SOFTWARE TESTING
(Kiểm thử phần mềm)

LỚP: CS112.L21.KHCL

GIẢNG VIÊN: NGUYỄN THANH SƠN

Thực hiện:

Phạm Minh Long

Đặng Văn Minh

Trương Thị Kim Thoa

Nội dung

1. Kiểm thử phần mền là gì?
2. Phân loại kiểm thử phần mềm
3. Phân loại kiểm thử động
4. Cách tạo bộ test và một số lưu ý

Kiểm thử phần mền là gì?

Là một tiến trình hay một tập hợp các tiến trình được thiết kế ra để:

- Đảm bảo phần mềm thực hiện đúng theo những thứ mà chúng đã được thiết kế.

-Trong quá trình sử dụng phần mềm không phát sinh bất cứ thứ gì không mong muốn.

Tại sao Kiểm thử phần mền quan trọng?

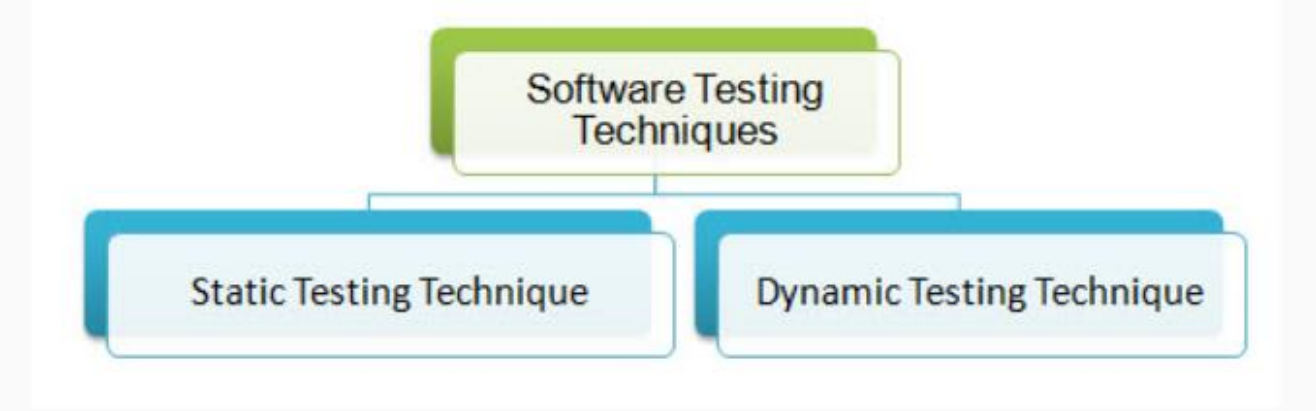
Kiểm thử phần mềm là một phần quan trọng trong quá trình phát triển hệ thống phần mềm:

- Giúp cho khách hàng thấy được sản phẩm đặt ra đã đáp ứng được yêu cầu hay chưa.

- Đảm bảo chất lượng phần mềm khi đưa ra sử dụng. Tránh các rủi ro khi cho khách hàng khi đưa phần mềm vào sử dụng.

- Giảm thời gian và chi phí phát sinh do bảo trì (fix lỗi) cho người viết phần mềm.

Phân loại kiểm thử phần mềm



Kiểm thử tĩnh(static testing)

Kiểm thử tĩnh: là phương pháp kiểm thử phần mềm thủ công.

Phương pháp này thường được những người phân tích thiết kế phần mềm (IT-BA) thực hiện.

Kiểm thử động(dynamic testing)

- Phương pháp thử phần mềm thông qua việc chạy chương trình để theo dõi trạng thái, kết quả của từng bước / toàn bộ các bước trong quá trình thực thi phần mềm.

- Cách thực hiện: dựa trên các ca kiểm thử (test case) xác định bằng sự hoạt động của đối tượng kiểm thử hay toàn bộ chương trình.

- Kiểm tra cách thức hoạt động động của mã lệnh, bao gồm luôn cả kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian.

Kiểm thử động – mục đích

- Trong kiểm thử động, tester quan tâm đến dữ liệu đầu vào và output đầu ra.

- Với mỗi tập dữ liệu đầu vào sẽ phát sinh ra một tập dữ liệu output

- Nếu chương trình nhận đầu vào để “chạy” sau đó đưa ra dữ liệu output giống như yêu cầu. Có thể nói đó là phần mềm đã hoạt động thành công.

Phần loại kiểm thử động

- Phân loại theo chiến lược kiểm thử:

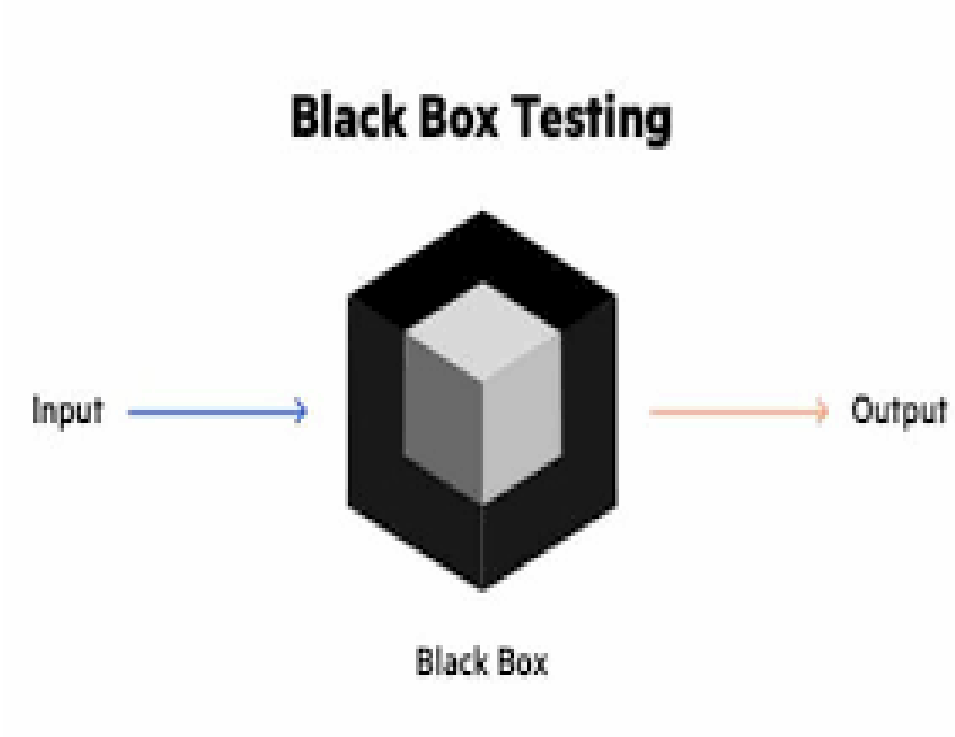
- Black box testing
- White box testing

- Phân loại theo mức độ

- Unit testing
- Integration test
- System test
- Acceptance test
- Release testing

Black box testing

- Xem chương trình như 1 “hộp đen”.
- Kiểm thử dựa trên đặc tả của phần mềm.
- Không quan tâm cấu trúc bên trong của chương trình, tập trung tìm các trường hợp mà chương trình không thực hiện theo đặc tả của nó.

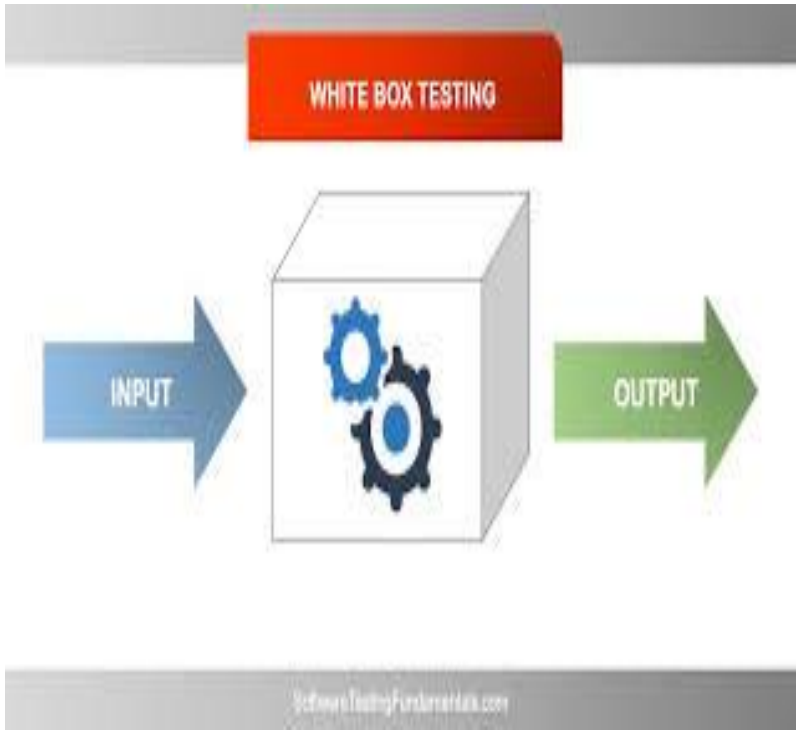


Đặc điểm:

- Không cần biết tới code và cấu trúc chương trình
- Đánh giá chương trình một cách khách quan

White box testing

- Còn được gọi là clear box testing, glass box testing, transparent box testing.
- Thường thiết kế các trường hợp kiểm thử dựa vào cấu trúc bên trong của phần mềm



White box testing – đặc điểm

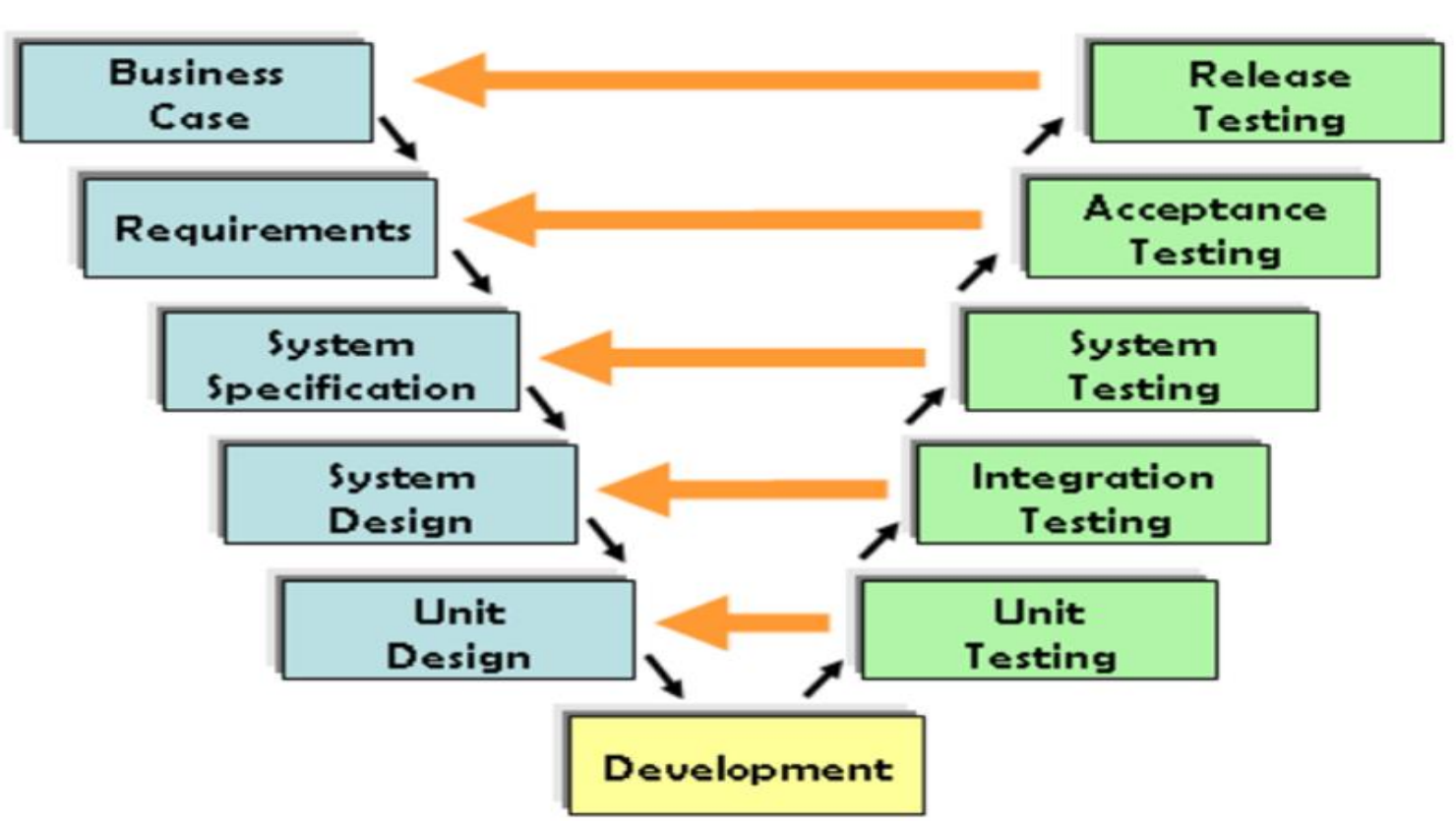
- WBT đòi hỏi kỹ thuật lập trình am hiểu cấu trúc bên trong của phần mềm (các logic nghiệp vụ, luồng dữ liệu, chức năng, kết quả).

- Phương thức: Chọn các đầu vào và xem các đầu ra.

- Phụ thuộc vào các cài đặt hiện tại của hệ thống và của phần mềm, nếu có sự thay đổi thì bài test cũng phải thay đổi theo.

- Được ứng dụng trong các kiểm tra ở cấp độ module (điển hình), tích hợp (có khả năng) và hệ thống của quá trình test phần mềm.

Kiểm thử phần mềm phân loại theo mức độ



Unit test(kiểm tra đơn vị)

Khái niệm: Kiểm thử đơn vị là loại kiểm thử phần mềm trong đó các đơn vị/thành phần đơn lẻ của phần mềm được kiểm tra như: Hàm (Function), Lớp (Class), Phương thức (Method). Kiểm thử đơn vị được thực hiện trong quá trình phát triển ứng dụng. Lỗi ở level này thường được fix ngay sau khi chúng được tìm ra mà không cần lưu lại và quản lý như các test level khác.

- **Sử dụng phương pháp:** Kiểm thử hộp trắng
- **Người thực hiện:** Thường là developer thực hiện

Unit test – mục đích

Tách riêng từng phần để kiểm tra và chứng minh các thành phần đó thực hiện chính xác các yêu cầu chức năng trong đặc tả.

Lỗi được sửa sớm trong chu trình phát triển phần mềm vì vậy tiết kiệm thời gian và chi phí sửa lỗi.

Mã nguồn được tái sử dụng nhiều hơn.

Tăng sự tin tưởng trong việc thay đổi hoặc bảo trì

Mã nguồn đáng tin cậy hơn.

Integration test – kiểm thử tích hợp

Khái niệm: Kiểm thử tích hợp là loại kiểm thử trong đó các module phần mềm hay từng chức năng riêng lẻ được tích hợp logic và được kiểm tra theo nhóm. Mỗi dự án phần mềm gồm nhiều modules, được code bởi nhiều người khác nhau, vì vậy kiểm thử tích hợp tập chung vào việc kiểm tra truyền dữ liệu giữa các module.

Mục đích: Phát hiện lỗi tương tác xảy ra giữa các Unit. Tập chung chủ yếu vào các giao diện và thông tin giữa các module. Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ.

Người thực hiện: Thường là Tester thực hiện

System test – kiểm thử hệ thống

- **Khái niệm:** Kiểm thử hệ thống là kiểm thử toàn bộ chức năng và giao diện của hệ thống.
- **Mục đích:** Đánh giá hệ thống có đáp ứng theo đúng yêu cầu nghiệp vụ, yêu cầu về chức năng đưa ra hay không.
- **Sử dụng phương pháp:** Kiểm thử hộp đen là phổ biến
- **Người thực hiện:** Thường là Tester thực hiện

System test – phân loại

- Kiểm thử chức năng (Functional Test)
- Kiểm thử hiệu năng (Performance Test)
- Kiểm thử bảo mật (Security Test)
- Kiểm thử khả năng phục hồi (Recovery Test)
- Kiểm thử tính khả dụng (Usability Test)

Acceptance test – kiểm thử chấp nhận

- **Khái niệm:** Kiểm thử chấp nhận là kiểm tra xem phần mềm đã thỏa mãn tất cả yêu cầu của khách hàng chưa? Và khách hàng có chấp nhận sản phẩm hay không?
- **Mục đích:** Để nghiệm thu hệ thống trước khi hệ thống được đưa vào hoạt động.
- **Sử dụng phương pháp:** Kiểm thử hộp đen
- **Người thực hiện:** Khách hàng hoặc bên thứ 3

Release test – kiểm thử phát hành

Release testing được thực hiện sau khi triển khai phần mềm lên hệ thống thật.

Các bộ phận liên quan sẽ chuẩn bị tập dữ liệu để kiểm thử trên hệ thống production. Đây là giai đoạn cực kỳ quan trọng, quyết định sản phẩm sẽ đưa ra để khách hàng sử dụng hay hoãn lại (nếu có thể) hoặc rollback lại version trước đó.

UNIT TEST

Cách tạo bộ test (unit test)

Bước 1: Xác định phạm vi dữ liệu của chương trình.

Bước 2: Xác định các đầu mút dữ liệu, dữ liệu phổ quát và các trường hợp đặc biệt có thể xảy của giá trị đầu vào tương ứng với giá trị đầu ra của chương trình cần kiểm tra.

Bước 3: Xây dựng bộ test case

Bước 4: Code chương trình kiểm tra dự trên bộ test case đã có

Một số chú ý

Chúng ta tập trung vào tạo bộ test để kiểm tra tính đúng và hiệu năng của chương trình:

- Kiểm tra tính đúng:

- Xem có bao nhiêu tình huống có thể xảy ra, mỗi tình huống cho ít nhất 1 test case(một số trường hợp cần lưu ý: các biên, trường hợp phổ quát, các trường hợp đặc biệt)
- Sử dụng bất kì ngôn ngữ nào để tạo ra các test case.

- Kiểm tra hiệu năng:

- Nới rộng bộ dữ liệu, làm dữ liệu lớn dần.
- Giới hạn về thời gian và bộ nhớ

Demo Unit test with Python

Add.py

Báo cáo chủ đề > Add.py > ..

```python
def Add(x, y):
    return x + y

```

```python
#Importing library and Add function
import unittest
import random
from Add import Add

#Creating class unittest
class TestAdd(unittest.TestCase):
    def test_1_manual(self):
        with open("Add.in", 'w+') as f_in:          #Creating new file input
            with open("Add.out", 'w+') as f_out:    #Creating new file output
                self.assertEqual(Add(3,5), 8)       #Testing Add function if its work as expected
                f_in.write('3 5')                   #Write down test case to input file
                f_out.write('{}'.format(Add(3,5)))  #Write down result to output file

    def test_2_auto(self):
        with open("Add.in", 'a+') as f_in:
            with open("Add.out", 'a+') as f_out:
                for _ in range(100):                    #Random 100 cases
                    x=random.randint(-100,100)
                    y=random.randint(-100,100)
                    self.assertEqual(Add(x,y), x+y)     #Comparision with a true function
                    f_in.write('\n{} {}'.format(x,y))
                    f_out.write('\n{}'.format(Add(x,y)))

if __name__ == "__main__":
    unittest.main()
    '''You can also run code in terminal. Exp: python -m unittest Add_test'''
```

Bài tập: Nộp file input và output cho bài tập Tảo
Về email:19522295@gm.uit.edu.vn
Định dạng file:
1 folder tên nhóm bên trong gồm có:
-1 folder input bên trong gồm có các file:
+ input1: file txt (tương ứng duy nhất 1 test case)
+input2:
+.....
-1 folder output bên trong gồm có các file:
+ output1: (là output tương ứng cho input1)
+ output:2
+ output.....
Lưu ý: Không giới hạn số lượng input và output số lượng input phải bằng số lượng output
Deadline: 0h:0ph 3/4/2021

TẢO BIỂN

Tảo biển sinh sản rất nhanh khi có môi trường thuận lợi với chúng và có những loài còn tiết ra môi trường những chất độc hại.

Một loại tảo nâu trong môi trường nước bị ô nhiễm nặng sinh sản theo quy luật sau:

- Ngày đầu tiên (ngày 0) có ***n*** cá thể ở mức 1,
- Ở mỗi ngày tiếp theo, mỗi cá thể mức ***i*** sinh ra ***i*** cá thể mức 1, các cá thể mới sinh sẽ sinh sôi, phát triển từ ngày hôm sau.
- Bản thân các cá thể mức ***i*** phát triển thành mức ***i***+1 và chu kỳ phát triển trong ngày chấm dứt.

Hãy xác định sau ***k*** ngày trong nước biển có bao nhiêu cá thể.

Dữ liệu: Vào từ thiết bị nhập chuẩn gồm một dòng chứa 2 số nguyên ***n*** và ***k*** ($1 \le n \le 1000$, $1 \le k \le 10^5$).

Kết quả: Đưa ra thiết bị xuất chuẩn một số nguyên – số lượng cá thể tảo theo mô đun 10^9+7.

Tài liệu tham khảo

1. Guru99[https://www.guru99.com/software-testing.html?fbclid=IwAR0mtAt6d0BJRCW5Vbi6RhlL-UiQyqAbnvCChdG4Ao-WEqPjG_QxQYmo_sk]

2. Viblo[https://viblo.asia/p/cac-test-level-trong-kiem-thu-phan-mem-djeZ14AJKWz?fbclid=IwAR0rV-Z7C9e6ATysHgP0drtYRmJXOH4wAdez4LbT6kahdaSWnaAYfhb26Qg]

3. codelearn[https://codelearn.io/sharing/toan-tap-kiem-thu-phan-mem?fbclid=IwAR2_KCYuqQnk1ylP7DwhJUeorSP8p7Lj5elQAECycLkWYiUcHFbFt2VK9gU]

Hết