การทดสอบซอฟต์แวร์

อุไร ทองหัวไผ่[1]

บทคัดย่อ

การทดสอบซอฟต์แวร์มีวัตถุประสงค์เพื่อค้นหาข้อผิดพลาดที่มีอยู่ในระบบ ผู้ที่ทำการทดสอบเรียกว่าทีมทดสอบประกอบไปด้วย นักทดสอบมืออาชีพ นักวิเคราะห์ระบบ นักออกแบบระบบ ผู้เชี่ยวชาญการจัดการโครงแบบ และ ผู้ใช้ การทดสอบความผิดพลาดของซอฟต์แวร์แบ่งเป็น 2 ระดับคือระดับโปรแกรม และระดับระบบ สำหรับระบบขนาดใหญ่การทดสอบแบ่งเป็น 6 ระยะได้แก่ การทดสอบมอดูลหรือทดสอบหน่วย การทดสอบรวม การทดสอบฟังก์ชัน การทดสอบประสิทธิภาพ การทดสอบการยอมรับ และการทดสอบการติดตั้ง การทดสอบซอฟต์แวร์สามารถทำให้ลูกค้ามั่นใจได้ว่าระบบสามารถกระทำหน้าที่และสามารถแก้ปัญหาได้ตามข้อกำหนดที่ระบุในเอกสารกำหนดความต้องการได้อย่างถูกต้อง เครื่องมือที่ใช้ในการทดสอบได้แก่ เครื่องมือจำลอง ระบบเฝ้าสังเกต และเครื่องมือการวิเคราะห์ เอกสารที่จำเป็นในการทดสอบได้แก่ แผนทดสอบ ข้อกำหนดการทดสอบและประเมินผล รายละเอียดการทดสอบและรายงานการวิเคราะห์การทดสอบ

คำสำคัญ: การทดสอบซอฟต์แวร์ แผนทดสอบ ทีมทดสอบ

[1]ผู้ช่วยศาสตราจารย์ ภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยรามคำแหง

วารสารเกษมบัณฑิต ปีที่ 16 ฉบับที่ 2 กรกฎาคม – ธันวาคม 2558

Software Testing

Urai Thonghuaphai[1]

Abstract

Software testing aims to find errors that exist in the system. Group of persons who perform the testing are the test team. Test team members consist of professional testers, system analysts, system designers, configuration management specialists, and users. Testing of software errors is divided into two levels i.e, the program level, and system level. In the development of a large system, testing involves 6 stages, namely, module testing or unit testing, integration testing, function testing, performance testing, acceptance testing and installation testing. System testing can ensure that the system will be functioning as required and can help solve the problem that occurs. Several tools are available for example, simulators, monitor, and analyzer. Several types of documentation are needed in the software testing, e.g., test plan, test specification and evaluation, test description, and test analysis report.

Key words: System testing, Test plan, Test team, Module testing, Unit testing.

[1]Asst. Prof Computer Science, Ramkhamhang University.

บทนำ

การทดสอบซอฟต์แวร์(software testing) เป็นขั้นตอนหลักของการพัฒนาซอฟต์แวร์ เป็นกระบวนการค้นหาข้อผิดพลาดที่มีอยู่ในระบบ ช่วยให้ซอฟต์แวร์ที่พัฒนาขึ้นมีความถูกต้อง สมบูรณ์ ปลอดภัย และมีประสิทธิภาพและคุณภาพที่ดี ผู้ที่ทำการทดสอบเรียกว่าทีมทดสอบ (test team) โดยทั่วไปประกอบไปด้วย นักทดสอบมืออาชีพ นักวิเคราะห์ นักออกแบบระบบ ผู้เชี่ยวชาญการจัดการโครงแบบ และ ผู้ใช้ การทดสอบซอฟต์แวร์ประกอบด้วยหลายขั้นตอน และแบ่งออกเป็นหลายระยะ ดังรายละเอียดต่อไปนี้

ขั้นตอนการทดสอบซอฟต์แวร์

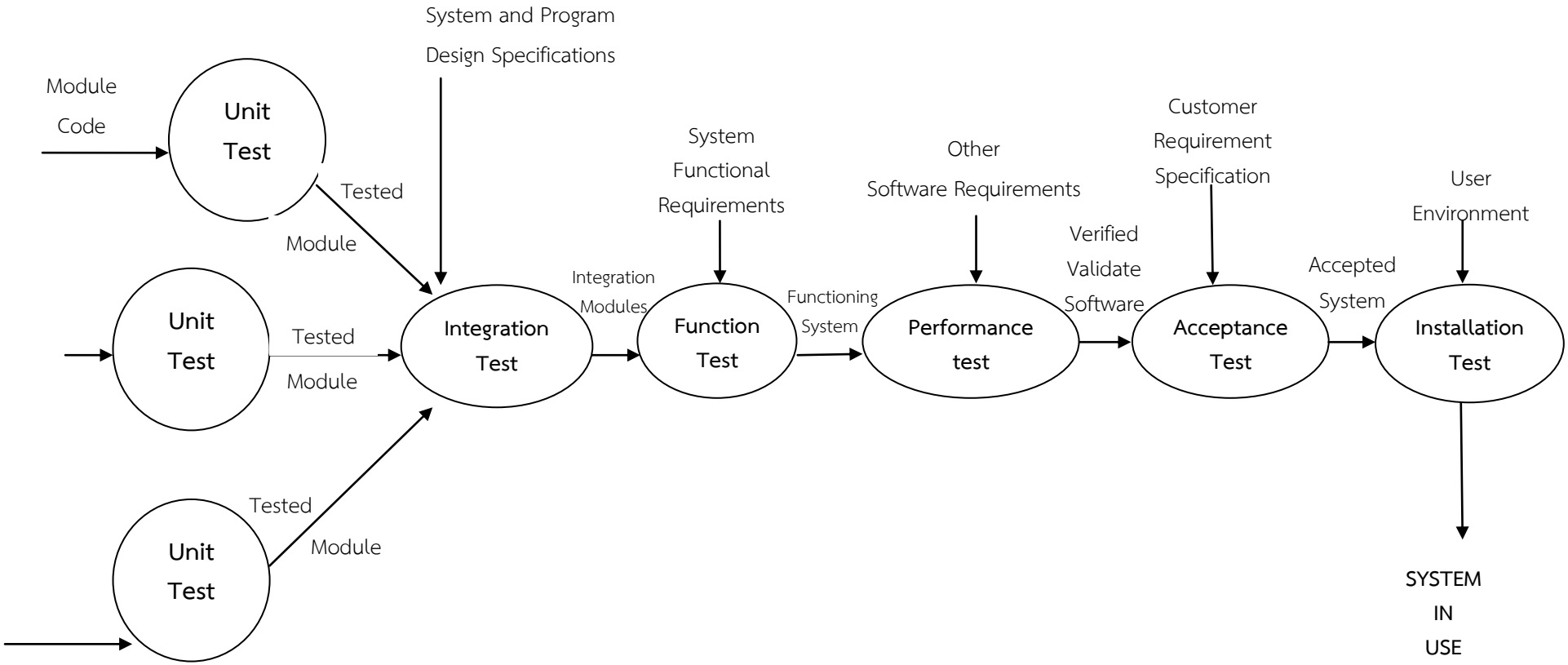
การทดสอบความผิดพลาดของซอฟต์แวร์แบ่งเป็น 2 ขั้นตอนคือ การทดสอบระดับโปรแกรมและการทดสอบระดับระบบ ดังมีรายละเอียดดังนี้

1. การทดสอบระดับโปรแกรม มีวัตถุประสงค์เพื่อตรวจสอบโปรแกรมที่ถูกสร้างขึ้นโดยโปรแกรมเมอร์เมื่อรวมกันเป็นระบบการทำงานแล้วสามารถกระทำงานได้ตามที่ออกแบบไว้ ความผิดพลาดที่อาจเกิดขึ้นในระดับนี้ได้แก่ ความผิดพลาดของขั้นตอนวิธี (algorithm error) ความผิดพลาดด้านไวยากรณ์ (syntax error) ความผิดพลาดทางด้านการคำนวณ (computation and precision error) ความผิดพลาดของเอกสาร (documentation error) ความผิดพลาดที่เกิดจากการปฏิบัติงานเกินสิ่งที่กำหนดไว้ (stress and overload error) ความผิดพลาดที่ระบบทำงานเกินประสิทธิภาพของระบบ (capacity or boundary error) ความผิดพลาดของเวลา (timing or coordination error) ความผิดพลาดในเรื่องประสิทธิภาพ (throughput หรือ performance error)ความผิดพลาดในการกู้คืน (recovery error) ความผิดพลาดในด้านฮาร์ดแวร์และซอฟต์แวร์ (hardware and system software error) ความผิดพลาดของมาตรฐานและกระบวนการ (standard and procedure error) เป็นต้น

2. การทดสอบระดับระบบ มีวัตถุประสงค์เพื่อตรวจสอบว่าระบบที่พัฒนาขึ้นสามารถแก้ปัญหาได้ตรงตามความต้องการที่ระบุในเอกสารกำหนดความต้องการ ความผิดพลาดที่อาจเกิดขึ้นในระดับนี้ได้แก่ การวิเคราะห์ความต้องการไม่ชัดเจน การแปลความหมายไม่ถูกต้อง การสื่อสารระหว่างทีมงานพัฒนาผิดพลาด การออกแบบระบบและการออกแบบโปรแกรมผิดพลาด การพัฒนาโปรแกรมและเอกสารผิดพลาด ความผิดพลาดที่อาจเกิดขึ้นนี้สามารถเกิดขึ้นจากลูกค้า นักออกแบบระบบ นักออกแบบโปรแกรม โปรแกรมเมอร์ ทีมงานทดสอบ รวมทั้งทีมงานในการบำรุงรักษาระบบ

ระยะของการทดสอบซอฟต์แวร์

การพัฒนาระบบงานขนาดใหญ่ การทดสอบซอฟต์แวร์แบ่งเป็น 6 ระยะได้แก่ การทดสอบมอดูลหรือทดสอบหน่วย การทดสอบรวม การทดสอบฟังก์ชัน การทดสอบประสิทธิภาพ การทดสอบการยอมรับและการทดสอบการติดตั้ง ดังภาพที่ 1



ภาพที่ 1 แสดงระยะของการทดสอบ

1. การทดสอบมอดูลหรือทดสอบหน่วย(module testing หรือ unit testing) เป็นระยะแรกของการทดสอบ หลังจากที่ผู้พัฒนาโปรแกรมได้เขียนคำสั่งโปรแกรมตามที่ได้รับมอบหมายเสร็จสิ้น ระยะนี้เป็นการค้นหาข้อผิดพลาดของมอดูลหรือหน่วยโปรแกรม อาจเป็นรูปแบบหรือไวยากรณ์ภาษาผิดพลาด ความหมายผิดพลาด สูตรการคำนวณผิดพลาด หรือลำดับของการทำงานผิดพลาด กระบวนการทดสอบหน่วยมีหลายวิธีการ ได้แก่

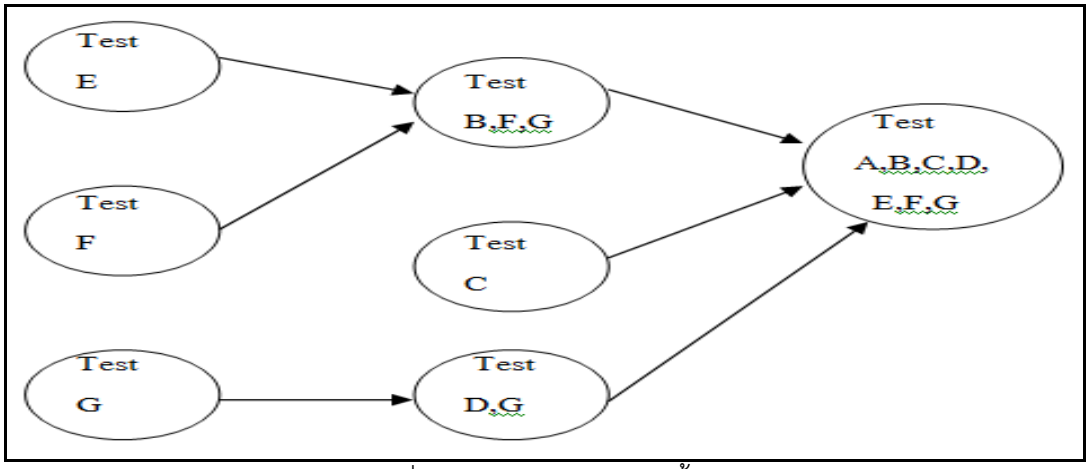
1.1 การทบทวนโปรแกรม (program review) เป็นกระบวนการทบทวนคำสั่งโปรแกรมและเอกสารที่โปรแกรมเมอร์พัฒนาขึ้นซึ่งแปลจากรายละเอียดที่ระบุในเอกสารการออกแบบโปรแกรมไปเป็นคำสั่งโปรแกรมและเอกสาร โดยกลุ่มผู้เชี่ยวชาญเพื่อตรวจสอบและค้นหาความผิดพลาดที่อาจเกิดขึ้นในโปรแกรมโดยวิธีการตรวจตลอดโปรแกรม (program walk-through) ซึ่งเป็นวิธีการทบทวนโปรแกรมที่ไม่ใช้คอมพิวเตอร์โดยผู้พัฒนา โปรแกรมจะนำคำสั่งโปรแกรมและเอกสารที่ได้จากการพัฒนาเสนอให้กับกลุ่มผู้เชี่ยวชาญเพื่อตรวจสอบความถูกต้องและค้นหาความผิดพลาดในโปรแกรมเป็นระยะ ๆ อย่างไม่เป็นทางการ หรืออาจใช้วิธีการตรวจสอบโปรแกรม (program inspection) เป็นการทบทวนโปรแกรมอย่างเป็นทางการมีเอกสารประกอบการตรวจสอบโดยการทบทวนโปรแกรมวิธีนี้ ผู้พัฒนาโปรแกรมต้องพัฒนาโปรแกรมให้เสร็จสมบูรณ์ก่อนจึงเสนอให้กลุ่มผู้เชี่ยวชาญเพื่อทำการทบทวนโปรแกรม

1.2 การพิสูจน์ความถูกต้องของโปรแกรม (proving programs correct) เป็นการทดสอบข้อเท็จจริงทางโครงสร้างของโปรแกรม พิจารณาการไหลของตรรกะโดยเขียนคำสั่งโปรแกรมใหม่ในเทอมของระบบตรรกะที่เป็นทางการ ซึ่งสามารถพิสูจน์นิพจน์ใหม่เหล่านี้ด้วยทฤษฎีบทคณิตศาสตร์ ซึ่งค่าความเป็นจริงของทฤษฎีสามารถนำมาใช้พิสูจน์ความถูกต้องของโปรแกรมได้

1.3 การทดสอบโปรแกรม (testing programs) เป็นการตรวจสอบความถูกต้องของโปรแกรมในมุมมองที่แตกต่าง โดยแสดงวิธีการที่โปรแกรมดำเนินการจากภายนอกโปรแกรม เป็นชุดของการทดลอง (experiment) ผลลัพธ์ที่ได้เป็นพื้นฐานในการตัดสินใจของโปรแกรมเพื่อทำงานในสถานการณ์ที่กำหนดในสภาพแวดล้อมจริง โดยทุก ๆ คำสั่งและการปฏิบัติการจะถูกประมวลผลอย่างน้อย 1 ครั้ง กลยุทธ์ของการทดสอบในลักษณะนี้เรียกว่ากล่องขาว(white box) การทดสอบโปรแกรมต้องกระทำอย่างละเอียด ข้อมูลทดสอบต้องแสดงพฤติกรรมที่เป็นไปได้ทั้งหมด

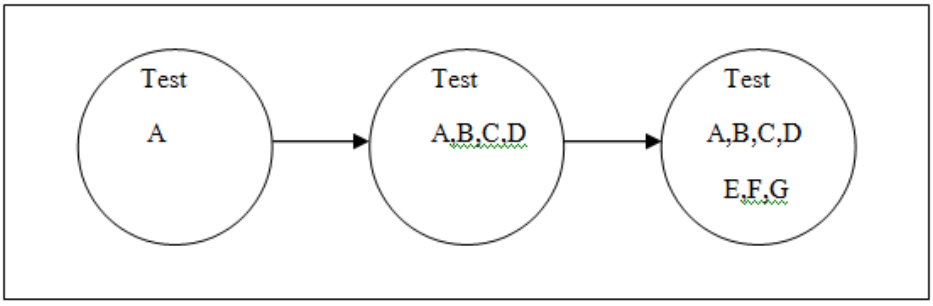
2. การทดสอบรวม (integration testing) เป็นการทดสอบการทำงานของมอดูลโปรแกรมทั้งหมด โดยนำมอดูลทั้งหมดมาทดสอบรวมกัน วิธีการทดสอบรวมนั้นจะถูกมองเป็นลำดับชั้นของมอดูลสอดคล้องกับการออกแบบเป็นมอดูล(modular design) แต่ละมอดูลจะอยู่ในชั้น (layer)ของการออกแบบ ซึ่งมีหลายวิธีการดังกล่าวต่อไปนี้

2.1 วิธีการจากล่างขึ้นบน(bottom-up approach) เป็นปรัชญาที่นิยมในการรวมมอดูลสำหรับทดสอบระบบขนาดใหญ่ วิธีการนี้มอดูลในระดับล่างสุดถูกทดสอบก่อน ต่อจากนั้นจะเรียกมอดูลในลำดับชั้นที่อยู่ก่อนนำมาทดสอบรวมกัน กระทำทีละลำดับชั้นจากล่างขึ้นบนจนกระทั่งทดสอบครบทุกมอดูล วิธีการนี้เป็นประโยชน์เมื่อมีรูทีนอัตถประโยชน์(utility routine)เป็นมอดูลในระดับล่างจำนวนมากที่ถูกเรียกใช้โดยมอดูลในระดับบน



ภาพที่ 2 แสดงวิธีการจากล่างขึ้นบน

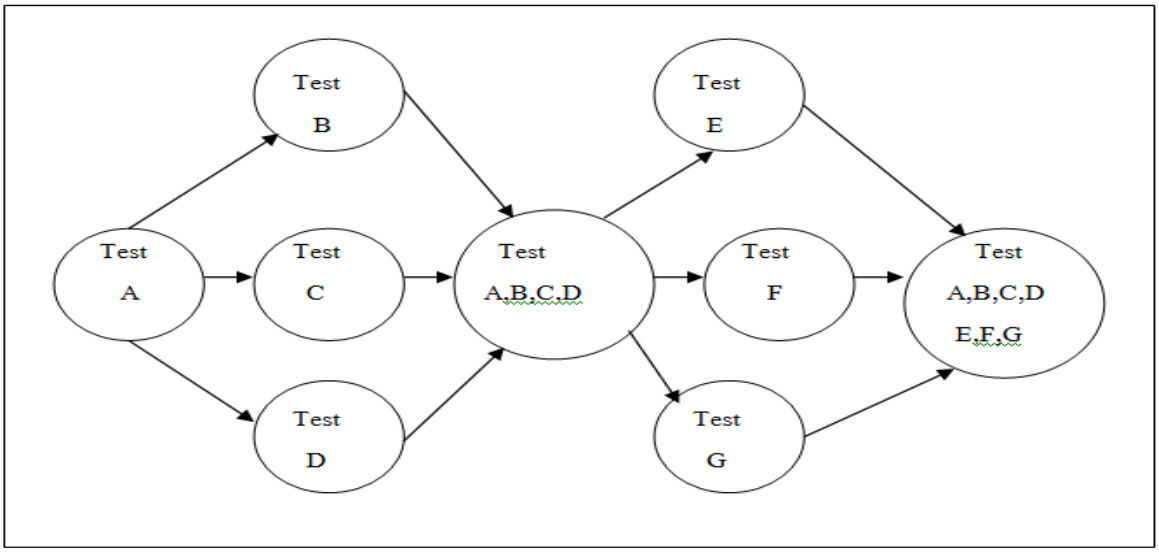
2.2 จากบนลงล่าง(top-down approach) การทดสอบวิธีการนี้จะเริ่มจากมอดูลระดับชั้นบนสุดซึ่งจะทดสอบตัวเอง ต่อจากนั้นมอดูลในลำดับชั้นถัดมาจะถูกเรียกเพื่อรวมและทดสอบเป็นหน่วยใหญ่ขึ้น และจะกระทำซ้ำในลำดับชั้นถัด ๆ มาจนกระทั่งทุกโมดูลรวมกันเพื่อทดสอบทั้งหมด วิธีการทดสอบวิธีนี้อาจเรียกมอดูลที่ยังไม่ได้ทำการทดสอบดังนั้นอาจต้องเขียนโปรแกรมจำลองกิจกรรมของมอดูลที่ขาดหายไป



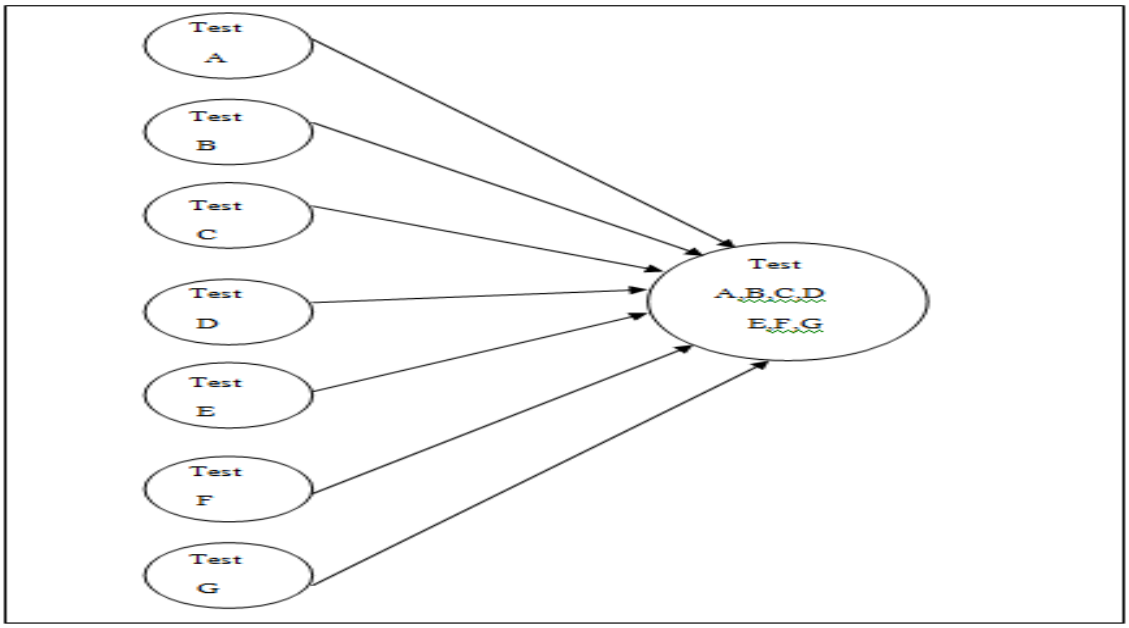
ภาพที่ 3 วิธีการจากบนลงล่าง

วิธีการหนึ่งในการหลีกเลี่ยงการจำลองกิจกรรมของมอดูลที่ขาดหายไปโดยการให้แต่ละระดับชั้นทำการทดสอบตนเองก่อนที่จะนำมารวมกัน เรียกว่า การทดสอบการปรับเปลี่ยนจากบนลงล่าง(modified top-down testing) ดังภาพที่ 4

2.3 วิธีการบิก-แบง(big-bang approach) เป็นการทดสอบรวมที่นำทุกมอดูลในระบบทั้งหมดทดสอบแยกออกจากกัน ต่อจากนั้นจึงนำมอดูลทั้งหมดมารวมกันเพื่อทดสอบรวมเพียงครั้งเดียว ดังภาพที่ 5

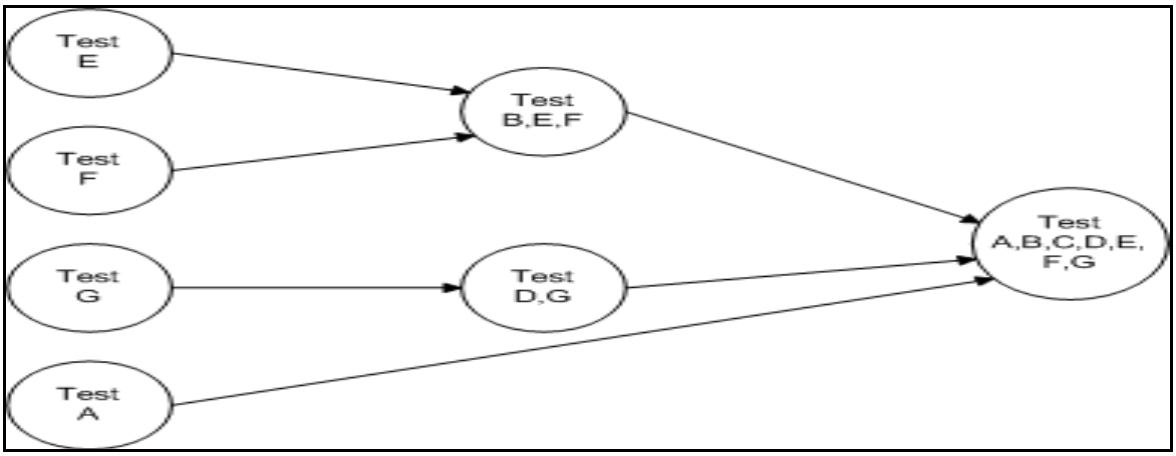


ภาพที่ 4 การทดสอบการปรับเปลี่ยนจากบนลงล่าง



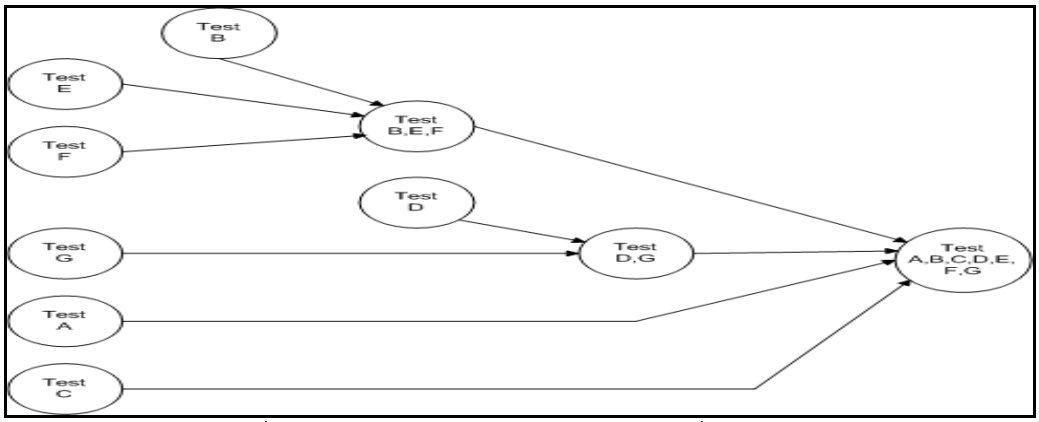
ภาพที่ 5 แสดง การทดสอบรวมวิธีการบิก-แบง

2.4 วิธีการแซนวิช(sandwich approach) เป็นวิธีการทดสอบรวมที่ผสมผสานระหว่างวิธีการจากบนลงล่างและวิธีการจากล่างขึ้นบน วิธีการนี้มองระบบเป็น 3 ชั้นเหมือนกับการทำแซนวิชได้แก่ ชั้นเป้าหมาย(target layer)ซึ่งเป็นชั้นที่อยู่ตรงกลาง ชั้นที่อยู่เหนือชั้นเป้าหมายและชั้นที่อยู่ล่างชั้นเป้าหมาย โดยการทดสอบรวมวิธีจากบนลงล่างถูกใช้ในระดับชั้นบนสุด และ วิธีการทดสอบรวมจากล่างขึ้นบนถูกใช้ในระดับชั้นล่างสุด การทดสอบถูกควบคุมโดยชั้นเป้าหมายซึ่งเลือกลำดับการทดสอบจากคุณลักษณะของระบบหรือโครงสร้างลำดับชั้นของระบบ ดังภาพที่ 6



ภาพที่ 6 แสดง การทดสอบรวมวิธีการแซนวิช

อาจปรับเปลี่ยนวิธีการทดสอบรวมเพื่อให้มอดูลในระดับชั้นเป้าหมายสามารถทดสอบตนเองก่อนที่นำมาทดสอบรวมได้ ดังภาพที่ 7



ภาพที่ 7 แสดงการทดสอบรวมวิธีการปรับเปลี่ยนแซนวิช

การเลือกวิธีการในการทดสอบรวมนั้นขึ้นอยู่กับคุณลักษณะของระบบและความคาดหวังของลูกค้าที่ต้องการเห็นภาพการทำงานของระบบ ผู้ทดสอบต้องสร้างแผนงานการทดสอบรวมให้ระบบสามารถทำงานพื้นฐานได้ตั้งแต่ระยะเริ่มต้นของกระบวนการทดสอบ ซึ่งทั้งผู้ทดสอบและโปรแกรมเมอร์จะทำงานขนานกันไป

3. การทดสอบฟังก์ชัน(function testing) เป็นการทดสอบมุ่งเน้นไปที่การทำงานตามที่ลูกค้าต้องการ โดยตรวจสอบจากเอกสารระบุความต้องการ โดยเปรียบเทียบระหว่างระบบที่สร้างขึ้นกับเอกสารที่ระบุความต้องการ ในระยะนี้ทีมงานทดสอบไม่สนใจโครงสร้างของระบบแต่สนใจเฉพาะกิจกรรมที่ระบบสามารถกระทำได้เท่านั้นเป็นลักษณะกล่องปิด(closed box)ดังนั้นในการทดสอบจึงจำเป็นอย่างยิ่งที่ต้องทราบถึงหน้าที่และกิจกรรมต่าง ๆ ที่ต้องการให้ระบบสามารถกระทำได้

กิจกรรมหรือหน้าที่ต่าง ๆ ที่ระบบกระทำนั้นประกอบด้วยกลุ่มมอดูลต่าง ๆ ที่ทำงานร่วมกันเรียกว่าเธรด (thread) ซึ่งการทดสอบในระยะนี้บางครั้งเรียกว่าการทดสอบเธรด(thread testing) สำหรับกลุ่มมอดูลที่มีขนาดเล็ก ผู้ทดสอบสามารถค้นพบหาความผิดพลาดได้ง่ายกว่ากลุ่มของมอดูลที่มีขนาดใหญ่ ดังนั้นกิจกรรมหรือหน้าที่อาจถูกกำหนดเป็นระดับที่ซ้อนกันเรียกว่าสปิน(spin)

วิธีการวิเคราะห์ความต้องการเพื่อสร้างกรณีทดสอบนั้นสามารถสร้างเป็นความสัมพันธ์ทางตรรกะระหว่างอินพุตและเอาต์พุต หรือระหว่างอินพุตและการเปลี่ยนรูป เรียกอินพุตว่าเหตุ(cause) เรียกเอาต์พุตและการเปลี่ยนรูปว่าผล(effect) ผลลัพธ์จะอยู่ในรูปของกราฟบูลีนซึ่งสะท้อนความสัมพันธ์เรียกกว่ากราฟเหตุและผล(cause and effect graph)

4. การทดสอบประสิทธิภาพ(performance testing) เป็นการเปรียบเทียบมอดูลที่ผ่านการทดสอบรวม (integrated module) กับความต้องการที่ไม่เป็นฟังก์ชัน โดยการทดสอบสามารถกระทำในสภาพแวดล้อมผู้ใช้งานจริง เรียกว่าระบบตรวจสอบความสมเหตุสมผล(validated system) หรือทดสอบในสภาพแวดล้อมจำลอง เรียกว่าระบบทวนสอบ(verified system) อ้างอิงกับวัตถุประสงค์ของลูกค้าบนพื้นฐานของความต้องการ ได้แก่

4.1 การทดสอบความตึงเครียด (stress test) เป็นการทดสอบความสามารถของระบบเมื่อเกิดความตึงเครียดในช่วงเวลาสั้น ๆ เช่น การทดสอบอุปกรณ์และผู้ใช้ระบบตามที่ระบุไว้พร้อมกันโดยปกติเป็นจำนวนที่มากที่สุดที่ระบบสามารถปฏิบัติได้

4.2 การทดสอบปริมาณ (volume test) เป็นการทดสอบปริมาณของข้อมูลที่มากที่สุดที่ระบบสามารถจัดการได้ รวมทั้งตรวจสอบขนาดและการใส่ข้อมูลในรายการข้อมูล (field) ระเบียน (record) และแฟ้มข้อมูล(file)ว่ามีขนาดที่เหมาะสมกับปริมาณข้อมูล

4.3 การทดสอบโครงแบบ (configuration test) เป็นการวิเคราะห์โครงแบบฮาร์ดแวร์และซอฟต์แวร์ที่ระบุในเอกสารความต้องการ โดยระบบอาจมีการให้บริการหลายกลุ่มหลายโครงแบบ ผู้ใช้คนเดียวหรือผู้ใช้หลายคน การทดสอบโครงแบบจะประเมินโครงแบบที่เป็นไปได้ทั้งหมดเพื่อให้แน่ใจว่าระบบสามารถทำงานตามความต้องการได้

4.4 การทดสอบความเข้ากันได้ (compatibility test) เป็นการทดสอบที่จำเป็นสำหรับระบบที่มีการปฏิสัมพันธ์กับระบบอื่น ถ้าระบบมีการติดต่อสื่อสารกับฐานข้อมูลขนาดใหญ่

การทดสอบความเข้ากันได้จะตรวจสอบความเร็ว ความถูกต้องในการดึงข้อมูล

4.5 การทดสอบความถดถอย (regression test) เป็นการทดสอบที่จำเป็นสำหรับการนำระบบงานใหม่แทนที่ระบบงานที่กำลังปฏิบัติงานอยู่ เพื่อรับประกันว่าระบบใหม่มีประสิทธิภาพดีกว่า ใช้ทดสอบสำหรับการพัฒนาที่มีหลายระยะ(phased development)

4.6 การทดสอบความปลอดภัย (security test) เป็นการทดสอบความปลอดภัยของระบบ ซึ่งได้ระบุไว้ในเอกสารความต้องการ โดยตรวจสอบการเข้าถึงฟังก์ชันการทำงาน การเข้าถึงข้อมูลของผู้ใช้ระดับต่าง ๆ

4.7 การทดสอบระยะเวลา (timing test) เป็นการประเมินผลเวลาตอบสนองของผู้ใช้งานที่กระทำหน้าที่ต่าง ๆ ของระบบ

4.8 การทดสอบสภาพแวดล้อม (environmental test) เป็นการพิจารณาความสามารถของระบบที่สามารถปฏิบัติงานในสถานที่ติดตั้ง รวมถึงความคลาดเคลื่อนที่เกิดจากความร้อน(heat) ความชื้นสัมพัทธ์ (humidity) การเคลื่อนไหว (motion) การปรากฏของสารเคมี (chemical presence) ความชื้น(moisture) ความสามารถในการเคลื่อนย้าย(portability) สนามไฟฟ้าหรือแม่เหล็ก(electrical and magnetic field) การหยุดชะงักของการใช้พลังงาน (disruption of power) หรือ คุณลักษณะอื่น ๆ เพื่อรับประกันว่าระบบมีประสิทธิภาพภายใต้เงื่อนไขที่กำหนด

4.9 การทดสอบคุณภาพ(quality test) เป็นการประเมินผลความน่าเชื่อถือ (reliability)การบำรุงรักษาระบบ(maintainability) และความพร้อมของระบบ(availability)

4.10 การทดสอบการกู้คืน (recovery test) เป็นการทดสอบการตอบสนองของระบบกรณีเกิดข้อผิดพลาด การสูญหายของข้อมูล อุปกรณ์ หรือพลังงานไฟฟ้า เพื่อให้เห็นว่าระบบสามารถกู้คืนได้อย่างถูกต้อง

4.11 การทดสอบการบำรุงรักษา (maintenance test) เป็นการตรวจสอบเครื่องมือหรือกระบวนการที่ช่วยสำหรับวิเคราะห์ความผิดพลาด ได้แก่ โปรแกรมวิเคราะห์ความผิดพลาด (diagnostic) แผนผังแสดงการใช้หน่วยความจำ (memory map) ร่องรอยของการทำรายการ (trace of transaction) แผนภาพของวงจร (diagram of circuitry) เป็นต้น โดยตรวจสอบเครื่องมือที่ใช้ในการบำรุงรักษาระบบที่ระบุไว้ในเอกสารความต้องการเหล่านี้มีอยู่จริงสามารถนำมาใช้งานได้อย่างถูกต้อง

4.12 การทดสอบเอกสาร (documentation test) เป็นการตรวจสอบเอกสารคู่มือผู้ใช้ เอกสารบำรุงรักษาระบบ เอกสารทางเทคนิคที่จำเป็น ที่มีอยู่ว่ามีความคงที่ ถูกต้อง และง่ายต่อการอ่าน มีรูปแบบตรงตามที่ระบุไว้

4.13 การทดสอบมนุษยปัจจัย (human factor test) มุ่งเน้นไปที่การปฏิสัมพันธ์ของผู้ใช้โดยทดสอบการแสดงผลทางจอภาพ ข่าวสารต่าง ๆ ที่แสดง รูปแบบของรายงาน หรือด้านอื่น ๆ ที่ทำให้ผู้ใช้สะดวกสบาย ใช้งานได้ง่าย

5. การทดสอบการยอมรับ (acceptance testing) การทดสอบในระยะนี้เป็นการเปรียบเทียบประสิทธิภาพของระบบกับความคาดหวังของลูกค้า โดยตรวจสอบจากเอกสารกำหนดความต้องการ (requirement definition document) ตรวจสอบคุณลักษณะของระบบเพื่อให้แน่ใจว่าระบบสามารถทำงานตามที่ได้กำหนดไว้ได้ วิธีการในการทดสอบการยอมรับมีหลายวิธีการ ดังกล่าวต่อไปนี้

5.1 ทดสอบเกณฑ์มาตรฐาน (benchmark test)การทดสอบด้วยวิธีนี้ลูกค้าจะเตรียมกลุ่มของกรณีทดสอบซึ่งแทนการปฏิบัติงานของระบบจริง ๆ ลูกค้าจะทดสอบประสิทธิภาพของระบบในแต่ละกรณีทดสอบ โดยใช้ผู้ใช้ที่กระทำงานจริง ๆ ในระบบหรืออาจเป็นทีมงานทดสอบที่ปฏิบัติงานเฉพาะกรณีก็ได้ วิธีการนี้ลูกค้าสามารถทดสอบความต้องการพิเศษเฉพาะ เช่น ทดสอบความเร็วของข่ายงานเสียงหรือทดสอบการใช้งานข้อมูล สำหรับระบบงานที่ต้องการจัดซื้อจากผู้จัดจำหน่าย ลูกค้าสามารถทดสอบระบบโดยใช้วิธีการนี้เพื่อตัดสินใจเลือกระบบที่ตรงกับความต้องการมากที่สุด

5.2 ทดสอบนำร่อง (pilot test) เป็นการทดสอบบนพื้นฐานของการทดลอง โดยผู้ใช้ทำการทดสอบฟังก์ชันทั้งหมดที่ต้องทำงานเป็นประจำทุกๆวัน การทดสอบวิธีนี้เป็นโครงสร้างและเป็นทางการน้อยกว่าวิธีการแรก แต่วิธีนี้เหมาะสำหรับระบบที่มีปรับปรุงหรือแก้ไขการทำงานของระบบเดิม เช่น ระบบสำนักงานอัตโนมัติ หรือระบบปฏิบัติงานเวอร์ชันใหม่ การทดสอบนำร่องอาจกระทำการทดสอบรอบแรก(alpha test) และทดสอบรอบสอง(beta test) โดย

- การทดสอบรอบแรก(alpha test) หมายถึง การนำเอาโปรแกรมไปให้ผู้ใช้ทดลองใช้โดยใช้ข้อมูลสมมุติ เพื่อให้ผู้ใช้ตรวจสอบเบื้องต้น
- ทดสอบรอบสอง(beta test) หมายถึง การนำเอาโปรแกรมไปให้ผู้ใช้ทดลองใช้เป็นครั้งที่สอง โดยใช้ข้อมูลจริงภายใต้สถานการณ์จริงในการทดสอบ

5.3 ทดสอบแบบขนาน(parallel test) ถ้าพัฒนาระบบใหม่แทนที่ระบบปัจจุบันหรือระบบใหม่เป็นส่วนหนึ่งของระยะของการพัฒนา (phase development) การทดสอบแบบขนานเป็นวิธีการหนึ่งที่เหมาะสมเพราะเป็นวิธีการทดสอบที่ทำให้ผู้ใช้คุ้นเคยกับระบบใหม่ก่อน วิธีการทดสอบแบบขนานนั้นระบบเก่าและระบบใหม่ทำงานขนานกันไป ผู้ใช้ระบบสามารถเปรียบเทียบและเห็นความแตกต่างระหว่างระบบใหม่กับระบบเก่าทำให้ผู้ใช้ระบบเกิดความเชื่อมั่น

6. การทดสอบการติดตั้ง(installation testing) เป็นการทดสอบระบบเพื่อให้แน่ใจว่าสามารถกระทำงานได้จริง โดยทดสอบการติดตั้งระบบในสภาพแวดล้อมจริง เป็นการตั้งค่าระบบในสภาพแวดล้อมของผู้ใช้ ติดตั้งอุปกรณ์ต่างๆ จัดสรรแฟ้มข้อมูลกำหนดการเข้าถึงฟังก์ชันและข้อมูล การติดต่อสื่อสารกับระบบอื่นได้ การทดสอบวิธีการนี้มุ่งไปที่ความสมบูรณ์(completeness)ของการติดตั้งระบบและการตรวจสอบ(verification) คุณลักษณะที่เป็นฟังก์ชันและไม่เป็นฟังก์ชันที่ส่งผลต่อสถานที่ติดตั้ง ถ้าลูกค้าพอใจในผลของการทดสอบ ระบบพร้อมส่งมอบให้ลูกค้าต่อไป

เครื่องมือทดสอบซอฟต์แวร์

เครื่องมือที่สามารถนำมาใช้ช่วยในการทดสอบซอฟต์แวร์นั้นมีมากมาย ส่วนมากเป็นเครื่องมืออัตโนมัติเพื่อใช้สำหรับจับ(capture) ข้อมูล เพื่อใช้สำหรับคำนวณประสิทธิภาพของระบบ เช่น

1 .เครื่องมือจำลอง (simulator) เป็นเครื่องมือที่ช่วยแสดงคุณลักษณะของอุปกรณ์และระบบที่ไม่มีเครื่องมือหรืออุปกรณ์จริง เช่น การจำลองการบินที่ไม่มีเครื่องบินจริง ๆ แต่มีอุปกรณ์จำลองที่สามารถให้ผู้ใช้ได้เรียนรู้การบินจากเครื่องบินจำลองนี้ เป็นต้น เครื่องมือจำลองจะรายงานสถานะเมื่อเกิดความผิดพลาด ช่วยหาแหล่งของข้อผิดพลาด ช่วยทดสอบความตึงเครียด(stress test)และทดสอบปริมาณ (volume test)

2. ระบบเฝ้าสังเกต (monitor) เป็นอุปกรณ์ในการดักจับข้อมูลที่ผ่านจากโพรเซสหรืออุปกรณ์ตัวหนึ่งไปยัง โพรเซสหรืออุปกรณ์อีกตัวหนึ่ง เช่น ระบบเฝ้าสังเกตข้อมูลเข้าและข้อมูลออกระหว่างโพรเซสเซอร์ 2 ตัว ซึ่งสามารถเก็บเหตุการณ์ก่อนและหลังจากเหตุการณ์ที่มีข้อผิดพลาดเกิดขึ้นโดยบันทึกเป็นสำเนาชั่วคราว(snapshot) ซึ่งช่วยให้ค้นหาแหล่งของความผิดพลาดและตรวจสอบประสิทธิภาพที่เหมาะสม

3.เครื่องมือการวิเคราะห์(analyzer) เป็นเครื่องมือในการวิเคราะห์ข้อมูลต่างๆตามระเบียบที่กำหนด ได้แก่

3.1การวิเคราะห์แบบคงที่(static analysis) เป็นเครื่องมือที่ช่วยทีมงานทดสอบและโปรแกรมเมอร์วิเคราะห์ความถูกต้องของโปรแกรมต้นฉบับ(source program)ก่อนทำงาน (run) ได้แก่

- เครื่องมือวิเคราะห์คำสั่ง(code analyzer) เป็นเครื่องมือสำหรับวิเคราะห์ความถูกต้องของรูปแบบหรือไวยากรณ์ภาษาโดยอัตโนมัติ ประโยคหรือคำสั่งใดที่ผิดรูปแบบภาษา จะถูกกำหนดแถบสว่างเพื่อบอกถึงตำแหน่งที่ผิดให้แก่ผู้ทดสอบ
- เครื่องมือตรวจสอบโครงสร้าง(structure checker) เป็นเครื่องมือที่ตรวจสอบโครงสร้างของมอดูล อาจสร้างในรูปของกราฟ หรือลำดับชั้นของมอดูลหรือกลุ่มของมอดูล เพื่อหาข้อบกพร่องของโครงสร้าง
- เครื่องมือวิเคราะห์ข้อมูล(data analyzer) เป็นเครื่องมือตรวจสอบโครงสร้างข้อมูล การประกาศข้อมูลและการโต้ตอบของมอดูล เครื่องมือชนิดนี้จะบันทึกความผิดพลาดที่เกิดขึ้นระหว่างมอดูล รวมทั้งการกำหนดข้อมูลและการใช้ข้อมูลที่ผิดพลาด
- เครื่องมือตรวจสอบลำดับ(sequence checker) เป็นเครื่องมือที่ตรวจสอบลำดับของเหตุการณ์ต่างๆที่เกิดขึ้นในมอดูล ถ้าเกิดความผิดพลาดจะทำเครื่องหมายหรือบันทึกตำแหน่งที่ผิดพลาดเพื่อให้ผู้ทดสอบทราบ

3.2 การวิเคราะห์แบบไดนามิก (dynamic analysis)

เป็นเครื่องมือที่ช่วยทีมงานทดสอบและโปรแกรมเมอร์ทราบเหตุการณ์ต่าง ๆ ที่เกิดขึ้นระหว่างการทำงานของโปรแกรมต้นฉบับ(source program) ซึ่งจะทำหน้าที่ดูและรายงานพฤติกรรมของโปรแกรม รายงานเวลาที่ปฏิบัติงานในแต่ละมอดูล รายงานจุดตัดสินใจที่มีการกระโดดข้ามไปในโปรแกรม บันทึกจำนวนครั้งที่มอดูลย่อยถูกเรียกหรือบรรทัดของคำสั่งที่ถูกปฏิบัติการ ซึ่งข่าวสารต่างๆที่ได้รับนี้ช่วยทีมงานทดสอบในการคำนวณประสิทธิภาพของระบบอีกทั้งสถิติต่างๆในการปฏิบัติงานของโปรแกรมสามารถนำไปเป็นข้อมูลใช้สร้างตัวแปรเฉพาะต่าง ๆ เช่น ค่าเริ่มต้น ค่าสุดท้าย ค่าน้อยสุด ค่ามากสุด หรือ จุดพัก (breakpoint) ในการทดสอบ เป็นต้น

เอกสารทดสอบซอฟต์แวร์

ระบบที่มีความซับซ้อนที่มีการประมวลผลแบบกระจายหรือการประมวลผลแบบทันที(real time) การทดสอบระบบที่มีผู้ใช้จำนวนมากในลักษณะนี้ทำให้การทดสอบมีความซับซ้อนและยาก วิธีการในการควบคุมความซับซ้อนและความยากในการทดสอบนั้นสามารถกระทำได้โดยออกแบบเอกสารทดสอบที่มีความสมบูรณ์และครอบคลุม เอกสารที่จำเป็นมีหลายชนิด ได้แก่ แผนทดสอบ (test plan) ข้อกำหนดการทดสอบและประเมินผล (test specification and evaluation) รายละเอียดการทดสอบ(test description) และรายงานการวิเคราะห์การทดสอบ(test analysis report) ดังรายละเอียดต่อไปนี้

1.แผนทดสอบ (test plan) เป็นเอกสารในการวางรูปแบบของการทดสอบระบบทั้งหมด โดยแบ่งระบบเป็นฟังก์ชันย่อย องค์ประกอบของแผนทดสอบประกอบด้วย

- วัตถุประสงค์ (objective) โดยทั่วไปการวางแผนการทดสอบเริ่มจากระบุวัตถุประสงค์ของการทดสอบโดยแนะนำวิธีการจัดการทดสอบ แนะนำเทคนิคต่าง ๆ ที่ใช้ในระหว่างการทดสอบ สร้างแผนงานและกำหนดระยะเวลาในการทดสอบ รวมทั้งระบุอุปกรณ์ที่จำเป็น วิธีการทดสอบ ผลลัพธ์ที่ต้องการ อธิบายลักษณะและขอบเขตของแต่ละการทดสอบ อธิบายถึงวิธีการของการทดสอบที่สามารถประเมินหน้าที่และประสิทธิภาพของระบบได้ บรรยายข้อมูลทดสอบและผลที่คาดว่าจะได้รับ

- การอ้างอิงเอกสาร(document reference) แผนการทดสอบต้องอ้างอิงกับเอกสารที่ผลิตขึ้นระหว่างการพัฒนาโครงงาน ซึ่งแผนทดสอบจะอธิบายความสัมพันธ์ระหว่างเอกสารระบุความต้องการ เอกสารการออกแบบ เอกสารการสร้างระบบ และกระบวนการทดสอบ

- การสรุประบบ (system summary) เป็นการกำหนดแผนงานและเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในการทดสอบโดยคร่าว ๆ เช่นข้อมูลนำเข้าและผลลัพธ์ที่สำคัญเท่านั้น

- การทดสอบที่สำคัญ(major test) เป็นการบรรยายวิธีการที่ใช้ในการทดสอบ ซึ่งแผนทดสอบมีความแตกต่างกันในการทดสอบฟังก์ชัน ทดสอบประสิทธิภาพ ทดสอบการยอมรับ และทดสอบการติดตั้ง

- แผนงาน (schedule) เป็นการวางแผนกรอบเวลาในรูปของแผนผังหลักชัย (milestone) หรือกราฟ

- กิจกรรม (activity graph) ประกอบด้วยระยะเวลาการทดสอบทั้งหมด เวลาเริ่มต้นและเวลาหยุดของการทดสอบแต่ละส่วนย่อย ความต้องการก่อนการทดสอบ เวลาสำหรับเตรียมและทบทวนรายงาน การวิเคราะห์การทดสอบ ถ้าการทดสอบมีสถานที่หลายแห่ง แผนการทดสอบต้องมีแผนงานการทดสอบของแต่ละ สถานที่ ซึ่งแสดงฮาร์ดแวร์ ซอฟต์แวร์ และบุคคลที่จำเป็นสำหรับการบริหารจัดการทดสอบ ช่วงเวลาของการใช้ทรัพยากรต่าง ๆ การฝึกอบรมพิเศษและการบำรุงรักษาที่จำเป็น

- วัสดุที่จำเป็น (material need) การวางแผนทดสอบต้องกำหนดวัสดุหรือปัจจัยที่จำเป็นในการส่งมอบระบบ เช่น ผู้ใช้ คู่มือผู้ควบคุมเครื่อง รายการตัวอย่าง อุปกรณ์ทดสอบ ตารางฐานข้อมูลหรือสื่อเก็บข้อมูล ซึ่งปัจจัยหรือวัสดุเหล่านี้จะถูกส่งไปยังสถานที่ทดสอบ เช่นการทดสอบระบบการจัดการฐานข้อมูล ผู้ทดสอบสร้างฐานข้อมูลสมมุติขึ้นมาเพื่อให้ผู้ใช้จริงทดลองใช้ก่อนที่ทีมงานทดสอบจริงจะทำการทดสอบ หรือการทดสอบความปลอดภัยหรือสิทธิต่าง ๆ ในการทำงาน ผู้ใช้อาจสร้างรหัสผ่าน(password) หรือ สิทธิในการเข้าถึงพิเศษสำหรับทีมงานทดสอบ ก่อนที่การทดสอบจะเริ่มขึ้น เป็นต้น

2. ข้อกำหนดการทดสอบและประเมินผล (test specification and evaluation) เริ่มจากเขียนรายการความต้องการของระบบย่อยซึ่งอ้างอิงจากเอกสารกำหนดความต้องการ เพื่อให้ผู้ทดสอบสามารถมองเห็นความสมนัย(correspondence) ระหว่างความต้องการและการทดสอบ โดยทั่วไปจะสร้างอยู่ในรูปของตารางหรือแผนผัง เป็นแนวทางที่ผู้ทดสอบสามารถนำมาใช้กำหนดวิธีการในการทดสอบได้

3.รายละเอียดการทดสอบ(test description) เป็นเอกสารที่ถูกเขียนขึ้นสำหรับ

ทดสอบข้อกำหนดต่างๆของความต้องการอย่างละเอียดและชัดเจน ประกอบด้วยข้อมูล กระบวนการ และวิธีการควบคุม สำหรับการทดสอบข้อมูลสามารถพิจารณาได้ในหลายส่วน ได้แก่ ข้อมูลนำเข้า (input data) คำสั่งนำเข้า (input command) ข้อมูลผลลัพธ์(output data) และข่าวสารที่สร้างโดยระบบ (system message) โดยทั่วไปเรียกกระบวนการทดสอบว่าบททดสอบ (test script) เนื่องจากมีรายละเอียดการทดสอบทีละขั้น (step-by-step) บททดสอบนี้จะอธิบายให้เห็นเหตุการณ์ต่างๆที่เกิดขึ้นจริงในการปฏิบัติการการปรากฏสารสนเทศบนหน้าจอ หรือผลลัพธ์ที่เกิดขึ้นจากการทำงานเป็นลำดับของกิจกรรมที่ต้องการทดสอบไปจนกระทั่งสิ้นสุดการทดสอบ

4. รายงานการวิเคราะห์การทดสอบ (test analysis report) เป็นเอกสารที่บรรยายผลลัพธ์ของการทดสอบและในกรณีที่เกิดข้อผิดพลาด เอกสารนี้จะเตรียมสารสนเทศที่จำเป็นเพื่อให้ทราบแหล่งของความผิดพลาดทำให้สามารถแก้ไขข้อผิดพลาดที่เกิดขึ้นได้ง่ายขึ้น นอกจากนี้ยังให้ข้อมูลที่จำเป็นในการตรวจสอบเมื่อโครงงานที่พัฒนาเสร็จสมบูรณ์เป็นการสร้างความเชื่อมั่นในประสิทธิภาพของระบบ ข้อผิดพลาดหรือข้อบกพร่องที่เกิดขึ้นสามารถบันทึกไว้ในแบบรายงานความคลาดเคลื่อน (discrepancy report form : DRF) ซึ่งประกอบไปด้วยสถานะของระบบก่อนเกิดข้อผิดพลาด เหตุการณ์ที่เกิดข้อผิดพลาด กิจกรรมหรือกระบวนการที่ปรากฏขึ้นนำไปสู่ข้อผิดพลาดที่เกิดขึ้น รายละเอียดของระบบที่ควรทำงานในกรณีที่ไม่เกิดความผิดพลาด การอ้างอิงที่เกี่ยวข้องกับความต้องการ ผลกระทบของข้อผิดพลาดที่เกิดขึ้นในระบบ ระดับความรุนแรง เป็นต้น ผู้พัฒนาสามารถใช้สารสนเทศของแบบรายงานความคลาดเคลื่อนเพื่อตัดสินใจถึงสิ่งที่ควรกระทำ แบบรายงานความคลาดเคลื่อนถูกกำหนดรูปแบบโดยทีมจัดการโครงแบบ (configuration management team)ซึ่งมีหน้าที่รับผิดชอบถึงการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นในระบบเมื่อมีข้อผิดพลาดเกิดขึ้น ทราบตำแหน่งที่เกิดความผิดพลาด และผลกระทบต่าง ๆ โดยอ้างอิงกับข้อกำหนดการทดสอบ สำหรับข้อผิดพลาดที่เกิดขึ้นมีระดับความรุนแรงแตกต่างกัน ซึ่งกรณีที่มีความรุนแรงระดับต่ำทีมงานทดสอบสามารถทำการทดสอบในขั้นตอนอื่นๆต่อไปนี้ แต่ถ้าเป็นความรุนแรงระดับสูงอาจตัดสินใจหยุดการทดสอบเพื่อแก้ไข

สรุป

การทดสอบความผิดพลาดของซอฟต์แวร์แบ่งเป็น 2 ขั้นตอนคือ การทดสอบระดับโปรแกรมและการทดสอบระดับระบบ แบ่งเป็น 6 ระยะได้แก่ การทดสอบมอดูลหรือทดสอบหน่วย การทดสอบรวม การทดสอบฟังก์ชัน การทดสอบประสิทธิภาพ การทดสอบการยอมรับและการทดสอบการติดตั้ง เครื่องมือทดสอบซอฟต์แวร์ได้แก่ เครื่องมือจำลอง (simulator) ระบบเฝ้าสังเกตุ (monitor) และเครื่องมือการวิเคราะห์(analyzer) เอกสารทดสอบซอฟต์แวร์ได้แก่ แผนทดสอบ (test plan) ข้อกำหนดการทดสอบและประเมินผล(test specification and evalution) รายละเอียดการทดสอบ(test description) และรายงานการวิเคราะห์การทดสอบ(test analysis report)

บรรณานุกรม

S.A. Kelkar. (2007**). Software Engineering - A concise Study**. Prentice Hall of India Private Limited. New Delhi.

Shari Lawrence Pfleeger. (1991). **Software Engineering - The Production of Quality Software**. Macmillan Publishing.