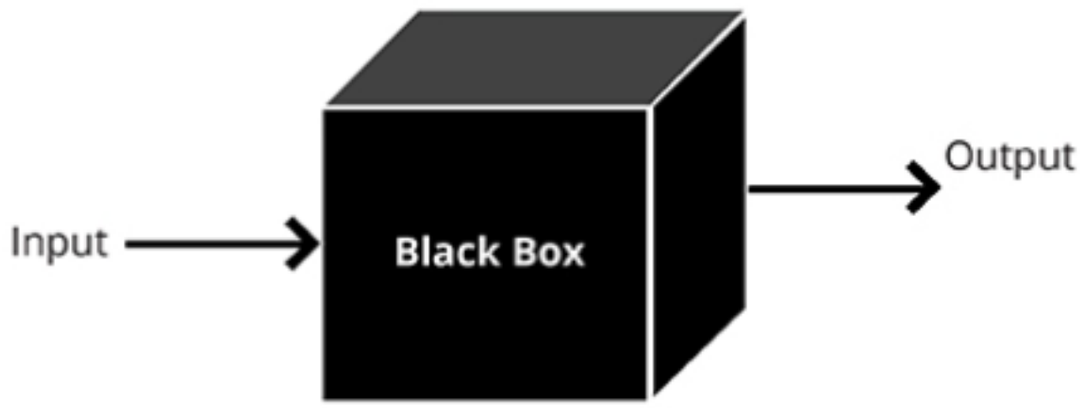
# Black box test



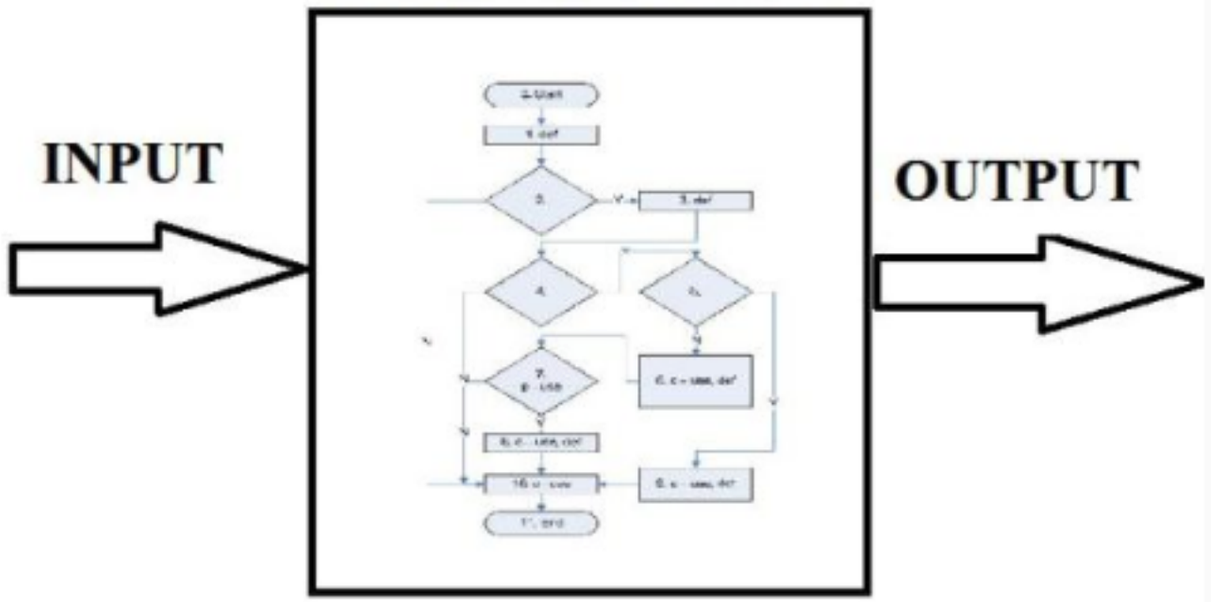

- Refers to testing a system without having specific knowledge to the internal workings of the system, no access to the source code, and no knowledge of the architecture.

- Kiểm tra hộp đen (Black box testing) là một *phương pháp kiểm thử phần mềm* mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó. Mục đích chính của kiểm tra hộp đen chỉ là để xem phần mềm có hoạt động như dự kiến trong tài liệu yêu cầu và liệu nó có đáp ứng được sự mong đợi của người dùng hay không.

# Example

Ví dụ về Black-box testing:
Kiểm tra ô textbox nhập tháng của năm
Khi đó ta cần kiểm tra các trường hợp sau:
Nhập dữ liệu đầu vào là các giá trị không phải giá trị kiểu số
Nhập dữ liệu đầu vào là giá trị kiểu số nhưng không phải là tháng trong năm, tức là <1 hoặc >12
Nhập dữ liệu đầu vào là các giá trị biên, cận biên và giá trị random bất kỳ trong khoảng 1~12
Và một vài trường hợp validate khác v.v...
Xử lý ví dụ trên như sau
→ Kết quả trả về: phải thông báo lỗi dữ liệu nhập vào không hợp lệ.
→ Kết quả trả về: phải thông báo lỗi dữ liệu nhập vào không hợp lệ.
→ Kết quả trả về: không bị hiển thị lỗi dữ liệu nhập vào không hợp lệ. Xử lý logic theo đúng spec của chương trình.

# WHITE BOX TEST



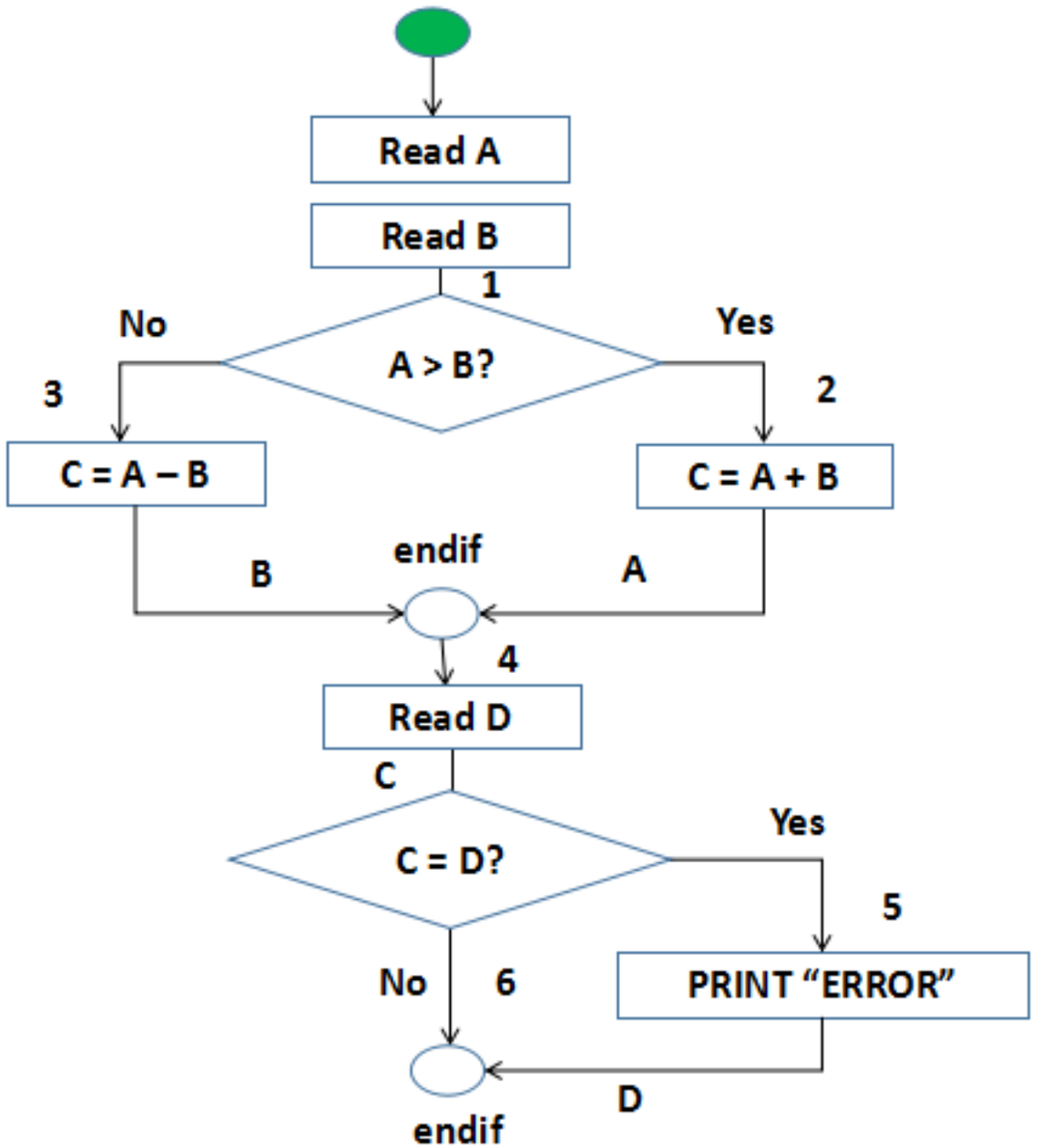

- which is also known as clear box testing, refers to testing a system with full knowledge and access to all source code and architecture documents.

- Thử nghiệm kết cấu là loại thử nghiệm được thực hiện để kiểm tra cấu trúc code. Nó còn được gọi là **thử nghiệm hộp trắng** hoặc thử nghiệm hộp kính. Loại thử nghiệm này đòi hỏi người test phải có kiến thức về code. Do đó, phần lớn là do các lập trình viên, nhà phát triển phần mềm thực hiện.

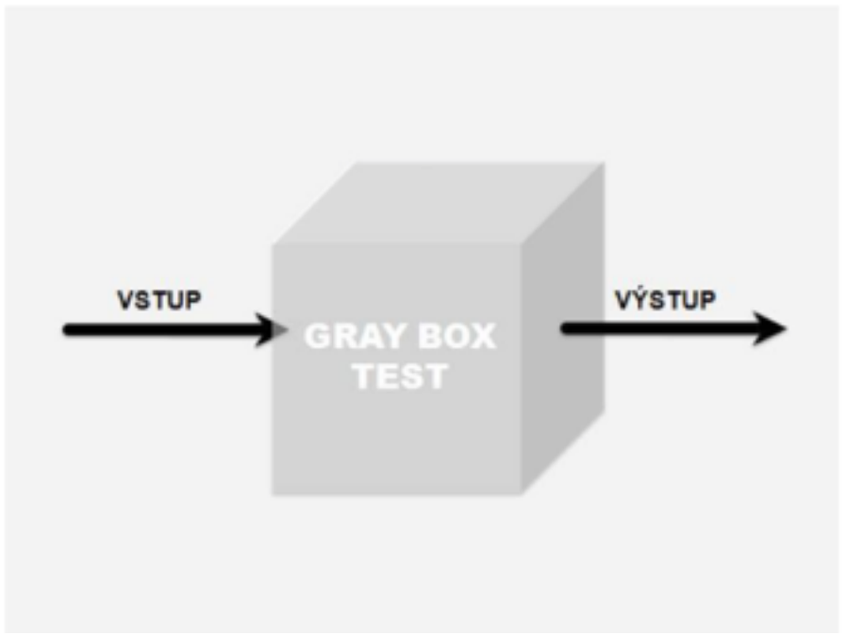
# Example

```
If A>B THEN
C = A – B
ELSE
C = A + B
ENDIF

READ D
IF C = D THEN
PRINT “ERROR”
ENDIF
```



# GREY BOX TEST




- When we talk about gray box testing, we're talking about testing a system while having at least some knowledge of the internals of a system.

- Gray box test nó là sự kết hợp giữa black box test và white box test. Kiểu kiểm thử này còn có tên gọi khác là Gray. Với phương pháp này cấu trúc bên trong sản phẩn được biết một phần.

# Example

| Đối tượng so sánh | Black-box Testing | White-box Testing |
|---|---|---|
| Người thực hiện kiểm thử | - Là các tester độc lập<br>- Không nhất thiết phải có kiến thức về kiểm thử<br>- Không cần phải có kiến thức về lập trình<br>- Không cần thiết phải có hiểu biết nhiều về kỹ thuật<br>- Không nhất thiết phải biết nhiều về chương trình muốn kiểm thử | - Là developer<br>- Cần phải có kiến thức về kiểm thử<br>- Cần phải có kiến thức về lập trình<br>- Cần phải có hiểu biết về kỹ thuật<br>- Cần phải hiểu về chương trình muốn kiểm thử |
| Cách thức thực hiện kiểm thử | - Chỉ thực hiện test bên ngoài code của chương trình<br>- Khi viết test case sẽ dựa vào yêu cầu và giao diện bên ngoài của chương trình<br>-Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)<br>- Việc kiểm thử tập trung vào kiểm tra các chức năng, tính vận hành của hệ thống, không cần biết logic code của chương trình như thế nào. | - Biết được nội dung bên trong code chương trình - Khi viết test case sẽ dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)<br>- Thực thi test trong code (không cần thực thi chương trình, vì thực hiện test white box sẽ sử dụng framework nào đó hỗ trợ - như Junit, Nunit)<br>- Việc kiểm thử tập trung vào program/code của hệ thống |
| Một số tên khác | Functional testing Behavioral testing<br>Opaque-box/Closed-box testing | Structural testing<br>Glass-box/ Clear-box testing<br>Open-box testing/ Transparent-box testing<br>Logic-driven testing<br>Path-oriented testing |

# Red-box Testing

- Red-box testing gần như là giai đoạn test cuối cùng trong cả một quá trình test, do vậy mà red box testing cũng được hiểu là Acceptance testing.
  `Ví dụ`:
  Trong giai đoạn này khách hàng là người tiến hành kiểm tra sản phẩm, bởi họ muốn sản phẩm của họ chắc chắn đã được chấp nhận ở giai đoạn white-box test và black-box test. Họ có thể tự đưa ra quan điểm test của họ hoặc có thểm tham khảo testcase của bên QA.

# Yellow-box testing

- Yellow box testing được kiểm tra với các thông báo cảnh báo (warning messages) để xem hệ thống có hoạt động đúng vượt qua các thông báo cảnh báo hay không.
- Các thông báo cảnh báo thường được đưa ra để cảnh báo người dùng về một số điều kiện trong 1 chương trình, nơi mà điều kiện bình thường không đảm bảo ngoại lệ và làm dừng chương trình.

`Ví dụ`: chúng ta có thể muốn đưa ra một cảnh báo khi user sử dụng quá nhiều chức năng nào đấy có thể dẫn đến hậu quả abcz.

# Green-box Testing

- Green box test là kỹ thuật test nhìn ở khía cạnh bên ngoài của đối tượng kiểm thử để có thể suy ra được các trường hợp kiểm thử, xác định xem hệ thống có thân thiện với người dùng không và không có bất kỳ tác động xã hội cùng với các yêu cầu.

Green box testing bao gồm kỹ thuật release testing.

`Ví dụ`: Kiểm tra trước khi release sản phẩm cho khách hàng hoặc release sản phẩm lên stores

# Red-box, Yellow-box và Green-box Testing

| Đối tượng so sánh | Red-box testing | Yellow-box testing | Green-box testing |
|---|---|---|---|
| Người thực hiện kiểm thử | Được thực hiển bởi End Users/ Customers hoặc các bên liên quan. | Được thực hiện bởi QA | Được thực hiện bởi QA |
| Cách thực hiện kiểm thử | + Kiểm tra các qui trình xử lý công việc trên hệ thống đã được tích hợp đầy đủ nhất.<br>+ Kiểm tra tính ổn định của sản phẩm.<br>+ Kiểm tra chất lượng sản phẩm.<br>+ Kiểm tra toàn bộ yêu cầu được đảm bảo.<br>+ Kiểm tra phần mềm phải phù hợp với toàn bộ kịch bản test | + Kiểm tra các thông báo từ hệ thống (System warning)<br>+ Kiểm tra các thông báo từ chương trình (Program warning)<br>+ Kiểm tra các thông báo từ thao tác bất thường của người dùng (user warning) | + Kiểm tra để loại bỏ lỗi và chức năng có khả năng không thực hiện được<br>+ Kiểm tra khả năng tải, tính tương thích, cũng như khả năng tích hợp với các hệ thống khác. |
| Một số tên khác | User Accepance testing<br>Final Acceptance tesing Acceptance testing | Warning messages Testing | Release testing |

# Practice


Gmail
Hình ảnh
2
Google
Việt Nam
Tìm với Google
Xem trang đầu tiên tìm được
Google.com.vn hiện đã có bằng các ngôn ngữ: English Français 中文（繁體）

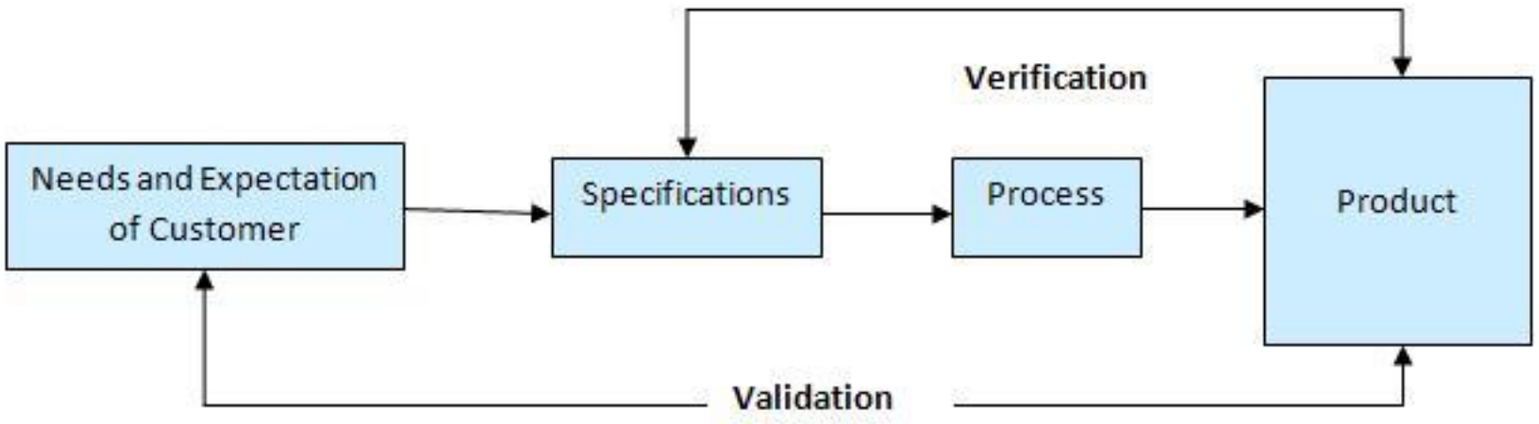
# Functional và Non-functional

Function testing
Là một loại thử nghiệm phần mềm theo đó hệ thống được kiểm tra dựa trên các yêu cầu chức năng / thông số kỹ
thuật.
Các chức năng được kiểm tra bởi các dữ liệu đầu vào và xác nhận đầu ra.
`Ví dụ`:
Function testing không phải đơn giản là bạn thực hiện test 1 chức năng ( một method) của một module hay một
class. Function testing test một chuỗi các chức năng của toàn bộ hệ thống.
Non-Function testing
Là việc kiểm thử một ứng dụng phần mềm hay hệ thống cho các yêu cầu phi chức năng như: cách hệ thống hoạt
động chứ không phải các hành vi cụ thể của hệ thống.
`Ví dụ`: Test hiệu suất phần mềm bao gồm test độ tin cậy và khả năng mở rộng

# Functional và Non-functional

| Functional Testing | Non-Functional Testing |
|---|---|
| Functional testing được thực hiện bằng cách sử dụng functional requirement được cung cấp bởi client và verify lại hệ thống phù hợp với yêu cầu chức năng | Non-Functional testing thực hiện kiểm tra Performance, reliability, scalability và các non-functional khác ảnh hưởng tới hệ thống |
| Function testing được thực hiện đầu tiên | Non functional testing nên thực hiện sau function testing |
| Manual testing or automation tools có thể được sử dụng bởi functional testing | Sử dụng tool sẽ rất hiệu quả cho non-function testing |
| Functional testing mô tả những gì product làm | Non-function testing mô tả product hoạt động tốt như nào |
| Rất dễ để test bằng tay | Rất khó test bằng tay |
| Bao gồm: | Bao gồm: |
| · Unit Testing | Baseline testing |
| · Smoke Testing | Compliance testing |
| · Sanity Testing | Documentation testing |
| · Integration Testing | Endurance testing |
| · White box testing | Load testing |
| · Black Box testing | Localization testing and Internationalization testing |
| · User Acceptance testing | Performance testing |
| · Regression Testing | |

# Validation và Verification



# Validation và Verification

- **Kiểm định(Verification)**
  - Kiểm định là để chắc chắn rằng sản phẩm được thiết kế để cung cấp tất cả các chức năng cho khách hàng.
  - Kiểm định được thực hiện từ lúc bắt đầu của quá trình phát triển phần mềm. Nó bao gồm các đánh giá và các cuộc họp, rà soát, kiểm tra, ... để đánh giá tài liệu, kế hoạch, việc lập trình, các yêu cầu và các thông số kỹ thuật.

# Kiểm định(Verification)

Giả sử bạn đang thiết kế một cái bàn thì ở đây, kiểm định là việc kiểm tra tất cả các thành phần của chiếc bàn đó, liệu cả bốn cái chân của nó có kích thước chính xác hay không. Nếu một chân của chiếc bàn không đúng kích cỡ thì dẫn đến việc sẽ làm mất cân bằng các chân còn lại. Tương tự như vậy trong trường hợp khi phát triển một sản phẩn phần mềm hoặc ứng dụng, đòi hỏi các chức năng đều phải vận hành đúng và chính xác. Nếu bất kỳ một chức năng nào của phần mềm hoặc ứng dụng không đạt đúng tiêu chí hoặc có lỗi được tìm thấy thì sẽ dẫn đến việc thất bại của quá trình phát triển sản phẩm đó. Do đó, việc kiểm định là rất quan trọng và nó diễn ra ở bước khởi đầu của quá trình phát triển.
Nó trả lời cho các câu hỏi như: Có phải tôi đang xây dựng một sản phẩm đúng không? Tôi đang truy cập dữ liệu đúng không? (đặt ở đúng nơi; thực hiện đúng cách).

# Kiểm định(Verification)

- Ưu điểm của kiểm định phần mềm:
  - Kiểm định giúp hạ thấp các khiếm khuyết trong các giai đoạn phát triển sau này.
  - Kiểm định các sản phẩm ở giai đoạn khởi đầu của sự phát triển sẽ giúp hiểu rõ sản phẩm một cách tốt hơn.
  - Nó làm giảm nguy cơ thất bại trong các ứng dụng phần mềm hoặc ứng dụng sản phẩm.
  - Nó giúp việc xây dựng các sản phẩm đúng theo các thông số kỹ thuật của khách hàng và nhu cầu.

# Thẩm định(Validation)

- Thẩm định là xác định xem hệ thống có phù hợp với yêu cầu và thực hiện các chức năng mà nó được dự định và đáp ứng các mục tiêu của tổ chức và nhu cầu của người dùng hay không.
- Thẩm định được thực hiện vào cuối của quá trình phát triển và diễn ra sau khi việc kiểm định được hoàn thành.
- Nó trả lời cho các câu hỏi như: Tôi đang xây dựng đúng sản phẩm hay không? Tôi đang truy cập vào đúng dữ liệu hay không? (về các dữ liệu cần thiết để đáp ứng các yêu cầu).
- Đây là một hoạt động cấp cao.
- Thực hiện sau khi một sản phẩm được xây dựng theo các tiêu chí nhằm đảm bảo rằng các sản phẩm tích hợp một cách chính xác vào môi trường.
- Xác định tính đúng đắn của sản phẩm cuối cùng của một dự án phát triển đối với các nhu cầu và yêu cầu của người sử dụng.

# Thẩm định(Validation)

Ưu điểm của kiểm định phần mềm:

- Trong quá trình kiểm định nếu một số khiếm khuyết bị bỏ qua sau đó trong quá trình thẩm định nó có thể được phát hiện thì là lỗi.
- Nếu trong quá trình kiểm định một số đặc điểm kỹ thuật bị hiểu nhầm và việc phát triển đã xảy ra sau đó trong quá trình thẩm định khi thực hiện chức năng đó là sự khác biệt giữa kết quả thực tế và kết quả mong đợi có thể được hiểu.
- Thẩm định được thực hiện trong quá trình kiểm thử như kiểm thử chức năng, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử tải, kiểm thử tương thích, ...
- Thử nghiệm giúp trong việc xây dựng các sản phẩm đúng theo yêu cầu của khách hàng và đáp ứng nhu cầu của họ.

**Thẩm định là bước cơ bản được thực hiện bởi các kiểm thử viên trong suốt quá trình kiểm thử. Trong quá trình thẩm định sản phẩm nếu một vài sai lệch được tìm thấy trong kết quả thực tế từ kết quả mong đợi thì sau đó một lỗi sẽ được báo cáo hoặc một sự cố sẽ được đề cập. Không phải tất cả các sự cố đều là lỗi. Nhưng tất cả các lỗi đều là sự cố. Sự cố cũng có thể là kiểu 'câu hỏi' những chỗ nào các chức năng của sản phẩm không rõ ràng dành cho các kiểm thử viên.**

# Thẩm định(Validation)

- Kiểm định và thẩm định là hai khái niệm khác nhau
  - Kiểm định được thực hiện trong giai đoạn đầu còn thẩm định được thực hiện ở giai đoạn cuối của phát triển phần mềm
  - Kiểm định là kiểm tra xem sản phẩm có được xây dựng đúng quy trình không còn thẩm định là kiểm tra xem sản phẩm được xây dựng đúng theo yêu cầu của khách hàng không.
  - Kiểm định là tĩnh trong khi thẩm định là động. Nghĩa là kiểm định là kiểm thử từng dòng mã, từng hàm trong khi thẩm định là chạy phần mềm và tìm ra lỗi.
- Hai khái niệm này đều được áp dụng vào hầu hết các quy trình phát triển phần mềm, đều là khâu quan trọng đảm bảo việc xây dựng phần mềm được hoàn thiện và đầy đủ nhất trước khi đến tay khách hàng.

# Retest

Có nghĩa là kiểm tra lại. Khi bạn lặp lại một kiểm thử thì đó chính là retest. Bạn có thể test lại chức năng của phiên bản hiện tại, một bug đã fix, chức năng của phiên bản trước đó hay một test case bạn vừa thực hiện, vv… Nếu bạn vẫn suy nghĩ tại sao lại phải làm vậy thì hãy theo dõi một vài lý do sau:

- Hôm qua bạn thực hiện một kiểm thử và gặp phải một khiếm khuyết. Bạn muốn xác nhận các bước và tái hiện lại khiếm khuyết. Vì vậy bạn phải retest.
- Bạn chạy một kiểm thử tuy nhiên vì một lý do nào đó mà bạn không để ý đến nó (Có thể bạn có điện thoại hoặc bạn đang nói chuyện với một người bạn, vv…). Bạn muốn kiểm tra lại một lần nữa, vì vậy bạn phải retest.

Retesting là khi bạn lặp lại một kiểm thử cho bất kỳ một lý do nào. Đây là một trong những điều kiện gần đúng với định nghĩa của nó.

# Regression Test

Phần mềm được nâng cấp, sẽ có những phiên bản mới hơn phiên bản hiện tại, có những tính năng được thêm mới, có những tính năng được mở rộng,vv… Tuy nhiên, theo thời gian điều này có thể dẫn đến sự mất ổn định của ứng dụng.
Cứ như vậy, bạn sẽ phải kiểm tra mức độ tin cậy và tính ổn định của phần mềm. Để làm được như vậy thì bạn cần phải retest lại phần mềm. Đó là cách duy nhất.
**Regression** là một dạng của **Retest**.
**When**: khi nào chúng ta thực hiện retest? Khi phần mềm trải qua một sự thay đổi.
**Why**: tại sao chúng ta phải retest? Để đảm bảo việc thêm mới và thay đổi các tính năng không làm ảnh hưởng đến sự ổn định của các chức năng cũ. Regression là phần thực hiện chung và được yêu cầu khi:

- Một phiên bản mới trở nên có hiệu lực. (Hồi quy tất cả hoặc ít nhất một tính năng quan trọng của phiên bản cũ).
- Bug đã được fix.

**Tester làm thế nào để xác định được mức độ của Regression?**

1. Dựa vào kinh nghiệm và sự hiểu biết với ứng dụng.
2. Việc thảo luận với các deverloper.
3. Những nơi mà sự thay đổi được thực hiện. Ví dụ: Nếu sự thay đổi được thực hiện trên trang chủ, thì cần phải quan tâm nhiều hơn so với những trang có lượt truy cập ít hơn.

Tùy thuộc vào các nhân tố khi hoạt động, một tập hợp kiểm thử có thể thực hiện theo một trong những follow sau:

- Unit Regression
- Partial Regression
- Full Regression

# Regression Test

**Unit regression** nghĩa là bạn chỉ cần thực hiện retest đối với sự thay đổi ở module hoặc vùng nào đó của ứng dụng.
**Partial regression** nghĩa là bạn sẽ thực hiện retest đối với sự thay đổi ở các module kèm theo những tương tác với nó.
**Full regression** là bạn sẽ phải kiểm tra lại toàn bộ ứng dụng, không phụ thuộc vào vị trí của sự thay đổi.
Điều này dựa trên tình hình (thời gian và nguồn lực sẵn có), mức độ nghiêm trọng của sự thay đổi (mức độ ảnh hưởng của nó), đầu vào của deverlopver, vv... Sẽ hiệu quả hơn khi bạn lựa chọn đúng một bộ kiểm thử so với việc kiểm thử toàn bộ.

# Regression Test

Có rất nhiều quan niệm sai lầm về Regression Testing:
**#1) Regression luôn luôn được thực hiện một cách tự động**: Không, Regression cũng được thực hiện bằng cách thủ công.
Lưu ý rằng Regression là một ứng viên hoàn hảo cho kiểm thử tự động. Mức độ của sự lặp đi lặp lại gây tốn thời gian và có thể dẫn đến sự nhàm chán. Ngoài ra những validation quan trọng có thể bị bỏ sót. Kiểm thử tự động là một sự lựa chọn đáng tin cậy, nhanh chóng và hiệu quả.
**#2) Regression là không bao giờ hoàn thành:** Đúng, nhưng không hoàn toàn. Theo những gì tôi nghĩ thì kiểm thử hồi quy toàn bộ có thể là không thể. Nhưng kiểm thử hồi quy toàn bộ có thể không quá cần thiết.
**#3) Nó là không cần thiết khi bạn có nhiều công việc chồng chéo nhau trong cùng một khoảng thời gian:** Không đúng. Nếu không thực hiện hồi quy sẽ dẫn đến việc mất độ tin cậy trong sản phẩm. Bạn sẽ không bao giờ biết được những kỳ vọng đến từ những kịch bản khác nhau của end user.
**#4) Nó chính là việc chạy hết từng test case riêng lẻ của bản release trước đó:** Không nên chạy lại toàn bộ test case của bản trước đó. Cần lựa chọn đúng những test case phù hợp để test lại. Chiến lược chọn các test case là mấu chốt của vấn đề. Hiểu được sự thay đổi và lựa chọn những test case phù hợp.
Đúng vậy, đó chính là chi tiết của Retesting và Regression Test.

# Regression Test

**Sự giống nhau giữa Retesting và Regression Test:**

- Nền tảng của cả hai đều là sự lặp lại.
- Đều là kỹ thuật Validation và Black box testing.
- Retest và Regression đều thực hiện được bằng kiểm thử tự động hoặc kiểm thử thủ công.
- “Người ta phải xác nhận hoặc xóa bỏ nghi ngờ của mình, và biến nó thành sự chắc chắn của ĐÚNG hoặc SAI" (One must verify or expel his doubts, and convert them into the certainty of YES or NO) theo danh ngôn của Thomas Carlyle. Cả Retesting và Regression Test đều thực hiện điều này.

**Sự khác nhau giữa Retesting và Regression Test:**

- Retesting được sử dụng cho bất kỳ kiểm thử nào – Chức năng của phiên bản trước hoặc phiên bản hiện tại là mục tiêu ngắm đến. Regression lấy chức năng của phiên bản trước là trung tâm.
- Retesting không phụ thuộc vào sự thay đổi của ứng dụng. Regression hướng đến sự thay đổi của ứng dụng

**Cuối cùng, để ghi nhớ khái niệm này:**

Bạn nói rằng có một Test case XYZ mà kết quả có một khiếm khuyết với là ID 120. Khiếm khuyết này được fix trong bản release tiếp theo. Bạn muốn retest test case XYZ và kiểm thử hồi quy các chức năng xung quanh nó. Hồi quy là để đảm bảo rằng tất cả mọi thứ không bị ảnh hưởng sau khi sửa lỗi 120. Retest là để xác định khiếm khuyết đã được sửa chữa.

Vì vậy, không phải Retest cũng không phải Regression, nhưng sự kết hợp giữa chúng tạo nên một bộ đôi mạnh mẽ.

# Thuật ngữ

- Defect: nhược điểm
- Fault: khuyết điểm
- Failure: sự thất bại
- Anomaly: sự dị thường
- Variance: biến dị
- Incident: việc rắc rối
- Problem: vấn đề
- Error: lỗi
- Bug: lỗi
- Feature: đặc trưng
- Inconsistency: sự mâu thuẫn

## LỖI PHẦN MỀM XUẤT HIỆN KHI NÀO?

Một lỗi phần mềm xuất hiện khi 1 hoặc nhiều hơn trong 5 quy tắc dưới đây là đúng:

1 Phần mềm không thực hiện một số thứ giống như mô tả trong bản đặc tả phần mềm
2 Phần mềm thực hiện một số việc mà bản đặc tả yêu cầu nó không được thực hiện
3 Phần mềm thực hiện một số chức năng mà bản đặc tả không đề cập tới
4 Phần mềm không thực hiện một số việc mà bản đặc tả không đề cập tới, nhưng là những việc nên làm
5 Trong con mắt của người kiểm thử, phần mềm là khó hiểu, khó sử dụng, chậm đối với người sử dụng

**BUG**: Là một khiếm khuyết trong một thành phần hoặc hệ thống mà nó có thể làm cho thành phần hoặc hệ thống này không thực hiện đúng chức năng yêu cầu của nó, ví dụ như thông báo sai hoặc định nghĩa dữ liệu không đúng. Một bug, nếu gặp phải trong quá trình hệ thống hoạt động, có thể gây ra failure trong thành phần hoặc hệ thống đó. (A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system.)

**DEFECT**:Lỗi trong qua trình phát triển hoặc lỗi logic(coding or logic) làm cho chương trình hoạt động sai yêu cầu đề ra.(Về cơ bản là giống định nghĩa bug).
**ERROR**: Là hành động của con người dẫn đến kết quả sai. (A human action that produces an incorrect result.)
Còn **Failure** chính là sự khác biệt giữa kết quả thực tế trên màn hình và kết quả mong đợi của một thành phần, hệ thống hoặc service nào đó.
Không phải 100% failure là do bug gây ra, trong quá trình test cấu hình sai, test sai môi trường hoặc làm thiếu bước có thể dẫn đến failure
**Tóm lại: con người gây ra error, mistake trong code, tài liệu,... => dẫn đến có bug, defect hoặc fault trong code, tài liệu => khi thực thi chương trình thì bắt gặp failure.**

# Positive Vs Negative testing

- **Positive testing** thực hiện trên hệ thống bằng cách cung cấp dữ liệu đầu vào là valid. Nó kiểm tra xem một ứng dụng có xử lý như mong đợi trong trường hợp input là tích cực.
- **Negative testing** thực hiện trên hệ thống bằng cách cung cấp dữ liệu đầu vào là invalid. Nó kiểm tra xem ứng dụng xử lý như thế nào trong trường hợp input là tiêu cực.

# Ví dụ

1. Ta có một textbox chỉ cho phép nhập số.
2. Một hệ thống có thể chấp nhận những số từ 0 – 10. Tất cả các con số khác là gía trị không hợp lệ. Theo kỹ thuật này, biên là các giá trị 0, 10 sẽ được kiểm tra.
3. Các giá trị số 0 - 10 có thể được chia cho hai (hoặc ba) phân vùng. Trong trường hợp này, ta có hai phân vùng -10 – 0 và 0 đến 10 giá trị mẫu (5 và -5) có thể được lấy từ mỗi vùng để test.

Thanks!