FutureSkill

# บทนำ
# การทดสอบซอฟต์แวร์

# การทดสอบซอฟต์แวร์ (Software Testing) คืออะไร

การทดสอบซอฟต์แวร์ (อังกฤษ: software testing) เป็นกระบวนการเพื่อช่วยให้ซอฟต์แวร์ที่พัฒนามีความถูกต้อง, ความสมบูรณ์, ปลอดภัย, และมีคุณภาพที่ดี

*– ที่มา: th.wikipedia.org/การทดสอบซอฟต์แวร์*

Software testing is a set of activities to discover defects and evaluate the quality of software artifacts.

*– ที่มา: ISTQB Certified Tester Foundation Level Syllabus, 1.1. What is Testing?*

# จุดประสงค์ของการทดสอบซอฟต์แวร์

- ตรวจสอบ และประเมินซอฟต์แวร์ หรือผลิตภัณฑ์ ตั้งแต่วิเคราะห์ความต้องการของผู้ใช้ การออกแบบผลิตภัณฑ์ และโค้ดที่พัฒนาขึ้นเพื่อสร้างซอฟต์แวร์
- ค้นหาข้อผิดพลาดที่อาจทำให้ซอฟต์แวร์ทำงานผิดปกติ
- เพื่อให้การทดสอบครอบคลุม
- ลดความเสี่ยงของการได้ซอฟต์แวร์ หรือผลิตภัณฑ์ที่ไม่ได้คุณภาพ
- ตรวจสอบว่าซอฟต์แวร์ตอบโจทย์ความต้องการของผู้ใช้

ที่มา: ISTQB Tester Foundation Syllabus v4.0

# จุดประสงค์ของการทดสอบซอฟต์แวร์

- ตรวจสอบว่าซอฟต์แวร์ปฏิบัติตามกฎข้อบังคับ หรือกฎหมายที่เกี่ยวข้องอย่างถูกต้อง
- เพื่อให้ข้อมูลแก่ผู้เกี่ยวข้องกับการผลิตซอฟต์แวร์ เพื่อประกอบการตัดสินใจต่าง ๆ
- เพื่อให้มั่นใจว่าซอฟต์แวร์มีคุณภาพ
- ตรวจสอบว่าซอฟต์แวร์พัฒนาได้เสร็จสมบูรณ์ และทำงานได้ตามความต้องการ

ที่มา: ISTQB Tester Foundation Syllabus v4.0

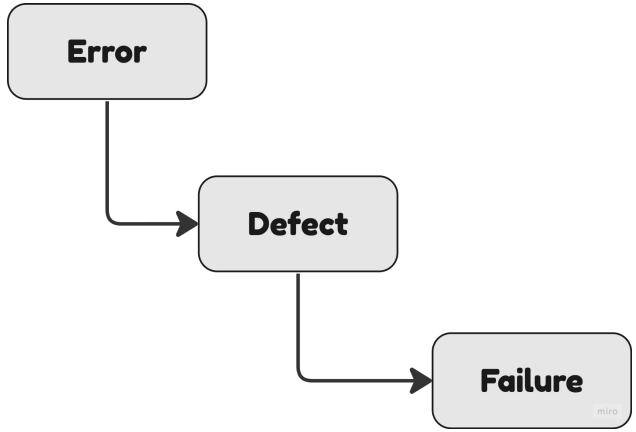
# หลักการของการทดสอบซอฟต์แวร์

- การทดสอบใช้หาข้อผิดพลาดที่เกิดขึ้น แต่ไม่สามารถพิสูจน์ได้ว่าซอฟต์แวร์ไม่มีข้อผิดพลาดเลย
- ไม่สามารถทำการทดสอบทุกสิ่งทุกอย่างได้ทั้งหมด
- การทดสอบที่เกิดขึ้นเร็วในขั้นตอนของการพัฒนาจะช่วยประหยัดเวลา และเงินที่ใช้ในการพัฒนาซอฟต์แวร์
- ข้อผิดพลาดมักกระจุกตัวกัน
- การทดสอบซ้ำ ๆ เดิมอาจไม่สามารถหาข้อผิดพลาดใหม่ ๆ ได้
- การทดสอบซอฟต์แวร์นั้นจะแตกต่างกันตามบริบท
- การตรวจสอบความถูกต้องของซอฟต์แวร์ ไม่สามารถการันตีว่าซอฟต์แวร์จะตอบโจทย์ความต้องการของได้เสมอไป ควรประเมินด้วยว่า ผู้ใช้งานสามารถใช้งานซอฟต์แวร์ได้ตามที่ต้องการ

ที่มา: ISTQB Tester Foundation Syllabus v4.0

## นิยามของ Bug และ Defect

Bug หรือ Defect คือ ข้อผิดพลาดที่เกิดขึ้นในซอฟต์แวร์ ซึ่งอาจเกิดจากข้อผิดพลาดในการออกแบบ การพัฒนา หรือการทดสอบ ซึ่งมักจะทำให้ซอฟต์แวร์ไม่ทำงานตามที่คาดหวังหรือมีพฤติกรรมที่ผิดปกติ ก่อให้เกิดปัญหาในการทำงานหรือเสียหายกับซอฟต์แวร์



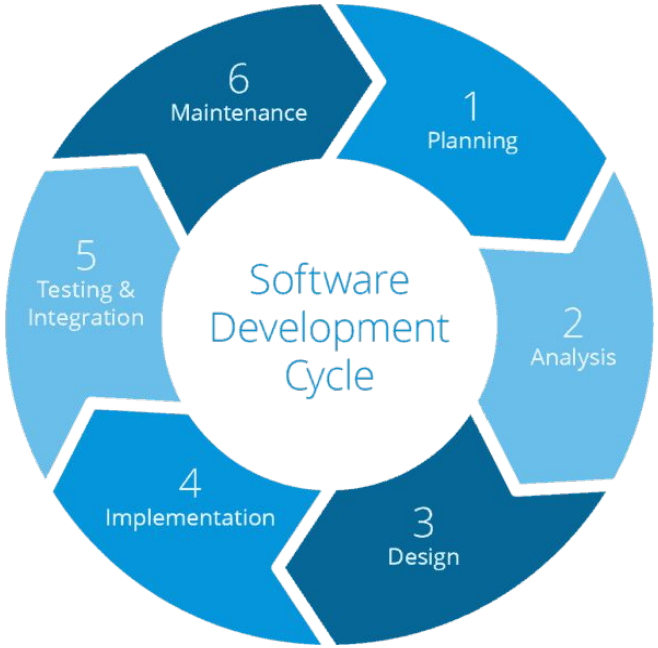
FutureSkill

# Software Development Life Cycle (SDLC)

## วงจรการพัฒนาซอฟต์แวร์ (SDLC)

กระบวนการพัฒนาซอฟต์แวร์ที่ถูกออกแบบมาเพื่อสามารถพัฒนาและส่งมอบซอฟต์แวร์ได้อย่างมีคุณภาพและสมบูรณ์

**ตัวอย่าง SDLC Models** เช่น Waterfall, V-model, Iterative-Incremental model เป็นต้น



FutureSkill

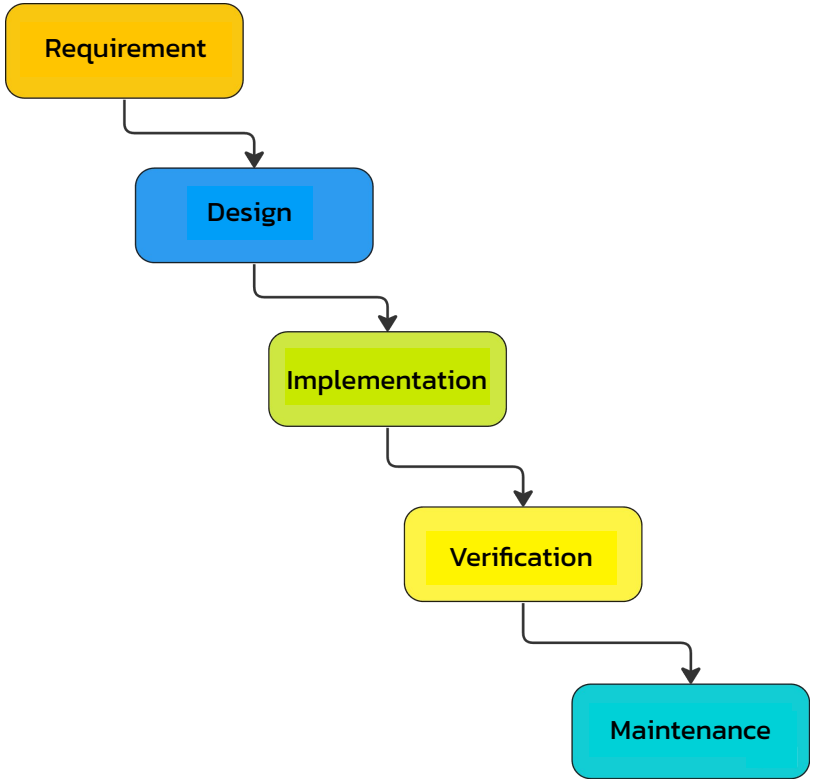
# SDLC Models

# Waterfall model

ขั้นตอนการทดสอบ (Testing) จะเกิดขึ้นหลังจากพัฒนาเสร็จเรียบร้อยแล้ว โดยที่ QA หรือ Tester จะทำการ

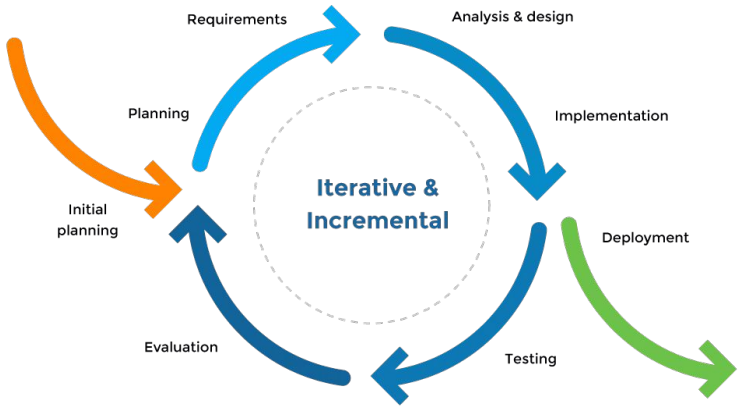
**Verification** เพื่อตรวจสอบให้มั่นใจว่าซอฟต์แวร์ หรือผลิตภัณฑ์ที่เราพัฒนาสามารถใช้งานได้ถูกต้องตามฟังก์ชันการใช้งาน

**Validation** เพื่อตรวจสอบให้มั่นใจว่าการทำงานของซอฟต์แวร์ หรือผลิตภัณฑ์ตรงตามความต้องการ และตอบโจทย์ผู้ใช้งาน



# Iterative-Incremental model

- การพัฒนาซอฟต์แวร์ หรือผลิตภัณฑ์ เป็นรอบ ๆ (cycle)
- แต่ละรอบมี time box หรือกำหนดเวลา เช่น รอบละ 2 สัปดาห์
- แบ่งการพัฒนาออกเป็นส่วนย่อย ๆ ที่สามารถเสร็จสมบูรณ์ได้ภายในเวลาที่กำหนด
- เพื่อให้ได้ feedback จากการทดสอบ และการใช้งานจากผู้ใช้งานได้รวดเร็ว
- และนำ feedback ที่ได้มาปรับปรุงพัฒนาซอฟต์แวร์เพื่อให้มีประสิทธิภาพมากขึ้น



# Agile Software Development

การพัฒนาซอฟต์แวร์ หรือผลิตภัณฑ์ที่เน้นความยืดหยุ่นและความร่วมมือในการทำงานร่วมกันของทีมพัฒนา โดยเน้นให้มีการส่งมอบซอฟต์แวร์ที่มีความคืบหน้าเป็นระยะ ๆ และรับความคิดเห็นและตอบสนองต่อความเปลี่ยนแปลงของลูกค้าและผู้ใช้งานอย่างรวดเร็ว

**Agile Manifesto**

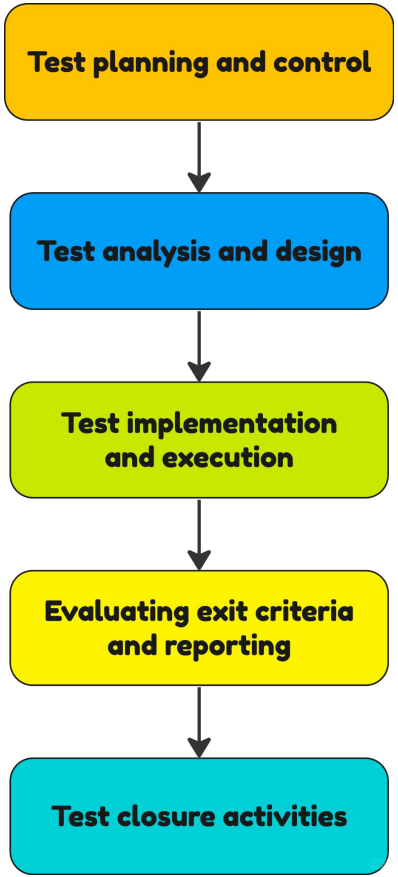
**Individuals and interactions** over processes and tools

**Working software** over comprehensive documentation

**Customer collaboration** over contract negotiation

**Responding to change** over following a plan

# Fundamental Test Process



# บทบาทหน้าที่ของ QA ในแต่ละขั้นตอนการทำงาน


**Requirement**

ช่วย Analyst วิเคราะห์ความต้องการ โดยคำนึงถึงผู้ใช้งาน และช่วยวิเคราะห์ความเสี่ยง และกรณีที่อาจจะทำให้เกิดข้อผิดพลาดในการพัฒนาซอฟต์แวร์

วางแผนการทดสอบ และสร้างเทสเคสเพื่อทดสอบซอฟต์แวร์หรือผลิตภัณฑ์

**Design**

วิเคราะห์เรื่อง user experience และให้ความเห็นเกี่ยวกับ user interface ที่อาจทำให้เกิดปัญหาได้ทั้งในการพัฒนา และการใช้งาน

**Implementation**

ทำงานร่วมกับ developer เพื่อป้องกันข้อผิดพลาดที่อาจเกิดขึ้น โดยการพูดคุยเรื่องการทดสอบ และความเสี่ยงที่อาจก่อปัญหาในการพัฒนา

**Verification**

ทำการทดสอบซอฟต์แวร์หรือผลิตภัณฑ์ โดยตรวจสอบการทำงาน ว่าถูกต้องตามความต้องการ และตอบโจทย์การใช้งาน

รายงานข้อผิดพลาดเพื่อทำการแก้ไข และทดสอบซ้ำ

รายงานผลการทดสอบ เพื่อพิจารณาว่า สามารถส่งมอบงานให้ผู้ใช้งานได้หรือไม่

**Maintenance**

รายงานข้อผิดพลาดที่เจอบน Production และทำการทดสอบซ้ำหลังจากข้อผิดพลาดได้รับการแก้ไขแล้ว

FutureSkill

# ประเภทของการทดสอบต่าง ๆ

# ระดับของการทดสอบระบบ (Test Levels)

| | |
|---|---|
| Unit Test | ทดสอบแต่ละ component |
| Integration Test | ทดสอบการรวมกันของ component |
| System Test | ทดสอบระบบทั้งหมด |
| Acceptance Test | ทดสอบระบบสุดท้าย |

ที่มา: https://www.guru99.com/levels-of-testing.html

FutureSkill

# ประเภทของการทดสอบ (Types of Testing)

## Functional Testing

- Unit testing
- Integration testing
- System testing
- System integration testing
- Regression testing
- User acceptance testing
- Exploratory testing
- Smoke testing
- Sanity testing

## Non-functional Testing

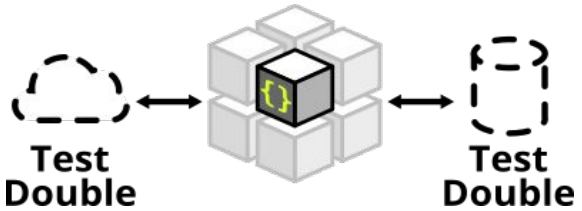
- Performance testing
- Security testing
- Compatibility testing

## Unit testing

การทดสอบแบบ Unit test คือ การทดสอบเฉพาะแต่ละ unit หรือ component ต่าง ๆ แยกกัน ทดสอบว่าแต่ละฟังก์ชันของโค้ด สามารถทำงานได้ถูกต้อง

ตัวอย่างเช่น ฟังก์ชันบวกของเครื่องคิดเลข unit test จะทดสอบว่าฟังก์ชันสามารถบวกเลขได้ถูกต้อง

Developer มักจะเป็นผู้ทำการทดสอบ unit test บน development environment

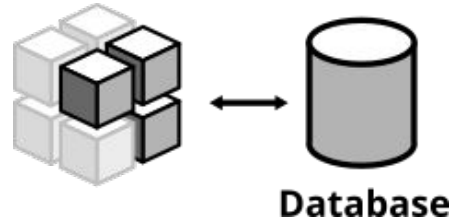


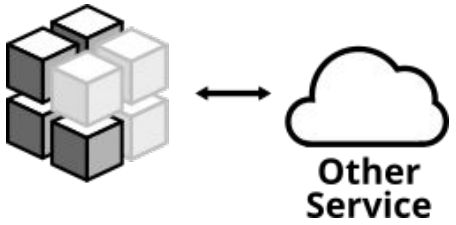
# Integration testing

การทดสอบส่วนที่แต่ละระบบเชื่อมต่อกัน หรือมีปฏิสัมพันธ์กัน

ตัวอย่างเช่น

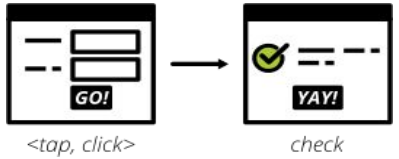
- การทดสอบ component integration test
- การทดสอบส่วนที่ต่อกับ database
- การทดสอบส่วนที่ต่อกับ service อื่น ๆ เช่น การเชื่อมต่อกับ 3rd party API

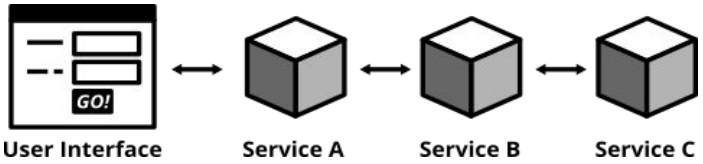


## System testing

การทดสอบการทำงานของซอฟต์แวร์ หรือระบบทั้งระบบ ซึ่งก็จะประกอบไปด้วย user interface test (UI test) และ end-to-end test (E2E)



- **System integration testing (SIT)**
  การทดสอบระบบที่เชื่อมต่อกัน รวมถึงที่เชื่อมต่อกับ service ภายนอกด้วย
- **Acceptance testing** หรือ **User acceptance testing (UAT)**
  การทดสอบที่ทำโดยตัวแทนของฝ่ายธุรกิจ หรือตัวแทนของผู้ใช้งาน เพื่อทดสอบว่าซอฟต์แวร์ หรือระบบที่พัฒนานั้นสามารถใช้งานได้ตามความต้องการทางธุรกิจ
- **Regression testing**
  การทดสอบที่ตรวจสอบว่า ซอฟต์แวร์ หรือระบบยังทำงานได้ปกติ หลังจากมีการเปลี่ยนแปลง เช่น การแก้ไขข้อผิดพลาด หรือมี feature ใหม่

- **Exploratory testing**
  การทดสอบที่ใช้เพื่อเรียนรู้การทำงานของซอฟต์แวร์ หรือระบบ โดยการคิดว่าจะทดสอบอะไร และทำการทดสอบในระยะเวลาที่จำกัด ขั้นตอนของการทำ Exploratory testing คือ การออกแบบการทดสอบ การทำการทดสอบ และการสังเกตและ เรียนรู้ระบบว่าสามารถทำงานได้อย่างไรบ้าง
- **Smoke testing**
  การทดสอบสั้นๆ ที่ตรวจสอบว่าการทำงาน หรือฟังก์ชันหลักของซอฟต์แวร์สามารถใช้งานได้ปกติหลังจากมีการ release หรือส่งมอบซอฟต์แวร์เวอร์ชันใหม่
- **Sanity testing**
  เป็นการทดสอบย่อย (และสั้นๆ) ของ regression testing ที่จะตรวจสอบเฉพาะจุด หรือการทำงานที่มีการเปลี่ยนแปลง เพื่อให้แน่ใจว่าซอฟต์แวร์ทำงานได้ปกติหลังจากการเปลี่ยนแปลงล่าสุด

# Non-functional Testing

**Performance testing**
การทดสอบที่ตรวจสอบ และประเมินว่าระบบมีการตอบสนอง และความเสถียรต่อการใช้งานมาก ๆ อย่างไร ซึ่งจะตรวจสอบความเร็วในการตอบสนองต่อผู้ใช้งาน ความ ทดทานต่อการใช้งาน (robustness) และความเชื่อถือได้ของการทำงาน (reliability) และอื่น ๆ

**Load testing**
หนึ่งในวิธีการทดสอบ performance testing ที่จะทดสอบว่าระบบสามารถทำงานได้ภายใต้การใช้งานปกติและการใช้งานอย่างหนัก
Load testing จะทำให้เราทราบว่าระบบสามารถรองรับการใช้งานได้มากน้อยเท่าใด และช่วยให้เราทราบว่าตรงจุดไหนคือจุดคอขวดที่ทำให้ระบบทำงานช้า

## Security testing

การทดสอบที่หาข้อผิดพลาด หรือช่องโหว่ของซอฟต์แวร์ หรือระบบ ที่อาจะทำให้เกิดความเสี่ยงเกิดภัยคุกคามและเกิดความเสียหายได้ เช่น ช่องโหว่ที่ทำให้ข้อมูลของผู้ใช้งานเสียหาย หรือนำข้อมูลออกไปใช้งานโดยไม่ได้รับอนุญาต

Security test จะทำการทดสอบเกี่ยวกับการเข้าถึงข้อมูล ความถูกต้องของข้อมูล การยืนยันตัวตนของผู้ใช้ สิทธิในการใช้งานและเข้าถึงข้อมูล เป็นต้น

## Compatibility testing

การทดสอบที่จะตรวจสอบว่า ซอฟต์แวร์ หรือระบบสามารถใช้งานได้บน hardware, operating systems, applications, network environments หรืออุปกรณ์ mobile device ที่แตกต่างกันได้

**Hardware**

**Operating Systems**

**Software**

**Network**

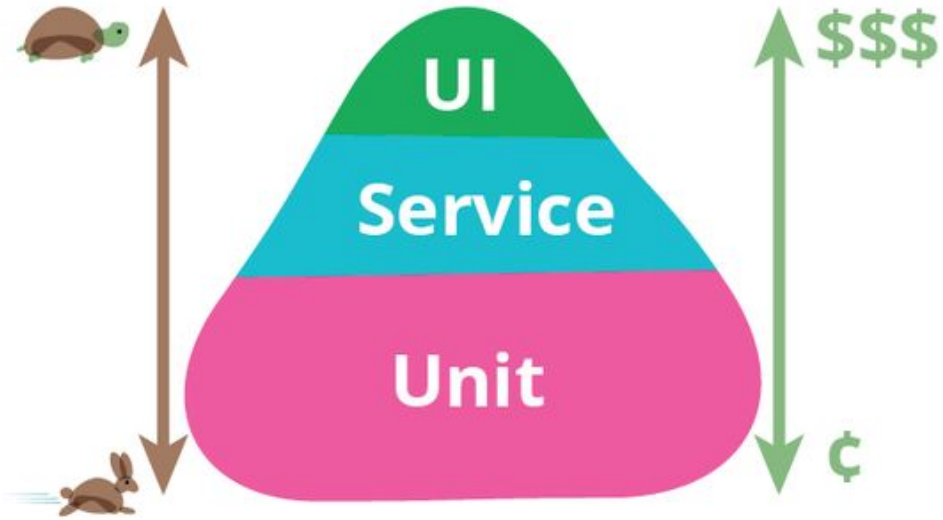
**Browser**

**Devices**

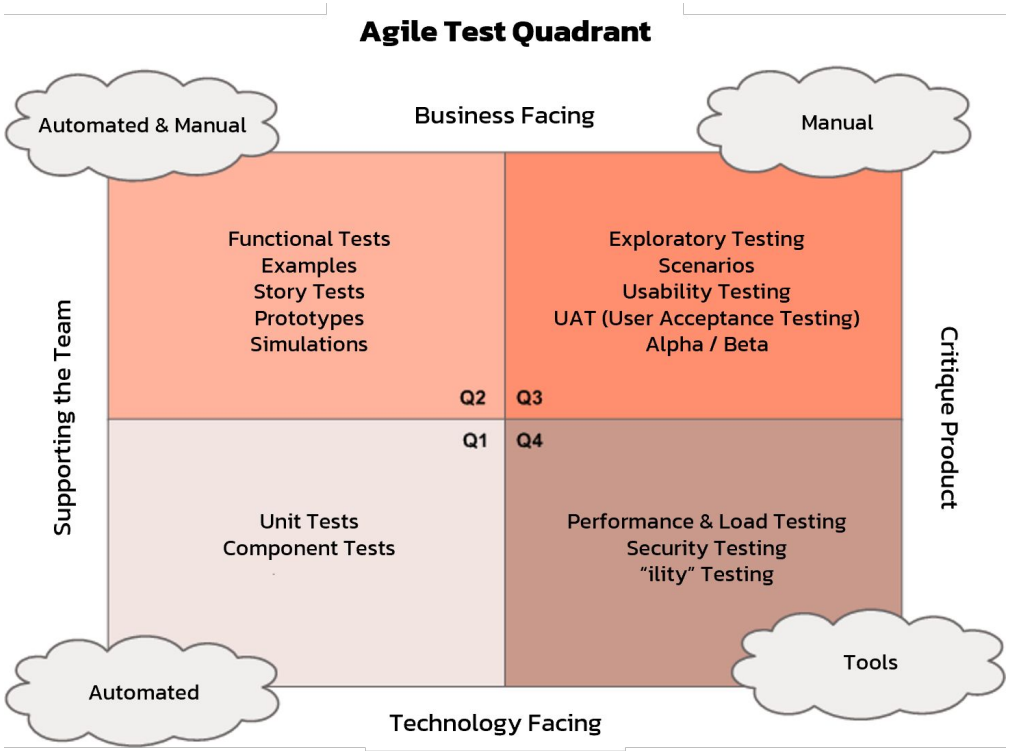
**Mobile**

**Versions**

# เทสปิรามิด (Test Pyramid)

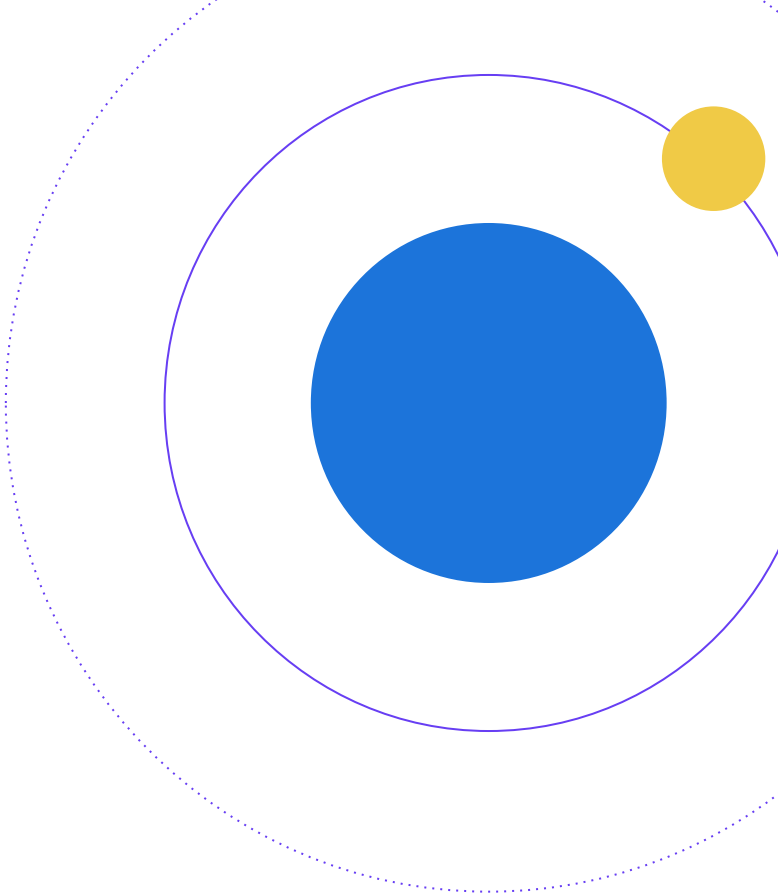


# เทสควอดรันต์ (Test Quadrants)



FutureSkill

# Agile Testing

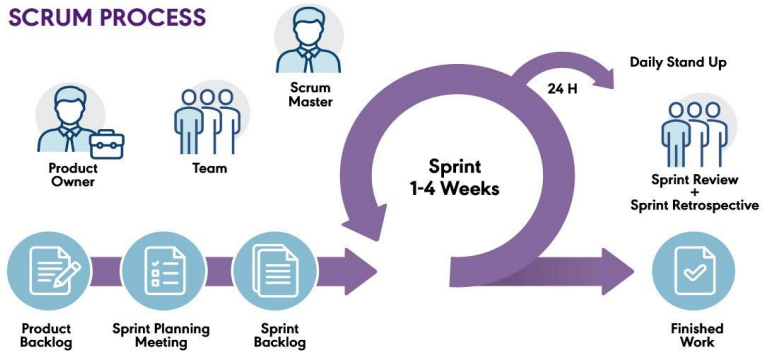


FutureSkill

# Agile Framework

# Scrum

Scrum เป็น Framework ที่ใช้ระบบการทำงานในรอบ ๆ สั้น ๆ เรียกว่า "Sprints" ซึ่งระยะเวลาแต่ละรอบจะอยู่ที่ 1 – 4 สัปดาห์ ในแต่ละรอบจะมีการวางแผนความต้องการที่ต้องส่งมอบ และทีมพัฒนาจะทำงานกันเพื่อให้ส่งซอฟต์แวร์ที่ใช้งานได้ตามความต้องการในทุก ๆ Sprint

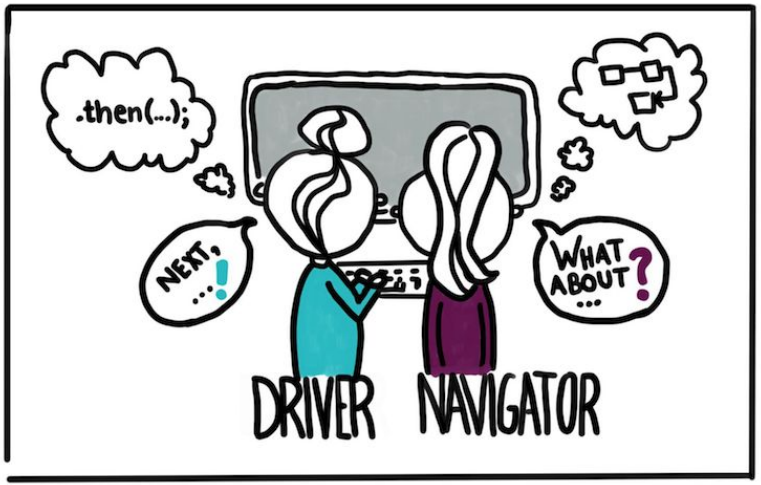


# Kanban

Kanban เป็น Framework ที่ใช้กระดาน Kanban ซึ่งแสดงงานที่กำลังทำอยู่ ทีมพัฒนาสามารถเลือกทำงานตามความต้องการและความเร่งด่วนของงานที่ได้รับ

## Extreme Programming (XP)

เน้นการทำงานในรอบที่สั้น ๆ (Iteration) และการใช้ Test Driven Development (TDD) และใช้ Pair Programming ในการพัฒนาซอฟต์แวร์ หรือผลิตภัณฑ์



FutureSkill

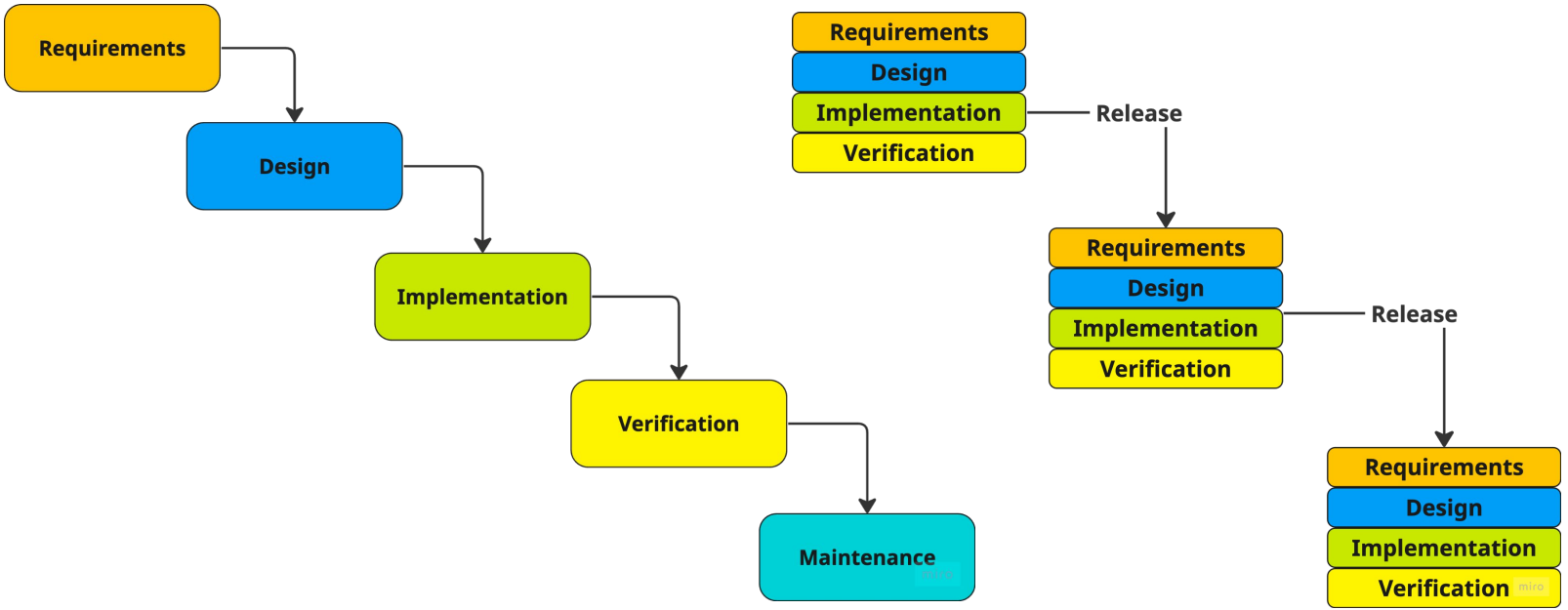
# Agile testing คือ

**Agile testing** หรือ การทดสอบแบบ Agile คือ แนวทางการทำการทดสอบซอฟต์แวร์ ที่ใช้หลักการและแนวคิดของการพัฒนาซอฟต์แวร์แบบ Agile โดยจะประกอบไปด้วยการใช้แนวปฏิบัติต่าง ๆ เพื่อส่งเสริม**การทำงานร่วมกันของทีมพัฒนา ความยืดหยุ่น** และ**ความพร้อมรับการเปลี่ยนแปลง** ซึ่งจะทำให้ทีมพัฒนาสามารถส่งมอบซอฟต์แวร์ที่มีคุณภาพสูง ในระยะเวลารวดเร็ว

<u>การทดสอบแบบ Agile เน้นเรื่อง</u>

- การพูดคุย และทำงานร่วมกันของทีม (Whole Team approach and Collaboration)
- การป้องกันไม่ให้เกิดข้อผิดพลาด มากกว่า หาข้อผิดพลาดในภายหลัง
- การได้รับฟีดแบคอย่างรวดเร็ว ทั้งในระหว่างการทำงาน และจากผู้ใช้งาน (Fast feedback)
- การเปิดเผยสถานะ และความโปร่งใสของการทำงาน (Visibility and Transparency)

# Traditional testing vs Agile testing



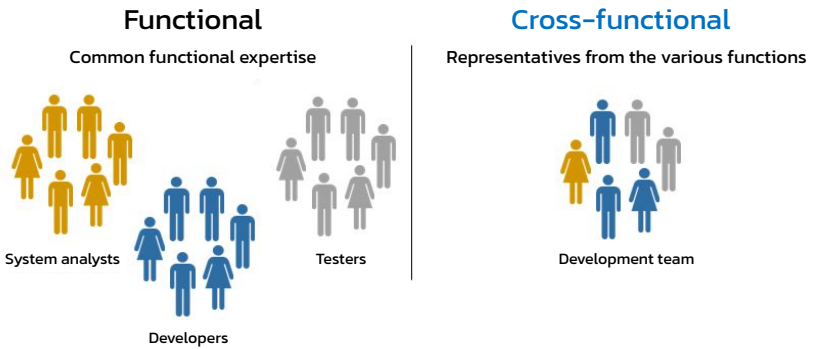

THE TESTING Manifesto
we value:
Testing throughout OVER testing at the end
Preventing bugs OVER finding bugs
Testing understanding OVER checking functionality
Building the best system OVER breaking the system
Team responsibility for quality OVER tester responsibility
www.GrowingAgile.co.za
@growingAgile

FutureSkill

# การทำงานร่วมกันในการพัฒนาซอฟต์แวร์แบบ Agile

# Whole Team Approach และ Agile Testing Mindset

- การทำงาน และความรับผิดชอบร่วมกันของทีม
- ทุก ๆ คนในทีม รับผิดชอบต่อคุณภาพของซอฟต์แวร์หรือผลิตภัณฑ์
- Cross-functional team หรือ ทีมพัฒนาที่รวมคนที่ทำหน้าที่ต่าง ๆ มาอยู่ในทีมเดียวกัน ทั้ง developers, business analyst, quality analyst เป็นต้น
- Agile tester ไม่ใช่ Quality police
- พยายามเรียนรู้ และพัฒนาความรู้และทักษะอยู่สม่ำเสมอ



FutureSkill

# Conclusion and What's next

# ความรู้พื้นฐานของการทดสอบซอฟต์แวร์

- นิยาม และจุดประสงค์ของการทดสอบซอฟต์แวร์
- วงจรการพัฒนาซอฟต์แวร์
- ประเภทของการทดสอบต่าง ๆ
- Agile Testing

## สิ่งที่ควรเรียนรู้เพิ่มเติมต่อจากนี้

- พื้นฐานการทดสอบระบบแบบ Manual Test
- การวางแผนการทดสอบ
- การสร้างเทสเคส
- ISTQB certification