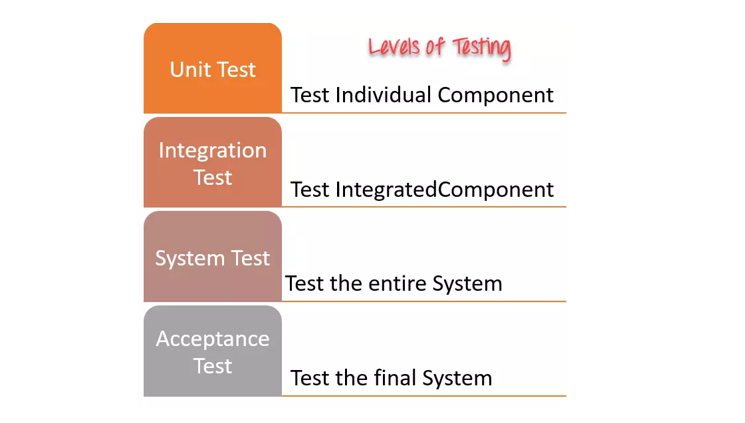
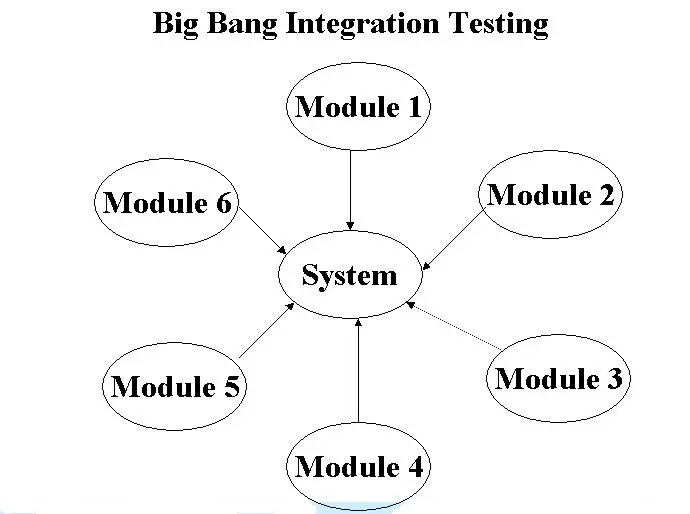
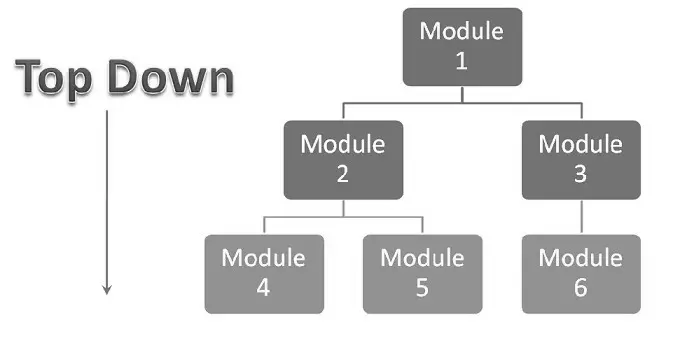
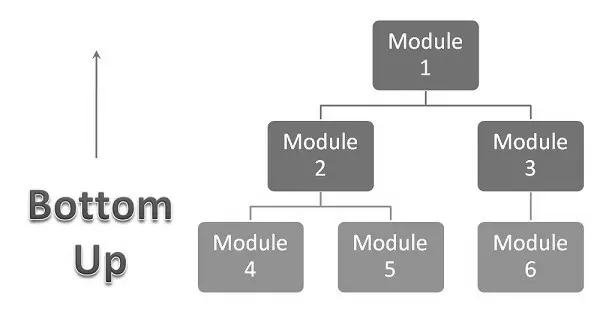
* **Kiểm thử tích hợp – Integration Testing**

****

* 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.
* **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.
* **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.
* **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.

* **Nêu vòng đời của kiểm thử thành phần.**

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 |

* 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.