บทความวิชาการ

การทดสอบซอฟต์แวร์

Software Testing

อรยา ปรีชาพานิช
อาจารย์ประจำ ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ มหาวิทยาลัยทักษิณ

1. บทนำ

การพัฒนาซอฟต์แวร์เพื่อใช้งานในองค์กร มีเป้าหมายเพื่อให้ได้ซอฟต์แวร์ที่มีประสิทธิภาพ และสามารถตอบสนองความต้องการใช้งานของผู้ใช้ได้อย่างแท้จริง แต่ด้วยปัจจัยแวดล้อมหลายประการ เช่น ระยะเวลาการพัฒนาซอฟต์แวร์ที่จำกัด งบประมาณที่ได้รับไม่เพียงพอ การสื่อสารระหว่างนักพัฒนาระบบและผู้ใช้งานไม่สอดคล้องกัน หรือการบูรณาการความต้องการใช้งานของผู้ใช้งานทุกกลุ่มที่เกี่ยวข้องไม่ครบถ้วนสมบูรณ์ เป็นต้น ล้วนมีผลต่อคุณภาพของซอฟต์แวร์ที่พัฒนาขึ้นในภาพรวม เนื่องจากผู้พัฒนามุ่งเน้นที่จะพัฒนาซอฟต์แวร์ให้แล้วเสร็จตามกำหนดการ ภายใต้เงื่อนไขที่ไม่เอื้ออำนวย จึงมักตัดสินใจลดขั้นตอนสำคัญในวงจรของการพัฒนาซอฟต์แวร์บางส่วนออกไป โดยเฉพาะอย่างยิ่งขั้นตอนของการทดสอบซอฟต์แวร์ ทั้งนี้เพราะผู้พัฒนามักมีใจเอนเอียง (Bias) ว่าซอฟต์แวร์ที่ทีมงานตนเองพัฒนาขึ้นมีคุณภาพดีอยู่แล้ว จึงไม่ได้ให้ความสำคัญกับการทดสอบซอฟต์แวร์ ประกอบกับผู้ใช้งานไม่มีความรู้และความเชี่ยวชาญในเรื่องของวงจรของการพัฒนาซอฟต์แวร์ที่ดีเพียงพอ ก็จะยินยอมให้มีการนำซอฟต์แวร์ดังกล่าวมาติดตั้งใช้งานภายในองค์กร และพบกับข้อผิดพลาดต่างๆ ที่เกิดขึ้นระหว่างการใช้งาน ซึ่งส่งผลให้ประสิทธิภาพการทำงานและประสิทธิผลขององค์กรลดลง และอาจก่อให้เกิดปัญหารวมทั้งค่าใช้จ่ายอื่นๆ ตามมาได้อีกมากมาย ในกรณีที่ผู้ที่เกี่ยวข้องเลือกที่จะแก้ปัญหาอย่างผิดวิธีการ ดังนั้นจึงกล่าวได้ว่า "การทดสอบซอฟต์แวร์" เป็นกระบวนการสำคัญที่สุดกระบวนการหนึ่งที่องค์กรจะต้องให้ความสำคัญ เพื่อให้เกิดความเชื่อมั่นในการใช้งานซอฟต์แวร์ และเพื่อให้การลงทุนทั้งหมดเป็นไปอย่างคุ้มค่ามากที่สุด

2. หลักการทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์ หมายถึง กระบวนการในการวิเคราะห์ ตรวจสอบ และติดตามผลการพัฒนาซอฟต์แวร์ เพื่อให้แน่ใจว่าซอฟต์แวร์ที่จะส่งมอบงานมีความถูกต้อง สมบูรณ์ และมี

ประสิทธิภาพตามที่ผู้ใช้งานได้คาดหวังไว้ โดยทั่วไปผู้ใช้งานจะร่วมกันกำหนดเกณฑ์มาตรฐานที่จะยอมรับซอฟต์แวร์ (Acceptance Criteria) ตั้งแต่ระยะแรกของโครงการ หลังจากที่สรุปข้อกำหนดคุณลักษณะความต้องการใช้งานซอฟต์แวร์ (Software Requirement Specification) เสร็จสิ้นแล้ว แต่ยังไม่ได้เริ่มขั้นตอนของการออกแบบระบบ ซึ่งถือว่าเป็นข้อตกลงร่วมกันระหว่างนักพัฒนาระบบกับผู้ใช้งานว่าซอฟต์แวร์ที่จะพัฒนาขึ้นนั้นต้องมีคุณสมบัติใดบ้าง และจะนำเกณฑ์มาตรฐานดังกล่าวมาใช้ในการตรวจรับงานซอฟต์แวร์

การทดสอบซอฟต์แวร์แบ่งได้เป็น 2 กระบวนการหลักดังนี้คือ

- กระบวนการตรวจสอบความถูกต้องของซอฟต์แวร์ที่ถูกพัฒนาขึ้นว่าเป็นไปตามข้อกำหนดคุณลักษณะความต้องการใช้งานซอฟต์แวร์ที่ได้กำหนดไว้หรือไม่ (Verification)
- กระบวนการตรวจสอบผลการพัฒนาซอฟต์แวร์ที่เกิดขึ้นจริงว่าตรงกับความคาดหวังของผู้ใช้หรือไม่ (Validation)

นั่นคือการทดสอบซอฟต์แวร์จะต้องตอบคำถามประเด็นสำคัญๆ ดังต่อไปนี้ได้

- ซอฟต์แวร์นี้สามารถตอบสนองความต้องการใช้งานของผู้ใช้ได้อย่างถูกต้องและครบถ้วนสมบูรณ์หรือไม่
- ซอฟต์แวร์นี้ทำงานได้ตามที่คาดหวังไว้หรือไม่
- ซอฟต์แวร์นี้สามารถทำงานร่วมกับซอฟต์แวร์อื่นๆภายในองค์กรอย่างมีประสิทธิภาพหรือไม่
- ซอฟต์แวร์นี้รองรับการขยายงานในอนาคตแล้วหรือไม่
- ซอฟต์แวร์นี้พร้อมที่จะติดตั้งใช้งานแล้วหรือไม่
- เป็นต้น

จากตัวอย่างคำถามข้างต้นจะเห็นได้ว่า การทดสอบซอฟต์แวร์มุ่งเน้นที่คุณภาพของซอฟต์แวร์ในประเด็นต่างๆ ที่ครอบคลุมมากกว่าการแก้ไขข้อบกพร่องของการเขียนโปรแกรมเพียงอย่างเดียว ดังนั้นในกรณีที่ผลการทดสอบออกมาว่าฟังก์ชันการทำงานของซอฟต์แวร์มีข้อบกพร่อง ก็ไม่ได้หมายความว่า โปรแกรมเมอร์เขียนโปรแกรมไม่ดี แต่อาจจะเกิดจากสาเหตุอื่นๆ เช่น เอกสาร

ข้อกำหนดคุณลักษณะความต้องการใช้งานของผู้ใช้ระบบไม่ชัดเจน ทำให้เกิดการตีความที่คลาดเคลื่อนกันระหว่างผู้ใช้และผู้พัฒนาระบบ หรืออาจเกิดจากความผิดพลาดในการสื่อสารระหว่างทีมงานพัฒนาระบบ เป็นต้น การทดสอบซอฟต์แวร์จึงเป็นกระบวนการสำคัญที่นำไปสู่การวิเคราะห์เพื่อหาแนวทางในการแก้ปัญหาที่ต้นเหตุ โดยพิจารณาถึงผลกระทบที่เกิดขึ้นในภาพรวมทั้งหมดและลำดับความสำคัญเร่งด่วน เพื่อให้ได้ซอฟต์แวร์ที่ถูกต้อง สมบูรณ์มากที่สุด

หลักการทดสอบซอฟต์แวร์ที่สำคัญคือ จะต้องเลือกทดสอบซอฟต์แวร์ในส่วนที่เป็นฟังก์ชันการทำงานหลัก (Core Function) นั่นคือถ้าเกิดข้อผิดพลาดขึ้นในฟังก์ชันดังกล่าวแล้วจะทำให้เกิดภาวะวิกฤตในการปฏิบัติงาน จากนั้นจึงจะไปทำการทดสอบฟังก์ชันเสริมต่างๆ ที่พัฒนาเพิ่มเติมเพื่อให้ซอฟต์แวร์มีความสมบูรณ์มากยิ่งขึ้นในลำดับถัดไป

นอกจากนั้นแล้วการทดสอบซอฟต์แวร์ยังจำเป็นที่จะต้องทดสอบทั้งในสภาวะแวดล้อมปกติ และในสภาวะแวดล้อมที่ไม่ปกติซึ่งจำลองขึ้นเพื่อทดสอบประสิทธิภาพในเรื่องความคงทนและความเสถียรของซอฟต์แวร์ เช่น การทดสอบฟังก์ชันการค้นคืนข้อมูลจากเครื่องคอมพิวเตอร์แม่ข่าย จะต้องทำการเรียกใช้งานฟังก์ชันดังกล่าวทั้งในช่วงเวลาปกติทั่วไป และช่วงเวลาที่มีการเข้าถึงข้อมูลหนาแน่นสูงสุดของวัน เพื่อพิจารณาเวลาที่ใช้ในการประมวลผล และผลลัพธ์ที่ได้ว่าถูกต้องสมบูรณ์หรือไม่ เป็นต้น

3. บทบาทหน้าที่ของผู้ทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์ควรจะจัดตั้งเป็นทีมงานขึ้นมาหนึ่งชุด โดยมีจำนวนบุคลากรมากหรือน้อยขึ้นอยู่กับขนาดและความซับซ้อนของซอฟต์แวร์นั้นๆ คุณสมบัติของบุคลากรในทีมงานทดสอบซอฟต์แวร์ประกอบด้วย

- มีความรู้ความเข้าใจในซอฟต์แวร์ที่กำลังพัฒนาเป็นอย่างดี
- มีความละเอียด รอบคอบ
- มีความกระตือรือร้น
- มีมุมมองที่หลากหลาย
- มีมนุษยสัมพันธ์ดี

คุณสมบัติที่สำคัญคือ ทีมงานทดสอบซอฟต์แวร์จะต้องเป็นผู้ที่มีจิตวิทยาในการสื่อสารกับผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ทุกระดับ เพื่อหลีกเลี่ยงการเกิดความรู้สึกต่อต้านและไม่ให้

ความร่วมมือในการทดสอบซอฟต์แวร์ เนื่องจากเข้าใจว่าเป็นการมุ่งเน้นที่จะหาข้อผิดพลาดในส่วนงานที่ตนเองเป็นผู้รับผิดชอบ ซึ่งในตารางที่ 1 ได้แสดงถึงบุคลากรในตำแหน่งต่างๆ พร้อมทั้งหน้าที่ความรับผิดชอบที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์

ตารางที่ 1 แสดงตำแหน่งของบุคลากรพร้อมทั้งหน้าที่ความรับผิดชอบที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์

ตำแหน่ง	หน้าที่ความรับผิดชอบ
เจ้าของและผู้บริหารองค์กร	-สนับสนุนด้านทรัพยากร เช่น บุคลากร งบประมาณ อุปกรณ์คอมพิวเตอร์ และเครื่องมือต่างๆ เป็นต้น -กำหนดความต้องการใช้งานที่นอกเหนือจากงานประจำวัน -กำหนดระยะเวลาการส่งมอบงาน
หัวหน้าโครงการ	-วางแผนการดำเนินงานโครงการ -ควบคุมการดำเนินงานให้เป็นไปตามแผนที่กำหนดไว้
นักพัฒนาซอฟต์แวร์	-ออกแบบ พัฒนาโปรแกรม รวมทั้งจัดทำเป็นโปรแกรมประยุกต์ที่พร้อมจะติดตั้งใช้งาน -ร่วมในการทบทวน/ทดสอบโปรแกรม -แก้ไขข้อบกพร่องและผลกระทบที่เกิดขึ้นจากการเขียนโปรแกรม
ผู้ประสานงานในการทดสอบซอฟต์แวร์	-จัดทำแผนการทดสอบซอฟต์แวร์ -จัดทำเอกสารข้อกำหนดคุณลักษณะการทดสอบซอฟต์แวร์ โดยอ้างอิงจากเอกสารข้อกำหนดคุณลักษณะความต้องการใช้งานซอฟต์แวร์
ผู้ที่ทำการทดสอบซอฟต์แวร์	-ทดสอบซอฟต์แวร์ -จัดทำเอกสารสรุปผลการทดสอบซอฟต์แวร์

4. การจัดทำแผนการทดสอบซอฟต์แวร์

จุดเริ่มต้นของการทดสอบซอฟต์แวร์ คือการจัดทำแผนการทดสอบซอฟต์แวร์ (Test Plan) เนื่องจากการทดสอบซอฟต์แวร์จะต้องใช้เวลาการปฏิบัติงานของบุคลากร และใช้งบประมาณในการดำเนินงานค่อนข้างมาก ดังนั้นจึงจำเป็นอย่างยิ่งที่จะต้องกำหนดรายละเอียดการดำเนินงานทั้งหมด เช่น องค์ประกอบของซอฟต์แวร์ที่จะทำการทดสอบ ตารางเวลาของการทดสอบ บุคลากรที่เกี่ยวข้องรวมทั้งกรอบแนวทางการปรับปรุงแก้ไขข้อผิดพลาดที่อาจเกิดขึ้นในโครงการเป็นต้น ซึ่งรายละเอียดทั้งหมดจะต้องสอดคล้องกับเอกสารข้อกำหนดคุณลักษณะความต้องการใช้งานซอฟต์-แวร์ และเอกสารการวิเคราะห์และออกแบบซอฟต์แวร์ โดยผ่านความเห็นชอบร่วมกันจากผู้ที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์ทุกฝ่าย ทั้งนี้กิจกรรมการทดสอบซอฟต์แวร์จะต้องถูกกำหนดไว้ในแผนการดำเนินงานโครงการ (Software Development Plan) ตั้งแต่เริ่มต้นโครงการไปจนกระทั่งจบโครงการ

จากมาตรฐานของ "American National Standards Institute and Institute for Electronic Engineer Standard 829/1983 for Software Test Document" ได้ระบุถึงองค์ประกอบต่างๆ ที่ควรกำหนดไว้อย่างชัดเจนในแผนการทดสอบซอฟต์แวร์ รายละเอียดดังปรากฏในตารางที่ 2 จากตารางดังกล่าวจะเห็นว่ามีการกำหนดองค์ประกอบต่างๆที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์ที่ครอบคลุมทั้งทางด้านเทคนิคและด้านการจัดการ ซึ่งบางองค์ประกอบเช่น กรณีทดสอบ (Test Case) จะมีรายละเอียดมากมายเพื่อให้การทดสอบซอฟต์แวร์ครอบคลุมความต้องการใช้งานมากที่สุด ดังนั้นนักพัฒนาซอฟต์แวร์ ผู้ใช้งาน และผู้ที่มีส่วนเกี่ยวข้องทุกฝ่าย จะต้องร่วมกันกำหนดชุดข้อมูลนำเข้า (Input) เงื่อนไขต่างๆ (Condition) ที่ใช้ในการประมวลผล รวมทั้งผลลัพธ์ที่ควรได้จากการประมวลผล (Expected Result) ทั้งในกรณีการใช้งานซอฟต์แวร์ในสถานการณ์ปกติ และในสถานการณ์ที่จะทำให้เกิดข้อผิดพลาดขึ้น เพื่อที่จะดำเนินการทดสอบให้แน่ใจว่าซอฟต์แวร์ที่พัฒนาขึ้นสามารถทำงานได้อย่างถูกต้องและรองรับสถานการณ์ต่างๆ ที่จะเกิดขึ้นในอนาคตได้

ตารางที่ 2 แสดงองค์ประกอบที่ควรกำหนดไว้ในแผนการทดสอบซอฟต์แวร์

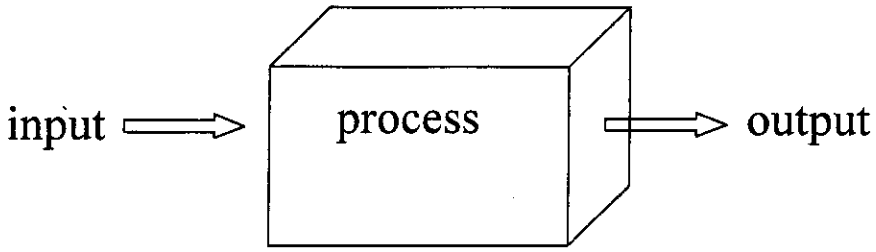
องค์ประกอบ	คำอธิบาย
บุคลากรและหน้าที่ความรับผิดชอบ	กำหนดบุคลากรที่ได้รับมอบหมายให้ทำหน้าที่ทดสอบซอฟต์แวร์ในส่วนต่างๆ พร้อมทั้งหน้าที่ความรับผิดชอบอย่างชัดเจน
สมมติฐาน	กำหนดสถานะของโปรแกรมและซอฟต์แวร์ในแต่ละช่วงระยะเวลาการดำเนินงานโครงการ
การทดสอบ	-กำหนดขอบเขตของการทดสอบ -กำหนดตารางเวลาของการทดสอบ -กำหนดลำดับความสำคัญก่อน-หลังในการทดสอบ -กำหนดกรณีทดสอบ(Test Case) ที่เกี่ยวข้องกับการใช้งานซอฟต์แวร์
การประสานงาน	กำหนดแนวทางการปฏิบัติงานร่วมกัน และช่องทางการสื่อสารระหว่างบุคลากรที่เกี่ยวข้อง
การวิเคราะห์ความเสี่ยง	กำหนดปัจจัยเสี่ยงต่อความสำเร็จของโครงการ เพื่อการติดตามผลและวิเคราะห์ความเสี่ยงที่เกิดขึ้นเป็นระยะๆ
รายงานผลการทดสอบซอฟต์แวร์	กำหนดรูปแบบการจัดทำเอกสารรายงานผลการทดสอบซอฟต์แวร์เพื่อนำไปใช้ในการปรับปรุงแก้ไขข้อบกพร่องที่เกิดขึ้น และใช้ในการทดสอบซ้ำ หลังจากมีการปรับปรุงแก้ไขซอฟต์แวร์แล้ว
สภาพแวดล้อม	กำหนดสภาพแวดล้อมที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์ เช่น -สภาพแวดล้อมของข้อมูลที่เกี่ยวข้อง -สภาพแวดล้อมในการปฏิบัติงาน -สภาพแวดล้อมในการทดสอบซอฟต์แวร์ เป็นต้น

5. วิธีการทดสอบซอฟต์แวร์

การทดสอบซอฟต์แวร์แบ่งได้เป็น 2 วิธีการหลักดังนี้คือ

5.1 Functional Testing หรือที่นักพัฒนาซอฟต์แวร์เรียกว่า **Black Box Testing** เนื่องจากการทดสอบนี้จะเน้นผลลัพธ์ที่เกิดขึ้น (Output) จากการประมวลผลโปรแกรม (Process) โดยไม่สนใจรูปแบบการเขียนโปรแกรมของโปรแกรมเมอร์

การทดสอบซอฟต์แวร์ด้วยวิธีการนี้จึงเหมือนกับการป้อนข้อมูลเข้าสู่**กล่องดำ**ที่ไม่สามารถมองเห็นกระบวนการทำงานภายในกล่อง และรอรับผลลัพธ์ที่เกิดขึ้น จากนั้นจึงนำผลลัพธ์นั้นไปตรวจสอบกับกรณีทดสอบ (Test Case) ที่ได้กำหนดไว้ล่วงหน้าแล้วว่าถูกต้องตรงกันหรือไม่ ดังนั้นจึงเห็นได้ว่าการทดสอบด้วยวิธีการนี้ผู้ทดสอบไม่จำเป็นจะต้องมีความรู้หรือทักษะทางด้านเทคนิคในการพัฒนาซอฟต์แวร์แต่อย่างใด

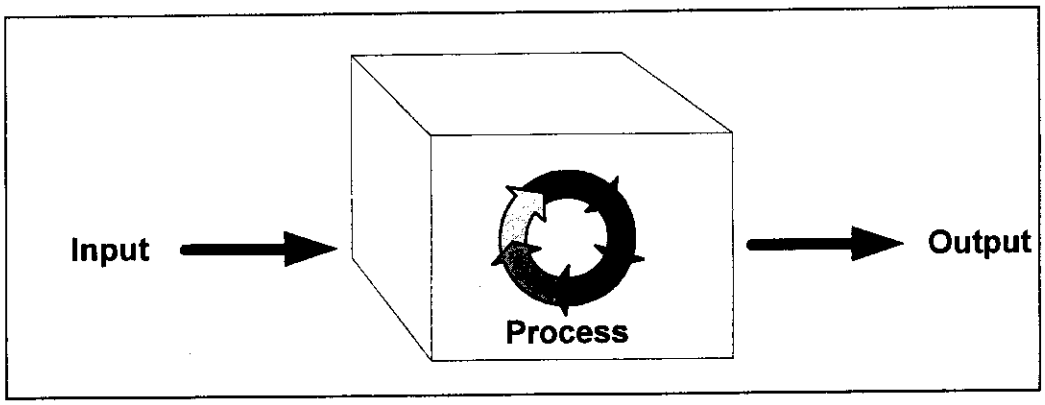


รูปที่ 1 แสดงภาพจำลองของวิธีการทดสอบแบบ Functional Testing

5.2 Structural Testing หรือที่นักพัฒนาซอฟต์แวร์เรียกว่า **White Box Testing / Glass Box Testing** เนื่องจากการทดสอบนี้จะเน้นที่โครงสร้างโปรแกรมและการเขียนโปรแกรมในเชิงตรรกะ รวมไปถึงการตรวจสอบเพื่อหาผลกระทบที่อาจเกิดขึ้นจากกระบวนการทำงานในแต่ละโมดูลของโปรแกรม ดังนั้นการกำหนดข้อมูลเพื่อใช้ในการทดสอบจะทำได้ก็ต่อเมื่อโปรแกรมได้ถูกพัฒนาขึ้นมาแล้ว

การทดสอบซอฟต์แวร์ด้วยวิธีการนี้จึงเหมือนกับการป้อนข้อมูลเข้าสู่**กล่องใส**ที่ผู้ทดสอบสามารถมองเห็นกระบวนการทำงานภายในกล่องได้อย่างชัดเจนว่ามีโครงสร้าง

การทำงานอย่างไร และกระบวนการทำงานนั้นมีประสิทธิภาพดีเพียงพอหรือไม่ ดังนั้นจึงเห็นได้ว่าการทดสอบด้วยวิธีการนี้ผู้ทดสอบจำเป็นจะต้องมีความรู้และทักษะทางด้านเทคนิคในการพัฒนาซอฟต์แวร์เป็นอย่างดี

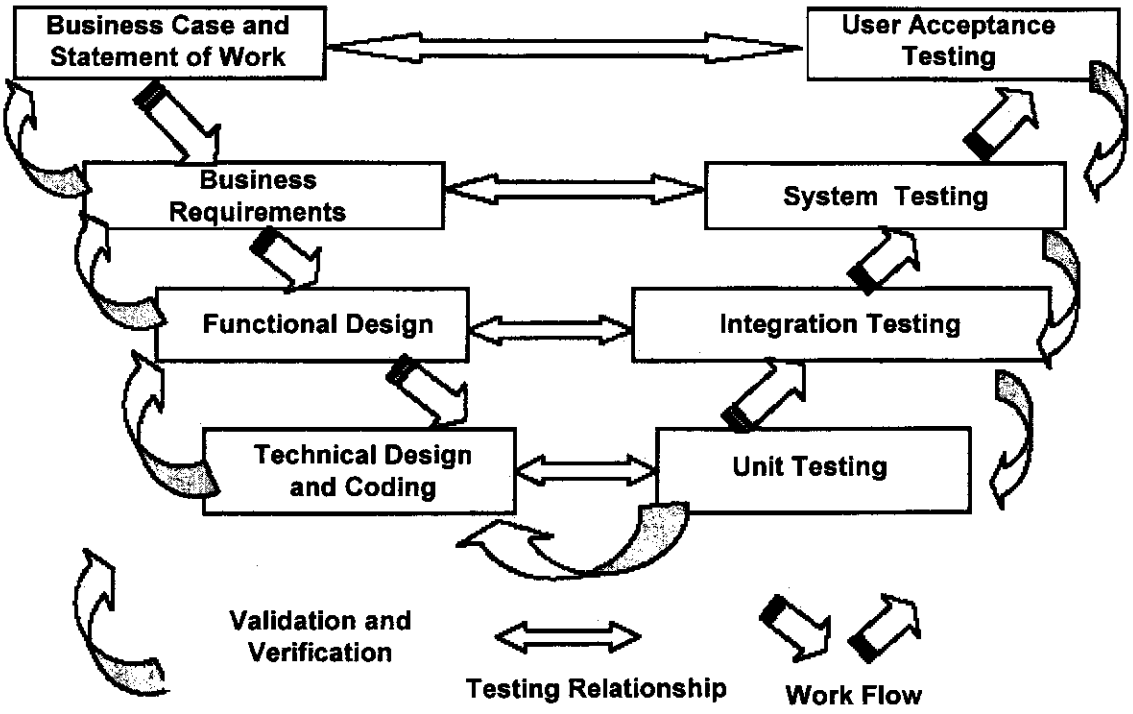


รูปที่ 2 แสดงภาพจำลองของวิธีการทดสอบแบบ Structural Testing

จากที่กล่าวมาแล้วข้างต้นจะเห็นได้ว่าการทดสอบซอฟต์แวร์แต่ละวิธีการมีเป้าหมายในการทดสอบที่แตกต่างกัน ดังนั้นเพื่อให้การทดสอบซอฟต์แวร์ได้ผลสัมฤทธิ์ที่น่าเชื่อถือมากที่สุด จึงควรดำเนินการทดสอบซอฟต์แวร์ทั้ง 2 วิธีการ ก่อนการส่งมอบและติดตั้งใช้งานจริง

6. ตัวแบบในการพัฒนาซอฟต์แวร์กับการทดสอบซอฟต์แวร์

ตัวแบบในการพัฒนาซอฟต์แวร์ในปัจจุบันมีหลากหลายแนวคิด แต่ตัวแบบที่มุ่งเน้นการทดสอบซอฟต์แวร์ในแต่ละขั้นตอนของการดำเนินงานโครงการ คือตัวแบบ V-Model โดยมีรูปแบบการทำงานดังรูปที่ 3 ซึ่งได้แสดงผลลัพธ์ที่เกิดขึ้นจากกระบวนการพัฒนาซอฟต์แวร์เป็นลำดับจากบนลงล่าง และแสดงกระบวนการทดสอบซอฟต์แวร์เป็นลำดับจากล่างขึ้นบน ตามลักษณะของตัวอักษร V และมีเส้นตรงที่มีลูกศร 2 ทิศทาง แสดงถึงความสัมพันธ์ระหว่างผลลัพธ์ที่เกิดขึ้นจากกระบวนการพัฒนาซอฟต์แวร์กับกระบวนการทดสอบซอฟต์แวร์ นอกจากนั้นแล้วทุกกระบวนการทำงานสามารถย้อนกลับไปทบทวน/ตรวจสอบกระบวนการที่ผ่านมาได้(Validation and Verification)



รูปที่ 3 แสดงตัวแบบในการพัฒนาซอฟต์แวร์ของ V-Model

กระบวนการทดสอบซอฟต์แวร์ประกอบด้วย 4 กระบวนหลักดังนี้คือ

6.1 Unit Testing เป็นการทดสอบแต่ละโมดูลย่อยของระบบ ว่าทำงานได้อย่างถูกต้องหรือไม่ โดยทั่วไปจะถือว่าแต่ละโมดูลมีความเป็นอิสระต่อกัน ดังนั้นในขั้นตอนนี้เราจะไม่คำนึงถึงการทำงานที่สัมพันธ์กับส่วนอื่นๆ ของระบบ

การทดสอบในขั้นตอนนี้จะอ้างอิงเอกสารการวิเคราะห์และออกแบบซอฟต์แวร์ ในส่วนที่เป็นการออกแบบทางเทคนิคของซอฟต์แวร์ (Technical Design Specification) เป็นหลัก ซึ่งในเอกสารดังกล่าวจะกำหนดไว้ว่าแต่ละโมดูลทำหน้าที่อะไร และมีกระบวนการทำงานอย่างไร

6.2 Integration Testing เป็นการทดสอบกระบวนการทำงานในแต่ละระบบย่อย ซึ่งประกอบด้วยโมดูลต่างๆที่ทำงานสัมพันธ์กัน ในขั้นตอนนี้จะเน้นไปที่การประสานเชื่อมโยงกันระหว่างโมดูล เพื่อให้แน่ใจว่า แต่ละระบบย่อยทำงานได้อย่างมีประสิทธิภาพ

การทดสอบในขั้นตอนนี้จะอ้างอิงเอกสารการวิเคราะห์และออกแบบซอฟต์แวร์ในส่วนที่เป็นการออกแบบฟังก์ชันการทำงานของซอฟต์แวร์ (Functional Design Specification) เป็นหลัก ซึ่งในเอกสารดังกล่าวจะกำหนดไว้ว่าแต่ละระบบย่อยประกอบด้วยโมดูลใดบ้าง และมีกระบวนการทำงานอย่างไร

6.3 System Testing เป็นการทดสอบการทำงานร่วมกันของทั้งระบบ โดยเน้นไปที่การประสานเชื่อมโยงกันระหว่างระบบย่อย รวมทั้งการตรวจสอบในภาพรวมของระบบว่าระบบได้ตอบสนองความต้องการใช้งานทั้งในส่วนของฟังก์ชันการทำงานและในส่วนของประสิทธิภาพของซอฟต์แวร์แล้วหรือไม่ โดยอ้างอิงเอกสารข้อกำหนดคุณลักษณะความต้องการใช้งานซอฟต์แวร์เป็นหลัก

ในกรณีที่ทดสอบแล้วพบข้อผิดพลาดและดำเนินการปรับปรุงแก้ไขเรียบร้อยแล้ว จะต้องทำการทดสอบทั้งระบบซ้ำอีกครั้ง เพื่อให้แน่ใจว่าการปรับแก้โปรแกรมในครั้งนี้ไม่ได้สร้างผลกระทบหรือปัญหาใหม่ๆ ให้เกิดขึ้นกับระบบ

6.4 User Acceptance Testing เป็นการทดสอบร่วมกันระหว่างผู้ใช้งานกับนักพัฒนาระบบ โดยใช้ข้อมูลจริงป้อนเข้าสู่ระบบ เพื่อจำลองสถานการณ์การใช้งานระบบในอนาคต ซึ่งอาจทำให้ค้นพบข้อผิดพลาดบางอย่างที่ไม่คาดคิดมาก่อน เนื่องจากการทดสอบในขั้นตอนที่ผ่านมาทั้งหมดล้วนแล้วแต่ใช้ข้อมูลจำลองทั้งสิ้น ดังนั้นจึงอาจจะเป็นไปได้ว่ายังมีบางประเด็นที่ซอฟต์แวร์ไม่ได้พัฒนาเพื่อรองรับสถานการณ์ดังกล่าว

กระบวนการทดสอบซอฟต์แวร์นี้จะดำเนินการไปอย่างต่อเนื่อง จนกว่าผู้ใช้งานกับนักพัฒนาระบบจะตกลงร่วมกันได้ว่า สามารถส่งมอบงานซอฟต์แวร์นี้ได้แล้ว และเตรียมพร้อมในการติดตั้งเพื่อใช้งานจริงในองค์กรต่อไป

7. สรุป

การทดสอบซอฟต์แวร์ เป็นกระบวนการหลักของวงจรการพัฒนาซอฟต์แวร์ ซึ่งจะช่วยในการประหยัดทรัพยากรที่ต้องใช้ในการดำเนินงานโครงการทั้งงบประมาณและเวลา ในกรณีที่สามารถพบข้อผิดพลาดหรือปัญหาที่เกิดขึ้นตั้งแต่ระยะแรกๆ ของโครงการ นั่นคือยิ่งพบปัญหาเร็วก็จะยิ่งหาทางแก้ไขปัญหาได้เร็วขึ้น รวมทั้งยังส่งผลกระทบกับส่วนอื่นๆ น้อยที่สุดเท่าที่จะเป็นไปได้ และที่สำคัญคือ การทดสอบซอฟต์แวร์สามารถสร้างความเชื่อมั่น และการยอมรับในการใช้งานซอฟต์แวร์เพื่อเป็นส่วนหนึ่งของการปฏิบัติงานในองค์กรได้อย่างมีประสิทธิภาพและประสิทธิผลต่อไป

เอกสารอ้างอิง

(1) Electric Power Research Institute. 2006. *Test Plan Development.* http://www.epri.com/eprisoftware/processguide/tpdev.html

(2) Ian Sommerville. 1998. *Software Engineering* (Fifth Edition). Addison-Wesley.

(3) International Institute for Software Testing. http://www.testinginstitute.com/

(4) John E. Bentley. 2004. *Software Testing Fundamentals – Concepts,* Roles, and Terminology. www2.sas.com/proceedings/sugi30/141-30.pdf

(5) Jon Fairlough. 1996. *Software Engineering Guides.* Printice Hall Europe.

(6) Wikipedia. 2006. *Software Testing.* http://en.wikipedia.org/wiki/Software_testing