การวัดซอฟต์แวร์เชิงวัตถุ

นงเยาว์ จินดาสวัสดิ์[1] นครทิพย์ พร้อมพูล[2] เชษฐ พัฒโนทัย[2] 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 จุฬา ลงกรณ์มหาวิทยาลัย 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 การกำหนดค่าให้กับมาตรวัดของ

คุณลักษณะต่างๆ 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 การวัดขนาดของซอฟต์แวร์ที่ผลิตได้ ซึ่งเป็นคุณลักษณะภายในสามารถวัดได้จากจำนวนบรรทัดทั้งหมดในโปรแกรม 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) รายละเอียดของมาตรวัดแต่ละประเภท มีดังนี้

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) มาตรวัดจำนวนเมธอดต่อคลาส (Weighted methods per class – WMC)

ในขั้นตอนของการวิเคราะห์และออกแบบระบบเชิงวัตถุสามารถคำนวณค่ามาตรวัดจำนวนเมธอดต่อคลาสได้ เพื่อนำค่ามาตรวัดนี้มาใช้ในการประมาณเวลา (Time) และความพยายาม (Effort) ที่ต้องใช้ในการพัฒนาซอฟต์แวร์ รวมถึงแก้ไข และบำรุงรักษาซอฟต์แวร์ได้ ค่ามาตรวัดจำนวนเมธอดต่อคลาสได้จากผลรวมของการคำนวณค่าความซับซ้อนของมาตรวัดไซโคลเมติกของแมคเคบทุกเมธอดภายในแต่ละคลาส

2) มาตรวัดความรับผิดชอบของคลาส (Response for a class – RFC)

ความรับผิดชอบของคลาส คือ จำนวนเมธอดภายในคลาสที่สามารถตอบสนองการเรียกใช้งานจากคลาสอื่นได้ หรือจากเมธอดบางตัวภายในคลาสนั้น มาตรวัดนี้แสดงค่าผลรวมของความซับซ้อนผ่านจำนวนเมธอดและจำนวนการเรียกใช้งานจากคลาสอื่น ถ้ามีเมธอดที่ถูกเรียกใช้งานได้เป็นจำนวนมากจะส่งผลให้คลาสมีความซับซ้อน 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 มาตรวัดจำนวนคลาสลูกเป็นคุณลักษณะที่สำคัญต่อการวัดคุณภาพในด้านความสามารถในการนำกลับมาใช้ใหม่เช่นเดียวกับมาตรวัดระดับความลึกของการสืบทอดคุณสมบัติของคลาส

5) มาตรวัดการเข้าคู่กันระหว่างวัตถุ (Coupling between objects – CBO)

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

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

การเกาะกันเป็นก้อน (Cohesion) ของคลาส คือ การที่จะบอกว่าเมธอดของคลาสนั้นมีความสัมพันธ์กับเมธอดอื่นๆ 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 จำนวนเมธอดที่ประกาศเป็น public ภายในคลาสนั้น ซึ่งการประกาศเมธอดเป็น public นั้นจะทำให้คลาสอื่นสามารถเข้ามาใช้งานเมธอดนั้นได้
2) มาตรวัด Number of Instance Methods in a Class (NIM) คือ จำนวนของเมธอดที่ประกาศภายในคลาสนั้น ซึ่งจะนับเมธอดที่มีการประกาศเป็นแบบ 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 จำนวนของคลาสเมธอดที่ประกาศในคลาสนั้น
5) มาตรวัด Number of class Variables in a Class (NVC) คือ จำนวนของตัวแปรประเภทคลาสที่ประกาศอยู่ในคลาสนั้น

มาตรวัดการสืบทอดของคลาส

1) มาตรวัด Number of Methods Overridden (NMO) คือ จำนวนของเมธอดที่ทำการ Override โดยคลาสลูก
2) มาตรวัด Number of Methods Inherited (NMI) คือ จำนวนเมธอดที่ถูก Inherit โดยคลาสลูก
3) มาตรวัด Number of Methods Added (NMA) คือ จำนวนเมธอดที่สร้างขึ้นใหม่ภายในคลาสลูก
4) มาตรวัด Specialization Index (SIX) คือ มาตรวัดที่เกิดจากการคำนวณจำนวนของเมธอดที่ถูก override คูณด้วยจำนวน Hierarchy หารด้วยจำนวนเมธอดทั้งหมด
5) มาตรวัด Average Parameters Per Method (APPM) คือ อัตราส่วนระหว่างจำนวนพารามิเตอร์ในทุกๆ เมธอดกับจำนวนเมธอดทั้งหมด

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 อัตราส่วนระหว่างผลรวมของเมธอดที่ประกาศเป็น private กับจำนวนเมธอดทั้งหมดในระบบ
2) มาตรวัด Attribute Hiding Factor (AHF) คือ อัตราส่วนระหว่างผลรวมของแอทริบิวต์ที่ประกาศเป็น private กับจำนวนแอทริบิวต์ทั้งหมดในระบบ
3) มาตรวัด Method Inheritance Factor (MIF) คือ อัตราส่วนระหว่างผลรวมของเมธอดที่ Inherit มาทั้งหมดในทุกๆ คลาสกับจำนวนเมธอดทั้งหมดในทุกๆ 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 จำนวนเมธอดในคลาสที่กำลังพิจารณา โดยไม่มีการถ่วงค่าให้กับ modifier ของเมธอดที่เป็น Public, Protected และ Private
4) มาตรวัด Number of the operations in a class – weighted (NOPC2) คือ จำนวนเมธอดภายในคลาสที่กำลังพิจารณา โดยมีการถ่วงน้ำหนักให้กับค่า modifier ของเมธอดที่เป็น 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)

ในการประเมินผลิตผลของซอฟต์แวร์ที่สามารถผลิตได้ นิยมใช้มาตรวัดจำนวนบรรทัด 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) ที่เราต้องการวัด จะประกอบไปด้วยเกณฑ์ (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) และ มาตรวัดจำนวนคลาสลูก (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

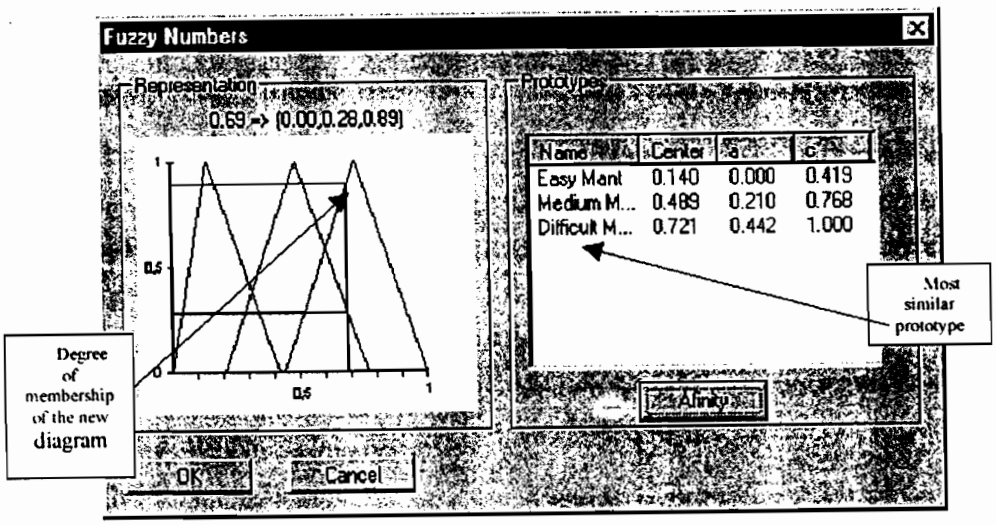
4.3.3 ผลงานวิจัยของ *Marcela Genero and Mario Piattini* [9][10][11]

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

การวัดระดับความสามารถในการบำรุงรักษาซอฟต์แวร์ของงานวิจัยนี้เป็นการเพิ่มข้อมูลการตัดสินใจในการปรับปรุงแผนภาพคลาสให้กับผู้พัฒนาซอฟต์แวร์ เพื่อให้พัฒนาซอฟต์แวร์ที่มีคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์เพิ่มมากขึ้น

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

ผลงานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านการบำรุงรักษาซอฟต์แวร์ ซึ่งทำการทดลองเพื่อศึกษาผลกระทบเมื่อนำหลักการออกแบบเชิงคุณภาพมาประยุกต์ใช้ ซึ่งพิจารณาถึงความสามารถในการบำรุงรักษาของการออกแบบเชิงวัตถุ โดยพิจารณาคุณลักษณะย่อยที่ประกอบด้วยความสามารถในการทำความเข้าใจ 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 มุ่งเน้นวัดคุณภาพในด้าน ความสามารถในการนำกลับมาใช้ใหม่, ความน่าเชื่อถือ และความสามารถในการบำรุงรักษาซอฟต์แวร์ และกำลังดำเนินการศึกษาคุณภาพในด้านอื่นๆ ผลงานวิจัยของกลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุ ได้แก่

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

งานวิจัยนี้เกี่ยวข้องกับคุณภาพในด้านความสามารถในการนำกลับมาใช้ใหม่ภายใน (Internal Reuse) โดยพิจารณาจากจำนวนครั้งของการเรียกใช้งานซ้ำของเมธอดที่สร้างขึ้นสำหรับใช้งานภายในซอฟต์แวร์ชิ้นหนึ่งๆ การวัดการนำกลับมาใช้ภายในคลาสใดๆ จะเกิดจากผลรวมของการนำกลับมาใช้ใหม่ภายในของทุกๆ เมธอดของคลาสนั้นๆ และการนำกลับมาใช้ใหม่ภายในของซอฟต์แวร์จะเกิดจากผลรวมของการนำกลับมาใช้ใหม่ภายในของทุกๆ คลาสในซอฟต์แวร์ งานวิจัยนี้ได้แบ่งกลุ่มเมธอดออกเป็น 2 กลุ่มได้แก่

- กลุ่ม Private คือ เมธอดที่มีตัวขยาย (modifier) เป็น Private เมธอดในกลุ่มนี้จะมีการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมธอด เฉพาะในส่วนของการนำกลับมาใช้ใหม่ภายในที่ไม่ผ่านกลไกการสืบทอดคุณสมบัติ
- กลุ่ม Public คือเมธอดที่มีตัวขยาย เป็น Public และ Protected เมธอดในกลุ่มนี้จะมีการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมธอด ทั้งในส่วนของการนำกลับมาใช้ใหม่ภายในที่ผ่านกลไกการสืบทอดคุณสมบัติ และการนำกลับมาใช้ใหม่ภายในที่ไม่ผ่านกลไกการสืบทอดคุณสมบัติ

มาตรวัดการนำกลับมาใช้ใหม่ภายในที่เกี่ยวข้องกับเมธอด M ของคลาส X ใดๆ สามารถแสดงได้ดังรูปที่ 4 โดย

NOI คือ จำนวนครั้งที่เมธอดอื่นถูกเรียกใช้ทั้งแบบโดยตรงและโดยอ้อมจากเมธอด M

a คือ จำนวนครั้งที่เมธอด M ถูกเรียกใช้โดยไม่ผ่านกลไกของการสืบทอดคุณสมบัติ

n คือ จำนวนของ descendant class ของคลาส X ที่ไม่ redefine เมธอด 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 จำนวนครั้งที่เมธอด M ถูกเรียกใช้ผ่านกลไกของการสืบทอดคุณสมบัติ จาก instance ของ descendant class ของคลาส X ที่ไม่ redefine เมธอด 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

$D = -3.348 - 1.594WMC + 2.914*10^{-4}LCOM + 4.151DIT - 0.159NOC - 6.678*10^{-2}NOA - 1.361NOHA + 1.863NOTA + 3.645NOHC + 2.022NOTC + 1.627NOHG - 2.294NMSC + 1.965NMRC + 0.204NOATPB$

สมการสำหรับคลาสที่มีข้อผิดพลาด

$D = -3.942 + 0.16WMC + 1.869*10^{-4}LCOM + 4.828DIT - 0.23NOC + 0.954NOA - 3.674*10^{-2}NOHA + 2.746NOTA - 0.216NOHC + 1.483NOTC + 0.783NOHG - 3.141NMSC + 1.057NMRC + 0.569NOATPB$

จากทั้ง 2 สมการเมื่อแทนค่ามาตรวัดลงในสมการ ถ้าค่าในสมการแรกมีค่ามากกว่าค่าในสมการที่สองแสดงว่าคลาสที่นำมาพิจารณานั้นไม่มีข้อผิดพลาดเกิดขึ้น แต่ถ้าค่าในสมการแรกมีค่าน้อยกว่าค่าในสมการที่สองแสดงว่าคลาสที่นำมาพิจารณานั้นมีข้อผิดพลาดเกิดขึ้น

ทั้งนี้เพื่อความสะดวกในการวัดคุณภาพความน่าเชื่อถือ และการคำนวณค่ามาตรวัดต่างๆ ของงานวิจัยนี้ ขณะนี้กลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุกำลังพัฒนาเครื่องมือเพื่อช่วยในการวัดค่าระดับความผิดพลาดของคลาส

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

งานวิจัยนี้วัดคุณภาพในด้านความสามารถในการบำรุงรักษาซอฟต์แวร์ ซึ่งพิจารณาตามความสามารถในการทำความเข้าใจได้ และความสามารถในการปรับเปลี่ยน และพิจารณามาตรวัดสำหรับแผนภาพคลาสและแผนภาพซีเควนซ์ในขั้นตอนการออกแบบระบบจำนวน 18 มาตรวัด สำหรับสร้างโมเดลการทำนายความสามารถในการบำรุงรักษาซอฟต์แวร์ 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

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

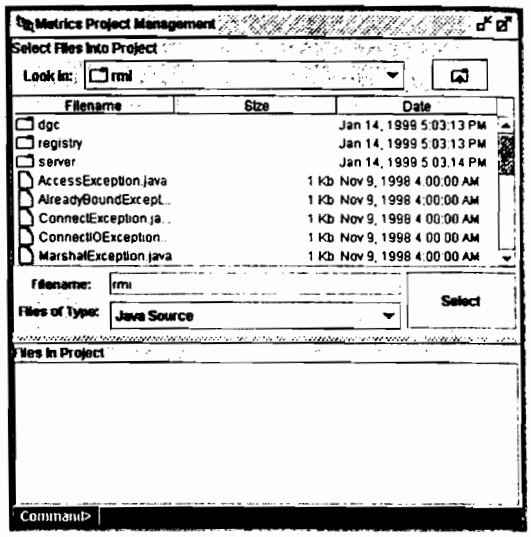
ประโยชน์ของงานวิจัยนี้คือนักวิเคราะห์, นักออกแบบ และนักพัฒนาซอฟต์แวร์สามารถนำโมเดลนี้ไปใช้วัดความสามารถในการทำความเข้าใจได้ของซอฟต์แวร์ในขั้นตอนการวิเคราะห์และออกแบบระบบเชิงวัตถุ และช่วยเพิ่มข้อมูลการตัดสินใจว่าควรแก้ไขแผนภาพคลาสและซีเควนซ์ก่อนนำไปพัฒนาเป็นซอฟต์แวร์หรือไม่ โดยในการวัดระดับความสามารถในการทำความเข้าใจได้นั้น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

5. เครื่องมือคำนวณมาตรวัดซอฟต์แวร์เชิงวัตถุ

ในปัจจุบันเครื่องมือวัดซอฟต์แวร์เชิงวัตถุแบบอัตโนมัติ JMetric [1] และ JavaNCSS เป็นเครื่องมือที่นำมาใช้ในการคำนวณหาค่ามาตรวัดขนาด และมาตรวัดคุณภาพ 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 ช่วยในการ

พัฒนา และมีคลาสสวิงที่ช่วยในการสร้างส่วนติดต่อกับผู้ใช้ (Graphic User Interface – GUI)

6.1 มาตรวัดที่สามารถคำนวณได้จากเครื่องมือ MTOOP รุ่นที่ 1

จำแนกประเภทของมาตรวัดออกเป็น 5 ประเภท ได้แก่

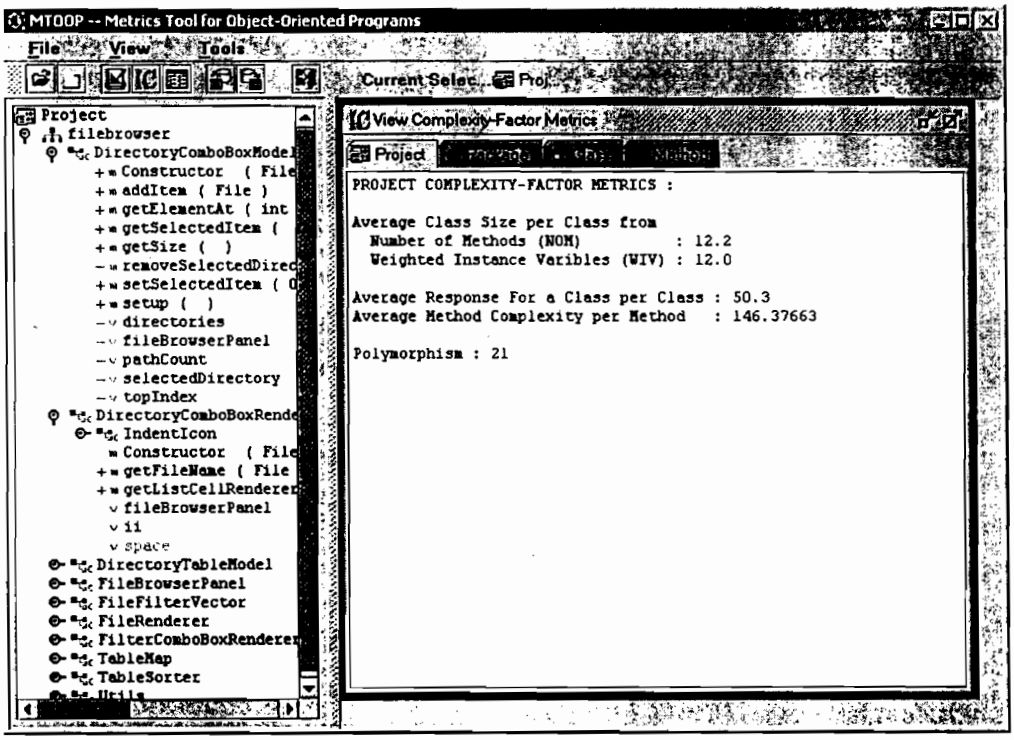
- มาตรวัดสำหรับโปรเจค ได้แก่ มาตรวัดจำนวนแพ็คเกจ, จำนวนคลาส, จำนวนเมธอด, จำนวนเมธอดต่อคลาส, จำนวนบรรทัด, จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนท์

- มาตรวัดสำหรับแพ็คเกจ ได้แก่ มาตรวัดจำนวนคลาส, จำนวนเมธอด, จำนวนบรรทัด, จำนวนสเตทเมนต์ และจำนวนตัวแปรอินสแตนท์

- มาตรวัดสำหรับคลาส ได้แก่ มาตรวัดระดับความลึกของการสืบทอดคุณสมบัติ, จำนวนเมธอด, จำนวนบรรทัด, จำนวนสเตทเมนต์, จำนวนตัวแปรอินสแตนท์ และการขาดความสัมพันธ์ภายในคลาส

- มาตรวัดสำหรับเมธอด ได้แก่ จำนวนพารามิเตอร์, จำนวนบรรทัด, จำนวนสเตทเมนต์, ค่าไซโคลเมติกของแมคเคบ, จำนวนตัวแปรของแต่ละเมธอด และขนาดความสัมพันธ์ระหว่างวัตถุ

- มาตรวัดสำหรับตัวแปรอินสแตนท์ ได้แก่ จำนวนครั้งที่ตัวแปรอินสแตนท์ถูกเรียกใช้ และจำนวนเมธอดที่เรียกใช้ตัวแปรอินสแตนท์



รูปที่ 7 แสดงหน้าจอการดูค่ามาตรวัด (View Metrics) [22]

6.2 มาตรวัดที่สามารถคำนวณได้จากเครื่องมือ MTOOP รุ่นที่ 2

จำแนกประเภทของมาตรวัดออกเป็น 5 ประเภท เช่นกัน ซึ่งจะแสดงเฉพาะมาตรวัดที่เพิ่มเข้ามาได้แก่

- มาตรวัดสำหรับโปรเจค ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งโปรเจค แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนต์ และค่าถ่วงน้ำหนักของตัวแปรอินส-แตนท์, ค่าเฉลี่ยของความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมธอด และค่าโพลีมอร์ฟิซึม

- มาตรวัดสำหรับแพ็คเกจ ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งแพ็คเกจ แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนท์ และค่าถ่วงน้ำหนักของตัวแปรอินสแตนท์, ค่าเฉลี่ยของความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมธอด และค่าโพลีมอร์ฟิซึม

- มาตรวัดสำหรับคลาส ได้แก่ มาตรวัดขนาดของคลาสเฉลี่ยทั้งแพ็คเกจ แบ่งเป็น ขนาดของคลาสที่วัดจากจำนวนสเตทเมนท์ และค่าถ่วงน้ำหนักของตัวแปรอินสแตนท์, ค่าความรับผิดชอบของคลาส, ค่าเฉลี่ยของความซับซ้อนของเมธอด และค่าโพลีมอร์ฟิซึม

- มาตรวัดสำหรับเมธอด ได้แก่ มาตรวัดขนาดของเมธอด และค่าความซับซ้อนของเมธอด

- มาตรวัดสำหรับตัวแปรอินสแตนท์ เป็นมาตรวัดเช่นเดียวกับเครื่องมือ MTOOP รุ่นที่ 1

ในรูปที่ 7 แสดงหน้าจอของเครื่องมือที่ใช้วัดคุณภาพโปรแกรมเชิงวัตถุ MTOOP รุ่นที่ 2

6.3 หลักการพัฒนาเครื่องมือวัด MTOOP

การคำนวณค่ามาตรวัดเริ่มจากสร้างคอมไพเลอร์ ที่ทำหน้าที่นำโปรแกรมต้นฉบับมาผ่านขบวนการสร้างตัวแปลภาษาเพื่อสร้างซินแท็กซ์ทรี โดยงานวิจัยนี้ได้เลือกเครื่องมือช่วยสร้างตัวแปลภาษาที่ชื่อว่า จาวาคอมไพเลอร์ คอมไพเลอร์ (Java Compiler Compiler-JavaCC) การแปลภาษาโปรแกรมต้นฉบับ มีขั้นตอนดังต่อไปนี้

6.3.1 ขั้นตอนเลกซิคัลอนาไลซิส (Lexical analysis)

ในขั้นตอนนี้จะมีหน่วยย่อยของตัวแปลภาษาที่เรียกว่า เลกซิคัลอนาไลเซอร์ (Lexical analyzer) หรือเรียกอีกชื่อหนึ่งว่า สแกนเนอร์ (Scanner) มีหน้าที่ทำการอ่านอักขระจากภาษาต้นแบบแล้วจัดการแยกกลุ่มอักขระเหล่านั้นออกเป็น โทเคน (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) ซึ่งเป็นโครงสร้างต้นไม้ที่มีการลดทอนให้เหลือแค่ส่วนของโทเคนที่จำเป็นต้องใช้ในการสร้างโค้ดเท่านั้น ในแต่ละโหนด (Node) ของเอเอสทีจะเก็บข้อมูลหรือคุณสมบัติต่างๆ ที่อยู่ภายในโหนด เช่น เป็นคลาส เป็นเมธอด 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) ทำหน้าที่ท่องไปบนซินแท็กซ์ทรี เพื่อเก็บข้อมูลต่างๆ จากโหนด โดยเก็บอยู่ในรูปแบบของตัวแปรแบบเวคเตอร์

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 และมีส่วนช่วยในการประเมินสถานะของโครงงานและการตรวจสอบคุณภาพซอฟต์แวร์ในด้านต่างๆ 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 และขณะนี้กลุ่มงานวิจัยมาตรวัดซอฟต์แวร์เชิงวัตถุกำลังศึกษาการวัดคุณภาพซอฟต์แวร์ในด้านอื่นๆ 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

นงเยาว์ จินดาสวัสดิ์ การศึกษาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) เกียรตินิยมอันดับ 1 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2543, ปัจจุบันเป็นนิสิตปริญญาโท สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

นครทิพย์ พร้อมพูล การศึกษา บริหารธุรกิจบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย, วิทยาศาสตรมหาบัณฑิต George Washington University, U.S.A.

เชษฐ พัฒโนทัย การศึกษาวิศวกรรมศาสตรบัณฑิต (วิศวกรรมไฟฟ้า) มหาวิทยาลัยขอนแก่น 2532, วิทยาศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) University of Miami, U.S.A., 2538

พรศิริ หมื่นไชยศรี การศึกษาวิทยาศาสตรบัณฑิต (สถิติ) มหาวิทยาลัยเชียงใหม่, สถิติศาสตรมหาบัณฑิต วิทยาศาสตรมหาบัณฑิต (วิทยาศาสตร์คอมพิวเตอร์) Oregon State University, U.S.A., 2537, ปรัชญาดุษฎีบัณฑิต (วิศวกรรมซอฟต์แวร์) Oregon State University, U.S.A., 2541