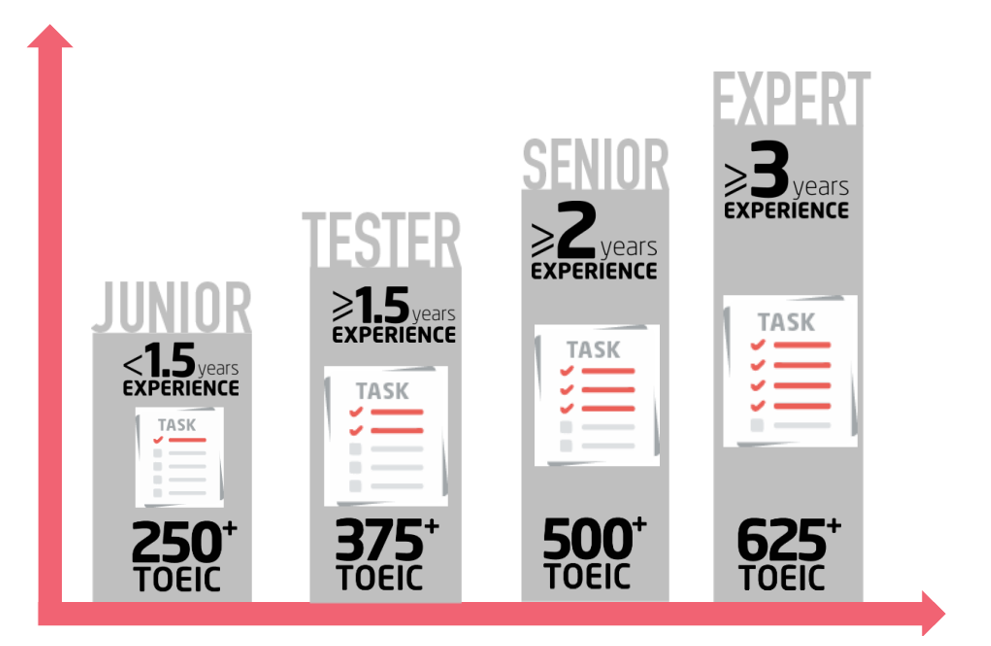
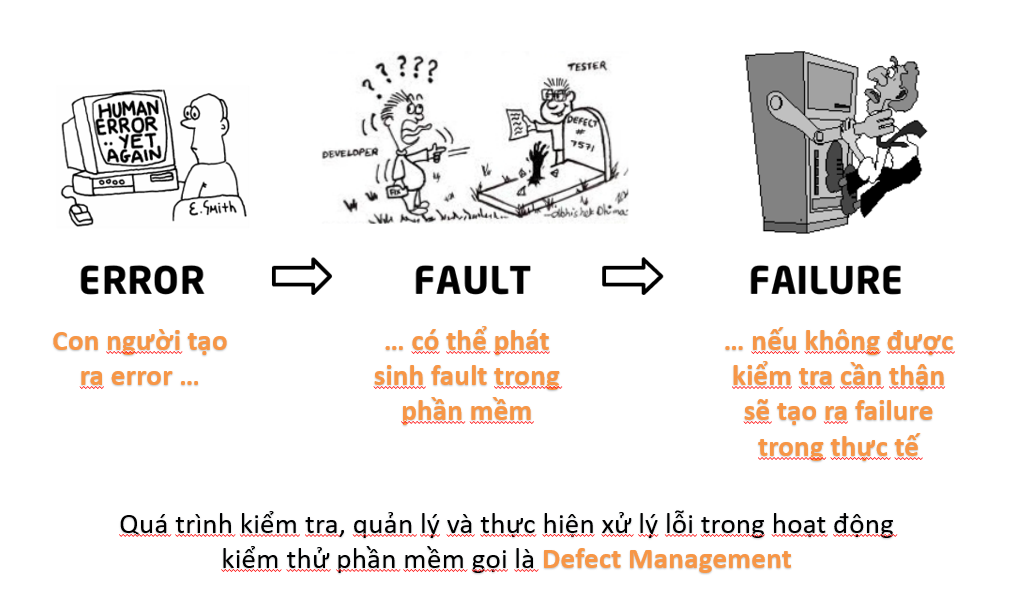
SOF303 - Kiểm thử cơ bản

1. Kiểm thử khái niệm:
   1. **Kiểm thử phần mềm là gì?** *Kiểm thử hay còn gọi là testing, là quá trình đánh giá một hệ thống hay các thành phần của nó với mục đích tìm xem liệu hệ thống có đáp ứng các yêu cầu được đã được chỉ định hay không. Nói một cách đơn giản, kiểm thử được thực hiện trên một hệ thống để xác định bất kỳ lỗ hổng, các lỗi hoặc các yêu cầu đang bị thiếu hay trái ngược với các yêu cầu thực tế đã được đề ra.* *Theo tiêu chuẩn ANSI / IEEE 1059, kiểm thử có thể được định nghĩa là quá trình phân tích các thành phần của phần mềm để phát hiện sự khác biệt giữa những điều kiện của phần mềm đang tồn tại thực tế và những điều kiện được yêu cầu (****đó là defects – thiếu sót/ errors – sai sót / bugs – lỗi)*** *và từ đó có thể đánh giá được chất lượng của chất lượng của phần mềm.*
   2. **Ai là người kiểm thử? Tester là gì?** *Người kiểm thử hay còn gọi là tester, họ là những người có nhiệm vụ tìm ra lỗi của phần mềm, điều này sẽ phụ thuộc vào quy trình và các bên liên quan trong dự án.*
      1. Trong hầu hết các trường hợp, các chuyên gia kiểm thử cho một hệ thống trong phạm vi năng lực tương ứng của họ có thể là:
         * Software Tester – Nhân viên kiểm thử phần mềm
         * Software Developer – Nhân viên phát triển phần mềm
         * Project Lead/Manager - Leader hoặc Manager của dự án
         * End User - Người dùng cuối cùng
   3. **Mục tiêu của kiểm thử phần mềm?**
      1. **Mục đích đầu tiên là tìm lỗi sản phẩm**
      2. **Ngăn ngừa lỗi còn quan trọng hơn tìm lỗi** (*Trong quá trình test mình sẽ gặp những lỗi tương tự nhau nhưng phát sinh ở những trường hợp khác nhau, đặc biệt là cùng từ 1 Dev tạo ra thì phải thảo luận với Dev đó để ngăn ngừa những lỗi đó tiếp theo*)
      3. **Đánh giá chất lượng sản phẩm** (Bạn phải báo cáo tiến độ với PM hoặc Lead để có thể đưa ra 1 kế hoạch chuẩn mực để sản phẩm có đến được tay khách hàng đúng hẹn hay không. Tìm bug và đảm bảo chất lượng sản phẩm tốt nhất trước khi đưa đến tay người dùng mới là điều mà các tester cần nỗ lực.)
      4. **Đưa ra những ý tưởng cải thiện sản phẩm**(Bạn hoàn toàn có thể cùng Dev đưa ra các ý kiến đóng góp cải thiện chức năng, bố cục để giúp sản phẩm được hoàn thiện hơn. Nhưng để có được những đóng góp giá trị bạn cần phải hiểu về sản phẩm của bạn đang làm và luôn nhớ **“Để tạo ra 1 sản phẩm giá trị không đến từ ngẫu hứng mà phải nỗ lực không ngừng nghỉ”**)
      5. **Hiểu và làm rõ yêu cầu thực tế của khách hàng:** Khi bàn giao cho **End-User(Người dùng cuối)** thì nhiệm vụ của tester cần biết khách hàng sử dụng sản phẩm của họ như thế nào và nên viết ra cách kịch bản kiểm thử và thiết kế các Test Case. Việc này sẽ giúp rất nhiều trong việc hoành thành tất cả các yêu cầu của khách hàng.
      6. **Nỗ lực để làm việc hiệu quả:** Sản phẩm có thể mất **3 tháng** để code nhưng chỉ có **1 tháng** để test vì vậy bạn phải tận dụng toàn bộ thời gian đo và kinh nghiệm để test hiệu quả nhất và đảm bảo đúng hạn.
   4. **Để đạt được sự tín nhiệm của khách hàng bằng cách cung cấp cho họ một sản phẩm chất lượng như nào?**
      1. Phải đạt được các mục tiêu của ý 3 ở trên để nâng cao chất lượng của sản phẩm và ban giao cho khách hàng một sản phẩm hoàn hảo nhất có thể.
      2. Luôn giữ cho nhiệt huyết công việc bùng cháy: có thể bạn nghĩ rằng việc mình làm thật nhỏ bé nhưng hãy luôn nghĩ rằng nhiệm vụ của bạn sẽ đem ra những sản phẩm thật chất lượng.
   5. **Tại sao lại phải kiểm thử phần mềm?**
      1. Kiểm thử phần mềm là khâu vô cùng quan trọng trong quá trình phát triển 1 sản phẩm công nghệ. Nó chỉ ra những lỗi và sai sót đã được thực hiện trong các giai đoạn phát triển.
      2. Nó giúp đảm bảo chất lượng sản phẩm khi đến được tay khách hàng.
      3. Khiến sản phẩm tăng chất lượng và độ uy tín của công ty bạn.
   6. **Vai trò của kiểm thử viên?**
      1. Trong các giai đoạn lập kế hoạch và chuẩn bị thử nghiệm, người kiểm thử nên xem xét và đóng góp vào các kế hoạch thử nghiệm, cũng như phân tích, xem xét và đánh giá các yêu cầu và thông số kỹ thuật thiết kế. Họ có thể tham gia hoặc thậm chí là những người chính xác định các điều kiện thử nghiệm và tạo ra các thiết kế thử nghiệm, các trường hợp kiểm tra, các đặc tả thủ tục kiểm tra và dữ liệu thử nghiệm, và có thể tự động hóa hoặc giúp tự động hóa các thử nghiệm.
      2. Trong suốt vòng đời kiểm thử phần mềm , họ xem xét công việc, hỗ trợ lẫn nhau, bao gồm các thông số kiểm tra, báo cáo lỗi và kết quả kiểm tra.
   7. **3 Nhân tố tạo chất lượng sản phẩm là gì?**
      1. **CON NGƯỜI - CÔNG CỤ - QUY TRÌNH**
   8. **Tesing và Debugging là gì?**
      1. **Tesing:** Hoạt động tìm lỗi được thực hiện bởi kiểm thử viên và tìm được càng nhiều lỗi càng tốt.
      2. **Debugging:** Hoạt động sửa lỗi được thực hiện bởi DEV – Loại bỏ các lỗi hiện hình và những lỗi tìm được.
   9. **Phần mềm phát sinh lỗi có ảnh hưởng gì không?**
      1. **Mất tiền để xử lý**
      2. **Mất thời gian để xử lý**
      3. **Gây ra tổn thương với con người:** như gây chết người và thiệt hại lớn về kinh tế.
      4. **Ảnh hưởng đến uy tín của công ty**
   10. **Khi nào thì kết thúc kiểm thử?**
       1. Hạn chót phát triển phần mềm (deadline).
       2. Đã thực hiện hết các test-case được đề ra.
       3. Hoàn thiện các chức năng và toàn bộ code đã đảm bảo được các yêu cầu đề ra.
       4. Tỷ lệ bug phải trong giới hạn mong muốn.
       5. Không có những bug nghiêm trọng.
       6. Quyết định của quản lý dự án PM.
   11. **Tính cách cần có để trở thành Tester:** - Tính kỷ luật – Tính kiên trì – Tính linh hoạt – Tính nhạy cảm – Tính cởi mở.
   12. **Thái độ Tester:** Luôn đặt câu hỏi – Tìm kiếm lỗi có ý nghĩa – Trung Thực – Đừng mong đợi mọi người hiểu công việc của mình.
   13. **Kỹ năng mềm:** Đọc tài liệu – Giao tiếp – Viết báo cáo – Lập kế hoạch – Làm việc nhóm – Đàm phán
   14. **Tương lai của Tester đi về đâu?**



1. Một số khái niệm cơ bản trong phát triển phần mềm:
   1. **Business Analysis / Business Analyst (BA)-** là người phân tích nghiệp vụ của dự án trong lĩnh vực CNTT. BA là cầu nối giữa đội phát triển phần mềm và khách hàng.
   2. **Project Management / Project Manager** **(PM)**- là người Quản Lý Dự Án.
   3. **Leader –** là người trưởng nhóm có tham gia lập trình sử dụng các ngôn ngữ để phát triển phần mềm và quản lý các TASK công việc cùng PM cho các DEV.
   4. **Developer (Dev)-** là người lập trình viên tham gia dự án và sử dụng các ngôn ngữ lập trình để phát triển phần mềm.
   5. **Tester:** Người kiểm thử dự án.
   6. **Quality Assurance (QA) -** là người chịu trách nhiệm đảm bảo chất lượng sản phẩm thông qua việc đưa ra **quy trình làm việc** giữa các bên liên quan.
2. Các khái niệm căn bản trong kiểm thử
3. **DEFECT:** mang ý nghĩa là bạn đã gặp trường hợp không dự tính trước được.
4. **CRASH:** mang ý nghĩa chương trình, hệ thống của bạn gặp một tai nạn trong quá trình thực hiện.
5. **BUG:** Là các sai sót không dự tính được trước, không thể kiểm soát trước khi phát hành sản phẩm. Bug được phát hiện sau khi đã đưa chương trình vào kiểm thử ở các trường hợp cụ thể (Gọi là test) hoặc sau khi phát hành sản phẩm. Bug có thể chỉ là một lỗi rất nhỏ, nhưng nó lại làm cho chương trình chạy sai hoàn toàn.
6. **Error:** Hành động của con người gây ra một kết quả không đúng. Bởi các lý do sau: Do một số nhầm lẫn trong việc tìm hiểu chức năng của phần mềm – tính toán sai giá trị khởi tạo.....(Viết chương trình xong không thể biên dịch)
7. **Fault:** Một bước sai, quy trình sai hay dữ liệu sai – là kếu quả của Error. – Thường mang ý nghĩa đã có sự không chính xác các bước đã được quy định, không tuân theo quy trình dẫn đến chương trình thực hiện một cách không xác định, hoặc chương trình không kiểm soát được.
8. **Failure(Thất bại):** Một lỗi được tìm thấy trong quá trình sử dụng thực tế - Là khái niệm trong trường hợp hệ thống, chương trình, hoặc một thành phần nào đó không đủ khả năng thực hiện do các yêu cầu để thực hiện nó là không đủ, không chính xác.
9. **Quy trình phát triển phần mềm và chạy dự án:**

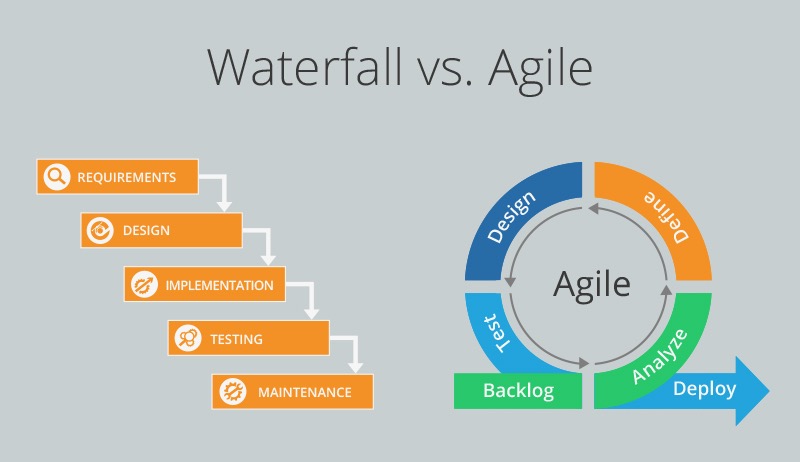
**Phương pháp phát triển phần mềm là gì?**

* 1. Maintenance: bao gồm điều chỉnh các lỗi mà chưa được phát hiện trong các giai đoạn trước của chu kỳ sống của một phần mềm.



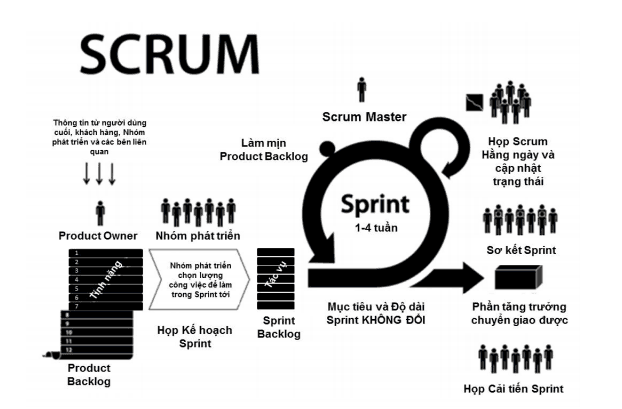
**Agile**: là một phương pháp phát triển phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng nhanh càng tốt. **Tuyên ngôn: Phát triển phần mềm linh hoạt.** Trong suốt dự án chúng ta sẽ có thêm các thông tin mới. Khi đã Agile thì ta thừa nhận chuyện này từ khi chưa bắt đầu. Chúng ta bắt đầu dự án với lượng thông tin cần thiết tối thiểu. Thường xuyên dành thời gian để xem lại những điều ta mới biết thêm cũng như điều gì đã thay đổi trong quá trình làm dự án. Quan trọng nhất, chúng ta cho phép có thay đổi đối với cả dự án và quy trình làm việc.





**Scrum**: là một quy trình phát triển phần mềm theo phương pháp Agile. Vì thế nó tuần thủ các nguyên tắc của Agile.

* Một cuộc họp ngắn được tổ chức hàng ngày của mỗi Nhóm Phát triển, trong thời gian đó các thành viên của nhóm kiểm tra công việc của họ, đồng bộ hóa công việc và tiến độ của mình và báo cáo các vấn đề để giải quyết. Nhóm và Scrum Master có thể phải tiến hành các hoạt động tiếp theo Daily Scrum để thích ứng với công việc sắp tới và tối tưu hóa Sprint.
* **Release** – Bản phát hành
* **ScrumMaster:** Người chịu trách nhiệm cho quy trình Scrum, bao gồm việc triển khai đúng quy tắc, và tối hóa lợi ích từ Scrum.
* **Sprint Backlog**: Danh sách các nhiệm vụ xác định công việc của nhóm trong một Sprint. Danh sách này được cập nhật trong suốt Sprint. Nhóm tự quản lý Sprint Backlog bằng việc cập nhật trạng thái thực thi các nhiệm vụ với người chịu trách nhiệm và thời gian còn lại để hoàn tất công việc tính tới thời điểm cập nhật.
* **Sprint Review meeting** - Họp Sơ kết Sprint: Đội sản xuất cùng với Product Owner đánh giá lại công việc của Sprint vừa kết thúc. Trong khi đánh giá, Đội sản xuất trình bày các phần việc đã hoàn tất, các chức năng đã hoàn thành; lắng nghe các phản hồi từ Product Owner và thảo luận về các chỉnh sửa (nếu có) sẽ thực hiện trong Sprint tiếp theo.



**Phương pháp nào tốt nhât?** Tùy vào như cầu, điều kiện và chiến lược riêng của từng doanh nghiệp để lựa chọn ra phương pháp phù hợp nhất.

1. **Quy trình kiểm thử:**

**Thởi điểm nào thì nên bắt đầu test?** Việc kiểm thử – test sớm sẽ giúp giảm chi phí và thời gian để xây dựng lại và sửa lỗi để bàn giao sản phẩm phần mềm. Trong một vòng đời của phần mềm, việc test nên được bắt đầu từ khi có những yêu cầu từ phía khách hàng và được kéo dài đến cho đến khi triển khai phầm mềm. Thời điểm bắt đầu kiểm thử còn phụ thuộc vào mô hình phát triển phần mềm đang được sử dụng.

Việc test được thể hiện theo nhiều dạng công việc khác nhau ở các giai đoạn khác trong suốt vòng đời phát triển phần mềm:

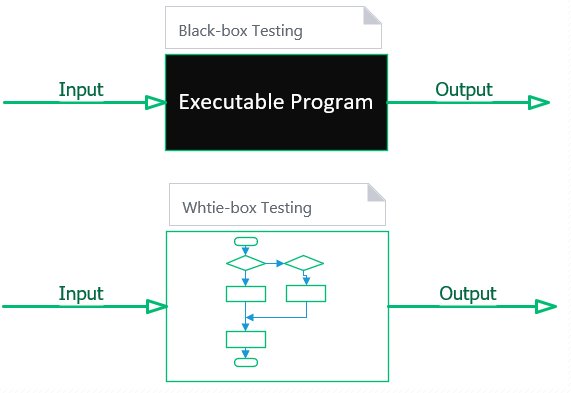
Trong quá trình tập hơp yêu cầu (requirement gathering phase) : phân tích và xác minh yêu cầu cũng được coi là kiểm thử – test requirement.

Trong giai đoạn thiết kế (design phase) : kiểm tra lại thiết kế với mục đích cải thiện thiết kế cũng được tính là kiểm thử.

Trong giai đoạn phát triển phần mềm (implement phase) : kiểm thử được thực hiện bởi lập trình viên – unit testing cũng được tính là kiểm thử.

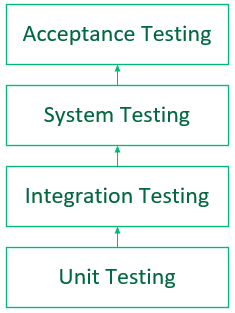
**Phân loại Tesing:**

Theo phương thức tiếp cận có thể chia làm 2 loại testing: phương thức kiểm thử hộp đen (black-box testing) và phương thức kiểm thử hộp trắng (white-box testing).

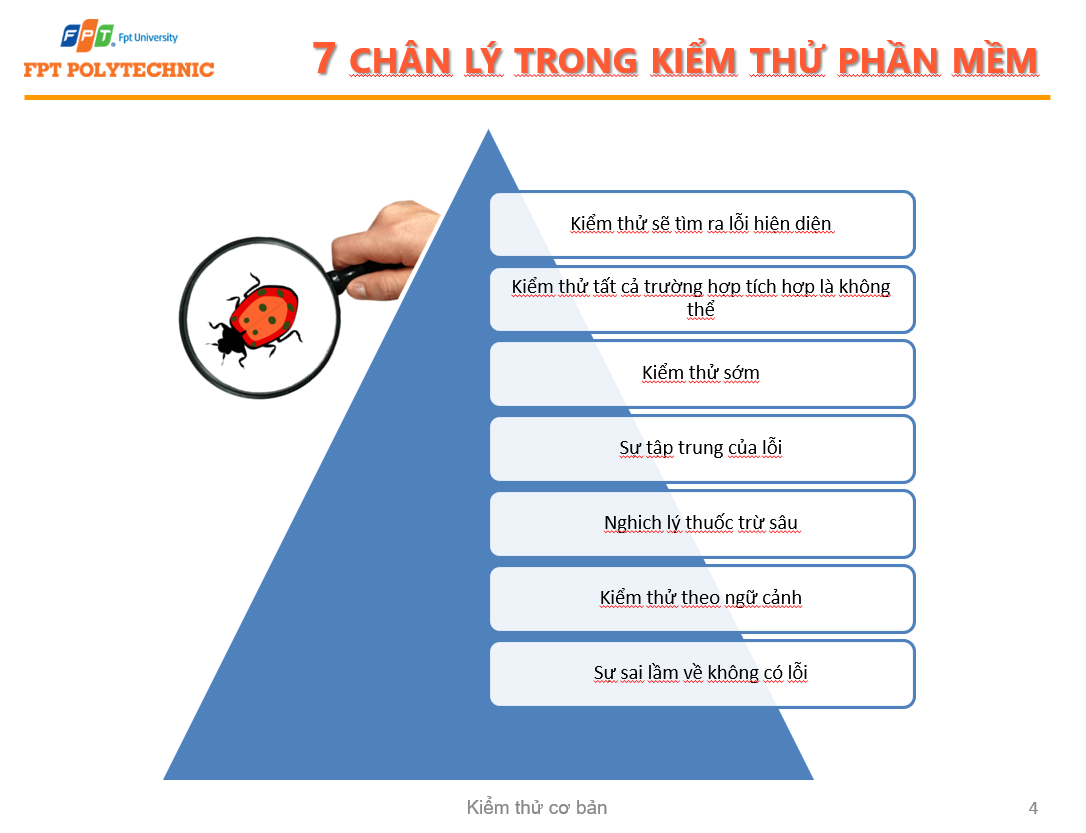


* **Black-box testing** : chúng ta không cần quan tâm những gì xử lý trong cái hộp đen đó làm, chỉ cần cho đầu vào và xác nhận đầu ra. Đây chính là những gì bộ phận QA độc lập thực hiện.
* **White-box testing** : chúng ta cần quan tâm những gì xử lý trong hộp trắng đó làm, tức là code được viết như thế nào, kiến trúc, logic xử lý ra sao.

Thông thường**, Black-box testing** sẽ là Acceptance Testing do bộ phận **QA** thực hiện, **White-box** testing sẽ là **Unit Testing**/ **Integration Testing**/ **Functional Testing** do **Tester** hoặc **Dev** thực hiện .



**Không thể kiểm thử tất hết tất cả mà cần phải tập trung vào nội dung cần thực hiện.**



**1 Kiểm thử đưa ra lỗi**

Kiểm thử có thể cho thấy rằng phần mềm đang có lỗi, nhưng không thể chứng minh rằng phần mềm không có lỗi. Kiểm thử được thực hiện bằng những kĩ thuật khác nhau. Kiểm thử làm giảm xác suất lỗi chưa tìm thấy vẫn còn trong phần mềm, ngay cả khi đã kiểm thử nghiêm ngặt phần mềm vẫn có thể còn lỗi. Vì vậy chúng ta phải tìm được càng nhiều lỗi càng tốt.

**2 Kiểm thử tất cả các trường hợp là không thể**

Nguyên lý này nói rằng, kiểm tra mọi thứ trong phần mềm một cách trọn vẹn là không thể. Kiểm thử với tất cả các kết hợp đầu vào và đầu ra, với tất cả các kịch bản là không thể trừ phi nó chỉ bao gồm ít trường hợp thì có thể kiểm thử toàn bộ. Thay vì kiểm thử toàn bộ, việc phân tích rủi ro và dựa trên sự mức độ ưu tiên chúng ta có thể tập trung việc kiểm thử vào một số điểm cần thiết, có nguy cơ lỗi cao hơn.

**3 Kiểm thử càng sớm càng tốt**

Nguyên lý này yêu cầu bắt đầu thử nghiệm phần mềm trong giai đoạn đầu của vòng đời phát triển phần mềm. Các hoạt động kiểm thử phần mềm từ giai đoạn đầu sẽ giúp phát hiện bug sớm hơn. Nó cho phép chuyển giao phần mềm theo yêu cầu đúng thời gian với chất lượng dự kiến.

**2.4 Sự tập trung của lỗi**

Thông thường, lỗi tập trung vào những module, thành phần chức năng chính của hệ thống. Nếu xác định được điều này bạn sẽ tập trung vào tìm kiếm lỗi quanh khu vực được xác định. Nó được coi là một trong những cách hiệu quả nhất để thực hiện kiểm tra hiệu quả.

**2.5 Nghịch lý thuốc trừ sâu**

Nếu bạn sử dụng cùng một tập hợp các trường hợp kiểm thử liên tục, sau đó một thời gian các trường hợp kiểm thử không tìm thấy lỗi nào mới. Hiệu quả của các trường hợp kiểm thử bắt đầu giảm xuống sau một số lần thực hiện, vì vậy luôn luôn chúng ta phải luôn xem xét và sửa đổi các trường hợp kiểm thử trên một khoảng thời gian thường xuyên.

**2.6 Kiểm thử phụ thuộc vào ngữ cảnh**

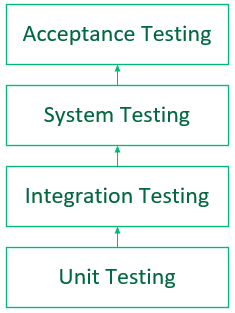
Theo nguyên tắc này thì việc kiểm thử phụ thuộc vào ngữ cảnh và chúng ta phải tiếp cận kiểm thử theo nhiều ngữ cảnh khác nhau.

Nếu bạn đang kiểm thử ứng dụng web và ứng dụng di động bằng cách sử dụng chiến lược kiểm thử giống nhau, thì đó là sai. Chiến lược để kiểm thử ứng dụng web sẽ khác với kiểm thử ứng dụng cho thiết bị di động của Android.

**2.7 Không có lỗi - Sai lầm**

Việc không tìm thấy lỗi trên sản phẩm không đồng nghĩa với việc sản phẩm đã sẵn sàng để tung ra thị trường. Việc không tìm thấy lỗi cũng có thể là do bộ trường hợp kiểm thử được tạo ra chỉ nhằm kiểm tra những tính năng được làm đúng theo yêu cầu thay vì nhằm tìm kiếm lỗi mới.

Các loại kiểm thử



* 1. **Unit Testing (kiểm thử đơn vị)**
     1. **Định nghĩa**: **Unit Testing (UT)** là một mức kiểm thử phần mềm với mục đích để xác nhận từng unit của phần mềm được phát triển đúng như được thiết kế. UT là mức test nhỏ nhất trong bất kỳ phần mềm nào. UT bản thân nó là cái gì đó khá trừu tượng vì tùy dự án, chúng ta có thể quy định “unit” ở mức độ nào. Thông thường, “unit” sẽ được quy định giới hạn trong một hàm (method) hay một class. Trong thực tế, tùy vào kinh nghiệm và kĩ năng, developer sẽ đưa ra quyết định viết các UT như thế nào cho phù hợp, có thể test đầu vào đầu ra của hàm, hay kiểm tra một phần hoặc toàn bộ class.
     2. **Tại sao phải unit test?**
        + Đôi khi do tính chủ quan của **dev**, họ thường không kiểm tra lại đoạn code mình đã phát triển dẫn đến lỗi hệ thống hoặc trả ra kết quả không mong đợi. Những ảnh hưởng trên dẫn đến việc tốn kém thời gian và tiền bạc để sửa lỗi.
        + **UT** còn giúp developer tăng khả năng tối ưu hóa code của mình hơn, nâng cao khả năng tư duy về code.
        + Việc xử lý những vấn đề sớm khi **UT** có thể xử lý rất nhiều vấn đề khác có thể xảy ra sau này trong quá trình **development** và **testing**.
        + Chi phí cho **UT** thì ít hơn nhiều so với các **phase testing** sau như là **system testing**, **acceptance testing** và nhất là chi phí khi issues/bus qua đến bên khách hàng.
        + Đo lường được những đoạn code nào đã pass, những đoạn code nào chưa pass. Và **dev** có thể cover ngay đoạn code đó sau khi **chạy** UT.
        + Giảm lượng **bugs** phát sinh ở các giai đoạn **testing** tiếp theo (**system testing**, **acceptance testing**, …).
        + Phát hiện sớm các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.
        + Tiết kiệm thời gian **development**. Việc viết **UT** và **execute** có thể sẽ tốn nhiều thời gian nhưng nó làm cho code đầy đủ hơn và phát sinh ít **issues/bugs** hơn. Thay vì coding nhanh và phát sinh rất nhiều **issues** thì hay coding song song với viết **UT** và **execute**, có thể sẽ tốn nhiều thời gian hơn bình thường nhưng sẽ hạn chế những phát sinh sau này.
        + Tạo hàng rào an toàn cho các đoạn **code**: bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.
     3. **Khi nào thực hiện? UT** là mức kiểm thử đầu tiên trong các mức kiểm thử phần mềm, nó được thực hiện trước khi **Integration Testing**.
     4. **Ai thực hiện? UT** thường do lập trình viên thực hiện. **UT** nên được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt quá trình phát triển phần mềm.
     5. **Mục đích**
        + Cô lập từng thành phần của chương trình và chứng minh các bộ phận riêng lẻ chính xác về các yêu cầu chức năng.
        + Tăng sự đảm bảo khi có sự thay đổi mã.
        + Code dễ sử dụng, dễ hiểu, có thể tái sử dụng nhiều hơn, debug dễ dàng.
        + Chi phí sửa lỗi thấp hơn so với các mức kiểm thử giai đoạn sau.
     6. **Demo Test Case trên chính dự án JAVA1.**
  2. **Integration Testing (kiểm thử tích hợp)**
     1. **Định nghĩa: Integration Testing (IT)** là một mức của kiểm thử phần mềm với mục đích kiểm tra một nhóm các module nhỏ liên quan đến nhau xem chúng có hoạt động được với nhau đúng chức năng như trong thiết kế hay không. Theo **ISTQB ( International Software Testing Qualifications Board):**
        + Kiểm thử tích hợp được thực hiện để phát hiện các lỗi về **giao diện** hoặc **trong tương tác giữa các thành phần** hoặc hệ thống tích hợp
        + Kiểm thử tích hợp thành phần: kiểm tra sự tương tác giữa các thành phần với điều kiện 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 con khác nhau và các hệ thống này đã pass ở lần kiểm thử trước đó.
     2. **Tại sao phải Integration Testing?** Mặc dù mỗi module đều được **unit test** nhưng các lỗi vẫn còn tồn tại với các lý do khác nhau:
        + Một Module nói chung được thiết kế bởi một developer có kiến thức và logic lập trình có thể khác với các lập trình viên khác. Kiểm thử tích hợp là cần thiết để đảm bảo tính hợp nhất của phần mềm.
        + Tại thời điểm phát triển module vẫn có thể có thay đổi trong đặc tả yêu cầu của khách hàng, những thay đổi này có thể không được kiểm tra ở giai đoạn unit test trước đó.
        + Giao diện và cơ sở dữ liệu của các module có thể chưa hoàn chỉnh khi được tíc hợp lại.
        + Khi tích hợp hệ thống các module có thể không tương thích với cấu hình chung của hệ thống.
        + Thiếu các xử lý ngoại lệ có thể xảy ra.
     3. **Khi nào thực hiện? IT** là mức thứ 2 trong các mức kiểm thử phần mềm. Nó được thực hiện sau **Unit Test** và trước **System Test**.
     4. **Ai thực hiện?** **IT** có thể được thực hiện bởi **developer**, một **test team chuyên biệt** hay một nhóm chuyên developer/kiểm thử viên tích hợp bao gồm cả kiểm thử phi chức năng.
     5. **Mục đích**
        + Kiểm tra sự tích hợp 1 nhóm các thành phần riêng lẻ có liên quan xem chúng có hoạt động đúng như mong đợi hay không.
        + Nhằm phát hiện lỗi giao tiếp xảy ra giữa các thành phần cũng như lỗi của bản thân từng thành phần (nếu có). Phát hiện lỗi giao tiếp xảy ra giữa các **Unit**.
        + Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ **(sub-system)** và cuối cùng là nguyên hệ thống hoàn chỉnh **(system)** chuẩn bị cho kiểm tra ở mức hệ thống **(System Test).**
        + Thành phần có thể là:  các **module**,  các ứng dụng riêng lẻ, các ứng dụng **Client/ Server.**
     6. **Phân loại** Có 4 loại kiểm tra trong **Integration Test.**
        + **Kiểm tra cấu trúc (structure)**: Tương tự **White Box Testing** (**kiểm tra nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng),** chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn các lệnh và nhánh bên trong.
        + **Kiểm tra chức năng (functional)**: Tương tự **Black Box Test** **(kiểm tra chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong)**, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
        + **Kiểm tra hiệu năng (performance)**: Kiểm tra việc vận hành của hệ thống.
        + **Kiểm tra khả năng chịu tải (stress)**: Kiểm tra các giới hạn của hệ thống.
  3. **System Testing (kiểm thử hệ thống)**
     1. **Định nghĩa** System Testing là một mức của kiểm thử phần mềm. Giai đoạn này sẽ hoàn thiện và hợp nhất phần mềm để kiểm thử. Theo **ISTQB** định nghĩa: quy trình của kiểm thử tích hợp hệ thống để xác nhận xem hệt hống phần mềm có đáp ứng đúng theo đặc tả yêu cầu.
     2. **Tại sao cần system test?** 
        + **Sau khi hoàn thành quá trình test tích hợp** chúng ta cần phải kiểm tra thêm về độ tương thích và tương tác với các thiết bị ngoại vi bên ngoài của ứng dụng để kiểm tra tính khả dụng của nó.
        + **System test** là việc xác minh kiểm tra kỹ lưỡng của mỗi đầu vào trong các ứng dụng để kiểm tra các kết quả mong muốn.
        + Thử nghiệm các kinh nghiệm của người dùng với các ứng dụng.
     3. **Khi nào thực hiện?** Kiểm thử hệ thống là mức kiểm thử thứ 3 trong các mức kiểm thử phần mềm được thực hiện sau kiểm thử tích hợp **(Integration Testing)** và trước kiểm thử chấp nhận (**Acceptance Testing)**.
     4. **Ai thực hiện?** Thông thường, các **tester** thực hiện kiểm thử hệ thống. Các **tester** này thường hoàn toàn độc lập với nhóm phát triển dự án.
     5. **Mục đích:** Mục đích của giai đoạn này là để đánh giá sự hoạt động của hệ thống có đúng theo như tài liệu đặc tả.
     6. **Phân loại System Testing**
        + **Basic testing :** để chứng tỏ hệ thống có thể cài đặt được, cấu hình được và hoạt động được.
        + **Functional Testing :** kiểm thử chức năng nhằm đảm bảo chức năng phần mềm được vận hành theo đúng mục đích (**requirements**) đưa ra.
        + **Scalability testing :** xác định giới hạn quy mô của hệ thống, như quy mô người dùng, quy mô địa lý, quy mô nguồn lực.
        + **Reliability testing :** đánh giá khả năng hệ thống giữ hoạt động trong thời gian dài mà không gây ra **failures**.
        + **Documentation testing :** đảm bảo system’s user guides là chính xác và khả dụng.
        + **Usability Testing :** kiểm tra tính khả dụng của ứng dụng, chủ yếu tập trung kiểm tra sự dễ sử dụng, tính linh hoạt, tính thân thiện của sản phẩm.
        + **Load and Stability Testing :**kiểm tra khả năng ổn định của hệ thống trong thời gian dài với toàn tải.
        + **Performance testing :** đánh giá hiệu năng của hệ thống chẳng hạn băng thông, phản hồi dưới các điều kiện khác nhau.
        + **Regression Testing :** kiểm thử hồi quy tập trung vào việc tìm kiếm lỗi sau khi xảy ra một thay đổi mã chính. Đảm bảo hệ thống vẫn ổn định khi tích hợp thêm các hệ thống con khác và khi bảo trì.
        + **Recovery Testing :** kiểm thử phục hồi là kiểm thử được thực hiện sau khi có các sự cố hệ thống dẫn đến chương trình lỗi, không hoạt động được. Kiểm thử phục hồi được thực hiện để đảm bảo chương trình sau khi khắc phục lỗi trên không xảy ra bug
        + **Migration Testing :** kiểm thử di động được thực hiện để đảm bảo tính linh động của phần mềm, phần mềm có thể di chuyển được chuyển từ cơ sở hạ tầng hệ thống cơ sở hạ tầng hệ thống cũ để hiện tại không có bất kỳ vấn đề.
        + **Hardware/Software Testing :** đây là khi các thử nghiệm tập trung chú ý về sự tương tác giữa các phần cứng và phần mềm trong quá trình thử nghiệm hệ thống.
        + **Robustness testing :** xác định xem khả năng phục hồi của hệ thống từ **input errors** và các tình huống **failure** khác.
        + **Inter-operability testing :** xác định xem hệ thống có thể tương thích (**inter-operate**) với các sản phẩm của bên thứ 3.
     7. **Sự khác nhau giữa Integration Test và System Testing** Testing Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Testing chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện **Unit Test** và **Integration Test** để bảo đảm mọi **Unit** và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện **System Testing.**
  4. **Acceptance Test (kiểm thử chấp nhận)**
     1. **Định nghĩa Acceptance Testing** về cơ bản là mô phỏng các thao tác của người dùng sử dụng sản phẩm để xem kết quả người dùng nhận được có đúng như mong muốn không.
     2. **Khi nào thực hiện? Acceptance Testing** 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.
     3. **Ai thực hiện?**
        + Kiểm thử chấp nhận được chia thành 2 mức khác nhau:
          - **Kiểm thử alpha:** được thực hiện bởi những người trong tổ chức nhưng không tham gia phát triển phần mềm.
          - **Kiểm thử beta:** được thực hiện bởi khách hàng/ người dùng cuối tại địa điểm của người dùng cuối (hoặc ủy quyền cho một nhóm thứ ba thực hiện).
        + **Lưu ý:** không nhất thiết phải thực hiện tất cả các loại kiểm tra nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tuỳ khả năng và thời gian cho phép của dự án, khi lập kế hoạch, quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.
     4. **Mục đích**
        + Đảm bảo phần mềm đáp ứng đúng yêu cầu nghiệp vụ của khách hàng. Sản phẩm nhận được sự chấp nhận từ khách hàng/ người dùng cuối.
  5. **Manual Testing và Automation Testing**
     1. **Manual Testing** : là việc thử nghiệm một phầm mềm hoàn toàn được làm bằng tay bởi người tester. Nó được thực hiện nhằm phát hiện lỗi trong phầm mềm đang được phát triển. Trong **manual testing, tester** sẽ thực hiện các trường hợp kiểm thử và tạo báo cáo kiểm thử hoàn toàn thủ công mà không có bất kỳ sự trợ giúp của công cụ tự động nào.
     2. **Automation Testing** : là phương pháp kiểm thử tự động. Người **tester** sẽ phải viết các kịch bản kiểm thử sau đó sử dụng các **tool** hỗ trợ để thực hiện kiểm thử, phương pháp này sẽ giúp việc kiểm thử hiệu quả và tốn ít thời gian hơn. **Automation testing** giúp chạy các kịch bản kiểm thử lặp lại nhiều lần và các **task** kiểm thử khác khó thực hiện bằng tay như **performance testing** và **stress testing.**
  6. **Ưu, nhược điểm Manual Testing**
     1. **Ưu điểm:**
        + Dễ dàng cho việc **test** giao diện, người **tester** sẽ có phản hồi nhanh và trực quan về giao diện ứng dụng.
        + Mất ít chi phí cho các tool tự động và quy trình.
        + Khi có thay đổi nhỏ manual testing manual testing không bị mất nhiều thời gian để thay đổi các trường hợp kiểm thử.
     2. **Nhược điểm:**
        + Kết quả kiểm thử ít tin cậy hơn vì có thể sai xót do yếu tố con người.
        + Qúa trình thực hiện các ca kiểm thử không được ghi lại, do vậy nó không có tính tái sử dụng.
        + Với một số **task** khó thực hiện thủ công như **performance testing** và **stress testing** thì **manual testing** rất khó để thực hiện.
  7. **Ưu, nhược điểm Automation Testing**
     1. **Ưu điểm**
        + Sử dụng **tool** tự động giúp tìm kiếm được nhiều lỗi hơn.
        + **Automation testing** nhanh và hiệu quả.
        + Qúa trình kiểm thử được ghi lại, điều đó giúp chạy lại kịch bản kiểm thử nhiều lần và thực hiện trên nhiều nền tảng khác nhau.
        + **Automation testing** được thực hiện bằng các công cụ phầm mềm, do đó nó hoạt động không mệt mỏi không giống như người kiểm thử **tester**.
        + **Automation testing** năng suất và chính xác.
        + Phạm vi kiểm thử rộng vì kiểm tra tự động không quên kiểm tra ngay cả đơn vị nhỏ nhất.
     2. **Nhược điểm**
        + Rất khó có cái nhìn đúng và trực quan về giao diện người dùng như màu sắc, font chữ, vị trí, kích thước các button nếu như không có yếu tố con người.
        + Chi phí cho các **tool** kiểm thử có thể tốn kém, có thể làm tăng chi phí trong khâu kiểm thử của dự án.
        + Nếu có một thay đổi nhỏ cũng sẽ mất thời gian để update kịch bản kiểm thử.
  8. **Sự khác nhau giữa Manual Testing và Automation Testing**

| **TIÊU CHÍ** | **AUTOMATION TESTING** | **MANUAL TESTING** |
| --- | --- | --- |
| Definition | Automation testing sử dụng các tool để thực hiện các trường hợp kiểm thử. | Thực hiện kiểm thử hoàn toàn thủ công không có sự trợ giúp của bất kỳ công cụ tự động nào, được thực hiện bời tester. |
| Processing time | Thời gian kiểm thử rút ngắn hơn so với manual testing | Manual testing tốn nhiều thời gian và nguồn nhân lực |
| Exploratory Testing | Không cho phép kiểm thử khám phá | Có thể kiểm thử khám phá trong manual testing |
| Reliability | Kết quả kiểm thử đáng tin cậy vì nó được thực hiện bằng các tool và các kịch bản | Kết quả kiểm thử không đáng tin cậy vì có khả năng xảy ra lỗi của con người |
| UI change | Chỉ là thay đổi nhỏ trong giao diện AUT nhưng các kịch bản kiểm thử tự động cần phải sửa đổi để hoạt động đúng như mong đợi | Những thay đổi nhỏ thư thay đổi về id, class sẽ không cản trở quá trình kiểm thử |
| Investment | Cần phải đầu tư cho các công cụ kiểm thử | Cần đầu tư về nguồn nhân lực |
| Test Report Visibility | Tất cả các bên liên quan có thể đăng nhập vào hệ thống xem kết quả đã kiểm thử | Kết quả được lưu lại trong excel hoặc word |
| Performance Testing | Được thực hiện trong kiểm thử Load testing, stress testing | Không khả thi trong kiểm thử Load testing, stress testing |
| Parallel Execution | Có thể thực hiện song song trên cấc nền tảng vận hành khác nhau và giảm thời gian thực hiện kiểm thử | Kiểm thử song song trên các nền tảng khác nhau sẽ phải tăng nguồn nhân lực |
| Programming knowledge | Yêu cầu phải có kiến thức lập trình | Không cần có kiến thức lập trình vẫn có thể thực hiện |
| Ideal approach | Automation testing rất hữu ích khi thường xuyên thực hiện chạy lại một kịch bản nhiều nhiều lần | Manual testing hữu ích khi chạy bộ test case một hoặc hai lần |

**Verification và Validation**

**Verification** là quy trình đánh giá một hệ thống hoặc một thành phần để xác định xem liệu các sản phẩm của một giai đoạn phát triển nhất định có đáp ứng được những yêu cầu được định tại thời điểm bắt đầu của giai đoạn đó không.

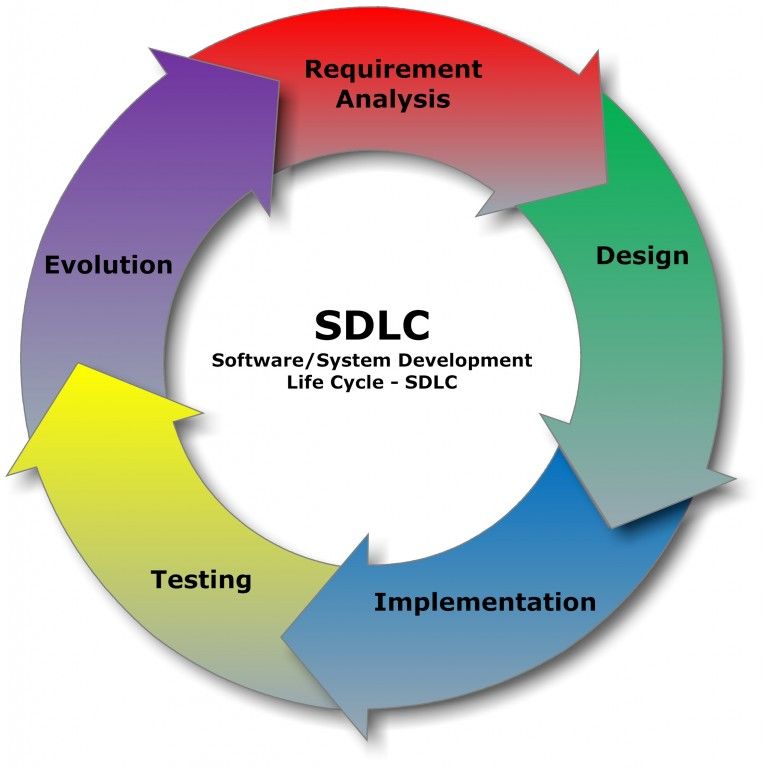
**Validation** là qui trình đánh giá một hệ thống hoặc một thành phần trong suốt quá trình phát triển hoặc lúc kết thúc của quá trình phát triển để xác định xem liệu nó có được làm ra đúng như những yêu cầu cụ thể như những yêu cầu lúc đầu đưa ra không.

**Verification là tĩnh (Static)** trong khi **Validation là động (Dynamic).** Chẳng hạn **Verification** phần mềm là kiểm thử từng dòng mã, từng hàm. Với **Validation**, chạy phần mềm và tìm lỗi. Vị trí lỗi có thể tìm thấy với **Verification**, mà không thể với **Validation**.

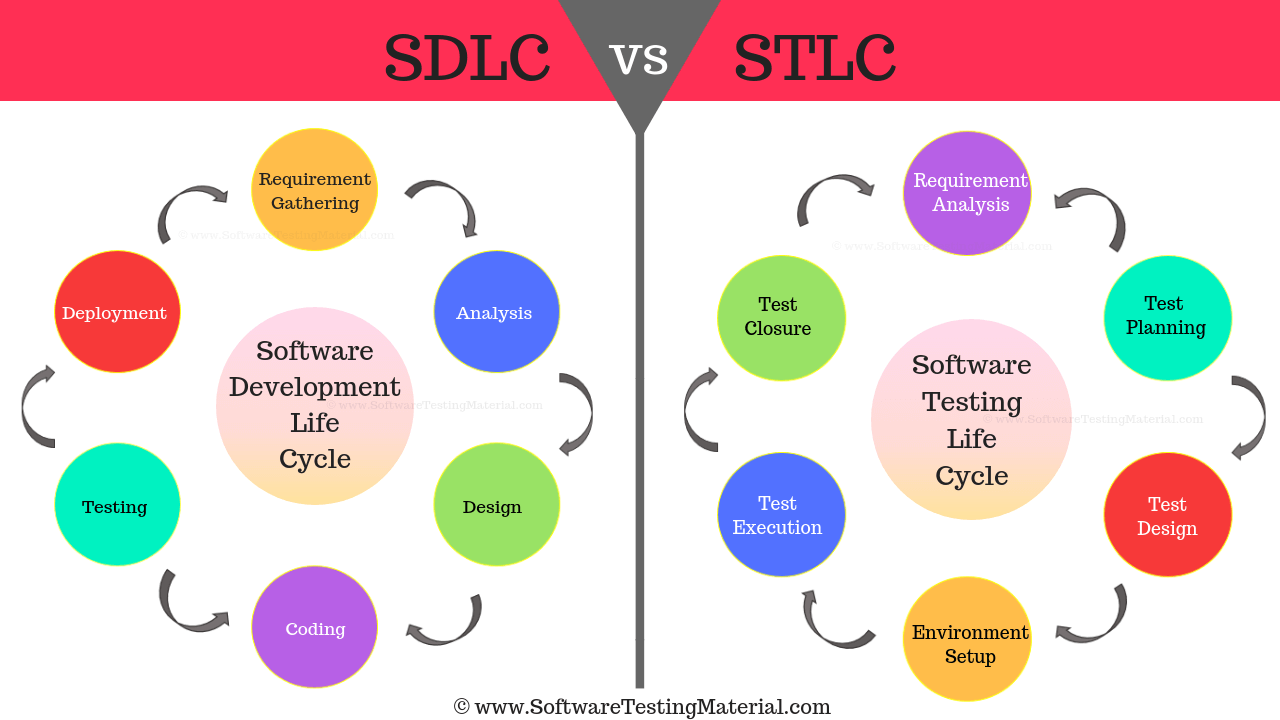
Các nhóm triển khai đọc tài liệu SRS và hoàn thiện Workshop 1 bằng báo cáo dựa trên những hiểu biết của mình về SRS. Nghiên cứu 3 khái niệm **Test Plan – Test Control – Test Closure** phân công công việc nghiên cứu cho các thành viên và dựng ra các File **Test Plan .doc** – **Test Design** .xlsx– **Test Case.xlsx** – **Test Report xlsx. Trả lời các câu hỏi của giảng viên để lấy điểm. WS1.**

**18/03/2020 lấy điểm.**

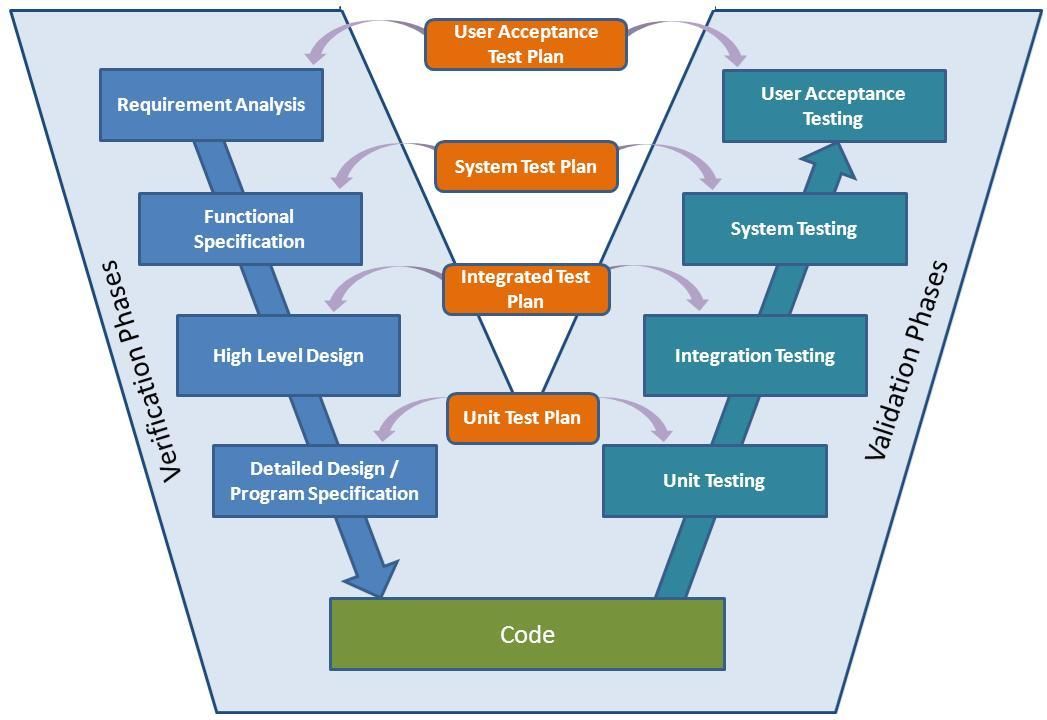
1. **TEST LEVEL:**
2. **SDLC là gì?** **SDLC (Software Development Life Cycle)** hay còn gọi là chu kì phát triển phần mềm. Nó là chuỗi các hoạt động do nhà phát triển thực hiện để thiết kế và phát triển phần mềm. Mặc dù **SDLC** sử dụng thuật ngữ “**Development**”, không chỉ là các công việc mã hóa được thực hiện bởi **deverloper** mà nó còn kết hợp các công việc được đóng góp **bởi kiểm thử viên** và các **bên liên quan**. **Testcases** được tạo ở **SDLC**.



1. **STLC là gì? STLC (Software Testing Life Cycle)** hay còn được là vòng đời kiểm thử phần mềm. Nó bao gồm một loại các hoạt động được thực hiện bởi người kiểm thử để kiểm tra sản phẩm phần mềm của bạn. Mặc dù **STLC** sử dụng thuật ngữ “**testing**” nhưng nó không chỉ liên quan đến kiểm thử viên. Trong một số trường hợp nó còn liên quan đến các nhà phát triển. Trong **STLC** testsase được thực thi



1. **Mô hình thác nước (Waterfall Model) là gì**? Mô hình thác nước là một mô hình tuần tự các giai đoạn khác nhau của hoạt động phát triển phần mềm. Mỗi giai đoạn được thiết kế để thực hiện các hoạt động trong chu kì phát triển phần mềm. Giai đoạn kiểm thử trong mô hình này chỉ được bắt đầu khi đã triển khai hệ thống xong. Kiểm thử được hoàn thành trong SDLC.
2. **V model là gì? Vmodel** là sự mở rộng của mô hình thác nước. Không giống như mô hình thác nước. Ở V model, tương ứng với một giai đoạn kiểm thử là một giai đoạn phát triển phần mềm, thử nghiệm trong mô hình chữ V được thực hiện song song với chu kì phát triển phần mềm. **(HLD = SRS)**



1. **Functional Testing - Non-functional Testing**

Kiểm thử chức năng Functional Testing là gì

*Kiểm thử chức năng là một trong số 4 loại kiểm thử chính thuộc* ***level kiểm thử thứ 2****- Kiểm thử tích hợp* ***(Integration Test)*** *trong kiểm thử phần mềm.*

**Định nghĩa:**

Kiểm thử chức năng là một loại kiểm thử hộp đen **(black box)** và **test case** của nó được dựa trên đặc tả của ứng dụng phần mềm/thành phần đang **test**. Các chức năng được test bằng cách nhập vào các giá trị và kiểm tra kết quả đầu ra, và ít quan tâm đến cấu trúc bên trong của ứng dụng.

Nó là một qui trình cố gắng tìm ra **các khác biệt giữa đặc tả bên ngoài của phần mềm và thực tế** mà phần mềm cung cấp. Với các đặc tả bên ngoài của phần mềm là đặc tả chính xác về hành vi của phần mềm theo góc nhìn của người dùng.

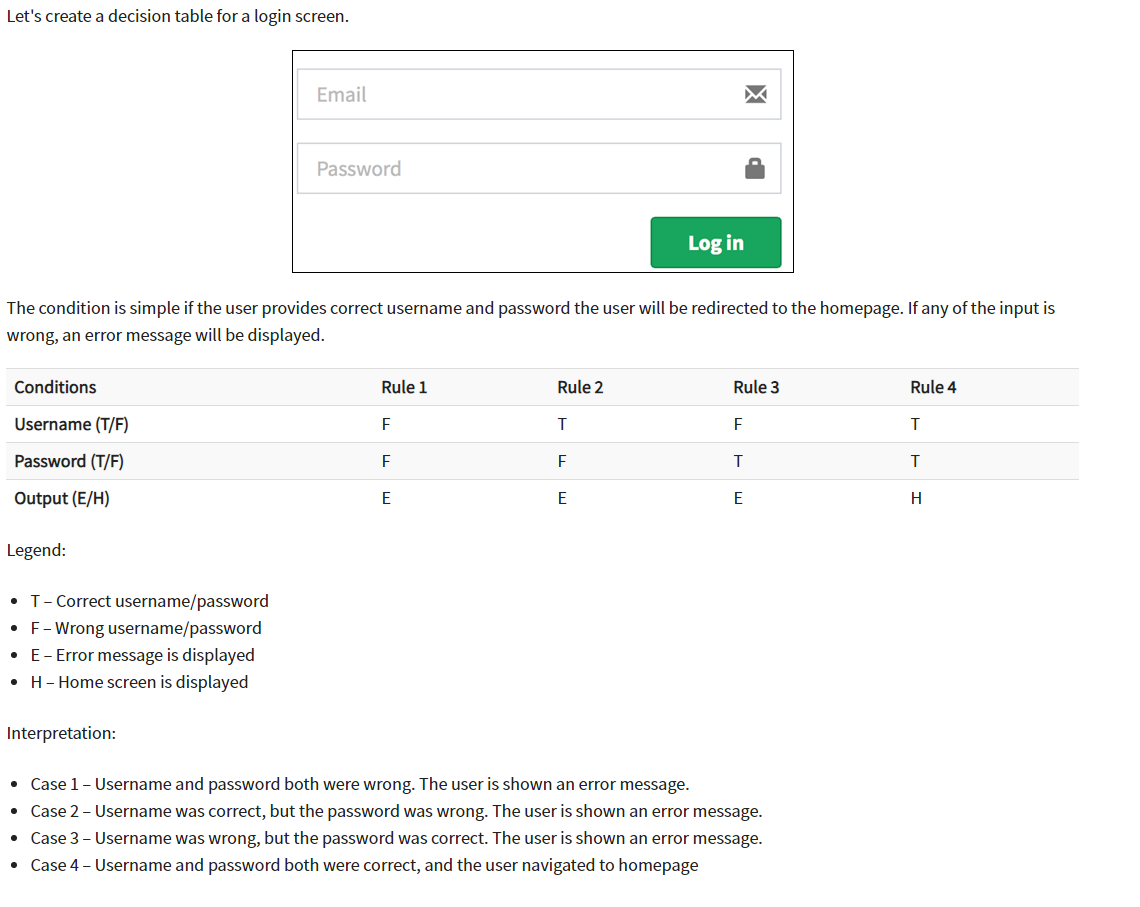
**Mục đích**

* Với kiểm thử đơn vị ta phát hiện sự khác biệt giữa đặc tả giao tiếp của đơn vị và thực tế mà đơn vị này cung cấp.
* Với kiểm thử hệ thống ta chỉ ra rằng chương trình không tương thích với các mục tiêu ban đầu của nó.
* Với kiểm thử chức năng ta sẽ hoàn thiện nốt phần cần xác minh còn lại là chỉ ra rằng chương trình không tương thích với các đặc tả bên ngoài của nó.
* Các lợi ích : tránh kiểm thử dư thừa - ngăn chặn sự quan tâm nhiều vào quá nhiều loại lỗi tại từng thời điểm.

**Các kỹ thuật thường dùng trong kiểm thử chức năng:**

Kiểm thử chức năng thường sử dụng 1 kỹ thuật kiểm thử hộp đen : (Các nhóm tự tìm hiểu thêm cái này)

* Kỹ thuật phân lớp tương đương (Equivalence Class Partitioning).
* Kỹ thuật dùng các bảng quyết định (Decision Tables)



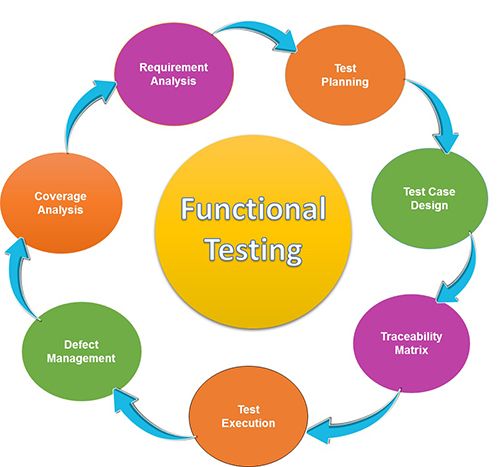
* Kỹ thuật kiểm thử các bộ thần kỳ (Pairwise)
* Kỹ thật phân tích vùng miền (domain analysis)
* Kỹ thuật dựa trên đặc tả Use Case (Use case)

**Các cách tiếp cận để kiểm thử chức năng phần mềm :**

Kiểm thử chức năng khác với kiểm thử hệ thống, kiểm thử chức năng **là “xác nhận – verify – một ứng dụng bằng cách kiểm tra nó dựa vào các tài liệu thiết kế hoặc đặc tả kỹ thuật”,** còn kiểm thử hệ thống là **“xác minh – validate – chương trình bằng cách kiểm tra nó dựa vào các yêu cầu của Người dùng hoặc hệ thống đã công bố”**.

**Kiểm thử chức năng thường bao gồm 5 bước (trích dẫn các bước cần thiết):**

* Việc xác định các chức năng mà phần mềm mong muốn sẽ thực hiện
* Việc tạo ra các dữ liệu đầu vào dựa trên các tài liệu đặc tả kỹ thuật của các chức năng
* Việc xác định kết quả đầu ra dựa trên các tài liệu đặc tả kỹ thuật của các chức năng
* Việc thực hiện các trường hợp kiểm thử
* Việc so sánh kết quả thực tế và kết quả mong muốn.

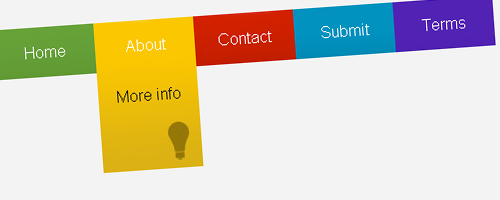


**Các kỹ thuật cụ thể:**

1. User Navigation Testing
2. Transaction Screen Testing
3. Transaction Flow Testing
4. Report Screen Testing
5. Report Flow Testing
6. Database Create/Retrieve/Update/Delete Testing

**1. Kiểm thử khả năng duyệt chức năng của người dùng (User Navigation Test)**

Các màn hình phục vụ duyệt thực hiện chức năng là màn hình log on/log off, menu bar và hệ thống cây phân cấp các option để thực hiện chức năng, toolbar, tất cả các mối liên kết từ màn hình này tới màn hình khác để thể hiện sự liên tục của hoạt động nghiệp vụ đang cần thực hiện.

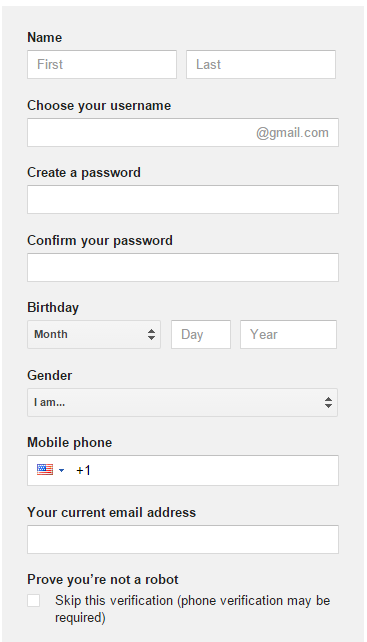


Kiểm thử khả năng duyệt chức năng của người dùng tập trung trên :

* Khả năng người dùng login vào hệ thống với quyền hạn thích hợp.
* Di chuyển qua các màn hình mong muốn 1 cách đúng đắn và logout khỏi phần mềm.

**2. Kiểm thử màn hình thao tác (Transaction screen Test)**

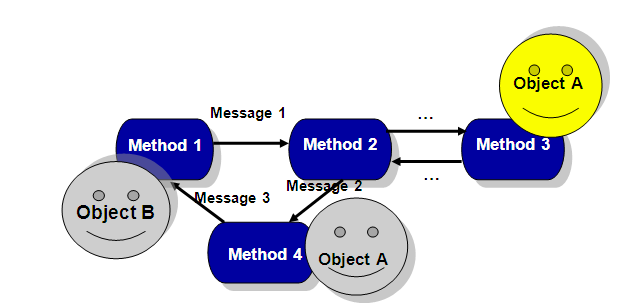
Màn hình thao tác có các **field** nhập liệu, list chọn option, các options, các button chức năng (**Add**, **Change**, **Delete**, **Submit**, **Cancel**, **OK**...). Một vài loại kết quả có thể được hiển thị trên màn hình thao tác sau khi người dùng click button chức năng nào đó.



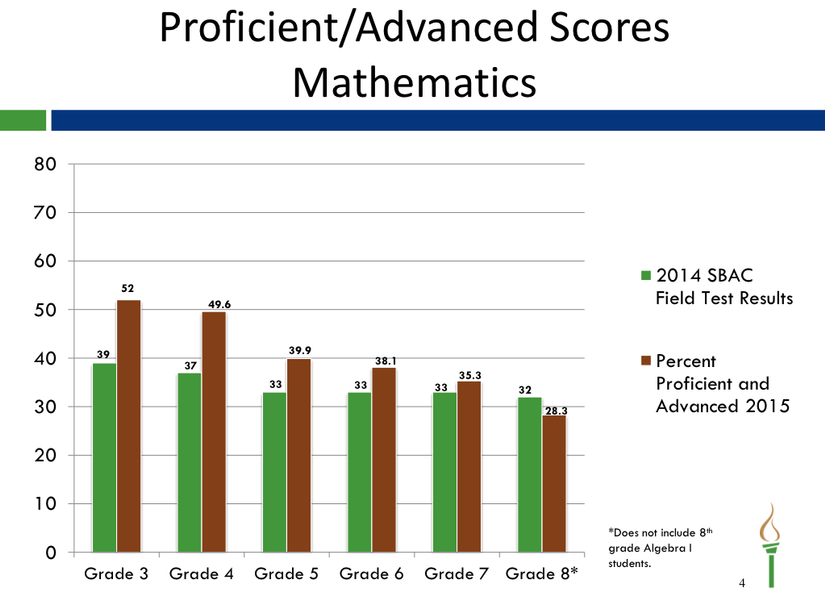
**Công việc của người kiểm thử :**

* Thiết kế **testcase** để xác thực hoạt động của mỗi field dữ liệu, list, option và button trên màn hình giao tác theo các yêu cầu nghiệp vụ, tài liệu người dùng và tài liệu người quản trị.
* **Nếu kết quả được hiển thị trên màn hình thao tác, thì kỹ thuật kiểm thử hộp đen với testcase gồm (data input, ouput kỳ vọng) sẽ được dùng để xác thực kết quả hiển thị.**

**3. Kiểm thử luồng thao tác (Transaction Flow Test)**

Kiểm tra kết quả tổng hợp của nhiều màn hình thao tác theo thứ tự duyệt đúng có hoàn thành hoạt động nghiệp vụ tương ứng không ? Thí dụ nghiệp vụ cập nhật profile khách hàng gồm các màn hình thao tác sau : màn hình 1 cập nhật tên, địa chỉ, contact. Màn hình 2 cập nhật credit. Màn hình 3 cập nhật thông tin thanh toán và khuyến mãi. Màn hình 4 tổng kết profile và thực hiện cập nhật. Màn hình 5 để xem kết quả profile đã cập nhật. Kết quả cuối cùng của trình tự các màn hình là file hay database sẽ ₫ược cập nhật để chứa các thông tin mà người dùng ₫ã cập nhật thông qua các màn hình thao tác.**Nhiệm vụ của người kiểm thử :** Xác thực rằng nếu người dùng thực hiện đúng trình tự các màn hình thao tác và hoàn tất được chúng thì hệ thống sẽ cung cấp kết quả đúng. Ngược lại, nếu người dùng không tuân thủ bất kỳ 1 qui luật nghiệp vụ nào trong 1 màn hình thao tác nào thì hệ thống sẽ không cung cấp kết quả gì cho người dùng.

**4. Kiểm thử màn hình report (Report screen Test)**

Màn hình report cho phép tìm kiếm dữ liệu và hiển thị kết quả (không cần nhập dữ liệu như màn hình giao tác). Khó khăn trong kiểm thử màn hình report nằm ở chỗ có nhiều cách mà người dùng có thể đặc tả dữ liệu cần được tìm kiếm (tiêu chuẩn) và cách thức dữ liệu này ₫ược hiển thị (sắp xếp và định dạng).Công việc của người kiềm thử:

* chú ý đặc biệt vào dữ liệu tìm kiếm và hiển thị vì người dùng có thể chọn sai dữ liệu hay tệ hơn là không có kết quả nào được hiển thị.

**5. Kiểm thử luồng report (Report Flow Test)**

Kiểm thử các khác biệt giữa kết quả hiển thị trong màn hình report và các phương thức report khác (như máy in, file,..). Nhiệm vụ của người kiểm thử : Xác định xem phần mềm gởi cùng kết quả ra màn hình report và máy in ? Xác thực kết quả report trên tất cả phương thức báo cáo khác nhau được hỗ trợ bởi phần mềm. Xác định xem khả năng máy in có hỗ trợ font, vùng chọn được người dùng xác định trong màn hình report ?

**6. Kiểm thử việc Create/Retrieve/Update/Delete database**

Thường được thực hiện thông qua 2 bước : Kiểm thử việc thiết kế, khởi tạo database ban đầu thông qua tiện ích bên ngoài phần mềm ứng dụng cần kiểm thử. Kiểm thử việc phần mềm sử dụng database ₫ã ₫ược thiết kế và khởi tạo ₫úng. Đòi hỏi sự hợp tác và cộng tác giữa người kiểm thử và người quản trị database.

Kiểm thử phi chức năng (Non-functional testing) là gì?

Kiểm thử phi chức năng là một loại kiểm thử để kiểm tra các khía cạnh phi chức năng (hiệu suất, khả năng sử dụng, độ tin cậy, v.v.) của một ứng dụng phần mềm. Nó được thiết kế để kiểm tra sự sẵn sàng của một hệ thống theo các tham số không chức năng không bao giờ được giải quyết bằng thử nghiệm chức năng.

Một ví dụ về kiểm tra phi chức năng như là kiểm tra xem có bao nhiêu người có thể đăng nhập đồng thời vào một phần mềm.

Kiểm tra phi chức năng cũng quan trọng không kém như kiểm tra chức năng và ảnh hưởng đến sự hài lòng của khách hàng.

**Trong hướng dẫn này, chúng ta sẽ tìm hiểu những vấn đề sau:**

* Mục tiêu của kiểm thử phi chức năng
* Đặc điểm của kiểm thử phi chức năng
* Các thông số kiểm thử phi chức năng
* Loại kiểm thử
* Các loại kiểm thử phi chức năng.

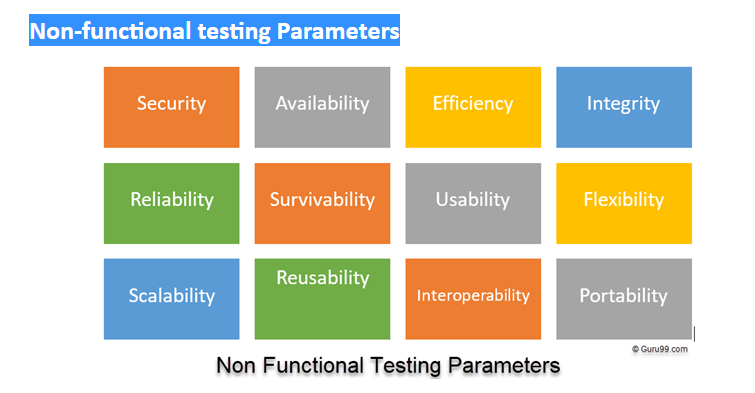
**Mục tiêu của kiểm thử phi chức năng**

* Kiểm thử phi chức năng nên tăng khả năng sử dụng, hiệu quả, khả năng bảo trì và tính di động của sản phẩm.
* Giúp giảm rủi ro sản xuất và chi phí liên quan đến các khía cạnh phi chức năng của sản phẩm.
* Tối ưu hóa cách cài đặt sản phẩm, thiết lập, thực thi, quản lý và giám sát.
* Thu thập và sản xuất các phép đo, và số liệu cho nghiên cứu và phát triển nội bộ.
* Cải thiện và nâng cao kiến thức về hành vi và công nghệ sản phẩm đang sử dụng.

**Đặc điểm của kiểm thử phi chức năng**

* Kiểm tra phi chức năng nên được đo lường, vì vậy không có chỗ cho đặc tính chủ quan như tốt, tốt hơn, tốt nhất, v.v.
* Những con số chính xác không thể biết được khi bắt đầu xử lý yêu cầu
* Điều quan trọng cần ưu tiên các yêu cầu
* Đảm bảo rằng các thuộc tính chất lượng được xác định chính xác

**Các thông số kiểm thử phi chức năng**



**1. Bảo vệ (Security):**

Tham số xác định cách hệ thống được bảo vệ an toàn trước các cuộc tấn công có chủ ý và đột ngột từ các nguồn bên trong và bên ngoài. Điều này được kiểm thử thông qua Kiểm tra bảo mật.

**2. Độ tin cậy (Reliability):**

Mức độ mà bất kỳ hệ thống phần mềm nào liên tục thực hiện các chức năng được chỉ định mà không gặp sự cố. Điều này được kiểm thử bởi Kiểm tra độ tin cậy

**3. Khả năng sống sót (Survivability):**

Tham số kiểm tra rằng hệ thống phần mềm tiếp tục hoạt động và tự phục hồi trong trường hợp lỗi hệ thống. Điều này được kiểm tra bằng Recovery Recovery

**4. Khả dụng (Availability):**

Tham số xác định mức độ mà người dùng có thể phụ thuộc vào hệ thống trong quá trình hoạt động. Điều này được kiểm tra bằng Kiểm tra ổn định.

**5. Khả năng sử dụng (Usability):**

Người dùng có thể dễ dàng học hỏi, vận hành, chuẩn bị đầu vào và đầu ra thông qua tương tác với một hệ thống. Điều này được kiểm tra bởi Kiểm tra khả năng sử dụng

**6. Khả năng mở rộng (Scalability):**

Thuật ngữ này đề cập đến mức độ mà bất kỳ ứng dụng phần mềm nào cũng có thể mở rộng khả năng xử lý của nó để đáp ứng nhu cầu gia tăng. Điều này được kiểm tra bằng khả năng mở rộng

**7. Khả năng tương tác (Interoperability):**

Tham số phi chức năng này kiểm tra giao diện hệ thống phần mềm với các hệ thống phần mềm khác. Điều này được kiểm tra bởi Kiểm tra khả năng tương tác

**8. Hiệu quả (Efficiency):**

Mức độ mà bất kỳ hệ thống phần mềm nào cũng có thể xử lý dung lượng, số lượng và thời gian đáp ứng.

**9. Mềm dẻo (Flexibility):**

Thuật ngữ này đề cập đến sự dễ dàng mà ứng dụng có thể hoạt động trong các cấu hình phần cứng và phần mềm khác nhau. Giống như RAM tối thiểu, yêu cầu CPU.

**10. Tính di động (Portability):**

Tính linh hoạt của phần mềm để chuyển từ môi trường phần cứng hoặc phần mềm hiện tại của nó.

**11. Tái sử dụng (Reusability):**

Nó đề cập đến một phần của hệ thống phần mềm có thể được chuyển đổi để sử dụng trong một ứng dụng khác.

**Non-functional Testing Types**

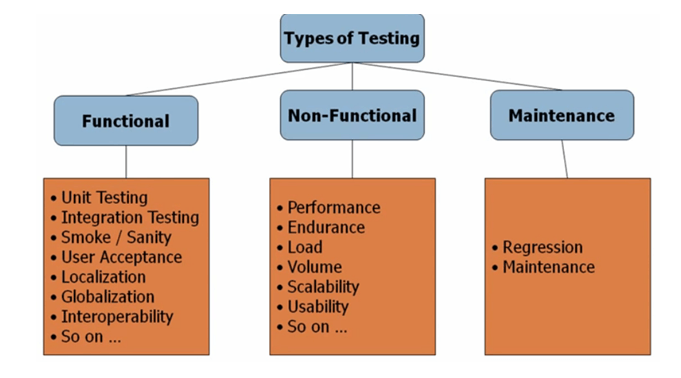
* **Performance Testing (Kiểm tra năng suất)**
* **Load Testing (Kiểm tra tải)**
* **Failover Testing (Kiểm tra chuyển đổi dự phòng)**
* **Security Testing (Kiểm tra bảo mật)**
* **Compatibility Testing (Kiểm tra tương thích)**
* **Usability Testing (Kiểm tra khả năng sử dụng)**
* **Stress Testing (Kiểm tra về áp lực)**
* **Maintainability Testing (Kiểm tra bảo trì)**
* **Scalability Testing (Kiểm tra khả năng mở rộng)**
* **Volume Testing (Kiểm tra khối lượng)**
* **Disaster Recovery Testing (Kiểm tra khắc phục thảm họa)**
* **Compliance Testing (Kiểm tra tuân thủ)**
* **Portability Testing (Kiểm tra tính di động)**
* **Efficiency Testing (Kiểm tra hiệu quả)**
* **Reliability Testing (Kiểm tra độ tin cậy)**
* **Baseline Testing (Kiểm tra cơ bản)**
* **Endurance Testing (Kiểm tra độ bền)**
* **Documentation Testing (Kiểm tra tài liệu)**
* **Recovery Testing (Kiểm tra phục hồi)**
* **Internationalization Testing (Kiểm tra quốc tế hóa)**
* **Localization Testing (Kiểm tra nội địa hóa)**

Loại thử nghiệm có 3 loại thử nghiệm

Chức năng (**Functional**)

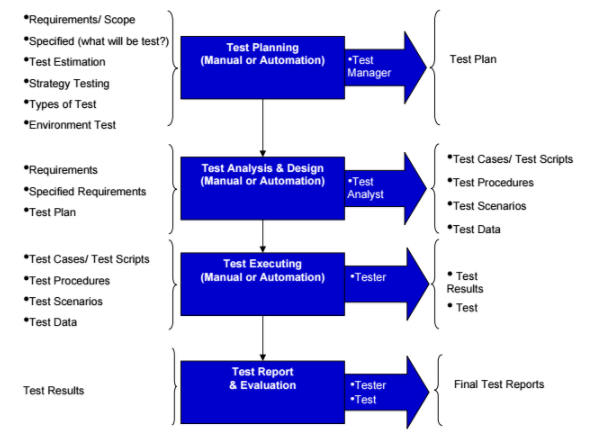
Không chức năng (**Non - Functional**)

Bảo trì (**Maintenance**)

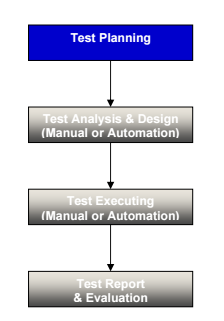


1. **Kế hoạch kiểm thử**

**Quy trình kiểm thử tổng quát**

****

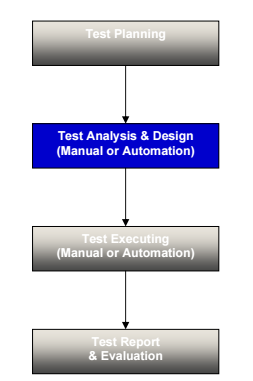
**Xây dựng kế hoạch kiểm thử**

****

**Test Manager hoặc Test Leader sẽ xây dựng kế hoạch ban đầu về kiểm thử.**

* Định nghĩa phạm vi kiểm thử
* Định nghĩa các chiến lược kiểm thử
* Nhận dạng các rủi ro và yếu tố bất ngờ
* Nhận dạng các hoạt động kiểm thử và xây dựng lịch kiểm thử
* Nhận dạng môi trường kiểm thử
* .... Kế hoạch kiểm thử cần được:
* Xem lại bởi QA team, Developer, Business Analysis.PM và Customer
* Chấp thuận bởi: Project Manager và Customer
* Hiểu chỉnh trong suốt chu kỳ kiểm thử để phản ánh các thay đổi nếu cần thiết

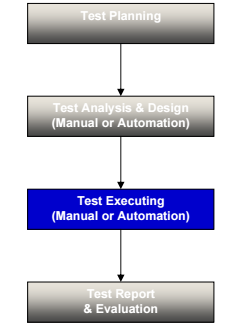
**Phân tích và thiết kế kiểm thử**

****

**Test Analysis** hoặc **test Design** sẽ thiết kế (định nghĩa) các testcase từ các yêu cầu liên quan (ví dụ từ thông tin trong usecase)

* Thiết kế (định nghĩa) các testcase từ các yêu cầu chức năng và yêu cầu không chức năng của phần mềm.
* Các testcase cần bao phủ tất cả yêu cầu trong chiến lược kiểm thử.
* Nếu cần kiểm thử tự động, Test Design sẽ xây dựng các kịch bản dựa trên các testcase/test procedures. Các testcase cần được:
* Xem xét lại bởi Project Leader, Developer có liên quan, các Tester khác, Test Leader, Business Analysis và Customer.
* Chấp thuận bởi Test Leader hoặc Customer
* Hiệu chỉnh/cập nhật nếu Tester đã tìm được những lỗi mà không nằm trong testcase hiện có.

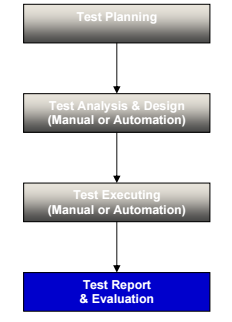
**Thi hành kiểm thử**

****

**Tester sẽ được bố trí công việc bởi Test Leader để thi hành kiểm thử.**

* Thi hành kiểm thử theo tưng testcase
* Thực hiện kiểm thử đặc biệt
* Thực hiện kịch bản kiểm thử mà không được định nghĩa trong testcase
* Kiểm thử lại các lỗi đã được sửa
* Tester sẽ tạo các báo cáo về lỗi trong suốt quá trình kiểm thử lỗi và theo dõi chúng cho đến khi chúng đã được xử lý.
* Ở công đoạn kiểm thử độ chấp thuận, Customer sẽ thi hành kiểm thử để kiểm định xem hệ thống phần mềm có thỏa mãn các nhu cầu người dùng không?

**Test Report và Evaluation**

****

**Test Manager hoặc Test Leaser sẽ phân tích các lỗi trong hệ thống theo dõi các lỗi**

* Tạo các báo cáo lỗi
* Đánh giá các kết quả kiểm thử, thống kê các yêu cầu thay đổi
* Tính và phân phối các thông tin đo lường hoạt động kiểm thử
* Tạo bảng tổng kết đánh giá hoạt động kiểm lỗi
* Xác định xem đã đạt tiêu chí thành công và hoàn thành kiểm thử chưa.

Kế hoạch kiểm thử

**3.1 Định nghĩa**

**Kế hoạch kiểm thử thường được để trong 1 file và chứa các kết quả của các hoạt động sau:**

* Nhận dạng các chiến lược được dùng để kiểm tra và đảm bảo rằng sản phẩm thỏa mãn đặc tả thiết kế phần mềm và các yêu cầu khác về phần mềm.
* Định nghĩa các mục tiêu và phạm vi của nỗ lực kiểm thử
* Nhận dạng phương pháp luận mà đội kiểm thử sẽ dùng để thực hiện công việc kiểm thử.
* Nhận dạng phần cứng, phần mềm và các tiện ích cần cho kiểm thử
* Nhận dạng các tính chất và các chức năng sẽ được kiểm thử
* Xác định các hệ số rủi ro gây nguy hại cho việc kiểm thử
* Lập lịch kiểm thử và phân phối công việc cho mỗi thành viên tham gia.
* ... Test Manager hoặc Test Leader sẽ xây dựng kế hoạch kiểm thử.

**3.2 Nhu cầu cần phải có kế hoạch kiểm thử**

**Kế hoạch kiểm thử cần phải được xây dựng sớm như có thể có trong mỗi chu kì phát triển phần mềm để:**

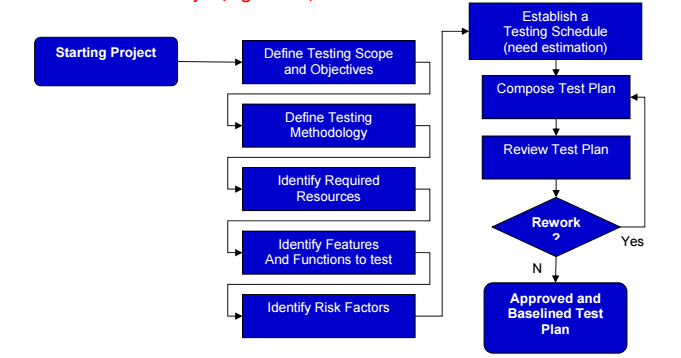
* Tập hợp và tổ chức các thông tin kiểm thử cần thiết
* Cung cấp thông tin về quy trình kiểm thử sẽ xảy ra trong tổ chức kiểm thử
* Cho mỗi thành viên trong đội kiểm thử có hướng đi đúng
* Gán các trách nhiệm rõ ràng cụ thể cho mỗi thành viên đội kiểm thử
* Có lịch biểu làm việc rõ ràng và các thành viên có thể làm việc với nhau tốt

**3.3 Kế hoạch kiểm thử**

**Cần chứa các thông tin sau:**

* Phạm vi/mục tiêu kiểm thử
* Các chiến lược được dùng
* Các tài nguyên phần cứng và phần mềm phục vụ kiểm thử
* Các nhu cầu về nhân viên và huấn luyện nhân viên
* Các tính chất cần được kiểm thử
* Các rủi ro và sự cố bất ngờ
* Lịch kiểm thử cụ thể
* Các kênh thông tin liên lạc
* Cấu hình cho từng phần tử như kế hoạch kiểm thử, testcase, thủ tục kiểm thử...
* Môi trường kiểm thử
* Tiêu chí đầu vào và tiêu chí dừng kiểm thử
* Các kết quả phân phối

**3.4 Quy trình xây dựng kế hoạch kiểm thử**

****

**Ghi chú quan trọng: Sau khi xây dựng xong kế hoạch kiểm thử, ta có thể thay đổi nó nhưng phải tuân thủ quy trình yêu cầu thay đổi.**

**3.5 Các hoạt động chính trong việc xây dựng kế hoạch kiểm thử**

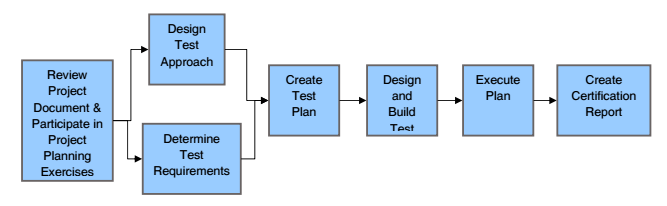
* Định nghĩa mục đích, phạm vi, chiến lược, cách tiếp cận, các điều kiện chuyển, các rủi ro, kế hoạch giảm nhẹ và tiêu chí chấp thuận
* Định nghĩa cách thức thiết lập môi trường và các tài nguyên được dùng cho việc kiểm thử
* Thiết lập cơ chế theo dõi lỗi phát hiện
* Chuẩn bị ma trận theo dõi bao phủ mọi yêu cầu phần mềm
* Báo cáo trạng thái kiểm thử
* Phát hành leo thang

**4. Các thành phần chính trong kế hoạch kiểm thử**

**4.1 Mục đích và phạm vi kiểm thử**

* Đặc tả mục đích của tài liệu về kế hoạch kiểm thử
* Cung cấp vắn tắt về phạm vi mà project được hỗ trợ như platform, loại database, hay danh sách vắn tắt về các loại project con trong project kiểm thử

**4.2 Cách tiếp cận và các chiến lược được dùng**

* Đắc tả về phương pháp luận kiểm thử sẽ được dùng để thực hiện kiểm thử
* **Ví dụ: Tổng quan về Testing Process Approach cho Project ABC **

**Đề cập các cấp độ kiểm thử cần thực hiện Các kỹ thuật được dùng cho mỗi kiểu kiểm thử trong project:**

* Kiểm thử tích hợp (Integration Testing)
* Kiểm thử hệ thống (System Testing)
* Kiểm thử độ chấp thuận (Acceptant Testing)
* Kiểm thử chức năng của người dùng (Functionality Testing)
* Kiểm thử hồi quy (Regresstion Testing)
* Kiểm thử việc phục hồi sau lỗi (Failover and Recovery Testing)
* Kiểm thử việc kiểm soát an ninh và truy xuất (Security and Access Control Testing)
* Kiểm thử việc cấu hình và cài đặt (Configuration and Installation Testing)
* Kiểm thử đặc biệt
* Kiểm thử hiệu xuất (Performance Testing)

**4.3 Các tính chất cần được kiểm thử**

Danh sách các tính chất của phần mềm cần được kiểm thử, đây là catalog chưa tất cả các testcase (bao gồm chỉ số testcase, tiêu đề testcase) cũng như tất cả các trạng thái cơ bản.

**4.4 Các tính chất không cần được kiểm thử**

Danh sách các vùng phần mềm được loại trừ khỏi kiểm thử, cũng như các testcase đã được định nghĩa nhưng không cần kiểm thử.

**4.5 Rủi ro và các sự cố bất ngờ**

* Danh sách tất cả rủi ro có thể xảy ra trong chu kì kiểm thử
* Phương pháp mà ta cần thực hiện để tối thiểu hóa hay sống chung với rủi ro.

**4.6 Tiêu chí đình chỉ và phục hồi kiểm thử**

* Tiêu chí đình chỉ kiểm thử là các điều kiện mà nếu thỏa mãn thì kiểm thử sẽ dừng lại
* Tiêu chí phục hồi là những điều kiện được đòi hỏi để tiếp tục việc kiểm thử đã được ngừng trước đó.

**4.7 Môi trường kiểm thử**

Đặc tả đầy đủ về các môi trường kiểm thử, bao gồm đặc tả phần cứng, phần mềm, mạng, database, hệ điều hành và các thuộc tính môi trường khác ảnh hưởng đến kiểm thử.

**4.8 Lịch kiểm thử**

Lịch kiểm thử ở dạng ước lượng, nên chứa các thông tin: các cột mốc với ngày xác định + Kết quả phân phối của từng cột mốc

**4.9 Tiêu chí dừng kiểm thử và chấp nhận**

**Bất kỳ chuẩn chât lượng mong muốn nào mà phần mềm phải thỏa mãn cho việc phân phối đến khách hàng. Có thể bao gồm các thứ sau:**

* Các yêu cầu mà phần mềm phải được kiểm thử dưới các môi trường xác định
* Số lỗi tối thiểu ở các cấp an ninh và ưu tiên khác nhau, số phủ kiểm thử tối thiểu
* Ký kết và đồng thuận của các bên liên quan

**4.10 Nhân sự**

**Vai trò và trách nhiệm từng người :**

* Danh sách các vai trò xác định của các thành viên đội kiểm thử trong hoạt ₫ộng kiểm thử.
* Các trách nhiệm của từng vai trò.
* Công tác huấn luyện.
* Danh sách các huấn luyện cần thiết cho các QC

**4.11 Các tiện ích phục vụ kiểm thử**

* Danh sách tất cả các tiện ích cần dùng trong suốt chu kỳ kiểm thử.
* Với project kiểm thử tự ₫ộng, các tiện ích cần ₫ược liệt kê với chỉ số version cùng thông tin license.

**4.12 Các kết quả phân phối**

**Danh sách tất cả tài liệu hay artifacts dự ₫ịnh phân phối nội bộ sau khi mỗi cột mốc kết thúc hay sau khi project kết thúc.**

**5. Một số điểm chính cần nhớ**

* Mục đích và phạm vi kiểm thử
* Cách tiếp cận và các chiến lược kiểm thử được dùng.
* Các tính chất cần được kiểm thử/ không cần kiểm thử
* Lịch kiểm thử
* Nhân sự
* Môi trường kiểm thử
* Tạm dừng/Tiếp tục kiểm thử
* Kiểm thử ₫ộ chấp thuận
* Các tiện ích kiểm thử cần dùng
* Rủi ro và yếu tố bất ngờ
* Các kết quả phân phối nội bộ

1. **1**
2. **1**
3. **1**
4. **1**
5. **1**
6. **1**