บทที่ 1

บทนำ

ในบทนี้จะกล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของงานวิจัย ขอบเขตของการวิจัย ขั้นตอนในการวิจัย และประโยชน์ที่คาดว่าจะได้รับจากงานวิจัยนี้ ซึ่งแต่ละส่วนมีรายละเอียดดังนี้

1.1 ความเป็นมาและความสำคัญของปัญหา

ในกระบวนการพัฒนาซอฟต์แวร์ (Software Development Process) จะเริ่มต้นด้วยการเก็บรวบรวมความต้องการและวิเคราะห์ (Requirements gathering and analysis phase) ถึงความต้องการของซอฟต์แวร์ที่จะทำการสร้างขึ้นมา จากนั้นจะเข้าสู่ช่วงการออกแบบ (Design phase) ซอฟต์แวร์ด้วยวิธีต่างๆ เช่น การออกแบบเชิงโครงสร้าง (Structural design) หรือการออกแบบเชิงวัตถุ (Object-oriented design) เป็นต้น แต่ในปัจจุบันได้รับความนิยมการออกแบบเชิงวัตถุกันมากขึ้นเนื่องมาจากสามารถแบ่งระบบที่จะออกแบบออกเป็นส่วนย่อยอิสระจากกันได้ ซึ่งทำให้ง่ายต่อการพัฒนาในขั้นต่อไป หลังจากทำการออกแบบซอฟต์แวร์เสร็จจะเข้าสู่ช่วงการพัฒนาซอฟต์แวร์ (Implementation phase) หรือช่วงของการเขียนโปรแกรม ซึ่งจะเขียนโปรแกรมตามที่ได้ออกแบบไว้ในขั้นตอนของการออกแบบซอฟต์แวร์ หลังจากนั้นจะเข้าสู่ช่วงการทดสอบซอฟต์แวร์ (Testing phase) เพื่อทดสอบการทำงานของซอฟต์แวร์ที่ได้ และช่วงของการบำรุงรักษาซอฟต์แวร์ (Maintainence phase) ซึ่งเป็นขั้นตอนสุดท้าย

ขั้นตอนของการทดสอบซอฟต์แวร์ เป็นขั้นตอนหนึ่งที่มีความสำคัญเป็นอย่างยิ่งที่จะทำให้ได้มาซึ่งซอฟต์แวร์ที่มีคุณภาพ โดยสามารถที่จะทำการทดสอบได้ในทุกกระบวนการของการพัฒนาซอฟต์แวร์ โดยหากยิ่งทำการทดสอบระบบในช่วงเริ่มต้นของการพัฒนาระบบ ซึ่งก็คือช่วงของการวิเคราะห์และออกแบบซอฟต์แวร์ จะทำให้ทราบถึงปัญหาและข้อผิดพลาดที่อาจจะเกิดได้เร็วขึ้น สามารถแก้ไขได้ในช่วงต้นๆ ของการพัฒนา ทำให้ซอฟต์แวร์ที่ผลิตออกมามีคุณภาพน่าเชื่อถือ ตรงตามความต้องการของผู้ใช้ [1] นอกจากนี้ต้นทุนโดยรวมในการพัฒนาลดต่ำลงด้วย

กระบวนการทดสอบซอฟต์แวร์จะมีการสร้างกรณีทดสอบ (Test cases) เพื่อใช้ในการทดสอบซอฟต์แวร์ว่าสามารถทำงานได้ถูกต้องหรือไม่ โดยการสร้างกรณีทดสอบระบบแบ่งได้เป็น 2 แบบ [2] คือ การสร้างกรณีทดสอบที่พิจารณาจากโปรแกรม (Program-based testing) และการสร้างกรณีทดสอบที่พิจารณาจากข้อกำหนดความต้องการของซอฟต์แวร์ (Specification-based testing) ซึ่งการสร้างกรณีทดสอบที่พิจารณาจากโปรแกรมจะเหมาะกับการทดสอบระดับ

หน่วย (Unit testing) จะยากในการติดตามผลการทดสอบ นอกจากนี้ยังเป็นการยากสำหรับนักทดสอบระบบ (Tester) เนื่องจากการทดสอบระบบจากโปรแกรมเป็นเทคนิคที่ขึ้นอยู่กับภาษา (Code-dependent) นักทดสอบระบบต้องเข้าใจในวากยสัมพันธ์ (Syntax) ของภาษาที่ใช้ ซึ่งทำให้เสียเวลากับการอ่านโปรแกรมที่เขียนโดยผู้อื่นและเสียเวลาทำความเข้าใจในการทำงานของโปรแกรมด้วย

จากเหตุผลดังกล่าว ผู้วิจัยจึงมีแนวความคิดที่จะทำการทดสอบซอฟต์แวร์ในขั้นตอนของการวิเคราะห์และออกแบบซอฟต์แวร์ ซึ่งจะทำการสร้างกรณีทดสอบจากข้อกำหนดความต้องการของซอฟต์แวร์ โดยพิจารณาจากแผนภาพสเตทชาร์ต (Statechart diagram) เนื่องจากเป็นแผนภาพที่แสดงถึงสถานะการทำงานภายในของคลาส บอกถึงพฤติกรรมของคลาสว่ามีการทำงานอย่างไร ซึ่งในแผนภาพคลาส (Class diagram)ไม่สามารถบอกได้ โดยทั้งสองแผนภาพเป็นประเภทหนึ่งของแผนภาพยูเอ็มแอล (UML: Unified- Modeling Language) [3] ซึ่งเป็นภาษาสัญลักษณ์มาตรฐานที่ถูกคิดขึ้นมาสำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ นอกจากนี้ผู้วิจัยได้พัฒนาเครื่องมือสำหรับสร้างกรณีทดสอบจากแผนภาพสเตทชาร์ต เนื่องจากการสร้างกรณีทดสอบจะเสียเวลาในการสร้างถ้าทำการสร้างด้วยมือ แต่ถ้ามีเครื่องมือที่มาช่วยในการสร้างกรณีทดสอบจะช่วยผู้ทดสอบทำงานได้รวดเร็วและง่ายขึ้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อสร้างกรณีทดสอบจากแผนภาพสเตทชาร์ต ซึ่งเป็นแผนภาพที่แสดงการทำงานของคลาสใดๆ ที่เป็นคลาสที่ขึ้นอยู่กับสถานะ

1.2.2 เพื่อพัฒนาเครื่องมือสนับสนุนการสร้างกรณีทดสอบจากแผนภาพสเตทชาร์ตที่ต้องการ

1.3 ขอบเขตของการวิจัย

1.3.1 พิจารณาการทดสอบโดยใช้แผนภาพคลาสและแผนภาพสเตทชาร์ต ในการพิจารณาสร้างกรณีทดสอบของคลาสโดยดูจากแผนภาพสเตทชาร์ตที่ถูกต้องตามวากยสัมพันธ์ ซึ่งการออกแบบแผนภาพคลาสและแผนภาพสเตทชาร์ตจะต้องถูกต้องตามกฎของยูเอ็มแอล รุ่น 1.3

1.3.2 พิจารณาคลาสที่ขึ้นอยู่กับสถานะ (State-dependent) เป็นคลาสที่มีพฤติกรรมที่หลากหลายในแต่ละสถานะที่เกิดขึ้น ซึ่งเหตุการณ์ที่รับเข้ามา ทำให้ทรานสิชันของ

สถานะเกิดการเปลี่ยนแปลงและสร้างผลลัพธ์ของเหตุการณ์นั้นไปควบคุมการทำงานคลาสอื่น

1.3.3 จะไม่พิจารณาการทำงานที่มีต่อกันระหว่างคลาส เนื่องจากเป็นการทดสอบระดับหน่วยคือ ทดสอบคลาสเดียวโดยไม่ขึ้นต่อคลาสอื่น ซึ่งกรณีที่ในแผนภาพสเตทชาร์ตมีเหตุการณ์จากภายนอกเข้ามาซึ่งเป็นของคลาสที่เกี่ยวข้องกัน จะสร้างการทดสอบโดยสร้างตัวดำเนินการของคลาสนั้นขึ้นมา(Stub) ซึ่งเหมือนกับว่าเป็นคลาสนั้นจริงๆ มาดำเนินการ โดยจะทำการเก็บข้อมูลของคลาสที่เกี่ยวข้องและข้อมูลที่จำเป็นต้องใช้ของคลาสที่เกี่ยวข้องทั้งหมด

1.3.4 แผนภาพคลาสและแผนภาพสเตทชาร์ตที่นำมาพิจารณา จะประกอบไปด้วยสถานะ (State)ในแผนภาพสเตทชาร์ตที่เป็นแบบสถานะธรรมดา (Simple state) และแบบสถานะซับซ้อน (Composite state) ซึ่งสถานะซับซ้อนคือ สถานะที่ประกอบด้วยหลายๆ สถานะรวมอยู่ โดยมีการทำงานเหมือนสถานะธรรมดา ซึ่งงานวิจัยนี้จะไม่พิจารณาสถานะย่อยภายในของสถานะธรรมดานั้น เช่น A มีสถานะย่อย B และ C ผู้วิจัยจะพิจารณา B และ C ด้วยแต่จะไม่พิจารณาสถานะย่อยภายใน B และ C ในกรณีที่ B หรือ C มีสถานะย่อยภายในอีก และไม่พิจารณาสถานะที่มีการทำงานพร้อมกัน (Concurrent state) ด้วย

1.3.5 แผนภาพสเตทชาร์ตที่นำมาพิจารณาในการสร้างกรณีทดสอบจะต้องเป็นแผนภาพที่ประกอบด้วยเหตุการณ์แบบเช้จอีเวนท์ (Change event) ซึ่งเป็นเหตุการณ์ที่ทำให้เกิดการเปลี่ยนสถานะ โดยมีค่าของประพจน์เป็นจริงหรือเท็จเท่านั้น นอกจากนี้ยังพิจารณาเหตุการณ์แบบเงื่อนไขการ์ด (Guard condition) ที่ประกอบด้วยตัวดำเนินการทางตรรกะ 3 แบบ คือ and or และ not เท่านั้น ตัวอย่างเช่น [A and B] หรือ [A and B or not C] โดยให้ A B และ C แทนประโยค (Clause) ที่อยู่ในเงื่อนไขการ์ด ซึ่งทำการพิจารณาเงื่อนไขการ์ดที่ประกอบด้วยประโยคอย่างมาก 3 ประโยคเท่านั้น และพิจารณาทรานสิชัน (Transition) ที่เป็นแบบเอนนาเบิลทรานสิชัน (Enable transition) ซึ่งเป็นทรานสิชันที่มีเหตุการณ์ที่ทำให้เกิดการเปลี่ยนจากสถานะหนึ่งเป็นอีกสถานะหนึ่ง และทรานสิชันกับตัวเอง (Self transition) เป็นทรานสิชันที่เมื่อมีเหตุการณ์ใดๆ เข้ามากระตุ้นแล้วเกิดการกระทำขึ้นภายนอกสถานะ เมื่อเสร็จสิ้นการทำงานก็จะเข้าสู่สถานะเดิม

1.3.6 กรณีทดสอบที่ได้ครอบคลุมทุกประพจน์ที่เกิดขึ้นในแผนภาพสเตทชาร์ตที่ทำการทดสอบ โดยใช้หลักการแบบครอบคลุมประพจน์ (Full predicate coverage criteria)

1.3.7 เครื่องมือที่สร้างรองรับกับระบบที่มีการออกแบบโดยใช้แผนภาพสเตทชาร์ตที่ไม่เกิน 10 สถานะในแต่ละคลาสที่พิจารณา

1.3.8 พัฒนาเครื่องมือบนระบบปฏิบัติการวินโดว์ โดยใช้ไมโครซอฟต์วิชวลสตูดิโอดอทเน็ต รุ่น 2003 (Microsoft Visual Studio .NET 2003) เป็นเครื่องมือในการพัฒนา

1.3.9 ทำการประเมินกรณีทดสอบที่ได้ด้วยมือ (Manual) ว่ากรณีทดสอบที่ได้ตรงตามหลักการการทดสอบ คือครอบคลุมทุกประพจน์ในโมเดลการทดสอบหรือไม่

1.4 ขั้นตอนในการวิจัย

1.4.1 ศึกษาแผนภาพคลาสและแผนภาพสเตทชาร์ต

1.4.2 ศึกษาวิธีการในการทดสอบซอฟต์แวร์เชิงวัตถุ

1.4.3 ศึกษาวิธีการในการแปลงแผนภาพสเตทชาร์ตให้อยู่ในรูปแบบข้อมูลที่สามารถสร้างกรณีทดสอบได้

1.4.4 ออกแบบและพัฒนาขั้นตอนวิธีในการสร้างกรณีทดสอบโดยพิจารณาจากคลาสใดๆ ที่ขึ้นอยู่กับสถานะในแผนภาพคลาส โดยทำการสร้างกรณีทดสอบจากแผนภาพสเตท-ชาร์ตของคลาสที่พิจารณา

1.4.5 ออกแบบและพัฒนาเครื่องมือซอฟต์แวร์เพื่อสนับสนุนการสร้างกรณีทดสอบ

1.4.6 ประเมินผลการทดสอบ

1.4.7 สรุปผลการวิจัยและจัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่ได้รับ

1.5.1 กรณีทดสอบที่ได้ครอบคลุมประพจน์ที่เกิดขึ้นภายในแผนภาพสเตทชาร์ต นอกจากนี้ยังครอบคลุมทุกทรานสิชันภายในแผนภาพอีกด้วย

1.5.2 กรณีทดสอบที่ได้จะช่วยให้ผู้พัฒนาซอฟต์แวร์ สามารถนำมาทดสอบสถานะการทำงานของคลาสว่าเมื่อมีเหตุการณ์เข้ามากระตุ้นให้เกิดการทำงาน แล้วจะเปลี่ยนไป

เป็นสถานะอะไร ตามเงื่อนไขของเหตุการณ์ที่เกิดขึ้น ซึ่งเป็นการตรวจสอบการทำงานของโปรแกรมว่าทำงานถูกต้องหรือไม่

1.5.3 ประหยัดเวลาและเพิ่มคุณภาพของซอฟต์แวร์ เนื่องจากสามารถพบข้อบกพร่องและแก้ไขข้อผิดพลาดของซอฟต์แวร์ได้ในช่วงเริ่มต้นของการพัฒนาซอฟต์แวร์

1.5.4 เครื่องมือที่สร้างได้ ช่วยลดภาระให้กับผู้ทดสอบระบบ ทำให้ผู้ทดสอบทำงานง่ายขึ้น เนื่องจากผู้ทดสอบไม่ต้องพิจารณาการสร้างกรณีทดสอบด้วยตนเอง