บทที่ 5

การออกแบบและพัฒนาเครื่องมือคำนวณความเสถียรเชิงตรรกะและเครื่องมือประมาณค่าความเสถียรเชิงตรรกะ

เครื่องมือคำนวณความเสถียรเชิงตรรกะของงานวิจัยนี้ได้ทำการพัฒนาขึ้น โดยการแก้ไขเครื่องมือคำนวณเสถียรภาพ ChangeImpact[18] โดยแก้ไขวิธีการคำนวณ และเพิ่มส่วนต่อประสานผู้ใช้ให้ง่ายต่อการใช้งาน โดยรายละเอียดของการแก้ไขและรายละเอียดของโปรแกรมจะอยู่ในหัวข้อที่ 5.1.2

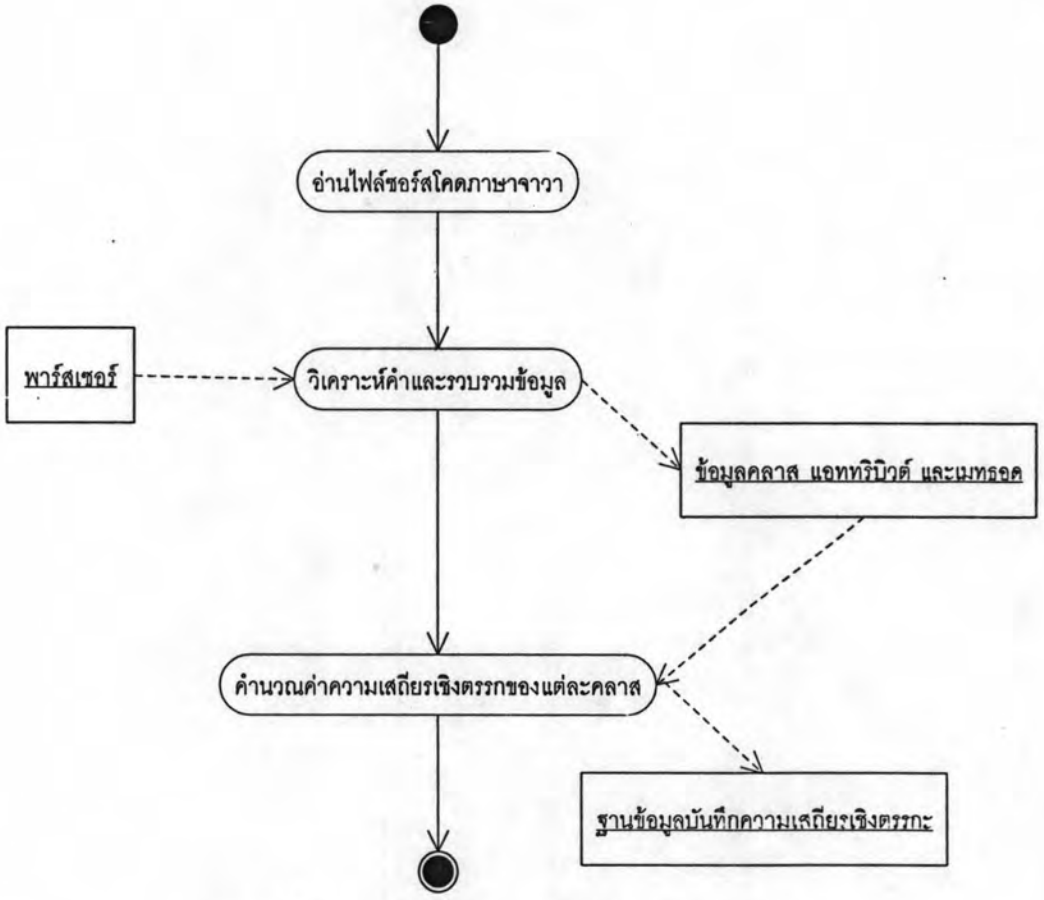
สำหรับเครื่องมือการประมาณค่าความเสถียรเชิงตรรกะจากตัววัดแผนภาพตามโมเดลที่ได้ของงานวิจัยนี้จะอาศัยโปรแกรม SDMetrics[14] ที่ใช้ในการวัดค่าตัววัดแผนภาพในขั้นตอนที่ 3.3 โดยโปรแกรม SDMetrics มีคุณสมบัติในการอ่านข้อมูลแผนภาพที่ได้รับการแปลงให้อยู่ในรูปแบบเอกซ์เอมแอล แล้วทำการคำนวณค่าข้อมูลต่างๆ ของแผนภาพเพื่อนำไปใช้ในการวัดค่าตัววัดแผนภาพต่อไป และอนุญาตให้ผู้ใช้สามารถทำการแก้ไขเพิ่มเติมวิธีการอ่านค่า และการคำนวณต่างๆ ได้ รายละเอียดของโปรแกรมและรายละเอียดการแก้ไขโปรแกรมอยู่ในหัวข้อที่ 5.2

5.1 การออกแบบและพัฒนาเครื่องมือคำนวณความเสถียรเชิงตรรกะ

เครื่องมือคำนวณความเสถียรเชิงตรรกะที่พัฒนาขึ้นโดยการแก้ไข และปรับปรุงเครื่องมือคำนวณเสถียรภาพ ChangeImpact[18] โดยทำการแก้ไขขั้นตอนและวิธีการคำนวณให้สอดคล้องกับ รูปแบบการเปลี่ยนแปลงในตารางที่ 3.1

5.1.1 โครงสร้างของเครื่องมือคำนวณเสถียรภาพ

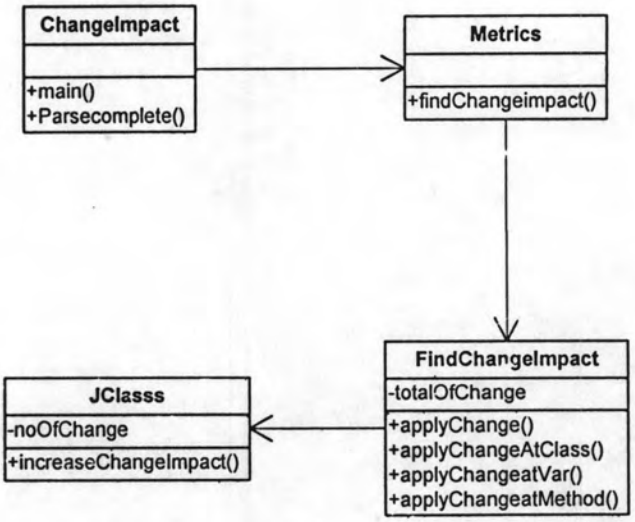
เครื่องมือคำนวณเสถียรภาพที่นำมาพัฒนา มีการทำงานโดยจะทำการอ่านไฟล์ซอร์สโคดภาษาจาวาที่ระบุไว้ แล้วทำการวิเคราะห์และเก็บรวบรวมข้อมูลของแต่ละคลาส โดยอาศัยพาร์สเซอร์ภาษาจาวาในการวิเคราะห์แล้วจึงแยกเก็บข้อมูลของแต่ละคลาส แอททริบิวต์ และเมทธอด จากนั้นจึงทำการวัดค่าความเสถียรเชิงตรรกะจากชุดข้อมูลที่รวบรวมได้ แล้วบันทึกผลลัพธ์ที่ได้ลงฐานข้อมูลที่ได้กำหนดไว้ แผนภาพกิจกรรมการทำงานของเครื่องมือคำนวณเสถียรภาพแสดงได้ดังรูปที่ 5.1



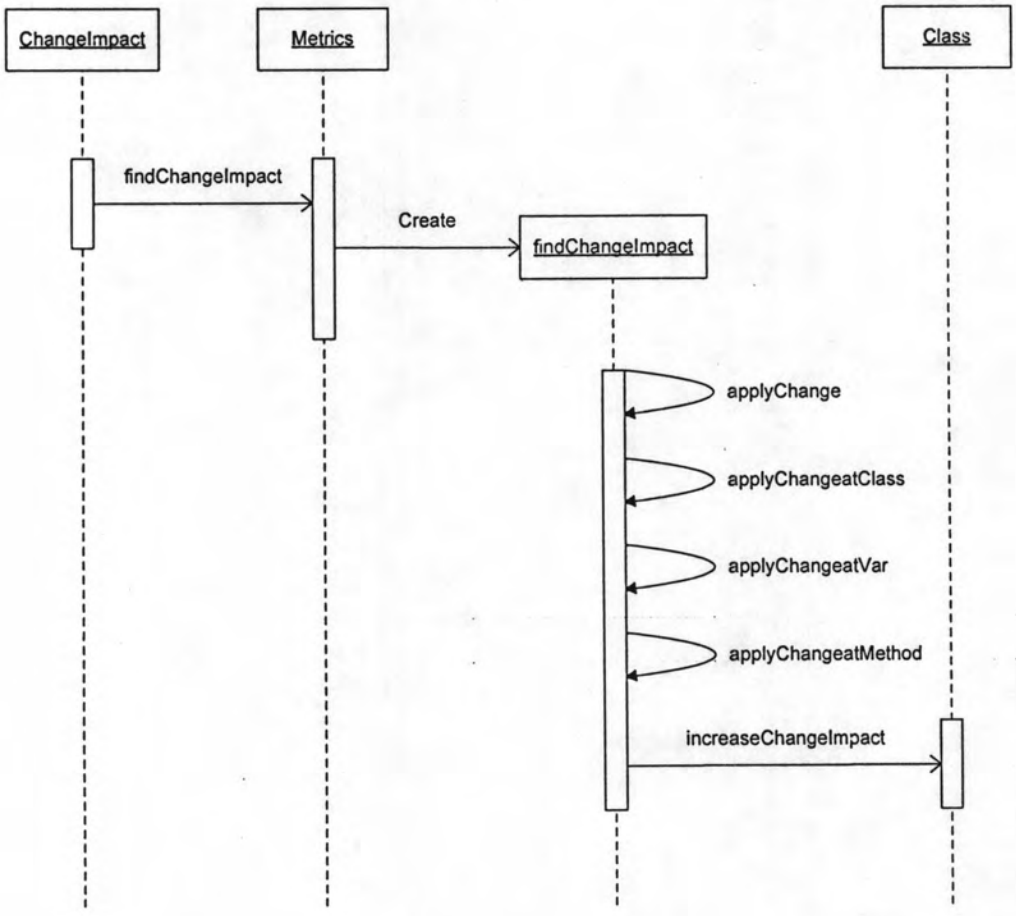
รูปที่ 5.1 การทำงานของเครื่องมือคำนวณเสถียรภาพ

งานวิจัยนี้ได้ทำการแก้ไขวิธีการในการคำนวณความเสถียรเชิงตรรกะของเครื่องมือคำนวณเสถียรภาพเดิม โดยมีรายละเอียดในหัวข้อที่ 5.1.2

รูปที่ 5.2 แสดงแผนภาพคลาสของเครื่องมือคำนวณเสถียรภาพเฉพาะในส่วนของการคำนวณค่าความเสถียร โดยมีคลาส ChangeImpact เป็นคลาสหลักเมื่อมีการเรียกใช้งานจะทำการอ่านรายชื่อไฟล์ซอร์สโคดจากไฟล์ที่กำหนดแล้วทำการวิเคราะห์คำ เมื่อทำการวิเคราะห์เสร็จสิ้นจะทำการเก็บข้อมูลของคลาสทั้งหมดไว้ในคลาส metrics จากนั้นทำการตรวจสอบผลกระทบผ่านทางคลาส metrics ซึ่งจะทำการสร้างคลาส FindChangeImpact ขึ้นและทำการตรวจสอบผลกระทบพร้อมทั้งเพิ่มค่าจำนวนครั้งที่แต่ละคลาสได้รับผลกระทบ แล้วจึงทำการคำนวณความเสถียรเชิงตรรกะและทำการบันทึกลงฐานข้อมูล ขั้นตอนการทำงานเพื่อคำนวณความเสถียรเชิงตรรกะของเครื่องมือคำนวณเสถียรภาพแสดงได้โดยแผนภาพซีเควนซ์ในรูปที่ 5.3



รูปที่ 5.2 แผนภาพคลาสของเครื่องมือคำนวณเสถียรภาพเฉพาะส่วนที่เกี่ยวข้อง



รูปที่ 5.3 แผนภาพซีเควนซ์การทำงานของเครื่องมือคำนวณเสถียรภาพ

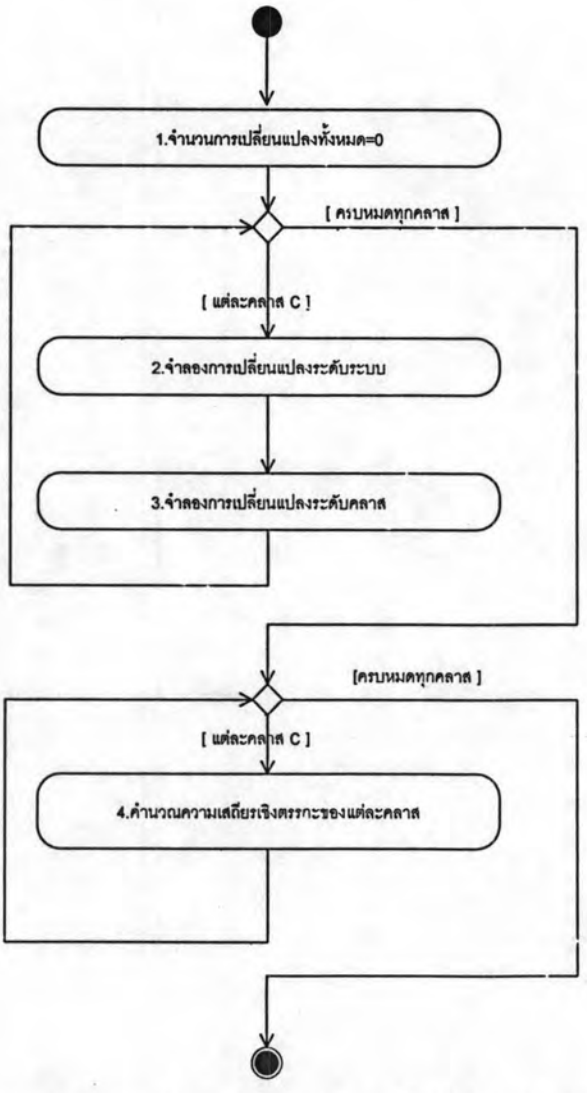
5.1.2 การพัฒนาเครื่องมือคำนวณความเสถียรเชิงตรรกะ

การพัฒนาเครื่องมือคำนวณความเสถียรเชิงตรรกะในงานวิจัยนี้แบ่งเป็น 2 ส่วนดังนี้

1. การพัฒนากระบวนการคำนวณความเสถียรเชิงตรรกะ
2. การพัฒนาส่วนต่อประสานผู้ใช้และการเลือกไฟล์ซอร์สโคด

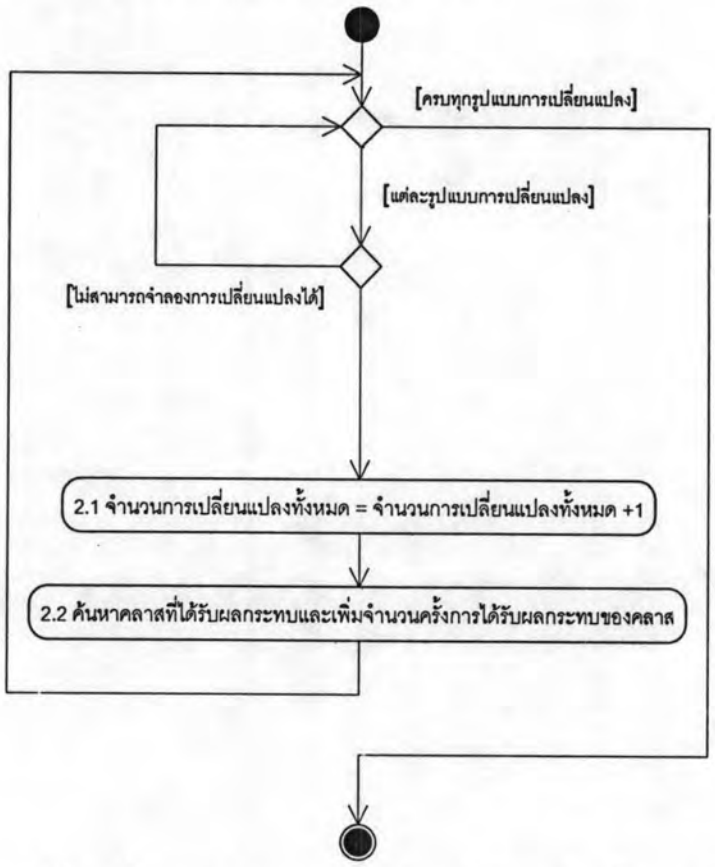
5.1.2.1 การพัฒนากระบวนการคำนวณความเสถียรเชิงตรรกะ

การพัฒนากระบวนการคำนวณความเสถียรเชิงตรรกะกระทำโดยทำการแก้ไขคลาส FindChangeImpact ของเครื่องมือคำนวณเสถียรภาพ ให้มีขั้นตอนการทำงานตามแผนภาพกิจกรรมการคำนวณความเสถียรเชิงตรรกะในรูปที่ 5.4 ในส่วนของการจำลองรูปแบบการเปลี่ยนแปลง



รูปที่ 5.4 ขั้นตอนการคำนวณความเสถียรเชิงตรรกะของเครื่องมือคำนวณความเสถียรเชิงตรรกะ

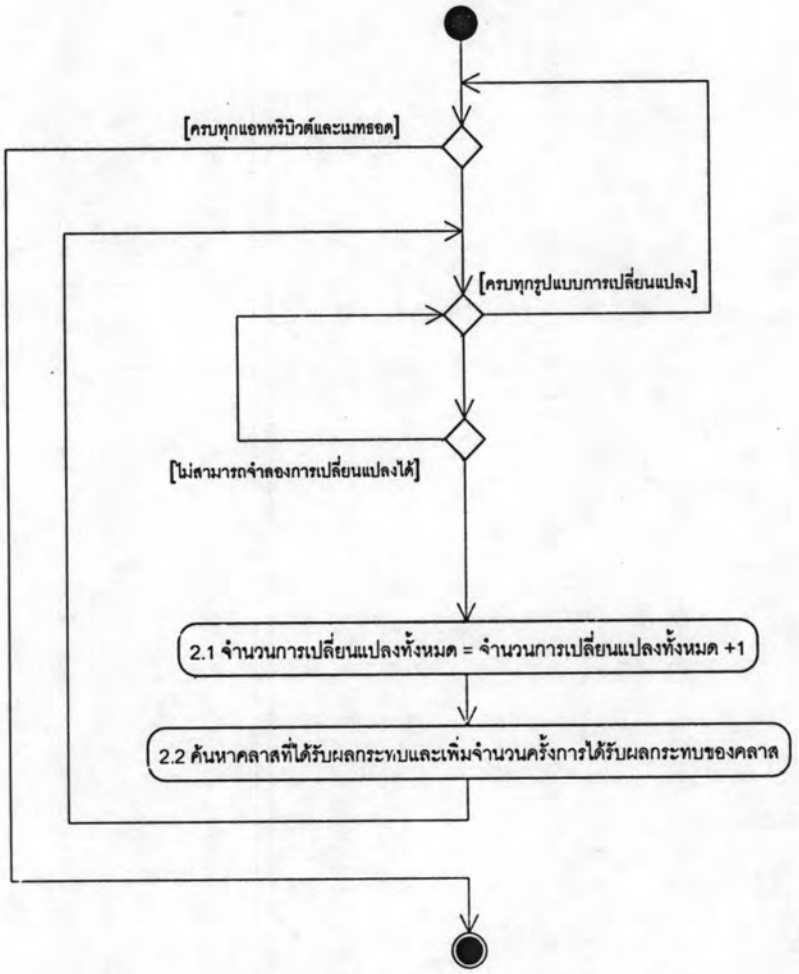
จากรูปที่ 5.4 ขั้นตอนการคำนวณความเสถียรเชิงตรรกะเริ่มโดยทำการเลือกคลาสขึ้นมาหนึ่งคลาสจากทั้งหมด จากนั้นทำการจำลองการเปลี่ยนแปลงระดับของระบบให้แก่คลาสตามการเปลี่ยนแปลงที่ 1-14 ในตารางที่ 3.1 แล้วจึงทำการจำลองการเปลี่ยนแปลงในระดับคลาสตามการเปลี่ยนแปลงที่ 15-27 ในตารางที่ 3.1 จากนั้นเลือกคลาสถัดไปมาทำการจำลองทั้งระดับของระบบและระดับคลาสเช่นเดียวกันและทำซ้ำจนครบทุกคลาส เมื่อทำการจำลองการเปลี่ยนแปลงครบทุกคลาสแล้วจึงทำการคำนวณความเสถียรเชิงตรรกะของแต่ละคลาสตามสมการในหัวข้อ 2.1.2 รูปที่ 5.5 แสดงขั้นตอนการจำลองการเปลี่ยนแปลงในระดับระบบ



รูปที่ 5.5 ขั้นตอนการจำลองการเปลี่ยนแปลงในระดับของระบบ

การเปลี่ยนแปลงระดับคลาสในรูปที่ 5.4 จะประกอบไปด้วยการเปลี่ยนแปลงที่แอททริบิวต์และการเปลี่ยนแปลงที่เมทธอด ซึ่งมีลำดับการทำงานดังรูปที่ 5.6 โดยการเปลี่ยนแปลงที่ แอททริบิวต์คือการเปลี่ยนแปลงที่ 15-20 และการเปลี่ยนแปลงที่เมทธอดคือการเปลี่ยนแปลงที่ 21-27 ในตารางที่ 3.1 และทุกครั้งที่ทำการจำลองการเปลี่ยนแปลงจะต้องทำการค้นหาคลาสที่ได้รับผลกระทบจากการเปลี่ยนแปลงนั้นๆ เพื่อเพิ่มค่าจำนวนครั้งที่คลาสใดๆ ได้รับ

ผลกระทบจากการเปลี่ยนแปลง นอกจากนี้ต้องทำการเพิ่มค่าจำนวนการเปลี่ยนแปลงทั้งหมดเสมอดังจะเห็นได้จากรูปที่ 5.5 และรูปที่ 5.6

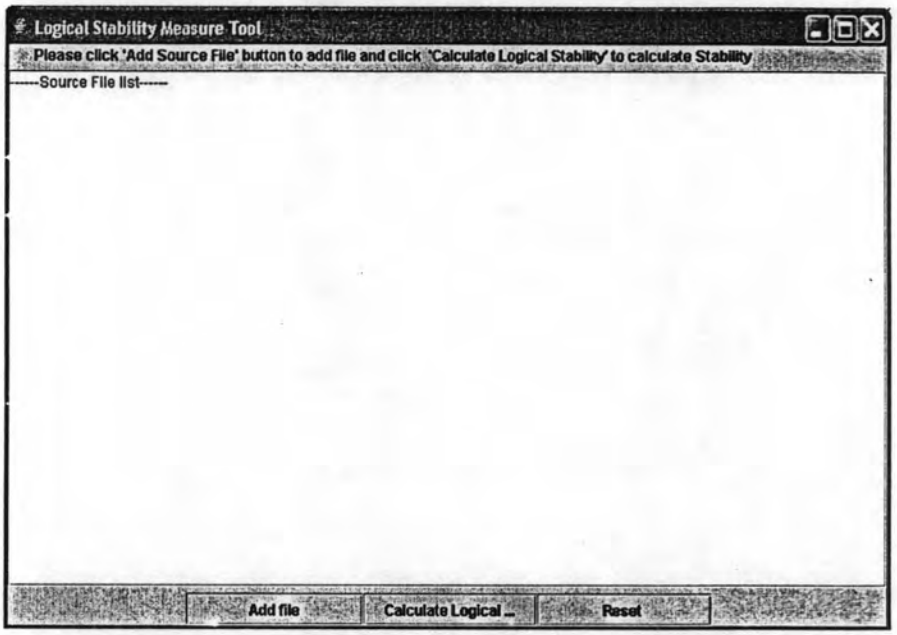


รูปที่ 5.6 ขั้นตอนการจำลองการเปลี่ยนแปลงระดับของคลาส

5.1.2.2 การพัฒนาส่วนต่อประสานผู้ใช้และการเลือกไฟล์ซอร์สโคด

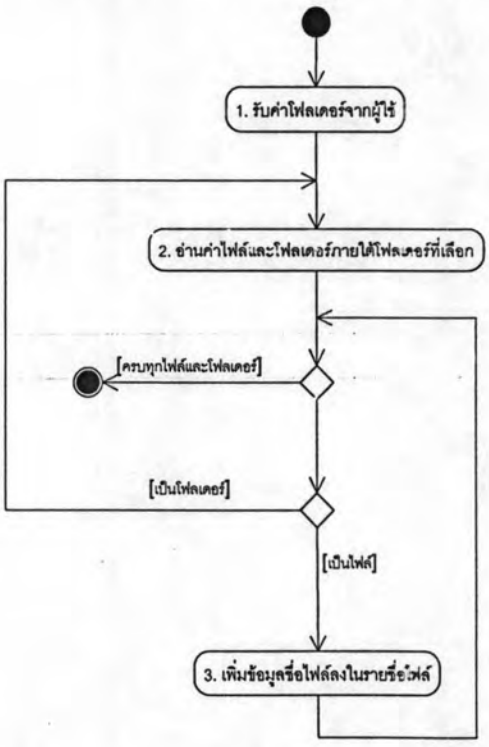
การพัฒนาส่วนต่อประสานผู้ใช้และการเลือกไฟล์ซอร์สโคดทำขึ้นเพื่ออำนวยความสะดวกในการเลือกไฟล์ของซอฟต์แวร์ที่ต้องการคำนวณความเสถียรเชิงตรรกะของแต่ละคลาสจากเดิมที่ต้องทำการแก้ไขข้อมูลในไฟล์ config.dat โดยให้มีส่วนต่อประสานผู้ใช้และให้ผู้ใช้เลือกเพียงไดเรกทอรีที่มีไฟล์ซอร์สโคดทั้งหมดอยู่ และโปรแกรมจะทำการอ่านไฟล์ในไดเรกทอรีนั้นและไดเรกทอรีย่อยด้วย

ส่วนต่อประสานผู้ใช้จะมีลักษณะดังรูปที่ 5.7 โดยรายละเอียดของส่วนต่อประสานและการใช้งานโปรแกรมคำนวณความเสถียรเชิงตรรกะอยู่ในภาคผนวก ง



รูปที่ 5.7 ส่วนต่อประสานผู้ใช้ของเครื่องมือคำนวณความเสถียรเชิงตรรกะ

การเลือกไฟล์ซอร์สโคดจะอาศัยการทำงานแบบวนกลับ (recursive) ดังแสดงในรูปที่ 5.8 โดยเมื่อผู้ใช้เลือกไดเรคทอรีที่ต้องการโปรแกรมจะดึงข้อมูลไฟล์และไดเรคทอรีที่อยู่ภายในไดเรคทอรีออกมาแล้วทำการตรวจสอบทีละไฟล์หรือไดเรกทอรี ถ้าเป็นไฟล์ให้ทำการเพิ่มข้อมูลในรายชื่อไฟล์ ถ้าเป็นไดเรกทอรีให้ทำการวนกลับตรวจสอบไดเรกทอรีต่อไปจนกว่าจะครบทั้งหมด



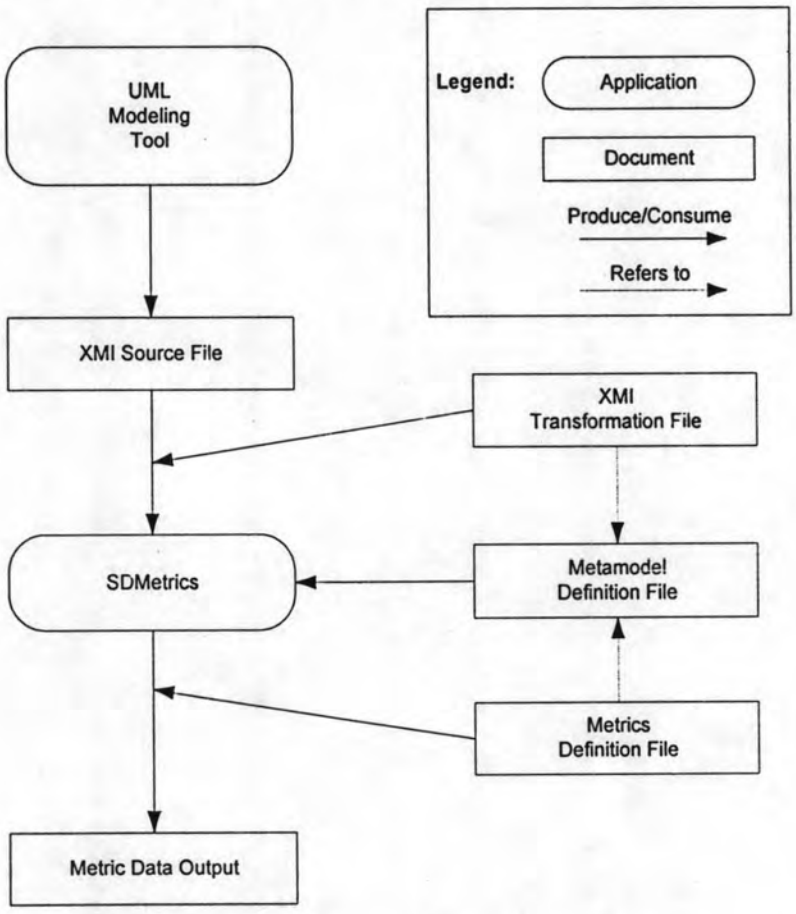
รูปที่ 5.8 ขั้นตอนการค้นหาไฟล์ทั้งหมดในไดเรกทอรีที่ต้องการ

5.2 การออกแบบและพัฒนาเครื่องมือประมาณค่าความเสถียรเชิงตรรกะ

การพัฒนาเครื่องมือคำนวณความเสถียรเชิงตรรกะของงานวิจัยนี้จะอาศัยโปรแกรม SDMetrics ซึ่งเป็นโปรแกรมที่มีความสามารถในการคำนวณค่าตัววัดแผนภาพยูเอ็มแอลที่อยู่ในรูปแบบของเอกซ์เอมแอลหรือเอกซ์เอมไอได้ และรองรับการแก้ไขเพิ่มเติมตัววัดได้

5.2.1 โครงสร้างการทำงานของโปรแกรม SDMetrics

โครงสร้างการทำงานของโปรแกรม SDMetrics แสดงได้ดังรูปที่ 5.9



รูปที่ 5.9 โครงสร้างการทำงานของโปรแกรม SDMetrics

จากรูปที่ 5.9 โปรแกรม SDMetrics จะสามารถทำการวัดค่าตัววัดแผนภาพได้โดยอาศัยข้อมูลจากไฟล์นิยาม 3 ไฟล์ได้แก่ XMI Transformation file Metamodel Definition file และ Metrics Definition file โดยโปรแกรม SDMetrics จะทำการวัดตัววัดแผนภาพโดยอ่านข้อมูลจากไฟล์เอกซ์เอมไอ แล้วทำการดึงข้อมูลที่รู้จักซึ่งระบุไว้ใน Metamodel Definition file โดยใช้วิธีการดึงข้อมูลตามที่ระบุไว้ใน XMI Transformation file จากนั้นทำการวัดตัววัดแผนภาพตามที่ระบุไว้

ใน Metrics Definition File จากนั้นทำการแสดงผลตัววัดแผนภาพที่วัดได้ โดยไฟล์นิยามทั้ง 3 มีรายละเอียดและโครงสร้างตามหลักภาษาเอกซ์เอมแอล

5.2.2 การอ่านค่าข้อมูลแผนภาพและคำนวณตัววัดแผนภาพ

การอ่านข้อมูลแผนภาพจากไฟล์เอกซ์เอมไอ และเอกซ์เอมแอลเพื่อนำมาใช้ในการคำนวณตัววัดแผนภาพนั้นสามารถอธิบายเป็นขั้นตอนได้ดังนี้

ขั้นตอนที่ 1 กำหนดข้อมูลที่ต้องการอ่าน

ขั้นตอนที่ 2 กำหนดวิธีการอ่านข้อมูลนั้นๆ

ขั้นตอนที่ 3 กำหนดวิธีการนำข้อมูลที่อ่านได้ไปใช้คำนวณ

การกำหนดข้อมูลที่ต้องการอ่านค่าในขั้นตอนที่ 1 นั้นทำได้โดยการกำหนดค่าลงในไฟล์ Metamodel Definition โดยไฟล์ Metamodel Definition คือไฟล์ที่เป็นตัวกำหนดชนิดและรายละเอียดของข้อมูลแผนภาพที่จำเป็นในการคำนวณตัววัดแผนภาพและแสดงผล ซึ่งมีรูปแบบโครงสร้างดังตารางที่ 5.1

ตารางที่ 5.1 โครงสร้างของไฟล์ Metamodel Definition

```
<sdmetricsmetamodel version="2.0" >
        <modelelement name="element1">
                <attribute name="attr1" type="data" multiplicity="one" />
                <attribute name="attr2" type="ref" multiplicity="many" />
                        ..
        </modelelement>
        <modelelement name="element2" parent="element1">
                ...
        </modelelement>
        ...
</sdmetricsmetamodel>
```

จากตารางที่ 5.1 โครงสร้างของไฟล์ Metamodel definition เป็นไปตามข้อกำหนดของภาษาเอกซ์เอมแอล โดยต้องมีการประกาศประเภทของข้อมูลในไฟล์ และเวอร์ชันของไฟล์เสียก่อนดังในบรรทัดแรก ซึ่งเป็นการประกาศประเภทของเอกสารเป็น เอกสาร sdmetricsmetamodel เวอร์ชัน 2 บรรทัดถัดมาเป็นการประกาศองค์ประกอบของโมเดล (Model Element) หรือข้อมูล

แผนภาพที่ต้องการโดยรายละเอียดข้อมูลที่ต้องการต้องประกาศภายใต้แท็ก <modelelement> และต้องทำการกำหนดชื่อให้แก่ทุกองค์ประกอบ

องค์ประกอบโมเดลทุกองค์ประกอบจะมีข้อมูลแอททริบิวต์ขององค์ประกอบ ซึ่งจะเก็บข้อมูลขององค์ประกอบ และข้อมูลอ้างอิงกับองค์ประกอบอื่นๆ การประกาศแอททริบิวต์ขององค์ประกอบต้องอยู่ภายใต้แท็ก <attribute> ซึ่งมีแอททริบิวต์เอกซ์เอมแอล 3 แอททริบิวต์ดังนี้

- name คือชื่อของแอททริบิวต์ที่ประกาศ
- type คือชนิดของแอททริบิวต์มี 2 ชนิดคือ ข้อมูล (data) และข้อมูลอ้างอิง (ref)
- multiplicity คือจำนวนการเก็บข้อมูลในแต่ละแอททริบิวต์ ได้แก่ เก็บข้อมูลค่าเดียว (one) หรือหลายค่า (many)

ตัวอย่างการประกาศนิยามขององค์ประกอบแสดงในตารางที่ 5.2 ซึ่งเป็นการประกาศนิยามของโอเปอร์เรชัน

ตารางที่ 5.2 การประกาศนิยามของโอเปอร์เรชัน

```
<modelelement name="operation">
        <attribute name="id" />
        <attribute name="name" />
        <attribute name="context" type="ref" />
        <attribute name="visibility" />
</modelelement>
```

จากตารางที่ 5.2 โอเปอร์เรชันจะประกอบไปด้วยแอททริบิวต์แบบค่าเดียว 4 แอททริบิวต์ด้วยกันดังนี้

- id คือหมายเลขเอกซ์เอมไอของโอเปอร์เรชันแต่ละตัว
- name คือชื่อของโอเปอร์เรชัน
- context คือค่าอ้างอิงถึงองค์ประกอบที่เป็นเจ้าของโอเปอร์เรชัน
- visibility คือค่าขอบเขตของโอเปอร์เรชัน

ขั้นตอนที่ 2 ของการอ่านข้อมูลแผนภาพคือการกำหนดวิธีในการอ่าน โดยการกำหนดไฟล์ XMI Transformation ซึ่งเป็นไฟล์ที่ระบุข้อกำหนดการเปลี่ยนรูปเพื่ออ่านค่าข้อมูลที่ต้องการตามที่

ได้กำหนดไว้ในไฟล์ Metamodel Definition โดยจะทำการอ่านข้อมูลจากไฟล์แผนภาพที่แปลงให้อยู่ในรูปแบบของเอกซ์เอมแอล โครงสร้างของไฟล์ XMI Transformation แสดงในตารางที่ 5.3 ทั้งนี้ข้อกำหนดการเปลี่ยนรูปจะแตกต่างกันไปตามแต่ผู้ใช้กำหนด

ตารางที่ 5.3 โครงสร้างของไฟล์ XMI Transformation

```
<xmitransformations version="2.0" >
        <xmitransformation ...xmitransformation attributes... />
                <trigger ...trigger attributes... />
                <trigger ...trigger attributes... />
                ...
        </xmitransformation>
        <xmitransformation ...xmitransformation attributes... />
                <trigger ...trigger attributes... />
                ...
        </xmitransformation>
        ...
</xmitransformations>
```

จากตารางที่ 5.3 การกำหนดการเปลี่ยนรูปข้อมูลหรือการอ่านค่าข้อมูลสามารถกระทำได้โดยแท็ก <xmitransformation> ซึ่งจะเป็นการกำหนดการเปลี่ยนรูปข้อมูลขององค์ประกอบ 1 องค์ประกอบเท่านั้น ภายในแท็ก xmitransformation สามารถประกาศแอททริบิวต์ได้ดังนี้

- modelelement คือชื่อขององค์ประกอบที่ต้องการเปลี่ยนรูป
- xmipattern คือรูปแบบเอกซ์เอมไอขององค์ประกอบในไฟล์ข้อมูลแผนภาพที่ต้องการเปลี่ยนรูป
- recurse คือการกำหนดว่าองค์ประกอบที่ต้องการเปลี่ยนรูปสามารถเป็นเจ้าของขององค์ประกอบอื่นได้หรือไม่
- condition คือการกำหนดเงื่อนไขให้ดำเนินการเปลี่ยนรูปเมื่อเงื่อนไขถูกต้องเท่านั้น

การเปลี่ยนรูปข้อมูลขององค์ประกอบสามารถกระทำโดยอาศัยแท็ก trigger ซึ่งมีแอททริบิวต์ที่จำเป็นดังนี้

- type คือการกำหนดประเภทการเปลี่ยนรูปข้อมูล เช่น รหัสเอกซ์เอมไอขององค์ประกอบ หรือชื่อขององค์ประกอบ เป็นต้น
- name คือชื่อขององค์ประกอบที่ต้องการเปลี่ยนรูปตามที่ระบุไว้ใน Metamodel Definition

ตัวอย่างการสร้างข้อกำหนดการเปลี่ยนรูปขององค์ประกอบโอเปอร์เรชันแสดงในตารางที่ 5.4

ตารางที่ 5.4 ตัวอย่างข้อกำหนดการเปลี่ยนรูปขององค์ประกอบโอเปอร์เรชัน

```
<xmitransformation modelelement="operation"
                   xmipattern="Foundation.Core.Operation" recurse="true">
        <trigger name="id" type="attrval" attr="xmi.id" />
        <trigger name="name" type="ctext" src="Foundation.Core.ModelElement.name" />
        <trigger name="visibility" type="cattrval" src="Foundation.Core.ModelElement.visibility"
                 attr="xmi.value"/>
        <trigger name="context" type="gcattrval" src="Foundation.Core.Feature.owner" attr="xmi.idref"/>
</xmitransformation>
```

จากตารางที่ 5.4 องค์ประกอบโอเปอร์เรชันจะมีรูปแบบในภาษาเอกซ์เอมไอเป็น Foundation.Core.Operation และมีแอททริบิวต์ดังนี้

- id คือรหัสขององค์ประกอบเอกซ์เอมไอประจำแต่ละองค์ประกอบในข้อมูลแผนภาพโดยรับค่าจากการดึงรหัสขององค์ประกอบโอเปอร์เรชันมาโดยตรง
- name คือชื่อของโอเปอร์เรชันโดยรับค่าจากการดึงข้อความจากองค์ประกอบลูกของ โอเปอร์เรชัน
- visibility คือขอบเขตการเข้าถึงของโอเปอร์เรชันโดยรับค่าจากการดึงข้อมูลแอททริบิวต์ขององค์ประกอบลูกของโอเปอร์เรชัน
- context คือรหัสขององค์ประกอบที่เป็นเจ้าของโอเปอร์เรชันโดยรับค่าจากการดึงข้อมูล แอททริบิวต์ขององค์ประกอบหลานของโอเปอร์เรชัน

หลังจากทำการเปลี่ยนรูปข้อมูลแผนภาพให้อยู่ในรูปแบบข้อมูลที่สามารถนำไปใช้ในการคำนวณค่าตัววัดแผนภาพได้แล้ว ขั้นตอนต่อไปคือทำการกำหนดวิธีการนำข้อมูลที่ได้ไปใช้คำนวณ

ตัววัดแผนภาพที่ต้องการ ซึ่งรายละเอียดการคำนวณตัววัดแผนภาพจะถูกกำหนดไว้ในไฟล์ metrics definition ซึ่งมีโครงสร้างดังตารางที่ 5.5

ตารางที่ 5.5 โครงสร้างรายละเอียดตัววัดแผนภาพ

```
<metric name="metricname" domain="metricdomain" category="metriccategory" internal="true/false">
        <description>Description of the metric.</description>
        <'metric definition' ...>
</metric>
```

จากตารางที่ 5.5 การกำหนดรายละเอียดของตัววัดแผนภาพจะต้องกำหนดภายใต้แท็ก <metric> ซึ่งจะมีแอททริบิวต์ดังนี้

- name คือชื่อของตัววัดแผนภาพ
- domain คือโดเมนของตัววัดแผนภาพว่าเป็นตัววัดแผนภาพในระดับใด เช่น คลาส หรือแพคเกจ เป็นต้น
- category คือกลุ่มประเภทของตัววัดแผนภาพที่ผู้ใช้สามารถกำหนดได้เองเช่น ขนาด ความซับซ้อน หรือการขึ้นต่อกัน เป็นต้น
- internal คือตัวเลือกในการเลือกให้แสดงผลตัววัดแผนภาพแก่ผู้ใช้หรือไม่

แอททริบิวต์ทั้ง 4 เป็นตัวบอกคุณสมบัติของตัววัดแผนภาพ โดยมีข้อจำกัดบางประการคือ ตัววัดแผนภาพที่อยู่ในโดเมนเดียวกันไม่สามารถมีชื่อเหมือนกันได้ แต่ตัววัดที่อยู่คนละโดเมนสามารถมีชื่อเหมือนกันได้ นอกจากคุณสมบัติของตัววัดแล้วผู้ใช้ยังสามารถกำหนดรายละเอียดหรือคำอธิบายของตัววัดแผนภาพได้ด้วยแท็ก <description>

รายละเอียดการคำนวณตัววัดแผนภาพจะถูกกำหนดไว้ใน <metric definition> ซึ่งจะมีแท็กที่ใช้แตกต่างกันออกไปตามประเภทของตัววัดแผนภาพ โดยประเภทของตัววัดแผนภาพสามารถจำแนกได้เป็น 2 ประเภทหลักๆ คือตัววัดที่ได้จากข้อมูลโดยตรง และตัววัดที่ได้จากค่าของตัววัดอื่น

การประกาศรายละเอียดการคำนวณของตัววัดที่ได้จากข้อมูลโดยตรงจะใช้แท็ก <projection> ซึ่งมีการทำงานหลักโดยการนับจำนวนขององค์ประกอบที่ต้องการ โดยสามารถกำหนดขอบเขตและเงื่อนไขของการนับได้โดยอาศัยแอททริบิวต์เพิ่มเติมของแท็กดังนี้

- relation คือการจำกัดองค์ประกอบที่ใช้ในการคำนวณเฉพาะขององค์ประกอบที่มีความสัมพันธ์ตามที่กำหนดกับองค์ประกอบที่ต้องการคำนวณตัววัด เช่น เป็นองค์ประกอบย่อย (context) เป็นต้น
- relset คือการจำกัดองค์ประกอบที่ใช้ในการคำนวณเฉพาะที่อยู่ในกลุ่มขององค์ประกอบที่ได้ระบุไว้ก่อน
- target คือการจำกัดชนิดขององค์ประกอบที่ใช้ในการคำนวณ
- element คือการจำกัดองค์ประกอบที่มีความสัมพันธ์ทางอ้อมกับองค์ประกอบที่ต้องการคำนวณตัววัด
- eltype คือการจำกัดชนิดขององค์ประกอบที่มีความสัมพันธ์ทางอ้อมกับองค์ประกอบที่ต้องการคำนวณตัววัด
- condition และ targetcondition คือการจำกัดองค์ประกอบที่มีคุณสมบัติตรงตามที่ระบุ เช่น มีขอบเขตการเข้าถึงเป็นพับบลิค เป็นต้น
- scope คือการจำกัดองค์ประกอบที่ใช้ในการคำนวณเฉพาะองค์ประกอบที่มีขอบเขตตามที่กำหนด เช่น อยู่ในแพคเกจเดียวกับองค์ประกอบที่ต้องการคำนวณตัววัด เป็นต้น
- sum คือการนับผลรวมตัววัดขององค์ประกอบที่ใช้ในการคำนวณ เช่น จำนวนโอเปอเรเตอร์ (NumOps) ทั้งหมดของคลาสที่อยู่ในแพคเกจที่สนใจ เป็นต้น
- stat คือการนับค่าตัววัดที่มากที่สุดหรือน้อยที่สุดขององค์ประกอบที่ใช้ในการคำนวณ เช่น จำนวนโอเปอเรเตอร์ (NumOps) ที่มากที่สุดของคลาสที่อยู่ในแพคเกจที่สนใจ เป็นต้น
- recurse คือการกำหนดให้การคำนวณค่าตัววัดทำการทำซ้ำกับองค์ประกอบประเภทเดียวกันกับองค์ประกอบที่ต้องการคำนวณตัววัดและมีความสัมพันธ์ตามที่ระบุ เช่น นับจำนวนคลาสในแพคเกจที่ต้องการรวมไปถึงแพคเกจลูก และต่อไป เป็นต้น
- nesting คือการกำหนดให้การคำนวณค่าตัววัดทำการนับระดับความลึกขององค์ประกอบ

การคำนวณค่าตัววัดแผนภาพจากตัววัดแผนภาพอื่นต้องทำการประกาศรายละเอียดภายใต้แท็ก <compoundmetric> โดยจะทำการคำนวณตามรายละเอียดที่กำหนด แท็ก <compoundmetric> ประกอบไปด้วยแอททริบิวต์ดังนี้

- term คือแอททริบิวต์สำหรับกำหนดรายละเอียดการคำนวณ
- fallback คือแอททริบิวต์สำหรับส่งกลับค่าที่ระบุในกรณีที่การคำนวณค่าวัดได้ผลลัพธ์ที่ไม่ใช่ตัวเลข
- condition คือแอททริบิวต์สำหรับกำหนดเงื่อนไขในการคำนวณ ถ้าเงื่อนไขถูกต้องจะทำการคำนวณตามที่กำหนดแต่ถ้าเงื่อนไขไม่ถูกต้องจะทำงานตามการทำงานสำรอง (alt)
- alt คือแอททริบิวต์สำหรับกำหนดการทำงานสำรองในกรณีที่เงื่อนไขการทำงานไม่ถูกต้อง

5.2.3 การแก้ไขและพัฒนาความสามารถในการประมาณค่าความเสถียรเชิงตรรกะให้แก่โปรแกรม SDMetrics

การแก้ไขและพัฒนาความสามารถในการประมาณค่าความเสถียรเชิงตรรกะให้แก่โปรแกรม SDMetrics สามารถกระทำได้โดยทำการแก้ไขไฟล์นิยามทั้ง 3 โดยในงานวิจัยนี้ได้ทำการเพิ่มเติมตัววัดแผนภาพ ดังนี้

1. ตัววัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ใช้งาน (NumAss_User)
2. ตัววัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ถูกใช้งาน (NumAss_Provider)
3. ตัววัดจำนวนความสัมพันธ์คลาสแม่ของคลาสที่สนใจ (NOP)

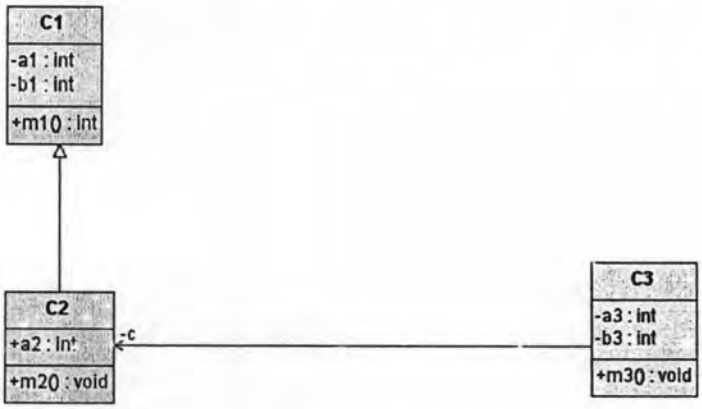
นอกจากนี้ได้ทำการแก้ไขเพื่อให้โปรแกรมสามารถทำการประมาณค่าความเสถียรเชิงตรรกะของแต่ละคลาสจากตัววัดแผนภาพ และแสดงผลการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ได้

5.2.2.1 การเพิ่มตัววัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ใช้งานและตัววัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ถูกใช้งาน

ความสัมพันธ์แอสโซซิเอชันเป็นความสัมพันธ์ระหว่างคลาส 2 คลาสซึ่งแสดงโดยเส้นเชื่อมระหว่างคลาสในแผนภาพคลาส โดยปลายทั้งสองของเส้นจะเชื่อมกับคลาสที่มีความสัมพันธ์แอสโซซิเอชันกัน ความสัมพันธ์แอสโซซิเอชันสามารถแบ่งเป็น 2 แบบ คือความสัมพันธ์แบบทางเดียวและแบบสองทาง

ความสัมพันธ์แบบทางเดียว คือกรณีที่คลาสหนึ่งเป็นผู้ใช้งานอีกคลาสหนึ่งเท่านั้น ในขณะที่ความสัมพันธ์แบบสองทางจะเป็นการใช้งานซึ่งกันและกัน โดยเมื่อทำการแปลงแผนภาพคลาสให้อยู่ในรูปแบบของเอกซ์เอมไอแล้วปลายเส้นความสัมพันธ์แอสโซซิเอชันจะถูกแปลงให้เป็นองค์ประกอบของแผนภาพเช่นเดียวกับคลาส หรือแอททริบิวต์ เรียกว่าจุดสิ้นสุดแอสโซซิเอชัน (AssociationEnd) และมีคุณสมบัติต่างๆ เช่นเดียวกับองค์ประกอบอื่นๆ

คุณสมบัติประการหนึ่งของจุดสิ้นสุดแอสโซซิเอชันที่สามารถใช้ในการแบ่งแยกว่าคลาสใดเป็นผู้เรียกใช้และคลาสใดเป็นผู้ถูกเรียกใช้คือคุณสมบัติ isNavigable ซึ่งจุดสิ้นสุดแอสโซซิเอชันที่เชื่อมต่อกับคลาสที่ถูกเรียกใช้จะมีค่าคุณสมบัตินี้เป็นจริง ดังนั้นในการสร้างตัววัดแผนภาพจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ใช้งานนั้นจะนับจำนวนแอสโซซิเอชันของคลาสที่สนใจเมื่อจุดสิ้นสุดแอสโซซิเอชันที่เชื่อมต่อกับคลาสที่สนใจมีค่าคุณสมบัติ isNavigable เป็นเท็จ



รูปที่ 5.10 ตัวอย่างคลาสที่มีความสัมพันธ์แอสโซซิเอชัน

รูปที่ 5.10 แสดงตัวอย่างคลาสที่มีความสัมพันธ์แอสโซซิเอชัน โดยคลาส C3 เป็นผู้เรียกใช้และคลาส C2 เป็นผู้ถูกเรียกใช้ซึ่งเมื่อทำการแปลงแผนภาพให้อยู่ในรูปแบบไฟล์เอกซ์เอมแอลแล้วจะได้ส่วนของเอกสารเอกซ์เอมแอลที่แสดงความสัมพันธ์แอสโซซิเอชันระหว่างทั้ง 2 คลาสดังในตารางที่ 5.6

ตารางที่ 5.6 ส่วนของเอกสารเอกซ์เอมแอลที่แสดงความสัมพันธ์แอสโซซิเอชัน

```
<UML:Association xmi.id='_9_5_1_24400562_1172516764739_988991_168'>
        <UML:Association.connection>
                <UML:AssociationEnd xmi.id='_9_5_1_24400562_1172516764739_786514_169' name='c'
isNavigable='true' participant='_9_5_1_24400562_1172516764729_273078_153'>
                </UML:AssociationEnd>
                <UML:AssociationEnd xmi.id='_9_5_1_24400562_1172516764739_201817_171'
participant='_9_5_1_24400562_1172516764729_480783_155'/>
        </UML:Association.connection>
</UML:Association>
```

จากตารางที่ 5.6 จะพบว่าภายใต้องค์ประกอบแอสโซซิเอชันจะประกอบไปด้วย 2 จุดสิ้นสุดแอสโซซิเอชัน โดยจะมีจุดสิ้นสุดแอสโซซิเอชันหนึ่งที่มีคุณสมบัติ isNavigable เป็นค่าจริง ซึ่งจุดสิ้นสุดแอสโซซิเอชันนั้นเชื่อมต่อกับคลาส C2 ซึ่งเป็นคลาสที่ถูกเรียกใช้ ทั้งนี้สามารถตรวจสอบได้ว่าจุดสิ้นสุดแอสโซซิเอชันเชื่อมต่อกับคลาสใดโดยดูจากคุณสมบัติ participant ซึ่งจะระบุหมายเลขรหัสเอกซ์เอมไอขององค์ประกอบที่เชื่อมต่อ ในที่นี้เป็นหมายเลขรหัสเอกซ์เอมไอของคลาส C2 ในแผนภาพคลาส

การเก็บค่าคุณสมบัติ isNavigable สามารถทำได้โดยการระบุค่าแอททริบิวต์ navigable เป็นหนึ่งในข้อมูลองค์ประกอบที่ต้องการไว้ในไฟล์ Metamodel Definition และกำหนดวิธีการเก็บค่าไว้ในไฟล์ XMI Transfomation ดังตารางที่ 5.7 และ 5.8

ตารางที่ 5.7 รายละเอียดองค์ประกอบจุดสิ้นสุดแอสโซซิเอชัน

```
<modelelement name="associationend">
        <attribute name="navigable" type="data">Boolean indicating if the association end is navigable.</attribute>
        <attribute name="associationendtype" type="ref">A link to the element attached to the association
end.</attribute>
</modelelement>
```

ตารางที่ 5.8 รายละเอียดการเก็บค่าองค์ประกอบจุดสิ้นสุดแอสโซซิเอชัน

```
<xmitransformation modelelement="associationend" xmipattern="UML:AssociationEnd" >
        <trigger name="navigable" type="attrval" attr="isNavigable" />
        <trigger name="associationendtype" type="attrval" attr="participant"/>
        <trigger name="context" type="attrval" attr="association"/>
</xmitransformation>
```

หลังจากเก็บค่าคุณสมบัติที่จำเป็นได้แล้วทำการสร้างตัววัดโดยอาศัยคุณสมบัติที่เก็บได้ การวัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ใช้งาน สามารถทำได้โดยการนับจำนวนจุดสิ้นสุดแอสโซซิเอชันที่เชื่อมต่อกับคลาสที่สนใจและมีค่าคุณสมบัติ isNavigable เป็นเท็จ ในขณะที่การวัดจำนวนความสัมพันธ์แอสโซซิเอชันเมื่อคลาสที่สนใจเป็นผู้ถูกใช้งานสามารถทำได้โดยการนับจำนวนจุดสิ้นสุดแอสโซซิเอชันที่เชื่อมต่อกับคลาสที่สนใจ และมีค่าคุณสมบัติ isNavigable เป็นจริง รายละเอียดของตัววัดแผนภาพทั้งสองที่สร้างขึ้นแสดงดังตารางที่ 5.9

ตารางที่ 5.9 รายละเอียดตัววัดแผนภาพ

```
<metric name="NumAss_user" domain="class">
<description>Number of association relation when the class is a user.</description>
<projection relation="associationendtype" target="associationend" condition="navigable!='true'" />
</metric>

<metric name="NumAss_provider" domain="class">
<description> Number of association relation when the class is a provider or being used.</description>
<projection relation="associationendtype" target="associationend" condition="navigable='true'" />
</metric>
```

5.2.2.2 การเพิ่มตัววัดจำนวนความสัมพันธ์คลาสแม่ของคลาสที่สนใจ

จำนวนความสัมพันธ์คลาสแม่ หรือจำนวนความสัมพันธ์เจเนอรัลไลเซชันเมื่อคลาสที่สนใจเป็นผู้เรียกใช้ สามารถคำนวณได้โดยการนับจำนวนความสัมพันธ์แบบเจเนอรัลไลเซชันของคลาสที่สนใจเมื่อคลาสที่สนใจเป็นผู้เรียกใช้ที่ปรากฏอยู่ในแผนภาพคลาส โดยเช่นเดียวกับ

ความสัมพันธ์แอสโซซิเอชัน ความสัมพันธ์แบบเจเนอรัลไลเซชันก็เป็นองค์ประกอบหนึ่งของแผนภาพเช่นเดียวกัน และมีรูปแบบข้อมูลเมื่อแปลงเป็นเอกสารเอกซ์เอมแอลดังในตารางที่ 5.10

ตารางที่ 5.10 ส่วนของเอกสารเอกซ์เอมแอลที่แสดงความสัมพันธ์เจเนอรัลไลเซชัน

```
<UML:Generalization xmi.id='_9_5_1_24400562_1172516764739_898503_167'
child='_9_5_1_24400562_1172516764729_273078_153' parent='_9_5_1_24400562_1172516764729_645070_151'/>
```

คุณสมบัติที่สำคัญขององค์ประกอบความสัมพันธ์เจเนอรัลไลเซชัน ซึ่งเป็นคุณสมบัติที่ใช้ในการระบุความสัมพันธ์ระหว่างองค์ประกอบคลาสคือ คุณสมบัติ child ซึ่งระบุรหัสขององค์ประกอบคลาสที่เป็นคลาสลูกผ่านทางความสัมพันธ์นี้ และคุณสมบัติ parent ซึ่งระบุรหัสขององค์ประกอบคลาสที่เป็นคลาสแม่ผ่านทางความสัมพันธ์นี้ ดังนั้นเพื่อนำค่าคุณสมบัติทั้งสองไปใช้ในการสร้างตัววัดแผนภาพจำนวนความสัมพันธ์คลาสแม่ จึงต้องทำการระบุค่าแอททริบิวต์ genchild และ genparent เป็นข้อมูลองค์ประกอบที่ต้องการไว้ในไฟล์ Metamodel Definition และกำหนดวิธีการเก็บค่าไว้ในไฟล์ XMI Transfomation ดังตารางที่ 5.11 และ 5.12

ตารางที่ 5.11 รายละเอียดองค์ประกอบความสัมพันธ์เจเนอรัลไลเซชัน

```
<modelelement name="generalization">
        <attribute name="genchild" type="ref">Link to the child in the generalization.</attribute>
        <attribute name="genparent" type="ref">Link to the parent in the generalization.</attribute>
</modelelement>
```

ตารางที่ 5.12 รายละเอียดการเก็บค่าองค์ประกอบความสัมพันธ์เจเนอรัลไลเซชัน

```
<xmitransformation modelelement="generalization" xmipattern="UML:Generalization" >
        <trigger name="genchild" type="attrval" attr="child" />
        <trigger name="genparent" type="attrval" attr="parent" />
</xmitransformation>
```

การคำนวณค่าตัววัดแผนภาพจำนวนคลาสแม่ของคลาสที่สนใจ สามารถคำนวณได้โดยการนับจำนวนคลาสที่มีความสัมพันธ์แบบเจเนอรัลไลเซชันในฐานะคลาสแม่กับคลาสที่สนใจ เมื่อคลาสที่สนใจอยู่ในสถานะคลาสลูก ตารางที่ 5.13 แสดงรายละเอียดของตัววัดแผนภาพที่สร้างขึ้น

ตารางที่ 5.13 รายละเอียดตัววัดแผนภาพจำนวนคลาสแม่

```
<metric name="NOP" domain="class">
        <description> Number of parent class</description>
        <projection relation="genchild" target="generalization" element="genparent" eltype="class" />
</metric>
```

5.2.2.3 การเพิ่มความสามารถในการประมาณค่าความเสถียรเชิงตรรกะตามโมเดลที่พัฒนาขึ้น

โมเดลประมาณค่าความเสถียรเชิงตรรกะที่พัฒนาขึ้นจะอาศัยตัววัดแผนภาพต่างๆ ในการคำนวณ ดังนั้นในการเพิ่มความสามารถให้โปรแกรม SDMetrics สามารถประมาณค่าความเสถียรเชิงตรรกะของแต่ละคลาสสามารถทำได้โดย สร้างเป็นตัววัดแผนภาพของคลาสซึ่งอาศัยค่าจากตัววัดแผนภาพอื่นในการคำนวณโดยอาศัยแท็ก <compoundmetric>

เนื่องจากงานวิจัยนี้ได้ทำการสร้างโมเดลประมาณค่าความเสถียรเชิงตรรกะของคลาสจากแผนภาพคลาสและแผนภาพซีเควนซ์สำหรับโปรแกรม 3 ประเภท ดังนั้นจึงต้องทำการสร้างตัววัดประมาณค่าความเสถียรเชิงตรรกะ 3 ตัววัดด้วยกันตามโมเดลของโปรแกรมแต่ละประเภท โดยมีรายละเอียดดังตารางที่ 5.14 ถึง 5.16 และทำการสร้างตัววัดประมาณค่าความเสถียรเชิงตรรกะทั้ง 3 แยกจากกันอยู่ในไฟล์ข้อกำหนดตัววัดแผนภาพ 3 ไฟล์

ตารางที่ 5.14 รายละเอียดการคำนวณโมเดลสำหรับประมาณค่าความเสถียรเชิงตรรกะของคลาสสำหรับโปรแกรมด้านการคำนวณ

```
<metric name="EstLSta" domain="class">
        <description>Estimated Logical Stability.</description>
        <compoundmetric term="0.9805680980-0.0032740467*ClassifInst +0.0048858207*NumOps-
        0.0064901626*NumAnc+0.0080432844*AttrInh  -0.0010171931*MsgSent +0.0005569513*NumDesc"
        fallback="0"/>
</metric>
```

ตารางที่ 5.15 รายละเอียดการคำนวณโมเดลสำหรับประมาณค่าความเสถียรเชิงตรรกะของคลาสสำหรับโปรแกรมด้านการจัดการข้อความ

```
<metric name="EstLSta" domain="class">
        <description>Estimated Logical Stability.</description>
        <compoundmetric term="1.0426495664 - 0.0239384916*NumPubOps - 0.0049619233*NumOps"
        fallback="0"/>
</metric>
```

ตารางที่ 5.16 รายละเอียดการคำนวณโมเดลสำหรับประมาณค่าความเสถียรเชิงตรรกะของคลาสสำหรับโปรแกรมด้านการจัดการรูปภาพ

```
<metric name="EstLSta" domain="class">
        <description>Estimated Logical Stability.</description>
        <compoundmetric term="0.9908283241 - 0.0829356623*IfImpl - 0.0541333205*NOP
        - 0.0004169792*MsgSelf - 0.0212684802*CLD " fallback="0"/>
</metric>
```

การเลือกใช้โมเดลให้เหมาะสมกับโปรแกรมที่ต้องการประมาณค่า ผู้ใช้ต้องทำการเลือกเองโดยทำการเลือกไฟล์ข้อกำหนดตัววัดแผนภาพที่ต้องการ

จากการเพิ่มความสามารถในการประมาณค่าความเสถียรเชิงตรรกะของคลาสตามโมเดลที่พัฒนาขึ้น ทำให้สามารถเพิ่มความสามารถในการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ได้อีกด้วย โดยความเสถียรเชิงตรรกะของซอฟต์แวร์จะเท่ากับค่าเฉลี่ยของความเสถียรเชิงตรรกะของทุกคลาสในซอฟต์แวร์ โดยในการเพิ่มความสามารถในการแสดงค่าประมาณของความเสถียรเชิงตรรกะของซอฟต์แวร์ จะต้องทำการเพิ่มข้อกำหนดองค์ประกอบซอฟต์แวร์ลงในไฟล์ MetaModel Definition และ XMI Transformation เสียก่อน เพื่อให้โปรแกรม SDMetrics รู้จักองค์ประกอบโปรแกรมและแสดงผลในระดับซอฟต์แวร์ได้ รายละเอียดของข้อกำหนดองค์ประกอบซอฟต์แวร์แสดงในตารางที่ 5.17 และ 5.18

ตารางที่ 5.17 รายละเอียดองค์ประกอบซอฟต์แวร์

```
<modelelement name="software" />
```

ตารางที่ 5.18 รายละเอียดการเก็บค่าองค์ประกอบซอฟต์แวร์

```
<xmitransformation modelelement="software" xmipattern="UML:Model" recurse="true" />
```

การหาค่าเฉลี่ยของค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในโปรแกรมแบ่งออกเป็น 2 กรณีด้วยกันคือ กรณีที่มีการแบ่งแพคเกจ และกรณีที่ไม่มีการแบ่งแพคเกจ

สำหรับกรณีที่มีการแบ่งแพคเกจต้องหาค่าเฉลี่ยของค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในแพคเกจเสียก่อน โดยในการหาค่าเฉลี่ยของค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในแพคเกจ จะอาศัยตัววัดแผนภาพ NumCls_tc ซึ่งเป็นจำนวนคลาสทั้งหมดในแพคเกจ และตัววัดแผนภาพ SumStab_p ซึ่งเป็นผลรวมของค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในแพคเกจ และค่าประมาณความเสถียรเชิงตรรกะของแพคเกจจะเท่ากับ ค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในแพคเกจหารด้วยจำนวนคลาสทั้งหมดในแพคเกจ รายละเอียดของตัววัดแผนภาพ NumCls_tc SumStab_p และค่าประมาณความเสถียรเชิงตรรกะของแพคเกจแสดงในตารางที่ 5.19 ถึง 5.21 ตามลำดับ

ตารางที่ 5.19 รายละเอียดตัววัดแผนภาพจำนวนคลาสทั้งหมดในแพคเกจ

```
<metric name="NumCls_tc" domain="package" category="Size">
        <description>The number of classes in the package, its subpackages, and so on</description>
        <projection relation="context" target="class" recurse="true"/>
</metric>
```

ตารางที่ 5.20 รายละเอียดตัววัดแผนภาพผลรวมค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในแพคเกจ

```
<metric name="SumStab_p" domain="package" internal="true">
        <description>The Summation of class logical stability in this package, sub package and so on.
        </description>
        <projection relation="context" target="class" sum="EstLSta" recurse="true"/>
</metric>
```

ตารางที่ 5.21 รายละเอียดตัววัดค่าประมาณความเสถียรเชิงตรรกะของแพคเกจ

```
<metric name="EstStab" domain="package">
        <description>The estimate logical stability of this package.</description>
        <compoundmetric conditions="NumCls_tc>0" term="SumStab/NumCls_tc" fallback="0">
        </compoundmetric>
</metric>
```

หลังจากสามารถคำนวณค่าประมาณความเสถียรเชิงตรรกะของแพคเกจได้แล้ว จะสามารถคำนวณค่าประมาณความเสถียรเชิงตรรกะของซอฟต์แวร์ได้โดย หาผลรวมของค่าประมาณความเสถียรเชิงตรรกะของทุกแพคเกจแล้วหารด้วยจำนวนแพคเกจทั้งหมด ดังนั้นต้องทำการสร้างตัววัดแผนภาพเพิ่มเติมอีก 2 ตัววัด คือ ตัววัดจำนวนแพคเกจ(NumPck) และผลรวมของค่าประมาณความเสถียรเชิงตรรกะของทุกแพคเกจ(SumStab) ซึ่งมีรายละเอียดในตารางที่ 5.22 และ 5.23 ตามลำดับ

ตารางที่ 5.22 รายละเอียดตัววัดแผนภาพจำนวนแพคเกจทั้งหมดในซอฟต์แวร์

```
<metric name="Numpck" domain="software">
        <description>Number of All the Package in Software</description>
        <projection relation="context" target="package"  condition="NumCls_tc>0" recurse="true"/>
</metric>
```

ตารางที่ 5.23 รายละเอียดตัววัดแผนภาพผลรวมค่าประมาณความเสถียรเชิงตรรกะของทุกแพคเกจในซอฟต์แวร์

```
<metric name="SumStab" domain="software" internal="true" >
        <description>Sum of the estimate logical stability of package</description>
        <projection relation="context" target="package" sum="EstStab" condition="NumCls_tc>0" recurse="true"/>
</metric>
```

สำหรับกรณีที่ซอฟต์แวร์ไม่มีการแบ่งแพคเกจ หรือไม่มีแพคเกจ ต้องทำการหาจำนวนคลาสทั้งหมดของซอฟต์แวร์ และผลรวมของค่าประมาณความเสถียรเชิงตรรกะของทุกคลาส โดยทำการสร้างตัววัดแผนภาพอีก 2 ตัวคือ NumCls และ SumStab_c ซึ่งมีรายละเอียดในตารางที่ 5.24 และตารางที่ 5.25 ตามลำดับ

ตารางที่ 5.24 รายละเอียดตัววัดแผนภาพจำนวนคลาสทั้งหมดในซอฟต์แวร์

```
<metric name="NumCls" domain="software">
        <description>Number of All the class in Software</description>
        <projection relation="context" target="class" recurse="true"/>
</metric>
```

ตารางที่ 5.25 รายละเอียดตัววัดแผนภาพผลรวมค่าประมาณความเสถียรเชิงตรรกะของทุกคลาส

```
<metric name="SumStab_c" domain="software" internal="true">
        <description>Sum of the estimate logical stability of package</description>
        <projection relation="context" target="class" sum="EstLSta"  recurse="true"/>
</metric>
```

ความเสถียรเชิงตรรกะของซอฟต์แวร์จะมีค่าเท่ากับผลรวมค่าประมาณความเสถียรเชิงตรรกะของทุกแพคเกจในซอฟต์แวร์หารด้วยจำนวนแพคเกจ หรือเท่ากับผลรวมค่าประมาณความเสถียรเชิงตรรกะของทุกคลาสในซอฟต์แวร์หารด้วยจำนวนคลาสทั้งหมด โดยรายละเอียดของตัววัดค่าประมาณความเสถียรเชิงตรรกะของซอฟต์แวร์อยู่ในตารางที่ 5.26

ตารางที่ 5.26 รายละเอียดตัววัดค่าประมาณความเสถียรเชิงตรรกะของซอฟต์แวร์

```
<metric name="EstLStab" domain="software">
        <description>The Estimate Logical Stability of program calculated by sum of all class estimate logical stability divide by number of classes</description>
        <compoundmetric condition="Numpck>0" term="SumStab_p/Numpck" alt="SumStab_c/NumCls" fallback="0"/>
</metric>
```