Rubrics cuối kỳ

TRẮC NGHIỆM

Tổng quan kiểm thử phần mềm: 1.1, 1.2, 1.3 - Chương 1 (25%)

* Trình bày được mục đích kiểm thử phần mềm, các thuật ngữ, các giai đoạn và các chiến lược kiểm thử phần mềm - 1.1 -
* Trình bày các loại tài liệu quan trọng trong kiểm thử phần mềm - 1.2
* Trình bày các kỹ thuật kiểm thử hộp đen và kiểm thử hộp trắng - 1.3
* *Trình bày được tầm quan trọng và quy trình kiểm thử tự động - 1.4*
  + Hiểu các khái niệm, thuật ngữ, vai trò và quy trình kiểm thử phần mềm.
  + Hiểu tầm quan trọng của từng loại tài liệu kiểm thử phần mềm.
  + Hiểu các giai đoạn và chiến lược tiếp cận kiểm thử phần mềm.
  + Hiểu các kỹ thuật trong kiểm thử hộp đen, kiểm thử hộp trắng và kiểm thử hộp xám.

Kiểm thử hộp đen: 2.1 - Chương 2 (20%)

* Phân tích yêu cầu để thiết kế test case và dự đoán lỗi tiềm ẩn - 2.1

Kiểm thử hộp trắng: 2.1 - Chương 3 (10%)

* Phân tích yêu cầu để thiết kế test case và dự đoán lỗi tiềm ẩn - 2.1

Kiểm thử tự động: 1.1, 1.2

* Hiểu quy trình và tầm quan trọng kiểm thử tự động - 1.4

TỰ LUẬN

**Phân tích được đầy đủ mã nguồn và thiết kế được các test case tương ứng.**

Kiểm thử hộp đen: 2.1, 2.2, 2.3

* Phân tích yêu cầu để thiết kế test case và dự đoán lỗi tiềm ẩn - 2.1
* *Viết Unit Test thành thạo với ngôn ngữ Java, C# - 2.2*
* *Lập trình sử dụng Selenium để kiểm thử các ứng dụng web 2.3*
* Viết báo cáo kết quả kiểm thử - 2.4

Kiểm thử hộp trắng: 2.1, 2.2, 2.3

* Phân tích yêu cầu để thiết kế test case và dự đoán lỗi tiềm ẩn - 2.1
* *Viết Unit Test thành thạo với ngôn ngữ Java, C# - 2.2*
* *Lập trình sử dụng Selenium để kiểm thử các ứng dụng web 2.3*
* Viết báo cáo kết quả kiểm thử - 2.4

Nội dung

# CÁC QUY TRÌNH KIỂM THỬ PHẦN MỀM:

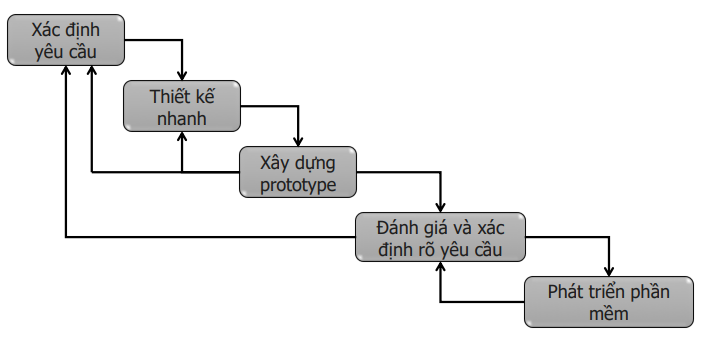
Phân loại quy trình phát triển phần mềm:

* Quy trình có kế hoạch (plan-driven process): tất cả các hoạt động đã được lên kế hoạch trước; tiến độ được xác định thông qua kế hoạch này
* Quy trình phát triển nhan (agile process): các kế hoạch tăng trưởng dần; linh hoạt với các thay đổi của khách hàng

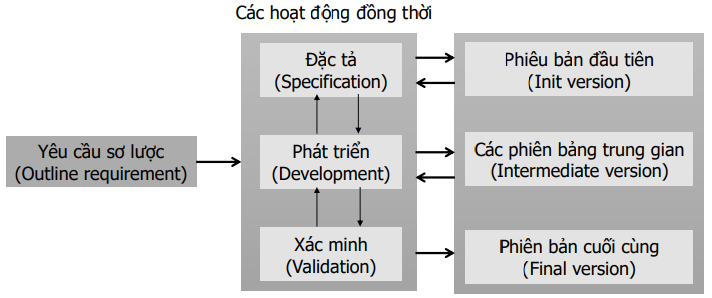
→ Thường kết hợp 2 quy trình trong thực thế

Giới thiệu các quy trình phát triển phần mềm phổ biến:

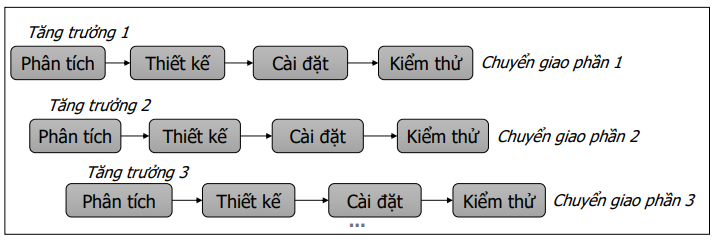
* Quy trình thác nước: các giai đoạn Xác định yêu cầu, Phân tích, Thiết kế, Cài đặt, Kiểm thử, Triển khai được thực hiện tuần tự, nối tiếp; xong cái này mới đến cái tiếp theo, không được đi ngược → **gặp khó khăn khi muốn thay đổi một giai đoạn đã qua**
* Quy trình prototype: phát triển dựa trên yêu cầu hệ thống, có sự tương tác cao với end-user; tuy nhiên **không hướng tới mục tiêu tái sử dụng** trong giai đoạn phát triển phần mềm

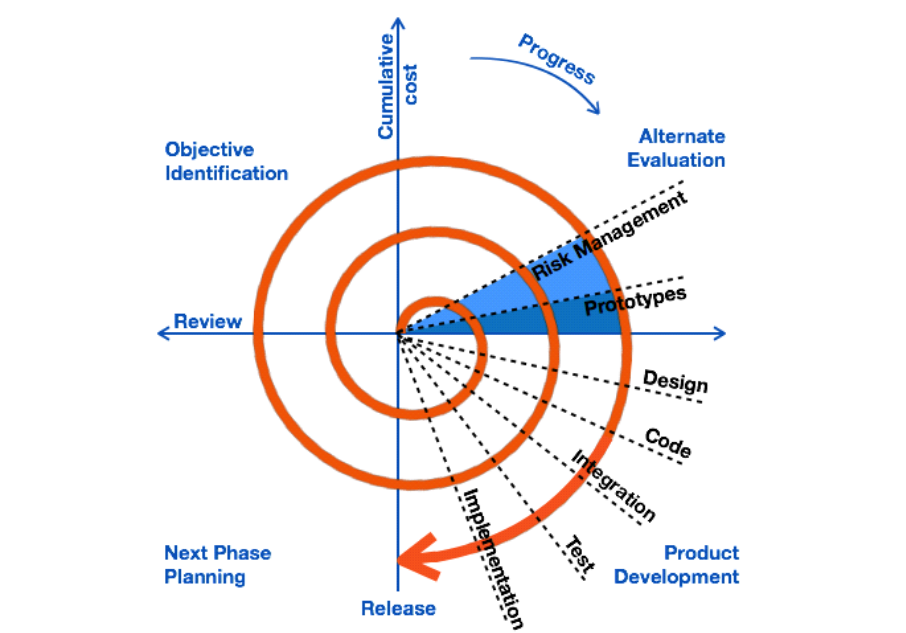


* Quy trình tiến hóa: như quy trình prototype nhưng có nhiều phiên bản prototype liên tiếp nhau để tái sử dụng, cải thiện chương trình sau nhiều chu trình phát triển



* Quy trình tăng trưởng: chia hệ thống thành các phần nhỏ để phát triển, sau khi hoàn thành Feature 1 rồi tiếp tục phát triển Feature 2 **(không đụng vào feature đã hoàn thành nữa)** đến Feature cuối cùng; **phù hợp với các đặc tả chức năng độc lập**



* Quy trình xoắn ốc: **mô hình tập trung phân tích rủi ro** và đề xuất kế hoạch giải quyết rủi ro đó thông qua nhiều chu trình con được nối tiếp nhau; ít được sử dụng
* Quy trình Scrum: phát triển phần mềm theo mô hình linh hoạt với nguyên tắc chính là chia dự án thành nhiều sprint (vòng lặp). Mỗi sprint mất 2-4 tuần (30 ngày)
  + Các thành phần trong 1 Scrum:
* Scrum Master: phân chia sprint, giải quyết trở ngại, xung đột
* Team: các thành viên có vai trò như nhau, không có leader
* Project Owner: thường là khách hàng
* Đặc trưng của chu trình Scrum:
* Product Backlog: danh sách chức năng từ Project Owner
* Sprint Backlog: chọn 1 danh sách chức năng nhỏ cho 1 sprint
* Daily meeting: Họp 15 phút để báo cáo và đánh giá nhanh mỗi ngày
* Khi thực hiện kiểm thử trong quy trình Scrum cần biết điểm dừng (dự án không thể tiếp tục phát triển khi thiếu tài nguyên)

**Câu hỏi ôn tập:**

* **Quy trình phát triển phần mềm Scrum thích hợp với những dự án nào?** Có nhiều thay đổi và yêu cầu tốc độ cao.
* **Trong quy trình Scrum, ai là người quyết định thứ tự ưu tiên các nhiệm vụ trong product backlog?** Product Owner
* **Trong quy trình Scrum, ai là người quyết định thứ tự ưu tiên các nhiệm vụ trong sprint backlog?** Team (Scrum Master là người hỗ trợ, đảm bảo quá trình Scrum diễn ra suôn sẻ, tránh trở ngại

# TỔNG QUAN KIỂM THỬ PHẦN MỀM

## **Định nghĩa:**

* Kiểm thử (testing) là quá trình đánh giá hệ thống hoặc các thành phần của hệ thống, nhằm kiểm tra nó đúng với bản đặc tả yêu cầu và tìm ra lỗi trước khi đưa vào sử dụng; kiểm thử thực thi chương trình với dữ liệu tạo trước

## Người thực hiện kiểm thử phần mềm:

* Tester: kiểm thử chức năng, hiệu năng, bảo mật,...
* Developer: kiểm thử đơn vị (unit testing) và kiểm thử tích hợp (integration testing)
* Project leader/Manager: lập kế hoạch kiểm thử, đánh giá kết quả kiểm thử, và đảm bảo rằng quy trình được thực hiện đầy đủ
* End-user: tham gia kiểm thử để đảm bảo phần mềm hoạt động như mong đợi của họ trong môi trường thực tế

## Kiểm thử bắt đầu và kết thúc:

* **Bắt đầu:** Càng sớm càng tốt vì càng để lâu, lỗi càng tăng
* **Kết thúc kiểm thử khi:**
* Hết hạn kiểm thử
* Đã hết test case cần thực thi
* Tỷ lệ lỗi có thể chấp nhận được
* Không còn chức năng để kiểm thử

Nếu phần mềm không phát hiện lỗi khi kiểm thử khả năng rơi vào một trong hai trường hợp:

* Phần mềm quá tốt, không tồn tại lỗi trong phần mềm
* Tester quá tệ, không thể phát hiện lỗi trong phần mềm

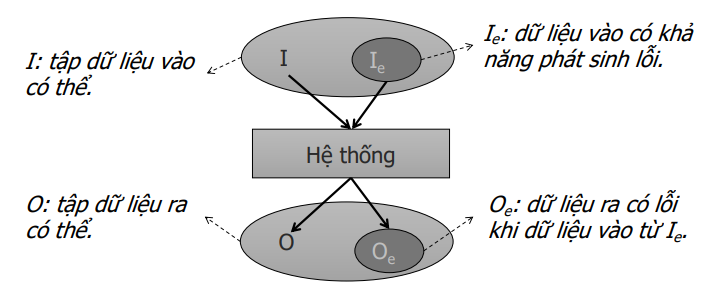
# MỤC ĐÍCH CỦA KIỂM THỬ PHẦN MỀM

* Đảm bảo và chứng minh nhóm phát triển đã phát triển đúng với yêu cầu khách hàng (validation testing: kiểm thử xác nhận)
* Phát hiện ra lỗi, thiếu sót hoặc những hoạt động của phần mềm không đúng (defect testing: kiểm thử lỗi) với đặc tả của nó

# MỘT SỐ THUẬT NGỮ:

## Validation testing và Defect testing:

* Validation testing: kiểm thử đảm bảo hệ thống hoạt động như mong đợi trong môi trường thực tế (kiểm thử với dữ liệu đúng không nằm trong Ie để hệ thống cho đầu ra dữ liệu mong muốn)
* Defect testing: kiểm thử đảm bảo phần mềm không có lỗi, có ít lỗi hoặc giảm số lượng lỗi được giảm thiểu tối đa (kiểm thử với dữ liệu trong Ie mà hệ thống có thể sinh ra lỗi)



**Error (lỗi):** lỗi do con người gây ra

**Fault (sai sót):** những sai sót do dư hoặc thiếu hoặc không đúng các yêu cầu phần mềm cần thực hiện.

**Failure (hỏng):** xảy ra khi các sai sót (fault) được thực thi (kết quả của fault)

**Incident (biến cố):** là những trường hợp mà phần mềm có những **thực thi đáng nghi ngờ** (không nằm trong kế hoạch), nguyên nhân của nó có thể là do *cấu hình môi trường kiểm thử không đúng*, *dữ liệu dùng kiểm thử sai* hoặc *do lỗi của kiểm thử viên*.

**Stub:** một mẫu code để mô phỏng hoạt động của **thành phần đang thiếu.**

**Driver:** một mẫu code để truyền test case (dữ liệu kiểm thử) đến mẫu code khác (module cần kiểm thử), nhận kết quả và xuất ra.

***Ví dụ:*** *Trong kiểm thử tích hợp, khi một module mới được thêm vào hệ thống và cần được kiểm thử, một driver được tạo ra để gọi module mới và cung cấp dữ liệu đầu vào để kiểm tra chức năng của nó.*

**Quality Control (QC):** là những hoạt động, kỹ thuật như thực hiện kiểm tra, kiểm thử trên sản phẩm để đảm bảo chất lượng sản phẩm; tìm ra lỗi để kịp thời sửa đổi

**Quality Assurance (QA):** là những kế hoạch, hoạt động nhằm đảm bảo quy trình phát triển sản phẩm đang được thực hiện chính xác; giảm thiểu số lượng lỗi xuất hiện

## Verification & Validation (V&V):

V&V là một dãy các hoạt động trong QA, kiểm thử là một trong những hoạt động đó và đóng vai trò cực kì quan trọng

* **Verification:** quá trình đảm bảo sản phẩm phần mềm đúng với đặc tả của nó, do dev thực hiện
* **Validation:** quá trình đảm bảo sản phẩm phần mềm đáp ứng được yêu cầu người dùng, do tester thực hiện

# VÒNG ĐỜI KIỂM THỬ PHẦN MỀM

Yêu cầu Acceptance test

Đặc tả System test

Thiết kế Integration test

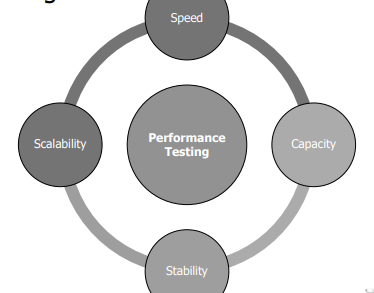
Thi hành (dev) Unit test

# CÁC GIAI ĐOẠN KIỂM THỬ

* **Development testing:** Hệ thống được ***dev, nhóm phát triển*** kiểm thử trong suốt quá trình phát triển để phát hiện lỗi (bugs) hoặc thiếu sót (defects) trong chương trình
* **Release testing: *một nhóm tester riêng biệt*** sẽ kiểm thử một phiên bản đã hoàn tất của hệ thống trước khi nó được triển khai đến người dùng
* **User testing:** ***khách hàng hoặc người dùng tiềm năng*** của hệ thống sẽ kiểm thử hệ thống trong một môi trường riêng của họ để nhận phản hồi về
  + Tính tin cậy
  + Hiệu năng
  + Tiện dụng
  + Mạnh mẽ

# CÁC MỨC ĐỘ KIỂM THỬ

* **Unit test (kiểm thử đơn vị):** kiểm tra trên từng đơn vị đơn giản (thường là hàm, phương thức hoặc lớp), do dev trực tiếp thực hiện
* **Integration test (kiểm thử tích hợp):** kiểm tra sự tương tác giữa các thành phần riêng lẻ đã được tích hợp. Các kiểm thử trong kiểm tra tích hợp:
  + **Kiểm thử cấu trúc (structural)**
  + **Kiểm thử chức năng (functional)**
  + **Kiểm thử hiệu năng (performance)**
* **System test (kiểm thử hệ thống):** kiểm thử toàn bộ hệ thống sau khi đã được tích hợp đầy đủ nhằm đảm bảo phần mềm tuân thủ các đặc tả yêu cầu. Các kiểm thử trong kiểm thử hệ thống:
  + **Kiểm thử hiệu năng (performance):** tối ưu hiệu năng, khả năng chịu tải nhằm xác định một số vấn về về thắt cổ chai (bottleneck) hoặc hiệu năng



* **Load test:** 1 phần của kiểm thử hiệu năng nhằm kiểm tra khả năng chịu tải bằng cách chạy hàng loạt test, tăng dần sức tải đến khi hệ thống không thể thực thi được nửa (test case thường được thiết kế xung quanh giá trị biên)
* **Stress test:** 1 phần của kiểm thử hiệu năng, thực hiện kiểm tra hệ thống với những giá trị ngoài phạm vi (abnormal conditions) **để xác định những điểm gãy** (breaking point) của phần mềm
  + **Kiểm thử cấu hình (configuration)**
  + **Kiểm thử an toàn (security):** đảm bảo toàn vẹn dữ liệu
  + **Kiểm thử phục hồi (recovery):** đảm bảo có đủ khả năng phục hồi trước những tình huống bất thường xảy ra
* Acceptance test (kiểm thử chấp nhận): được thực hiện bởi khách hàng, end-user để đảm bảo sản phẩm hoạt động đúng với những gì khách hàng mong đợi
  + **Alpha testing:** kiểm thử ngay trên môi trường phát triển phần mềm, dev sẽ quan sát người dùng sử dụng phần mềm và ghi nhận lỗi phát sinh để sửa chữa; giúp phát hiện ra những vấn đề thực tế khi dùng phần mềm mà nhóm phát triển chưa biết
  + **Beta testing:** kiểm thử trên môi trường thực của người dùng (có thể công khai cho người dùng quan tâm dùng thử) để nhận được lỗi từ khách hàng nhằm phát hiện những vấn đề tương tác giữa phần mềm trong môi trường nó được sử dụng; beta testing thực hiện sau alpha testing

# CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM

## Incremental & Big Bang

* **Incremental:** kiểm thử từ phần riêng biệt và tăng dần từng chút một lên; tuy có thể phát hiện ra lỗi sớm và dễ dàng tìm ra nguyên nhân nhưng tốn nhiều thời gian và nhân lực
* **Big Bang:** kiểm thử toàn bộ cùng một lúc; điều này có để khiến tester gặp khó khăn trong việc tìm ra nguyên nhân và dễ mắc phải failure nhưng công việc có thể được hoàn thành sớm (trước khi bắt đầu)

## Top-down & Bottom-up

Cả hai chiến lược này đều thuộc chiến lược Incremental

* **Top-down:** module ở mức cao nhất được kiểm thử trước
* **Bottom-up:** module ở mức thấp nhất được kiểm thử trước

## Static & Dynamic

* **Static:** xem mã nguồn nhưng không thực thi để kiểm thử, thực hiện quy trình Verification trong chiến lược này
* **Dynamic:** xem mã nguồn và thực thi để kiểm thử, thực hiện quy trình Validation trong chiến lược này

# CÁC TÀI LIỆU KIỂM THỬ PHẦN MỀM

* **Test plan:** kế hoạch kiểm thử dùng để phát thảo chiến lược kiểm thử phần mềm (resources, môi trường, phạm vi, lịch trình); được phát triển trong giai đoạn đầu phát triển phần mềm. Test plan có thể bao gồm:
  + Giả thiết khi kiểm thử
  + **Danh sách test case kiểm thử**
  + **Danh sách đặc trưng kiểm thử**
  + **Danh sách sản phẩm kiểm thử**
  + **Tài nguyên kiểm thử**
  + Cách ưu tiên tiếp cận kiểm thử
  + Rủi ro trong kiểm thử
  + Lịch trình, cột mốc cần đạt được
* **Test report:** là kết quả kiểm thử chính thức được ghi lại một cách có tổ chức, mô tả điều kiện, môi trường kiểm thử và so sánh kết quả kiểm thử với mục tiêu được đề ra
* **Test scenario (kịch bản kiểm thử)**: đảm bảo các luồng xử lý từ đầu đến cuối và bao gồm nhiều bước được thực thi
* **Test case:** tài liệu thể hiện một hành vi nào đó trong chương trình được phát triển cho một kịch bản cụ thể để xác minh tính đúng đắn của yêu cầu đặc tả; **gồm ID, kịch bản, mô tả, test step, test data, expected result, actual result** và có thể có điều kiện tiên quyết (prerequisite, tham số kiểm thử, thông tin môi trường, ghi chú.
* **Traceability Matrix:** là một bản lưu vết (trace) các yêu cầu trong vòng đời phát triển phần mềm giúp theo dõi và quản lý mối quan hệ giữa các yêu cầu đặc tả và test case
  + **Mục đích:**
* Đảm bảo phần mềm phát triển như yêu cầu
* Dễ dàng tìm ra nguyên nhân chính gây bug
* Lưu vết được tài liệu được phát triển trong vòng đời phát triển phần mềm

# CÁC KỸ THUẬT KIỂM THỬ PHẦN MỀM

## Kiểm thử hộp đen - Black-box Testing

Kiểm thử hộp đen (black-box testing) hay còn được gọi là kiểm thử dựa trên đặc tả (specification-based testing) vì cơ sở duy nhất để thực hiện kiểm thử là bảng đặc tả yêu cầu chức năng của từng thành phần phần mềm; **kiểm thử tập trung** vào phần mềm **làm được những gì** chứ **không cần biết** phần mềm **làm điều đó như thế nào**

### Ưu điểm:

* Không cần truy cập mã nguồn
* Góc nhìn dev và user được tách biệt
* Cho phép nhiều người tham gia test

### Khuyết điểm:

* Không có hiệu quả cao
* Khó thiết kế test case
* Khó kiểm thử phủ
* Không có định hướng kiểm thử rõ ràng

### Quy trình kiểm thử hộp đen:

* Phân tích đặc tả chức năng của các thành phần phần mềm
* Thiết kế test case
* Thực thi test case
* So sánh actual result với expected result
* Lập báo cáo

### Các kỹ thuật kiểm thử hộp đen:

Kỹ thuật có đầu vào cụ thể và độc lập (không có quan hệ ràng buộc):

* Phân vùng tương đương
* Phân tích giá trị biên

Kỹ thuật có logic và quy tắc, nghiệp vụ

* Bảng quyết định
* Dịch chuyển trạng thái

## Phân vùng tương đương - Equivalence Partitioning (EP)

**Ý tưởng:** nếu một giá trị đại diện trong nhóm thì các giá trị còn lại cũng đúng và ngược lại; phù hợp với các bài toán có giá trị đầu vào là 1 một miền xác định

Phân vùng tương đương chia 1 tập các giá trị điều kiện kiểm thử thành các tập con có các giá trị tương đương nhau và kiểm thử để **giảm thiểu số lượng test case không cần thiết**; có thể dùng trong tất cả cấp độ kiểm thử

Điều kiện để phân vùng tương đương:

* Giá trị tương tự nhau (intuitive similarity)
* Đặc tả mô tả chương trình xử lý theo cùng 1 cách (specified as equivalent)
* Cùng một hướng đi (cùng nhánh if) (equivalent path)
* Giá trị cho cùng 1 kết quả với những giải thiết đưa ra (risk-based)

Khi thiết kế test case, phải đảm bảo tất cả các phân vùng (valid, invalid) được kiểm thử ít nhất 1 lần

## Phân tích giá trị biên - Boundary Value Analysis (BVA)

Là kỹ thuật kiểm thử dựa vào các giá trị tại biên giữa các phân vùng tương đương, gồm 2 trường hợp:

**Kiểm thử hộp trắng (white box):** kỹ thuật kiểm tra logic và cấu trúc mã nguồn. Tester hiểu và biết về công việc trong mã nguồn để làm rõ đơn vị và các mã nguồn

**Kiểm thử hộp xám (gray box):** kiểm thử tương tác người dùng với giao diện nhưng tester cần biết chút ý về cấu trúc mã nguồn và truy cập vào mã nguồn, cơ sở dữ liệu để chuẩn bị dữ liệu và kịch bản kiểm thử tốt hơn