บทที่ 4

การออกแบบและพัฒนาเครื่องมือสำหรับทดสอบงานวิจัย

ในส่วนนี้จะกล่าวถึงการออกแบบและพัฒนาเครื่องมือที่ใช้สำหรับทดสอบงานวิจัย เนื่องจากวัตถุประสงค์ของงานวิจัยนี้ต้องการเปรียบเทียบประสิทธิภาพของซอฟต์แวร์ขณะประมวลผลและเปรียบเทียบผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ตามความต้องการเชิงฟังก์ชันระหว่างโครงสร้างคลาสในความสัมพันธ์แบบแอสโซซิเอชันและโครงสร้างคลาสในความสัมพันธ์แบบเจเนอรัลไลเซชัน ดังนั้น ผู้วิจัยได้จัดทำเครื่องมือเพื่อใช้สำหรับทดสอบงานวิจัย แยกออกเป็น 2 ชุด คือ (1) การออกแบบและพัฒนาเครื่องมือสำหรับวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล และ (2) การออกแบบและพัฒนาเครื่องมือสำหรับวัดผลกระทบในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ โดยเครื่องมือทั้ง 2 ชุดนี้ออกแบบโดยใช้หลักการเชิงวัตถุ โดยพัฒนาภายใต้โปรแกรมภาษาจาวา (Java Programming Language) และใช้ชุดโปรแกรมสำหรับพัฒนาภาษาโปรแกรมจาวา (Integrated Development Environment - IDE) ที่มีชื่อว่าอิคลิพซ์ (Eclipse) เวอร์ชัน 3.2.2 ซึ่งเป็นชุดเครื่องมือที่เป็นโอเพนซอร์ส (Open Source) ที่สามารถดาวน์โหลดมาใช้งานได้ฟรีจากเว็บไซต์ http://www.eclipse.org โดยชุดโปรแกรมนี้มีคอมไพล์เลอร์ (Compiler) อินเทอร์พรีเตอร์ (Interpreter) เครื่องมือบริหารจัดการไฟล์ (Editor File Manager) ดีบักเกอร์ (Debugger) และคลาสไลบรารี (Class Libraries) ของโปรแกรมภาษาจาวาสำหรับสร้างโปรแกรมที่ช่วยให้การทำงานมีประสิทธิภาพมากขึ้น (Holzner, 2004)

นอกเหนือจากการติดตั้งชุดโปรแกรมสำหรับพัฒนาโปรแกรมภาษาจาวาแล้ว เพื่อให้เครื่องมือที่พัฒนาขึ้นและหน่วยตัวอย่างที่ใช้ในงานวิจัยสามารถทำงานได้ ต้องดำเนินการติดตั้งจาวารันไทม์เอนเวอร์รอนเมนต์ (Java Runtime Environment - JRE) โดยสามารถดาวน์โหลดเพื่อติดตั้งได้จาก http://www.java.sun.com เพื่อให้โปรแกรมภาษาจาวาสามารถทำงานได้ ซึ่งโปรแกรมภาษาจาวาใช้แนวความคิดของการสร้างเครื่องจักรสมมติ (Java Virtual Machine) เพื่อให้โปรแกรมภาษาจาวา ที่ถูกคอมไพล์ด้วยจาวาคอมไพล์เลอร์ เป็นชุดคำสั่งของเครื่องจักรสมมติภาษาจาวาหรือที่เรียกว่าชุดคำสั่งไบต์โค้ด (Byte Code) ซึ่งจะอยู่ในรูปของจาวาคลาส (Java Classes) และเมื่อถึงขั้นตอนการประมวลผลจึงเพียงแต่แปลจากคำสั่งของเครื่องจักรสมมติภาษาจาวาเป็นคำสั่งที่หน่วยประมวลผลนั้นทำงานได้เรียกว่าเนทิฟว์โค้ด (Native Code) เพื่อให้โปรแกรมสามารถทำงานบนเครื่องคอมพิวเตอร์ได้ (วีระศักดิ์ ซึงถาวร, 2549) เมื่อทำการติดตั้งโปรแกรมหรือซอฟต์แวร์ที่

เกี่ยวข้องกับการพัฒนาเครื่องมือเรียบร้อยแล้ว สามารถอธิบายการออกแบบและพัฒนาเครื่องมือแต่ละชุด ดังนี้

4.1 การออกแบบและพัฒนาเครื่องมือสำหรับวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล

จากวัตถุประสงค์ของงานวิจัย ต้องการเปรียบเทียบประสิทธิภาพของซอฟต์แวร์ขณะประมวลผลระหว่างโครงสร้างคลาสในความสัมพันธ์แบบแอสโซซิเอชันและโครงสร้างคลาสในความสัมพันธ์แบบเจเนอรัลไลเซชันขณะประมวลผล ดังนั้นการออกแบบและพัฒนาเครื่องมือสำหรับทดสอบงานวิจัยส่วนนี้ สามารถอธิบายได้ดังนี้

4.1.1 ความต้องการเชิงฟังก์ชันของเครื่องมือ (Functional Requirement)

การวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผลของโครงสร้างคลาสในความสัมพันธ์แบบแอสโซซิเอชันและโครงสร้างคลาสในความสัมพันธ์เจเนอรัลไลเซชัน ทำได้โดย การคำนวณจากจำนวนการเรียกใช้งานหรือการส่งเมสเสจ (Message Calling) ระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์ทั้งหมดขณะประมวลผลการทำงานของซอฟต์แวร์ และคำนวณจากระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์ (Response Time) ซึ่งผู้วิจัยได้ออกแบบและและพัฒนาเครื่องมือที่ใช้ในการทดสอบงานวิจัยเป็นซอฟต์แวร์ประเภทโปรแกรมประยุกต์แบบวินโดว์ (Windows Application Program) โดยมีการเรียกใช้งานจาวาคลาสไลบรารี (Java Class Libraries) ของชุดโปรแกรมสำหรับพัฒนาโปรแกรมภาษาจาวาหรือที่เรียกว่าแอพพลิเคชันโปรแกรมเมอร์อินเทอร์เฟช (Application Programmer Interface - APIs) ในส่วนของเจเอฟซี (Java Foundation Classes - JFC) ซึ่งจะมีคอมโพเนนท์ (Component) สวิง (Swing) เพื่อใช้ในการสร้างจอภาพแบบกราฟฟิก (Graphic User Interface) ได้

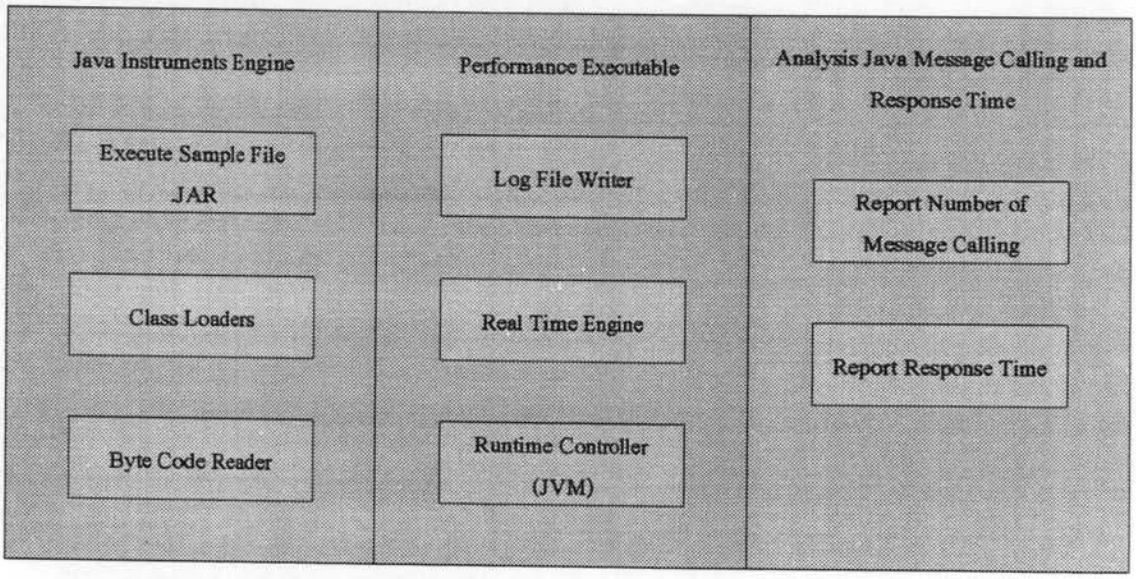
เพื่อให้เครื่องมือที่ออกแบบและพัฒนาขึ้นสามารถทำงานได้โดยตอบวัตถุประสงค์ของงานวิจัยได้ ดังนั้น เครื่องมือประกอบด้วยฟังก์ชันการทำงานต่าง ๆ ดังนี้

1. เมื่อผู้ใช้ต้องการคำนวณหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล ซอฟต์แวร์จะเรียกไฟล์คอทจาร์ (.JAR) เข้ามาประมวลผลและคำนวณหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล
2. การประมวลผลการทำงานของซอฟต์แวร์หน่วยตัวอย่าง เครื่องมือต้องสามารถติดตามการเรียกใช้งานหรือการส่งเมสเสจ และสามารถคำนวณจำนวนการเรียกใช้งานหรือการส่งเมสเสจระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์ทั้งหมดขณะประมวลผลการทำงานของซอฟต์แวร์

3. การประมวลผลการทำงานของซอฟต์แวร์หน่วยตัวอย่าง เครื่องมือต้องสามารถคำนวณระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์
4. เครื่องมือสามารถแสดงผลจำนวนการเรียกใช้งานหรือการส่งเมสเสจ รวมทั้งระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์ได้
5. เครื่องมือสามารถบันทึกผลจำนวนการเรียกใช้งานหรือการส่งเมสเสจ รวมทั้งระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์ได้

หมายเหตุ

ไฟล์ดอทจาร์ (.JAR) คือ การบีบอัดไฟล์ดอทคลาส (.CLASS) จำนวนหลาย ๆ คลาสเก็บลงในไฟล์เพียงไฟล์เดียว เพื่อให้ซอฟต์แวร์มีขนาดเล็กลง และง่ายในการนำข้อมูลเข้าสู่เครื่องมือเพื่อประมวลผลประสิทธิภาพของซอฟต์แวร์



รูปที่ 4-1 สถาปัตยกรรมของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล

จากรูปที่ 4-1 แสดงสถาปัตยกรรมของเครื่องมือที่ใช้สำหรับคำนวณหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผลที่พัฒนาภายใต้ชุดโปรแกรมพัฒนาภาษาจาวาอิคลิพซ์ โดยแบ่งทำงานออกเป็น 3 ส่วน คือ

- ส่วนจาวาอินสทรูเมินท์เอนจิน (Java Instruments Engine) เป็นส่วนที่ใช้สำหรับการอ่านข้อมูลนำเข้าของซอฟต์แวร์ที่ใช้เป็นหน่วยตัวอย่าง โดยอ่านข้อมูลนำเข้าเป็นดอทจาร์ (.JAR) จากนั้นคลาสโหลดเดอร์ (Class Loader) จะโหลดไฟล์ดอทคลาส

(.CLASS) ในจาร์ไฟล์ขึ้นมาทำงาน ซึ่งจะเป็นการอ่านกระบวนการทำงานในลักษณะที่เป็นไบต์โค้ด (Byte Code Reader)

- ส่วนการประมวลผลเพื่อทดสอบประสิทธิภาพของซอฟต์แวร์ (Performance Executable) เมื่อมีการประมวลผลไบต์โค้ดของซอฟต์แวร์ที่เป็นหน่วยตัวอย่าง จะมีการเก็บบันทึกลอกไฟล์ (Log File Writer) ของการเรียกใช้เมสเสจของคลาสทั้งหมดที่เกี่ยวข้องกับการทำงานของซอฟต์แวร์ นอกจากนี้จะมีการเก็บระยะเวลา (Real Time Engine) ที่ใช้ในการประมวลผลซอฟต์แวร์โดยการคำนวณเวลาจะถูกควบคุมโดยการทำงานของเครื่องจักรสมมติของโปรแกรมภาษาจาวา (Runtime Controller - JVM)
- ส่วนการวิเคราะห์จำนวนการเรียกใช้งานเมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์โดยสมบูรณ์ (Analysis Java Message Calling and Response Time) ทำได้โดยการนำลอกไฟล์ที่บันทึกไว้มาวิเคราะห์และรายงานผลจำนวนการรับส่งเมสเสจ (Report Number of Message Calling) ของคลาสทั้งหมดที่เกี่ยวข้องกับการทำงานของซอฟต์แวร์ รวมทั้งรายงานระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์ (Report Response Time)

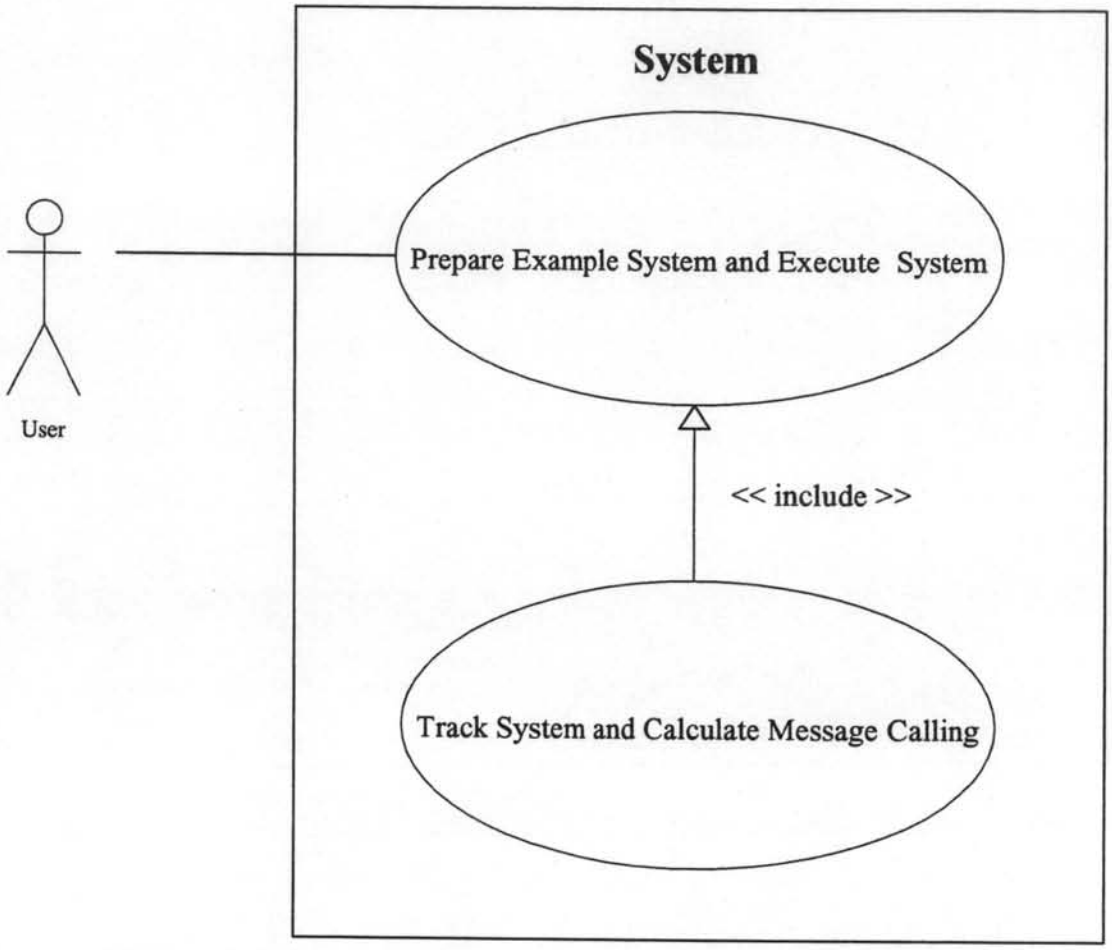
4.1.2 ประเภทของเมสเสจ

การวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล ทำได้โดย การคำนวณจากจำนวนการเรียกใช้งานหรือการส่งเมสเสจ (Message Calling) ระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์ทั้งหมด ซึ่งสามารถแบ่งประเภทของเมสเสจได้เป็น 3 ประเภท คือ เมสเสจที่เป็นแอททริบิวต์ เมสเสจที่เป็นอาร์เรย์และเมสเสจที่เป็นเมธอด ดังรายละเอียดที่กล่าวมาแล้วในบทที่ 2

โดยผู้วิจัยได้นำประเภทของเมสเสจที่กล่าวมาออกแบบวิธีการคำนวณหาจำนวนการเรียกใช้งานหรือส่งเมสเสจระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์และพัฒนาเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล

4.1.3 แผนภาพยูสเคสและคำอธิบายยูสเคส (Use Case Diagram and Use Case Description) ของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

รูปที่ 4-2 เป็นแผนภาพยูสเคสของเครื่องมือสำหรับคำนวณหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล ซึ่งสามารถคำนวณได้จากการรับส่งหรือเรียกใช้เมสเสจระหว่างคลาสทั้งหมดที่เกี่ยวข้องกับการทำงานของซอฟต์แวร์และคำนวณจากระยะเวลาที่ซอฟต์แวร์ประมวลผลโดยสมบูรณ์ โดยสามารถอธิบายรายละเอียดการทำงานของยูสเคสได้ ดังตารางที่ 4-1



รูปที่ 4-2 แผนภาพของยูสเคสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

ตารางที่ 4-1 คำอธิบายยูสเคสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

1. ยูสเคสการจัดเตรียมหน่วยตัวอย่างและประมวลผล (Prepare Example System and Execute System)	
เป้าหมาย (Goal)	จัดเตรียมข้อมูลหน่วยตัวอย่างที่ใช้ให้อยู่ในรูปแบบดอทจาร์ (.JAR) ก่อนเข้าสู่การคำนวณประสิทธิภาพของซอฟต์แวร์
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Pre - condition)	หน่วยตัวอย่างสามารถประมวลผลได้อย่างถูกต้อง
เงื่อนไขหลัง (Post - condition)	จัดเตรียมข้อมูลที่ใช้สำหรับการคำนวณประสิทธิภาพของซอฟต์แวร์ขณะประมวลผลเรียบร้อยแล้ว

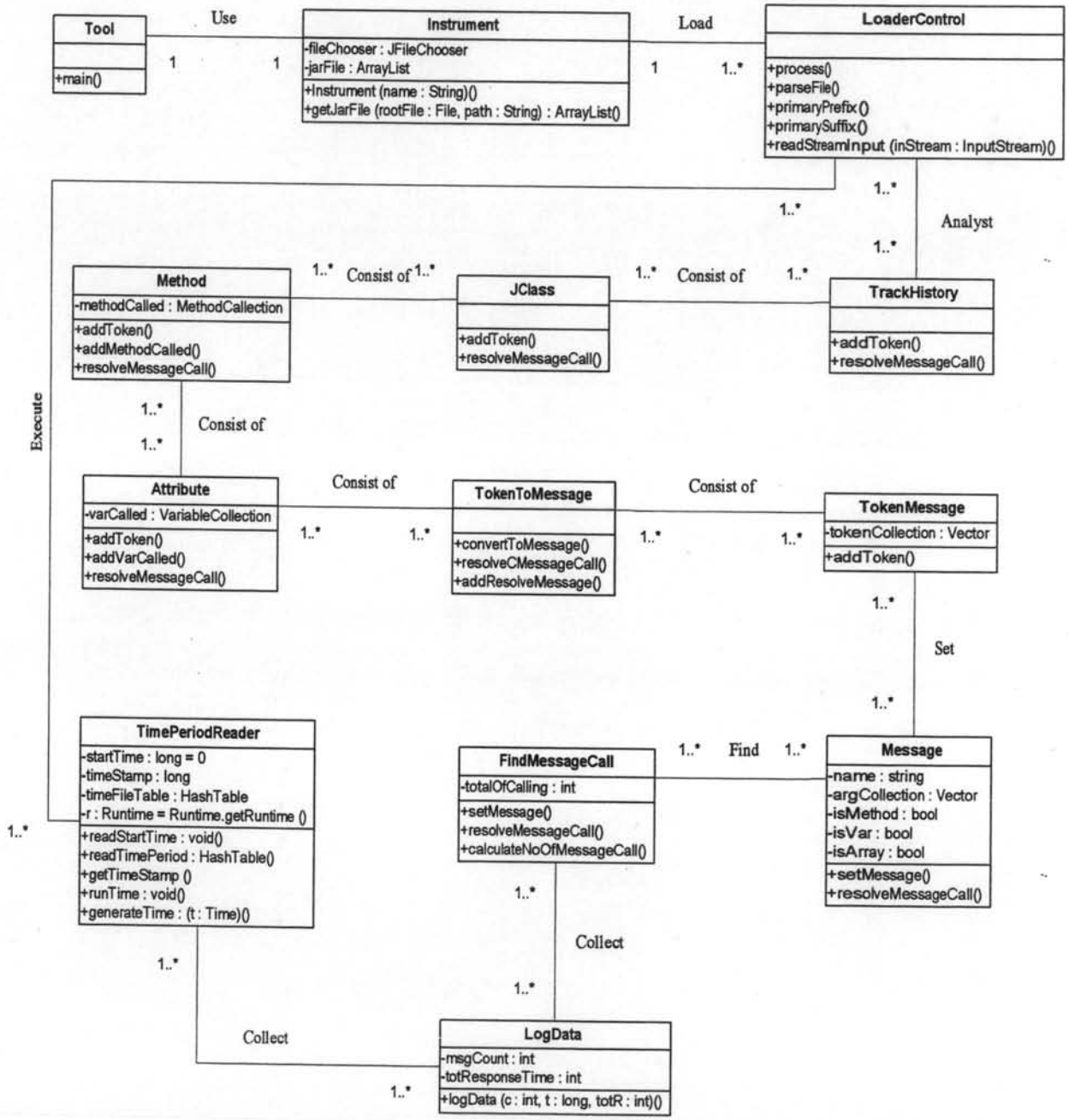
ตารางที่ 4-1 คำอธิบายยูสเคสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

เมนซัคเซส ซีนาริโอ (Main Success Scenario)	1. ผู้ใช้เรียกข้อมูลที่เป็นนามสกุลดอทจาร์ (File.jar) ของหน่วยตัวอย่างที่ต้องการคำนวณหาประสิทธิภาพของซอฟต์แวร์ 2. ระบบแสดงรายชื่อคลาส และจำนวนคลาสทั้งหมดที่จะประมวลผลทางหน้าจอ 3. ผู้ใช้ทำกระบวนการอินสทรูเมินท์เอนจิน เพื่อให้คลาสโหลดเดอร์โหลดไฟล์ดอทคลาส (.CLASS) ในจาร์ไฟล์ขึ้นมาทำงาน 4. ผู้ใช้ประมวลผลซอฟต์แวร์หน่วยตัวอย่าง และระบบจัดเก็บลอกไฟล์การรับส่ง / เรียกใช้งานเมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์หน่วยตัวอย่าง
2. ยูสเคสการวิเคราะห์และคำนวณการเรียกใช้เมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์ (Track System and Calculate Message Calling and Response Time)	
เป้าหมาย (Goal)	คำนวณการรับส่ง / เรียกใช้งานเมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Pre - condition)	ระบบจัดเตรียมข้อมูลที่ใช้สำหรับการคำนวณการรับส่ง / เรียกใช้งานเมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์ ได้แก่ ลอกไฟล์ที่จัดเก็บได้จากการประมวลผลซอฟต์แวร์หน่วยตัวอย่างเรียบร้อยแล้ว
เงื่อนไขหลัง (Post - condition)	ระบบวิเคราะห์และรายงานผลการคำนวณการรับส่ง / เรียกใช้งานเมสเสจและระยะเวลาในการประมวลผลซอฟต์แวร์เรียบร้อยแล้ว
เมนซัคเซส ซีนาริโอ (Main Success Scenario)	1. ระบบโหลดลอกไฟล์ขึ้นมาวิเคราะห์และสร้างรายงานผลจำนวนการรับส่ง / เรียกใช้งานเมสเสจ 2. ระบบโหลดลอกไฟล์ขึ้นมาวิเคราะห์และสร้างรายงานผลระยะเวลาในการประมวลผลซอฟต์แวร์

4.1.4 แผนภาพคลาสและคำอธิบายคลาส (Class Diagram and Class Description) ของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

รูปที่ 4-3 เป็นแผนภาพคลาสของเครื่องมือสำหรับคำนวณหาประสิทธิภาพของซอฟต์แวร์ ซึ่งคำนวณได้จากจำนวนการเรียกใช้งานหรือการส่งเมสเสจระหว่างคลาสที่เป็นองค์ประกอบของ

ซอฟต์แวร์ทั้งหมดขณะประมวลผล และระยะเวลาที่ซอฟต์แวร์หนึ่ง ๆ ถูกประมวลผลโดยสมบูรณ์ ซึ่งจะแสดงเฉพาะแผนภาพคลาสที่เกี่ยวข้องกับกระบวนการส่วนที่วิเคราะห์และประมวลผลเพื่อหาค่าประสิทธิภาพของซอฟต์แวร์เท่านั้น โดยสามารถแสดงรายละเอียดของแต่ละคลาส ได้ดังตารางที่ 4-2



รูปที่ 4-3 แผนภาพคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

ชื่อคลาส (Class Name)	Tool
คำอธิบายคลาส (Class Description)	เป็นคลาสหลักของการหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล โดยวัดจากจำนวนของเมสเสจที่ถูกเรียกใช้งานในระบบ โดยคลาสนี้จะเป็นคลาสแรกที่ถูกเรียกขึ้นมาใช้งาน
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
main	เป็นเมธธอดหลัก และเป็นเมธธอดแรกที่ถูกเรียกใช้งานเมื่อต้องการหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล
ชื่อคลาส (Class Name)	Instrument
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ในการโหลดหน่วยตัวอย่างเข้ามาในเครื่องมือ ซึ่งจะโหลดเข้ามาเป็นไฟล์ดอทจาร์ (.JAR)
แอททริบิวต์ (Attribute)	
fileChooser	เป็นแอททริบิวต์ที่ทำหน้าที่ในการเก็บไฟล์ที่ผู้ใช้เลือกที่จะนำเข้าเครื่องมือ
jarFile	เป็นแอททริบิวต์ที่เก็บไฟล์ดอทจาร์ที่ผู้ใช้โหลดเข้ามาในเครื่องมือ
เมธธอด (Method)	
Instrument	เป็นเมธธอดที่ทำหน้าที่โหลดชื่อไฟล์ดอทจาร์
getJarFile	เป็นเมธธอดที่ทำหน้านำเข้าไฟล์ดอทจาร์เข้าสู่เครื่องมือ

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

ชื่อคลาส (Class Name)	LoaderControl
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่โหลดไฟล์ดอทคลาส (.CLASS) ในจาร์ไฟล์ขึ้นมาทำงาน ซึ่งจะเป็นการอ่านกระบวนการทำงานในลักษณะที่เป็นไบต์โค้ด รวมทั้งควบคุมกระบวนการในการหาประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
process	เป็นเมธธอดที่ควบคุมกระบวนการทำงานของซอร์สโค้ดก่อนการท่องโปรแกรมเพื่อหาจำนวนการเรียกใช้งานเมสเสจในระบบ
readStreamInput	เป็นเมธธอดที่ใช้ในการอ่านอินพุทสตรีมของไฟล์ดอทคลาสขึ้นมาทำงาน
parseFile	เป็นเมธธอดที่เริ่มต้นการท่องซอร์สโค้ด
primaryPrefix	เป็นเมธธอดที่เก็บโทเคนในส่วนพรีฟิกซ์ของเมสเสจจากรหัสโปรแกรม
primarySuffix	เป็นเมธธอดที่เก็บโทเคนในส่วนซัฟฟิกซ์ของเมสเสจจากรหัสโปรแกรม
ชื่อคลาส (Class Name)	TrackHistory
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ติดตามการท่องซอร์สโค้ดของซอฟต์แวร์และจัดการกับโทเคนที่เก็บข้อมูลได้จากการท่องซอร์สโค้ด
แอททริบิวต์ (Attribute)	
-	-

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

เมธธอด (Method)	
addToken	เป็นเมธธอดที่รับโทเคนของเมสเสจที่เก็บได้ ส่งไปยังคลาส JClass
resolveMessageCall	เป็นเมธธอดที่ส่งคำสั่งการแปลงโทเคนที่เก็บไว้เป็นเมสเสจ และการจับคู่เมสเสจไปยังคลาส JClass
ชื่อคลาส (Class Name)	TokenToMessage
คำอธิบายคลาส (Class Description)	เป็นคลาสที่นำโทเคนของเมสเสจที่เก็บได้ มาแปลงเป็นเมสเสจ
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
convertToMessage	เป็นเมธธอดที่ทำหน้าที่ในการแปลงโทเคนของเมสเสจที่เก็บได้ ให้เป็นเมสเสจ
resolveMessageCall	เป็นเมธธอดที่ใช้ในการจับคู่เมสเสจหลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว
addResolveMessage	เป็นเมธธอดที่ทำหน้าที่เพิ่มเมสเสจที่เป็นแอททริบิวต์และเมธธอดที่ถูกเรียกใช้งาน
ชื่อคลาส (Class Name)	TokenMessage
คำอธิบายคลาส (Class Description)	เป็นคลาสที่เก็บโทเคนของเมสเสจ ก่อนที่จะนำเอาไปแปลงเป็นเมสเสจ
แอททริบิวต์ (Attribute)	
tokenCollection	เป็นแอททริบิวต์ที่ใช้เก็บรายละเอียดโทเคนของเมสเสจ
เมธธอด (Method)	
addToken	เป็นเมธธอดที่ทำหน้าที่ในการเก็บอ๊อบเจคที่เป็นสตริงของโทเคน เพื่อที่จะนำมาใช้แปลงเป็นเมสเสจต่อไป

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

ชื่อคลาส (Class Name)	Message
คำอธิบายคลาส (Class Description)	เป็นคลาสที่เก็บการเรียกใช้เมสเสจทั้งหมดของคลาสที่เป็นองค์ประกอบของซอฟต์แวร์หน่วยตัวอย่าง หลังจากมีการแปลงโทเคนเรียบร้อยแล้วแล้ว
แอททริบิวต์ (Attribute)	
name	เป็นแอททริบิวต์ที่ทำหน้าที่จัดเก็บชื่อเมสเสจที่เป็นแอททริบิวต์หรือชื่อเมสเสจที่เป็นเมธธอด
argCollection	เป็นแอททริบิวต์ที่เก็บอาร์กิวเมนต์ของเมสเสจที่เป็นเมทธอด หรือเก็บอินเด็กซ์ของอาร์เรย์
isMethod	เป็นแอททริบิวต์ที่เก็บเมสเสจที่เป็นเมธธอด
isAttr	เป็นแอททริบิวต์ที่เก็บเมสเสจที่เป็นแอททริบิวต์
isArray	เป็นแอททริบิวต์ที่เก็บเมสเสจที่เป็นอาร์เรย์
เมธธอด (Method)	
setMessage	เป็นเมธธอดที่ทำหน้าที่ในการจัดเก็บเมสเสจ
resolveMessageCall	เป็นเมธธอดที่ใช้ในการจับคู่เมสเสจหลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว
ชื่อคลาส (Class Name)	JClass
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้สำหรับจัดเก็บโครงสร้างของคลาสในรหัสโปรแกรม
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
addToken	เป็นเมธธอดที่รับโทเคนของเมสเสจที่เก็บได้ ส่งไปยังคลาสเมธธอด
resolveMessageCall	เป็นเมธธอดที่ส่งคำสั่งการแปลงโทเคนที่เก็บไว้เป็นเมสเสจและการจับคู่เมสเสจไปยังคลาสเมธธอด

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

ชื่อคลาส (Class Name)	Method
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้สำหรับจัดเก็บโครงสร้างของเมธธอดในรหัสโปรแกรม
แอททริบิวต์ (Attribute)	
methodCalled	เป็นแอททริบิวต์ที่เก็บจำนวนเมธธอดที่ถูกเรียกใช้งานในคลาส
เมธธอด (Method)	
addToken	เป็นเมธธอดที่รับโทเคนที่เก็บได้
addMethodCalled	เป็นเมธธอดสำหรับเพิ่มจำนวนเมธธอดที่ถูกเรียกใช้งาน
resolveMessageCall	เป็นเมธธอดที่ส่งคำสั่งการแปลงโทเคนเป็นเมสเสจ
ชื่อคลาส (Class Name)	Attribute
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้สำหรับจัดเก็บโครงสร้างของแอททริบิวต์ในซอร์สโค้ด
แอททริบิวต์ (Attribute)	
varCalled	เป็นแอททริบิวต์ที่เก็บจำนวนแอททริบิวต์ที่ถูกเรียกใช้งานในคลาส
เมธธอด (Method)	
addToken	เป็นเมธธอดที่รับโทเคนที่เก็บได้
addMethodCalled	เป็นเมธธอดสำหรับเพิ่มจำนวนแอททริบิวต์ที่ถูกเรียกใช้งาน
resolveMessageCall	เป็นเมธธอดที่ส่งคำสั่งการแปลงโทเคนเป็นเมสเสจ

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

ชื่อคลาส (Class Name)	FindMessageCall
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ในการคำนวณหาจำนวนการรับส่ง / เรียกใช้งานเมสเสจของคลาสทั้งหมดที่เป็นองค์ประกอบของซอฟต์แวร์
แอททริบิวต์ (Attribute)	
totalOfCalling	เป็นแอททริบิวต์ที่ทำหน้าที่จัดเก็บจำนวนการรับส่ง / เรียกใช้งานเมสเสจของคลาสทั้งหมดที่เป็นองค์ประกอบของซอฟต์แวร์
เมธธอด (Method)	
setMessage	เป็นเมธธอดที่ทำหน้าที่ในการจัดเก็บเมสเสจ
resolveMessageCall	เป็นเมธธอดที่ใช้ในการจับคู่เมสเสจหลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว
calculateNumberOfMesageCall	เป็นเมธธอดที่ทำหน้าที่ในการคำนวณหาจำนวนการรับส่ง / เรียกใช้เมสเสจทั้งหมดของคลาสทั้งหมดที่เป็นองค์ประกอบของซอฟต์แวร์ โดยรับค่าจากเมธธอด setMessage มาคำนวณ
ชื่อคลาส (Class Name)	TimePeriodReader
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ในการคำนวณหาระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์
แอททริบิวต์ (Attribute)	
startTime	เป็นแอททริบิวต์ที่ทำหน้าที่จัดเก็บเวลาเริ่มต้นที่ซอฟต์แวร์ประมวลผล โดยตั้งค่าเริ่มต้นให้เท่ากับศูนย์
timeStamp	เป็นแอททริบิวต์ที่ทำหน้าที่เก็บระยะเวลารวมที่ซอฟต์แวร์ประมวลผลเสร็จสมบูรณ์
timeFileTable	เป็นแอททริบิวต์ที่ทำหน้าที่เก็บระยะเวลาขณะที่ซอฟต์แวร์มีการประมวลผล
Runtime = Runtime.getRuntime ()	เป็นแอททริบิวต์ที่ใช้เชื่อมต่อระยะเวลาของจาวารันไทม์เพื่อให้เวลาของการประมวลผลผูกอยู่กับจาวารันไทม์

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

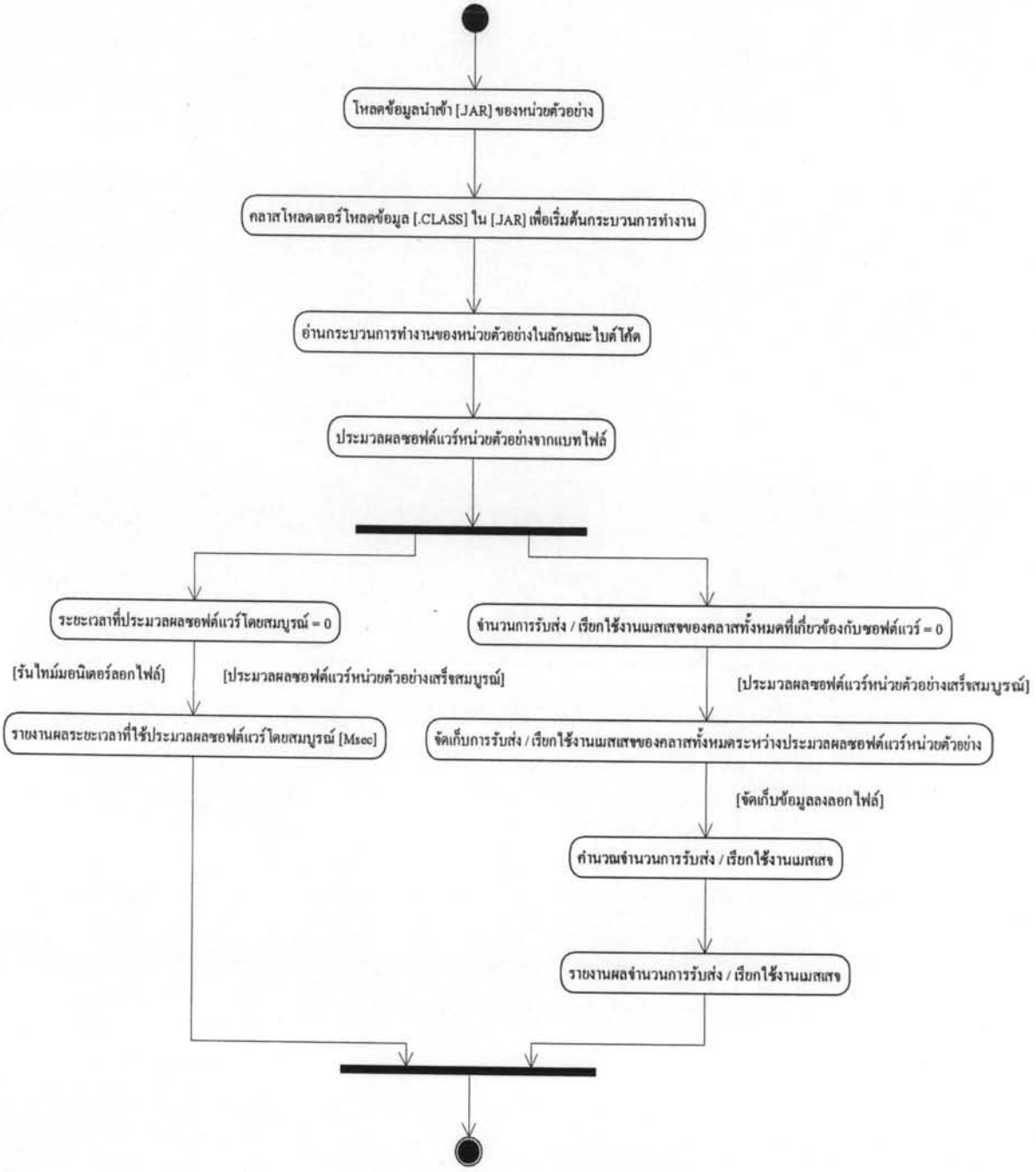
เมธธอด (Method)	
setMessage	เป็นเมธธอดที่ทำหน้าที่ในการจัดเก็บเมสเสจ
resolveMessageCall	เป็นเมธธอดที่ใช้ในการจับคู่เมสเสจหลังจากแปลงโทเคนเป็นเมสเสจเรียบร้อยแล้ว
calculateNumberOfMesageCall	เป็นเมธธอดที่ทำหน้าที่ในการคำนวณหาจำนวนการรับส่ง / เรียกใช้เมสเสจทั้งหมดของคลาสทั้งหมดที่เป็นองค์ประกอบของซอฟต์แวร์ โดยรับค่าจากเมธธอด setMessage มาคำนวณ
ชื่อคลาส (Class Name)	TimePeriodReader
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ในการคำนวณหาระยะเวลาที่ซอฟต์แวร์ถูกประมวลผลโดยสมบูรณ์
แอททริบิวต์ (Attribute)	
startTime	เป็นแอททริบิวต์ที่ทำหน้าที่จัดเก็บเวลาเริ่มต้นที่ซอฟต์แวร์ประมวลผล โดยตั้งค่าเริ่มต้นให้เท่ากับศูนย์
timeStamp	เป็นแอททริบิวต์ที่ทำหน้าที่เก็บระยะเวลารวมที่ซอฟต์แวร์ประมวลผลเสร็จสมบูรณ์
timeFileTable	เป็นแอททริบิวต์ที่ทำหน้าที่เก็บระยะเวลาขณะที่ซอฟต์แวร์มีการประมวลผล
Runtime = Runtime.getRuntime ()	เป็นแอททริบิวต์ที่ใช้เรียกการเชื่อมต่อระยะเวลาของจาวารันไทม์เพื่อให้ระยะเวลาของการประมวลผลซอฟต์แวร์ผูกอยู่กับจาวารันไทม์
เมธธอด (Method)	
readStartTime	เป็นเมธธอดที่ทำหน้าที่เริ่มกระบวนการอ่านเวลาเริ่มต้นที่ซอฟต์แวร์เริ่มประมวลผล
readTimePeriod	เป็นเมธธอดที่ทำหน้าที่บอกระยะเวลาที่ซอฟต์แวร์ใช้ระหว่างที่มีการประมวลผล
getTimeStamp	เป็นเมธธอดที่ทำหน้าที่บอกระยะเวลาสิ้นสุดที่ซอฟต์แวร์ประมวลผลเสร็จสมบูรณ์

ตารางที่ 4-2 คำอธิบายคลาสของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์ (ต่อ)

runTime	เป็นเมธธอดที่ทำหน้าที่ประมวลผลระยะเวลาที่ใช้ในการประมวลผลซอฟต์แวร์
generateTime	เป็นเมธธอดที่ทำหน้าที่บอกระยะเวลารวมทั้งหมดที่ซอฟต์แวร์ใช้ในการประมวลผลเสร็จสมบูรณ์
ชื่อคลาส (Class Name)	LogData
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ทำหน้าที่ในการจัดเก็บจำนวนการรับส่ง / เรียกใช้งานเมสเสจ และระยะเวลาที่ใช้ในการประมวลผลซอฟต์แวร์ เพื่อบอกประสิทธิภาพของซอฟต์แวร์
แอททริบิวต์ (Attribute)	
msgCount	เป็นแอททริบิวต์ที่จัดเก็บจำนวนการรับส่ง / เรียกใช้งานเมสเสจทั้งหมด
totResponseTime	เป็นแอททริบิวต์ที่จัดเก็บระยะรวมทั้งหมดที่ใช้ในการประมวลผลซอฟต์แวร์หนึ่ง ๆ ให้เสร็จสมบูรณ์
เมธธอด (Method)	
logData	เป็นเมธธอดที่ทำหน้าที่ในการจัดเก็บข้อมูลจำนวนการรับส่ง / เรียกใช้งานเมสเสจ และระยะเวลาที่ใช้ในการประมวลผลซอฟต์แวร์

4.1.5 แผนภาพแอคติวิตี้ (Activity Diagram) ของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

จากรูปที่ 4-4 เป็นแผนภาพแอคติวิตี้ของเครื่องมือสำหรับคำนวณหาประสิทธิภาพของซอฟต์แวร์ ซึ่งจะแสดงขั้นตอนวิธีการดำเนินงานในส่วนของการคำนวณจำนวนการเรียกใช้งานหรือการส่งเมสเสจระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์ทั้งหมดขณะประมวลผล และระยะเวลาที่ซอฟต์แวร์หนึ่ง ๆ ถูกประมวลผลโดยสมบูรณ์



รูปที่ 4-4 แผนภาพแอคติวิตี้ของเครื่องมือวัดประสิทธิภาพของซอฟต์แวร์

4.2 การออกแบบและพัฒนาเครื่องมือสำหรับวัดผลกระทบในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์

จากวัตถุประสงค์ของงานวิจัย ต้องการเปรียบเทียบผลกระทบที่เกิดจากการเปลี่ยนแปลงซอฟต์แวร์ตามความต้องการเชิงฟังก์ชันระหว่างโครงสร้างคลาสในความสัมพันธ์แบบแอสโซซิเอชันและโครงสร้างคลาสในความสัมพันธ์แบบเจเนอรัลไลเซชัน ดังนั้นการออกแบบและพัฒนาเครื่องมือสำหรับทดสอบงานวิจัยส่วนนี้ สามารถอธิบายได้ดังนี้

4.2.1 ความต้องการเชิงฟังก์ชันของเครื่องมือ (Functional Requirement)

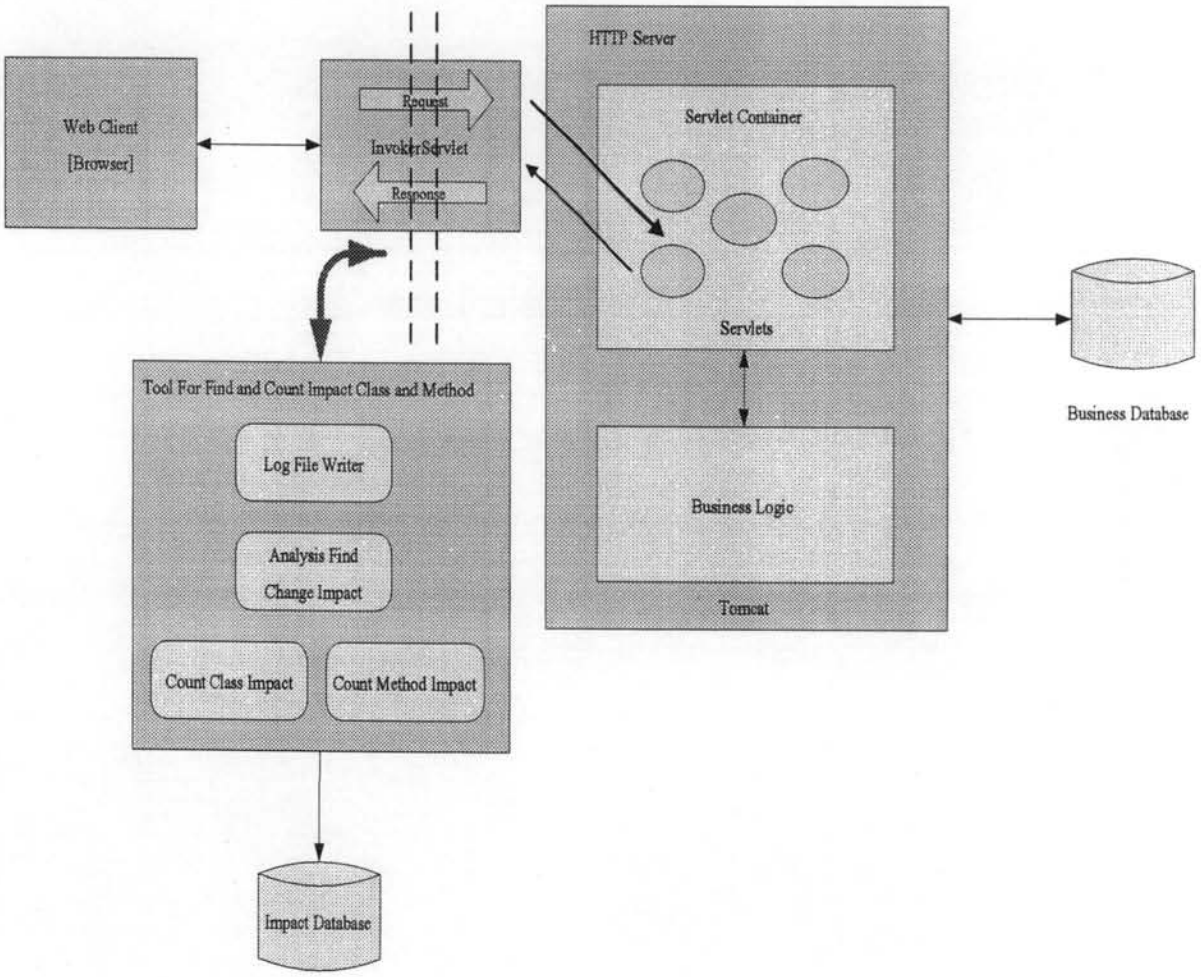
การหาผลกระทบของซอฟต์แวร์ที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เชิงฟังก์ชันของโครงสร้างคลาสในความสัมพันธ์แบบแอสโซซิเอชันและโครงสร้างคลาสในความสัมพันธ์เจเนอรัลไลเซชัน ทำได้โดยการคำนวณจากจำนวนคลาสและจำนวนเมธธอดที่ได้รับผลกระทบเมื่อมีการเปลี่ยนแปลงความต้องการเชิงฟังก์ชัน (Functional Requirement) ของซอฟต์แวร์ ซึ่งผู้วิจัยได้ออกแบบและและพัฒนาเครื่องมือที่ใช้ในการทดสอบงานวิจัยเป็นจาวาแพคเกจ (Java Package) ไว้สำหรับให้ซอฟต์แวร์กรณีศึกษานำไปเรียกใช้งาน โดยเป็นโปรแกรมประเภทเว็บแอพพลิเคชัน (Web Application Programming Language) โดยมีการเรียกใช้งานจาวาคลาสไลบรารี (Java Class Libraries) ของชุดโปรแกรมสำหรับพัฒนาโปรแกรมภาษาจาวา ในส่วนของเซิร์ฟเลท (Servlets) ที่ใช้เขียนโปรแกรมภาษาจาวาเพื่อใช้งานร่วมกับเว็บเซิร์ฟเลท (Web Servlets) และเพื่อให้เครื่องมือที่พัฒนาและซอฟต์แวร์หน่วยตัวอย่างสามารถประมวลผลได้ จึงต้องมีการติดตั้งจาวาเซิร์ฟเวอร์ (Java Server) ใช้เป็นเว็บเซิร์ฟเวอร์ที่ใช้งานร่วมกับเซิร์ฟเลท โดยงานวิจัยนี้ผู้วิจัยได้เลือกใช้แอพพลิเคชันอะพาชชีทอมแคท เวอร์ชัน 5.5 (Apache Tomcat version 5.5) นอกเหนือจากนี้ต้องมีการติดตั้งฐานข้อมูลมายเอสคิวแอลเซิร์ฟเวอร์เวอร์ชัน 4.1 (MySQL Server version 4.1) เพิ่มเติมด้วย

เพื่อให้เครื่องมือที่ออกแบบและพัฒนาขึ้นสามารถทำงานได้โดยตอบวัตถุประสงค์ของงานวิจัยได้ ผู้วิจัยจึงได้จัดทำเป็นจาวาแพคเกจเพื่อให้ซอฟต์แวร์กรณีศึกษานำไปติดตั้งแบบแอดอิน (Add - in) แล้วให้หน่วยตัวอย่างที่ต้องการทดสอบนำไปเรียกใช้งาน โดยเครื่องมือประกอบด้วยฟังก์ชันการทำงานต่าง ๆ ดังนี้

1. เมื่อต้องการหาจำนวนผลกระทบของซอฟต์แวร์ที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เชิงฟังก์ชัน ผู้ใช้ต้องนำซอฟต์แวร์แพคเกจเข้าไปติดตั้งในอินโวคเซิร์ฟเลท (InvokeSerlet) โดยเมื่อมีการเพิ่มหน่วยตัวอย่างที่เป็นความต้องการเชิงฟังก์ชันของซอฟต์แวร์จาวาแพคเกจจะเรียกซอร์สโค้ดที่ถูกเปิดใช้งานอยู่จากโปรแกรม

สำหรับพัฒนาภาษาโปรแกรมจาวาอิคลิพซ์ขึ้นมาวิเคราะห์และคำนวณหาค่าผลกระทบที่เกิดจากการเปลี่ยนแปลงซอฟต์แวร์

2. เครื่องมือสามารถคำนวณและแสดงผลจำนวนคลาสที่ถูกกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ได้
3. เครื่องมือสามารถคำนวณและแสดงผลจำนวนเมธธอดที่ถูกกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ได้
4. เครื่องมือสามารถบันทึกผลจำนวนคลาสและจำนวนเมธธอดที่ถูกกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ลงในฐานข้อมูลมายเอสคิวแอลได้



รูปที่ 4-5 สถาปัตยกรรมเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์

จากรูปที่ 4-5 แสดงสถาปัตยกรรมของเครื่องมือวัดผลกระทบในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ โดยมีรายละเอียดการทำงานดังนี้

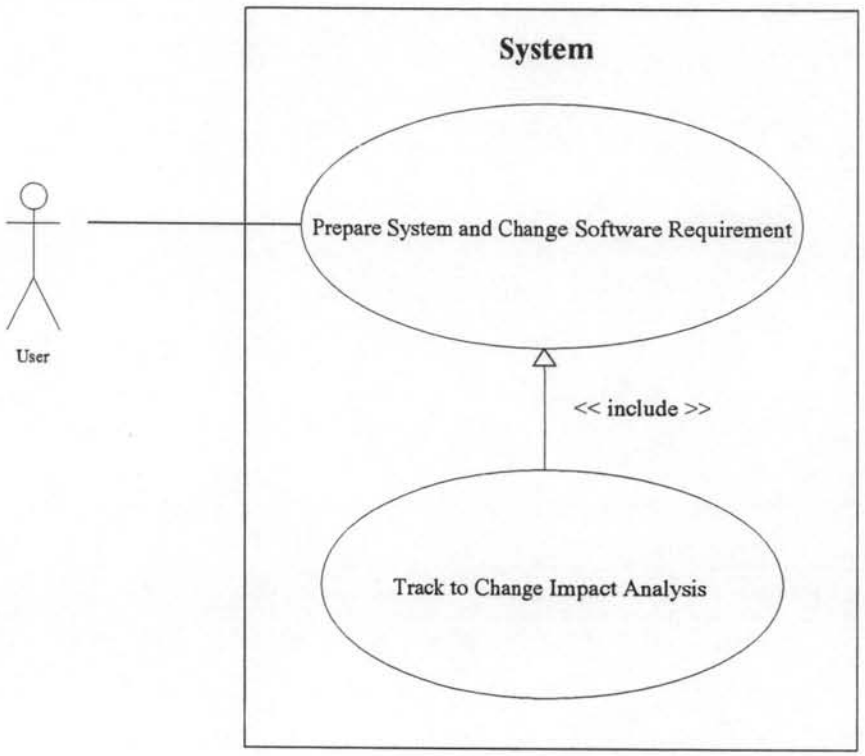
เซิร์ฟเลท (Servlets) คือรูปแบบของโปรแกรมภาษาจาวา โดยเซิร์ฟเลทต้องทำงานภายใต้สภาวะแวดล้อมพิเศษที่เรียกว่าเซิร์ฟเลทคอนเทเนอร์ (Servlet Container) ซึ่งจะถูกสร้างขึ้นในเฮชทีทีพีเซิร์ฟเวอร์ (HTTP Server) เพื่อทำงานเซิร์ฟเลท (วีระศักดิ์ ซึงถาวร, 2547) และภายในมีการอิมพลีเมนต์ออบเจคท์คลาสเชิงธุรกิจของซอฟต์แวร์กรณีศึกษาด้วย

โดยเซิร์ฟเลทจะถูกเรียกทำงานจากโปรแกรมไคลเอนต์ (Client) ที่เป็นเว็บเบราเซอร์ (Browser) หรือจาวาแอพพลิเคชัน (Java Application) ซึ่งไคลเอนต์จะติดต่อกับเซิร์ฟเลทได้โดยใช้วิธีการรีเควสท์ (Request) และรีสพอนซ์ (Response) ผ่านอินโวคเซิร์ฟเลท (InvokeSerlet)

เครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์จะถูกติดตั้งแบบแอดอินภายในอินโวคเซิร์ฟเลท เพื่อตรวจสอบการทำงานของซอฟต์แวร์กรณีศึกษา เมื่อมีการเปลี่ยนแปลงซอฟต์แวร์กรณีศึกษาตามความต้องการเชิงฟังก์ชัน จาวาแอพพลิเคชันของซอฟต์แวร์กรณีศึกษาจะติดต่อกับเซิร์ฟเลทและออบเจคท์คลาสเชิงธุรกิจได้โดยใช้รีเควสท์และรีสพอนซ์ผ่านทางอินโวคเซิร์ฟเลท ซึ่งเครื่องมือจะประมวลผลเพื่อวิเคราะห์และจัดเก็บจำนวนคลาสและจำนวนเมธธอดที่ได้รับผลกระทบจากประมวลผลซอฟต์แวร์กรณีศึกษา ซึ่งทำได้โดยการนำลอกไฟล์ที่บันทึกไว้มาวิเคราะห์และบันทึกผลจำนวนคลาสและจำนวนเมธธอดที่ถูกกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ลงในฐานข้อมูลมายเอสคิวแอล

4.2.2 แผนภาพยูสเคสและคำอธิบายยูสเคส (Use Case Diagram and Use Case Description) ของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

รูปที่ 4-6 เป็นแผนภาพยูสเคสของเครื่องมือสำหรับคำนวณหาจำนวนคลาสและจำนวนเมทธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เชิงฟังก์ชัน โดยสามารถอธิบายรายละเอียดการทำงานของยูสเคสได้ ดังตารางที่ 4-3



รูปที่ 4-6 แผนภาพของยูสเคสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

ตารางที่ 4-3 คำอธิบายยูสเคสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

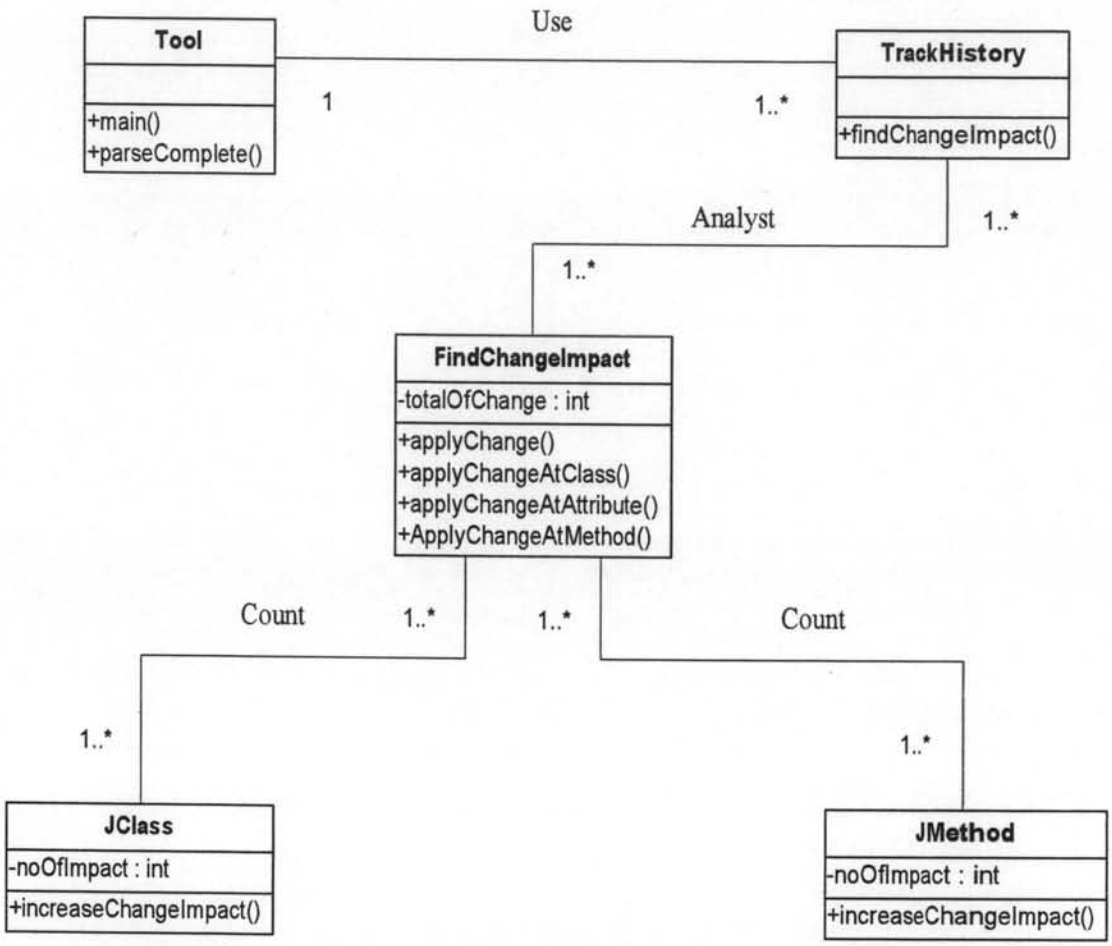
1. ยูสเคสการจัดเตรียมหน่วยตัวอย่างและเปลี่ยนแปลงความต้องการของซอฟต์แวร์ (Prepare System and Change Software Requirement)	
เป้าหมาย (Goal)	จัดเตรียมหน่วยตัวอย่างและเปลี่ยนแปลงความต้องการของซอฟต์แวร์
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Pre - condition)	นำเครื่องมือที่เป็นจาวาแพคเกจเข้ามาติดตั้งให้ซอฟต์แวร์กรณีศึกษาและระบบเรียกใช้ข้อมูลจากชุดเครื่องมือพัฒนาโปรแกรมอิคลิพซ์ รวมทั้งรันจาวาเซิร์ฟเวอร์และติดต่อฐานข้อมูลเรียบร้อยแล้ว
เงื่อนไขหลัง (Post - condition)	ระบบจัดเตรียมข้อมูลสำหรับการคำนวณผลกระทบที่เกิดจากการเปลี่ยนแปลงซอฟต์แวร์เรียบร้อยแล้ว
เมนซัคเซส ซีนาริโอ (Main Success Scenario)	1. ผู้ใช้เปลี่ยนแปลงความต้องการของซอฟต์แวร์ที่เป็นหน่วยตัวอย่าง 2. ระบบประมวลผลซอฟต์แวร์กรณีศึกษาตามความต้องการของซอฟต์แวร์ที่เปลี่ยนแปลงไป

ตารางที่ 4-3 คำอธิบายยูสเคสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ (ต่อ)

2. ยูสเคสการติดตามและวิเคราะห์ผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ (Track to Change Impact Analysis)	
เป้าหมาย (Goal)	วิเคราะห์และรายงานจำนวนคลาสและจำนวนเมธธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
แอกเตอร์ (Actor)	ผู้ใช้
เงื่อนไขก่อน (Pre - condition)	ระบบเปลี่ยนแปลงความต้องการของซอฟต์แวร์และประมวลผลซอฟต์แวร์เรียบร้อยแล้ว
เงื่อนไขหลัง (Post - condition)	ระบบวิเคราะห์และบันทึกผลจำนวนคลาสและเมธธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ลงในฐานข้อมูล
เมนซัคเซส ซีนาริโอ (Main Success Scenario)	1. ระบบคำนวณจำนวนคลาสและจำนวนเมธธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ 2. ระบบบันทึกจำนวนคลาสและจำนวนเมธธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ลงในฐานข้อมูล

4.2.3 แผนภาพคลาสและคำอธิบายคลาส (Class Diagram and Class Description) ของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

รูปที่ 4-7 เป็นแผนภาพคลาสของเครื่องมือสำหรับคำนวณหาจำนวนคลาสและจำนวนเมทธอดที่ได้รับผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เชิงฟังก์ชัน โดยจะแสดงเฉพาะคลาสที่เป็นกระบวนการในการวิเคราะห์และหาผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เท่านั้น และรายละเอียดของแต่ละคลาสแสดงได้ดังตารางที่ 4-4



รูปที่ 4-7 แผนภาพคลาสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

ตารางที่ 4-4 คำอธิบายคลาสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

ชื่อคลาส (Class Name)	Tool
คำอธิบายคลาส (Class Description)	เป็นคลาสหลักของการหาผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
main	เป็นเมธธอดหลัก และเป็นเมธธอดแรกที่ถูกเรียกใช้งานเมื่อต้องการหาผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
parseComplete	เป็นเมธธอดที่ถูกเรียกใช้งานหลังจากเก็บข้อมูลจากซอร์สโค้ด เพื่อเริ่มขั้นตอนการเปลี่ยนแปลงซอร์สโค้ดตามความต้องการของซอฟต์แวร์และหาผลกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
ชื่อคลาส (Class Name)	TrackHistory
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้จัดการกับข้อมูลต่าง ๆ ที่เก็บมาจากซอร์สโค้ดและส่งคำสั่งในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
แอททริบิวต์ (Attribute)	
-	-
เมธธอด (Method)	
findChangeImpact	เป็นเมธธอดสำหรับส่งคำสั่งการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ และหาผลกระทบที่เกิดจากการเปลี่ยนแปลงไปยังคลาส FindChangeImpact

ตารางที่ 4-4 คำอธิบายคลาสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ (ต่อ)

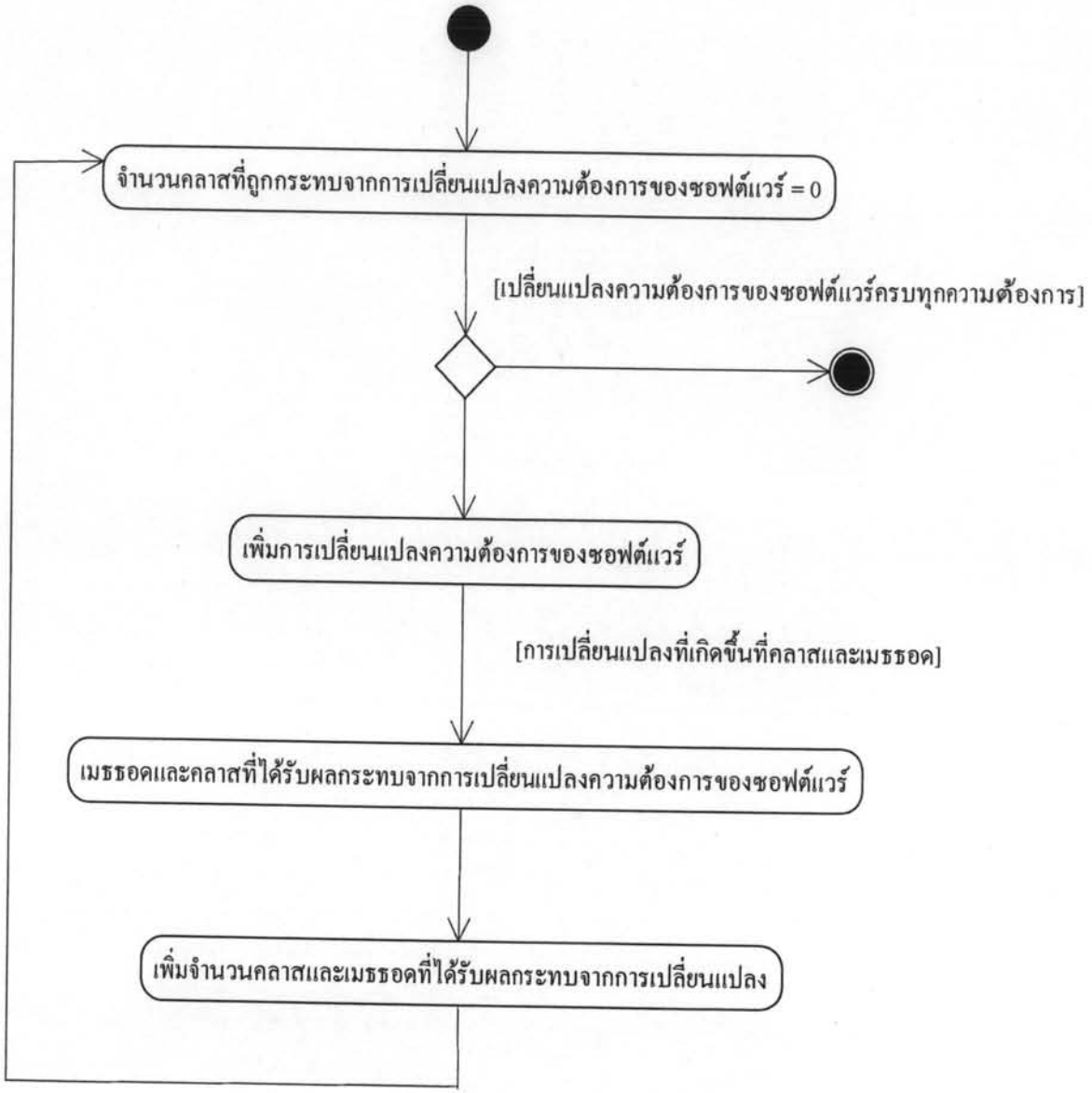
ชื่อคลาส (Class Name)	FindChangeImpact
คำอธิบายคลาส (Class Description)	เป็นคลาสสำหรับดำเนินการการเปลี่ยนแปลงความต้องการและหาผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
แอททริบิวต์ (Attribute)	
totalOfChange	เป็นแอททริบิวต์ที่ใช้จัดเก็บจำนวนการเปลี่ยนแปลงทั้งหมด
เมธธอด (Method)	
applyChange	เป็นเมธธอดที่ใช้ควบคุมการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ให้กับทุกคลาสในซอร์สโค้ด
applyChangeAtClass	เป็นเมธธอดที่ใช้ควบคุมการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ให้กับแต่ละคลาสในซอร์สโค้ด และหาผลกระทบที่เกิดจากการเปลี่ยนแปลงนั้น
applyChangeAtAttribute	เป็นเมธธอดที่ใช้ควบคุมการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ให้กับแอททริบิวต์ในซอร์สโค้ด และหาผลกระทบที่เกิดจากการเปลี่ยนแปลงนั้น
applyChangeAtMethod	เป็นเมธธอดที่ใช้ควบคุมการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ให้กับเมธธอดในซอร์สโค้ด และหาผลกระทบที่เกิดจากการเปลี่ยนแปลงนั้น
ชื่อคลาส (Class Name)	JClass
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้สำหรับจัดเก็บโครงสร้างของคลาสในซอร์สโค้ด
แอททริบิวต์ (Attribute)	
noOfImpact	เป็นแอททริบิวต์ที่ใช้สำหรับจัดเก็บจำนวนผลกระทบที่เกิดขึ้นกับคลาสในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์

ตารางที่ 4-4 คำอธิบายคลาสของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์ (ต่อ)

เมธธอด (Method)	
increaseChangeImpact	เป็นเมธธอดที่ทำหน้าที่เพิ่มจำนวนผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
ชื่อคลาส (Class Name)	JMethod
คำอธิบายคลาส (Class Description)	เป็นคลาสที่ใช้สำหรับจัดเก็บโครงสร้างของเมธธอดในซอร์สโค้ด
แอททริบิวต์ (Attribute)	
noOfImpact	เป็นแอททริบิวต์ที่ใช้สำหรับจัดเก็บจำนวนผลกระทบที่เกิดขึ้นกับเมธธอดในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์
เมธธอด (Method)	
increaseChangeImpact	เป็นเมธธอดที่ทำหน้าที่เพิ่มจำนวนผลกระทบที่เกิดจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์

4.2.4 แผนภาพแอคติวิตี้ (Activity Diagram) ของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

จากรูปที่ 4-8 เป็นแผนภาพแอคติวิตี้ของเครื่องมือหาผลกระทบที่เกิดขึ้นจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ ซึ่งจะแสดงขั้นตอนวิธีการดำเนินงานในส่วนของการหาจำนวนคลาสและจำนวนเมธธอดที่ถูกกระทบจากการเปลี่ยนแปลงความต้องการของซอฟต์แวร์เท่านั้น



รูปที่ 4-8 แผนภาพแอคติวิตี้ของเครื่องมือวัดผลกระทบจากการเปลี่ยนแปลงซอฟต์แวร์

4.3 การทดสอบความถูกต้องของเครื่องมือ

เมื่อผู้วิจัยได้พัฒนาเครื่องมือที่ใช้ทดสอบงานวิจัยเรียบร้อยแล้ว ผู้วิจัยได้มีการจัดการทดสอบความถูกต้องของเครื่องมือ เพื่อลดความผิดพลาดในการประมวลผลของเครื่องมือที่พัฒนาขึ้น ซึ่งงานวิจัยมีการสร้างเครื่องมือสำหรับทดสอบงานวิจัย 2 ชุด คือ (1) เครื่องมือสำหรับวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล และ (2) เครื่องมือสำหรับวัดผลกระทบในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์ โดยสามารถอธิบายได้ดังนี้

4.3.1 การทดสอบเครื่องมือสำหรับวัดประสิทธิภาพของซอฟต์แวร์ขณะประมวลผล

ผู้วิจัยเริ่มทดสอบเครื่องมือตั้งแต่ขั้นตอนการพัฒนาเป็นโมดูลย่อย ๆ โดยใช้หลักการทดสอบแบบหน่วยต่อหน่วย (Unit Testing) ซึ่งการทดสอบในขั้นตอนนี้เป็นการช่วยลดความผิดพลาดของเครื่องมือที่พัฒนาขึ้นในแต่ละโมดูลย่อย โดยผู้วิจัยเลือกใช้กรณีทดสอบอย่างง่าย เช่น การคำนวณจำนวนการรับส่งหรือเรียกใช้เมสเสจระหว่างคลาสที่เป็นองค์ประกอบของซอฟต์แวร์ทั้งหมด ผู้วิจัยได้ดำเนินการสร้างซอร์สโค้ดที่ไม่มีความซับซ้อนในการทำงาน และไม่มีการเรียกใช้งานจาวาคลาสไลบรารีขึ้นมา จากนั้นประมวลผลซอร์สโค้ดที่สร้างขึ้นผ่านจอเฝ้าคุม (Console) เพื่อสร้างลอกไฟล์การประมวลผลการทำงานของซอร์สโค้ดที่สร้างขึ้น แล้วนับจำนวนเมสเสจที่มีการรับส่งระหว่างประมวลผลเพื่อเปรียบเทียบกับจำนวนการเรียกใช้เมสเสจที่ประมวลผลผ่านเครื่องมือที่สร้างขึ้น ส่วนการทดสอบระยะเวลาที่ใช้ในการประมวลผลซอฟต์แวร์โดยสมบูรณ์นั้น ผู้วิจัยได้เรียกใช้ฟังก์ชันการจัดการเวลาจากจาวาคลาสไลบรารีมาใช้ เนื่องด้วยเป็นส่วนที่ชุดโปรแกรมพัฒนาภาษาจาวามีไว้ให้เรียกใช้บริการอยู่แล้ว

4.3.2 การทดสอบเครื่องมือสำหรับวัดผลกระทบในการเปลี่ยนแปลงความต้องการของซอฟต์แวร์

เนื่องจากเครื่องมือที่พัฒนาขึ้นมาในส่วนนี้เป็นลักษณะของซอฟต์แวร์แพคเกจที่นำไปติดตั้งแบบแอดอินในซอฟต์แวร์กรณีศึกษาที่ต้องการทำวิจัย ดังนั้นผู้วิจัยได้พัฒนาซอร์สโค้ดที่ไม่ความซับซ้อนมากนักขึ้นมา แล้วประมวลผลซอร์สโค้ดผ่านจอเฝ้าคุม เพื่อสร้างลอกไฟล์จัดเก็บคลาสและเมธธอดที่ถูกเรียกขึ้นมาทำงานขณะประมวลผลซอร์สโค้ด จากนั้นจึงเพิ่มความต้องการเชิงฟังก์ชันเข้าสู่ซอร์สโค้ดที่สร้างขึ้นแล้วประมวลผลผ่านจอเฝ้าคุมอีกครั้งหนึ่ง แล้วจัดเก็บลอกไฟล์ จากนั้นนำลอกไฟล์จากการประมวลผลซอร์สโค้ดทั้งสองครั้งมาเปรียบเทียบกันเพื่อหาคลาสและเมธธอดที่ถูกเรียกขึ้นมาทำงาน ถ้าการประมวลผลทั้งสองครั้งลอกไฟล์มีค่าไม่แตกต่างกันแสดงว่าการเพิ่มความต้องการเชิงฟังก์ชันของซอฟต์แวร์ไม่มีผลกระทบซอร์สโค้ดที่สร้างขึ้นมา หากลอกไฟล์มีค่าแตกต่างกันแสดงว่าการเพิ่มความต้องการมีผลกระทบกับซอร์สโค้ดที่สร้างขึ้น จากนั้นนำซอร์สโค้ดที่สร้างขึ้นมาติดตั้งซอฟต์แวร์แพคเกจของเครื่องมือแล้วประมวลผลผ่านซอฟต์แวร์แพคเกจ จากนั้นนำผลลัพธ์ที่ได้จากซอฟต์แวร์แพคเกจและผลลัพธ์จากลอกไฟล์มาเปรียบเทียบกันว่าผลลัพธ์ที่ได้มีความถูกต้องตรงกันหรือไม่