บทที่ 1 หลักการทดสอบซอฟต์แวร์ (Software Testing Concepts)

ในบทนี้จะเกริ่นนำถึงหลักการ ความหมายของการทดสอบซอฟต์แวร์ วัตถุประสงค์ของการทดสอบ ความเชื่อและความจริงเกี่ยวกับการทดสอบซอฟต์แวร์ ข้อจำกัดของการทดสอบ ใครเป็นผู้ทดสอบซอฟต์แวร์ ความแตกต่างของผู้ทดสอบอิสระ กับผู้พัฒนาซอฟต์แวร์ กระบวนการและกิจกรรมของการทดสอบ คำถามเกี่ยวกับการทดสอบในแง่มุมต่างๆ ทัศนคติของนักทดสอบที่ดี

1.1 ความหมายของการทดสอบ

คงไม่มีใครปฏิเสธได้ว่าไม่รู้จักคำว่า “ทดสอบ” เพราะตั้งแต่เริ่มเข้าเรียนได้ระยะหนึ่ง นักเรียนทุกคนก็ต้องพบกับการทดสอบสิ่งที่ได้เรียนไปแล้วว่าประสบผลสำเร็จในการเรียนเพียงใด ยิ่งโตขึ้น เรียนชั้นสูงขึ้นก็พบกับการทดสอบมากขึ้น หรือในบางกรณี อาจมีการทดสอบก่อนที่จะเข้าเรียน เพื่อจัดระดับที่เหมาะสม เช่น การเรียนภาษาอังกฤษ หรือการศึกษาต่อในระดับสูง ก็ต้องวัดว่ามีศักยภาพในการเรียนต่อสำหรับสาขานั้นๆ หรือไม่ ดังนั้น การทดสอบจึงทำขึ้นเพื่อวัดคุณภาพของสิ่งที่ต้องการ ที่กล่าวมานี้เป็นการทดสอบความรู้โดยทั่วไป แต่ในการเรียนวิชานี้จะเน้นที่การทดสอบซอฟต์แวร์ที่สร้างขึ้นเท่านั้น

การทดสอบซอฟต์แวร์ เป็นกระบวนการในการวิเคราะห์ส่วนต่างๆภายในซอฟต์แวร์เพื่อตรวจสอบคุณภาพ เช่น ฟังก์ชันการทำงาน ความถูกต้อง ประสิทธิภาพต่างๆ ของซอฟต์แวร์ รวมถึงการเอ็กซิคิวท์โปรแกรมเพื่อค้นหาข้อผิดพลาด ซึ่งส่วนที่ไม่เป็นไปตามความต้องการของระบบหรือผู้ใช้ ข้อผิดพลาดที่เกิดขึ้นเรียกว่า Bug การทดสอบรวมถึงการแสดงว่าซอฟต์แวร์ทำงานผิดพลาดด้วย จากสถิติจะพบ 10-20 errors ต่อโปรแกรม 1000 บรรทัดโดยเฉลี่ย

วิวัฒนาการของการทดสอบซอฟต์แวร์

- ในยุคแรก ราวก่อนปี ค.ศ. 1956 การทดสอบจะเน้นแค่ค้นหาและแก้ไขข้อผิดพลาดของซอฟต์แวร์เท่านั้น
- ช่วงปี ค.ศ. 1957-1978 เน้นการแสดงว่าซอฟต์แวร์ทำงานได้ตามข้อกำหนดความต้องการ
- ช่วงปี ค.ศ. 1979-1982 เน้นการแสดงว่าซอฟต์แวร์ไม่สามารถทำงานได้ตามข้อกำหนดความต้องการ
- ช่วงปี ค.ศ. 1983-1987 เน้นการประเมินคุณภาพของซอฟต์แวร์
- ตั้งแต่ ปี ค.ศ. 1988 เน้นการป้องกันข้อผิดพลาดที่อาจเกิดขึ้นและการประกันคุณภาพ

(ข้อมูลจากบทความ “The Growth of Software Testing” โดย David Gelperin และ Bill Hetzel Communications of the ACM 1988)

1.2 วัตถุประสงค์และความสำคัญของการทดสอบซอฟต์แวร์

- เพื่อตรวจวัดผลลัพธ์หรือซอฟต์แวร์ว่าตรงกับความต้องการหรือไม่
- เพื่อจัดทำเอกสารแสดงความแตกต่างของผลลัพธ์ที่ควรจะเป็นกับผลลัพธ์ที่เกิดขึ้นจริงในการทดสอบ
- เพื่อช่วยในการแก้ปัญหาความแตกต่างข้างต้นโดยการแนะนำข้อมูลที่เกี่ยวกับข้อผิดพลาดที่เหมาะสม

การทดสอบกระทำเพื่อให้ทราบถึงปัญหาและระดับคุณภาพของซอฟต์แวร์ หากมีกระบวนการทดสอบที่ดี ย่อมมีความมั่นใจว่าจะได้ซอฟต์แวร์ที่ดี ตรงตามความต้องการของผู้ใช้ หากไม่มีการทดสอบหรือการทดสอบที่ไม่มีประสิทธิภาพแล้ว อาจเกิดความเสียหายในการใช้ซอฟต์แวร์นั้นๆ ดังตัวอย่างที่แสดงให้เห็นว่าข้อผิดพลาดในซอฟต์แวร์ก่อให้เกิดผลเสียอย่างไรบ้าง ต่อไปนี้

- กุมภาพันธ์ พ.ศ. 2551 ปัญหาซอฟต์แวร์ของระบบจัดเรียงกระเป๋าในสนามบินแห่งหนึ่ง ทำให้ผู้โดยสารหลายพันคนไม่สามารถเช็คอินได้ทันเวลาเครื่องบินออกเดินทาง มีรายงานว่าปัญหาเกิดในขณะที่มีการปรับปรุงซอฟต์แวร์ให้ดีขึ้น แม้ว่าจะมีการทดสอบก่อนแล้ว ปัญหายังคงมีอยู่อีกหลายเดือน
- ข่าวในเดือนธันวาคม ปี พ.ศ.2550 แจ้งว่ามีปัญหาสำคัญที่ยังคงเกิดขึ้นในระบบเงินเดือนใหม่ของโรงเรียนขนาดใหญ่แห่งหนึ่ง เชื่อว่าพนักงานหนึ่งในสามได้รับเช็คเงินเดือนที่ไม่ถูกต้อง ทำให้ต้องใช้เงินเพิ่มกว่า 40% ในการแก้ไขปัญหานี้และทำให้ระบบย่อยอื่นๆล่าช้าไปด้วย
- ในเดือนพฤศจิกายน พ.ศ. 2550 หน่วยงานภาครัฐ ส่วนภูมิภาคได้รับรายงานว่าซอฟต์แวร์ราคาหลายร้อยล้านเหรียญสหรัฐ ไม่สามารถทำงานตามที่ต้องการได้ จึงฟ้องร้องเรียกค่าเสียหายจากบริษัทผู้ขาย ซึ่งบริษัทผู้ขายต้องไปฟ้องร้องต่อผู้รับช่วงอีกทอดหนึ่ง
- อุปกรณ์ทางการแพทย์หลายหมื่นชิ้นถูกเรียกคืน ในเดือนมีนาคม พ.ศ. 2550 เพื่อแก้ไขข้อผิดพลาดของซอฟต์แวร์ ตามรายงานข่าวแจ้งว่าซอฟต์แวร์ที่ใช้ในการแจ้งเตือนเมื่อกำลังไฟต่ำไม่มีความน่าเชื่อถือพอ ทำให้ต้องเรียกคืนสินค้าทั้งหมด
- ในเดือนกันยายน พ.ศ. 2549 มีข่าวเกี่ยวกับการเลือกตั้งหลักของรัฐบาลสหรัฐ เครื่องคอมพิวเตอร์ที่ใช้ในการแสดงตัวของผู้มีสิทธิ์ออกเสียงเลือกตั้งในบางเขตเกิดขัดข้อง ทำให้ผลการเลือกตั้งต้องล่าช้าออกไป แหล่งข่าวแจ้งว่า สาเหตุเป็นเพราะการทดสอบไม่เพียงพอ
- เดือนสิงหาคม พ.ศ. 2549 รัฐบาลสหรัฐ ฝ่ายให้กู้เงินสำหรับนักเรียน ข้อมูลส่วนตัวของผู้กู้ยืมจำนวน 21,000 คนได้ถูกเปิดเผยทางเว็บไซต์ เนื่องจากความผิดพลาดทางซอฟต์แวร์ ทำให้เกิดความไม่พอใจแก่เจ้าของข้อมูลเป็นอย่างมาก
- เดือน มิถุนายน พ.ศ. 2549 บริษัทยักษ์ใหญ่ด้านการสื่อสาร จัดทำใบเสร็จที่สูงกว่าความเป็นจริงหลายพันเหรียญสหรัฐสำหรับลูกค้า 11,000 ราย การแก้ไขข้อผิดพลาดในการคำนวณใช้เวลาหลายวันในการวิเคราะห์หาสาเหตุ
- เดือนพฤษภาคม พ.ศ.2549 มีข่าวการซื้อซอฟต์แวร์ด้านสุขภาพ มูลค่าหลายร้อยล้านเหรียญสหรัฐ แต่เมื่อใช้งานไปสักระยะหนึ่ง ลูกค้าแจ้งปัญหาที่พบในการใช้ซอฟต์แวร์นั้น เช่น ผลลัพธ์ที่ไม่ถูกต้อง ข้อมูลที่ไม่ถูกต้องที่บุคลากรทางการแพทย์ต้องใช้

- ต้นปี พ.ศ. 2549 ปัญหาของซอฟต์แวร์เกี่ยวกับการเงินของรัฐบาลมีผลทำให้การออกรายงานเพื่อแสดงสถานะการเงินของผู้ลงสมัครแข่งขันเลือกตั้ง สู่สาธารณะ มีความล่าช้า ต้องปิดเว็บไซต์นั้นจนกว่าจะแก้ปัญหาได้
- บัญชีธนาคารหลายล้านบัญชี ของธนาคารใหญ่ในอเมริกาเหนือ มีปัญหาจากการติดตั้งซอฟต์แวร์ที่ทำการทดสอบน้อยเกินไป ทำให้เกิดข้อผิดพลาดในการคำนวณยอดคงเหลือ มีผลต่อความเชื่อถือในธนาคารดังกล่าว และต้องใช้เวลาสองสัปดาห์ในการแก้ไข นอกจากนี้ยังมีปัญหาอื่นๆ เช่น การส่งข้อความถึงลูกค้าโดยอัตโนมัติที่ตั้งค่าไว้ในโปรแกรม รวมแล้วธนาคารต้องใช้เงินมากกว่า 100 ล้านเหรียญสหรัฐในการแก้ไขปัญหาเหล่านี้
- เดือนเมษายน พ.ศ. 2546 บริษัทเงินทุนให้กู้ยืมสำหรับนักเรียน ในอเมริกา คำนวณยอดชำระเงินรายเดือนของการกู้เงิน 800,000 ราย เนื่องจากความผิดพลาดทางซอฟต์แวร์ แม้ว่าผู้กู้จะได้รับการแจ้งให้จ่ายเงินเพิ่มมากกว่าความเป็นจริง บริษัทยังคงยืนยันถึงดอกเบี้ยที่ได้น้อยลงถึง 8 ล้านเหรียญสหรัฐ ความผิดพลาดนี้ถูกค้นพบเมื่อผู้กู้เริ่มรายงานถึงความผิดปกติใน ใบแจ้งที่พวกเขาได้รับ
- เดือนกุมภาพันธ์ พ.ศ. 2546 กระทรวงการคลังของสหรัฐ ส่งเช็คให้แก่ผู้ประกันสังคม 50,000 ฉบับ โดยไม่มีชื่อผู้รับ แหล่งข่าวแจ้งว่าชื่อที่หายไป เกิดจากข้อผิดพลาดในการปรับปรุงซอฟต์แวร์ จึงต้องแก้ไขและส่งเช็คฉบับใหม่ทดแทนฉบับเดิม เพื่อให้ผู้รับสามารถนำเช็คไปขึ้นเงินได้

1.3 ความเชื่อเกี่ยวกับการทดสอบซอฟต์แวร์

- **ความเชื่อที่ 1** การทดสอบซอฟต์แวร์มีราคาแพงเกินไป
- **ความจริง** มีการกล่าวว่าจ่ายน้อยกว่าสำหรับการทดสอบระหว่างการพัฒนา หรือจ่ายมากกว่าในการบำรุงรักษาหรือแก้ไขข้อผิดพลาดในภายหลัง การทดสอบตั้งแต่แรกจะทำให้ประหยัดทั้งเวลาและเงินในหลายๆด้าน การลดต้นทุนโดยไม่ทำการทดสอบ อาจจะทำให้การออกแบบซอฟต์แวร์ที่ไม่เหมาะสมไร้ซึ่งประโยชน์ในการใช้งาน
- **ความเชื่อที่ 2** การทดสอบใช้เวลานาน
- **ความจริง** ในระหว่างการพัฒนาซอฟต์แวร์ การทดสอบไม่ได้ใช้เวลามาก อย่างไรก็ตาม การวินิจฉัยและแก้ไขปัญหาที่พบในระหว่างการทดสอบที่เหมาะสมจะใช้เวลานาน แต่ก็เป็นงานที่สร้างสรรค์และมีประโยชน์
- **ความเชื่อที่ 3** การทดสอบไม่สามารถเริ่มได้หากผลิตภัณฑ์ยังไม่เสร็จสมบูรณ์
- **ความจริง** ไม่มีข้อสงสัยเลยว่าการทดสอบจะขึ้นกับตัวโปรแกรม แต่การรีวิวเอกสารความต้องการและการสร้างกรณีทดสอบ เป็นอิสระจากโปรแกรม อย่างไรก็ดีการพัฒนาแบบวนซ้ำหรือแบบเพิ่มขึ้นทีละส่วน อาจจะช่วยลดความขึ้นแก่กันของการทดสอบกับซอฟต์แวร์ที่พัฒนาแล้ว

- **ความเชื่อที่ 4** การทดสอบที่สมบูรณ์สามารถทำได้
- **ความจริง** เป็นไปได้ที่จะทำการทดสอบทุกเส้นทางโดยทีมทดสอบ แต่ข้อมูลทุกตัวไม่สามารถทดสอบได้หมด อาจจะมีเหตุการณ์บางอย่างที่ไม่เคยถูกทดสอบโดยทีมทดสอบหรือลูกค้าระหว่างพัฒนาซอฟต์แวร์เลย แต่จะเกิดเหตุการณ์นั้นขึ้นเมื่อนำซอฟต์แวร์นั้นไปใช้งานจริงเท่านั้น
- **ความเชื่อที่ 5** ถ้าซอฟต์แวร์ผ่านการทดสอบแล้วแสดงว่าซอฟต์แวร์นั้นไร้ข้อผิดพลาด
- **ความจริง** เป็นความเข้าใจปกติของลูกค้า ผู้จัดการโครงการ และทีมบริหารจัดการที่จะคิดอย่างนั้น ไม่มีใครรับประกันได้ว่าซอฟต์แวร์นั้นจะไม่มีข้อผิดพลาด 100% ถึงแม้ว่าทีมทดสอบจะมีความสามารถในการทดสอบเพียงใดก็ตาม ในความเป็นจริงนั้น ไม่มีซอฟต์แวร์ใด ที่ไม่มีข้อผิดพลาด (เพียงแต่ยังหาไม่พบเท่านั้น) เนื่องจากการที่จะบอกว่าซอฟต์แวร์นั้นไม่มีข้อผิดพลาดเลย ต้องหมายความว่าได้ทดสอบทุกๆ input combination แล้ว ซึ่งเป็นไปไม่ได้ที่จะสามารถทดสอบได้ครบถ้วนทุกกรณี ทั้งทางทฤษฎีและปฏิบัติ ดังนั้นซอฟต์แวร์ที่ผ่านการทดสอบมาอย่างดี เป็นการสร้างความมั่นใจแก่ผู้ใช้งาน ว่าจะไม่มีข้อผิดพลาดร้ายแรงเกิดขึ้นเท่านั้น หรือมีข้อผิดพลาดน้อยที่สุด
- **ความเชื่อที่ 6** ข้อผิดพลาดที่ไม่ถูกค้นพบ ถือเป็นความผิดพลาดของผู้ทดสอบ
- **ความจริง** เป็นการไม่ถูกต้องที่จะโทษผู้ทดสอบหากยังมีข้อผิดพลาดหลงเหลืออยู่ในซอฟต์แวร์อีกแม้ว่าจะทำการทดสอบแล้วก็ตาม ความเชื่อนี้สัมพันธ์กับเงื่อนไขของเวลา เงิน และ การเปลี่ยนแปลงความต้องการ อย่างไรก็ตามวิธีการทดสอบอาจทำให้ผู้ทดสอบพลาดใน การค้นพบข้อผิดพลาดเหล่านั้น
- **ความเชื่อที่ 7** ผู้ทดสอบต้องรับผิดชอบคุณภาพของผลิตภัณฑ์
- **ความจริง** ปกติคนจะเชื่อว่าคุณภาพของซอฟต์แวร์ขึ้นกับทีมทดสอบ ความรับผิดชอบของทีมทดสอบ คือรายงานการพบข้อผิดพลาดให้แก่ผู้มีส่วนร่วม จากนั้นเป็นการตัดสินใจของผู้มีส่วนร่วมว่าจะแก้ไขข้อผิดพลาดหรือจะส่งมอบให้ลูกค้าเลยโดยไม่แก้ไข ซึ่งจะสร้างความกดดันมากกว่าให้ทีมทดสอบ เพราะจะต้องถูกกล่าวหาเรื่องคุณภาพของซอฟต์แวร์
- **ความเชื่อที่ 8** ควรใช้ Test Automation ทุกที่ที่ใช้ได้และเพื่อลดเวลา
- **ความจริง** เป็นความจริงที่ว่า Test Automation ลดเวลาในการทดสอบ แต่เป็นไปไม่ได้ที่จะเริ่ม Test Automation เมื่อใดก็ได้ระหว่างการพัฒนาซอฟต์แวร์ Test Automation ควรจะเริ่มเมื่อได้มีการทดสอบด้วยมือแล้วและไม่เปลี่ยนแปลง เพราะหากมีการเปลี่ยนแปลงความต้องการจะไม่สามารถใช้ Test Automation ได้เลย

- **ความเชื่อที่ 9** ใครๆก็สามารถทดสอบซอฟต์แวร์ได้
- **ความจริง** คนนอกวงการไอที มักจะคิดและเชื่อว่าใครก็สามารถทดสอบซอฟต์แวร์ได้และการทดสอบไม่ใช่งานสร้างสรรค์ จริงๆแล้วความหมายของนักทดสอบซอฟต์แวร์ (Software tester หรือในบางองค์กรอาจเรียกเป็น Quality Assurance Engineers) ไม่ใช่เป็นเพียงแค่คนที่นำข้อมูลเข้า และรอดูผลลัพธ์เท่านั้น หากแต่เป็นผู้ที่มีความรู้ด้านการทดสอบซอฟต์แวร์เป็นอย่างดี สามารถออกแบบกรณีทดสอบได้ครอบคลุมทั้ง valid และ invalid input เรียกว่า effective test cases สามารถวางแผนการทดสอบให้เหมาะสมกับแต่ละระยะเวลาของการพัฒนาซอฟต์แวร์ และเลือกวิธีการทดสอบที่เหมาะสมกับซอฟต์แวร์ที่จะทดสอบ นอกจากนี้ยังต้องเป็นผู้รอบรู้ในหลายศาสตร์ของคอมพิวเตอร์อีกด้วย เพื่อใช้ในการวิเคราะห์ปัญหาที่พบและแนวทางแก้ไข แล้วยังต้องมีความสามารถด้านการสื่อสาร
- **ความเชื่อที่ 10** งานของนักทดสอบคือหาข้อผิดพลาดเท่านั้น
- **ความจริง** การหาข้อผิดพลาดเป็นงานของนักทดสอบแต่ก็เป็นงานของผู้เชี่ยวชาญโดเมนนั้นๆในซอฟต์แวร์แต่ละตัว ผู้พัฒนาต้องรับผิดชอบส่วนของซอฟต์แวร์ที่ได้รับมอบหมาย แต่นักทดสอบจะเข้าใจการทำงานทั้งหมดของซอฟต์แวร์ ความขึ้นแก่กัน และผลกระทบของโมดูลหนึ่งที่มีต่ออีกโมดูลหนึ่ง

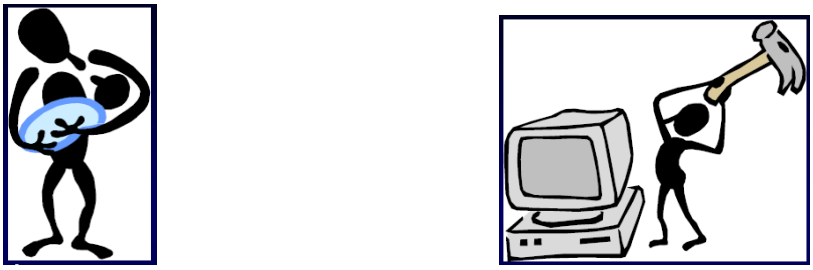
1.4 ข้อจำกัดของการทดสอบซอฟต์แวร์

- เนื่องด้วยเวลาอันจำกัดในการทดสอบ เป็นไปไม่ได้ที่จะมั่นใจว่าจะทดสอบได้ครบทุกด้าน
- ไม่สามารถเชื่อได้ว่าข้อกำหนดซอฟต์แวร์มีความถูกต้อง 100%
- ระบบการทดสอบหรือเครื่องมือในการทดสอบอาจไม่ถูกต้อง
- ไม่มีเครื่องมือการทดสอบใดที่สามารถใช้ได้กับทุกๆซอฟต์แวร์
- วิศวกรทดสอบอาจจะไม่เข้าใจการทำงานของระบบซอฟต์แวร์ที่ทดสอบทั้งหมด
- ทรัพยากรในการทดสอบไม่เพียงพอ
- ไม่มั่นใจได้ว่าทดสอบครบ 100% แล้ว

1.5 ใครเป็นผู้ทดสอบซอฟต์แวร์

- ผู้จัดการทีมทดสอบ มีหน้าที่ในการควบคุม บริหารการทดสอบ ให้คำปรึกษาแก่นักทดสอบ วางแผนการทดสอบ
- นักทดสอบ กำหนดกรณีทดสอบ เขียนข้อกำหนดการทดสอบ และทำการทดสอบ

- กลุ่มผู้ทดสอบอิสระ หมายถึงนักทดสอบที่ไม่ขึ้นกับหน่วยงาน อาจจ้างจากภายนอก การที่ต้องมีกลุ่มนี้เพราะจะไม่มีอคติต่อการพัฒนาระบบ ต้องเน้นที่การตรวจสอบคุณภาพซอฟต์แวร์อย่างแท้จริง
- นักพัฒนาซอฟต์แวร์ จะทดสอบซอฟต์แวร์ในระดับหน่วย (Unit test) และรวมหน่วย (Integration test) เท่านั้น
- กลุ่มผู้ประกันคุณภาพหรือวิศวกรคุณภาพ มีหน้าที่ทดสอบระบบ กำหนดมาตรฐานการทดสอบและกระบวนการควบคุมคุณภาพ



รูปที่ 1.1 ความแตกต่างระหว่างการทดสอบโดยนักพัฒนาซอฟต์แวร์และนักทดสอบอิสระ

จากรูปที่ 1.1 เป็นการแสดงให้เห็นถึงความแตกต่างระหว่างการทดสอบโดยผู้พัฒนาซอฟต์แวร์ (รูปด้านซ้ายมือ) ซึ่งเป็นผู้ที่เข้าใจการทำงานของซอฟต์แวร์เป็นอย่างดี การทดสอบจะค่อนข้างทะนุถนอม ไม่ต้องการให้เกิดข้อผิดพลาด มีข้อจำกัดตรงกำหนดเวลา ส่วนนักทดสอบอิสระ (รูปด้านขวามือ) จะต้องเรียนรู้การทำงานของซอฟต์แวร์อย่างถ่องแท้ จะพยายามทดสอบเพื่อหาข้อผิดพลาดที่จะทำให้ซอฟต์แวร์ไม่ทำงาน เพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพอย่างแท้จริง

1.6 กิจกรรมและกระบวนการทดสอบ

กิจกรรมการทดสอบ ประกอบด้วย

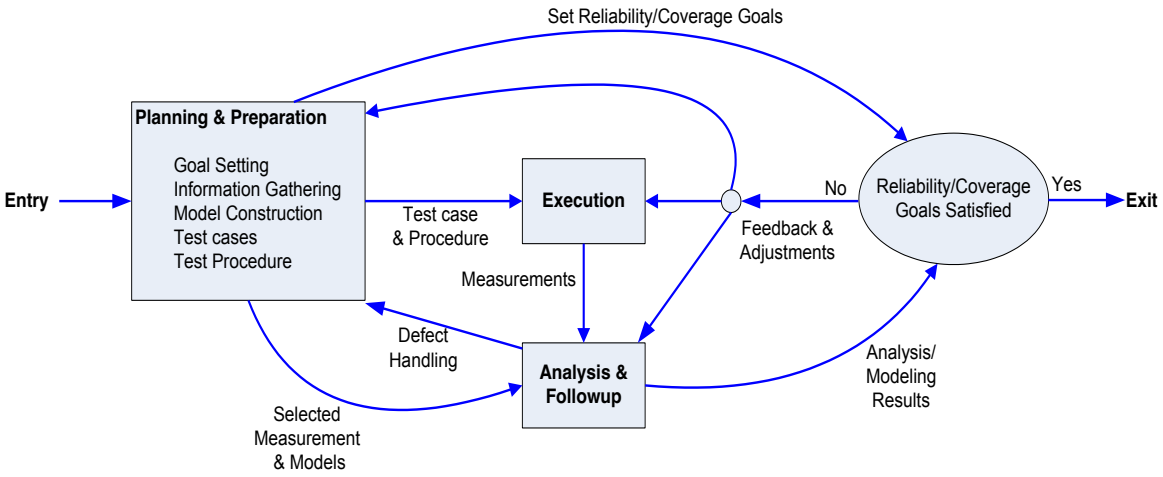
- Test Planning การวางแผนการทดสอบ หมายถึงตารางเวลา กิจกรรมที่ต้องทำ การมอบหมายงานทดสอบ ความต้องการและหัวข้อต่างๆ หลักและวิธีการทดสอบ เครื่องมือที่จะใช้ทดสอบ เป็นต้น
- Test Design and Specification การออกแบบและข้อกำหนด ต้องออกแบบกรณีทดสอบให้ครอบคลุมสิ่งที่ต้องการทดสอบให้มากที่สุด
- Test Set-up ต้องจัดเตรียมเครื่องมือและสภาพแวดล้อมของการทดสอบ ชุดทดสอบให้พร้อม
- Test Operation and Execution ดำเนินการทดสอบด้วยมือหรือเครื่องมืออัตโนมัติ

กระบวนการทดสอบทั่วไป ประกอบด้วย 3 กิจกรรมหลัก คือ

- ***Test planning and preparation*** การวางแผนและเตรียมการทดสอบซอฟต์แวร์ เป็นการกำหนดวัตถุประสงค์ ขอบเขต แนวทาง ขั้นตอนของการทดสอบซอฟต์แวร์ บ่งบอกถึงสิ่งที่ต้องการจะ

ทดสอบและระดับของการทดสอบ การวางแผนควรทำตั้งแต่ช่วงต้นๆ ของการพัฒนาและอาจมีการปรับเปลี่ยนเมื่อเวลาผ่านไป

- ***Test execution and related activities*** การดำเนินการทดสอบและกิจกรรมอื่นๆ ที่เกี่ยวข้องเป็นการลงมือปฏิบัติตามสิ่งที่กำหนดไว้ในขั้นแรก และอาจมีกิจกรรมอื่นๆที่จำเป็น
- ***Analysis and follow-up*** เป็นการวิเคราะห์และติดตามผลการทดสอบ



รูปที่ 1.2 แสดงขั้นตอนทั่วไปในการทดสอบซอฟต์แวร์

(จาก Software Quality Engineering โดย Jeff Tian)

จากรูปที่ 1.2 อธิบายได้ว่า การทดสอบต้องมีการวางแผนและเตรียมการ ประกอบด้วยการกำหนดวัตถุประสงค์ในการทดสอบ การรวบรวมข้อมูลที่ใช้ในการทดสอบ การสร้างโมเดลการออกแบบกรณีทดสอบ การกำหนดขั้นตอนการทดสอบ จากนั้นนำกรณีทดสอบและขั้นตอนการทดสอบไปลงมือปฏิบัติเพื่อให้ได้ผลลัพธ์ ซึ่งคือค่าต่างๆ ที่วัดได้จากการทดสอบ นำไปสู่ขั้นตอนการวิเคราะห์ผลการทดสอบว่ามีสาเหตุจากอะไร และควรดำเนินการแก้ไขต่อไปอย่างไร โดยใคร เป็นต้น พร้อมทั้งติดตามว่าได้มีการแก้ไขตามที่ควรจะเป็นหรือไม่ ในขณะเดียวกันขั้นตอนแรกนั้นอาจมีการกำหนดวัตถุประสงค์เกี่ยวกับความน่าเชื่อถือหรือความครอบคลุมของการทดสอบไว้ด้วย เมื่อทำการทดสอบไปแล้ว หากบรรลุวัตถุประสงค์ที่ตั้งไว้ ก็อาจหยุดการทดสอบได้ หากยังไม่พอใจ สามารถปรับค่าต่างๆ และย้อนกลับไปทำการทดสอบซ้ำอีกเรื่อยๆ จนกว่าจะได้ผลเป็นที่น่าพอใจ

1.7 คำถามที่น่ารู้เกี่ยวกับการทดสอบ

1.7.1 คำถามพื้นฐาน

- *สิ่งที่ถูกทดสอบคือสิ่งใด*

  มักเป็นโปรแกรมซอฟต์แวร์หรือโค้ดที่ถูกเขียนด้วยภาษาคอมพิวเตอร์หลากหลายภาษา

- *จะทดสอบอะไร และข้อผิดพลาดชนิดใด*

  ถ้าต้องการตรวจสอบความต้องการของโปรแกรมด้านไวยากรณ์ (syntax error) หรือด้านตรรกะ คือการเป็นเหตุเป็นผล (logical error) ก็จะต้องพิจารณาวิธีการทดสอบที่เหมาะสม

- *จะทดสอบระดับใด หรือจะหยุดทดสอบเมื่อใด*

  พิจารณาจากความครอบคลุมของการทดสอบว่าสามารถพบข้อผิดพลาดได้มากที่สุดทั้งระดับสูง (design) หรือระดับต่ำ (code) และอาจใช้เป็นเงื่อนไขการหยุดทดสอบได้

1.7.2 คำถามเกี่ยวกับเทคนิคการทดสอบ

- *ใช้เทคนิคการทดสอบแบบไหน*

  ขึ้นอยู่กับคำถามที่ 2 ของหัวข้อ 1.4.1 นั่นคือจะทดสอบอะไร และต้องการค้นหาข้อผิดพลาดชนิดใด ว่าจะเหมาะสม (จะกล่าวรายละเอียดต่อไป) เช่น การใช้ Black-box testing หรือ white-box testing หรือต้องใช้ทั้ง 2 ประเภท เป็นต้น

- *เทคนิคการทดสอบนั้นๆ มีโมเดลอะไรช่วยในการทดสอบ*

  วิธีการทดสอบที่เป็นระบบมักจะมีพื้นฐานบนโมเดลบางอย่างที่สามารถช่วยให้เข้าใจวิธีการทดสอบที่สัมพันธ์กันดีขึ้น

- *มีเทคนิคการทดสอบในโดเมนอื่นๆ ที่จะนำมาปรับใช้กับการทดสอบนี้ได้หรือไม่*

  อาจนำวิธีการทดสอบที่เคยทำในโครงการอื่นที่เป็นงานคนละโดเมนกันมาประยุกต์ใช้กับการทดสอบที่กำลังทำอยู่ได้

- *ถ้ามีเทคนิคการทดสอบให้ใช้ได้หลายเทคนิค สามารถนำเทคนิคเหล่านั้นมารวมหรือประสานงานกันเพื่อเพิ่มประสิทธิภาพและประสิทธิผลได้หรือไม่*

1.7.3 คำถามเกี่ยวกับกิจกรรมและการบริหารจัดการการทดสอบ

- *ใครเป็นผู้ดำเนินการทดสอบ*

  การทดสอบต่างระดับอาจใช้ผู้ทดสอบที่ต่างกัน ในระดับแรกคือการทดสอบในระดับโมดูลหรือยูนิตโดยมากผู้พัฒนาจะเป็นผู้ทดสอบ แต่ในระดับสูงขึ้นไปมักต้องใช้ผู้ทดสอบที่ไม่ใช่ผู้พัฒนาเอง เพื่อป้องกันอคติต่อการทดสอบ ดังที่กล่าวในรูปที่ 1.1

- *เมื่อใดจึงจะทำการทดสอบ*

  หากเป็นการทดสอบซอฟต์แวร์โดยการเอ็กซิคิวท์เพื่อดูผลลัพธ์ (Software testing) มักต้องรอจนกว่าจะเขียนโปรแกรมเสร็จบางส่วนแล้วเริ่มทดสอบ แต่การทดสอบโดยพิจารณาจากเอกสาร (Software inspection) เพื่อดูแนวโน้มที่จะเกิดข้อผิดพลาดสามารถกระทำได้ตั้งแต่เริ่มการพัฒนา เอกสารที่สามารถตรวจสอบได้ ได้แก่ สัญญาการพัฒนาซอฟต์แวร์ เอกสาร

ความต้องการ เอกสารการออกแบบระบบ โปรแกรม แผนการทดสอบ กรณีทดสอบ เป็นต้น ส่วนแผนของการทดสอบควรทำตั้งแต่เริ่มต้นโครงการ

- *เมื่อทำการทดสอบเสร็จ จะต้องทำอะไรต่อไป*

  จากรูป 1.2 จะเห็นว่ามีการวิเคราะห์และติดตามผล

- *เป็นไปได้หรือไม่ที่จะทดสอบแบบอัตโนมัติ*

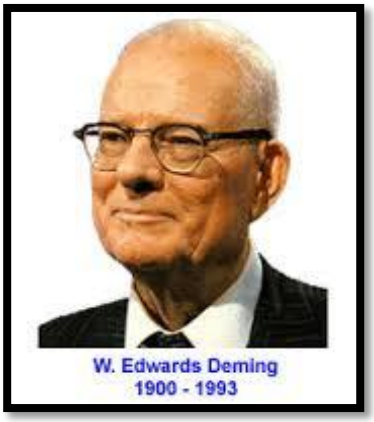
  หากทำได้จะพิจารณาเครื่องมือช่วยทดสอบแบบอัตโนมัติที่สามารถใช้ได้อย่างเหมาะสม

- *เครื่องมือใดบ้างที่ช่วยบริหารจัดการกระบวนการทดสอบและกิจกรรมที่เกี่ยวข้อง*
- *อะไรคือความสัมพันธ์ระหว่างการทดสอบและหลักการของข้อผิดพลาดที่เกี่ยวข้อง*
- *อะไรคือ ฮาร์ดแวร์/ซอฟต์แวร์/สภาพแวดล้อมของการทดสอบ*
- *อะไรคือประเภทของผลิตภัณฑ์หรือส่วนแบ่งของตลาดสำหรับผลิตภัณฑ์ที่ทดสอบอยู่*

1.8 ปรมาจารย์ด้านคุณภาพ

นักวิจัยผู้สนใจการพัฒนาคุณภาพของงานได้มีการศึกษาเกี่ยวกับคุณภาพมาเป็นเวลานาน ขอยกตัวอย่างผู้เชี่ยวชาญด้านคุณภาพของอุตสาหกรรมการผลิต (โดยเฉพาะรถยนต์ที่มีการนำไปปฏิบัติอย่างแพร่หลายในหลายๆประเทศทั่วโลก)

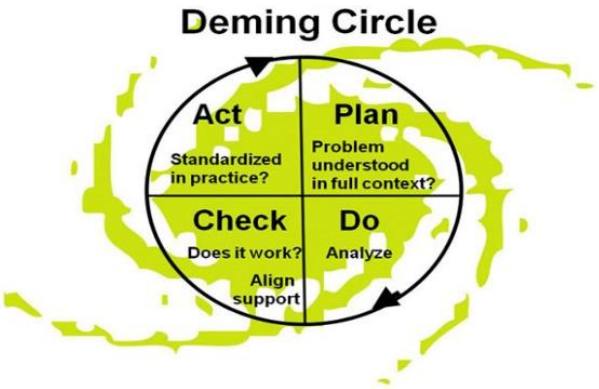
Dr. William E. Deming (1900 – 1993)



เป็นศาสตราจารย์ด้านสถิติ มหาวิทยาลัยนิวยอร์ก

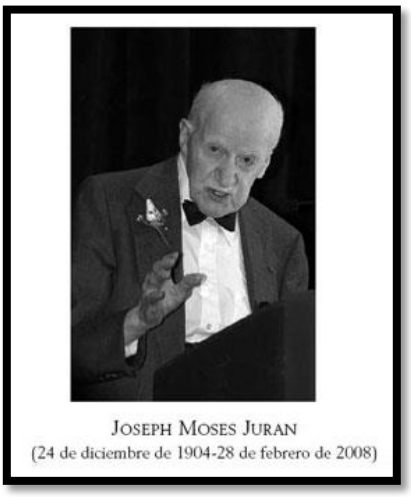
ได้รับเชิญจากรัฐบาลญี่ปุ่นเพื่อช่วยปรับปรุงคุณภาพและประสิทธิภาพการผลิต

เมื่ออุตสาหกรรมรถยนต์ฟอร์ดนำวิธีการของเดมิงไปใช้ ทำให้ประสิทธิภาพและยอดขายเพิ่มขึ้นอย่างมากเดมิงได้กำหนดขั้นตอนการบริหารคุณภาพ 4 ขั้นตอน ปรัชญาและงานสอนของเดมิง สรุปได้ 14 ข้อ



"The Father of Quality Evolution"

Dr. Joshep M. Juran (1904 – 2008)



เป็นปรมาจารย์ทางด้านคุณภาพชาวอเมริกันที่มีชื่อเสียงและมีวิสัยทัศน์ที่กว้างไกล เป็นหนึ่งในทีมของเดมิง

แนวคิดของจูรานกับเดมิงมีส่วนที่คล้ายกันและแตกต่างกัน ในส่วนที่คล้ายกันทั้งสองเห็นความสำคัญของบทบาทของผู้บริหารระดับสูงในการจัดการคุณภาพทั้งองค์กรและตระหนักถึงปัญหาคุณภาพในการผลิต ทั้งสองยังเห็นความสำคัญของลูกค้าภายในและภายนอกองค์กร เห็นความสำคัญของการปรับปรุงคุณภาพอย่างต่อเนื่อง และความจำเป็นในการฝึกอบรม รวมทั้งเทคนิคและเครื่องมือในการปรับปรุงคุณภาพต่างๆ

ส่วนความแตกต่างนั้น ประเด็นใหญ่อยู่ที่เดมิงเน้นกระบวนการ (process) มากกว่า ในขณะที่จูรานสนใจที่ผลผลิต (output)

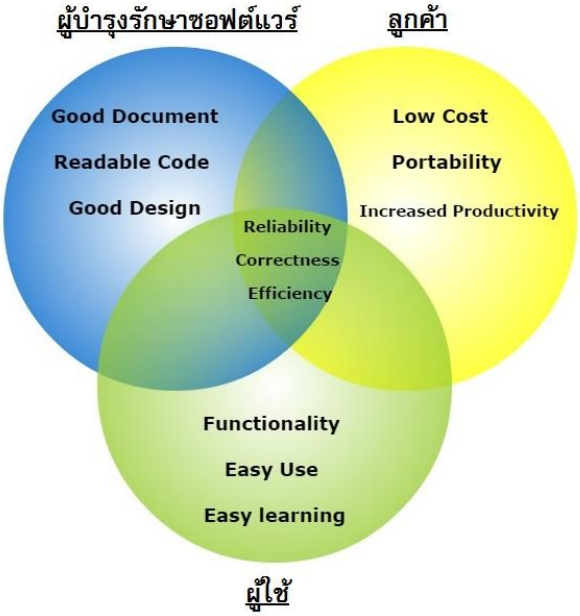
สามเหลี่ยมของจูรัน

Philip B. Crosby (1926 – 2001)

เป็นนักปฏิบัติและนักคิดด้านการจัดการคุณภาพที่มีชื่อเสียงชาวอเมริกัน
ผู้ให้ความหมาย “คุณภาพ” คือ การทำได้ตามข้อกำหนด
(Conformance to Requirements)
ครอสบีให้ความสำคัญกับ
การดำเนินงานที่ข้อบกพร่องเป็นศูนย์ (Zero Defect)
การทำตามมาตรฐาน (Conformance to Standards)
ทำให้ถูกแต่แรก (Do It Right The First Time)

1.9 คุณภาพของซอฟต์แวร์

คุณภาพของซอฟต์แวร์มองได้หลายมิติ จากรูป 1.3 จะแสดงถึงคุณภาพของซอฟต์แวร์จากมุมมองของผู้ที่มีส่วนเกี่ยวข้อง ได้แก่ ผู้ใช้งาน (user) ลูกค้า (customer) และผู้ดูแลรักษาระบบ (maintainer)



รูปที่ 1.3 คุณภาพของซอฟต์แวร์จากมุมมองของผู้ที่เกี่ยวข้อง

- **ผู้บำรุงรักษาซอฟต์แวร์** มองว่าคุณภาพ หมายถึงเอกสารประกอบทุกประเภทต้องอ่านเข้าใจง่าย โปรแกรมอ่านง่าย มีหมายเหตุ อธิบาย และซอฟต์แวร์มีการออกแบบที่ดี เพื่อที่เวลาแก้ไขโปรแกรมจะได้สามารถศึกษาจากเอกสารได้ ช่วยให้การแก้ไขหรือปรับปรุงทำได้ง่าย
- **ผู้ใช้** มองคุณภาพซอฟต์แวร์ตรงที่ทำงานง่าย ใช้งานได้ง่าย เรียนรู้ได้ง่าย นั่นคือใช้เวลาพอสมควรในการเรียนรู้การใช้งาน
- **ลูกค้า** (ผู้ที่จ้างพัฒนาซอฟต์แวร์ อาจใช้หรือไม่ได้ใช้ซอฟต์แวร์ก็ได้ เช่น ผู้บริหาร) มองคุณภาพของซอฟต์แวร์ ด้านการลงทุนต่ำ สามารถนำไปใช้ในหลากหลายระบบและเพิ่มผลผลิต

แต่มุมมองของกลุ่มคนเหล่านี้จะมีจุดที่เหมือนกันคือ ซอฟต์แวร์นั้นต้องมีความน่าเชื่อถือ ทำงานได้ตรงตามที่ต้องการทำได้ถูกต้องและมีประสิทธิภาพ

ซอฟต์แวร์ที่มีคุณภาพต่ำจะก่อให้เกิดต้นทุนต่าง ๆ ดังนี้

- เพิ่มเวลาในการหาและแก้ไขปัญหา การมีข้อผิดพลาดเยอะต้องใช้เวลามาก ทำให้เสียเวลา
- เพิ่มค่าใช้จ่ายในการแก้ไขแบบกระจาย เพราะส่วนใหญ่เราใช้ซอฟต์แวร์ผ่านเครือข่าย เนื่องจากเป็นระบบแบบกระจาย เช่น บริษัทนั้นมีสาขากระจายอยู่ทั่วประเทศ ถ้าซอฟต์แวร์มีปัญหาก็ต้องตามไปแก้ไขทุกสาขา

- เพิ่มค่าใช้จ่ายของฝ่ายดูแลลูกค้า (customer support) ยิ่งซอฟต์แวร์มีปัญหามาก ผู้ใช้ก็ต้องติดต่อเข้ามายังฝ่ายดูแลลูกค้ามาก ซึ่งจะทำให้ฝ่ายดูแลลูกค้าต้องทำงานหนัก
- เป็นข้อบกพร่องของซอฟต์แวร์
- ซอฟต์แวร์นั้นไม่ประสบความสำเร็จในตลาด ชื่อเสียงของบริษัทก็ไม่ดี บริษัทก็ขาดทุน

1.9.1 การประกันคุณภาพซอฟต์แวร์

Quality Control การควบคุมคุณภาพซอฟต์แวร์

- นิยามของกระบวนการและมาตรฐานต่างๆ
- การประชุมด้านเทคนิคอย่างเป็นทางการ
- มีการวางแผนการทดสอบและทบทวน

Quality Assurance การประกันคุณภาพซอฟต์แวร์

- การวิเคราะห์และรายงาน
- การวัดค่าต่างๆ เช่น วัดความเร็ว ปริมาณข้อมูล วัดประสิทธิภาพ เพื่อให้รู้ว่าซอฟต์แวร์ของเราอยู่ในระดับที่ต้องการหรือไม่

1.9.2 Verification & Validation

Software Testing มีหลักการทดสอบที่เรียกว่า “Verification and Validation” โดยมีลักษณะดังนี้คือ

- **Verification** คือ กระบวนการตรวจสอบว่า ซอฟต์แวร์หรือ component ที่จะออกมานั้นเป็นไปตามข้อกำหนด (Specification) หรือไม่ ซึ่งจะทำในช่วงก่อนการเริ่มทำการพัฒนาซอฟต์แวร์ โดยทำ Verification นี้จะประกอบด้วยการพิจารณาจากเอกสาร โดยวิธี Technical Reviews, Inspection, Desk checking เอกสารที่นำมาพิจารณา เช่น สัญญา เอกสารความต้องการ เอกสารการออกแบบ โปรแกรม แผนการทดสอบ เป็นต้น
- **Validation** คือ กระบวนการที่จะตรวจสอบว่า ซอฟต์แวร์หรือ component ที่ออกมานั้นเป็นไปตามความต้องการใช้งานของผู้ใช้หรือไม่ ซึ่งจะทำในตอนสิ้นสุดของกระบวนการพัฒนา

1.10 ทัศนคติที่จะทำให้เป็นนักทดสอบระบบที่ดี

- เป็นอิสระจากคนเขียนโปรแกรม เพราะปกติคนเขียนโปรแกรมไม่ต้องการให้เกิดข้อผิดพลาดขึ้น ถ้าเราทำตามที่คนเขียนโปรแกรมบอก อาจไม่เจอข้อผิดพลาดได้
- มองโปรแกรมในมุมมองของผู้ใช้ เพราะว่าคนใช้งานซอฟต์แวร์ คือ customer ไม่ใช่คนพัฒนา tester จึงต้องมองว่าถ้าเป็นคนใช้จริงๆ เราต้องการอะไรบ้าง ไม่ใช่มองแค่ผ่านๆ

- ต้องรู้วัตถุประสงค์ของการทดสอบ เพื่อทดสอบให้ซอฟต์แวร์นั้นทำงานได้ตามที่กำหนดไว้ หรือตามที่ลูกค้าต้องการ
- ต้องทดสอบให้ซอฟต์แวร์เกิดข้อผิดพลาดที่ไม่ต้องการให้เกิด พยายามให้ซอฟต์แวร์ทำงานไม่ได้ เพื่อให้รู้ว่าซอฟต์แวร์ทำในสิ่งที่ไม่ควรทำหรือไม่
- นักทดสอบที่ดีควรมีความรู้ในหลายๆด้าน เพราะหลังการทดสอบต้องมีการวิเคราะห์หาสาเหตุที่แท้จริงของข้อผิดพลาด เช่น บางทีการที่ซอฟต์แวร์ไม่ทำงานอาจเกิดจากความผิดพลาดของระบบเครือข่ายก็ได้
- ควรมีประสบการณ์ในการวิเคราะห์และแก้ไขปัญหา

แบบฝึกหัดท้ายบทที่ 1

1. การทดสอบซอฟต์แวร์สำคัญต่อการพัฒนาซอฟต์แวร์อย่างไร
2. ผู้ทดสอบควรมีหลักการในการทดสอบอย่างไร เพื่อให้ประสบความสำเร็จในการทดสอบ
3. การเลือกเทคนิควิธีการในการทดสอบพิจารณาจากสิ่งใด
4. เมื่อทำการทดสอบตามแผนที่วางไว้แล้ว เหตุใดจึงต้องทำการวิเคราะห์ผลจากการทดสอบ
5. นักทดสอบซอฟต์แวร์ควรมีความรู้ด้านใดบ้าง
6. ลักษณะนิสัยที่นักทดสอบซอฟต์แวร์ควรมี คือสิ่งใด
7. ถ้าผู้ทดสอบเป็นคนเดียวกับผู้เขียนโปรแกรมจะมีข้อดีข้อเสียอย่างไร
8. นิสิตเห็นด้วยหรือไม่กับคำกล่าวที่ว่าการทดสอบเป็นเรื่องง่าย ไม่จำเป็นต้องมีความรู้เฉพาะทางก็สามารถทำได้ พร้อมเหตุผลสนับสนุน
9. การประกันคุณภาพและควบคุมคุณภาพของซอฟต์แวร์มีผลอย่างไรต่อการทดสอบซอฟต์แวร์
10. จงอธิบายสาเหตุที่การทดสอบต้องมีทั้ง Verification และ Validation
11. การทดสอบจะกระทำไปจนถึงจุดใดจึงจะหยุดได้
12. จงเปรียบเทียบงานของนักพัฒนาซอฟต์แวร์กับนักทดสอบซอฟต์แวร์