



SOFTWARE ENGINEERING

# Software Testing

**Dr. Rathachai Chawuthai**

Department of Computer Engineering  
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

# Agenda

- Introduction
- Software Testing
- Test Documentation

# Introduction



# Discussion

---

ทำไมต้องมี Software Testing ?

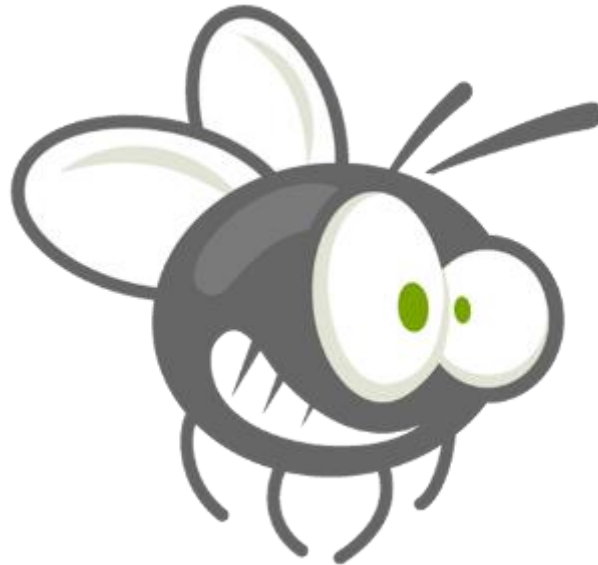
# Qualitied Software

---

- Reliability ความน่าเชื่อถือ
- Efficiency ประสิทธิภาพ (การใช้ทรัพยากรคุ้มค่า)
- Maintainability ความสามารถในการดูแลรักษา
- Compatibility ความสามารถเข้ากันได้กับสิ่งอื่นหรือเวอร์ชันอื่น
- Usability สามารถใช้ได้ง่ายเข้าใจได้ง่าย
- Performance ความสามารถในการตอบสนองผู้ใช้ (เร็ว, เยอะ)

# Bug ?

---



# Bug ?

(033) PRO 2 2.130476415  
correct 2.130676415

Relays 6-2 in 033 failed special speed test  
in relay .. 10.00 test.

Relays changed

1100 Started Cosine Tape (Sine check)  
1525 Started Mult + Adder Test.

1545



Relay #70 Panel F  
(moth) in relay.

First actual case of bug being found.  
~~1630~~ 1630 Antangent started.  
1700 closed down.

US NAVAL HISTORICAL CENTER

# Terms

---

- Defect ปัญหาที่พบในการทดสอบระบบ
- Error กิจกรรมของมนุษย์ที่ทำให้เกิดผลลัพธ์ที่ผิด
- Bug การปรากฏตัวของ error นั้นขณะซอฟต์แวร์ทำงาน
- Fault สถานะของซอฟต์แวร์ที่เกิดจาก error ตัวนั้น
- Failure ความผิดพลาดจากการทำงานของซอฟต์แวร์



A person makes an **error**



that creates a **fault** in the software



that can cause a **failure** in operation.



# Defect Types

---

- **Requirement**
  - เป็น Defect ที่เกิดจากการแก้ไข Requirement ของทาง Business โดยไม่แจ้งทีมที่เกี่ยวข้อง
- **Coding**
  - เป็น Defect ที่เกิดจากการ Coding ของทาง Developer ที่ไม่ตรวจสอบในส่วนนั้นๆ
- **Graphic Design**
  - เป็น Defect ที่เกิดจากการ Design ที่ไม่รองรับกับ Browser ต่างๆ หรือ เมื่อ developer นำ Design มาประกอบกับ Code แล้วทำให้การแสดงผลไม่ถูกต้อง
- **Data Test**
  - เป็น Defect ที่เกิดจาก Test data อาจไม่มีใน Environment หรือระบบอาจไม่รองรับ data นี้
- **Other**
  - ข้อจำกัดของระบบ, ข้อจำกัดของ Environment

# Defect Severity

---

- Critical

- Defect ที่ไม่สามารถทดสอบโปรแกรมในส่วนของ Function นั้นต่อได้เลย

- High

- Defect ที่เกิดจากการใส่ข้อมูลถูกต้อง แต่ระบบแสดงผลผิดพลาด เช่น Error ต่างๆ

- Medium

- ระบบจะแสดงผลถูกต้องเมื่อใส่ข้อมูลถูกต้อง แต่เมื่อใส่ข้อมูลไม่ถูกต้อง ระบบจะแสดงผลผิดพลาด เช่น Filed ที่มีการ Validate ผล เมื่อใส่ ค่าว่าง,อักขระพิเศษ ( ' , % , & ) และ Script ที่มีผลต่อการแสดงผลของระบบ ใดๆ จะแสดงผลผิดพลาด

- Low

- Defect ที่เกิดจากการแสดงผลของข้อความ หรือ เรื่องของการ Design ซึ่ง Defect เหล่านี้จะไม่มีผลกระทบกับการทำงานของระบบ

# Software Testing

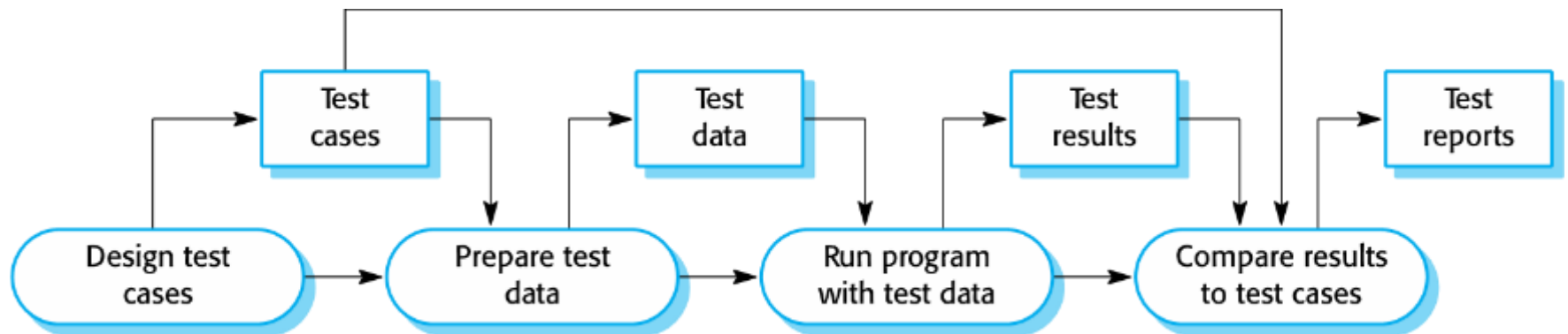
---

- เป็นการทดสอบความสมบูรณ์ของโปรแกรม รวมทั้งความน่าเชื่อถือและความถูกต้องของผลลัพธ์จากโปรแกรมที่พัฒนาขึ้น
- ตรวจสอบประสิทธิภาพการทำงานของซอฟต์แวร์ไปด้วย
- เป็นผลให้สัมพันธ์กับคุณภาพของซอฟต์แวร์ตามไปด้วย
- เป็นกิจกรรมที่จัดทำขึ้นเพื่อประเมินและปรับปรุงคุณภาพของซอฟต์แวร์ โดยการหาข้อผิดพลาดและปัญหาที่เกิดขึ้นและทำการแก้ไขปัญหา
- ส่วนใหญ่ทำโดย Testers
- มีทำโดย Developers และ Users
- มีแบบแผนการปฏิบัติ

# Testing Process

---

- A model of the software testing process




# Software Testing



# Test?

---



เพิ่มรายการสินค้าใหม่

ชื่อสินค้า

หมวดสินค้า  ▼

รายละเอียด

ราคา  บาท

Tag (ค้นด้วย ;)

วันที่เริ่มแสดง   
(YYYY-MM-DD)

เพิ่ม

# Black Box Testing

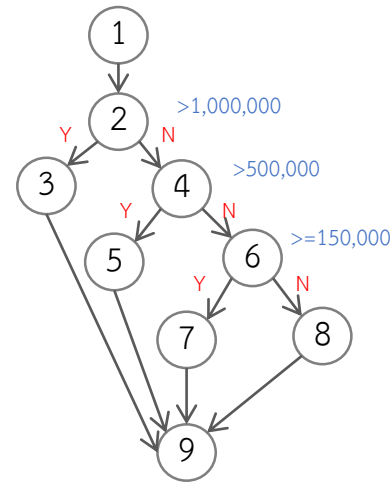


Test case	Precondition	Event	Expected Result	Note
TC1	Your cart is empty	Click btn Add item	1 item in your cart	S1=>S2
TC2	$n \geq 1$ items in your cart	Click btn Add item	$n+1$ items in your cart	S2=>S2
TC3	1 item in your cart	Click btn Remove item	Your cart is empty	S2=>S1
TC4	$n \geq 2$ items in your cart	Click btn Remove item	$n-1$ items in your cart	S2=>S2
TC5	$n \geq 1$ items in your cart	Click btn Check out	Display screen Check out	S2=>S3
TC6	Direct to screen Check out	Click btn Back	Display screen Shopping	S3=>S2
TC7	Direct to screen Check out	Click btn Payment	Display screen Payment	S3=>S4

# White Box Testing

```
float taxCal (float salary){  
    float tax = 0.0;  
    if(salary>1,000,000){  
        tax = salary*0.25 ;  
    }else if(salary>500,000){  
        tax = salary*0.15 ;  
    }else if(salary>=150,000){  
        tax = salary*0.05 ;  
    }else{  
        tax = 0.0  
    }  
    return tax;  
}
```

Flowchart



Path Coverage

- 1,2,3,9
- 1,2,4,5,9
- 1,2,4,6,7,8
- 1,2,4,6,8,9

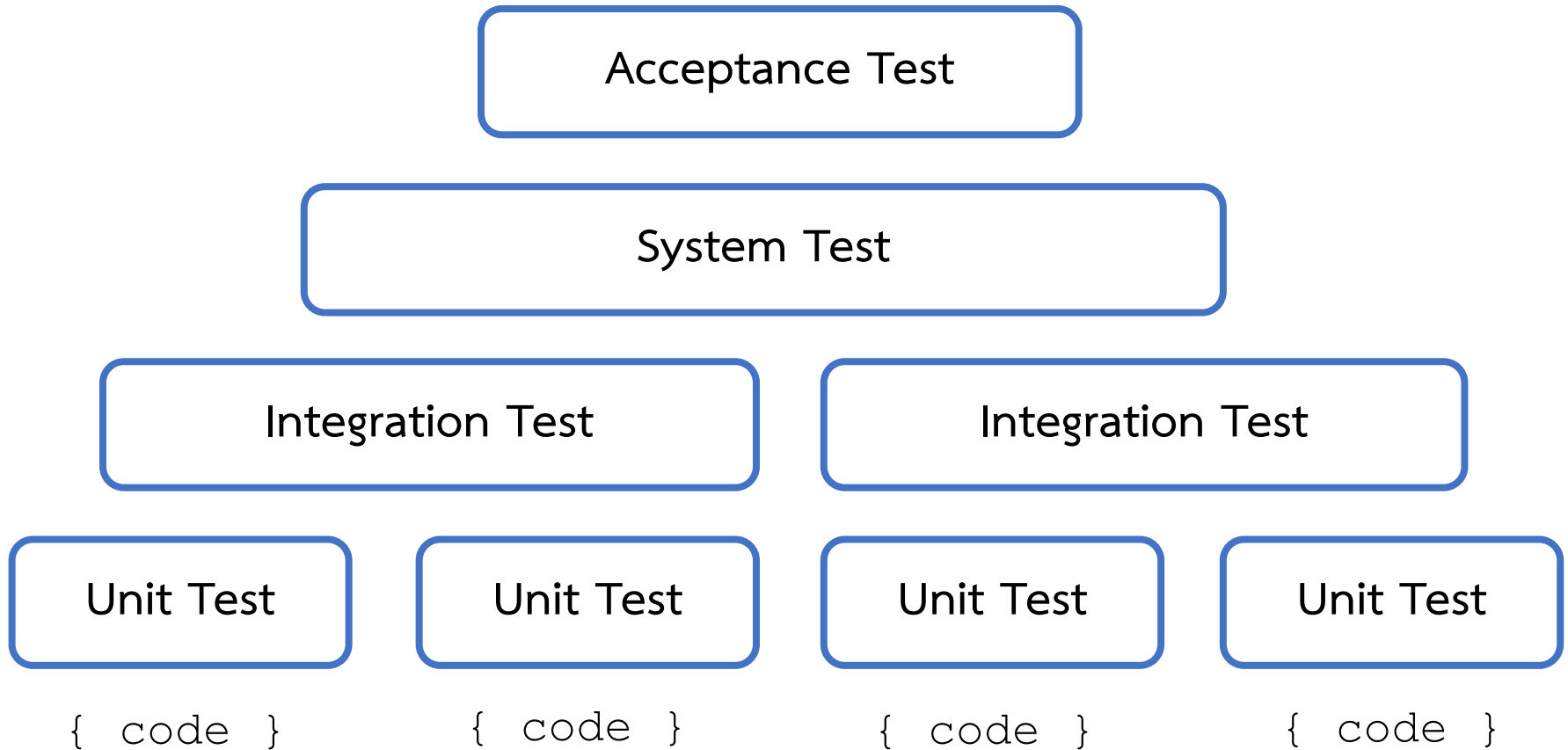
Test Cases

- |                      |                    |
|----------------------|--------------------|
| • salary = 0         | • salary = 499,999 |
| • salary = 999,999   | • salary = 500,000 |
| • salary = 1,000,000 | • salary = 500,001 |
| • salary = 1,000,001 | • อื่นๆ            |



# Software Testing

---



# Unit Test

## Function ใน class ชื่อ “Tax”

```
float taxCal (float salary){  
    float tax = 0.0;  
    if(salary>1,000,000){  
        tax = salary*0.25 ;  
    }else if(salary>500,000){  
        tax = salary*0.15 ;  
    }else if(salary>=150,000){  
        tax = salary*0.05 ;  
    }else{  
        tax = 0.0  
    }  
    return tax;  
}
```

## Unit Test Script (เป็น White Box Testing)

```
public class MyTest{  
    @Test  
    public void TestTax25(){  
        Tax tester = new Tax();  
        assertEquals("salary > 1,000,000",  
            25000.25, tester.taxCal(1000001));  
  
        assertEquals("salary = 1,000,000",  
            25000.00, tester.taxCal(1000000));  
    }  
    // more cases  
}
```

# Integration Test

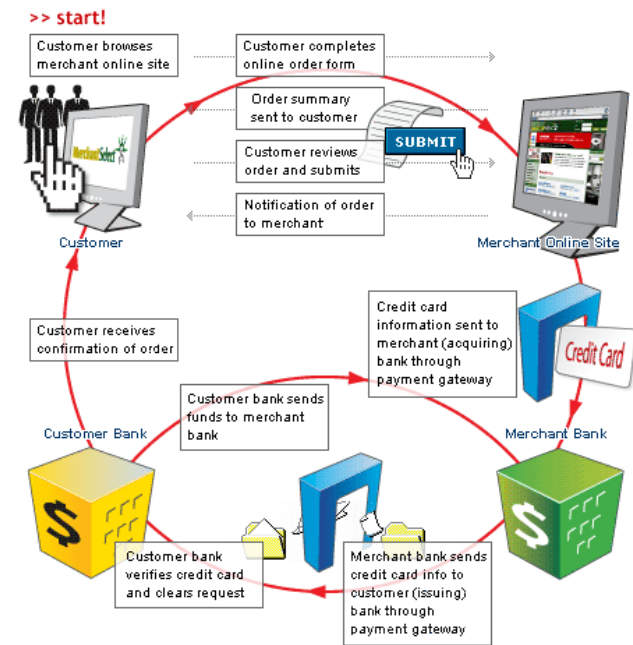
---

- เพื่อให้เห็นว่าระบบย่อยทั้งหมดทำงานร่วมกันได้
- ทดสอบการทำงานของ classes, modules, หรือ subsystems ต่างๆ เมื่อมาประกอบรวมกันทำงานแล้ว
- ถ้ามี API ก็ต้องทดสอบร่วมกับ API ด้วย
- ถ้ามี Database ก็ต้องทดสอบกับ Database ด้วย



# System Test

- เป็นการทดสอบการเชื่อมต่อระหว่างระบบของซอฟต์แวร์ที่พัฒนาขึ้น หรือทดสอบกับระบบอื่นๆ
- ทดสอบระบบการโอนเงิน กับระบบธนาคาร
- ทดสอบระบบการโอนเงิน กับระบบบัญชีผู้ใช้
- Alpha Testing คือ จำลองการทำงานระบบให้เหมือนจริงในฝั่งนักพัฒนา
- Beta Testing หรือ Pilot Testing ทดสอบกับระบบจริงๆ ด้วยสิ่งแวดล้อมจริงก่อนส่งมอบ



# Acceptance Test

---

- ทดสอบระบบจาก Requirement หรือ User Story ของลูกค้า
- ระบบต้องสามารถใช้งานได้จริงและสมบูรณ์ตรงตาม Business Logic ที่ตกลงกันไว้
- ลูกค้า และ/หรือ คนที่ให้ requirement มีส่วนร่วมในการเขียน Test Case และทดสอบ
- ทดสอบทุก Roles ของผู้ใช้
- สภาพแวดล้อม (Hardware, Software, และ Infrastructure) ต้องเหมือนจริงมากที่สุด.
- ตัวอย่าง
  - ผู้ดูแลระบบสามารถนำข้อมูลสินค้าเข้าในระบบ และเมื่อนำสินค้าเข้าสู่ระบบแล้ว ผู้ใช้จะสามารถเห็นข้อมูลของสินค้านั้นได้ และสามารถค้นหาได้

# Performance Testing

---

- **Load testing**

- การทดสอบซอฟต์แวร์หรือระบบว่า ระบบจะมีความเร็วมากน้อยแค่ไหน ภายใต้สภาวะและขนาดของภาระที่คาดว่าจะเกิดขึ้นจริง เช่น หากมีผู้ใช้งานเข้ามาใช้ระบบพร้อมกัน 100 คน ระบบจะตอบสนองเร็วหรือช้าแค่ไหน  
(concurrent users)

- **Stress testing**

- การทดสอบระบบที่นอกเหนือจากการทำงานปกติ เพื่อทดสอบความเสถียร ในความพร้อมและการจัดการข้อผิดพลาด เมื่อระบบมีการทำงานหนัก

- **Spike testing**

- การทดสอบระบบเมื่อมีการเพิ่มจำนวนผู้ใช้งานอย่างรวดเร็ว

- **Soak testing หรือ Endurance testing**

- การทดสอบระบบว่า ระบบยังสามารถทำงานได้ดีหรือไม่ เมื่อมีการใช้งานในเวลานาน throughput and/or response times ยังดีเหมือนกับตอนเริ่มต้นหรือไม่

- **Capacity testing**

- การทดสอบเพื่อกำหนดหาว่า จะมีผู้ใช้กี่คนที่ระบบสามารถรองรับได้ โดยที่ระบบสามารถยังทำงานได้

# Performance Testing

---

- **Recovery testing**

- การทดสอบระบบว่า ระบบสามารถฟื้นตัวจากการล่มได้เร็วหรือดีแค่ไหน

- **Smoke testing**

- การเริ่มต้นทดสอบระบบในการทดสอบประสิทธิภาพ เพื่อดูว่า การระบบสามารถทำงานได้ปกติในสภาวะปกติ

- **Volume testing**

- การทดสอบระบบโดยใช้จำนวนข้อมูล เพื่อแสดงให้เห็นว่า จำนวนข้อมูลเท่าไรที่ระบบไม่สามารถรองรับได้

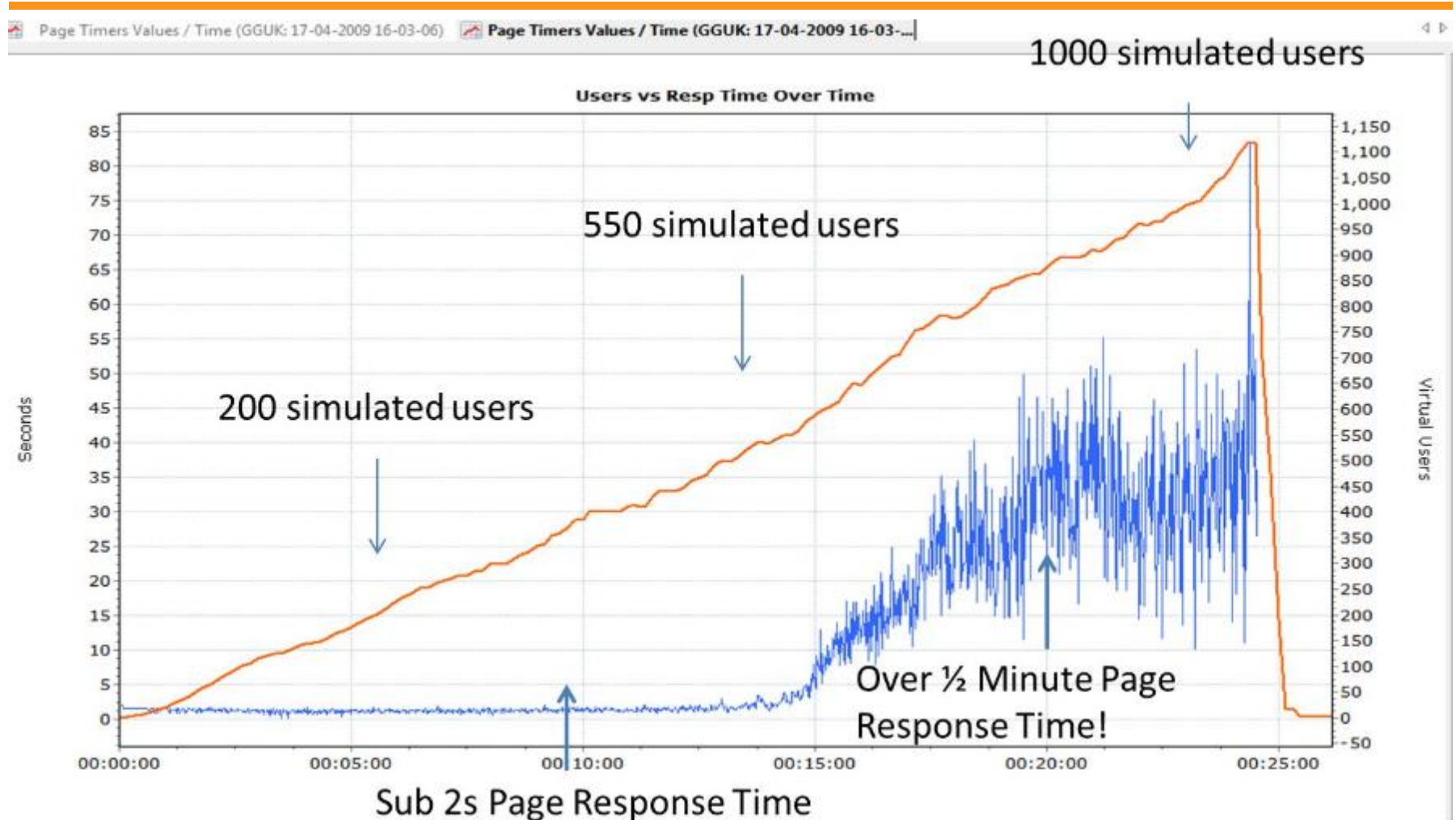
- **Network Sensitivity testing**

- การทดสอบขีดจำกัดของ WAN และ การทำงานของ network สามารถที่จะคาดการณ์ผลกระทบในส่วน of WAN และ การสื่อสารบน bandwidth

- **Scalability testing**

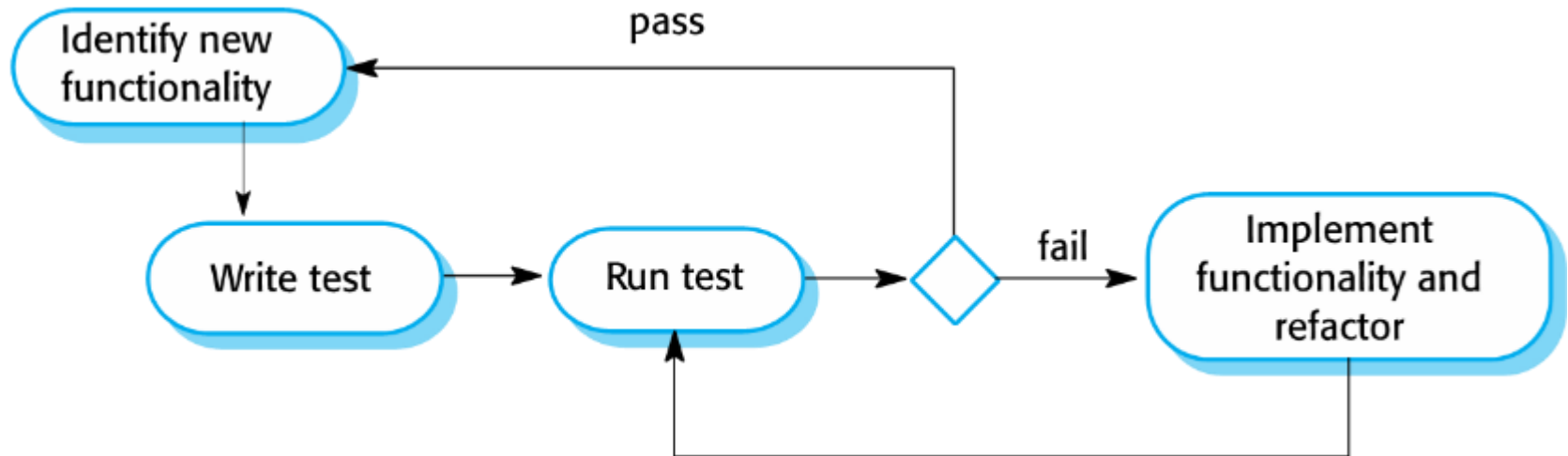
- การทดสอบเพื่อวัดความสามารถในการประยุกต์ใช้เมื่อนำไปใช้กับระบบที่ใหญ่ขึ้น หรือ ระบบอื่นๆที่จะทำไปใช้

# Load Test Report





# Test-Driven Development



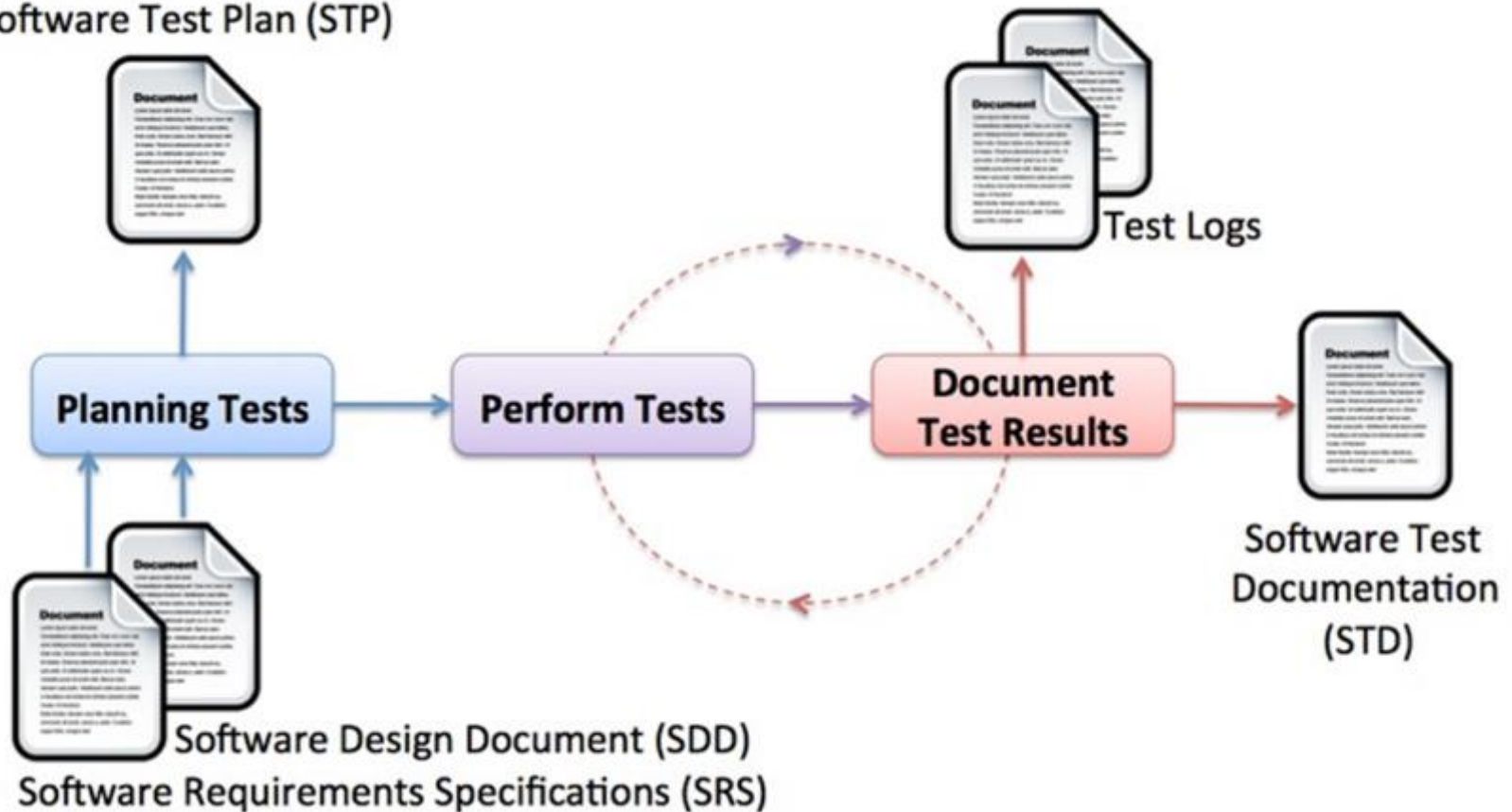
- เป็นการเขียน Test Script ขึ้นมาก่อนแล้วจึงเขียน Code เพื่อให้แต่ละ Test Case ผ่าน
- ประโยชน์ของ Test-Driven Development
  - เขียน Code ได้อย่างมีทิศทาง
  - ครอบคลุมทุก Requirements
  - ไม่ได้ถือว่าเสียเวลา เพราะเหมือนกับการทำ Unit Test ก่อนเริ่มเขียน Code

# Testing Documentation



# Test Documentation

## Software Test Plan (STP)



# Software Test Plan

---

- Introduction
- Test items
- Features to be tested
- Testing approach
- Item pass/fail criteria
- Suspension and resumption
- Deliverables
- Tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Costs and schedule
- Risks and contingencies.

# Software Testing Plan

- ตัวอย่าง: Testing Mobile Business Applications

Approach	Type of Testing	Manual Testing		Automated Testing on Device
		Using Device	Using Emulators	
Standard Testing	Unit Testing	No	Yes	No
	Integration Testing	No	Yes	No
	System Testing	Yes	No	No
	Regression testing	Yes	No	Yes
	Acceptance testing	Yes	No	No
Special type of testing to address specific challenges	Compatibility Testing	Yes	No	Yes
	GUI Testing	Yes	No	No
Type of testing more relevant for enterprise mobile business application	Performance Testing	Yes	No	Yes
	Security Testing	Yes	No	Yes
	Synchronization Testing	Yes	No	No

# Test Case Description

---

- Test items
- Input specifications
- Output specifications
- Environmental needs
- Special procedural requirements/rules
- Intercase dependencies.

# Test Case Description

- Verify the login of Gmail

Project Name:	Google Email	
Module Name:	Login	
Reference Document:	If any	
Created by:	Rajkumar	
Date of creation:	DD-MMM-YY	
Date of review:	DD-MMM-YY	

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Valid Password>	Successful login	Gmail inbox is shown		
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Valid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			

# Test Case Description

- Verify the login of Gmail

TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT
Enter valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login
		2. Enter Password	<Valid Password>	
		3. Click "Login" button		
Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown
		2. Enter Password	<Invalid Password>	
		3. Click "Login" button		
Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown
		2. Enter Password	<Valid Password>	
		3. Click "Login" button		
Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown
		2. Enter Password	<Invalid Password>	
		3. Click "Login" button		




# Test Log

---

- **Description**
  - Test item identification
  - Test environment description
- **Activity and event entries**
  - Date and time
  - Author
  - Test procedure identifier
  - Staff present
  - Pass/fail
  - Error messages generated
  - Environmental information
  - Anomalous events

Test Cases



# Defect Tracking

 Marker / MAR-131

## [Pricing] - Update the price to \$29

[Edit](#) [Comment](#) [Assign](#) [To Do](#) [In Progress](#) [Workflow](#) [Admin](#)

### Details

Type:	 Bug	Status:	<b>TO DO</b> <a href="#">(View workflow)</a>
Priority:	 High	Resolution:	Unresolved
Labels:	None		
Environment:	— Browser Chrome 54.0.2840.71 Screen Size 1920 x 1200 Viewport Size 1607 x 920 Zoom L...		

### Description

**Summary:**  
The price mentioned on the pricing page is not correct

**Steps to Reproduce:**  
Go to the pricing page


**Expected Results**  
The price for the basic plan should be \$29


**Actual Results:**  
The price for the basic plan is currently \$25


—


**Source URL:** <https://www.shopify.com/pricing>

### People

Assignee:  gary  
[Assign to me](#)

Reporter:  Christophe Han

Votes:  0

Watchers:  1 [Stop watching th](#)

### Dates

Created: 1 minute ago

Updated: 1 minute ago

### Agile


[View on Board](#)


### HipChat discussions

Do you want to discuss this issue? [Connect](#)

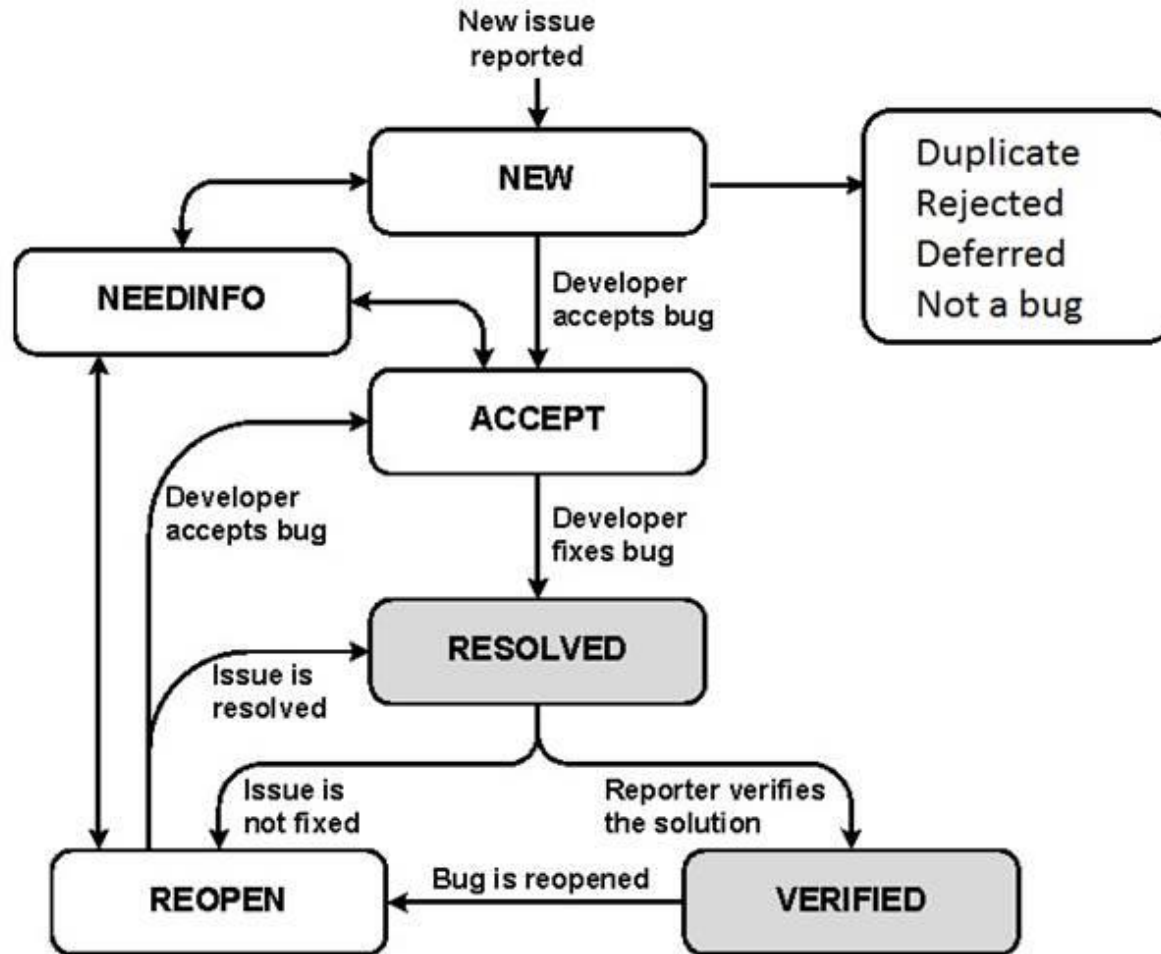
[Connect](#) [Dismiss](#)

### Attachments

 Drop files to attach, or [browse](#).



# Defect Status Flow



# Summary



# Summary

---



- Test Techniques
  - Black Box Testing
  - White Box Testing
- Test Types
  - Unit Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing
  - Performance Testing \*
  - Test-First Development

“

Quality is never an accident;  
it is always the result of  
intelligent effort.

”

John Ruskin