำกระบวนการรีแฟคทอริงตามสมมติฐานข้อที่ 3 สามารถวิเคราะห์ข้อมูลทางสถิติได้ทั้งหมด 8 รูปแบบ ผู้วิจัยขอไม่แสดงตารางข้อมูลของการทำกระบวนการรีแฟคทอริงกับหน่วยทดลองที่ไม่สามารถวิเคราะห์ข้อมูลทางสถิตินี้ได้

ซึ่งจากตารางที่แสดงไว้ในภาคผนวก แสดงให้เห็นว่าค่ามาตรวัดเชิงวัตถุทั้ง 14 ค่า สามารถยอมรับ H_0 และปฏิเสธ H_0 โดยสามารถสรุปได้ดังตารางต่อไปนี้

Refactoring		Average Method Size
1. Extract Method vs. Replace Temp with Query	Sig.	0.926
	t, Z	-.095
2. Extract Method vs. Introduce Explaining Variable	Sig.	0.661
	t, Z	0.450
3. Extract Method vs. Move Method	Sig.	0.337
	t, Z	0.982
4. Extract Method vs. Move Field	Sig.	0.410
	t, Z	0.842
5. Extract Method vs. Extract Class	Sig.	0.217
	t, Z	-1.234
6. Extract Method vs. Replace Magic Number with Symbolic Constant	Sig.	0.747
	t, Z	-.328
7. Extract Method vs. Decompose Conditional	Sig.	0.423
	t, Z	.817
8. Extract Method vs. Rename Method	Sig.	0.179
	t, Z	1.381

ตารางที่ 4-18 แสดงตารางค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของสมมติฐานที่ 3

จากตารางที่ 4-18 เป็นตารางที่แสดงค่า Sig. (1-tailed) และค่าสถิติ t หรือ Z ของมาตรวัดเชิงวัตถุ เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของกระบวนการรีแฟคทอริงที่ทำในลำดับที่ต่างกันตามสมมติฐานที่ 3 ซึ่งเป็นการทดสอบสมมติฐานลำดับของกระบวนการรีแฟคทอริง 8 รูปแบบ ที่ผ่านการทดสอบการแจกแจงข้อมูลแล้ว จะเห็นได้ว่าไม่สามารถปฏิเสธสมมติฐานได้เนื่องจากกระบวนการรีแฟคทอริงในกลุ่มนี้ไม่ได้ทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นอย่างมีนัยสำคัญ

4.6 การวิเคราะห์ข้อมูลเพิ่มเติม (Exploration)

จากผลสรุปดังตารางที่ 4-8 และตารางที่ 4-9 ถึงตารางที่ 4-17 นั้นแสดงให้เห็นว่าค่ามาตรวัดเชิงวัตถุบางค่าเท่านั้นที่ดีขึ้นจากกระบวนการรีแฟคทอริงในแต่ละวิธี ในทางกลับกันอาจมีกระบวนการรีแฟคทอริงบางวิธีหรือกรณีเมื่อนำกระบวนการรีแฟคทอริง 2 วิธี มาใช้ร่วมกันแล้วทำให้ค่ามาตรวัดเชิงวัตถุบางค่าด้อยลงด้วย ดังนั้นผู้วิจัยจึงตั้งสมมติฐานขึ้นใหม่ 2 กรณี (สำหรับกรณีการสลับลำดับวิธีการทำกระบวนการรีแฟคทอริงที่ไม่เกิดความแตกต่างของค่ามาตรวัดเชิงวัตถุ ในผลการทดลองของสมมติฐานที่ 3 ผู้วิจัยจึงไม่นำประเด็นเรื่องการสลับลำดับวิธีการทำกระบวนการรีแฟคทอริงมาวิเคราะห์ข้อมูลเพิ่มเติม) ดังนี้

4. H_0: ค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ไม่ได้มีค่ามาตรวัดเชิงวัตถุน้อยลง

   H_1: ค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง มีค่ามาตรวัดเชิงวัตถุน้อยลง

5. H_0: ค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับหลังผ่านกระบวนการรีแฟคทอริง 2 วิธี ไม่ได้มีค่ามาตรวัดเชิงวัตถุน้อยลง

   H_1: ค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากซอร์สโค้ดเดียวกันที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี หลังผ่านกระบวนการรีแฟคทอริง 2 วิธี มีค่ามาตรวัดเชิงวัตถุน้อยลง

ซึ่งสามารถเขียนเป็นรูปแบบมาตรฐานของสมมติฐานดังกล่าว โดยแยกการทดสอบสมมติฐานเป็น 2 กรณี ดังนี้

1. กรณีค่ามาตรวัดเชิงวัตถุที่มีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ด้อยลง ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \geq \mu_1$$
$$H_1: \mu_2 < \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริง (สำหรับสมมติฐานที่ 5 μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี)

μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง (สำหรับสมมติฐานที่ 5 μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 2 วิธี)

สำหรับมาตรวัดเชิงวัตถุที่มีค่าลดลงแล้วส่งผลให้คุณภาพซอฟต์แวร์ด้อยลงประกอบไปด้วย 1) คอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) 2) เมท็อดไฮดิงแฟคเทอร์ (Method Hiding Factor: MHF) 3) แอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) 4) เมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) และ 5) โพลีมอร์ฟิซึมแฟคเทอร์ (Polymorphism Factor: POF) (Quenel และ Lövdahl, 2004)

2. กรณีค่ามาตรวัดเชิงวัตถุที่มีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ด้อยลง ดังนั้นสามารถตั้งสมมติฐานได้ดังนี้

$$H_0: \mu_2 \leq \mu_1$$
$$H_1: \mu_2 > \mu_1$$

โดยที่ μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริง (สำหรับสมมติฐานที่ 5 μ_1 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 1 วิธี)

μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง (สำหรับสมมติฐานที่ 5 μ_2 คือค่าเฉลี่ยของค่ามาตรวัดเชิงวัตถุเมื่อผ่านกระบวนการรีแฟคทอริง 2 วิธี)

สำหรับมาตรวัดเชิงวัตถุที่มีค่าเพิ่มขึ้นแล้วส่งผลให้คุณภาพซอฟต์แวร์ด้อยลงประกอบไปด้วย 1) เวทเทดเมท็อดเปอคลาส (Weighted Methods per Class: WMC) 2) เดพออฟอินเฮอริแทนทรี (Depth of Inheritance Tree: DIT) 3) นัมเบอร์ออฟชิลเดรน (Number of Children: NOC) 4) คัพพลิงบีทวีนอ็อบเจกต์คลาส (Coupling Between Object Classes: CBO) 5) เรสปอนซ์ฟอร์อะคลาส (Response for a Class: RFC) 6) แลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) 7) เอเวอร์เรจเมท็อดไซซ์ (Average Method Size: AMS) 8) แอตทริบิวต์อินเฮอริแทนแฟค

เทอร์ (Attribute Inheritance Factor: AIF) และ 9) คัพพลิงแฟคเทอร์ (Coupling Factor: COF) (Quenel และ Lövdahl, 2004)

จากตารางที่ 4-8 และตารางที่ 4-9 ถึงตารางที่ 4-17 แสดงค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z เพื่อใช้บอกยอมรับและปฏิเสธ H_0 ของแต่ละสมมติฐาน โดยผลการทดสอบสมมติฐานชุดใหม่ที่มีค่าสถิติ Sig. (1-tailed) และค่าสถิติ t หรือ Z ซึ่งจะปฏิเสธสมมติฐาน H_0 เมื่อ Sig. (1-tailed) < ค่าระดับนัยสำคัญ (ผู้วิจัยกำหนดไว้ที่ 0.05) โดยผู้วิจัยแสดงในตารางด้วยเครื่องหมายที (T)

สำหรับสมมติฐานที่ 4 การเปรียบเทียบค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริง เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง จากตารางที่ 4-8 จะเห็นได้ว่าค่ามาตรวัดเชิงวัตถุที่มีค่าด้อยลงคือ 1) คอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) 2) แลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) 3) แอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) และ 4) เมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) โดยเป็นผลมาจากกระบวนการรีแฟคทอริงเกือบทุกวิธี สามารถวิเคราะห์ได้ดังนี้

- กระบวนการรีแฟคทอริงเกือบทุกวิธียกเว้นกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดทำให้มาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) มีค่าด้อยลง ผู้วิจัยพบว่าการทำกระบวนการรีแฟคทอริงเกือบทุกวิธีจะทำให้เกิดจำนวนบรรทัดในซอร์สโค้ดเพิ่มขึ้น แต่กระบวนการรีแฟคทอริงที่ผ่านการทำจากเครื่องมือรีชาร์ปเปอร์ 2.0 (Resharper 2.0) ไม่ได้มีส่วนเพิ่มคำอธิบาย (Comment) ให้กับซอร์สโค้ด แต่กระบวนการรีแฟคทอริงในแต่ละวิธีส่วนมากจะเป็นการเพิ่มบรรทัดของซอร์สโค้ดขึ้น จึงทำให้ค่ามาตรวัดคอมเมนต์เปอร์เซ็นต์เทจมีค่าลดลงดังกล่าว
- กระบวนการรีแฟคทอริงที่มีการยุ่งเกี่ยวกับการเพิ่มหรือลดจำนวนฟิลด์ในระบบ โดยเฉพาะอย่างยิ่งกระบวนการรีแฟคทอริงวิธีมูฟฟิลด์และกระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ทำให้มาตรวัดแลคออฟโคฮีชันอินเมท็อด (Lack of Cohesion in Method: LCOM) และมาตรวัดแอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) มีค่าด้อยลง (ซึ่งจะพบได้อีกในสมมติฐานที่ 5 ผู้วิจัยจะกล่าวต่อไป) ผู้วิจัยพบว่าการทำกระบวนการรีแฟคทอริง 2 วิธีนี้ จะทำให้ความสามารถการเอ็นแคบซูเลชัน (Encapsulation) ของระบบลดลง เนื่องจากกระบวนการรีแฟคทอริง 2 วิธีนี้จะไปเพิ่มจำนวนฟิลด์เฉพาะที่ ซึ่งฟิลด์เหล่านี้จะถูกใช้เฉพาะในแต่ละคลาสเท่านั้น จากประเด็นนี้จึงเป็นผลให้อัตราส่วนของฟิลด์โดยรวมเทียบกับฟิลด์ที่มีคุณสมบัติการเอ็นแคบซูเลชันลดลง
- กระบวนการรีแฟคทอริงที่มีการยุ่งเกี่ยวกับการเพิ่มหรือลดจำนวนเมท็อดในระบบ โดยเฉพาะอย่างยิ่งกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด และกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีทำให้มาตรวัดเมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF)

มีค่าด้อยลง (ซึ่งจะพบได้อีกในสมมติฐานที่ 5 ผู้วิจัยจะกล่าวต่อไป) ผู้วิจัยพบว่าการทำกระบวนการรีแฟคทอริง 2 วิธีนี้ จะทำให้ความสามารถการอินเฮอริแทน (Inheritance) ของระบบลดลง เนื่องจากกระบวนการรีแฟคทอริง 2 วิธีนี้จะทำให้ระบบมีจำนวนเมท็อดที่ใช้งานในแต่ละคลาสเพิ่มมากขึ้น จึงเป็นผลให้อัตราส่วนของเมท็อดโดยรวมเทียบกับเมท็อดที่มีคุณสมบัติการอินเฮอริแทนลดลง

สำหรับสมมติฐานที่ 5 การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงริง 1 วิธี เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริงริง 2 วิธี จากตารางที่แสดงค่าสถิติสำหรับสมมติฐานที่ 5 แสดงให้เห็นว่าค่ามาตรวัดเชิงวัตถุที่ด้อยลงสามารถพบได้หลายค่ามาตรวัดเชิงวัตถุแตกต่างกันไปใน 9 ตารางข้างต้น แต่เมื่อมองไปที่กระบวนการรีแฟคทอริงวิธีวิธีเอ็กแทรกเมท็อด กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี กระบวนการรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธ ซิมโบลิกคอนสแตนท์ และกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอล จะพบว่าในตาราง 9 ตารางนี้จะพบค่ามาตรวัดเชิงวัตถุที่ด้อยลงเหมือนกันอย่างมีนัยสำคัญและสอดคล้องกับสมมติฐานที่ 4 คือ

- กระบวนการรีแฟคทอริงเกือบทุกวิธีใน 9 ตาราง (ดูตารางที่ 4-9 ถึงตารางที่ 4-17 ประกอบ) ทำให้มาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ (Comment Percentage) มีค่าด้อยลง ผู้วิจัยพบว่าการทำกระบวนการรีแฟคทอริงเกือบทุกวิธีจะทำให้เกิดจำนวนบรรทัดในซอร์สโค้ดเพิ่มขึ้น แต่กระบวนการรีแฟคทอริงที่ผ่านการทำจากเครื่องมือรีชาร์ปเปอร์ 2.0 (Resharper 2.0) ไม่ได้มีส่วนเพิ่มคำอธิบาย (Comment) ให้กับซอร์สโค้ด แต่กระบวนการรีแฟคทอริงในแต่ละวิธีส่วนมากจะเป็นการเพิ่มบรรทัดของซอร์สโค้ดขึ้น จึงทำให้ค่ามาตรวัดคอมเมนต์เปอร์เซ็นต์เทจมีค่าลดลงดังกล่าว
- การทำกระบวนรีแฟคทอริงวิธีเอ็กแทรกเมท็อด กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี และกระบวนการรีแฟคทอริงวิธีดีคอมโพสคอนดิชันนอลทำให้มาตรวัดเมท็อดอินเฮอริแทนแฟคเทอร์ (Method Inheritance Factor: MIF) มีค่าด้อยลง (ดูตารางที่ 4-9 ถึงตารางที่ 4-17 ประกอบ) โดยผู้วิจัยพบว่าความสามารถการอินเฮอริแทน (Inheritance) ของระบบลดลง เนื่องจากกระบวนการรีแฟคทอริง 2 วิธีนี้จะทำให้ระบบมีจำนวนเมท็อดที่ใช้งานในแต่ละคลาสเพิ่มมากขึ้น จึงเป็นผลให้อัตราส่วนของเมท็อดโดยรวมเทียบกับเมท็อดที่มีคุณสมบัติการอินเฮอริแทนลดลง เช่นเดียวกับเหตุผลข้างต้น
- กระบวนรีแฟคทอริงวิธีรีเพลสเมจิกนัมเบอร์วิธซิมโบลิกคอนสแตนท์ทำให้มาตรวัดแอตทริบิวต์ไฮดิงแฟคเทอร์ (Attribute Hiding Factor: AHF) มีค่าด้อยลง (ดูตารางที่ 4-9 ถึงตารางที่ 4-17 ประกอบ) โดยผู้วิจัยพบว่าการทำความสามารถการเอ็นแคบซูเลชัน (Encapsulation) ของระบบลดลง เนื่องจากกระบวนการรีแฟคทอริง 2 วิธีนี้จะไปเพิ่มจำนวนฟิลด์เฉพาะที่ ซึ่งฟิลด์เหล่านี้

จะถูกใช้เฉพาะในแต่ละคลาสเท่านั้น จากประเด็นนี้จึงเป็นผลให้อัตราส่วนของฟิลด์โดยรวมเทียบกับฟิลด์ที่มีคุณสมบัติการเอ็นแคบซูเลชันลดลงเช่นเดียวกับเหตุผลข้างต้น

สำหรับการเปลี่ยนแปลงของมาตรวัดเชิงวัตถุที่เหลือ ไม่ว่าจะเป็นการดีขึ้นหรือด้อยลง ผู้วิจัยไม่สามารถกล่าวสรุปได้เนื่องจากการเปลี่ยนแปลงของมาตรวัดเชิงวัตถุดังกล่าวไม่เกิดขึ้นอย่างมีนัยสำคัญ โดยผลการทดลองอาจมีผลที่แตกต่างกันได้เมื่อผู้ที่สนใจนำกลุ่มหน่วยทดลองนอกเหนือข้อจำกัดของงานวิจัยนี้

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 บทนำ

บทนี้นำเสนอสรุปผลการวิเคราะห์เพื่อตอบวัตถุประสงค์ของงานวิจัย การอภิปรายประเด็นต่างๆ ที่เกิดขึ้นในงานวิจัย การนำงานวิจัยนี้ไปใช้ประโยชน์ในเชิงทฤษฎีและเชิงประยุกต์ ข้อจำกัดของงานวิจัยและข้อเสนอแนะเพื่อเป็นโอกาสในการศึกษาต่อไปในภายภาคหน้า

5.2 การทดลองและลักษณะของหน่วยทดลอง

งานวิจัยนี้เป็นการวิจัยเชิงทดลอง (Experimental Research) โดยใช้หน่วยทดลองเป็นโครงงานที่จัดทำโดยนิสิตปริญญาบัณฑิต ในวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ซึ่งเป็นวิชาแรกด้านการเชิงวัตถุ (Object – Oriented) ในปีการศึกษา 2547 และ 2548 รวมทั้งหมด 32 หน่วยทดลอง โดยมีข้อกำหนดว่าแต่ละโครงงานจะต้องมีจำนวนคลาสอย่างน้อย 5 คลาส แต่ละหน่วยทดลองจะถูกนำมาพิจารณาร่องรอยไม่ดีในซอร์สโค้ดและพบว่าสามารถทำกระบวนการรีแฟคทอริงกับหน่วยทดลองทั้งหมด 10 วิธี แต่ไม่สามารถวิเคราะห์ข้อมูลของกระบวนการรีแฟคทอริงวิธีพูลอัพเมท็อดได้ เนื่องจากกระบวนการรีแฟคทอริงวิธีนี้ถูกพบในหน่วยทดลองเพียง 1 หน่วยทดลองเท่านั้น ทำให้ไม่สามารถนำไปวิเคราะห์ข้อมูลทางสถิติได้ เมื่อหน่วยทดลองผ่านกระบวนการรีแฟคทอริงตามสมมติฐานทั้ง 3 ข้อเรียบร้อยแล้ว จะถูกนำมาวัดค่าด้วยมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) 6 มาตรวัด มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) 1 มาตรวัด มาตรวัดเชิงวัตถุของ Abreu (1996) 6 มาตรวัด และมาตรวัดคอมเมนต์เปอร์เซ็นต์เทจ 1 มาตรวัด (Rosenberg และ Hyatt, 1995)

5.3 บทสรุป

งานวิจัยนี้เป็นการเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดโดยใช้มาตรวัดเชิงวัตถุ ซึ่งสามารถวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์คือ ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability) โดยอ้างอิงตามงานวิจัยของ Quenel และ Lövdahl (2004) ซึ่งได้กล่าวถึงผลการวิเคราะห์มาตรวัดเชิงวัตถุดังกล่าวกับคุณภาพซอฟต์แวร์ไว้ โดยสามารถแจกแจงตามแบบจำลองของ McCall (1977) ได้ดังตารางที่ 5-1

มาตรวัดเชิงวัตถุ / คุณภาพซอฟต์แวร์	Comment Percentage	Weighted Methods per Class	Depth of Inheritance Tree	Number of Children	Coupling Between Object Classes	Response for a Class	Lack of Cohesion in Method	Average Method Size	Method Hiding Factor	Attribute Hiding Factor	Method Inheritance Factor	Attribute Inheritance Factor	Polymorphism Factor	Coupling Factor
Maintainability	-	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓
Reusability	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	-	✓
Understandability	✓	✓	-	-	-	-	-	✓	-	-	-	-	-	-

ตารางที่ 5-1 แสดงตารางแบบจำลองความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์ที่เลือกใช้ในงานวิจัยนี้ (Quenel และ Lövdahl (2004))

หมายเหตุ ✓ หมายถึงมาตรวัดเชิงวัตถุมีความสัมพันธ์กับคุณภาพซอฟต์แวร์

\- หมายถึงมาตรวัดเชิงวัตถุไม่มีความสัมพันธ์กับคุณภาพซอฟต์แวร์

สำหรับการเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี การเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 1 วิธี กับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี และการเปรียบเทียบคุณภาพซอฟต์แวร์ของระหว่างซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริง 2 วิธี ที่มีการสลับลำดับกัน โดยการพิจารณาคุณภาพซอฟต์แวร์ทั้ง 3 ด้านจะใช้การพิจารณาจากมาตรวัดเชิงวัตถุทั้ง 14 ค่า ที่มีค่าดีขึ้นหรือด้อยลงตามรูปแบบการทำกระบวนการรีแฟคทอริง สามารถสรุปผลการเปรียบเทียบได้ดังนี้

5.3.1. คุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ผลการเปรียบเทียบคุณภาพซอฟต์แวร์ที่คำนวณค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง ปรากฏว่าคุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริงมีค่าที่ดีขึ้น ซึ่งเป็นไปตามที่ผู้วิจัยคาดหวังไว้ก่อนที่จะทดลองว่าซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงน่าจะมีคุณภาพซอฟต์แวร์สูงกว่าซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริง สำหรับการพิจารณาคุณภาพซอฟต์แวร์ที่ดีขึ้นหรือด้อยลงจะถูกพิจารณาจากค่ามาตรวัด

เชิงวัตถุที่ดีขึ้นและด้อยลงตามค่าสถิติที่ได้ทดลองในบทที่ 4 โดยสามารถแสดงจำนวนค่ามาตรวัดเชิงวัตถุที่ดีขึ้นและด้อยลงของกระบวนการรีแฟคทอริงแต่ละวิธี ได้ดังตารางที่ 5-2

No	Subject Name	จำนวนมาตรวัดเชิงวัตถุที่ดีขึ้น	จำนวนมาตรวัดเชิงวัตถุที่ด้อยลง	จำนวนมาตรวัดเชิงวัตถุที่เหลือ
1	Extract Method	5	2	7
2	Replace Temp with Query	3	2	9
3	Introduce Explaining Variable	1	1	12
4	Move Method	7	2	5
5	Move Field	7	4	3
6	Extract Class	8	1	5
7	Replace Magic Number with Symbolic Constant	3	2	9
8	Decompose Conditional	3	2	9
9	Rename Method	-	-	14

ตารางที่ 5-2 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ดีขึ้นและด้อยลง ตามวัตถุประสงค์ที่ 1

หมายเหตุ สดมภ์จำนวนมาตรวัดเชิงวัตถุที่เหลือ หมายถึงจำนวนมาตรวัดเชิงวัตถุที่ไม่สามารถวิเคราะห์ผลได้และไม่มีค่าที่ดีขึ้นหรือด้อยลงอย่างมีนัยสำคัญ

จากตารางที่ 5-2 จะเห็นได้ว่ากระบวนการรีแฟคทอริงแต่ละวิธีจะมีผลทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นหรือด้อยลงในจำนวนไม่เท่ากัน ทั้งนี้เป็นผลมาจากแต่ละวิธีของกระบวนการรีแฟคทอริงจะมีการเปลี่ยนแปลงลักษณะโครงสร้างซอร์สโค้ดที่แตกต่างกันไป โดยกระบวนการรีแฟคทอริงที่มีส่วนแก้ไขโครงสร้างหลักๆ ภายในซอร์สโค้ดจะมีผลทำให้ค่ามาตรวัดเชิงวัตถุมีการเปลี่ยนแปลงค่อนข้างมาก เช่น กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ หรือกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส โดยผู้วิจัยเห็นว่าโดยส่วนใหญ่กระบวนการรีแฟคทอริงจะมีผลทำให้ค่ามาตรวัดเชิงวัตถุมีค่าที่ดีขึ้น แต่จะมีบางกรณีที่จะเห็นได้ว่าการทำกระบวนการรีแฟคทอริงจะมีผลทำให้ค่ามาตรวัดเชิงวัตถุด้อยลง ซึ่งผู้วิจัยจะกล่าวในลำดับถัดไป (ในหัวข้อการวิเคราะห์ข้อมูลเพิ่มเติม)

ในการวิเคราะห์คุณภาพซอฟต์แวร์ก่อนทำกระบวนการรีแฟคทอริงและซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงนั้น จะพิจารณาจากค่ามาตรวัดเชิงวัตถุที่ดีขึ้นในทางบวกกับคุณภาพซอฟต์แวร์ โดยสามารถวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์คือ ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability) สำหรับการตอบ

วัตถุประสงค์ที่ 1 สามารถแยกการวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์ โดยแบ่งออกเป็น 9 รูปแบบตามแบบแผนการทดลองที่กล่าวมาแล้ว ซึ่งแต่ละรูปแบบนั้นส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นในระดับที่ไม่เท่ากัน สามารถแสดงได้ดังตารางที่ 5-3

No	Subject Name	จำนวนมาตรวัดเชิงวัตถุที่มีค่าดีขึ้นสำหรับคุณภาพซอฟต์แวร์		
		Maintainability	Reusability	Understandability
1	Extract Method	5	4	1
2	Replace Temp with Query	3	2	1
3	Introduce Explaining Variable	1	0	1
4	Move Method	6	5	2
5	Move Field	6	5	2
6	Extract Class	7	6	2
7	Replace Magic Number with Symbolic Constant	3	2	1
8	Decompose Conditional	3	2	1
9	Rename Method	0	0	0

ตารางที่ 5-3 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ดีขึ้นอย่างมีนัยสำคัญ ตามคุณภาพซอฟต์แวร์ด้านต่างๆ ตามวัตถุประสงค์ที่ 1

หมายเหตุ จำนวนมาตรวัดเชิงวัตถุที่ชี้วัดการดีขึ้นของคุณภาพซอฟต์แวร์ในแต่ละด้าน โดยความสามารถในการบำรุงรักษา (Maintainability) มีระดับสูงสุดคือ 12 มาตรวัด ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) มีระดับสูงสุดคือ 11 มาตรวัด ความสามารถในการทำความเข้าใจ (Understandability) มีระดับสูงสุดคือ 3 มาตรวัด และจำนวนมาตรวัดเชิงวัตถุที่ไม่เพิ่มขึ้นจะมีค่าเท่ากับ 0

จากตารางที่ 5-3 แสดงให้เห็นว่ากระบวนการรีแฟคทอริงทุกวิธีมีผลทำให้คุณภาพซอฟต์แวร์ดีขึ้น ยกเว้นกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด เพราะกระบวนการรีแฟคทอริงวิธีนี้เป็นการเปลี่ยนแปลงชื่อที่ไม่สื่อความหมายในซอร์สโค้ดไม่ได้ทำให้โครงสร้างของซอร์สโค้ดเปลี่ยนแปลงไป ซึ่งจากเหตุผลดังกล่าวผู้วิจัยจึงคาดว่ามาตรวัดเชิงวัตถุที่เลือกใช้ในงานวิจัยนี้ไม่มีความสามารถวัดการสื่อความหมายของชื่อได้จึงทำให้ไม่สามารถชี้ให้เห็นถึงการเปลี่ยนแปลงที่ดีขึ้นจากกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดนี้

จากผลการทดลองในสมมติฐานที่ 1 นี้ สรุปได้ว่ากระบวนการรีแฟคทอริงที่มีการปรับโครงสร้างซอร์สโค้ดหลักๆ เช่น กระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อด กระบวนการ

รีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ และกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส จะมีผลทำให้คุณภาพซอฟต์แวร์เพิ่มขึ้นสูงกว่าเมื่อเทียบกับกระบวนการรีแฟคทอริงวิธีอื่นๆ ใน 9 วิธีนี้

5.3.2. คุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริง 1 วิธี เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง 2 วิธี ผลการเปรียบเทียบคุณภาพซอฟต์แวร์ที่คำนวณค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริง 1 วิธี เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง 2 วิธี ปรากฏว่าคุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริง 2 วิธี มีค่าที่ดีขึ้น ซึ่งเป็นไปตามที่ผู้วิจัยคาดหวังไว้ก่อนที่จะทดลองว่าซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริง 2 วิธี น่าจะมีคุณภาพซอฟต์แวร์สูงกว่าซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงเพียงวิธีเดียว สำหรับการพิจารณาคุณภาพซอฟต์แวร์ที่ดีขึ้นหรือด้อยลงจะถูกพิจารณาจากค่ามาตรวัดเชิงวัตถุที่ดีขึ้นและด้อยลงตามค่าสถิติที่ได้ทดลองในบทที่ 4 โดยสามารถแสดงจำนวนค่ามาตรวัดเชิงวัตถุที่ดีขึ้นและด้อยลงของกระบวนการรีแฟคทอริงแต่ละคู่ ได้ดังตารางที่ 5-4

กระบวนการรีแฟคทอริงที่ทำเพิ่ม / กระบวนการรีแฟคทอริงที่ทำหลัก	Extract Method			Replace Temp with Query			Introduce Explaining Variable			Move Method			Move Field			Extract Class			Replace Magic Number with Symbolic Constant			Decompose Conditional			Rename Method		
	+	-	na	+	-	na	+	-	na	+	-	na	+	-	na	+	-	na	+	-	na	+	-	na	+	-	na
Extract Method				2	2	10	2	1	11	9	2	3	9	2	3	9	1	4	4	2	8	2	2	10	1	2	11
Replace Temp with Query	2	3	9				1	-	13	6	1	7	7	2	5	8	1	5	-	1	13	3	2	9	-	-	14
Introduce Explaining Variable	3	1	10	-	1	13				7	2	5	6	2	6	7	-	7	-	-	14	2	2	10	-	-	14
Move Method	4	2	8	1	1	12	1	1	12				-	-	14	-	-	14	3	3	8	3	2	9	1	-	13
Move Field	5	1	8	1	1	12	-	-	14	4	-	10				-	-	14	3	3	8	5	3	6	1	1	12
Extract Class	5	2	7	2	2	10	1	-	13	4	2	8	4	2	8				3	3	8	3	2	9	-	-	14
Replace Magic Number with Symbolic Constant	5	2	7	2	1	11	-	-	14	7	1	6	7	2	5	8	1	5				3	2	9	-	-	14
Decompose Conditional	5	2	7	2	2	10	-	-	14	8	2	4	7	2	5	8	1	5	-	1	13				1	-	13
Rename Method	5	2	7	2	3	9	-	2	12	7	2	5	8	2	4	8	1	5	2	2	10	4	2	8			

129

ตารางที่ 5-4 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ดีขึ้นและด้อยลง ตามวัตถุประสงค์ที่ 2

หมายเหตุ สดมภ์ช่องที่มีเครื่องหมายบวก หมายถึงจำนวนค่ามาตรวัดเชิงวัตถุที่มีค่าดีขึ้นอย่างมีนัยสำคัญ

สดมภ์ช่องที่มีเครื่องหมายลบ หมายถึงจำนวนค่ามาตรวัดเชิงวัตถุที่มีค่าด้อยลงอย่างมีนัยสำคัญ

สดมภ์ช่องที่มีอักษรย่อ na หมายถึงจำนวนค่ามาตรวัดเชิงวัตถุที่ไม่สามารถวิเคราะห์ผลได้และไม่มีค่าที่ดีขึ้นหรือด้อยลงอย่างมีนัยสำคัญ

จากตารางที่ 5-4 จะเห็นได้ว่ากระบวนการรีแฟคทอริงแต่ละคู่จะมีผลทำให้ค่ามาตรวัดเชิงวัตถุดีขึ้นหรือด้อยลงในจำนวนไม่เท่ากัน ทั้งนี้เป็นผลมาจากแต่ละวิธีของกระบวนการรีแฟคทอริงจะมีการเปลี่ยนแปลงลักษณะโครงสร้างซอร์สโค้ดที่แตกต่างกันไป เช่นเดียวกับวัตถุประสงค์ที่ 1

ในการวิเคราะห์คุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ทำกระบวนการรีแฟคทอริง 1 วิธี และซอร์สโค้ดที่ทำกระบวนการรีแฟคทอริง 2 วิธีนั้น จะพิจารณาจากค่ามาตรวัดเชิงวัตถุที่ดีขึ้นกับคุณภาพซอฟต์แวร์ โดยสามารถวิเคราะห์ความสัมพันธ์ระหว่างค่ามาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์คือ ความสามารถในการบำรุงรักษา (Maintainability) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) และความสามารถในการทำความเข้าใจ (Understandability) สำหรับการตอบวัตถุประสงค์ที่ 2 สามารถแยกการวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุกับคุณภาพซอฟต์แวร์ โดยแบ่งออกเป็น 72 รูปแบบตามแบบแผนการทดลองที่กล่าวมาแล้ว ซึ่งแต่ละรูปแบบของการทำกระบวนการรีแฟคทอริงนั้นส่งผลให้คุณภาพซอฟต์แวร์ดีขึ้นในระดับที่ไม่เท่ากัน สามารถแสดงได้ดังตารางที่ 5-5

กระบวนการรีแฟคทอริงที่ทำเพิ่ม / กระบวนการรีแฟคทอริงที่ทำหลัก	Extract Method			Replace Temp with Query			Introduce Explaining Variable			Move Method			Move Field			Extract Class			Replace Magic Number with Symbolic Constant			Decompose Conditional			Rename Method		
	M	R	U	M	R	U	M	R	U	M	R	U	M	R	U	M	R	U	M	R	U	M	R	U	M	R	U
Extract Method				2	1	1	2	1	1	8	7	2	8	7	2	7	6	2	4	3	1	2	1	1	1	0	1
Replace Temp with Query	2	1	1				1	0	1	5	4	2	6	5	2	7	6	2	0	0	0	3	1	2	0	0	0
Introduce Explaining Variable	3	2	1	0	0	0				6	5	2	5	4	2	6	6	1	0	0	0	2	1	1	0	0	0
Move Method	4	2	2	1	1	0	1	0	1				0	0	0	0	0	0	3	3	0	3	2	1	1	1	0
Move Field	5	4	1	1	1	0	0	0	0	4	3	1				0	0	0	3	3	0	5	3	2	1	1	0
Extract Class	5	4	1	2	1	1	1	0	1	4	3	1	4	3	1				3	3	0	3	2	1	0	0	0
Replace Magic Number with Symbolic Constant	5	4	1	2	1	1	0	0	0	6	5	2	6	5	2	7	6	2				3	2	1	0	0	0
Decompose Conditional	5	4	1	2	1	1	0	0	0	7	6	2	6	5	2	7	6	2	1	1	0				1	1	0
Rename Method	5	4	1	2	1	1	0	0	0	6	5	2	7	6	2	7	6	2	2	1	1	4	2	2			

131

ตารางที่ 5-5 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ดีขึ้นอย่างมีนัยสำคัญตามคุณภาพซอฟต์แวร์ตามวัตถุประสงค์ที่ 2

หมายเหตุ อักษรย่อ M หมายถึงคุณภาพซอฟต์แวร์ความสามารถการบำรุงรักษา (Maintainability)

อักษรย่อ R หมายถึงคุณภาพซอฟต์แวร์ความสามารถการนำกลับมาใช้ใหม่ (Reusability)

อักษรย่อ U หมายถึงคุณภาพซอฟต์แวร์ความสามารถการทำความเข้าใจ (Understandability)

วิธีการแปลความหมายตารางให้เริ่มอ่านจากด้านซ้ายไปบรรจบกับด้านบน เช่น แถวที่ 1 สดมภ์ที่ 2 สามารถแปลความหมายได้ว่ารูปแบบหน่วยทดลองที่ถูกทำกระบวนการรีแฟคทอริงวิธีเอ็กแทรกเมท็อดกับกระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรี โดยหน่วยทดลองที่ถูกทำด้วยกระบวนการรีแฟคทอริงทั้ง 2 วิธีนี้ มีผลความสามารถการบำรุงรักษา (Maintainability) เพิ่มขึ้น 2 มาตรวัด ความสามารถการนำกลับมาใช้ใหม่ (Reusability) เพิ่มขึ้น 1 มาตรวัด และความสามารถการทำความเข้าใจ (Understandability) เพิ่มขึ้น 1 มาตรวัด

จากตารางที่ 5-5 แสดงให้เห็นว่าการที่ซอร์สโค้ดจากหน่วยทดลองได้รับการทำกระบวนการรีแฟคทอริงเพิ่มขึ้นเป็น 2 วิธี จะมีผลให้คุณภาพซอฟต์แวร์ดีขึ้นเป็นส่วนใหญ่ แต่จะมีบางกรณีเท่านั้นที่ไม่พบการเพิ่มขึ้นของคุณภาพซอฟต์แวร์ ซึ่งทางผู้วิจัยคาดว่าน่าจะมีสาเหตุมาจาก 1) กระบวนการรีแฟคทอริงไม่ได้ทำให้โครงสร้างของซอร์สโค้ดเปลี่ยนแปลงไป ทำให้มาตรวัดเชิงวัตถุที่ใช้ในงานวิจัยนี้ไม่สามารถวัดค่าการเปลี่ยนแปลงที่เกิดขึ้นได้ เช่น กระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิล 2) ปริมาณของกระบวนการรีแฟคทอริงที่ทำกับหน่วยทดลองแต่ละหน่วยทดลองบางวิธีน้อยเกินกว่าที่จะทำให้เห็นถึงการเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ เช่น กระบวนการรีแฟคทอริงวิธีรีเพลสเทมป์วิธคิวรีถูกทำในหน่วยทดลอง 12 หน่วยทดลอง (ดูตารางที่ 4-4 ประกอบ) แต่เมื่อมองถึงปริมาณการทำกระบวนการรีแฟคทอริงในแต่ละหน่วยทดลองจะเห็นได้มีปริมาณเพียง 14 ครั้ง (ดูตารางที่ 4-3 ประกอบ)

สำหรับการทำกระบวนการรีแฟคทอริงเพิ่มขึ้นเป็น 2 วิธี จะมีผลให้คุณภาพซอฟต์แวร์ดีขึ้นเป็นส่วนใหญ่ ผู้วิจัยเห็นว่าการเพิ่มขึ้นของคุณภาพซอฟต์แวร์ที่ดีขึ้นมีผลขึ้นอยู่กับวิธีที่ทำกระบวนรีแฟคทอริงด้วย จากตารางที่ 5-5 จะเห็นได้ว่าการทำกระบวนการรีแฟคทอริง 2 วิธีนี้ ผลของคุณภาพซอฟต์แวร์จะเพิ่มขึ้นตามกระบวนการรีแฟคทอริงวิธีที่ 2 ที่ทำเพิ่มเข้าไป ตัวอย่างเช่น การทำกระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิลเพียง 1 วิธี จะเพิ่มคุณภาพความสามารถการบำรุงรักษา (Maintainability) และความสามารถการทำความเข้าใจ

(Understandability) เพียง 1 มาตรวัด แต่จะไม่เพิ่มคุณภาพความสามารถการนำกลับมาใช้ใหม่ (Reusability) แต่เมื่อนำกระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิลทำร่วมกับกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดกลับให้ผลคุณภาพเพิ่มขึ้นคือ ความสามารถการบำรุงรักษา (Maintainability) 6 มาตรวัด ความสามารถการนำกลับมาใช้ใหม่ (Reusability) 5 มาตรวัด และความสามารถการทำความเข้าใจ (Understandability) 2 มาตรวัด ซึ่งสามารถเห็นได้ชัดเจนว่ามีผลมาจากกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดที่ทำให้คุณภาพซอฟต์แวร์ดีขึ้น

5.3.3. คุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x ผลการเปรียบเทียบคุณภาพซอฟต์แวร์ที่คำนวณค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริง 2 วิธีที่มีการสลับลำดับกัน ปรากฏว่าคุณภาพซอฟต์แวร์ที่พิจารณาจากค่ามาตรวัดเชิงวัตถุหลังทำกระบวนการรีแฟคทอริงไม่มีค่าแตกต่างไปจากเดิม ซึ่งไม่เป็นไปตามที่ผู้วิจัยคาดหวังไว้ โดยผู้วิจัยคาดหวังว่าซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริง 2 วิธีที่สลับลำดับกันจะมีคุณภาพซอฟต์แวร์แตกต่างกัน

ในการวิเคราะห์คุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ทำกระบวนการรีแฟคทอริง 2 วิธีที่สลับลำดับนั้น จะเห็นได้ว่าคุณภาพซอฟต์แวร์ด้านความสามารถการบำรุงรักษา (Maintainability) ความสามารถการนำกลับมาใช้ใหม่ (Reusability) และความสามารถการทำความเข้าใจ (Understandability) ไม่แตกต่างกัน เนื่องจากเนื้อหาซอร์สโค้ดจากหน่วยทดลองที่ผ่านการสลับลำดับวิธีทำกระบวนการรีแฟคทอริงนี้ไม่มีความแตกต่างกัน ดังนั้นจึงไม่มีค่ามาตรวัดเชิงวัตถุชนิดใดที่มีค่าแตกต่างกัน

5.3.4. สรุปการวิเคราะห์ข้อมูลเพิ่มเติม (Exploration) จากการวิเคราะห์ข้อมูลเพิ่มเติมในสมมติฐานที่ 4 สามารถวิเคราะห์ความสัมพันธ์ระหว่างมาตรวัดเชิงวัตถุที่ด้อยลงกับคุณภาพซอฟต์แวร์ สามารถแสดงได้ดังตารางที่ 5-6

No	Subject Name	มาตรวัดเชิงวัตถุที่มีค่าด้อยลงสำหรับคุณภาพซอฟต์แวร์		
		Maintainability	**Reusability**	**Understandability**
1	Extract Method	Method Inheritance Factor	Method Inheritance Factor	CommentPercentage
2	Replace Temp with Query	Method Inheritance Factor	Method Inheritance Factor	CommentPercentage
3	Introduce Explaining Variable	-	-	CommentPercentage
4	Move Method	Method Inheritance Factor	Method Inheritance Factor	CommentPercentage
5	Move Field	Lack of Cohesion in Method, Attribute Hiding Factor, Method Inheritance Factor	Lack of Cohesion in Method, Attribute Hiding Factor, Method Inheritance Factor	CommentPercentage

ตารางที่ 5-6 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ด้อยลงอย่างมีนัยสำคัญ ตามคุณภาพซอฟต์แวร์ด้านต่างๆ ตามวัตถุประสงค์ที่ 4

No	Subject Name	มาตรวัดเชิงวัตถุที่มีค่าด้อยลงสำหรับคุณภาพซอฟต์แวร์		
		Maintainability	**Reusability**	**Understandability**
6	Extract Class	-	-	CommentPercentage
7	Replace Magic Number with Symbolic Constant	Attribute Hiding Factor	Attribute Hiding Factor	CommentPercentage
8	Decompose Conditional	Attribute Hiding Factor	Attribute Hiding Factor	CommentPercentage
9	Rename Method	-	-	-

ตารางที่ 5-6 แสดงตารางจำนวนมาตรวัดเชิงวัตถุที่ด้อยลงอย่างมีนัยสำคัญ ตามคุณภาพซอฟต์แวร์ด้านต่างๆ ตามวัตถุประสงค์ที่ 4 (ต่อ)

จากตารางที่ 5-6 แสดงให้เห็นว่ากระบวนการรีแฟคทอริงเกือบทุกวิธีมีผลทำให้คุณภาพซอฟต์แวร์ด้านความสามารถในการทำความเข้าใจ (Understandability) ด้อยลง ยกเว้นกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อด ผู้วิจัยเห็นว่ามีสาเหตุมาจากการทำกระบวนการรีแฟคทอริงโดยใช้เครื่องมือรีชาร์ปเปอร์ 2.0 (Resharper 2.0) ไม่ได้มีส่วนเพิ่มคำอธิบายให้กับซอร์สโค้ดแต่จะทำให้จำนวนบรรทัดเพิ่มขึ้นจึงเป็นผลให้ค่ามาตรวัดคอมเมนต์เปอร์เซ็นต์เทจมีค่าด้อยลง ซึ่งเป็นมาตรวัดมีส่วนเกี่ยวข้องกับคุณภาพซอฟต์แวร์ด้านนี้ สำหรับกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดที่เป็นการเปลี่ยนชื่อให้สื่อความหมายจะไม่พบการด้อยค่าลงเพราะกระบวนการรีแฟคทอริงวิธีรีเนมเมท็อดนี้ไม่ได้ทำให้จำนวนบรรทัดเพิ่มขึ้นอย่างมีนัยสำคัญจึงไม่พบการด้อยลงของคุณภาพซอฟต์แวร์

สำหรับคุณภาพซอฟต์แวร์ด้านความสามารถในการบำรุงรักษา (Maintainability) และความสามารถในการนำกลับมาใช้ใหม่ (Reusability) มีส่วนด้อยลง ผู้วิจัยพบว่าการด้อยค่าลงของคุณภาพซอฟต์แวร์ 2 ด้านนี้มีสาเหตุมาจากกระบวนการรีแฟคทอริงดังกล่าวมีส่วนแก้ไขเมท็อดหรือแก้ไขฟิลด์ ภายในระบบจึงมีผลทำให้ค่ามาตรวัดเชิงวัตถุที่เกี่ยวข้องกับการวัดค่าสิ่งเหล่านี้คือ มาตรวัดแลคออฟโคฮีชันอินเมท็อด มาตรวัดแอตทริบิวต์ไฮดิงแฟคเทอร์ และมาตรวัดเมท็อดอินเฮอริแทนแฟคเทอร์ลดเป็นอย่างน้อย 1 มาตรวัด สำหรับสมมติฐานที่ 5 จะมีผลที่สอดคล้องกันคือ กระบวนการรีแฟคทอริงโดยส่วนใหญ่มีผลทำให้คุณภาพซอฟต์แวร์ด้านความสามารถในการทำความเข้าใจ (Understandability) ด้อยลง โดยผู้วิจัยเห็นว่าน่าจะมีสาเหตุเช่นเดียวกับสมมติฐานที่ 4 คือการทำกระบวนการรีแฟคทอริงโดยใช้เครื่องมือรีชาร์ปเปอร์ 2.0 (Resharper 2.0) ไม่ได้มีส่วนเพิ่มคำอธิบายให้กับซอร์สโค้ดแต่จะทำให้จำนวนบรรทัดเพิ่มขึ้นจึงเป็นผลให้ค่ามาตรวัดคอมเมนต์เปอร์เซ็นต์เทจมีค่าด้อยลง และผู้วิจัยสังเกตเห็นว่าการทำกระบวนการรีแฟคทอริง 2 วิธี จะมีผลทำ

ให้การด้อยค่าลงมีจำนวนที่เพิ่มขึ้นซึ่งเป็นสาเหตุจากกระบวนการรีแฟคทอริงที่ทำร่วมเข้าไปเป็นวิธีที่ 2 เช่น การทำกระบวนการรีแฟคทอริงวิธีอินโทรดิวซ์เอ็กเพลนนิงแวริเอเบิลเมื่อนำกระบวนการรีแฟคทอริงวิธีมูฟเมท็อดหรือวิธีมูฟฟิลด์มาทำร่วมนั้น จะพบการด้อยค่าลงมีจำนวนที่เพิ่มขึ้น

5.4 การนำงานวิจัยไปประยุกต์ใช้ (Contribution)

งานวิจัยนี้นอกจากนำไปใช้ในเชิงทฤษฎีแล้ว ยังสามารถนำไปประยุกต์ใช้ได้ ดังต่อไปนี้

5.4.1 การนำงานวิจัยไปใช้ในเชิงทฤษฎี (Theoretical Contribution) งานวิจัยนี้เป็นการต่อยอดองค์ความรู้ด้านการพัฒนาซอฟต์แวร์ให้มีคุณภาพมากขึ้น เนื่องจากผู้วิจัยทบทวนวรรณกรรมในอดีต (Literature Review) แล้วพบว่ากระบวนรีแฟคทอริงเป็นหนึ่งในกระบวนการที่ใช้ช่วยปรับปรุงซอร์สโค้ด โดยมีงานวิจัยของ Demeyer และคณะ (2004) ได้นำเสนอผลที่ดีขึ้นจากการทำกระบวนการรีแฟคทอริงในซอร์สโค้ดในด้านการเชื่อมต่อ (Coupling) และการทำงานร่วมกัน (Cohesion) โดยไม่ได้กล่าวสรุปถึงคุณภาพซอฟต์แวร์ในแต่ละด้าน ดังนั้นผู้วิจัยจึงได้ศึกษาเปรียบเทียบกระบวนการรีแฟคทอริงแต่ละวิธีว่ามีผลส่งให้คุณภาพซอฟต์แวร์ด้านใดเพิ่มขึ้นบ้าง และการนำกระบวนการรีแฟคทอริงมาใช้ร่วมกันจะมีผลกับคุณภาพซอฟต์แวร์อย่างไรบ้าง ซึ่งประเด็นนี้จะเป็นส่วนที่สนับสนุนว่าการทำกระบวนการรีแฟคทอริงมีผลทำให้คุณภาพดีขึ้นจริง ทั้งคุณภาพในด้านซอฟต์แวร์เชิงวัตถุและในด้านของซอฟต์แวร์ทั่วไป

สำหรับงานวิจัยของ Mens และคณะ (2004) ได้กล่าวถึงลำดับขั้นตอนในการทำกระบวนการรีแฟคทอริงบางคู่ที่ไม่สามารถทำงานร่วมกันได้ ซึ่งผลการวิเคราะห์ในงานวิจัยนี้ได้ถูกนำมาพัฒนาไปเป็นส่วนหนึ่งในเครื่องมือที่ใช้ทำกระบวนการรีแฟคทอริง ตัวอย่างเช่น เครื่องมือรีชาร์ปเปอร์ 2.0 (ReSharper 2.0) สำหรับเครื่องมือการทำกระบวนการรีแฟคทอริงในปัจจุบันจะถูกพัฒนาให้สามารถทำกระบวนการรีแฟคทอริงแล้วไม่เกิดการขัดแย้งกันแล้วก็ตาม แต่ในประเด็นของการทำกระบวนการรีแฟคทอริงที่สลับลำดับวิธีการทำยังเป็นประเด็นที่น่าสนใจ ดังนั้นผู้วิจัยจึงได้นำแนวทางดังกล่าวมาศึกษาว่าเมื่อนำกระบวนการรีแฟคทอริงในแต่ละวิธีมาจับคู่แล้วสลับลำดับวิธีในการทำ จะมีผลต่อคุณภาพซอฟต์แวร์หรือไม่ ซึ่งจากผลการทดลองจะเห็นได้ว่าการสลับลำดับวิธีการทำกระบวนการรีแฟคทอริงจะไม่มีผลทำให้คุณภาพซอฟต์แวร์เปลี่ยนแปลงไป ทั้งนี้เพื่อเป็นแนวทางให้กับผู้ที่สนใจในเรื่องกระบวนการรีแฟคทอริง คุณภาพซอฟต์แวร์ และมาตรวัดเชิงวัตถุได้นำข้อมูลเหล่านี้ไปใช้ในการศึกษาต่อไป

5.4.2 การนำงานวิจัยไปใช้ในเชิงประยุกต์ (Practical Contribution) จากผลการทดลองของงานวิจัยนี้แสดงให้เห็นว่า การทำกระบวนการรีแฟคทอริงมีผลทำให้คุณภาพซอฟต์แวร์ดีขึ้นไม่ว่าจะเป็นการเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับหลังทำกระบวนการรีแฟคทอริงในแต่ละวิธี และการเปรียบเทียบคุณภาพซอฟต์แวร์จากซอร์สโค้ดที่ทำ

กระบวนการรีแฟคทอริงเพียงวิธีเดียวกับ 2 วิธีมีกระบวนการรีแฟคทอริงบางวิธีเท่านั้น ที่ไม่ทำให้คุณภาพซอฟต์แวร์ดีขึ้น ซึ่งผู้วิจัยคาดว่าจะมีสาเหตุดังที่กล่าวมาแล้วข้างต้น สำหรับกรณีการทำกระบวนการรีแฟคทอริงที่สลับลำดับวิธีจะเห็นได้ว่าไม่ว่าจะเลือกทำกระบวนการรีแฟคทอริงวิธีใดก่อนก็ไม่ทำให้ผลของคุณภาพซอฟต์แวร์เปลี่ยนแปลงไป

จากผลสรุปดังกล่าวสามารถนำผลสรุปจากงานวิจัยนี้ไปใช้แนะนำให้กับนิสิตหรือนักพัฒนาที่เพิ่งเริ่มต้นพัฒนาซอฟต์แวร์เชิงวัตถุว่าประเด็นไหนในหลักการแบบเชิงวัตถุที่ควรให้ความสำคัญ เช่น การสร้างระบบซอฟต์แวร์เชิงวัตถุควรพิจารณาหน้าที่การทำงานของแต่ละคลาสและเมท็อดให้มีความชัดเจน เพราะจะเห็นได้ว่าการทำกระบวนการรีแฟคทอริงวิธีมูฟเมท็อด กระบวนการรีแฟคทอริงวิธีมูฟฟิลด์ และกระบวนการรีแฟคทอริงวิธีเอ็กแทรกคลาส มีผลให้คุณภาพซอฟต์แวร์ดีขึ้นมากจึงควรแนะนำและปลูกฝังให้นักพัฒนาที่เริ่มเรียนรู้เข้าใจเพื่อที่จะให้ซอฟต์แวร์ที่พัฒนาออกมามีคุณภาพซอฟต์แวร์ที่ดี เพื่อเป็นส่วนช่วยให้เข้าใจแนวคิดของซอฟต์แวร์เชิงวัตถุมากยิ่งขึ้น

จากผลสรุปในงานวิจัยนี้ยังสามารถใช้เป็นคำแนะนำสำหรับนิสิตหรือนักพัฒนาที่เพิ่งเริ่มต้นได้ในกรณีที่ต้องทำกระบวนการรีแฟคทอริงกับซอร์สโค้ด แต่ไม่ทราบว่าควรเริ่มต้นทำกระบวนการรีแฟคทอริงวิธีใด เริ่มต้นทำส่วนใดของซอร์สโค้ด และทำกระบวนการรีแฟคทอริงวิธีใดจะเพิ่มคุณภาพด้านไหน ในประเด็นนี้เป็นสิ่งสำคัญที่ควรจะแนะนำให้นิสิตหรือนักพัฒนาที่เพิ่งเริ่มต้นเข้าใจหลักการใช้กระบวนการรีแฟคทอริง นอกจากจะเป็นส่วนช่วยให้เข้าใจแนวคิดของซอฟต์แวร์เชิงวัตถุแล้วยังช่วยให้นิสิตหรือนักพัฒนาที่เพิ่งเริ่มต้นเลือกใช้กระบวนการรีแฟคทอริงได้ตรงกับปัญหาที่เกิดขึ้น โดยอาจจะเริ่มต้นแนะนำกระบวนการรีแฟคทอริงวิธีที่สำคัญๆ ก่อน เช่น กระบวนการรีแฟคทอริงวิธีที่พบในนิสิตหรือนักพัฒนาที่เพิ่งเริ่มต้นมากที่สุดในงานวิจัยนี้ หรือวิธีที่ทำให้คุณภาพซอฟต์แวร์ดีขึ้นมากที่สุดในงานวิจัยนี้

ถึงแม้ว่าการพิจารณาร่องรอยไม่ดีในซอร์สโค้ดเป็นเพียงคู่มือที่ใช้แนะแนวทาง โดยการพิจารณาจำเป็นต้องพึ่งการตัดสินใจจากผู้ใช้อีกที แต่จากผลสรุปในงานวิจัยนี้สามารถใช้เป็นข้อควรระวังสำหรับการพิจารณาเลือกใช้กระบวนการรีแฟคทอริงในแต่ละวิธี โดยถ้าผู้ที่สนใจเลือกใช้กระบวนการรีแฟคทอริงไปใช้อย่างไม่ถูกต้องก็จะเปรียบเสมือนการใช้ยาไม่ถูกกับโรค ซึ่งจะมีผลทำให้คุณภาพซอฟต์แวร์ด้อยลงได้ ดังนั้นการเลือกใช้กระบวนการรีแฟคทอริงแต่ละวิธีให้ถูกกับปัญหาของซอร์สโค้ด จะช่วยให้ซอร์สโค้ดมีคุณภาพซอฟต์แวร์ที่ดีขึ้นด้วย

จากผลการสรุปในประเด็นการทำกระบวนการรีแฟคทอริงสลับลำดับวิธีที่ไม่มีผลกับการเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ที่ดีขึ้น แต่ผู้วิจัยพบว่าลำดับวิธีในการทำกระบวนการรีแฟคทอริงจะมีผลกับแรงงาน (Effort) ในการทำกระบวนการรีแฟคทอริงที่ไม่เท่ากัน โดยผู้วิจัย

เห็นว่าถ้าผู้ที่ใช้กระบวนการรีแฟคทอริงพิจารณาลำดับวิธีในการทำกระบวนการรีแฟคทอริงให้เหมาะสม จะไม่ทำให้เกิดการทำกระบวนการรีแฟคทอริงที่ซ้ำซ้อนกัน

5.5 ข้อจำกัดและข้อเสนอแนะของงานวิจัย

งานวิจัยนี้มีข้อจำกัดบางประการ ซึ่งผู้วิจัยขอสรุปและเสนอแนะแนวทาง ดังต่อไปนี้

1. การนำผลการทดลองไปใช้ให้คำนึงด้วยว่า หน่วยทดลองของงานวิจัยนี้เป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิตที่ผ่านการเรียนวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ไม่ใช่งานที่ผลิตจากองค์กรพัฒนาซอฟต์แวร์ ดังนั้นในการนำไปขยายผลกับงานในองค์กรพัฒนาซอฟต์แวร์อาจได้ผลที่แตกต่างออกไป

2. จากผลการทดลองผู้วิจัยพบกระบวนการรีแฟคทอริงที่ต้องทำในซอร์สโค้ดหน่วยทดลองทั้งหมด 10 วิธี ซึ่งผู้วิจัยได้นำมาวิเคราะห์วัดผลต่อมาแล้วจึงสรุปได้ผลการทดลองดังกล่าว ดังนั้นการนำผลการทดลองไปขยายกับกระบวนการรีแฟคทอริงวิธีอื่นๆ อาจได้ผลที่แตกต่างออกไป

3. การเลือกใช้กระบวนการรีแฟคทอริงกับซอฟต์แวร์ควรชั่งน้ำหนักเรื่องคุณภาพซอฟต์แวร์ที่ต้องการให้มีในระบบด้วย ซึ่งผู้วิจัยเห็นว่ากระบวนการรีแฟคทอริงแต่ละวิธีมีจุดเด่น จุดด้อยในคุณภาพซอฟต์แวร์ที่แตกต่างกัน

4. ในงานวิจัยนี้ผู้วิจัยได้เลือกมาตรวัดเชิงวัตถุทั้งหมด 14 มาตรวัด โดยแต่ละมาตรวัดเป็นมาตรวัดที่ใช้ในการวัดซอฟต์แวร์เชิงวัตถุ ดังนั้นหากนำมาตรวัดทางซอฟต์แวร์นอกเหนือจากนี้มาใช้วัดอาจทำให้ได้ผลที่แตกต่างออกไป

5. ควรมีการศึกษาต่อไปโดยเปลี่ยนแปลงแผนแบบการทดลอง เช่น นำซอร์สโค้ดที่ได้จากองค์กรผลิตซอฟต์แวร์ เปรียบเทียบซอร์สโค้ดที่ทำกระบวนการรีแฟคทอริงมากกว่า 2 วิธีขึ้นไป หรือนำมาตรวัดทางซอฟต์แวร์อื่นๆ เข้ามาใช้วัดเพิ่มขึ้น เป็นต้น ทั้งนี้เพื่อดูว่าคุณภาพซอฟต์แวร์เปลี่ยนแปลงไปอย่างไร

6. งานวิจัยนี้เป็นการเปรียบเทียบคุณภาพซอฟต์แวร์ โดยใช้มาตรวัดเชิงวัตถุเป็นตัวชี้วัด จึงควรมีการต่อยอด โดยการนำมาตรวัดเชิงวัตถุมาวิเคราะห์ความสัมพันธ์ของตัวแปรมากกว่า 2 ตัว ที่มีการถ่วงน้ำหนักให้กับมาตรวัดเชิงวัตถุแต่ละค่า แล้วจึงคำนวณเพื่อให้ได้ผลของคุณภาพซอฟต์แวร์ ทั้งนี้เพื่อให้ได้ผลการเปลี่ยนแปลงของคุณภาพซอฟต์แวร์ที่แท้จริง

รายการอ้างอิง

ภาษาไทย

กัลยา วาณิชย์บัญชา. 2544. หลักสถิติ. พิมพ์ครั้งที่ 6. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

กัลยา วานิชย์บัญชา. 2548. การใช้ SPSS for Windows ในการวิเคราะห์ข้อมูล. พิมพ์ครั้งที่ 7. กรุงเทพมหานคร: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

ศิริวรรณ เสรีรัตน์, สมชาย หิรัญกิตติ, จิรศักดิ์ จิยะจันทน์, ชวลิต ประภวานนท์, ณดา จันทร์สม และวลัยลักษณ์ อัตธีรวงศ์. 2541. การวิจัยธุรกิจ. กรุงเทพมหานคร: สำนักพิมพ์เพชรจรัสแสงแห่งโลกธุรกิจ.

อำนาจ วังจีน และ พรรณี บุญสุยา 2548. สถิติทั่วไป. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร: สำนักวิชาการศึกษาทั่วไป มหาลัยศรีปทุม

ภาษาอังกฤษ

Abreu, F.B. and Melo, W. 1996. Evaluating the Impact of Object-Oriented Design on Software Quality. In Proceedings of the 3rd International Software Metrics Symposium : 90 – 99.

ANSI standard (ANSI/ASQC A3/1978). 1978. Quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given need.

Babbie, E. 2001. The Practice of Social Research. Ninth Edition. USA: Wadsworth/Thomson Learning.

Beck, K. 1999. Smalltalk Best Practice Patterns. Upper Saddle River: Prentice-Hall.

Beck, K. 1999. Extreme Programming Explained United States of America: Addison-Wesley.

Berard, E.V. 1993. Essays on Object-Oriented Software Engineering, Volume 1, Englewood Cliffs, New Jersey: Prentice Hall.

Bieman, J. and Ott, L. 1994. Measuring Functional Cohesion. IEEE Transactions on Software Engineering. 20, 8: 644—657.

Bois, B.D., Demeyer, S., and Verelst, J. 2004. Refactoring - Improving Coupling and Cohesion of Existing Code. In Proceedings of the 11th Working Conference 8, 12: 144 – 151.

Bois, B.D. and Mens, T. 2003. Describing the Impact of Refactorings on Internal Program Quality. Bart Du Bois Lab On ReEngineering Universiteit Antwerpen Middelheimlaan

Borland, JBuilder 2006. Available from: www.borland.com/jbuilder/[2006, March 5]

Brant, J.and Robert, D. 1999. Refactoring Browser Tool Available from: http://st-www.cs.uiuc.edu/~brant/RefactoringBrowser[1999, July 18]

Bodin, F., 1994. Sage++: An Object - Oriented Toolkit and Class Library for Building Fortran and C++ Restructuring Tools. In Proceedings of the 2nd Object-Oriented Numerics Conference. Sunriver, Oregon.

Chen, J. Y. and Lu, J. F. 1993. A new metric for object-oriented design. Journal of Information and Software technology. 35, 232 – 240.

Chidamber, R.S.and Kemerer, C.F. 1994. A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering Volume 20, Issue 6, IEEE Computer Society. : 476 – 493.

Cincom Smalltalk VisualWorks 2006. Available from: www.cincomsmalltalk.com/[2006, March 5]

Dagpinar, M.and Jahnke, J.H. 2003. Predicting Maintainability with Object-Oriented Metrics - An Empirical Comparison. Proceedings of the 10th Working Conference on Reverse Engineering. : 155 – 164.

Demacro, T. 1982. Controlling Software Projects. New York: Yourdon Press.

Demeyer, S. 2002. Maintainability versus Performance: What's the Effect of Introducing Polymorphism? Technical Report Laboratory. On Reengineering. University Antwerpen Belgium.

Demeyer, S. Du Bois, B. Verelst, J. 2004. Refactoring - Improving Coupling and Cohesion of Existing Code. Reverse Engineering, 2004. Proceedings. 11th Working Conference. : 144-151.

Dhama, H. 1995. Quantitative models of cohesion and coupling in software. Source. Journal of Systems and Software archive. 29.

DIN 55350-11, German Industry Standard DIN 55350 Part 11 : Quality comprises all characteristics and significant features of a product of an activity which relate to the satisfying of given requirements.

Dudziak, T. and Wloka, J. 2002. Tool-Supported Discovery and Refactoring of Structural Weaknesses in Code. Master's thesis, Faculty of Computer Science, Technical University of Berlin.

Eclipse.org, Eclipse Version 1.0.5.4 2006. Available from: www.eclipse.org/[2006, March 5]

Emden, E. V. and Moonen, L. 2002. Java Quality Assurance by Detecting Code Smells. In Proceedings of the 9th Working Conference on Reverse Engineering. : 97 – 108.

Fenton, N. E.and Pfleeger, S.L. 1997. Software Metrics, A Rigorous approach & Practical Approach. Second Edition United Kingdom: InternationalThomson Computer Press.

Fowler, M. 1999. Refactoring Improving the Design of Existing Code. United States of America: Addison Wesley.

Harrison, R., Counsell, S., and. Nithi, R, 1997. An Overview of Object-Oriented Design Metrics. Proceedings of the 8th International Workshop on Software Technology and Engineering Practice.

IEEE Standard for Software Quality (IEEE Std 729 - 1983). 1983.

Instantiations, jFactor 2006. Available from: www.instantiations.com/jfactor/[2006, March 5]

IntelliJ, IDEA 2006. Available from: www.intellij.com/idea/[2006, March 5]

International Standard. ISO/IEC 9126-1. 2006. Institute of Electrical and Elec- tronics Engineers, Part 1, 2, 3: Quality model, Available from: http://www.iso.ch[2006, January 11]

Joshi, P. and Joshi, R.K. 2006. Microscopic Coupling Metrics for Refactoring. In Proceedings of the 10th European Conference on Software Maintenance and Reengineering.

Kan, S. H. 2000. Metrics and Models in Software Quality Engineering. United States of America: Addison-Wesley.

Kataoka, Y., Ernst, M.D., Griswold, W.G. and Notkin, D. 2001. "Automated Support for Program Refactoring Using Invariants." In Proceedings of the IEEE International Conference on Software Maintenance. : 736 - 743.

Kataoka, Y., Imai, T., Andou, H., and Fukaya, T. 2002. A Quantitative Evaluation of Maintainability Enhancement by Refactoring In Proceedings of the IEEE International Conference on Software Maintenance. : 576 – 585.

Khosravi, K. and Guéhéneuc, Y.G. 2004. A Quality Model for Design Patterns Technical Report 1249, University of Montreal.

Kitchenham, B. and Pfleeger, S. L. 1996. Software quality: The elusive target. IEEE Software. : 12 - 21.

Kitchenham, B.and Walker, J., 1989. A Quantitative Approach to Monitoring Software Development. Software Engineering Journal. : 2 – 14.

Lanubile, F. and Visaggio G., 1997. Extracting Reusable Functions by Flow Graph-based Program Slicing. IEEE Transactions on Software Engineering 23, 4 : 246 – 258.

Li, W. and Henry, S. 1993. Object- Oriented Metrics that Predict Maintainability. Journal of Systems and Software. : 23, 111– 122.

Lorenz, M. and Kidd, J, 1994. Object-Oriented Software Metrics. United States of America: Prentice-Hall International.

Mancl, D. 2001. Refactoring for Software Migration. IEEE Communications Magazine.

McCabe, T. J. and Butler, C. W., 1989. Design Complexity Measurement and Testing Communications of the ACM. 32, 12: 1415 – 1425.

McCall, J. A. 1977. An Introduction to Software Quality Metrics. New York: A Petrocelli Book.

Mens, T., Demeyer, S., Bois, B.D., Stenten, H. and Gorp, P.V. 2003. Refactoring: Current Research and Future Trends. ETAPS Fifth Workshop on Language Descriptions, Tools and Applications.

Mens, T., Demeyer, S. and Janssens, D., 2002. Formalising Behaviour Preserving Program Transformations in: Graph Transformation. Lecture Notes in Computer Science 2505. : 286 – 301.

Mens, T., Taentzer, G. and Runge, O. 2004. Detecting Structural Refactoring Conflicts Using Critical Pair Analysis. Workshop on Software Evolution through Transformations, Rome, Italy. : 113 – 128.

Neill, C.J. and Gill, B. 2003. Refactoring Reusable Business Components. IEEE IT Professional. 5, 1. : 33 - 38.

Neill, C. J. and Gill, B. 2003. Refactoring Reusable Business Components. IT Pro. : 33-38.

Opdyke, W. F. 1992. Refactoring: A Program Restructuring Aid in Designing Object-Oriented Application Frameworks. PhD thesis, University of Illinois at Urbana-Champaign.

Opdyke, W. F. 1999. Refactoring C++ programs. Technical report, Lucent Technologies/ Bell Labs. Available from: www.cs.uiuc.edu/users/opdyke/wfo.990201.c++.refac.html[1999, June 25]

Pipka, J.U. 2002. Refactoring in a 'Test First' – World. In Proceedings of the 3rd International conference on eXtreme Programming and Flexible Processes in Software Engineering.

Pressman, R. S. 2001. Software Engineering a practitioner's Approach. United States of America: McGraw-Hill.

Proietti, M. and Pettorossi, A. 1991. Semantics Preserving TransformationRules for Prolog. Proceedings ACM Symposium on Partial Evaluation and Semantics Based Program Manipulation. : 274 - 284.

Quenel, G. and Lövdahl, H. 2004. Object oriented software quality models Seminar in Software Quality.

Roberts, D., Brant J. and Johnson R., 1997. A Refactoring Tool for Smalltalk. Theory and Practice of Object Systems. 3 : 253 – 263.

Rosenberg, L.H. and Hyatt, L.E. 1995. Software Quality Metrics for Object-Oriented Environments Crosstalk Journal.

Schroeder, M. 1999. A Practical Guide to Object-Oriented Metrics. IT Pro : 30-36.

Simon, F., Steinbrückner, F., and Lewerent, C. 2001. Metrics Based Refactoring. In Proceedings of the 5th European Conference on Software Maintenance and ReengineeringIEEE Computer Society Press. : 30 – 38.

Sommerville, I. 2001. Software Engineering. Sixth Edition. United States of America: Addison Wesley.

Tahvildari, L. and Kontogiannis, K. 2002. A Methodology for Developing Transformations Using the Maintainability Soft-Goal Graph. In Proceedings of the 9th IEEE Working Conference on Reverse Engineering (WCRE), Richmond, Virginia, USA. : 77 – 86.

Tahvildari, L. and Kontogiannis, K. 2003. A Metric-Based Approach to Enhance Design Quality through Meta-Pattern Transformations. In Proceedings of the 7th IEEE European Conference on Software Maintenance and Reengineering. : 183 – 192.

TogetherSoft, ControlCenter 2006. Available from: www.togethersoft.com/[2006, March 5]

Tokuda, L. and Batory, D.S. 1999. Evolving Object-Oriented Designs with Refactorings. In Proceedings of ASE-99: The 14th IEEE Conference on Automated Software Engineering. IEEE Computer Society Press.

Tourwe´, T. and Mens, T. 2003. Identifying Refactoring Opportunities Using Logic Meta Programming. In Proceedings of the 7th European Conference on Software Maintenance and Reengineering. : 91 - 100.

XRef-Tech, XRefactory 2006. Available from: http://xref-tech.com/speller/main.html[2006, March 5]

ภาคผนวก

ภาคผนวก ก
เอกสารแสดงการออกแบบซอฟต์แวร์มาตรวัดเชิงวัตถุ

1. เอกสารแสดงการออกแบบซอฟต์แวร์มาตรวัดเชิงวัตถุ

จากข้อกำหนดของความต้องการ (Software Requirement Specification) สามารถออกแบบซอฟต์แวร์มาตรวัดเชิงวัตถุ โดยประกอบไปด้วยมาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) มาตรวัดเชิงวัตถุของ Abreu (1996) มาตรวัดความซับซ้อนและมาตรวัดอัตราส่วนร้อยละของคำอธิบาย ได้ดังนี้

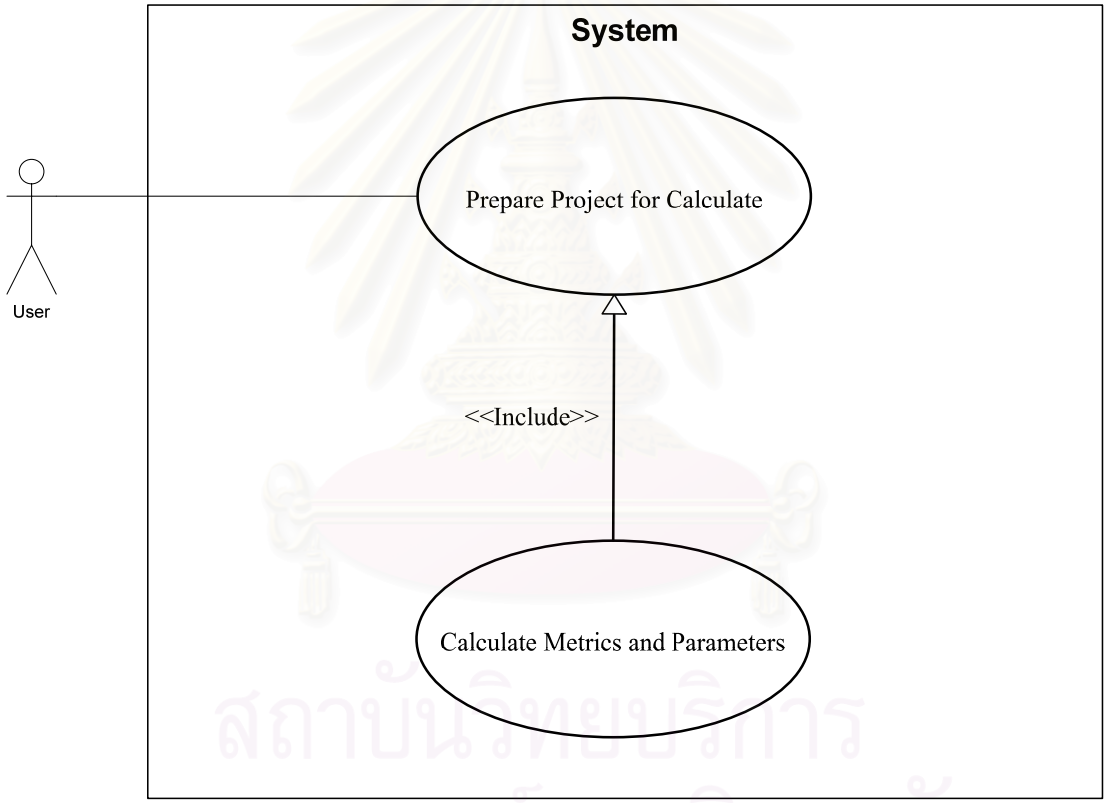
1.1 ความต้องการซอฟต์แวร์ (Requirement Description)

ระบบการคำนวณค่ามาตรวัดมาตรวัดเชิงวัตถุ (Object-Oriented Metrics) นี้จัดทำขึ้นเพื่อให้ผู้ใช้ (User) สามารถคำนวณค่ามาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) มาตรวัดเชิงวัตถุของ Abreu (1996) มาตรวัดความซับซ้อนและมาตรวัดอัตราส่วนร้อยละของคำอธิบาย โดยระบบประกอบด้วยฟังก์ชันการทำงานต่างๆ ดังต่อไปนี้

1. เมื่อผู้ใช้ต้องการคำนวณค่ามาตรวัดเชิงวัตถุ ระบบจะเรียกซอร์สโค้ดดอทซีเอส (.CS) มาคำนวณจากซอฟต์แวร์ที่เปิดใช้งานอยู่จากเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 (Visual Studio .NET 2005)
2. ระบบสามารถคำนวณค่ามาตรวัดเชิงวัตถุของ Chidamber และ Kemerer (1994) มาตรวัดเชิงวัตถุของ Lorenz และ Kidd (1994) มาตรวัดเชิงวัตถุของ Abreu (1996) มาตรวัดความซับซ้อน และมาตรวัดอัตราส่วนร้อยละของคำอธิบายได้
3. ระบบสามารถแสดงผลค่ามาตรวัดเชิงวัตถุได้
4. ระบบสามารถบันทึกค่ามาตรวัดเชิงวัตถุได้

1.2 แผนภาพยูสเคสและคำอธิบายยูสเคส (Use Case Diagram and Use Case Description)

แผนภาพยูสเคส (Use Case Diagram)



เอกสารคำอธิบายยูสเคส (Use Case Description)

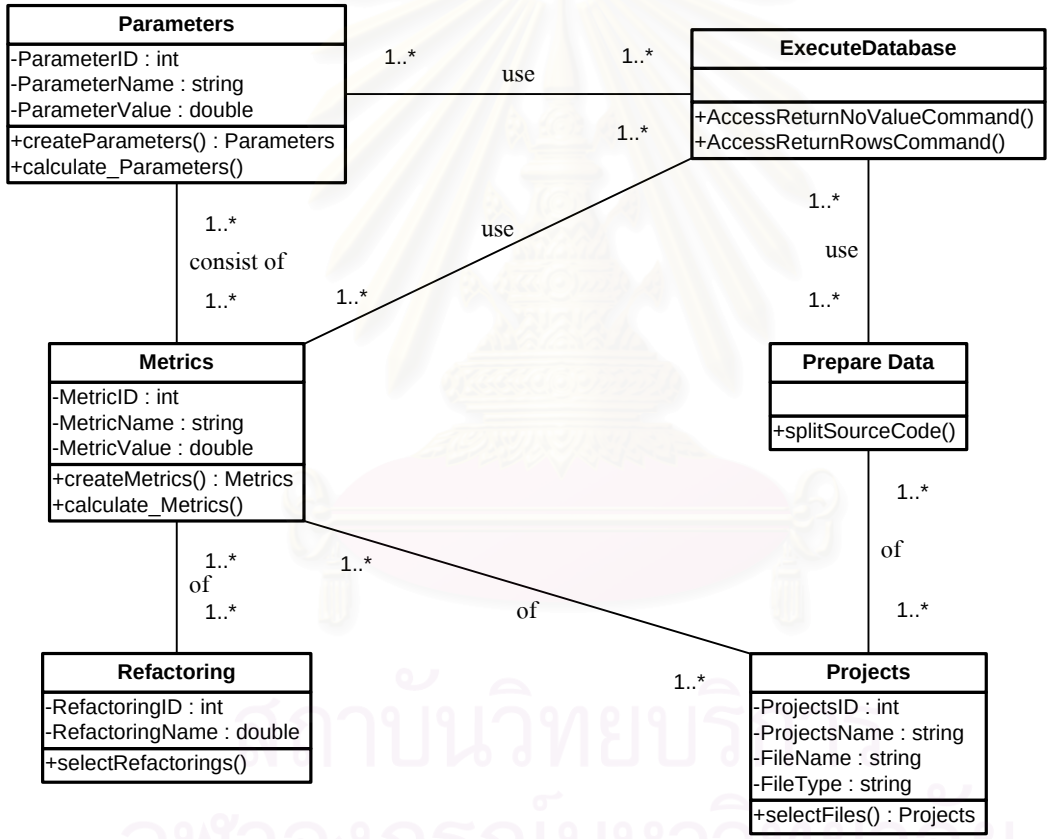
1. ยูสเคสการจัดเตรียมข้อมูลก่อนการคำนวณค่ามาตรวัดเชิงวัตถุ (Prepare Project for Calculate)	
เป้าหมาย (Goal)	จัดเตรียมข้อมูลของซอร์สโค้ดก่อนเข้าสู่การคำนวณค่ามาตรวัดเชิงวัตถุ
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Precondition)	ระบบติดต่อ (Connect) กับข้อมูลจากเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 เรียบร้อยแล้ว
เงื่อนไขหลัง (Postcondition)	ระบบจัดเตรียมข้อมูลสำหรับการคำนวณค่ามาตรวัดเชิงวัตถุเรียบร้อยแล้ว
เมนซัคเซส ซีนาริโอ (Main Success Scenario)	1. ผู้ใช้เรียกข้อมูลที่เป็นเอกสารนามสกุลซีเอส (File.cs) ของโปรเจ็กต์ที่ต้องการทำรีแฟคทอริง จากเครื่องมือวิชวลสตูดิโอดอทเน็ต 2005 2. ระบบแสดงเอกสารนามสกุลซีเอสทางหน้าจอ 3. ผู้ใช้เลือกรูปแบบการทำกระบวนการรีแฟคทอริง 4. ระบบจัดเตรียมเอกสารนามสกุลซีเอสให้อยู่ในรูปแบบที่สามารถคำนวณได้ โดยการแยกซอร์สโค้ดแต่ละเอกสารที่ได้รับมา 5. ระบบนับจำนวนเอกสาร และบันทึกข้อมูลรูปแบบการทำกระบวนการรีแฟคทอริง ข้อมูลชื่อโครงงานและข้อมูลชื่อเอกสาร

2. ยูสเคสการคำนวณค่ามาตรวัดเชิงวัตถุและค่าพารามิเตอร์ (Calculate Metrics and Parameters)	
เป้าหมาย (Goal)	จัดการคำนวณค่ามาตรวัดเชิงวัตถุและค่าพารามิเตอร์
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Precondition)	ระบบจัดเตรียมข้อมูลสำหรับการคำนวณค่ามาตรวัดเชิงวัตถุเรียบร้อยแล้ว
เงื่อนไขหลัง (Postcondition)	ระบบแสดงผลการคำนวณค่ามาตรวัดเชิงวัตถุทางหน้าจอเรียบร้อยแล้ว

เมนซัคเซส ซีนาริโอ **(Main Success Scenario)**	1. ผู้ใช้สร้างพารามิเตอร์สำหรับใช้ในการคำนวณค่ามาตรวัดเชิงวัตถุ 2. ได้พารามิเตอร์สำหรับใช้ในการคำนวณค่ามาตรวัดเชิงวัตถุตามที่สร้างไว้ 3. ระบบคำนวณค่าพารามิเตอร์สำหรับใช้ในการคำนวณค่ามาตรวัดเชิงวัตถุ 4. ระบบบันทึกข้อมูลค่าพารามิเตอร์ลงฐานข้อมูล 5. ผู้ใช้สร้างมาตรวัดเชิงวัตถุ 6. ได้มาตรวัดเชิงวัตถุตามที่สร้างไว้ 7. ระบบร้องขอข้อมูลค่าพารามิเตอร์จากฐานข้อมูล 8. ได้ข้อมูลค่าพารามิเตอร์ตามที่ร้องขอไว้ 9. ระบบคำนวณค่ามาตรวัดเชิงวัตถุจากข้อมูลค่าพารามิเตอร์ 10. ระบบบันทึกข้อมูลค่ามาตรวัดเชิงวัตถุลงฐานข้อมูล

1.3 แผนภาพคลาสและคำอธิบายคลาส (Class Diagram and Class Description)

แผนภาพคลาส (Class Diagram)



คำอธิบายคลาส (Class Description)

1. Metrics				
Description	มาตรวัดเชิงวัตถุ			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
MetricID	รหัสมาตรวัดเชิงวัตถุ	Number	-	Private
MetricName	ชื่อมาตรวัดเชิงวัตถุ	String	-	Private
MetricValue	ค่ามาตรวัดเชิงวัตถุ	Number	-	Private
Methods				
1. createMetrics				
Description	ฟังก์ชันที่ใช้สร้างข้อมูลมาตรวัดเชิงวัตถุ			
Visibility	Public			
Parameters	**Name**	**Description**	**Data Type**	
	MetricName	ชื่อมาตรวัดเชิงวัตถุ	String	
	Refactoring	รูปแบบกระบวนการรีแฟคทอริง	Refactoring	

	Project	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ	Project
Return Type	Metrics		
2. calculate_Metrics			
Description	คำนวณค่ามาตรวัดเชิงวัตถุ		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Parameters	พารามิเตอร์ที่ใช้ในการคำนวณค่ามาตรวัดเชิงวัตถุ	Parameters
	Refactoring	รูปแบบกระบวนการรีแฟคทอริง	Refactoring
	Project	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ	Project
Return Type	-		

2. Parameters				
Description	พารามิเตอร์ที่ใช้ในการคำนวณค่ามาตรวัดเชิงวัตถุ			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
ParameterID	รหัสพารามิเตอร์	Number	-	Private

ParameterName	ชื่อพารามิเตอร์	String	-	Private
ParameterValue	ค่าพารามิเตอร์	Number	-	Private

Methods			
1. createParameters			
Description	ฟังก์ชันที่ใช้สร้างข้อมูลพารามิเตอร์		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	ParameterName	ชื่อพารามิเตอร์	String
	Refactoring	รูปแบบกระบวนการรีแฟคทอริง	Refactoring
	Project	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ	Project
Return Type	Parameters		
2. calculate_Parameters			
Description	คำนวณค่าพารามิเตอร์		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	Refactoring	รูปแบบกระบวนการรีแฟคทอริง	Refactoring
	Project	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ	Project
Return Type	-		

3. ExecuteDatabase				
Description	การจัดการฐานข้อมูล			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
-	-	-	-	Private
Methods				
1. AccessReturnNoValueCommand				
Description	ฟังก์ชันจัดการกับฐานข้อมูลโดยไม่มีการส่งข้อมูลกลับคืนกลับสู่ระบบ			
Visibility	Public			
Parameters	**Name**	**Description**	**Data Type**	
	QueryString	ชุดคำสั่งฐานข้อมูล	String	
	TableName	ชื่อตาราง	String	
Return Type	-			

2. AccessReturnRowsCommand			
Description	ฟังก์ชันจัดการกับฐานข้อมูลโดยมีการส่งข้อมูลกลับคืนกลับสู่ระบบ		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	QueryString	ชุดคำสั่งฐานข้อมูล	String
	TableName	ชื่อตาราง	String
Return Type	Parameters		

4. Refactorings				
Description	รูปแบบกระบวนการรีแฟคทอริง			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
RefactoringID	รหัสกระบวนการรีแฟคทอริง	Number	-	Private
RefactoringName	ชื่อกระบวนการรีแฟคทอริง	String	-	Private
Methods				
1. selectRefactorings				

Description	ฟังก์ชันการเลือกรูปแบบกระบวนการรีแฟคทอริง		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	RefactoringID	รหัสกระบวนการรีแฟคทอริง	Number
Return Type	-		

5. Projects				
Description	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ			
Superclass	-			
Subclass	-			
Attributes				
Name	**Description**	**Data Type**	**Initial Value**	**Visibility**
ProjectsID	รหัสโครงงาน	Number	-	Private
ProjectsName	ชื่อโครงงาน	String	-	Private
FileName	ชื่อเอกสารในโครงงาน	String	-	Private
FileType	ชนิดเอกสารในโครงงาน	String	-	Private
Methods				
1. selectFiles				

Description	ฟังก์ชันการเลือกเอกสารในโครงงาน		
Visibility	Public		
Parameters	**Name**	**Description**	**Data Type**
	FileType	ชนิดเอกสารในโครงงาน	String
Return Type	Projects		

6. PrepareData	
Description	การจัดเตรียมข้อมูลก่อนคำนวณค่ามาตรวัดเชิงวัตถุ
Superclass	-
Subclass	-

Attributes

Name	Description	Data Type	Initial Value	Visibility
-	-	-	-	Private

Methods

1. splitSourceCode	
Description	ฟังก์ชันแยกซอร์สโค้ดของเอกสารในโครงงาน
Visibility	Public

	Name	Description	Data Type
Parameters	RefactoringID	รูปแบบกระบวนการรีแฟคทอริง	Refactoring
	Project	โครงงานที่นำมาคำนวณค่ามาตรวัดเชิงวัตถุ	Project
Return Type	-		

Association	
1. of	
Description	มาตรวัดเชิงวัตถุของแต่ละรูปแบบกระบวนการรีแฟคทอริง
Association Type	Association
From - To	Metrics - Refactoring
Cardinality	1..* : 1..*
2. of	
Description	มาตรวัดเชิงวัตถุของแต่ละโครงงาน
Association Type	Association
From - To	Metrics - Projects
Cardinality	1..* : 1..*
3. of	
Description	การจัดเตรียมข้อมูลก่อนคำนวณค่ามาตรวัดเชิงวัตถุของแต่ละโครงงาน

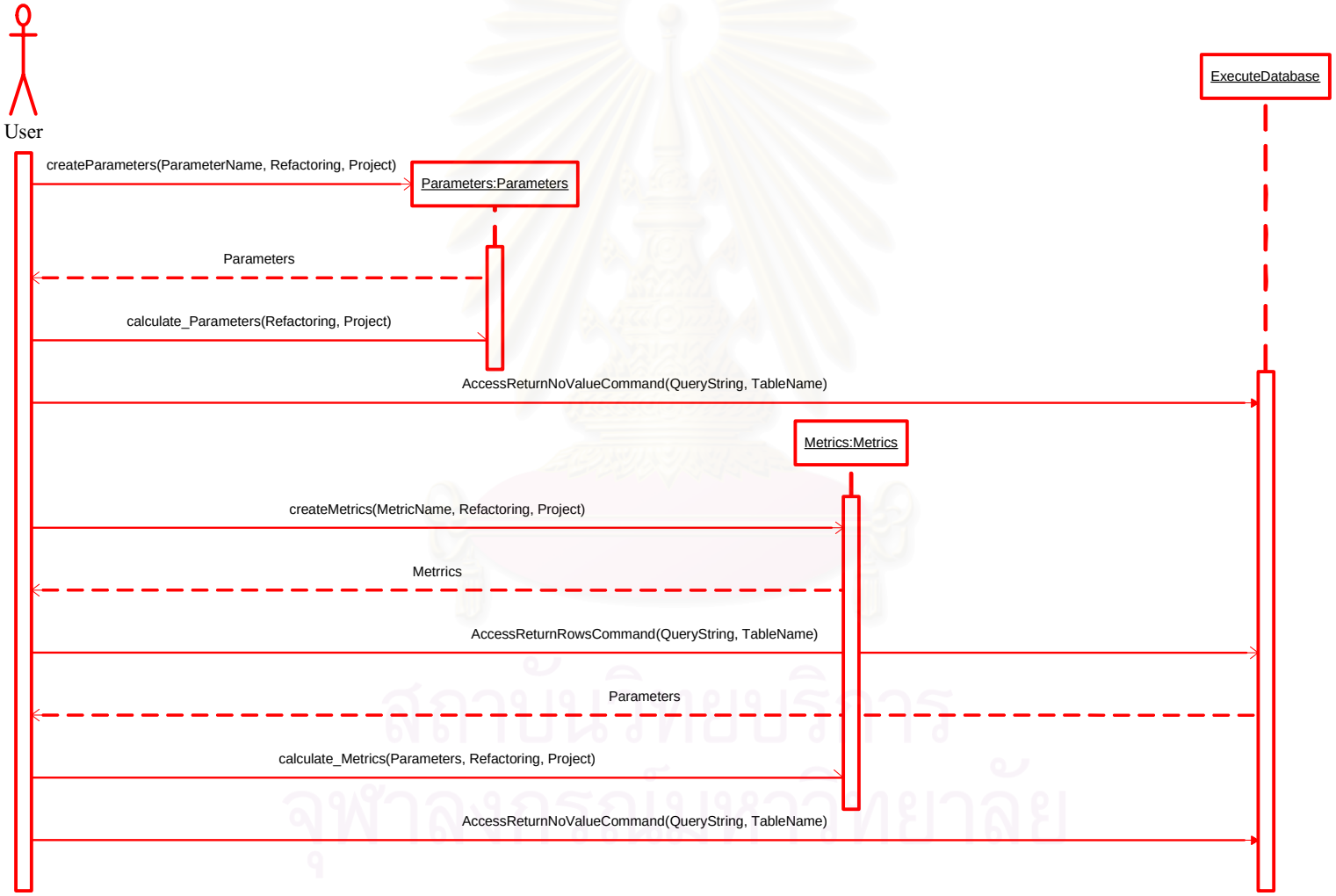
Association Type	Association
From - To	Prepare Data - Projects
Cardinality	1..* : 1..*

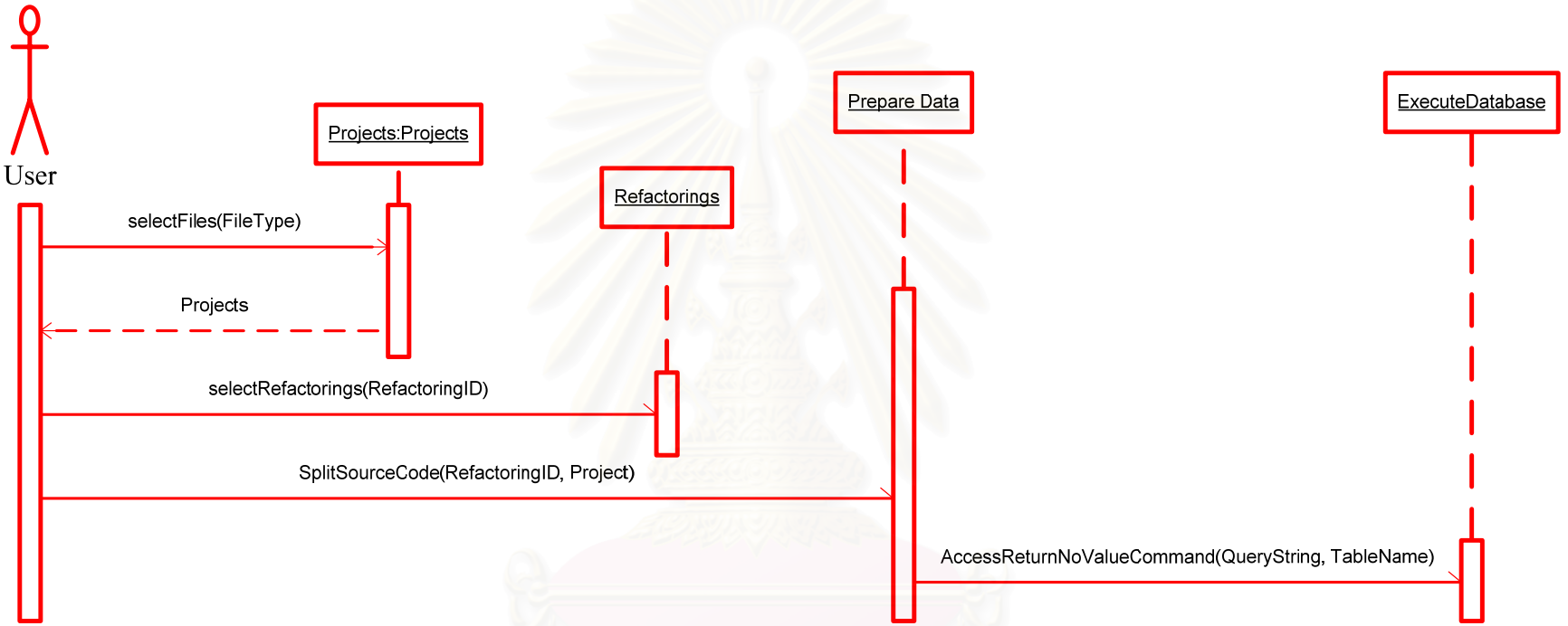
4. consist of	
Description	มาตรวัดเชิงวัตถุประกอบไปด้วยพารามิเตอร์
Association Type	Association
From - To	Metrics - Parameters
Cardinality	1..* : 1..*
5. use	
Description	มาตรวัดเชิงวัตถุเรียกใช้งานฐานข้อมูล
Association Type	Association
From - To	Metrics – Execute Database
Cardinality	1..* : 1..*
6. use	
Description	พารามิเตอร์เรียกใช้งานฐานข้อมูล
Association Type	Association
From - To	Parameters – Execute Database

Cardinality	1..* : 1..*

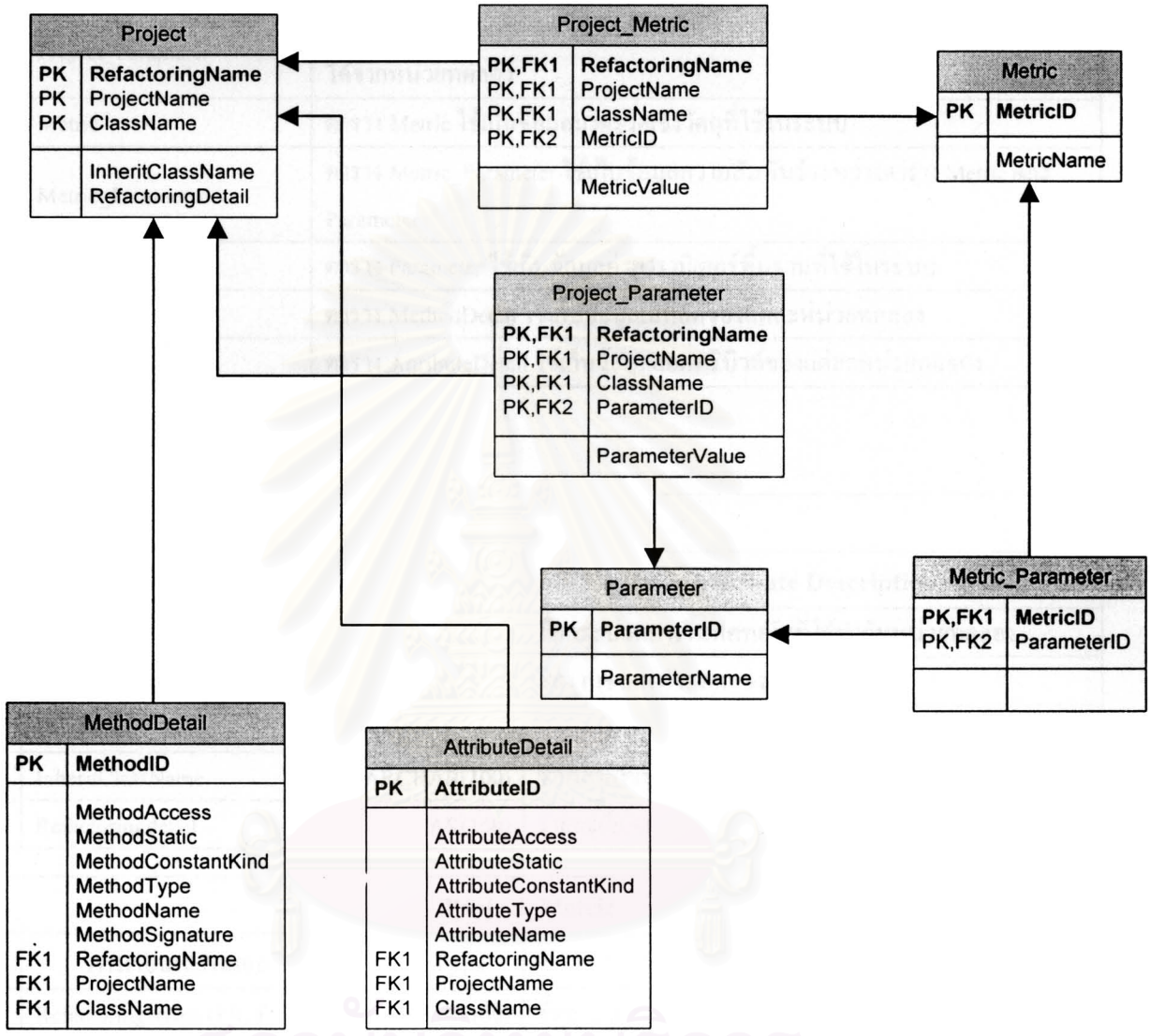
7. use	
Description	การจัดเตรียมข้อมูลก่อนคำนวณค่ามาตรวัดเชิงวัตถุเรียกใช้งานฐานข้อมูล
Association Type	Association
From - To	Prepare Data – Execute Database
Cardinality	1..* : 1..*

1.4 แผนภาพซีเควนซ์ (Sequence Diagram)





1.1 แผนภาพอีอาร์ (ER Diagram)



พจนานุกรมข้อมูล (Data Dictionary)

1. คำอธิบายตารางของระบบ

No.	Table Name	Table Description
1	Project	ตาราง Project ใช้เก็บข้อมูลของหน่วยทดลองและกระบวนการรีแฟคทอริงที่ทำกับหน่วยทดลอง
2	Project_Metric	ตาราง Project_Metric ใช้เก็บข้อมูลค่ามาตรวัดเชิงวัตถุที่คำนวณได้จากหน่วยทดลอง

No.	Table Name	Table Description
3	Project_Parameter	ตาราง Project_Parameter ใช้เก็บข้อมูลค่าพารามิเตอร์พื้นฐานของระบบที่คำนวณได้จากหน่วยทดลอง
4	Metric	ตาราง Metric ใช้เก็บข้อมูลมาตรวัดเชิงวัตถุที่ใช้ในระบบ
5	Metric_Parameter	ตาราง Metric_Parameter ใช้เก็บข้อมูลความสัมพันธ์ระหว่างตาราง Metric และ Parameter
6	Parameter	ตาราง Parameter ใช้เก็บข้อมูลค่าพารามิเตอร์พื้นฐานที่ใช้ในระบบ
7	MethodDetail	ตาราง MethodDetail ใช้เก็บข้อมูลเมท็อดของแต่ละหน่วยทดลอง
8	AttributeDetail	ตาราง AttributeDetail ใช้เก็บข้อมูลแอตทริบิวต์ของแต่ละหน่วยทดลอง

2. คำอธิบายแอตทริบิวต์ของระบบ

Project			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	RefactoringName (PK)	VARCHAR(100)	ชื่อกระบวนการรีแฟคทอริงที่ใช้ทำกับหน่วยทดลอง
2	ProjectName (PK)	VARCHAR(100)	ชื่องานจากหน่วยทดลอง
3	ClassName (PK)	VARCHAR(100)	ชื่อคลาสในระบบ
4	InheritClassName	VARCHAR(100)	ชื่อคลาสที่เป็นคลาสรูปแบบอินเฮอริแทน
5	Refactoringdetail	VARCHAR(100)	รายละเอียดกระบวนการรีแฟคทอริง

Project_Metric			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	RefactoringName (PK, FK1)	VARCHAR(100)	ชื่อกระบวนการรีแฟคทอริงที่ใช้ทำกับหน่วยทดลอง
2	ProjectName (PK, FK1)	VARCHAR(100)	ชื่องานจากหน่วยทดลอง
3	ClassName (PK, FK1)	VARCHAR(100)	ชื่อคลาสในระบบ
4	MetricID (PK, FK2)	COUNTER	รหัสมาตรวัดวัตถุเชิงวัตถุที่คำนวณในระบบ
5	MetricValue	REAL	ค่ามาตรวัดเชิงวัตถุที่คำนวณได้ในแต่ละหน่วยทดลอง

Project_Parameter			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	RefactoringName (PK, FK1)	VARCHAR(100)	ชื่อกระบวนการรีแฟคทอริงที่ใช้ทำกับหน่วยทดลอง
2	ProjectName (PK, FK1)	VARCHAR(100)	ชื่องานจากหน่วยทดลอง
3	ClassName (PK, FK1)	VARCHAR(100)	ชื่อคลาสในระบบ

Project_Parameter			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
4	ParameterID (PK, FK2)	COUNTER	รหัสพารามิเตอร์พื้นฐานที่คำนวณในระบบ
5	ParameterValue	REAL	ค่าพารามิเตอร์พื้นฐานที่คำนวณได้ในแต่ละหน่วยทดลอง

Metric			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	MetricID (PK)	COUNTER	รหัสมาตรวัดคุณเชิงวัตถุที่คำนวณในระบบ
2	MetricName	VARCHAR(100)	ชื่อมาตรวัดเชิงวัตถุที่คำนวณในระบบ

Metric_Parameter			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	MetricID (PK, FK1)	COUNTER	รหัสมาตรวัดคุณเชิงวัตถุที่คำนวณในระบบ
2	ParameterID (PK, FK2)	COUNTER	รหัสพารามิเตอร์พื้นฐานที่คำนวณในระบบ

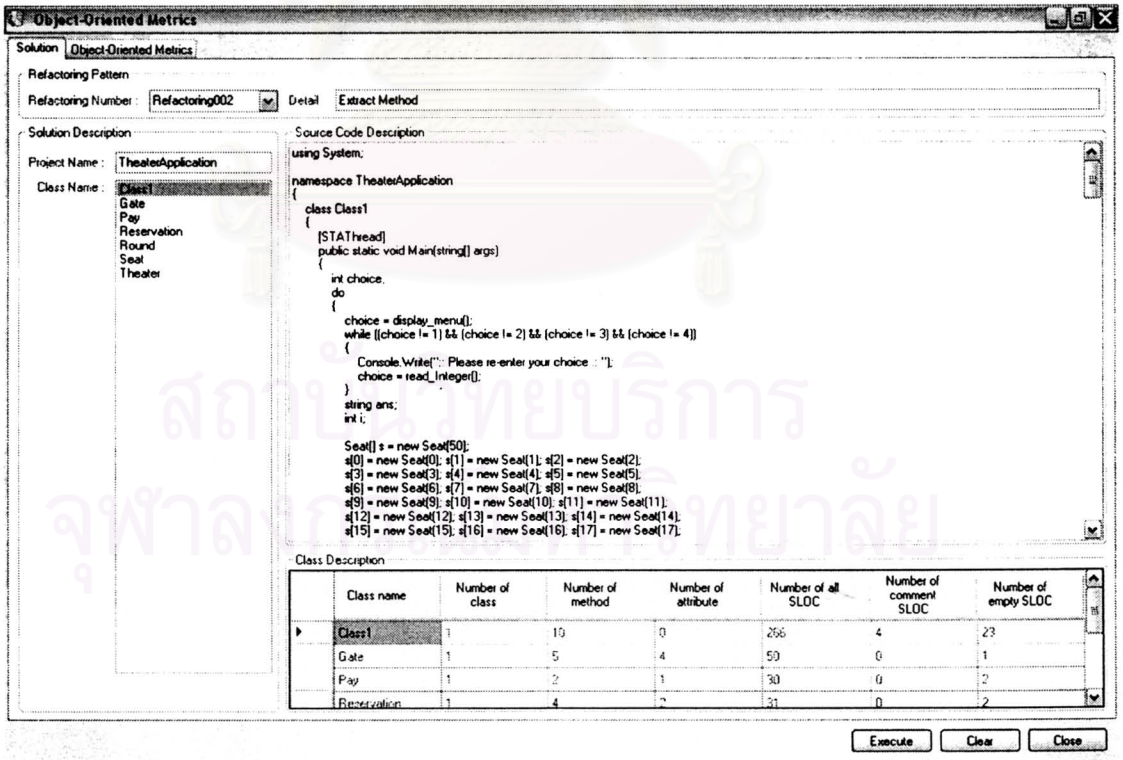
Parameter			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	ParameterID (PK)	COUNTER	รหัสพารามิเตอร์พื้นฐานที่คำนวณในระบบ
2	ParameterName	VARCHAR(100)	ชื่อพารามิเตอร์พื้นฐานที่คำนวณในระบบ

MethodDetail			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	MethodID (PK)	COUNTER	รหัสเมท็อดที่อยู่ในแต่ละคลาส
2	MethodAccess	VARCHAR(100)	คุณสมบัติการเข้าถึงของเมท็อดในแต่ละคลาส
3	MethodStatic	VARCHAR(100)	คุณสมบัติสถิตของเมท็อดในแต่ละคลาส
4	MethodConstantKind	VARCHAR(100)	คุณสมบัติค่าคงที่ของเมท็อดในแต่ละคลาส
5	MethodType	VARCHAR(100)	ชนิดของเมท็อดในแต่ละคลาส
6	MethodName	VARCHAR(100)	ชื่อเมท็อดในแต่ละคลาส
7	MethodSignature	VARCHAR(100)	คุณลักษณะของเมท็อดในแต่ละคลาส
8	RefactoringName (FK1)	VARCHAR(100)	ชื่อกระบวนการรีแฟคทอริงที่ใช้ทำกับหน่วยทดลอง
9	ProjectName (FK1)	VARCHAR(100)	ชื่องานจากหน่วยทดลอง
10	ClassName (FK1)	VARCHAR(100)	ชื่อคลาสในระบบ

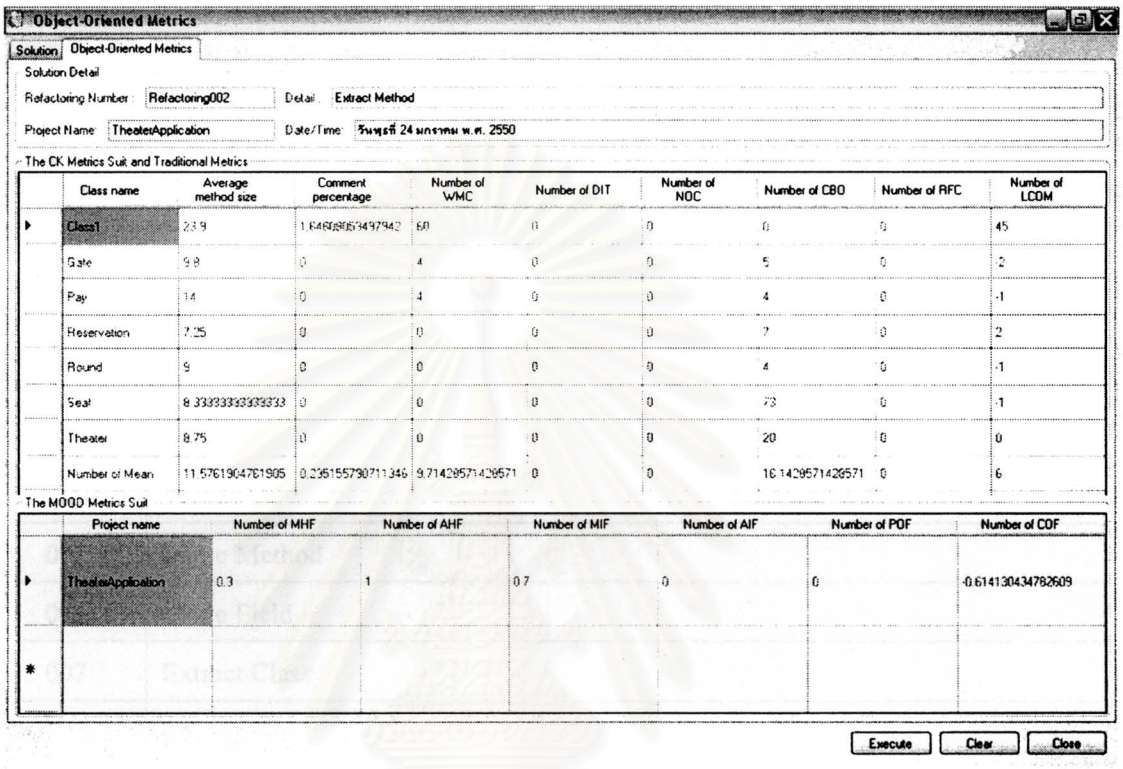
AttributeDetail			
No.	**Attribute Name**	**Data Type**	**Attribute Description**
1	AttributeID	COUNTER	รหัสแอตทริบิวต์ที่อยู่ในแต่ละคลาส
2	AttributeAccess	VARCHAR(100)	คุณสมบัติการเข้าถึงของแอตทริบิวต์ในแต่ละคลาส
3	AttributeStatic	VARCHAR(100)	คุณสมบัติสถิตของแอตทริบิวต์ในแต่ละคลาส
4	AttributeConstantKind	VARCHAR(100)	คุณสมบัติค่าคงที่ของแอตทริบิวต์ในแต่ละคลาส
5	AttributeType	VARCHAR(100)	ชนิดของแอตทริบิวต์ในแต่ละคลาส
6	AttributeName	VARCHAR(100)	ชื่อแอตทริบิวต์ในแต่ละคลาส
7	RefactoringName (FK1)	VARCHAR(100)	ชื่อกระบวนการรีแฟคทอริงที่ใช้ทำกับหน่วยทดลอง
8	ProjectName (FK1)	VARCHAR(100)	ชื่องานจากหน่วยทดลอง
9	ClassName (FK1)	VARCHAR(100)	ชื่อคลาสในระบบ

2. แผนภาพหน้าจอซอฟต์แวร์มาตรวัดเชิงวัตถุ

แสดงหน้าจอการทำงานของซอฟต์แวร์ที่พัฒนาขึ้นในงานวิจัยนี้ ซึ่งหน้าที่ 1 คือการบอกรายละเอียดของหน่วยทดลองที่เข้ามาคำนวณค่ามาตรวัดเชิงวัตถุ



แสดงหน้าจอการทำงานของซอฟต์แวร์ที่พัฒนาขึ้นในงานวิจัยนี้ ซึ่งหน้าที่ 2 คือการบอกรายละเอียดของค่ามาตรวัดเชิงวัตถุที่คำนวณจากหน่วยทดลอง



สามารถอ่านรายละเอียดซอร์สโค้ดและรายละเอียดเอกสารการออกแบบในรูปแบบสำเนาอิเล็กทรอนิกส์ (Softcopy) ได้จากซีดีรอม (CD-ROM) ที่แนบมาพร้อมกับงานวิจัยนี้

ภาคผนวก ข
เอกสารรายชื่อรูปแบบการทำกระบวนการรีแฟคทอริง

1. รูปแบบกระบวนการรีแฟคทอริงตามวัตถุประสงค์ที่ 1 คือ การเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดก่อนทำกระบวนการรีแฟคทอริงกับซอร์สโค้ดหลังทำกระบวนการรีแฟคทอริงแต่ละวิธี

รหัส	กระบวนการรีแฟคทอริง
001	No Refactoring
002	Extract Method
003	Replace Temp with Query
004	Introduce Explaining Variable
005	Move Method
006	Move Field
007	Extract Class
008	Replace Magic Number with Symbolic Constant
009	Decompose Condition
010	Rename Method
011	Pull Up Method

2. รูปแบบกระบวนการรีแฟคทอริงตามวัตถุประสงค์ที่ 2 คือ การเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x กับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y

รหัส	กระบวนการรีแฟคทอริง
012	Extract Method + Replace Temp with Query
013	Extract Method + Introduce Explaining Variable
014	Extract Method + Move Method
015	Extract Method + Move Field
016	Extract Method + Extract Class
017	Extract Method + Replace Magic Number with Symbolic Constant

รหัส	กระบวนการรีแฟคทอริง
018	Extract Method + Decompose Condition
019	Extract Method + Rename Method
020	Extract Method + Pull Up Method
021	Replace Temp with Query + Introduce Explaining Variable
022	Replace Temp with Query + Move Method
023	Replace Temp with Query + Move Field
024	Replace Temp with Query + Extract Class
025	Replace Temp with Query + Replace Magic Number with Symbolic Constant
026	Replace Temp with Query + Decompose Condition
027	Replace Temp with Query + Rename Method
028	Replace Temp with Query + Pull Up Method
029	Introduce Explaining Variable + Move Method
030	Introduce Explaining Variable + Move Field
031	Introduce Explaining Variable + Extract Class
032	Introduce Explaining Variable + Replace Magic Number with Symbolic Constant
033	Introduce Explaining Variable + Decompose Condition
034	Introduce Explaining Variable + Rename Method
035	Introduce Explaining Variable + Pull Up Method
036	Move Method + Move Field
037	Move Method + Extract Class
038	Move Method + Replace Magic Number with Symbolic Constant
039	Move Method + Decompose Condition
040	Move Method + Rename Method
041	Move Method + Pull Up Method
042	Move Field + Extract Class
043	Move Field + Replace Magic Number with Symbolic Constant
044	Move Field + Decompose Condition
045	Move Field + Rename Method

รหัส	กระบวนการรีแฟคทอริง
046	Move Field + Pull Up Method
047	Extract Class + Replace Magic Number with Symbolic Constant
048	Extract Class + Decompose Condition
049	Extract Class + Rename Method
050	Extract Class + Pull Up Method
051	Replace Magic Number with Symbolic Constant + Decompose Condition
052	Replace Magic Number with Symbolic Constant + Rename Method
053	Replace Magic Number with Symbolic Constant + Pull Up Method
054	Decompose Condition + Rename Method
055	Decompose Condition + Pull Up Method
056	Rename Method + Pull Up Method

3. รูปแบบกระบวนการรีแฟคทอริงตามวัตถุประสงค์ที่ 3 คือ การเปรียบเทียบคุณภาพซอฟต์แวร์ของซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y กับซอร์สโค้ดที่ผ่านกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x

รหัส	กระบวนการรีแฟคทอริง
057	Replace Temp with Query + Extract Method
058	Introduce Explaining Variable + Extract Method
059	Move Method + Extract Method
060	Move Field + Extract Method
061	Extract Class + Extract Method
062	Replace Magic Number with Symbolic Constant + Extract Method
063	Decompose Condition + Extract Method
064	Rename Method + Extract Method
065	Pull Up Method + Extract Method
066	Introduce Explaining Variable + Replace Temp with Query
067	Move Method + Replace Temp with Query

รหัส	กระบวนการรีแฟคทอริง
068	Move Field + Replace Temp with Query
069	Extract Class + Replace Temp with Query
070	Replace Magic Number with Symbolic Constant + Replace Temp with Query
071	Decompose Condition + Replace Temp with Query
072	Rename Method + Replace Temp with Query
073	Pull Up Method + Replace Temp with Query
074	Move Method + Introduce Explaining Variable
075	Move Field + Introduce Explaining Variable
076	Extract Class + Introduce Explaining Variable
077	Replace Magic Number with Symbolic Constant + Introduce Explaining Variable
078	Decompose Condition + Introduce Explaining Variable
079	Rename Method + Introduce Explaining Variable
080	Pull Up Method + Introduce Explaining Variable
081	Move Field + Move Method
082	Extract Class + Move Method
083	Replace Magic Number with Symbolic Constant + Move Method
084	Decompose Condition + Move Method
085	Rename Method + Move Method
086	Pull Up Method + Move Method
087	Extract Class + Move Field
088	Replace Magic Number with Symbolic Constant + Move Field
089	Decompose Condition + Move Field
090	Rename Method + Move Field
091	Pull Up Method + Move Field
092	Replace Magic Number with Symbolic Constant + Extract Class
093	Decompose Condition + Extract Class
094	Rename Method + Extract Class
095	Pull Up Method + Extract Class

รหัส	กระบวนการรีแฟคทอริง
096	Decompose Condition + Replace Magic Number with Symbolic Constant
097	Rename Method + Replace Magic Number with Symbolic Constant
098	Pull Up Method + Replace Magic Number with Symbolic Constant
099	Rename Method + Decompose Condition
100	Pull Up Method + Decompose Condition
101	Pull Up Method + Rename Method

ภาคผนวก ค

ผลการวิเคราะห์ข้อมูล

1. การทดสอบการแจกแจงของข้อมูล

จากการทดสอบการแจกแจงข้อมูลสามารถแสดงรายละเอียดของข้อมูลได้ดังนี้

1. ข้อมูลสำหรับการทดสอบสมมติฐานที่ 1: การเปรียบเทียบค่ามาตรวัดเชิงวัตถุก่อนทำกระบวนการรีแฟคทอริงเปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริง

1.1 Extract Method - No Refactoring

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.279	26	.000	.767	26	.000
WMC	.399	26	.000	.613	26	.000
CBO	.367	26	.000	.570	26	.000
RFC	.539	26	.000	.198	26	.000
LCOM	.326	26	.000	.655	26	.000
AverageMethodSize	.241	26	.000	.658	26	.000
MethodHidingFactor	.117	26	.200(*)	.965	26	.492
AttributeHidingFactor	.539	26	.000	.198	26	.000
MethodInheritanceFactor	.096	26	.200(*)	.966	26	.521
AttributeInheritanceFactor	.524	26	.000	.326	26	.000
PolymorphismFactor	.539	26	.000	.198	26	.000
CouplingFactor	.244	26	.000	.860	26	.002

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.

1.2 Replace Temp with Query - No Refactoring

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.231	12	.075	.819	12	.016
CBO	.530	12	.000	.327	12	.000
LCOM	.277	12	.011	.756	12	.003
AverageMethodSize	.268	12	.017	.795	12	.008
MethodHidingFactor	.134	12	.200(*)	.975	12	.959
MethodInheritanceFactor	.187	12	.200(*)	.944	12	.552
AttributeInheritanceFactor	.530	12	.000	.327	12	.000
CouplingFactor	.530	12	.000	.327	12	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f AttributeHidingFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.3 Introduce Explaining Variable - No Refactoring

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.349	15	.000	.544	15	.000
WMC	.514	15	.000	.424	15	.000
CBO	.535	15	.000	.284	15	.000
LCOM	.384	15	.000	.676	15	.000
AverageMethodSize	.242	15	.018	.788	15	.003
AttributeHidingFactor	.526	15	.000	.329	15	.000
AttributeInheritanceFactor	.438	15	.000	.530	15	.000
CouplingFactor	.535	15	.000	.284	15	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e MethodHidingFactor is constant. It has been omitted.
f MethodInheritanceFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.4 Move Method - No Refactoring

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.192	23	.028	.914	23	.050
WMC	.265	23	.000	.834	23	.001
DIT	.169	23	.087	.921	23	.071
NOC	.167	23	.097	.932	23	.124
CBO	.188	23	.035	.927	23	.094
RFC	.200	23	.018	.840	23	.002
LCOM	.321	23	.000	.536	23	.000
AverageMethodSize	.293	23	.000	.619	23	.000
MethodHidingFactor	.519	23	.000	.303	23	.000
AttributeHidingFactor	.474	23	.000	.295	23	.000
MethodInheritanceFactor	.367	23	.000	.510	23	.000
AttributeInheritanceFactor	.457	23	.000	.352	23	.000
PolymorphismFactor	.493	23	.000	.471	23	.000
CouplingFactor	.338	23	.000	.645	23	.000

a Lilliefors Significance Correction

1.5 Move Field - No Refactoring

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.173	21	.102	.923	21	.099
WMC	.267	21	.000	.834	21	.002
DIT	.171	21	.112	.941	21	.233
NOC	.173	21	.100	.955	21	.428
CBO	.241	21	.002	.890	21	.022
RFC	.197	21	.032	.857	21	.006
LCOM	.247	21	.002	.882	21	.016
AverageMethodSize	.253	21	.001	.776	21	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MethodHidingFactor	.505	21	.000	.283	21	.000
AttributeHidingFactor	.462	21	.000	.316	21	.000
MethodInheritanceFactor	.375	21	.000	.493	21	.000
AttributeInheritanceFactor	.459	21	.000	.353	21	.000
PolymorphismFactor	.478	21	.000	.417	21	.000
CouplingFactor	.336	21	.000	.544	21	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction

1.6 Extract Class - No Refactoring

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.208	22	.014	.860	22	.005
WMC	.280	22	.000	.816	22	.001
DIT	.171	22	.095	.931	22	.128
NOC	.170	22	.098	.944	22	.237
CBO	.261	22	.000	.730	22	.000
RFC	.199	22	.024	.849	22	.003
LCOM	.289	22	.000	.756	22	.000
AverageMethodSize	.269	22	.000	.728	22	.000
MethodHidingFactor	.539	22	.000	.221	22	.000
AttributeHidingFactor	.470	22	.000	.303	22	.000
MethodInheritanceFactor	.499	22	.000	.367	22	.000
AttributeInheritanceFactor	.472	22	.000	.346	22	.000
PolymorphismFactor	.480	22	.000	.405	22	.000
CouplingFactor	.366	22	.000	.450	22	.000

a Lilliefors Significance Correction

1.7 Replace Magic Number with Symbolic Constant - No Refactoring

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.336	19	.000	.590	19	.000
CBO	.334	19	.000	.768	19	.000
LCOM	.452	19	.000	.552	19	.000
AverageMethodSize	.217	19	.019	.784	19	.001
AttributeHidingFactor	.323	19	.000	.536	19	.000
AttributeInheritanceFactor	.325	19	.000	.587	19	.000
CouplingFactor	.448	19	.000	.457	19	.000

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f MethodHidingFactor is constant. It has been omitted.
g MethodInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.8 Decompose Conditional - No Refactoring

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.326	22	.000	.594	22	.000
CBO	.323	22	.000	.770	22	.000
LCOM	.414	22	.000	.564	22	.000
AverageMethodSize	.227	22	.004	.769	22	.000
AttributeHidingFactor	.314	22	.000	.563	22	.000
AttributeInheritanceFactor	.326	22	.000	.621	22	.000
CouplingFactor	.445	22	.000	.502	22	.000

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f MethodHidingFactor is constant. It has been omitted.
g MethodInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.9 Rename Method - No Refactoring

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.504	32	.000	.401	32	.000
WMC	.454	32	.000	.413	32	.000
CBO	.364	32	.000	.702	32	.000
RFC	.539	32	.000	.172	32	.000
LCOM	.399	32	.000	.417	32	.000
AverageMethodSize	.495	32	.000	.452	32	.000
AttributeHidingFactor	.521	32	.000	.262	32	.000
MethodInheritanceFactor	.529	32	.000	.252	32	.000
AttributeInheritanceFactor	.448	32	.000	.375	32	.000
PolymorphismFactor	.539	32	.000	.172	32	.000
CouplingFactor	.337	32	.000	.600	32	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d MethodHidingFactor is constant. It has been omitted.

2. ข้อมูลสำหรับการทดสอบสมมติฐานที่ 2: การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y

1.10 (Extract Method + Replace Temp with Query) – Extract Method

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.187	9	.200(*)	.852	9	.078
CBO	.519	9	.000	.390	9	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
LCOM	.350	9	.002	.695	9	.001
AverageMethodSize	.282	9	.038	.801	9	.021
MethodHidingFactor	.163	9	.200(*)	.963	9	.834
MethodInheritanceFactor	.280	9	.041	.869	9	.121
AttributeInheritanceFactor	.497	9	.000	.402	9	.000
CouplingFactor	.519	9	.000	.390	9	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f AttributeHidingFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.11 (Extract Method + Introduce Explaining Variable) - Extract Method

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CyclomaticComplexity	.532	13	.000	.311	13	.000
CommentPercentage	.394	13	.000	.597	13	.000
WMC	.532	13	.000	.311	13	.000
CBO	.532	13	.000	.311	13	.000
LCOM	.389	13	.000	.677	13	.000
AverageMethodSize	.292	13	.003	.566	13	.000
AttributeHidingFactor	.462	13	.000	.364	13	.000
AttributeInheritanceFactor	.453	13	.000	.588	13	.000
CouplingFactor	.532	13	.000	.311	13	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e MethodHidingFactor is constant. It has been omitted.
f MethodInheritanceFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.12 (Extract Method + Move Method) - Extract Method

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CyclomaticComplexity	.283	22	.000	.812	22	.001
CommentPercentage	.165	22	.121	.935	22	.159
WMC	.283	22	.000	.812	22	.001
DIT	.171	22	.095	.931	22	.128
NOC	.170	22	.098	.944	22	.237
CBO	.290	22	.000	.765	22	.000
RFC	.199	22	.024	.849	22	.003
LCOM	.301	22	.000	.757	22	.000
AverageMethodSize	.195	22	.029	.796	22	.000
MethodHidingFactor	.134	22	.200(*)	.939	22	.192
AttributeHidingFactor	.510	22	.000	.336	22	.000
MethodInheritanceFactor	.134	22	.200(*)	.939	22	.192

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AttributeInheritanceFactor	.448	22	.000	.499	22	.000
PolymorphismFactor	.539	22	.000	.221	22	.000
CouplingFactor	.284	22	.000	.695	22	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction

1.13 (Extract Method + Move Field) - Extract Method

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CyclomaticComplexity	.261	21	.001	.848	21	.004
CommentPercentage	.171	21	.111	.912	21	.059
WMC	.261	21	.001	.848	21	.004
DIT	.171	21	.112	.941	21	.233
NOC	.173	21	.100	.955	21	.428
CBO	.239	21	.003	.783	21	.000
RFC	.197	21	.032	.857	21	.006
LCOM	.290	21	.000	.710	21	.000
AverageMethodSize	.261	21	.001	.823	21	.002
MethodHidingFactor	.104	21	.200(*)	.940	21	.215
AttributeHidingFactor	.496	21	.000	.269	21	.000
MethodInheritanceFactor	.104	21	.200(*)	.939	21	.213
AttributeInheritanceFactor	.496	21	.000	.477	21	.000
CouplingFactor	.327	21	.000	.579	21	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.14 (Extract Method + Extract Class) - Extract Method

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CyclomaticComplexity	.283	22	.000	.812	22	.001
CommentPercentage	.213	22	.011	.861	22	.005
WMC	.283	22	.000	.812	22	.001
DIT	.171	22	.095	.931	22	.128
NOC	.170	22	.098	.944	22	.237
CBO	.290	22	.000	.707	22	.000
RFC	.199	22	.024	.849	22	.003
LCOM	.324	22	.000	.778	22	.000
AverageMethodSize	.151	22	.200(*)	.913	22	.055
AttributeHidingFactor	.498	22	.000	.261	22	.000
AttributeInheritanceFactor	.499	22	.000	.465	22	.000
CouplingFactor	.362	22	.000	.462	22	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c MethodInheritanceFactor is constant. It has been omitted.
d PolymorphismFactor is constant. It has been omitted.

1.15 (Extract Method + Replace Magic Number with Symbolic Constant) - Extract Method

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.361	19	.000	.384	19	.000
CBO	.388	19	.000	.621	19	.000
LCOM	.440	19	.000	.545	19	.000
AverageMethodSize	.283	19	.000	.829	19	.003
AttributeHidingFactor	.329	19	.000	.501	19	.000
AttributeInheritanceFactor	.314	19	.000	.646	19	.000
CouplingFactor	.448	19	.000	.417	19	.000

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f MethodHidingFactor is constant. It has been omitted.
g MethodInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.16 (Extract Method + Decompose Condition) -Extract Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.270	21	.000	.773	21	.000
WMC	.308	21	.000	.765	21	.000
CBO	.252	21	.001	.787	21	.000
LCOM	.369	21	.000	.601	21	.000
AverageMethodSize	.215	21	.013	.760	21	.000
MethodHidingFactor	.117	21	.200(*)	.912	21	.060
AttributeHidingFactor	.496	21	.000	.269	21	.000
MethodInheritanceFactor	.144	21	.200(*)	.944	21	.259
AttributeInheritanceFactor	.526	21	.000	.360	21	.000
CouplingFactor	.240	21	.003	.635	21	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.17 (Extract Method + Rename Method) - Extract Method

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.454	26	.000	.357	26	.000
WMC	.445	26	.000	.468	26	.000
CBO	.363	26	.000	.727	26	.000
LCOM	.412	26	.000	.507	26	.000
AverageMethodSize	.482	26	.000	.407	26	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MethodHidingFactor	.512	26	.000	.379	26	.000
AttributeHidingFactor	.443	26	.000	.397	26	.000
MethodInheritanceFactor	.497	26	.000	.453	26	.000
AttributeInheritanceFactor	.434	26	.000	.377	26	.000
PolymorphismFactor	.539	26	.000	.198	26	.000
CouplingFactor	.350	26	.000	.621	26	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.

1.18 (Replace Temp with Query + Extract Method) - Replace Temp with Query

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.390	9	.000	.645	9	.000
WMC	.256	9	.091	.925	9	.439
CBO	.386	9	.000	.642	9	.000
LCOM	.248	9	.117	.846	9	.067
AverageMethodSize	.278	9	.044	.754	9	.006
MethodHidingFactor	.143	9	.200(*)	.972	9	.913
AttributeHidingFactor	.519	9	.000	.390	9	.000
MethodInheritanceFactor	.209	9	.200(*)	.923	9	.417
AttributeInheritanceFactor	.519	9	.000	.390	9	.000
PolymorphismFactor	.519	9	.000	.390	9	.000
CouplingFactor	.338	9	.004	.786	9	.014

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.

1.19 (Replace Temp with Query + Introduce Explaining Variable) - Replace Temp with Query

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.412	4	.	.685	4	.008
LCOM	.441	4	.	.630	4	.001
AverageMethodSize	.277	4	.	.889	4	.378

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e CBO is constant. It has been omitted.
f RFC is constant. It has been omitted.
g MethodHidingFactor is constant. It has been omitted.
h AttributeHidingFactor is constant. It has been omitted.
i MethodInheritanceFactor is constant. It has been omitted.
j AttributeInheritanceFactor is constant. It has been omitted.
k PolymorphismFactor is constant. It has been omitted.
l CouplingFactor is constant. It has been omitted.

1.20 (Replace Temp with Query + Move Method) - Replace Temp with Query

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.286	8	.053	.829	8	.058
WMC	.328	8	.011	.842	8	.078
DIT	.275	8	.075	.828	8	.056
NOC	.218	8	.200(*)	.904	8	.313
CBO	.233	8	.200(*)	.900	8	.291
RFC	.164	8	.200(*)	.947	8	.680
LCOM	.211	8	.200(*)	.942	8	.636
AverageMethodSize	.347	8	.005	.692	8	.002
MethodHidingFactor	.355	8	.004	.772	8	.014
AttributeHidingFactor	.513	8	.000	.418	8	.000
MethodInheritanceFactor	.166	8	.200(*)	.969	8	.887
AttributeInheritanceFactor	.416	8	.000	.473	8	.000
CouplingFactor	.299	8	.034	.715	8	.003

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.21 (Replace Temp with Query + Move Field) - Replace Temp with Query

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.295	8	.039	.840	8	.076
WMC	.328	8	.011	.842	8	.078
DIT	.275	8	.075	.828	8	.056
NOC	.218	8	.200(*)	.904	8	.313
CBO	.209	8	.200(*)	.920	8	.432
RFC	.164	8	.200(*)	.947	8	.680
LCOM	.305	8	.027	.807	8	.034
AverageMethodSize	.369	8	.002	.631	8	.000
MethodHidingFactor	.454	8	.000	.496	8	.000
MethodInheritanceFactor	.216	8	.200(*)	.943	8	.643
AttributeInheritanceFactor	.513	8	.000	.418	8	.000
CouplingFactor	.281	8	.063	.719	8	.004

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b AttributeHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.22 (Replace Temp with Query + Extract Class) - Replace Temp with Query

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.247	8	.161	.871	8	.154
WMC	.332	8	.010	.837	8	.070
DIT	.275	8	.075	.828	8	.056
NOC	.218	8	.200(*)	.904	8	.313
CBO	.255	8	.135	.761	8	.011

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
RFC	.164	8	.200(*)	.947	8	.680
LCOM	.303	8	.029	.714	8	.003
AverageMethodSize	.358	8	.003	.608	8	.000
AttributeInheritanceFactor	.513	8	.000	.418	8	.000
CouplingFactor	.388	8	.001	.643	8	.000

a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c AttributeHidingFactor is constant. It has been omitted.
d MethodInheritanceFactor is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.23 (Replace Temp with Query + Replace Magic Number with Symbolic Constant) - Replace Temp with Query

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.356	7	.008	.800	7	.041
CBO	.337	7	.016	.747	7	.012
LCOM	.504	7	.000	.453	7	.000
AverageMethodSize	.241	7	.200(*)	.900	7	.330
AttributeHidingFactor	.466	7	.000	.492	7	.000
AttributeInheritanceFactor	.252	7	.200	.737	7	.009
CouplingFactor	.380	7	.003	.557	7	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f MethodHidingFactor is constant. It has been omitted.
g MethodInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.24 (Replace Temp with Query + Decompose - Conditional) Replace Temp with Query

Tests of Normality(b,c,d,e,f,g,h,i,j,k)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.133	7	.200(*)	.990	7	.993
WMC	.280	7	.104	.808	7	.049
CBO	.344	7	.012	.777	7	.024
LCOM	.273	7	.124	.892	7	.286
AverageMethodSize	.279	7	.106	.796	7	.037
MethodHidingFactor	.190	7	.200(*)	.956	7	.787
AttributeHidingFactor	.504	7	.000	.453	7	.000
MethodInheritanceFactor	.144	7	.200(*)	.971	7	.905
AttributeInheritanceFactor	.416	7	.001	.511	7	.000
CouplingFactor	.234	7	.200(*)	.922	7	.484

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.25 (Replace Temp with Query + Rename Method) - Replace Temp with Query

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.464	12	.000	.554	12	.000
WMC	.530	12	.000	.327	12	.000
DIT	.530	12	.000	.327	12	.000
NOC	.530	12	.000	.327	12	.000
CBO	.513	12	.000	.352	12	.000
RFC	.530	12	.000	.327	12	.000
LCOM	.476	12	.000	.568	12	.000
AverageMethodSize	.371	12	.000	.649	12	.000
MethodHidingFactor	.530	12	.000	.327	12	.000
AttributeHidingFactor	.530	12	.000	.327	12	.000
MethodInheritanceFactor	.474	12	.000	.516	12	.000
AttributeInheritanceFactor	.448	12	.000	.401	12	.000
PolymorphismFactor	.530	12	.000	.327	12	.000
CouplingFactor	.489	12	.000	.386	12	.000

a Lilliefors Significance Correction

1.26 (Introduce Explaining Variable + Extract Method) - Introduce Explaining Variable

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.249	13	.027	.875	13	.061
WMC	.350	13	.000	.716	13	.001
CBO	.323	13	.001	.633	13	.000
LCOM	.254	13	.022	.700	13	.001
AverageMethodSize	.300	13	.002	.621	13	.000
MethodHidingFactor	.130	13	.200(*)	.938	13	.428
AttributeHidingFactor	.529	13	.000	.326	13	.000
MethodInheritanceFactor	.102	13	.200(*)	.953	13	.643
AttributeInheritanceFactor	.501	13	.000	.345	13	.000
CouplingFactor	.332	13	.000	.807	13	.008

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.27 (Introduce Explaining Variable + Replace Temp with Query) - Introduce Explaining Variable

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.304	4	.	.804	4	.109
CBO	.441	4	.	.630	4	.001
LCOM	.281	4	.	.941	4	.662
AverageMethodSize	.198	4	.	.972	4	.851
MethodHidingFactor	.149	4	.	.994	4	.979
MethodInheritanceFactor	.168	4	.	.990	4	.956
CouplingFactor	.441	4	.	.630	4	.001

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f AttributeHidingFactor is constant. It has been omitted.
g AttributeInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.28 (Introduce Explaining Variable + Move Method) - Introduce Explaining Variable

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.238	11	.083	.888	11	.131
WMC	.298	11	.007	.832	11	.024
DIT	.159	11	.200(*)	.930	11	.413
NOC	.253	11	.048	.879	11	.101
CBO	.244	11	.067	.898	11	.176
RFC	.205	11	.200(*)	.824	11	.020
LCOM	.335	11	.001	.609	11	.000
AverageMethodSize	.336	11	.001	.572	11	.000
AttributeHidingFactor	.516	11	.000	.398	11	.000
MethodInheritanceFactor	.307	11	.005	.738	11	.001
AttributeInheritanceFactor	.392	11	.000	.677	11	.000
CouplingFactor	.197	11	.200(*)	.898	11	.176

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.29 (Introduce Explaining Variable + Move Field) - Introduce Explaining Variable

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.281	11	.015	.865	11	.066
WMC	.260	11	.036	.838	11	.029

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
DIT	.159	11	.200(*)	.930	11	.413
NOC	.253	11	.048	.879	11	.101
CBO	.198	11	.200(*)	.935	11	.468
RFC	.205	11	.200(*)	.824	11	.020
LCOM	.151	11	.200(*)	.964	11	.819
AverageMethodSize	.337	11	.001	.665	11	.000
AttributeHidingFactor	.472	11	.000	.550	11	.000
MethodInheritanceFactor	.343	11	.001	.703	11	.001
AttributeInheritanceFactor	.446	11	.000	.601	11	.000
CouplingFactor	.168	11	.200(*)	.879	11	.100

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.30 (Introduce Explaining Variable + Extract Class) - Introduce Explaining Variable

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.308	11	.004	.747	11	.002
WMC	.281	11	.015	.796	11	.008
DIT	.306	11	.005	.738	11	.001
NOC	.274	11	.020	.753	11	.002
CBO	.402	11	.000	.650	11	.000
RFC	.355	11	.000	.586	11	.000
LCOM	.264	11	.031	.847	11	.039
AverageMethodSize	.293	11	.009	.827	11	.022
AttributeHidingFactor	.462	11	.000	.439	11	.000
MethodInheritanceFactor	.492	11	.000	.487	11	.000
AttributeInheritanceFactor	.404	11	.000	.551	11	.000
PolymorphismFactor	.528	11	.000	.345	11	.000
CouplingFactor	.377	11	.000	.671	11	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.

1.31 (Introduce Explaining Variable + Replace Magic Number with Symbolic Constant) - Introduce Explaining Variable

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.450	12	.000	.524	12	.000
WMC	.498	12	.000	.481	12	.000
DIT	.530	12	.000	.327	12	.000
NOC	.530	12	.000	.327	12	.000
CBO	.502	12	.000	.357	12	.000
RFC	.530	12	.000	.327	12	.000
LCOM	.448	12	.000	.590	12	.000
AverageMethodSize	.454	12	.000	.370	12	.000
AttributeHidingFactor	.386	12	.000	.801	12	.010
MethodInheritanceFactor	.530	12	.000	.327	12	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AttributeInheritanceFactor	.401	12	.000	.717	12	.001
CouplingFactor	.433	12	.000	.661	12	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.32 (Introduce Explaining Variable + Decompose Conditional) - Introduce Explaining Variable

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.346	11	.001	.764	11	.003
WMC	.347	11	.001	.676	11	.000
CBO	.262	11	.033	.663	11	.000
LCOM	.236	11	.087	.880	11	.106
AverageMethodSize	.366	11	.000	.612	11	.000
MethodHidingFactor	.165	11	.200(*)	.849	11	.042
AttributeHidingFactor	.528	11	.000	.345	11	.000
MethodInheritanceFactor	.197	11	.200(*)	.856	11	.052
AttributeInheritanceFactor	.528	11	.000	.345	11	.000
CouplingFactor	.152	11	.200(*)	.963	11	.807

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.33 (Introduce Explaining Variable + Rename Method) - Introduce Explaining Variable

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.441	12	.000	.588	12	.000
WMC	.496	12	.000	.487	12	.000
CBO	.259	12	.025	.833	12	.023
LCOM	.434	12	.000	.701	12	.001
AverageMethodSize	.456	12	.000	.456	12	.000
AttributeHidingFactor	.509	12	.000	.421	12	.000
AttributeInheritanceFactor	.414	12	.000	.685	12	.001
CouplingFactor	.369	12	.000	.583	12	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e MethodHidingFactor is constant. It has been omitted.
f MethodInheritanceFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.34 (Move Method + Extract Method) - Move Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.337	23	.000	.640	23	.000
WMC	.448	23	.000	.358	23	.000
CBO	.391	23	.000	.682	23	.000
LCOM	.267	23	.000	.844	23	.002
AverageMethodSize	.195	23	.023	.811	23	.001
MethodHidingFactor	.158	23	.139	.877	23	.009
AttributeHidingFactor	.504	23	.000	.353	23	.000
MethodInheritanceFactor	.203	23	.015	.883	23	.011
AttributeInheritanceFactor	.426	23	.000	.556	23	.000
CouplingFactor	.466	23	.000	.245	23	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.35 (Move Method + Replace Temp with Query) - Move Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.271	8	.085	.847	8	.089
WMC	.513	8	.000	.418	8	.000
CBO	.483	8	.000	.542	8	.000
LCOM	.164	8	.200(*)	.965	8	.853
AverageMethodSize	.421	8	.000	.580	8	.000
MethodHidingFactor	.270	8	.089	.804	8	.031
AttributeHidingFactor	.513	8	.000	.418	8	.000
MethodInheritanceFactor	.315	8	.019	.802	8	.030
AttributeInheritanceFactor	.513	8	.000	.418	8	.000
CouplingFactor	.501	8	.000	.472	8	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.36 (Move Method + Introduce Explaining Variable) - Move Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.212	11	.181	.894	11	.155
WMC	.432	11	.000	.617	11	.000
CBO	.336	11	.001	.813	11	.014
LCOM	.465	11	.000	.557	11	.000
AverageMethodSize	.307	11	.005	.595	11	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MethodHidingFactor	.528	11	.000	.345	11	.000
AttributeHidingFactor	.516	11	.000	.398	11	.000
MethodInheritanceFactor	.524	11	.000	.347	11	.000
AttributeInheritanceFactor	.414	11	.000	.627	11	.000
CouplingFactor	.459	11	.000	.484	11	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.37 (Move Method + Move Field) - Move Method

Tests of Normality(b,c,d,e,f,g,h,i,j,k)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
LCOM	.538	20	.000	.236	20	.000
AttributeHidingFactor	.538	20	.000	.236	20	.000
AttributeInheritanceFactor	.501	20	.000	.286	20	.000

a Lilliefors Significance Correction
b CommentPercentage#Minute is constant. It has been omitted.
c WMC#Minute is constant. It has been omitted.
d DIT#Minute is constant. It has been omitted.
e NOC#Minute is constant. It has been omitted.
f CBO#Minute is constant. It has been omitted.
g RFC#Minute is constant. It has been omitted.
h MethodHidingFactor is constant. It has been omitted.
i MethodInheritanceFactor is constant. It has been omitted.
j PolymorphismFactor is constant. It has been omitted.
k CouplingFactor is constant. It has been omitted.

1.38 (Move Method + Extract Class) - Move Method

Tests of Normality(b,c,d,e,f,g,h,i,j,k)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
LCOM	.525	22	.000	.284	22	.000
AverageMethodSize	.539	22	.000	.221	22	.000
AttributeHidingFactor	.524	22	.000	.356	22	.000
AttributeInheritanceFactor	.486	22	.000	.400	22	.000

a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h MethodHidingFactor is constant. It has been omitted.
i MethodInheritanceFactor is constant. It has been omitted.
j PolymorphismFactor is constant. It has been omitted.
k CouplingFactor is constant. It has been omitted.

1.39 (Move Method + Replace Magic Number with Symbolic Constant) - Move Method

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.365	17	.000	.528	17	.000
CBO	.396	17	.000	.580	17	.000
LCOM	.530	17	.000	.296	17	.000
AverageMethodSize	.202	17	.063	.923	17	.164
AttributeHidingFactor	.368	17	.000	.413	17	.000
AttributeInheritanceFactor	.277	17	.001	.695	17	.000
CouplingFactor	.465	17	.000	.310	17	.000

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f MethodHidingFactor is constant. It has been omitted.
g MethodInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.40 (Move Method + Decompose Conditional) - Move Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.220	19	.016	.848	19	.006
WMC	.280	19	.000	.840	19	.005
CBO	.305	19	.000	.647	19	.000
LCOM	.238	19	.006	.678	19	.000
AverageMethodSize	.239	19	.006	.804	19	.001
MethodHidingFactor	.086	19	.200(*)	.957	19	.507
AttributeHidingFactor	.522	19	.000	.372	19	.000
MethodInheritanceFactor	.132	19	.200(*)	.941	19	.269
AttributeInheritanceFactor	.524	19	.000	.374	19	.000
CouplingFactor	.186	19	.082	.866	19	.012

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.41 (Move Method + Rename Method) - Move Method

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.423	22	.000	.329	22	.000
WMC	.502	22	.000	.247	22	.000
DIT	.539	22	.000	.221	22	.000
NOC	.539	22	.000	.221	22	.000
CBO	.282	22	.000	.767	22	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
RFC	.539	22	.000	.221	22	.000
LCOM	.465	22	.000	.331	22	.000
AverageMethodSize	.407	22	.000	.415	22	.000
MethodHidingFactor	.502	22	.000	.361	22	.000
AttributeHidingFactor	.505	22	.000	.437	22	.000
MethodInheritanceFactor	.478	22	.000	.333	22	.000
AttributeInheritanceFactor	.439	22	.000	.355	22	.000
PolymorphismFactor	.539	22	.000	.221	22	.000
CouplingFactor	.314	22	.000	.612	22	.000

1.42 (Move Field + Extract Method) - Move Field

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.245	21	.002	.863	21	.007
WMC	.389	21	.000	.594	21	.000
CBO	.331	21	.000	.678	21	.000
LCOM	.307	21	.000	.622	21	.000
AverageMethodSize	.206	21	.020	.698	21	.000
MethodHidingFactor	.172	21	.107	.923	21	.100
AttributeHidingFactor	.499	21	.000	.257	21	.000
MethodInheritanceFactor	.138	21	.200(*)	.923	21	.100
AttributeInheritanceFactor	.529	21	.000	.345	21	.000
PolymorphismFactor	.539	21	.000	.228	21	.000
CouplingFactor	.466	21	.000	.452	21	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.

1.43 (Move Field + Replace Temp with Query) - Move Field

Tests of Normality(b,c,d,e,f,g,h,i,j,k)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.200	8	.200(*)	.918	8	.410
WMC	.513	8	.000	.418	8	.000
CBO	.430	8	.000	.678	8	.001
LCOM	.334	8	.009	.754	8	.009
AverageMethodSize	.316	8	.018	.778	8	.017
MethodHidingFactor	.284	8	.056	.804	8	.031
MethodInheritanceFactor	.279	8	.067	.843	8	.080
CouplingFactor	.391	8	.001	.544	8	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
i AttributeHidingFactor is constant. It has been omitted.
j AttributeInheritanceFactor is constant. It has been omitted.
k PolymorphismFactor is constant. It has been omitted.

1.44 (Move Field + Introduce Explaining Variable) - Move Field

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.196	10	.200(*)	.948	10	.648
WMC	.370	10	.000	.756	10	.004
CBO	.410	10	.000	.477	10	.000
LCOM	.307	10	.008	.879	10	.126
AverageMethodSize	.355	10	.001	.771	10	.006
MethodHidingFactor	.524	10	.000	.366	10	.000
AttributeHidingFactor	.431	10	.000	.418	10	.000
MethodInheritanceFactor	.367	10	.000	.767	10	.006
AttributeInheritanceFactor	.395	10	.000	.561	10	.000
CouplingFactor	.393	10	.000	.746	10	.003

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.45 (Move Field + Move Method) - Move Field

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.290	20	.000	.677	20	.000
WMC	.507	20	.000	.359	20	.000
CBO	.330	20	.000	.734	20	.000
LCOM	.344	20	.000	.545	20	.000
AverageMethodSize	.416	20	.000	.400	20	.000
MethodHidingFactor	.538	20	.000	.236	20	.000
AttributeHidingFactor	.451	20	.000	.273	20	.000
MethodInheritanceFactor	.331	20	.000	.671	20	.000
AttributeInheritanceFactor	.485	20	.000	.387	20	.000
CouplingFactor	.239	20	.004	.845	20	.004

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.46 (Move Field + Extract Class) - Move Field

Tests of Normality(b,c,d,e,f,g,h,i,j)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.538	20	.000	.236	20	.000
LCOM	.519	20	.000	.398	20	.000
AverageMethodSize	.533	20	.000	.261	20	.000
AttributeHidingFactor	.500	20	.000	.366	20	.000
AttributeInheritanceFactor	.483	20	.000	.390	20	.000

a Lilliefors Significance Correction
b WMC is constant. It has been omitted.

c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e CBO is constant. It has been omitted.
f RFC is constant. It has been omitted.
g MethodHidingFactor is constant. It has been omitted.
h MethodInheritanceFactor is constant. It has been omitted.
i PolymorphismFactor is constant. It has been omitted.
j CouplingFactor is constant. It has been omitted.

1.47 (Move Field + Replace Magic Number with Symbolic Constant) - Move Field

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.341	16	.000	.544	16	.000
CBO	.384	16	.000	.599	16	.000
LCOM	.485	16	.000	.544	16	.000
AverageMethodSize	.221	16	.036	.919	16	.160
AttributeHidingFactor	.355	16	.000	.431	16	.000
AttributeInheritanceFactor	.244	16	.012	.718	16	.000
CouplingFactor	.455	16	.000	.327	16	.000

a Lilliefors Significance Correction
b WMC#Minute is constant. It has been omitted.
c DIT#Minute is constant. It has been omitted.
d NOC#Minute is constant. It has been omitted.
e RFC#Minute is constant. It has been omitted.
f MethodHidingFactor#Minute is constant. It has been omitted.
g MethodInheritanceFactor#Minute is constant. It has been omitted.
h PolymorphismFactor#Minute is constant. It has been omitted.

1.48 (Move Field + Decompose Conditional) - Move Field

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.235	17	.013	.831	17	.006
WMC	.285	17	.001	.814	17	.003
CBO	.307	17	.000	.673	17	.000
LCOM	.247	17	.007	.773	17	.001
AverageMethodSize	.218	17	.032	.749	17	.000
MethodHidingFactor	.149	17	.200(*)	.932	17	.231
AttributeHidingFactor	.449	17	.000	.421	17	.000
MethodInheritanceFactor	.114	17	.200(*)	.927	17	.197
AttributeInheritanceFactor	.443	17	.000	.522	17	.000
CouplingFactor	.199	17	.071	.845	17	.009

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.49 (Move Field + Rename Method) - Move Field

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.297	21	.000	.826	21	.002
WMC	.389	21	.000	.688	21	.000
CBO	.269	21	.000	.821	21	.001
LCOM	.260	21	.001	.781	21	.000
AverageMethodSize	.363	21	.000	.648	21	.000
MethodHidingFactor	.458	21	.000	.530	21	.000
AttributeHidingFactor	.532	21	.000	.258	21	.000
MethodInheritanceFactor	.358	21	.000	.750	21	.000
AttributeInheritanceFactor	.530	21	.000	.266	21	.000
CouplingFactor	.314	21	.000	.706	21	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.50 (Extract Class + Extract Method) - Extract Class

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.276	21	.000	.811	21	.001
WMC	.413	21	.000	.554	21	.000
CBO	.356	21	.000	.539	21	.000
LCOM	.309	21	.000	.670	21	.000
AverageMethodSize	.260	21	.001	.627	21	.000
MethodHidingFactor	.170	21	.115	.954	21	.398
AttributeHidingFactor	.536	21	.000	.240	21	.000
MethodInheritanceFactor	.125	21	.200(*)	.946	21	.285
AttributeInheritanceFactor	.526	21	.000	.284	21	.000
PolymorphismFactor	.539	21	.000	.228	21	.000
CouplingFactor	.234	21	.004	.814	21	.001

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.

1.51 (Extract Class + Replace Temp with Query) - Extract Class

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.304	8	.028	.775	8	.016
CBO	.513	8	.000	.418	8	.000
LCOM	.181	8	.200(*)	.945	8	.660
AverageMethodSize	.199	8	.200(*)	.871	8	.153
MethodHidingFactor	.178	8	.200(*)	.967	8	.873
MethodInheritanceFactor	.307	8	.026	.858	8	.114
CouplingFactor	.513	8	.000	.418	8	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f AttributeHidingFactor is constant. It has been omitted.
g AttributeInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.52 (Extract Class + Introduce Explaining Variable) - Extract Class

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.522	11	.000	.364	11	.000
WMC	.526	11	.000	.352	11	.000
DIT	.528	11	.000	.345	11	.000
NOC	.528	11	.000	.345	11	.000
CBO	.528	11	.000	.345	11	.000
RFC	.528	11	.000	.345	11	.000
LCOM	.491	11	.000	.501	11	.000
AverageMethodSize	.308	11	.005	.597	11	.000
AttributeHidingFactor	.462	11	.000	.439	11	.000
MethodInheritanceFactor	.528	11	.000	.345	11	.000
AttributeInheritanceFactor	.479	11	.000	.496	11	.000
PolymorphismFactor	.528	11	.000	.345	11	.000
CouplingFactor	.528	11	.000	.345	11	.000

a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.

1.53 (Extract Class + Move Method) - Extract Class

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.297	22	.000	.713	22	.000
WMC	.425	22	.000	.597	22	.000
CBO	.355	22	.000	.619	22	.000
LCOM	.309	22	.000	.595	22	.000
AverageMethodSize	.293	22	.000	.572	22	.000
MethodHidingFactor	.463	22	.000	.421	22	.000
AttributeHidingFactor	.524	22	.000	.356	22	.000
MethodInheritanceFactor	.227	22	.004	.861	22	.005
AttributeInheritanceFactor	.524	22	.000	.369	22	.000
CouplingFactor	.482	22	.000	.368	22	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.54 (Extract Class + Move Field) - Extract Class

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.256	20	.001	.714	20	.000
WMC	.507	20	.000	.454	20	.000
CBO	.345	20	.000	.632	20	.000
LCOM	.286	20	.000	.822	20	.002
AverageMethodSize	.287	20	.000	.735	20	.000
MethodHidingFactor	.480	20	.000	.424	20	.000
AttributeHidingFactor	.519	20	.000	.385	20	.000
MethodInheritanceFactor	.213	20	.018	.861	20	.008
AttributeInheritanceFactor	.457	20	.000	.554	20	.000
CouplingFactor	.489	20	.000	.381	20	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.55 (Extract Class + Replace Magic Number with Symbolic Constant) - Extract Class

Tests of Normality(b,c,d,e,f,g,h,i)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.343	17	.000	.518	17	.000
CBO	.396	17	.000	.580	17	.000
LCOM	.537	17	.000	.262	17	.000
AverageMethodSize	.211	17	.043	.892	17	.050
AttributeHidingFactor	.364	17	.000	.413	17	.000
AttributeInheritanceFactor	.262	17	.003	.694	17	.000
CouplingFactor	.464	17	.000	.327	17	.000

a Lilliefors Significance Correction
b CyclomaticComplexity#Minute is constant. It has been omitted.
c WMC#Minute is constant. It has been omitted.
d DIT#Minute is constant. It has been omitted.
e NOC#Minute is constant. It has been omitted.
f RFC#Minute is constant. It has been omitted.
g MethodHidingFactor#Minute is constant. It has been omitted.
h MethodInheritanceFactor#Minute is constant. It has been omitted.
i PolymorphismFactor#Minute is constant. It has been omitted.

1.56 (Extract Class + Decompose Conditional) - Extract Class

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.310	19	.000	.638	19	.000
WMC	.300	19	.000	.789	19	.001
CBO	.277	19	.000	.756	19	.000
LCOM	.243	19	.004	.641	19	.000
AverageMethodSize	.287	19	.000	.671	19	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
MethodHidingFactor	.106	19	.200(*)	.984	19	.980
AttributeHidingFactor	.476	19	.000	.383	19	.000
MethodInheritanceFactor	.130	19	.200(*)	.951	19	.417
AttributeInheritanceFactor	.474	19	.000	.545	19	.000
CouplingFactor	.274	19	.001	.629	19	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.57 (Extract Class + Rename Method) - Extract Class

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
WMC	.439	22	.000	.513	22	.000
DIT	.539	22	.000	.221	22	.000
NOC	.539	22	.000	.221	22	.000
CBO	.326	22	.000	.708	22	.000
RFC	.539	22	.000	.221	22	.000
LCOM	.531	22	.000	.256	22	.000
AverageMethodSize	.447	22	.000	.469	22	.000
AttributeHidingFactor	.527	22	.000	.340	22	.000
MethodInheritanceFactor	.539	22	.000	.221	22	.000
AttributeInheritanceFactor	.507	22	.000	.429	22	.000
CouplingFactor	.288	22	.000	.641	22	.000

a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c MethodHidingFactor is constant. It has been omitted.
d PolymorphismFactor is constant. It has been omitted.

1.58 (Replace Magic Number with Symbolic Constant + Extract Method) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.208	20	.023	.843	20	.004
WMC	.444	20	.000	.529	20	.000
CBO	.360	20	.000	.553	20	.000
LCOM	.319	20	.000	.691	20	.000
AverageMethodSize	.272	20	.000	.676	20	.000
MethodHidingFactor	.117	20	.200(*)	.968	20	.704
AttributeHidingFactor	.454	20	.000	.482	20	.000
MethodInheritanceFactor	.113	20	.200(*)	.969	20	.725
AttributeInheritanceFactor	.469	20	.000	.314	20	.000
PolymorphismFactor	.538	20	.000	.236	20	.000
CouplingFactor	.215	20	.016	.833	20	.003

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.
d RFC#Minute is constant. It has been omitted.

1.59 (Replace Magic Number with Symbolic Constant + Replace Temp with Query) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c,d,e,f,g,h)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.282	7	.097	.829	7	.078
CBO	.504	7	.000	.453	7	.000
LCOM	.209	7	.200(*)	.936	7	.605
AverageMethodSize	.251	7	.200(*)	.807	7	.048
MethodHidingFactor	.166	7	.200(*)	.926	7	.518
MethodInheritanceFactor	.328	7	.022	.718	7	.006
CouplingFactor	.504	7	.000	.453	7	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC is constant. It has been omitted.
c DIT is constant. It has been omitted.
d NOC is constant. It has been omitted.
e RFC is constant. It has been omitted.
f AttributeHidingFactor is constant. It has been omitted.
g AttributeInheritanceFactor is constant. It has been omitted.
h PolymorphismFactor is constant. It has been omitted.

1.60 (Replace Magic Number with Symbolic Constant + Introduce Explaining Variable) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.461	12	.000	.481	12	.000
WMC	.496	12	.000	.491	12	.000
DIT	.530	12	.000	.327	12	.000
NOC#Minute	.530	12	.000	.327	12	.000
CBO#Minute	.508	12	.000	.335	12	.000
RFC#Minute	.530	12	.000	.327	12	.000
LCOM#Minute	.397	12	.000	.600	12	.000
AverageMethodSize	.451	12	.000	.396	12	.000
AttributeHidingFactor	.478	12	.000	.573	12	.000
MethodInheritanceFactor	.530	12	.000	.327	12	.000
AttributeInheritanceFactor	.427	12	.000	.589	12	.000
CouplingFactor	.487	12	.000	.472	12	.000

a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.61 (Replace Magic Number with Symbolic Constant + Move Method) - Replace Magic Number with Symbolic Constant

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.238	17	.011	.883	17	.036
WMC	.292	17	.000	.781	17	.001
DIT	.203	17	.062	.878	17	.030
NOC	.125	17	.200(*)	.926	17	.186
CBO	.224	17	.024	.892	17	.049
RFC	.200	17	.069	.834	17	.006
LCOM	.326	17	.000	.642	17	.000
AverageMethodSize	.283	17	.001	.659	17	.000
MethodHidingFactor	.537	17	.000	.262	17	.000
AttributeHidingFactor	.518	17	.000	.405	17	.000
MethodInheritanceFactor	.220	17	.028	.896	17	.059
AttributeInheritanceFactor	.510	17	.000	.442	17	.000
CouplingFactor	.238	17	.011	.644	17	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.62 (Replace Magic Number with Symbolic Constant + Move Field) - Replace Magic Number with Symbolic Constant

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.243	16	.013	.866	16	.023
WMC	.279	16	.002	.814	16	.004
DIT	.207	16	.065	.888	16	.052
NOC	.119	16	.200(*)	.942	16	.370
CBO	.285	16	.001	.858	16	.018
RFC	.201	16	.082	.842	16	.010
LCOM	.224	16	.031	.917	16	.149
AverageMethodSize	.278	16	.002	.815	16	.004
MethodHidingFactor	.536	16	.000	.273	16	.000
AttributeHidingFactor	.511	16	.000	.441	16	.000
MethodInheritanceFactor	.236	16	.017	.877	16	.035
AttributeInheritanceFactor	.376	16	.000	.726	16	.000
CouplingFactor	.231	16	.022	.664	16	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.63 (Replace Magic Number with Symbolic Constant + Extract Class) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.298	17	.000	.796	17	.002
WMC	.294	17	.000	.791	17	.002
DIT	.203	17	.062	.878	17	.030
NOC	.125	17	.200(*)	.926	17	.186
CBO	.303	17	.000	.708	17	.000
RFC	.200	17	.069	.834	17	.006
LCOM	.290	17	.000	.753	17	.000
AverageMethodSize	.304	17	.000	.704	17	.000
AttributeHidingFactor	.520	17	.000	.397	17	.000
AttributeInheritanceFactor	.518	17	.000	.399	17	.000
CouplingFactor	.460	17	.000	.344	17	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c MethodInheritanceFactor is constant. It has been omitted.
d PolymorphismFactor is constant. It has been omitted.

1.64 (Replace Magic Number with Symbolic Constant + Decompose Conditional) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.442	18	.000	.516	18	.000
WMC	.445	18	.000	.338	18	.000
CBO	.209	18	.037	.835	18	.005
RFC	.538	18	.000	.253	18	.000
LCOM	.298	18	.000	.731	18	.000
AverageMethodSize	.314	18	.000	.660	18	.000
MethodHidingFactor	.104	18	.200(*)	.975	18	.880
AttributeHidingFactor	.472	18	.000	.476	18	.000
MethodInheritanceFactor	.153	18	.200(*)	.920	18	.128
AttributeInheritanceFactor	.531	18	.000	.276	18	.000
PolymorphismFactor	.538	18	.000	.253	18	.000
CouplingFactor	.264	18	.002	.679	18	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.

1.65 (Replace Magic Number with Symbolic Constant + Rename Method) - Replace Magic Number with Symbolic Constant

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.442	22	.000	.422	22	.000
WMC	.493	22	.000	.249	22	.000
CBO	.297	22	.000	.788	22	.000
LCOM	.480	22	.000	.435	22	.000
AverageMethodSize	.486	22	.000	.252	22	.000
AttributeHidingFactor	.453	22	.000	.473	22	.000
AttributeInheritanceFactor	.455	22	.000	.279	22	.000
CouplingFactor	.228	22	.004	.819	22	.001

a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.
d RFC#Minute is constant. It has been omitted.
e MethodHidingFactor#Minute is constant. It has been omitted.
f MethodInheritanceFactor#Minute is constant. It has been omitted.
g PolymorphismFactor#Minute is constant. It has been omitted.

1.66 (Decompose Conditional + Extract Method) - Decompose Conditional

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.283	22	.000	.758	22	.000
WMC	.350	22	.000	.562	22	.000
CBO	.345	22	.000	.561	22	.000
LCOM	.338	22	.000	.600	22	.000
AverageMethodSize	.231	22	.003	.770	22	.000
MethodHidingFactor	.114	22	.200(*)	.950	22	.311
AttributeHidingFactor	.537	22	.000	.233	22	.000
MethodInheritanceFactor	.122	22	.200(*)	.931	22	.129
AttributeInheritanceFactor	.527	22	.000	.357	22	.000
PolymorphismFactor	.539	22	.000	.221	22	.000
CouplingFactor	.216	22	.009	.831	22	.002

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.
d RFC#Minute is constant. It has been omitted.

1.67 (Decompose Conditional + Replace Temp with Query) - Decompose Conditional

Tests of Normality(b,c,d,e,f)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.319	7	.030	.812	7	.054
CBO	.406	7	.001	.737	7	.009
LCOM	.199	7	.200(*)	.916	7	.439

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.304	7	.050	.696	7	.003
MethodHidingFactor	.210	7	.200(*)	.891	7	.279
AttributeHidingFactor	.504	7	.000	.453	7	.000
MethodInheritanceFactor	.340	7	.014	.813	7	.055
AttributeInheritanceFactor	.427	7	.000	.652	7	.001
CouplingFactor	.375	7	.004	.762	7	.017

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC#Minute is constant. It has been omitted.
c DIT#Minute is constant. It has been omitted.
d NOC#Minute is constant. It has been omitted.
e RFC#Minute is constant. It has been omitted.
f PolymorphismFactor#Minute is constant. It has been omitted.

1.68 (Decompose Conditional + Introduce Explaining Variable) - Decompose Conditional

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.244	11	.066	.789	11	.007
WMC	.376	11	.000	.803	11	.010
CBO	.346	11	.001	.767	11	.003
LCOM	.441	11	.000	.455	11	.000
AverageMethodSize	.286	11	.012	.831	11	.024
MethodHidingFactor	.353	11	.000	.800	11	.009
AttributeHidingFactor	.528	11	.000	.345	11	.000
MethodInheritanceFactor	.362	11	.000	.769	11	.004
AttributeInheritanceFactor	.528	11	.000	.345	11	.000
CouplingFactor	.443	11	.000	.603	11	.000

a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.
d RFC#Minute is constant. It has been omitted.
e PolymorphismFactor#Minute is constant. It has been omitted.

1.69 (Decompose Conditional + Move Method) - Decompose Conditional

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.199	19	.047	.903	19	.054
WMC	.262	19	.001	.801	19	.001
DIT	.187	19	.078	.884	19	.026
NOC	.168	19	.164	.933	19	.194
CBO	.268	19	.001	.836	19	.004
RFC	.207	19	.031	.811	19	.002
LCOM	.299	19	.000	.577	19	.000
AverageMethodSize	.174	19	.133	.915	19	.090
MethodHidingFactor	.188	19	.075	.942	19	.287
AttributeHidingFactor	.522	19	.000	.372	19	.000
MethodInheritanceFactor	.091	19	.200(*)	.984	19	.979
AttributeInheritanceFactor	.501	19	.000	.471	19	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CouplingFactor	.242	19	.005	.762	19	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.70 (Decompose Conditional + Move Field) - Decompose Conditional

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.192	17	.098	.911	17	.102
WMC	.247	17	.007	.849	17	.010
DIT	.187	17	.119	.904	17	.078
NOC	.174	17	.182	.962	17	.661
CBO	.248	17	.007	.699	17	.000
RFC	.211	17	.043	.827	17	.005
LCOM	.236	17	.012	.858	17	.014
AverageMethodSize	.152	17	.200(*)	.954	17	.528
MethodHidingFactor	.202	17	.063	.946	17	.403
AttributeHidingFactor	.517	17	.000	.398	17	.000
MethodInheritanceFactor	.120	17	.200(*)	.961	17	.656
AttributeInheritanceFactor	.493	17	.000	.501	17	.000
CouplingFactor	.284	17	.001	.756	17	.001

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.71 (Decompose Conditional + Extract Class) - Decompose Conditional

Tests of Normality(b,c,d)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.220	19	.016	.855	19	.008
WMC	.293	19	.000	.809	19	.002
DIT	.187	19	.078	.884	19	.026
NOC	.168	19	.164	.933	19	.194
CBO	.253	19	.002	.726	19	.000
RFC	.207	19	.031	.811	19	.002
LCOM	.291	19	.000	.614	19	.000
AverageMethodSize	.277	19	.000	.866	19	.012
AttributeHidingFactor	.476	19	.000	.383	19	.000
AttributeInheritanceFactor	.453	19	.000	.633	19	.000
CouplingFactor	.384	19	.000	.374	19	.000

a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c MethodInheritanceFactor is constant. It has been omitted.
d PolymorphismFactor is constant. It has been omitted.

1.72 (Decompose Conditional + Replace Magic Number with Symbolic Constant) - Decompose Conditional

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.415	18	.000	.514	18	.000
WMC	.520	18	.000	.262	18	.000
CBO	.229	18	.014	.848	18	.008
RFC	.538	18	.000	.253	18	.000
LCOM	.430	18	.000	.483	18	.000
AverageMethodSize	.330	18	.000	.748	18	.000
MethodHidingFactor	.339	18	.000	.801	18	.002
AttributeHidingFactor	.314	18	.000	.615	18	.000
MethodInheritanceFactor	.380	18	.000	.532	18	.000
AttributeInheritanceFactor	.397	18	.000	.622	18	.000
PolymorphismFactor	.538	18	.000	.253	18	.000
CouplingFactor	.435	18	.000	.524	18	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.

1.73 (Decompose Conditional + Rename Method) - Decompose Conditional

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.442	22	.000	.282	22	.000
WMC	.480	22	.000	.261	22	.000
CBO	.258	22	.001	.837	22	.002
LCOM	.346	22	.000	.564	22	.000
AverageMethodSize	.310	22	.000	.799	22	.000
MethodHidingFactor	.311	22	.000	.829	22	.001
AttributeHidingFactor	.456	22	.000	.526	22	.000
MethodInheritanceFactor	.315	22	.000	.841	22	.002
AttributeInheritanceFactor	.515	22	.000	.310	22	.000
CouplingFactor	.277	22	.000	.699	22	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.74 (Rename Method + Extract Method) - Rename Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.291	21	.000	.686	21	.000
WMC	.410	21	.000	.572	21	.000
CBO	.363	21	.000	.483	21	.000
LCOM	.327	21	.000	.632	21	.000
AverageMethodSize	.275	21	.000	.656	21	.000
MethodHidingFactor	.106	21	.200(*)	.982	21	.951

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AttributeHidingFactor	.533	21	.000	.256	21	.000
MethodInheritanceFactor	.159	21	.177	.963	21	.571
AttributeInheritanceFactor	.519	21	.000	.314	21	.000
CouplingFactor	.269	21	.000	.805	21	.001

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

1.75 (Rename Method + Replace Temp with Query) - Rename Method

Tests of Normality(b,c)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.172	12	.200(*)	.914	12	.238
DIT	.530	12	.000	.327	12	.000
NOC	.530	12	.000	.327	12	.000
CBO	.527	12	.000	.342	12	.000
RFC	.530	12	.000	.327	12	.000
LCOM	.274	12	.013	.879	12	.084
AverageMethodSize	.251	12	.036	.785	12	.006
MethodHidingFactor	.120	12	.200(*)	.961	12	.792
AttributeHidingFactor	.530	12	.000	.327	12	.000
MethodInheritanceFactor	.243	12	.048	.695	12	.001
AttributeInheritanceFactor	.495	12	.000	.488	12	.000
CouplingFactor	.530	12	.000	.329	12	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b WMC#Minute is constant. It has been omitted.
c PolymorphismFactor#Minute is constant. It has been omitted.

1.76 (Rename Method + Introduce Explaining Variable) - Rename Method

Tests of Normality(b,c,d,e,f,g,h,i)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.332	12	.001	.638	12	.000
WMC	.499	12	.000	.479	12	.000
LCOM	.406	12	.000	.683	12	.001
AverageMethodSize	.186	12	.200(*)	.857	12	.044
AttributeHidingFactor	.478	12	.000	.522	12	.000
AttributeInheritanceFactor	.441	12	.000	.633	12	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT#Minute is constant. It has been omitted.
c NOC#Minute is constant. It has been omitted.
d CBO#Minute is constant. It has been omitted.
e RFC#Minute is constant. It has been omitted.
f MethodHidingFactor#Minute is constant. It has been omitted.
g MethodInheritanceFactor#Minute is constant. It has been omitted.
h PolymorphismFactor#Minute is constant. It has been omitted.
i CouplingFactor#Minute is constant. It has been omitted.

1.77 (Rename Method + Move Method) -Rename Method

Tests of Normality

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.162	22	.136	.896	22	.025
WMC	.352	22	.000	.600	22	.000
DIT	.156	22	.176	.927	22	.104
NOC	.142	22	.200(*)	.844	22	.003
CBO	.237	22	.002	.887	22	.017
RFC	.304	22	.000	.474	22	.000
LCOM	.287	22	.000	.667	22	.000
AverageMethodSize	.250	22	.001	.708	22	.000
MethodHidingFactor	.496	22	.000	.375	22	.000
AttributeHidingFactor	.489	22	.000	.397	22	.000
MethodInheritanceFactor	.264	22	.000	.555	22	.000
AttributeInheritanceFactor	.477	22	.000	.330	22	.000
PolymorphismFactor	.539	22	.000	.221	22	.000
CouplingFactor	.323	22	.000	.647	22	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction

1.78 (Rename Method + Move Field) - Rename Method

Tests of Normality(b)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.178	21	.081	.913	21	.064
WMC	.271	21	.000	.827	21	.002
DIT#Minute	.171	21	.112	.941	21	.233
NOC#Minute	.173	21	.100	.955	21	.428
CBO#Minute	.212	21	.015	.863	21	.007
RFC#Minute	.197	21	.032	.857	21	.006
LCOM	.247	21	.002	.796	21	.001
AverageMethodSize	.227	21	.006	.829	21	.002
MethodHidingFactor	.433	21	.000	.527	21	.000
AttributeHidingFactor	.533	21	.000	.255	21	.000
MethodInheritanceFactor	.260	21	.001	.864	21	.007
AttributeInheritanceFactor	.534	21	.000	.251	21	.000
CouplingFactor	.242	21	.002	.729	21	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b PolymorphismFactor is constant. It has been omitted.

1.79 (Rename Method + Extract Class) - Rename Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.209	22	.014	.861	22	.005
WMC	.281	22	.000	.816	22	.001
DIT	.292	22	.000	.589	22	.000
NOC	.186	22	.047	.764	22	.000
CBO	.257	22	.001	.759	22	.000
RFC	.239	22	.002	.750	22	.000

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
LCOM	.241	22	.002	.754	22	.000
AverageMethodSize	.269	22	.000	.728	22	.000
AttributeHidingFactor	.512	22	.000	.294	22	.000
MethodInheritanceFactor	.539	22	.000	.221	22	.000
AttributeInheritanceFactor	.500	22	.000	.411	22	.000
CouplingFactor	.364	22	.000	.469	22	.000

a Lilliefors Significance Correction
b MethodHidingFactor is constant. It has been omitted.
c PolymorphismFactor is constant. It has been omitted.

1.80 (Rename Method + Replace Magic Number with Symbolic Constant) - Rename Method

Tests of Normality(b,c,d,e,f,g)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.409	22	.000	.524	22	.000
WMC	.539	22	.000	.221	22	.000
CBO	.394	22	.000	.652	22	.000
LCOM	.451	22	.000	.423	22	.000
AverageMethodSize	.490	22	.000	.313	22	.000
AttributeHidingFactor	.325	22	.000	.580	22	.000
AttributeInheritanceFactor	.383	22	.000	.645	22	.000
CouplingFactor	.499	22	.000	.374	22	.000

a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e MethodHidingFactor is constant. It has been omitted.
f MethodInheritanceFactor is constant. It has been omitted.
g PolymorphismFactor is constant. It has been omitted.

1.81 (Rename Method + Decompose Conditional) - Rename Method

Tests of Normality(b,c,d,e)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CommentPercentage	.385	22	.000	.540	22	.000
WMC	.453	22	.000	.309	22	.000
CBO	.239	22	.002	.768	22	.000
LCOM	.263	22	.000	.707	22	.000
AverageMethodSize	.283	22	.000	.664	22	.000
MethodHidingFactor	.113	22	.200(*)	.973	22	.779
AttributeHidingFactor	.437	22	.000	.487	22	.000
MethodInheritanceFactor	.135	22	.200(*)	.943	22	.223
AttributeInheritanceFactor	.524	22	.000	.284	22	.000
CouplingFactor	.279	22	.000	.685	22	.000

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b DIT is constant. It has been omitted.
c NOC is constant. It has been omitted.
d RFC is constant. It has been omitted.
e PolymorphismFactor is constant. It has been omitted.

3. ข้อมูลสำหรับการทดสอบสมมติฐานที่ 3: การเปรียบเทียบค่ามาตรวัดเชิงวัตถุเมื่อทำกระบวนการรีแฟคทอริงวิธีที่ x ตามด้วยวิธีที่ y เปรียบเทียบกับหลังทำกระบวนการรีแฟคทอริงวิธีที่ y ตามด้วยวิธีที่ x

1.82 (Extract Method + Replace Temp with Query) - (Replace Temp with Query + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.118	10	.200(*)	.967	10	.858

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.83 (Extract Method + Introduce Explaining Variable) - (Introduce Explaining Variable + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.186	13	.200(*)	.961	13	.762

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.84 (Extract Method + Move Method) - (Move Method + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.140	23	.200(*)	.964	23	.542

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.85 (Extract Method + Move Field) - (Move Field + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.162	21	.159	.924	21	.105

a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.86 (Extract Method + Extract Class) - (Extract Class + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.175	22	.079	.895	22	.024

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.

g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.87 (Extract Method + Replace Magic Number with Symbolic Constant) - (Replace Magic Number with Symbolic Constant + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.119	20	.200(*)	.961	20	.564

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.88 (Extract Method + Decompose Conditional) - (Decompose Conditional + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.221	22	.007	.923	22	.087

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

1.89 (Extract Method + Rename Method) - (Rename Method + Extract Method)

Tests of Normality(b,c,d,e,f,g,h,i,j,k,l,m,n)

	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
AverageMethodSize	.205	27	.005	.928	27	.063

* This is a lower bound of the true significance.
a Lilliefors Significance Correction
b CommentPercentage is constant. It has been omitted.
c WMC is constant. It has been omitted.
d DIT is constant. It has been omitted.
e NOC is constant. It has been omitted.
f CBO is constant. It has been omitted.
g RFC is constant. It has been omitted.
h LCOM is constant. It has been omitted.
i MethodHidingFactor is constant. It has been omitted.
j AttributeHidingFactor is constant. It has been omitted.
k MethodInheritanceFactor is constant. It has been omitted.
l AttributeInheritanceFactor is constant. It has been omitted.
m PolymorphismFactor is constant. It has been omitted.
n CouplingFactor is constant. It has been omitted.

สามารถอ่านรายละเอียดการแจกแจงข้อมูลและการทดสอบสมมติฐานในรูปแบบสำเนาอิเล็กทรอนิกส์ (Softcopy) ได้จากซีดีรอม (CD-ROM) ที่แนบมาพร้อมกับงานวิจัยนี้

ภาคผนวก ง
เอกสารรายชื่อหน่วยทดลอง

1. หน่วยทดลองซึ่งเป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิต ในวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ปีพ.ศ. 2547 ทั้งหมด 19 หน่วยทดลอง

- ระบบศูนย์คอมพิวเตอร์
- ระบบร้านจำหน่ายสื่อบันเทิง
- ระบบคาราโอเกะ
- ระบบโรงภาพยนตร์
- ระบบรถเงินสด
- ระบบซุปเปอร์มาร์เก็ต
- ระบบการประกันภัย
- ระบบร้านไอศกรีม
- ระบบดูดวง
- ระบบการส่งสิ่งของ
- ระบบร้านพิซซ่า
- ระบบร้านสหกรณ์
- ระบบโรงแรม
- ระบบสถาบันกวดวิชา
- ระบบโรงภาพยนตร์
- ระบบร้านเช่าหนังสือ
- ระบบสหกรณ์
- ระบบบัตรเติมเงินเกมส์ออนไลน์
- ระบบไปรษณีย์

2. หน่วยทดลองซึ่งเป็นงานที่ได้รับมอบหมาย (Assignment) ของนิสิตปริญญาบัณฑิต ในวิชาพื้นฐานการโปรแกรมเชิงวัตถุ (Object - Oriented Programming Foundation) ปีพ.ศ. 2548 ทั้งหมด 13 หน่วยทดลอง

- ระบบโรงภาพยนตร์
- ระบบจัดการสถานีบริการน้ำมัน
- ระบบการคิดอัตราค่าที่พักโรงแรม
- ระบบจองห้องพักโรงแรม
- ระบบสถาบันกวดวิชา
- ระบบหอพัก
- ระบบจัดการพนักงาน
- ระบบจองตั๋วเครื่องบิน
- ระบบฟิตเนส
- ระบบขายซอฟต์แวร์สำเร็จรูป
- ระบบร้านขายเสื้อผ้า
- ระบบแสงและเสียง
- ระบบร้านอาหาร

ประวัติผู้เขียนวิทยานิพนธ์

นายศิรธันย์ ศุภธนะรัตน์ เกิดวันที่ 25 กันยายน พ.ศ. 2525 สำเร็จการศึกษาวิศวกรรมบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีพ.ศ. 2547 จากนั้นได้เข้าศึกษาต่อในระดับปริญญาโท สาขาการพัฒนาซอฟต์แวร์ด้านธุรกิจ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย