HUTECH
Đại học Công nghệ Tp.HCM

KIỂM THỬ PHẦN MỀM

Bài 3:
KỸ THUẬT THIẾT KẾ TEST CASE

Thời gian: 3 tiết

NỘI DUNG

3.1. CÁC KHÁI NIỆM CHÍNH

- ✓ Thiết kế test–case: Là quá trình xây dựng các phương pháp kiểm thử để phát hiện lỗi của phần mềm.
- ✓ Thiết kế test-case quan trọng vì :
  - Các ca kiểm thử tốt có khả năng phát hiện ra lỗi nhiều nhất.
  - Tạo các ca kiểm thử có chi phí rẻ, tốn ít thời gian và công sức nhất.
- ✓ Chiến lược kiểm thử hợp lý là kết hợp sức mạnh của phương pháp Kiểm thử hộp đen và kiểm thử hộp trắng
- ✓ Sử dụng phương pháp thiết kế Tast case hướng hộp đen và bổ sung thêm kiểm thử tính logic chương trình, với phương pháp hộp trắng.

3.1. CÁC KHÁI NIỆM CHÍNH

Các thành phần dữ liệu Test case:

- ✓ Test case là tài liệu tập hợp dữ liệu đầu vào, các điều kiện để thực hiện với kết quả mong đợi.
- ✓ Người kiểm thử chạy chương trình cho mỗi test theo tài liệu được cung cấp, và so sánh kết quả thực tế với kết quả mong đợi.
- ✓ Khi kết quả không phù hợp với các kết quả mong đợi, một lỗi được xác định.
- ✓ Phương pháp phân vùng các lớp tương đương được áp dụng hiệu quả trong hộp đen.
- ✓ Hai tài liệu cần thiết cho việc thiết kế testcase:
  - ✓ Đặc tả chức năng module
  - ✓ Mã nguồn của module

3.1. CÁC KHÁI NIỆM CHÍNH

Thủ tục thiết kế testcase:

- ✓ Phân tích module dựa vào kỹ thuật kiểm thử hộp trắng.
- ✓ Áp dụng kỹ thuật kiểm thử hộp đen vào đặc tả của module để bổ sung thêm các testcase khác.

Để thực hiện kiểm thử các module, chú ý 2 điểm chính:

[1] Làm sao thiết kế được tập các testcase hiệu quả.

[2] Cách thức và thứ tự tích hợp các module:

- Viết testcase cho module nào
- Dùng loại tiện ích nào cho kiểm thử.
- Coding và kiểm thử các module theo thứ tự nào.
- Chi phí tạo ra các testcase.
- Chi phí debug để tìm và sửa lỗi.

3.1. CÁC KHÁI NIỆM CHÍNH

Có 2 phương án để kiểm thử các module:

1. Kiểm thử không tăng tiến (Nonincremental testing): kiểm thử các module chức năng độc lập, sau đó kết hợp chúng lại để tạo ra chương trình.

2. Kiểm thử tăng tiến (Incremental testing): Kết hợp module cần kiểm thử vào bộ phận đã kiểm thử (lúc đầu là null) để kiểm thử module cần kiểm thử trong ngữ cảnh.

3.2.THIẾT KẾ TEST CASE THEO WHITE-BOX

3.2.1. Tổng quan về kiểm thử hộp trắng

3.2.2. Ưu điểm/nhược điểm kiểm thử hộp trắng

3.2.3. Các mức độ áp dụng

3.2.1. TỔNG QUAN VỀ KIỂM THỬ HỘP TRẮNG

- ✓ Kiểm thử Hộp Trắng/Clear Box Testing/Open Box Testing/Glass Box Testing/Transparent Box Testing, Code-Based Testing/Structural Testing
- ✓ Là phương pháp kiểm thử về cấu trúc thiết kế.
- ✓ Tester chọn đầu vào để thực hiện thông qua mã và xác định đầu ra thích hợp.
- ✓ Dựa vào thuật giải cụ thể, vào cấu trúc dữ liệu để xác định đơn vị đó có thực hiện đúng.
- ✓ Tester cần có kỹ năng, kiến thức về ngôn ngữ lập trình, về thuật giải trong phần mềm.

3.2.2. ƯU ĐIỂM/NHƯỢC ĐIỂM KIỂM THỬ HỘP TRẮNG

Ưu điểm

- ✓ Test ở giai đoạn sớm hơn, không đợi có GUI để test
- ✓ Test kỹ càng hơn, bao phủ hầu hết các đường dẫn
- ✓ Thích hợp trong tìm lỗi và các vấn đề trong mã lệnh
- ✓ Cho phép tìm các lỗi ẩn bên trong
- ✓ Các lập trình viên có thể tự kiểm tra
- ✓ Giúp tối ưu việc mã hoá
- ✓ Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.

3.2.2. ƯU ĐIỂM/NHƯỢC ĐIỂM KIỂM THỬ HỘP TRẮNG

Nhược điểm:

- ✓ Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình và thực hiện.
- ✓ Maintenance test script có thể là một gánh nặng nếu thể hiện thay đổi quá thường xuyên.
- ✓ Vì phương pháp này liên quan với ứng dụng đang được test, nên các công cụ để phục vụ có thể không sẵn có.

3.2.3. CÁC MỨC ĐỘ ÁP DỤNG

Kiểm tra Hộp trắng áp dụng cho các mức độ sau:

- ✓ Unit Testing (Kiểm thử đơn vị): Để kiểm tra đường dẫn trong một đơn vị.
- ✓ Integration Testing (Test tích hợp): Để kiểm tra đường dẫn giữa các đơn vị.
- ✓ System Testing (Test hệ thống): Để kiểm tra các đường dẫn giữa các hệ thống con.

Tuy nhiên, chủ yếu áp dụng cho các kiểm thử đơn vị.

3.3.THIẾT KẾ TEST CASE THEO BLACK-BOX

3.2.1. Tổng quan về kiểm thử hộp đen

3.2.2. Ưu điểm/nhược điểm kiểm thử hộp đen

3.2.3. Qui trinh kiểm thử hộp đen

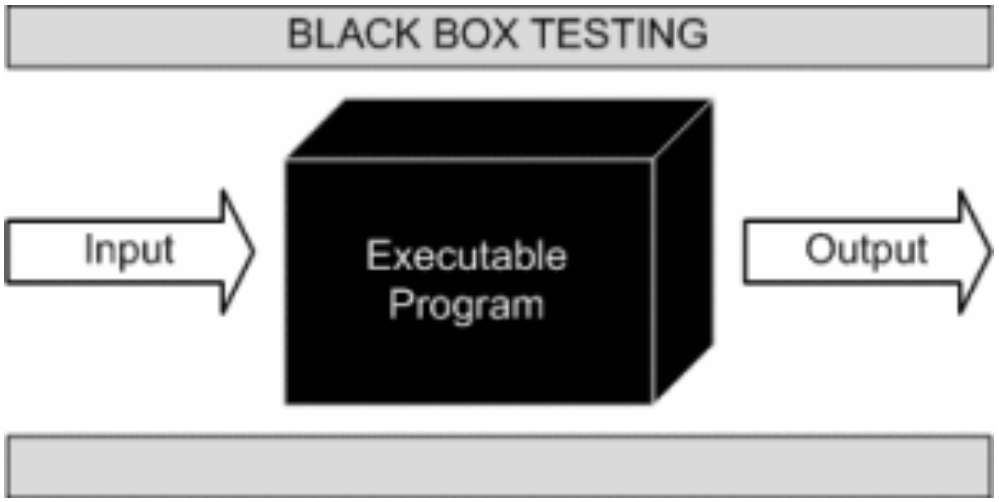
3.3.1. TỔNG QUAN VỀ KIỂM THỬ HỘP ĐEN

- ✓ Phương pháp kiểm thử được thực hiện khi không biết cấu tạo bên trong của phần mềm, tester xem hệ thống như chiếc hộp đen hay kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.
- ✓ Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình.
- ✓ Các tester không dùng kiến thức về cấu trúc lập trình bên trong hệ thống, xem hệ thống là một cấu trúc hoàn chỉnh, không thể can thiệp được.

3.3.1. TỔNG QUAN VỀ KIỂM THỬ HỘP ĐEN

Black Box Testing thực hiện trong Function test và System test. Cố gắng tìm ra các lỗi sau:

- ✓ Chức năng không chính xác hoặc thiếu.
- ✓ Lỗi giao diện.
- ✓ Lỗi trong cấu trúc DL hoặc truy cập CSDL bên ngoài.
- ✓ Hành vi hoặc hiệu suất lỗi.
- ✓ Khởi tạo và chấm dứt các lỗi.



3.3.2. ƯU ĐIỂM/NHƯỢC ĐIỂM KIỂM THỬ HỘP ĐEN

Ưu điểm:

- ✓ Thực hiện từ quan điểm của người dùng.
- ✓ Các tester không có "mối ràng buộc" nào với code.
- ✓ Tester có thể không phải IT chuyên nghiệp, không cần phải biết ngôn ngữ lập trình.
- ✓ Các tester có thể được thực hiện độc lập với developer, khách quan và tránh thiên vị.
- ✓ Hệ thống thật sự với toàn bộ yêu cầu được kiểm thử chính xác.
- ✓ Thiết kế Test case kiểm thử khá nhanh

3.3.2. ƯU ĐIỂM/NHƯỢC ĐIỂM KIỂM THỬ HỘP ĐEN

Nhược điểm của kiểm thử hộp đen

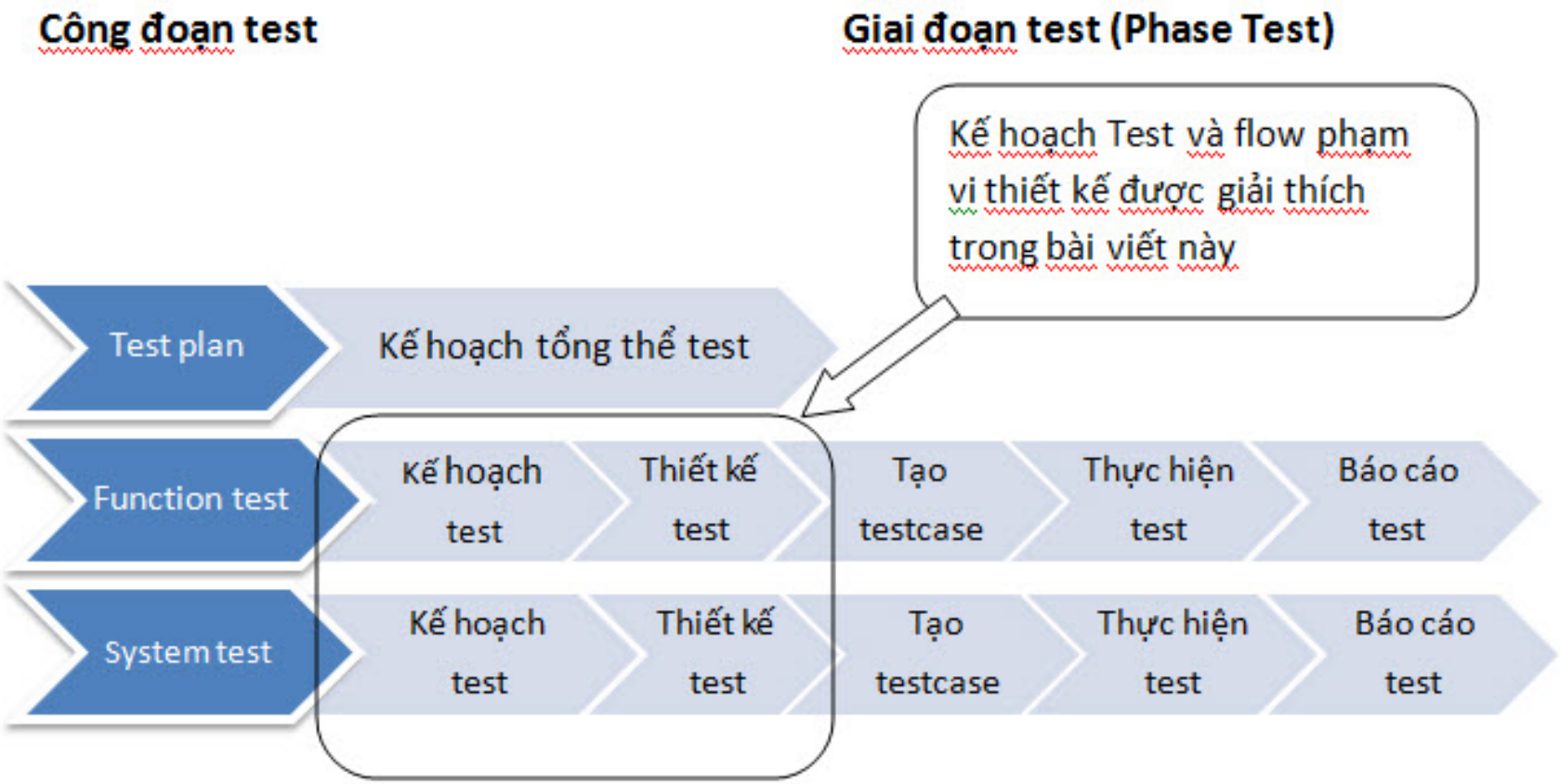
- ✓ Dữ liệu đầu vào khối lượng khá lớn
- ✓ Dự án không có thông số rõ ràng thì việc thiết kế test case rất khó.
- ✓ Chỉ có một số nhỏ các đầu vào được kiểm tra.
- ✓ Kiểm thử black box như "là đi trong mê cung tối không mang đèn" vì tester không biết phần mềm đang test đã được xây dựng như thế nào.
- ✓ Có nhiều trường hợp khi một tester viết rất nhiều trường hợp test những vẫn không phát hiện test hết

3.2.3. QUI TRINH KIỂM THỬ HỘP ĐEN

Qui trình kiểm thử hộp đen tổng quát gồm các bước:

- ✓ Phân tích đặc tả về chức năng phần mềm cần thực hiện.
- ✓ Dùng kỹ thuật định nghĩa các testcase để định nghĩa testcase. là xác định 3 thông tin sau :
  - Giá trị dữ liệu nhập.
  - Trạng thái cần có để thực hiện testcase.
  - Giá trị dữ liệu xuất mà phần mềm phải tạo được.
- ✓ Kiểm thử các testcase đã định nghĩa.
- ✓ So sánh kết quả thu được với kết quả kỳ vọng trong từng testcase, từ đó lập báo cáo về kết quả kiểm thử.

3.2.3. QUI TRINH KIỂM THỬ HỘP ĐEN



Phase test trong công đoạn test

3.2.3. QUI TRINH KIỂM THỬ HỘP ĐEN

- ✓ **Kế hoạch test**: Chỉ rõ mục đích và phạm vi của công đoạn test.
- ✓ **Thiết kế test**: Quyết định sẽ sử dụng gì cho mục đích.
- ✓ **Tạo testcase**: Tạo document ghi trạng thái trước khi bắt đầu test và **kết quả mong đợi**

3.2.3. QUI TRINH KIỂM THỬ HỘP ĐEN

- ✓ **Thực hiện test**: Vừa xem testcase vừa cho chạy phần mềm, đánh dấu kết quả pass hoặc fail vào cột trạng thái testcase (trường hợp fail thì tạo bản báo cáo lỗi: trình bày nội dung mô tả hiện tượng khác với kết quả và hiện tượng đó phát sinh trong trường hợp như thế nào như: thao tác, điều kiện,...)
- ✓ **Báo cáo test**: Tóm tắt kết quả mục thực hiện, hiệu quả của việc test, ..và **dữ liệu lỗi** (số lỗi được tìm ra, số lỗi theo mức độ quan trọng,...) để đánh giá xem có thỏa mãn tiêu chuẩn pass/ fail của test không. Ngoài ra cũng đề xuất thêm rủi ro có thể sinh ra

3.4. CÁC HỆ THỐNG QUẢN LÝ TESTCASE

3.4.1. Giới thiệu

3.4.2. Testcase

3.4.3. Các thành phần testcase

3.4.1. GIỚI THIỆU

- ✓ "Excel - Google sheet“ là tool được sử dụng ở nhiều công ty, nhiều dự án, tốt cho những project vừa và nhỏ, nhóm có 1,2 tester/QA.

3.4.1. GIỚI THIỆU

- ✓ Với những project lớn, thì phải dùng test case management tool.
  - Công cụ để quản lý test case, trong phần mềm quản lý dự án, như dùng Git để quản lý code, dùng Jira để quản lý task....):
    - Miễn phí: TestLink...
    - Tính phí: Testrail, Testlodge
  - Visual Studio Team Service hay Terik quy tất cả về một mối", từ source code, test case, document, process, automation test, CI/CD,..

TestLink

TestRail

3.4.2. TESTCASE

- ✓ Sau khi có requirement, test plan thì tester/QA xây dựng bộ test case cho phần mềm.
- ✓ Test case là thể hiện được cách hoạt động của chức năng. Một test case tốt là một test case biết chức năng đó đang hoạt động đúng hay sai.
- ✓ Bộ test case như một checklist, mỗi khi test chỉ cần dựa trên checklist, không bỏ sót trường hợp.
- ✓ Test case là một tài liệu cực kì quan trọng, Tester mới nhận việc sẽ dựa vào test case để tiếp tục công việc.
- ✓ Vậy nên việc xây dựng một test case hiệu quả là cực kì quan trọng, test case phải chi tiết, chính xác và được cập nhật liên tục.

3.4.2. TESTCASE

- ✓ Việc có bộ test case sẽ giúp tạo test dễ dàng, một checklist có các case, sẽ dễ dàng lọc ra những bug đang có trong phần mềm.
- ✓ Từ bộ test case cho biết độ bao phủ đã tốt hay chưa. Mỗi *khi phát sinh bug mà flow đó chưa có trong test case thì hãy bổ sung ngay*.
- ✓ Không thể xây dựng một bộ test case hoàn chỉnh ngay từ đâu, test case được bổ sung trong quá trình phát triển phần mềm
- ✓ Một test case viết tốt thì khi chuyển qua automation, build script sẽ nhanh và tốn ít thời gian hơn.

3.4.3. CÁC THÀNH PHẦN TESTCASE

- ✓ **Test case ID**: ID để dễ gọi tên (*case của chức năng "Sign Up" thì sẽ là SU_001)*
- ✓ **Test case summary**: Tóm tắt, giúp nhận diện được mục đích của test case (VD: *"Verify that user can sign up with valid data successfully“)*
- ✓ **Test case description**: Mô tả chi tiết test case dùng để làm gì (tương tự "Test case summary", có thể bỏ qua)
- ✓ **Pre-conditions**: Điều kiện trước khi Test như: môi trường test, test data,...(*VD kiểm tra "Verify sign in function when there's no internet connection" thì Pre-conditon là: "1. There's no internet connection on device. 2. User is in sign in screen“)*

3.4.3. CÁC THÀNH PHẦN TESTCASE

- ✓ **Test data**: Dữ liệu Test, có test case cần test data, hoặc không.
- ✓ **Reproduce Steps**: Mô tả lại từng bước để thực hiện test case (rất quan trọng và chiếm nhiều thời gian khi viết test case). "Đừng giả định bất cứ điều gì khi viết, hãy nghĩ rằng đây là lần đầu tiên thao tác, thì test case mới rõ được.
- ✓ **Expected result**: Tương ứng với một "reproduce steps" sẽ có một expected result, không nhất thiết là bước nào cũng phải có. (vd: test case "Verify that user can sign up successfully". Có steps "Press "Sign In" button" thì tương ứng phải phải có expected result "Sign in successfully. User go to home screen“

3.4.3. CÁC THÀNH PHẦN TESTCASE

- ✓ **Observed/Actual result**: kết quả thực tế. Thường thì phần này ta sẽ thêm vào khi run test case.
- ✓ **Test case status:** Kết quả khi "run" một test case: Pass/Fail/Blocked/Skipped
- ✓ **Bug ID**: Nếu test case bị failed thì sẽ tạo một ticket bug và thêm vào bug ID.
- ✓ **Environment**: Môi trường test (như Dev, Staging, UAT, Production)
- ✓ **Notes/Comment**: Lưu ý/phản hồi ghi lại để xem lại

HUTECH
Đại học Công nghệ Tp.HCM

KIỂM THỬ PHẦN MỀM

Bài 3:
KỸ THUẬT THIẾT KẾ TEST CASE

Q&A