**Lời nói đầu**

Thực tập tốt nghiệp là điều vô cùng cần thiết và quan trọng với mỗi sinh viên. Thật may mắn khi nhận được sự quan tâm của nhà trường để em có khoảng thời gian thực tập đầy ý nghĩa tại Công ty TMA Solution – Chi nhánh Quy Nhơn và được tiếp cận với những công nghệ kỹ thuật thực tế, hiện đại.

Mặc dù chỉ có một khoảng thời gian ngắn để thực tập tại công ty nhưng được sự chỉ bảo tận tình của anh Trần Duy Tân cùng các anh chị trong công ty mà em đã học hỏi được rất nhiều kiến thức mới, bổ ích mà ở trường em chưa từng học đến. Với sự giúp đỡ của anh Trần Duy Tân em đã hoàn thành đề tài thực tập tốt nghiệp “Tìm hiểu những kiến thức tổng quan nhất về kiểm thử, tìm lỗi và cách thiết kế Testcase trong kiểm thử phần mềm”.

Sau quá trình thực tập tại công ty, để hoàn thành đề tài trên, em xin chân thành cảm ơn lãnh đạo của công ty đã tạo điều kiện tốt nhất cho em được tiếp xúc với môi trường thực tế và anh Trần Duy Tân đã trainning cho em rất nhiều kiến thức bổ ích. Qua đó em được hướng dẫn, tìm hiểu những kiến thức về kiểm thử, tìm lỗi và cách thiết kế Testcase trong kiểm thử phần mềm.

Trong quá trình thực tập và làm báo cáo thực tập, em khó tránh khỏi những sai sót, rất mong quý Thầy Cô khoa Kỹ Thuật và Công Nghệ và anh chị trong công ty bỏ qua. Đồng thời, do kiến thức cũng như kinh nghiệm còn hạn chế nên bài báo cáo của em không tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp của quý Thầy Cô và anh chị trong công ty.

Cuối cùng em kính chúc quý Thầy Cô dồi dào sức khỏe và thành công trên con đường sự nghiệp. Đồng thời chúc toàn thể anh chị trong Công ty TMA Solution – Chi nhánh Quy Nhơn luôn dồi dào sức khỏe, đạt được nhiều thành công tốt đẹp trong công việc và cuộc sống.

***Em xin chân thành cảm ơn.***

Quy Nhơn, ngày…tháng…năm 2018

Sinh viên

Nguyễn Cao Kỳ

[**CHƯƠNG I: GIỚI THIỆU TỔNG QUAN VỀ CÔNG TY** 1](#_Toc526432376)

[**I.** **Quá trình hình thành và phát triển** 1](#_Toc526432377)

[**II.** **Cơ cấu tổ chức** 2](#_Toc526432378)

[**III. Chức năng nhiệm vụ, Lĩnh vực kinh doanh** 2](#_Toc526432379)

[**1.** **Các lĩnh vực trọng tâm** 2](#_Toc526432380)

[**2.** **Công nghệ** 3](#_Toc526432381)

[**3.** **Dịch vụ chính** 3](#_Toc526432382)

[**4.** **Niềm tự hào** 4](#_Toc526432383)

[**5.** **Bằng khen** 4](#_Toc526432384)

[**CHƯƠNG II: CÁC KIẾN THỨC CƠ BẢN** 4](#_Toc526432385)

[**I.** **Phần mềm** 5](#_Toc526432386)

[**II. Kiểm thử phần mềm và một số khái niệm liên quan** 5](#_Toc526432387)

[**1. Kiểm thử phần mềm** 5](#_Toc526432388)

[**2. Một số khái niệm liên quan** 5](#_Toc526432389)

[**3. Vòng đời phần mềm** (Software Lifecycle) 6](#_Toc526432390)

[**4. Quy trình phát triển phần mềm** 6](#_Toc526432391)

[**4.1. Mô hình thác nước** (Waterfall model) 6](#_Toc526432392)

[**4.2. Mô hình chữ V** (V model) 7](#_Toc526432393)

[**4.3. Mô hình Agile** (Agile model) 9](#_Toc526432394)

[**5.** **Các cấp độ kiểm thử** (Test Level) 14](#_Toc526432395)

[**5.1.** **Kiểm thử mức đơn vị** (Unit test) 14](#_Toc526432396)

[**5.2.** **Kiểm thử tích hợp** (Integration test) 15](#_Toc526432397)

[**5.3.** **Kiểm thử hồi quy** (Regression test) 15](#_Toc526432398)

[**5.4.** **Kiểm thử mức hệ thống** (System test) 15](#_Toc526432399)

[**5.5.** **Kiểm thử chấp nhận sản phẩm** (Acceptance test) 15](#_Toc526432400)

[**6.** **Các kỹ thuật kiểm thử phần mềm** 16](#_Toc526432401)

[**6.1.** **Nguyên tắc cơ bản kiểm thử phần mềm** 16](#_Toc526432402)

[**6.2.** **Kỹ thuật kiểm thử hộp đen** (Black-Box testing) 17](#_Toc526432403)

[**6.3.** **Kỹ thuật kiểm thử hộp trắng** (White-Box testing) 17](#_Toc526432404)

[**6.4.** **Kỹ thuật kiểm thử hộp xám** (Grey-Box testing) 18](#_Toc526432405)

[**7. Kỹ thuật thiết kế ca kiểm thử** 19](#_Toc526432406)

[**7.1.** **Cấu trúc của ca kiểm thử** 19](#_Toc526432407)

[**7.2.** **Phân vùng tương đương** 20](#_Toc526432408)

[**7.3.** **Phân vùng giá trị biên** 21](#_Toc526432409)

[**7.4.** **Đoán lỗi** 23](#_Toc526432410)

[**8. Tạo report Bug** 23](#_Toc526432411)

[**8.1.** **Bug và report Bug** 23](#_Toc526432412)

[**8.2.** **Cấu trúc một Bug report** 24](#_Toc526432413)

[**8.3.** **Severity và Priority** 24](#_Toc526432414)

[**CHƯƠNG III. ÁP DỤNG** 26](#_Toc526432415)

[**I.** **Giới thiệu về màn hình tạo tài khoản Google** 26](#_Toc526432416)

[**II.** **Kết quả** 27](#_Toc526432417)

**CHƯƠNG I: GIỚI THIỆU TỔNG QUAN VỀ CÔNG TY**

1. **Quá trình hình thành và phát triển**

TMA Solutions (gọi tắt TMA, tiếng Việt: Công ty TNHH Giải Pháp Phần Mềm Tường Minh) là một công ty tại Việt Nam, kinh doanh các dịch vụ liên quan đến phát triển phần mềm.

Vào tháng 3 năm 1997, bà Bùi Ngọc Anh thành lập công ty TMA với 6 kỹ sư tại phòng khách nhà bà. Vào thời điểm đó, một công ty công nghệ thông tin ở Canada có ý thuê nhà bà Ngọc Anh làm chi nhánh cho công ty, đồng thời cũng muốn nhờ bà hoàn thành giúp các thủ tục pháp lý để mở chi nhánh cho mình. Nhưng cuối cùng, vì lý do tài chính nên công ty này đã hủy hợp đồng, thế là bà Ngọc Anh, cùng với vốn kiến thức đã thu thập được trong thời gian qua đã cho ra đời công ty TMA.

Đến năm 1998, TMA đón nhận khách hàng đầu tiên từ Mỹ và Canada. Vào năm này, số lượng thành viên của TMA đã tăng lên gấp ba lần, tức 18 người.

Năm 1999, con số 18 người này lại một lần nữa tăng lên gấp 3, TMA đạt được mức 54 nhân viên, và dời trụ sở sang quận Phú Nhuận. Hiện nay, trụ sở này vẫn là trụ sở chính của công ty.

Vào năm 2000, TMA có thêm khách hàng mới từ Úc, Singapore, Ấn Độ và Nhật Bản.

Năm 2001, TMA có thêm trụ sở mới, cũng ở quận Phú Nhuận. Đồng thời cũng có thêm khách hàng từ Nhật Bản.

Năm 2003, TMA thành lập trung tâm nghiên cứu và phát triển R&D. Số lượng nhân viên lúc này cũng đạt được 200 người.

Năm 2005, TMA thành lập trụ sở ở nước ngoài đầu tiên tại Canada. Xét về mặt các trụ sở trong nước, cùng với trụ sở tại đường Đặng Văn Ngữ (quận Phú Nhuận) được thành lập vào năm 2004, đến năm 2005, TMA lại mở thêm một trụ sở mới trên đường Trần Hữu Trang (quận Phú Nhuận), đẩy số lượng trụ sở lên 5, góp phần mở rộng quy mô công ty hơn.

Năm 2006-2008, TMA thành lập thêm 3 chi nhánh mới ở Nhật bản, Mỹ và ở châu Âu. Vào lúc này, TMA bước đầu thâm nhập thị trường châu Âu với các khách hàng từ Đức, Pháp, Đan Mạch.

Trong khoảng năm 2009-2010, TMA hoàn thành thêm trụ sở thứ sáu của mình tại công viên phần mềm Quang Trung. Đồng thời cho ra đời Trung tâm Giải pháp Di Động TMA (TMA Mobile Solutions). Thêm vào đó, TMA mở thêm một chi nhánh mới tại Úc. Vào năm 2009, TMA cũng có mặt tại triển lãm CommunicAsia2009 ở Singapore. Ngày 29/7/2010, Trung tâm Nghiên cứu & Phát triên CNTT-TT (ICT R&D Center - iRDC) của TMA được thành lập nhằm phục vụ hợp tác phát triển công nghệ và sản phẩm mới (iRDC)

Năm 2011, TMA thành lập Tech Lab và Trung tâm Thực tập Sinh viên (SDC) nhằm đào tạo và nâng cao trình độ các thế hệ sinh viên, đặc biệt là sinh viên CNTT. Vào năm này, TMA cũng thành lập Bảo tàng và Thư viên sách cũ tại Lab6 (công viên phần mềm Quang Trung).

Năm 2012, TMA tham gia triển lãm CommunicAsia2012 tại Singapore. Cùng năm, TMA cũng tham gia triển lãm CNTT CeBIT tại Hannover, Đức. Lúc này, số lượng nhân viên của TMA là 1200 người.

Năm 2013, số lượng nhân viên tại TMA là 1400 người.

Năm 2015, với các dự án không ngừng tăng lên, số lượng kỹ sư CNTT đang làm việc tại TMA là 1900 người.

Năm 2017, kỉ niệm 20 năm thành lập và phát triển vững mạnh, số lượng kỹ sư CNTT đang làm việc tại TMA đạt hơn 2000 người.

Và sau quá trình tìm hiểu kỹ lưỡng và được sự ủng hộ của lãnh đạo tỉnh Bình Định, năm 2018 TMA quyết định thành lập Lab 8 tại tỉnh Bình Định. Theo đó, TMA sẽ xây dựng TMA Bình Định IT Park theo tiêu chí xanh, thân thiện với môi trường, có tổng diện tích 15 ha, thuộc khuôn viên Khu Đô thị Khoa học và Giáo dục Quy Hoà, P. Ghềnh Ráng, TP. Quy Nhơn. Để triển khai sớm dự án lớn này, và trong khi chờ đợi xây dựng, bước đầu TMA sẽ sử dụng tòa nhà 75 Mai Xuân Thưởng, TP. Quy Nhơn để làm văn phòng, mọi thủ tục đang được gấp rút hoàn thành để có thể khai trương vào đầu quý 2 – 2018. Số thành viên tại TMA là 2400 kỹ sư.

TMA với gần 2400 kỹ sư và hơn 20 năm phát triển vững mạnh đã góp phần khẳng định tên tuổi Việt Nam trên bản đồ ngành công nghệ thông tin thế giới. Đây cũng chính là tâm nguyện của Chú Lệ, của cô Ngọc Anh, BGĐ và các thế hệ nhân viên TMA. Đến nay, có thể nói giấc mơ đầu đã trở thành hiện thực, nhưng khát khao vươn cao, vươn xa hơn của TMA chưa bao giờ dừng lại.

1. **Cơ cấu tổ chức**

Là một công ty tư nhân, 100% vốn trong nước.

* Chủ tịch tập đoàn : Ông NGUYỄN HỮU LỆ
* CEO : Bà BÙI NGỌC ANH
* CEO TMA North America : Ông ALEX NEWCOMBE
* CEO TMA Japan : Ông SATO NOBOU
* CEO TMA Australia : Ông PHAN TÂM

## **III. Chức năng nhiệm vụ, Lĩnh vực kinh doanh**

1. **Các lĩnh vực trọng tâm**

Phần mềm mạng và viễn thông (Telecommunications)

Với 12 năm kinh nghiệm. Cung cấp những phần mềm dịch vụ R&D đến những công ty truyền thông hàng đầu như: Nortel, NTT,….

Những sản phẩm tiêu biểu:

* Service edge router.
* Security router.
* Wireless router.
* PBX, IP telephone.
* Contact center.
* Soft-switch.
* Communication server.
* Network management.
* CTI server.
* IMS/SIP.
* NGN.
* GGSN.

Ứng dụng quản lý kinh doanh (Business Applications).

* Ngôn ngữ lập trình: C/C++, Java, C#, ASP, ASP.NET, PHP, Perl, Ruby.
* Database: SQL Server, Oracle, MySQL, PostgreSQL.
* Middleware and web services: J2EE Framework, Net framework, SOAP, CORBA.

Sản phẩm/ dịch vụ cho di động (Mobile Products/Service). Hơn 9 năm cung cấp những dịch vụ phát triển ứng dụng trên mobile.

Đội ngũ trên 100 lập trình viên.

* + - Microsoft Solutions: TMA là một Microsoft Gold Certified Partner từ năm 2007 và hơn 11 năm kinh nghiệm trong các công nghệ của Microsoft.
    - Giải pháp mã nguồn mở: TMA Solutions có nhiều năm kinh nghiệm với các giải pháp mã nguồn mở. Căn cứ vào nhu cầu khách hàng, đội ngũ của chúng tôi đã cung cấp nhiều giải pháp mã nguồn mở sử dụng được xây dựng trước các thành phần và các gói để rút ngắn thời gian phát triển, tăng tính linh hoạt và tiết kiệm chi phí.
    - Giải pháp Java: TMA đã hơn 10 năm kinh nghiệm trong việc tận dụng các công nghệ Java để cung cấp giải pháp phần mềm cho các khánh hàng của công ty. Công ty cũng đã chuyển thành công nhiều ứng dụng hiện có để môi trường Java có khả năng mở rộng tốt hơn và hiệu suất. Thông qua nhiều dự án thành công công ty đã xây dựng được một đội ngũ gần Swing100 lập trình viên Java và kiến trúc sư.

1. **Công nghệ**

* J2EE, J2SE, J2ME.
* JSP/Servlet, Javascript, JSF, Facelets.
* Tag libs, Java Beans, Custom Tags.
* Ajax, JavaMail, JMS.
* Java Internationlization.

Phần mềm nhúng: TMA có hơn 8 năm kinh nghiệm trong các dự án phần mềm nhúng cho nhiều công ty lớn: Nortel, Juniper Networks, Flextronics, NTT, Toshiba.

Đào tạo theo yêu cầu (IT Training): TMA có trung tâm đào tạo được gọi là TMA Training Center, giúp các sinh viên mới ra trường tự tin khi xin việc và thành công trong các công ty lớn bằng các trang bị cho các bạn các kỹ năng cần thiết trong môi trường làm việc chuyên nghiệp:

* Kỹ năng giao tiếp và thuyết trình.
* Sử dụng ngoại ngữ trong công việc.
* Quy trình phát triển phần mềm.
* Công nghệ chuyên ngành.
* Kinh nghiệm thực tiễn.

1. **Dịch vụ chính**

Dịch vụ giải pháp phần mềm: Tư vấn lựa chọn giải pháp; Tích hợp phần mềm; Phần mềm mạng và viễn thông; Ứng dụng di động; Ứng dụng quản lý kinh doanh.

Dịch vụ triển khai mạng viễn thông: Lắp đặt, nâng cấp mạng và thiết bị viễn thông; Kiểm tra toàn diện trước khi đưa vào hệ thống và vận hành; Hỗ trợ kĩ thuật và bảo trì.

Dịch vụ đánh giá an toàn thông tin: Kiểm tra sự an toàn thông tin và năng lực bào mật của doanh nghiệp, tìm ra những rủi ro, đề ra những giải pháp để khắc phục.

Dịch vụ cơ sở hạ tầng thông tin: Tư vấn, lựa chọn giải pháp cho hệ thống thông tin và mạng máy tính; Thiết kế và cài đặt hệ thống mạng máy tính, mạng WIFI, WAN, VPN và điện thoại nội bộ; Tư vấn triển khai và đào tạo về sử dụng phần mềm mã nguồn mở.

1. **Niềm tự hào**

Công ty có quy mô lớn nhất thành phố Hồ Chí Minh và thứ hai Việt Nam trên 1000 kỹ sư.

15 năm liền đạt huy chương vàng xuất khẩu phần mềm.

Công ty phần mềm viễn thông hàng đầu Đông Nam Á.

Được các công ty hàng đầu thế giới tin tưởng và hợp tác lâu dài: Avaya, IBM, Alcatel-Lucent, Nortel, Juniper Networks, Flextronics, Genband, NTT, Toshiba, Andrew, Telus, NEC, etc.

Được nhiều đài truyền hình và báo chí quốc tế giới thiệu: CNN (Mỹ), NHK (Nhật), Global (Brazil), National (Thái Lan), Nikkei Computer (Nhật), để minh chứng cho sự phát triển của ngành phần mềm Việt Nam.

Chứng minh năng lực của các kỹ sư Việt Nam trong lĩnh vực công nghệ cao.

1. **Bằng khen**

Bằng Khen của Bộ Thông tin và Truyền thông về thành tích cải tiến chất lượng năm 2007.

Bằng khen của Ủy ban Nhân dân Thành phố Hồ Chí Minh: Đã có thành tích xuất sắc trong lĩnh vực CNTT-TT, góp phần tích cực vào sự phát triển CNTT-TT của Thành phố (10 năm liên tục từ 2003 đến 2012).

Chứng nhận và cúp của Hội Tin học Thành phố Hồ Chí Minh (HCA).

Huy Chương Vàng Xuất khẩu Phần Mềm (15 năm liền từ 2004 đến 2018).

Top 5 Đơn Vị Gia Công Xuất khẩu Phần Mềm Hàng Đầu năm 2009, 2010 và 2012.

Bằng khen của VINASA: Đã có thành tích xuất sắc đóng góp cho hoạt động của Hiệp hội và cho sự phát triển của ngành phần mềm và dịch vụ công nghệ thông tin VN năm 2012.

Là một trong 15 công ty hàng đầu thế giới trong việc áp dụng hiệu quả quy trình gia công phần mềm (Báo cáo của công ty tư vấn Mỹ Aberdeen, 09/2002).

Đạt các chứng chỉ chất lượng quốc tế (CMMI-L5, TL 9000, ISO 9001:2000, ISO 27001:2005).

**CHƯƠNG II: CÁC KIẾN THỨC CƠ BẢN**

Kiểm thử nhằm đánh giá chất lượng hoặc tính chấp nhận được của sản phẩm. Ngoài ra, kiểm thử còn giúp phát hiện lỗi hoặc bất cứ vấn đề gì về sản phẩm. Chúng ta cần kiểm thử vì biết rằng con người luôn có thể mắc sai lầm. Điều này đặc biệt đúng trong lĩnh vực phát triển phần mềm và các hệ thống điều khiển bởi phần mềm. Chương này sẽ giới thiệu các khái niệm trong lĩnh vực kiểm thử phần mềm.

1. **Phần mềm**

Phần mềm thường được mô tả bởi ba thành phần cấu thành:

**1.** Tập các lệnh (chương trình máy tính) trên máy tính khi thực hiện sẽ tạo ra các dịch vụ và đem lại những kết quả mong muốn cho người dùng.

**2.** Các cấu trúc dữ liệu (lưu giữ trên các bộ nhớ) làm cho chương trình thao tác hiệu quả với các thông tin thích hợp và nội dung thông tin được số hóa.

**3.** Các tài liệu để mô tả thao tác, cách sử dụng và bảo trì phần mềm (hướng dẫn sử dụng, tài liệu kỹ thuật, tài liệu phân tích, thiết kế, kiểm thử, v.v.).

**II. Kiểm thử phần mềm và một số khái niệm liên quan**

**1. Kiểm thử phần mềm**

Kiểm thử phần mềm là một cuộc kiểm tra được tiến hành để cung cấp cho các bên liên quan thông tin về chất lượng của sản phẩm hoặc dịch vụ được kiểm thử. Kiểm thử có thể cung cấp cho doanh nghiệp một quan điểm, một cách nhìn độc lập về phần mềm để từ đó cho phép đánh giá và thấu hiểu được những rủi ro trong quá trình triển khai phần mềm.

Trong kỹ thuật kiểm thử không chỉ giới hạn ở việc thực hiện một chương trình hoặc ứng dụng với mục đích đi tìm các lỗi phần mềm (bao gồm các lỗi và các thiếu sót) mà còn là một quá trình phê chuẩn và xác minh một chương trình máy tính/ứng dụng/sản phẩm nhằm:

* Đáp ứng được mọi yêu cầu hướng dẫn khi thiết kế và phát triển phần mềm.
* Thực hiện công việc đúng như kỳ vọng.
* Có thể triển khai được với những đặc tính tương tự.
* Đáp ứng được mọi nhu cầu của các bên liên quan.

Tùy thuộc vào từng phương pháp, việc kiểm thử có thể được thực hiện bất cứ lúc nào trong quá trình phát triển phần mềm. Theo truyền thống thì các nỗ lực kiểm thử được tiến hành sau khi các yêu cầu được xác định và việc lập trình được hoàn tất nhưng trong Agile (là một tập hợp các phương pháp phát triển phần mềm linh hoạt dựa trên việc lặp đi lặp lại và gia tăng giá trị) thì việc kiểm thử được tiến hành liên tục trong suốt quá trình xây dựng phần mềm. Như vậy, mỗi một phương pháp kiểm thử bị chi phối theo một quy trình phát triển phần mềm nhất định.

**2. Một số khái niệm liên quan**

Chất lượng phần mềm (Software quality): Là mức độ mà một hệ thống, thành phần hay quy trình đáp ứng các yêu cầu của đặc tả phần mềm, các nhu cầu mong đợi của khách hàng hoặc người sử dụng.

Đảm bảo chất lượng phần mềm (Software quality assurance): Là một quy trình có kế hoạch và hệ thống của tất cả các hành động cần thiết để cung cấp các thông tin đầy đủ để đảm bảo các sản phẩm có phù hợp với các yêu cầu về kỹ thuật hay không. Mục đích cuối cùng là để đánh giá quy trình sản xuất sản phẩm phần mềm.

Lỗi (Error): Lỗi là những vấn đề mà con người mắc phải trong quá trình phát triển các sản phẩm phần mềm.

Sai (Fault): Sai là kết quả của lỗi, hay nói khác đi, lỗi sẽ dẫn đến sai.

Thất bại (Failure): Thất bại xuất hiện khi một lỗi được thực thi.

Sự cố (Incident): Khi thất bại xuất hiện, nó có thể hiển thị hoặc không, tức là rõ ràng hoặc không rõ ràng đối với người dùng hoặc người kiểm thử. Sự cố là triệu chứng liên kết với một thất bại và thể hiện cho người dùng hoặc người kiểm thử về sự xuất hiện của thất bại này.

Ca kiểm thử (Test case): Ca kiểm thử gồm một tập các dữ liệu đầu vào và một xâu các giá trị đầu ra mong đợi đối với phần mềm, mục đích là dựa vào đó để kiểm tra xem phần mềm có thỏa các yêu cầu đặt ra hay không.

Kịch bản kiểm thử (Test script): Một kịch bản kiểm thử là một nhóm mã lệnh dạng đặc tả kịch bản dùng để tự động hóa một quy trình hay một ca kiểm tra, giúp cho việc kiểm tra nhanh hơn, hoặc cho những trường hợp mà kiểm tra bằng tay sẽ rất khó khăn hoặc không khả thi.

**3. Vòng đời phần mềm** (Software Lifecycle)

Mô hình vòng đời phát triển phần mềm là một loạt các giai đoạn mà phần mềm trải qua từ bắt đầu phát triển đến khi phần mềm bị loại bỏ hoàn toàn. Mô hình vòng đời phát triển phần mềm thường gồm những giai đoạn sau:

* Giai đoạn yêu cầu: Là giai đoạn đầu tiên trong quá trình xây dựng phần mềm, nhằm xác định yêu các yêu cầu phải có trong phần mềm giữa nhóm phát triển và khách hàng.
* Giai đoạn phân tích: Giai đoạn này phân tích các yêu cầu của khách hàng được mô tả chi tiết kết quả đầu ra, quá trình phát triển phần mềm dưới dạng “tài liệu đặc tả”.
* Giai đoạn thiết kế: Giai đoạn này căn cứ vào tài liệu đặc tả để mô tả chi tiết cách phần mềm thực hiện các công việc cụ thể.
* Giai đoạn lập trình: Giai đoạn này là quá trình thực hiện viết chương trình bằng một ngôn ngữ cụ thể.
* Giai đoạn kiểm thử hệ thống: Giai đoạn này hoạt động sau khi giai đoạn lập trình kết thúc. Mục đích chính là phát hiện lỗi phần mềm càng nhiều càng tốt để đạt được chất lượng phần mềm ở mức chấp nhận được sau khi chỉnh sửa.
* Giai đoạn bảo trì và loại bỏ: Pha này bảo trì sửa lỗi có thể còn xuất hiện trong chương trình sau khi cài đặt cho khách hàng và cập nhật sửa đổi phần mềm theo ý khách hàng hoặc thích nghi với điều kiện ràng buộc để nâng cao hiệu quả làm việc. Nếu chi phí bảo trì quá lớn có thể dẫn đến loại bỏ phần mềm.

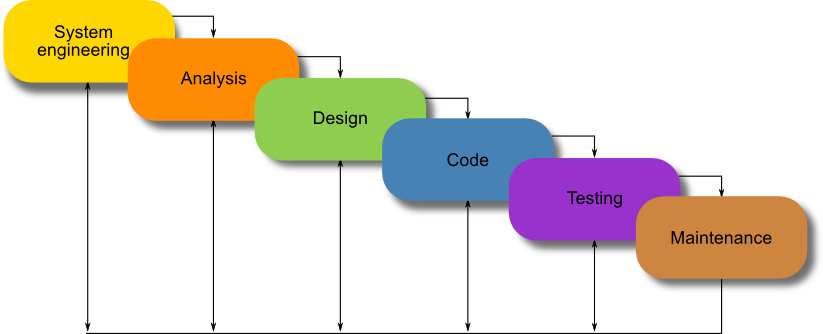
**4. Quy trình phát triển phần mềm**

Phần mềm được phát triển dựa trên các mô hình để xác định các hoạt động và quy trình theo trình tự nhất định. Một số mô hình phát triển phần mềm tiêu biểu sau:

**4.1. Mô hình thác nước** (Waterfall model)

**4.1.1. Khái niệm**

Mô hình thác nước hay còn gọi là mô hình vòng đời truyền thống do tác giả Royce đề xuất năm 1970. Mô hình này yêu cầu tiếp cận một cách hệ thống, tuần tự và chặt chẽ đối với việc phát triển phần mềm, bắt đầu ở mức hệ thống và tiến dần xuống phân tích, thiết kế, mã hóa, kiểm thử, và bảo trì.

****

*Hình 2.1: Mô hình thác nước*

Mô hình thác nước là mô hình áp dụng theo tính tuần tự của giai đoạn phát triển phần mềm. Giai đoạn sau chỉ được thực hiện khi giai đoạn trước được hoàn thành.

**4.1.2. Ưu điểm và nhược điểm của mô hình thác nước**

1. **Ưu điểm**

* Đơn giản, dễ hiểu và dễ sử dụng.
* Đối với cácdự án nhỏ, mô hình thác nước hoạt động tốt và mang lại kết quả phù hợp.
* Vì các giai đoạn của mô hình thác nước cứng nhắc và chính xác, mỗi pha thực hiện một lần và nó rất dễ để duy trì.
* Các tiêu chí đầu vào và đầu ra được xác định rõ rang, do đó nó dễ dàng và có hệ thống để tiến hành chất lượng.
* Kết quả được ghi chép tốt.

1. **Nhược điểm**

* Không thể chấp nhận thay đổi yêu cầu.
* Nó trở nên khó khăn để di chuyển trở lại giai đoạn.
* Việc giao hàng của sản phẩm cuối cùng là muộn vì không có mẫu thử nghiệm được chứng minh trung gian.
* Đối với dự án lớn và phức tạp, mô hình này không tốt vì rủi ro cao hơn.
* Không thích hợp cho các dự án mà yêu cầu được thay đổi thường xuyên.
* Không làm việc cho các dự án dài và đang xảy ra.
* Kể từ khi thử nghiệm được thực hiện ở giai đoạn sau, nó không cho phép xác định những thách thức và rủi ro trong giai đoạn trước đó nên cách giảm thiểu rủi ro rất khó chuẩn bị.

**4.2. Mô hình chữ V** (V model)

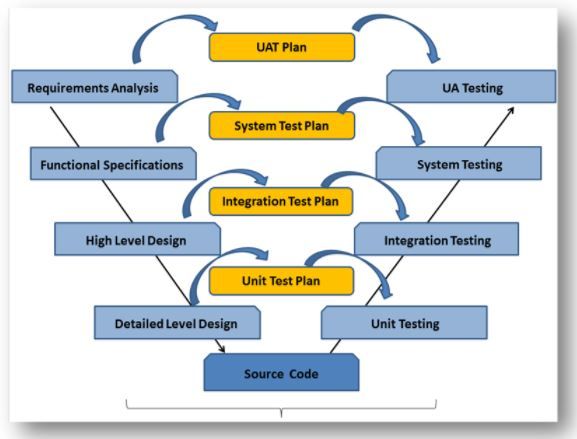
* + 1. **Khái niệm**

Mô hình chữ V hiện nay là một trong những quy trình phát triển phần mềm được sử dụng rộng rãi nhất. Trong mô hình chữ V việc thực hiện kiểm tra được diễn ra ngay từ giai đoạn lấy yêu cầu. V mô hình cũng được gọi là mô hình xác minh (verification) và mô hình xác nhận (validation).

Để hiểu được mô hình chữ V, trước hết chúng ta hãy hiểu xác minh (verification) và xác nhận hợp lệ (validation) trong phần mềm là gì.

* Xác minh (Verification): Xác minh là một kỹ thuật phân tích tĩnh. Trong kiểm thử, kỹ thuật này được thực hiện mà không phải chạy code. Nó bao gồm một số hoạt đông như xem lại (review), kiểm tra (inspection) và kiểm tra từ đầu tới cuối (walkthrough).
* Xác nhận (Validation): Xác nhận là một kỹ thuật phân tích động, trong đó việc kiểm thử được thực hiện bằng cách thực hiện code. Nó bao gồm kỹ thuật kiểm tra chức năng (function) và phi chức năng (non-function).

Trong mô hình chữ V, các hoạt động phát triển và đảm bảo chất lượng được thực hiện đồng thời. Không có pha rời rạc được gọi là kiểm thử, thay vào đó kiểm thử được bắt đầu ngay từ giai đoạn lấy yêu cầu. Các hoạt động xác minh và xác nhận đi liền với nhau. Để hiểu mô hình chữ V, chúng ta hãy nhìn vào hình dưới đây:



*Hình 2.2. Mô hình chữ V*

Mô hình chữ V được thực hiện từ trái qua phải, mô tả dãy các hoạt động và kiểm thử cơ bản. Mô hình này nêu bật các mức kiểm thử liên quan đến các pha phát triển phần mềm tương ứng từng giai đoạn.

* + 1. **Ưu điểm và nhược điểm của mô hình chữ V**

1. **Ưu điểm**

* Quá trình phát triển và quy trình quản lý có tổ chức và hệ thống.
* Hoạt động tốt cho các dự án có quy mô vừa và nhỏ.
* Kiểm tra từ khi bắt đầu phát triển vì vậy sự mơ hồ được xác định ngay từ đầu.
* Dễ dàng quản lý vì mỗi giai đoạn có các mục tiêu được xác định rõ ràng.

1. **Nhược điểm**

* Không thích hợp cho các dự án lớn và phức tạp.
* Không phù hợp nếu các yêu cầu thường xuyên thay đổi.
* Không có phần mềm làm việc được sản xuất ở giai đoạn trung gian.
* Không có điều khoản cho việc phân tích rủi ro nên có sự không chắc chắn và có tính rủi ro.

**4.3. Mô hình Agile** (Agile model)

* + 1. **Khái niệm**

Mô hình Agile là một tập hợp các phương thức phát triển lặp và tăng dần trong đó các yêu cầu và giải pháp được phát triển thông qua sự liên kết cộng tác giữa các nhóm tự quản và liên chức năng. Agile là cách thức làm phần mềm linh hoạt để làm sao đưa sản phẩm đến tay người dùng càng sớm càng tốt và được xem như là sự cải tiến so với những mô hình cũ như mô hình “Thác nước (waterfall)”.

* + 1. **Tuyên ngôn của Agile**

Cá nhân và sự tương hỗ quan trọng hơn quy trình và công cụ.

Sản phẩm xài được quan trọng hơn tài liệu về sản phẩm.

Cộng tác với khách hàng quan trọng hơn đàm phán hợp đồng.

Phản hồi với sự thay đổi quan trọng hơn bám theo kế hoạch.

* + 1. **Ưu điểm và nhược điểm của mô hình Agile**

1. **Ưu điểm**

* Đạt được sự hài lòng của khách hàng bằng cách bàn giao nhanh chóng, liên tục các sản phẩm phần mềm có ích.
* Con người và tương tác được nhấn mạnh hơn là quá trình và công cụ. Khách hàng, nhà phát triển và người thử nghiệm liên tục trao đổi với nhau.
* Phần mềm làm việc được bàn giao thường xuyên (vài tuần chứ không phải vài tháng).
* Cuộc đối thoại trực tiếp (face-to-face) là hình thức giao tiếp tốt nhất.
* Gần gũi với nhau hơn, hợp tác hàng ngày giữa các khách hàng và các lập trình viên.
* Chú ý liên tục đến sự xuất sắc về kỹ thuật và bản thiết kế tốt.
* Thường xuyên thích nghi với hoàn cảnh thay đổi.

1. **Nhược điểm**

* Trong trường hợp một số sản phẩm phần mềm, đặc biệt là các sản phẩm lớn, rất khó để đánh giá những nỗ lực bắt buộc khi bắt đầu chu trình phát triển phần mềm.
* Thiếu sự nhấn mạnh vào thiết kế và tài liệu cần thiết.
* Dự án có thể dễ dàng bị tắt nếu đại diện khách hàng không rõ kết quả cuối cùng mà họ muốn.
* Chỉ những lập trình viên cao cấp mới có thể đưa ra các quyết định cần thiết trong quá trình phát triển. Do đó nó không có nơi cho các lập trình mới, trừ khi kết hợp với các nguồn lực có kinh nghiệm.

**4.3.4. Một số phương pháp của Agile**

**a. Phương pháp Scrum**

* **Khái niệm:** Scrum là một khung làm việc để phát triển bền vững các sản phẩm phức tạp. Có thể hiểu đây là khung tổ chức công việc tổng quát hướng đến phát triển các sản phẩm phức tạp, chủ yếu là phần mềm. Tuy vậy, Scrum có thể được dùng như là nền tảng tổ chức các công việc khác nhau, từ quản trị dự án linh hoạt nói chung, đến phát triển sản phẩm, thực hiện các chiến dịch marketing, tổ chức dạy học, sản xuất ô tô module hóa hoặc những công việc cá nhân khác.
* **Phân tích các thành phần trong Scrum:**

Scrum là một phương pháp linh hoạt (Agile), vì thế nó tuân thủ các nguyên tắc của Tuyên ngôn Agile (Manifesto for Agile Software Development). Ngoài ra Scrum hoạt động dựa trên ba giá trị cốt lõi, còn gọi là Ba chân của Scrum bao gồm Minh bạch, Thanh tra và Thích nghi:

* *Minh bạch* *(Transparency):* Trong Scrum, tính minh bạch được đề cao như là giá trị cốt lõi cơ bản nhất. Muốn thành công với Scrum, thông tin liên quan tới quá trình phát triển phải minh bạch và thông suốt. Các thông tin đó có thể là: tầm nhìn về sản phẩm, yêu cầu khách hàng, tiến độ công việc, các khúc mắc và rào cản v.v. Từ đó mọi người ở các vai trò các nhau có đủ thông tin cần thiết để tiến hành các quyết định có giá trị để nâng cao hiệu quả công việc. Các công cụ và cuộc họp trong Scrum luôn đảm bảo thông tin được minh bạch cho các bên.
* *Thanh tra (Inspection):* Công tác thanh tra liên tục các hoạt động trong Scrum đảm bảo cho việc phát lộ các vấn đề cũng như giải pháp để thông tin đa dạng và hữu ích đến được với các bên tham gia dự án. Truy xét kĩ càng và liên tục là cơ chế khởi đầu cho việc thích nghi và các cải tiến liên tục trong Scrum.
* *Thích nghi* *(Adaptation):* Scrum rất linh hoạt như các phương pháp phát triển linh hoạt (Agile software development) khác. Nhờ đó nó mang lại tính thích nghi rất cao. Dựa trên các thông tin minh bạch hóa từ các quá trình thanh tra và làm việc, Scrum có thể phản hồi lại các thay đổi một cách tích cực, nhờ đó mang lại thành công cho dự án.

Trong Scrum, đội ngũ tham gia phát triển phần mềm được phân chia ra ba vai trò với trách nhiệm rõ ràng để đảm bảo tối ưu hóa các công việc đặc thù. Ba vai trò này bao gồm: Product Owner (chủ sản phẩm), Scrum Master và Development Team (Đội sản xuất hay nhóm phát triển):

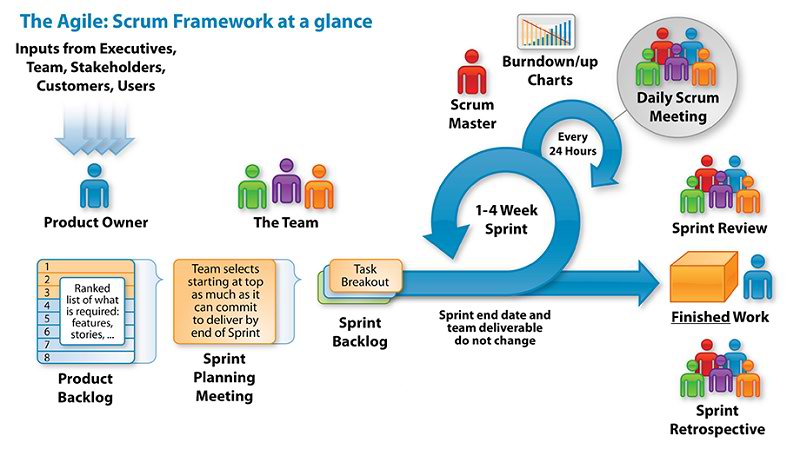
* *Product Owner* *(Chủ sản phẩm):* Là người chịu trách nhiệm về sự thành công của dự án, người định nghĩa các yêu cầu và đánh giá cuối cùng đầu ra của các nhà phát triển phần mềm.
* *Scrum Master:* Là người có hiểu biết sâu sắc về Scrum và đảm bảo nhóm có thể làm việc hiệu quả với Scrum.
* *Development Team* *(Đội sản xuất, hay nhóm phát triển):* Một nhóm liên chức năng (cross-functional) tự quản lý để tiến hành chuyển đổi các yêu cầu được tổ chức trong Product Backlog thành chức năng của hệ thống.

Scrum định nghĩa quy tắc cho bốn sự kiện chủ chốt nhằm tạo môi trường và quy cách hoạt động và cộng tác cho các thành viên trong dự án. Các sự kiện này diễn ra trước khi Sprint bắt đầu (là sự kiện lập kế hoạch – Sprint Planning), trong khi Sprint diễn ra (sự kiện Daily Scrum) và sau khi Sprint kết thúc (sự kiện Sprint Review và Sprint Retrospective):

* *Sprint Planning (Họp Kế hoạch Sprint):* Nhóm phát triển gặp gỡ với Product Owner để lên kế hoạch làm việc cho một Sprint (xem thêm phần Sprint bên dưới). Công việc lập kế hoạch bao gồm việc chọn lựa các yêu cầu cần phải phát triển, phân tích và nhận biết các công việc phải làm kèm theo các ước lượng thời gian cần thiết để hoàn tất các tác vụ. Scrum sử dụng cách thức lập kế hoạch từng phần và tăng dần theo thời gian, theo đó, việc lập kế hoạch không diễn ra duy nhất một lần trong vòng đời của dự án mà được lặp đi lặp lại, có sự thích nghi với các tình hình thực tiễn trong tiến trình đi đến sản phẩm.
* *Daily Scrum* *(Họp Scrum hằng ngày):* Scrum Master tổ chức cho Đội sản xuất họp hằng ngày trong khoảng 15 phút để Nhóm Phát triển chia sẻ tiến độ công việc cũng như chia sẻ các khó khăn gặp phải trong quá trình phát triển phần mềm suốt một Sprint.
* *Sprint Review* *(Họp sơ kết Sprint):* Cuối Sprint, nhóm phát triển cùng với Product Owner sẽ rà soát lại các công việc đã hoàn tất (Done) trong Sprint vừa qua và đề xuất các chỉnh sửa hoặc thay đổi cần thiết cho sản phẩm.
* *Sprint Retrospective* *(Họp cải tiến Sprint):* Dưới sự trợ giúp của Scrum Master, nhóm phát triển sẽ rà soát lại toàn diện Sprint vừa kết thúc và tìm cách cải tiến quy trình làm việc cũng như bản thân sản phẩm.

Scrum sử dụng các công cụ rất đơn giản nhưng hiệu quả để trợ giúp công việc. Chúng bao gồm bản yêu cầu của Product Owner được gọi là Product Backlog, bản kế hoạch của từng Sprint (Sprint Backlog) và phần sản phẩm chuyển giao được cuối mỗi Sprint (Increment):

* *Product Backlog:* Đây là danh sách ưu tiên các tính năng (Feature) hoặc đầu ra khác của dự án, có thể hiểu như là danh sách yêu cầu (Requirement) của dự án. Product Owner chịu trách nhiệm sắp xếp độ ưu tiên cho từng hạng mục (Product Backlog Item) trong Product Backlog dựa trên các giá trị do Product Owner định nghĩa (thường là giá trị thương mại – Business value).
* *Sprint Backlog***:** Đây là bản kế hoạch cho một Sprint; là kết quả của buổi họp lập kế hoạch (Sprint Planning). Với sự kết hợp của Product Owner, nhóm sẽ phân tích các yêu cầu theo độ ưu tiên từ cao xuống thấp để hiện thực hóa các hạng mục trong Product Backlog dưới dạng danh sách công việc (Todo list).
* *Phần tăng trưởng (Increment):* Sau mỗi Sprint, nhóm phát triển phải chuyển giao một phần công việc hoàn chỉnh cho ProductOwner hoặc khách hàng, gọi là Increment (Phần sản phẩm hoàn chỉnh chuyển giao được). Phần này chính là phần công việc hoàn chỉnh tính tới thời điểm chuyển giao, tích hợp tốt với những phần công việc trước đó và đáp ứng tiêu chuẩn hoàn thành của nhóm.
* **Quá trình vận hành của Scrum:**



*Hình 2.3. Sơ đồ vận hành của Scrum*

Product Owner tạo ra Product Backlog chứa các yêu cầu của dự án với các hạng mục được sắp theo thứ tự ưu tiên. Đội sản xuất sẽ thực hiện việc hiện thực hóa dần các yêu cầu của Product Owner với sự lặp đi lặp lại các giai đoạn nước rút từ 1 đến 4 tuần làm việc (gọi là Sprint) với đầu vào là các hạng mục trong Product Backlog, đầu ra là các gói phần mềm hoàn chỉnh có thể chuyển giao được (Potentially Shippable Product Increment).

Trước khi cả nhóm cùng đua nước rút trong Sprint, đội sản xuất cùng họp với Product Owner để lập kế hoạch cho từng Sprint. Kết quả của buổi lập kế hoạch (theo cách làm của Scrum) là Sprint Backlog chứa các công việc cần làm trong suốt một Sprint.

Trong suốt quá trình phát triển, nhóm sẽ phải cập nhật Sprint Backlog và thực hiện cuộc họp hằng ngày kéo dài 15 phút (Daily Scrum) để chia sẻ tiến độ công việc cũng như các vướng mắc trong quá trình làm việc cùng nhau. Nhóm được trao quyền để tự quản lí và tổ chức lấy công việc của mình để hoàn thành công việc trong Sprint.

Khi kết thúc Sprint, nhóm tạo ra các gói phần mềm có chức năng hoàn chỉnh, sẵn sàng chuyển giao (shippable) cho khác hàng. Buổi họp sơ kết Sprint (Sprint Review) ở cuối Sprint sẽ giúp khách hàng thấy được nhóm đã có thể chuyển giao những gì, còn những gì phải làm hoặc còn gì phải thay đổi hay cải tiến. Sau khi kết thúc việc đánh giá Sprint, Scrum Master và nhóm cùng tổ chức họp cải tiến Sprint (Sprint Retrospective) để tìm kiếm các cải tiến trước khi Sprint tiếp theo bắt đầu, điều này sẽ giúp nhóm liên tục học hỏi và trưởng thành qua từng Sprint.

Các Sprint sẽ được lặp đi lặp lại cho tới khi nào các hạng mục trong Product Backlog đều được hoàn tất hoặc khi Product Owner quyết định có thể dừng dự án căn cứ tình hình thực tế.

Do sử dụng chiến thuật “có giá trị hơn làm trước” nên các hạng mục mang lại nhiều giá trị hơn cho chủ dự án luôn được hoàn tất trước. Do đó Scrum luôn mang lại giá trị cao nhất cho người đầu tư cho dự án. Do quy trình luôn luôn được cải tiến, nhóm Scrum thường có năng suất lao động rất cao. Đây là hai lợi ích to lớn mà Scrum mang lại cho tổ chức.

**b. Phương pháp Kanban**

****

*Hình 2.4 Phương pháp Kanban*

* **Khái niệm:**

Kanban dịch từ tiếng Nhật có nghĩa là “bảng thông tin”. Còn đúng chính xác thuật ngữ chuyên môn kinh tế thì phải là “Phương pháp quản lý Kanban” (Kanban method). Đây là một thuật ngữ bắt nguồn từ công ty chế tạo xe hơi Toyota. Nơi có phương thức quản lý xí nghiệp thông minh, tạo đòn bẩy phát triển kinh tế của Nhật bản và là tiêu chuẩn quản lý của các tập đoàn sản xuất lớn của Nhật hiện tại. Phương thức quản lý ở Toyota bao gồm 1 phần rất quan trọng là “Phương thức quản lý Kanban”.

Kanban là công cụ kiểm soát sản xuất, có thể có nhiều màu sắc để chỉ định nguyên liệu và các công đoạn khác nhau. Đối với trạm công việc này kanban là một phiếu (thẻ) đặt hàng, còn đối với trạm kế tiếp nó trở thành một phiếu vận chuyển – chỉ định rõ phải nhận bộ phận, chi tiết hay nguyên liệu nào từ trạm trước nó với số lượng bao nhiêu.

* **Nguyên tắc của Kanban:**

Mỗi thùng hàng phải chứa một thẻ Kanban trên đó ghi tên chi tiết, nơi sản xuất, nơi chuyển đến và số lượng.

Chi tiết luôn được “kéo” bởi công đoạn sau.

Không bắt đầu sản xuất khi không nhận được Kanban.

Mỗi khay, thùng phải đựng đúng số lượng được chỉ định.

Không được giao chi tiết phế phẩm cho công đoạn sau.

Số lượng kanban cần được giảm thiểu.

Khoảng thời gian giữa các lần giao cần được giảm thiểu.

* **Ưu điểm và nhược điểm của Kanban:**

**Ưu điểm**

* Độ chính xác giờ giấc.
* Độ chính xác sản phẩm.
* Tiết kiệm tối đa vật tư và nguyên liệu.
* Vòng đời sản phẩm quay nhanh vì khả năng phân tán lao động cao.

**Nhược điểm:**

* Đòi hỏi phải có một hệ thống cơ sở hạ tầng của xã hội tốt, hoàn hảo.
* Đòi hỏi toàn dây chuyền sản xuất phải có một hệ thống nhân viên và kỹ thuật viên có trình độ và kiến thức cao, ý thức kỷ luật lao động cao, bởi vì chỉ cần một nhân viên của bộ phận vệ tinh vô kỷ luật, kiểm tra một con ốc không kỹ thì cả dây chuyền phải ngưng làm việc.
* Đòi hỏi Chính phủ, Nhà nước phải có một hệ thống văn bản pháp luật hỗ trợ sản xuất rành mạch, minh bạch và nghiêm minh, một hệ thống nhân viên chính phủ giữ đúng kỷ cương tôn trọng pháp luật vì ví dụ chỉ cần một nhân viên hải quan nhũng nhiểu làm khó dễ trong lúc chuyển vận hàng hoá phụ kiện là sẽ kéo theo việc ngưng hoạt động toàn bộ dây chuyền liên quan đến toàn bộ quy trình hoạt động ngay lúc đó.
* Đòi hỏi chế độ bảo mật kỹ thuật đối với các bộ phận vệ tinh nghiêm ngặt, nếu không rất dễ bị lộ kỹ thuật ra ngoài.

1. **Các cấp độ kiểm thử** (Test Level)

Các mức kiểm thử phần mềm thông thường:

*Unit Test:* Kiểm thử mức đơn vị.

*Integration Test:* Kiểm thử tích hợp.

*Regression Test:* Kiểm thử hồi quy.

*System Test:* Kiểm thử mức hệ thống.

*Acceptance Test:* Kiểm thử chấp nhận sản phẩm.

* 1. **Kiểm thử mức đơn vị** (Unit test)

Một đơn vị kiểm thử là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Theo định nghĩa này, các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method) đều có thể được xem là đơn vị kiểm thử.

Vì đơn vị kiểm thử được chọn để kiểm thử thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức, kiểm thử, ghi nhận và phân tích kết quả kiểm thử. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn vị đang kiểm thử. Một nguyên lý đúc kết từ thực tiễn: Thời gian tốn cho kiểm thử đơn vị sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức kiểm thử sau đó.

Kiểm thử đơn vị thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Thông thường, kiểm thử đơn vị đòi hỏi kiểm thử viên có kiến thức về thiết kế và mã nguồn của chương trình. Mục đích của kiểm thử đơn vị là bảo đảm thông tin được xử lý và xuất ra là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của đơn vị kiểm thử. Điều này thường đòi hỏi tất cả các nhánh bên trong đơn vị kiểm thử đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một đơn vị kiểm thử, ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then … else là một nhánh. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết các đơn vị kiểm thử đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Cũng như các mức kiểm thử khác, kiểm thử đơn vị cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử hoặc kịch bản kiểm thử, trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong chờ sẽ xuất ra. Các ca kiểm thử và kịch bản này nên được giữ lại để tái sử dụng.

Kiểm thử đơn vị thường sử dụng các Unit Test Framework, đó là các khung chương trình được viết sẵn để hộ trợ cho việc test các mô đun, các đơn vị phần mềm.

* 1. **Kiểm thử tích hợp** (Integration test)

Kiểm thử tích hợp kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi kiểm thử đơn vị kiểm thử các thành phần và đơn vị phần mềm riêng lẻ thì kiểm thử tích hợp kết hợp chúng lại với nhau và kiểm thử sự giao tiếp giữa chúng. Kiểm thử tích hợp có 2 mục tiêu chính:

* Phát hiện lỗi giao tiếp xảy ra giữa các đơn vị kiểm thử.
* Tích hợp các đơn vị kiểm thử đơn lẻ thành các hệ thống nhỏ (subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (system) chuẩn bị cho kiểm thử ở mức hệ thống.
  1. **Kiểm thử hồi quy** (Regression test)

Kiểm thử hồi quy không phải là một mức kiểm thử, như các mức khác đã nói ở trên. Nó đơn thuần kiểm tra lại phần mềm sau khi có một sự thay đổi xảy ra, để bảo đảm phiên bản phần mềm mới thực hiện tốt các chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt. Kiểm thử hồi quy có thể thực hiện tại mọi mức kiểm thử. Ví dụ: một phần mềm đang phát triển khi kiểm tra cho thấy nó chạy tốt các chức năng A, B và C. Khi có thay đổi code của chức năng C, nếu chỉ kiểm tra chức năng C thì chưa đủ, cần phải kiểm tra lại tất cả các chức năng khác liên quan đến chức năng C, trong ví dụ này là A và B. Lý do là khi C thay đổi, nó có thể sẽ làm A vàB không còn làm việc đúng nữa.

* 1. **Kiểm thử mức hệ thống** (System test)

Mục đích Kiểm thử mức hệ thống là kiểm tra thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không. Điểm khác nhau then chốt giữa kiểm thử tích hợp và kiểm thử hệ thống là kiểm thử hệ thống chú trọng các hành vi và lỗi trên toàn hệ thống, còn kiểm thử tích hợp chú trọng sự giao tiếp giữa các đơn vị hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện kiểm thử đơn vị và kiểm thử tích hợp để bảo đảm mọi đơn vị phần mềm và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện kiểm thử hệ thống. Kiểm thử hệ thống kiểm tra cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi “bế tắc” (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn kiểm thử hệ thống, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm thử để chấp nhận hoặc dùng thử (Alpha/Beta Test).

* 1. **Kiểm thử chấp nhận sản phẩm** (Acceptance test)

Thông thường, sau giai đoạn kiểm thử hệ thống là kiểm thử chấp nhận, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của kiểm thử chấp nhận là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng). Kiểm thử chấp nhận có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của kiểm thử hệ thống và kiểm thử chấp nhận gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

1. **Các kỹ thuật kiểm thử phần mềm**

Có thể chia các kỹ thuật kiểm thử phần mềm thành ba loại: các kỹ thuật kiểm thử hộp đen (Black-Box testing), kỹ thuật kiểm thử hộp trắng (White-Box testing) và kỹ thuật kiểm thử hộp xám (Grey-Box testing). Các kiểm thử hộp đen tìm các lỗi như thiếu các chức năng, khả năng sử dụng và các yêu cầu phi chức năng. Các kỹ thuật kiểm thử hộp trắng yêu cầu hiểu biết về cấu trúc chương trình bên trong và các kiểm thử nhận được từ đặc tả thiết kế bên trong hoặc từ mã. Trong khi đó kỹ thuật kiểm thử hộp xám là sự kết hợp của kiểm thử hộp đen và kiểm thử hộp trắng.

* 1. **Nguyên tắc cơ bản kiểm thử phần mềm**

Trong lúc kiểm thử, công nghệ phần mềm phát sinh một chuỗi các trường hợp kiểm thử được sử dụng để “tách từng phần” phần mềm. Kiểm thử là một bước trong quy trình phần mềm mà có thể được xem xét bởi đội ngũ phát triển bằng cách phá vỡ thay vì xây dựng. Các kỹ sư phần mềm chính là những người xây dựng và kiểm thử yêu cầu họ vượt qua các khái niệm cho trước về độ chính xác và giải quyết mâu thuẫn khi các lỗi được xác định.

* + 1. **Mục tiêu kiểm thử**

Các nguyên tắc được xem như mục tiêu kiểm thử là:

* Kiểm thử là một quá trình thực thi chương trình với mục đích tìm lỗi.
* Một trường hợp kiểm thử tốt là trường hợp kiểm thử mà có khả năng cao việc tìm thấy các lỗi chưa từng được phát hiện.
* Một kiểm thử thành công là kiểm thử mà phát hiện lỗi chưa từng được phát hiện.
  + 1. **Luồng thông tin kiểm thử**

Hai kiểu của đầu vào được truyền cho quá trình kiểm thử:

* Cấu hình phần mềm: gồm các đặc tả yêu cầu, đặc tả thiết kế, và mã nguồn.
* Cấu hình kiểm thử: gồm có kế hoạch kiểm thử, các thủ tục, trường hợp kiểm thử, và các công cụ kiểm thử.
  + 1. **Thiết kế trường hợp kiểm thử**

Thiết kế kiểm thử phần mềm có thể là một quá trình thu thập, phân tích và thực hiện yêu cầu. Mục tiêu của kiểm thử là phải thiết kế các trường hợp kiểm thử có khả năng cao nhất trong việc phát hiện nhiều lỗi nhất với thời gian và công sức tối thiểu. Như vậy, vấn đề quan trọng nhất trong kiểm thử phần mềm là thiết kế và tạo ra các trường hợp kiểm thử có hiệu quả

Kiểm thử phần mềm còn có các ràng buộc về thời gian, chi phí, v.v. Chìa khoá của kiểm thử là trả lời của câu hỏi: “Tập con của tất cả các trường hợp kiểm thử có thể có xác suất phát hiện lỗi cao nhất là gì?”. Việc nghiên cứu các phương pháp thiết kế trường hợp kiểm thử sẽ cung cấp câu trả lời cho câu hỏi này.

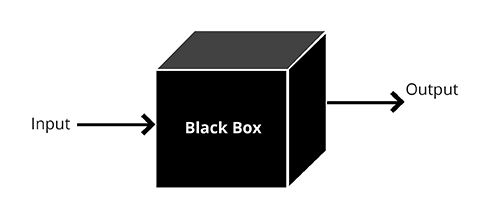
Bất kỳ sản phẩm công nghệ nào có thể được kiểm thử trong hai cách:

* Biết về các chức năng cụ thể mà sản phẩm đã được thiết kế để thực hiện.
* Biết cách hoạt động bên trong của sản phẩm, kiểm thử có thể được thực hiện để đảm bảo rằng “tất cả các thành phần ăn khớp nhau”.
* Cách tiếp cận kiểm thử đầu tiên được gọi là kiểm thử hộp đen và cách thứ hai là kiểm thử hộp trắng.
  1. **Kỹ thuật kiểm thử hộp đen** (Black-Box testing)
     1. **Khái niệm**

Kiểm thử hộp đen là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.

Phương pháp này được đặt tên như vậy bởi vì các chương trình phần mềm, trong con mắt của các tester, giống như một hộp đen; bên trong mà người ta không thể nhìn thấy. Phương pháp này cố gắng tìm ra các lỗi trong các loại sau:

* Chức năng không chính xác hoặc thiếu.
* Lỗi giao diện.
* Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.
* Hành vi hoặc hiệu suất lỗi.
* Khởi tạo và chấm dứt các lỗi.

****

*Hình 2.5. Minh họa kiểm tra hộp đen*

* + 1. **Ưu điểm và nhược điểm của kỹ thuật kiểm thử hộp đen**

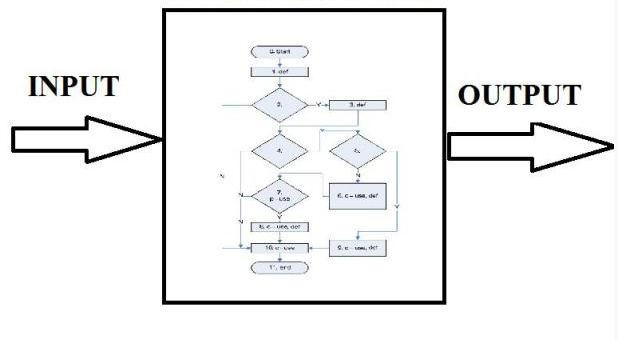
1. **Ưu điểm**

* Kỹ sư kiểm thử có thể không phải IT chuyên nghiệp.
* Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.
* Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

1. **Nhược điểm**

* Dữ liệu đầu vào yêu cầu một khối lượng mẫu khá lớn.
* Khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.
* Khả năng để bản thân kỹ sư lạc lối trong khi kiểm thử là khá cao.
  1. **Kỹ thuật kiểm thử hộp trắng** (White-Box testing)
     1. **Khái niệm**

Kiểm thử hộp trắng là kỹ thuật kiểm thử dựa trên đặc tả bên trong của chương trình, dựa vào mã nguồn, cấu trúc chương trình. Kiểm thử hộp trắng thường phát hiện các lỗi lập trình. Loại kiểm thử này khá khó thực hiện và chi phí cao.

****

*Hình 2.6. Minh họa kiểm tra hộp trắng*

Có 2 kỹ thuật kiểm tra hộp trắng phổ biến:

* Kiểm thử luồng dữ liệu.
* Kiểm thử luồng điều khiển.

**6.3.2. Ưu điểm và nhược điểm của kỹ thuật kiểm thử hộp trắng**

**a. Ưu điểm**

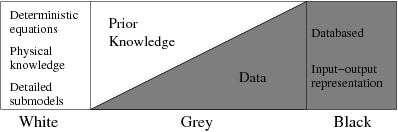
* Test có thể bắt đầu ở giai đoạn sớm hơn, không cần phải chờ đợi cho GUI để có thể test.
* Test kỹ càng hơn, có thể bao phủ hầu hết các đường dẫn.
* Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh.
* Cho phép tìm kiếm các lỗi ẩn bên trong.
* Các lập trình viên có thể tự kiểm tra.
* Giúp tối ưu việc mã hoá.
* Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.

**b. Nhược điểm**

* Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình và thực hiện.
* Giai đoạn bảo trì có thể là một gánh nặng nếu thể hiện thay đổi quá thường xuyên.
* Vì phương pháp thử nghiệm này liên quan chặt chẽ với ứng dụng đang được test, nên các công cụ để phục vụ cho mọi loại triển khai/nền tảng có thể không sẵn có.
  1. **Kỹ thuật kiểm thử hộp xám** (Grey-Box testing)

**6.4.1. Khái niệm**

Kiểm thử hộp xám là một phương pháp kiểm thử mà đòi hỏi tester phải có một lượng kiến thức nhất định về các luồng hoạt động ở bên trong hệ thống. Khác với kiểm thử hộp đen, phương pháp mà tester chỉ quan tâm duy nhất để việc kiểm thử thông qua giao diện người dùng, kiểm thử hộp xám đòi hỏi tester phải truy cập vào các tài liệu thiết kế hệ thống cũng như hệ thống cơ sở dữ liệu của hệ thống. Do đó mà tester có thể chuẩn bị tốt hơn những dữ liệu cho việc kiểm thử cũng như các trường hợp kiểm thử trong quá trình lên kế hoạch kiểm thử hệ thống.



*Hình 2.7. Minh họa kiểm tra hộp xám*

* + 1. **Ưu điểm và nhược điểm của kỹ thuật kiểm thử hộp trắng**
  1. **Ưu điểm**
* Vì là sự kết hợp giữa kiểm thử hộp trắng và kiểm thử hộp đen nên có được ưu điểm của cả hai phương pháp này.
* Các tester sử dụng phương pháp này không dựa vào các dòng lệnh của hệ thống mà chủ yếu dựa trên các tài liệu định nghĩa giao diện cũng như các tài liệu đặc tả chức năng.
* Trong phương pháp này các tester có thể thiết kế nên những trường hợp kiểm thử đặc biệt xung quanh các giao thức kết nối và các loại dữ liệu khác nhau.
* Việc kiểm thử được hoàn thành từ góc nhìn của người dùng chứ không phải từ nhà thiết kế.
  1. **Nhược điểm**

Vì phương pháp này không dựa trên việc truy cập code của hệ thống nên sẽ không tránh được việc độ bao phủ của các trường hợp kiểm thử bị giới hạn.

Khi sử dụng phương pháp này thì nhiều trường hợp kiểm thử có thể bị dư thừa nếu mà những nhà thiết kế phần mềm đã chạy các trường hợp kiểm thử này trước đó.

Việc kiểm tra tất cả các luồng đầu vào của hệ thống là không thể thực hiện được vì bị giới hạn về mặt thời gian kiểm thử và sẽ dẫn đến có rất nhiều các luồng hoạt động của hệ thống không được kiểm tra.

**7. Kỹ thuật thiết kế ca kiểm thử**

Quá trình phát triển ca kiểm thử có thể giúp tìm ra lỗi trong các yêu cầu hoặc thiết kế của ứng dụng, vì nó đòi hỏi phải tư duy hoàn toàn thông qua các hoạt động của ứng dụng. Vì lý do này, việc chuẩn bị ca kiểm thử sớm nhất có thể trong quy trình phát triển phần mềm là rất hữu ích. Các trường hợp kiểm thử phải bao phủ được toàn bộ luồng xử lý chức năng mô tả trong tài liệu phân tích và thiết kế; các yêu cầu về bảo mật an toàn thông tin, yêu cầu hiệu năng của hệ thống.

* 1. **Cấu trúc của ca kiểm thử**

*Test Case ID:* Giá trị cần để xác định số lượng trường hợp cần để kiểm thử.

*Testcase Description:* Mô tả sơ lược về mục đích của ca kiểm thử đó.

*PreRequisites:* Điều kiện tiền đề nếu có.

*Test Data:* Những dữ liệu đầu vào cần chuẩn bị để test.

*Step:* Các bước thực hiện 1 ca kiểm thử.

*Execution Step:* Mô tả các bước thực hiện kiểm thử.

*Expected results:* Kết quả mong đợi từ các bước thực hiện trên.

*Actual result:* Kết quả thực tế khi chạy chương trình.

*Result:* Đánh giá về kết quả, thông thường sẽ là pass, fail.

*Note:* Cột này dùng để ghi chú những thông tin liên quan khi thực hiện ca kiểm thử.

* Các bước xác định ca kiểm thử:

**Bước 1:** Xác định mục đích kiểm thử: cần hiểu rõ đặc tảyêu cầu củakhách hàng.

**Bước 2:** Xác định chức năng cần kiểm tra: cần phải biết làm thếnào phầnmềm được sử dụng bao gồm các hoạt động, tổ chức chức năng khác nhau. Các bước thực hiện chỉ mô tả các bước thực hiện đứng từ phía người dùng cuối bao gồm nhập dữ liệu, nhấn button, v.v.

**Bước 3:** Xác định các yêu cầu phi chức năng: yêu cầu phần cứng, hệ

điều hành, các khía cạnh an ninh.

**Bước 4:** Xác định biểu mẫu cho Ca kiểm thử: bao gồm giao diện UI,chức năng, khả năng tương thích và hiệu suất.

**Bước 5:** Xác định tínhảnh hưởng giữa các nguyên tắc mô-đun: mỗi mộtca kiểm thử nên được thiết kế để có thể che phủ được sự ảnh hưởng của các mô-đun với nhau ở mức độ cao nhất.

* 1. **Phân vùng tương đương**
     1. **Phương pháp**

Phân vùng tương đương là phương pháp chia các điều kiện đầu vào thành những vùng tương đương nhau. Tất cả các giá trị trong một vùng tương đương sẽ cho một kết quả đầu ra giống nhau. Vì vậy chúng ta có thể test một giá trị đại diện trong vùng tương đương.

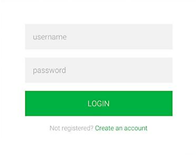
Mục đích: Giảm đáng kể số lượng ca kiểm thử cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.

Thiết kế ca kiểm thử bằng kỹ thuật phân vùng tương đương tiến hành theo 2 bước:

* Xác định các lớp tương đương: ta chia miền dữ liệu kiểm thử thành các miền con sao cho dữ liệu trong mỗi miền con có cùng tính chất đối với chương trình. Sau khi chia miền dữ liệu của chương trình thành các miền con tương đương, ta chỉ cần chọn một phần tử đại diện của mỗi miền con này làm bộ dữ liệu kiểm thử. Các miền con này chính là các lớp tương đương.
* Xây dựng các ca kiểm thử tương ứng với mỗi lớp tương đương.
  + 1. **Ví dụ**

Ví dụ về 1 Form đăng nhập bao gồm:

* Username
* Password

****

*Hình 2.8. Minh họa form đăng nhập*

Yêu cầu:

* Thiết kế ca kiểm thử sao cho người dùng nhập vào ô text-box username chỉ cho nhập ký tự chữ với độ dài trong khoảng [6-20].
* Nếu nhập giá trị với số ký tự không nằm trong khoảng [6-20] => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
* Nếu để trống ô hoặc nhập ký tự khác ký tự chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ”.

Dựa vào yêu cầu bài toán ta có thể có các lớp tương đương (phân vùng) sau:

* *Phân vùng 1:* Nhập giá trị hợp lệ từ 6 => 20 ký tự.
* *Phân vùng 2:* Nhập giá trị không hợp lệ < 6 ký tự.
* *Phân vùng 3:* Nhập giá trị không hợp lệ > 20 ký tự.
* *Phân vùng 4:* Trường hợp để trống không nhập gì hay nhập ký tự không phải dạng chữ.

Sau khi áp dụng phân vùng tương đương có thể chọn được các ca kiểm thử sau:

* *Case 1:* Nhập giá trị từ 6 => 20 => pass.
* *Case 2:* Nhập giá trị < 6 ký tự (có thể chọn nhập 1, 2, 3, 4 hoặc 5 ký tự) => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
* *Case 3:* Nhập giá trị > 20 ký tự (có thể chọn nhập 21, 22, 23… ký tự) => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
* *Case 4:* Để trống không nhập gì hay nhập ký tự không phải dạng chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ”.
  + 1. **Ưu điểm và nhược điểm**

1. **Ưu điểm:** Vì mỗi vùng tương đương ta chỉ cần kiểm tra trên các phần tử đại diện nên số lượng ca kiểm thử được giảm đi khá nhiều nhờ đó mà thời gian thực hiện kiểm thử cũng giảm đáng kể.
2. **Nhược điểm:** Không phải với bất kỳ bài toán nào đều có thể áp dụng kỹ thuật này. Có thể bị thiếu sót lỗi ở biên nếu chỉ chọn giá trị ở khoảng giữa của miền tương đương.
   1. **Phân vùng giá trị biên**

**7.3.1. Phương pháp**

Hầu hết các lỗi được tìm thấy khi kiểm tra ở các giá trị biên. Vì vậy phương pháp này tập trung vào việc kiểm thử các giá trị biên này.

Đây là phương pháp test mà chúng ta sẽ test tất cả các giá trị ở vùng biên của dữ liệu vào và dữ liệu ra. Chúng ta sẽ tập trung vào các giá trị biên chứ không test toàn bộ dữ liệu. Thay vì chọn nhiều giá trị trong lớp đương tương để làm đại diện, phân tích giá trị biên yêu cầu chọn một hoặc vài giá trị là các cạnh của lớp tương đương để làm điều kiện test.

Phân tích giá trị biên là trường hợp đặc biệt của phân vùng tương đương, dựa trên những phân vùng tương đương kiểm thử viên sẽ xác định giá trị biên giữa những phân vùng này và lựa chọn ca kiểm thử phù hợp. Mục tiêu là lựa chọn các ca kiểm thử để thực thi giá trị biên.

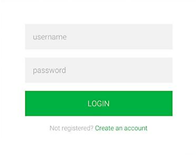
Các case chuẩn được lựa chọn dựa vào quy tắc sau:

* Giá trị biên nhỏ nhất – 1.
* Giá trị biên nhỏ nhất.
* Giá trị bất kì ở giữa.
* Giá trị biên lớn nhất.
* Giá trị biên lớn nhất + 1.

**7.3.2. Ví dụ**

Ví dụ về 1 Form đăng nhập bao gồm:

* Username
* Password

****

*Hình 2.9. Minh họa form đăng nhập*

Áp dụng kỹ thuật phân tích giá trị biên ta chọn được các case sau:

* *Case 1:* Nhập giá trị với 5 ký tự => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
* *Case 2:* Nhập giá trị với 6 ký tự => pass.
* *Case 3:* Nhập giá trị bất kì ở giữa => pass.
* *Case 4:* Nhập giá trị với 20 ký tự => pass.
* *Case 5:* Nhập giá trị với 21 ký tự => hiển thị lỗi “Bạn chỉ được phép nhập chuỗi từ 6 => 20 ký tự”.
* *Case 6:* Để trống không nhập gì hay nhập ký tự không phải dạng chữ => hiển thị lỗi “Tên người dùng chưa hợp lệ! Vui lòng nhập ký tự chữ”.
  + 1. **Ưu điểm và nhược điểm**

1. **Ưu điểm:** Thay vì phải kiểm tra hết toàn bộ các giá trị trong từng vùng tương đương, kỹ thuật phân tích giá trị biên tập trung vào việc kiểm thử các giá trị biên của miền giá trị đầu vào để thiết kế ca kiểm thử do “lỗi thường tiềm ẩn tại các ngõ ngách và tập hợp tại biên” nên sẽ tiết kiệm thời gian thiết kế ca kiểm thử và thực hiện kiểm thử.
2. **Nhược điểm:** Phương pháp phân tích giá trị biên chỉ hiệu quả trong trường hợp các đối số đầu vào độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn.
   1. **Đoán lỗi** 
      1. **Phương pháp**

Trong kiểm thử phần mềm, đoán lỗi là một phương pháp kiểm thử, trong đó các trường hợp kiểm thử được sử dụng để tìm lỗi trong các chương trình đã được phát triển dựa vào kinh nghiệm trong các lần kiểm thử trước. Phạm vi của các trường hợp kiểm thử thường được dựa vào các kiểm thử viên có kiến thức liên quan, là những người đã có kinh nghiệm sử dụng và trực giác để xác định những tình huống thường gây ra lỗi trong phần mềm. Các lỗi điển hình như chia cho không, null pointer, hoặc các biến không hợp lệ, v.v.

Phương pháp đoán lỗi không có quy tắc rõ ràng, ca kiểm thử có thể được thiết kế tùy thuộc vào tình hình, hoặc hoặc luồng công việc trong các tài liệu mô tả chức năng hoặc khi một lỗi không mong muốn / không được mô tả trong tài liệu được tìm thấy trong khi hoạt động kiểm thử. Phương pháp này chỉ phù hợp với những kiểm thử viên có kinh nghiệm. Khi một kiểm thử viên được đưa cho một chương trình, họ phỏng đoán dựa vào trực giác, dựa vào kinh nghiệm, dữ liệu lịch sử về các lỗi đã từng xảy ra với chương trình trước đó, v.v. và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

* + 1. **Ưu điểm và nhược điểm**

1. **Ưu điểm:** Sử dụng phương pháp đoán lỗi có thể giúp kiểm thử viên tìm ra những lỗi điển hình thường xảy ra trong phần mềm hoặc những lỗi không thể tìm thấy khi thiết kế ca kiểm thử theo hình thức thông thường.
2. **Nhược điểm:** Phương pháp đoán lỗi thường được thực hiện bởi các kiểm thử viên có kinh nghiệm và không theo một quy tắc nhất định, thiết kế ca kiểm thử dựa nhiều vào cảm tính.

**8. Tạo report Bug**

Report Bug là một phần rất quan trọng và không thể thiếu trong quy trình thực hiện kiểm thử. Khi phần mềm xảy ra lỗi, kiểm thử viên phải tạo được ra các report Bug và gửi cho nhà phát triển phần mềm đó. Một report Bug được viết rõ ràng và rành mạch, sẽ luôn gây ấn tượng và hiệu ứng tốt hơn đối với một report Bug sơ xài và cẩu thả. Làm cho người sửa bug đó và cả người xác nhận lại bug đó không có cảm giác khó chịu khi phải đọc một report Bug sơ xài.

* 1. **Bug và report Bug**

*Bug:* Bug của phần mềm là những sai lầm, hỏng hóc, lỗi, khiếm khuyết để tạo ra một kết quả sai, hoặc không lường đến được, có thể coi nó như một thứ gì đó không hoạt động đúng theo thiết kế.

*Report Bug:* Văn bản chứa đầy đủ các thông tin về một lỗi của một sản phẩm được kiểm thử viên gửi cho một tổ chức hay cá nhân liên quan để sửa được gọi là report Bug.

* 1. **Cấu trúc một Bug report**

*Project:* Tên của dự án phần mềm.

*Reported by:* Kiểm thử viên tạo ra report Bug.

*Bug Name, Bug ID và Date:* Tên của bug, ID và ngày tạo report.

*Assigned to:* Cá nhân hoặc tổ chức phát triển phần mềm đó.

*Status:* Trạng thái thực hiện của report.

*Summary/Description*: Mô tả ngắn gọn về bug.

*Environments (OS/Browser):* Môi trường chạy thử phần mềm.

*Step to reproduce:* Mô tả lại các bước thực hiện gây ra bug.

*Actual results:* Kết quả thực tế.

*Expected results:* Kết quả mong đợi.

*Severity*: Mức độ nghiêm trọng của bug.

*Priority*: Mức độ ưu tiên của bug.

*Attachment:* Đính kèm với bug (tệp, đường dẫn URL,ảnh, v.v.).

* Một số yêu cầu khi tạo Bug report:
* Tiêu đề phải rõ rang: Khi lập trình viên đọc bug, thứ đầu tiên đập vào mắt là Bug name. Nó cũng là phần được đọc nhiều nhất, không phải là description. Một bug name tốt phải ngắn gọn và diễn tả được bug một cách tối giản.
* Phải mô phỏng lại được quá trình gây ra bug: Nếu không mô phỏng lại được, bug sẽ không thể được khắc phục.
* Không viết luận trong description: Viết ngắn gọn và vào trọng tâm. Cố gắng viết ít chữ nhất có thể nhưng vẫn đầy đủ ý.
  1. **Severity và Priority**

Có hai phần quan trọng trong những bug report đó là:

* Severity- Mức độnghiêm trọng.
* Priority- Mức độ ưu tiên.

Mặc dù hai yếu tố này không phải là yếu tố sống còn trong quản lý bug. Tuy nhiên, việc hiểu đúng về mức độ nghiêm trọng, độ ưu tiên của sản phẩm cho thấy chúng ta thực sự hiểu rõ và quan tâm đến chất lượng sản phẩm cũng như thể hiện sự chuyên nghiệp của một kỹ sư kiểm thử.

* + 1. **Mức độ nghiêm trọng** (Severity)

Severity là mức độ mà các bug có thể ảnh hưởng đến các phần mềm. Nói cách khác, nó xác định các tác động mà một bug nhất định có trên hệ thống.

Severity có thể được phân thành các loại sau đây:

* *Critical (S1):* Bug ảnh hưởng đến các chức năng quan trọng hoặc dữ liệu quan trọng. Nó không có giải pháp để thay thế. Ví dụ: Cài đặt không thành công, hoàn thành sự thất bại của một tính năng, v.v.
* *Major (S2):* Thiệt hại ảnh hưởng đến chức năng chính hoặc dữ liệu chính. Nó có giải pháp để thay thế nhưng không rõ ràng hoặc khó khăn. Ví dụ: Một tính năng không thể thực thi trực tiếp nhưng là khả thi nếu có 10 bước gián tiếp phức tạp được được thực hiện để có được kết quả như mong muốn.
* *Minor (S3):* Bug ảnh hưởng đến chức năng nhỏ hoặc dữ liệu không quan trọng. Nó có một giải pháp thay thế dễ dàng. Ví dụ: Một tính năng nhỏ không được thực thi nhưng nhiệm vụ tương tự có thể dễ dàng thực hiện từ một chức năng khác.
* *Trivial (S4):* Bug không ảnh hưởng đến chức năng hoặc dữ liệu. Nó thậm chí không cần một giải pháp để thay thế do không ảnh hưởng đến năng suất hoặc hiệu quả mà chỉ là sự bất tiện. Ví dụ: Sai lệch bố cục nhỏ, lỗi chính tả / lỗi ngữ pháp, v.v.
  + 1. **Mức độ ưu tiên** (Priority)

Priority xác định thứ tự mà chúng ta nên giải quyết một bug. Chúng ta nên sửa nó ngay bây giờ, hoặc nó có thể được hoãn lại cho đến khi một bug nghiêm trọng khác đã được giải quyết. Tình trạng ưu tiên này được thiết lập bởi các kiểm thử viên cho nhà phát triển đề cập đến các khung thời gian để sửa chữa những bug. Mức độ ưu tiên càng cao thì phải sửa chữa nó trong thời gian càng sớm. Tình trạng ưu tiên được thiết lập dựa trên các yêu cầu của khách hàng.

Priority có thể phân thành các loại sau đây:

* *Urgent (P0):* Phải được sửa càng sớm càng tốt.
* *High (P1):* Phải được sửa trong một vài phiên bản tiếp theo.
* *Medium (P2):* Nên được sửa ở những phiên bản tiếp theo.
* *Low (P3):* Có thể được sửa ở một phiên bản nào đó.

**CHƯƠNG III. ÁP DỤNG**

Từ những phương pháp thiết kế Testcase em đã tìm hiểu ở Chương II, em áp dụng để xây dựng Testcase cho một bài toán cụ thể sau: “Viết Testcase cho màn hình đăng ký Gmail”.

* + - 1. **Giới thiệu về màn hình tạo tài khoản Google**

Sau đây là một số quan điểm test một form đăng ký tài khoản cũng như là cách viết testcase cho một form đăng ký bất kỳ trong quá trình thực hiện test để bao quát được tất cả các trường hợp xảy ra.

Form đăng ký bao gồm các trường sau:

Họ (First name):

* Nhập vào các chữ, số có độ dài từ 1 đến 45 ký tự (Dùng phương pháp giá trị biên).
* Nhập chữ có dấu và chữ in hoa.
* Nhập kết hợp chữ, số và ký tự đặc biệt.
* Nhập khoảng trống trước và sau ký tự.
* Không nhập bất cứ ký tự nào.
* Nhập khoảng trống.
* Nhập trên 45 ký tự.
* Nhập ký tự đặc biệt.

Tên (Last name):

* Nhập vào các chữ, số có độ dài từ 1 đến 44 ký tự (Dùng phương pháp giá trị biên).
* Nhập chữ có dấu và chữ in hoa.
* Nhập kết hợp chữ, số.
* Nhập khoản trống trước và sau ký tự.
* Không nhập bất cứ ký tự nào.
* Nhập khoảng trống
* Nhập trên 44 ký tự.
* Nhập kết hợp số, chữ và ký tự đặc biệt.

Tên người dùng (Username):

* Nhập vào các chữ từ 6 đến 30 ký tự (Dùng phương pháp giá trị biên).
* Nhập kết hợp chữ, số và dấu chấm.
* Nhập khoảng trống trước và sau ký tự.
* Không nhập bất cứ ký tự nào.
* Nhập khoảng trống.
* Nhập dưới 6 ký tự hoặc trên 30 ký tự.
* Nhập ký tự đặc biệt hoặc số.
* Nhập chữ có dấu.
* Nhập trùng tài khoản đã đăng ký.
* Nhập không đúng dạng gmail. Ví dụ: @ffgfg, df@sftt, trt@@,...

Mật khẩu (Password):

* Nhập trên 8 ký tự (Dùng phương pháp giá trị biên).
* Nhập giá trị vào thì có được mã hóa thành dấu \*\*\*\* hay không.
* Không nhập bất cứ ký tự nào.
* Nhập khoảng trống.
* Nhập dưới 8 ký tự.

Xác nhận lại mật khẩu (Confirm password):

* Nhập trùng với mật khẩu (Nếu mật khẩu đúng).
* Nhập lại mật khẩu chữ thường thành chữ hoa.
* Nhập lại mật khẩu nhưng có khoảng trống.
* Nhập lại mật khẩu nhưng thiếu ký tự.
* Nhập sai mật khẩu.

Nút “Tiếp theo” (Next):

* Nhập đúng tất cae các trường.
* Không nhập thông tin vào các trường.
* Nhập thông tin vào 1 trường còn lại để trống
* Nhập sai yêu cầu trong các trường.
  + - 1. **Kết quả**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC-ID | Test Scenario | Pre-condition | Test step | Step condition to perform | Data | Priority | Expected result |
|
|
| Gmail\_Screen\_001 | Create Account Successful by First Name | Stay at Create Account Screen | 1. Input value into First Name 2. Input valid value into rest field 3. Click Next button | 1..1 Input 1-45 randomly Chars or Number | dfbgfhgfhgfh or 123456454 | High | Go to the new page |
| Gmail\_Screen\_002 | 1.2 Input space before and after Chars | grgxscfdfr | Medium | The Space is trimmed   Move on the next page |
| Gmail\_Screen\_003 | 1.3 Input mix Chars + Number+ Special Character |  | High | Go to the new page |
| Gmail\_Screen\_004 | Create Account Unsuccessful by First Name | Stay at Create Account Screen | 1. Input value into First Name 2. Input valid value into rest field 3. Click Next button | 1.1 Don't Input Anything |  | Low | Obligatory you must input first name then go to the new page |
| Gmail\_Screen\_005 | 1.2 Input Space |  | Low | Obligatory you must input first name then go to the new page |
| Gmail\_Screen\_006 | 1.3 Input on 45 Chars |  | Medium | Sure you must input first name then go to the new page |
| Gmail\_Screen\_007 | 1.4 Input Special Character | <>:()\_+="**@**#$% | Medium | Sure you must input first name then go to the new page |