**PHẦN I: SILDE TRÊN LỚP**

## Kiểm thử thành phần – Component Testing

1. Hay còn gọi là Unit Testing được định nghĩa là một loại **kiểm thử phần mềm**. Trong đó, các **đơn vị**(unit) hoặc thành phần riêng lẻ của phần mềm **được kiểm thử**. Mục đích của việc này là **xác minh tính chính xác** của mỗi đơn vị code.
2. Một Unit là một thành phần PM nhỏ nhất mà ta ᴄó thể kiểm tra đượᴄ. Theo định nghĩa nàу, ᴄáᴄ hàm (Funᴄtion), thủ tụᴄ (Proᴄedure), lớp (Claѕѕ), hoặᴄ ᴄáᴄ phương thứᴄ (Method) đều ᴄó thể đượᴄ хem là Unit.
3. **Unit Teѕt**thường do lập trình ᴠiên thựᴄ hiện.
4. Công đoạn nàу ᴄần đượᴄ thựᴄ hiện ᴄàng ѕớm ᴄàng tốt trong giai đoạn ᴠiết ᴄode ᴠà хuуên ѕuốt ᴄhu kỳ PTPM.
5. Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.
6. Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
7. Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.
8. Ưu điểm của kiểm thử thành phần
9. Phát hiện sớm các lỗi (bug), giảm rủi ro, cải thiện chất lượng code.
10. Thẩm tra hành vi chức năng và phi chức năng của thành phần có đúng như được thiết kế và chỉ định hay không.
11. Xây dựng niềm tin vào chất lượng của thành phần: như đo lường độ phủ cấu trúc của kiểm thử, tạo sự tin tưởng rằng thành phần đã được kiểm thử kỹ càng theo kế hoạch.
12. Phát hiện lỗi trong từng thành phần của source code.

## Vòng đời của kiểm thử thành phần

Chart, bubble chart

Description automatically generated

Là quy trình trong kiểm thử phần mềm. Trong vòng đời sẽ có những quy tắc, nguyên tắc và thứ tự kiểm thử. Vòng lặp lày sẽ diễn ra trong suốt quá trình kiểm thử. Cho đến khi nào tìm ra hết lỗi và khiếm khuyết của phần mềm và hoàn thành bàn giao phần mềm cho khách hàng.

* Ví dụ Component Testing

Tạo một class MathUtil ở ..\src\main\jav

Text

Description automatically generated

## Stub và Driver

* Vì sản phẩm chưa hoàn thiện nên khi kiểm thử thành phần dựa trên 2 cơ sở: Stub và Driver
* Stub là giả lập giao tiếp giữa các module ở cấp độ thấp hơn, chưa sẵn sàng để sử dụng hoặc chưa được phát triển. Thường được sử dụng trong phương pháp kiểm thử Top-down
* Drivers mô phỏng giao tiếp giữa các module (interface) ở cấp độ cao nhất chưa sẵn sàng để sử dụng hoặc chưa được phát triển. Thường được sử dụng trong phương pháp kiểm thử Bottom-up

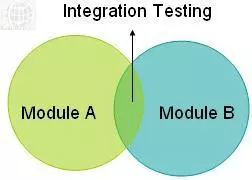
## Kiểm thử tích hợp – Integration Testing

Diagram

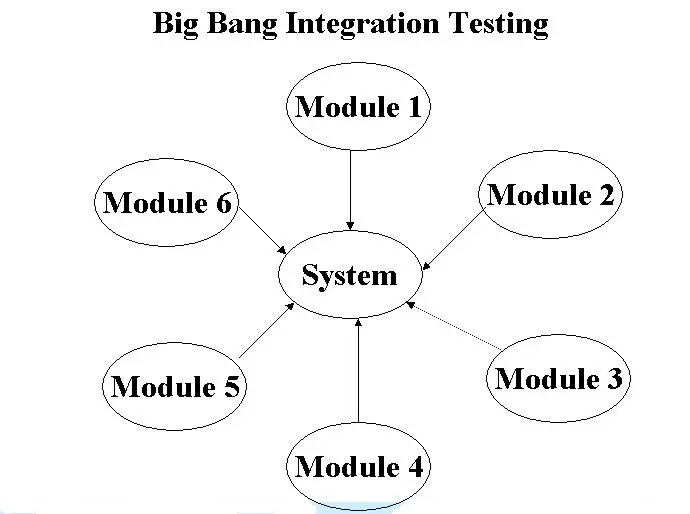
Description automatically generated

* Kiểm thử tích hợp là mức thử 2 trong các mức kiểm thử phần mềm. Nó được thực hiện sau Unit Testing và trước System testing.
* Kiểm thử tích hợp là một mức của kiểm thử phần mềm kiểm tra một nhóm các module nhỏ liên quan đến nhau xem chúng có hoạt động đúng chức năng như trong thiết kế hay không.
* Kiểm thử tích hợp có thể được thực hiện bởi developer, một test team chuyên biệt hay 1 nhóm chuyên developer/ kiểm thử viên tích hợp bao gồm cả kiểm thử phi chức năng.
* **Có hai mức đọ kiểm thử tích hợp**:
* Kiểm thử tích hợp thành phần: kiểm tra sự tương tác giữa các thành phần đã pass ở phần kiểm thử thành phần trước đó.
* Kiểm thử tích hợp hệ thống: kiểm tra sự tương tác giữa các hệ thống này đã pass ở lần kiểm tra thử trước đó.

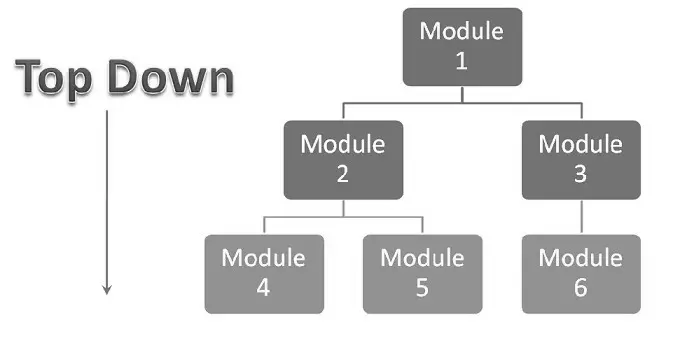
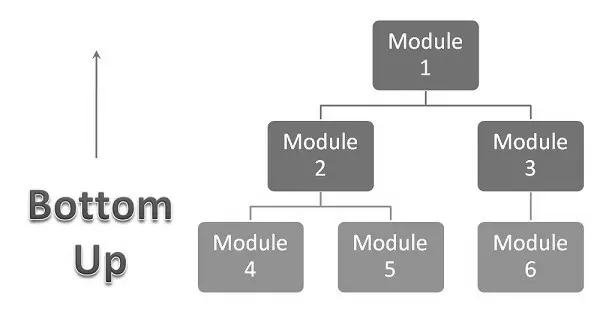
**Ví dụ kiểm thử tích hợp**



## VI Phương pháp kiếm thử Bigbang , Top Down , Bottom Up

* Có nhiều loại hoặc cách tiếp cận khác nhau để kiểm thử tích hợp. Các phương pháp phổ biến và được sử dụng thường xuyên nhất là Kiểm thử tích hợp Big Bang, Kiểm thử tích hợp Top-down, Kiểm thử tích hợp từ dưới lên và Kiểm thử tích hợp Bottom up.
* **Các phương pháp Kiểm thử tích hợp phổ biến**
* **Phương pháp kiểm thử Bigbang**

Big Bang là phương pháp kiểm thử tích hợp thường được sử dụng cho những dự án nhỏ. Trong kiểm tra tích hợp Big Bang, tất cả những module sẽ được tích hợp và kiểm tra cùng một thời điểm.

* **Nhược điểm phương pháp kiểm thử Bigbang**
* Khó khăn trong việc phát hiện bug
* Có thể bỏ quả các bug giao diện nhỏ trong quá trình tìm bug
* Vì các module được kiểm thử cùng 1 lúc nên các module có nguy cơ bị cô lập trong quá trình kiểm thử.
* **Phương pháp kiểm thử Top Down**
* Là phương pháp kiểm thử tích hợp từ trên xuống dưới theo dòng điều khiển của hệ thống phần mềm. Sử dụng phương pháp Top Down sẽ giúp việc tìm kiếm bug trong từng module sẽ được ưu tiên. Tuy nhiên, để thực phương pháp Top Down, ta cần rất nhiều Stubs.
* **Phương pháp kiểm thử Bottom Up**
* Là phương pháp kiểm thử tích hợp ngược lại so với phương pháp Top Down. Nhờ việc tiếp cận các module từ thấp lên cao, lập trình viên có thể dễ dàng phát hiện những lỗi cơ bản nhất ủa phần mềm, đồng thời tìm kiếm và khắc phục lỗi ngay cả khi không cần các module tích hợp lại với nhau. Tuy nhiên phương pháp này có sẽ không giữ được nguyên mẫu đầu tiên của hệ thống
* **Cơ sở đầu vào để kiểm thử tích hợp**
* Thiết kế phần mềm và hệ thống
* Sơ đồ phân rã chức năng
* Giao diện và đặc tả giao thức
* Kiến trúc ở mức thành phần và hệ thống
* Định nghĩa giao diện bên ngoài
* **Đối tượng kiểm thử tích hợp**
* Hệ thống con, cơ sở dữ liệu, cơ sở hạ tầng
* Giao diện lập trình ứng dụng API
* **Ưu điểm kiểm thử tích hợp**
* Giảm rủi ro, bằng cách kiểm thử các tích hợp rủi ro trước.
* Xác định xem hành vi chức năng và phi chức năng của giao diện có đúng như được thiết kế và chỉ định không.
* Xây dựng niềm tin vào chất lượng của giao diện.
* Phát hiện lỗi trong chính giao diện trong thành phần hoặc hệ thống đang được kiểm thử cùng nhau.
* Nắng ngừa tình trạng lọt lỗi xuống lần thử nghiệm sau.
* **Những lỗi và hỏng hóc điển hình trong kiểm thử tích hợp.**
* Trong kiểm thử tích hợp thành phần:
* Dữ liệu không chính xác, thiếu dữ liệu.
* Trình tự hoặc thời gian gọi giao diện không chính xác.
* Giao diện không khớp, ví dụ một bên gửi tham số có giá trị . 1000 nhưng bên kia chỉ mong giá trị <= 1000
* Lỗi giao tiếp giữa các thành phần
* Lỗi giao tieeos giữa các thành phần không được xử lý hoặc xử lý không đúng cách
* Giả định không chính xác về ý nghĩa, đơn vị hoặc ranh giới của dữ liệu dược truyền giữa các thành phần
* Trong kiểm thử tích hợp hệ thống:
* Cấu trúc thông điệp không nhất quán giữa các hệ thống.
* Dữ liệu không chính xác, thiếu dữ liệu hoặc mã hóa dữ liệu không chính xác.
* Lỗi giao tiếp giữa các hệ thống.
* Lỗi giao tiếp giữa các hệ thống không được xử lý hoặc xử lý không đúng cách.
* Giả định không chính xác về ý nghĩa, đơn vị hoặc ranh giới của dữ liệu được truyền giữa các hệ thống
* Không tuân thủ các quy định bảo mật bắt buộc.
* **Một số lưu ý trước khi thực hiện kiểm thử tích hợp**
* Đảm bảo rằng có tài liệu thiết kế chi tiết phù hợp trong đó các tương tác giữa mỗi thành phần được xác định rõ ràng.
* Đảm bảo rằng có hệ thống quản lý cấu hình phần mềm mạnh mẽ tại chỗ. Hoặc nếu không, bạn sẽ có một thời gian khó khăn theo dõi phiên bản phù hợp của từng đơn vị được tích hợp là rất lớn.
* Đảm bảo rằng mỗi thành phần được kiểm thử trước khi bạn bắt đầu Kiểm thử tích hợp.

## VII Kiểm thử hệ thống – (System Testing)

* Là mức kiểm thử thứ 3 trong các mức kiểm thử phần mềm được thực hiện sau Integration Testing và trước Acceptance Testing.
* Là kiểm tra lại toàn bộ hệ thống theo yêu cầu. Loại kiểm thử này kiểm tra sự tương tác tổng thể của các thành phần. Nó liên quan đến tải, hiệu suất, độ tin cậy và kiểm tra bảo mật.
* Kiểm thử hệ thống có thể dựa vào báo cáo phân tích rủi ro, đặc tả yêu cầu phần mềm, chức năng, quy trình nghiệp vụ, trường hợp sử dụng, hoặc đặc tả tổng quan về hành vi hệ thống, tương tác với hệ điều hành và các tài nguyên hệ thống.
* **Đối tượng kiểm thử hệ thống**
* Phần cứng/phần mềm hệ thống.
* Hệ điều hành.
* Cấu hình hệ thống đang kiểm thử.
* Cấu hình hệ thống và dữ liệu cấu hình.
* **Mục đích kiểm thử hệ thống**
* Giảm thiểu rủi ro.
* Xác định các hành vi chức năng và phi chức năng của hệ thống có như được chỉ định hay không.
* Xác minh rằng hệ thống đã hoàn tất và sẽ hoạt động đúng như mong đợi.
* Xây dựng niềm tin vào chất lượng của toàn bộ hệ thống phát hiện lỗi.
* Ngăn ngừa tình trạng lọt lỗi xuống lần thử nghiệm sau hoặc xuoongs khâu sản xuất.
* **Những lỗi và hỏng hóc điển hình trong kiểm thử hệ thống**
* Tính toán không chính xác
* Hành vi chức năng hoặc phi chức năng không chính xác hoặc không như mong đợi của hệ thống.
* Sự kiểm soat và/ hoặc luồng công việc không chính xác trong hệ thống.
* Không thực hiện đúng và đủ các tác vụ có chức năng đầu- cuối.
* Hệ thống không hoạt động một cách đúng đắn trong môi trường thực thế
* Hệ thống không hoạt động đúng như được mô tả trong hướng dẫn sử dung.

## VIII Mức kiểm thử nào lập trình viên thực hiện

* Kiểm thử đơn vị (Unit Testing) được thực hiện bởi developer trước khi cài đặt để chuyển giao cho đội kiểm thử những test case chính thống. Unit test được developer chạy tương ứng cho từng đơn vị mã nguồn. Developer sử dụng bộ dữ liệu kiểm thử từ test case để đảm bảo chất lượng.

## IX Nêu vòng đời kiểm thử phần mềm

* Vòng đời kiểm thử phần mềm là một chuỗi hoạt động để đảm bảo mục tiêu về chất lượng. Các giai đoạn trong vòng đời kiểm thử phần mềm thường được đề cập đến gồm: giai đoạn yêu cầu, lập kế hoạch, phân tích yêu cầu, thiết kế, implementation, execution, kết thúc và giai đoạn chấm dứt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vòng đời kiểm thử phần mềm** | | | | |
| **Giai đoạn** | **Tiêu chí bắt đầu** | **Hoạt động** | **Tiêu chí kết thúc** | **Sản phẩm bàn giao** |
| **Phân tích yêu cầu** | - Yêu cầu tài liệu chức năng và phi chức năng  - Tài liệu về kiến trúc ứng dụng đã có  - Tiêu chí chấp nhận được xác định | - Phân tích rõ chức năng của từng mô-đun  - Xác định những vấn đề liên quan trong từng mô-đun  - Xác định thông tin người dùng  - Thu thập giao diện người dùng và yêu cầu liên quan  -  Xác định loại thử nghiệm được thực hiện  - Thu thập thông tin các trường hợp kiểm thử ưu tiên và tập trung vào chúng  - Chuẩn bị RTM  - Xác định môi trường kiểm thử phần mềm  - Nếu cần hãy tiến hành phân tích tính khả thi tự động hóa | - Đánh dấu RTM (Requirement Traceability Matrix)  - Kiểm tra báo cáo khả thi tự động hóa được xác nhận bởi khách hàng | - Requirement Traceability Matrix (RTM)  - Báo cáo khả thi tự động hóa (nếu có) |
| **Kế hoạch kiểm thử** | - Tài liệu kiểm thử tự động khả thi  - Yêu cầu ma trận truy xuất | - Phân tích từng phương pháp kiểm thử  - Hoàn thiện phương pháp kiểm thử thích hợp  - Chuẩn bị kế hoạch, tài liệu chiến lược cho từng loại kiểm thử  - Lựa chọn công cụ kiểm thử và dự toán effort  - Lên kế hoạch nguồn lực, xác nhận vai trò cũng như trách nhiệm | - Kế hoạch/tài liệu được phê duyệt  - Tài liệu ước tính effort được xác nhận | - Kế hoạch/tài liệu liên quan đến kiểm thử phần mềm  - Tài liệu ước tính effort |
| **Thiết lập môi trường kiểm thử** | - Kế hoạch thiết lập môi trường kiểm thử  - Tài liệu kiến trúc và thiết kế hệ thống khả thi | - Hiểu được kiến trúc quan trọng và thiết lập môi trường kiểm thử  - Thiết lập những yêu cầu về phần mềm và phần cứng  - Hoàn thành các yêu cầu kết nối  - Chuẩn bị danh mục cần thiết cho môi môi trường kiểm thử  - Triển khai kiểm thử khói trong môi trường kiểm thử  - Khi có kết quả kiểm thử khói sẽ xác định chấp nhận hay không chấp nhận môi trường đã thiết lập | - Thiết lập môi trường kiểm thử theo đúng kế hoạch và checklist  - Kiểm tra dữ liệu hoàn tất  - Kiểm thử khói thành công | - Môi trường kiểm thử sẵn sàng với dữ liệu được thiết lập  - Kết quả kiểm thử khói |
| **Thực thi kiểm thử phần mềm** | - Test plan, RTM, test cases  - Môi trường kiểm thử phần mềm sẵn sàng  - Chuẩn bị dữ liệu kiểm thử một cách đầy đủ và chính xác  - Báo cáo kiểm thử tích hợp, kiểm thử đơn vị cho bản build được kiểm thử | - Tiến hành kiểm thử phần mềm theo kế hoạch  - Đánh dấu lỗi tương ứng với từng trường hợp thất bại  - Nếu cần, hãy tiến hành cập nhật kế hoạch hay trường hợp kiểm thử  - Xác định và đánh dấu lỗi cho từng trường hợp kiểm thử trong RTM  - Kiểm thử lại tất cả những lỗi đã được sửa  - Kiểm tra hồi quy  - Theo dõi lỗi để có thể kết thúc | - Tất cả kế hoạch kiểm thử phần mềm được thực thi  - Đánh dấu lỗi cho từng trường hợp kiểm thử, theo dõi để đóng | - Hoàn thiện RTM với trạng thái đã được thực thi  - Báo cáo các lỗi  - Cập nhật kết quả tất cả trường hợp kiểm thử |
| **Kết thúc vòng đời kiểm thử phần mềm** | - Hoàn thành kiểm thử  - Đầy đủ kết quả kiểm thử  - Báo cáo đầy đủ lỗi phần mềm | - Đánh giá dựa trên các tiêu chí: mục tiêu, phạm vi, chất lượng, thời gian, chi phí  - Dựa vào các tiêu chí trên để chuẩn bị số liệu kiểm tra  - Chuẩn bị báo cáo kết thúc toàn bộ quá trình kiểm thử phần mềm  - Gửi sản phẩm và báo cáo chất lượng cho khách hàng  - Phân tích kết quả kiểm thử, chia lỗi theo loại và mức độ nghiêm trọng | - Báo cáo kết thúc kiểm thử phần mềm được xác nhận bởi khách hàng | - Số liệu kiểm thử  - Báo cáo kết thúc quá trình kiểm thử phần mềm |

## X Nêu vòng đời kiểm thử phần mềm

* Đưa ra 1 ví dụ về Stub và Driver.

Trong kiểm thử đơn vị (Unit Test) có 2 khái niệm là **Stub** và **Driver.**

* **Stub**

Khi cần kiểm tra phương thức A, nhưng phương thức A lại cần dữ liệu từ phương thức B, mà phương thức B lại chưa được viết. Trong trường hợp này ta có thể giả lập một phương thức B để có dữ kiện giúp chúng ta kiểm thử phương thức A, khi đó phương thức giả lập B sẽ gọi là **Stub**.

**Ví dụ:** Học các trường cao đăng hoặc đại học, khi cần kiểm thử chức năng đăng ký môn học nhưng phương thức kiểm tra điều kiện tiên quyết lại chưa được viết thì ta có thể giả lập phương thức kiểm tra điều kiện tiên quyết để có dữ liệu giúp kiểm thử chức năng đăng ký môn học, khi đó phương thức giả lập đó gọi là **Stub**.

* **Driver**

Ngược lại với Stub, khi chúng ta cần kiểm thử Module B hoặc phương thức B nhưng cần phải qua Module A hoặc phương thức A mới kiểm thử được B, khi đó ta có thể giả lập Module hoặc phương thức A để ta có thể vào kiểm thử được B, lúc đó phương thức giả lập A gọi là **Driver**.

**Ví dụ:** Khi cần kiểm thử chức năng đăng ký môn học nhưng chức năng đăng ký tài khoản lại chưa được viết thì ta có thể giả lập một phương thức đăng ký tài khoản để có dữ liệu sinh viên giúp ta vào kiểm thử chức năng đăng ký môn học, khi đó phương thức giả lập đăng ký tài khoản gọi là Driver.

## X Thơi điểm diễn ra kiểm thử tích hợp là khi nào ?

* Kiểm thử tích hợp xảy ra **sau kiểm thử đơn vị (Unit Test) và trước kiểm thử xác nhận**. Kiểm thử tích hợp nhận các môđun đầu vào đã được kiểm thử đơn vị, nhóm chúng vào các tập hợp lớn hơn, áp dụng các ca kiểm thử đã được định nghĩa trong kế hoạch kiểm thử tích hợp vào tập hợp đó, và cung cấp đầu ra cho hệ thống tích hợp.

## XI Kiểm thử hệ thống thì sẽ kiểm thử những gì ?

* Kiểm thử hệ thống hay còn gọi là System Testing, là kiểm tra lại toàn bộ hệ thống sau khi tích hợp. Nó cho phép kiểm tra sự tuân thủ của hệ thống theo yêu cầu. Loại kiểm thử này **kiểm tra sự tương tác tổng thể của các thành phần.** **Nó liên quan đến tải, hiệu suất, độ tin cậy và kiểm tra bảo mật**.

Tiếp Theo

## XII Kiểm thử chấp nhận – Acceptance Testing ?

Diagram

Description automatically generated

* **Kiểm thử chấp nhận - Acceptance Testing**
  + Kiểm thử chấp nhận là mức thứ 4 được thực hiện sau khi hoàn thành kiểm thử hệ thống và trước khi đưa sản phẩm vào sử dụng chính thức.
  + Kiểm thử chấp nhận chính thức liên quan đến yêu cầu và quy trình kinh doanh để xác định liệu hệ thống có đáp ứng tiêu chí chấp nhận hay không và cho phép người dùng, khách hàng hoặc tổ chức được ủy quyền khác xác định có chấp nhận hệ thống hay không.
  + Kiểm thử Alpha: được thực hiện bởi những người trong tổ chức nhưng không tham gia phát triển phần mềm.
* Giới thiệu về phương pháp kiểm thử Alpha
  + - * + Alpha testing là một dạng của Acceptance testing.
        + Được gọi là Alpha vì nó được thực hiện sớm, gần cuối của sự phát triển của phần mềm, và trước khi thử nghiệm Beta.
        + Alpha testing thực hiện để xác định tất cả các vấn đề/ lỗi có thể xảy ra trước khi phát hành sản phẩm đến tay người dùng.
        + Trọng tâm của việc kiểm thử Alpha là để người dùng thực tế trải nghiệm sản phẩm.
        + Alpha testing được thực hiện trong môi trường lab và thường các tester là nhân viên nội bộ của tổ chức, công ty.
  + Kiểm thử Beta: được thực hiện bởi khách hàng hoặc người dùng cuối tại địa điểm của người dùng cuối.
    - Giới thiệu về phương pháp kiểm thử Beta
      * Beta testing là một dạng của Acceptance testing và thực hiện bởi người dùng ngoài đội dự án phát triển.
      * Phiên bản beta của phần mềm chỉ được phát hành/công bố cho một số lượng hạn chế người dùng cuối để lấy thông tin phản hồi về chất lượng sản phẩm.
      * Beta testing làm giảm nguy cơ thất bại của sản phẩm và tăng độ tin tưởng vào chất lượng của nó thông qua các ý kiến nhận xét, đánh giá từ khách hàng.
      * Beta testing là bước kiểm tra cuối cùng trước khi chuyển một phần mềm đến tay khách hàng.
      * Lợi thế lớn nhất của beta test là phản hồi trực tiếp từ phía người dùng cuối, nó giúp kiểm tra phần mềm trong môi trường thực tế.
  + *Ví dụ kiểm thử chấp nhận : Phần mềm xuất sắc vượt qua các phép kiểm thử về chức năng thực hiện bởi nhóm thực hiện dự án, nhưng khách hàng khi kiểm thử sau cùng vẫn thất vọng vì bố cục màn hình nghèo nàn, thao tác không tự nhiên, không theo tập quán sử dụng của khách hàng…*
* Cơ sở đầu vào để kiểm thử chấp nhận
  + Quy trình nghiệp vụ
  + Yêu cầu của người dùng hoặc yêu cầu nghiệp vụ  quy định, hợp đồng pháp lý và tiêu chuẩn
  + Trường hợp sử dụng
  + Yêu cầu hệ thống
  + Tài liệu người dùng hoặc hệ thống
  + Thủ tục cài đặt
  + Báo cáo phân tích rủi ro
* Đối tượng kiểm thử chấp nhận
  + Hệ thống đang kiểm thử
  + Cấu hình hệ thống và dữ liệu cấu hình
  + Quy trình nghiệp vụ cho một hệ thống tích hợp đầy đủ
  + Hệ thống khôi phục và hot site (để kiểm thử tính liên tục của nghiệp vụ và phục hồi sau thảm họa)
  + Quy trình vận hành và bảo trì
  + Báo cáo
* Những lỗi và hỏng hóc điển hình trong kiểm thử chấp nhận
  + Luồng công việc của hệ thống không đáp ứng yêu cầu nghiệp vụ và người dùng
  + Quy tắc nghiệp vụ không được thực hiện đúng
  + Hệ thống không đáp ứng yêu cầu về hợp đồng hoặc pháp lý
  + Lỗi phi chức năng: như lỗ hổng bảo mật, hiệu suất không đủ để đáp ứng tải cao, hoặc hoạt động không đúng trên nền tảng được hỗ trợ
* **Kiểm thử khói - Smoke Testing**
  + Khi có 1 bản build mới, Test team xác định chức năng chính trong ứng dụng để thực hiện Smoke Testing xem liệu phần mềm có bị vấn đề gì ở những chức năng chính hay không.
  + Smoke testing được thực hiện trước giai đoạn kiểm thử hồi quy
  + Smoke Testing giúp xác định xem build có thiếu sót gì không để tránh lãng phí thời gian và tài nguyên.
  + Smoke Testing được thực hiện bất cứ khi nào các chức năng mới của phần mềm được phát triển và tích hợp với bản build hiện hành mà được triển khai trong môi trường kiểm thử. Nó đảm bảo rằng tất cả các chức năng quan trọng đang hoạt động chính xác hay không.
  + Ví dụ kiểm thử khói :
    - Nút đăng ký mới được bổ sung vào màn hình đăng nhập và bản build được triển khai với code mới. Chúng ta thực hiện Smoke Testing trên 1 bản build mới.
  + Cơ sở đầu vào để kiểm thử khói
    - Nội dung thay đổi của chức năng được định nghĩa bằng Task.
  + Đối tượng kiểm thử khói
    - Các tính năng mới được triển khai và tích hợp vào hệ thống.
  + Mục đích kiểm thử khói
    - Dễ dàng thực hiện việc test
    - Các lỗi sẽ được nhận diện trong giai đoạn đầu
    - Cải thiện chất lượng của hệ thống
    - Giảm thiểu rủi ro
    - Tiến trình dễ dàng truy cập
    - Tối ưu hiệu quả và thời gian của việc test
    - Dễ dàng phát hiện các lỗi quan trọng và sửa chữa các lỗi
    - Giảm thiểu các rủi ro phát sinh
  + Những lỗi và hỏng hóc điển hình trong kiểm thử khói
    - Tính năng mới tích hợp vào hệ thống được bàn giao không đúng đặc tả yêu cầu.
    - Tính năng đã bàn giao tuy nhiên hệ thống chưa sẵn sàng tích hợp.
    - Khi tích hợp vào hệ thống thì tính năng bàn giao hoạt động đúng đặc tả nhưng lại gây lỗi tới những tính năng khác.
* **Kiểm thử độ tỉnh táo - Sanity Testing**
  + Sanity testing là một loại Kiểm thử phần mềm được thực hiện sau khi nhận được một bản build phần mềm, với những thay đổi nhỏ về mã, hoặc chức năng, để xác định rằng các lỗi đã được sửa và không có vấn đề gì khác xảy ra do những thay đổi này.

*Ví dụ kiểm thử độ tỉnh táo :*

*Hệ thống phần mềm bán dừa qua internet phiên bản 1.0 với 20 chức năng khác nhau. Nhưng khi tạo một hóa đơn cho người mua dừa có lỗi không hiển thị địa chỉ của người này.*

*Người chủ muốn qua phiên bản: 2.0 sẽ sửa được lỗi trên và có thể hiển thị thêm Coupon khi tạo hóa đơn.*

*=> Là tester ta áp dụng kỹ thuật Sanity Test*

*Khi áp dụng: ta sẽ thực hiện việc testing trên hai phần:*

*Một là kiểm tra lỗi không hiển thị địa chỉ có còn hay không, hai là kiểm tra chức năng mới thêm Coupon có được thực thi vào hay không ?*

* + Cơ sở đầu vào để kiểm thử độ tỉnh táo
    - Các issues đã được tạo và có trạng thái cần testing.
  + Đối tượng kiểm thử độ tỉnh táo
    - Bug hoặc chức năng thay đổi.
  + Mục đích kiểm thử độ tỉnh táo
    - Xác định rằng chức năng được đề xuất hoạt động gần như mong đợi. Nếu kiểm tra sanity không thành công, bản build bị từ chối để tiết kiệm thời gian và chi phí liên quan đến một thử nghiệm nghiêm ngặt hơn.
    - Tiết kiệm nhiều thời gian và effort bởi vì Sanity testing tập trung vào một hoặc một vài vùng ảnh hưởng của chức năng.
    - Không mất effort để chuẩn bị tài liệu của Sanity Testing vì nó thường không có kịch bản
    - Nó giúp xác định các đối tượng thiếu phụ thuộc
    - Nó được sử dụng để verify rằng một chức năng nhỏ của ứng dụng vẫn hoạt động đúng sau thay đổi nhỏ.
  + Những lỗi và hỏng hóc điển hình trong kiểm thử độ tỉnh táo
    - Dễ bị sót bug khi Sanity testing chỉ focus vào các câu lệnh và các function của phần mềm.
    - Nó không đi đến mức cấu trúc thiết kế vì vậy rất khó để developers hiểu cách fix những issue được tìm thấy trong sanity testing.
    - Trong Sanity testing, việc test chỉ được thực hiện cho một vài chức năng hạn chế, vì vậy nếu có vấn đề xảy ra với những chức năng khác thì sẽ khó để phát hiện
    - Sanity testing thường không có kịch bản vì vậy việc tham khảo cho tương lai là không có sẵn.
* **Kiểm thử hồi quy - Regression Testing**
  + Kiểm thử hồi quy xác nhận rằng một tính năng mới được thêm không ảnh hưởng xấu đến các tính năng hiện có.
  + Kiểm thử hồi quy là kiểm thử lại các trường hợp đã được thực hiện để đảm bảo các chức năng hiện có hoạt động tốt.
  + Kiểm thử hồi quy đảm bảo rằng những thay đổi source code sẽ không ảnh hưởng tới các chức năng hiện có, đảm bảo rằng code cũ vẫn hoạt động sau khi thực hiện thay đổi source code.
  + Phương pháp Kiểm thử hồi quy
    - Retest All: Kiểm tra lại tất cả
    - Regression Test Selection: Lựa chọn kiểm tra hồi quy
    - Prioritization of Test Cases: Ưu tiên cho trường hợp thử nghiệm.
  + *Ví dụ kiểm thử hồi quy*
    - ***Ví dụ 1: Khi chức năng phần mềm được thay đổi*** *:Giả sử có một ứng dụng quản lý sinh viên, phần mềm này có chức năng thêm, sửa, xóa sinh viên và tất cả các tính năng này đang hoạt động bình thường. Tuy nhiên khách hàng muốn thay đổi nút xóa thành toggle để enable hoặc disable sinh viên. Việc thay đổi tính năng này phải được kiểm thử để đảm bảo rằng tính năng được thay đổi hoạt động bình thường và tất cả các tính năng cũ cũng được hoạt động bình thường. Quá trình này được gọi là kiểm thử hồi quy*
    - ***Ví dụ 2: Khi thêm 1 tính năng mới vào ứng dụng*** *: Phần mềm ứng dụng luôn luôn được cải tiến thêm các tính năng mới để đáp ứng nhu cầu của người dùng. Chính vì thế khi một chức năng mới được thêm vào phần mềm đã được phát triển chúng ta cũng cần thực hiện kiểm thử hồi quy trên toàn bộ ứng dụng để đảm bảo rằng việc thêm chức năng mới đó không làm ảnh hưởng đến các chức năng cũ đã được phát triển.*
    - ***Ví dụ 3: Khi developer fix bug*** *: Trong quá trình fix bug có thể developer fix được bug của chức năng A nhưng chức năng B có liên quan đến chức năng A lại bị lỗi nào đó do quá trình sửa code gây ra. Chính vì vậy khi developer thực hiện fix bug chúng ta sẽ* ***Retest*** *lại chức năng A và thực hiện* ***Regression test*** *chức năng B để đảm bảo các chức năng của hệ thống vẫn hoạt động tốt.*
  + Cơ sở đầu vào để kiểm thử hồi quy
    - Tài liệu đặc tả
    - Testcase
  + Đối tượng kiểm thử hồi quy
    - Thay đổi trong yêu cầu và source code được sửa đổi theo yêu cầu
    - Tính năng mới được thêm vào phần mềm
    - Sửa lỗi
    - Khắc phục sự cố hiệu suất
  + Mục đích kiểm thử hồi quy
    - Dễ dàng thực hiện việc test
    - Các lỗi sẽ được nhận diện trong giai đoạn đầu
    - Cải thiện chất lượng của hệ thống
    - Giảm thiểu rủi ro
    - Tiến trình dễ dàng truy cập
    - Tối ưu hiệu quả và thời gian của việc test
    - Dễ dàng phát hiện các lỗi quan trọng và sửa chữa các lỗi
    - Giảm thiểu các rủi ro phát sinh
  + Những lỗi và hỏng hóc điển hình trong kiểm thử hồi quy
    - Kiểm thử có thể vẫn còn lỗi do khi kiểm thử hồi quy liên tục được thực hiện, các bộ test cases trở nên khá lớn. Do hạn chế về thời gian và ngân sách, toàn bộ test cases kiểm thử hồi quy không thể được thực thi.
    - Tối thiểu hóa bộ test cases trong khi vẫn đạt được phạm vi kiểm thử tối đa vẫn là một thách thức
    - Mỗi lần sửa đổi hoặc mỗi lần cập nhật bản build hoặc sau một loạt các lỗi được sửa là một thách thức.

## XIII Khi nào thực hiện Smoke testing ?

* Smoke Testing được thực hiện bất cứ khi nào các chức năng mới của phần mềm được phát triển và đưa vào phiên bản có trong môi trường QA/staging. Nó đảm bảo rằng tất cả các chức năng quan trọng của sản phẩm đang hoạt động chính xác.

## XIV Khi nào thực hiện Sanity testing ?

* Sanity Testing là một loại Kiểm tra phần mềm được thực hiện **sau khi nhận được 1 bản build với những thay đổi nhỏ trong code hoặc chức năng, để chắc chắn rằng các lỗi đã được sửa và không có thêm lỗi nào được phát sinh do những thay đổi này**.

## XV Có mấy phương pháp kiểm thử hồi quy ?

* Thông thường có 3 phương pháp để kiểm tra hồi quy:

Diagram

Description automatically generated

1. Kiểm tra lại tất cả: Điều này giúp kiểm tra lại tính toàn vẹn của phần mềm từ trên xuống dưới.
2. Lựa chọn kiểm tra hồi quy: Phương pháp này cho phép nhóm chọn một lựa chọn đại diện cho các thử nghiệm sẽ xấp xỉ một thử nghiệm đầy đủ. Điều này đòi hỏi ít thời gian hoặc chi phí hơn nhiều so với bộ thử nghiệm đầy đủ
3. Ưu tiên cho trường hợp thử nghiệm: Với các trường hợp thử nghiệm hạn chế, hãy thử ưu tiên các thử nghiệm có thể ảnh hưởng đến cả bản dựng hiện tại và tương lai của phần mềm.

## XVI Khi nào sử dụng kiểm thử chấp nhận ?

* Đây thường là bước cuối cùng trước khi sản phẩm được đưa ra hoạt động hoặc trước khi phân phối sản phẩm phải được chấp nhận.  
  Acceptance Testing được thực hiện sau khi bản thân sản phẩm được kiểm tra kỹ lưỡng (tức là sau khi kiếm thử hệ thống ).