รายงานการเข้าอบรม

เรื่อง กลยุทธ์การทดสอบซอฟต์แวร์

จัดโดย

เขตอุตสาหกรรมซอฟต์แวร์ประเทศไทย (Software Park)
สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.)

ณ บริษัท สยามชำนาญกิจ จำกัด
อาคารพร้อมพันธุ์ 3 ถนนลาดพร้าว เขตจตุจักร กรุงเทพฯ

วันที่ 11 มิถุนายน พ.ศ. 2562

ผู้จัดทำ
รองศาสตราจารย์ทัศนีย์วรรณ์ ศรีประดิษฐ์

ได้รับทุนสนับสนุนจากกองทุนพัฒนาบุคลากรมหาวิทยาลัยสุโขทัยธรรมาธิราช
ประจำปีงบประมาณ 2562

รายงานการไปฝึกอบรม ดูงาน ประชุม / สัมมนา

ตามระเบียบมหาวิทยาลัยสุโขทัยธรรมาธิราช ว่าด้วยการให้ทุนฝึกอบรม ดูงาน

และประชุมทางวิชาการแก่ข้าราชการมหาวิทยาลัยสุโขทัยธรรมาธิราช

---

1. ผู้เข้าร่วมการฝึกอบรม

ชื่อ นางสาวทัศนีย์วรรณ์ ศรีประดิษฐ์ ตำแหน่งรองศาสตราจารย์ ระดับ 9

สังกัด สาขาวิชาวิทยาศาสตร์และเทคโนโลยี โทร. 8290

ไปเข้าร่วม ฝึกอบรม เรื่อง “กลยุทธ์การทดสอบซอฟต์แวร์” ณ บริษัท สยามชำนาญกิจ จำกัด อาคารพร้อมพันธุ์ ถนนลาดพร้าว เขตจตุจักร กรุงเทพฯ

จัดโดย เขตอุตสาหกรรมซอฟต์แวร์ประเทศไทย (Software Park) สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ (สวทช.)

ตั้งแต่วันที่ 11 มิถุนายน 2562 ถึงวันที่ 11 มิถุนายน 2562 รวมระยะเวลา 1 วัน

2. รายละเอียดเกี่ยวกับการฝึกอบรม

2.1 วัตถุประสงค์ของการเข้ารับการอบรม

1) เพื่อให้ได้รับความรู้และทักษะเกี่ยวกับกลยุทธ์การทดสอบซอฟต์แวร์อย่างมีคุณภาพ จากวิทยากรผู้เชี่ยวชาญที่มีประสบการณ์ในเรื่องนี้โดยตรง

2) เพื่อแลกเปลี่ยนความรู้และประสบการณ์ในการทดสอบซอฟต์แวร์ระหว่างผู้เข้ารับการอบรม

2.2 รูปแบบของการฝึกอบรม

แบ่งออกเป็น 2 ส่วน ได้แก่

1) วิทยากรบรรยายให้ได้รับความรู้ความเข้าใจด้านทฤษฎีและแนวทางการปฏิบัติงานที่เกี่ยวข้องกับการทดสอบซอฟต์แวร์

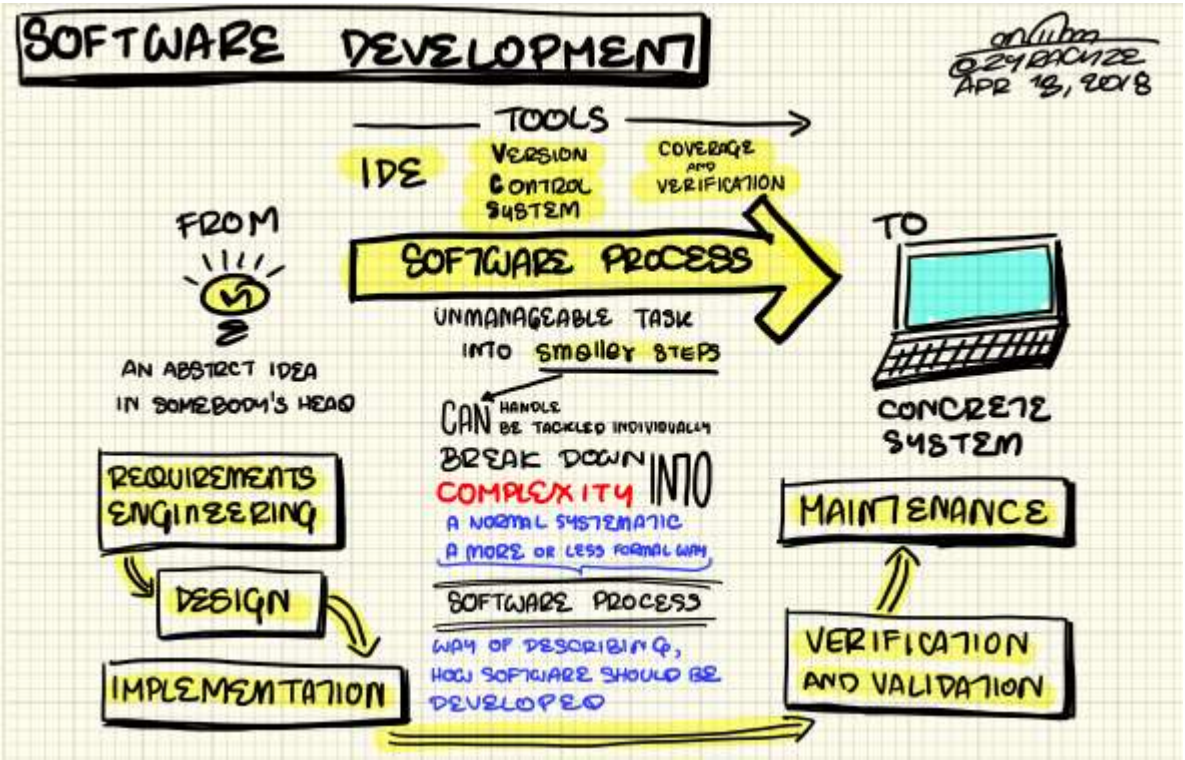
2) ฝึกทักษะและทดสอบความรู้ความเข้าใจของผู้เข้ารับการอบรม

วิทยากรบรรยาย

คุณประธาน ด่านสกุลเจริญกิจ

2.3 สาระสำคัญของการฝึกอบรม

กระบวนการพัฒนาซอฟต์แวร์ (software development process) หมายถึง กิจกรรม วิธีการ การปฏิบัติ และการเปลี่ยนแปลงที่ใช้ในการพัฒนาซอฟต์แวร์ ตลอดจนผลิตภัณฑ์ที่เกี่ยวข้อง



กระบวนการพัฒนาซอฟต์แวร์ ประกอบด้วยกิจกรรมพื้นฐาน 4 กิจกรรม ได้แก่

1. การกำหนดคุณสมบัติซอฟต์แวร์ (Software Specification)
2. การออกแบบและสร้างซอฟต์แวร์ (Software Design and Implementation)
3. การทวนสอบซอฟต์แวร์ (Software Validation)
4. การประเมินซอฟต์แวร์ (Software Evolution)

1. การกำหนดคุณสมบัติซอฟต์แวร์ เพื่อนิยามหน้าที่ต่างๆ ที่ต้องมีในซอฟต์แวร์ และระบุข้อจำกัดต่างๆ ที่เกี่ยวข้องกับกระบวนพัฒนาซอฟต์แวร์

2. การออกแบบและสร้างซอฟต์แวร์ เพื่อสร้าง และพัฒนาซอฟต์แวร์ให้ตรงกับข้อกำหนด

3. การทวนสอบซอฟต์แวร์ เพื่อวิเคราะห์และตรวจสอบการทำงานของซอฟต์แวร์ โดยดูภาพรวมของการทำงานว่ามีการตอบสนองความต้องการทั้งในส่วนของฟังก์ชันและประสิทธิภาพการทำงาน ว่าสอดคล้องกับลักษณะของความต้องการของซอฟต์แวร์หรือไม่ ซึ่งมักจะใช้การทดสอบแบบ functional testing (black box testing)

4. การประเมินซอฟต์แวร์ เพื่อเตรียมการบางอย่างเพื่อจัดการกับเหตุการณ์ที่คาดหมายว่าจะเกิดขึ้นในอนาคต เช่น เมื่อซอฟต์แวร์ใช้งานได้ระยะหนึ่งแล้ว ผู้ใช้หรือลูกค้าอาจมีความต้องการเพิ่มเติมหรือเปลี่ยนแปลงความต้องการบางอย่าง

การทดสอบซอฟต์แวร์ แบ่งออกเป็น 2 ประเภท ได้แก่

1. การทดสอบส่วนของฟังก์ชันงาน (Functional Testing) เป็นการพิสูจน์ความจริงในแต่ละฟังก์ชันงานว่าทำงานสอดคล้องกับข้อกำหนดความต้องการของระบบงานหรือไม่ โดยไม่สนใจว่าการทำงานภายในเป็นอย่างไร สนใจค่าที่ input เข้าไป กับ ผลลัพธ์ที่ออกมา เช่น การทำงานของเครื่องจักร โดยแค่ป้อนคำสั่งให้เครื่องจักร แล้วดูว่าเครื่องจักรทำงานถูกต้องหรือไม่ โดยที่ไม่สนใจกระบวนการทำงานว่าข้างในทำอะไรบ้าง เรียกอีกอย่างว่า Black box testing เป็นการทดสอบที่มองทั้งระบบเป็นเหมือนกล่องดำ (black box) ประกอบด้วย

1.1 การทดสอบความสามารถการเข้าใช้งาน (accessibility testing) ช่วยในการกำหนดว่าซอฟต์แวร์ดังกล่าวสามารถนำไปใช้งานได้จริงหรือไม่ โดยกรรมวิธีการทดสอบจะทดลองกับการทำงานจริง

1.2 การทดสอบในจุดเริ่ม (alpha testing) คือการทดสอบโดยจำลองการทำงาน/การปฏิบัติงานจริง โดยผู้ใช้งานหรือลูกค้าบนเครื่อง/ไซต์งานของนักพัฒนา การทดสอบนี้ถูกนำมาใช้ก่อนที่จะนำซอฟต์แวร์เข้าสู่ขั้นการทดสอบในชั้นถัดมา (beta)

1.3 การทดสอบในชั้นถัดมา (beta testing) คือการทดสอบในขั้นตอนสุดท้าย ก่อนที่จะส่งมอบซอฟต์แวร์ออกใช้งานตามวัตถุประสงค์เชิงพาณิชย์ ปกติแล้วผู้ดำเนินการทดสอบจะเป็นผู้ใช้งานจริงหรือลูกค้าเจ้าของระบบ

1.4 การทดสอบภัยการทำลาย (destructive testing) มีจุดมุ่งหมายในการหาประเด็นข้อผิดพลาดของซอฟต์แวร์ เพื่อทำความเข้าใจกับโครงสร้างของระบบและกำหนดตัวอย่างการทดสอบด้วยการจงใจใส่รายการผิดพลาดทั้งข้อมูลนำเข้า รูปแบบ จำนวน ฯลฯ โดยอาศัยทีมงานทดสอบ

1.5 การทดสอบแบบสโมค (smoke testing) เพื่อพิจารณาว่า “สิ่งใหม่” ที่เพิ่มเข้ามาจากทีมพัฒนามีความเสถียรและคงทนเพียงพอหรือไม่ ดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

1.6 การทดสอบสภาวะปกติ (sanity testing) เพื่อประเมินอย่างรวดเร็วในส่วนของซอฟต์แวร์ สภาพแวดล้อมเครือข่าย ระบบภายนอก ว่ายังทำงานตามปกติหรือไม่ ดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

1.7 การทดสอบการเสื่อมถอย (regression testing) ใช้สำหรับสืบค้นหาข้อผิดพลาดในซอฟต์แวร์ซึ่งไม่ครอบคลุม หลังจากได้ปรับปรุงโปรแกรมจนเสร็จแล้ว (เช่น การซ่อมแก้ไข จุดด่าง หรือบั๊ก ในฟังก์ชันการทำงาน) โดยการทดสอบโปรแกรมซ้ำอีกครั้ง ดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

2. การทดสอบส่วนที่ไม่ใช่ฟังก์ชันงาน (Non-Functional Testing) เป็นการประเมินความพร้อมของระบบ ซึ่งขึ้นอยู่กับเกณฑ์หลายๆ ด้าน การทดสอบนี้ช่วยให้สามารถทำการวัดและเปรียบเทียบได้ ประกอบด้วย

2.1 การทดสอบความเข้ากันได้ (compatibility testing) เพื่อตรวจสอบว่าซอฟต์แวร์สามารถทำงานบนฮาร์ดแวร์ ระบบปฏิบัติการ ฐานข้อมูล เว็บเซิร์ฟเวอร์ แอปพลิเคชันเซิร์ฟเวอร์ สภาพแวดล้อมที่แตกต่างกันได้อย่างราบรื่นหรือไม่ การทดสอบประเภทนี้ดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

2.2 การทดสอบการติดตั้ง (installation testing) เพื่อทำให้มั่นใจว่าระบบได้รับการติดตั้งอย่างถูกต้อง และทำงานได้จริง การทดสอบนี้ดำเนินการทดสอบโดยวิศวกรด้านการทดสอบซอฟต์แวร์ หรือเจ้าหน้าที่ที่เชี่ยวชาญ

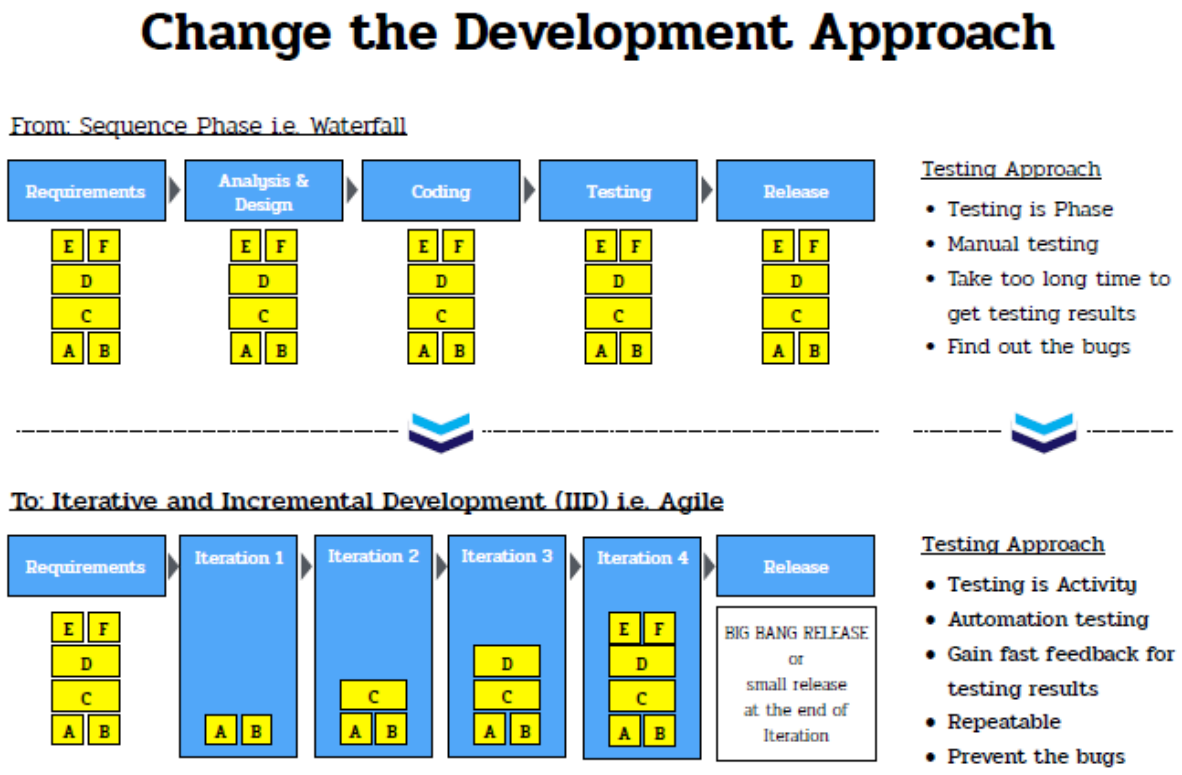
2.3 การทดสอบความเป็นสากล (internationalization testing) วิธีนี้ใช้ตรวจสอบดูขอบเขตของซอฟต์แวร์ว่าสนับสนุนการทำงานในระดับสากลหรือไม่ เช่น การใช้ภาษาที่แตกต่างกัน การใช้ชุดตัวอักษรที่ต่างกัน เป็นต้น ซึ่งดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

2.4 การทดสอบแบบจำกัดเฉพาะส่วน (localization testing) เพื่อปรับแอปพลิเคชันที่ใช้กันทั่วโลก ให้เหมาะกับวัฒนธรรมและท้องถิ่นใดโดยเฉพาะ ซึ่งดำเนินการทดสอบโดยทีมงานด้านการทดสอบ

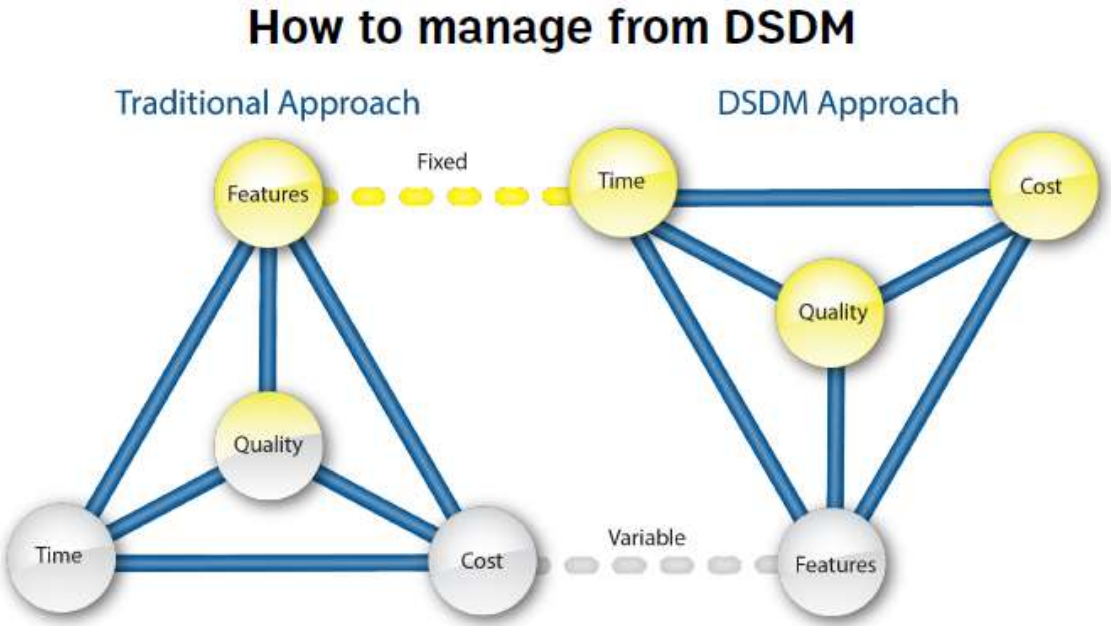
2.5 การทดสอบสมรรถนะการทำงาน (performance testing) จะทำการทดสอบคุณลักษณะด้านคุณภาพของซอฟต์แวร์ เช่น การคงสภาพ ความวางใจได้ และความพร้อมใช้งาน ซึ่งดำเนินการทดสอบโดยทีมวิศวกรซอฟต์แวร์หรือผู้เชี่ยวชาญ

2.6 การทดสอบด้านความปลอดภัย (security testing) เพื่อสร้างความมั่นใจว่า จะมีความปลอดภัยจากการดำเนินการทั้งภายในและภายนอก ทั้งโปรแกรมไม่พึงประสงค์ หรือจากคนมุ่งร้าย ซึ่งดำเนินการทดสอบโดยทีมงานด้านการทดสอบหรือหน่วยงานผู้เชี่ยวชาญด้านความปลอดภัย

2.7 การทดสอบความสามารถใช้งานได้ (usability testing) เพื่อทำความเข้าใจถึงวิธีการทำงานที่เป็นมิตรกับผู้ใช้ซอฟต์แวร์ ซึ่งดำเนินการทดสอบโดยผู้ใช้งานตัวจริง

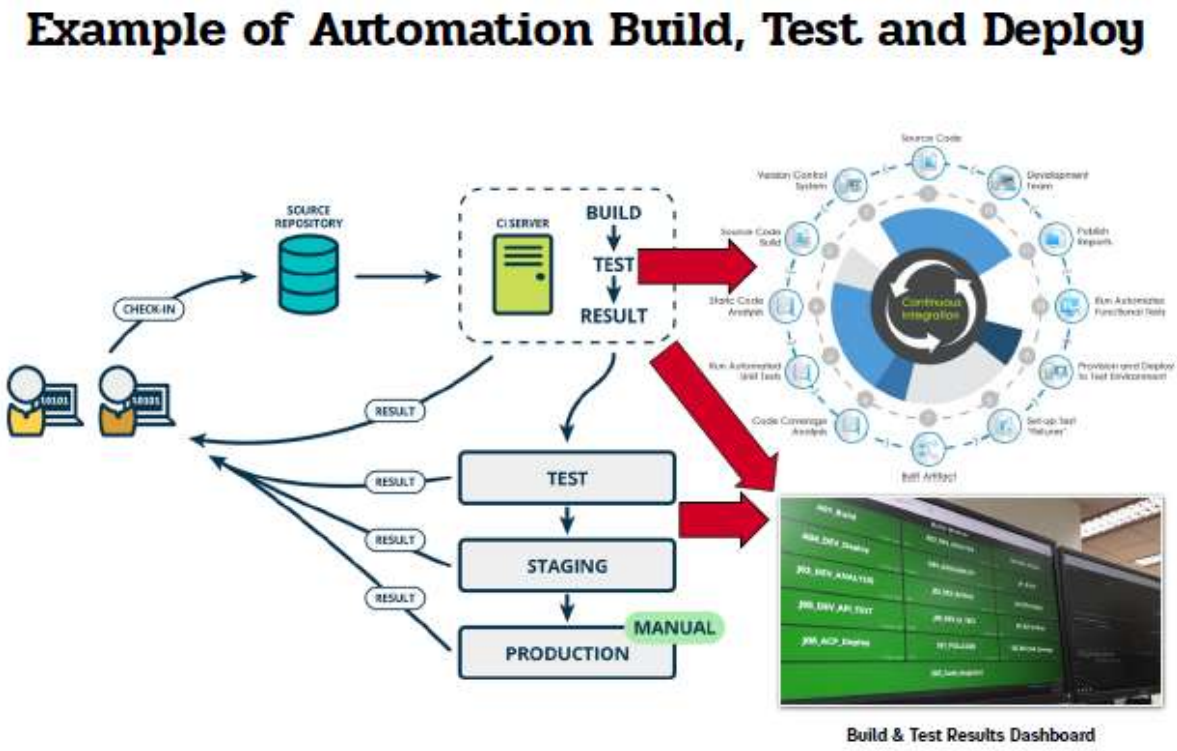


ในกระบวนการพัฒนาระบบแบบตามลำดับขั้น เช่น แบบน้ำตก (waterfall) นั้น กิจกรรมที่ต้องดำเนินการในทุกๆ ขั้นตอนจะมีปริมาณเท่ากันทุกขั้นตอน แต่หากเป็นการพัฒนาระบบแบบวนซ้ำหรือเพิ่มขึ้น เช่น อไจล์ (Agile) กิจกรรมที่ต้องทำการทดสอบแต่ละรอบจะไม่เท่ากัน ในการทดสอบรอบที่ 1 อาจทดสอบการทำงานเพียงบางส่วนเท่านั้น และปริมาณงานที่ต้องทำการทดสอบจะเพิ่มขึ้นในการทดสอบรอบต่อๆ ไป จนกระทั่งได้งานครบและผ่านการทดสอบครบทุกส่วนในรอบสุดท้าย



การบริหารจัดการโครงการด้วย Dynamic Software Development Methodology (DSDM) ซึ่งเป็นอีกหนึ่งวิธีการที่อยู่ภายใต้กระบวนการพัฒนาระบบแบบ Agile องค์ประกอบพื้นฐานของการบริหารจัดการโครงการ ได้แก่ ขอบเขตของงาน (Scope) หรือฟีเจอร์ (Feature) ระยะเวลาดำเนินการโครงการ (Time) ต้นทุน (Cost) และคุณภาพ (Quality) ซึ่งเวลาและงบประมาณจะคงที่ไม่เปลี่ยนแปลง แต่ขอบเขตของงานโดยทั่วไปส่วนใหญ่จะมีการปรับเปลี่ยนเสมอ และต้องดูแลคุณภาพของระบบให้สม่ำเสมอด้วย

Scrum Framework เป็นเครื่องมือในการดำเนินโครงการพัฒนา ทดสอบและส่งมอบซอฟต์แวร์เป็นรอบสั้นๆ ที่มีระยะเวลาหรือรอบการทำงานคงที่ เรียกว่า Sprint ดังนั้นจะต้องมีการกำหนดแผนการดำเนินงานของโครงการร่วมกับทีมลูกค้าและสื่อสารออกไปให้ทุกๆ คนที่เกี่ยวข้องรับทราบร่วมกัน

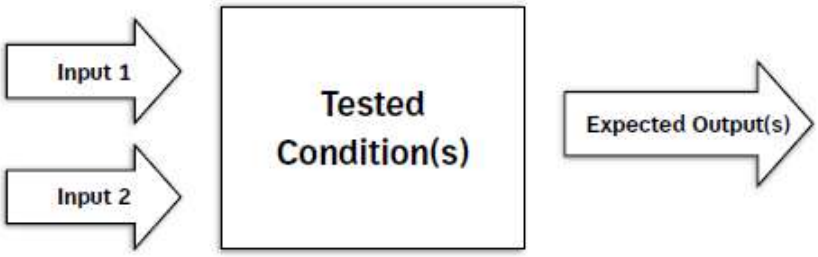


การทดสอบระบบแบบ Manual test เป็นวิธีการทดสอบแบบดั้งเดิม ซึ่งง่ายและใช้เวลาน้อย แต่เมื่อระบบเริ่มโตขึ้นเรื่อยๆ เวลาและความยากสำหรับการทดสอบแบบ manual ก็จะเพิ่งขึ้นเรื่อยๆ จนกระทั่งการทดสอบทำได้ยาก และใช้เวลามากขึ้น ทำให้การตรวจหาข้อบกพร่องของซอฟต์แวร์ทำได้ยากขึ้น การทดสอบแบบ manual จึงไม่ใช่แนวทางที่ยั่งยืน จำเป็นต้องหาวิธีการอื่นมาช่วยในการทดสอบ นั่นคือ การนำ Automated test มาใช้ร่วมกับ Manual test

Automation Test คือ การทดสอบโดยใช้เครื่องมือ (tool) ช่วยในการทดสอบ โดยกำหนดให้ tool ทำงานด้วยตัวของมันเอง ซึ่งจะช่วยลดเวลาของการทำ Manual Test แต่อาจจะต้องมีการใช้ Coding เพื่อเขียน Script ที่ใช้สำหรับ Run

Test Case/Test Scenario

*"Test case/Test Scenario - **input(s)** to test **the system** and the **expected output(s)** from these inputs if the system operates correctly under specified execution conditions"*

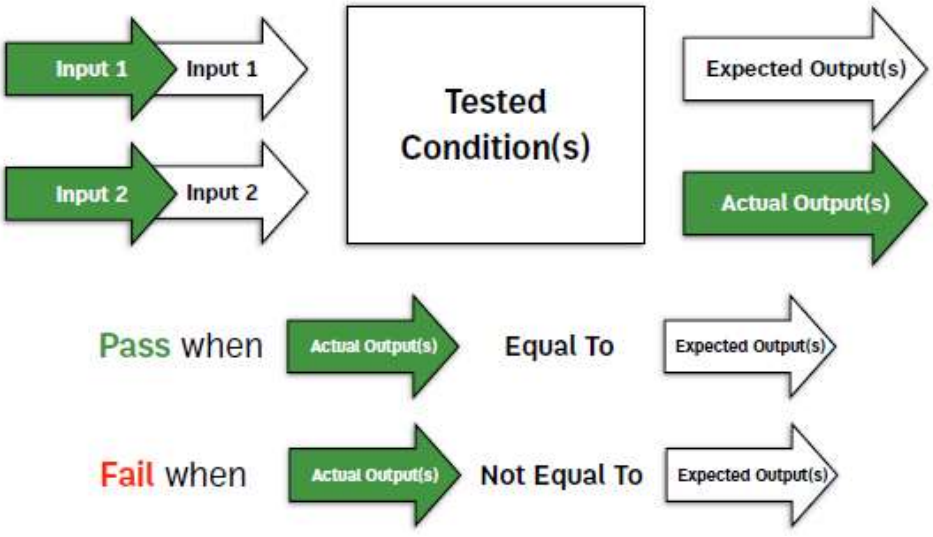


Test Case = Tested Conditions + Test Data Production
Expected Output — Tested Conditions — Input (s) – Data Production

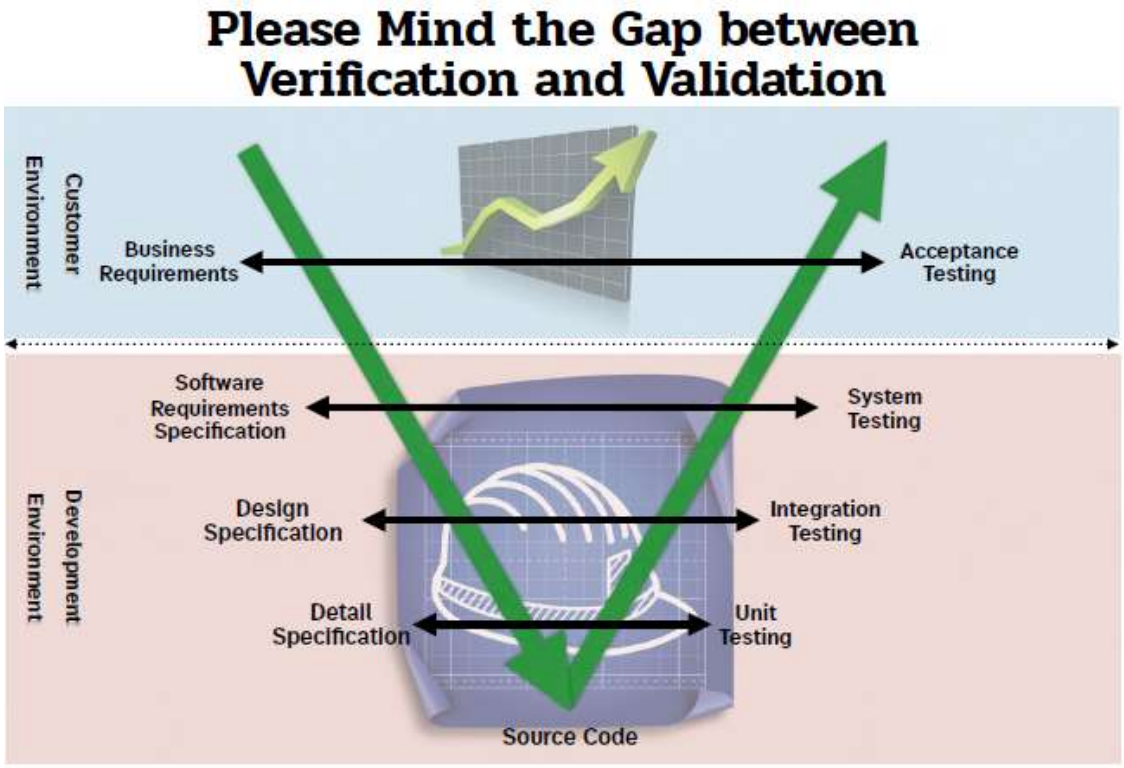
Test Case คือ กรณีที่ใช้ในการทดสอบ ประกอบด้วย เงื่อนไขที่ใช้ในการทดสอบ (tested conditions) ร่วมกับการทดสอบด้วยข้อมูลจริง (test data production)

When Test Case/Scenario is Executed

*The system is provided with the specified input(s) and **the actual output(s)** are compared with **the expected output(s)***



ผลการทดสอบจะผ่านการยอมรับ ก็ต่อเมื่อ ผลลัพธ์ที่เกิดขึ้นจริง (actual output) มีค่าเท่ากับผลลัพธ์ที่คาดหวัง (expected output) เท่านั้น



ในการทดสอบซอฟต์แวร์ จะใช้คำ 2 คำนี้พร้อมๆกัน อาจเป็นเพราะแยกไม่ออกว่าอันไหนเป็น Verification อันไหนเป็น Validation

Verification จะตั้งคำถามว่า **Are we building the product right?**

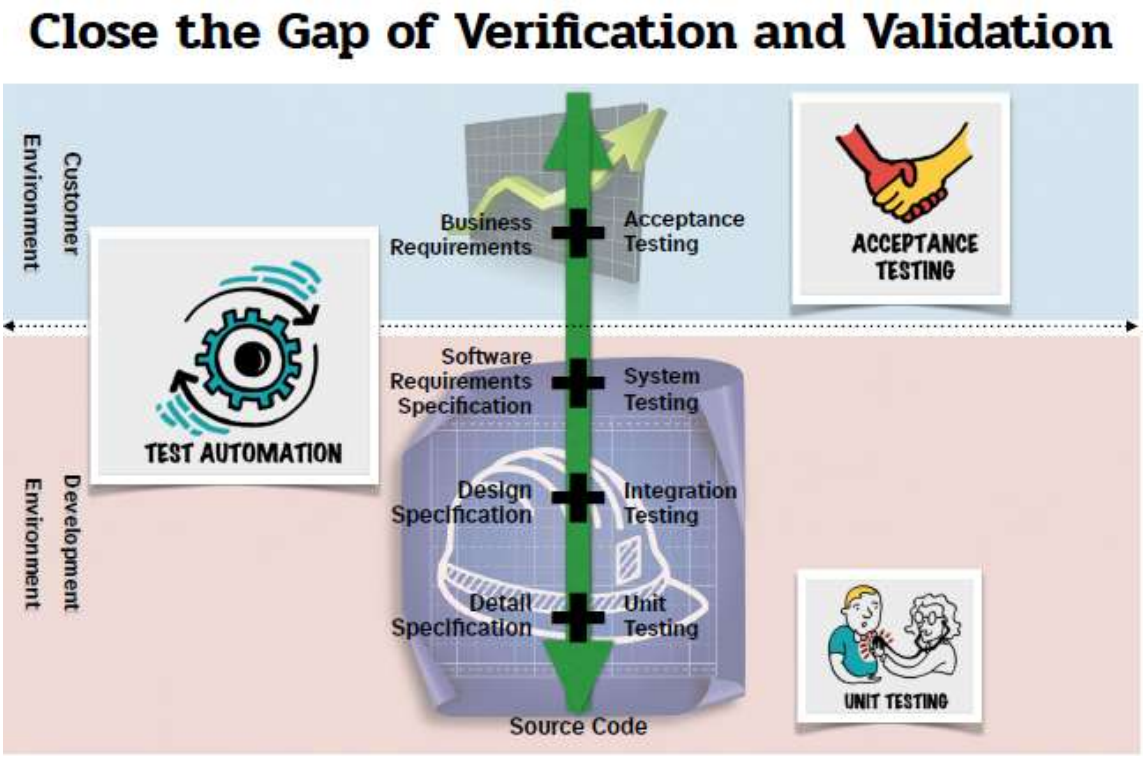
Validation กลับตั้งคำถามว่า **Are we building the right product?**

Verification จะต้องการ product ที่ถูกต้องตาม requirement และ design เนื่องจากกระบวนการในการทำ verification (ขั้นตอนการ review รูปแบบต่างๆ) จะเน้นที่การ remove defect ออกตั้งแต่ขั้นตอนแรกๆ เพื่อที่จะ effort ในการทำ testing ลง

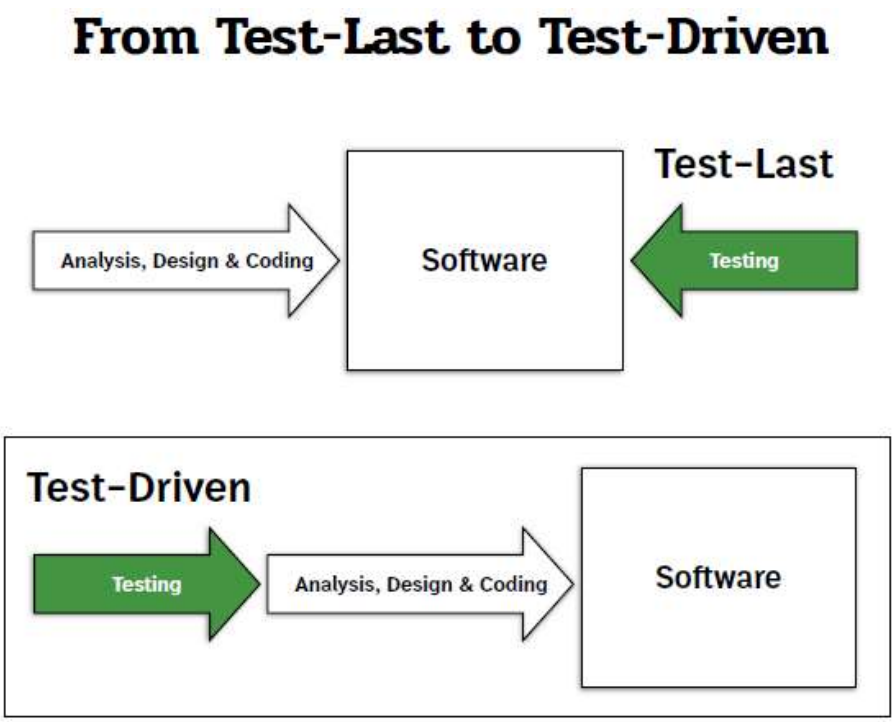
ส่วน Validation หมายถึง การทดสอบ product ว่าตรงกับ business requirement หรือไม่ (ไม่ใช่ software requirement) นั่นคือ Software ที่ได้จะต้องตรงกับสิ่งที่ลูกค้าต้องการ และสามารถใช้งานได้จริง

และความแตกต่างที่เห็นได้ชัดอีกอย่างหนึ่งคือ Verification จะเน้นที่การตรวจสอบ Work product (สิ่งที่เกิดขึ้นระหว่างกระบวนการ แต่ไม่ใช่สิ่งที่เอาไปใช้งานจริง) แต่ Validation จะตรวจสอบที่ Product (Software ที่ผลิตได้) เป็นหลัก

ดังนั้น ในการทำ Software Testing จำเป็นจะต้องทำทั้ง 2 กลุ่มนี้ ซึ่งอาจทำมากหรือทำน้อยให้พิจารณาจากขนาดและความซับซ้อนของ Project

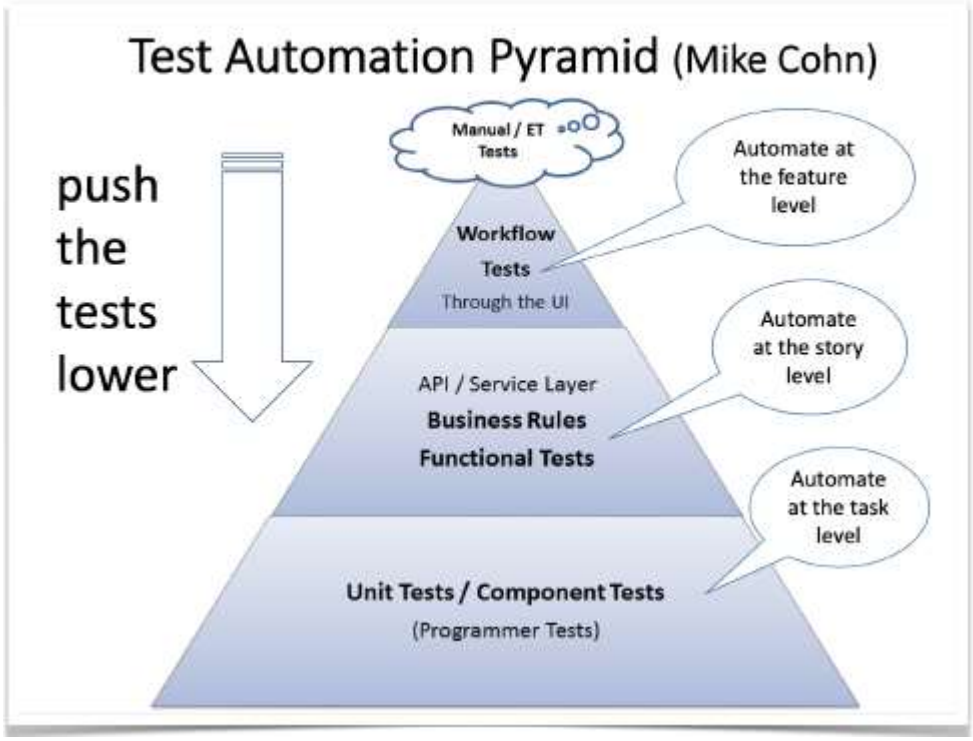


การปิด gap ระหว่าง Verification และ Validation จึงสามารถดำเนินการได้ด้วยการผสานขั้นตอนการทดสอบทั้งหมดให้เป็นหนึ่งเดียว และใช้กระบวนการทดสอบแบบอัตโนมัติ (test automation) ร่วมด้วย

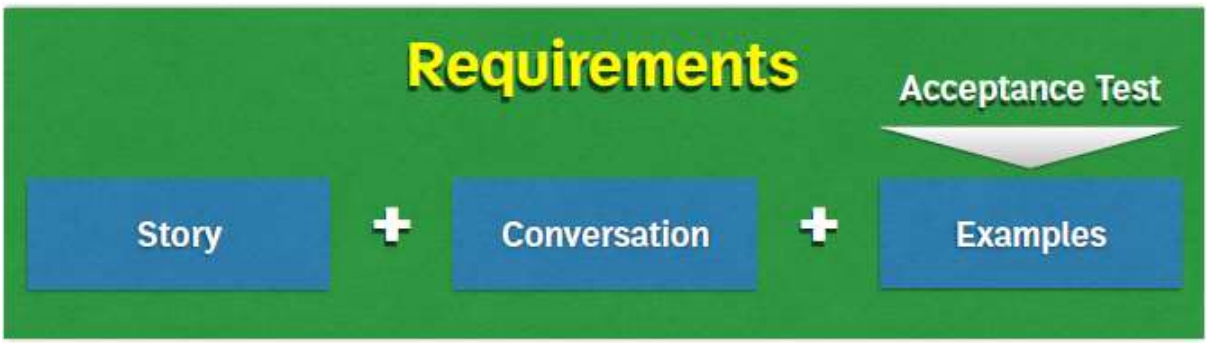


การทดสอบซอฟต์แวร์แบบเดิมๆ นั้น จะต้องผ่านกระบวนการวิเคราะห์ ออกแบบ และเขียนโปรแกรมมาก่อน จึงจะเข้าสู่กระบวนการทดสอบ แต่ Test-Driven เป็นกระบวนการพัฒนาซอฟต์แวร์ที่ต้องสร้าง Test ขึ้นมาก่อน แล้วจึงค่อยเริ่มเขียนโปรแกรม โปรแกรมที่ทำงานได้ถูกต้องจะต้องสามารถ run test ผ่าน

Push The Tests Lower



Acceptance Test



Stories are not enough

Acceptance test define by customer

Conversation

Wireframe, user cases...whatever is needed

Acceptance Test คือ กระบวนการทดสอบระบบก่อนใช้งานจริง เพื่อตรวจสอบว่าระบบสามารถตอบสนองตามความต้องการของลูกค้า (requirement) ได้ตรงตาม Business Flow จริงๆ ของลูกค้าในระดับที่ยอมรับได้ และทดสอบในสภาพแวดล้อมที่ใกล้เคียงกับการใช้งานจริงมากที่สุด

3. ประโยชน์ที่ได้รับ

3.1 ประโยชน์ที่ผู้รับทุนได้รับ

1) ได้รับความรู้และทักษะเกี่ยวกับการทดสอบซอฟต์แวร์อย่างมีคุณภาพ จากวิทยากรผู้เชี่ยวชาญที่มีประสบการณ์ในเรื่องนี้โดยตรง

2) ได้รับมุมมองและแนวคิดที่หลากหลายในเรื่องของการทดสอบซอฟต์แวร์อย่างมีคุณภาพ จากการแลกเปลี่ยนเรียนรู้และประสบการณ์ระหว่างผู้เข้ารับการอบรม

3.2 ประโยชน์ที่มหาวิทยาลัยได้รับ

1) บุคลากรของมหาวิทยาลัยได้รับความรู้เพิ่มขึ้น

2) ทำให้เห็นมุมมองของแนวความคิดที่หลากหลายจากผู้เข้ารับการอบรม ซึ่งอยู่ในแวดวงของวิชาชีพด้านเทคโนโลยีสารสนเทศ

4. ข้อเสนอแนะ

การเข้าร่วมการสัมมนาเพื่อรับฟังการถ่ายทอดผลงานวิจัยและการค้นพบใหม่ รวมถึงการแลกเปลี่ยนข่าวสาร วิทยาการความรู้ และประสบการณ์ โดยเฉพาะอย่างยิ่งในด้านเทคโนโลยีสารสนเทศและการสื่อสารที่มีความก้าวหน้าและเปลี่ยนแปลงไปอย่างรวดเร็ว เป็นการสร้างแนวคิดที่ดีและได้เห็นมุมมองของแนวความคิดที่หลากหลายและแตกต่างกันออกไป สามารถนำมาใช้ประโยชน์ได้ทั้งต่อผู้รับทุนเองและประโยชน์ที่มหาวิทยาลัยจะได้รับ