บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การเปลี่ยนแปลงซอฟต์แวร์ (Software Change) คือ การเปลี่ยนแปลงองค์ประกอบต่างๆ ของซอฟต์แวร์ เช่น ซอร์สโคด (Source Code) อินเตอร์เฟส (Interface) หรือ แผนภาพการออกแบบ (Design Diagram) เป็นต้น โดยมีวัตถุประสงค์เพื่อแก้ไขข้อผิดพลาดของซอฟต์แวร์ เพิ่มเติมความสามารถของซอฟต์แวร์ หรือปรับปรุงประสิทธิภาพของซอฟต์แวร์ โดยเมื่อมีการเปลี่ยนแปลงย่อมเกิดผลกระทบตามมา เรียกว่าผลกระทบจากการเปลี่ยนแปลง (Change Impact) โดยผลกระทบที่เกิดขึ้นสามารถกระจาย (ripple) ไปยังส่วนอื่นๆ ของซอฟต์แวร์ได้ ไม่จำกัดอยู่เพียงส่วนที่เกิดการเปลี่ยนแปลง การกระจายของผลกระทบไปยังส่วนอื่นๆ ของซอฟต์แวร์เรียกว่าการเกิดผลกระทบวงคลื่น (Ripple Effect) การเกิดผลกระทบวงคลื่นไปยังส่วนต่างๆ ของซอฟต์แวร์ทำให้การเปลี่ยนแปลงซอฟต์แวร์มีความยุ่งยากยิ่งขึ้น ทั้งนี้ผลกระทบที่เกิดขึ้นอาจทำให้ซอฟต์แวร์ทำงานผิดพลาด หรือไม่สามารถทำงานได้เลย ดังนั้นในการเปลี่ยนแปลงซอฟต์แวร์จึงจำเป็นต้องทำการวิเคราะห์ผลกระทบของการเปลี่ยนแปลง (Change Impact Analysis) เพื่อให้ทราบถึงผลกระทบที่อาจเกิดขึ้นหากทำการเปลี่ยนแปลง ทั้งนี้เพื่อวางแผนและจัดการกับผลกระทบที่อาจเกิดขึ้นไม่ให้กลายเป็นปัญหาหรือข้อผิดพลาดใหม่ขึ้นมา

ในการวิเคราะห์ผลกระทบของการเปลี่ยนแปลงมีวิธีการวิเคราะห์และประเมินระดับของผลกระทบหลายวิธี ตามแต่การนิยามความหมายของการวิเคราะห์ผลกระทบการเปลี่ยนแปลงของแต่ละผู้วิจัย เช่น ได้มีการให้คำนิยามไว้ใน [1] กล่าวว่าการวิเคราะห์ผลกระทบการเปลี่ยนแปลงคือการประเมินความเสี่ยงที่เกิดจากการเปลี่ยนแปลงรวมไปถึงการประเมินผลกระทบต่อทรัพยากรแรงงานและเวลา และได้มีการให้คำนิยามไว้ใน [2] ว่าการวิเคราะห์ผลกระทบการเปลี่ยนแปลงคือการหาผลลัพธ์ที่เกิดจากการเปลี่ยนแปลง และประเมินความต้องการในการแก้ไขให้สอดคล้องกับการเปลี่ยนแปลง เป็นต้น และในการวิเคราะห์ผลกระทบการเปลี่ยนแปลงมีตัวชี้วัดหนึ่งสามารถบอกถึงระดับของผลกระทบที่จะเกิดขึ้นเมื่อทำการแก้ไขซอฟต์แวร์ คือความเสถียรของซอฟต์แวร์ (Software Stability) โดยความเสถียรของซอฟต์แวร์คือ โอกาสในการไม่ได้รับผลกระทบจากการเปลี่ยนแปลงของซอฟต์แวร์หรืออาจระบุเป็นส่วนกลับของผลกระทบวงคลื่น โดยซอฟต์แวร์ที่มี

ความเสถียรสูงคือ ซอฟต์แวร์ที่มีโอกาสได้รับผลกระทบจากการเปลี่ยนแปลงต่ำทำให้สามารถทำการเปลี่ยนแปลงแก้ไขได้อย่างสะดวกและรวดเร็ว

ความเสถียรของซอฟต์แวร์สามารถแบ่งออกได้หลายแบบด้วยกันตามผลกระทบที่เกิดขึ้น แต่โดยส่วนใหญ่มักแบ่งออกเป็น ความเสถียรเชิงตรรกะ (Logical Stability)[3] และความเสถียรเชิงประสิทธิภาพ (Performance Stability)[3] โดยความเสถียรเชิงตรรกะคือ โอกาสในการไม่ได้รับผลกระทบที่ส่งผลให้การทำงานผิดพลาด และความเสถียรเชิงประสิทธิภาพคือ โอกาสในการไม่ได้รับผลกระทบที่ส่งผลให้ประสิทธิภาพในการทำงานเปลี่ยนไป โดยรายละเอียดของความเสถียรทั้ง 2 แบบจะอธิบายในบทที่ 2 ปัจจุบันมีผู้ทำการวิจัยเกี่ยวกับความเสถียรเชิงตรรกะอยู่มาก แต่โดยส่วนใหญ่เป็นการคำนวณความเสถียรเชิงตรรกะจากซอร์สโคดทั้งสิ้น ทำให้ไม่สามารถทำการวิเคราะห์เพื่อปรับปรุงซอฟต์แวร์ให้มีความเสถียรสูงได้ตั้งแต่ขั้นตอนการออกแบบ จำเป็นต้องรอจนซอฟต์แวร์พัฒนาเสร็จจึงสามารถทำการวัดค่าความเสถียรเชิงตรรกะได้ นอกจากนี้จากทฤษฎีค่าใช้จ่ายของการเปลี่ยนแปลงของ Boehm [4] ระบุว่าการเปลี่ยนแปลงซอฟต์แวร์ในช่วงหลังของการพัฒนาจะเสียค่าใช้จ่ายมากกว่าในช่วงแรกมาก ดังนั้นหากสามารถทำการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ได้ตั้งแต่ช่วงต้นของการพัฒนาจะช่วยในการตัดสินใจของผู้พัฒนาเพื่อทำการแก้ไขปรับปรุงการออกแบบซอฟต์แวร์ต่อไป

วิทยานิพนธ์นี้จึงมีวัตถุประสงค์เพื่อหาโมเดลในการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์จากแบบจำลองที่ใช้ในการออกแบบซอฟต์แวร์ โดยในวิทยานิพนธ์นี้ผู้วิจัยได้นิยามความเสถียรเชิงตรรกะเป็นส่วนกลับผลกระทบวงคลื่นเชิงตรรกะ และผู้วิจัยได้เลือกทำการวิจัยแผนภาพคลาสและแผนภาพซีเควนซ์ ทั้งนี้เพราะปัจจุบันการพัฒนาซอฟต์แวร์อาศัยแนวความคิดเชิงวัตถุและแผนภาพยูเอ็มแอลเป็นหลัก และแผนภาพทั้ง 2 เป็นแผนภาพที่ช่วยให้เห็นโครงสร้างและขั้นตอนการทำงานของซอฟต์แวร์ที่ออกแบบได้อย่างชัดเจน

ผลลัพธ์ของวิทยานิพนธ์นี้คือ โมเดลในการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์จากแผนภาพคลาสและแผนภาพซีเควนซ์ และซอฟต์แวร์ที่ใช้ในการประมาณค่าโดยรับข้อมูลเป็นแผนภาพคลาสและแผนภาพซีเควนซ์ที่ทำการแปลงเป็นรหัสข้อมูลเอกซ์เอ็มไอแล้ว จากนั้นทำการประมวลผลและประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ เพื่อทำการสรุปผลแจ้งแก่ผู้ใช้ถึงความเสถียรเชิงตรรกะของซอฟต์แวร์ที่ออกแบบ

1.2 วัตถุประสงค์

1. สร้างโมเดลในการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์จากแผนภาพคลาสและแผนภาพซีเควนซ์

2. พัฒนาเครื่องมือประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์เมื่อพัฒนาเสร็จจากแผนภาพคลาสและแผนภาพซีเควนซ์

1.3 ขอบเขตงานวิจัย

1. โมเดลในการประมาณค่าความเสถียรเชิงตรรกะในวิทยานิพนธ์นี้สำหรับซอฟต์แวร์ที่พัฒนาตามโมเดลการออกแบบซอฟต์แวร์เชิงวัตถุด้วย แผนภาพคลาสและแผนภาพซีเควนซ์ของยูเอ็มแอลเวอร์ชัน 1.4 และพัฒนาด้วยภาษาจาวาเท่านั้น
2. การวัดความเสถียรเชิงตรรกะจากซอร์สโคดทำการวิเคราะห์ผลกระทบจากการเปลี่ยนแปลงในตารางที่ 3.1 เท่านั้น
3. วิทยานิพนธ์นี้ทำการหาโมเดลประมาณค่าความเสถียรเชิงตรรกะเท่านั้นมิได้เสนอแนะแนวทางการปรับปรุงความเสถียรเชิงตรรกะ
4. พัฒนาซอฟต์แวร์สำหรับประมาณค่าความเสถียรเชิงตรรกะตามโมเดลที่ได้จากการทดลอง โดยมีข้อจำกัดดังนี้
   - ข้อมูลเข้าเป็นแผนภาพที่แปลงอยู่ในรูปแบบเอกซ์เอ็มไอเท่านั้น
   - ผลลัพธ์ที่ได้เป็นตัวเลขแสดงผลในรูปแบบตาราง โดยแสดงค่าความเสถียรเชิงตรรกะที่ประมาณได้ของแต่ละคลาส ค่าตัววัดแผนภาพของแต่ละคลาส และค่าความเสถียรเชิงตรรกะของซอฟต์แวร์
   - พัฒนาโดยภาษาจาวา
   - สามารถทำงานได้บนระบบปฏิบัติการไมโครซอฟท์วินโดวส์ และระบบปฏิบัติการยูนิกซ์ที่รองรับภาษาจาวา

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. ศึกษาและค้นหาโมเดลวัดค่าความเสถียรเชิงตรรกะที่เหมาะสมต่อการทดลองจากบทความวิชาการต่างๆ
2. สร้างและเลือกซอฟต์แวร์สำหรับการทดลองตามคุณสมบัติที่กำหนด
3. วัดค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ที่เลือกมา
4. แปลงซอฟต์แวร์ที่เลือกมาเป็นแผนภาพคลาสและแผนภาพซีเควนซ์ด้วยโปรแกรม MagicDraw UML 9.5
5. แปลงแผนภาพที่ได้ให้อยู่ในรูปแบบภาษาเอกซ์เอ็มไอด้วยโปรแกรม MagicDraw UML 9.5

6. วัดค่าตัววัดแผนภาพที่ได้ทำการแปลงให้อยู่ในรูปแบบภาษาเอกซ์เอ็มไอแล้วด้วยโปรแกรม SDMetrics 2.0
7. หาสมการในการประมาณค่าความเสถียรเชิงตรรกะจากแผนภาพด้วยวิธีการวิเคราะห์ถดถอยเชิงเส้นแบบหลายตัวแปร
8. ตรวจสอบความถูกต้องของโมเดลที่ได้จากการทดลอง
9. สร้างซอฟต์แวร์เพื่อทำการประมาณค่าความเสถียรเชิงตรรกะ
10. ตรวจสอบความถูกต้องของซอฟต์แวร์ที่พัฒนาขึ้น

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โมเดลในการประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์ที่พัฒนาเสร็จแล้วจากแผนภาพคลาสและแผนภาพซีเควนซ์
2. ได้ซอฟต์แวร์ในการวิเคราะห์แผนภาพคลาสและซีเควนซ์เพื่อประมาณค่าความเสถียรเชิงตรรกะของซอฟต์แวร์เมื่อพัฒนาเสร็จสิ้น และสรุปข้อมูลเพื่อให้ผู้ใช้ได้ตัดสินใจเพื่อปรับปรุงการออกแบบซอฟต์แวร์ต่อไป