(Slide 3) Testing là giai đoạn cho chúng ta thấy rằng chương trình của chúng ta sẽ định làm gì và phát hiện ra các lỗi chương trình trước khi đi vào sử dụng. Thì việc testing này được thực hiện với dữ liệu nhân tạo có nghĩa là chúng ta mang sản phẩm chạy qua những test case để test được tất cả trường hợp của sản phẩm.

Khi mà test sản phẩm thì có 2 thứ chúng ta cần thực hiện:

1.Việc đầu tiên và cơ bản nhất là nó đã đáp ứng đủ tất cả nhu cầu mà khách hàng đưa ra, đối với custom software, chúng ta phải test ít nhất là 1 lần để đối chiếu với tài liệu trong user requirement, đối với generic software, thì chúng ta phải tự thử nghiệm test hết tất cả trường hợp cũng như tương tác giữa chúng lấy 1 ví dụ đó là khi chúng ta làm một sản phẩm máys tính cầm tay nếu chỉ đơn thuần là dừng lại ở việc thực hiện các phép tính theo đúng form mẫu thì không có gì đáng nói, đáng nói ở đây là các thao tác sai phạm của khách hàng dẫn đến những lỗi phát sinh không mong muốn lấy

2. Tìm các loại input về cơ bản nó sẽ phát sinh ra lỗi khi user nhập vào. Khác với ở phần trên là các tương tác của các tính năng nút ấn đã có sẵn, việc người dùng nhập vào 1 chuỗi như thế nào là khó dự đoán được, đối với những user thông minh thì nó càng khó đoán hơn.

(Slide 5) Mức độ tin cậy phụ thuộc vào mục đích của hệ thống, kỳ vọng của người dùng và môi trường marketing hiện tại cho hệ thống:

1. Phần mềm càng quan trọng, thì nó càng phải đáng tin cậy. Ví dụ như 1 phần mềm kiểm soát an toàn thì mức độ quan trọng và tin cậy nó phải cao hơn những phần mềm chỉ dừng lại ở mức độ an toàn như sáng tạo ra những ý tưởng sản phẩm.

2. Do người dùng đã từng sử dụng những sản phẩm đầy lỗi và không có kì vọng cao về chất lượng phần mềm vậy nên họ sẽ không ngạc nhiên khi sản phẩm có lỗi. Cho dù vậy những người dùng vẫn kì vọng nó sẽ trở nên đáng tin cây hơn do đó phải kiểm tra kĩ lưỡng trước khi đưa vào sử dụng.

3. Môi trường Marketing, một công ty phần mềm đưa một hệ thống ra thị trường, điều quan trọng hơn hết là phải biết được đối thủ cạnh tranh của nó, từ giá cả, chất lượng dịch vụ. Trong môi trường cạnh tranh, công ty có thể quyết định đưa sản phẩm ra thị trường trước khi qua công đoan testing và debuging bởi vì họ muốn họ là con chim đầu đàn của dòng sản phẩm này. Đặc biệt, nếu người dùng sử dụng sản phẩm rẻ, họ có thể sẽ chịu mức độ tin cậy thấp hơn.

(Slide 6) Hình 8.2 cho thấy việc kiểm tra và kiểm tra phần mềm hỗ trợ V & V ở các giai đoạn khác nhau trong quy trình phần mềm. V& V nghĩa là Validation and Verification. Các mũi tên chỉ ra các giai đoạn trong quy trình mà các kỹ thuật có thể được sử dụng.

(Slide 7) Kiểm tra phần mềm có ba ưu điểm:

1. Trong quá trình thử nghiệm, các lỗi có thể bị ẩn, không ai có thể đoán trước được hết những lỗi sẽ xảy ra trong tương lai. Vì việc test không liên quan đến việc thực thi hệ thống, vậy nên 1 phiên test sẽ có thể phát hiện nhiều lỗi.

2. Các bản chưa hoàn chỉnh có thể được kiểm tra mà không cần trả thêm phí.

3. Ngoài việc tìm kiếm các khiếm khuyết của chương trình thì việc kiểm tra cũng có thể kiểm luôn những thứ khác như tiêu chuẩn của 1 chương trình, tính linh hoạt, và khả năng bảo trì. Hơn thế nữa nó có thể tìm thấy được sự thiếu hiệu quả ở thuật toán và phong cách lập trình kém.

(Slide 8) Thông thường, một hệ thống phần mềm thương mại phải trải qua ba giai đoạn thử nghiệm:

1. Development testing - kiểm tra phát triển, hệ thống sẽ được kiểm tra để phát hiện lỗi, ở giai đoạn này các nhà thiết kế hệ thống và lập trình viên đều được tham gia

2. Release testing - kiểm tra phát hành, lúc này một nhóm tester riêng biệt sẽ test hệ thống trước khi phát hành để kiểm tra sản phẩm đáp ứng nhu cầu của bên khách hàng hay không

3. User testing - kiểm tra người dùng, kiểm tra môi trương của người sử dụng. Ở giai đoạn này tester là một nhóm marketing trong nội bộ sẽ test để xem phần mềm có được phát hành hay không. Acceptance testing là giai đoạn kiểm tra từ phía khách hàng trong đó họ chính thức kiểm tra các chức năng và quyết định có chấp nhận sản phẩm hay cần phát triển thêm.

(Slide 9) Có ba giai đoạn development testing:

1. Unit testing - kiểm tra đơn vị, kiểm tra các lớp, unit testing tập trung vào các phương thức hoặc đối tượng.

2. Component testing - kiểm tra thành phần, các thành phần riêng rẽ sẽ tích hợp lại tạo ra các thành phần tổng hợp. Giai đoạn này tập trung vào kiểm tra giao diện thành phần cung cấp quyền truy cập và các chức năng thành phần.

3. System testing, lúc này các thành phần trong giai đoan component testing sẽ tích hợp vào hệ thống để kiểm tra. Tập trung vào kiểm tra tương tác của các thành phần này.

(Slide 10) Unit testing là quá trình kiểm tra các thành phần chương trình, như các phương thức hoặc các lớp đối tượng. Hình 8.2 là ví dụ của unit testing ở trạm thời tiết

(Slide 11) Một bài kiểm tra tự động có ba phần:

1. Phần thiết lập - setup part, trong đó chúng ta khởi tạo hệ thống với các testcase

2. Phần gọi - call part, gọi đối tượng hoặc phương thức để kiểm tra

3. Phần xác nhận - assertion part, so sánh các kết quả của việc gọi lại đối tượng, phương thức với kết quả mong đợi.

(Slide 12) Chọn test case:

1. Test case phải mô tả được đúng với những gì khách hàng mong đợi, nghĩa là thành phần mà chúng ta đang kiểm tra phải đúng với chức năng mà nó phải làm.

2. Nếu có lỗi, nó sẽ hiển thị bởi testcase

Do đó nên thiết kế 2 trường hợp thử nghiệm. Thứ nhất la trường hợp các thành phần hoạt động. Thứ 2 là dựa trên kinh nghiệm để test những vấn đề phát sinh phổ biến

(Slide 13) Có 2 chiến lược hiêu quả giúp chúng ta chọn testcase:

1. Kiểm tra phân vùng, các nhóm đầu vào sẽ có đặc điểm chung và xử lí cùng 1 cách. Chúng ta nên chọn các test case từ trong nhóm này.

2. Kiểm tra dựa trên hướng dẫn, chúng ta sẽ sử dụng hướng dẫn để chọn testcase. Hướng dẫn này là các kinh nghiệm về các loại lỗi mà lập trình viên mắc phải khi phát triển.

(Slide 14) Ví dụ: khi bạn đang kiểm tra các chương trình với phương trình, mảng hoặc danh sách, các nguyên tắc có thể giúp tìm ra lỗi bao gồm:

1. Kiểm tra phần mềm với các chuỗi chỉ có 1 giá trị duy nhất, việc này để test lỗi khi các lập trình viên đưa ra giả định chuỗi được tạo thành từ 1 số giá trị

2. Sử dụng trình tự khác nhau

3. Kiểm tra đạo hàm sao cho các phần tử đầu tiên, giữa và cuối của chuỗi được truy cập. Cách tiếp cận này cho thấy các vấn đề tại ranh giới phân vùng.

(Slide 15) các loại lỗi giao diện khác nhau có thể xảy ra các lỗi khác nhau:

1. Parameter Inteface - giao diện tham số. Đây là các giao diện trong đó dữ liệu hoặc tham chiếu chức năng được chuyển từ thành phần này sang thành phần khác.

2. Shared memory interfaces - giao diện bộ nhớ dùng chung. Đây là các giao diện trong đó khối bộ nhớ được chia sẻ giữa các thành phần. Giao diện này được dùng trong các hê thống nhúng, trong đó các cảm biến tạo dữ liệu được lấy và xử lí bởi các thành phần hệ thống khác

3. Procedural interfaces - giao diện thủ tục. Đây là các giao diện trong đó có 1 thành phần đóng gó 1 tập các thủ tục có thể được gọi bởi thành phần khác. Các đối tượng và các thành phần có thể tải sử dụng có hình thức giao diện này.

4. Message passing interfaces - giao diện truyền tin nhăn. Đây là các giao diện trong đó 1 thành phần yêu cầu sử dụng từ thành phần khác bằng cách gửi tin nhắn đến nó. Sẽ có thông báo trả lại cùng với kết quả.

(Slide 16) Hình 8.7 này thể hiện sự tương tác giữa các thành phần A B và C tạo nên 1 thành phần lớn hơn. Testcase lúc này không còn áp dụng cho các thành phần riêng rẻ A B hay C nữa mà là thành phần tổng hợp của A B và C vì có thể lỗi không phát sinh ở từng thành phần nhỏ mà phát sinh ở tương tác giữa các thành phần này

(Slide 17) Một số hướng dẫn chung để kiểm tra giao diện:

1. Kiểm tra code và xác định mỗi lần gọi đến thành phần bên ngoài. Thiết kế 1 tập test case trong đó giá trị của các tham số cho thành phần bên ngoài nằm ở cùng cực phạm vi của nó. Những giá trị ở ngưỡng thế này có thể show ra được sự không nhất quán giữa các giao diện

2. Con trỏ, con trỏ được truyền qua 1 giao diên luôn kiểm tra giao diện với các tham số con trỏ null.

3. Khi 1 thành phần được gọi thông giao Procedural interfaces - giao diện thủ tục, thiết kế các testcase để cố tình làm nó lỗi.

4. Sử dụng stress testing trong các hệ thống truyền tin nhắn. Có nghĩa là chúng ta nên thiết kế các testcase đẻ in ra nhiều message hơn để kiểm tra về vấn đề thời gian

5. Khi 1 số thành phần tương tác thông qua Shared Memory, các testcase thiết kế thay đổi thứ tự các thành phần. Cách này tiết lộ được giả định ngầm của lập trình viên về thứ tự dữ liệu.

(Slide 18) Trong quá trình thử nghiêm hệ thống, các thành phần được reuse sẽ được phát triển riêng và tích hợp vào các thành phần mới được phát triển. Hệ thống hoàn chỉnh sau đó mới được thử nghiệm

Các thành phần được phát triển bởi từng thành viên sẽ có phụ đề khách nhau ở giai đoạn này. Kểm tra hệ thống là test cả hệ thống chứ không riêng rẽ các thành phần.

(Slide 19) Biểu đồ thể hiện quá trình thu thập dữ lệu bằng sequence chart

(Slide 20) Sequence diagram giúp chúng ta thiết kế các testcase cụ thể, vì nó cho thấy được input, output và những yêu cầu được tạo ra:

1. 1 input của một yêu cầu nên có 1 xác nhận có liên quan. Trong quá trình thử nghiệm, chúng ta nên tạo dữ liệu tóm tắt có thể được sử dụng để kiểm tra xem báo cáo có được tổ chức chính xác không

2. 1 yêu cầu input cho 1 báo cáo trạm thời tiết dẫn đến 1 báo cáo tóm tắt được tạo ra. Chúng ta có thể kiểm tra điều này bằng cách tạo dữ liệu thô tương ứng với bạn tóm tắt mà chúng ta chuẩn bị kiểm tra và kiểm tra xem đối tượng trạm thời tiết có tạo ra bản tóm tắt chính xác không.