การทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยใช้มาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุแบบยูเอ็มแอล

นางสาวนงเยาว์ จินดาสวัสดิ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2546

ISBN 974-17-4314-9

PREDICTING SOFTWARE MAINTAINABILITY
USING UML-BASED OBJECT-ORIENTED ANALYSIS AND DESIGN METRICS

Miss Nongyao Jindasawat

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2003
ISBN 974-17-4314-9

หัวข้อวิทยานิพนธ์	การทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้มาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุแบบยูเอ็มแอล
โดย	นางสาวนงเยาว์ จินดาสวัสดิ์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี
อาจารย์ที่ปรึกษาร่วม	อาจารย์นครทิพย์ พร้อมพูล

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์)

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

..... อาจารย์ที่ปรึกษาร่วม
(อาจารย์นครทิพย์ พร้อมพูล)

..... กรรมการ
(อาจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

..... กรรมการ
(อาจารย์เชษฐ พัฒโนทัย)

นงเยาว์ จินดาสวัสดิ์ : การทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้มาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุแบบยูเอ็มแอล. (PREDICTING SOFTWARE MAINTAINABILITY USING UML-BASED OBJECT-ORIENTED ANALYSIS AND DESIGN METRICS) อ. ที่ปรึกษา: ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี, อ.ที่ปรึกษาร่วม: อาจารย์นครทิพย์ พร้อมพูล จำนวนหน้า 169 หน้า. ISBN 974-17-4314-9.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อหาความสัมพันธ์ระหว่างมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุและความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยใช้มาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุจากแผนภาพคลาสและแผนภาพซีเควนซ์จำนวน 18 มาตรวัด เพื่อสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่สามารถทำนายได้ 3 ระดับ คือ ระดับยาก ปานกลาง และง่าย จากนั้นทำการออกแบบและทำการทดลองเพื่อเก็บรวบรวมข้อมูลเพื่อนำไปสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ด้วยวิธีการวิเคราะห์จำแนกกลุ่ม ระบบที่นำมาใช้ในการทดลองจำนวน 40 ระบบ แบ่งออกเป็น ระบบที่ใช้เป็นข้อมูลสอนสำหรับการสร้างโมเดลการทำนายจำนวน 35 ระบบ และระบบที่ใช้เป็นข้อมูลทดสอบสำหรับตรวจสอบความถูกต้องของโมเดลจำนวน 5 ระบบ

ผู้วิจัยได้พัฒนาเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยภาษาจาวา ซึ่งข้อมูลนำเข้าสำหรับเครื่องมือที่พัฒนาขึ้นได้จากการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ด้วยโปรแกรมเรชันเนลโรสและแปลงแผนภาพให้อยู่ในรูปของเอกสารเอ็กซ์เอ็มแอลด้วยโปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล

ผลการทดลองพบว่ามาตรวัดการวิเคราะห์และออกแบบระบบเชิงวัตถุที่มีความสัมพันธ์กับความสามารถในการบำรุงรักษาซอฟต์แวร์ มีจำนวน 14 มาตรวัด ได้แก่ มาตรวัด NC ANAUW ANMUW ANAsso NaggH MaxHAgg NGenH MaxDIT NOS WMBO ANRM ANDM ANET และ ANCM จากการตรวจสอบความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ พบว่าระบบจำนวน 3 ระบบจาก 5 ระบบสามารถทำนายอยู่ในกลุ่มที่ถูกต้อง คิดเป็น 60 เปอร์เซ็นต์

ภาควิชาวิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต.....
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์	ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2546	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....

4470356421 : MAJOR COMPUTER SCIENCE

KEY WORD: MAINTAINABILITY / PREDICTION MODEL / SOFTWARE QUALITY / DESIGN METRICS / OBJECT-ORIENTED

NONGYAO JINDASAWAT : (PREDICTING SOFTWARE MAINTAINABILITY USING UML-BASED OBJECT-ORIENTED ANALYSIS AND DESIGN METRICS)

THESIS ADVISOR : ASSISTANT PROFESSOR PORNSIRI MUENCHAISRI, Ph.D., THESIS COADVISOR : NAKORNTHIP PROMPOON, 169 pp. ISBN 974-17-4314-9.

The objective of this thesis is to explore the correlation between object-oriented analysis and design metrics and maintainability using 18 metrics from UML class and sequence diagrams in order to construct a software maintainability prediction model. The obtained model can identify 3 levels of maintainability of UML class and sequence diagrams: difficult, medium and easy. The maintainability model, based on data collected from controlled experiments, is constructed by using discriminant analysis. Forty software design models are categorized to 35 training data sets for constructing maintainability model and 5 test data sets for validating the constructed model.

This research work also constructs an automated tool for measuring software metrics and for predicting maintainability. Input data for this tool, which is an XML document representing UML class and sequence diagrams, is exported from Rational Rose using Unisys Rose XML Tool.

The result shows that 14 object-oriented analysis and design metrics: NC, ANAUW, ANMUW, ANAsso, NaggH, MaxHAgg, NGenH, MaxDIT, NOS, WMBO, ANRM, ANDM and ANCM are related to maintainability. Validation result presents that three out of five new software design models are correctly predicted by the obtained model.

Department Computer Engineering Student's signature.....
Field of study Computer Science Advisor's signature.....
Academic year 2003 Co-advisor's nature.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจาก ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า และอาจารย์นครทิพย์ พร้อมพูล อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วมของข้าพเจ้า ขอกราบขอบพระคุณอาจารย์ทั้งสองท่านที่ได้ให้คำแนะนำ และข้อเสนอแนะต่างๆ ตลอดระยะเวลาของการจัดทำวิทยานิพนธ์ของข้าพเจ้าอย่างดียิ่ง จนสำเร็จลุล่วงได้ด้วยดี

ขอกราบขอบพระคุณรองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์ เป็นประธานกรรมการ อาจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ และอาจารย์เชษฐ พัฒโนทัย เป็นกรรมการสอบวิทยานิพนธ์ ซึ่งได้สละเวลาและให้คำแนะนำต่างๆ ในการสอบวิทยานิพนธ์ของข้าพเจ้าอย่างดียิ่ง

ขอขอบคุณ คุณเมทินี เขียวกันยะที่ให้คำแนะนำ ปรึกษาและความช่วยเหลือในทุกๆ ด้านและขอบคุณรุ่นน้องทุกคนที่ช่วยสละเวลามาทำข้อสอบในงานวิทยานิพนธ์ชิ้นนี้ สุดท้ายนี้ของกราบขอบพระคุณบิดา มารดา พี่ชาย พี่สาว และขอบคุณเพื่อนๆ ทุกคนที่คอยเป็นกำลังใจและให้ความสนับสนุนมาโดยตลอด

และท้ายที่สุดนี้ขอขอบพระคุณทางตลาดหลักทรัพย์แห่งประเทศไทย ที่ได้สนับสนุนในด้านการเงินให้กับ “งานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุ” ซึ่งเป็นงานวิจัยร่วมระหว่างภาครัฐบาล (ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย) กับตลาดหลักทรัพย์แห่งประเทศไทย

สารบัญ

หน้า

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

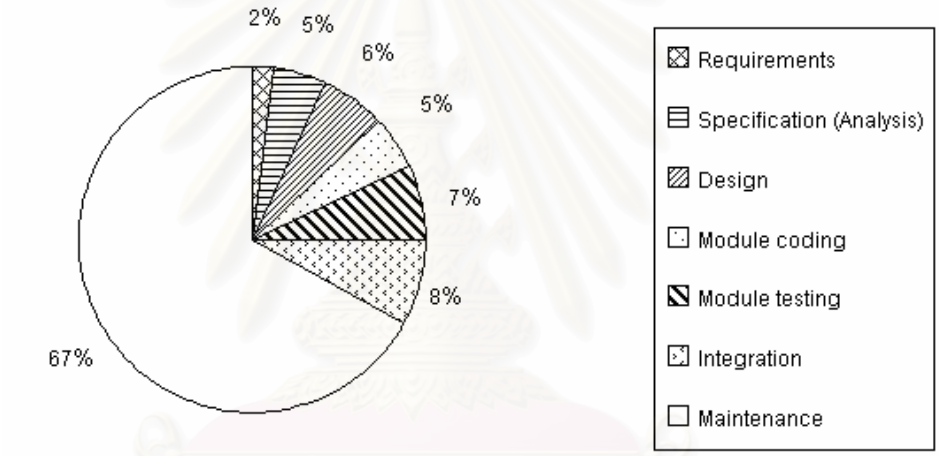
สารบัญภาพ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การพัฒนาซอฟต์แวร์ ประกอบด้วยขั้นตอนหลายขั้นตอน เริ่มตั้งแต่การเก็บรวบรวมความต้องการจากผู้ใช้งาน (User requirements) จากนั้นเป็นการวิเคราะห์ระบบ โดยมุ่งหาปัญหาที่ต้องการแก้ไขคืออะไร [4] ขั้นตอนต่อไปคือการออกแบบระบบที่มุ่งเน้นทั้งในส่วนความต้องการด้านหน้าที่ (Functional) และไม่ใช่หน้าที่ (Nonfunctional) และข้อกำหนดต่างๆ ในข้อกำหนดของปัญหา (Problem statement) [4] และเข้าสู่ขั้นตอนการพัฒนาระบบ ทดสอบระบบ และขั้นตอนสุดท้ายคือการบำรุงรักษาระบบ



ที่มา : Classical and Object-Oriented Software Engineering with UML and Java™ [14]

รูปที่ 1.1 แสดงเปอร์เซ็นต์ของเวลาที่ใช้ในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์

จากรูปที่ 1.1 แสดงเปอร์เซ็นต์ของเวลาที่ใช้ในแต่ละขั้นตอนของการพัฒนาซอฟต์แวร์ พบว่าขั้นตอนการบำรุงรักษาซอฟต์แวร์เป็นขั้นตอนที่ใช้ระยะเวลานานที่สุด เนื่องจากเป็นขั้นตอนที่เริ่มตั้งแต่พัฒนาซอฟต์แวร์เสร็จ นำซอฟต์แวร์ไปติดตั้งให้กับผู้ใช้งาน รวมถึงการแก้ไขข้อผิดพลาดที่เกิดขึ้น (Corrective) การปรับเปลี่ยนซอฟต์แวร์ให้เข้ากับสภาพแวดล้อมใหม่ๆ (Adaptive) การปรับปรุงฟังก์ชันการทำงานให้สมบูรณ์ขึ้น (Perfective) และการป้องกันและรักษาความน่าเชื่อถือของซอฟต์แวร์ (Preventive) [12] ทั้งหมดเป็นกิจกรรมที่เกิดขึ้นหลังจากพัฒนาซอฟต์แวร์เสร็จ เนื่องจากซอฟต์แวร์ที่พัฒนาได้ต้องมีการปรับปรุงแก้ไขเพื่อรองรับความต้องการของผู้ใช้งานที่มีอยู่ตลอดเวลา สาเหตุหนึ่งของความยากง่ายในการปรับปรุงแก้ไขซอฟต์แวร์มาจากการออกแบบโมเดลในขั้นตอนของการวิเคราะห์และออกแบบระบบ ดังนั้นถ้าสามารถวัดคุณภาพ

ของซอฟต์แวร์ในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบระบบได้ จะช่วยให้ทราบคุณภาพของซอฟต์แวร์ได้เร็วขึ้น ซึ่งหากโมเดลการวิเคราะห์และออกแบบระบบที่ได้มีคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ไม่น่าพอใจ นักพัฒนาระบบสามารถกลับไปแก้ไขโมเดลการวิเคราะห์และออกแบบระบบให้มีลักษณะเป็นที่พอใจก่อนนำไปพัฒนาเป็นซอฟต์แวร์ต่อไป ซึ่งช่วยให้การบำรุงรักษาซอฟต์แวร์มีความง่ายขึ้น และเป็นการลดระยะเวลาและต้นทุนในการพัฒนาซอฟต์แวร์โดยรวมอีกด้วย

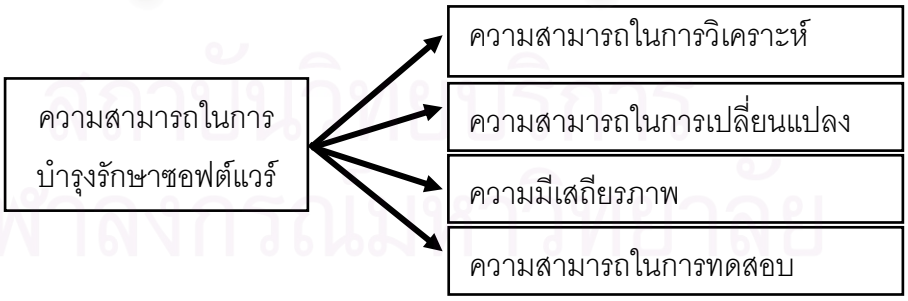
โมเดลวัดคุณภาพผลิตผลซอฟต์แวร์ (Software productivity) ตามมาตรฐานไอเอสโอ 9126 (ISO 9126) [9,13] ได้นิยามความสามารถในการบำรุงรักษาซอฟต์แวร์ (Maintainability) ไว้คือ “ความสามารถของซอฟต์แวร์เมื่อมีการแก้ไข (The capability of the software to be modified)” ซึ่งแบ่งเกณฑ์ (Criteria) ของความสามารถในการบำรุงรักษาซอฟต์แวร์ออกเป็น 4 เกณฑ์ ดังแสดงในรูปที่ 1.2 คือ

1) ความสามารถในการวิเคราะห์ (Analyzability) หมายถึง “ประสิทธิภาพของผลิตผลซอฟต์แวร์ในการวิเคราะห์ เพื่อตรวจหาความไม่สมบูรณ์หรือสาเหตุของความล้มเหลวในซอฟต์แวร์ หรือการระบุตำแหน่งที่ต้องทำการแก้ไข”

2) ความสามารถในการเปลี่ยนแปลง (Changeability) หมายถึง “ประสิทธิภาพของผลิตผลซอฟต์แวร์ที่สามารถระบุรายละเอียดวิธีการแก้ไขได้”

3) ความมีเสถียรภาพ (Stability) หมายถึง “ประสิทธิภาพของผลิตผลซอฟต์แวร์ซึ่งจะมีผลกระทบน้อยที่สุด เมื่อมีการแก้ไขซอฟต์แวร์”

4) ความสามารถในการทดสอบ (Testability) หมายถึง “ประสิทธิภาพของผลิตผลซอฟต์แวร์ที่สามารถใช้งานได้เมื่อซอฟต์แวร์มีการแก้ไข”



รูปที่ 1.2 แสดงโมเดลวัดคุณภาพความสามารถในการบำรุงรักษาซอฟต์แวร์ตามมาตรฐานไอเอสโอ 9126

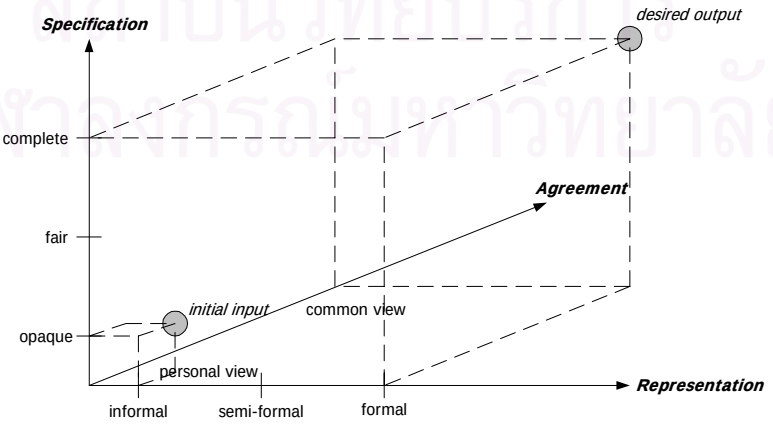
เกณฑ์ทั้งหมดของมาตรฐานไอเอสโอ 9126 นำมาใช้สำหรับวัดคุณภาพของผลิตผลซอฟต์แวร์ที่ได้หลังจากพัฒนาระบบเป็นซอฟต์แวร์แล้ว เพื่อให้สามารถวัดคุณภาพของผลิตผลซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบระบบได้ จึงทำการปรับเปลี่ยนเกณฑ์ที่ใช้ในการวัดความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยพิจารณาจากเกณฑ์ในด้านความสามารถในการทำความเข้าใจซอฟต์แวร์ (Understandability) และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ (Modifiability)

การเขียนข้อกำหนดความต้องการทางซอฟต์แวร์ (Software Requirements Specification) พบว่ามีวิธีการที่นิยมใช้อยู่ 3 วิธี ดังแสดงในรูปที่ 1.3 คือ [10]

1. แบบไม่มีแบบแผน (Informal) เป็นวิธีการแบบลักษณะมุมมองส่วนบุคคล (Personal view) คือ นักวิเคราะห์และออกแบบแต่ละคนจะมีรูปแบบในการนำเสนอข้อกำหนดความต้องการทางซอฟต์แวร์ที่แตกต่างกัน บางคนอาจนำเสนอด้วยรูปภาพ บางคนอาจนำเสนอด้วยภาษาธรรมชาติ (Natural language) อาจทำให้เกิดความคลุมเครือ และทำให้นักวิเคราะห์และออกแบบระบบเกิดความเข้าใจระบบไม่ตรงกันได้

2. แบบกึ่งแบบแผน (Semi formal) เป็นวิธีการที่ใช้แผนภาพแสดงข้อกำหนดความต้องการทางซอฟต์แวร์ เช่น อีอาร์ดี (Entities Relationship Diagram - ERD) และภาษายูเอ็มแอล (Unified Modeling Language - UML) ทำให้สามารถมองเห็นภาพรวมของระบบได้ชัดเจนขึ้น ซึ่งนักวิเคราะห์และออกแบบมีความเข้าใจที่ตรงกันมากกว่าวิธีแรก

3. แบบมีแบบแผน (Formal) เป็นวิธีการออกแบบที่ใกล้เคียงกับภาษาที่ใช้ในการโปรแกรม วิธีการนี้ทำให้นักวิเคราะห์และออกแบบมีความเข้าใจข้อกำหนดความต้องการทางซอฟต์แวร์ที่ตรงกันมากกว่าสองวิธีข้างต้น แต่ตัวภาษามีความเข้าใจยากและซับซ้อนในการออกแบบให้ตรงตามหลักการออกแบบ ภาษาที่นิยมใช้ ได้แก่ วีดีเอ็ม (Vienna Development Method – VDM) และภาษาเซด (Z language)



รูปที่ 1.3 แสดงมิติ 3 แบบของความต้องการทางซอฟต์แวร์ [10]

งานวิจัยนี้ทำการประเมินคุณภาพของซอฟต์แวร์ด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยพิจารณาจากเกณฑ์ความสามารถในการทำความเข้าใจซอฟต์แวร์ และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ ซึ่งสอดคล้องกับงานวิจัย [6,11] และใช้วิธีการเขียนข้อกำหนดความต้องการทางซอฟต์แวร์แบบกึ่งแบบแผนตามวิธีการออกแบบเชิงวัตถุ โดยใช้ภาษายูเอ็มแอลที่นำเสนอด้วยแผนภาพคลาส (Class diagram) และแผนภาพซีเควนซ์ (Sequence diagram) พร้อมทั้งสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์จากมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design Metrics) ที่คำนวณจากแผนภาพคลาสและแผนภาพซีเควนซ์ ด้วยวิธีการวิเคราะห์จำแนกกลุ่ม (Discriminant analysis) โมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ สามารถนำมาใช้วัดซอฟต์แวร์ในช่วงการวิเคราะห์และออกแบบระบบ ซึ่งเป็นขั้นตอนต้นๆ ของการพัฒนาซอฟต์แวร์ ทำให้ทราบถึงคุณภาพของซอฟต์แวร์ในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ได้เร็วขึ้น และช่วยให้ผู้พัฒนาซอฟต์แวร์สามารถแก้ไขโมเดลการวิเคราะห์และออกแบบระบบให้มีคุณภาพตามต้องการก่อนนำไปพัฒนาเป็นซอฟต์แวร์ต่อไป พร้อมทั้งสร้างเครื่องมือสำหรับคำนวณมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่สร้างบนพื้นฐานของมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุและโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ เพื่อความสะดวกในการคำนวณค่ามาตรวัด จึงต้องแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ให้อยู่ในรูปของภาษาเอ็กซ์เอ็มแอล (Extensible Markup Language - XML) ที่นำเสนอข้อมูลของแผนภาพทั้งสองด้วยข้อความ (Text-based) ก่อน เพื่อเป็นข้อมูลนำเข้าสำหรับเครื่องมือที่พัฒนาขึ้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อหาความสัมพันธ์ระหว่างมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุที่ได้จากแผนภาพคลาสและแผนภาพซีเควนซ์และความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยพิจารณาคุณลักษณะย่อยตามความสามารถในการทำความเข้าใจซอฟต์แวร์ และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ พร้อมทั้งสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยวิธีการวิเคราะห์จำแนกกลุ่ม

1.2.2 เพื่อพัฒนาเครื่องมือสำหรับคำนวณค่ามาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

1.3 ขอบเขตของการวิจัย

1.3.1 ศึกษาเกณฑ์ด้านความสามารถในการทำความเข้าใจซอฟต์แวร์และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ซึ่งเป็นเกณฑ์ย่อยของความสามารถในการบำรุงรักษาซอฟต์แวร์เท่านั้น

1.3.2 ใช้วิธีการเขียนข้อกำหนดความต้องการทางซอฟต์แวร์แบบกึ่งแบบแผนตามวิธีการออกแบบเชิงวัตถุ โดยใช้ภาษายูเอ็มแอลที่นำเสนอด้วยแผนภาพคลาสและแผนภาพซีเควนซ์

1.3.3 แผนภาพคลาสและแผนภาพซีเควนซ์ที่นำมาใช้งานวิจัย ต้องสร้างจากโปรแกรมเรชันเนลโรส (Rational Rose)

1.3.4 แผนภาพคลาสและแผนภาพซีเควนซ์ของระบบที่ใช้เป็นชุดข้อมูลสอน (Training data set) และชุดข้อมูลทดสอบ (Test data set) มีขนาดตั้งแต่ 5 คลาสขึ้นไป

1.3.5 เครื่องมือที่ใช้ในการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์เป็นเอกสารเอ็กซ์เอ็มแอล คือ โปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล (Unisys Rose XML Tool) [15]

1.3.6 ในการพัฒนาเครื่องมือใช้เครื่องไมโครคอมพิวเตอร์ที่มีระบบปฏิบัติการวินโดวส์

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษาการวิเคราะห์และออกแบบระบบด้วยภาษายูเอ็มแอล และวิธีใช้งานโปรแกรมสำหรับสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ ด้วยโปรแกรมเรชันเนลโรส

1.4.2 ศึกษาภาษาเอ็กซ์เอ็มแอล

1.4.3 ศึกษาวิธีการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์เป็นภาษาเอ็กซ์เอ็มแอลโดยใช้โปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล

1.4.4 ศึกษามาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุที่มีผลกระทบต่อความสามารถในการบำรุงรักษาโมเดลการวิเคราะห์และออกแบบระบบด้วยภาษายูเอ็มแอล

1.4.5 พัฒนาเครื่องมือด้วยภาษาจาวา เพื่อคำนวณหามาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุที่นำมาใช้ในการทดลอง

1.4.6 ออกแบบการทดลองเพื่อตรวจสอบมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุที่มีผลต่อความสามารถในการบำรุงรักษาโมเดลการวิเคราะห์และออกแบบระบบด้วยภาษายูเอ็มแอล

1.4.7 เก็บรวบรวมข้อมูลที่ได้จากการทดลอง

1.4.8 วิเคราะห์ข้อมูลและสรุปผลการทดลอง

1.4.9 ปรับปรุงเครื่องมือที่พัฒนาในขั้นตอนที่ 1.4.5 เพื่อให้สามารถวัดความสามารถในการบำรุงรักษาโมเดลการวิเคราะห์และออกแบบระบบด้วยภาษายูเอ็มแอลได้

1.4.10 จัดทำรายงานวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์จากโมเดลการวิเคราะห์และออกแบบระบบด้วยภาษายูเอ็มแอลได้

1.5.2 ได้มาตรวัดการวิเคราะห์และออกแบบระบบเชิงวัตถุที่มีความสัมพันธ์กับความสามารถในการบำรุงรักษาซอฟต์แวร์

1.5.3 ได้เครื่องมือสำหรับคำนวณมาตรวัดการวิเคราะห์และออกแบบเชิงวัตถุและวัดความสามารถในการบำรุงรักษาซอฟต์แวร์แบบอัตโนมัติ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องในงานวิจัยการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้มาตรวัดการวิเคราะห์และออกแบบระบบเชิงวัตถุแบบยูเอ็มแอล ได้แก่ ภาษายูเอ็มแอล การวัดซอฟต์แวร์เชิงวัตถุ ภาษาเอ็กซ์เอ็มแอล และสถิติสำหรับการวิจัย ซึ่งมีรายละเอียดดังต่อไปนี้

2.1.1 ภาษายูเอ็มแอล [3]

ภาษาที่ใช้ในการออกแบบในปัจจุบันที่นักพัฒนาซอฟต์แวร์นิยมใช้กัน คือ ภาษายูเอ็มแอล ซึ่งเป็นภาษามาตรฐานที่กำหนดขึ้นโดยโอเอ็มจี (Object Management Group - OMG) ภาษายูเอ็มแอลนำมาใช้เขียนโมเดลการวิเคราะห์และออกแบบระบบ ซึ่งมีแผนภาพทั้งหมด 9 แผนภาพ [3] ที่สำคัญ แบ่งออกเป็นแผนภาพเชิงโครงสร้าง (Structural diagrams) และแผนภาพเชิงพฤติกรรม (Behavioral diagrams) แผนภาพเชิงโครงสร้าง ได้แก่ แผนภาพคลาส แผนภาพวัตถุ (Object diagram) แผนภาพคอมโพเนนท์ (Component diagram) และแผนภาพดีพลอยเมนต์ (Deployment diagram) แผนภาพเชิงพฤติกรรม ได้แก่ แผนภาพยูสเคส (Use case diagram) แผนภาพซีเควนซ์ แผนภาพคอแลบอเรชั่น (Collaboration diagram) แผนภาพสเตทชาร์ต (Statechart diagram) และแผนภาพแอคทิวิตี (Activity diagram) งานวิจัยนี้ศึกษาเฉพาะแผนภาพคลาส และแผนภาพซีเควนซ์ ซึ่งรายละเอียดของทั้งสองแผนภาพมีดังนี้

2.1.1.1 แผนภาพคลาส

แผนภาพคลาสเป็นแผนภาพที่ใช้ในการวิเคราะห์และออกแบบระบบ และเป็นแผนภาพเชิงโครงสร้าง เพื่อให้ผู้ใช้งานสามารถมองเห็นองค์ประกอบทั้งหมดของระบบ ส่วนประกอบของคลาสที่สำคัญประกอบด้วย 2 ส่วน คือ โครงสร้างหลัก (Structure) และความสัมพันธ์ (Relation)

รูปที่ 2.1 แสดงโครงสร้างหลักของแผนภาพคลาส

จากรูปที่ 2.1 โครงสร้างหลักของแผนภาพคลาส ประกอบด้วย

1) ชื่อคลาส (Class name) แสดงชื่อของสิ่งที่เราสนใจในระบบ

2) คุณลักษณะ (Attribute) แสดงถึงคุณลักษณะของคลาส

3) เมทธอด (Method หรือ Operation) แสดงกิจกรรมหรือการทำงานของคลาส

ความสัมพันธ์ของแผนภาพคลาสมีทั้งหมด 5 แบบ ได้แก่

1) ความสัมพันธ์แบบแอสโซซิเอชัน (Association) ใช้เพื่ออธิบายความสัมพันธ์ระหว่างคลาสสองคลาส ซึ่งมีอินสแตนท์ (Instance) ของคลาสที่ได้ทำการส่งสาร (Message) ระหว่างกัน สัญลักษณ์ของความสัมพันธ์แบบแอสโซซิเอชัน แสดงด้วยเส้นทึบไม่มีหัวลูกศร ดังแสดงในตารางที่ 2.1

2) ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization) เป็นความสัมพันธ์ระหว่างคลาสที่มีลักษณะทั่วไป (เรียกว่าซุปเปอร์คลาส (Superclass) หรือแพเรนท์คลาส (Parent class) หรือ คลาสแม่) และคลาสที่มีลักษณะเฉพาะ (เรียกว่าสับคลาส (Subclass) หรือไชลคลาส (Child class) หรือ คลาสลูก) หรือเรียกอีกอย่างหนึ่งว่าความสัมพันธ์แบบอีสอะ (Is-a) ในกรณีนี้คลาสลูกใช้โครงสร้างและพฤติกรรมร่วมกับคลาสแม่ สัญลักษณ์ของความสัมพันธ์แบบเจเนอรัลไลเซชันแสดงด้วยเส้นทึบหัวลูกศรกลวงตรงกลางชี้ไปยังคลาสแม่ ดังแสดงในตารางที่ 2.1

3) ความสัมพันธ์แบบแอกกริเกชัน (Aggregation) และคอมโพซิชัน (Composition) เป็นความสัมพันธ์พิเศษชนิดหนึ่งของความสัมพันธ์แบบแอสโซซิเอชัน แสดงความสัมพันธ์ระหว่างคลาสที่แสดงสิ่งที่ใหญ่กว่า (The “whole”) ซึ่งประกอบด้วยคลาสที่เล็กกว่า (The “parts”) หรือเรียกอีกอย่างหนึ่งว่าความสัมพันธ์แบบแฮสอะ (Has-a) ความสัมพันธ์แบบคอมโพซิชันจะมีการแสดงความเป็นเจ้าของสูงกว่าความสัมพันธ์แบบแอกกริเกชัน กล่าวคือเมื่อคลาสที่ใหญ่กว่าถูกลบคลาสที่เล็กกว่าจะถูกลบด้วยทันที แต่ถ้าเป็นความสัมพันธ์แบบแอกกริเกชันเมื่อคลาสที่ใหญ่กว่าถูกลบคลาสที่เล็กกว่าจะไม่ถูกลบด้วย เนื่องจากมีความเป็นอิสระต่อกัน สัญลักษณ์ของความสัมพันธ์แบบแอกกริเกชัน และคอมโพซิชัน แสดงด้วยเส้นทึบหัวรูปขนมเปียกปูนกลวงและทึบตามลำดับ ดังแสดงในตารางที่ 2.1

4) ความสัมพันธ์แบบดีเพนเดนซี (Dependency) เป็นความสัมพันธ์ที่ใช้แสดงในกรณีที่คลาสหนึ่งมีการเปลี่ยนแปลงแล้วมีผลกระทบกับคลาสอื่นด้วย สัญลักษณ์ของความสัมพันธ์แบบดีเพนเดนซี แสดงด้วยเส้นประแบบมีทิศทาง (Directed line) ดังแสดงในตารางที่ 2.1

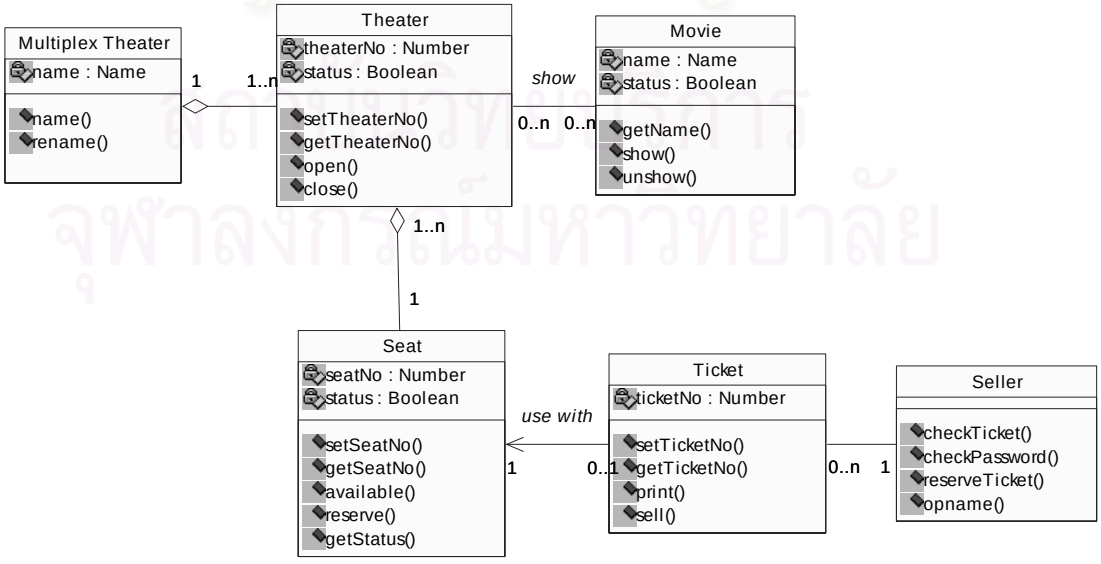
5) ความสัมพันธ์แบบเรียลไรเซชัน (Realization) ใช้เชื่อมระหว่างส่วนติดต่อผู้ใช้งาน (Interface) และคลาสหรือคอมโพเนนท์ ความสัมพันธ์แบบเรียลไรเซชัน แสดงด้วยสัญลักษณ์เส้นประหัวลูกศรกลวงตรงกลาง ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แสดงสัญลักษณ์ของความสัมพันธ์ทั้ง 5 แบบของแผนภาพคลาส

ความสัมพันธ์	สัญลักษณ์
แอสโซซิเอชัน	
เจเนอรัลไลเซชัน	
แอกกริเกชันและคอมโพซิชัน	
ดีเพนเดนซี	
เรียลไรเซชัน	

จากส่วนประกอบต่างๆ ของแผนภาพคลาส สามารถนำมารวมกันเป็นแผนภาพคลาสแสดงดังรูปที่ 2.2 ซึ่งเป็นตัวอย่างแผนภาพคลาสของระบบซอฟต์แวร์สำหรับโรงภาพยนตร์มัลติเพล็กซ์ ประกอบด้วยคลาสทั้งหมด 6 คลาส ได้แก่ คลาส Multiplex Theater คลาส Theater คลาส Seat คลาส Movie คลาส Ticket และคลาส Seller ที่มีความสัมพันธ์กัน

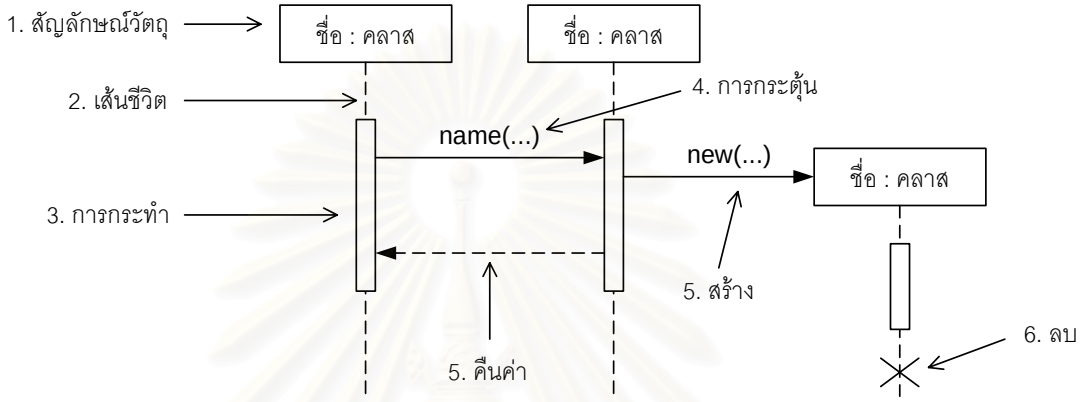
คลาส Multiplex Theater และคลาส Theater มีความสัมพันธ์แบบแอกกริเกชัน หมายความว่า โรงภาพยนตร์แบบ Multiplex ประกอบไปด้วยโรงภาพยนตร์หลายๆ โรงภาพยนตร์ คลาส Theater และคลาส Movie มีความสัมพันธ์แบบแอสโซซิเอชัน หมายความว่า โรงภาพยนตร์หนึ่งๆ ทำการฉายภาพยนตร์ คลาส Theater และ คลาส Seat มีความสัมพันธ์แบบแอกกริเกชัน หมายความว่า โรงภาพยนตร์แต่ละโรงภาพยนตร์ประกอบไปด้วยที่นั่งสำหรับผู้ที่มาชมภาพยนตร์ คลาส Seat และคลาส Ticket มีความสัมพันธ์แบบแอสโซซิเอชัน หมายความว่า ที่นั่งจะมีตั๋วประจำแต่ละที่ และสุดท้ายคลาส Ticket และคลาส Seller มีความสัมพันธ์แบบแอสโซซิเอชัน หมายความว่า เจ้าหน้าที่ขายตั๋วทำการขายตั๋วสำหรับชมภาพยนตร์



รูปที่ 2.2 ตัวอย่างแผนภาพคลาสแสดงซอฟต์แวร์สำหรับระบบโรงภาพยนตร์มัลติเพล็กซ์

2.1.1.2 แผนภาพซีเควนซ์

แผนภาพซีเควนซ์เป็นแผนภาพหนึ่งที่ออกแบบด้วยภาษายูเอ็มแอลสำหรับใช้ในการวิเคราะห์และออกแบบระบบเป็นแผนภาพเชิงพฤติกรรม ประกอบไปด้วยกลุ่มของวัตถุ (Object) ความสัมพันธ์ และสารที่ส่งระหว่างวัตถุ [3] โดยแผนภาพซีเควนซ์จะเน้นที่การลำดับเวลาของสารที่ส่งระหว่างกัน



รูปที่ 2.3 แสดงส่วนประกอบของแผนภาพซีเควนซ์

ส่วนประกอบของแผนภาพซีเควนซ์แสดงดังรูปที่ 2.3 ประกอบด้วย

1) สัญลักษณ์วัตถุ (Object symbol) แสดงถึงวัตถุที่ถูกสร้างขึ้นจากคลาสที่ประกาศอยู่หลังเครื่องหมาย ":"

2) เส้นชีวิต (Lifeline) แสดงถึงช่วงระยะเวลาการมีชีวิตอยู่ของวัตถุ

3) การกระทำ (Activation) แสดงช่วงระยะเวลาที่วัตถุกำลังดำเนินการอยู่

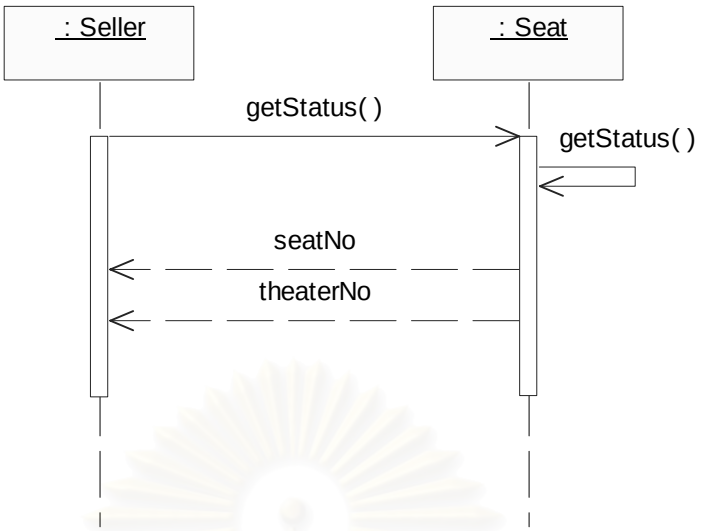
4) การกระตุ้น (Stimulus) แสดงถึงสารที่ส่งระหว่างวัตถุ

5) คืนค่า (Return) แสดงการย้อนกลับไปทำงานยังวัตถุเดิม

6) ลบ (Delete) แสดงจุดสิ้นสุดของเส้นชีวิต โดยได้รับสารที่เป็นการลบวัตถุนั้น

7) สร้าง (Create) แสดงการเริ่มต้นสร้างวัตถุ และเส้นชีวิต ที่ถูกสร้างขึ้นมาใหม่

จากส่วนประกอบต่างๆ เมื่อนำมารวมกันเป็นแผนภาพซีเควนซ์สามารถแสดงได้ดังตัวอย่างในรูปที่ 2.4 แผนภาพซีเควนซ์ของซอฟต์แวร์สำหรับระบบโรงภาพยนตร์มัลติเพล็กซ์แสดงการจองที่นั่งในโรงภาพยนตร์ โดยวัตถุของคลาส Seller ส่งสารที่ชื่อว่า getStatus() ไปยังวัตถุของคลาส Seat เพื่อเช็คสถานะของวัตถุของคลาส Seat ว่ามีการจองที่นั่งแล้วหรือยัง จากนั้นวัตถุของคลาส Seat ทำการเรียกสารภายในคลาส Seat ที่ชื่อว่า getStatus() แล้วส่งข้อมูล seatNo และ theaterNo ซึ่งเป็นค่าของเลขที่นั่งและเลขที่โรงภาพยนตร์ มาให้วัตถุของคลาส Seller



รูปที่ 2.4 ตัวอย่างของแผนภาพซีเควนซ์ แสดงการจองที่นั่งในโรงภาพยนตร์มัลติเพล็กซ์

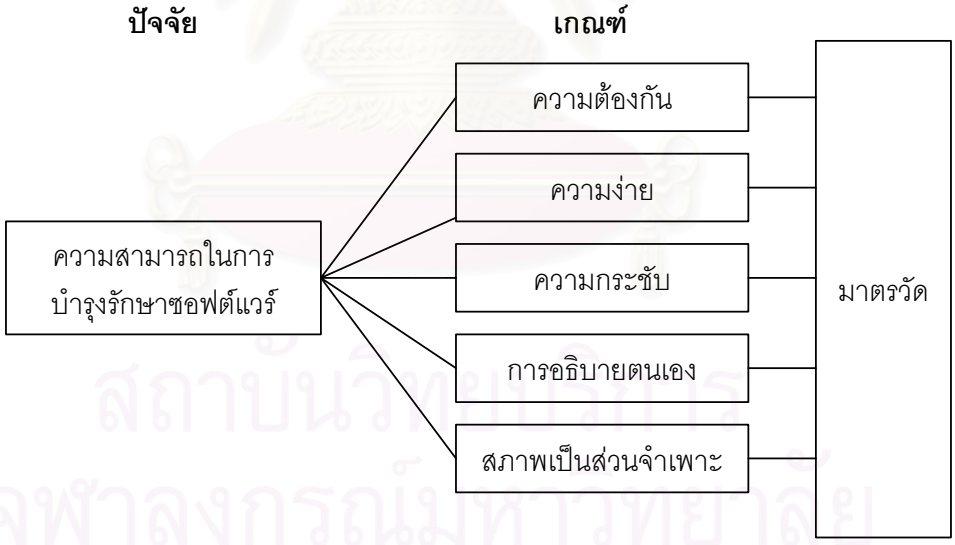
2.1.2 **การวัดซอฟต์แวร์เชิงวัตถุ** (Object-Oriented Software Measurement)

การวัดเป็นกระบวนการกำหนดค่าที่เป็นตัวเลขหรือสัญลักษณ์ให้กับคุณลักษณะของสิ่งที่เราสนใจ (Entities) [5] การวัดสิ่งต่างๆ สามารถวัดได้ 2 วิธี คือ การวัดทางตรง (Direct measure) เป็นวิธีการวัดที่ใช้วัดคุณลักษณะภายใน (Internal attribute) ของสิ่งที่เราสนใจ เช่น การวัดขนาดของซอฟต์แวร์ สามารถวัดได้จากการนับจำนวนบรรทัดทั้งหมดในตัวโปรแกรม วิธีการที่สองคือ การวัดทางอ้อม (Indirect measure) เป็นวิธีการที่ใช้วัดคุณลักษณะภายนอก (External attribute) แต่เนื่องจากการวัดคุณลักษณะภายนอก เช่น การวัดความสามารถในการใช้งาน (Usability) ความสามารถในการทดสอบ และความสามารถในการบำรุงรักษาซอฟต์แวร์ ไม่สามารถหาค่าได้โดยตรง จึงต้องอาศัยการวัดทางตรงมาช่วยในการคำนวณค่า การวัดทั้งสองวิธีจำเป็นต้องใช้มาตรวัด เพื่อวัดคุณลักษณะของสิ่งที่สนใจหรือแสดงลักษณะเฉพาะตัวของซอฟต์แวร์เหล่านั้น

มาตรวัดซอฟต์แวร์ถูกกำหนดขึ้นมาเพื่อช่วยในการวัดคุณภาพของซอฟต์แวร์ในด้านต่างๆ ในปัจจุบันได้มีนักวิจัยหลายท่านที่คิดค้นและนำเสนอมาตรวัดใหม่ๆ โดยมาตรวัดที่เป็นที่นิยมใช้กันอย่างแพร่หลาย และเป็นที่ยอมรับต่างๆ เหล่านี้สามารถแบ่งได้เป็น 2 ประเภท ได้แก่ มาตรวัดแบบดั้งเดิม (Traditional metrics) เช่น จำนวนบรรทัดของโปรแกรม (Lines Of Code – LOC) และค่าวัดไซโคลเมติกคอมเพล็กซิตี้ของแมคเคบ (Cyclomatic Complexity – CC) สำหรับวัดค่าความซับซ้อนของโปรแกรม และมาตรวัดเชิงวัตถุ ได้แก่ ระดับความลึกของการสืบทอดคุณสมบัติของคลาส (Depth of Inheritance Tree - DIT) จำนวนคลาสลูก (Number Of Children – NOC) และขนาดความสัมพันธ์ระหว่างวัตถุ (Coupling Between Objects – CBO) หลังจากหา

ค่ามาตรวัดต่างๆ ได้แล้วจะต้องนำมาตรวัดเหล่านี้มาทำการตรวจสอบ (Validation)ว่าค่ามาตรวัดที่ได้มีความสมเหตุสมผล (Validity) ความน่าเชื่อถือ (Reliability) สามารถนำไปใช้วัดคุณภาพซอฟต์แวร์ต่อไปได้

ปัจจุบันมีการนำมาตรวัดซอฟต์แวร์ไปใช้งานในด้านการวัดผลิตผลของซอฟต์แวร์ การประมาณค่าใช้จ่าย และกำลังคน หรืออธิบายคุณภาพของซอฟต์แวร์ในด้านความสามารถในการนำกลับมาใช้ใหม่ (Reusability) ความน่าเชื่อถือ และความสามารถในการบำรุงรักษาซอฟต์แวร์ เพื่อให้ทราบถึงความสามารถของซอฟต์แวร์ในด้านต่างๆ แต่เนื่องจากคุณภาพแต่ละตัวสามารถประกอบไปด้วยคุณลักษณะย่อยหลายตัว จึงมีนักวิจัยนำเสนอโมเดลสำหรับวัดคุณภาพซอฟต์แวร์ โดยโมเดลที่นิยมใช้ คือ โมเดลคุณภาพของแมคคอล (McCall) และโบห์ม (Boehm) รูปที่ 2.5 แสดงส่วนของโมเดลคุณภาพของแมคคอลด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ แสดงให้เห็นว่าปัจจัยคุณภาพของซอฟต์แวร์ขึ้นกับเกณฑ์หลายแบบ ซึ่งแต่ละแบบสามารถหาค่าได้จากมาตรวัด เช่น ปัจจัยของคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ของโมเดลแมคคอล ได้กำหนดเกณฑ์ไว้ 5 ปัจจัยด้วยกัน คือ ความต้องกัน (Consistency) ความง่าย (Simplicity) ความกระชับ (Conciseness) การอธิบายตนเอง (Self-descriptiveness) และสภาพเป็นส่วนจำเพาะ (Modularity)



รูปที่ 2.5 แสดงส่วนของโมเดลคุณภาพของแมคคอลด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ [5]

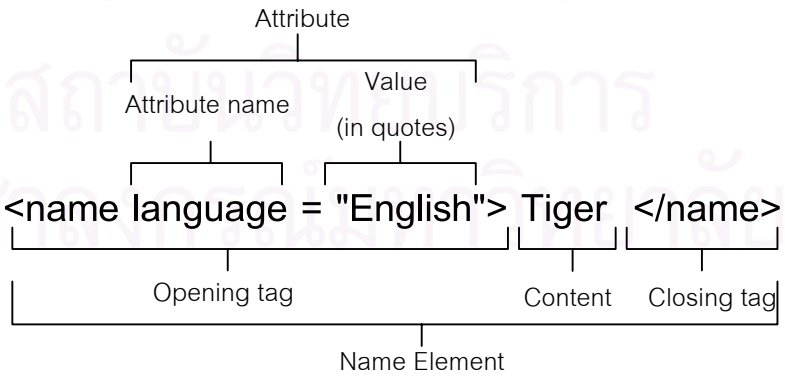
2.1.3 ภาษาเอ็กซ์เอ็มแอล [17]

เอ็กซ์เอ็มแอลเป็นภาษาที่ถูกออกแบบเพื่อใช้อธิบายข้อมูลของวัตถุ และพฤติกรรม (Behavior) ของโปรแกรม โดยเป็นภาษาที่มีพื้นฐานจากเอสจีเอ็มแอล (Standard

Generalized Markup Language - SGML) เอ็กซ์เอ็มแอลประกอบด้วยส่วนประกอบสองส่วน คือ คำอธิบายชนิดของเอกสารเอ็กซ์เอ็มแอล ที่เรียกว่า ดีทีดี (Document Type Definition - DTD) ที่ทำหน้าที่กำหนดไวยากรณ์ของเอกสารเอ็กซ์เอ็มแอล ส่วนที่สอง คือ เนื้อหาของเอกสารเอ็กซ์เอ็มแอล (XML document) ซึ่งเป็นโครงสร้างทางตรรกะ (Logical structure) สำหรับอธิบายคุณลักษณะต่างๆ ให้สอดคล้องกับดีทีดีในรูปของโครงสร้างลำดับชั้น (Hierarchy) การที่เอ็กซ์เอ็มแอลมีการแบ่งส่วนของดีทีดี และเนื้อหาของเอกสารเอ็กซ์เอ็มแอลแยกออกจากส่วนของการแสดงผล ทำให้ได้เอกสารที่มีเค้าร่าง (Schema) เป็นโครงสร้างที่ง่ายต่อการนำข้อมูลในเอกสารไปใช้งาน และนำกลับไปใช้ใหม่ได้หลายๆ ครั้ง ภายใต้สื่อต่างชนิดกัน

เอ็กซ์เอ็มแอลเป็นเอกสารที่มีความยืดหยุ่นสำหรับงานประยุกต์ที่มีพื้นฐานบนเว็บ และมีรูปแบบการนำเสนอในรูปแบบข้อความ จึงไม่ขึ้นกับระบบปฏิบัติการ และสถาปัตยกรรมของคอมพิวเตอร์ และมีแนวโน้มที่จะเป็นมาตรฐานใหม่ในระบบเปิด ขณะเดียวกันแท็ก (Tag) ของเอกสารที่เป็นข้อความสามารถแสดงคำอธิบายเชิงความหมายได้ ทำให้เอ็กซ์เอ็มแอลมีความยืดหยุ่นในการเขียนเมตาดาต้า (Metadata) เพื่อการจัดการข้อมูล

องค์กรดับเบิลยูทรีซี (World Wide Web Consortium – W3C) ได้เสนอแนวคิดในการกำหนดมาตรฐานของเอกสารบนเว็บขึ้น คือ ภาษาเอสจีเอ็มแอล แต่เอสจีเอ็มแอลมีรายละเอียดมากและซับซ้อนและใช้เวลาในการศึกษานานจึงเกิดภาษาเอ็กซ์เอ็มแอลขึ้น ซึ่งเป็นภาษาที่รวมเอาข้อดีของภาษาเอสจีเอ็มแอล และเอชทีเอ็มแอล (Hyper Text Markup Language – HTML) เข้าไว้ด้วยกัน เนื่องจากภาษาเอ็กซ์เอ็มแอลเป็นไฟล์ข้อความ ทำให้เอกสารเอ็กซ์เอ็มแอลเป็นภาษากลางที่ใช้ได้ในทุกแพลตฟอร์ม (Platform) และเป็นข้อดีของภาษานี้ในการเขียนโปรแกรมประยุกต์ด้านการจัดการข้อมูลต่างๆ



รูปที่ 2.6 แสดงองค์ประกอบของอิลิเมนต์ที่ชื่อ name

จากรูปที่ 2.6 แสดงองค์ประกอบต่างๆ ที่สำคัญภายในเอกสารเอ็กซ์เอ็มแอล ได้แก่ ชื่ออิลิเมนต์ (Element name) แท็กเปิด (Opening tag) แท็กปิด (Closing tag) คุณลักษณะ

ชื่อคุณลักษณะ (Attribute name) ค่าของคุณลักษณะ (Attribute value) และส่วนของเนื้อหา (Content) ซึ่งกฎในการเขียนเอกสารเอ็กซ์เอ็มแอล มีดังนี้

1. ต้องทำการประกาศเวอร์ชันของเอกสารเอ็กซ์เอ็มแอลในบรรทัดแรกทุกครั้ง

```
<?xml version = “1.0”?>
```

2. ภายในเอกสารเอ็กซ์เอ็มแอลจะต้องมีอิลิเมนต์ราก (Root element)

```
<endangered_species>
    <name>Tiger</name>
</endangered_species>
```

3. ทุกแท็กภายในเอกสารเอ็กซ์เอ็มแอลจะต้องมีแท็กปิด

```
<endangered_species>
  <name> Tiger </name>
<picture filename=“tiger.jpg”/>
</endangered_species>
```

4. อิลิเมนต์จะต้องมีการเรียงซ้อนกันเป็นคู่

5. ชื่ออิลิเมนต์ตัวใหญ่ตัวเล็กจะมีความหมายไม่เหมือนกัน (Case sensitive)

```
<name> Tiger </name>
<Name> Tiger </Name>
```

ถูก

```
<name> Tiger </Name>
```

ผิด

6. ค่าข้อมูลของคุณลักษณะจะต้องอยู่ในเครื่องหมายอัญประกาศ (“) เท่านั้น

```
<picture filename=“tiger.jpg”/>
```

7. เอนทิตีใดๆ ที่ถูกอ้างถึงในเอกสารเอ็กซ์เอ็มแอล จะต้องมีการประกาศไว้ที่ส่วนของดีทีดีก่อนเรียกใช้งาน

วิธีการเข้าถึงเอกสารเอ็กซ์เอ็มแอลต้องอาศัยตัวแจงส่วนโครงสร้างของเอกสารเอ็กซ์เอ็มแอล (XML parser) ซึ่งมีวิธีการดึงข้อมูลจากเอกสารเอ็กซ์เอ็มแอลหลายวิธี แต่ที่นิยมกันมากมี 2 วิธี [17] คือ

2.1.3.1 **ดอม** (Document Object Model - DOM) [17]

ดอมย่อมาจาก Document Object Model มีหลักการในการอ่านเอกสารเอ็กซ์เอ็มแอล มาวางเป็นต้นไม้ (Tree) ในหน่วยความจำ (Memory) ของเครื่องคอมพิวเตอร์ที่กำลังทำงาน โครงสร้างต้นไม้ที่ได้จะประกอบไปด้วยอิลิเมนต์ หรือคุณลักษณะต่างๆ ดังนั้นการเข้าถึงข้อมูลที่อยู่ในรูปแบบต้นไม้ต้องอาศัยการท่อง (Traverse) ไปตามกิ่งก้านของต้นไม้ วิธีการแบบนี้มีข้อจำกัดตรงที่จะทำการอ่านข้อมูลเอกสารทั้งหมดมาเก็บไว้ในหน่วยความจำเพียงครั้งเดียวทำให้เปลืองพื้นที่ในหน่วยความจำของเครื่องคอมพิวเตอร์ แต่ข้อดี คือ เขียนโค้ดได้ง่าย

2.1.3.2 **แซก** (Simple API for XML - SAX) [17]

แซกย่อมาจาก Simple API for XML ขั้นตอนการทำงานของแซก จะไม่โหลดข้อมูลทั้งหมดของเอกสารเอ็กซ์เอ็มแอล เข้ามาไว้ในหน่วยความจำแบบวิธีการของดอม แต่จะทำการอ่านเอกสารเอ็กซ์เอ็มแอลตั้งแต่ต้นเอกสารจนจบ พร้อมทั้งสร้างเหตุการณ์ (Event) ไว้ที่อิลิเมนต์แต่ละอิลิเมนต์ภายในเอกสาร เพื่อใช้สำหรับดึงข้อมูลภายในเอกสารเอ็กซ์เอ็มแอล ในการดึงข้อมูลจะอ่านข้อมูลเริ่มต้นตั้งแต่อิลิเมนต์รากไปถึงอิลิเมนต์สุดท้ายภายในเอกสารเอ็กซ์เอ็มแอล เรียกวิธีการแบบนี้ว่า อีเวนต์ดริฟเวน (Event-driven) การใช้งานแบบแซกเหมาะสำหรับการค้นหาข้อมูลในปริมาณน้อย

2.1.4 **สถิติสำหรับการวิจัย** (Statistics for Research) [1, 2]

สำหรับการวิจัยนั้น เมื่อผู้วิจัยได้ค้นคว้าเก็บรวบรวมข้อมูลดิบที่ได้จากการทดลองมาแล้วต้องนำข้อมูลเหล่านั้นมาวิเคราะห์ และแปลความหมาย วิธีการทางสถิติได้นำมาใช้ในการวิเคราะห์ข้อมูลดิบเหล่านั้น โดยใช้วิธีการต่างๆ เช่น การแจกแจงความถี่ การหาร้อยละ ตลอดจนถึงช่วยให้ทราบเกี่ยวกับคุณลักษณะต่างๆ ของข้อมูล เช่น การวัดแนวโน้มเข้าสู่ส่วนกลาง การวัดการกระจายของข้อมูล ดังนั้นสถิติจึงมีความเกี่ยวข้องกับการวิจัยอย่างมาก นอกจากนี้ยังมีบทบาทในการนำเสนอรายงานการวิจัย เช่น การจัดทำตาราง การสร้างกราฟเส้น และแผนภูมิรูปภาพ

งานวิจัยนี้ได้นำวิธีการทางสถิติมาช่วยในการทำงานวิจัยและวิเคราะห์ข้อมูล ซึ่งวิธีการทางสถิติที่นำมาใช้ประกอบด้วย การหาค่าเฉลี่ย (Mean) การหาค่าส่วนเบี่ยงเบนมาตรฐาน (Standard deviation) การวิเคราะห์ความสัมพันธ์ (Correlation analysis) การวิเคราะห์ความถดถอย (Regression analysis) และการวิเคราะห์จำแนกกลุ่ม (Discriminant analysis) ซึ่งรายละเอียดมีดังนี้

2.1.4.1 **การหาค่าเฉลี่ย** [2]

การหาค่าเฉลี่ย เป็นวิธีการวัดแนวโน้มสู่ส่วนกลาง (Central tendency) วิธีหนึ่งซึ่งเป็นการคำนวณค่ากลางของข้อมูลว่าอยู่ที่ใด โดยจะใช้ค่าเฉลี่ยเป็นตัวแทนของข้อมูลที่นำมาคำนวณ สูตรในการคำนวณหาค่าเฉลี่ย คือ

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

โดยที่ $\bar{x}$ คือ ค่าเฉลี่ยของข้อมูลชุดใดๆ

x_i คือ ค่าข้อมูลในลำดับที่ i เมื่อ $1 \le i \le n$

n คือ จำนวนข้อมูลทั้งหมด

2.1.4.2 **การหาค่าส่วนเบี่ยงเบนมาตรฐาน** [2]

การพิจารณาหรือสรุปถึงลักษณะของข้อมูลโดยใช้ค่ากลางหรือค่าเฉลี่ยเพียงอย่างเดียวอาจไม่สามารถทราบถึงลักษณะของข้อมูลได้ชัดเจน เนื่องจากอาจมีชุดของข้อมูลที่มีค่ากลางเท่ากัน แต่ลักษณะของข้อมูลในแต่ละชุดแตกต่างกัน นั่นคือมีการกระจายของข้อมูลไม่เหมือนกัน ดังนั้นค่าส่วนเบี่ยงเบนมาตรฐาน เป็นวิธีการวัดการกระจายวิธีหนึ่งที่นิยมใช้ ซึ่งมีสูตรในการคำนวณ คือ

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

โดยที่ $SD.$ คือ ค่าเบี่ยงเบนมาตรฐาน

$\bar{x}$ คือ ค่าเฉลี่ยของข้อมูลทั้งหมด

x_i คือ ค่าข้อมูลในลำดับที่ i เมื่อ $1 \le i \le n$

n คือ จำนวนข้อมูลทั้งหมด

2.1.4.3 **การวิเคราะห์ความสัมพันธ์** [2]

การหาความสัมพันธ์ของข้อมูลจะใช้สถิติไคสแควร์ (Chi-square) สำหรับข้อมูลนามบัญญัติ (Nominal scale) การวิเคราะห์ความสัมพันธ์จัดอันดับแบบสเปียร์แมน (Spearman rank correlation) สำหรับข้อมูลอันดับ (Ordinal scale) และการวิเคราะห์ความสัมพันธ์แบบเพียร์สัน (Pearson correlation) สำหรับข้อมูลอันตรภาค (Interval scale) เนื่องจากงานวิจัยนี้มี

ลักษณะข้อมูลแบบอัตราส่วน (Ratio scale) จึงใช้วิธีการวิเคราะห์ความสัมพันธ์แบบเพียร์สัน ดังนั้นจะกล่าวถึงการวิเคราะห์เพียงแบบเดียวเท่านั้น

การวิเคราะห์ความสัมพันธ์แบบเพียร์สันเป็นการหาความสัมพันธ์ระหว่างตัวแปรเชิงปริมาณ 2 ตัว โดยมีข้อตกลงเบื้องต้นว่าตัวแปรทั้งสองมีการแจกแจงแบบปกติ และมีความสัมพันธ์กันในแบบเชิงเส้น (Linear relationship) สำหรับสถิติที่ใช้วัดความสัมพันธ์ระหว่างตัวแปรอิสระ (Independent variable) (x) และตัวแปรตาม (Dependent variable) (y) ว่ามากหรือน้อยนั้นจะเรียกว่า สัมประสิทธิ์สหสัมพันธ์ (Correlation coefficient) มีสูตรการคำนวณดังนี้

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

โดยที่ r คือ สัมประสิทธิ์สหสัมพันธ์

x_i คือ ตัวแปรตัวที่ 1 หรือตัวแปรอิสระ ลำดับที่ i เมื่อ $1 \le i \le n$

y_i คือ ตัวแปรตัวที่ 2 หรือตัวแปรตาม ลำดับที่ i เมื่อ $1 \le i \le n$

n คือ จำนวนข้อมูลทั้งหมด

ค่า r ที่ได้จากการคำนวณจะมีค่าระหว่าง -1 ถึง 1 ความหมายของค่า r คือ

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

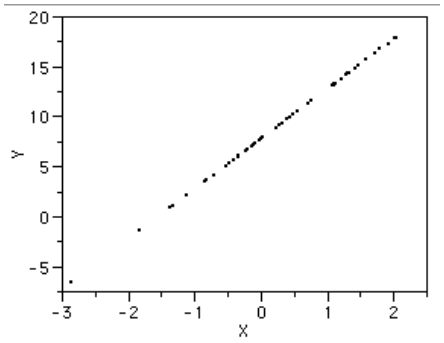
2. ค่า r เป็นบวกแสดงว่า x และ y มีความสัมพันธ์ในทิศทางเดียวกัน คือ ถ้าค่าของ x เพิ่มค่าของ y จะเพิ่มด้วย แต่ถ้าค่าของ x ลดค่าของ y จะลดด้วย

3. ถ้า r มีค่าเข้าใกล้ 1 หมายถึง x และ y สัมพันธ์ในทิศทางเดียวกันและมีความสัมพันธ์กันมาก

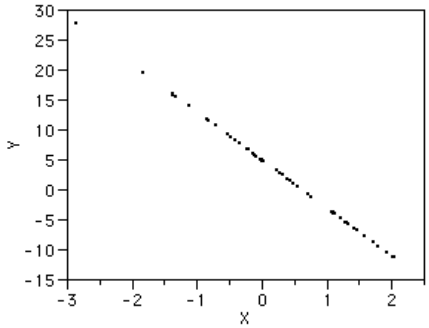
4. ถ้า r มีค่าเข้าใกล้ -1 หมายถึง x และ y สัมพันธ์ในทิศทางตรงกันข้ามและมีความสัมพันธ์กันมาก

5. ถ้า r = 0 แสดงว่า x และ y ไม่มีความสัมพันธ์กัน

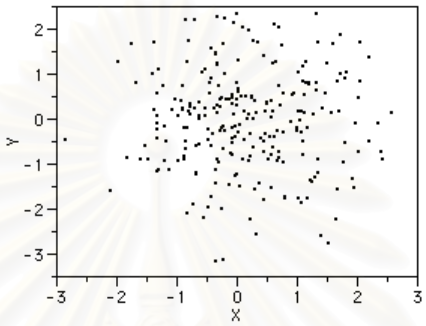
6. ถ้า r เข้าใกล้ 0 แสดงว่า x และ y มีความสัมพันธ์กันน้อย



r = 1



r = -1



r = 0

รูปที่ 2.7 แสดงค่าของ r ที่มีค่า $-1 \le x \le 1$ [16]

จากรูปที่ 2.7 แสดงกราฟความสัมพันธ์ระหว่างค่า x และ y ที่มีค่าสัมประสิทธิ์สหสัมพันธ์เป็น 1 -1 และ 0 ตามลำดับ

2.1.4.4 การวิเคราะห์การถดถอย [2]

การวิเคราะห์ความถดถอยเป็นการศึกษาถึงความสัมพันธ์ของตัวแปรตั้งแต่ 2 ตัวขึ้นไป เพื่อประมาณหรือพยากรณ์ค่าของตัวแปรหนึ่งจากตัวแปรอื่นที่มีความสัมพันธ์กันกับตัวแปรที่ต้องการประมาณนั้น วัตถุประสงค์ของการวิเคราะห์การถดถอย เพื่อศึกษาถึงความสัมพันธ์ระหว่างตัวแปร และใช้ความสัมพันธ์ที่วิเคราะห์ได้มาประมาณค่าหรือพยากรณ์ค่าตัวแปรตาม (y) ในอนาคตเมื่อกำหนดค่าตัวแปรอิสระ (x) ซึ่งความสัมพันธ์ระหว่างตัวแปร 2 ตัว จะแสดงอยู่ในรูปสมการเชิงเส้นดังนี้

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

โดยที่ y คือ ตัวแปรตาม

x คือ ตัวแปรอิสระ

β_0 คือ ส่วนตัดแกน y หรือค่าของ y เมื่อ x มีค่าเป็นศูนย์

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

e คือ ความคลาดเคลื่อนอย่างสุ่ม (Random error)

2.1.4.5 การวิเคราะห์จำแนกกลุ่ม [1]

การวิเคราะห์จำแนกกลุ่ม เป็นเทคนิคที่ทำการแบ่งกลุ่มข้อมูล หรือหน่วยตัวอย่างออกเป็นกลุ่มย่อยๆ หลายๆ กลุ่ม ซึ่งใช้หลักเกณฑ์ของการวิเคราะห์ความถดถอยเชิงพหุที่สัมพันธ์อยู่ในรูปเชิงเส้น [1,2] โดยที่ตัวแปรตามเป็นตัวแปรเชิงกลุ่ม ส่วนตัวแปรต้นเป็นตัวแปรเชิงปริมาณ โดยที่คนหรือหน่วยทดลอง (Subject) ที่อยู่ในกลุ่มเดียวกันจะต้องมีความคล้ายคลึงกัน แล้วนำตัวแปรเหล่านี้มาศึกษาหาความสัมพันธ์ที่อยู่ในรูปเชิงเส้น จากนั้นนำสมการเชิงเส้นดังกล่าวมาพยากรณ์ หรือประมาณว่าข้อมูลชุดใหม่ควรอยู่กลุ่มใด

ตัวแปรอิสระหรือตัวแปรที่ทำให้กลุ่มแตกต่างกัน จะเรียกว่าตัวแปรจำแนกกลุ่ม (Discriminator variable) ควรเป็นตัวแปรเชิงปริมาณ ซึ่งอาจมีเพียง 1 ตัวหรือตั้งแต่ 2 ตัวขึ้นไป ความสัมพันธ์ระหว่างตัวแปรตามกับตัวแปรจำแนกกลุ่มจะอยู่ในรูปเชิงเส้นดังนี้

$$D = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + e$$

โดยที่ D คือ ตัวแปรตามและเป็นข้อมูลเชิงกลุ่ม

$x_1, x_2, ..., x_n$ คือ ตัวแปรอิสระหรือตัวแปรจำแนกกลุ่ม เมื่อ $n \geq 1$

$\beta_0, \beta_1, .., \beta_n$ คือ ค่าสัมประสิทธิ์ของตัวแปรอิสระ เมื่อ $n \geq 0$

e คือ ค่าความคลาดเคลื่อน

ในการวิเคราะห์จำแนกกลุ่ม จะเรียกสมการนี้ว่าฟังก์ชันจำแนกกลุ่ม บางครั้งเรียก สมการจำแนกกลุ่มหรือเรียกกว่าฟังก์ชันการจำแนกกลุ่มของฟิชเชอร์ (Fisher Discriminant Function) ซึ่ง อาร์เอฟิชเชอร์ (R.A.Fisher) เป็นผู้คิดค้นขึ้น

2.2 งานวิจัยที่เกี่ยวข้อง

จากการศึกษาเบื้องต้นพบว่ามีงานวิจัยที่เกี่ยวข้องจำนวนทั้งสิ้น 3 งานวิจัย โดยแบ่งออกเป็นงานวิจัยที่เกี่ยวข้องกับมาตรวัดเชิงวัตถุในหัวข้อ 2.2.1 และงานวิจัยที่เกี่ยวข้องกับการทดลองด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ในหัวข้อ 2.2.2 และ 2.2.3 ซึ่งมีรายละเอียดดังต่อไปนี้

2.2.1 งานวิจัย “Developing Software Metrics Applicable to UML Models” [8]

งานวิจัยนี้ ได้นำเสนอมาตรวัดซอฟต์แวร์ (Software metrics) ใหม่ที่เรียกว่ามาตรวัดยูเอ็มแอล (UML metrics) จำนวน 27 มาตรวัด โดยวัดตามคุณลักษณะ (Characteristics) ของ

โมเดลยูเอ็มแอล ซึ่งแบ่งออกเป็น 4 ประเภท ได้แก่ มาตรวัดสำหรับโมเดล (Metrics for model) จำนวน 9 มาตรวัด มาตรวัดสำหรับคลาส (Metrics for class) จำนวน 13 มาตรวัด มาตรวัดสำหรับสาร (Metrics for message) จำนวน 2 มาตรวัด มาตรวัดสำหรับยูสเคส (Metrics for use case) จำนวน 3 มาตรวัด (รายละเอียดแสดงในภาคผนวก ก) และได้พัฒนาเครื่องมือยูเอ็มพี (UML Metrics Producer - UMP) สำหรับคำนวณหามาตรวัดทั้ง 27 มาตรวัด ด้วยวิธีการเขียนโปรแกรมพัฒนาบนโปรแกรมเรชันเนลโรส โดยใช้ภาษาเบสิกสคริป (BasicScript) และเอพีไอ (Application Programming Interface - API) ของโปรแกรมเรชันเนลโรส เพื่อดึงข้อมูลมาคำนวณหาค่ามาตรวัด และได้แบ่งวิธีการแสดงค่ามาตรวัดตามประเภทของมาตรวัดทั้ง 4 ประเภท พร้อมกับสร้างรายงานแสดงค่ามาตรวัดทั้งหมดในรูปแบบของเอกสารเอ็กซ์เอ็มแอล (รายละเอียดแสดงในภาคผนวก ก)

ผลลัพธ์ที่ได้จากงานวิจัยนี้ คือ ได้มาตรวัดยูเอ็มแอลจำนวน 27 มาตรวัด ที่สามารถคำนวณค่าได้ในขั้นการวิเคราะห์และออกแบบระบบของวงจรการพัฒนาระบบ (System Development Life Cycle: SDLC) และเครื่องมือยูเอ็มพี สำหรับคำนวณมาตรวัดทั้ง 27 ตัว พร้อมกับรายงานค่ามาตรวัดในรูปแบบเอกสารเอ็กซ์เอ็มแอล

2.2.2 **งานวิจัย** “Empirical Validation of Measures for Class Diagram Structural Complexity through Controlled Experiments” [6]

งานวิจัยนี้เป็นงานวิจัยที่ต่อเนื่องจากงานวิจัย “Early measures for UML class diagrams” [7] ซึ่งได้รวบรวมมาตรวัดเชิงวัตถุที่เกี่ยวข้องกับแผนภาพคลาสจากงานวิจัยต่างๆ ที่มีนักวิจัย 4 กลุ่มได้นำเสนอไว้ ได้แก่ มาตรวัดของ Chidamber และ Kemerer มาตรวัดของ Lorenz และ Kidd มาตรวัดของ Brito e Abreu และ Melo และมาตรวัดของ Marchesi จากนั้นนำมาตรวัดทั้งหมดมาวิเคราะห์ตามตารางการเปรียบเทียบมาตรวัดขอบข่ายของแพ็คเกจ (Package-scope metrics) และตารางการเปรียบเทียบมาตรวัดขอบข่ายของคลาส (Class-scope metrics) (รายละเอียดแสดงในภาคผนวก ค) และนำเสนอมาตรวัดเชิงวัตถุโดยพิจารณาตามความสัมพันธ์ของแผนภาพคลาสแบบแอสโซซิเอชัน แอกกริเกชัน เจเนอรัลไลเซชัน และดีเพนเดนซี (รายละเอียดแสดงในภาคผนวก ข)

งานวิจัยนี้ได้นำมาตรวัดที่ได้นำเสนอใน [7] และมาตรวัดพื้นฐาน (Traditional metrics) ได้แก่ จำนวนคลาส (Number of classes) และ จำนวนคุณลักษณะ (Number of attributes) มาตรวจสอบเชิงประสบการณ์ (Empirical validation) หาความสัมพันธ์ระหว่างความซับซ้อนของโครงสร้างแผนภาพคลาส (Class diagram structural complexity) กับความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยการศึกษาความสามารถในการบำรุงรักษาซอฟต์แวร์ในงานวิจัยนี้

ใช้โมเดลคุณภาพตามมาตรฐานไอเอสโอ 9126 แต่พิจารณาคุณลักษณะย่อยๆ ของความสามารถในการบำรุงรักษาซอฟต์แวร์เพียง 3 คุณลักษณะย่อย ได้แก่ ความสามารถในการทำความเข้าใจซอฟต์แวร์ ความสามารถในการวิเคราะห์ซอฟต์แวร์ และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ และทำการทดลองเพื่อทดสอบสมมติฐานที่ว่า “มีความเกี่ยวข้องกันระหว่างกลุ่มของมาตรวัดที่นำเสนอและการประเมินลักษณะย่อยของความสามารถในการบำรุงรักษาซอฟต์แวร์ของหน่วยตัวอย่างอย่างมีนัยสำคัญ” โดยการทดลองให้หน่วยทดลอง ทำการกำหนดระดับคุณลักษณะย่อยของความสามารถในการบำรุงรักษาซอฟต์แวร์ 7 ระดับ ดังแสดงในรูปที่ 2.8 และนำข้อมูลมาสร้างโมเดลการทำนาย (Prediction model) ความสามารถในการบำรุงรักษาซอฟต์แวร์ของแผนภาพคลาสด้วยวิธีการทางปัญญาประดิษฐ์ (Artificial Intelligent : AI) ที่มีชื่อว่า “The Fuzzy Prototypical Knowledge Discovery (FPKD)”

ยากต่อการเข้าใจมากที่สุด	ยากต่อการเข้าใจมาก	ยากต่อการเข้าใจ	ปานกลาง	ง่ายต่อการเข้าใจ	ง่ายต่อการเข้าใจมาก	ง่ายต่อการเข้าใจมากที่สุด

รูปที่ 2.8 แสดงระดับคุณลักษณะย่อยของความสามารถในการบำรุงรักษาซอฟต์แวร์ 7 ระดับ

ผลลัพธ์ที่ได้จากงานวิจัยนี้ คือ มีความสัมพันธ์กันอย่างมากระหว่างมาตรวัดที่ได้นำเสนอและความสามารถในการบำรุงรักษาของแผนภาพคลาส โดยในการวิเคราะห์หาความสัมพันธ์ใช้วิธีการของสเปียร์แมน (Spearman’s correlation) และได้โมเดลการทำนายความสามารถในการบำรุงรักษาสำหรับแผนภาพคลาส แต่งานวิจัยนี้ได้ทำการทดลองเพียงแค่ครั้งเดียว ดังนั้นเพื่อให้ผลมีความถูกต้องมากขึ้นควรมีการทำการทดลองซ้ำ

2.2.3 **งานวิจัย** “A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Design” [11]

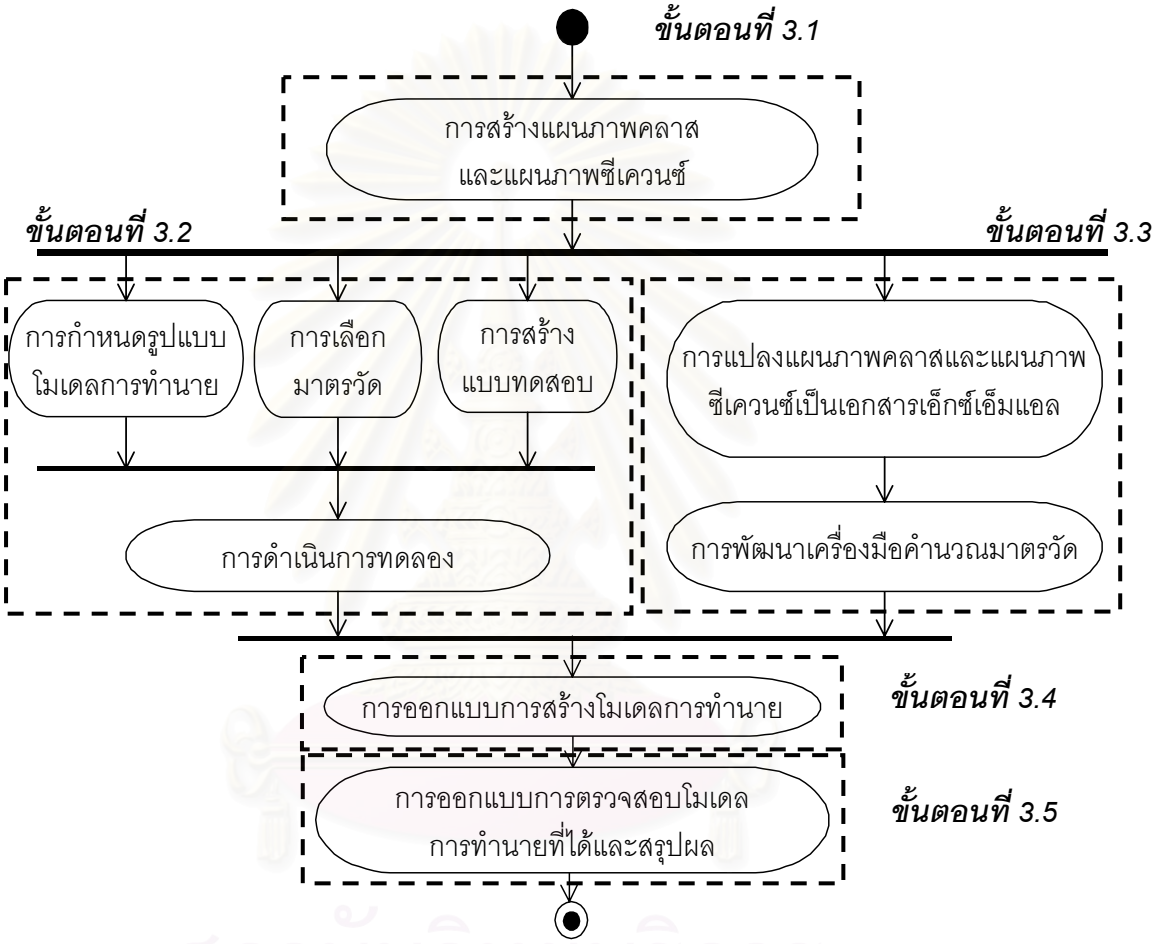
งานวิจัยนี้ทำการทดลองเพื่อดูผลกระทบของการนำเอาหลักการออกแบบเชิงคุณภาพมาประยุกต์ใช้ โดยในงานวิจัยนี้พิจารณาตามหลักการ (Principles) ของคอด (Coad) และยอร์ดอน (Yourdon) ในส่วนของความสามารถในการบำรุงรักษาการออกแบบเชิงวัตถุ การวิจัยนี้พิจารณาความสามารถในการบำรุงรักษาการออกแบบเชิงวัตถุ ในด้านความสามารถในการทำความเข้าใจ และความสามารถในการปรับเปลี่ยนซอฟต์แวร์ โดยการเปรียบเทียบผลกระทบของหลักการออกแบบในมุมมองของ “การออกแบบเชิงวัตถุที่ดี (Good design)” และ “การออกแบบเชิงวัตถุที่ไม่ดี (Bad design)” ตามหลักการของคอดและยอร์ดอน มาตรวัดที่นำมาใช้มีอยู่ด้วยกัน

6 มาตรวัด ได้แก่ มาตรวัด Und_Time มาตรวัด Und_Corr มาตรวัด Mod_Time มาตรวัด Mod_Comp มาตรวัด Mod_Corr และมาตรวัด Mod_Rate

ผลลัพธ์ที่ได้จากงานวิจัยนี้ คือ การออกแบบเชิงคุณภาพที่ทำตามหลักการของคอดและยอร์ดอนมีความง่ายต่อการเข้าใจและการแก้ไข ซึ่งทำให้ง่ายต่อการบำรุงรักษาซอฟต์แวร์ด้วย

บทที่ 3
การออกแบบการทดลองและการดำเนินการทดลอง

งานวิจัยนี้แบ่งขั้นตอนในการดำเนินงานวิจัยออกเป็น 5 ส่วน สามารถแสดงได้ด้วยแผนภาพแอคทิวิตีดังแสดงในรูปที่ 3.1



รูปที่ 3.1 แสดงแผนภาพขั้นตอนการดำเนินงานวิจัย

ขั้นตอนที่ 3.1 ของการดำเนินงานเริ่มจากการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์สำหรับระบบในชุดข้อมูลสอนและระบบในชุดข้อมูลทดสอบ ขั้นตอนที่ 3.2 แสดงขั้นตอนของการออกแบบการทดลอง ขั้นตอนที่ 3.3 แสดงขั้นตอนของการออกแบบเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ขั้นตอนที่ 3.4 แสดงขั้นตอนของการออกแบบการสร้างโมเดลการทำนาย และขั้นตอนที่ 3.5 แสดงขั้นตอนของการออกแบบการตรวจสอบโมเดลการทำนายที่ได้และสรุปผล รายละเอียดของแต่ละขั้นตอนแสดงในหัวข้อที่ 3.1 3.2 3.3 3.4 และ 3.5 ตามลำดับ

3.1 การสร้างแผนภาพคลาสและแผนภาพซีเควนซ์

ขั้นตอนนี้ทำการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ทั้ง 40 ระบบ ด้วยโปรแกรมเรชันเนลโรส โดยระบบทั้ง 40 ระบบจะแยกเป็นระบบที่ใช้เป็นชุดข้อมูลสอนจำนวน 35 ระบบเพื่อนำไปใช้สร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ และระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 5 ระบบเพื่อนำไปใช้ในการตรวจสอบความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาที่สร้างได้

3.2 การออกแบบการทดลอง

การออกแบบการทดลองนั้นจะเป็นการวางแผนการทำงาน ซึ่งจะมีรายละเอียดเกี่ยวกับการกำหนดรูปแบบโมเดลการทำนาย การเลือกมาตรวัด การเลือกหน่วยทดลอง การสร้างแบบทดสอบ และการดำเนินการทดลอง

3.2.1 การกำหนดรูปแบบโมเดลการทำนาย

งานวิจัยนี้ มีวัตถุประสงค์เพื่อสร้างโมเดลการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ของโมเดลการวิเคราะห์และออกแบบระบบด้วยยูเอ็มแอล ซึ่งโมเดลที่ได้สามารถแบ่งระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ออกเป็น 3 ระดับ คือ ระดับง่าย ปานกลาง และยาก และมีรูปแบบของโมเดลการทำนายอยู่ในรูปของสมการเชิงเส้น

$$y = a_1 x_1 + a_2 x_2 + ... + a_i x_i + ... + a_n x_n + c$$

เมื่อ ค่า y (ตัวแปรตาม) คือ ระดับความสามารถในการบำรุงรักษาซอฟต์แวร์

ค่า x_i (ตัวแปรต้น) คือ มาตรวัดที่มีความสัมพันธ์กับความสามารถในการบำรุงรักษาซอฟต์แวร์ เมื่อ $1 \le i \le n$

ค่า a_i คือ สัมประสิทธิ์ของมาตรวัดแต่ละตัว เมื่อ $1 \le i \le n$

ค่า c คือ ค่าคงที่

3.2.2 การเลือกมาตรวัด

มาตรวัดหรือตัวแปรต้นที่ใช้ในงานวิจัยนี้ ประกอบไปด้วยมาตรวัดทั้งหมด 18 มาตรวัด แบ่งออกเป็นมาตรวัดสำหรับแผนภาพคลาสจำนวน 12 มาตรวัด และมาตรวัดสำหรับแผนภาพซีเควนซ์จำนวน 6 มาตรวัด ดังแสดงในตารางที่ 3.1

จากตารางที่ 3.1 มาตรวัดที่มีเครื่องหมายดอกจันทร์ (*) กำกับ คือ มาตรวัดที่งานวิจัยนี้นำเสนอ ซึ่งได้จากการปรับปรุงมาตรวัดที่มีอยู่เดิมให้เป็นมาตรวัดสำหรับแผนภาพคลาสและแผนภาพซีเควนซ์ ส่วนมาตรวัดที่ไม่มีเครื่องหมายดอกจันทร์กำกับคือมาตรวัดที่มีนักวิจัยท่านอื่นนำเสนอไว้แล้ว

ตารางที่ 3.1 แสดงมาตรวัดที่ใช้ในงานวิจัยจำนวน 18 มาตรวัด

มาตรวัดสำหรับแผนภาพคลาส		
คลาส		Number of classes (NC)
คุณลักษณะ		Average number of attributes-unweighted (ANAUW) *
		Average number of attributes-weighted (ANAW) *
เมทธอด		Average number of methods-unweighted (ANMUW) *
		Average number of methods-weighted (ANMW) *
ความสัมพันธ์	แอสโซซิเอชัน	Average number of association relationships (ANAsso) *
	แอกกริเกชัน	Average number of aggregation relationships (ANAgg) *
		Number of aggregation hierarchies (NaggH)
		Maximum number of aggregation hierarchies (MaxHAgg)
	เจเนอรัลไลเซชัน	Average number of generalization relationships (ANGen) *
		Number of generalization hierarchies (NGenH)
		Maximum number of Depth of Inheritance Tree (MaxDIT)
มาตรวัดสำหรับแผนภาพซีเควนซ์		
ซีนาริโอ		Number of Scenarios (NOS) *
สาร		Weighted messages between objects (WMBO) *
		Average number of return messages (ANRM) *
		Average number of directly dispatched messages (ANDM) *
		Average number of the elements in transitive closure of directly dispatched messages (ANET) *
เงื่อนไข		Average number of condition messages (ANCM) *

มาตรวัดสำหรับแผนภาพคลาส ได้แบ่งแยกออกเป็น 4 กลุ่มย่อย โดยพิจารณาตามองค์ประกอบภายในแผนภาพคลาส ได้แก่ มาตรวัดในกลุ่มของคลาส คุณลักษณะ เมทธอด และความสัมพันธ์ ซึ่งความสัมพันธ์แยกออกเป็น ความสัมพันธ์แบบแอสโซซิเอชัน แอกกริเกชัน และเจเนอรัลไลเซชัน

3.2.2.1 นิยามของมาตรวัดสำหรับแผนภาพคลาส

- มาตรวัด Number of classes (NC) [7] คือ จำนวนของคลาสทั้งหมดภายในแผนภาพคลาส

- มาตรวัด Average number of attributes-unweighted (ANAUW) คือ ผลรวมของจำนวนคุณลักษณะภายในคลาสทั้งหมด โดยไม่มีการถ่วงค่าให้กับตัวดัดแปร (modifier) ของคุณลักษณะที่เป็นพับลิค (Public) โพรเทค (Protected) และไพรเวท (Private) หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Average number of attributes-weighted (ANAW) คือ ผลรวมของจำนวนคุณลักษณะภายในคลาสทั้งหมด โดยมีการถ่วงค่าให้กับตัวดัดแปร ของคุณลักษณะที่เป็นพับลิค เท่ากับ 1.0 โพรเทค เท่ากับ 0.5 และ ไพรเวท เท่ากับ 0.0 หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Average number of methods-unweighted (ANMUW) คือ ผลรวมของจำนวนเมทธอดในคลาสทั้งหมด โดยไม่มีการถ่วงค่าให้กับตัวดัดแปร ของเมทธอดที่เป็นพับลิค โพรเทค และ ไพรเวท หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Average number of methods-weighted (ANMW) คือ ผลรวมของจำนวนเมทธอดภายในคลาสทั้งหมด โดยมีการถ่วงค่าให้กับตัวดัดแปร ของเมทธอดที่เป็นพับลิค เท่ากับ 1.0 โพรเทค เท่ากับ 0.5 และไพรเวท เท่ากับ 0.0 หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Average number of association relationships (ANAsso) คือ ผลรวมของจำนวนความสัมพันธ์แบบแอสโซซิเอชัน หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Average number of aggregation relationships (ANAgg) คือ ผลรวมของจำนวนความสัมพันธ์แบบแอกกริเกชัน หารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Number of aggregation hierarchies (NaggH) [7] คือ ผลรวมของจำนวนลำดับชั้นของความสัมพันธ์แบบแอกกริเกชัน ภายในแผนภาพคลาส

- มาตรวัด Maximum number of aggregation hierarchies (MaxHAgg) [7] คือ ค่าความสูงของลำดับชั้นของความสัมพันธ์แบบแอกกริเกชันภายในแผนภาพคลาสที่มีความสูงมากที่สุด

- มาตรวัด Average number of generalization relationships (ANGen) คือ ผลรวมของจำนวนความสัมพันธ์แบบเจเนอรัลไลเซชันหารด้วยจำนวนคลาสทั้งหมด

- มาตรวัด Number of generalization hierarchies (NGenH) [7] คือ ผลรวมของจำนวนลำดับชั้นของความสัมพันธ์แบบเจเนอรัลไลเซชันภายในแผนภาพคลาส

- มาตรวัด Maximum number of Depth of Inheritance Tree (MaxDIT) [7] คือ ค่าความสูงของลำดับขั้นของความสัมพันธ์แบบเจเนอรัลไลเซชันภายในแผนภาพคลาสที่มีความสูงมากที่สุด

มาตรวัดสำหรับแผนภาพซีเควนซ์ ได้แบ่งแยกออกเป็น 3 กลุ่มย่อย ได้แก่ มาตรวัดในกลุ่มของซีนาริโอ (Scenario) สาร และเงื่อนไข

3.2.2.2 นิยามของมาตรวัดสำหรับแผนภาพซีเควนซ์

- มาตรวัด Number of Scenarios (NOS) คือ จำนวนซีนาริโอทั้งหมดภายในแผนภาพซีเควนซ์
- มาตรวัด Weighted messages between objects (WMBO) คือ ผลรวมของจำนวนสารทั้งหมดภายในซีนาริโอหารด้วยจำนวนวัตถุทุกซีนาริโอทั้งหมดหารด้วยจำนวนซีนาริโอ
- มาตรวัด Average number of return messages (ANRM) คือ ผลรวมของจำนวนสารแบบคืนค่าทั้งหมดหารด้วยจำนวนซีนาริโอ
- มาตรวัด Average number of directly dispatched messages (ANDM) คือ ผลรวมของจำนวนสารที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่นทุกซีนาริโอหารด้วยจำนวนซีนาริโอทั้งหมด
- มาตรวัด Average number of the elements in transitive closure of directly dispatched messages (ANET) คือ จำนวนสารในการส่งผ่านแบบปิด (Transitive closure) ของสารที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่น
- มาตรวัด Average number of condition messages (ANCM) คือ ผลรวมของจำนวนสารแบบที่มีเงื่อนไขทั้งหมดหารด้วยจำนวนซีนาริโอทั้งหมดภายในแผนภาพซีเควนซ์

3.2.3 การเลือกหน่วยทดลอง

หน่วยทดลองของงานวิจัยนี้คือ นิสิตระดับปริญญาโทในสาขาวิทยาศาสตร์คอมพิวเตอร์ (Computer Science) และสาขาวิศวกรรมซอฟต์แวร์ (Software Engineering) ชั้นปีที่ 1 และ 2 ของภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย จำนวน 60 คนเป็นผู้ทำการทดลอง ซึ่งในที่นี้จะเรียกว่าหน่วยทดลอง โดยหน่วยทดลองทั้งหมดได้ผ่านการเรียนในวิชา 2110638 เทคโนโลยีเชิงวัตถุ (Object-Oriented Technology) หรือวิชา 2110623 วิศวกรรมความต้องการซอฟต์แวร์ (Software Requirements

Engineering) เพื่อให้หน่วยทดลองมีความรู้พื้นฐานเกี่ยวกับการออกแบบเชิงวัตถุ และความรู้เกี่ยวกับแผนภาพคลาสและแผนภาพซีเควนซ์ของยูเอ็มแอล

3.2.4 การสร้างแบบทดสอบ

ระบบที่นำมาใช้ในการทดลองมีทั้งหมด 40 ระบบ ซึ่งแบ่งออกเป็นระบบที่ใช้เป็นชุดข้อมูลสอน สำหรับการสร้างโมเดลการทำนายจำนวน 35 ระบบและระบบที่ใช้เป็นชุดข้อมูลทดสอบ สำหรับตรวจสอบความถูกต้องของโมเดลจำนวน 5 ระบบ ซึ่งทั้ง 40 ระบบจะมีแบบทดสอบซึ่งประกอบไปด้วย 2 ส่วน คือ เอกสารความต้องการของระบบ และแผนภาพคลาสและแผนภาพซีเควนซ์ที่สร้างด้วยโปรแกรมเรชันเนลโรส ซึ่งประกอบด้วยข้อสอบสำหรับวัดระดับความสามารถในการทำความเข้าใจซอฟต์แวร์ได้จำนวน 20 ข้อ และข้อสอบสำหรับวัดระดับความสามารถในการปรับเปลี่ยนซอฟต์แวร์ได้จำนวน 10 ข้อ (รายละเอียดของตัวอย่างแบบทดสอบอยู่ในภาคผนวก ง) ซึ่งระบบทั้ง 40 ระบบมีจำนวนคลาสตั้งแต่ 6 คลาสขึ้นไปจนถึงมากที่สุด 36 คลาสดังแสดงในตารางที่ 3.2 และ 3.3

ตารางที่ 3.2 แสดงรายชื่อระบบที่ใช้เป็นชุดข้อมูลสอน

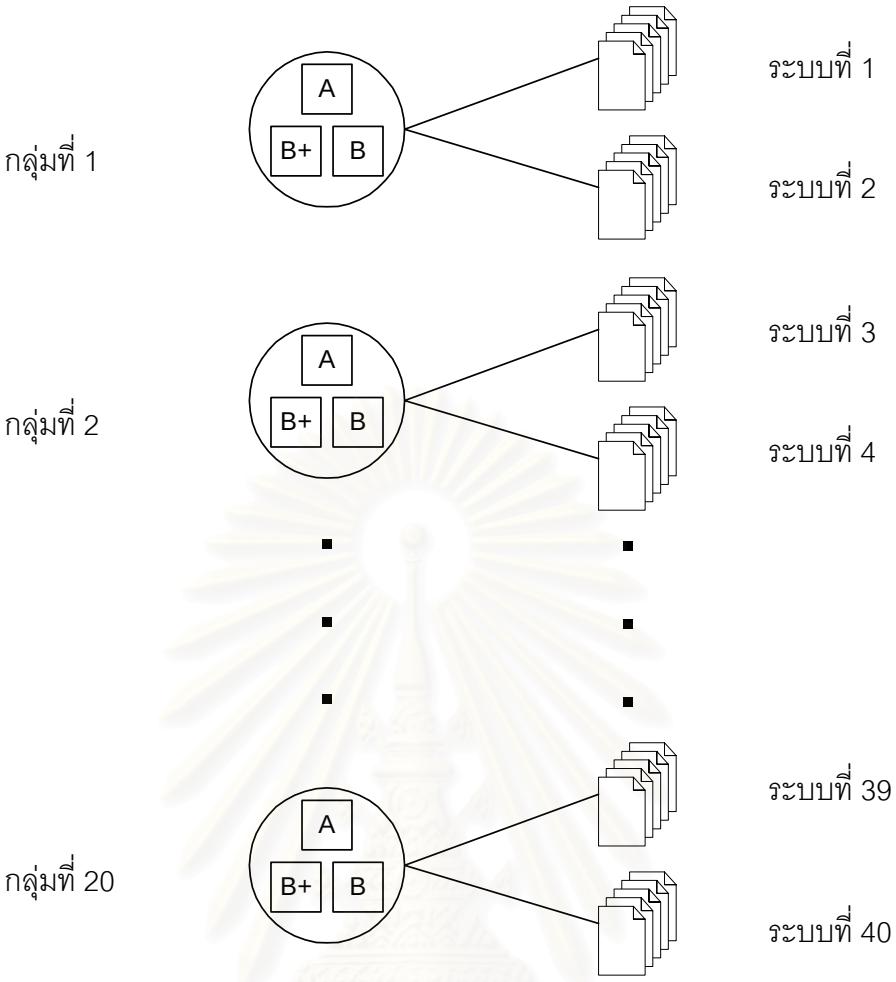
ระบบที่	ชื่อระบบ	จำนวนคลาส
1	ระบบลงทะเบียนสัมมนา	6
2	ระบบส่วนลดของการขาย	6
3	ระบบขายสินค้าผ่านเว็บ	7
4	ระบบลิฟต์	7
5	ระบบจัดซื้อวัตถุดิบ	7
6	ระบบสารสนเทศคลังยา	8
7	ระบบขายสินค้า	8
8	ระบบตัวแทนจำหน่ายรูปวาด	8
9	ระบบสั่งซื้ออุปกรณ์สุนัขออนไลน์	8
10	ระบบจำลองซื้อขายหุ้นผ่านอินเทอร์เน็ต	8
11	ระบบการลงทะเบียนเรียนผ่านเว็บ	8
12	ระบบซื้อขายหุ้นผ่านโทรศัพท์เคลื่อนที่	9
13	ระบบขายตั๋วหนังออนไลน์	9
14	ระบบเช่ารถ	9
15	ระบบจัดการร้านอาหาร	10

ตารางที่ 3.2 แสดงรายชื่อระบบที่ใช้เป็นชุดข้อมูลสอน (ต่อ)

ระบบที่	ชื่อระบบ	จำนวนคลาส
16	ระบบห้องสมุด	10
17	ระบบคงคลังสินค้า	10
18	ระบบงานโฆษณา	11
19	ระบบจองท่องเที่ยว	11
20	ระบบสินค้าบนเว็บ	11
21	ระบบกลุ่มงานสินค้าคงคลัง	12
22	ระบบงานเงินเดือน	12
23	ระบบขนส่งผู้โดยสารระหว่างเมือง	13
24	ระบบร้านหนังสือออนไลน์	13
25	ระบบเช่าวีดีโอ	14
26	ระบบกลุ่มงานสั่งซื้อ	14
27	ระบบโรงพยาบาลส่วนนัดหมายผู้ป่วย	15
28	ระบบจ่ายเงินล่วงหน้า	15
29	ระบบห้องสมุดใหญ่	16
30	ระบบร้านเช่าวีดีโอ	17
31	ระบบสถานีบริการหลวง	18
32	ระบบเอทีเอ็ม	24
33	ระบบงานต้นทุนการผลิต	24
34	ระบบการผลิตเสื้อผ้าส่งออก	25
35	ระบบเครื่องคิดเลข	36

ตารางที่ 3.3 แสดงรายชื่อระบบที่ใช้เป็นชุดข้อมูลทดสอบ

ระบบที่	ชื่อระบบ	จำนวนคลาส
36	ระบบจ่ายเงิน	6
37	ระบบการฝากถอนคำนวณเงินดำรงฐานะ	9
38	ระบบโรงภาพยนตร์แบบมัลติเพล็กซ์	10
39	ระบบการส่งออก	13
40	ระบบซื้อขายซีดีออนไลน์	23



รูปที่ 3.2 แสดงแผนภาพการแบ่งกลุ่มทำการทดลอง

3.2.5 การดำเนินการทดลอง

การดำเนินการทดลอง เริ่มจากแบ่งหน่วยทดลองออกเป็น 20 กลุ่มกลุ่มละ 3 คน เนื่องจากต้องการลดความคลาดเคลื่อนของการทดลองที่อาจเกิดขึ้นจากหน่วยทดลอง จึงกำหนดให้แต่ละกลุ่มมีหน่วยทดลองที่ได้เกรด A B+ และ B เพื่อให้แต่ละกลุ่มมีลักษณะคล้ายคลึงกันมากที่สุด จากนั้นให้หน่วยทดลองทั้ง 3 คนในแต่ละกลุ่มได้ทำข้อสอบคนละ 2 ระบบ การกำหนดข้อสอบให้กับหน่วยทดลองในแต่ละกลุ่มนั้นจะทำการเรียงลำดับข้อสอบตามจำนวนคลาสของข้อสอบ และทำการแบ่งข้อสอบออกเป็นสองกลุ่ม จากนั้นสุ่มข้อสอบจากทั้งสองกลุ่มให้กับหน่วยทดลองในแต่ละกลุ่ม ดังนั้นหน่วยทดลองในแต่ละกลุ่มจะได้ข้อสอบจำนวนสองชุดที่เป็นระบบที่มีโดเมน (Domain) แตกต่างกัน ดังแสดงในรูปที่ 3.2

ในการทำแบบทดสอบแต่ละระบบจะให้เวลาในการทำข้อสอบสำหรับวัดระดับความสามารถในการทำความเข้าใจซอฟต์แวร์ได้ 30 นาที และข้อสอบสำหรับวัดระดับ

ความสามารถในการปรับเปลี่ยนซอฟต์แวร์ได้ 40 นาที ในชุดที่ 1 และหยุดพักอีก 15 นาทีแล้วจึงทำข้อสอบในชุดที่ 2

3.3 การออกแบบเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

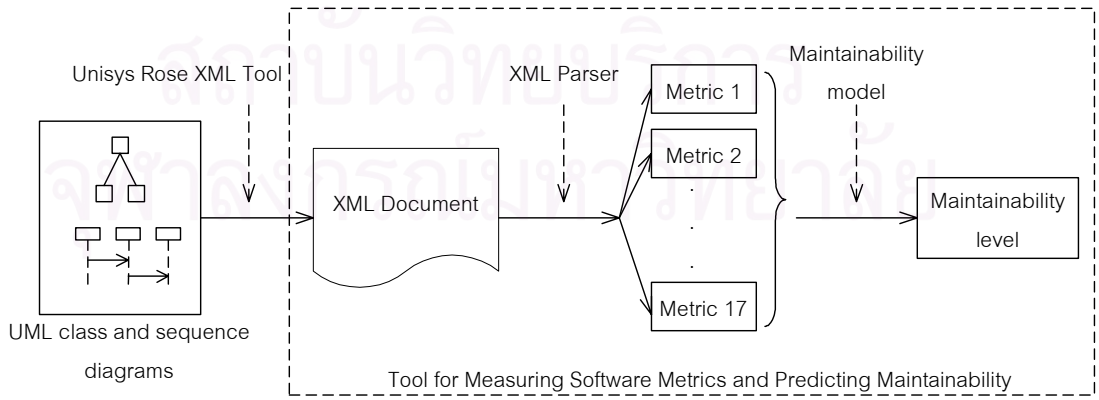
ขั้นตอนการออกแบบเครื่องมือสำหรับการคำนวณค่ามาตรวัดและการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ มีขั้นตอนดังนี้

3.3.1 การแปลงแผนภาพคลาสและแผนภาพซีเควนซ์เป็นเอกสารเอ็กซ์เอ็มแอล

ขั้นตอนแรกเริ่มจากนำแผนภาพคลาสและแผนภาพซีเควนซ์ จำนวน 40 ระบบที่สร้างโดยใช้โปรแกรมเรชันเนลโรสมาแปลงให้อยู่ในรูปเอกสารเอ็กซ์เอ็มแอลด้วยโปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล

3.3.2 การพัฒนาเครื่องมือคำนวณมาตรวัด

ขั้นตอนที่สองเป็นการพัฒนาเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยภาษาจาวาโดยอาศัยตัวแจงส่วนโครงสร้างของเอกสารเอ็กซ์เอ็มแอล ในการดึงข้อมูลจากเอกสารเอ็กซ์เอ็มแอลมาคำนวณค่ามาตรวัดที่ต้องการและเพิ่มความสามารถของเครื่องมือโดยให้สามารถทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ได้ ดังแสดงในรูปที่ 3.3 ซึ่งรายละเอียดเกี่ยวกับขั้นตอนการพัฒนาเครื่องมือสำหรับการคำนวณค่ามาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์อยู่ในบทที่ 4



รูปที่ 3.3 แสดงขั้นตอนการพัฒนาเครื่องมือสำหรับการคำนวณค่ามาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

3.4 การออกแบบการสร้างโมเดลการทำนาย

งานวิจัยนี้สร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้ระบบที่ใช้เป็นชุดข้อมูลสอน 35 ระบบและชุดข้อมูลทดสอบ 5 และได้เพิ่มการทดลองโดยปรับจำนวนระบบที่ใช้เป็นชุดข้อมูลสอน และชุดข้อมูลทดสอบดังนี้

1) สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 25 ระบบและชุดข้อมูลทดสอบจำนวน 15 ระบบ ระบบที่นำมาใช้เป็นชุดข้อมูลสอน ได้แก่ ระบบที่ 1, 2, 4, 5, 6, 7, 10, 12, 13, 15, 16, 17, 19, 21, 22, 24, 25, 26, 27, 28, 31, 32, 33, 34 และ 35 ระบบที่นำมาใช้เป็นชุดข้อมูลทดสอบ ได้แก่ ระบบที่ 3, 8, 9, 11, 14, 18, 20, 23, 29, 30, 36, 37, 38, 39 และ 40

2) สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 30 ระบบและชุดข้อมูลทดสอบจำนวน 10 ระบบ ระบบที่นำมาใช้เป็นชุดข้อมูลสอน ได้แก่ ระบบที่ 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34 และ 35 ระบบที่นำมาใช้เป็นชุดข้อมูลทดสอบ ได้แก่ ระบบที่ 3, 11, 14, 20, 29, 36, 37, 38, 39 และ 40

3) สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบ ระบบที่นำมาใช้เป็นชุดข้อมูลสอน ได้แก่ ระบบที่ 1 ถึงระบบที่ 35 ระบบที่นำมาใช้เป็นชุดข้อมูลทดสอบ ได้แก่ ระบบที่ 36 ถึงระบบที่ 40

3.4.1 ขั้นตอนการสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

ค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ของแต่ละระบบคำนวณได้จากค่าเฉลี่ยของผลรวมระหว่างคะแนนข้อสอบสำหรับวัดระดับความสามารถในการทำความเข้าใจซอฟต์แวร์และข้อสอบสำหรับวัดระดับความสามารถในการปรับเปลี่ยนซอฟต์แวร์จากหน่วยทดลองทั้ง 3 คน การสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ มีขั้นตอนดังนี้

ขั้นตอนที่ 1 นำข้อมูลมาตรวัดทั้ง 18 มาตรวัดของชุดข้อมูลสอนที่ได้จากขั้นตอนที่ 3.2 มาวิเคราะห์หาความสัมพันธ์แบบเพียร์สัน สำหรับคู่มาตรวัดที่มีค่าความสัมพันธ์สูงจะถูกเลือกมาเพียงมาตรวัดเดียว ด้วยการวิเคราะห์การถดถอยระหว่างค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ (y) ที่ได้จากขั้นตอนที่ 3.1 กับค่ามาตรวัด (x) เนื่องจากค่าอาร์สแควร์ที่ปรับแล้ว (Adjusted r Square) เป็นค่าที่แสดงว่าตัวแปรอิสระ (x) อธิบายความแปรปรวนของตัวแปรตาม (y) ได้ดีเพียงใด ดังนั้นมาตรวัดที่มีค่าอาร์สแควร์ที่ปรับแล้ว สูงแสดงว่าสามารถอธิบายค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ได้ดี ค่ามาตรวัดนั้นจะถูกเลือกไว้เพื่อสร้างโมเดลการ

ทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ส่วนมาตรวัดที่มีค่าอาร์สแควร์ที่ปรับแล้วน้อยกว่าจะถูกตัดทิ้ง

ขั้นตอนที่ 2 คำนวณค่าคัทออฟ (Cutoff) ซึ่งค่าคัทออฟ ค่าแรกได้จากค่าเฉลี่ยของคะแนนสอบบวกด้วยค่าส่วนเบี่ยงเบนมาตรฐาน ส่วนค่าคัทออฟ ค่าที่สองได้จากค่าเฉลี่ยของข้อมูลลบด้วยค่าส่วนเบี่ยงเบนมาตรฐาน

ขั้นตอนที่ 3 ปรับค่าคะแนนของแต่ละระบบเป็นค่า 0 1 และ 2 ซึ่งเป็นค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยค่าข้อมูลที่มีค่าน้อยกว่าค่าคัทออฟ ค่าที่หนึ่งจะถูกกำหนดให้มีค่าเป็น 0 ส่วนค่าข้อมูลที่มีค่าระหว่างค่าคัทออฟ ทั้งสองค่าจะถูกกำหนดให้มีค่าเป็น 1 และค่าข้อมูลที่มีค่ามากกว่าค่าคัทออฟ ค่าที่สองจะถูกกำหนดให้มีค่าเป็น 2 โดยค่า 0 แทนระบบที่มีระดับความสามารถในการบำรุงรักษาซอฟต์แวร์น้อย ค่า 1 แทนระบบที่มีระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ปานกลาง และค่า 2 แทนระบบที่มีระดับความสามารถในการบำรุงรักษาซอฟต์แวร์มาก ดังนั้นในขั้นตอนนี้ได้ค่า y สำหรับสร้างโมเดลการทำนายจำนวน 35 ระบบ และค่า y สำหรับตรวจสอบความถูกต้องของโมเดลจำนวน 5 ระบบ

ขั้นตอนสุดท้าย นำค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ และค่ามาตรวัดมาวิเคราะห์ทางสถิติด้วยวิธีการวิเคราะห์จำแนกกลุ่มด้วยโปรแกรมเอสพีเอสเอสสำหรับวินโดวส์ (SPSS for windows) [1, 2] เพื่อหามาตรวัดที่มีความสัมพันธ์กับความสามารถในการบำรุงรักษาซอฟต์แวร์ และสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ตามสมการที่ (1)

$$\text{Maintainability level} = a_1x_1 + a_2x_2 + \ldots + a_ix_i + \ldots + a_nx_n + c \quad \text{---------- (1)}$$

โดยที่ Maintainability level คือ ระดับความสามารถในการบำรุงรักษาซอฟต์แวร์

x_i คือ ค่ามาตรวัดตัวที่ i เมื่อ $1 \le i \le n$

a_i คือ ค่าสัมประสิทธิ์ของค่ามาตรวัดตัวที่ i เมื่อ $1 \le i \le n$

และ c คือ ค่าคงที่

3.5 การออกแบบการตรวจสอบโมเดลการทำนายที่ได้และสรุปผล

ในขั้นตอนนี้จะนำค่ามาตรวัดจากชุดข้อมูลทดสอบจำนวน 5 ระบบ มาสร้างโมเดลการทำนายเพื่อหาระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ และทำการเปรียบเทียบค่าที่ได้จากโมเดลการทำนายกับค่าที่ได้จากการทดลองขั้นตอนที่ 3.2 พร้อมทั้งหาเปอร์เซ็นต์ความผิดพลาดที่เกิดขึ้น และสรุปผลการทดลอง (รายละเอียดอยู่ในบทที่ 5)

บทที่ 4
การออกแบบและพัฒนาเครื่องมือสำหรับการคำนวณมาตรวัด และทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

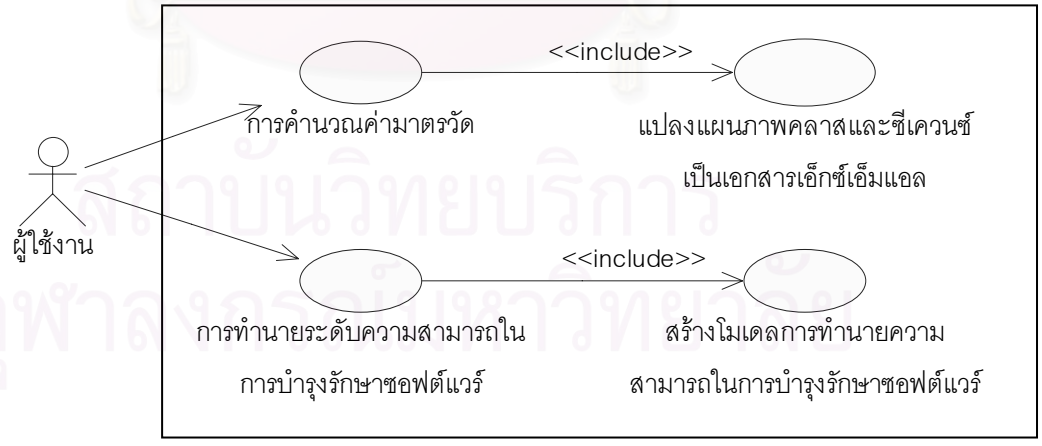
ในบทนี้จะอธิบายรายละเอียดเกี่ยวกับขั้นตอนการออกแบบและพัฒนาเครื่องมือขั้นตอนการคำนวณค่ามาตรวัด และการตรวจสอบค่ามาตรวัดที่ได้จากเครื่องมือ ดังนี้

4.1 การออกแบบและพัฒนาเครื่องมือ

เครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ (Java Tool for Measuring Metrics and Predicting Maintainability) พัฒนาด้วยโปรแกรมภาษาจาวาบนระบบปฏิบัติการวินโดว์ โดยใช้คลาสไลบรารีของ Java™ 2 SDK เวอร์ชัน 1.4.2 (j2sdk 1.4.2) ส่วนนี้นำเสนอแผนภาพยูสเคส แผนภาพคลาส และแผนภาพซีเควนซ์ของการออกแบบเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งมีรายละเอียดดังนี้

4.1.1 แผนภาพยูสเคส

เครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์มีฟังก์ชันการทำงานหลักอยู่ 2 ฟังก์ชัน คือ การคำนวณค่ามาตรวัด และการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 แสดงแผนภาพยูสเคสของเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

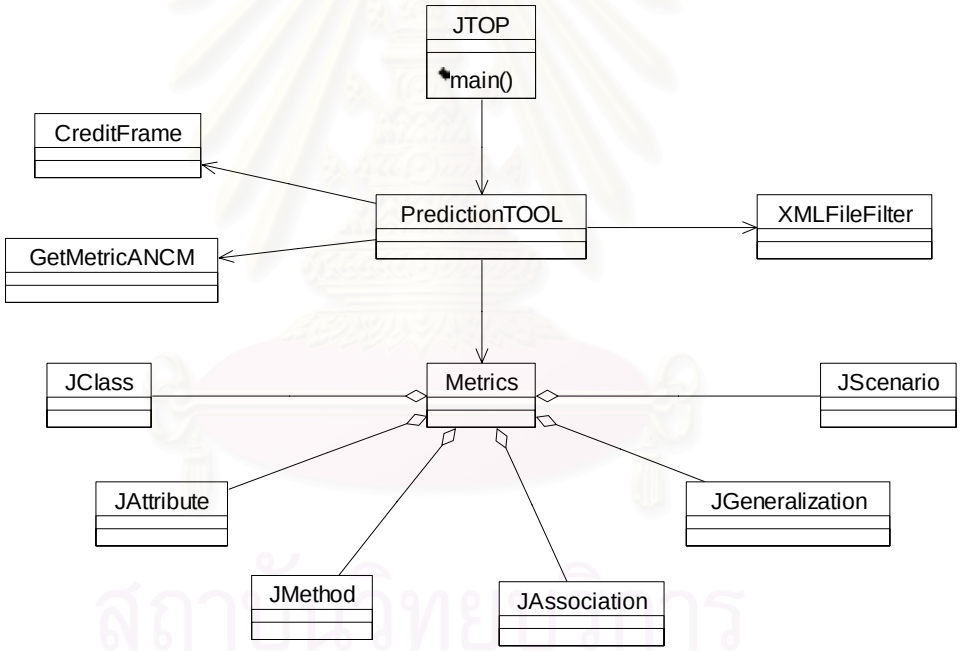
1) ฟังก์ชันการคำนวณค่ามาตรวัด รับข้อมูลเข้าเป็นเอกสารเอ็กซ์เอ็มแอลที่ได้จากการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ด้วยโปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอลมา

คำนวณค่ามาตรวัด โดยใช้ตัวแจงส่วนโครงสร้างของเอกสารเอ็กซ์เอ็มแอลชนิดแซกในการดึงข้อมูลจากเอกสารเอ็กซ์เอ็มแอล ผลที่ได้จากฟังก์ชันการคำนวณ คือ ค่ามาตรวัดจำนวน 17 มาตรวัด

2) ฟังก์ชันการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ รับข้อมูลเข้าเป็นค่ามาตรวัดที่ได้จากฟังก์ชันการคำนวณค่ามาตรวัด และนำค่ามาตวัดที่ได้เข้าสู่สมการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ผลที่ได้ คือ ค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์

4.1.2 แผนภาพคลาส

จากรูปที่ 4.2 แสดงคลาสทั้งหมด 12 คลาส ประกอบด้วยคลาส JTOP PredictionTOOL CreditFrame GetMetricANCM XMLFileFilter Metrics JClass JAttribute JMethod JAssociation JGeneralization และ JScenario



รูปที่ 4.2 แสดงแผนภาพคลาสของเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

4.1.2.1 คลาส JTOP

คลาส JTOP เป็นคลาสหลักของโปรแกรม ประกอบด้วย เมทธอด main() เพื่อเริ่มต้นรันโปรแกรม และมีความสัมพันธ์แบบแอสโซซิเอชันกับคลาส PredictionTOOL(รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับ (Source code) ของคลาส JTOP อยู่ในภาคผนวก จ)

4.1.2.2 **คลาส** PredictionTOOL

คลาสนี้เป็นคลาสที่แสดงหน้าจอหลักของเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งประกอบด้วยคอมโพเนนท์ของคลาสจาวาสวิง (Java Swing) ประกอบเป็นหน้าจอ มีเมทธอด calMetricsMenuItemActionPerformed() ใช้คำนวณค่ามาตรวัด และเมทธอด predictMenuItemActionPerformed() ใช้ทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ และมีความสัมพันธ์แบบแอสโซซิเอชันกับคลาส CreditFrame GetMetricANCM XMLFileFilter และ Metrics (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส PredictionTOOL อยู่ในภาคผนวก จ)

4.1.2.3 **คลาส** CreditFrame

คลาส CreditFrame เป็นคลาสที่แสดงหน้าจออธิบายเกี่ยวกับรายละเอียดของโปรแกรมชื่อผู้พัฒนาโปรแกรม และเวอร์ชันของโปรแกรม

4.1.2.4 **คลาส** GetMetricANCM

คลาส GetMetricANCM เป็นคลาสที่ถ่ายทอดมาจาก javax.swing.JDialog ซึ่งเป็นหน้าจอโต้ตอบ (Dialog) สำหรับรับค่ามาตรวัด ANCM จากผู้ใช้งานเพื่อนำมาคำนวณค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส GetMetricANCM อยู่ในภาคผนวก จ)

4.1.2.5 **คลาส** XMLFileFilter

คลาส XMLFileFilter เป็นคลาสที่ใช้เพื่อกำหนดนามสกุลของไฟล์ที่ต้องการเลือกมาเข้าสู่โปรแกรม โดยถูกกำหนดเป็นไฟล์นามสกุลเอ็กซ์เอ็มแอลเท่านั้น

4.1.2.6 **คลาส** Metrics

คลาส Metrics เป็นคลาสที่ทำการคำนวณค่ามาตรวัดทั้ง 17 มาตรวัดจากเอกสารเอ็กซ์เอ็มแอลที่นำเข้ามาในโปรแกรม ซึ่งคลาส Metrics มีความสัมพันธ์แบบแอกกริเกชันกับคลาส JClass JAttribute JMethod JAssociation JGeneralization และ JScenario (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส Metrics อยู่ในภาคผนวก จ)

4.1.2.7 **คลาส** JClass

คลาส JClass เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับคลาสจากเอกสารเอ็กเอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด NC (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JClass อยู่ในภาคผนวก จ)

4.1.2.8 **คลาส** JAttribute

คลาส JAttribute เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับคุณลักษณะจากเอกสารเอ็กซ์เอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด ANAUW และ ANAW (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JAttribute อยู่ในภาคผนวก จ)

4.1.2.9 **คลาส** JMethod

คลาส JMethod เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับเมทธอดจากเอกสารเอ็กซ์เอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด ANMUW และ ANMW (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JMethod อยู่ในภาคผนวก จ)

4.1.2.10 **คลาส** JAssociation

คลาส JAssociation เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับความสัมพันธ์แบบแอสโซซิเอชัน แอกกริเกชัน และ คอมโพซิชัน จากเอกสารเอ็กซ์เอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด ANAsso ANAgg NaggH และ MaxHAgg (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JAssociation อยู่ในภาคผนวก จ)

4.1.2.11 **คลาส** JGeneralization

คลาส JGeneralization เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับความสัมพันธ์แบบเจเนอรัลไลเซชัน จากเอกสารเอ็กซ์เอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด ANGen NGenH และ MaxDIT (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JGeneralization อยู่ในภาคผนวก จ)

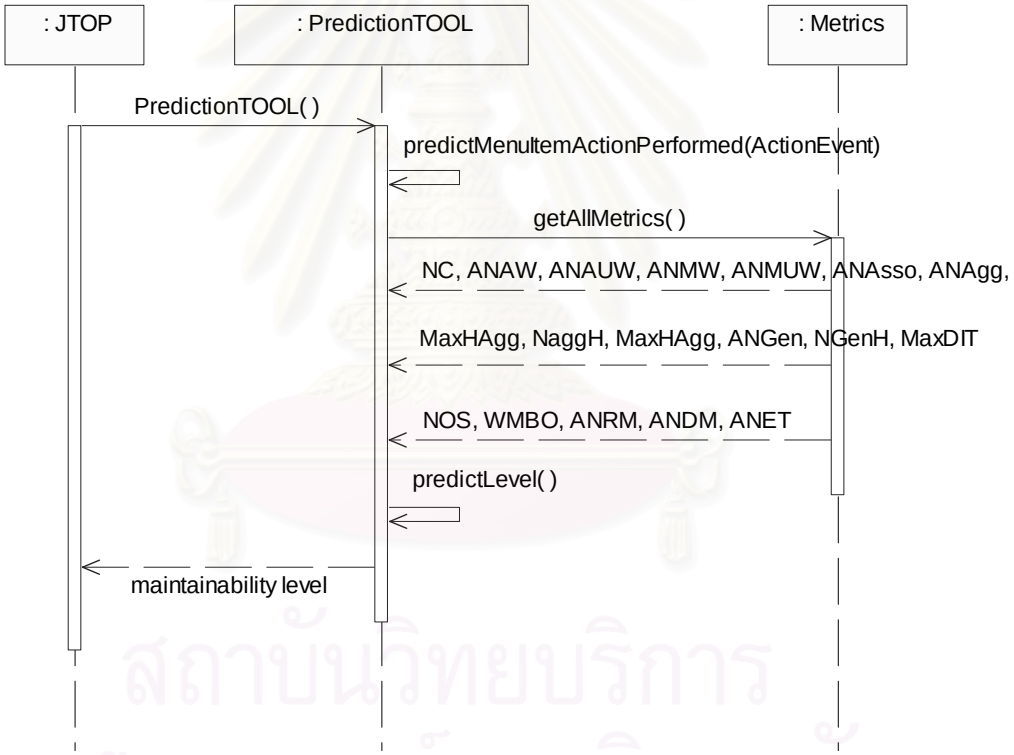
4.1.2.12 **คลาส** JScenario

คลาส JScenario เป็นคลาสที่ใช้เก็บรายละเอียดข้อมูลเกี่ยวกับมาตรวัดสำหรับแผนภาพซีเควนซ์ จากเอกสารเอ็กซ์เอ็มแอล เพื่อส่งค่าให้คลาส Metrics คำนวณค่ามาตรวัด NOS

WMBO ANRM ANDM และ ANET (รายละเอียดของคุณลักษณะ เมทธอด และรหัสต้นฉบับของคลาส JScenario อยู่ในภาคผนวก จ)

4.1.3 แผนภาพซีเควนซ์

จากรูปที่ 4.3 และ 4.4 แสดงแผนภาพซีเควนซ์ของการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ และการคำนวณค่ามาตรวัดตามลำดับ แผนภาพซีเควนซ์ของการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ประกอบด้วยวัตถุที่สร้างจากคลาส JTOP PredictionTOOL และ Metrics ส่วนแผนภาพซีเควนซ์ของการคำนวณค่ามาตรวัดประกอบด้วยวัตถุที่สร้างจากคลาส JTOP PredictionTOOL Metrics JClass JAttribute JMethod JScenario JAssociation และ JGeneralization

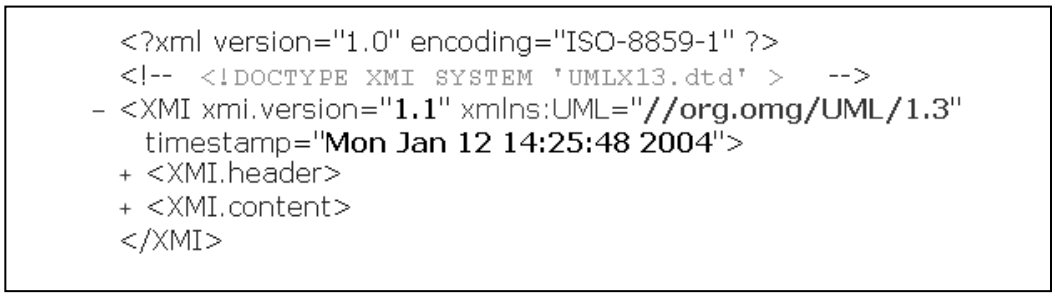


รูปที่ 4.3 แผนภาพซีเควนซ์แสดงการทำนายค่าความสามารถในการบำรุงรักษาซอฟต์แวร์

4.2 การคำนวณค่ามาตรวัด

เอกสารเอ็กซ์เอ็มแอลที่ได้จากการแปลงแผนภาพคลาสและแผนภาพซีเควนซ์ด้วยโปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล มีส่วนประกอบหลักอยู่ 2 ส่วน ดังแสดงในรูปที่ 4.5 ประกอบไปด้วยส่วนหัว (Header) และส่วนเนื้อหา (Content)

รูปที่ 4.4 แผนภาพซีเควนซ์แสดงการคำนวณค่ามาตรวัด



รูปที่ 4.5 แสดงส่วนประกอบหลักภายในเอกสารเอ็กซ์เอ็มแอล

การคำนวณค่ามาตรวัดจะพิจารณาข้อมูลเอกสารเอ็กซ์เอ็มแอลในส่วนเนื้อหาเท่านั้น ซึ่งการคำนวณมาตรวัดแบ่งออกเป็นการคำนวณค่ามาตรวัดสำหรับแผนภาพคลาส และมาตรวัดสำหรับแผนภาพซีเควนซ์ ซึ่งมีรายละเอียดดังนี้

4.2.1 การคำนวณค่ามาตรวัดสำหรับแผนภาพคลาส

การคำนวณค่ามาตรวัดสำหรับแผนภาพคลาสนั้น จะพิจารณาจากข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับคลาส ดังแสดงในรูปที่ 4.6 ได้แก่

1) คลาส ซึ่งอธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Class ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส (Namespace) เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของคลาส คุณลักษณะ visibility คือ ค่าตัวดัดแปร ของคลาส ซึ่งมีอยู่ 3 ค่า คือ พับลิค โพรเทค และไพรเวท และคุณลักษณะ isSpecification บอกว่าคลาสนั้นคือคลาสลูกหรือไม่ ถ้าคลาสนั้นเป็นคลาสลูกจะมีค่าเป็นจริง แต่ถ้าไม่ใช่จะมีค่าเป็นเท็จ

2) คุณลักษณะภายในคลาส อธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Attribute ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของคุณลักษณะภายในคลาส คุณลักษณะ visibility คือ ค่าตัวดัดแปร ของคุณลักษณะภายในคลาส

3) เมทธอด อธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Operation ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของเมทธอดภายในคลาส คุณลักษณะ visibility คือ ค่าตัวดัดแปร ของเมทธอดภายในคลาส

4) ความสัมพันธ์ อธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Association ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของความสัมพันธ์ ซึ่งภายในแท็กอิลิเมนต์นี้จะมีแท็กอิลิ-เมนต์ย่อยภายใน ชื่อ UML:AssociationEnd ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะแอกกริเก-

ชัน ซึ่งจะเป็นตัวแสดงว่าถ้ามีความสัมพันธ์แบบแอกกริเกชัน ค่าของคุณลักษณะแอกกริเกชันจะมีค่าเป็น none แต่ถ้ามีความสัมพันธ์แบบแอกกริเกชัน ค่าของคุณลักษณะแอกกริเกชัน จะมีค่าเป็นแอกกริเกท (Aggregate) และถ้ามีความสัมพันธ์แบบคอมโพซิชัน ค่าของคุณลักษณะแอกกริเกชันจะมีค่าเป็นคอมโพซิท (composite)

ซึ่งการคำนวณค่ามาตรวัดทั้ง 12 มาตรวัดของแผนภาพคลาส ได้แก่ มาตรวัด NC ANAUW ANAW ANMUW ANMW ANAsso ANAgg NaggH MaxHAgg ANGen NGenH และ MaxDIT คำนวณจากข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับคลาสดังกล่าว

4.2.2 การคำนวณค่ามาตรวัดสำหรับแผนภาพซีเควนซ์

การคำนวณค่ามาตรวัดสำหรับแผนภาพซีเควนซ์นั้น จะพิจารณาจากข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับซีเควนซ์ ดังแสดงในรูปที่ 4.7 ได้แก่

1) ซีนารีโอ อธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Interaction ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของซีนารีโอนั้น

2) สาร อธิบายด้วยแท็กอิลิเมนต์ ชื่อ UML:Message ซึ่งมีคุณลักษณะที่สำคัญ คือ คุณลักษณะ xmi.id ซึ่งเป็นเนมสเปส เพื่อใช้ในการอ้างถึงภายในเอกสารเอ็กซ์เอ็มแอล คุณลักษณะ name คือ ชื่อของสารภายในมีแท็กอิลิเมนต์ย่อยที่สำคัญคือ UML:Message.sender อ้างไปยังวัตถุของคลาสที่เป็นผู้ส่งสาร และ UML:Message.receiver อ้างไปยังวัตถุของคลาสที่เป็นผู้รับสาร

ซึ่งการคำนวณค่ามาตรวัดทั้ง 5 มาตรวัดของแผนภาพซีเควนซ์ ได้แก่ มาตรวัด NOS WMBO ANRM ANDM และ ANET คำนวณจากข้อมูลภายในเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับซีเควนซ์ดังกล่าว

4.3 การตรวจสอบค่ามาตรวัดที่ได้จากเครื่องมือ

ผู้วิจัยได้ทำการตรวจสอบความถูกต้องของค่ามาตรวัดทั้ง 17 มาตรวัดที่คำนวณได้จากเครื่องมือสำหรับการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์กับค่ามาตรวัดทั้ง 17 มาตรวัดที่ได้จากวิธีการนับและการคำนวณด้วยมือ ด้วยระบบที่นำมาใช้ในการทดลองทั้ง 40 ระบบ และทำการเปรียบเทียบค่าของมาตรวัดที่คำนวณได้จากทั้งสองวิธี ผลการเปรียบเทียบพบว่าเครื่องมือที่พัฒนาขึ้นสามารถคำนวณค่ามาตรวัดได้ถูกต้องตรงกับการคำนวณด้วยการนับและการคำนวณด้วยมือ

```
<!-- ==================== websale::Warehouse [Class] ====================
>
- <UML:Class xmi.id="S.1" name="Warehouse" visibility="public" isSpecification="false"
  isRoot="true" isLeaf="true" isAbstract="false" isActive="false">
  - <UML:ModelElement.namespace>
      <Foundation.Core.Namespace xmi.idref="G.0" />
      <!-- websale -->
    </UML:ModelElement.namespace>
  - <UML:Classifier.feature>
      <!-- ==================== websale::Warehouse.productCode [Attribute]
      ==================== -->
    + <UML:Attribute xmi.id="S.2" name="productCode" visibility="private"
        isSpecification="false" ownerScope="instance" changeability="changeable"
        targetScope="instance">
      <!-- ==================== websale::Warehouse.productName [Attribute]
      ==================== -->
    + <UML:Attribute xmi.id="S.3" name="productName" visibility="private"
        isSpecification="false" ownerScope="instance" changeability="changeable"
        targetScope="instance">
      <!-- ==================== websale::Warehouse.productQty [Attribute]
      ==================== -->
    + <UML:Attribute xmi.id="S.4" name="productQty" visibility="private"
        isSpecification="false" ownerScope="instance" changeability="changeable"
        targetScope="instance">
     <UML:Operation xmi.id="S.6" name="deliveryOrder" visibility="public"
       isSpecification="false" ownerScope="instance" isQuery="false"
       concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
       specification="" />
     <!-- ==================== websale::Warehouse::getProductDetail
     [Operation] ==================== -->
     <UML:Operation xmi.id="S.7" name="getProductDetail" visibility="public"
       isSpecification="false" ownerScope="instance" isQuery="false"
       concurrency="sequential" isRoot="false" isLeaf="false" isAbstract="false"
       specification="" />
   </UML:Classifier.feature>
- <UML:Association xmi.id="G.10" name="view{3E23A9D801A1}" visibility="public"
    isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false">
  - <UML:Association.connection>
      <!-- ==================== websale::view{3E23A9D801A1} [AssociationEnd]
      ==================== -->
    - <UML:AssociationEnd xmi.id="G.11" name="" visibility="public" isSpecification="false"
        isNavigable="true" ordering="unordered" aggregation="none"
        targetScope="instance" changeability="changeable">
      - <UML:AssociationEnd.multiplicity>
        - <UML:Multiplicity>
          - <UML:Multiplicity.range>
              <UML:MultiplicityRange lower="1" upper="-1" />
            </UML:Multiplicity.range>
          </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
      - <UML:AssociationEnd.type>
          <Foundation.Core.Classifier xmi.idref="S.14" />
          <!-- websale::ProductRepository -->
        </UML:AssociationEnd.type>
      </UML:AssociationEnd>
      <!-- ==================== websale::view{3E23A9D801A1} [AssociationEnd]
      ==================== -->
    - <UML:AssociationEnd xmi.id="G.12" name="" visibility="public" isSpecification="false"
        isNavigable="false" ordering="unordered" aggregation="none"
        targetScope="instance" changeability="changeable">
      - <UML:AssociationEnd.multiplicity>
```

รูปที่ 4.6 แสดงเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับคลาส

```
- <UML:Collaboration.interaction>
  - <UML:Interaction xmi.id="G.13" name="{Logical View}2_orderFail" visibility="public"
      isSpecification="false">
    - <UML:Interaction.message>
      - <UML:Message xmi.id="G.22" name="view( )" visibility="public"
          isSpecification="false">
        - <UML:Message.sender>
            <Behavioral_Elements.Collaborations.ClassifierRole xmi.idref="G.53" />
          </UML:Message.sender>
        - <UML:Message.receiver>
            <Behavioral_Elements.Collaborations.ClassifierRole xmi.idref="G.54" />
          </UML:Message.receiver>
        - <UML:Message.message3>
            <Behavioral_Elements.Collaborations.Message xmi.idref="G.23" />
          </UML:Message.message3>
        - <UML:Message.communicationConnection>
            <Behavioral_Elements.Collaborations.AssociationRole xmi.idref="G.15" />
          </UML:Message.communicationConnection>
        - <UML:Message.action>
            <Behavioral_Elements.Common_Behavior.Action xmi.idref="XX.24" />
          </UML:Message.action>
        </UML:Message>
```

รูปที่ 4.7 แสดงเอกสารเอ็กซ์เอ็มแอลที่แสดงรายละเอียดเกี่ยวกับซีเควนซ์

บทที่ 5
ผลการทดลองและการนำไปใช้งาน

ในบทนี้จะกล่าวถึงผลที่ได้จากการทดลองในบทที่ 3 ประกอบด้วย ค่ามาตรวัดทั้ง 40 ระบบ ค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 40 ระบบ โมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ การตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ผลการเปรียบเทียบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ และนำเสนอวิธีการนำโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ไปใช้งาน ซึ่งมีรายละเอียดดังนี้

5.1 ค่ามาตรวัดทั้ง 40 ระบบ

ผลจากการคำนวณค่ามาตรวัดทั้ง 40 ระบบได้ผลข้อมูลแสดงดังตารางที่ 5.1 และ 5.2 โดยตารางที่ 5.1 แสดงข้อมูลมาตรวัดของระบบที่ใช้เป็นชุดข้อมูลสอน จำนวน 35 ระบบ และตารางที่ 5.2 แสดงข้อมูลมาตรวัดของระบบที่ใช้เป็นชุดข้อมูลทดสอบ จำนวน 5 ระบบ

5.2 ค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 40 ระบบ

จากการดำเนินการทดลองในขั้นตอนที่ 3.2.5 ได้ผลคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ของชุดข้อมูลสอนดังแสดงในตารางที่ 5.3 และผลคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ของชุดข้อมูลทดสอบดังแสดงในตารางที่ 5.4 ซึ่งประกอบไปด้วยคะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์ คะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์ คะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ซึ่งเป็นผลรวมของคะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์และคะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์ และค่าที่ปรับเป็นค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งค่าคัทออฟค่าแรกมีค่าเท่ากับ 21.000 และค่าคัทออฟค่าที่สองมีค่าเท่ากับ 28.700

5.3 โมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

งานวิจัยนี้ได้ทดลองสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ตามขั้นตอนที่ได้อธิบายไว้ในหัวข้อ 3.4 ผลการทดลองของการสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ มีรายละเอียดดังนี้

ตารางที่ 5.1แสดงค่ามาตรวัดของระบบที่ใช้เป็นชุดข้อมูลสอน จำนวน 18 มาตรวัด

	มาตรวัด																	
ระบบที่	NC	ANAUW	ANAW	ANMUW	ANMW	ANAsso	ANAgg	NaggH	MaxHAgg	ANGen	NGenH	Max DIT	NOS	WMBO	ANRM	ANDM	ANET	ANCM
1	6.000	2.500	0.000	2.333	2.333	0.667	0.167	1.000	1.000	0.000	0.000	0.000	2.000	1.667	2.500	1.500	0.000	0.500
2	6.000	2.833	0.000	1.333	1.333	0.500	0.167	1.000	1.000	0.000	0.000	0.000	1.000	1.200	3.000	1.000	0.000	0.000
3	7.000	2.857	0.214	1.571	1.571	0.571	0.000	0.000	0.000	0.286	1.000	1.000	2.000	1.325	2.000	2.000	0.000	0.000
4	7.000	1.857	0.071	1.714	1.714	0.429	0.429	2.000	1.000	0.286	1.000	1.000	2.000	1.800	90.000	1.000	2.000	0.000
5	7.000	2.857	0.000	1.714	1.714	1.714	0.143	1.000	1.000	0.000	0.000	0.000	3.000	1.500	1.333	0.333	0.000	0.333
6	8.000	3.000	0.000	1.875	1.875	0.750	0.250	2.000	1.000	0.000	0.000	0.000	3.000	1.028	0.000	1.000	0.000	0.667
7	8.000	1.875	0.000	3.000	3.000	0.625	0.250	1.000	1.000	0.250	1.000	1.000	1.000	2.167	1.000	7.000	4.000	1.000
8	8.000	3.125	1.125	3.500	3.500	0.375	0.000	0.000	0.000	0.625	1.000	3.000	2.000	1.450	1.000	1.000	0.000	0.000
9	8.000	4.250	0.375	3.625	3.625	0.375	0.250	2.000	1.000	0.250	1.000	1.000	4.000	1.938	2.250	2.250	0.000	0.000
10	8.000	2.125	0.000	3.875	3.875	1.000	0.000	0.000	0.000	0.000	0.000	0.000	3.000	2.167	2.000	2.667	0.000	3.000
11	8.000	2.500	0.125	4.625	4.625	0.750	0.250	2.000	1.000	0.000	0.000	0.000	3.000	1.600	1.333	1.000	0.000	1.333
12	9.000	0.889	0.333	3.222	3.222	0.333	0.333	3.000	1.000	0.556	2.000	1.000	3.000	1.111	3.333	1.333	0.000	1.667
13	9.000	1.778	0.000	5.333	5.333	0.111	0.667	1.000	2.000	0.000	0.000	0.000	3.000	4.944	1.333	2.000	0.000	0.667
14	9.000	2.889	0.222	2.000	2.000	0.667	0.000	0.000	0.000	0.222	1.000	1.000	3.000	1.250	0.667	1.000	0.000	0.667
15	10.000	2.000	0.000	3.100	3.100	0.800	0.100	1.000	1.000	0.000	0.000	0.000	3.000	1.250	1.000	1.667	0.000	0.667

ตารางที่ 5.1 แสดงค่ามาตรวัดของระบบที่ใช้เป็นชุดข้อมูลสอน จำนวน 18 มาตรวัด (ต่อ)

	มาตรวัด																	
ระบบที่	NC	ANAUW	ANAW	ANMUW	ANMW	ANAsso	ANAgg	NaggH	MaxHAgg	ANGen	NGenH	Max DIT	NOS	WMBO	ANRM	ANDM	ANET	ANCM
16	10.000	1.800	0.250	2.800	2.800	0.400	0.000	0.000	0.000	0.500	2.000	1.000	2.000	1.500	1.000	4.000	0.000	1.000
17	10.000	2.000	0.000	2.900	2.850	0.800	0.000	0.000	0.000	0.300	1.000	1.000	3.000	3.111	1.667	1.000	0.000	1.000
18	11.000	2.727	0.409	4.364	4.364	0.364	0.364	1.000	2.000	0.364	2.000	1.000	2.000	2.167	0.000	2.500	0.000	0.000
19	11.000	4.545	0.000	1.182	1.182	0.909	0.273	3.000	1.000	0.000	0.000	0.000	2.000	1.500	0.500	2.000	0.000	1.000
20	11.000	2.636	0.000	3.091	3.091	1.091	0.455	2.000	1.000	0.000	0.000	0.000	2.000	1.875	0.500	2.500	0.000	0.500
21	12.000	1.917	0.292	3.750	3.750	0.250	0.250	2.000	2.000	0.250	2.000	1.000	4.000	1.333	0.000	3.250	0.000	0.250
22	12.000	1.583	0.000	3.583	3.583	0.250	0.250	1.000	2.000	0.000	0.000	0.000	3.000	1.694	0.667	4.000	0.000	0.333
23	13.000	2.000	0.077	2.462	2.462	0.846	0.154	2.000	1.000	0.308	1.000	1.000	3.000	2.111	0.000	1.333	0.000	1.333
24	13.000	1.846	0.385	2.769	2.462	0.385	0.462	3.000	1.000	0.385	2.000	1.000	3.000	1.833	2.000	1.000	0.000	0.000
25	14.000	2.143	0.250	1.214	1.143	0.286	0.286	2.000	2.000	0.429	2.000	1.000	2.000	1.500	2.000	2.000	3.000	0.000
26	14.000	1.286	0.000	4.000	4.000	0.500	0.286	3.000	1.000	0.000	0.000	0.000	5.000	1.933	0.400	2.600	0.000	0.000
27	15.000	2.067	0.333	1.000	1.000	0.533	0.333	3.000	1.000	0.333	1.000	2.000	3.000	1.417	0.000	1.000	0.000	0.333
28	15.000	3.400	0.367	1.533	1.533	0.533	0.067	1.000	1.000	0.467	2.000	2.000	4.000	1.558	0.500	3.000	0.000	0.750
29	16.000	1.500	0.000	1.500	1.500	0.313	0.000	0.000	0.000	0.625	3.000	1.000	4.000	0.938	1.250	0.500	0.000	0.750
30	17.000	3.353	0.559	2.824	2.824	0.353	0.176	2.000	1.000	0.471	2.000	2.000	3.000	1.733	1.667	3.000	0.000	1.000

ตารางที่ 5.1 แสดงค่ามาตรวัดของระบบที่ใช้เป็นชุดข้อมูลสอน จำนวน 18 มาตรวัด (ต่อ)

	มาตรวัด																	
ระบบที่	NC	ANAUW	ANAW	ANMUW	ANMW	ANAsso	ANAgg	NaggH	MaxHAgg	ANGen	NGenH	Max DIT	NOS	WMBO	ANRM	ANDM	ANET	ANCM
31	18.000	1.556	0.000	1.167	1.167	0.722	0.167	2.000	1.000	0.000	0.000	0.000	2.000	1.000	0.000	2.000	0.000	0.000
32	24.000	0.792	0.042	1.250	1.250	0.917	0.458	3.000	1.000	0.167	1.000	1.000	4.000	1.263	1.500	0.500	0.000	0.500
33	24.000	2.125	0.042	2.750	2.750	0.500	0.250	3.000	1.000	0.083	1.000	1.000	3.000	2.833	0.000	3.000	0.333	0.333
34	25.000	1.560	0.16	1.400	1.400	0.480	0.280	4.000	1.000	0.360	3.000	1.000	2.000	1.000	1.000	2.000	0.000	1.000
35	36.000	1.528	0.361	3.500	3.431	0.389	0.000	0.000	0.000	0.667	4.000	2.000	2.000	1.744	0.000	0.667	0.000	0.000

ตารางที่ 5.2 แสดงค่ามาตรวัดของระบบที่ใช้เป็นชุดข้อมูลทดสอบ จำนวน 18 มาตรวัด

	มาตรวัด																	
ระบบที่	NC	ANAUW	ANAW	ANMUW	ANMW	ANAsso	ANAgg	NaggH	MaxHAgg	ANGen	NGenH	Max DIT	NOS	WMBO	ANRM	ANDM	ANET	ANCM
36	6.000	3.000	0.167	5.833	4.833	0.167	0.167	1.000	1.000	0.500	1.000	2.000	1.000	2.000	3.000	1.000	0.000	1.000
37	9.000	2.222	0.333	3.667	3.667	0.556	0.222	2.000	1.000	0.333	2.000	1.000	5.000	1.250	0.200	0.800	0.000	1.400
38	10.000	1.200	0.100	2.900	2.900	0.500	0.200	1.000	2.000	0.200	1.000	1.000	4.000	1.075	1.000	1.750	0.000	1.000
39	13.000	1.769	0.154	1.308	1.308	0.769	0.000	0.000	0.000	0.538	2.000	1.000	4.000	0.875	3.000	1.750	0.000	0.500
40	23.000	0.957	0.109	1.043	1.043	0.522	0.217	3.000	1.000	0.348	3.000	1.000	1.000	0.909	0.000	3.000	0.000	0.000

ตารางที่ 5.3 แสดงคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ของชุดข้อมูลสอน

ระบบที่	คะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการบำรุงรักษา (คะแนนเต็ม 40 คะแนน)*	ค่าที่ปรับเป็นค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์
1	13.000	8.667	21.667	1
2	12.333	12.000	24.333	1
3	17.33	10.67	28.00	1
4	12.667	11.000	23.67	1
5	13.500	12.000	25.500	1
6	14.667	15.667	30.33	2
7	13.000	16.000	29.00	2
8	13.333	11.000	24.333	1
9	12.333	12.333	24.667	1
10	15.000	14.667	29.667	2
11	14.333	13.667	28.000	1
12	12.667	13.333	26.00	1
13	10.333	11.000	21.333	1
14	16.000	14.333	30.333	2
15	16.333	14.000	30.33	2
16	12.333	12.000	24.333	1
17	13.000	11.000	24.000	1
18	8.333	9.000	17.33	0
19	13.667	12.000	25.67	1
20	15.000	13.833	28.833	2
21	8.000	8.667	16.667	0
22	15.333	12.667	28.000	1
23	15.333	13.000	28.33	1
24	13.333	12.333	25.67	1
25	14.000	14.000	28.00	1

ตารางที่ 5.3 แสดงคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ของชุดข้อมูลสอน (ต่อ)

ระบบที่	คะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการบำรุงรักษา (คะแนนเต็ม 40 คะแนน)*	ค่าที่ปรับเป็นค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์
26	13.000	12.333	25.333	1
27	15.667	10.333	26.00	1
28	13.000	10.667	23.67	1
29	12.333	15.000	27.33	1
30	12.333	11.000	23.333	1
31	12.667	11.667	24.33	1
32	16.000	10.333	26.33	1
33	12.333	11.667	24.000	1
34	12.667	7.333	20.00	0
35	9.000	8.333	17.333	0

* ค่าคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์เกิดจากผลรวมของคะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์และคะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์

ตารางที่ 5.4 แสดงคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์ของชุดข้อมูลทดสอบ

ระบบที่	คะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์ (คะแนนเต็ม 20 คะแนน)	คะแนนความสามารถในการบำรุงรักษา (คะแนนเต็ม 40 คะแนน)*	ค่าที่ปรับเป็นค่าระดับความสามารถในการบำรุงรักษาซอฟต์แวร์
36	15.333	15.333	30.667	2
37	14.667	12.333	27.00	1
38	13.333	6.667	20.00	0
39	16.000	12.333	28.333	1
40	9.000	8.667	17.67	0

* ค่าคะแนนความสามารถในการบำรุงรักษาซอฟต์แวร์เกิดจากผลรวมของคะแนนความสามารถในการทำความเข้าใจซอฟต์แวร์และคะแนนความสามารถในการปรับเปลี่ยนซอฟต์แวร์

5.3.1 สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบ

จากการนำข้อมูลมาตรวัดทั้ง 18 มาตรวัดของชุดข้อมูลสอนมาวิเคราะห์หาความสัมพันธ์แบบเพียร์สัน ซึ่งมีความสัมพันธ์ ดังแสดงในตารางที่ 5.5 พบว่ามีมาตรวัดจำนวน 5 คู่ที่มีค่าความสัมพันธ์กันสูง ดังแสดงในตารางที่ 5.6 เมื่อพิจารณาค่าอาร์สแควร์ที่ปรับแล้วที่ได้จากวิธีการวิเคราะห์การถดถอย ดังแสดงในตารางที่ 5.7 พบว่ามีมาตรวัดจำนวน 4 มาตรวัดที่ถูกตัดออก คือ มาตรวัด ANAW ANMW ANAgg และ ANGen ดังนั้นในขั้นตอนนี้จะเหลือมาตรวัดจำนวน 14 มาตรวัด คือ มาตรวัดNC ANAUW ANMUW ANAsso NaggH MaxHAgg NGenH MaxDIT NOS WMBO ANRM ANDM ANET และ ANCM ที่นำไปวิเคราะห์ทางสถิติด้วยวิธีการวิเคราะห์จำแนกกลุ่มด้วยโปรแกรมเอสพีเอสเอสสำหรับวินโดวส์

ตารางที่ 5.8 แสดงผลที่ได้จากการนำค่ามาตรวัดจำนวน 14 มาตรวัดและค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ไปวิเคราะห์จำแนกกลุ่มด้วยโปรแกรมเอสพีเอสเอสสำหรับวินโดวส์ ค่าตัวเลขที่แสดงในตารางคือค่าสัมประสิทธิ์ของมาตรวัดแต่ละมาตรวัดในคอลัมน์ที่ 1 โดยค่าสัมประสิทธิ์ในคอลัมน์ที่ 2 เป็นค่าสัมประสิทธิ์สำหรับฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับยาก ค่าสัมประสิทธิ์ในคอลัมน์ที่ 3 เป็นค่าสัมประสิทธิ์สำหรับฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับปานกลาง และค่าสัมประสิทธิ์ในคอลัมน์ที่ 4 เป็นค่าสัมประสิทธิ์สำหรับฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับง่าย

จากตารางที่ 5.8 สามารถเขียนสมการสำหรับฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์แต่ละระดับได้ดังนี้

$$F_{difficult} = 0.98\times NC+7.856\times ANAUW+10.146\times ANMUW+27.655\times ANAsso$$
$$+0.287\times NaggH+11.547\times MaxHAgg+13.096\times NGenH-4.062\times MaxDIT$$
$$-0.737\times NOS-2.393\times WMBO+0.192\times ANRM+0.215\times ANDM-0.672\times ANET$$
$$-2.51\times ANCM-60.615$$

$$F_{medium} = 0.825\times NC+6.56\times ANAUW+4.614\times ANMUW+20.022\times ANAsso$$
$$+0.254\times NaggH+4.801\times MaxHAgg+3.332\times NGenH+0.282\times MaxDIT$$
$$+3.656\times NOS+0.897\times WMBO+3.858\times ANRM+0.868\times ANDM+2.414\times ANET$$
$$-1.282\times ANCM-37.568$$

$$F_{easy} = 0.913\times NC+8.781\times ANAUW+7.588\times ANMUW+19.309\times ANAsso$$
$$+0.729\times NaggH+5.252\times MaxHAgg+4.003\times NGenH-2.543\times MaxDIT$$

$+3.399\times NOS-0.034\times WMBO+2.941\times ANRM-0.368\times ANDM+3.096\times ANET$

$-1.353\times ANCM-46.637$

เมื่อ $F_{difficult}$ คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับยาก

F_{medium} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับปานกลาง

F_{easy} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับง่าย

ผลลัพธ์ในตารางที่ 5.9 วิธีครอสแวริเดชัน (Cross-validation method) หมายถึงการทำนายโดยใช้ข้อมูล 34 ระบบในการสร้างโมเดลการทำนาย เพื่อพยากรณ์ระบบที่เหลือ 1 ระบบ ซึ่งจะมีการคำนวณทั้งหมดเท่ากับจำนวน 35 ครั้ง เพื่อคำนวณเปอร์เซ็นต์ความถูกต้องในการจัดกลุ่ม ในกลุ่มระบบที่อยู่ในระดับยากจำนวน 4 ระบบ ทำนายว่าอยู่ในระดับยาก 2 ระบบ หรือทำนายถูกร้อยละ 50 ((2/4)×100) ทำนายว่าอยู่ในระดับปานกลางจำนวน 1 ระบบ หรือทำนายผิดร้อยละ 25 ((1/4)×100) ทำนายว่าอยู่ในระดับง่ายจำนวน 1 ระบบหรือทำนายผิดร้อยละ 25 ((1/4)×100) ในกลุ่มระบบที่อยู่ในระดับปานกลางจำนวน 25 ระบบ ทำนายว่าอยู่ในระดับปานกลาง 19 ระบบ หรือทำนายถูกร้อยละ 76 ((19/25)×100) ทำนายว่าอยู่ในระดับง่ายจำนวน 6 ระบบ หรือทำนายผิดร้อยละ 24 ((6/25)×100) และในกลุ่มระบบที่อยู่ในระดับง่ายจำนวน 6 ระบบ ทำนายว่าอยู่ในระดับปานกลางจำนวน 5 ระบบ หรือทำนายผิดร้อยละ 83.3 ((5/6)×100) ทำนายว่าอยู่ในระดับง่ายจำนวน 1 ระบบ หรือทำนายถูกร้อยละ 16.7 ((1/6)×100)

ดังนั้นการทำนายด้วยวิธีครอสแวริเดชัน สามารถทำนายถูกต้องคิดเป็นค่าเฉลี่ย 62.9% (((2+19+1)/35)×100)

5.4 การตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

หลังจากสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ได้นำโมเดลการทำนายที่ได้มาทำการตรวจสอบความถูกต้องของการทำนายด้วยระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 5 ระบบ จากผลการตรวจสอบความถูกต้องของโมเดลการทำนายความสามารในการบำรุงรักษาซอฟต์แวร์พบว่ามีระบบจำนวน 3 ระบบที่สามารถทำนายอยู่ในระดับที่ถูกต้อง ดังแสดงในตารางที่ 5.10

งานวิจัยนี้ได้ทดลองปรับจำนวนข้อมูลในชุดข้อมูลสอนและชุดข้อมูลทดสอบเป็น 25 ระบบและ 15 ระบบ กับ 30 ระบบและ 10 ระบบ เพื่อพิจารณาความแตกต่างของจำนวนความถูกต้อง ผลการทดลองของชุดข้อมูลทั้ง 2 กลุ่มแสดงอยู่ในภาคผนวก ช

ตารางที่ 5.5 แสดงค่าความสัมพันธ์ด้วยวิธีการวิเคราะห์หาความสัมพันธ์แบบเพียร์สัน

		NC	ANAUW	ANAW	ANMUW	ANMW	ANASS	ANAGG	NAGGH	MAXHAGG	ANGEN	NGENH	MAXDIT	NOS	WMBO	ANRM	ANDM	ANET	ANCM
NC	Pearson Correlation	1	-.369*	.075	-.145	-.152	-.163	-.007	.273	-.062	.341*	.596**	.353*	.123	-.071	-.356*	-.088	-.109	-.146
	Sig. (2-tailed)	.	.029	.667	.405	.383	.350	.967	.112	.722	.045	.000	.037	.481	.684	.036	.613	.534	.402
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANAUW	Pearson Correlation	-.369*	1	.259	-.099	-.093	.188	-.189	-.126	-.049	-.123	-.233	.062	-.132	-.041	.027	.009	-.115	-.081
	Sig. (2-tailed)	.029	.	.133	.571	.597	.280	.277	.472	.782	.481	.179	.723	.449	.816	.879	.959	.509	.644
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANAW	Pearson Correlation	.075	.259	1	.154	.146	-.404*	-.175	-.102	-.145	.684**	.451**	.820**	-.032	-.147	.038	-.068	-.098	-.218
	Sig. (2-tailed)	.667	.133	.	.376	.403	.016	.316	.561	.404	.000	.007	.000	.856	.400	.830	.697	.574	.208
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANMUW	Pearson Correlation	-.145	-.099	.154	1	.999**	-.268	.183	-.195	.189	-.086	-.090	-.102	.198	.579**	.009	.267	-.132	.177
	Sig. (2-tailed)	.405	.571	.376	.	.000	.119	.293	.262	.276	.624	.606	.561	.254	.000	.961	.121	.450	.308
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANMW	Pearson Correlation	-.152	-.093	.146	.999**	1	-.261	.174	-.201	.189	-.097	-.104	-.107	.198	.576**	.000	.276	-.134	.187
	Sig. (2-tailed)	.383	.597	.403	.000	.	.130	.317	.246	.276	.580	.552	.539	.253	.000	.999	.109	.442	.281
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANASS	Pearson Correlation	-.163	.188	-.404*	-.268	-.261	1	-.160	-.016	-.234	-.482**	-.479**	-.392*	-.030	-.152	-.042	-.224	-.117	.272
	Sig. (2-tailed)	.350	.280	.016	.119	.130	.	.358	.928	.176	.003	.004	.020	.865	.382	.809	.195	.504	.113
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANAGG	Pearson Correlation	-.007	-.189	-.175	.183	.174	-.160	1	.655**	.716**	-.291	-.187	-.257	.055	.371*	-.047	.033	.160	-.196
	Sig. (2-tailed)	.967	.277	.316	.293	.317	.358	.	.000	.000	.089	.283	.136	.752	.028	.789	.851	.358	.258
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
NAGGH	Pearson Correlation	.273	-.126	-.102	-.195	-.201	-.016	.655**	1	.487**	-.181	-.026	-.108	.199	-.125	-.102	-.037	.017	-.075
	Sig. (2-tailed)	.112	.472	.561	.262	.246	.928	.000	.	.003	.299	.880	.536	.251	.475	.560	.834	.921	.667
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
MAXHAGG	Pearson Correlation	-.062	-.049	-.145	.189	.189	-.234	.716**	.487**	1	-.324	-.174	-.291	.067	.217	-.124	.250	.206	-.261
	Sig. (2-tailed)	.722	.782	.404	.276	.276	.176	.000	.003	.	.057	.317	.090	.704	.211	.479	.148	.234	.131
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANGEN	Pearson Correlation	.341*	-.123	.684**	-.086	-.097	-.482**	-.291	-.181	-.324	1	.879**	.852**	-.023	-.188	.044	-.067	.107	-.073
	Sig. (2-tailed)	.045	.481	.000	.624	.580	.003	.089	.299	.057	.	.000	.000	.895	.280	.801	.704	.542	.676
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
NGENH	Pearson Correlation	.596**	-.233	.451**	-.090	-.104	-.479**	-.187	-.026	-.174	.879**	1	.675**	.016	-.187	-.051	-.023	.078	-.089
	Sig. (2-tailed)	.000	.179	.007	.606	.552	.004	.283	.880	.317	.000	.	.000	.927	.281	.769	.893	.657	.610
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
MAXDIT	Pearson Correlation	.353*	.062	.820**	-.102	-.107	-.392*	-.257	-.108	-.291	.852**	.675**	1	.009	-.101	-.123	-.026	.082	-.173
	Sig. (2-tailed)	.037	.723	.000	.561	.539	.020	.136	.536	.090	.000	.000	.	.960	.566	.483	.882	.641	.321
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
NOS	Pearson Correlation	.123	-.132	-.032	.198	.198	-.030	.055	.199	.067	-.023	.016	.009	1	.053	-.093	-.173	-.398*	.082
	Sig. (2-tailed)	.481	.449	.856	.254	.253	.865	.752	.251	.704	.895	.927	.960	.	.763	.596	.320	.018	.641
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
WMBO	Pearson Correlation	-.071	-.041	-.147	.579**	.576**	-.152	.371*	-.125	.217	-.188	-.187	-.101	.053	1	-.029	.192	.072	.095
	Sig. (2-tailed)	.684	.816	.400	.000	.000	.382	.028	.475	.211	.280	.281	.566	.763	.	.869	.268	.682	.586
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANRM	Pearson Correlation	-.356*	.027	.038	.009	.000	-.042	-.047	-.102	-.124	.044	-.051	-.123	-.093	-.029	1	-.132	.001	.224
	Sig. (2-tailed)	.036	.879	.830	.961	.999	.809	.789	.560	.479	.801	.769	.483	.596	.869	.	.448	.996	.197
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANDM	Pearson Correlation	-.088	.009	-.068	.267	.276	-.224	.033	-.037	.250	-.067	-.023	-.026	-.173	.192	-.132	1	.475**	.152
	Sig. (2-tailed)	.613	.959	.697	.121	.109	.195	.851	.834	.148	.704	.893	.882	.320	.268	.448	.	.004	.384
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANET	Pearson Correlation	-.109	-.115	-.098	-.132	-.134	-.117	.160	.017	.206	.107	.078	.082	-.398*	.072	.001	.475**	1	-.073
	Sig. (2-tailed)	.534	.509	.574	.450	.442	.504	.358	.921	.234	.542	.657	.641	.018	.682	.996	.004	.	.677
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
ANCM	Pearson Correlation	-.146	-.081	-.218	.177	.187	.272	-.196	-.075	-.261	-.073	-.089	-.173	.082	.095	.224	.152	-.073	1
	Sig. (2-tailed)	.402	.644	.208	.308	.281	.113	.258	.667	.131	.676	.610	.321	.641	.586	.197	.384	.677	.
	N	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35

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

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

ตารางที่ 5.6 แสดงค่าความสัมพันธ์ระหว่าง 2 มาตรวัด

มาตรวัดที่สัมพันธ์กัน	ความสัมพันธ์แบบเพียร์สัน	ระดับนัยสำคัญ (2 ข้าง)
ANMUW & ANMW	0.999	0.01
ANGen & NGenH	0.879	0.01
ANGen & MaxDIT	0.852	0.01
ANAW & MaxDIT	0.820	0.01
ANAgg & MaxHAgg	0.716	0.01

ตารางที่ 5.7 แสดงค่าอาร์สแควร์ที่ปรับแล้วของมาตรวัด

มาตรวัด	ค่าอาร์สแควร์ที่ปรับแล้ว	ค่าความผิดพลาดของการประมาณ
NGenH	0.275	7.664190
MaxDIT	0.121	8.438642
ANGen	0.109[1]	8.498086
ANAW	0.053[1]	8.759590
MaxHAgg	-0.030	9.137311
ANMUW	-0.028	9.127916
ANMW	-0.028[1]	9.127125
ANAgg	0.001[1]	8.997236

ตารางที่ 5.8 แสดงค่าสัมประสิทธิ์ของมาตรวัดแต่ละตัวของฟังก์ชันการจำแนกกลุ่มของฟิชเชอร์

มาตรวัด	ความสามารถในการบำรุงซอฟต์แวร์		
	ระดับยาก	ระดับปานกลาง	ระดับง่าย
NC	0.980	0.825	0.913
ANAUW	7.856	6.560	8.781
ANMUW	10.146	4.614	7.588
ANAss	27.655	20.022	19.309
NaggH	0.287	0.254	0.729
MaxHAgg	11.547	4.801	5.252

ตารางที่ 5.8 แสดงค่าสัมประสิทธิ์ของมาตรวัดแต่ละตัวของฟังก์ชันการจำแนกกลุ่มของฟิชเชอร์ (ต่อ)

มาตรวัด	ความสามารถในการบำรุงซอฟต์แวร์		
	ระดับยาก	ระดับปานกลาง	ระดับง่าย
NGenH	13.096	3.332	4.003
MaxDIT	-4.062	0.282	-2.543
NOS	-0.737	3.656	3.399
WMBO	-2.393	0.897	-0.034
ANRM	0.192	3.858	2.941
ANDM	0.215	0.868	-0.368
ANET	-0.672	2.414	3.096
ANCM	-2.510	-1.282	-1.353
(Constant)	-60.615	-37.568	-46.637

ตารางที่ 5.9 แสดงเปอร์เซ็นต์ความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์เมื่อใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			รวม
			ยาก	ปานกลาง	ง่าย	
ครอสแวริเดชัน	จำนวน	ยาก	2	1	1	4
		ปานกลาง	0	19	6	25
		ง่าย	0	5	1	6
	เปอร์เซ็นต์	ยาก	50.0	25.0	25.0	100.0
		ปานกลาง	0.0	76.0	24.0	100.0
		ง่าย	0.0	83.3	16.7	100.0

ตารางที่ 5.10 แสดงผลการตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้เมื่อใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			
			ยาก	ปานกลาง	ง่าย	รวม
กลุ่มที่ 3	จำนวน	ยาก	1	1	0	2
		ปานกลาง	0	1	1	2
		ง่าย	0	0	1	1
	เปอร์เซ็นต์	ยาก	50.0	50.0	0.0	100.0
		ปานกลาง	0.0	50.0	50.0	100.0
		ง่าย	0.0	0.0	100.0	100.0

5.5 ผลการเปรียบเทียบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

จากการทดลองสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยกลุ่มข้อมูลทั้ง 3 กลุ่ม คือ กลุ่มที่ 1 สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 25 ระบบและชุดข้อมูลทดสอบจำนวน 15 ระบบ กลุ่มที่ 2 สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 30 ระบบและชุดข้อมูลทดสอบจำนวน 10 ระบบ กลุ่มที่ 3 สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบ

ตารางที่ 5.11 แสดงจำนวนความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 3 กลุ่มข้อมูล

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			รวม
			ยาก	ปานกลาง	ง่าย	
ครอสแวริเดชัน กลุ่มที่ 1	จำนวน	ยาก	0	1	2	3
		ปานกลาง	2	12	5	19
		ง่าย	0	3	0	3
ครอสแวริเดชัน กลุ่มที่ 2	จำนวน	ยาก	2	2	0	4
		ปานกลาง	0	16	6	22
		ง่าย	0	4	0	4
ครอสแวริเดชัน กลุ่มที่ 3	จำนวน	ยาก	2	1	1	4
		ปานกลาง	0	19	6	25
		ง่าย	0	5	1	6

ตารางที่ 5.11 แสดงจำนวนความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 3 กลุ่มด้วยวิธีครอสแวริเดชัน พบว่ากลุ่มที่ 1 มีจำนวนความถูกต้องในการทำนายเท่ากับ 12 ใน 25 ระบบ หรือคิดเป็น 48 เปอร์เซ็นต์ กลุ่มที่ 2 มีจำนวนความถูกต้องในการทำนายเท่ากับ 18 ใน 30 ระบบ หรือคิดเป็น 60 เปอร์เซ็นต์ กลุ่มที่ 3 มีจำนวนความถูกต้องในการทำนายเท่ากับ 22 ใน 35 ระบบ หรือคิดเป็น 62.9 เปอร์เซ็นต์

เมื่อนำระบบในชุดทดสอบของแต่ละกลุ่มมาตรวจสอบความถูกต้องในการทำนาย พบว่ากลุ่มที่ 1 มีจำนวนระบบที่ทำนายถูกต้องเท่ากับ 8 ใน 15 หรือคิดเป็น 53.3 เปอร์เซ็นต์ กลุ่มที่ 2 มีจำนวนระบบที่ทำนายถูกต้องเท่ากับ 6 ใน 10 หรือคิดเป็น 60 เปอร์เซ็นต์ กลุ่มที่ 3 มีจำนวนระบบที่ทำนายถูกต้องเท่ากับ 3 ใน 5 หรือคิดเป็น 60 เปอร์เซ็นต์

จากผลการเปรียบเทียบทั้ง 3 กลุ่มพบว่าจำนวนความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยวิธีครอสแวริเดชันในกลุ่มที่ 3 มีความถูกต้องสูงที่สุดคือ 62.9 เปอร์เซ็นต์ ส่วนการตรวจสอบความถูกต้องในการทำนายความสามารถในการ

บำรุงรักษาซอฟต์แวร์ด้วยระบบในชุดทดสอบของแต่ละกลุ่ม พบว่ากลุ่มที่ 2 และกลุ่มที่ 3 มีความถูกต้องในการทำนายเท่ากันคือ 60 เปอร์เซ็นต์

ดังนั้นงานวิจัยนี้จึงใช้โมเดลการทำนายที่สร้างจากการใช้ระบบที่ใช้เป็นชุดข้อมูลสอนจำนวน 35 ระบบ และระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 5 ระบบ (กลุ่มที่ 3) เป็นโมเดลในการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์

ตารางที่ 5.12 แสดงผลการตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 3 กลุ่มข้อมูล

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			
			ยาก	ปานกลาง	ง่าย	รวม
กลุ่มที่ 1	จำนวน	ยาก	2	1	1	4
		ปานกลาง	2	4	1	7
		ง่าย	1	1	2	4
กลุ่มที่ 2	จำนวน	ยาก	1	1	0	2
		ปานกลาง	1	3	1	5
		ง่าย	1	0	2	3
กลุ่มที่ 3	จำนวน	ยาก	1	1	0	2
		ปานกลาง	0	1	1	2
		ง่าย	0	0	1	1

5.6 การแปลผลที่ได้จากการทดลอง

จากโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้จากการสร้างโดยใช้ระบบที่ใช้เป็นชุดข้อมูลสอนจำนวน 35 ระบบและระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 5 ระบบ ตามสมการ $F_{difficult}$ F_{medium} และ F_{easy} ดังนี้

$$F_{difficult} = 0.98\times NC+7.856\times ANAUW+10.146\times ANMUW+27.655\times ANAsso$$
$$+0.287\times NaggH+11.547\times MaxHAgg+13.096\times NGenH-4.062\times MaxDIT$$
$$-0.737\times NOS-2.393\times WMBO+0.192\times ANRM+0.215\times ANDM-0.672\times ANET$$
$$-2.51\times ANCM-60.615$$

$$F_{medium} = 0.825\times NC+6.56\times ANAUW+4.614\times ANMUW+20.022\times ANAsso$$
$$+0.254\times NaggH+4.801\times MaxHAgg+3.332\times NGenH+0.282\times MaxDIT$$
$$+3.656\times NOS+0.897\times WMBO+3.858\times ANRM+0.868\times ANDM+2.414\times ANET$$
$$-1.282\times ANCM-37.568$$

$$F_{easy} = 0.913\times NC+8.781\times ANAUW+7.588\times ANMUW+19.309\times ANAsso$$
$$+0.729\times NaggH+5.252\times MaxHAgg+4.003\times NGenH-2.543\times MaxDIT$$

+3.399×NOS-0.034×WMBO+2.941×ANRM-0.368×ANDM+3.096×ANET

-1.353×ANCM-46.637

พบว่ามาตรวัด NC ANAUW ANMUW ANAsso NaggH MaxHAgg NGenH MaxDIT NOS WMBO ANRM ANDM ANET และ ANCM มีผลต่อการจำแนกกลุ่มของการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ และมาตรวัด ANAW, ANMW, ANAgg และ ANGen ไม่มีผลต่อการจำแนกกลุ่มของการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์

เมื่อพิจารณาโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์พบว่าค่าสัมประสิทธิ์ของมาตรวัดแต่ละมาตรวัดทั้ง 3 สมการมีผลต่อการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ในแต่ละกลุ่มต่างกัน เช่น จากค่าสัมประสิทธิ์ของมาตรวัด NC ในสมการ $F_{difficult}$ มีค่าเท่ากับ 0.98 ในสมการ F_{medium} มีค่าเท่ากับ 0.825 และในสมการ F_{easy} มีค่าเท่ากับ 0.913 แสดงว่ามาตรวัด NC มีผลต่อการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ในกลุ่มระดับยากสูงกว่าในกลุ่มระดับง่ายและปานกลาง และมีผลต่อการทำนายในทิศทางเดียวกัน เนื่องจากมีค่าสัมประสิทธิ์เป็นค่าบวก หมายความว่าถ้าค่า NC มากจะมีผลต่อการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ในกลุ่มนั้นมาก ในกรณีที่ค่าสัมประสิทธิ์เป็นค่าลบ หมายความว่าถ้าค่า NC มากจะมีผลต่อการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ในกลุ่มนั้นน้อย

จากการพิจารณาเปอร์เซ็นต์ความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ด้วยวิธีครอสแวริเดชันมีค่าเปอร์เซ็นต์ความถูกต้องเท่ากับ 62.9 เปอร์เซ็นต์ และเปอร์เซ็นต์ความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ด้วยระบบที่ใช้เป็นชุดข้อมูลสอบ 5 ระบบ พบว่ามีค่าเปอร์เซ็นต์ความถูกต้องเท่ากับ 60 เปอร์เซ็นต์ ซึ่งถือว่ามีค่าความถูกต้องอยู่ในระดับปานกลาง จากผลการดำเนินการทดลองพบว่ามีปัจจัยหลายอย่างที่มีความเกี่ยวข้องและมีผลกระทบต่อผลการทดลอง และเปอร์เซ็นต์ความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาที่ได้ ปัจจัยเหล่านั้น ได้แก่

1) ระบบที่นำมาใช้ในการทดลอง

แผนภาพคลาส และแผนภาพซีเควนซ์ของระบบทั้ง 40 ระบบที่นำมาใช้ในการทดลองสร้างจากระบบทั่วๆ ไปซึ่งมีโดเมนของระบบที่แตกต่างกัน แต่เนื่องจากระบบที่นำมาใช้ในการทดลองมีความซับซ้อนของระบบน้อยและสามารถเข้าใจได้ง่าย ดังนั้นจึงมีผลทำให้หน่วยทดลองสามารถเข้าใจและสามารถแก้ไขระบบได้ง่ายด้วย

ระบบที่นำมาใช้มีจำนวนคลาสมากสุดเพียง 25 คลาส ซึ่งเป็นข้อจำกัดของงานวิจัยนี้เนื่องจากไม่สามารถหาระบบที่มีการใช้งานอยู่จริงมาใช้เป็นระบบในการทดลองได้

2) หน่วยทดลอง

หน่วยทดลองในงานวิจัยนี้ คือ นิสิตระดับปริญญาโท ซึ่งส่วนใหญ่มีประสบการณ์ด้านการออกแบบซอฟต์แวร์ด้วยยูเอ็มแอลในระดับปานกลางถึงมาก และผ่านการเรียนในวิชาเทคโนโลยีเชิงวัตถุ และวิศวกรรมความต้องการซอฟต์แวร์ ทำให้ผลที่ได้จากการทำข้อสอบของหน่วยทดลองแต่ละคนมีความน่าเชื่อถือระดับหนึ่ง

หน่วยทดลองแต่ละคนทำข้อสอบคนละ 2 ระบบ และข้อสอบทั้ง 2 ระบบเป็นข้อสอบที่มีโดเมนของระบบแตกต่างกัน ดังนั้นผลกระทบจากการเรียนรู้ในตัวข้อสอบจึงมีน้อยมาก และในระหว่างการทำข้อสอบมีการหยุดพักเพื่อให้หน่วยทดลองไม่เหนื่อยล้าจากการทำข้อสอบเกินไป ระหว่างสอบได้จัดให้หน่วยทดลองที่ทำข้อสอบระบบต่างกันนั่งใกล้กัน เพื่อป้องกันการปรึกษากัน

3) แบบข้อสอบ

แบบข้อสอบที่สร้างขึ้นสำหรับระบบทั้ง 40 ระบบเน้นทดสอบในด้านความสามารถในการเข้าใจ และความสามารถในการปรับเปลี่ยนของแผนภาพคลาสและแผนภาพซีเควนซ์ ซึ่งข้อสอบส่วนใหญ่เป็นคำถามแบบอัตนัยให้หน่วยทดลองเขียนตอบ และป้องกันไม่ให้หน่วยทดลองเดาคำตอบได้ เนื่องจากต้องสร้างคำถามให้สอดคล้องกับระบบแต่ละระบบ ดังนั้นคำถามที่สร้างขึ้นอาจมีความยากและง่ายแตกต่างกัน

4) มาตรวัด

มาตรวัดที่นำเสนอในงานวิจัยนี้เป็นมาตรวัดสำหรับระบบ ซึ่งผู้วิจัยจึงได้ทำการดัดแปลงมาจากมาตรวัดที่ใช้สำหรับคลาส โดยการหาผลรวมและเฉลี่ยเพื่อให้เป็นมาตรวัดสำหรับระบบ ซึ่งมาตรวัดที่ได้อาจไม่มีความเหมาะสมในการทดลองกรณีที่ค่ามาตรวัดสำหรับคลาสแต่ละคลาสในระบบมีความแตกต่างกันมาก แต่เมื่อนำมาหาค่าเฉลี่ยแล้วอาจทำให้ค่าที่ได้กลายเป็นค่าเฉลี่ยแบบปกติ ทำให้ไม่มีความแตกต่างกันระหว่างระบบ ซึ่งทำให้เกิดความผิดพลาดในการสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ได้

5.7 การนำโมเดลการทำนายไปใช้งาน

ตัวอย่างการนำโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้จากผลการทดลองของกลุ่มที่ 3 โดยใช้ชุดข้อมูลสอนจำนวน 35 ระบบและชุดข้อมูลทดสอบจำนวน 5 ระบบไปใช้งาน มีขั้นตอนการใช้งานดังนี้

1. คำนวณค่ามาตรวัด 8 มาตรวัด คือ มาตรวัด NC ANAUW ANMUW ANAsso NaggH MaxHAgg NGenH และ MaxDIT จากแผนภาพคลาสและค่ามาตรวัด 6 มาตรวัด คือ มาตรวัด NOS WMBO ANRM ANDM ANET และ ANCM จากแผนภาพซีเควนซ์

2. แทนค่ามาตรวัดแต่ละตัวลงในฟังก์ชันการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ทั้ง 3 ฟังก์ชัน คือ ฟังก์ชัน $F_{difficult}$ F_{medium} และ F_{easy} ที่แสดงอยู่ในหัวข้อที่ 5.3

3. เปรียบเทียบค่าที่คำนวณได้จากฟังก์ชัน $F_{difficult}$ F_{medium} และ F_{easy} ว่าค่าที่คำนวณได้จากฟังก์ชันใดมีค่าสูงที่สุด ระบบนั้นจะถูกทำนายให้อยู่ในกลุ่มนั้น ตัวอย่างเช่น ถ้าค่าที่คำนวณได้จากฟังก์ชัน $F_{difficult}$ มีค่าสูงกว่าค่าที่ได้คำนวณได้จากฟังก์ชัน F_{medium} และ F_{easy} ระบบนั้นจะถูกทำนายให้อยู่ในกลุ่มที่มีระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ยาก เป็นต้น

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

วิทยานิพนธ์นี้ ได้ทำการสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ที่สามารถทำนายได้ 3 ระดับ คือ ระดับยาก ปานกลาง และง่าย โดยใช้มาตรวัดการวิเคราะห์และออกแบบระบบเชิงวัตถุจากแผนภาพคลาสและแผนภาพซีเควนซ์จำนวน 18 มาตรวัด ซึ่งแบ่งเป็นมาตรวัดที่คำนวณจากแผนภาพคลาส 12 มาตรวัดและมาตรวัดที่คำนวณจากแผนภาพซีเควนซ์ 6 มาตรวัด พร้อมทั้งออกแบบและทำการทดลองเพื่อเก็บรวบรวมข้อมูลเพื่อนำไปสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ด้วยวิธีการวิเคราะห์จำแนกกลุ่ม โดยระบบที่นำมาใช้ในการทดลองจำนวน 40 ระบบ แบ่งออกเป็น ระบบที่ใช้เป็นข้อมูลสอนสำหรับการสร้างโมเดลการทำนายจำนวน 35 ระบบ และระบบที่ใช้เป็นข้อมูลทดสอบสำหรับตรวจสอบความถูกต้องของโมเดลจำนวน 5 ระบบ

ผลการทดลองพบว่ามาตรวัดการวิเคราะห์และออกแบบระบบเชิงวัตถุที่มีความสัมพันธ์กับความสามารถในการบำรุงรักษาซอฟต์แวร์ มีจำนวน 14 มาตรวัด ได้แก่ มาตรวัด NC ANAUW ANMUW ANAsso NaggH MaxHAgg NGenH MaxDIT NOS WMBO ANRM ANDM ANET และ ANCM จากนั้นนำระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 5 ระบบมาตรวจสอบความถูกต้องในการทำนายของโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ พบว่าระบบจำนวน 3 ระบบจาก 5 ระบบสามารถทำนายอยู่ในกลุ่มที่ถูกต้อง คิดเป็น 60 เปอร์เซ็นต์ นอกจากนี้ผู้วิจัยยังได้ทดลองปรับจำนวนข้อมูลในชุดข้อมูลสอนและชุดข้อมูลทดสอบเป็น 25 ระบบและ 15 ระบบ กับ 30 ระบบและ 10 ระบบ เพื่อพิจารณาความแตกต่างของค่าเปอร์เซ็นต์ความถูกต้อง ซึ่งผลการทดลองพบว่าการทดลองทั้ง 3 ครั้งให้ผลค่าเปอร์เซ็นต์ความถูกต้องใกล้เคียงกัน

จากนั้นได้พัฒนาเครื่องมือเพื่อการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ด้วยภาษาจาวา ให้สามารถคำนวณค่ามาตรวัดจำนวน 17 มาตรวัด และทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ได้ ซึ่งข้อมูลนำเข้าสำหรับเครื่องมือที่พัฒนาขึ้น ได้จากการสร้างแผนภาพคลาสและแผนภาพซีเควนซ์ด้วยโปรแกรมเรชันเนลโรสและแปลงแผนภาพให้อยู่ในรูปของภาษาเอ็กซ์เอ็มแอล (Extensible Markup Language - XML) ด้วยโปรแกรมยูนิซิสโรสเอ็กซ์เอ็มแอล

6.2 ข้อเสนอแนะ

1) โมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ สร้างจากมาตรวัดสำหรับแผนภาพคลาสและแผนภาพซีเควนซ์เท่านั้น ดังนั้นงานวิจัยนี้จึงเป็นแนวทางในการนำค่ามาตรวัดที่คำนวณจากแผนภาพอื่นๆ ของยูเอ็มแอล มาพิจารณาประกอบ เพื่อเป็นประโยชน์ในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ให้ดีขึ้นต่อไป

2) งานวิจัยนี้ทำนายระดับความสามารถในการบำรุงรักษาเพียง 3 ระดับ คือ ระดับง่าย ปานกลาง และยาก ดังนั้นถ้าสามารถทำนายระดับความสามารถในการบำรุงรักษาได้ละเอียดขึ้น จะช่วยเพิ่มความสามารถในการตัดสินใจให้กับผู้พัฒนาโปรแกรมได้

3) โมเดลการทำนายที่สร้างขึ้น บอกเพียงแค่ระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ แต่ไม่ได้แนะแนวทางในการแก้ไขโมเดลการวิเคราะห์และออกแบบระบบ ดังนั้นงานวิจัยนี้จะเป็นประโยชน์มากขึ้น ถ้าสามารถบอกตำแหน่งที่ต้องปรับปรุง หรือแก้ไขภายในแผนภาพคลาสและแผนภาพซีเควนซ์ให้ผู้พัฒนาโปรแกรมทราบได้

6.3 ผลงานตีพิมพ์

ระหว่างดำเนินงานวิทยานิพนธ์นี้ได้ตีพิมพ์ผลงานวิจัยในวารสารวิชาการ และการประชุมทั้งในและต่างประเทศ รวมทั้งสิ้น 3 ผลงาน ได้แก่ (รายละเอียดอยู่ในภาคผนวก ซ)

รายการอ้างอิง

1. กัลยา วินิชย์บัญชา. การวิเคราะห์ตัวแปรหลายตัวด้วย SPSS for Windows. กรุงเทพ: โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2544.
2. กัลยา วินิชย์บัญชา. การวิเคราะห์สถิติ : สถิติสำหรับการบริหารและวิจัย. กรุงเทพ: โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.
3. Booch, G., J. Rumbuagh and I. Jacobson. The Unified Modeling Language User Guide. Addison Wesley Longman, Inc., 1999.
4. Dennis, A., B. H. Wixom and D. Tegarden. System Analysis and Design: An Object-Oriented Approach with UML. John Wiley & Sons, Inc., 2002
5. Fenton, E. N. and S. L. Pfleeger. Software Metrics: A Rigorous and Practical Approach. PWS Publishing Company, 1997.
6. Genero, M. and M. Piattini. Empirical validation of measures for class diagram structural complexity through controlled experiments. Proceedings of 11th International on Computer Science Society. 2001.
7. Genero, M., M. Piattini and C. Calero. Early measures for UML class diagrams. L'OBJECT: Software, Databases, Networks 6 (2000): 489-515.
8. Hyoseob, K. and C. Boldyreff. Developing Software Metrics Applicable to UML Models. Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE). 2002.
9. ISO/IEC 9126-1:2000, Information technology – Software product quality – Part 1: Quality model, International Organization for Standardization.
10. Klaus, P. "The three dimensions of requirements engineering: A framework and its applications", Information Systems.19 (1994): 243-258.
11. Lionel, C. B., C. Bunse and J. W. Daly. A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. IEEE Transactions on Software Engineering. 27 (2001): 513-530.

12. Pressman, R. S. Software Engineering: A practitioner's approach. McGraw-Hill, Inc., 1992.
13. Punter, T., R. V. Solingen, J. Trienekens. Software Product Evaluation. Proceedings of the 4th IT Evaluation Conference (EVIT-97). 1997.
14. Stephen, R. S. Classical and Object-Oriented Software Engineering with UML and Java™. McGraw-Hill, 1999.
15. Unisys. Unisys Rose XML Tool [Online]. Available from: http://www.unisys.com/: [2003, April 11].
16. Maloney, E. Values of the Pearson Correlation [Online]. Available from: http://cnx.rice.edu/content/m10950/latest/: [2003, October 3].
17. Quin, L. Extensible Markup Language (XML) [Online]. Available from: http://www.w3c.org/: [2003, April 11].

ภาคผนวก

ภาคผนวก ก
มาตรวัดยูเอ็มแอล

1. **มาตรวัดยูเอ็มแอล (UML Metrics)** จำนวน 27 มาตรวัด แบ่งออกเป็น 4 ประเภท ได้แก่

1.1 มาตรวัดสำหรับโมเดล (Metrics for Model) จำนวน 9 มาตรวัด ได้แก่

1.1.1 มาตรวัด Number of the packages in a model (NPM) คือ จำนวนแพ็คเกจในโมเดล

1.1.2 มาตรวัด Number of the classes in a model (NCM) คือ จำนวนคลาสในโมเดล

1.1.3 มาตรวัด Number of actors in a model (NAM)คือ จำนวนผู้ดำเนินการ (Actor) ในโมเดล

1.1.4 มาตรวัด Number of the use cases in a model (NUM) คือ จำนวนยูสเคสในโมเดล

1.1.5 มาตรวัด Number of the objects in a model (NOM) คือ จำนวนวัตถุในโมเดล

1.1.6 มาตรวัด Number of the messages in a model (NMM) คือ จำนวนสารในโมเดล

1.1.7 มาตรวัด Number of the associations in a model (NASM) คือ จำนวนความสัมพันธ์แบบแอสโซซิเอชันในโมเดล

1.1.8 มาตรวัด Number of the aggregations in a model (NAGM) คือ จำนวนความสัมพันธ์แบบแอกกริเกชันในโมเดล

1.1.9 มาตรวัด Number of the inheritance relations in a model (NIM) คือ จำนวนความสัมพันธ์แบบเจเนอรัลไรเซชันในโมเดล

1.2 มาตรวัดสำหรับคลาส (Metrics for Class) จำนวน 13 มาตรวัด ได้แก่

1.2.1 มาตรวัด Number of the attributes in a class – unweighted (NATC1) คือ จำนวนคุณลักษณะภายในคลาสที่กำลังพิจารณา โดยไม่มีการถ่วงค่าให้กับ ตัวดัดแปร ของคุณลักษณะที่เป็นพับลิค โพรเทค และไพรเวท

1.2.2 มาตรวัด Number of the attributes in a class – weighted (NATC2) คือ จำนวนคุณลักษณะภายในคลาสที่กำลังพิจารณา โดยมีการถ่วงค่าให้กับ ตัว

ดัดแปร ของคุณลักษณะที่เป็นพับลิคเท่ากับ 1.0 โพรเทคเท่ากับ 0.5 และไพรเวทเท่ากับ 0.0

1.2.3 มาตรวัด Number of the operations in a class – unweighted (NOPC1) คือ จำนวนเมทธอดในคลาสที่กำลังพิจารณา โดยไม่มีการถ่วงค่าให้กับ ตัวดัดแปรของเมทธอดที่เป็นพับลิค โพรเทค และไพรเวท

1.2.4 มาตรวัด Number of the operations in a class – weighted (NOPC2) คือ จำนวนเมทธอดภายในคลาสที่กำลังพิจารณา โดยมีการถ่วงค่าให้กับ ตัวดัดแปร ของเมทธอดที่เป็นพับลิคเท่ากับ 1.0 โพรเทคเท่ากับ 0.5 และไพรเวทเท่ากับ 0.0

1.2.5 มาตรวัด Number of the associations linked to a class (NASC) คือ จำนวนความสัมพันธ์แบบแอสโซซิเอชันของคลาสที่กำลังพิจารณา โดยนับความสัมพันธ์แบบแอกกริเกชันรวมด้วย

1.2.6 มาตรวัด Coupling between classes (CBC) คือ จำนวนความสัมพันธ์แอสโซซิเอชันกันภายในคลาสกำลังพิจารณาและคุณลักษณะ

1.2.7 มาตรวัด Depth of inheritance tree (DIT) คือ ระดับความลึกของการสืบทอดคุณสมบัติ โดยนับจากจำนวนระดับ (level) จากคลาสที่กำลังพิจารณาจนถึงราก (Root)

1.2.8 มาตรวัด Number of the superclasses of a class (NSUPC) คือ จำนวนคลาสบรรพบุรุษ (Parent class) แบบโดยตรง (Direct) ของคลาสที่กำลังพิจารณา

1.2.9 มาตรวัด Number of the elements in the transitive closure of the superclasses of a class (NSUPC*) คือ จำนวน Transitive closure ของคลาสบรรพบุรุษของคลาสที่กำลังพิจารณา

1.2.10 มาตรวัด Number of the subclasses of a class (NSUBC) คือ จำนวนคลาสลูก แบบโดยตรงของคลาสที่กำลังพิจารณา

1.2.11 มาตรวัด Number of the elements in the transitive closure of the subclasses of a class NSUBC*) คือ จำนวน Transitive closure ของคลาสลูกหลานของคลาสที่กำลังพิจารณา

1.2.12 มาตรวัด Number of messages sent by the instantiated objects of a class (NMSC) คือ จำนวนสารที่ส่งออกไปโดยวัตถุที่ถูกสร้างจากคลาสที่กำลังพิจารณา

1.2.13 มาตรวัด Number of messages received by the instantiated objects of a class (NMRC) คือ จำนวนสารที่รับเข้ามาโดยวัตถุที่ถูกสร้างจากคลาสที่กำลังพิจารณา

1.3 **มาตรวัดสำหรับสาร (Metrics for Message)** จำนวน 2 มาตรวัด ได้แก่

1.3.1 มาตรวัด Number of the directly dispatched messages of a message (NDM) คือ จำนวนสารที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่น

1.3.2 มาตรวัด Number of the elements in the transitive closure of the directly dispatched messages of a message (NDM*) คือ จำนวนอิลิเมนต์ใน Transitive closure ของสารที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่น

1.4 **มาตรวัดสำหรับยูสเคส (Metrics for Use Case)** จำนวน 3 มาตรวัด ได้แก่

1.4.1 มาตรวัด Number of actors associated with a use case (NAU) คือ จำนวนผู้ดำเนินการที่สัมพันธ์กับยูสเคสที่กำลังพิจารณา

1.4.2 มาตรวัด Number of message associated with a use case (NMU) คือ จำนวนสารที่สัมพันธ์กับยูสเคสที่กำลังพิจารณา

1.4.3 มาตรวัด Number of system classes associated with a use case (NSCU) คือ จำนวนคลาสที่เกี่ยวข้องกับยูสเคสที่กำลังพิจารณา

ภาคผนวก ข
มาตรวัดเชิงวัตถุสำหรับแผนภาพคลาส

1. **พิจารณาจากความสัมพันธ์แบบแอสโซซิเอชัน** ได้แก่
   - 1.1 **มาตรวัดขอบข่ายของคลาส** ได้แก่
     - 1.1.1 มาตรวัด Number of Associations of a Class (NAC) คือ จำนวนผลรวมของความสัมพันธ์แบบแอสโซซิเอชันของคลาสในแผนภาพคลาส
   - 1.2 **มาตรวัดขอบข่ายของแพ็คเกจ** ได้แก่
     - 1.2.1 Number of Associations in a Package (NAP) คือ จำนวนผลรวมของความสัมพันธ์แบบแอสโซซิเอชันภายในแพ็คเกจ
     - 1.2.2 Number of Associations vs. Classes in a Package (NAVCP) คือ อัตราส่วนระหว่างจำนวนความสัมพันธ์แบบแอสโซซิเอชันในแพ็คเกจหารด้วยจำนวนคลาสภายในแพ็คเกจ

2. **พิจารณาจากความสัมพันธ์แบบแอกกริเกชัน** ได้แก่
   - 2.1 **มาตรวัดขอบข่ายของคลาส** ได้แก่
     - 2.1.1 Height of Aggregation (HAgg) คือ ทางเดิน (Path) ที่ยาวที่สุดจากคลาสไปยังลีฟ (leaves)
     - 2.1.2 Number of Direct Parts (NODP) คือ จำนวนผลรวมของคลาส “direct path” ซึ่งประกอบด้วย composite class
     - 2.1.3 Number of Parts (NP) คือ จำนวนของคลาสลูกหลาน (descendant) ของคลาสนั้น
     - 2.1.4 Number of Wholes (NW) คือ จำนวนของคลาสบรรพบุรุษ (predecessor) ของคลาสนั้น
     - 2.1.5 Multiple Aggregation (MAgg)
   - 2.2 **มาตรวัดขอบข่ายของแพ็คเกจ** ได้แก่
     - 2.2.1 Number of Aggregation Relationships (NAggR) คือ จำนวนความสัมพันธ์แบบแอกกริเกชันภายในแพ็คเกจ

3. **พิจารณาจากความสัมพันธ์แบบดีเพนเดนซี** ได้แก่

3.1 **มาตรวัดขอบข่ายของคลาส** ได้แก่

3.1.1 Number of Dependencies In (NDepIn) คือ จำนวนของคลาสที่ขึ้นอยู่กับคลาสที่กำลังพิจารณา

3.1.2 Number of Dependencies Out (NDepOut) คือ จำนวนของคลาสซึ่งคลาสที่กำลังพิจารณาไปขึ้นอยู่กับคลาสเหล่านั้น

3.2 **มาตรวัดขอบข่ายของแพ็คเกจ** ได้แก่

3.2.1 Number of Dependencies In (NDepIn) ความหมายเหมือนมาตรวัด NDepIn

3.2.2 Number of Dependencies Out (NDepOut) ความหมายเหมือนมาตรวัด NDepOut

ภาคผนวก ค
ตารางเปรียบเทียบมาตรวัด

Scope → / Proposals ↓	Classes					
	Attributes	Methods	Relationships			
			Gen	Agg	Assoc	Dep
Lorenz and Kidd [LOR94]	X	X	X			
Chidamber and Kemerer [CHI 94]		X	X			
Brito e Abreu and Melo [BRI 96]						
Marchesi [MAR 98]	X		X	X		

รูปที่ ค-1 แสดงตารางการเปรียบเทียบมาตรวัดขอบข่ายของคลาส

Scope → / Proposals ↓	Packages					
	Attributes	Methods	Relationships			
			Gen	Agg	Assoc	Dep
Lorenz and Kidd [LOR94]						
Chidamber and Kemerer [CHI 94]						
Brito e Abreu and Melo [BRI 96]	X	X	X			
Marchesi [MAR 98]	X		X	X		

รูปที่ ค-2 แสดงตารางการเปรียบเทียบมาตรวัดขอบข่ายของแพ็คเกจ

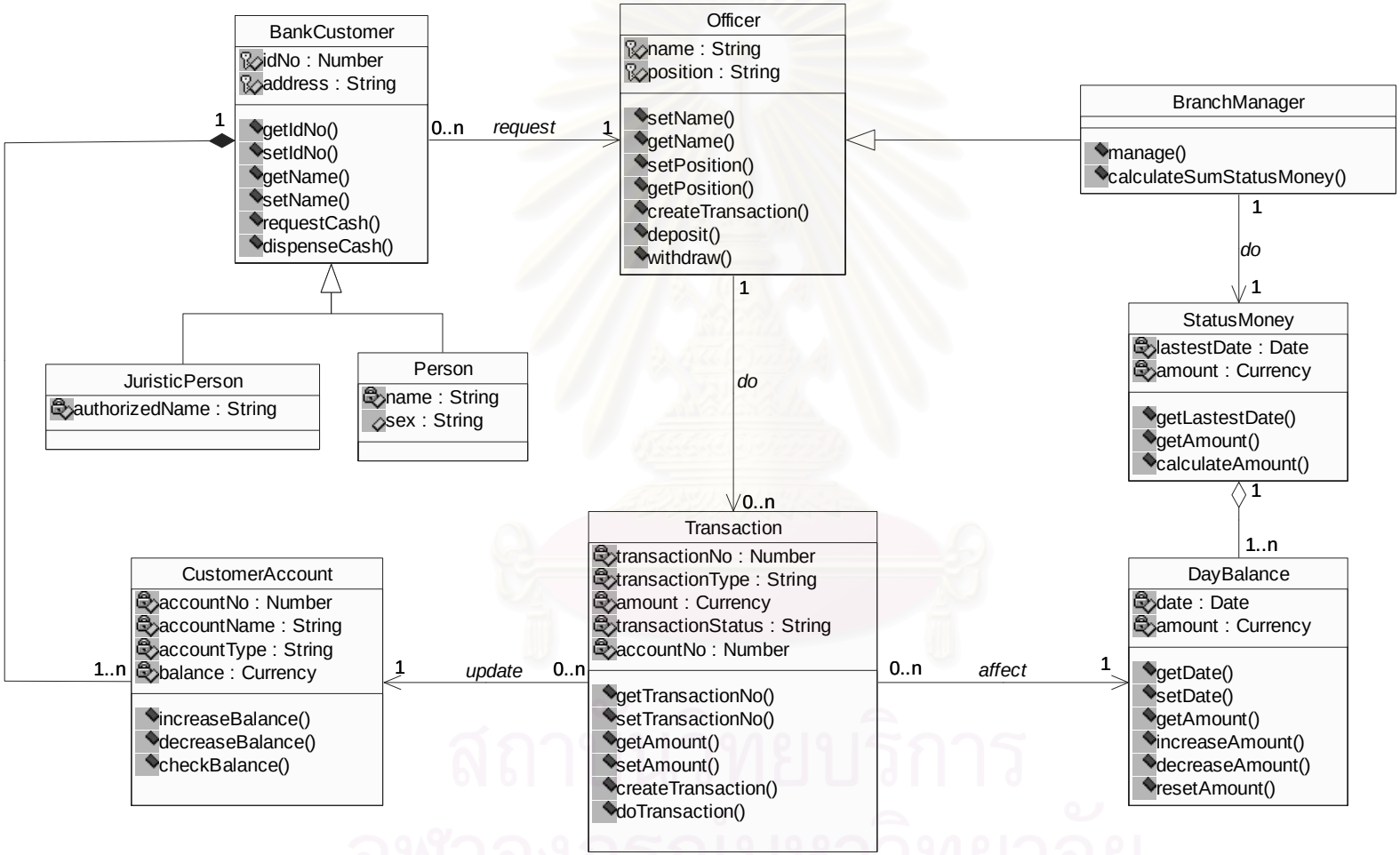
ภาคผนวก ง.
ตัวอย่างแบบทดสอบ

ระบบการฝากถอนคำนวณเงินดำรงฐานะของธนาคารพาณิชย์

ฟังก์ชันการทำงานของระบบ

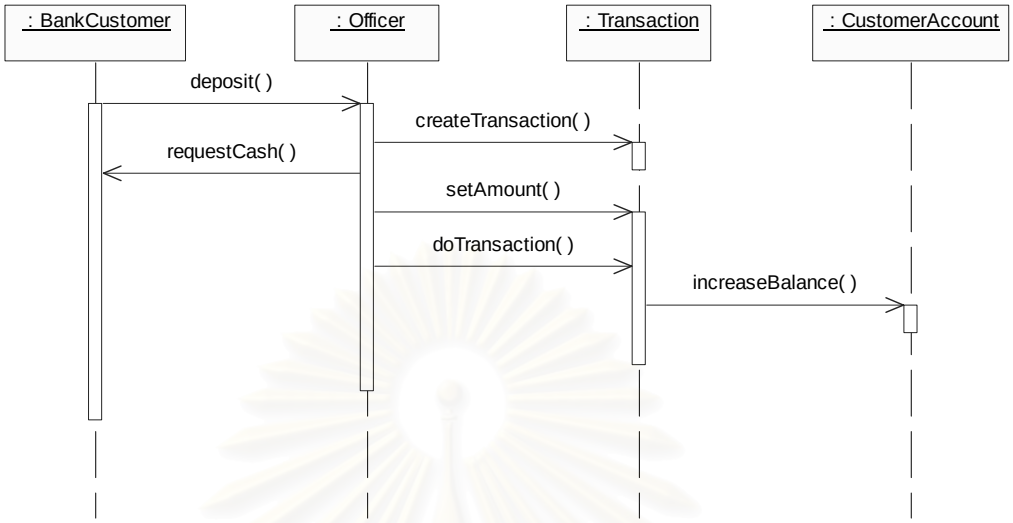
ในสาขาของธนาคารพาณิชย์แห่งหนึ่ง จะมีระบบงานมาตรฐานคือ การรับฝากเงินจากประชาชน หรือนิติบุคคล การให้ถอนเงินจากบัญชีเงินฝาก และการคำนวณยอดคงค้างรายวัน ในทางกฎหมายเรียกยอดคงค้างนี้ว่า เงินดำรงฐานะ ซึ่งงานทั้งหมดจะดำเนินการโดยเจ้าหน้าที่ของธนาคารเท่านั้น โดยสำหรับงานรับฝาก และถอนเงินจะดำเนินการโดยเจ้าหน้าที่ประจำเคาน์เตอร์ ในขณะที่การคำนวณเงินดำรงฐานะซึ่งเป็นรายได้/รายจ่ายที่เกิดจากการฝาก/ถอน จะดำเนินการโดยผู้จัดการสาขา ในเวลา 16:00 น. ของวันดังกล่าว

แผนภาพคลาส

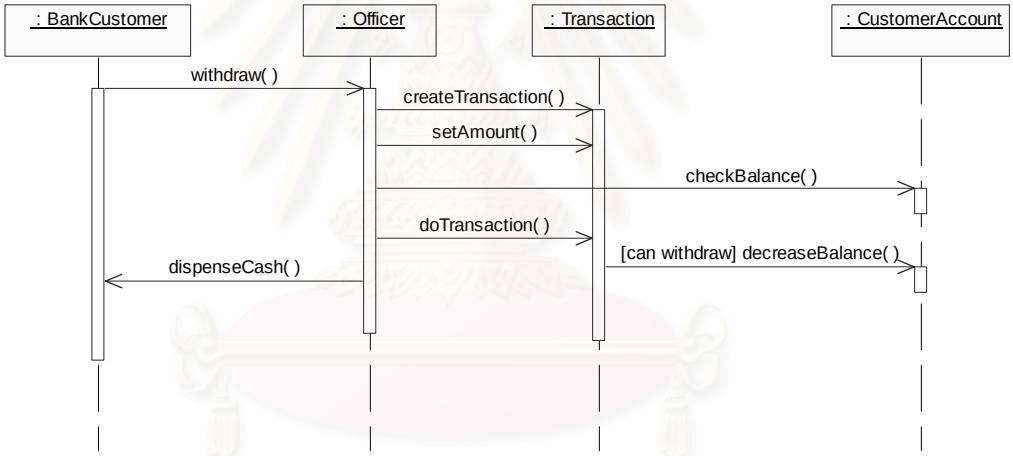

BankCustomer
idNo : Number
address : String
getIdNo()
setIdNo()
getName()
setName()
requestCash()
dispenseCash()
1
0..n
request
1
Officer
name : String
position : String
setName()
getName()
setPosition()
getPosition()
createTransaction()
deposit()
withdraw()
BranchManager
manage()
calculateSumStatusMoney()
1
do
1
JuristicPerson
authorizedName : String
Person
name : String
sex : String
1
do
0..n
StatusMoney
lastestDate : Date
amount : Currency
getLastestDate()
getAmount()
calculateAmount()
1
1..n
Transaction
transactionNo : Number
transactionType : String
amount : Currency
transactionStatus : String
accountNo : Number
getTransactionNo()
setTransactionNo()
getAmount()
setAmount()
createTransaction()
doTransaction()
CustomerAccount
accountNo : Number
accountName : String
accountType : String
balance : Currency
increaseBalance()
decreaseBalance()
checkBalance()
1..n
1
update
0..n
0..n
affect
1
DayBalance
date : Date
amount : Currency
getDate()
setDate()
getAmount()
increaseAmount()
decreaseAmount()
resetAmount()

แผนภาพซีเควนซ์

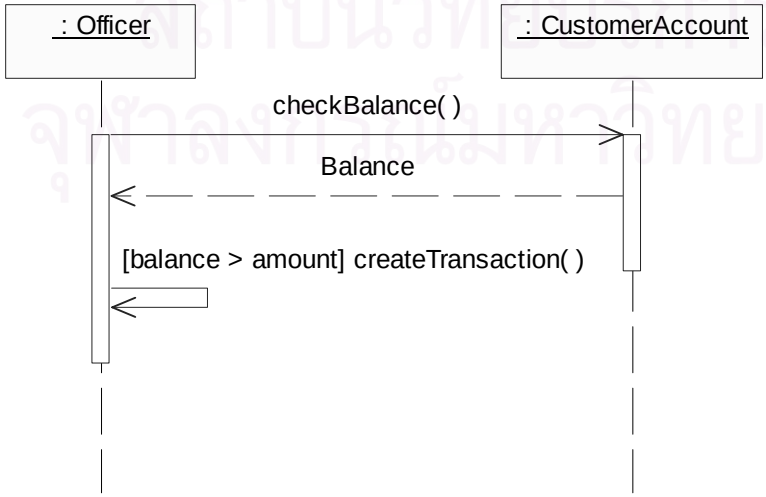
- แผนภาพซีเควนซ์ของการฝากเงิน



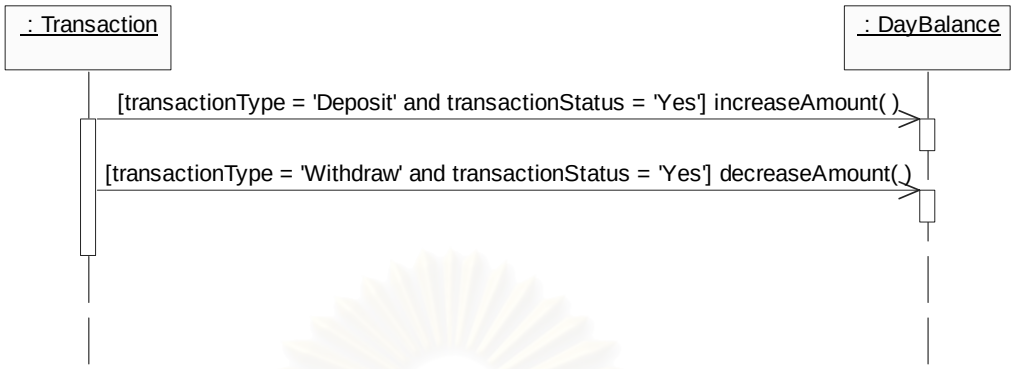
- แผนภาพซีเควนซ์ของการถอนเงิน



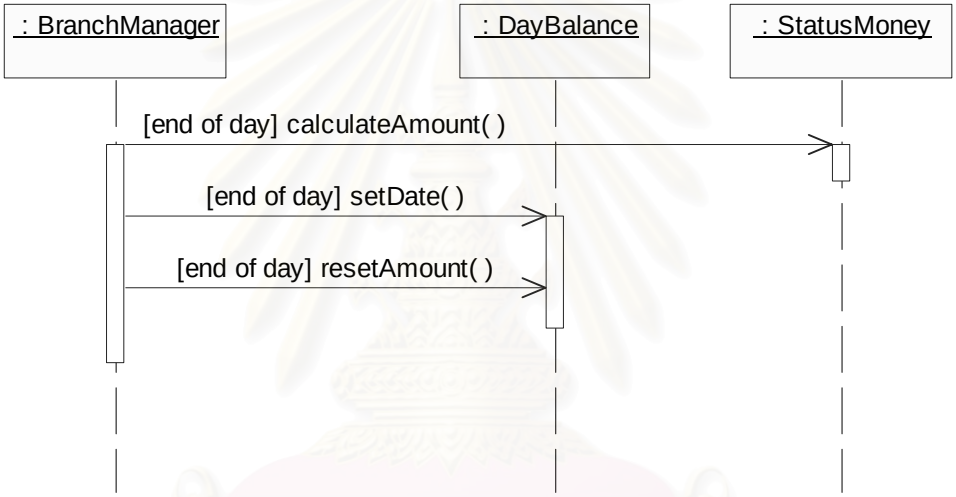
- แผนภาพซีเควนซ์ของการตรวจสอบยอดคงค้างของบัญชีเพื่อทำการถอนเงิน



- แผนภาพซีเควนซ์ของการรวมยอดการฝากถอนรายวัน



- แผนภาพซีเควนซ์ของการคำนวณเงินดำรงฐานะ



ชื่อ .....นามสกุล.....

เวลาเริ่มทำ .....เวลาสิ้นสุดการทำ .....

ชุดที่ 1 ข้อละ 1 คะแนน

ชื่อระบบการฝากถอนคำนวณเงินดำรงฐานะของธนาคารพาณิชย์

คำถามที่เกี่ยวข้องกับแผนภาพคลาส

ก. จงเติมคำลงในช่องว่าง

1. ผู้จัดการสาขามีแอทริบิวต์อะไรบ้าง

.....

2. หากต้องการทราบว่าการทำรายการในขณะนั้นเป็นการฝากเงินหรือการถอนเงิน จะดูได้จากแอทริบิวต์ใดของคลาสใด

.....

3. แอทริบิวต์ใดของคลาส Transaction ที่ไม่มีส่วนเกี่ยวข้องต่อความสัมพันธ์ ‘affect’ กับคลาส DayBalance

.....

4. แอทริบิวต์ใดของคลาส Transaction ที่มีผลต่อการเรียกใช้หรือไม่เรียกใช้เมทธอด increaseAmount() และ decreaseAmount() ในคลาส DayBalance

.....

5. ท่านคิดว่าแอทริบิวต์ amount ของคลาส StatusMoney เป็นค่าที่ได้จากการคำนวณของ attribute ใดของคลาสใด

.....

6. จากแผนภาพคลาส เมทธอดที่ใช้สำหรับการแก้ไขข้อมูลชื่อของผู้จัดการสาขาคือเมทธอดใด ของคลาสใด

.....

7. คลาส BranchManager ประกอบด้วยเมทธอดใดบ้าง

.....

8. จากแผนภาพคลาส ท่านคิดว่าเมทธอด getAmount ของคลาส DayBalance จะถูกเรียกใช้จากเมทธอดใดของคลาสใด

.....

9. จากแผนภาพคลาส ผู้จัดการสาขาสามารถเป็นผู้ทำรายการฝาก-ถอนได้หรือไม่

.....

10. จากแผนภาพคลาส เจ้าหน้าที่ธนาคารที่ไม่ได้เป็นผู้จัดการสาขา สามารถคำนวณเงินดำรงฐานะแทนผู้จัดการสาขาได้หรือไม่

.....

11. จากแผนภาพคลาส เจ้าหน้าที่ของธนาคารสามารถสอบถามยอดเงินคงเหลือในบัญชีเงินฝากได้หรือไม่

.....

ข. จงเลือกคำตอบที่ถูกต้องที่สุด

12. ข้อใดคือความแตกต่างระหว่างแอทริบิวต์ amount ในคลาส Transaction และคลาส DayBalance
    ก. แอทริบิวต์ amount ในคลาส Transaction เก็บจำนวนเงินของการฝากหรือถอน**หนึ่งรายการ** ของลูกค้า 1 คน ภายใน 1 วัน
    ส่วน แอทริบิวต์ amount ในคลาส DayBalance เกิดจากการคำนวณจำนวนเงินของการฝากและถอนทุกรายการ ของ**ลูกค้าทุกคน** ภายใน 1 วัน
    ข. แอทริบิวต์ amount ในคลาส Transaction เก็บจำนวนเงินของการฝากหรือถอน**หนึ่งรายการ** ของลูกค้า 1 คน ภายใน 1 วัน
    ส่วน แอทริบิวต์ amount ในคลาส DayBalance เกิดจากการคำนวณจำนวนเงินของการฝากและถอนทุกรายการ ของ**ลูกค้า 1 คน** ภายใน 1 วัน
    ค. แอทริบิวต์ amount ในคลาส Transaction เกิดจากการคำนวณจำนวนเงินของการฝากและถอน**ทุกรายการ** ของลูกค้า 1 คน ภายใน 1 วัน
    ส่วน แอทริบิวต์ amount ในคลาส DayBalance เกิดจากการคำนวณจำนวนเงินของการฝากและถอนทุกรายการ ของ**ลูกค้าทุกคน** ภายใน 1 วัน
    ง. แอทริบิวต์ amount ในคลาส Transaction เกิดจากการคำวณจำนวนเงินของการฝากและถอน**ทุกรายการ** ของลูกค้า 1 คน ภายใน 1 วัน
    ส่วน แอทริบิวต์ amount ในคลาส DayBalance เกิดจากการคำนวณจำนวนเงินของการฝากและถอนทุกรายการ ของ**ลูกค้า 1 คน** ภายใน 1 วัน
13. ท่านคิดว่าเมทธอด resetAmount() ของคลาส DayBalance ทำหน้าที่อะไร
    ก. เซ็ตค่ายอดเงินฝากของธนาคารใน 1 วัน
    ข. เซ็ตค่าเงินดำรงฐานะของธนาคารใน 1 วัน
    ค. เซ็ตค่าเงินในรายการฝากและถอน (transaction) ของลูกค้า
    ง. เซ็ตค่ายอดเงินคงค้างรายวันของธนาคาร
14. จากความสัมพันธ์ ‘update’ ระหว่างคลาส Transaction และ คลาส CustomerAccount ท่านคิดว่าคลาส Transaction จะเรียกใช้เมทธอดใดของคลาส CustomerAccount
    ก. เมทธอด checkBalance()
    ข. ใช้ทุกเมทธอดในคลาส CustomerAccount
    ค. เมทธอด increaseBalance() และ decreaseBalance()
    ง. ไม่ความสัมพันธ์กับเมทธอดใด แต่มีความสัมพันธ์กับบางแอทริบิวต์ในคลาส CustomerAccount
15. จากแผนภาพคลาส ประโยคใดต่อไปนี้ถูกต้อง
    ก. ลูกค้าต้องมีรายการฝาก-ถอนเงินกับทางธนาคารอย่างน้อย 1 รายการ
    ข. ถ้าวันใดไม่มีการฝาก-ถอนเงินเลย จะไม่มีการคำนวณเงินดำรงฐานะ
    ค. รายการฝาก-ถอน 1 รายการจะเกี่ยวข้องกับบัญชีเงินฝากเพียง 1 บัญชีเท่านั้น
    ง. ยอดเงินคงค้างรายวันเป็นประเภทหนึ่งของเงินดำรงฐานะ

คำถามที่เกี่ยวข้องกับแผนภาพซีเควนซ์

ก. จงเติมคำตอบลงในช่องว่าง

16. จากแผนภาพซีเควนซ์ของการฝากเงิน ท่านคิดว่าหัวลูกศรของ message requestCash() ชี้ไปในทิศทางที่ถูกต้องหรือไม่

……...

17. ในการถอนเงิน การทำรายการจะทำได้สำเร็จ เมื่อเงื่อนไขใดเป็นจริง

……...

18. จากแผนภาพซีเควนซ์ของการตรวจสอบยอดคงค้างของบัญชีเพื่อทำการถอนเงิน มีการคืนค่าแอทริบิวต์ใดให้แก่คลาสใดบ้าง

……...

19. จากแผนภาพซีเควนซ์ของการรวมยอดฝากถอนรายวัน จงอธิบายความหมายของ transactionStatus = ‘Yes’

……...

20. กิจกรรมที่เกิดขึ้นหลังเวลา 16:00 น. มีคลาสใดที่เกี่ยวข้องบ้าง

……...

ชื่อ .....นามสกุล.....

เวลาเริ่มทำ .....เวลาสิ้นสุดการทำ .....

ชุดที่ 2

ชื่อระบบการฝากถอนคำนวณเงินดำรงฐานะของธนาคารพาณิชย์

คำถามที่เกี่ยวข้องกับแผนภาพคลาส

1. ถ้าบัญชีเงินฝากแยกออกเป็นบัญชีออมทรัพย์, บัญชีกระแสรายวัน และ บัญชีฝากเงินประจำ ควรแก้ไขแผนภาพคลาสอย่างไร (1 คะแนน)

2. สมมติว่า กรณีลูกค้าธนาคารต้องการถอนเงินออกจากบัญชี จะต้องใส่รหัสผ่านก่อน จึงจะสามารถถอนเงินได้ จะต้องทำการแก้ไขแผนภาพคลาสตำแหน่งใด (1 คะแนน)

3. ถ้าสามารถทำรายการถอนเงินได้เพียง 1 รายการต่อ 1 วัน ต้องทำการแก้ไขแผนภาพคลาสอย่างไร จงอธิบาย (1 คะแนน)

4. ถ้าลูกค้าธนาคารต้องการสอบถามยอดเงินคงเหลือในบัญชี จะต้องแก้ไขที่แผนภาพคลาสอย่างไร (1 คะแนน)

5. ถ้าการคำนวณเงินดำรงฐานะสามารถทำโดยพนักงานคนใดก็ได้ จะทำการแก้ไขแผนภาพคลาสนี้อย่างไร จงอธิบาย (2 คะแนน)

6. ถ้าต้องการให้ลูกค้า สามารถทำการโอนเงินไปให้ลูกค้าอีกคนหนึ่งได้ จะต้องทำการแก้ไขแผนภาพคลาสอย่างไรบ้าง จงอธิบาย (3 คะแนน)

คำถามที่เกี่ยวข้องกับแผนภาพซีเควนซ์

7. จากข้อที่ 6 เราจะเขียนแผนภาพซีเควนซ์แสดงการโอนเงินได้อย่างไร (3 คะแนน)

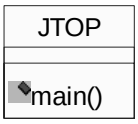
8. ถ้าต้องการคำนวณเงินดำรงฐานะในทุกๆ สัปดาห์ในวันอาทิตย์ ต้องทำการแก้ไขแผนภาพซีเควนซ์ การคำนวณเงินดำรงฐานะ อย่างไร (2 คะแนน)

9. จากแผนภาพซีเควนซ์ของการถอนเงิน ถ้ากำหนดให้การถอนเงิน 1 ครั้ง สามารถถอนเงินได้ไม่เกินจำนวน 20,000 บาท ควรแก้ไขแผนภาพซีเควนซ์นี้อย่างไร จงอธิบาย (2 คะแนน)

10. จากแผนภาพซีเควนซ์การตรวจสอบยอดคงค้างของบัญชีเพื่อทำการถอนเงิน ให้เขียนแผนภาพต่อเพิ่มเติมตามเงื่อนไขต่อไปนี้ คือ "ในกรณีที่จำนวนเงินคงเหลือในบัญชีน้อยกว่าจำนวนเงินที่ถอน ให้ส่งข้อความว่า Can't withdraw ไปยังผู้ถอนเงิน" จงวาดแผนภาพซีเควนซ์ใหม่ (1 คะแนน)

ภาคผนวก จ.
รายละเอียดภายในคลาส

1. คลาส JTOP



รูปที่ จ-1 แสดงคลาส JTOP

ชื่อคลาส	JTOP
คุณลักษณะ	ไม่มี
เมทธอด	- main() ทำหน้าที่เริ่มต้นรันโปรแกรม

1.1 รหัสต้นฉบับของคลาส JTOP

```
/* คลาสหลักสำหรับเริ่มต้น run โปรแกรม มีเมทธอด main เป็นเมทธอดหลักให้เรียกใช้งาน */

public class JTOP {
public static void main(String args[]) throws Exception {
        PredictionTOOL pre = new PredictionTOOL();
                pre.show();
} }
```

2. คลาส PredictionTOOL

ชื่อคลาส	PredictionTOOL
คุณลักษณะ	- numNC สำหรับเก็บค่ามาตรวัด NC - numNOS สำหรับเก็บค่ามาตรวัด NOS - numNGenH สำหรับเก็บค่ามาตรวัด NGenH - numNaggH สำหรับเก็บค่ามาตรวัด NaggH - numMaxHAgg สำหรับเก็บค่ามาตรวัด MaxHAgg - numMaxDIT สำหรับเก็บค่ามาตรวัด MaxDIT - numANAUW สำหรับเก็บค่ามาตรวัด ANAUW - numANAW สำหรับเก็บค่ามาตรวัด ANAW - numANMUW สำหรับเก็บค่ามาตรวัด ANMUW

	- numANMW สำหรับเก็บค่ามาตรวัด ANMW - numWMBO สำหรับเก็บค่ามาตรวัด WMBO - numANRM สำหรับเก็บค่ามาตรวัด ANRM - numANDM สำหรับเก็บค่ามาตรวัด ANDM - numANET สำหรับเก็บค่ามาตรวัด ANET - numANCM สำหรับเก็บค่ามาตรวัด ANCM - numANAss สำหรับเก็บค่ามาตรวัด ANAsso - numANAgg สำหรับเก็บค่ามาตรวัด ANAgg - numANGen สำหรับเก็บค่ามาตรวัด ANGen - selectedFileName ใช้สำหรับเก็บชื่อของไฟล์เอกสารเอ็กซ์เอ็มแอล
เมทธอด	- PredictionTOOL() ทำหน้าสร้างวัตถุของคลาส PredictionTOOL - initComponents() ทำหน้าที่เริ่มต้นสร้างหน้าจอ - clearTable() ทำหน้าที่ลบข้อมูลที่แสดงภายในตารางทั้งหมด - clickOpen() ทำหน้าที่แสดงหน้าจอสำหรับเลือกไฟล์ข้อมูลเอกสารเอ็กซ์เอ็มแอล - jButton1ActionPerformed() ทำหน้าที่เรียกเมทธอด clickOpen() - aboutMenuItemActionPerformed() ทำหน้าที่แสดงชื่อผู้สร้างโปรแกรม - predictMenuItemActionPerformed() ทำหน้าที่ทำนายค่าความสามารถในการบำรุงรักษาซอฟต์แวร์ - calMetricsMenuItemActionPerformed() ทำหน้าที่คำนวณค่ามาตรวัด - openMenuItemActionPerformed() ทำหน้าที่แสดงหน้าจอสำหรับเลือกไฟล์ข้อมูล - calculateMenuActionPerformed() ทำหน้าที่คำนวณค่ามาตรวัด - exitMenuItemActionPerformed() ทำหน้าที่ออกจากโปรแกรม - exitForm() ทำหน้าที่ออกจากโปรแกรม - getExtension() ทำหน้าที่ดึงข้อมูลนามสกุลของไฟล์มาจากคลาส XMLFileFilter

รูปที่ จ-2 แสดงคลาส PredictionTOOL

2.1 รหัสต้นฉบับของคลาส PredictionTOOL

```
public class PredictionTOOL extends javax.swing.JFrame {
    /** Creates new form PredictionTOOL */
    private int numNC, numNOS, numNGenH, numNaggH, numMaxHAgg, numMaxDIT;
    private float numANAUW, numANAW, numANMUW, numANMW, numWMBO, numANRM;
    private float numANGen, numANAss, numANAgg, numANDM, numANET, numANCM=0;
    String selectedFileName = "";
    java.io.File selectedFile = null;
    public PredictionTOOL() {
        initComponents();
```

```
}
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
private void initComponents() {
    jTabbedPane1 = new javax.swing.JTabbedPane();
    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jTextField1 = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTable1 = new javax.swing.JTable();
    jButton1 = new javax.swing.JButton();
    jPanel2 = new javax.swing.JPanel();
    jScrollPane5 = new javax.swing.JScrollPane();
    jTable5 = new javax.swing.JTable();
    jPanel3 = new javax.swing.JPanel();
    jScrollPane3 = new javax.swing.JScrollPane();
    jTable3 = new javax.swing.JTable();
    jLabel4 = new javax.swing.JLabel();
    jPanel4 = new javax.swing.JPanel();
    jScrollPane4 = new javax.swing.JScrollPane();
    jTable4 = new javax.swing.JTable();
    jLabel3 = new javax.swing.JLabel();
    jPanel5 = new javax.swing.JPanel();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTable2 = new javax.swing.JTable();
    jLabel2 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    menuBar = new javax.swing.JMenuBar();
    fileMenu = new javax.swing.JMenu();
    openMenuItem = new javax.swing.JMenuItem();
    jSeparator1 = new javax.swing.JSeparator();
    exitMenuItem = new javax.swing.JMenuItem();
```

```
        calculateMenu = new javax.swing.JMenu();
        calMetricsMenuItem = new javax.swing.JMenuItem();
        jSeparator3 = new javax.swing.JSeparator();
        predictMenuItem = new javax.swing.JMenuItem();
        helpMenu = new javax.swing.JMenu();
        aboutMenuItem = new javax.swing.JMenuItem();
        setTitle("Java TOOL for Prediction ");
        setIconImage(getIconImage());
        setLocationRelativeTo(jPanel1);
        setMaximizedBounds(new java.awt.Rectangle(2147483647, 2147483647, 2147483647,
2147483647));
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        jPanel1.setLayout(null);
        jPanel1.setForeground((java.awt.Color) javax.swing.UIManager.getDefaults().get("primary3"));
        jPanel1.setMinimumSize(new java.awt.Dimension(200, 200));
        jLabel1.setText("File Name :");
        jPanel1.add(jLabel1);
        jLabel1.setBounds(90, 20, 70, 16);
        jTextField1.setEditable(false);
        jPanel1.add(jTextField1);
        jTextField1.setBounds(160, 20, 300, 20);
        jTable1.setFont(new java.awt.Font("Dialog", 1, 12));
        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {"    NC", null},
                {"    ANAUW ", null},
                {"    ANAW", null},
                {"    ANMUW", null},
                {"    ANMW", null},
                {"    ANAss", null},
                {"    ANAgg", null},
```

```
            {"    NaggH", null},
            {"    MaxHAgg", null},
            {"    ANGen", null},
            {"    NGenH", null},
            {"    MaxDIT", null},
            {"    NOS", null},
            {"    WMBO", null},
            {"    ANRM", null},
            {"    ANDM", null},
            {"    ANET", null},
            {"    ANCM", null}
        },
        new String [] {
            "Metrics", "Value"
        }
    ) {
        Class[] types = new Class [] {
            java.lang.String.class, java.lang.String.class
        };
        boolean[] canEdit = new boolean [] {
            false, false
        };
        public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
        }
        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });
    jTable1.setRowHeight(20);
    jScrollPane1.setViewportView(jTable1);
    jPanel1.add(jScrollPane1);
    jScrollPane1.setBounds(40, 60, 530, 380);
    jButton1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/JToolInterface/pics/open.jpg")));
```

```
        jButton1.setToolTipText("open xml file");
        jButton1.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED, null, null,
java.awt.Color.black, java.awt.Color.lightGray));
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        jPanel1.add(jButton1);
        jButton1.setBounds(470, 15, 30, 30);
        jTabbedPane1.addTab("Calculate Metrics", jPanel1);
        jPanel2.setLayout(null);
        jTable5.setFont(new java.awt.Font("Dialog", 1, 12));
        jTable5.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {"   NC  "},
                {"   ANAUW"},
                {"   ANMUW"},
                {"   ANAss  "},
                {"   NaggH"},
                {"   MaxHAgg"},
                {"   NGenH"},
                {"   MaxDIT"},
                {"   NOS"},
                {"   WMBO"},
                {"   ANRM"},
                {"   ANDM"},
                {"   ANET"},
                {"   ANCM"},
                {"   (Constant)"},
                {"   SUM"}
            },
            new String [] {
                "Variable"
```

```
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class
    };
    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});
jTable5.setRowHeight(20);
jScrollPane5.setViewportView(jTable5);
jPanel2.add(jScrollPane5);
jScrollPane5.setBounds(20, 60, 80, 340);
jPanel3.setLayout(null);
jPanel3.setBackground(new java.awt.Color(255, 204, 204));
jPanel3.setBorder(new javax.swing.border.EtchedBorder());
jPanel3.setFont(new java.awt.Font("Dialog", 1, 12));
jTable3.setFont(new java.awt.Font("Dialog", 1, 12));
jTable3.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {new Float(0.98), null},
        {new Float(7.856), null},
        {new Float(10.146), null},
        {new Float(27.655), null},
        {new Float(0.287), null},
        {new Float(11.547), null},
        {new Float(13.096), null},
        {new Float(-4.062), null},
        {new Float(-0.737), null},
        {new Float(-2.393), null},
        {new Float(0.192), null},
        {new Float(0.215), null},
        {new Float(-0.672), null},
        {new Float(-2.51), null},
        {new Float(-60.615), null},
```

```
        {null, null}
    },
    new String [] {
        "Coefficient", "Metrics value"
    }
) {
    Class[] types = new Class [] {
        java.lang.Float.class, java.lang.Float.class
    };
    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});
jTable3.setRowHeight(20);
jScrollPane3.setViewportView(jTable3);
jPanel3.add(jScrollPane3);
jScrollPane3.setBounds(0, 40, 160, 340);
jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel4.setText("Fuction for difficult");
jPanel3.add(jLabel4);
jLabel4.setBounds(20, 10, 120, 16);
jPanel2.add(jPanel3);
jPanel3.setBounds(100, 20, 160, 380);
jPanel4.setLayout(null);
jPanel4.setBackground(new java.awt.Color(153, 204, 255));
jPanel4.setBorder(new javax.swing.border.EtchedBorder());
jTable4.setFont(new java.awt.Font("Dialog", 1, 12));
jTable4.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {new Float(0.825), null},
        {new Float(6.56), null},
        {new Float(4.614), null},
        {new Float(20.022), null},
        {new Float(0.254), null},
        {new Float(4.801), null},
```

```
            {new Float(3.332), null},
            {new Float(0.282), null},
            {new Float(3.656), null},
            {new Float(0.897), null},
            {new Float(3.858), null},
            {new Float(0.868), null},
            {new Float(2.414), null},
            {new Float(-1.282), null},
            {new Float(-37.568), null},
            {null, null}
        },
        new String [] {
            "Coefficient", "Metrics value"
        }
    ) {
        Class[] types = new Class [] {
            java.lang.Float.class, java.lang.Float.class
        };
        public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
        }
    });
    jTable4.setRowHeight(20);
    jScrollPane4.setViewportView(jTable4);
    jPanel4.add(jScrollPane4);
    jScrollPane4.setBounds(0, 40, 160, 340);
    jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel3.setText("Fuction for medium");
    jPanel4.add(jLabel3);
    jLabel3.setBounds(20, 10, 120, 15);
    jPanel2.add(jPanel4);
    jPanel4.setBounds(260, 20, 160, 380);
    jPanel5.setLayout(null);
    jPanel5.setBackground(new java.awt.Color(255, 255, 204));
    jPanel5.setBorder(new javax.swing.border.EtchedBorder());
```

```
jTable2.setFont(new java.awt.Font("Dialog", 1, 12));
jTable2.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {new Float(0.913), null},
        {new Float(8.781), null},
        {new Float(7.588), null},
        {new Float(19.309), null},
        {new Float(0.729), null},
        {new Float(5.252), null},
        {new Float(4.003), null},
        {new Float(-2.543), null},
        {new Float(3.399), null},
        {new Float(-0.034), null},
        {new Float(2.941), null},
        {new Float(-0.368), null},
        {new Float(3.096), null},
        {new Float(-1.353), null},
        {new Float(-46.637), null},
        {null, null}
    },
    new String [] {
        "Coefficient", "Metrics value"
    }
) {
    Class[] types = new Class [] {
        java.lang.Float.class, java.lang.Float.class
    };
    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});
jTable2.setRowHeight(20);
jScrollPane2.setViewportView(jTable2);
jPanel5.add(jScrollPane2);
jScrollPane2.setBounds(0, 40, 160, 340);
```

```
        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel2.setText("Fuction for easy");
        jPanel5.add(jLabel2);
        jLabel2.setBounds(20, 10, 120, 16);
        jPanel2.add(jPanel5);
        jPanel5.setBounds(420, 20, 160, 380);
        jLabel5.setFont(new java.awt.Font("Dialog", 1, 14));
        jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jPanel2.add(jLabel5);
        jLabel5.setBounds(120, 400, 370, 30);
        jTabbedPane1.addTab("Predict Matintainability", jPanel2);
        getContentPane().add(jTabbedPane1, java.awt.BorderLayout.CENTER);
        menuBar.setBorder(null);
        fileMenu.setText("File");
        openMenuItem.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/JToolInterface/pics/open2.jpg")));
        openMenuItem.setText("Open");
        openMenuItem.setToolTipText("open xml file");
        openMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                openMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(openMenuItem);
        fileMenu.add(jSeparator1);
        exitMenuItem.setIcon(new javax.swing.ImageIcon(""));
        exitMenuItem.setText("Exit");
        exitMenuItem.setToolTipText("exit program");
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(exitMenuItem);
        menuBar.add(fileMenu);
```

```
        calculateMenu.setText("Calculate");
        calculateMenu.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                calculateMenuActionPerformed(evt);
            }
        });
        calMetricsMenuItem.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/JToolInterface/pics/cal.gif")));
        calMetricsMenuItem.setText("Calculate Metrics");
        calMetricsMenuItem.setToolTipText("");
        calMetricsMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                calMetricsMenuItemActionPerformed(evt);
            }
        });
        calculateMenu.add(calMetricsMenuItem);
        calculateMenu.add(jSeparator3);
        predictMenuItem.setText("Prediction Maintainability");
        predictMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                predictMenuItemActionPerformed(evt);
            }
        });
        calculateMenu.add(predictMenuItem);
        menuBar.add(calculateMenu);
        helpMenu.setText("Help");
        aboutMenuItem.setText("About");
        aboutMenuItem.setToolTipText("credit");
        aboutMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                aboutMenuItemActionPerformed(evt);
            }
        });
        helpMenu.add(aboutMenuItem);
        menuBar.add(helpMenu);
```

```
    setJMenuBar(menuBar);
    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-609)/2, (screenSize.height-550)/2, 609, 550);
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // Add your handling code here:
    clickOpen();
}
private void aboutMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // Add your handling code here:
    CreditFrame credit = new CreditFrame();
    java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
    credit.setBounds((screenSize.width-450)/2, (screenSize.height-400)/2, 450, 400);
    credit.show();
}
private void predictMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // Add your handling code here:
    jTabbedPane1.setSelectedIndex(1);
    // show metrics value into table
     jTable3.setValueAt(new Float(numNC),0,1);
     jTable3.setValueAt(new Float(numANAUW),1,1);
     jTable3.setValueAt(new Float(numANMUW),2,1);
     jTable3.setValueAt(new Float(numANAss),3,1);
     jTable3.setValueAt(new Float(numNaggH),4,1);
     jTable3.setValueAt(new Float(numMaxHAgg),5,1);
     jTable3.setValueAt(new Float(numNGenH),6,1);
     jTable3.setValueAt(new Float(numMaxDIT),7,1);
     jTable3.setValueAt(new Float(numNOS),8,1);
     jTable3.setValueAt(new Float(numWMBO),9,1);
     jTable3.setValueAt(new Float(numANRM),10,1);
     jTable3.setValueAt(new Float(numANDM),11,1);
     jTable3.setValueAt(new Float(numANET),12,1);
     jTable3.setValueAt(new Float(numANCM),13,1);
     jTable4.setValueAt(new Float(numNC),0,1);
     jTable4.setValueAt(new Float(numANAUW),1,1);
```

```
        jTable4.setValueAt(new Float(numANMUW),2,1);
        jTable4.setValueAt(new Float(numANAss),3,1);
        jTable4.setValueAt(new Float(numNaggH),4,1);
        jTable4.setValueAt(new Float(numMaxHAgg),5,1);
        jTable4.setValueAt(new Float(numNGenH),6,1);
        jTable4.setValueAt(new Float(numMaxDIT),7,1);
        jTable4.setValueAt(new Float(numNOS),8,1);
        jTable4.setValueAt(new Float(numWMBO),9,1);
        jTable4.setValueAt(new Float(numANRM),10,1);
        jTable4.setValueAt(new Float(numANDM),11,1);
        jTable4.setValueAt(new Float(numANET),12,1);
        jTable4.setValueAt(new Float(numANCM),13,1);
        jTable2.setValueAt(new Float(numNC),0,1);
        jTable2.setValueAt(new Float(numANAUW),1,1);
        jTable2.setValueAt(new Float(numANMUW),2,1);
        jTable2.setValueAt(new Float(numANAss),3,1);
        jTable2.setValueAt(new Float(numNaggH),4,1);
        jTable2.setValueAt(new Float(numMaxHAgg),5,1);
        jTable2.setValueAt(new Float(numNGenH),6,1);
        jTable2.setValueAt(new Float(numMaxDIT),7,1);
        jTable2.setValueAt(new Float(numNOS),8,1);
        jTable2.setValueAt(new Float(numWMBO),9,1);
        jTable2.setValueAt(new Float(numANRM),10,1);
        jTable2.setValueAt(new Float(numANDM),11,1);
        jTable2.setValueAt(new Float(numANET),12,1);
        jTable2.setValueAt(new Float(numANCM),13,1);
      // show metrics value into table
      double value1 = 0, value2 = 0, value3 = 0;
      value1 =
0.98*numNC+7.856*numANAUW+10.146*numANMUW+27.655*numANAss+0.287*numNaggH+11.5
47*numMaxHAgg+13.096*numNGenH-4.062*numMaxDIT-0.737*numNOS-
2.393*numWMBO+0.192*numANRM+0.215*numANDM-0.672*numANET-2.51*numANCM-60.615;
      value2 =
0.825*numNC+6.56*numANAUW+4.614*numANMUW+20.022*numANAss+0.254*numNaggH+4.801
```

```
*numMaxHAgg+3.332*numNGenH+0.282*numMaxDIT+3.656*numNOS+0.897*numWMBO+3.858*n
umANRM+0.868*numANDM+2.414*numANET-1.282*numANCM-37.568;
        value3 =
0.913*numNC+8.781*numANAUW+7.588*numANMUW+19.309*numANAss+0.729*numNaggH+5.25
2*numMaxHAgg+4.003*numNGenH-2.543*numMaxDIT+3.399*numNOS-
0.034*numWMBO+2.941*numANRM-0.368*numANDM+3.096*numANET-1.353*numANCM-46.637;
        jTable3.setValueAt(new Float(value1),15,1);
        jTable4.setValueAt(new Float(value2),15,1);
        jTable2.setValueAt(new Float(value3),15,1);
        if (value1 < value2) {
            if (value2 < value3) {
                // value 3 is max
                jLabel5.setText("This system is easy to maintain");
            }else {
                // value 2 is max
                jLabel5.setText("This system is medium to maintain");
            }
        }else {
            if (value1 < value3) {
                // value 3 is max
                jLabel5.setText("This system is easy to maintain");
            }else {
                // value 1 is max
                jLabel5.setText("This system is difficult to maintain");
            }
        }
    }
    private void calMetricsMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            if (!selectedFileName.equals("") || (!(selectedFile == null))) {
                JFrame f = new JFrame("Please input ");
                java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
                f.setBounds((screenSize.width-400)/2, (screenSize.height-200)/2, 400, 200);
                GetMetricANCM dialog = new GetMetricANCM(f, true);
                dialog.setBounds((screenSize.width-400)/2, (screenSize.height-200)/2, 400, 200);
```

```
dialog.show();
//int returnvalue = 0;
     //if (returnVal == dialog.APPROVE_OPTION) {
    String num = dialog.getNumANCM();
    SAXBuilder builder = new SAXBuilder();
    Document doc = builder.build(selectedFile);
    Element root = doc.getRootElement();
    Metrics metric= new Metrics(root,num);
    // set value of each metrics
    numNC = metric.getNC();
    numNOS = metric.getNOS();
    numNGenH = metric.getnumNGenH();
    numNaggH = metric.getnumNaggH();
    numMaxHAgg = metric.getnumMaxHAgg();
    numMaxDIT = metric.getnumMaxDIT();
    numANAUW = metric.getANAUW();
    numANAW = metric.getANAW();
    numANMUW = metric.getANMUW();
    numANMW = metric.getANMW();
    numWMBO = metric.getWMBO();
    numANRM = metric.getANRM();
    numANGen = metric.getANGen();
    numANAss = metric.getANAss();
    numANAgg = metric.getANAgg();
    numANDM = metric.getnumANDM();
    numANET = metric.getnumANET();
    numANCM = metric.getnumANCM();
    // Insert value into table
    jTable1.setValueAt("    "+numNC,0,1);
    jTable1.setValueAt("    "+numANAUW,1,1);
    jTable1.setValueAt("    "+numANAW,2,1);
    jTable1.setValueAt("    "+numANMUW,3,1);
    jTable1.setValueAt("    "+numANMW,4,1);
    jTable1.setValueAt("    "+numANAss,5,1);
    jTable1.setValueAt("    "+numANAgg,6,1);
```

```
                jTable1.setValueAt("    "+numNaggH,7,1);
                jTable1.setValueAt("    "+numMaxHAgg,8,1);
                jTable1.setValueAt("    "+numANGen,9,1);
                jTable1.setValueAt("    "+numNGenH,10,1);
                jTable1.setValueAt("    "+numMaxDIT,11,1);
                jTable1.setValueAt("    "+numNOS,12,1);
                jTable1.setValueAt("    "+numWMBO,13,1);
                jTable1.setValueAt("    "+numANRM,14,1);
                jTable1.setValueAt("    "+numANDM,15,1);
                jTable1.setValueAt("    "+numANET,16,1);
                jTable1.setValueAt("    "+numANCM,17,1);
                // Insert value into table
            //}
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    // Add your handling code here:
}
private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // Add your handling code here:
    clickOpen();
}
private void calculateMenuActionPerformed(java.awt.event.ActionEvent evt) {
    // Add your handling code here:
}
private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {
    System.exit(0);
}
private void clearTable() {
            jTable1.setValueAt("",0,1);
```

```
            jTable1.setValueAt("",1,1);
            jTable1.setValueAt("",2,1);
            jTable1.setValueAt("",3,1);
            jTable1.setValueAt("",4,1);
            jTable1.setValueAt("",5,1);
            jTable1.setValueAt("",6,1);
            jTable1.setValueAt("",7,1);
            jTable1.setValueAt("",8,1);
            jTable1.setValueAt("",9,1);
            jTable1.setValueAt("",10,1);
            jTable1.setValueAt("",11,1);
            jTable1.setValueAt("",12,1);
            jTable1.setValueAt("",13,1);
            jTable1.setValueAt("",14,1);
            jTable1.setValueAt("",15,1);
            jTable1.setValueAt("",16,1);
            jTable1.setValueAt("",17,1);
}
public String getExtension(java.io.File f) {
    if(f != null) {
        String filename = f.getName();
            int i = filename.lastIndexOf('.');
            if(i>0 && i<filename.length()-1) {
                    return filename.substring(i+1).toLowerCase();
            };
        }
        return null;
}
private void clickOpen() {
    jTabbedPane1.setSelectedIndex(0);
    javax.swing.JFileChooser chooser = new javax.swing.JFileChooser("c:");
    chooser.setFileFilter(new XMLFileFilter());
    int returnVal = chooser.showOpenDialog(this);
        selectedFile = null;
    if (returnVal == chooser.APPROVE_OPTION) {
```

```
        selectedFile = chooser.getSelectedFile();
        selectedFileName = chooser.getSelectedFile().getAbsolutePath();
        String extension = getExtension(selectedFile);
        if (extension == null) {
          selectedFile = new java.io.File(selectedFile.getAbsolutePath()+".xml");
        }
        selectedFileName = selectedFile.toString();
        jTextField1.setText(selectedFileName);
    }
    clearTable();
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    new PredictionTOOL().show();
}
// Variables declaration - do not modify
private javax.swing.JMenuItem aboutMenuItem;
private javax.swing.JMenuItem calMetricsMenuItem;
private javax.swing.JMenu calculateMenu;
private javax.swing.JMenuItem exitMenuItem;
private javax.swing.JMenu fileMenu;
private javax.swing.JMenu helpMenu;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
```

```
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator3;
    private javax.swing.JTabbedPane jTabbedPane1;
    private javax.swing.JTable jTable1;
    private javax.swing.JTable jTable2;
    private javax.swing.JTable jTable3;
    private javax.swing.JTable jTable4;
    private javax.swing.JTable jTable5;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenuItem openMenuItem;
    private javax.swing.JMenuItem predictMenuItem;
    // End of variables declaration
   }
```

3. **คลาส** Metrics

ชื่อคลาส	Metrics
คุณลักษณะ	- numNC สำหรับเก็บค่ามาตรวัด NC - numNOS สำหรับเก็บค่ามาตรวัด NOS - numNGenH สำหรับเก็บค่ามาตรวัด NGenH - numNaggH สำหรับเก็บค่ามาตรวัด NaggH - numMaxHAgg สำหรับเก็บค่ามาตรวัด MaxHAgg - numMaxDIT สำหรับเก็บค่ามาตรวัด MaxDIT - numANAUW สำหรับเก็บค่ามาตรวัด ANAUW - numANAW สำหรับเก็บค่ามาตรวัด ANAW - numANMUW สำหรับเก็บค่ามาตรวัด ANMUW - numANMW สำหรับเก็บค่ามาตรวัด ANMW - numWMBO สำหรับเก็บค่ามาตรวัด WMBO

	- numANRM สำหรับเก็บค่ามาตรวัด ANRM - numANDM สำหรับเก็บค่ามาตรวัด ANDM - numANET สำหรับเก็บค่ามาตรวัด ANET - numANCM สำหรับเก็บค่ามาตรวัด ANCM - numANAss สำหรับเก็บค่ามาตรวัด ANAsso - numANAgg สำหรับเก็บค่ามาตรวัด ANAgg - numANGen สำหรับเก็บค่ามาตรวัด ANGen
เมทธอด	- ProcessElement() ทำหน้าที่กำหนดเงื่อนไขในการคำนวณค่ามาตรวัด - calculateMetrics() ทำหน้าที่คำนวณค่ามาตรวัด - getNC() ทำหน้าที่ให้ค่ามาตรวัด NC - getANAUW() ทำหน้าที่ให้ค่ามาตรวัด ANAUW - getANAW() ทำหน้าที่ให้ค่ามาตรวัด ANAW - getANMUW() ทำหน้าที่ให้ค่ามาตรวัด ANMUW - getANMW() ทำหน้าที่ให้ค่ามาตรวัด ANMW - getANAss() ทำหน้าที่ให้ค่ามาตรวัด ANAsso - getANAgg() ทำหน้าที่ให้ค่ามาตรวัด ANAgg - getMaxHAgg() ทำหน้าที่ให้ค่ามาตรวัด MaxHAgg - getnumNaggH() ทำหน้าที่ให้ค่ามาตรวัด NaggH - getnumMaxHAgg() ทำหน้าที่ให้ค่ามาตรวัด MaxHAgg - getANGen() ทำหน้าที่ให้ค่ามาตรวัด ANGen - getnumNGenH() ทำหน้าที่ให้ค่ามาตรวัด NGenH - getnumMaxDIT() ทำหน้าที่ให้ค่ามาตรวัด MaxDIT - getNOS() ทำหน้าที่ให้ค่ามาตรวัด NOS - getWMBO() ทำหน้าที่ให้ค่ามาตรวัด WMBO - getANRM() ทำหน้าที่ให้ค่ามาตรวัด ANRM - getnumANDM() ทำหน้าที่ให้ค่ามาตรวัด ANDM - getnumANET() ทำหน้าที่ให้ค่ามาตรวัด ANET - getnumANCM() ทำหน้าที่ให้ค่ามาตรวัด ANCM - Metrics() ทำหน้าที่สร้างวัตถุของคลาส Metrics

รูปที่ จ-3 แสดงคลาส Metrics

3.1 รหัสต้นฉบับของคลาส Metrics

```
public class Metrics {
        private JClass jclass;
        private JAttribute jattribute;
```

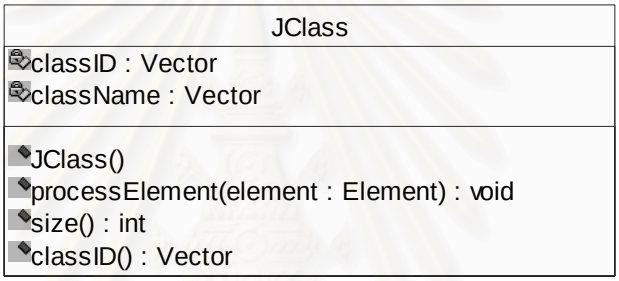
```
	private JMethod jmethod;
	private JScenario jscenario;
	private JGeneralization jgen;
	private JAssociation jassociation;
	private int numNC, numNOS, numNGenH, numNaggH, numMaxHAgg, numMaxDIT;
	private float numANAUW, numANAW, numANMUW, numANMW, numWMBO, numANRM;
	private float numANGen, numANAss, numANAgg, numANDM, numANET, numANCM;
	public Metrics(Element element, String num){
		Float tmp = new Float(num);
		numANCM = tmp.floatValue();
		jclass = new JClass();
		jattribute = new JAttribute();
		jmethod = new JMethod();
		jscenario = new JScenario();
		jgen = new JGeneralization();
		jassociation = new JAssociation();
		processElement(element);
		calculateMetrics();
	}
	private void processElement(Element element) {
		String elementName = element.getName();
		if (elementName.equals("Class") || elementName.equals("AssociationClass"))  {
			jclass.processElement(element);
		}
		if (elementName.equals("Attribute"))  {
			jattribute.processElement(element);
		}
		if (elementName.equals("Operation"))  {
			jmethod.processElement(element);
		}
		if (elementName.equals("Interaction")||elementName.equals("Message")||
elementName.equals("Message.sender") || elementName.equals("Message.receiver")) {
			jscenario.processElement(element);
		}
```

```
                if (elementName.equals("ClassifierRole") ||
elementName.equals("Foundation.Core.Classifier")) {
                        jscenario.processCollaboration(element);
                }
                if (elementName.equals("Generalization") ||
elementName.equals("Generalization.child") || elementName.equals("Generalization.parent"))  {
                        jgen.processElement(element);
                }
                if (elementName.equals("Association")) {
                        jassociation.processElement(element);
                }
                if (elementName.equals("AssociationEnd")) {
                        jassociation.processAssoEnd(element);
                }
                if (elementName.equals("AssociationEnd.type")) {
                        jassociation.processAssoEndType(element);
                }
                List kids = element.getChildren();
                Iterator iterator = kids.iterator();
                while (iterator.hasNext()) {
                        Element kid = (Element)iterator.next();
                        processElement(kid);
                }
        }
        private void calculateMetrics(){
                numNC = jclass.size();
                numANAUW = jattribute.size();
                numANAW = 0;
                for (int i = 0; i<jattribute.size(); i++) {
                        String modifier = jattribute.attributeModifer(i);
                        if (modifier.equals("public"))
                                numANAW += 1;
                        else if (modifier.equals("protected"))
                                numANAW += 0.5;
                }
```

```
            numANAUW = numANAUW/(float)numNC;
            numANAW = numANAW/(float)numNC;
            numANMUW = jmethod.size();
            numANMW = 0;
            for (int i = 0; i<jmethod.size(); i++) {
                        String modifier = jmethod.methodModifier(i);
                        if (modifier.equals("public"))
                                    numANMW += 1;
                        else if (modifier.equals("protected"))
                                    numANMW += 0.5;
            }
numANMUW = numANMUW/(float)numNC;
numANMW = numANMW/(float)numNC;
numANAss = (float)jassociation.getNumAsso(jclass.classID())/(float)numNC;
numANAgg = (float)jassociation.getNumAgg()/(float)numNC;
numNaggH = jassociation.getNumChild();
numMaxHAgg = jassociation.getMaxDepth();
numANGen = ((float)jgen.getChidren()/(float)numNC);
numNGenH = jgen.getRootParent();
numMaxDIT = jgen.getMaxDepth();
numNOS = jscenario.size();
numWMBO = jscenario.calWMBO(jclass.classID())/(float)numNOS;
numANRM = (float)jscenario.getMessageReturn()/(float)numNOS;
numANDM = (float)jscenario.getANDM()/(float)numNOS;
numANET = (float)jscenario.getANET()/(float)numNOS;
}
public int getNC()      {   return numNC;       }
public float getANAUW() {   return numANAUW;    }
public float getANAW()  {   return numANAW;     }
public float getANMUW() {   return numANMUW;    }
public float getANMW()  {   return numANMW;     }
public float getANAss() {   return numANAss;    }
public float getANAgg() {   return numANAgg;    }
public float getMaxHAgg() { return numMaxHAgg;  }
public int getnumNaggH(){   return numNaggH;    }
```

```
        public int getnumMaxHAgg(){ return numMaxHAgg; }
        public float getANGen() {   return numANGen;        }
        public int getnumNGenH(){   return numNGenH;        }
        public int getnumMaxDIT(){  return numMaxDIT;       }
        public int getNOS()     {   return numNOS;          }
        public float getWMBO()  {   return numWMBO;         }
        public float getANRM()  {   return numANRM;         }
        public float getnumANDM(){  return numANDM;         }
        public float getnumANET(){  return numANET;         }
        public float getnumANCM(){  return numANCM;         }
}
```

4. คลาส JClass



รูปที่ จ-4 แสดงคลาส JClass

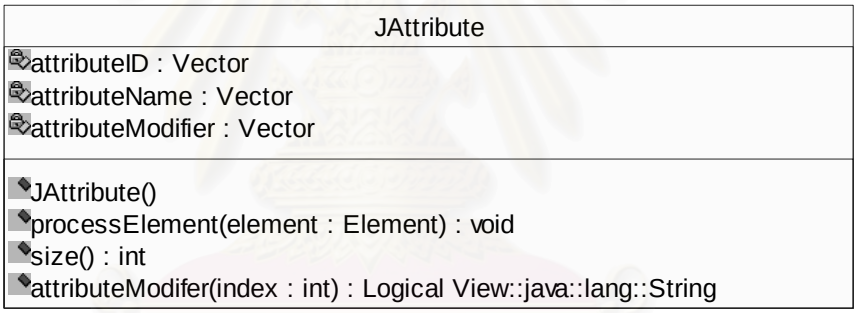
ชื่อคลาส	JClass
คุณลักษณะ	- classID สำหรับเก็บหมายเลขอ้างอิงของคลาส - className สำหรับเก็บชื่อคลาส
เมทธอด	- JClass() ทำหน้าที่สร้างวัตถุของคลาส JClass - processElement() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ classID และ className - size() ทำหน้าที่ให้ค่าขนาดของคุณลักษณะ classID - classID() ทำหน้าที่ให้ค่าของคุณลักษณะ classID

4.1 รหัสต้นฉบับของคลาส JClass

```
public class JClass {
        private Vector classID;
        private Vector className;
        public JClass(){
```

```
                classID = new Vector();
                className = new Vector();
        }
        public void processElement(Element element){
                String elementName = element.getName();
                if (element.getParent().getParent().getName().equals("Model")) {
        classID.addElement(element.getAttributeValue("xmi.id").toString());
        className.addElement(element.getAttributeValue("name").toString());
                }
        }
        public int size() {             return classID.size();      }
        public Vector classID() {   return classID;    }
}
```

5. **คลาส** JAttribute



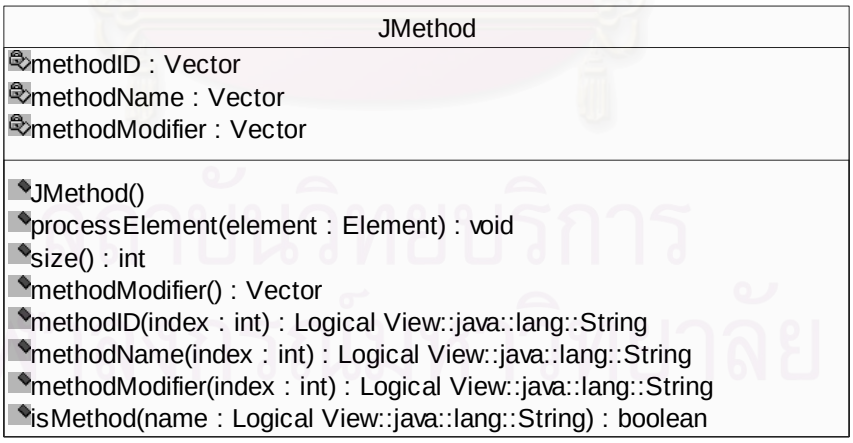
รูปที่ จ-5 แสดงคลาส JAttribute

ชื่อคลาส	JAttribute
คุณลักษณะ	- attributeID สำหรับเก็บหมายเลขอ้างอิงของคุณลักษณะ - attributeName สำหรับเก็บชื่อของคุณลักษณะ - attributeModifier สำหรับเก็บค่าตัวดัดแปรของคุณลักษณะ
เมทธอด	- JAttribute() ทำหน้าที่สร้างวัตถุของคลาส JAttribute - processElement() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ attributeID, attributeName และ attributeModifier - size() ทำหน้าที่ให้ค่าขนาดของคุณลักษณะ attributeID - attributeModifier() ทำหน้าที่ให้ค่าของคุณลักษณะ attributeModifier

5.1 รหัสต้นฉบับของคลาส JAttribute

```
public class JAttribute {
        private Vector attributeID;
        private Vector attributeName;
        private Vector attributeModifier;
        public JAttribute(){
                attributeID = new Vector();
                attributeName = new Vector();
                attributeModifier = new Vector();
        }
        public void processElement(Element element){
        attributeID.addElement(element.getAttributeValue("xmi.id").toString());
        attributeName.addElement(element.getAttributeValue("name").toString());
                attributeModifier.addElement(element.getAttributeValue("visibility"));
        }
public String attributeModifer(int index) { return attributeModifier.get(index).toString();}
public int size() { return attributeID.size();    }
}
```

6. คลาส JMethod



รูปที่ จ-6 แสดงคลาส JMethod

ชื่อคลาส	JMethod
คุณลักษณะ	- methodID สำหรับเก็บหมายเลขอ้างอิงของเมทธอด - methodName สำหรับเก็บชื่อของเมทธอด

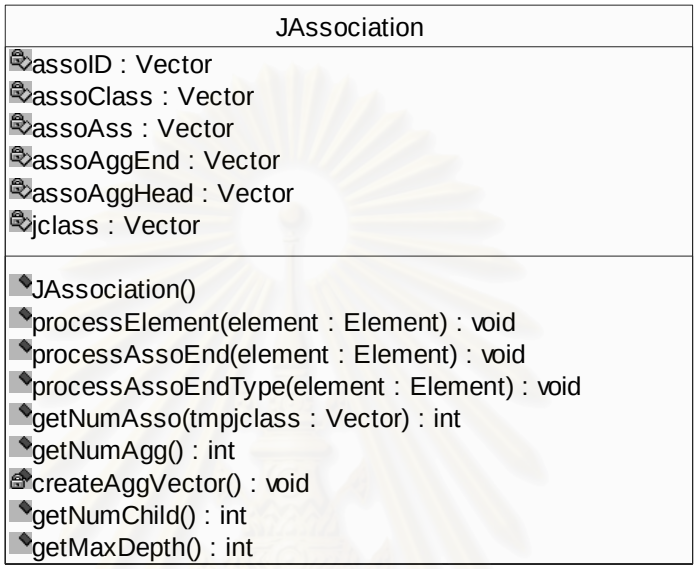
	- methodModifier สำหรับเก็บค่าตัวดัดแปรของเมทธอด
เมทธอด	- JMethod() ทำหน้าที่สร้างวัตถุของคลาส JMethod - processElement() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ methodID, methodName และ methodModifier - size() ทำหน้าที่ให้ค่าขนาดของคุณลักษณะ methodID - methodModifier() ทำหน้าที่ให้ค่าของคุณลักษณะ methodModifier - methodID() ทำหน้าที่ให้ค่าของคุณลักษณะ methodID - methodName() ทำหน้าที่ให้ค่าของคุณลักษณะ methodName - methodModifier() ทำหน้าที่ให้ค่าของคุณลักษณะ methodModifier - isMethod() ทำหน้าที่ตรวจสอบค่าที่เข้ามาว่าเป็นเมทธอดหรือไม่

6.1 รหัสต้นฉบับของคลาส JMethod

```
public class JMethod {
        private Vector methodID;
        private Vector methodName;
        private Vector methodModifier;
        public JMethod(){
                methodID = new Vector();
                methodName = new Vector();
                methodModifier = new Vector();
        }
        public void processElement(Element element){
                String elementName = element.getName();
                String id = element.getAttributeValue("xmi.id").toString();
                methodID.addElement(id);
                String name = element.getAttributeValue("name").toString();
                methodName.addElement(name);
                String attributeName = element.getAttributeValue("visibility");
        methodModifier.addElement(attributeName);
        }
        public String methodID(int index) {   return methodID.get(index).toString();        }
        public String methodName(int index) {   return methodName.get(index).toString();  }
        public String methodModifier(int index) {  return methodModifier.get(index).toString();  }
```

```
        public int size() { return methodID.size();     }
        public Vector methodModifier() {    return methodModifier;     }
        public boolean isMethod(String name) {  return methodName.contains(name);  }
}
```

7. **คลาส** JAssociation



รูปที่ จ-7 แสดงคลาส JAssociation

ชื่อคลาส	JAssociation
คุณลักษณะ	- assoID สำหรับเก็บหมายเลขอ้างอิงของความสัมพันธ์แบบแอสโซซิเอชัน - assoClass สำหรับเก็บชื่อคลาสที่มีความสัมพันธ์แบบแอสโซซิเอชัน - assoAss สำหรับเก็บค่าความสัมพันธ์แบบแอสโซซิเอชัน - assoAggEnd สำหรับเก็บค่าความสัมพันธ์แบบแอกกริเกชันด้านท้าย - assoAggHead สำหรับเก็บค่าความสัมพันธ์แบบแอกกริเกชันด้านหัว - jclass สำหรับเก็บวัตถุของคลาส JClass
เมทธอด	- JAssociation() ทำหน้าที่สร้างวัตถุของคลาส JAssociation - processElement() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ assoID, assoClass - processAssoEnd() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ assoAggEnd - processAssoEndType() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ assoAss - getNumAsso() ทำหน้าที่นับจำนวนความสัมพันธ์แบบแอสโซซิเอชัน - getNumAgg() ทำหน้าที่นับจำนวนความสัมพันธ์แบบแอกกริเกชัน

	- createAggVector() ทำหน้าที่สร้างเวกเตอร์สำหรับความสัมพันธ์แบบแอกกริเกชัน - getNumChild() ทำหน้าที่คำนวณจำนวนคลาสลูก - getMaxDepth() ทำหน้าที่คำนวณค่าความลึกของความสัมพันธ์แบบแอกกริเกชัน

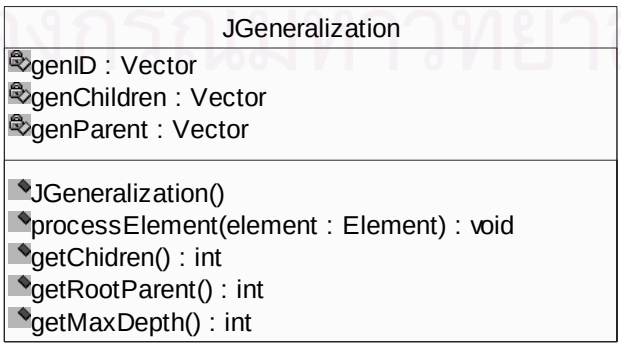
7.1 **รหัสต้นฉบับของคลาส** JAssociation

```
public class JAssociation {
        private Vector assoID;
        private Vector assoClass;
        private Vector assoAss;
        private Vector assoAggEnd;
        private Vector assoAggHead;
        private Vector jclass;
        public JAssociation() {
        assoID = new Vector();
        assoClass = new Vector();
        assoAss = new Vector();
        assoAggEnd = new Vector();
        assoAggHead = new Vector();
        jclass = new Vector();
        }
public void processElement(Element element){
        assoID.addElement(element.getAttributeValue("xmi.id"));
}
public void processAssoEnd(Element element) {
        if (!(element.getParent().getParent().getName().equals("AssociationClass")))
        assoClass.addElement(element.getAttributeValue("aggregation"));
}
public void processAssoEndType(Element element) {
        if (element.getParent().getParent().getParent().getName().equals("Association")) {
        assoAss.addElement(element.getChild("Foundation.Core.Classifier").getAttributeValue("xmi.i
dref"));
        }
}
public int getNumAsso(Vector tmpjclass) {
```

```
            int count = 0;
            jclass = tmpjclass;
            for (int i=0; i<assoClass.size();i+=2)
if    (assoClass.elementAt(i).equals("none")    &&    assoClass.elementAt(i+1).equals("none")    &&
jclass.contains(assoAss.elementAt(i)))
            count++;
            return count;
}
public int getNumAgg() {
            int count = 0;
            for (int i=0; i<assoClass.size();i++)
            if (assoClass.elementAt(i).equals("composite") ||
                        assoClass.elementAt(i).equals("aggregate")
                        && jclass.contains(assoAss.elementAt(i)))
                        count++;
                        return count;
            }
            private void createAggVector() {
                        for (int j=0; j<assoAss.size();j+=2) {
            if (assoClass.elementAt(j).equals("composite") ||
                        assoClass.elementAt(j).equals("aggregate")
                        && jclass.contains(assoAss.elementAt(j))) {
                                    assoAggEnd.addElement(assoAss.elementAt(j+1));
                                    assoAggHead.addElement(assoAss.elementAt(j));
                        }
if (assoClass.elementAt(j+1).equals("composite") || assoClass.elementAt(j+1).equals("aggregate") &&
jclass.contains(assoAss.elementAt(j+1))) {
            assoAggEnd.addElement(assoAss.elementAt(j));
            assoAggHead.addElement(assoAss.elementAt(j+1));
            }
}
}
public int getNumChild() {
            createAggVector();  Vector temp = new Vector();       boolean flg = false;
            for (int i=0; i<assoAggHead.size(); i++) {
```

```
                flg = false;
                if (assoAggEnd.contains(assoAggHead.elementAt(i))) {  flg = true;   }
                if (!(flg)) {           temp.addElement(assoAggHead.elementAt(i)); }
        }
        Vector tmp = new Vector();
        for (int j=0; j<temp.size(); j++)
                if (!(tmp.contains(temp.elementAt(j))))
                        tmp.addElement(temp.elementAt(j));
                return tmp.size();
        }
public int getMaxDepth() {
        int max = 0;                int count = 0;
        String head = "";           String end = "";
        for (int i=0; i<assoAggHead.size(); i++) {
                end = assoAggEnd.elementAt(i).toString();
                count=1;
                for (int j=0; j<assoAggHead.size(); j++) {
                        if (assoAggHead.elementAt(j).equals(end)) {
                                count++;    j=0;
                                end = assoAggEnd.elementAt(j).toString();
                }
        }
        if (count > max)   {     max = count;    }
}       return max;
} }
```

8. คลาส JGeneralization



รูปที่ จ-8 แสดงคลาส JGeneralization

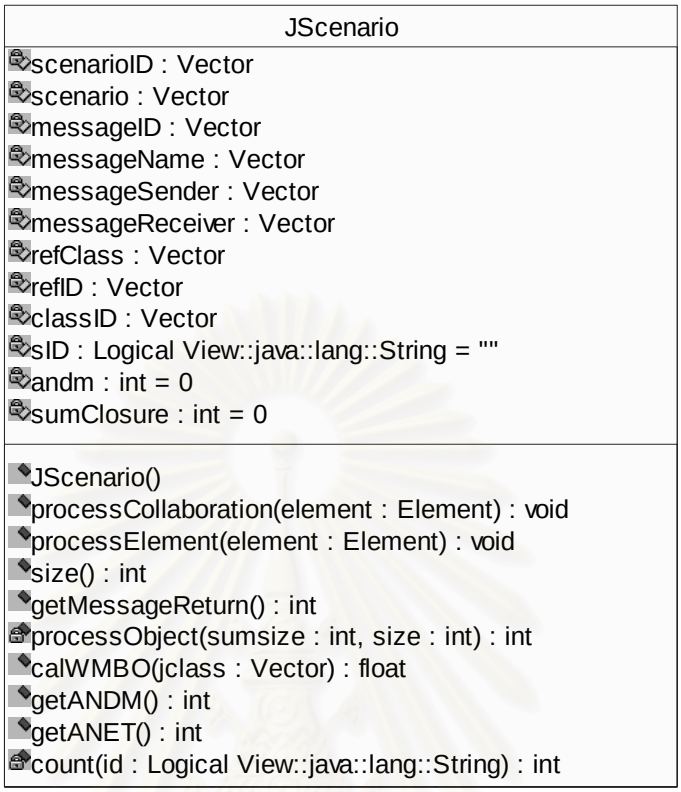
ชื่อคลาส	JGeneralization
คุณลักษณะ	- genID สำหรับเก็บหมายเลขอ้างอิงของความสัมพันธ์แบบเจเนอรัลไรเซชัน - genChildren สำหรับเก็บชื่อคลาสลูกที่มีความสัมพันธ์แบบเจเนอรัลไรเซชัน - genParent สำหรับเก็บชื่อคลาสแม่ที่มีความสัมพันธ์แบบเจเนอรัลไรเซชัน
เมทธอด	- JGeneralization() ทำหน้าที่สร้างวัตถุของคลาส JGeneralization - processElement() ทำหน้าที่เก็บข้อมูลลงในคุณลักษณะ genID, genChildren และ genParent - getChildren() ทำหน้าที่ให้ข้อมูลคุณลักษณะ genChildren - getRootParent() ทำหน้าที่ให้ข้อมูลคุณลักษณะ genParent - getMaxDepth() ทำหน้าที่นับจำนวนความลึกของความสัมพันธ์แบบเจเนอรัลไรเซชัน

8.1 **รหัสต้นฉบับของคลาส** JGeneralization

```
public class JGeneralization {
        private Vector genID;
        private Vector genChildren;
        private Vector genParent;
        public JGeneralization() {
                genID = new Vector();
                genChildren = new Vector();
                genParent = new Vector();
        }
        public void processElement(Element element){
                String elementName = element.getName();
                if (elementName.equals("Generalization"))
                        genID.addElement(element.getAttributeValue("xmi.id").toString());
                if (elementName.equals("Generalization.child"))
        genChildren.addElement(element.getChild("Foundation.Core.GeneralizableElement").getAttr
ibuteValue("xmi.idref").toString());
                if (elementName.equals("Generalization.parent"))
        genParent.addElement(element.getChild("Foundation.Core.GeneralizableElement").getAttrib
uteValue("xmi.idref").toString());
        }
```

```
public int getChidren() {
        Vector temp = new Vector();
        for (int i=0; i<genChildren.size(); i++)
                if (!temp.contains(genChildren.elementAt(i)))
                        temp.addElement(genChildren.elementAt(i));
        return temp.size();
}
public int getRootParent() {
        Vector temp = new Vector();          boolean flg = false;
        for (int i=0; i<genParent.size(); i++) {
                flg = false;
        if (genChildren.contains(genParent.elementAt(i)))    {  flg = true;   }
        if (!(flg))  { temp.addElement(genParent.elementAt(i));   }
        }
        Vector tmp = new Vector();
        for (int j=0; j<temp.size(); j++)
                if (!(tmp.contains(temp.elementAt(j))))
                        tmp.addElement(temp.elementAt(j));
        return tmp.size();
}
public int getMaxDepth() {
        int max = 0;                    int count = 0;
        String head = "";               String end = "";
        for (int i=0; i<genParent.size(); i++) {
                end = genChildren.elementAt(i).toString();
                count=1;
                for (int j=0; j<genParent.size(); j++) {
                        if (genParent.elementAt(j).equals(end)) {
                                count++;   j=0;
                                end = genChildren.elementAt(j).toString();
                        }
                }
                if (count > max) {     max = count;   }
        }
        return max;         }}
```

9. **คลาส** JScenario



รูปที่ จ-9 แสดงคลาส JScenario

ชื่อคลาส	JScenario
คุณลักษณะ	- scenarioID สำหรับเก็บหมายเลขอ้างอิงของซีนาริโอ - scenario สำหรับเก็บชื่อซีนาริโอ - messageID สำหรับเก็บหมายเลขอ้างอิงของสาร - messageName สำหรับเก็บชื่อสาร - messageSender สำหรับเก็บหมายเลขอ้างอิงของคลาสที่เป็นผู้ส่งสาร - messageReceiver สำหรับเก็บหมายเลขอ้างอิงของคลาสที่เป็นผู้รับสาร - refClass สำหรับเก็บหมายเลขอ้างอิงของคลาสที่ถูกอ้างถึงภายในซีนาริโอ - refID สำหรับเก็บการอ้างถึงหมายเลขอ้างอิงของซีนาริโอภายในซีนาริโอ - classID สำหรับเก็บหมายเลขอ้างอิงของคลาสทั้งหมด - sID สำหรับเก็บจำนวนซีนาริโอ - andm สำหรับเก็บค่ามาตรวัด andm - sumClosure สำหรับเก็บค่ามาตรวัด ANET

เมทธอด	- JScenario() ทำหน้าที่สร้างวัตถุของคลาส JScenario - processCollaboration ทำหน้าที่คำนวณค่าคุณลักษณะ refID และ refClass - processElement() ทำหน้าที่คำนวณค่าคุณลักษณะ scenarioID, scenario, messageID, messageName, messageSender และ messageReceiver - size() ทำหน้าที่ให้ขนาดของคุณลักษณะ scenarioID - getMessageReturn() ทำหน้าที่ให้ข้อมูลสารที่เป็นการคืนค่า - processObject() ทำหน้าที่นับจำนวนวัตถุของแต่ละซีนาริโอ - calWMBO() ทำหน้าที่คำนวณค่ามาตรวัด WMBO - getANDM() ทำหน้าที่ให้ค่าคุณลักษณะ andm - getANET() ทำหน้าที่คำนวณและให้ค่ามาตรวัด ANET - count() ทำหน้าที่คำนวณและให้ค่าจำนวนซีนาริโอ

9.1 **รหัสต้นฉบับของคลาส** JScenario

```
public class JScenario {
        private Vector scenarioID;
        private Vector scenario;
        private Vector messageID;
        private Vector messageName;
        private Vector messageSender;
        private Vector messageReceiver;
        private Vector refClass;
        private Vector refID;
        private Vector classID;
        private String sID = "";
        private int andm=0,sumClosure = 0;
        public JScenario(){
                scenarioID = new Vector();
                scenario = new Vector();
                messageID = new Vector();
                messageName = new Vector();
                messageSender = new Vector();
                messageReceiver = new Vector();
                refClass = new Vector();
```

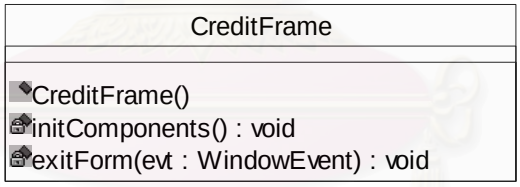
```
		refID = new Vector();
		classID = new Vector();
	}
	public void processCollaboration(Element element) {
		String elementName = element.getName();
		if (elementName.equals("ClassifierRole")) {
			if (element.getParent().getParent().getName().equals("Collaboration"))
				refClass.addElement(element.getAttributeValue("xmi.id"));
		}
		if (elementName.equals("Foundation.Core.Classifier")) {
			if
(element.getParent().getParent().getParent().getParent().getName().equals("Collaboration"))
				refID.addElement(element.getAttributeValue("xmi.idref"));
		}
	}
	public void processElement(Element element){
		String elementName = element.getName();
		if (elementName.equals("Interaction")) {
			String id = element.getAttributeValue("xmi.id").toString();
			sID = id;
			scenarioID.addElement(id);
		}
		if (elementName.equals("Message")) {
			messageID.addElement(element.getAttributeValue("xmi.id").toString());
			messageName.addElement(element.getAttributeValue("name").toString());
			scenario.addElement(sID);
		}
		if (elementName.equals("Message.sender")) {
	messageSender.addElement(element.getChild(("Behavioral_Elements.Collaborations.Classif
ierRole")).getAttributeValue("xmi.idref").toString());
		}
		if (elementName.equals("Message.receiver")) {
	messageReceiver.addElement(element.getChild("Behavioral_Elements.Collaborations.Class
ifierRole").getAttributeValue("xmi.idref").toString());
		}
```

```
}
public int size() {		return scenarioID.size();	}
public int getMessageReturn() {
		int count=0;
		for (int j=0;j<messageID.size();j++)
			if (messageName.elementAt(j).toString().indexOf("(") == -1)
				count++;
		return count;
}
private int processObject(int sumsize,int size) {
		Vector object = new Vector();
		object.removeAllElements();
		for (int i = sumsize; i<sumsize+size ; i++) {
			int index1 = refClass.indexOf(messageSender.elementAt(i));
			if (classID.contains(refID.elementAt(index1))) {
				if (!object.contains(messageSender.elementAt(i)))
object.addElement(messageSender.elementAt(i).toString());
			}
			int index2 = refClass.indexOf(messageReceiver.elementAt(i));
			if (classID.contains(refID.elementAt(index2))) {
				if (!object.contains(messageReceiver.elementAt(i)))
object.addElement(messageReceiver.elementAt(i).toString());
			}
		}
		return object.size();
}
private int count(String id) {
		int num=0;
		for (int k=0; k<scenario.size();k++)
			if (scenario.elementAt(k).equals(id))
				num++;
		return num;
}
public float calWMBO(Vector jclass) {
		classID = jclass;
```

```
int msg = 0, size=0, sumsize=0, temp=0, closure = 0;
float sum = 0;
String first="", receiver = "";
andm = 0;
sumClosure = 0;
for (int i=0; i<scenarioID.size(); i++) {
        msg = 0;
        temp = 0;
        closure = 0;
        first = scenarioID.elementAt(i).toString();
        size = count(first);
        for (int j=sumsize; j<sumsize+size; j++) {
                if (messageName.elementAt(j).toString().indexOf("(") != -1)
                        msg++;
                if (j != messageSender.size()-1) {
        if ((!messageReceiver.elementAt(j).equals(messageSender.elementAt(j)))
&&  messageReceiver.elementAt(j).equals(messageSender.elementAt(j+1)) &&
(messageName.elementAt(j).toString().indexOf("(") != -1) &&
(messageName.elementAt(j+1).toString().indexOf("(") != -1))
                                        temp++;
                }
        }
        int number = 0;
        for (int k=sumsize; k<sumsize+size; k++) {
                number = 0;
        if (messageName.elementAt(k).toString().indexOf("(") != -1) {
                receiver = messageReceiver.elementAt(k).toString();
                for (int l=k+1;l<messageSender.size();l++) {
        if ((!messageReceiver.elementAt(l).equals(messageSender.elementAt(l)))
        && messageReceiver.elementAt(l-1).equals(messageSender.elementAt(l))
        && (messageName.elementAt(l).toString().indexOf("(") != -1)) {
                number++;
        if (receiver.equals(messageReceiver.elementAt(l))) {
                closure += number;
                k = l;
```

```
                        }
                } else
                        l = messageSender.size();
                }
                 }
        }
                sum += (float)msg/(float)processObject(sumsize,size);
                sumClosure += closure;
                sumsize += size;
                andm += temp;
        }
        return sum;
        }
        public int getANDM() {      return andm;      }
        public int getANET() {      return sumClosure;          }
}
```

10. คลาส CreditFrame



รูปที่ จ-10 แสดงคลาส CreditFrame

ชื่อคลาส	CreditFrame
คุณลักษณะ	ไม่มี
เมทธอด	- CreditFrame() ทำหน้าที่สร้างวัตถุของคลาส CreditFram() - initComponents() ทำหน้าที่สร้างคอมโพเนนท์ภายในหน้าจอและแสดงผล - exitForm() ทำหน้าที่ทำลายคลาส CreditFrame

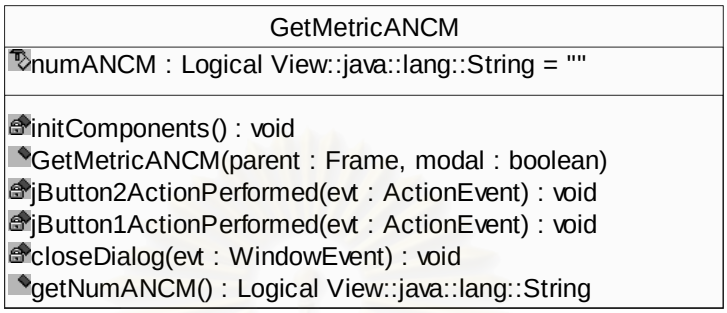
10.1 รหัสต้นฉบับของคลาส CreditFrame

```
public class CreditFrame extends javax.swing.JFrame {
   public CreditFrame() {
```

```
    initComponents();
}
private void initComponents() {//GEN-BEGIN:initComponents
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    getContentPane().setLayout(null);
    setTitle("About");
    setForeground(new java.awt.Color(153, 204, 255));
    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
            exitForm(evt);
        }
    });
    jLabel1.setText("Program Java Tool fOr Prediction (JTOP) Ver. 1.0");
    getContentPane().add(jLabel1);
    jLabel1.setBounds(80, 30, 290, 16);
    jLabel2.setText("Created by :: Nongyao  Jindasawat");
    getContentPane().add(jLabel2);
    jLabel2.setBounds(20, 260, 200, 16);
    jLabel3.setText("Sofware Engineering Laboratory, Department of Computer Engineering,");
    getContentPane().add(jLabel3);
    jLabel3.setBounds(20, 280, 410, 16);
    jLabel4.setText("Faculty of Engineering, Chulalongkorn University");
    getContentPane().add(jLabel4);
    jLabel4.setBounds(20, 300, 360, 16);
    pack();
}//GEN-END:initComponents
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    setVisible(false);
    dispose();
}//GEN-LAST:event_exitForm
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
```

```
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
}
```

11. คลาส GetMetricANCM



รูปที่ จ-11 แสดงคลาส GetMetricANCM

ชื่อคลาส	GetMetricANCM
คุณลักษณะ	numANCM สำหรับเก็บค่ามาตรวัด ANCM
เมทธอด	- initComponents() ทำหน้าที่สร้างคอมโพเนนท์ภายในหน้าจอและแสดงผล - GetMetricANCM() ทำหน้าที่สร้างวัตถุของคลาส GetMetricANCM - jButton2ActionPerformed() ทำหน้าที่รับยกเลิกค่าที่ผู้ใช้ป้อนทางหน้าจอ - jButton1ActionPerformed() ทำหน้าที่รับค่ามาตรวัดจากผู้ใช้งานส่งไปให้คลาส JPredicitonTOOL - closeDialog() ทำหน้าที่ปิดหน้าจอรับค่าข้อมูลทางหน้าจอ - getNumANCM() ทำหน้าที่ให้ค่าคุณลักษณะ numANCM

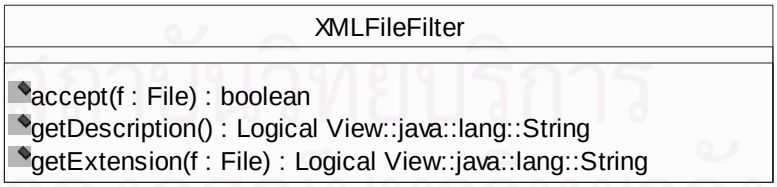
11.1 รหัสต้นฉบับของคลาส GetMetricANCM

```
public class GetMetricANCM extends javax.swing.JDialog {
    String numANCM = "";
    public GetMetricANCM(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
    }
    private void initComponents() {//GEN-BEGIN:initComponents
        jLabel1 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
```

```
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    getContentPane().setLayout(null);
    setTitle("INPUT ANCM VALUE");
    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
            closeDialog(evt);
        }
    });
    jLabel1.setText("Please input the value of metric ANCM");
    getContentPane().add(jLabel1);
    jLabel1.setBounds(90, 30, 220, 16);
    getContentPane().add(jTextField1);
    jTextField1.setBounds(130, 70, 140, 20);
    jButton1.setText("Ok");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton1);
    jButton1.setBounds(100, 110, 81, 26);
    jButton2.setText("Cancel");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton2);
    jButton2.setBounds(220, 110, 73, 26);
    pack();
}//GEN-END:initComponents
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    dispose();
```

```
    }//GEN-LAST:event_jButton2ActionPerformed
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        if (jTextField1.getText().equals(""))
            numANCM = "0";
        else
            numANCM = jTextField1.getText();
        setVisible(false);
        dispose();
    }//GEN-LAST:event_jButton1ActionPerformed
    private void closeDialog(java.awt.event.WindowEvent evt) {
        setVisible(false);
        dispose();
    }//GEN-LAST:event_closeDialog
    public String getNumANCM() {
        return numANCM;
    }
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField jTextField1;
}
```

12. คลาส XMLFileFilter



รูปที่ จ-12 แสดงคลาส XMLFileFilter

ชื่อคลาส	GetMetricANCM
คุณลักษณะ	ไม่มี
เมทธอด	- accept() ทำหน้าที่ตรวจสอบว่าไฟล์มีนามสกุลตรงตามที่กำหนดไว้หรือไม่ - getDescription() ทำหน้าที่ดึงคำอธิบายของนามสกุลไฟล์ - getExtension() ทำหน้าที่ดึงข้อมูลนามสกุลของไฟล์

12.1 รหัสต้นฉบับของคลาส XMLFileFilter

```
class XMLFileFilter extends javax.swing.filechooser.FileFilter {
    public boolean accept(java.io.File f) {
        if (f.isDirectory()) {
            return true;
        }
        String extension = getExtension(f);
        if (extension != null) {
            extension = extension.toLowerCase();
            if (extension.equals("xml"))
                return true;
        }
        return false;
    }
    public String getDescription() {
        return "XML file";
    }
     public String getExtension(java.io.File f) {
            if(f != null) {
                String filename = f.getName();
                int i = filename.lastIndexOf('.');
                if(i>0 && i<filename.length()-1) {
                        return filename.substring(i+1).toLowerCase();
                };
            }
            return null;
    }
}
```

ภาคผนวก ฉ.
การใช้งานเครื่องมือเพื่อการคำนวณมาตรวัด และทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

ในภาคผนวกนี้จะอธิบายถึงการติดตั้งเครื่องมือเพื่อการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ (Java Tool for Calculating Metrics and Predicting Maintainability) และการใช้งาน ซึ่งแบ่งออกเป็น 3 ขั้นตอน คือ การอ่านไฟล์ข้อมูลเอ็กซ์เอ็มแอล การคำนวณค่ามาตรวัด และการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ดังรายละเอียดต่อไปนี้

1. การติดตั้งเครื่องมือเพื่อการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

เครื่องมือเพื่อการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ ประกอบด้วยไฟล์ข้อมูลหลัก 2 ไฟล์ คือ JTOP.bat ซึ่งเป็นไฟล์ MS-DOS Batch File เพื่อใช้เรียกไฟล์ JTOP.jar ซึ่งเป็นไฟล์ Executable Jar File ให้เริ่มต้นรันโปรแกรม ขั้นตอนการติดตั้งเครื่องมือเพื่อการทำนาย มีดังนี้

1.1 คัดลอกไฟล์ข้อมูลทั้ง 2 ไฟล์ไว้ที่เครื่องคอมพิวเตอร์ซึ่งติดตั้งโปรแกรม Java™ 2 Standard Edition Runtime Environment (JRE) ไว้แล้ว ในที่นี้ได้ทำการคัดลอกไฟล์ทั้งสองไว้ที่ C:\JTOP แต่ถ้าเครื่องคอมพิวเตอร์ยังไม่ติดตั้งโปรแกรมดังกล่าวสามารถ Download ได้ที่ http://java.sun.com

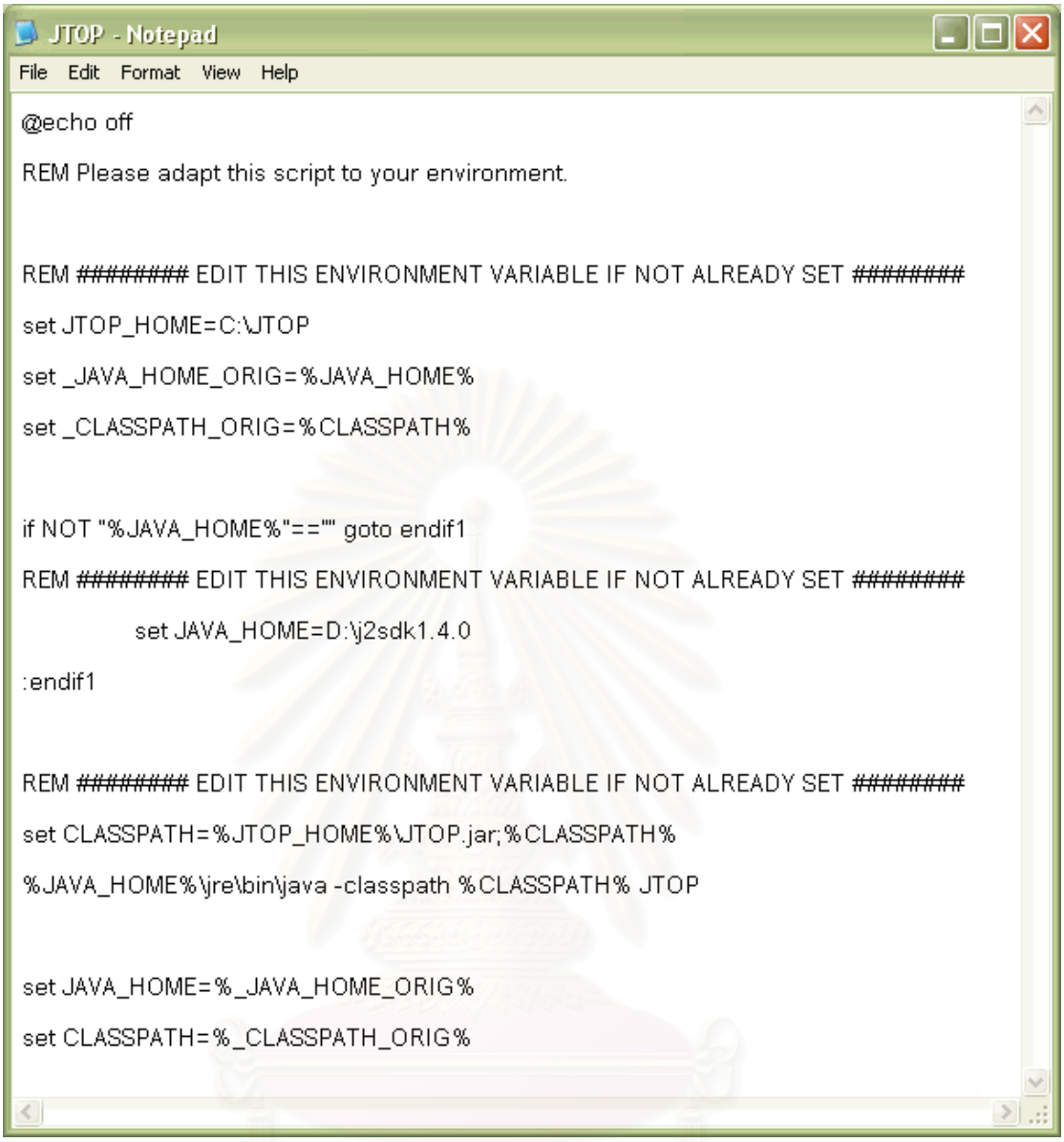
1.2 เปิดไฟล์ JTOP.bat ด้วยโปรแกรม Notepad หรือโปรแกรม Text Editor อื่นๆ เพื่อแก้ไขค่าตัวแปรตามให้เข้ากับสภาพแวดล้อมของเครื่องคอมพิวเตอร์ที่ใช้รันโปรแกรม

1.3 จากรูปที่ ฉ-1 ทำการแก้ไขค่าข้อมูลของตัวแปร JTOP_HOME ในบรรทัดที่ 4 ตามไดเรกทอรีที่ติดตั้งเครื่องมือเพื่อการคำนวณไว้ ในที่นี้กำหนดค่าไว้เป็น C:\JTOP

1.4 แก้ไขค่าข้อมูลของตัวแปร JAVA_HOME ในบรรทัดที่ 9 ตามไดเรกทอรีที่ติดตั้งโปรแกรม JRE ไว้ ในที่นี้กำหนดค่าไว้เป็น D:\j2sdk1.4.0

1.5 แก้ไขค่าข้อมูลของตัวแปร CLASSPATH ในบรรทัดที่ 12 ตามไดเรกทอรีที่เก็บไฟล์แอพลิเคชั่น java ไว้ ในที่นี้กำหนดค่าไว้เป็น \jre\bin\java

1.6 ขั้นตอนสุดท้าย ทำการบันทึกไฟล์ JTOP.bat

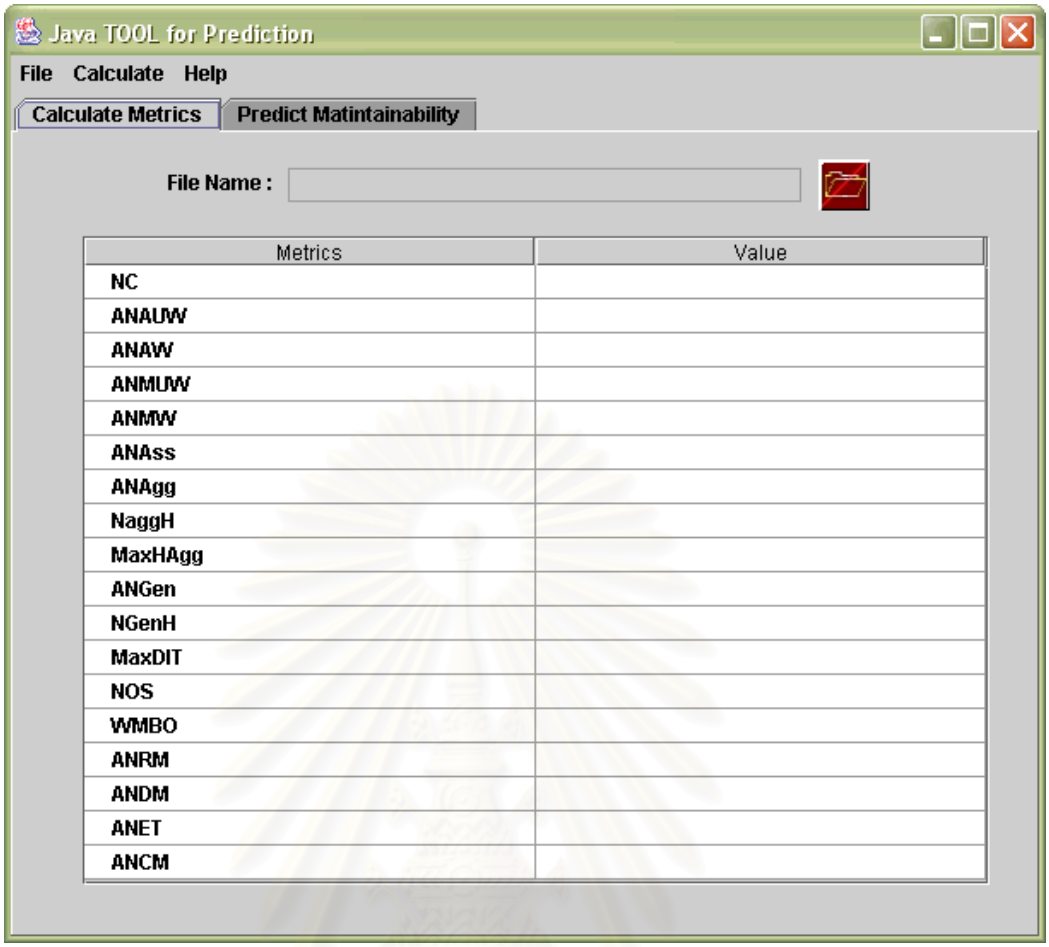


รูปที่ ฉ-1 แสดงไฟล์ข้อมูล JTOP.bat ที่เปิดด้วยโปรแกรม Notepad

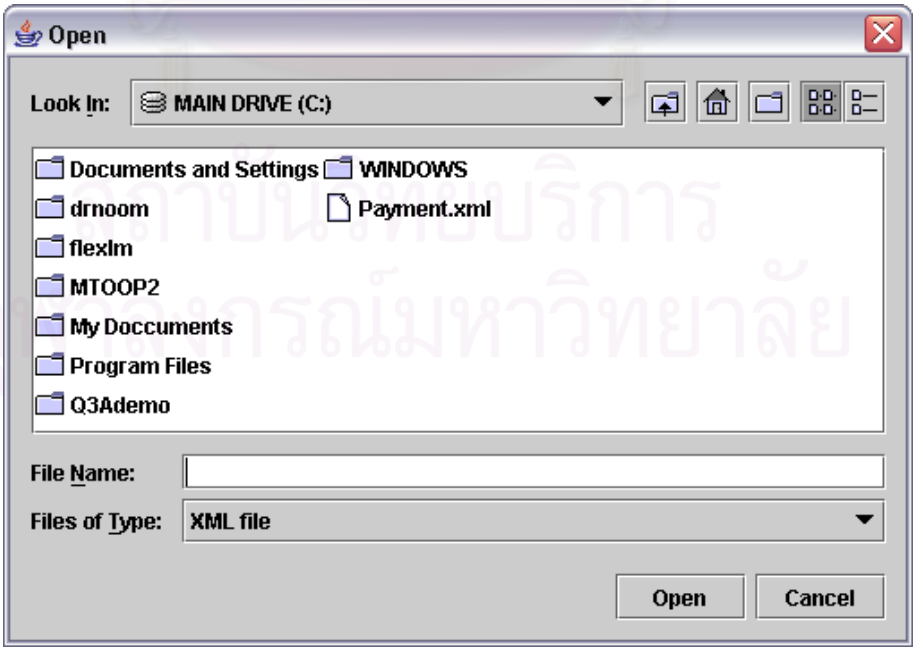
2. การใช้งานเครื่องมือเพื่อการคำนวณมาตรวัดและทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

2.1 การอ่านไฟล์ข้อมูลเอ็กซ์เอ็มแอล

เมื่อผู้ใช้งานเข้าสู่โปรแกรมจะแสดงหน้าจอโปรแกรมดังแสดงในรูปที่ ฉ-2 จากนั้นผู้ใช้งานสามารถเลือกไฟล์ข้อมูลเอ็กซ์เอ็มแอลได้จากเมนูเปิดไฟล์ (File -> Open) หรือคลิกที่ปุ่ม  จะแสดงหน้าจอให้เลือกไฟล์ข้อมูลดังแสดงในรูปที่ ฉ-3



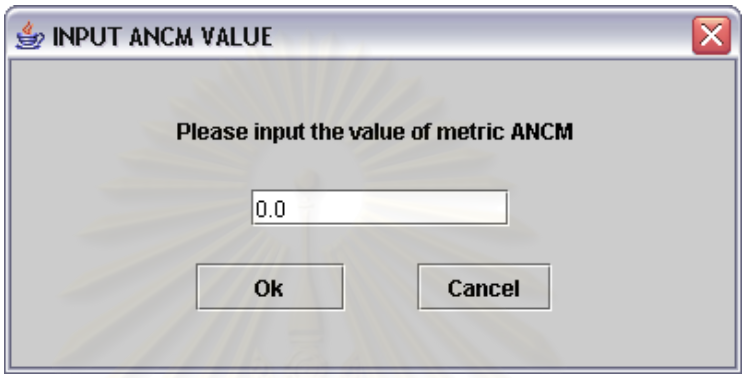
รูปที่ ฉ-2 แสดงหน้าจอเครื่องมือเพื่อการคำนวณมาตรวัด
และทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์



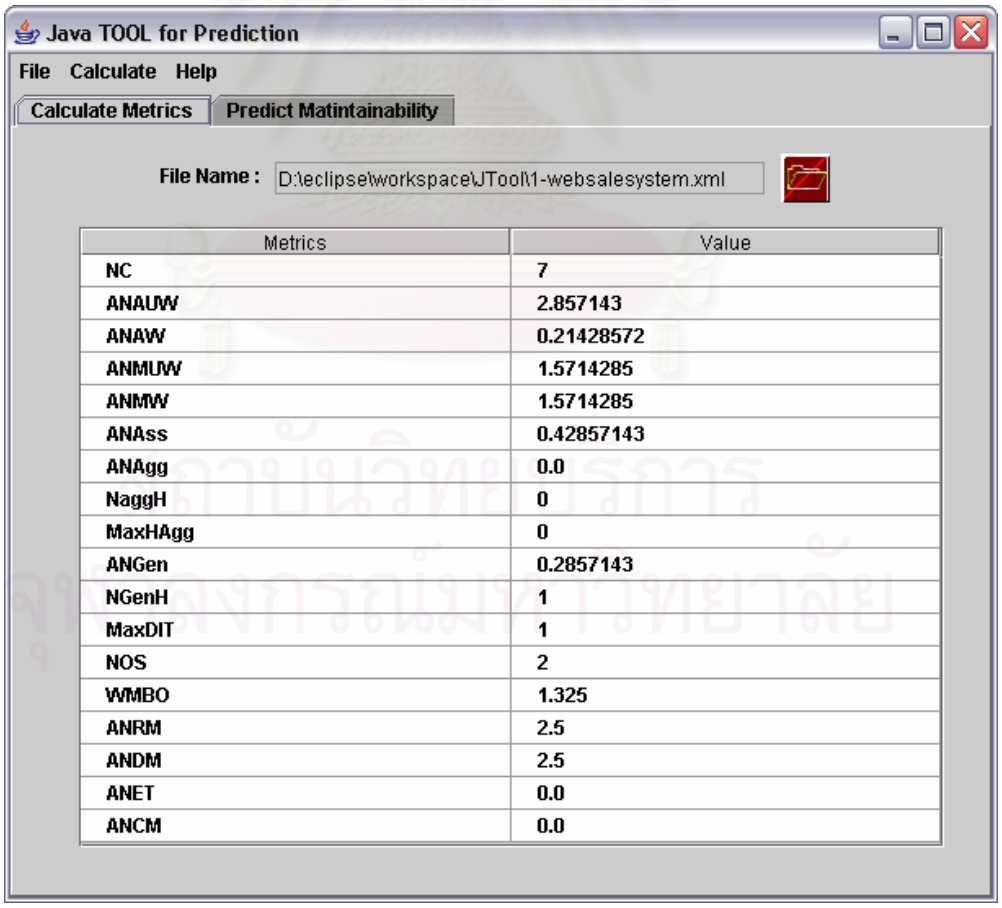
รูปที่ ฉ-3 แสดงหน้าจอเลือกไฟล์ข้อมูลเอ็กซ์เอ็มแอล

2.2 การคำนวณค่ามาตรวัด

หลังจากผู้ใช้งานเลือกไฟล์ข้อมูลเอ็กซ์เอ็มแอลแล้ว สามารถให้โปรแกรมคำนวณค่ามาตรวัดได้โดยเลือกเมนูคำนวณมาตรวัด (Calculate -> Calculate Metrics) จะแสดงหน้าจอให้ใส่ข้อมูลมาตรวัด ANCM ดังแสดงในรูปที่ ฉ-4 แล้วกดปุ่ม Ok จากนั้นค่ามาตรวัดที่คำนวณได้ทั้ง 17 มาตรวัดจะแสดงในตารางข้อมูล ดังแสดงในรูปที่ ฉ-5



รูปที่ ฉ-4 แสดงหน้าจอใส่ข้อมูลมาตรวัด ANCM

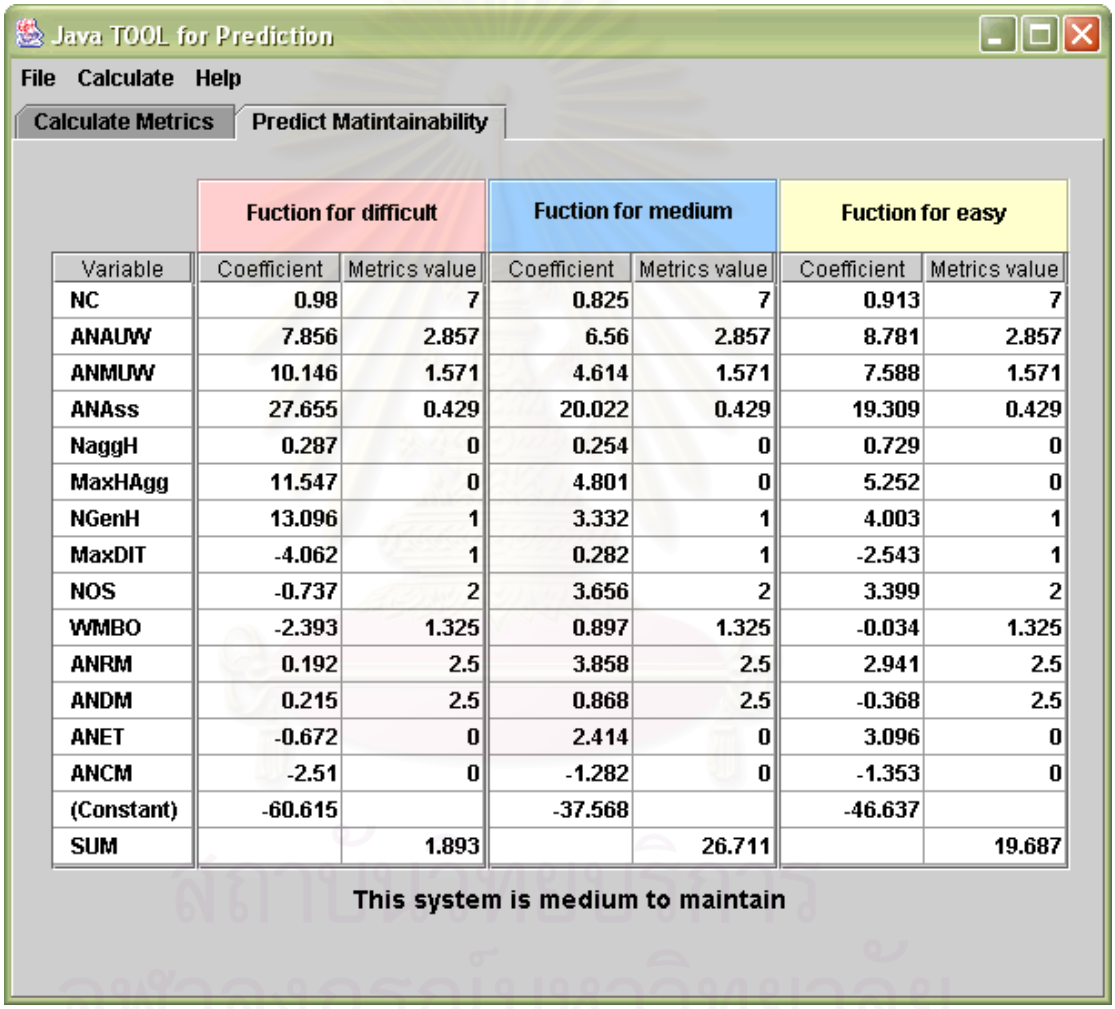


Metrics	Value
NC	7
ANAUW	2.857143
ANAW	0.21428572
ANMUW	1.5714285
ANMW	1.5714285
ANAss	0.42857143
ANAgg	0.0
NaggH	0
MaxHAgg	0
ANGen	0.2857143
NGenH	1
MaxDIT	1
NOS	2
WMBO	1.325
ANRM	2.5
ANDM	2.5
ANET	0.0
ANCM	0.0

รูปที่ ฉ-5 แสดงหน้าจอข้อมูลมาตรวัดที่คำนวณได้

2.3 การทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

เมื่อผู้ใช้งานต้องการให้โปรแกรมทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ให้เลือกเมนูทำนาย (Calculate -> Prediction Maintainability) ผลการทำนายจะแสดงดังรูปที่ ฉ-6 ซึ่งจะแสดงค่าที่ได้จากการคำนวณทั้ง 3 ฟังก์ชัน และบอกผลที่ได้จากการทำนายว่าระบบนี้มีระดับความสามารถในการบำรุงรักษาอยู่ที่ระดับใด หากผู้ใช้งานต้องการออกจากโปรแกรมให้เลือกเมนูออกจากโปรแกรม (File -> Exit)



รูปที่ ฉ-6 แสดงหน้าจอการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

ภาคผนวก ช.
ผลการทดลอง

1. สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 25 ระบบและชุดข้อมูลทดสอบจำนวน 15 ระบบ

ผู้วิจัยได้ทดลองสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้ระบบในกลุ่มชุดข้อมูลสอนจำนวน 25 ระบบ และชุดข้อมูลทดสอบจำนวน 15 ระบบ ซึ่งระบบเพิ่มเข้ามาเป็นชุดข้อมูลทดสอบ ได้แก่ ระบบที่ 3, 8, 9, 11, 14, 18, 20, 23, 29, และ 30

1.1 สร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

ผลการวิเคราะห์หาความสัมพันธ์ด้วยวิธีการวิเคราะห์ความสัมพันธ์แบบเพียร์สันพบว่ามีมาตรวัดจำนวน 7 คู่ที่มีค่าความสัมพันธ์กันสูง และเมื่อพิจารณาค่าอาร์สแควร์ที่ปรับแล้วที่ได้จากวิธีการวิเคราะห์การถดถอย พบว่ามีมาตรวัดจำนวน 4 มาตรวัดที่ถูกตัดออก คือ มาตรวัด ANAW, MaxDIT, ANMW และ ANGen ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์แต่ละระดับ คือ

$$F_{difficult} = -0.572 \times NC + 6.018 \times ANAUW + 21.388 \times ANMUW + 137.451 \times ANAsso - 11.76 \times ANAgg + 3.537 \times NaggH + 37.005 \times MaxHAgg + 31.638 \times NGenH - 11.225 \times NOS - 0.083 \times WMBO - 1.634 \times ANRM + 1.25 \times ANDM - 10.686 \times ANET - 16.753 \times ANCM - 106.538$$

$$F_{medium} = -0.098 \times NC + 7.771 \times ANAUW + 12.62 \times ANMUW + 94.197 \times ANAsso + 12.948 \times ANAgg + 1.406 \times NaggH + 18.906 \times MaxHAgg + 17.383 \times NGenH - 2.681 \times NOS + 0.761 \times WMBO + 3.767 \times ANRM + 2.867 \times ANDM - 4.675 \times ANET - 13.398 \times ANCM - 66.356$$

$$F_{easy} = -0.316 \times NC + 8.138 \times ANAUW + 16.187 \times ANMUW + 107.967 \times ANAsso + 19.686 \times ANAgg + 0.268 \times NaggH + 24.66 \times MaxHAgg + 19.546 \times NGenH - 4.998 \times NOS - 0.051 \times WMBO + 1.137 \times ANRM + 1.114 \times ANDM - 5.954 \times ANET - 13.372 \times ANCM - 75.811$$

เมื่อ $F_{difficult}$ คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับยาก

F_{medium} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับปานกลาง

F_{easy} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับง่าย

การทำนายด้วยวิธีครอสแวริเดชัน สามารถทำนายถูกต้องคิดเป็นค่าเฉลี่ย 48% (((0+12+0)/25)×100) และเมื่อนำระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 15 ระบบมาทำนายด้วยโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ พบว่ามีระบบจำนวน 8 ระบบที่สามารถทำนายอยู่ในระดับที่ถูกต้อง ดังแสดงในตารางที่ ช-2

ตารางที่ ช-1 แสดงเปอร์เซ็นต์ความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์เมื่อใช้ชุดข้อมูลสอนจำนวน 25 ระบบและชุดข้อมูลทดสอบจำนวน 15 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			รวม
			ยาก	ปานกลาง	ง่าย	
ครอสแวริเดชัน	จำนวน	ยาก	0	1	2	3
		ปานกลาง	2	12	5	19
		ง่าย	0	3	0	3
	เปอร์เซ็นต์	ยาก	0.0	33.3	66.7	100.0
		ปานกลาง	10.5	63.2	26.3	100.0
		ง่าย	0.0	100.0	0.0	100.0

ตารางที่ ช-2 แสดงผลการตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้เมื่อใช้ชุดข้อมูลสอนจำนวน 25 ระบบและชุดข้อมูลทดสอบจำนวน 15 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			รวม
			0	1	2	
กลุ่มที่ 1	จำนวน	ยาก	2	1	1	4
		ปานกลาง	2	4	1	7
		ง่าย	1	1	2	4
	เปอร์เซ็นต์	ยาก	50.0	25.0	25.0	100.0
		ปานกลาง	28.6	57.1	14.3	100.0
		ง่าย	25.0	25.0	50.0	100.0

2. สร้างโมเดลการทำนายโดยใช้ชุดข้อมูลสอนจำนวน 30 ระบบและชุดข้อมูลทดสอบจำนวน 10 ระบบ

ผู้วิจัยได้ทดลองสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์โดยใช้ระบบในกลุ่มชุดข้อมูลสอนจำนวน 30 ระบบ และชุดข้อมูลทดสอบจำนวน 10 ระบบ ซึ่งระบบเพิ่มเข้ามาเป็นชุดข้อมูลทดสอบ ได้แก่ ระบบที่ 3, 11, 14, 20 และ 29

2.1 สร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์

ผลการวิเคราะห์หาความสัมพันธ์ด้วยวิธีการวิเคราะห์ความสัมพันธ์แบบเพียร์สัน พบว่ามีมาตรวัดจำนวน 5 คู่ที่มีค่าความสัมพันธ์กันสูง และเมื่อพิจารณาค่าอาร์สแควร์ที่ปรับแล้วที่ได้จากวิธีการวิเคราะห์การถดถอย พบว่ามีมาตรวัดจำนวน 5 มาตรวัดที่ถูกตัดออก คือ มาตรวัด ANAW, ANMW, ANAgg ANGen และ MaxDIT ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์แต่ละระดับ คือ

$$F_{difficult} = 0.342 \times NC + 4.502 \times ANAUW + 14.89 \times ANMUW + 94.108 \times ANAsso + 1.836 \times NaggH + 21.544 \times MaxHAgg + 19.934 \times NGenH - 5.71 \times NOS - 1.675 \times WMBO + 1.942 \times ANRM + 2.012 \times ANDM - 5.328 \times ANET - 12.928 \times ANCM - 85.089$$

$$F_{medium} = 0.346 \times NC + 4.991 \times ANAUW + 9 \times ANMUW + 69.092 \times ANAsso + 1.631 \times NaggH + 12.321 \times MaxHAgg + 10.438 \times NGenH - 0.608 \times NOS + 1.443 \times WMBO + 4.373 \times ANRM + 1.933 \times ANDM - 1.199 \times ANET - 9.778 \times ANCM - 52.407$$

$$F_{easy} = 0.411 \times NC + 6.529 \times ANAUW + 10.509 \times ANMUW + 67.93 \times ANAsso + 1.513 \times NaggH + 13.249 \times MaxHAgg + 9.048 \times NGenH + 0.0087 \times NOS + 1.195 \times WMBO + 4.339 \times ANRM + 0.835 \times ANDM - 0.326 \times ANET - 9.45 \times ANCM - 60.387$$

เมื่อ $F_{difficult}$ คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับยาก

F_{medium} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับปานกลาง

F_{easy} คือ ฟังก์ชันทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ระดับง่าย

ตารางที่ ช-3 แสดงเปอร์เซ็นต์ความถูกต้องในการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์เมื่อใช้ชุดข้อมูลสอนจำนวน 30 ระบบและชุดข้อมูลทดสอบจำนวน 10 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			รวม
			ยาก	ปานกลาง	ง่าย	
ครอสแวริเดชัน	จำนวน	ยาก	2	2	0	4
		ปานกลาง	0	16	6	22
		ง่าย	0	4	0	4
	เปอร์เซ็นต์	ยาก	50.0	50.0	0.0	100.0
		ปานกลาง	0.0	72.7	27.3	100.0
		ง่าย	0.0	100.0	0.0	100.0

การทำนายด้วยวิธีครอสแวริเดชัน สามารถทำนายถูกต้องคิดเป็นค่าเฉลี่ย 60% (((2+16+0)/30)×100) และเมื่อนำระบบที่ใช้เป็นชุดข้อมูลทดสอบจำนวน 10 ระบบมาทำนายด้วยโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้ พบว่ามีระบบจำนวน 6 ระบบที่สามารถทำนายอยู่ในระดับที่ถูกต้อง ดังแสดงในตารางที่ ช-4

ตารางที่ ช-4 แสดงผลการตรวจสอบโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ที่ได้เมื่อใช้ชุดข้อมูลสอนจำนวน 30 ระบบและชุดข้อมูลทดสอบจำนวน 10 ระบบ

ความสามารถในการบำรุงรักษาซอฟต์แวร์			จำนวนข้อมูลในกลุ่มทำนาย			
			ยาก	ปานกลาง	ง่าย	รวม
กลุ่มที่ 2	จำนวน	ยาก	1	1	0	2
		ปานกลาง	1	3	1	5
		ง่าย	1	0	2	3
	เปอร์เซ็นต์	ยาก	50.0	50.0	0.0	100.0
		ปานกลาง	20.0	60.0	20.0	100.0
		ง่าย	33.3	0.0	66.7	100.0

ภาคผนวก ซ.
ผลงานตีพิมพ์

ระหว่างดำเนินงานวิจัย ได้เขียนบทความเพื่อตีพิมพ์ผลงานในวารสารวิชาการและการประชุมวิชาการทั้งในและต่างประเทศ ดังนี้

1. นางสาวนงเยาว์ จินดาสวัสดิ์, อาจารย์นครทิพย์ พร้อมพูล, อาจารย์เชษฐ พัฒโนทัย, ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี. “การวัดซอฟต์แวร์เชิงวัตถุ”. วารสารวิชาการเนคเทค ปีที่ 4 ฉบับที่ 13 เดือนมีนาคม – มิถุนายน 2546.

2. Matinee Kiewkanya, Nongyao Jindasawat, Nakornthip Prompoon, Pornsiri Muenchaisri. “Constructing Understandability Model from Design Metrics”. The 2003 International Conference on Software Engineering and Knowledge Engineering (SEKE’2003), San Francisco, California USA, July 1-3, 2003.

3. Nongyao Jindasawat, Matinee Kiewkanya, Pornsiri Muenchaisri. “Using Structural Complexity Design Metrics to Construct Modifiability Model”. The 7th National Computer Science and Engineering Conference 2003 (NCSEC 2003), Burapha University, Chonburi, Thailand, October 28-30, 2003.

การวัดซอฟต์แวร์เชิงวัตถุ

นางสาวนงเยาว์ จินดาสวัสดิ์[1], อาจารย์นครทิพย์ พร้อมพูล [2],
อาจารย์เชษฐ พัฒโนทัย [2], ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี[2]
นิสิตปริญญาโท[1] และอาจารย์[2] ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ABSTRACT – This paper presents the concept of object-oriented software measurement. Firstly, we start with an overview of object-oriented software measurement that consists of definition, types and benefits of software measurement. A collection of software metrics consisting of traditional metrics and object-oriented metrics are presented next. Then, we summarize research works using software metrics to measure software products, cost and effort, and software quality. Finally, an automated tool for measuring software metrics called "Measurement Tool for Object-Oriented Programs version 2 (MTOOP v.2)" is introduced.

KEY WORDS – Software Measurement, Metrics, Software Quality, Software Engineering, Object-Oriented Programming

บทคัดย่อ – บทความนี้นำเสนอแนวคิดที่เกี่ยวกับการวัดซอฟต์แวร์เชิงวัตถุ ในส่วนแรกเป็นการอธิบายความหมาย ประเภทของการวัดซอฟต์แวร์เชิงวัตถุ และประโยชน์ของการวัดซอฟต์แวร์ จากนั้นนำเสนอมาตรวัดที่ได้รวบรวมมาจากงานวิจัยต่างๆ ซึ่งได้แก่ มาตรวัดดั้งเดิม และมาตรวัดเชิงวัตถุสำหรับวัดคุณภาพของซอฟต์แวร์ และนำเสนอผลงานวิจัยที่มีการนำมาตรวัดซอฟต์แวร์ไปใช้งานในด้านการวัดผลิตผลของซอฟต์แวร์ การวัดค่าใช้จ่ายและกำลังคน และการวัดคุณภาพซอฟต์แวร์ พร้อมทั้งนำเสนอเครื่องมือสำหรับคำนวณมาตรวัดซอฟต์แวร์เชิงวัตถุที่มีชื่อว่า Measurement Tool for Object-Oriented Programs รุ่นที่ 2

คำสำคัญ – การวัดซอฟต์แวร์, มาตรวัด, คุณภาพซอฟต์แวร์, วิศวกรรมซอฟต์แวร์, การโปรแกรมเชิงวัตถุ

---

1. บทนำ

การวัดซอฟต์แวร์เป็นวิธีการประเมินคุณลักษณะและคุณภาพซอฟต์แวร์ โดยแสดงออกมาในรูปของค่าตัวเลขที่ใช้อธิบายความหมายของคุณลักษณะและคุณภาพของซอฟต์แวร์เหล่านั้น เนื่องจากการวัดไม่สามารถหาค่าจากซอฟต์แวร์โดยตรงได้ จึงต้องใช้มาตรวัด (Metric) ที่มีความสัมพันธ์กับคุณภาพที่กำหนดไว้ ดังนั้นกลุ่มงานวิจัย “มาตรวัดซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Metrics)” ซึ่งเป็นกลุ่มงานวิจัยร่วมระหว่างภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และตลาดหลักทรัพย์แห่งประเทศไทย (The Stock Exchange of Thailand - SET) ได้เล็งเห็นความสำคัญของการวัดซอฟต์แวร์ และได้ร่วมกันศึกษาการวัดคุณภาพในด้านความสามารถในการนำกลับมาใช้ใหม่ (Reusability), ความน่าเชื่อถือ (Reliability) และความสามารถในการบำรุงรักษา (Maintainability) โดยเน้นการวัดคุณภาพจากโมเดลการวิเคราะห์และออกแบบด้วยยูเอ็มแอล (Unified Modeling Language – UML) ซึ่งเป็นการวัดในช่วงการวิเคราะห์และออกแบบระบบ และเป็นขั้นตอนต้นๆ ของการพัฒนาซอฟต์แวร์ ทำให้ทราบถึงคุณลักษณะและคุณภาพของซอฟต์แวร์ได้เร็วขึ้น ดังนั้นผู้พัฒนาซอฟต์แวร์สามารถแก้ไขโมเดลการวิเคราะห์และออกแบบให้มีคุณภาพตามต้องการก่อนนำไปพัฒนาเป็นซอฟต์แวร์ต่อไป และได้พัฒนาเครื่องมือคำนวณมาตรวัดแบบอัตโนมัติเพื่อใช้งาน

สำหรับเนื้อหาในบทความนี้ ประกอบด้วย ส่วนแรกกล่าวถึงความหมาย ประเภทของการวัดซอฟต์แวร์ และประโยชน์ที่ได้รับ ในส่วนที่สองได้รวบรวมมาตรวัดซอฟต์แวร์ต่างๆ ที่มีนักวิจัยได้นำเสนอไว้ ส่วนที่สามกล่าวถึงผลงานวิจัยที่นำมาตรวัดซอฟต์แวร์ไปใช้งานจริง ส่วนที่สี่นำเสนอเครื่องมือคำนวณมาตรวัดซอฟต์แวร์เชิงวัตถุแบบอัตโนมัติ และบทสรุปเป็นส่วนสุดท้าย

2. การวัดซอฟต์แวร์เชิงวัตถุ

2.1 ความหมายของการวัดซอฟต์แวร์

การวัด คือ การกำหนดค่าตัวเลขหรือสัญลักษณ์ให้กับคุณลักษณะ (Attribute) ของสิ่งที่สนใจ (Entity) [17] ซึ่งการกำหนดค่าให้กับคุณลักษณะของสิ่งที่สนใจนั้น คือ การกำหนดค่าให้กับมาตรวัดของคุณลักษณะต่างๆ เหล่านั้นนั่นเอง คุณลักษณะของสิ่งที่สนใจสามารถแบ่งได้เป็น 2 ประเภท คือ คุณลักษณะภายใน (Internal Attribute) และคุณลักษณะภายนอก (External Attribute) ซึ่งคุณลักษณะภายในเป็นคุณลักษณะที่สามารถหาค่าได้โดยตรงจากสิ่งที่สนใจ แต่คุณลักษณะภายนอกเกิดจากการคำนวณจากคุณลักษณะภายในที่เกี่ยวข้องกับคุณลักษณะภายนอกนั้น ซึ่งการวัดซอฟต์แวร์สามารถจำแนกออกเป็น 3 ประเภท คือ

2.1.1 การวัดกระบวนการพัฒนาซอฟต์แวร์ (Processes measurement)

สิ่งที่นำมาใช้ในการวัดกระบวนการพัฒนาซอฟต์แวร์ จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับกระบวนการทำงานทั้งหมดของการพัฒนาซอฟต์แวร์ เช่น การสร้างรายละเอียดของโครงงาน (specification) , รายละเอียดของการออกแบบ หรือการทดสอบโปรแกรม (Testing) เป็นต้น ซึ่งในแต่ละขั้นตอนจะประกอบไปด้วยคุณลักษณะภายใน และคุณลักษณะภายนอก ที่แตกต่างกัน ตัวอย่างของคุณลักษณะภายในของการทดสอบโปรแกรม เช่น ค่าใช้จ่ายทั้งหมดที่ใช้ในการทดสอบโปรแกรม หรือจำนวนข้อผิดพลาดที่ค้นพบในขั้นตอนการทดสอบโปรแกรม เป็นต้น ส่วนคุณลักษณะภายนอกของการทดสอบโปรแกรม เช่น ค่าใช้จ่ายเฉลี่ยของการค้นพบข้อผิดพลาดในขั้นตอนการทดสอบโปรแกรม พบว่าการคำนวณค่าใช้จ่ายเฉลี่ยนั้น เกิดจากการนำค่าใช้จ่ายทั้งหมดที่ใช้ในการทดสอบโปรแกรมหารด้วยจำนวนข้อผิดพลาดทั้งหมดที่ค้นพบ ซึ่งทั้งสองค่าที่นำมาหารกัน เกิดจากการวัดคุณลักษณะภายในของการทดสอบโปรแกรม

2.1.2 การวัดผลผลิตของซอฟต์แวร์ (Software Products Measurement)

สิ่งที่ใช้ในการวัดผลผลิตของซอฟต์แวร์จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับผลผลิตของซอฟต์แวร์ที่ได้ในทุกขั้นตอนของการพัฒนาซอฟต์แวร์ เช่น รายละเอียดโครงงาน, การออกแบบ หรือโค้ดโปรแกรม เป็นต้น ตัวอย่างคุณลักษณะของโค้ดโปรแกรม เช่น การวัดขนาดของซอฟต์แวร์ที่ผลิตได้ซึ่งเป็นคุณลักษณะภายในสามารถวัดได้จากจำนวนบรรทัดทั้งหมดในโปรแกรม หรือความซับซ้อนของโปรแกรมซึ่งเป็นคุณลักษณะภายใน [17] เป็นต้น

2.1.3 การวัดทรัพยากร (Resources Measurement)

สิ่งที่สนใจนำมาวัดในประเภทนี้ จะเน้นเฉพาะส่วนที่เกี่ยวข้องกับทรัพยากรที่ต้องใช้ในการพัฒนาซอฟต์แวร์ เช่น คน, เครื่องมือที่ใช้ในการพัฒนาและฮาร์ดแวร์ เป็นต้น ตัวอย่างคุณลักษณะของฮาร์ดแวร์ เช่น จำนวนหน่วยความจำของเครื่องคอมพิวเตอร์ที่ใช้ในการพัฒนาซอฟต์แวร์ ซึ่งเป็นคุณลักษณะภายใน และความน่าเชื่อถือของเครื่องคอมพิวเตอร์ ที่สามารถทำงานอย่างต่อเนื่องได้โดยไม่ล้มเหลวระหว่างการทำงาน ซึ่งเป็นคุณลักษณะภายนอกของฮาร์ดแวร์

2.2 ประเภทของการวัดซอฟต์แวร์

การวัดซอฟต์แวร์ประเภทต่างๆ ที่ได้กล่าวไว้ในข้างต้น สามารถทำการวัดได้ 2 วิธี คือ การวัดทางตรง (Direct Measure) และการวัดทางอ้อม (Indirect Measure) ดังนี้

2.2.1 การวัดทางตรง

การวัดทางตรงเป็นวิธีการที่ใช้วัดคุณลักษณะภายใน (Internal Attribute) และสามารถวัดค่าได้จากสิ่งนั้นโดยตรงไม่ได้เกิดจากการคำนวณมาจากค่าข้อมูลอื่น เช่น ความยาวของโค้ดโปรแกรม, จำนวนข้อผิดพลาดที่พบในช่วงการทดสอบโปรแกรม เป็นต้น

2.2.2 การวัดทางอ้อม

การวัดทางอ้อมเป็นวิธีการที่ใช้วัดคุณลักษณะภายนอก ซึ่งเป็นการวัดในเชิงคุณภาพ เช่น การวัดความสามารถในการบำรุงรักษาซอฟต์แวร์ หรือความสามารถในการนำกลับมาใช้ใหม่ เป็นต้น แต่เนื่องจากการวัดคุณลักษณะภายนอกนั้นไม่สามารถหาค่าได้โดยตรง จึงต้องอาศัยการวัดทางตรงมาใช้ในการคำนวณหาค่า เช่น ความสามารถในการนำกลับมาใช้ใหม่ พิจารณาจำนวนความหนาแน่นของข้อผิดพลาดในโปรแกรม เป็นต้น

2.3 ประโยชน์ของการวัดซอฟต์แวร์

ประโยชน์ที่ได้รับจากการวัดซอฟต์แวร์ขึ้นอยู่กับมุมมองและสถานะในการทำงานของแต่ละคน เช่น ผู้พัฒนาซอฟต์แวร์ (Software Developers) ต้องการทราบว่าความต้องการของผู้ใช้งานมีความถูกต้องและสมบูรณ์หรือไม่, การออกแบบมีคุณภาพหรือไม่ และโค้ดโปรแกรมพร้อมที่จะนำไปทดสอบแล้วหรือไม่ เป็นต้น แต่ถ้าเป็นหัวหน้าโครงงาน (Project Managers) อาจต้องการทราบว่าซอฟต์แวร์พัฒนาเสร็จทันส่งลูกค้าหรือไม่ หรือใช้งบประมาณเกินที่กำหนดหรือไม่ ส่วนในมุมมองของลูกค้า (Customers) อาจนำวิธีการวัดมาตรวจสอบซอฟต์แวร์ที่รับมอบว่าตรงตามความต้องการที่ได้เสนอไปหรือไม่ เป็นต้น ประโยชน์ส่วนใหญ่ของการวัดซอฟต์แวร์นำไปใช้งานด้านการวัดคุณภาพซอฟต์แวร์ ซึ่งจะกล่าวถึงในส่วนที่สี่เรื่องการนำมาตรวัดไปใช้งาน

3. มาตรวัดซอฟต์แวร์

มาตรวัดซอฟต์แวร์ถูกกำหนดขึ้นมาเพื่อช่วยให้การวัดมีความง่ายขึ้น ในปัจจุบันได้มีนักวิจัยหลายท่านที่คิดค้นและนำเสนอมาตรวัดใหม่ๆ ขึ้นมา มาตรวัดที่เป็นที่นิยมใช้กันอย่างแพร่หลาย และเป็นที่ยอมรับต่างๆ เหล่านี้สามารถแบ่งได้เป็น 2 ประเภท คือ มาตรวัดแบบดั้งเดิม (Traditional Metrics) และมาตรวัดเชิงวัตถุ (Object-Oriented Metrics) รายละเอียดของมาตรวัดแต่ละประเภทมีดังนี้

3.1 มาตรวัดดั้งเดิม

มาตรวัดที่นิยมใช้ คือ มาตรวัดจำนวนบรรทัด และมาตรวัดความซับซ้อน ซึ่งเป็นมาตรวัดที่นำเสนอโดย McCabe [1][19] ใช้สำหรับวัดโปรแกรมที่พัฒนาขึ้น

3.1.1 มาตรวัดจำนวนบรรทัด (Lines of code – LOC)

การหาค่าจำนวนบรรทัดนั้น จะไม่นับรวมบรรทัดว่าง, บรรทัดที่เป็นเครื่องหมายบล็อก ({}) และบรรทัดที่เป็นข้อความอธิบาย (Comment) และการประกาศตัวแปรหลายตัวอยู่บนบรรทัดเดียวกันจะนับจำนวนบรรทัดเท่ากับจำนวนตัวแปรที่ประกาศ

3.1.2 มาตรวัดวัดไซโคลเมติกของแมคเคบ (Cyclomatic complexity – V(G))

เป็นการวัดค่าความซับซ้อนของการใช้คำสั่งควบคุมในโปรแกรม ได้แก่ คำสั่ง if, while, repeat และ for ถ้าโปรแกรมมีการใช้คำสั่งควบคุมเป็นจำนวนมาก จะทำให้ยากต่อคุณภาพในด้านความสามารถการทดสอบ (Testability), ความสามารถในการทำความเข้าใจ (Understandability) และความสามารถในการบำรุงรักษาซอฟต์แวร์

3.2 มาตรวัดเชิงวัตถุ

นักวิจัยหลายท่านได้นำเสนอมาตรวัด และศึกษาถึงความสัมพันธ์ระหว่างมาตรวัดและคุณภาพในด้านต่างๆ ซึ่งมาตรวัดที่จะกล่าวถึง ได้แก่ มาตรวัดของ Chidamber and Kemerer [13][18], มาตรวัด Lorenz and Kidd [12], มาตรวัด Brito e Abreu and Melo [12], มาตรวัด Marcela, Mario and Coral [12] และมาตรวัดสำหรับ Hyoseob Kim and Cornelia Boldreff [5] ดังนี้

3.2.1 มาตรวัดของ Chidamber and Kemerer

ในปี 1993 Chidamber and Kemerer [13][18] ได้นำเสนอมาตรวัดเชิงวัตถุ ซึ่งเป็นที่ยอมรับและนิยมใช้กันมากจนถึงปัจจุบัน คือมาตรวัดพื้นฐาน 6 มาตรวัด ดังรายละเอียดต่อไปนี้

1) มาตรวัดจำนวนเมทธอดต่อคลาส (Weighted methods per class – WMC)

ในขั้นตอนของการวิเคราะห์และออกแบบระบบเชิงวัตถุสามารถคำนวณค่ามาตรวัดจำนวนเมทธอดต่อคลาสได้ เพื่อนำค่ามาตรวัดนี้มาใช้ในการประมาณเวลา (Time) และความพยายาม (Effort) ที่ต้องใช้ในการพัฒนาซอฟต์แวร์ รวมถึงแก้ไข และบำรุงรักษาซอฟต์แวร์ได้ ค่ามาตรวัดจำนวนเมทธอดต่อคลาสได้จากผลรวมของการคำนวณค่าความซับซ้อนของมาตรวัดไซโคลเมติกของแมคเคบทุกเมทธอดภายในแต่ละคลาส

2) มาตรวัดความรับผิดชอบของคลาส (Response for a class – RFC)

ความรับผิดชอบของคลาส คือ จำนวนเมทธอดภายในคลาสที่สามารถตอบสนองการเรียกใช้งานจากคลาสอื่นได้ หรือจากเมทธอดบางตัวภายในคลาสนั้น มาตรวัดนี้แสดงค่าผลรวมของความซับซ้อนผ่านจำนวนเมทธอดและจำนวนการเรียกใช้งานจากคลาสอื่น ถ้ามีเมทธอดที่ถูกเรียกใช้งานได้เป็นจำนวนมากจะส่งผลให้คลาสมีความซับซ้อน ทั้งในด้านการทดสอบ (Testing) และด้านการแก้ไขข้อผิดพลาด (Debugging) ของซอฟต์แวร์มากขึ้นด้วย

3) มาตรวัดระดับความลึกของการสืบทอดคุณสมบัติของคลาส (Depth of inheritance hierarchy - DIT)

เป็นการหาค่าระดับความลึกของการสืบทอดคุณสมบัติ (Inheritance) ของแต่ละคลาส การสืบทอดคุณสมบัติที่มากขึ้นทำให้ความซับซ้อนของตัวโปรแกรมเพิ่มมากขึ้น และทำให้ยากต่อการทำนายพฤติกรรมของคลาส และการทำความเข้าใจระบบ ค่าระดับความลึกของการสืบทอดคุณสมบัติ ได้จากการนับจำนวนของระดับขั้น (Level) การสืบทอดคุณสมบัติของแต่ละคลาสที่กำลังพิจารณา ซึ่งคุณสมบัติของการสืบทอดคุณสมบัติเป็นลักษณะที่สำคัญต่อการวัดคุณภาพในด้านความสามารถนำกลับมาใช้ใหม่ (Reusability)

4) มาตรวัดจำนวนคลาสลูก (Number of children – NOC)

เป็นการนับจำนวนคลาสลูกทั้งหมดที่สืบทอดลงมาจากคลาสแม่ที่กำลังพิจารณาในขณะนั้น มาตรวัดจำนวนคลาสลูกเป็นคุณลักษณะที่สำคัญต่อการวัดคุณภาพในด้าน

ความสามารถในการนำกลับมาใช้ใหม่เช่นเดียวกับมาตรวัดระดับความลึกของการสืบทอดคุณสมบัติของคลาส

5) มาตรวัดการเข้าคู่กันระหว่างวัตถุ (Coupling between objects – CBO)

การเข้าคู่กัน (Coupling) สำหรับซอฟต์แวร์เชิงวัตถุ นิยมให้ค่าของการเข้าคู่กันมีค่าน้อยๆ เพื่อให้คลาสแยกเป็นอิสระจากกันมากที่สุด เพื่อความสะดวกในการแก้ไขข้อมูลจะได้ไม่ส่งผลกระทบต่อคลาสอื่น การเข้าคู่กันระหว่างคลาสเป็นการแสดงความสัมพันธ์ระหว่างวัตถุเมื่อมีการเรียกใช้ตัวแปรหรือเมทธอดระหว่างกัน ดังนั้นถ้าค่าของการเข้าคู่กันมีค่าสูงจะลดความเป็นโมดูลของคลาส ทำให้ความสามารถในการนำกลับมาใช้ใหม่ และการบำรุงรักษาทำได้ยากขึ้นตามมา

6) มาตรวัดระดับของการขาดความสัมพันธ์ภายในคลาส (Lack of cohesion of methods – LCOM)

การเกาะกันเป็นก้อน (Cohesion) ของคลาส คือ การที่จะบอกว่าเมทธอดของคลาสนั้นมีความสัมพันธ์กับเมทธอดอื่นๆ อย่างไร การที่มีค่าของการเกาะกันเป็นก้อนสูง แสดงว่าคลาสมีการออกแบบที่ดี การเกาะกันเป็นก้อนของคลาสจะเป็นการบอกแนวโน้มของการเอ็นแคบซูเลชัน (Encapsulation) ถ้าค่าของการขาดการเกาะกันเป็นก้อนมีค่าสูงจะทำให้แนวโน้มของการเอ็นแคบซูเลชันมีค่าต่ำ ซึ่งจะทำให้มีความซับซ้อนของคลาสสูง [22]

3.2.2 มาตรวัดของ Lorenz and Kidd

Lorenz และ Kidd [12] เสนอมาตรวัดที่เรียกว่า design metrics ซึ่งแบ่งเป็น 3 กลุ่ม คือ

มาตรวัดขนาดคลาส

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

2) มาตรวัด Number of Instance Methods in a Class (NIM) คือ จำนวนของเมทธอดที่ประกาศภายในคลาสนั้น ซึ่งจะนับเมทธอดที่มีการประกาศเป็นแบบ public, protected และ private

3) มาตรวัด Number of Instance Variables in a Class (NIV) คือจำนวนตัวแปรที่ประกาศภายในคลาสนั้น

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

5) มาตรวัด Number of class Variables in a Class (NVC)คือ จำนวนของตัวแปรประเภทคลาสที่ประกาศอยู่ในคลาสนั้น

มาตรวัดการสืบทอดของคลาส

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

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

3) มาตรวัด Number of Methods Added (NMA) คือ จำนวนเมทธอดที่สร้างขึ้นใหม่ภายในคลาสลูก

4) มาตรวัด Specialization Index (SIX) คือ มาตรวัดที่เกิดจากการคำนวณจำนวนของเมทธอดที่ถูก override คูณด้วยจำนวน Hierarchy หารด้วยจำนวนเมทธอดทั้งหมด

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

3.2.3 มาตรวัดของ Brito e Abreu and Melo

Brito e Abreu และ Melo [12] นำเสนอกลุ่มของมาตรวัด MOOD (Metrics for Object Oriented Design) เน้นมาตรวัดสำหรับกลไกการออกแบบเชิงวัตถุ ได้แก่ encapsulation, inheritance, polymorphism และ message passing ดังนี้

1) มาตรวัด Method Hiding Factor (MHF) คือ อัตราส่วนระหว่างผลรวมของเมทธอดที่ประกาศเป็น private กับจำนวนเมทธอดทั้งหมดในระบบ

2) มาตรวัด Attribute Hiding Factor (AHF) คือ อัตราส่วนระหว่างผลรวมของแอทริบิวต์ที่ประกาศเป็น private กับจำนวนแอทริบิวต์ทั้งหมดในระบบ

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

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

5) มาตรวัด Polymorphism Factor (PF) คือ อัตราส่วนระหว่างจำนวนการพ้องรูปที่เป็นไปได้ทั้งหมด กับจำนวนการพ้องรูปของคลาสใดๆ ที่มีค่ามากที่สุด

3.2.4 มาตรวัดของ Marcela, Mario and Coral

Marcela Genero, Mario Piattini และ Coral Calero [12] นำเสนอมาตรวัดเชิงวัตถุสำหรับวัดความซับซ้อนของแผนภาพคลาส โดยเน้นที่มาตรวัดความสัมพันธ์ ได้แก่ ความสัมพันธ์แบบ aggregation, association และ dependency

มาตรวัด Association

1) มาตรวัด Number of Associations of a Class (NAC) คือ มาตรวัด NAC คำนวณจากจำนวนความสัมพันธ์แบบ association ทั้งหมดของคลาสที่กำลังพิจารณาในขณะนั้น ซึ่งเป็นมาตรวัดที่ใช้วัดความซับซ้อนของคลาส

2) มาตรวัด Number of Association in Package (NAP) คือ มาตรวัด NAP คำนวณจากจำนวนความสัมพันธ์แบบ association รวมทั้งหมดภายในแพ็คเกจนั้น มาตรวัดนี้เป็นการวัดขนาดของแพ็คเกจ

3) มาตรวัด Number of Associations vs. Classes in a Package (NAVCP) คือ มาตรวัด NAVCP คำนวณจากอัตราส่วนระหว่างมาตรวัด NAP หารด้วยจำนวนคลาสทั้งหมดในแพ็คเกจนั้น

มาตรวัด Aggregation

1) มาตรวัด Height of Aggregation (HAgg) คือ เป็นการวัดความสูงของคลาสภายในระดับขั้นของความสัมพันธ์แบบ aggregation ซึ่งเป็นค่าของเส้นทางที่ยาวที่สุดจากคลาสนั้นไปถึงลีฟ

2) มาตรวัด Number of Direct Parts (NODP) คือ การนับจำนวน “Direct part” คลาส ที่รวมกันเป็น composite คลาส

3) มาตรวัด Number of Parts (NP) คือ จำนวนผลรวมของคลาสที่เป็น “part” คลาส ของคลาสนั้น

4) มาตรวัด Number of Wholes (NW) คือ จำนวนคลาสที่เป็น “whole” คลาส

5) มาตรวัด Multiple Aggregation (MAgg) คือ การแสดงค่าของจำนวน “whole” คลาสที่มีคลาสอยู่ในระดับ aggregation

6) มาตรวัด Number of Aggregation Relationships in package (NAggR) คือ จำนวนความสัมพันธ์แบบ aggregation ภายในแพ็คเกจ

มาตรวัด Dependency

1) มาตรวัด Number of Dependencies In (NDepIn) คือ จำนวนของคลาสที่ขึ้นอยู่กับคลาสอื่น

2) มาตรวัด Number of Dependencies Out (NDepOut) คือ จำนวนของคลาสอื่นที่ขึ้นอยู่กับคลาสนั้น

3.2.5 มาตรวัดของ Hyoseob Kim and Cornelia Boldyreff

Hyoseob Kim and Cornelia Boldyreff [5] ได้รวบรวมมาตรวัดสำหรับวัดยูเอ็มแอลไว้ สามารถจำแนกประเภทของมาตรวัดออกเป็น มาตรวัดสำหรับโมเดล, มาตรวัดสำหรับคลาส, มาตรวัดสำหรับความสัมพันธ์, มาตรวัดสำหรับ Message และมาตรวัดสำหรับยูสเคส ดังนี้

มาตรวัดสำหรับโมเดล (Metrics for Model)

1) มาตรวัด Number of the packages in a model (NPM) คือ จำนวนแพ็คเกจในโมเดล

2) มาตรวัด Number of the classes in a model (NCM) คือ จำนวนคลาสในโมเดล

3) มาตรวัด Number of actors in a model (NAM) คือ จำนวนผู้ดำเนินการ (Actor) ในโมเดล

4) มาตรวัด Number of the use cases in a model (NUM) คือ จำนวนยูสเคสในโมเดล

5) มาตรวัด Number of the objects in a model (NOM) คือ จำนวนวัตถุ (Object) ในโมเดล

6) มาตรวัด Number of the messages in a model (NMM) คือ จำนวนสาร (Message) ในโมเดล

7) มาตรวัด Number of the associations in a model (NASM) คือ จำนวนความสัมพันธ์แบบ Association ในโมเดล

8) มาตรวัด Number of the aggregations in a model (NAGM) คือ จำนวนความสัมพันธ์แบบ Aggregation ในโมเดล

9) มาตรวัด Number of the inheritance relations in a model (NIM) คือ จำนวนความสัมพันธ์แบบ Inheritance ในโมเดล

มาตรวัดสำหรับคลาส (Metrics for Class)

1) มาตรวัด Number of the attributes in a class – unweighted (NATC1) คือ จำนวนคุณลักษณะภายในคลาสที่กำลังพิจารณา โดยไม่มีการถ่วงค่าให้กับ modifier ของคุณลักษณะที่เป็น Public, Protected และ Private

2) มาตรวัด Number of the attributes in a class – weighted (NATC2) คือ จำนวนคุณลักษณะภายในคลาสที่กำลังพิจารณา โดยมีการถ่วงค่าให้กับ modifier ของคุณลักษณะที่เป็น Public เท่ากับ 1.0, Protected เท่ากับ 0.5 และ Private เท่ากับ 0.0

3) มาตรวัด Number of the operations in a class – unweighted (NOPC1) คือ จำนวนเมทธอดในคลาสที่กำลังพิจารณา โดยไม่มีการถ่วงค่าให้กับ modifier ของเมทธอดที่เป็น Public, Protected และ Private

4) มาตรวัด Number of the operations in a class – weighted (NOPC2) คือ จำนวนเมทธอดภายในคลาสที่กำลังพิจารณา โดยมีการถ่วงน้ำหนักให้กับค่า modifier ของเมทธอดที่เป็น Public เท่ากับ 1.0, Protected เท่ากับ 0.5 และ Private เท่ากับ 0.0

5) มาตรวัด Number of the associations linked to a class (NASC) คือ จำนวนความสัมพันธ์แบบ Association ของคลาสที่กำลังพิจารณา โดยนับความสัมพันธ์แบบ Aggregation รวมด้วย

6) มาตรวัด Coupling between classes (CBC) คือ จำนวนความสัมพันธ์ (Association) ภายในคลาสที่กำลังพิจารณา

7) มาตรวัด Number of the superclasses of a class (NSUPC) คือ จำนวนคลาสบรรพบุรุษ (Parent class) แบบโดยตรง (Direct) ของคลาสที่กำลังพิจารณา

8) มาตรวัด Number of the elements in the transitive closure of the superclasses of a class (NSUPC*) คือ จำนวน Transitive closure ของคลาสบรรพบุรุษของคลาสที่กำลังพิจารณา

9) มาตรวัด Number of the subclasses of a class (NSUBC) คือ จำนวนคลาสลูกหลาน (Child class) แบบโดยตรงของคลาสที่กำลังพิจารณา

10) มาตรวัด Number of the elements in the transitive closure of the subclasses of a class NSUBC*) คือ จำนวน Transitive closure ของคลาสลูกหลานของคลาสที่กำลังพิจารณา

11) มาตรวัด Number of messages sent by the instantiated objects of a class (NMSC) คือ จำนวนสารที่ส่งออกไปโดยวัตถุ (Object) ที่ถูกสร้าง (Instantiate) จากคลาสที่กำลังพิจารณา

12) มาตรวัด Number of messages received by the instantiated objects of a class (NMRC) คือ จำนวนสารที่รับเข้ามาโดยวัตถุที่ถูกสร้างจากคลาสที่กำลังพิจารณา

มาตรวัดสำหรับสาร (Metrics for Message)

1) มาตรวัด Number of the directly dispatched messages of a message (NDM) คือ จำนวนสาร (Message) ที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่น

2) มาตรวัด Number of the elements in the transitive closure of the directly dispatched messages of a message (NDM*) คือ จำนวนอิลิเมนต์ใน Transitive closure ของสารที่กำลังพิจารณาส่งไปกระตุ้นสารตัวอื่น

มาตรวัดสำหรับความสัมพันธ์ (Metrics for Relationship)

- ความสัมพันธ์แบบ Association

1) มาตรวัด Number of Associations of a Class (NAC) คือ จำนวนผลรวมของความสัมพันธ์แบบ Associations ของคลาสในแผนภาพคลาส

2) มาตรวัด Number of Associations in a Package (NAP) คือ จำนวนผลรวมของความสัมพันธ์แบบ Associations ภายในแพ็คเกจ

3) มาตรวัด Number of Associations vs. Classes in a Package (NAVCP) คือ อัตราส่วนระหว่างจำนวนความสัมพันธ์แบบ Associations ในแพ็คเกจหารด้วยจำนวนคลาสภายในแพ็คเกจ

- ความสัมพันธ์แบบ Aggregation

1) มาตรวัด Height of Aggregation (HAgg) คือ ทางเดิน (Path) ที่ยาวที่สุดจากคลาสไปยังลีฟ (leaves)

2) มาตรวัด Number of Direct Parts (NODP) คือ จำนวนผลรวมของคลาส “direct path” ซึ่งประกอบด้วย composite class

3) มาตรวัด Number of Parts (NP) คือ จำนวนของคลาสลูกหลาน (descendant) ของคลาสนั้น

4) มาตรวัด Number of Wholes (NW) คือ จำนวนของคลาสบรรพบุรุษ (predecessor) ของคลาสนั้น

5) มาตรวัด Multiple Aggregation (MAgg) คือ จำนวนของคลาส “whole” ที่มีคลาสที่พิจารณาอยู่เป็นส่วนประกอบ

6) มาตรวัด Number of Aggregation Relationships (NAggR) คือ จำนวนความสัมพันธ์แบบ Aggregation ภายในแพ็คเกจ

- ความสัมพันธ์แบบ Dependency

1) มาตรวัด Number of Dependencies In (NDepIn) คือ จำนวนของคลาสที่ขึ้นอยู่กับคลาสที่กำลังพิจารณา

2) มาตรวัด Number of Dependencies Out (NDepOut) คือ จำนวนของคลาสซึ่งคลาสที่กำลังพิจารณาไปขึ้นอยู่กับคลาสเหล่านั้น

3) มาตรวัด Number of Dependencies In (NDepIn) มีความหมายเหมือนกับมาตรวัด (NDepIn)

4) มาตรวัด Number of Dependencies Out (NDepOut) มีความหมายเหมือนกับมาตรวัด (NDepOut)

มาตรวัดสำหรับยูสเคส (Metrics for Use Case)

1) มาตรวัด Number of actors associated with a use case (NAU) คือ จำนวนผู้ดำเนินการที่สัมพันธ์กับยูสเคสที่กำลังพิจารณา

2) มาตรวัด Number of message associated with a use case (NMU) คือ จำนวนสารที่สัมพันธ์กับยูสเคสที่กำลังพิจารณา

3) มาตรวัด Number of system classes associated with a use case (NSCU) คือ จำนวนคลาสที่เกี่ยวข้องกับยูสเคสที่กำลังพิจารณา

4. การนำมาตรวัดซอฟต์แวร์ไปใช้งาน

ปัจจุบันมีการนำมาตรวัดซอฟต์แวร์ไปใช้งานในด้านการวัดผลิตผลของซอฟต์แวร์, การประมาณค่าใช้จ่าย (Cost) และกำลังคน (Effort) หรืออธิบายคุณภาพของซอฟต์แวร์ในด้านความสามารถในการนำกลับมาใช้ใหม่, ความน่าเชื่อถือ และความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งเป็นคุณลักษณะที่สำคัญของการพัฒนาซอฟต์แวร์เชิงวัตถุ

4.1 การวัดผลิตผลของซอฟต์แวร์ (Software Productivity measurement)

ในการประเมินผลิตผลของซอฟต์แวร์ที่สามารถผลิตได้นิยมใช้มาตรวัดจำนวนบรรทัด หรือ Lines of code เช่น ต้องการทราบว่านักพัฒนาซอฟต์แวร์ (Programmer) แต่ละคนสามารถพัฒนาโปรแกรมได้เท่าใด จะทำการนับจำนวนบรรทัดโปรแกรมที่นักพัฒนาโปรแกรมผู้นั้นสามารถเขียนได้ หรือใช้ฟังก์ชันพอยต์ (Function point) สำหรับวัดผลิตผลที่นักพัฒนาโปรแกรมสามารถผลิตได้ เป็นต้น

4.2 การวัดค่าใช้จ่าย และกำลังคน (Cost and effort measurement)

Boehm [3] ได้พัฒนาโมเดล COCOMO (Constructive Cost Model) เพื่อใช้สำหรับการประมาณค่าใช้จ่าย และกำลังคนที่ต้องใช้ในกระบวนการพัฒนาซอฟต์แวร์ ซึ่งโมเดลนี้จะใช้มาตรวัดขนาด (Size) คือ มาตรวัด Source lines of code (SLOC) ต่อมาได้พัฒนาโมเดล COCOMO II ซึ่งเป็นโมเดลสำหรับซอฟต์แวร์เชิงวัตถุ โดยใช้มาตรวัดขนาดที่เรียกว่า มาตรวัด Object point แทนมาตรวัดตัวเดิม

4.3 การวัดคุณภาพซอฟต์แวร์ (Software Quality Measurement)

การวัดคุณภาพซอฟต์แวร์ทำเพื่อให้ทราบถึงความสามารถของซอฟต์แวร์ในด้านต่างๆ แต่เนื่องจากคุณภาพแต่ละตัวนั้นประกอบไปด้วยคุณลักษณะหลายตัว ดังนั้นจึงได้มีนักวิจัย คิดโมเดลสำหรับวัดคุณภาพซอฟต์แวร์ขึ้นมา ซึ่งโมเดลที่นิยมใช้ คือ โมเดลคุณภาพของ McCall และ

Boehm ซึ่งจะแสดงให้เห็นว่า คุณภาพคือปัจจัย (Factor) ที่เราต้องการวัดจะประกอบไปด้วยเกณฑ์ (Criteria) หลายแบบซึ่งแต่ละแบบสามารถหาค่าได้จากมาตรวัด เช่น ปัจจัยของคุณภาพในด้านความน่าเชื่อถือของโมเดล McCall ได้กำหนดเกณฑ์ไว้ 4 ตัวด้วยกัน คือ Accuracy, Error tolerance, Consistency และ Simplicity เป็นต้น มีผลงานวิจัยมากมายที่เสนอแนะวิธีการวัดคุณภาพซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบ และโปรแกรม และวิธีการนำการวัดซอฟต์แวร์ไปใช้งาน ได้แก่

4.3.1 ผลงานวิจัยของ *Lionel C. Briand, John Daly และ Jurgen Wust* [8]

ผลงานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านความน่าเชื่อถือ ซึ่งได้ทำการทดลองแสดงความสัมพันธ์ระหว่างมาตรวัดซอฟต์แวร์ในขั้นตอนของการออกแบบกับความน่าจะเป็นในการพบข้อผิดพลาดในคลาส โดยมาตรวัดซอฟต์แวร์ที่นำมาพิจารณาประกอบด้วยมาตรวัดการเข้าคู่ (coupling) การยึดเหนี่ยว (cohesion) และการถ่ายทอดคุณลักษณะ (inheritance) และสร้างโมเดลที่ช่วยในการทำนายว่าแต่ละคลาสมีความน่าจะเป็นของการเกิดข้อผิดพลาดได้มากหรือน้อยเพียงใด

Measure	Coeff.	Std. Error	p
RFC	0.242	0.042	<.0001
DAC	-1.110	0.429	0.0098
OCAEC	-0.985	0.373	0.0083
FMMEC	0.449	0.148	0.0025
RFC1_L	0.348	0.071	<.0001
NIH-ICP_L	-0.092	0.023	<.0001
ACMIC_L	1.060	0.556	0.0568
NOC	-2.645	0.789	0.0008
NOP	4.377	0.966	<.0001
NMI	-0.429	0.084	<.0001
SIX	-14.073	5.241	0.0072

รูปที่ 1 แสดงสัมประสิทธิ์ของมาตรวัดแต่ละตัวในโมเดลการทำนาย [8]

ผลการทดลองแสดงมาตรวัด 11 มาตรวัดที่มีความสัมพันธ์ต่อการทำนายความผิดพลาดของคลาส และสัมประสิทธิ์ของมาตรวัดแต่ละตัวแสดงในรูปที่ 1 ผลงานวิจัยนี้นำไปใช้วัดความน่าเชื่อถือของคลาสแต่ละคลาสภายในแผนภาพคลาส ในขั้นตอนการวิเคราะห์และออกแบบระบบ โดยทำการคำนวณค่ามาตรวัดทั้ง 11 มาตรวัดของแต่ละคลาส พร้อมทั้งคำนวณค่าจากผลคูณของมาตรวัดแต่ละมาตรวัดกับสัมประสิทธิ์ตามโมเดลการทำนาย ซึ่งค่าที่คำนวณได้เป็นค่าที่บอกถึงความน่าจะเป็นของการเกิดข้อผิดพลาดแต่ละคลาสนั้น ซึ่งจะช่วยให้นักออกแบบระบบสามารถนำแผนภาพคลาสมาแก้ไข ก่อนนำไปพัฒนาในขั้นตอนต่อไปได้

4.3.2 ผลงานวิจัยของ *Khaled El Emam, Walcelio Melo และ Javam Machado* [6]

ผลงานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านความน่าเชื่อถือ และทำการตรวจสอบ (Validation) มาตรวัดการออกแบบเชิงวัตถุ (Object-Oriented Design Metrics) ที่สัมพันธ์กับแนวโน้มการเกิดข้อผิดพลาด (Fault-proneness) ของระบบที่พัฒนาด้วยภาษาจาวา ผลการวิจัยพบว่าจากมาตรวัดระดับความลึกของการสืบทอดคุณสมบัติ (DIT) และมาตรวัดจำนวนคลาสลูก (NOC) ของ Chidamber and Kermerer และมาตรวัดการเข้าคู่จำนวน 8 มาตรวัด มีเพียง 3 มาตรวัดเท่านั้นที่มีผลต่อแนวโน้มการเกิดข้อผิดพลาด คือ มาตรวัดขนาด (Size), มาตรวัด OCMEC ซึ่งเป็นมาตรวัดการเข้าคู่ และมาตรวัด DIT

ผลงานวิจัยนี้ เป็นพื้นฐานของการวัดคุณภาพซอฟต์แวร์เชิงวัตถุในด้านความน่าเชื่อถือ และสามารถนำมาตรวัดทั้ง 3 มาตรวัดไปคำนวณค่าจากคลาสแต่ละคลาสภายในโปรแกรม เพื่อหาแนวโน้มของการเกิดข้อผิดพลาดภายในคลาสได้

4.3.3 ผลงานวิจัยของ *Marcela Genero and Mario Piattini* [9][10][11]

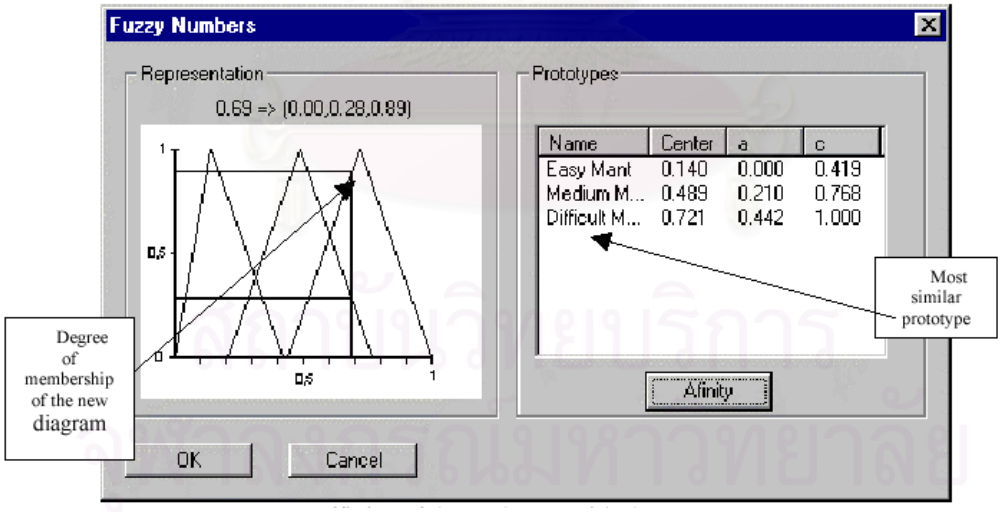
ผลงานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านการบำรุงรักษาซอฟต์แวร์โดยได้ทำการทดลอง (Empirical validation) เพื่อหาความสัมพันธ์ระหว่างความซับซ้อนของโครงสร้างแผนภาพคลาส (class diagram structural complexity) กับความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยพิจารณาคุณลักษณะความสามารถในการทำความเข้าใจ,

ความสามารถในการวิเคราะห์ และความสามารถในการปรับเปลี่ยน และสร้างโมเดลการทำนาย (Prediction model) ความสามารถในการบำรุงรักษาซอฟต์แวร์จากแผนภาพคลาส โดยให้หน่วยตัวอย่างกำหนดระดับความยากง่ายของแผนภาพคลาส ออกเป็น 7 ระดับตั้งแต่ง่ายที่สุดไปจนถึงยากที่สุด ด้วยตัวเลขตั้งแต่ 1 ถึง 7

ผลจากการทดลองได้นำข้อมูลที่ได้จากการแบ่งระดับโดยหน่วยตัวอย่างมาปรับให้เป็น 3 ระดับ คือ ระดับง่าย, ปานกลาง และยาก ดังแสดงในรูปที่ 2 และสร้างโมเดลการทำนายโดยใช้วิธีการ Fuzzy Prototypical Knowledge Discovery (FPKD)

	Understandability	**Analisability**	**Modifiability**
Difficult			
Average	6	6	6
Maximum	6	6	7
Minimum	6	5	6
Medium			
Average	5	5	5
Maximum	5	6	5
Minimum	4	4	4
Easy			
Average	2	2	3
Maximum	3	3	3
Minimum	2	2	2

รูปที่ 2 แสดงตารางต้นแบบของการแบ่งระดับความสามารถในการบำรุงรักษาระบบ [9]



รูปที่ 3 แสดงหน้าจอการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ [9]

รูปที่ 3 แสดงหน้าจอการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ จากผลการทดลองมีมาตรวัดที่สัมพันธ์ต่อการทำนายระดับความสามารถในการบำรุงรักษาซอฟต์แวร์อยู่ 11 มาตรวัด คือ Number of Classes (NC), Number of Attributes (NA), Number of Methods (NM), Number of Associations (NAssoc), Number of Aggregation (NAgg), Number of Dependencies (NDep), Number of Generalizations

(NGen), Number of Aggregation Hierarchies (NAggH), Number of Generalizations Hierarchies (NgenH), Maximum DIT และ Maximum HAgg

การวัดระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ของงานวิจัยนี้ เป็นการเพิ่มข้อมูลการตัดสินใจในการปรับปรุงแผนภาพคลาสให้กับผู้พัฒนาซอฟต์แวร์ เพื่อให้พัฒนาซอฟต์แวร์ที่มีคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์เพิ่มมากขึ้น

4.3.4 ผลงานวิจัยของ *Lionel C. Briand, Christian Bunse and John W. Daly* [7]

ผลงานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านการบำรุงรักษาซอฟต์แวร์ ซึ่งทำการทดลองเพื่อศึกษาผลกระทบเมื่อนำหลักการออกแบบเชิงคุณภาพมาประยุกต์ใช้ ซึ่งพิจารณาถึงความสามารถในการบำรุงรักษาของการออกแบบเชิงวัตถุ โดยพิจารณาคุณลักษณะย่อยที่ประกอบด้วย ความสามารถในการทำความเข้าใจ และความสามารถในการปรับเปลี่ยน เพื่อทำการเปรียบเทียบผลกระทบของหลักการออกแบบในมุมมองของ “การออกแบบเชิงวัตถุที่ดี (good design)” และ “การออกแบบเชิงวัตถุที่ไม่ดี (bad - design)” ตามหลักการออกแบบ (design principle) ของ Coad and Yourdon มาตรวัดที่นำมาใช้มีอยู่ด้วยกัน 6 มาตรวัด ได้แก่ มาตรวัด Und_Time, มาตรวัด Und_Corr, มาตรวัด Mod_Time, มาตรวัด Mod_Comp, มาตรวัด Mod_Corr และมาตรวัด Mod_Rate ผลการทดลองพบว่าการออกแบบตามหลักการออกแบบ ของ Coad และ Yourdon มีความสามารถในการเข้าใจระบบได้ดีกว่าระบบที่ไม่ได้ออกแบบตามหลักการของ Coad และ Yourdon

ยังมีผลงานวิจัยอีกหลายงานที่นำมาตรวัดซอฟต์แวร์ไปใช้ในการวัดคุณภาพในด้านความสามารถในการทดสอบ (Testability), ความสามารถในการนำไปใช้งาน (Usability) และความถูกต้อง (Correctness) เป็นต้น

4.3.5 ผลงานของกลุ่มงานวิจัย “มาตรวัดซอฟต์แวร์เชิงวัตถุ (Object-Oriented Software Metrics)” [14][15][16]

ขณะนี้กลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุ ของทางภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย มุ่งเน้นวัดคุณภาพในด้านความสามารถในการนำกลับมาใช้ใหม่, ความน่าเชื่อถือ และความสามารถในการบำรุงรักษาซอฟต์แวร์ และกำลังดำเนินการศึกษาคุณภาพในด้านอื่นๆ ผลงานวิจัยของกลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุ ได้แก่

4.3.5.1 ผลงานวิจัยของ *Matinee Kiewkanya และ Pornsiri Muenchaisri* [15]

งานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านความสามารถในการนำกลับมาใช้ใหม่ภายใน (Internal Reuse) โดยพิจารณาจากจำนวนครั้งของการเรียกใช้งานซ้ำของเมทธอดที่สร้างขึ้นสำหรับใช้งานภายในซอฟต์แวร์ชิ้นหนึ่งๆ การวัดการนำกลับมาใช้ภายในคลาสใดๆ จะเกิดจากผลรวมของการนำกลับมาใช้ใหม่ภายในของทุกๆ เมทธอดของคลาสนั้นๆ และการนำกลับมาใช้ใหม่ภายในของซอฟต์แวร์จะเกิดจากผลรวมของการนำกลับมาใช้ใหม่ภายในของทุกๆ คลาสในซอฟต์แวร์ งานวิจัยนี้ได้แบ่งกลุ่มเมทธอดออกเป็น 2 กลุ่ม ได้แก่

- กลุ่ม Private คือ เมทธอดที่มีตัวขยาย (modifier) เป็น Private เมทธอดในกลุ่มนี้จะมีการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมทธอด เฉพาะในส่วนของการนำกลับมาใช้ใหม่ภายในที่ไม่ผ่านกลไกการสืบทอดคุณสมบัติ
- กลุ่ม Public คือเมทธอดที่มีตัวขยาย เป็น Public และ Protected เมทธอดในกลุ่มนี้จะมีการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมทธอด ทั้งในส่วนของการนำกลับมาใช้ใหม่ภายในที่ผ่านกลไกการสืบทอดคุณสมบัติ และการนำกลับมาใช้ใหม่ภายในที่ไม่ผ่านกลไกการสืบทอดคุณสมบัติ

มาตรวัดการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมทธอด M ของคลาส X ใดๆ สามารถแสดงได้ดังรูปที่ 4 โดย

NOI คือ จำนวนครั้งที่เมทธอดอื่นถูกเรียกใช้ทั้งแบบโดยตรงและโดยอ้อมจากเมทธอด M

a คือ จำนวนครั้งที่เมทธอด M ถูกเรียกใช้โดยไม่ผ่านกลไกของการสืบทอดคุณสมบัติ

n คือ จำนวนของ descendant class ของคลาส X ที่ไม่ redefine เมทธอด M

Method Type	Internal reuse involves method M of class X	
	Static View	Dynamic View
Private	*NOI*	$a(NOI+1)-1$
Non -private	$(n+1)(NOI+1)-$	$\left(a+\sum_{i=1}^{n} b_i\right)(NOI+1)-1$

รูปที่ 4 แสดงมาตรวัดการนำกลับมาใช้ใหม่ภายใน

Type of metrics	Object-oriented design metrics
Size measurement metrics	WMC, NOATPB, NOATPT
Coupling measurement metrics	NMSC, NMRC, NOA, NOHA, NOTA, NOHC,NOTC,NOHG, NOTG
Inheritance measurement metrics	DIT, NOC
Cohesion measurement metric	LCOM

รูปที่ 5 แสดงมาตรวัดการออกแบบเชิงวัตถุ

b_i คือ จำนวนครั้งที่เมทธอด M ถูกเรียกใช้ผ่านกลไกของการสืบทอดคุณสมบัติ จาก instance ของ descendant class ของคลาส X ที่ไม่ redefine เมทธอด M

ผลการวิจัยได้นำเสนอมาตรวัดใหม่สำหรับการวัดการนำกลับมาใช้ใหม่ ดังรูปที่ 4 พร้อมทั้งกำหนดคำนิยาม (Definition) และข้อสมมติ (assumption) ตามทฤษฎีการวัดซอฟต์แวร์ (software measurement theory) เพื่อใช้วัดการนำกลับมาใช้ใหม่ภายในของซอฟต์แวร์ โดยวัดคุณภาพซอฟต์แวร์จากแผนภาพคลาสและแผนภาพซีเควนซ์ ผลที่ได้จากการคำนวณค่ามาตรวัดแต่ละตัวสามารถนำไปใช้ประกอบการจัดการโครงงาน (Project management) ในเรื่องของค่าใช้จ่าย, เวลา และกำลังคนที่ต้องใช้ในการพัฒนาซอฟต์แวร์

4.3.5.2 ผลงานวิจัยของ Matupayas Thongmak and Pornsiri Muenchaisri [16]

งานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านความน่าเชื่อถือ โดยพิจารณามาตรวัดที่วัดได้จากแผนภาพคลาสและแผนภาพซีเควนซ์ในขั้นตอนการออกแบบระบบ มาตรวัดที่นำมาพิจารณามีทั้งหมด 15 มาตรวัด ได้แก่ มาตรวัดขนาด 3 มาตรวัด, มาตรวัดการเข้าคู่ 9 มาตรวัด, มาตรวัดการสืบทอดคุณสมบัติ 2 มาตรวัดและมาตรวัด Cohesion 1 มาตรวัด ดังรูปที่ 5 และสร้างโมเดลการทำนายความผิดพลาดของคลาส

ผลการทดลองได้โมเดลการทำนายสำหรับจำแนกกลุ่มของคลาสที่มีข้อผิดพลาด และกลุ่มของคลาสที่ไม่มีข้อผิดพลาด ดังสมการ

สมการสำหรับคลาสที่ไม่มีข้อผิดพลาด

$D = -3.348 - 1.594WMC + 2.914*10^{-8}LCOM + 4.151DIT - 0.159NOC - 6.678*10^{-2}NOA - 1.361NOHA + 1.863NOTA + 3.645NOHC + 2.022NOTC + 1.627NOHG - 2.294NMSC + 1.965NMRC + 0.204NOATPB$

สมการสำหรับคลาสที่มีข้อผิดพลาด

$D = -3.942 + 0.16WMC + 1.869*10^{-8}LCOM + 4.828DIT - 0.23NOC + 0.954NOA - 3.674*10^{-2}NOHA + 2.746NOTA - 0.216NOHC + 1.483NOTC + 0.783NOHG - 3.141NMSC + 1.057NMRC + 0.569NOATPB$

จากทั้ง 2 สมการเมื่อแทนค่ามาตรวัดลงในสมการ ถ้าค่าในสมการแรกมีค่ามากกว่าค่าในสมการที่สองแสดงว่าคลาสที่นำมาพิจารณานั้นไม่มีข้อผิดพลาดเกิดขึ้น แต่ถ้าค่าในสมการแรกมีค่าน้อยกว่าค่าในสมการที่สองแสดงว่าคลาสที่นำมาพิจารณานั้นมีข้อผิดพลาดเกิดขึ้น

ทั้งนี้เพื่อความสะดวกในการวัดคุณภาพความน่าเชื่อถือและการคำนวณค่ามาตรวัดต่างๆ ของงานวิจัยนี้ ขณะนี้กลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุกำลังพัฒนาเครื่องมือเพื่อช่วยในการวัดค่าระดับความผิดพลาดของคลาส

4.3.5.3 ผลงานวิจัยของ Matinee Kiewkanya, Nongyao Jindasawat, Nakornthip Prompoon and Pornsiri Muenchaisri [14]

งานวิจัยนี้วัดคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งพิจารณาตามความสามารถในการทำความเข้าใจได้ และความสามารถในการปรับเปลี่ยน และพิจารณามาตรวัดสำหรับแผนภาพคลาสและแผนภาพซีเควนซ์ในขั้นตอนการออกแบบระบบจำนวน 18 มาตรวัดสำหรับสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ โดยใช้วิธีการ Discriminant Analysis

งานวิจัยนี้อยู่ในระหว่างการดำเนินการผลของคุณภาพด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ แต่ได้ทำการทดลองหาความสัมพันธ์ระหว่างมาตรวัดและคุณภาพด้านความสามารถในการทำความเข้าใจได้ ซึ่งผลการทดลองได้โมเดลจำแนกความสามารถในการทำความเข้าใจได้ออกเป็น 3 ระดับ คือ

- ระบบที่มีความสามารถในการทำความเข้าใจยาก

F_1 = 3.111NC + 14.963ANAUW-3.198ANMUW + 4.454ANAss + 54.366ANAgg + 36.587MAXHAgg + 187.922ANGen - 52.827MAXDIT - 15.061WMBO + 16.679NOS + 14.691ANDM + 13.105ANET - 96.976

- ระบบที่มีความสามารถในการทำความเข้าใจปานกลาง

F_2 = 1.62NC + 12.096ANAUW - 7.774ANMUW + 11.231ANAss + 85.822ANAgg + 26.719MAXHAgg + 141.52ANGen - 45.229MAXDIT - 3.07WMBO + 25.469NOS + 21.118ANDM - 4.367ANET - 105.755

- ระบบที่มีความสามารถในการทำความเข้าใจง่าย

F_3 = 1.612NC + 6.486ANAUW + 5.04ANMUW + 82.146ANAss + 32.366ANAgg + 11.018MAXHAgg + 68.621ANGen - 4.393MAXDIT - 2.311WMBO + 10.453NOS + 4.885ANDM + 8.044ANET - 88.325

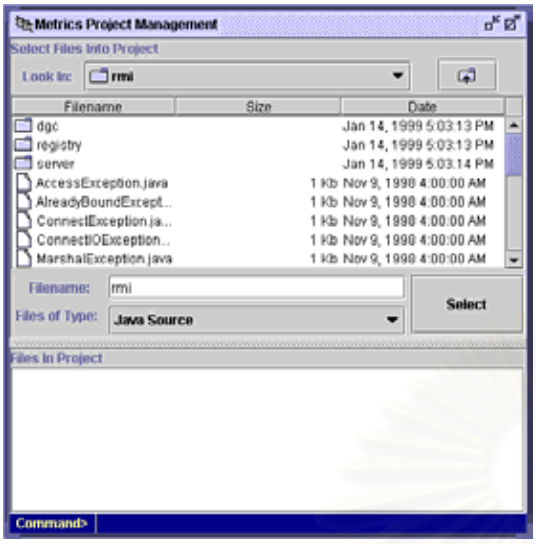
ประโยชน์ของงานวิจัยนี้คือนักวิเคราะห์, นักออกแบบและนักพัฒนาซอฟต์แวร์สามารถนำโมเดลนี้ไปใช้วัดความสามารถในการทำความเข้าใจได้ของซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบระบบเชิงวัตถุ และช่วยเพิ่มข้อมูลการตัดสินใจว่าควรแก้ไขแผนภาพคลาสและซีเควนซ์ก่อนนำไปพัฒนาเป็นซอฟต์แวร์หรือไม่ โดยในการวัดระดับความสามารถในการทำความเข้าใจได้นั้น จะต้องทำการคำนวณค่ามาตรวัด NC, ANAUW, ANMUW, ANAss, ANAgg, MAXHAgg, ANGen และMAXDIT จากแผนภาพคลาส ส่วนมาตรวัด WMBO, NOS, ANDM และ ANET คำนวณได้จากแผนภาพซีเควนซ์ในขั้นตอนการวิเคราะห์และออกแบบระบบด้วยยูเอ็มแอล ซึ่งในการทำนายจะต้องแทนค่ามาตรวัดทุกตัวเพื่อคำนวณค่าทั้ง 3 สมการ โดยค่าของการทำนายระดับความสามารถในการทำความเข้าใจได้ขึ้นอยู่กับค่าของสมการที่มีค่ามากที่สุด ขณะนี้กลุ่มงานวิจัย กำลังพัฒนาเครื่องมือเพื่อช่วยในการคำนวณค่ามาตรวัดต่างๆ และวัดระดับความสามารถในการทำความเข้าใจได้

5. เครื่องมือคำนวณมาตรวัดซอฟต์แวร์เชิงวัตถุ

ในปัจจุบันเครื่องมือวัดซอฟต์แวร์เชิงวัตถุแบบอัตโนมัติ JMetric [1] และ JavaNCSS เป็นเครื่องมือที่นำมาใช้ในการคำนวณหาค่ามาตรวัดขนาดและมาตรวัดคุณภาพ เพื่อนำค่ามาตรวัดที่คำนวณได้ไปใช้ในการประเมินซอฟต์แวร์

5.1 เครื่องมือวัด JMetric

เครื่องมือวัดนี้พัฒนาด้วยภาษาจาวา โดยทีมงาน JMetric ของ School of Information Technoloty ที่ Swinburne University of Technology ประเทศออสเตรเลีย [3] เครื่องมือ JMetric นี้คำนวณค่ามาตรวัดจากโปรแกรมภาษาจาวา ซึ่งมาตรวัดที่สามารถคำนวณได้ ได้แก่ มาตรวัดจำนวนบรรทัด (Lines of Code), มาตรวัดจำนวนสเตทเมนท์ (Statement Count), มาตรวัดระดับของการขาดความสัมพันธ์ภายใน (LCOM) และมาตรวัดไซโคลเมติกของแมคเคบ รวมถึงการคำนวณโดยนับจำนวนแพ็คเกจ, จำนวนคลาส, จำนวนเมธทอด และจำนวนตัวแปรอีกด้วย รูปที่ 6 แสดงหน้าจอโปรแกรม JMetric ซึ่งในภาพเป็นหน้าจอการเลือกโปรแกรมภาษาจาวาที่ต้องการนำมาคำนวณค่ามาตรวัด



รูปที่ 6 แสดงหน้าจอโปรแกรม JMetric

5.2 เครื่องมือวัด JavaNCSS

เครื่องมือวัดนี้พัฒนาด้วยภาษาจาวา โดย Christoph Clemens Lee [4] คำนวณมาตรวัดจากโปรแกรมภาษาจาวาเช่นเดียวกับเครื่องมือวัด JMetric ซึ่งมาตรวัดหลักที่คำนวณ คือ มาตรวัดจำนวนสเตทเมนท์ของซอร์สโค้ด (Non-Commenting Source Statements) คำนวณโดยนับจำนวนสเตทเมนท์ทั้งหมดของตัวโปรแกรม ซึ่งจะไม่นับสเตทเมนท์ที่เป็นคำอธิบาย (Comment) และมาตรวัดไซโคลเมติกของแมคเคบ

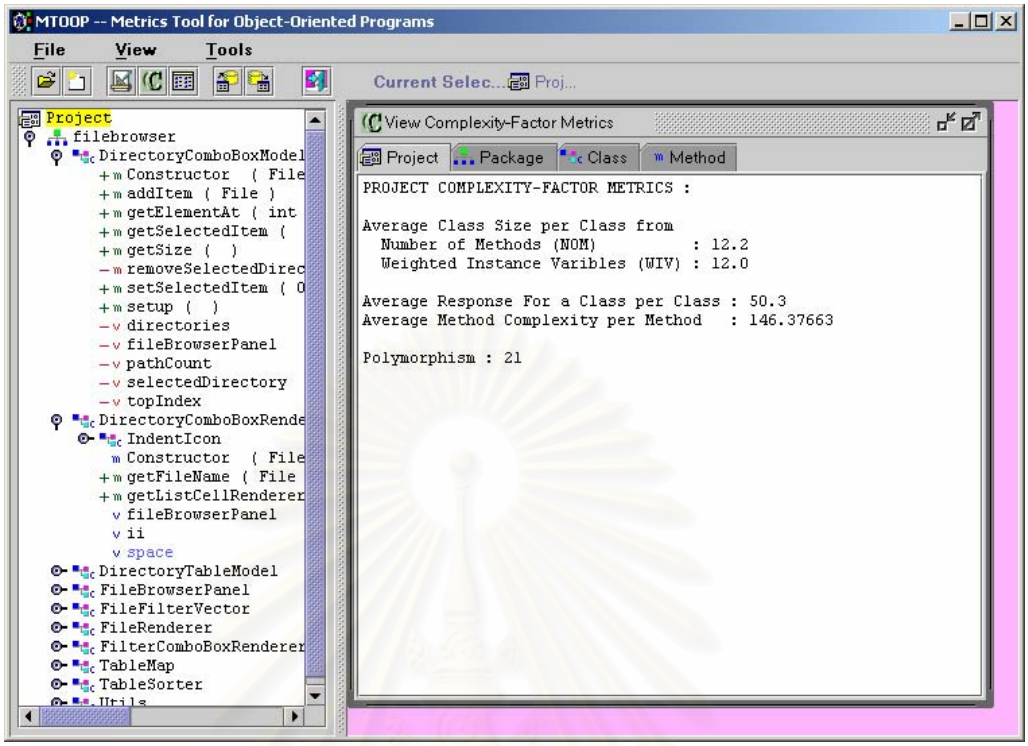
6. เครื่องมือวัด MTOOP

เครื่องมือที่ใช้วัดคุณภาพโปรแกรมเชิงวัตถุนี้ มีชื่อว่า Measurement Tool for Object-Oriented Programs (MTOOP) เป็นเครื่องมือที่พัฒนาขึ้นโดยกลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุ เพื่อใช้ในการคำนวณค่ามาตรวัดสำหรับโปรแกรมภาษาจาวาแบบอัตโนมัติ ซึ่งเครื่องมือ MTOOP ในปัจจุบันได้พัฒนาขึ้นมาใช้งานถึงรุ่นที่ 3 แล้ว โดยเครื่องมือ MTOOP รุ่นที่ 1 พัฒนาขึ้นโดยคุณสมหวัง แซ่ตั้ง [22] เป็นเครื่องมือที่ใช้ในการวัดมาตรวัดพื้นฐาน ส่วนเครื่องมือ MTOOP รุ่นที่ 2 พัฒนาโดยคุณวัฒนชัย รอดกำเนิด [21] ซึ่งปรับปรุงและพัฒนาเครื่องมือ MTOOP รุ่นที่ 1 เพิ่มเติม โดยได้เพิ่มมาตรวัดที่เป็นปัจจัยที่ใช้ในการวัดค่าความซับซ้อนของโปรแกรม และเครื่องมือ MTOOP รุ่นที่ 3 พัฒนาโดยคุณเมธาวี แดงเพ็ง [20] ได้เพิ่มความสามารถในการคำนวณมาตรวัดสำหรับใช้วัดคุณภาพในด้านการนำกลับมาใช้ใหม่ของโปรแกรมภาษาจาวา ซึ่งเครื่องมือ MTOOP รุ่นที่ 3 ขณะนี้อยู่ในขั้นตอนของการดำเนินการพัฒนาอยู่ ดังนั้นเครื่องมือ MTOOP ที่จะกล่าวถึงในที่นี้จึงหมายความถึงเครื่องมือ MTOOP ในรุ่นที่ 1 และ 2 ซึ่งพัฒนาขึ้นด้วยโปรแกรมภาษาจาวา บนระบบปฏิบัติการวินโดวส์ โดยใช้คลาสไลบรารีของจาวาดีเวลลอปเมนต์ทูลคิต (Java Development Tool Kit – JDK) เวอร์ชัน 1.2 ช่วยในการพัฒนา และมีคลาสสวิงที่ช่วยในการสร้างส่วนติดต่อกับผู้ใช้ (Graphic User Interface – GUI)

6.1 มาตรวัดที่สามารถคำนวณได้จากเครื่องมือ MTOOP รุ่นที่ 1

จำแนกประเภทของมาตรวัดออกเป็น 5 ประเภท ได้แก่

- มาตรวัดสำหรับโปรเจค ได้แก่ มาตรวัดจำนวนแพ็คเกจ, จำนวนคลาส, จำนวนเมทธอด, จำนวนเมทธอดต่อคลาส, จำนวนบรรทัด, จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนท์
- มาตรวัดสำหรับแพ็คเกจ ได้แก่ มาตรวัดจำนวนคลาส, จำนวนเมทธอด, จำนวนบรรทัด, จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนท์
- มาตรวัดสำหรับคลาส ได้แก่ มาตรวัดระดับความลึกของการสืบทอดคุณสมบัติ, จำนวนเมทธอด, จำนวนบรรทัด, จำนวนสเตทเมนต์, จำนวนตัวแปรอินสแตนท์ และการขาดความสัมพันธ์ภายในคลาส
- มาตรวัดสำหรับเมทธอด ได้แก่ จำนวนพารามิเตอร์, จำนวนบรรทัด, จำนวนสเตทเมนต์, ค่าไซโคลเมติกของแมคเคบ, จำนวนตัวแปรของแต่ละเมทธอด และขนาดความสัมพันธ์ระหว่างวัตถุ
- มาตรวัดสำหรับตัวแปรอินสแตนท์ ได้แก่ จำนวนครั้งที่ตัวแปรอินสแตนท์ถูกเรียกใช้ และจำนวนเมทธอดที่เรียกใช้ตัวแปรอินสแตนท์



รูปที่ 7 แสดงหน้าจอการดูค่ามาตรวัด (View Metrics) [22]

6.2 มาตรวัดที่สามารถคำนวณได้จากเครื่องมือ MTOOP รุ่นที่ 2

จำแนกประเภทของมาตรวัดออกเป็น 5 ประเภท เช่นกัน ซึ่งจะแสดงเฉพาะมาตรวัดที่เพิ่มเข้ามาได้แก่

- มาตรวัดสำหรับโปรเจค ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งโปรเจค แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนต์ และค่าถ่วงน้ำหนักของตัวแปรอินสแตนท์, ค่าเฉลี่ยของความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมทธอด และค่าโพลีมอร์ฟิซึม
- มาตรวัดสำหรับแพ็คเกจ ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งแพ็คเกจ แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนท์ และค่าถ่วงน้ำหนักของตัวแปรอินสแตนท์, ค่าเฉลี่ยของความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมทธอด และค่าโพลีมอร์ฟิซึม
- มาตรวัดสำหรับคลาส ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งแพ็คเกจ แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนท์ และค่าถ่วงน้ำหนักของตัวแปรอินสแตนท์, ค่าความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมทธอด และค่าโพลีมอร์ฟิซึม
- มาตรวัดสำหรับเมทธอด ได้แก่ มาตรวัดขนาดของเมทธอด และค่าความซับซ้อนของเมทธอด
- มาตรวัดสำหรับตัวแปรอินสแตนท์ เป็นมาตรวัดเช่นเดียวกับเครื่องมือ MTOOP รุ่นที่ 1

ในรูปที่ 7 แสดงหน้าจอของเครื่องมือที่ใช้วัดคุณภาพโปรแกรมเชิงวัตถุ MTOOP รุ่นที่ 2

6.3 หลักการพัฒนาเครื่องมือวัด MTOOP

การคำนวณค่ามาตรวัด เริ่มจากสร้างคอมไพเลอร์ ที่ทำหน้าที่นำโปรแกรมต้นฉบับมาผ่านขบวนการสร้างตัวแปลภาษาเพื่อสร้างซินแท็กซ์ทรี โดยงานวิจัยนี้ได้เลือกเครื่องมือช่วยสร้างตัวแปลภาษาที่ชื่อว่า จาวาคอมไพเลอร์คอมไพเลอร์ (Java Compiler Compiler-JavaCC) การแปลภาษาโปรแกรมต้นฉบับ มีขั้นตอนดังต่อไปนี้

6.3.1 ขั้นตอนเลกซิคัลอนาไลซิส (Lexical analysis)
ในขั้นตอนนี้จะมีหน่วยย่อยของตัวแปลภาษาที่เรียกว่า เลกซิคัลอนาไลเซอร์ (Lexical analyzer) หรือเรียกอีกชื่อหนึ่งว่า สแกนเนอร์ (Scanner) มีหน้าที่ทำการอ่านอักขระจากภาษาต้นแบบแล้วจัดการแยกกลุ่มอักขระเหล่านั้นออกเป็นโทเคน (Token) ข้อมูลเข้าของขั้นตอนนี้ คือ โค้ดโปรแกรมภาษาจาวา ข้อมูลออกของขั้นตอนนี้ คือ โทเคนของโปรแกรมภาษาจาวานั้น

6.3.2 ขั้นตอนซินแท็กซ์อนาไลซิส (Syntax analysis)
หน่วยย่อยของตัวแปลภาษาในขั้นตอนนี้เรียกว่า ซินแท็กซ์อนาไลเซอร์ (Syntax analyzer) หรือเรียกอีกชื่อหนึ่งว่าพาร์เซอร์ (Parser) มีหน้าที่ตรวจสอบว่าโทเคนที่ได้จากขั้นตอนเลกซิคัลอนาไลซิส เรียงกันถูกต้องตามหลักไวยากรณ์ของภาษาที่กำหนดไว้หรือไม่ โดยจะได้เป็นโครงสร้างต้นไม้ที่ เรียกว่า เอเอสที (Abstract Syntax Tree-AST) ซึ่งเป็นโครงสร้างต้นไม้ที่มีการลดทอนให้เหลือแต่ส่วนของโทเคนที่จำเป็นต้องใช้ในการสร้างโค้ดเท่านั้น ในแต่ละโหนด (Node) ของเอเอสทีจะเก็บข้อมูลหรือคุณสมบัติต่างๆ ที่อยู่ภายในโหนด เช่น เป็นคลาส เป็นเมทธอด เป็นคำสั่งเงื่อนไข เป็นต้น ข้อมูลเข้าของขั้นตอนนี้ คือ โทเคนของโปรแกรมภาษาจาวาที่ได้จากขั้นตอนเลกซิคัลอนาไลซิส ข้อมูลออกของขั้นตอนนี้ คือ โครงสร้างต้นไม้ที่เรียกว่า เอเอสที

6.3.3 ขั้นตอนการคำนวณค่ามาตรวัด
หลังจากที่ได้โครงสร้างต้นไม้แล้ว สามารถคำนวณค่าข้อมูลหรือคุณสมบัติต่างๆ ของมาตรวัดได้ โดยการท่องไปตามโหนด (Traverse node) ต่างๆ และทำการคำนวณค่ามาตรวัด โดยในขั้นตอนนี้ได้สร้างแพ็กเกจสำหรับคำนวณค่ามาตรวัด ซึ่งเป็นชุดของคลาสหน่วยรู้จำ โดยมีคลาสมาตรวัด (Metrics) ทำหน้าที่ท่องไปบนซินแท็กซ์ทรีเพื่อเก็บข้อมูลต่างๆ จากโหนด โดยเก็บอยู่ในรูปแบบของตัวแปรแบบเวกเตอร์

6.3.4 ขั้นตอนการตรวจสอบค่ามาตรวัด
เมื่อคำนวณค่ามาตรวัดที่ได้จากเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุแล้ว ได้ทำการตรวจสอบค่ามาตรวัดที่คำนวณได้ โดยเปรียบเทียบกับค่ามาตรวัดที่คำนวณได้จากเครื่องมือ JMetric [1] และ JavaNCSS

6.4 แนวทางการพัฒนาเครื่องมือวัดอื่นๆ
เนื่องจากขณะนี้ทางกลุ่มงานวิจัยได้มุ่งเน้นการวัดคุณภาพของซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบเชิงวัตถุ ดังนั้นจึงได้มีแนวทางการพัฒนาเครื่องมือให้สามารถคำนวณค่ามาตรวัดเชิงวัตถุที่คำนวณจากขั้นตอนการวิเคราะห์และออกแบบเชิงวัตถุด้วยยูเอ็มแอลได้ เพื่อให้สอดคล้องกับงานวิจัยในหัวข้อที่ 4.3.5 ทั้งหมด และสามารถนำเครื่องมือที่ได้จากงานวิจัยดังกล่าวไปใช้งานได้จริง เพื่อเป็นประโยชน์ต่อไป ซึ่งขณะนี้กำลังอยู่ในขั้นตอนของการดำเนินงาน

7. สรุป
การวัดซอฟต์แวร์เป็นวิธีการที่สามารถทำได้ในทุกๆ ขั้นตอนของการพัฒนาซอฟต์แวร์ และมีส่วนช่วยในการประเมินสถานะของโครงงาน และการตรวจสอบคุณภาพซอฟต์แวร์ในด้านต่างๆ ในการวัดซอฟต์แวร์สามารถวัดได้ทั้งกระบวนการพัฒนา, ผลผลิต, และทรัพยากรของซอฟต์แวร์ ซึ่งการวัดจะนำมาตรวัดมาช่วยในการคำนวณค่าของคุณภาพในด้านต่างๆ บทความนี้ได้นำเสนอผลงานวิจัยต่างๆ ที่นำวิธีการวัดซอฟต์แวร์ไปใช้งานจริง โดยเน้นการนำมาตรวัดเชิงวัตถุไปใช้วัดคุณภาพของซอฟต์แวร์ในด้านความสามารถในการนำกลับมาใช้ใหม่, ความน่าเชื่อถือ, ความสามารถในการบำรุงรักษาของซอฟต์แวร์ ในขั้นตอนการวิเคราะห์และออกแบบด้วยยูเอ็มแอล และได้นำเสนอเครื่องมือสำหรับคำนวณมาตรวัดซอฟต์แวร์เชิงวัตถุที่มีชื่อว่า Measurement Tool for Object-Oriented Programs รุ่นที่ 2 (MTOOP v.2) ที่นำไปคำนวณมาตรวัดแบบอัตโนมัติ สำหรับเครื่องมือ MTOOP รุ่นที่ 3 ที่กำลังพัฒนาอยู่ในขณะนี้ ได้รวบรวมมาตรวัดใหม่ๆ เพิ่มเติมเข้าไป เพื่อให้สามารถวัดคุณภาพในด้านการนำกลับมาใช้ใหม่ได้เพิ่มขึ้น และขณะนี้กลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุกำลังศึกษาการวัดคุณภาพ

ซอฟต์แวร์ในด้านอื่นๆ นอกเหนือจากที่ได้นำเสนอ และพัฒนาเครื่องมือสำหรับวัดคุณภาพของโมเดลการออกแบบที่เขียนด้วยยูเอ็มแอล (Unified Modeling Language – UML) เพื่อสนับสนุนผลงานวิจัยการวัดคุณภาพซอฟต์แวร์ในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์

เอกสารอ้างอิง

[1] A. AJ Cain, "JMetric", http://www.it.swin.edu.au/projects/jmetric/.

[2] A. H. Watson and T.J. McCabe, "Structure Testing: A Testing Methodology Using the Cyclomatic Complexity Metric", *National Institute of Standards and Technology Special Publication,* pp. 500-235, September 1996.

[3] B. W. Boehm, B. Clark, E. Horowitz et al., "Cost models for future life cycle processes: CCOMO 2.0", Annals of Software Engineering, pp. 1-24, November 1995.

[4] C. C. Lee, http://www.kclee.com/clemens/java/javancss/

[5] H. Kim and C. Boldyreff, "Developing Software Metrics Applicable to UML Models", *Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE)*, June 2002.

[6] K.El. Emam, W. Melo and J.C. Machado, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics", *Journal of Systems and Software*, February 2001.

[7] L. C. Briand, C. Bunse, and J. W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs", *IEEE Transactions on Software Engineering*, Vol, 27, No.6, pp. 513-530, June 2001.

[8] L.C. Briand, J. Daly, V. Porter and J. Wust, "Predicting Fault-Prone Classes with Design Measures in Object-Oriented Systems", *Proceeding of 9th International Symposium on Software Reliability Engineering*, 1998.

[9] M. Genero and M. Piattini, "Empirical Validation of Measures for class diagram Structural Complexity through Controlled Experiments", *Proceedings of 11th International on Computer Science Society*, pp. 95–104, 2001.

[10] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using Metrics to Predict OO Information Systems Maintainability", *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001)*, Interlaken, Switzerland, June 4-8, 2001.

[11] M. Genero, M. Piattini and C. Calero, "Assurance of Conceptual Data Model Quality Based on Early Measures", *Proceedings of 2nd Asia-Pacific Conference on Quality Software*, pp. 97-103, 2001.

[12] M. Genero, M. Piattini and C. Calero, "Early Measures for UML class diagrams", *L'Object.* Vol. 6. No. 4, 2000.

[13] M. Hitz and B. Montazeri, "Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective", *IEEE Transaction on Software Engineering,* Vol. 22, No. 4, pp. 267-271, April 1996.

[14] M. Kiewkanya, N. Jindasawat, N. Prompoon, P. Muenchaisri, "Constructing Understandability Model from Design Metrics", *Proceedings of the Fifteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2003)*, pp. 208-215, 1-3 July 2003.

[15] M. Kiewkanya, P. Muenchaisri, "Internal-Reuse Measurement of Object-Oriented Software from UML Class and Sequence Diagrams", *Proceedings*

of the 6th National Computer Science and Engineering Conference (NCSEC 2002), 29-31 October 2002.

[16] M. Thongmak, P. Muenchaisri, "Predicting Faulty Classes using Design Metrics with Discriminant Analysis", *Proceedings of the International Conference on Software Engineering Research and Practice (SERP'03)*, pp. 621-627, 23-26 June 2003.

[17] N. E. Fenton and S. L. Pfleeger, "Software Metrics : A Rigorous and Practical Approach", PWS Publishing Company, 1997.

[18] S. R. Chidamber and C. F. Kemerer, "A Metric Suit for Object-Oriented Design", *IEEE Transaction on Software Engineering,* Vol. 20, No. 6, pp. 476-493, June 1994.

[19] T.J. McCabe, "A Complexity Measure", *IEEE Transaction on Software Engineering,* Vol. 2, No. 4, pp. 308-320, December 1976.

[20] เมธาวี แดงเพ็ง, "การออกแบบและพัฒนาเครื่องมือวัดการนำกลับมาใช้ใหม่สำหรับซอฟต์แวร์ภาษาจาวา" (ปีการศึกษา 2545) สาขาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, หลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย

[21] วัฒนชัย รอดกำเนิด, "การออกแบบและพัฒนาเครื่องมือวัดปัจจัยของความซับซ้อนของโปรแกรมเชิงวัตถุภาษาจาวา" (ปีการศึกษา 2544) สาขาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, หลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย

[22] สมหวัง แซ่ตั้ง, "การออกแบบและพัฒนาเครื่องมือวัดซอฟต์แวร์สำหรับโปรแกรมเชิงวัตถุ" (ปีการศึกษา 2543) สาขาวิทยาศาสตร์คอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย, หลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

Constructing Understandability Model from Design Metrics

Matinee Kiewkanya[1], Nongyao Jindasawat[2], Nakornthip Prompoon[3], Pornsiri Muenchaisri[4]
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University, Thailand
Email: g4122109@yahoo.com[1], nongyao.j@student.chula.ac.th[2],
Nakornthip.S@chula.ac.th[3], Pornsiri.Mu@chula.ac.th[4]

Abstract

UML class and sequence diagrams produced at early phase are the key design artifacts used as blueprints for object-oriented software development. It is obvious that understandability of these blueprints heavily affects on the ease of software implementation and maintenance. The result gained from measuring understandability of these diagrams in early phase can be used as a prime indicator whether to redesign them for making easy implementation and reducing cost of software maintenance in later phases. The purpose of this work is to explore the correlation between structural complexity design metrics and understandability of UML class and sequence diagrams in order to construct an understandability model. The understandability model was built based on data collected from a controlled experiment. The obtained model can identify 3 levels of understandability of UML class and sequence diagrams: difficult, medium, easy.

Keywords: Understandability, Structural complexity metrics, UML class diagram, UML sequence diagram

1. Introduction

The Unified Modeling Language (UML) [3] is accepted as an industrial standard for modeling object-oriented design. It defines notations and semantics of modeling elements and the relationship between these elements. In its current form, class and sequence diagrams are two major artifacts acted as blueprints of object-oriented software. Class diagram, a conceptual model of object-oriented software, shows the classes of the system, their inter-relationships, and the operations and attributes of the classes. While class diagram represents static structure, dynamic structure of software is represented by sequence diagram. Sequence diagram is utilized for modeling software behavior in each scenario. Therefore, the quality of object-oriented software ultimately implemented is heavily dependent on the quality of both diagrams. One of the influential qualities is understandability.

From now on, we interchange the term UML class and sequence diagrams with the term software design model. In this paper, understandability refers to the degree to which the software design model can provide its clear meaning to evaluator. The ease of understanding of software design model leads to the ease of software implementation and maintenance in later phases.

General approach for capturing software understandability is the utilization of software metrics. Traditional metrics including size and comment percentage and object-oriented metrics which consist of Weight Method per Class (WMC), Response For a Class (RFC), Depth of Inheritance Tree (DIT) were suggested to be metrics suitable to measure understandability in [6]. Hitz and Muntazeri proposed that the relevance of coupling as a metric of design quality is related to understandability [11]. Structural complexity of the diagram is claimed to be one of factors affects its understandability [9,10]. The metrics for structural complexity of class diagram were presented by Genero et al.[8]. The metrics were called “Class Diagram-Scope metrics” and were identified into two categories: open-end metrics and close-ended metrics. An empirical study to validate these metrics was presented in [7,9,10]. Furthermore, in [8], they analysed a set of existing object-oriented metrics that can be applied for assessing class diagram complexity containing the metrics of Chidamber and Kemerer [15], Lorenz and Kidd [12], Brito e Abreu and Melo [1] and Marchesi [13]. Kim and Boldreff proposed 27 new metrics to measure various characteristics of UML models [2]. They also presented a CASE tool to facilitate the use of the proposed UML metrics.

Class and sequence diagrams are considered together in this work because they are jointly used to explain software aspect. The goal of this work is to explore a set of structural complexity metrics which are applicable to UML class and sequence diagrams and to use these metrics for creating an

---

This work was supported by "Chulalongkorn-Industry Linkage Research Fund".

Table 1. The structural complexity metrics used in the experiment.

Metrics for class diagram		
Classes		Number of classes (NC)
Attributes		Average number of attributes-unweighted (ANAUW) *
		Average number of attributes-weighted (ANAW) *
Methods		Average number of methods-unweighted (ANMUW) *
		Average number of methods-weighted (ANMW) *
Relationships	Association	Average number of association relationships (ANAss) *
	Aggregation	Average number of aggregation relationships (ANAgg) *
		Number of aggregation hierarchies (NaggH)
		Maximum number of aggregation hierarchies (MaxHAgg)
	Generalization	Average number of generalization relationships (ANGen) *
		Number of generalization hierarchies (NGenH)
		Maximum number of Depth of Inheritance Tree (MaxDIT)
Metrics for sequence diagram		
Scenarios		Number of scenarios (NOS) *
Messages		Weighted messages between objects (WMBO) *
		Average number of return messages (ANRM) *
		Average number of the directly dispatched messages (ANDM) *
		Average number of the elements in the transitive closure of the directly dispatched messages (ANET) *
Conditions		Average number of condition messages (ANCM) *

understandability model. The obtained model consists of 3 functions for 3 groups of understandability: difficult, medium, easy. Software developers can utilize the model to identify understandability level of software design model. When software design model is categorized into medium or difficult level, software developers can decide whether to redesign it in order to improve understandability.

This paper is organized as follows. The next section presents selected metrics which are expected to relate to the understandability of UML class and sequence diagrams. Section 3 describes a controlled experiment to construct an understandability model. Then, experimental results are presented in section 4. Section 5 discusses various issues that threaten the validity of the experiment and the attempt to alleviate them. Conclusions and future works are given in the last section.

2. Metric Selection

The first step to analyze software quality using metrics is to identify a collection of metrics that reflect on software characteristics which are being analyzed. This work is interested in examining the understandability of object-oriented system from the design point of view represented by UML class and sequence diagrams. The structural complexity metrics of UML class and sequence diagrams are focused since structural complexity of diagrams is claimed to be an important factor affects on their understandability. A collection of selected metrics is listed in Table 1.

The metrics used in this work consist of metrics for class diagram and metrics for sequence diagram. Metrics for class diagram are categorized into metrics related to classes, attributes, methods and relationships. Metrics for sequence diagram are categorized into metrics related to scenarios, messages and conditions. Existing metrics are NC, NAggH , MaxHAgg, NGenH and MaxDIT proposed by Genero et al. [8]. Metrics symbolized by '*' in Table 1 are modified from the metrics proposed by Genero et al. [8] and Kim & Boldreff [2]. These metrics will be consequentially validated with experimental result indicating whether they can be indicators of understandability. Definition of all metrics is presented in Appendix A.

3. A Controlled Experiment

The sample software design models will be classified into 3 groups according to their understandability level: difficult, medium, easy. Understandability level of each software design model is determined using examination score which is obtained from a controlled experiment. This section describes the experiment carried out to construct an understandability model.

3.1 Experimental Aim and Definition

The main goal of this experiment is to construct an understandability model of UML class and sequence diagrams from structural complexity metrics. In this work, understandability is defined as the degree to which the software design model can provide its clear meaning to evaluator.

3.2 Subjects

Experimental subjects were 30 graduate students from the Department of Computer Engineering at Chulalongkorn University, Bangkok, Thailand, who passed classes on Software Requirements Engineering and Object-Oriented Technology. During lectures, students were taught basic software engineering principles and object-oriented development techniques. The lectures were supplemented by practical lessons where the students had the opportunity to design real-world object-oriented software using UML diagrams. The subjects were classified into 10 groups by considering grades they obtained from classes mentioned above. Each group has one A, one B+ and one B students. This was performed to reduce the difference of subject ability in understanding the software design with UML among groups.

3.3 Experimental Material

Twenty software design models with different domains were used in this experiment. Each software documentation included the general software description, the class diagram, the sequence diagrams, the examination, and debriefing questionnaire. The examination of each software design model contained 20 questions for assessing subject understandability of the class and sequence diagrams. All of questions related to understanding of software structure and behavior described by the elements in class and sequence diagrams including attributes, methods, classes, relationships, scenarios, messages and conditions. Over ninety percents of questions were open-ended questions in order to prevent guessing. The debriefing questionnaire was used to capture personal information, experience, motivation, and subjective opinion of each subject. Examples of examination questions and debriefing questionnaire are given in Appendix B and C.

3.4 Experimental Task

There were two tasks to be performed by the participants. First, 3 subjects in each group were asked to complete the examinations of 2 software design models that randomly assigned for the group. In the same group, each subject received a documentation set which is different from the one examined by the subject sitting next to him/her. Each examination was performed until 30 minutes has passed or the subject already completed it before the timeout was reached. This timeout period was determined from a pilot test. There was 15-minute break between 2 examinations. The second task was to complete a debriefing questionnaire.

3.5 Data Collection

Independent variables are the design metrics introduced in section 2 which indicate the structural complexity of UML class and sequence diagrams.

Dependent variable is the understandability level of UML class and sequence diagrams of each software. The mean of 3 subjects' score for the examination of each software design model was converted by experts into 0,1, or 2 which indicates the understandability levels: difficult, medium or easy respectively.

4. Experimental Results

The data collected from the experiment was analyzed using statistical techniques, Correlation analysis and Discriminant analysis. Statistical analysis was automated using SPSS package [4]. This section presents statistical results.

4.1 Correlation Analysis

Section 2 presents first 18 metrics to be examined for understandability model. Correlation between each pair of metrics was considered in order to discard metrics that provide redundant information (i.e. the metric measures similar property as other metrics). This can be automated by applying Pearson's correlation test with significant at the 0.01 level. Result of correlation analysis is shown in Table 2.

For each couple of highly correlated metrics, only one of them will be selected. Linear regression with one independent variable was performed for each metric. Then, adjusted R square value was used to determine the best choice. Adjusted R square value of independent variable indicates that it can explain the variance of dependent

variable well or not. The metric which has higher adjusted R square value will be chosen. Following this selection process, ANAW, ANMW, NAggH and AGenH were discarded.

Table 2. Correlation analysis of candidate metrics.

Correlated Metrics	Pearson Correlation	Significant (2-tailed)
ANAW & ANAUW	0.625	$p < 0.01$
ANMW & ANMUW	0.997	$p < 0.01$
ANAgg & NAggH	0.712	$p < 0.01$
ANGen & NGenH	0.880	$p < 0.01$

4.2 Constructing Understandability Model

As mentioned earlier, in this experiment, understandability was classified into 3 levels: difficult, medium, easy. The numbers of sample software design models in each level gathered from the experiment were 7, 7, and 6 respectively. UML class and sequence diagrams of all software were measured using 14 remainder metrics. An understandability model was created from collected data applying Discriminant Analysis.

Discriminant analysis is multivariate technique concerned with separating distinct groups of object (or observations) and with allocating new objects to previously defined groups [14]. Discriminant analysis employs a concept very similar to the regression equation, and it is called the discriminant function [16]. Discriminant function uses a weighted combination of prediction variable (independent variable) values to classify an object into the criterion variable (dependent variable) groups. In this work, criterion variable is understandability level and predictor variables are the 14 metrics. Each object's score on the discriminant function, called discriminant score, will depend upon its values on the various predictor variables.

All metric variables in the model must also pass the tolerance criterion which was set to 0.001 for this experiment. A metric variable is not considered in the final model if it causes the tolerance of another variable to drop below the pre-set tolerance criterion. Table 3 shows 2 metrics that were not entered to the model.

Table 3.The result of variables failing tolerance test.

	Within-Groups Variance	Tolerance	Minimum Tolerance
ANRM	1.108	0.000	.000
ANCM	.367	0.000	.000

Table 4 displays standardized canonical discriminant function coefficients of 2 best functions for classifying 3 groups of understandability level. The metric which has high significant for group distinction should provide high coefficient (coefficient sign is not considered). Table 4 reveals that, for the first function, MaxDIT, ANGen, ANET, ANAss, MaxHAgg, ANMUW, ANDM, NOS, ANAUW, NC, WMBO and ANAgg are the metrics heavily affect on group distinction respectively. In the second function, NOS, ANMUW, ANDM, ANAgg, NC, MaxDIT, WMBO, ANAss, ANGen, ANET, MaxHAgg and ANAUW are the predictor variables heavily influence on group distinction respectively.

Table 4. Standardized canonical discriminant function coefficients.

	Function	
	1	2
NC	-.801	-1.321
ANAUW	-1.062	-.066
ANMUW	1.559	-2.325
ANAss	2.064	-.891
ANAgg	-.584	1.514
MaxHAgg	-1.777	-.289
ANGen	-3.078	-.520
MaxDIT	4.241	-1.271
NOS	-1.203	3.094
WMBO	.600	.899
ANRM	-1.275	2.085
ANET	2.192	.483

Three Fisher's linear discriminant functions are formed in order to classify three groups of understandability level as the follows :

Difficult Level's function:

F_1 = 3.111NC + 14.963ANAUW-3.198ANMUW + 4.454ANAss + 54.366ANAgg + 36.587MAXHAgg + 187.922ANGen - 52.827MAXDIT - 15.061WMBO + 16.679NOS + 14.691ANDM + 13.105ANET - 96.976

Medium Level's function:

F_2 = 1.62NC + 12.096ANAUW - 7.774ANMUW + 11.231ANAss + 85.822ANAgg + 26.719MAXHAgg + 141.52ANGen - 45.229MAXDIT - 3.07WMBO + 25.469NOS + 21.118ANDM - 4.367ANET - 105.755

Easy Level's function:

F_3 = 1.612NC + 6.486ANAUW + 5.04ANMUW + 82.146ANAss + 32.366ANAgg + 11.018MAXHAgg + 68.621ANGen - 4.393MAXDIT - 2.311WMBO + 10.453NOS + 4.885ANDM + 8.044ANET - 88.325

4.3 How to Use the Understandability Model

For a new software design model, the metrics NC, ANAUW, ANMUW, ANAss, ANAgg, MAXHAgg, ANGen and MAXDIT will be measured from UML class diagram, and WMBO, NOS, ANDM and ANET will be measured from sequence diagrams. Function F1, F2 and F3

will be calculated. Then the software design model will be allocated to the group that provides highest value among 3 functions. For example, if F3 value is more than F1 and F2 value, the understandability of software design model will be categorized into group 3 which is easy level.

4.4 Validating Understandability Model

The obtained understandability model was employed to classify the group of sample software design models that used for generating understandability model. The result revealed that 100% of sample software design models were accurately classified. The experiment was repeated with 5 new software design models in order to validate the understandability model. By applying the understandability model, the 4 out of 5 new software design models were correctly classified.

5. Threats to Validity

Following several empirical studies [5,7,9,10], this section discusses the various issues that threaten the validity of the experiment and the way attempted to alleviate them.

5.1 Threats to Construct Validity

The construct validity is the degree to which the independent and the dependent variables are accurately measured by the measurement instruments used in the experiment. The dependent variable used in this experiment is the level of understandability obtained from the accuracy of examination answers. The understandability level of each software design model was classified by experts similar to classification of student grades. This could be considered significant. For construct validity of the independent variables, all metrics capture the number of elements in UML class and sequence diagrams and relationships between them. So, they are related to the structural complexity of UML class and sequence diagrams. The construct validity of independent variables can be considered valid.

5.2 Threats to Internal Validity

The internal validity is the degree of confidence in a cause-effect relationship between factors of interest and the observed results.

- **Differences among subjects.** Each software design model was evaluated by group of 3 subjects. Although the ability of understanding the software design with UML is not exactly equivalent among each group. Differences among groups are reduced by assigning one A, one B+ and one B students.
- **Knowledge of the universe of discourse among UML class and sequence diagrams.** UML class and sequence diagrams were designed from different universe of discourse, but they were simple enough to be easily understood by the subjects. So, knowledge of the domain does not affect internal validity.
- **Accuracy of subject responses.** Subjects have at least medium experience in modeling and understanding the software design with UML. This reality is confirmed by responses of debriefing questionnaires. Their responses to examination are considered valid.
- **Learning effects.** Learning effect is little relevant because each subject performed only 2 examinations.
- **Fatigue effects.** Each subject performed 2 examinations with 15-minute break between them. Each examination took less than 30 minutes. The fatigue is not relevant.
- **Persistence effects.** Subjects had never performed a similar experiment. So, persistence effect is avoided.
- **Subject motivation.** All subjects participated this experiment voluntarily. The responses of debriefing questionnaires indicate that 93% of subjects pay heavily attention to perform examination.
- **Other factors.** Plagiarism and influence between subjects could be controlled. Two subjects who sat adjacently performed different examinations. The test was controlled by 2 proctors. The subjects were asked to avoid talking to each other.

5.3 Threats to External Validity

External validity is the degree to which the research results can be generalized to the population under study and to other research settings.

- **Materials and tasks used.** Examination questions tried to capture subjects understanding of UML class and sequence diagrams. All questions were approved by experts. The experiment tried to use UML class and sequence diagrams which represent real software, but the software used are small and simple. The software, has maximum number of classes, contains only 25 classes. This is a limitation of the study since it is difficult to find UML class and sequence diagrams of real world software.
- **Experimental Subject.** To solve the problem of lacking expert participation, the students were used as experimental subjects. We are aware that more experiments with experts should be carried out in order to be able to generalize the results. Nevertheless, this experiment does not require high level of industrial experience. Students are usually accepted as valid subject [17].

6. Conclusions and Future Works

Understandability is one of important quality characteristics of UML class and sequence diagrams which are used as blueprints of object-oriented software. Predicting understandability at the design level will help software designers to alter the design of the software for better performance that leads to the ease of implementation and reduction of maintenance cost. The prime goal of this work is to construct understandability model for UML class and sequence diagrams. The model was built using structural complexity metrics applicable to UML class and sequence diagrams. Twelve structural complexity metrics are good predictor variables for classifying understandability level into 3 groups: difficult, medium and easy. Four out of five new software design models were correctly classified with the obtained understandability model. The experimental result may be considered as the preliminary finding and may provide useful information to software engineering practitioners and researchers. This experiment may be repeated with more number of sample software design models. New metrics should be explored for future experiment. Understandability level can be tried on more than 3 levels. An automated tool for extracting the metric values from UML class and sequence diagrams is under construction.

Appendix A. Definition of Metrics Used in the Experiment

Metric used in the experiment are defined as following :

1. NC: This metric counts the total number of classes.
2. ANAUW: This metric is the proportion of the total number of attributes and the total number of classes.
3. ANAW: This metric is the weighted version of ANAUW. Each attribute is weighted depending on its visibility, i.e. 1.0 for public, 0.5 for protected and 0.0 for private attributes.
4. ANMUW: This metric is the proportion of the total number of methods and the total number of classes.
5. ANMW: This metric is the weighted version of ANMUW. Each method is weighted depending on its visibility as same as weighting attribute in ANAW.
6. ANAss: It is calculated from the total number of association relationships divided by the total number of classes.
7. ANAgg: It is calculated from the total number of aggregation relationships divided by the total number of classes.
8. NaggH: It counts the number of aggregation hierarchies.
9. MaxHAgg: It is the maximum between the HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves.
10. ANGen: It is calculated from the total number of generalization relationships divided by the total number of classes.
11. NGenH: It counts the number of generalization hierarchies.
12. MaxDIT: It is the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalization hierarchy is the longest path from the class to the root of the hierarchy.
13. NOS: It is the total number of scenarios which exactly same as the total number of sequence diagrams.
14. WMBO: It is the total number of average number of messages per instanced objects in all scenario divided by the total number of scenario.
15. ANRM: This metric is the proportion of the total number of return messages in all scenarios and the total number of scenarios.
16. ANDM: It is calculated from the total number of directly dispatched messages (NDM) of each message in all scenarios divided by the total number of scenarios. According to the UML semantics, a message can be an activator of other messages. For example, in Figure 1, the message a() activates the message c(), the NDM value of message a() is 1.
17. ANET: It is calculated from the total number of the elements in the transitive closure of the directly dispatched messages (NET) of each message in all scenarios divided by the total number of scenarios. For example, in Figure 1, NET(a()) = |{ c(), b() }| = 2.
18. ANCM: This is defined as the proportion of the total number of condition messages in all scenarios and the total number of scenarios.

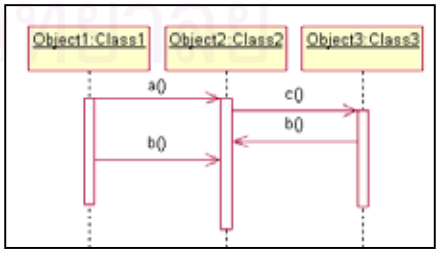


Figure 1 : An example of sequence diagram.

Appendix B. Examination Questions

Questions in examination related to understanding of software structure and behavior described by the elements in class and sequence diagrams. Table 5 displays examples of questions.

Table 5. Examples of examination questions.

Questions for class diagram	
Classes	What are the services provided by class C?
Attributes	What are possible values of attribute A?
	Attribute A of class C can be modified its value from other classes or not ?
	Which attributes of which classes affect to value of attribute A of class C?
Methods	What is functionality of method M?
	Which method of which class should invoke method M of class C?
Relationships	Please describe the relation between class C1 and C2.
	Please describe the multiplicity of class C2 and C2.
Questions for sequence diagram	
Messages	To perform work W, which messages are sent?
	In this sequence diagram, which attribute values are returned from class C?
Conditions	Please explain condition for performing work W?
	Which message is sent if condition CC is fault?
Others	Please describe briefly this sequence diagram.
	Which classes relate to this sequence diagram?
	Which objects are destroyed at end of this sequence diagram?

Appendix C. A Debriefing Questionnaire

The debriefing questionnaire used in the experiment shows as the follows.

Personal Details and Experience

1. What is your age?

Please answer the following 3 questions based on this experience scale:

None Little Average Substantial Professional
1 2 3 4 5

2. What is your experience with software engineering practice?
3. What is your experience with design documents in general?
4. What is your experience with modeling with UML for objected-oriented design?

Motivation and Performance

Please answer the following 3 questions based on this scale:

Not Poorly Fairly Well Highly
1 2 3 4 5

1. Estimate how motivated you were to perform well in this experiment.
2. Estimate how well you understood what was required of you.
3. Estimate your overall understanding of the design documents.
4. What did you understand least about the design documents in this experiment and why?
5. In your opinion, what caused you the most difficulty to understand the design documents?
6. Estimate the accuracy (in percent) of your answer to the examinations.
7. If you could not complete all examinations, please indicate why.

References

[1] F. Brito e Abreu and W. Melo, "Evaluating the Impact of Object-Oriented Design on Software Quality", Proc. of the 3rd International Metric Symposium, 1996.

[2] H. Kim and C. Boldyreff, "Developing Software Metrics Applicable to UML Models", Proc. of the 6th ECOOP Workshop on Quantitative Approaches in Objected-Oriented Software Engineering, June 11, 2002.

[3] J. Rumbaugh, I. Jacobson and G. Booch, "The Unified Modeling Language Reference Manual", Addison Wesley Publishing Company, 1999.

[4] K. Vanichbuncha, Multivariate analysis using SPSS for Windows, Chulalongkorn University Press, 2001.

[5] L. C. Briand, C. Bunse, and J. W. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs", IEEE Transactions on Software Engineering, Vol, 27, No.6, pp. 513-530, June 2001.

[6] L. H. Rosenberg and L. E. Hyatt, "Software Quality Metrics for Object-Oriented Environments", Crosstalk, Vol. 10, 1997.

[7] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using Metrics to predict OO Information Systems Maintainability", Proc. of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001), Interlaken, Switzerland, June 4-8, 2001.

[8] M. Genero, M. Piattini and C. Calero, "Early Measures for UML Class Diagrams", L' Object, Hermes Science Publications, Vol.6 No.4, pp.489-515, 2000.

[9] M. Genero, M. Piattini, "A Controlled Experiment for Validating Class diagram Structural Complexity Metrics", Proc. of the 8th International Conference on Object-Oriented Information Systems (OOIS 2002), Montpellier, France, pp. 372-383, September 2-5, 2002.

[10] M. Genero, M. Piattini, C. Calero, "An Empirical Study to Validate Metrics for Class Diagrams", Proc. of International Database Engineering and Applications Symposium (IDEAS'02), Edmonton, Canada, July 17-19,2002.

[11] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Designs", Proc. of International Symposium on Applied Corporate Computing, Monterrey, Maxico, October, 1995.

[12] M. Lorenz and J. Kidd, Object-Oriented Software Metrics: A Practical Guide. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[13] M. Marchesi, "OOA Metrics for the Unified Modeling Language", Proceedings of the 2nd Euromicro Conference on Software maintenance an reengineering, pp. 67-73,1998.

[14] R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Prentice Hall International, 1988.

[15] S. Chidamber and C. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, pp. 476-4936, June 1994.

[16] S. K. Kachigan, Multivariate statistical analysis : A Conceptual Introduction, Radius Press, New York, 1982.

[17] V.R. Basili, L.C. Briand, and W.L.Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Trans. Software Eng., Vol.22. No.10, pp. 751-761, October 1996.

Using Structural Complexity Design Metrics to Construct Modifiability Model

Nongyao Jindasawat[1], Matinee Kiewkanya[2], Pornsiri Muenchaisri[3]
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University, Thailand
Email: nongyao.j@student.chula.ac.th[1], g4122109@yahoo.com[2], Pornsiri.Mu@chula.ac.th[3]

Abstract

UML class and sequence diagrams obtained at early phase are principle blueprints for object-oriented software development. The modifiability of these diagrams heavily affects on the maintainability of software ultimately implemented. Assessing modifiability at the design level will help software designers to decide if the design of the software should be altered for better performance.

This paper focuses on modifiability of UML class and sequence diagrams. It refers to the ease with which UML class and sequence diagrams can be changed due to changing and adding of software functionality. A controlled experiment was carried out to investigate the correlation between a set of structural complexity design metrics and modifiability of UML class and sequence diagrams in order to construct a modifiability model. The modifiability model was built based on data collected from a controlled experiment. The obtained model can identify 3 levels of the ease of modifiability of UML class and sequence diagrams: easy, medium, difficult.

Keywords: Modifiability, Structural complexity metrics, UML class diagram, UML sequence diagram, Discriminant Analysis

1. Introduction

Software requirements are often changed. This requires software to be modified several times after their initial development. Many literatures indicated that 50-70% of the total life cycle cost of software is spent on software maintenance [3,5,18]. Modifiability of software, the ease with which a software can be changed, is one of an important sub-characteristic of software maintainability. Assessing modifiability at the design level will help software designers to decide if the design of the software should be altered which leads to the ease of maintenance.

Briand et al. proposed a controlled experiment which investigated 2 important components of object-oriented design maintainability: understandability and modifiability, by comparing the effect of "good" and "bad" design principles [9]. Their experiment showed that the system designed according to Coad and Yourdon's object-oriented design principle is significantly easier to maintain. In an empirical study presented by Daly et al., subjects were asked to modify object-oriented with 3 levels of inheritance depth and equivalent object-oriented software with no inheritance in order to examine the effect of inheritance on the maintainability of object-oriented software [6]. The result revealed that 2 out of 3 subjects performed faster when maintaining the object-oriented software with inheritance. Kung et al. presented 4 types of the code change in object-oriented software class library including data change, method change, class change and class library change [1]. They also provided an automated solution to identify different kind of code changes and their impact.

The Unified Modeling Language (UML) [7] is the result of the unification process of earlier object-oriented models and notations. UML models capture the static and dynamic aspects of software. In its current form, class and sequence diagrams are two major artifacts acted as blueprints of object-oriented software. Obviously, the modifiability of these blueprints heavily affects on the ease of software maintenance. From now on, we interchange the term UML class and sequence diagrams with the term software design model.

The goal of this work is to explore a set of structural complexity design metrics and to use these metrics for creating a modifiability model. The metrics applicable to UML models were introduced as follows. The metrics for structural complexity of class diagram were presented by Genero et al.[11]. The metrics were called "Class Diagram-Scope metrics" and were identified into two categories: open-end metrics and close-ended metrics. An empirical study to validate these metrics was presented in [10,12]. Furthermore, in [11], they analysed a set of existing object-oriented metrics that can be applied for assessing class diagram complexity containing the metrics of Chidamber and Kemerer [16], Lorenz and Kidd [13], Brito e Abreu and Melo [2] and Marchesi [14]. Kim and Boldreff proposed 27 new metrics to measure various characteristics of UML models [4].

This work was supported by "Chulalongkorn-Industry Linkage Research Fund 2002".

Table 1. The structural complexity metrics used in the experiment.

Metrics for class diagram		
Classes		Number of classes (NC)
Attributes		Average number of attributes-unweighted (ANAUW) *
		Average number of attributes-weighted (ANAW) *
Methods		Average number of methods-unweighted (ANMUW) *
		Average number of methods-weighted (ANMW) *
Relationships	Association	Average number of association relationships (ANAss) *
	Aggregation	Average number of aggregation relationships (ANAgg) *
		Number of aggregation hierarchies (NaggH)
		Maximum number of aggregation hierarchies (MaxHAgg)
	Generalization	Average number of generalization relationships (ANGen) *
		Number of generalization hierarchies (NGenH)
		Maximum number of Depth of Inheritance Tree (MaxDIT)
Metrics for sequence diagram		
Scenarios		Number of scenarios (NOS) *
Messages		Weighted messages between objects (WMBO) *
		Average number of return messages (ANRM) *
		Average number of the directly dispatched messages (ANDM) *
		Average number of the elements in the transitive closure of the directly dispatched messages (ANET) *
Conditions		Average number of condition messages (ANCM) *

This work selects metrics proposed by Genero et al. and Kim & Boldreff to use in a controlled experiment in order to construct a modifiability model. The obtained model consists of 3 functions for 3 groups of modifiability: easy, medium, difficult. Software developers can utilize the model to identify modifiability level of software design model. When software design model is categorized into medium or difficult level, software developers can decide whether to redesign it in order to improve modifiability.

This paper is organized as follows. The next section presents selected metrics which are expected to relate to the modifiability of UML class and sequence diagrams. Section 3 describes a controlled experiment to construct a modifiability model. Then, experimental results are presented in section 4. Section 5 discusses various issues that threaten the validity of the experiment and the attempt to alleviate them. Conclusions and future works are given in the last section.

2. Metric Selection

The first step to analyze software quality using metrics is to identify a collection of metrics that reflect on software characteristics which are being analyzed. This work explores the correlation between the modifiability of object-oriented design model represented by UML class and sequence diagrams and the structural complexity design metrics. A collection of selected metrics is listed in Table 1. The metrics used in this work consist of metrics for class diagram and metrics for sequence diagram. Metrics for class diagram are categorized into metrics related to classes, attributes, methods and relationships. Metrics for sequence diagram are categorized into metrics related to scenarios, messages and conditions. Existing metrics are NC, NAggH , MaxHAgg, NGenH and MaxDIT proposed by Genero et al. [11]. Metrics symbolized by '*' in Table 1 are modified from the metrics proposed by Genero et al. [11] and Kim & Boldreff [4]. These metrics will be consequentially validated with experimental result indicating whether they can be indicators of modifiability. Definition of all metrics is presented in Appendix A.

3. A Controlled Experiment

The sample software design models will be classified into 3 groups according to their modifiability level: easy, medium, difficult. Modifiability level of each software design model is determined using examination score which is obtained from a controlled experiment. This section describes the experiment carried out to construct a modifiability model.

3.1 Experimental Aim and Definition

The main goal of this experiment is to construct a modifiability model of UML class and sequence diagrams from structural complexity metrics. In this work, modifiability is defined as the ease with which UML class and sequence can be changed.

3.2 Subjects

Experimental subjects were 30 graduate students from the Department of Computer Engineering at Chulalongkorn University, Bangkok, Thailand, who passed classes on Software Requirements Engineering and Object-Oriented Technology. During lectures, students were taught basic software engineering principles and object-oriented development techniques. The lectures were supplemented by practical lessons where the students had the opportunity to design real-world object-oriented software using UML diagrams. The subjects were classified into 10 groups by considering grades they obtained from classes mentioned above. Each group has one A, one B+ and one B students. This was performed to reduce the difference of subject ability in modifying the software design with UML among groups.

3.3 Experimental Material

Twenty software design models with different domains were used in this experiment. Each software documentation included the general software description, the class diagram, the sequence diagrams, the examination, and debriefing questionnaire. The examination of each software design model contained 10 questions for assessing modifiability of the class and sequence diagrams. Subjects were asked to modify design diagrams with tasks covering on changing and adding software functionality. All elements in class and sequence diagrams including attributes, methods, classes, relationships, messages and conditions can be changed by subjects in order to complete the examinations. The debriefing questionnaire was used to capture personal information, experience, motivation, and subjective opinion of each subject. The debriefing questionnaire used in this experiment is given in Appendix B.

3.4 Experimental Task

There were two tasks to be performed by the participants. First, 3 subjects in each group were asked to complete the examinations of 2 software design models that randomly assigned for the group. In the same group, each subject received a documentation set which is different from the one examined by the subject sitting next to him/her. Each examination was performed until 40 minutes has passed or the subject already completed it before the timeout was reached. This timeout period was determined from a pilot test. There was 15-minute break between 2 examinations. The second task was to complete a debriefing questionnaire.

3.5 Data Collection

Independent variables are the design metrics introduced in section 2 which indicate the structural complexity of UML class and sequence diagrams.

Dependent variable is the modifiability level of UML class and sequence diagrams of each software. The mean of 3 subjects' score for the examination of each software design model was converted by experts into 0,1, or 2 which indicates the modifiability levels: easy, medium or difficult, respectively.

4. Experimental Results

The data collected from the experiment was analyzed using statistical techniques, Correlation analysis and Discriminant analysis. Statistical analysis was automated using SPSS package [8]. This section presents statistical results.

4.1 Correlation Analysis

Section 2 presents first 18 metrics to be examined for modifiability model. Correlation between each pair of metrics was considered in order to discard metrics that provide redundant information (i.e. the metric measures similar property as other metrics). This can be automated by applying Pearson's correlation test with significant at the 0.01 level. Result of correlation analysis is shown in Table 2.

For each couple of highly correlated metrics, only one of them will be selected. Linear regression with one independent variable was performed for each metric. Then, adjusted R square value was used to determine the best choice. Adjusted R square value of independent variable indicates that it can explain the variance of dependent

variable well or not. The metric which has higher adjusted R square value will be chosen. Following this selection process, ANAW, ANMW, NAggH and AGenH were discarded.

Table 2. Correlation analysis of candidate metrics.

Correlated Metrics	Pearson Correlation	Significant (2-tailed)
ANAW & ANAUW	0.625	$p < 0.01$
ANMW & ANMUW	0.997	$p < 0.01$
ANAgg & NAggH	0.712	$p < 0.01$
ANGen & NGenH	0.880	$p < 0.01$

4.2 Constructing Modifiability Model

As mentioned earlier, in this experiment, modifiability was classified into 3 levels: easy, medium and difficult. The numbers of sample software design models in each level gathered from the experiment were 6, 7, and 7 respectively. UML class and sequence diagrams of all software were measured using 14 remainder metrics. A modifiability model was created from collected data applying Discriminant Analysis.

Discriminant analysis is multivariate technique concerned with separating distinct groups of object (or observations) and with allocating new objects to previously defined groups [15]. Discriminant analysis employs a concept very similar to the regression equation, and it is called the discriminant function [17]. Discriminant function uses a weighted combination of prediction variable (independent variable) values to classify an object into the criterion variable (dependent variable) groups. In this work, criterion variable is modifiability level and predictor variables are the 14 metrics. Each object's score on the discriminant function, called discriminant score, will depend upon its values on the various predictor variables.

All metric variables in the model must also pass the tolerance criterion which was set to 0.001 for this experiment. A metric variable is not considered in the final model if it causes the tolerance of another variable to drop below the pre-set tolerance criterion. Table 3 shows 2 metrics that were not entered to the model.

Table 3.The result of variables failing tolerance test.

	Within-Groups Variance	Tolerance	Minimum Tolerance
ANET	0.903	0.000	0.000
ANCM	0.368	0.000	0.000

Table 4 displays standardized canonical discriminant function coefficients of 2 best functions for classifying 3 groups of modifiability level. The metric which has high significant for group distinction should provide high coefficient (coefficient sign is not considered). Table 4 reveals that, for the first function, MaxDIT, ANGen, ANDM, ANAgg, NOS, MaxHAgg, NC, ANMUW, WMBO, ANRM, ANAUW and ANAss are the metrics heavily affect on group distinction respectively. In the second function, NOS, MaxHAgg, ANAss, ANDM, ANMUW, MaxDIT, WMBO, ANAgg, ANGen, ANRM, NC and ANAUW are the predictor variables heavily influence on group distinction respectively.

Table 4. Standardized canonical discriminant function coefficients.

	Function	
	1	2
NC	-2.072	-0.629
ANAUW	-0.546	0.229
ANMUW	1.425	2.220
ANAss	-0.289	3.221
ANAgg	6.643	-1.657
MaxHAgg	-2.194	3.296
ANGen	-10.085	1.086
MaxDIT	10.249	2.024
WMBO	-1.109	-1.940
NOS	4.554	-3.979
ANDM	9.544	-2.989
ANRM	-0.598	0.697

Three Fisher's linear discriminant functions are formed in order to classify three groups of modifiability level as the follows:

Easy Level's function:

F_1 = -15.046NC - 23.251ANAUW + 33.09ANMUW + 18.524ANAss + 2019.12ANAgg - 198.833MAXHAgg - 1874.907ANGen + 713.21MAXDIT - 69.56WMBO + 203.132NOS + 654.628ANDM - 11.118ANRM - 727.104

Medium Level's function:

F_2 = -17.481NC - 28.427ANAUW + 33.833ANMUW - 54.17ANAss + 2434.853ANAgg - 262.912MAXHAgg - 2266.81ANGen + 837.74MAXDIT - 69.186WMBO + 254.403NOS + 795.76ANDM - 17.837ANRM - 1010.549

Difficult Level's function:

F_3 = -23.97NC - 35.807ANAUW + 57.567ANMUW – 10.96ANAss + 3114.299ANAgg - 301.974MAXHAgg - 2987.459ANGen + 1137.737MAXDIT - 126.533WMBO + 305.663NOS + 1014.745ANDM - 23.193ANRM - 1668.33

4.3 How to Use the Modifiability Model

For a new software design model, the metrics NC, ANAUW, ANMUW, ANAss, ANAgg, MAXHAgg, ANGen and MAXDIT will be measured from UML class diagram, and WMBO, NOS, ANDM and ANRM will be measured from sequence diagrams. Function F1, F2 and F3

will be calculated. Then the software design model will be allocated to the group that provides highest value among 3 functions. For example, if F3 value is more than F1 and F2 value, the modifiability of software design model will be categorized into group 3 which is difficult level.

4.4 Validating Modifiability Model

The obtained modifiability model was employed to classify the group of sample software design models that used for generating modifiability model. The result revealed that 100% of sample software design models were accurately classified. The experiment was repeated with 5 new software design models in order to validate the modifiability model. By applying the modifiability model, the 3 out of 5 new software design models were correctly classified.

5. Threats to Validity

Following several empirical studies [9,10,12], this section discusses the various issues that threaten the validity of the experiment and the way attempted to alleviate them.

5.1 Threats to Construct Validity

The construct validity is the degree to which the independent and the dependent variables are accurately measured by the measurement instruments used in the experiment. The dependent variable used in this experiment is the level of modifiability obtained from the accuracy of examination answers. The modifiability level of each software design model was classified by experts similar to classification of student grades. This could be considered significant. For the construct validity of the independent variables, all metrics capture the number of elements in UML class and sequence diagrams and relationships between them. So, they are related to the structural complexity of UML class and sequence diagrams. The construct validity of independent variables can be considered valid.

5.2 Threats to Internal Validity

The internal validity is the degree of confidence in a cause-effect relationship between factors of interest and the observed results.

- **Differences among subjects.** Each software design model was evaluated by group of 3 subjects. Although the ability of modifying the software design with UML is not exactly equivalent among each group. Differences among groups are reduced by assigning one A, one B+ and one B students to a group.
- **Knowledge of the universe of discourse among UML class and sequence diagrams.** UML class and sequence diagrams were designed from different universe of discourse, but they were simple enough to be easily understood by the subjects. So, knowledge of the domain does not affect internal validity.
- **Accuracy of subject responses.** Subjects have at least medium experience in modeling the software design with UML. This reality is confirmed by responses of debriefing questionnaires. Their responses to examination are considered valid.
- **Learning effects.** Learning effect is little relevant because each subject performed only 2 examinations.
- **Fatigue effects.** Each subject performed 2 examinations with 15-minute break between them. Each examination took less than 40 minutes. The fatigue is little relevant.
- **Persistence effects.** Subjects had never performed a similar experiment. So, persistence effect is avoided.
- **Subject motivation.** All subjects participated this experiment voluntarily. The responses of debriefing questionnaires indicate that 93% of subjects pay heavily attention to perform examination.
- **Other factors.** Plagiarism and influence between subjects could be controlled. Two subjects who sat adjacently performed different examinations. The test was controlled by 2 proctors. The subjects were asked to avoid talking to each other.

5.3 Threats to External Validity

External validity is the degree to which the research results can be generalized to the population under study and to other research settings.

- **Materials and tasks used.** Examination questions tried to capture modifiability of UML class and sequence diagrams. All questions were approved by experts. The experiment tried to use UML class and sequence diagrams which represent real software, but the software used are small and simple. The software, has maximum number of classes, contains only 25 classes. This is a limitation of the study since it is difficult to find UML class and sequence diagrams of real world software.
- **Experimental Subject.** To solve the problem of lacking expert participation, the students were used as experimental subjects. We are aware that more experiments with experts should be carried out in order to be able to generalize the results. Nevertheless, this experiment does not require high level of industrial experience. Students are usually accepted as valid subject [19].

6. Conclusions and Future Works

Predicting modifiability at the design level will help software designers to alter the design of the software for better performance that leads to the ease of maintenance. The goal of this work is to construct modifiability model for UML class and sequence diagrams using structural complexity design metrics. Twelve structural complexity design metrics are good predictor variables for classifying modifiability level into 3 groups: easy, medium and difficult. Three out of five new software design models were correctly classified with the obtained modifiability model. However, the experimental result may be considered as the preliminary finding and may provide useful information to software engineering practitioners and researchers. This experiment will be repeated with 20 new sample software design models for increasing reliability of the modifiability model. New metrics should be explored for future experiment. Modifiability level can be tried on more than 3 levels. An automated tool for extracting the metric values from UML class and sequence diagrams is under construction.

Understandability of the same set of software design models was explored in another controlled experiment. The collected data of both understandability and modifiability of these software design models will be considered together in order to construct a maintainability prediction model for object-oriented software.

Acknowledgement

We would like to thank Ajarn Chate Patanothai and Ajarn Nakornthip Prompoon for their valuable suggestions and also to all students who are subjects in this experiment.

Appendix A. Definition of metrics used in the experiment

Metric used in the experiment are defined as following :

1. NC: This metric counts the total number of classes.
2. ANAUW: This metric is the proportion of the total number of attributes and the total number of classes.
3. ANAW: This metric is the weighted version of ANAUW. Each attribute is weighted depending on its visibility, i.e. 1.0 for public, 0.5 for protected and 0.0 for private attributes.
4. ANMUW: This metric is the proportion of the total number of methods and the total number of classes.
5. ANMW: This metric is the weighted version of ANMUW. Each method is weighted depending on its visibility as same as weighting attribute in ANAW.
6. ANAss: It is calculated from the total number of association relationships divided by the total number of classes.
7. ANAgg: It is calculated from the total number of aggregation relationships divided by the total number of classes.
8. NaggH: It counts the number of aggregation hierarchies.
9. MaxHAgg: It is the maximum between the HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves.
10. ANGen: It is calculated from the total number of generalization relationships divided by the total number of classes.
11. NGenH: It counts the number of generalization hierarchies.
12. MaxDIT: It is the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalization hierarchy is the longest path from the class to the root of the hierarchy.
13. NOS: It is the total number of scenarios which exactly same as the total number of sequence diagrams.
14. WMBO: It is the total number of average number of messages per instanced objects in all scenario divided by the total number of scenario.
15. ANRM: This metric is the proportion of the total number of return messages in all scenarios and the total number of scenarios.
16. ANDM: It is calculated from the total number of directly dispatched messages (NDM) of each message in all scenarios divided by the total number of scenarios. According to the UML semantics, a message can be an activator of other messages. For example, in Figure 1, the message a() activates the message c(), the NDM value of message a() is 1.
17. ANET: It is calculated from the total number of the elements in the transitive closure of the directly dispatched messages (NET) of each message in all scenarios divided by the total number of scenarios. For example, in Figure 1, NET(a()) = |{ c(), b() }| = 2.
18. ANCM: This is defined as the proportion of the total number of condition messages in all scenarios and the total number of scenarios.

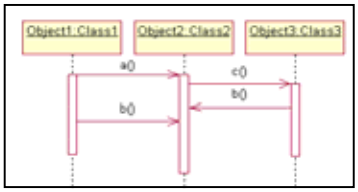


Figure 1 : An example of sequence

Appendix B. A Debriefing Questionnaire

The debriefing questionnaire used in the experiment shows as the follows.

Personal Details and Experience.

1. What is your age?

Please answer the following 3 questions based on this experience scale:

None	Little	Average	Substantial	Professional
1	2	3	4	5

2. What is your experience with software engineering practice?
3. What is your experience with design documents in general?
4. What is your experience with modeling with UML for objected-oriented design?

Motivation and Performance

Please answer the following 3 questions based on this scale:

Not	Poorly	Fairly	Well	Highly
1	2	3	4	5

1. Estimate how motivated you were to perform well in this experiment.
2. Estimate how well you understood what was required of you.
3. Estimate your overall understanding of the design documents.
4. What did you understand least about the design documents in this experiment and why?
5. In your opinion, what caused you the most difficulty to understand the design documents?
6. Estimate the accuracy (in percent) of your answer to the examinations.
7. If you could not complete all examinations, please indicate why.

References

[1] D. Kung, J. Gao, P. Hsia, F. Wen, Y. Toyoshima, and C. Chen, “Change impact identification in object oriented software maintenance,” Proc. Of International Conference on Software Maintenance, 19-23 September 1994, pp. 202-211, 1994.

[2] F. Brito e Abreu and W. Melo, “Evaluating the Impact of Object-Oriented Design on Software Quality”, Proc. of the 3rd International Metric Symposium, 1996.

[3] G. Parikh and N. Zvegintzov, Tutorial on Software Maintenance, IEEE CS Press, Los Alamitos, Calif., Order No.453, 1983.

[4] H. Kim and C. Boldyreff, “Developing Software Metrics Applicable to UML Models”, Proc. of the 6th ECOOP Workshop on Quantitative Approaches in Objected-Oriented Software Engineering, June 11, 2002.

[5] I. Sommerville, Software Engineering, Addison Wesley, 1996.

[6] J. Daly, A Brooks, J Miller, M Roper, and M Wood, “The Effect of Inheritance on the Maintainability of Object-Oriented Software: An Empirical Study,” Proc. Of the International Conference on Software Maintenance (ICSM ’95), 1995.

[7] J. Rumbaugh, I. Jacobson and G. Booch, “The Unified Modeling Language Reference Manual”, Addison Wesley Publishing Company, 1999.

[8] K. Vanichbuncha, Multivariate analysis using SPSS for Windows, Chulalongkorn University Press, 2001.

[9] L. C. Briand, C. Bunse, and J. W. Daly, “A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs”, IEEE Transactions on Software Engineering, Vol, 27, No.6, pp. 513-530, June 2001.

[10] M. Genero, J. Olivas, M. Piattini, and F. Romero, “Using Metrics to predict OO Information Systems Maintainability”, Proc. of the 13th International Conference on Advanced Information Systems Engineering (CAiSE 2001), Interlaken, Switzerland, June 4-8, 2001.

[11] M. Genero, M. Piattini and C. Calero, “Early Measures for UML Class Diagrams”, L’ Object, Hermes Science Publications, Vol.6 No.4, pp.489-515, 2000.

[12] M. Genero, M. Piattini, C. Calero, “An Empirical Study to Validate Metrics for Class Diagrams”, Proc. of International Database Engineering and Applications Symposium (IDEAS’02), Edmonton, Canada, July 17-19,2002.

[13] M. Lorenz and J. Kidd, Object-Oriented Software Metrics: A Practical Guide. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[14] M. Marchesi, “OOA Metrics for the Unified Modeling Language”, Proceedings of the 2nd Euromicro Conference on Software maintenance an reengineering, pp. 67-73,1998.

[15] R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Prentice Hall International, 1988.

[16] S. Chidamber and C. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, pp. 476-4936, June 1994.

[17] S. K. Kachigan, Multivariate statistical analysis : A Conceptual Introduction, Radius Press, New York, 1982.

[18] S. Schach, Software Engineering, Irwin Publishers, 1996.

[19] V.R. Basili, L.C. Briand, and W.L.Melo, “A Validation of Object-Oriented Design Metrics as Quality Indicators”, IEEE Trans. Software Eng., Vol.22. No.10, pp. 751-761, October 1996.

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวนงเยาว์ จินดาสวัสดิ์ เกิดวันที่ 11 เมษายน พ.ศ.2522 สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) เกียรตินิยมอันดับ 1 สาขาวิทยาการคอมพิวเตอร์ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เมื่อปีการศึกษา 2543 และเข้าศึกษาต่อระดับปริญญาโท ปีการศึกษา 2544 หลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย