Kiểm thử viên được Chứng nhận

Đề cương Cấp độ Nền tảng

v4.0.1

Hội đồng Thẩm định Kiểm thử Phần mềm Quốc tế

Thông báo Bản quyền

Lịch sử Sửa đổi

Phiên bản	Ngày	Ghi chú
CTFL v4.0.1	15.09.2024	CTFL v4.0.1 – Bản đính chính
CTFL v4.0	21.04.2023	CTFL v4.0 – Phiên bản phát hành chính thức
CTFL v3.1.1	01.07.2021	CTFL v3.1.1 – Cập nhật thông tin bản quyền và logo
CTFL v3.1	11.11.2019	CTFL v3.1 – Bản cập nhật duy trì với các thay đổi nhỏ
ISTQB 2018	27.04.2018	CTFL v3.0 – Phiên bản ứng viên cho phát hành chính thức
ISTQB 2011	01.04.2011	Bản Phát hành Cập nhật Đề cương CTFL
ISTQB 2010	30.03.2010	Bản Phát hành Cập nhật Đề cương CTFL
ISTQB 2007	01.05.2007	Bản Phát hành Cập nhật Đề cương CTFL
ISTQB 2005	01.07.2005	Đề cương Chứng chỉ Kiểm thử viên Cấp độ Nền tảng, phiên bản v1.0
ASQF V2.2	07.2003	Đề cương ASQF Cấp độ Nền tảng Phiên bản v2.2 “Chương trình Giảng dạy các Nguyên lý Cơ bản của Kiểm thử Phần mềm”
ISEB V2.0	25.02.1999	Đề cương ISEB Kiểm thử Phần mềm Cấp độ Nền tảng phiên bản v2.0

Mục lục

Lời cảm ơn

Tài liệu này được chính thức phát hành bởi Chủ sở hữu Sản phẩm (Product Owner) / trưởng nhóm công tác Eric Riou du Cosquer vào ngày 15.09.2024.

Tài liệu này được xây dựng bởi nhóm đến từ các Nhóm Công tác chung giữa ISTQB Cấp độ Nền tảng và Agile: Renzo Cerquozzi (phó trưởng nhóm), Wim Decoutere, Jean-François Riverin, Arnika Hryszko, Martin Klonk, Meile Posthuma, Eric Riou du Cosquer (trưởng nhóm), Adam Roman, Lucjan Stapp, Stephanie Ulrich (phó trưởng nhóm), Eshraka Zakaria.

Phiên bản 4.0 của tài liệu này đã được Đại Hội đồng ISTQB® chính thức phát hành vào ngày 21 tháng 4 năm 2023.

Tài liệu này được xây dựng bởi nhóm đến từ các Nhóm Công tác chung giữa ISTQB Cấp độ Nền tảng và Agile: Laura Albert, Renzo Cerquozzi (phó trưởng nhóm), Wim Decoutere, Klaudia Dussa-Zieger, Chintaka Indikadahena, Arnika Hryszko, Martin Klonk, Kenji Onishi, Michaël Pilaeten (đồng trưởng nhóm), Meile Posthuma, Gandhinee Rajkomar, Stuart Reid, Eric Riou du Cosquer (đồng trưởng nhóm), Jean-François Riverin, Adam Roman, Lucjan Stapp, Stephanie Ulrich (phó trưởng nhóm), Eshraka Zakaria.

Nhóm tác giả trân trọng cảm ơn Stuart Reid, Patricia McQuaid và Leanne Howard vì đã thực hiện rà soát kỹ thuật, cùng các nhóm rà soát và các Hội đồng Thành viên vì những góp ý và đóng góp của họ.

Những người sau đây đã tham gia việc rà soát, góp ý, và bỏ phiếu thông qua đề cương này: Adam Roman, Adam Scierski, Ágota Horváth, Ainsley Rood, Ale Rebon Portillo, Alessandro Collino, Alexander Alexandrov, Amanda Logue, Ana Ochoa, André Baumann, André Verschelling, Andreas Spillner, Anna Miazek, Armin Born, Arnd Pehl, Arne Becher, Attila Gyúri, Attila Kovács, Beata Karpinska, Benjamin Timmermans, Blair Mo, Carsten Weise, Chinthaka Indikadahena, Chris Van Bael, Ciaran O'Leary, Claude Zhang, Cristina Sobrero, Dandan Zheng, Dani Almog, Daniel Säther, Daniel van der Zwan, Danilo Magli, Darvay Tamás Béla, Dawn Haynes, Dena Pauletti, Dénes Medzihradszky, Doris Dötzer, Dot Graham, Edward Weller, Erhardt Wunderlich, Eric Riou Du Cosquer, Florian Fieber, Fran O'Hara, François Vaillancourt, Frans Dijkman, Gabriele Haller, Gary Mogyorodi, Georg Sehl, Géza Bujdosó, Giancarlo Tomasig, Giorgio Pisani, Gustavo Márquez Sosa, Helmut Pichler, Hongbao Zhai, Horst Pohlmann, Ignacio Trejos, Ilia Kulakov, Ine Lutterman, Ingvar Nordström, Iosif Itkin, Jamie Mitchell, Jan Giesen, Jean-Francois Riverin, Joanna Kazun, Joanne Tremblay, Joëlle Genois, Johan Klintin, John Kurowski, Jörn Münzel, Judy McKay, Jürgen Beniermann, Karol Frühauf, Katalin Balla, Kevin Kooh, Klaudia Dussa-Zieger, Klaus Erlenbach, Klaus Olsen, Krisztián Miskó, Laura Albert, Liang Ren, Lijuan Wang, Lloyd Roden, Lucjan Stapp, Mahmoud Khalaili, Marek Majernik, Maria Clara Choucair, Mark Rutz, Markus Niehammer, Martin Klonk, Márton Siska, Matthew Gregg, Matthias Hamburg, Mattijs Kemmink, Maud Schlich, May Abu-Sbeit, Meile Posthuma, Mette Bruhn-Pedersen, Michal Tal, Michel Boies, Mike Smith, Miroslav Renda, Mohsen Ekssir, Monika Stocklein Olsen, Murian Song, Nicola De Rosa, Nikita Kalyani, Nishan Portoyan, Nitzan Goldenberg, Ole Chr. Hansen, Patricia McQuaid, Patricia Osorio, Paul Weymouth, Pawel Kwasik, Peter Zimmerer, Petr Neugebauer, Piet de Roo, Radoslaw Smilgin, Ralf Bongard, Ralf Reissing, Randall Rice, Rik Marselis, Rogier Ammerlaan, Sabine Gschwandtner, Sabine Uhde, Salinda Wickramasinghe, Salvatore Reale, Sammy Kolluru, Samuel Ouko, Stephanie Ulrich, Stuart Reid, Surabhi Bellani, Szilard Szell, Tamás Gergely, Tamás Horváth, Tatiana Sergeeva, Tauhida Parveen, Thaer Mustafa, Thomas Eisbrenner, Thomas Harms, Thomas Heller, Tobias Letzkus, Tomas Rosenqvist, Werner Lieblang, Yaron Tsubery, Zhenlei Zuo and Zsolt Hargitai.

Nhóm công tác ISTQB Cấp độ Nền tảng (Phiên bản 2018): Klaus Olsen (trưởng nhóm), Tauhida Parveen (phó trưởng nhóm), Rex Black (quản lý dự án), Eshraka Zakaria, Debra Friedenberg, Ebbe Munk, Hans Schaefer, Judy McKay, Marie Walsh, Meile Posthuma, Mike Smith, Radoslaw Smilgin, Stephanie Ulrich,

Steve Toms, Corne Kruger, Dani Almog, Eric Riou du Cosquer, Igal Levi, Johan Klintin, Kenji Onishi, Rashed Karim, Stevan Zivanovic, Sunny Kwon, Thomas Müller, Vipul Kocher, Yaron Tsubery, và tất cả các Hội đồng Thành viên vì những đề xuất của họ.

Nhóm công tác ISTQB Cấp độ Nền tảng (Phiên bản 2011): Thomas Müller (trưởng nhóm), Debra Friedenberg. Nhóm nòng cốt cảm ơn nhóm rà soát (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquer, Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal), và tất cả các Hội đồng Thành viên vì những đề xuất cho phiên bản hiện tại của đề cương này.

Nhóm công tác ISTQB Cấp độ Nền tảng (Phiên bản 2010): Thomas Müller (trưởng nhóm), Rahul Verma, Martin Klonk và Armin Beer. Nhóm nòng cốt cảm ơn nhóm rà soát (Rex Black, Mette Bruhn-Pedersen, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veenendaal), và tất cả các Hội đồng Thành viên vì những đề xuất của họ.

Nhóm công tác ISTQB Cấp độ Nền tảng (Phiên bản 2007): Thomas Müller (trưởng nhóm), Dorothy Graham, Debra Friedenberg, và Erik van Veenendaal. Nhóm nòng cốt cảm ơn nhóm rà soát (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson và Wonil Kwon), và tất cả các Hội đồng Thành viên vì những đề xuất của họ.

Nhóm công tác ISTQB Cấp độ Nền tảng (Phiên bản 2005): Thomas Müller (trưởng nhóm), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson, và Erik van Veenendaal. Nhóm nòng cốt cảm ơn nhóm rà soát và tất cả các Hội đồng Thành viên vì những đề xuất của họ

0. Giới thiệu

0.1. Mục đích của Đề cương này

Đề cương này là cơ sở cho Chứng chỉ Kiểm thử Phần mềm Quốc tế ở Cấp độ Nền tảng.

Tổ chức ISTQB® cung cấp đề cương này như sau:

1. Cho các hội đồng thành viên, để dịch sang ngôn ngữ địa phương và công nhận các nhà cung cấp đào tạo. Các hội đồng thành viên có thể điều chỉnh đề cương theo nhu cầu ngôn ngữ cụ thể của mình và sửa đổi các tài liệu tham chiếu cho phù hợp với các ấn phẩm tại địa phương.
2. Cho các tổ chức cấp chứng chỉ, để xây dựng các câu hỏi thi bằng ngôn ngữ địa phương, phù hợp với các mục tiêu học tập của đề cương này.
3. Cho các nhà cung cấp đào tạo, để xây dựng tài liệu giảng dạy và xác định phương pháp giảng dạy phù hợp.
4. Cho các ứng viên muốn thi chứng chỉ này, để chuẩn bị cho kỳ thi chứng chỉ (dù thông qua khóa đào tạo hoặc tự học).

Phục vụ cộng đồng quốc tế về kỹ thuật phần mềm và hệ thống, góp phần thúc đẩy sự phát triển của nghề kiểm thử phần mềm và hệ thống, đồng thời làm cơ sở cho các sách và bài viết.

0.2. Tester được Chứng nhận ở Cấp độ Nền tảng về Kiểm thử Phần mềm

Chứng chỉ Cấp độ Nền tảng hướng đến bất kỳ ai tham gia vào kiểm thử phần mềm. Điều này bao gồm những người ở các vai trò như kiểm thử viên (tester), chuyên viên phân tích kiểm thử, kỹ sư kiểm thử, chuyên viên tư vấn kiểm thử, quản lý kiểm thử, lập trình viên phần mềm, và các thành viên trong nhóm phát triển. Chứng chỉ Cấp độ Nền tảng này cũng phù hợp với bất kỳ ai muốn trang bị hiểu biết cơ bản về kiểm thử phần mềm, chẳng hạn như quản lý dự án (PM), quản lý chất lượng (QAM), chủ sở hữu sản phẩm (PO), quản lý phát triển phần mềm, chuyên viên phân tích nghiệp vụ (BA), giám đốc khối CNTT, và các chuyên gia tư vấn quản lý. Những người sở hữu Chứng chỉ Cấp độ Nền tảng sẽ có thể tiếp tục theo học các chứng chỉ kiểm thử phần mềm ở cấp độ cao hơn.

0.3. Lộ trình Nghề nghiệp cho Tester

Chương trình ISTQB® cung cấp hỗ trợ cho các chuyên gia kiểm thử ở mọi giai đoạn trong sự nghiệp của họ, mang lại cả chiều rộng và chiều sâu về kiến thức. Những cá nhân đã đạt chứng chỉ ISTQB® Cấp độ Nền tảng có thể quan tâm đến các Cấp độ Nâng cao Cốt lõi (TA – Chuyên viên Phân tích Kiểm thử, TTA – Chuyên viên Kiểm thử Kỹ thuật, và TM – Quản lý Kiểm thử), và sau đó là Cấp độ Chuyên gia (Quản lý Kiểm thử hoặc Cải tiến Quy trình Kiểm thử).

Bất kỳ ai mong muốn phát triển kỹ năng về thực hành kiểm thử trong phát triển phần mềm Agile có thể cân nhắc các chứng chỉ Agile Technical Tester hoặc Agile Test Leadership at Scale.

Nhánh Specialist (Chuyên môn hoá) cung cấp sự đào sâu vào các lĩnh vực có phương pháp và hoạt động kiểm thử chuyên biệt (ví dụ: kiểm thử tự động, kiểm thử AI, kiểm thử dựa trên mô hình, kiểm thử ứng dụng di động), các lĩnh vực liên quan đến các loại kiểm thử cụ thể (ví dụ: kiểm thử hiệu năng, kiểm thử khả dụng, kiểm thử chấp nhận, kiểm thử bảo mật), hoặc nhóm kiểm thử theo kiến thức thực tiễn cho từng lĩnh vực ngành nghề nhất định (ví dụ: ô tô hoặc trò chơi điện tử).

Vui lòng truy cập www.istqb.org để có thông tin mới nhất về Chương trình Chứng chỉ Tester của ISTQB.

0.4. Giá trị Mang lại cho Doanh nghiệp

Phần này liệt kê 14 Giá trị mang lại cho doanh nghiệp mà một người đạt chứng chỉ Cấp độ Nền tảng được kỳ vọng có thể đạt được.

Một Tester được Chứng nhận Cấp độ Nền tảng có thể…

FL-BO1 Hiểu kiểm thử là gì và tại sao kiểm thử có ích

FL-BO2 Hiểu các khái niệm cơ bản về kiểm thử phần mềm

FL-BO3 Xác định cách tiếp cận kiểm thử và các hoạt động cần triển khai tùy theo bối cảnh kiểm thử

FL-BO4 Đánh giá và cải thiện chất lượng tài liệu

FL-BO5 Tăng hiệu quả và hiệu suất của các hoạt động kiểm thử

FL-BO6 Tuỳ chỉnh quy trình kiểm thử phù hợp với vòng đời phát triển phần mềm

FL-BO7 Hiểu các nguyên tắc quản lý kiểm thử

FL-BO8 Viết và truyền đạt các báo cáo lỗi rõ ràng, dễ hiểu

FL-BO9 Hiểu các yếu tố ảnh hưởng đến mức độ ưu tiên và công sức liên quan đến kiểm thử

FL-BO10 Làm việc như một phần của nhóm đa chức năng

FL-BO11 Biết các rủi ro và lợi ích liên quan đến tự động hóa kiểm thử

FL-BO12 Xác định các kỹ năng thiết yếu cần cho kiểm thử

FL-BO13 Hiểu tác động của rủi ro đối với kiểm thử

FL-BO14 Báo cáo hiệu quả về tiến độ kiểm thử và chất lượng

0.5. Mục tiêu Học tập Có thể Thi và Mức độ Nhận thức Kiến thức

Mục tiêu học tập hỗ trợ các giá trị mang lại cho doanh nghiệp và được sử dụng để xây dựng các kỳ thi Chứng chỉ Tester Cấp độ Nền tảng. Nhìn chung, toàn bộ nội dung của các chương 1–6 trong đề cương này đều có thể được đưa vào đề thi ở mức K1. Nghĩa là thí sinh có thể được yêu cầu nhận biết, ghi nhớ, hoặc nhớ lại một từ khóa hoặc khái niệm được đề cập trong bất kỳ chương nào trong sáu chương này. Các mức mục tiêu học tập cụ thể được trình bày ở phần đầu mỗi chương và được phân loại như sau:

- K1: Ghi nhớ
- K2: Hiểu
- K3: Áp dụng

Thông tin chi tiết hơn và các ví dụ về mục tiêu học tập được trình bày trong Phụ lục A. Tất cả các thuật ngữ được liệt kê là từ khóa ngay dưới tiêu đề các chương phải được ghi nhớ (K1), ngay cả khi không được đề cập một cách rõ ràng trong các mục tiêu học tập.

0.6. Kỳ thi Chứng chỉ Cấp độ Nền tảng

Kỳ thi Chứng chỉ Cấp độ Nền tảng được xây dựng dựa trên đề cương này. Các câu trả lời cho câu hỏi thi có thể yêu cầu sử dụng nội dung dựa trên nhiều hơn một phần trong đề cương này. Tất cả các phần của đề cương đều có thể được đưa vào đề thi, ngoại trừ phần Giới thiệu và các Phụ lục. Các tiêu chuẩn và sách được đưa vào phần tài liệu tham khảo (Chương 7), nhưng nội dung của chúng không nằm trong phạm vi bài thi, ngoại trừ những phần đã được tóm tắt trong đề cương từ các tiêu chuẩn và sách đó. Tham khảo các tài liệu "Cấu trúc và quy định kỳ thi" và "Bảng cấu trúc kỳ thi".

0.7. Công nhận

Hội đồng Thành viên ISTQB® có thể công nhận các nhà cung cấp đào tạo có tài liệu khóa học tuân theo đề cương này. Các nhà cung cấp đào tạo nên lấy hướng dẫn công nhận từ Hội đồng Thành viên hoặc tổ chức thực hiện việc công nhận. Một khóa học được công nhận được xem là phù hợp với đề cương này và được phép bao gồm kỳ thi ISTQB® như một phần của khóa học. Các hướng dẫn công nhận cho đề cương này tuân theo Hướng dẫn Công nhận chung do Nhóm công tác Quản lý Quy trình và Tuân thủ ban hành.

0.8. Áp dụng các Tiêu chuẩn

Có các tiêu chuẩn được tham chiếu trong đề cương Cấp độ Nền tảng này (ví dụ: các tiêu chuẩn IEEE hoặc ISO). Các tham chiếu này cung cấp một nền tảng tham chiếu (như trong các tham chiếu đến ISO 25010 liên quan đến các đặc tính chất lượng) hoặc cung cấp nguồn thông tin bổ sung nếu người đọc mong muốn. Các tài liệu tiêu chuẩn không nhằm mục đích đưa vào kỳ thi. Tham khảo chương 7 để biết thêm thông tin về các tiêu chuẩn.

0.9. Cập nhật Thông tin

Ngành công nghệ phần mềm thay đổi nhanh chóng. Để thích ứng với những thay đổi này và cung cấp cho các bên liên quan quyền truy cập đến thông tin phù hợp và cập nhật nhất, các nhóm công tác ISTQB đã tạo các liên kết trên trang web www.istqb.org, dẫn đến các tài liệu hỗ trợ và các thay đổi của tiêu chuẩn. Thông tin này không nằm trong phạm vi thi của đề cương Cấp độ Nền tảng.

0.10. Mức độ Chi tiết

Mức độ chi tiết trong đề cương này cho phép các khóa đào tạo và kỳ thi có tính nhất quán trên phạm vi toàn cầu. Để đạt được mục tiêu này, đề cương bao gồm:

- Các mục tiêu giảng dạy tổng quát, mô tả mục đích của Cấp độ Nền tảng
- Danh sách các thuật ngữ (từ khóa) mà học viên phải ghi nhớ
- Các mục tiêu học tập cho từng lĩnh vực kiến thức, mô tả các kết quả học tập về mặt nhận thức cần đạt được
- Mô tả các khái niệm chính, bao gồm tham chiếu đến các nguồn được công nhận

Nội dung của đề cương này không phải là tài liệu mô tả toàn bộ lĩnh vực kiến thức kiểm thử phần mềm; nó phản ánh mức độ chi tiết cần được bao phủ trong các khóa đào tạo Cấp độ Nền tảng. Đề cương tập trung vào các khái niệm và kỹ thuật kiểm thử có thể áp dụng được cho mọi dự án phần mềm, không phụ thuộc vào vòng đời phát triển phần mềm (SDLC) đang được sử dụng.

0.11. Cấu trúc của Đề cương này

Đề cương này có sáu chương với nội dung có thể được đưa vào đề thi. Tiêu đề cấp cao nhất của mỗi chương chỉ rõ thời lượng đào tạo cho chương đó. Thời lượng không được cung cấp cho các cấp dưới của chương. Đối với các khóa đào tạo được công nhận, đề cương yêu cầu tối thiểu 1,135 phút (18 giờ 55 phút) giảng dạy, được phân bổ cho sáu chương như sau:

- Chương 1: Nền tảng Kiểm thử (180 phút)
  - Học viên tìm hiểu các nguyên tắc cơ bản liên quan đến kiểm thử, lý do tại sao cần kiểm thử, và các mục tiêu của kiểm thử là gì.
  - Học viên hiểu quy trình kiểm thử, các hoạt động kiểm thử chính, và sản phẩm kiểm thử.
  - Học viên hiểu các kỹ năng thiết yếu của kiểm thử.
- Chương 2: Kiểm thử Trong suốt Vòng đời Phát triển Phần mềm (130 phút)
  - Học viên tìm hiểu cách kiểm thử được tích hợp vào các phương pháp tiếp cận phát triển khác nhau.
  - Học viên tìm hiểu các khái niệm về phương pháp tiếp cận kiểm thử trước, cũng như DevOps.
  - Học viên tìm hiểu về các mức kiểm thử khác nhau, các loại kiểm thử, và kiểm thử bảo trì.
- Chương 3: Kiểm thử tĩnh (80 phút)
  - Học viên tìm hiểu các kiến thức cơ bản về kiểm thử tĩnh, phản hồi và quy trình rà soát.
- Chương 4: Phân tích và Thiết kế Kiểm thử (390 phút)
  - Học viên tìm hiểu cách áp dụng các kỹ thuật kiểm thử hộp đen, hộp trắng, và dựa trên kinh nghiệm để thiết kế test case dựa vào các sản phẩm công việc phần mềm khác nhau.
  - Học viên tìm hiểu về cách tiếp cận kiểm thử dựa trên sự cộng tác
- Chương 5: Quản lý Các Hoạt động Kiểm thử (335 phút)
  - Học viên tìm hiểu cách lập kế hoạch kiểm thử nói chung, và cách ước lượng công sức kiểm thử.
  - Học viên tìm hiểu cách rủi ro có thể ảnh hưởng đến phạm vi kiểm thử.
  - Học viên tìm hiểu cách giám sát và kiểm soát các hoạt động kiểm thử.
  - Học viên tìm hiểu cách quản lý cấu hình hỗ trợ kiểm thử.
  - Học viên tìm hiểu cách báo cáo lỗi một cách rõ ràng và dễ hiểu.
- Chương 6: Công cụ Kiểm thử (20 phút)
  - Học viên tìm hiểu cách phân loại các công cụ và hiểu các rủi ro cũng như lợi ích của việc tự động hóa kiểm thử.

1. Nền tảng Kiểm thử – 180 phút

Từ khóa

chất lượng, cơ sở kiểm thử, dữ liệu kiểm thử, đảm bảo chất lượng, độ bao phủ, đối tượng kiểm thử, giám sát kiểm thử, gỡ lỗi (debugging), hoàn tất kiểm thử, khả năng truy vết, kết quả kiểm thử, khía cạnh kiểm thử, kiểm soát kiểm thử, kiểm thử, lập kế hoạch kiểm thử, lỗi (defect), mục tiêu kiểm thử, nguyên nhân gốc rễ, phân tích kiểm thử, quy trình kiểm thử, sai sót, sản phẩm kiểm thử, sự cố (failure), thiết kế test case, thẩm định (validation), thực thi kiểm thử, triển khai kiểm thử, trình tự thực hiện kiểm thử, trường hợp kiểm thử (test case), xác minh (verification)

Mục tiêu Học tập của Chương 1:

1.1 Kiểm thử là gì?

FL-1.1.1 (K1) Xác định các mục tiêu kiểm thử điển hình

FL-1.1.2 (K2) Phân biệt kiểm thử và gỡ lỗi

1.2 Tại sao Cần Kiểm thử?

FL-1.2.1 (K2): Minh hoạ lý do kiểm thử là cần thiết

FL-1.2.2 (K1): Ghi nhớ mối quan hệ giữa kiểm thử và đảm bảo chất lượng

FL-1.2.3 (K2): Phân biệt nguyên nhân gốc rễ, sai sót, lỗi, và sự cố

1.3 Các Nguyên tắc Kiểm thử

FL-1.3.1 (K2) Giải thích bảy nguyên tắc kiểm thử

1.4 Hoạt động Kiểm thử, Sản phẩm Kiểm thử, và Vai trò Kiểm thử

FL-1.4.1 (K2) Giải thích các hoạt động kiểm thử và nhiệm vụ liên quan

FL-1.4.2 (K2) Giải thích ảnh hưởng của bối cảnh đến quy trình kiểm thử

FL-1.4.3 (K2) Phân biệt các sản phẩm kiểm thử được tạo ra từ các hoạt động kiểm thử

FL-1.4.4 (K2) Giải thích giá trị của việc duy trì khả năng truy vết

FL-1.4.5 (K2) So sánh các vai trò khác nhau trong kiểm thử

1.5 Kỹ năng Thiết yếu và các Thực hành Tốt trong Kiểm thử

FL-1.5.1 (K2) Minh hoạ các kỹ năng cần thiết cho kiểm thử

FL-1.5.2 (K1) Ghi nhớ các lợi ích của cách tiếp cận toàn đội ngũ

FL-1.5.3 (K2) Phân biệt lợi ích và các hạn chế của tính độc lập trong kiểm thử

1.1. Kiểm thử là gì?

Các hệ thống phần mềm là một phần không thể thiếu trong cuộc sống hàng ngày của chúng ta. Hầu hết mọi người đều từng gặp phải phần mềm không hoạt động như mong đợi. Phần mềm hoạt động không đúng có thể gây ra nhiều vấn đề, bao gồm mất tiền, mất thời gian, hoặc ảnh hưởng đến uy tín doanh nghiệp, và trong những trường hợp nghiêm trọng, thậm chí có thể gây thương tích hoặc tử vong. Kiểm thử phần mềm giúp đánh giá chất lượng phần mềm và giảm thiểu rủi ro xảy ra sự cố của phần mềm trong quá trình vận hành.

Kiểm thử phần mềm là một tập hợp nhiều hoạt động nhằm phát hiện lỗi và đánh giá chất lượng của các sản phẩm công việc (work products) của phần mềm. Các sản phẩm công việc này, khi được kiểm thử, được gọi là đối tượng kiểm thử (test object). Có một hiểu lầm phổ biến về kiểm thử phần mềm là, nó chỉ bao gồm việc thực thi test case (tức là chạy phần mềm và kiểm tra kết quả kiểm thử). Tuy nhiên, kiểm thử phần mềm còn bao gồm nhiều hoạt động khác và cần được đồng bộ với các hoạt động trong vòng đời phát triển phần mềm (xem Chương 2).

Một hiểu lầm khác là kiểm thử chỉ tập trung vào các hoạt động xác minh (verification) đối tượng kiểm thử. Thực tế, kiểm thử không chỉ bao gồm xác minh, tức là kiểm tra xem hệ thống có đáp ứng các yêu cầu đã được mô tả hay không, mà còn bao gồm thẩm định (validation), tức là đánh giá xem hệ thống có đáp ứng nhu cầu của người dùng và các bên liên quan trong môi trường vận hành hay không.

Các hoạt động kiểm thử có thể là kiểm thử động hoặc kiểm thử tĩnh. Kiểm thử động cần thực thi phần mềm, trong khi đó, kiểm thử tĩnh thì không. Kiểm thử tĩnh bao gồm các hoạt động rà soát và phân tích tĩnh (xem Chương 3). Kiểm thử động sử dụng các kỹ thuật và cách tiếp cận khác nhau để thiết kế test case (xem Chương 4).

Kiểm thử không chỉ là các hoạt động kỹ thuật. Mà còn cần được lập kế hoạch, quản lý, ước lượng, giám sát và kiểm soát một cách phù hợp với bối cảnh của từng dự án (xem Chương 5).

Kiểm thử viên (tester) cũng cần sử dụng các công cụ hỗ trợ kiểm thử (xem Chương 6), nhưng điều quan trọng cần nhớ là kiểm thử phần lớn là một hoạt động mang tính trí tuệ, đòi hỏi tester phải có kiến thức chuyên môn, kỹ năng phân tích, tư duy phản biện, và tư duy hệ thống (Myers 2011, Roman 2018).

Tiêu chuẩn ISO/IEC/IEEE 29119-1 cung cấp thêm thông tin về các khái niệm kiểm thử phần mềm.

1.1.1. Mục tiêu Kiểm thử

Các mục tiêu kiểm thử điển hình bao gồm:

- Đánh giá các sản phẩm công việc như tài liệu yêu cầu, user story, tài liệu thiết kế, và mã nguồn
- Phát hiện các sự cố và tìm lỗi
- Đảm bảo mức độ bao phủ cần thiết của đối tượng kiểm thử
- Giảm mức rủi ro liên quan đến việc chất lượng phần mềm không đạt yêu cầu
- Xác minh xem các yêu cầu đã được đáp ứng hay chưa
- Xác minh rằng đối tượng kiểm thử đã tuân thủ các yêu cầu trong hợp đồng, pháp lý, và các quy định
- Cung cấp đầy đủ thông tin giúp các bên liên quan đưa ra quyết định sáng suốt
- Tăng niềm tin vào chất lượng của đối tượng kiểm thử
- Thẩm định xem đối tượng kiểm thử đã hoàn chỉnh và hoạt động như mong đợi của các bên liên quan hay chưa

Mục tiêu kiểm thử có thể thay đổi tùy theo bối cảnh, bao gồm: sản phẩm công việc được kiểm thử, mức kiểm thử, rủi ro liên quan, mô hình phát triển phần mềm (SDLC) được áp dụng, và các yếu tố liên quan đến bối cảnh doanh nghiệp, như: cấu trúc tổ chức, các yếu tố cạnh tranh, hoặc thời gian đưa sản phẩm ra thị trường.

1.1.2. Kiểm thử và Gỡ lỗi

Kiểm thử và gỡ lỗi là hai hoạt động riêng biệt. Các hoạt động kiểm thử có thể phát hiện các sự cố do lỗi trong phần mềm gây ra (kiểm thử động) hoặc có thể trực tiếp phát hiện lỗi trong đối tượng kiểm thử (kiểm thử tĩnh).

Khi kiểm thử động (xem Chương 4) phát hiện một sự cố, hoạt động gỡ lỗi sẽ tập trung vào việc tìm nguyên nhân của sự cố này (gọi là lỗi), phân tích nguyên nhân, và loại bỏ chúng. Quy trình gỡ lỗi điển hình bao gồm các hoạt động:

- Tái hiện lại sự cố
- Chẩn đoán (để tìm lỗi)
- Sửa lỗi

Sau đó, kiểm thử xác nhận sẽ kiểm tra xem việc sửa lỗi đã thực sự giải quyết được vấn đề hay chưa. Tốt nhất, kiểm thử xác nhận nên được thực hiện bởi người đã thực hiện việc kiểm thử ban đầu. Sau đó, việc kiểm thử hồi quy cũng có thể được thực hiện để kiểm tra xem việc sửa lỗi có gây ra sự cố ở các phần khác trong đối tượng kiểm thử hay không (xem mục 2.2.3). Khi kiểm thử tĩnh phát hiện lỗi, việc gỡ lỗi sẽ tập trung vào việc loại bỏ lỗi đó. Không cần phải tái hiện hay chẩn đoán, vì kiểm thử tĩnh phát hiện lỗi trực tiếp mà không cần chạy phần mềm nên không kích hoạt sự cố (xem Chương 3).

1.2. Tại sao Cần Kiểm thử?

Kiểm thử, là một hình thức kiểm soát chất lượng, giúp đạt được các mục tiêu kiểm thử đã thỏa thuận trong phạm vi, thời gian, mức chất lượng, và ngân sách đã đặt ra. Đóng góp của kiểm thử vào thành công của dự án không chỉ giới hạn ở các hoạt động của đội kiểm thử. Bất kỳ bên liên quan nào cũng có thể sử dụng kỹ năng kiểm thử của mình để góp phần đưa dự án đến thành công. Việc kiểm thử các thành phần, hệ thống, và các sản phẩm công việc liên quan (ví dụ: tài liệu) giúp phát hiện các lỗi trong phần mềm.

1.2.1. Đóng góp của Kiểm thử vào Thành công

Kiểm thử cung cấp một phương thức hiệu quả về chi phí để phát hiện lỗi. Những lỗi này sau đó có thể được loại bỏ (thông qua quá trình gỡ lỗi – không phải là hoạt động kiểm thử), vì vậy kiểm thử gián tiếp góp phần nâng cao chất lượng của đối tượng kiểm thử.

Kiểm thử cũng cung cấp một cách để đánh giá trực tiếp chất lượng của đối tượng kiểm thử tại các giai đoạn khác nhau trong SDLC. Những đánh giá này được sử dụng như một phần hoạt động quản lý dự án tổng thể, góp phần vào các quyết định chuyển sang giai đoạn tiếp theo trong SDLC, chẳng hạn như quyết định phát hành một phiên bản nào đó hay không.

Kiểm thử cung cấp sự đại diện gián tiếp cho người dùng trong dự án phát triển phần mềm. Tester đảm bảo rằng hiểu biết của họ về nhu cầu của người dùng phải được xem xét trong suốt vòng đời phát triển phần mềm. Một phương án thay thế là chọn một nhóm người dùng đại diện tham gia trực tiếp vào dự án, nhưng điều này thường không khả thi do chi phí cao và khó tìm được người dùng phù hợp.

Việc kiểm thử cũng có thể cần thực hiện để đáp ứng các yêu cầu được nêu trong hợp đồng, liên quan đến pháp lý, hoặc để tuân thủ các tiêu chuẩn quy định.

1.2.2. Kiểm thử và Đảm bảo Chất lượng (QA)

Mặc dù nhiều người thường sử dụng hai thuật ngữ “kiểm thử” (testing) và “đảm bảo chất lượng" (QA) thay thế cho nhau, nhưng thực tế chúng không giống nhau.

Kiểm thử là cách tiếp cận hướng sản phẩm và mang tính khắc phục, tập trung vào các hoạt động hỗ trợ để phần mềm đạt được mức chất lượng phù hợp. Kiểm thử là một hình thức kiểm soát chất lượng (QC - quality control) quan trọng, bên cạnh các hình thức khác bao gồm các phương pháp chính thức (kiểm tra mô hình và chứng minh tính đúng đắn), mô phỏng, và tạo mẫu.

Đảm bảo chất lượng (QA - quality assurance) là cách tiếp cận hướng quy trình và mang tính phòng ngừa, tập trung vào việc triển khai và cải tiến các quy trình phát triển phần mềm. QA dựa trên nguyên tắc nếu một quy trình tốt được tuân thủ đúng cách, thì sẽ tạo ra một sản phẩm tốt. QA áp dụng cho cả quá trình phát triển và kiểm thử, và là trách nhiệm của tất cả mọi người trong dự án.

Kết quả kiểm thử được sử dụng bởi cả QA và kiểm thử. Với kiểm thử, chúng được dùng để sửa lỗi. Với QA, chúng cung cấp thông tin về mức độ hiệu quả của quy trình phát triển và kiểm thử phần mềm đang được áp dụng.

1.2.3. Sai sót, Lỗi, Sự cố, và Nguyên nhân Gốc rễ

Trong quá trình làm việc, con người có thể mắc sai sót (error / mistake), từ đó tạo ra lỗi (còn gọi là khiếm khuyết – defect / fault / bug), và các lỗi này có thể dẫn đến sự cố (failure) khi chương trình được thực thi. Con người mắc sai sót vì nhiều lý do khác nhau, chẳng hạn như áp lực thời gian, độ phức tạp của các sản phẩm công việc, quy trình làm việc, hạ tầng hoặc các tương tác tác nhau, hoặc đơn giản là do họ mệt mỏi hoặc thiếu sự đào tạo bài bản.

Lỗi có thể tồn tại trong tài liệu (như tài liệu đặc tả yêu cầu, hay kịch bản kiểm thử), mã nguồn, hoặc các sản phẩm công việc hỗ trợ (ví dụ: bản build phần mềm). Khi lỗi trong sản phẩm công việc được tạo ra ở những giai đoạn sớm trong SDLC, nếu không được phát hiện, thường sẽ dẫn đến lỗi trong các sản phẩm công việc ở những giai đoạn sau. Khi một lỗi trong mã nguồn được thực thi, hệ thống có thể không thực hiện đúng chức năng cần thiết, hoặc thực hiện điều không nên làm, gây ra sự cố. Một số lỗi luôn gây ra sự cố khi mã nguồn được thực thi, trong khi một số khác chỉ gây ra sự cố trong những điều kiện cụ thể, và cũng có những lỗi có thể không bao giờ gây ra sự cố.

Sai sót và lỗi không phải là nguyên nhân duy nhất gây ra sự cố. Sự cố cũng có thể xảy ra do điều kiện môi trường, ví dụ như bức xạ hoặc trường điện từ gây ra lỗi trong firmware (phần mềm điều khiển được tích hợp sẵn trong thiết bị phần cứng).

Nguyên nhân gốc rễ là lý do căn bản dẫn đến sự xuất hiện của một vấn đề (ví dụ: một tình huống dẫn đến sai sót). Nguyên nhân gốc rễ được xác định thông qua các hoạt động phân tích nguyên nhân gốc rễ (RCA – root cause analysis), thường được thực hiện khi có sự cố xảy ra hoặc khi một lỗi nghiêm trọng được phát hiện. Người ta tin rằng các sự cố hoặc lỗi tương tự trong tương lai có thể được ngăn ngừa hoặc giảm tần suất xảy ra bằng cách xử lý nguyên nhân gốc rễ, chẳng hạn như loại bỏ nguyên nhân đó.

1.3. Các Nguyên tắc Kiểm thử

Trong nhiều năm qua, đã có một số nguyên tắc kiểm thử được đề xuất nhằm cung cấp hướng dẫn chung áp dụng cho mọi hoạt động kiểm thử. Đề cương này mô tả bảy nguyên tắc như sau:

1. **Kiểm thử cho thấy sự tồn tại của lỗi, chứ không phải sự vắng mặt của lỗi**. Kiểm thử có thể cho thấy rằng lỗi đang tồn tại trong đối tượng kiểm thử, nhưng không thể chứng minh rằng nó không có lỗi (Buxton 1970). Kiểm thử giúp giảm xác suất các lỗi chưa được phát hiện còn sót lại trong đối tượng kiểm thử, nhưng ngay cả khi không phát hiện lỗi nào, quá trình kiểm thử đó cũng không thể chứng minh tính đúng đắn của đối tượng kiểm thử (như đáp ứng các đặc tả hoặc yêu cầu).

2. **Kiểm thử vét cạn là không thể**. Việc kiểm thử toàn bộ mọi thứ là không khả thi, trừ một số trường hợp đơn giản (Manna 1978). Thay vì cố gắng thực hiện kiểm thử vét cạn, nên áp dụng các kỹ thuật kiểm thử (xem chương 4), ưu tiên hóa test case (xem mục 5.1.5), và kiểm thử dựa vào rủi ro (xem mục 5.2) để tập trung công sức kiểm thử.

3. **Kiểm thử sớm giúp tiết kiệm thời gian và chi phí**. Lỗi được loại bỏ sớm trong quy trình phát triển phần mềm sẽ không gây ra các lỗi tiếp theo trong các sản phẩm công việc được tạo ra sau đó. Chi phí chất lượng sẽ giảm do ít xảy ra sự cố hơn ở các giai đoạn sau trong SDLC (Boehm 1981). Để phát hiện lỗi sớm, các hoạt động kiểm thử tĩnh (xem chương 3) và kiểm thử động (xem chương 4) nên được bắt đầu càng sớm càng tốt.

4. **Lỗi có xu hướng tập trung**. Một số ít thành phần trong hệ thống thường chứa phần lớn các lỗi được phát hiện hoặc gây ra phần lớn các sự cố khi vận hành (Enders 1975). Hiện tượng này là một minh họa cho nguyên lý Pareto 80/20. Các cụm lỗi được dự đoán, cũng như quan sát được trên thực tế, qua quá trình kiểm thử hoặc vận hành, là đầu vào quan trọng cho phương pháp kiểm thử dựa vào rủi ro (xem mục 5.2).

5. **Test case giảm hiệu quả theo thời gian**. Nếu cùng một bộ test case được thực hiện lặp đi lặp lại nhiều lần, chúng sẽ ngày càng kém hiệu quả trong việc phát hiện các lỗi mới (Beizer 1990). Để khắc phục vấn đề này, cả test case và dữ liệu kiểm thử hiện có nên cần được rà soát, sửa đổi, và có thể cần bổ sung thêm test case mới. Tuy nhiên, trong một số trường hợp, việc lặp lại cùng một bộ test case có thể mang lại kết quả có lợi, ví dụ như kiểm thử hồi quy tự động (xem mục 2.2.3).

6. **Kiểm thử phụ thuộc vào ngữ cảnh**. Không có cách tiếp cận kiểm thử nào phù hợp mọi trường hợp. Các hoạt động kiểm thử sẽ được thực hiện khác nhau tùy thuộc vào từng bối cảnh cụ thể (Kaner 2011).

7. **Ngộ nhận về việc không có lỗi**. Có một ngộ nhận (quan niệm sai lầm) là nhiều người cho rằng các hoạt động xác minh (verification) sẽ đảm bảo sự thành công của một hệ thống. Việc kiểm thử kỹ lưỡng mọi yêu cầu được mô tả và sửa tất cả những lỗi được phát hiện vẫn có thể tạo ra một hệ thống phần mềm không đáp ứng nhu cầu và kỳ vọng của người dùng, không giúp đạt được các mục tiêu kinh doanh của khách hàng, và có thể do kém hơn so với các hệ thống cạnh tranh khác. Bên cạnh xác minh, các hoạt động thẩm định (validation) cũng cần được thực hiện (Boehm 1981).

1.4. Hoạt động Kiểm thử, Sản phẩm Kiểm thử, và Vai trò Kiểm thử

Kiểm thử phụ thuộc vào ngữ cảnh, nhưng ở mức tổng thể, vẫn có các tập hoạt động kiểm thử chung mà nếu thiếu chúng thì việc kiểm thử ít có khả năng đạt được các mục tiêu kiểm thử. Các tập hoạt động này hình thành nên một quy trình kiểm thử. Quy trình kiểm thử có thể được điều chỉnh để phù hợp với một tình huống cụ thể dựa trên nhiều yếu tố khác nhau. Những hoạt động kiểm thử nào sẽ được thực hiện, cách chúng được thực hiện, và thời điểm chúng diễn ra trong quá trình kiểm thử thường được xác định trong quá trình lập kế hoạch kiểm thử cho từng tình huống cụ thể (xem thêm mục 5.1).

Những phần sau mô tả các khía cạnh chung của một quy trình kiểm thử dưới dạng các hoạt động và nhiệm vụ kiểm thử, ảnh hưởng của ngữ cảnh, sản phẩm kiểm thử, khả năng truy vết giữa cơ sở kiểm thử và sản phẩm kiểm thử, và các vai trò trong kiểm thử.

Tiêu chuẩn ISO/IEC/IEEE 29119-2 cung cấp thêm thông tin về các quy trình kiểm thử.

1.4.1. Hoạt động và Nhiệm vụ Kiểm thử

Một quy trình kiểm thử thường bao gồm các nhóm hoạt động chính được mô tả dưới đây. Mặc dù nhiều hoạt động này được trình bày theo một trình tự lôgic, nhưng chúng thường được thực hiện theo cách lặp lại hoặc song song. Các hoạt động kiểm thử này thường cần được điều chỉnh để phù hợp với từng hệ thống và dự án.

Lập kế hoạch kiểm thử bao gồm việc xác định các mục tiêu kiểm thử và sau đó lựa chọn cách tiếp cận phù hợp nhất để đạt được những mục tiêu đó, dựa vào các ràng buộc của bối cảnh tổng thể. Việc lập kế hoạch kiểm thử được giải thích chi tiết hơn trong mục 5.1.

Giám sát kiểm thử và kiểm soát kiểm thử. Giám sát kiểm thử bao gồm việc liên tục theo dõi mọi hoạt động kiểm thử và so sánh tiến độ thực tế với kế hoạch. Kiểm soát kiểm thử bao gồm việc thực hiện mọi hành động cần thiết để đạt được các mục tiêu kiểm thử. Việc giám sát và kiểm soát kiểm thử được giải thích chi tiết hơn trong mục 5.3.

Phân tích kiểm thử bao gồm việc phân tích cơ sở kiểm thử để xác định các tính năng có thể kiểm thử được. Các khía cạnh kiểm thử liên quan cũng được xác định và ưu tiên hoá, có xét đến các rủi ro và mức rủi ro liên quan (xem mục 5.2). Cơ sở kiểm thử và đối tượng kiểm thử cũng được đánh giá để tìm lỗi tiềm ẩn, và đánh giá khả năng kiểm thử của chúng. Phân tích kiểm thử thường được hỗ trợ bởi việc áp dụng các kỹ thuật kiểm thử (xem chương 4). Phân tích kiểm thử trả lời câu hỏi “kiểm thử cái gì?” dưới dạng các tiêu chí bao phủ có thể đo lường được.

Thiết kế test case bao gồm việc chi tiết hóa các khía cạnh kiểm thử thành các test case và các sản phẩm kiểm thử khác (ví dụ: test charter - thẻ kiểm thử). Hoạt động này thường bao gồm việc xác định các hạng mục bao phủ, chúng đóng vai trò hướng dẫn cách xác định đầu vào cho test case. Các kỹ thuật kiểm thử (xem chương 4) có thể được áp dụng để hỗ trợ cho hoạt động này. Thiết kế test case cũng bao gồm việc xác định yêu cầu về dữ liệu kiểm thử, thiết kế môi trường kiểm thử, và xác định hạ tầng cùng những công cụ cần thiết. Thiết kế test case trả lời câu hỏi “kiểm thử như thế nào?”.

Triển khai kiểm thử bao gồm việc tạo ra hoặc thu thập các sản phẩm kiểm thử cần thiết cho việc thực thi kiểm thử (ví dụ: dữ liệu kiểm thử). Các test case có thể được sắp xếp thành vào những trình tự thực hiện kiểm thử (test procedure), và các trình tự này sau đó thường được gom thành các bộ test case (test suite). Các test script thủ công và tự động cũng được tạo ra ở hoạt động này. Các trình tự thực hiện kiểm thử được ưu tiên hoá và sắp xếp vào lịch thực thi kiểm thử để thực hiện hiệu quả (xem mục 5.1.5). Môi trường kiểm thử được thiết lập (cài đặt hoặc tạo ra) và xác nhận chúng đã được cấu hình đúng.

Thực thi kiểm thử bao gồm việc chạy các test case theo lịch thực thi kiểm thử (gọi là test run). Việc thực thi kiểm thử có thể là thủ công hoặc tự động. Việc thực thi kiểm thử có thể có nhiều hình thức, bao gồm kiểm thử liên tục hoặc kiểm thử theo cặp (pair testing). Kết quả thực tế sẽ được so sánh với kết quả mong đợi để đánh giá kết quả kiểm thử. Các kết quả kiểm thử này được ghi lại. Mọi bất thường đều được phân tích để xác định nguyên nhân tiềm ẩn. Phân tích này giúp báo cáo mọi bất thường dựa trên các sự cố quan sát được (xem mục 5.5).

Hoàn tất kiểm thử thường diễn ra tại các mốc dự án (ví dụ: sau khi phát hành một phiên bản, kết thúc một chu kỳ phát triển, hoàn tất một mức kiểm thử nào đó). Đối với những lỗi chưa được giải quyết, cần phải tạo yêu cầu thay đổi hoặc các hạng mục khác trong danh mục công việc (product backlog) để quản lý. Mọi sản phẩm kiểm thử có thể hữu ích trong tương lai sẽ được xác định và lưu trữ hoặc bàn giao lại cho các nhóm phù hợp. Môi trường kiểm thử được đưa về trạng thái đã thống nhất trước. Các hoạt động kiểm thử được phân tích để xác định những bài học kinh nghiệm và cải tiến cho các chu kỳ phát triển, bản phát hành, hoặc dự án tiếp theo (xem mục 2.1.6). Báo cáo hoàn tất kiểm thử được tạo và gửi đến các bên liên quan.

1.4.2. Quy trình Kiểm thử theo Ngữ cảnh

Các hoạt động kiểm thử không được thực hiện riêng lẻ. Các hoạt động kiểm thử là một phần không thể tách rời của các quy trình phát triển phần mềm được thực hiện trong một tổ chức. Kiểm thử cũng được đóng góp bởi các bên liên quan và mục tiêu cuối cùng của nó là hỗ trợ đáp ứng các nhu cầu công việc. Do đó, cách thức thực hiện các hoạt động kiểm thử phụ thuộc vào nhiều yếu tố ngữ cảnh, bao gồm:

- Các bên liên quan (nhu cầu, kỳ vọng, yêu cầu, mức độ sẵn sàng hợp tác, v.v.)
- Thành viên trong nhóm (kỹ năng, kiến thức, mức độ kinh nghiệm, khả năng đáp ứng, nhu cầu đào tạo, v.v.)
- Lĩnh vực nghiệp vụ (mức độ quan trọng của đối tượng được kiểm thử, các rủi ro đã xác định, nhu cầu thị trường, quy định pháp lý cụ thể, v.v.)
- Yếu tố kỹ thuật (loại phần mềm, kiến trúc sản phẩm, công nghệ sử dụng, v.v.)
- Ràng buộc dự án (phạm vi, thời gian, ngân sách, nguồn lực, v.v.)
- Các yếu tố về tổ chức (cấu trúc tổ chức, chính sách hiện có, các thực hành đang áp dụng, v.v.)
- Vòng đời phát triển phần mềm (các thực hành kỹ thuật, phương pháp phát triển phần mềm, v.v.)
- Công cụ (tính sẵn có, khả năng sử dụng, mức độ tuân thủ, v.v.)

Các yếu tố trên sẽ ảnh hưởng nhiều vấn đề liên quan đến kiểm thử, bao gồm: chiến lược kiểm thử, kỹ thuật kiểm thử được sử dụng, mức độ tự động hóa kiểm thử, mức độ bao phủ cần thiết, mức độ chi tiết của sản phẩm kiểm thử, báo cáo kiểm thử, v.v.

1.4.3. Sản phẩm Kiểm thử (Testware)

Các sản phẩm kiểm thử được tạo ra như các sản phẩm công việc đầu ra từ các hoạt động kiểm thử được mô tả trong mục 1.4.1. Có sự khác biệt đáng kể giữa các công ty trong cách họ tạo ra, định hình, đặt tên, tổ chức, và quản lý các sản phẩm công việc của mình. Quản lý cấu hình phù hợp (xem mục 5.4) giúp đảm bảo sự nhất quán và tính toàn vẹn của các sản phẩm công việc này. Danh sách các sản phẩm công việc dưới đây chưa bao quát hết tất cả các trường hợp:

- **Sản phẩm của hoạt động lập kế hoạch kiểm thử** bao gồm: kế hoạch kiểm thử, lịch kiểm thử, danh sách rủi ro, tiêu chí bắt đầu và tiêu chí kết thúc (xem mục 5.1). Danh sách rủi ro là danh sách các rủi ro cùng với khả năng xảy ra, mức độ ảnh hưởng, và các thông tin về việc giảm thiểu rủi ro (xem mục 5.2). Lịch kiểm thử, danh sách rủi ro, tiêu chí bắt đầu, và tiêu chí kết thúc thường là một phần trong kế hoạch kiểm thử.
- **Sản phẩm của hoạt động giám sát và kiểm soát kiểm thử** bao gồm: báo cáo tiến độ kiểm thử (xem mục 5.3.2), tài liệu về các chỉ đạo kiểm soát (xem mục 5.3), và thông tin về rủi ro (xem mục 5.2).
- **Sản phẩm của hoạt động phân tích kiểm thử** bao gồm: các khía cạnh kiểm thử được ưu tiên hoá (ví dụ: tiêu chí chấp nhận, xem mục 4.5.2), và các báo cáo lỗi liên quan đến cơ sở kiểm thử (nếu các lỗi này không được sửa trực tiếp).
- **Sản phẩm của hoạt động thiết kế test case** bao gồm: các test case (được ưu tiên hoá), thẻ kiểm thử (test charter), các hạng mục bao phủ, các yêu cầu về dữ liệu kiểm thử, và yêu cầu về môi trường kiểm thử.
- **Sản phẩm của hoạt động triển khai kiểm thử** bao gồm: trình tự thực hiện kiểm thử (test procedure), kịch bản kiểm thử (test script) thủ công và tự động, bộ test case (test suite), dữ liệu kiểm thử, lịch thực thi kiểm thử, và các hạng mục môi trường kiểm thử. Ví dụ một số hạng mục môi trường kiểm thử bao gồm: mô-đun giả (stub), thành phần điều khiển kiểm thử (driver), trình mô phỏng (simulator), và ảo hoá dịch vụ (service virtualization).
- **Sản phẩm của hoạt động thực thi kiểm thử** bao gồm: nhật ký kiểm thử, và báo cáo lỗi (xem mục 5.5).
- **Sản phẩm của hoạt động hoàn tất kiểm thử** bao gồm: báo cáo hoàn tất kiểm thử (xem mục 5.3.2), các hạng mục hành động để cải tiến cho các dự án hoặc chu kỳ phát triển tiếp theo, các bài học kinh nghiệm đã được tài liệu hoá, và các yêu cầu thay đổi (ví dụ: các mục trong product backlog - danh mục công việc cần làm để tạo ra sản phẩm).

1.4.4. Khả năng Truy vết giữa Cơ sở Kiểm thử và Sản phẩm Kiểm thử

Để thực hiện các hoạt động giám sát và kiểm soát kiểm thử hiệu quả, việc thiết lập và duy trì khả năng truy vết trong suốt quy trình kiểm thử giữa các phần tử trong cơ sở kiểm thử, sản phẩm kiểm thử liên quan (ví dụ: khía cạnh kiểm thử, rủi ro, test case), kết quả kiểm thử, và lỗi là điều rất quan trọng.

Khả năng truy vết chính xác hỗ trợ việc đánh giá độ bao phủ, do đó rất hữu ích khi trong cơ sở kiểm thử xác định được các tiêu chí bao phủ có thể đo lường được. Các tiêu chí bao phủ có thể đóng vai trò như các chỉ số hiệu suất chính (KPI) để thúc đẩy các hoạt động cho thấy mức độ đạt được mục tiêu kiểm thử (xem mục 1.1.1). Ví dụ:

- Truy vết test case tới yêu cầu được mô tả, có thể xác minh rằng các yêu cầu này đã được bao phủ bởi test case.
- Truy vết kết quả kiểm thử tới rủi ro, có thể được sử dụng để đánh giá mức rủi ro còn lại của đối tượng kiểm thử.

Ngoài việc đánh giá độ bao phủ, khả năng truy vết tốt còn giúp xác định tác động ảnh hưởng của các thay đổi, hỗ trợ đánh giá tuân thủ, và đáp ứng các tiêu chí quản trị liên quan đến công nghệ thông tin. Nó cũng giúp các báo cáo tiến độ và báo cáo hoàn tất kiểm thử dễ hiểu hơn bằng cách thể hiện trạng thái của từng thành phần trong cơ sở kiểm thử. Đồng thời, nó hỗ trợ việc truyền đạt những khía cạnh kỹ thuật trong kiểm thử đến các bên liên quan một cách dễ hiểu hơn. Khả năng truy vết cung cấp thông tin để đánh giá chất lượng sản phẩm, năng lực của quy trình, và tiến độ của dự án so với mục tiêu kinh doanh.

1.4.5. Các Vai trò trong Kiểm thử

Trong đề cương này, hai vai trò chính trong kiểm thử được đề cập: vai trò quản lý kiểm thử và vai trò thực hiện kiểm thử. Các hoạt động và nhiệm vụ được giao cho hai vai trò này phụ thuộc vào nhiều yếu tố như bối cảnh của dự án và sản phẩm, kỹ năng của những người đảm nhận vai trò, và tuỳ quan điểm của tổ chức.

Vai trò quản lý kiểm thử chịu trách nhiệm tổng thể về quy trình kiểm thử, đội ngũ kiểm thử, và việc dẫn dắt các hoạt động kiểm thử. Vai trò này chủ yếu tập trung vào các hoạt động lập kế hoạch, giám sát, kiểm soát, và hoàn tất kiểm thử. Cách thực hiện vai trò này khác nhau tùy ngữ cảnh. Ví dụ, trong phát triển phần mềm Agile, một số nhiệm vụ quản lý kiểm thử có thể do các thành viên trong nhóm Agile đảm nhận. Còn những nhiệm vụ trải rộng trên nhiều nhóm hoặc toàn bộ tổ chức thì có thể do các test manager, nằm ngoài đội phát triển phần mềm, thực hiện.

Vai trò thực hiện kiểm thử chịu trách nhiệm tổng thể về các khía cạnh kỹ thuật của kiểm thử. Vai trò này chủ yếu tập trung vào các hoạt động phân tích yêu cầu, thiết kế, triển khai, và thực thi kiểm thử (ví dụ thực thi bộ test case).

Những cá nhân khác nhau có thể đảm nhận các vai trò này tại các thời điểm khác nhau. Ví dụ, vai trò quản lý kiểm thử có thể do trưởng nhóm, quản lý kiểm thử, hoặc quản lý nhóm phát triển đảm nhận. Một cá nhân có thể đảm nhận đồng thời cả vai trò quản lý kiểm thử và thực hiện kiểm thử.

1.5. Kỹ năng Thiết yếu và Thực hành Tốt trong Kiểm thử

Kỹ năng là khả năng thực hiện thành thạo một việc, nó được hình thành từ kiến thức, thực hành, và tố chất cá nhân. Tester giỏi cần có một số kỹ năng thiết yếu để thực hiện tốt công việc. Họ cũng còn là những thành viên làm việc nhóm hiệu quả và có khả năng thực hiện kiểm thử ở các mức độ độc lập khác nhau.

1.5.1. Các kỹ năng Cần thiết Chung trong Kiểm thử

Mặc dù mang tính chung chung, các kỹ năng sau đặc biệt quan trọng đối với tester:

- Kiến thức về kiểm thử (để nâng cao hiệu quả trong kiểm thử, ví dụ như việc áp dụng các kỹ thuật kiểm thử)
- Sự cẩn thận, tỉ mỉ, tính tò mò, chú ý đến chi tiết, làm việc có phương pháp (để phát hiện lỗi, đặc biệt là các lỗi khó tìm)
- Kỹ năng giao tiếp tốt, lắng nghe chủ động, khả năng làm việc nhóm (để tương tác hiệu quả với các bên liên quan, truyền đạt thông tin, được hiểu đúng, và để báo cáo, thảo luận về lỗi)
- Tư duy phân tích, tư duy phản biện, tính sáng tạo (để nâng cao hiệu quả trong kiểm thử)
- Kiến thức kỹ thuật (để nâng cao hiệu quả thực thi kiểm thử, ví dụ bằng cách sử dụng công cụ hỗ trợ kiểm thử phù hợp)
- Kiến thức về lĩnh vực nghiệp vụ (để hiểu và giao tiếp với người dùng cuối hoặc các đại diện nghiệp vụ như PO, BA)

Tester thường là người truyền “tin xấu”. Một đặc điểm phổ biến của con người là có xu hướng đổ lỗi cho người mang tin xấu, do đó kỹ năng giao tiếp trở nên đặc biệt quan trọng đối với tester. Việc truyền đạt kết quả kiểm thử có thể bị xem là sự chỉ trích sản phẩm hoặc người tạo ra nó. Thiên kiến xác nhận (confirmation bias) có thể khiến con người khó chấp nhận thông tin trái với niềm tin hiện tại của họ. Một số người xem các hoạt động kiểm thử là hoạt động mang tính phá hoại, mặc dù các hoạt động này đóng góp lớn vào sự thành công của dự án và chất lượng sản phẩm. Để cải thiện nhận thức này, thông tin về lỗi và sự cố cần được truyền đạt theo cách mang tính chất đóng góp xây dựng.

1.5.2. Cách Tiếp cận Toàn Đội ngũ

Một trong những kỹ năng quan trọng của tester là khả năng làm việc nhóm hiệu quả và đóng góp tích cực vào các mục tiêu chung. Cách tiếp cận toàn đội ngũ (whole team approach) được xây dựng dựa trên kỹ năng này. Đây là một thực hành xuất phát từ phương pháp phát triển phần mềm cực đoan – Extreme Programming (XP) (xem mục 2.1).

Trong cách tiếp cận này, các thành viên trong nhóm có đủ kiến thức và kỹ năng để có thể thực hiện bất kỳ nhiệm vụ nào, và tất cả mọi người trong nhóm đều chịu trách nhiệm về chất lượng. Các thành viên làm việc cùng nhau trong một không gian (vật lý hoặc ảo), vì cùng vị trí làm việc sẽ giúp tăng cường khả năng giao tiếp và tương tác với nhau. Cách tiếp cận toàn đội giúp cải thiện động lực nhóm, tăng cường giao tiếp và phối hợp, đồng thời tạo ra sự cộng hưởng bằng cách tận dụng các kỹ năng đa dạng của mọi thành viên vì lợi ích của dự án.

Tester làm việc chặt chẽ với các thành viên khác để đảm bảo đạt được mức chất lượng mong muốn. Bao gồm việc hợp tác với đại diện nghiệp vụ để xây dựng test case phù hợp ở mức chấp nhận và làm việc với lập trình viên để thống nhất về chiến lược kiểm thử cũng như quyết định cách tiếp cận về tự động hóa kiểm thử. Qua đó, tester có thể chia sẻ kiến thức kiểm thử cho các thành viên khác và tác động đến quá trình phát triển phần mềm.

Tùy theo bối cảnh, cách tiếp cận toàn đội ngũ không phải lúc nào cũng phù hợp. Ví dụ, trong một số trường hợp như hệ thống cần tính an toàn cao, có thể cần mức kiểm thử độc lập cao hơn.

1.5.3. Tính Độc lập của Kiểm thử

Khi thực hiện kiểm thử ở một mức độ độc lập nhất định, sẽ giúp tester phát hiện lỗi hiệu quả hơn, do sự khác biệt về thiên kiến nhận thức giữa tester và người tạo ra đối tượng kiểm thử (Salman 1995). Tuy nhiên, tính độc lập không thể thay thế sự am hiểu, ví dụ, các lập trình viên có thể phát hiện nhiều lỗi trong chính mã nguồn do mình viết.

Các sản phẩm công việc có thể được kiểm thử bởi chính người tạo ra nó (không có tính độc lập), đồng nghiệp trong cùng nhóm (mức độ độc lập thấp), tester bên ngoài nhóm nhưng trong cùng tổ chức (mức độ độc lập cao), và tester bên ngoài tổ chức (mức độ độc lập rất cao). Đối với hầu hết các dự án, cách tốt nhất là thực hiện kiểm thử với nhiều mức độ độc lập khác nhau (ví dụ: lập trình viên thực hiện kiểm thử thành phần và kiểm thử tích hợp thành phần; đội tester riêng thực hiện kiểm thử ở mức hệ thống và mức tích hợp hệ thống; đại diện nghiệp vụ (như PO, PM, BA) thực hiện kiểm thử ở mức chấp nhận).

Lợi ích chính của tính độc lập trong kiểm thử là tester độc lập có khả năng phát hiện được các loại lỗi và sự cố khác so với lập trình viên, do sự khác biệt về nền tảng, góc nhìn kỹ thuật, và thiên kiến của họ. Ngoài ra, tester độc lập có thể xác minh, chất vấn, hoặc bác bỏ các giả định được đưa ra bởi các bên liên quan trong quá trình thảo luận đặc tả và triển khai hệ thống.

Tuy nhiên, kiểm thử độc lập cũng tồn tại một số hạn chế. Tester độc lập có thể bị cô lập khỏi nhóm phát triển, dẫn đến thiếu hợp tác giữa các bên, nảy sinh các vấn đề giao tiếp hoặc các mối quan hệ đối đầu với đội phát triển. Các lập trình viên có thể mất đi ý thức trách nhiệm về chất lượng. Tester độc lập cũng có thể bị xem là nút thắt cổ chai, hoặc bị đổ lỗi cho việc chậm trễ trong việc phát hành sản phẩm.

2. Kiểm thử Trong suốt Vòng đời Phát triển Phần mềm – 130 phút

Từ khóa

đối tượng kiểm thử, kiểm thử bảo trì, kiểm thử chấp nhận, kiểm thử chức năng, kiểm thử hồi quy, kiểm thử hệ thống, kiểm thử hộp đen, kiểm thử hộp trắng, kiểm thử phi chức năng, kiểm thử thành phần, kiểm thử tích hợp, kiểm thử tích hợp hệ thống, kiểm thử tích hợp thành phần, kiểm thử xác nhận, loại kiểm thử, mức kiểm thử, shift left (kiểm thử sớm)

Mục tiêu Học tập của Chương 2:

2.1 Kiểm thử theo Bối cảnh Vòng đời Phát triển Phần mềm

FL-2.1.1 (K2) Giải thích ảnh hưởng của vòng đời phát triển phần mềm được lựa chọn đối với các hoạt động kiểm thử

FL-2.1.2 (K1) Ghi nhớ các thực hành kiểm thử tốt, áp dụng cho mọi vòng đời phát triển phần mềm

FL-2.1.3 (K1) Ghi nhớ các ví dụ về các phương pháp phát triển hướng kiểm thử (TDD)

FL-2.1.4 (K2) Tóm tắt cách DevOps có thể ảnh hưởng đến các hoạt động kiểm thử

FL-2.1.5 (K2) Giải thích khái niệm shift left

FL-2.1.6 (K2) Giải thích cách các buổi retrospective (họp rút kinh nghiệm) có thể được sử dụng như một cơ chế cải tiến quy trình làm việc

2.2 Các Mức Kiểm thử và các Loại Kiểm thử

FL-2.2.1 (K2) Phân biệt các mức kiểm thử khác nhau

FL-2.2.2 (K2) Phân biệt các loại kiểm thử khác nhau

FL-2.2.3 (K2) Phân biệt kiểm thử xác nhận và kiểm thử hồi quy

2.3 Kiểm thử Bảo trì

FL-2.3.1 (K2) Tóm tắt kiểm thử bảo trì và các yếu tố kích hoạt nó

2.1. Kiểm thử theo Bối cảnh Vòng đời Phát triển Phần mềm (SDLC)

Một mô hình SDLC là một biểu diễn trừu tượng, ở mức khái quát cao của một quy trình phát triển phần mềm. Mô hình SDLC xác định cách các giai đoạn phát triển và các loại hoạt động khác nhau, được thực hiện trong quy trình này liên quan với nhau, cả về mặt lôgic lẫn theo trình tự thời gian. Một số ví dụ về mô hình SDLC bao gồm: Mô hình phát triển tuần tự (ví dụ: mô hình thác nước – waterfall, mô hình chữ V – V-model), mô hình phát triển lặp (ví dụ: mô hình xoắn ốc – spiral, phát triển nguyên mẫu – prototyping), và mô hình phát triển gia tăng (ví dụ: Unified Process - Quy trình Hợp nhất).

Một số hoạt động trong quy trình phát triển phần mềm cũng có thể được mô tả chi tiết hơn thông qua các phương pháp phát triển phần mềm và các thực hành Agile. Ví dụ bao gồm: phát triển hướng kiểm thử chấp nhận (ATDD), phát triển hướng hành vi (BDD), thiết kế hướng lĩnh vực (DDD), lập trình cực hạn (extreme programming - XP), phát triển hướng tính năng (FDD), Kanban, Lean IT, Scrum, và phát triển hướng kiểm thử (TDD).

2.1.1. Ảnh hưởng của Vòng đời Phát triển Phần mềm đến việc Kiểm thử

Để đạt hiệu quả tốt, các hoạt động kiểm thử cần được điều chỉnh phù hợp với mô hình SDLC. Việc lựa chọn SDLC ảnh hưởng đến:

- Phạm vi và thời điểm của các hoạt động kiểm thử (ví dụ: các mức kiểm thử và các loại kiểm thử)
- Mức độ chi tiết của tài liệu kiểm thử
- Lựa chọn kỹ thuật kiểm thử và cách tiếp cận kiểm thử
- Mức độ tự động hóa kiểm thử
- Vai trò và trách nhiệm của tester

Trong các mô hình phát triển tuần tự, ở các giai đoạn đầu, tester thường tham gia vào việc rà soát yêu cầu, phân tích kiểm thử, và thiết kế test case. Mã nguồn có thể thực thi được thường được tạo ra ở những giai đoạn về sau, do đó việc kiểm thử động thường không thể thực hiện được trong các giai đoạn đầu của SDLC.

Trong một số mô hình phát triển lặp và phát triển gia tăng, giả định rằng mỗi chu kỳ phát triển đều tạo ra một phần gia tăng sản phẩm hoặc một nguyên mẫu hoạt động được. Điều này có nghĩa là trong mỗi chu kỳ phát triển, cả kiểm thử tĩnh và kiểm thử động đều có thể được thực hiện ở tất cả các mức kiểm thử. Việc cung cấp các phần gia tăng sản phẩm thường xuyên đòi hỏi phản hồi nhanh và kiểm thử hồi quy ở mức độ cao.

Phát triển phần mềm Agile giả định rằng thay đổi yêu cầu có thể diễn ra trong suốt dự án. Vì vậy, sản phẩm công việc thường được tài liệu hoá theo cách gọn nhẹ, và việc tự động hóa kiểm thử được áp dụng rộng rãi để giúp kiểm thử hồi quy trở nên dễ dàng hơn. Ngoài ra, phần lớn các hoạt động kiểm thử thủ công thường được thực hiện bằng các kỹ thuật kiểm thử dựa trên kinh nghiệm (xem Mục 4.4), vì chúng không yêu cầu phân tích kiểm thử và thiết kế test case chi tiết từ trước.

2.1.2. Vòng đời Phát triển Phần mềm và các Thực hành Kiểm thử Tốt

Các thực hành kiểm thử tốt, không phụ thuộc vào mô hình SDLC được chọn, bao gồm:

- Mỗi hoạt động phát triển phần mềm đều có một hoạt động kiểm thử tương ứng, nhằm đảm bảo tất cả các hoạt động phát triển đều được kiểm soát chất lượng.
- Các mức kiểm thử khác nhau (xem mục 2.2.1) có các mục tiêu kiểm thử riêng biệt, giúp hoạt động kiểm thử đạt được mức độ bao phủ phù hợp, đồng thời tránh trùng lặp.
- Phân tích và thiết kế test case cho một mức kiểm thử cụ thể nên bắt đầu ngay trong giai đoạn phát triển tương ứng của SDLC, nhằm đảm bảo kiểm thử tuân theo nguyên tắc kiểm thử sớm (xem mục 1.3).
- Tester tham gia rà soát các sản phẩm công việc ngay khi có bản nháp, để việc kiểm thử và phát hiện lỗi sớm có thể hỗ trợ cho phương pháp tiếp cận shift left (xem mục 2.1.5).

2.1.3. Kiểm thử như một Yếu tố Dẫn dắt Phát triển Phần mềm

TDD, ATDD, và BDD là các phương pháp phát triển phần mềm tương tự nhau, trong đó test case được xem như một phương tiện để định hướng cho việc phát triển phần mềm. Các phương pháp này đều áp dụng nguyên tắc kiểm thử sớm (xem mục 1.3) và tuân theo phương pháp shift left (xem mục 2.1.5), vì các test case được xác định trước khi viết mã nguồn. Chúng hỗ trợ các mô hình phát triển lặp. Các cách tiếp cận này có những đặc điểm như sau:

Phát triển Hướng Kiểm thử (TDD):

- Định hướng việc lập trình thông qua các test case (thay vì dựa vào thiết kế phần mềm chi tiết) (Beck 2003)
- Test case được viết trước, sau đó mã nguồn được viết để đáp ứng các test case đó, rồi tiếp tục tái cấu trúc cả test case và mã nguồn

Phát triển Hướng Kiểm thử Chấp nhận (ATDD) (xem mục 4.5.3):

- Xây dựng test case dựa trên các tiêu chí chấp nhận như một phần của quá trình thiết kế hệ thống (Gärtner 2011)
- Test case được viết trước khi phát triển phần tương ứng của ứng dụng để đáp ứng các test case này

Phát triển Hướng Hành vi (BDD):

- Mô tả hành vi mong muốn của ứng dụng bằng các test case được viết dưới dạng ngôn ngữ tự nhiên đơn giản, dễ hiểu đối với các bên liên quan – thường sử dụng định dạng Given/When/Then (Chelimsky 2010)
- Sau đó, các test case này nên được tự động chuyển đổi thành các test case có thể thực thi được

Đối với tất cả các phương pháp phát triển phần mềm trên, test case có thể được thực hiện dưới dạng test case tự động nhằm đảm bảo chất lượng cho mã nguồn trong những lần thay đổi hoặc tái cấu trúc sau này.

2.1.4. DevOps và Kiểm thử

DevOps là một cách tiếp cận ở cấp độ tổ chức nhằm tạo ra sự phối hợp hiệu quả giữa việc phát triển (bao gồm các hoạt động kiểm thử) và vận hành phần mềm để cùng đạt được các mục tiêu chung. DevOps đòi hỏi sự thay đổi về văn hóa trong tổ chức nhằm thu hẹp khoảng cách giữa các bộ phận phát triển (gồm kiểm thử) và vận hành, đồng thời xem các bộ phận này có giá trị ngang nhau. DevOps thúc đẩy tính tự chủ của nhóm, phản hồi nhanh, áp dụng chuỗi công cụ tích hợp, và các thực hành về kỹ thuật như tích hợp liên tục (continuous integration – CI) và phân phối liên tục (continuous delivery – CD). Điều này giúp các nhóm phát triển phần mềm có thể xây dựng, kiểm thử, và phát hành chương trình chất lượng cao nhanh hơn thông qua quy trình phân phối DevOps (Kim 2016).

Nhìn từ góc độ kiểm thử, một số lợi ích của DevOps bao gồm:

- Phản hồi nhanh về chất lượng mã nguồn, cũng như việc các thay đổi có ảnh hưởng tiêu cực đến mã nguồn hiện có hay không
- Phản hồi nhanh về chất lượng mã nguồn, cũng như việc các thay đổi có ảnh hưởng tiêu cực đến mã nguồn hiện có hay không
- CI thúc đẩy việc áp dụng phương pháp shift left trong kiểm thử (xem mục 2.1.5) bằng cách khuyến khích các lập trình viên tạo ra mã nguồn chất lượng cao, kèm theo test case ở mức thành phần và phân tích tĩnh.
- Thúc đẩy các quy trình tự động hóa như CI/CD, giúp thiết lập môi trường kiểm thử ổn định
- Tăng khả năng quan sát các đặc tính chất lượng phi chức năng (ví dụ: hiệu năng, độ tin cậy)

- Tự động hóa thông qua quy trình phân phối CD, giúp giảm nhu cầu kiểm thử thủ công lặp đi lặp lại
- Giảm thiểu nguy cơ phát sinh lỗi trở lại, nhờ phạm vi và quy mô của các test case hồi quy tự động

DevOps cũng đi kèm với một số thách thức và rủi ro, bao gồm:

- Cần xác định và thiết lập quy trình phân phối DevOps
- Cần triển khai và duy trì các công cụ CI/CD
- Kiểm thử tự động đòi hỏi thêm nguồn lực và có thể khó thiết lập cũng như duy trì

Mặc dù DevOps đi kèm với mức độ tự động hóa kiểm thử cao, nhưng kiểm thử thủ công – đặc biệt kiểm thử từ góc độ người dùng – vẫn luôn cần thiết.

2.1.5. Shift Left

Nguyên tắc kiểm thử sớm (xem mục 1.3) đôi khi được gọi là shift left, vì đây là cách tiếp cận trong đó các hoạt động kiểm thử được thực hiện sớm hơn trong SDLC. Về cơ bản, shift left đề xuất các hoạt động kiểm thử nên được tiến hành sớm hơn (ví dụ: không chờ đến khi mã nguồn được lập trình xong, hoặc các thành phần đã được tích hợp), tuy nhiên điều này không có nghĩa là các hoạt động kiểm thử ở các giai đoạn về sau trong SDLC bị xem nhẹ.

Một số thực hành tốt minh họa cách thực hiện “shift left” trong kiểm thử bao gồm:

- Rà soát tài liệu đặc tả từ góc nhìn của tester. Các hoạt động rà soát này thường giúp phát hiện các lỗi tiềm ẩn như: mô tả không rõ ràng, thiếu sót, và không nhất quán
- Viết test case trước khi viết mã nguồn, và thực thi mã nguồn trong môi trường chạy kiểm thử (test harness) trong quá trình lập trình
- Sử dụng CI và tốt hơn nữa là CD, vì chúng cung cấp phản hồi nhanh, và các test case thành phần được tự động hoá đi kèm với mã nguồn khi được đưa vào kho mã nguồn (như GitHub, Bitbucket)
- Thực hiện phân tích tĩnh mã nguồn trước khi tiến hành kiểm thử động, hoặc tích hợp nó như một phần của quy trình tự động
- Thực hiện kiểm thử phi chức năng ngay từ mức kiểm thử thành phần, khi có thể. Đây cũng là một dạng shift left, vì các loại kiểm thử phi chức năng này thường được thực hiện muộn hơn trong SDLC, khi đã có hệ thống hoàn chỉnh và môi trường kiểm thử thích hợp tương ứng

Áp dụng shift left, có thể dẫn đến việc cần bổ sung thêm việc đào tạo, nguồn lực, và chi phí ở giai đoạn đầu của quy trình, nhưng được kỳ vọng sẽ giúp tiết kiệm nguồn lực và chi phí trong các giai đoạn về sau. Để triển khai shift left hiệu quả, điều quan trọng là các bên liên quan cần hiểu, đồng thuận, và ủng hộ cách tiếp cận này.

2.1.6. Retrospectives và Cải tiến Quy trình

Các buổi retrospective (họp rút kinh nghiệm) thường được tổ chức vào cuối dự án hoặc cuối mỗi chu kỳ phát triển, tại các mốc phát hành, hoặc khi cần thiết. Thời điểm và cách thức tổ chức retrospective phụ thuộc vào mô hình SDLC cụ thể đang được áp dụng trong công ty. Trong các cuộc họp này, những người tham gia (không chỉ tester mà còn có thể bao gồm lập trình viên, kiến trúc hệ thống, chủ sở hữu sản phẩm – PO, chuyên viên phân tích nghiệp vụ – BA, v.v.) sẽ thảo luận về:

- Điều gì đã thành công và nên được duy trì?
- Điều gì chưa thành công và có thể được cải thiện?
- Làm thế nào để áp dụng các cải tiến và duy trì các điểm tốt trong tương lai?

Kết quả của các buổi retrospective cần được ghi lại và thường là một phần trong báo cáo hoàn tất kiểm thử (xem mục 5.3.2). Retrospective đóng vai trò quan trọng trong việc triển khai cải tiến liên tục, và điều quan trọng là các đề xuất cải tiến phải được theo dõi và thực hiện.

Một số lợi ích điển hình của retrospective đối với hoạt động kiểm thử bao gồm:

- Nâng cao hiệu quả, hiệu suất kiểm thử (ví dụ: thông qua việc áp dụng các đề xuất cải tiến quy trình)
- Nâng cao chất lượng sản phẩm kiểm thử (ví dụ: thông qua việc cùng nhau rà soát các quy trình kiểm thử)
- Tăng cường gắn kết đội nhóm và học hỏi (ví dụ: vì cơ hội nêu vấn đề và các đề xuất cải tiến)
- Cải thiện chất lượng cơ sở kiểm thử (ví dụ: vì những thiếu sót về phạm vi và chất lượng của yêu cầu có thể được xử lý và khắc phục)
- Tăng cường sự hợp tác giữa bộ phận phát triển và bộ phận kiểm thử (ví dụ: vì việc thường xuyên xem xét và tối ưu cách hợp tác)

2.2. Các mức Kiểm thử và các Loại Kiểm thử

Mức kiểm thử (test level) là các nhóm các hoạt động kiểm thử được tổ chức và quản lý cùng nhau. Mỗi mức kiểm thử là một thể hiện của quy trình kiểm thử, được thực hiện đối với phần mềm ở một giai đoạn phát triển nhất định, từ các thành phần riêng lẻ đến toàn bộ hệ thống hoặc, khi phù hợp, đến các hệ thống gồm nhiều hệ thống.

Các mức kiểm thử có mối liên hệ với các hoạt động khác trong SDLC. Trong các mô hình SDLC tuần tự, các mức kiểm thử thường được xác định sao cho tiêu chí kết thúc của mức này sẽ là một phần của tiêu chí đầu vào của mức tiếp theo. Điều này có thể không áp dụng được trong một số mô hình lặp. Các hoạt động phát triển phần mềm có thể được thực hiện ở nhiều mức kiểm thử, và các mức kiểm thử cũng có thể diễn ra đồng thời.

Loại kiểm thử (test type) là các nhóm hoạt động kiểm thử liên quan đến các đặc tính chất lượng cụ thể, và hầu hết các hoạt động này có thể được thực hiện ở mọi mức kiểm thử.

2.2.1. Các Mức Kiểm thử

Trong đề cương này, năm mức kiểm thử sau sẽ được mô tả:

- **Kiểm thử thành phần** (còn gọi là unit testing – kiểm thử đơn vị) tập trung kiểm thử các thành phần một cách độc lập. Thường cần sự hỗ trợ chuyên biệt như môi trường chạy kiểm thử (test harness) hoặc các công cụ kiểm thử đơn vị. Kiểm thử thành phần thường được thực hiện bởi lập trình viên trong môi trường phát triển phần mềm của họ.
- **Kiểm thử tích hợp thành phần** (component integration testing) tập trung kiểm thử các điểm kết nối và sự tương tác giữa các thành phần. Mức kiểm thử này phụ thuộc nhiều vào chiến lược tích hợp như bottom-up, top-down, hoặc big-bang.
- **Kiểm thử hệ thống** (system testing) tập trung vào hành vi tổng thể và khả năng của toàn bộ hệ thống hoặc sản phẩm, thường bao gồm kiểm thử chức năng cho các luồng nghiệp vụ (kiểm thử end-to-end) và kiểm thử phi chức năng đối với các đặc tính chất lượng. Đối với một số đặc tính chất lượng phi chức năng, nên kiểm thử chúng trên một hệ thống hoàn chỉnh trong môi trường kiểm thử mang tính đại diện (ví dụ: tính khả dụng – usability). Có thể sử dụng việc mô phỏng cho các hệ thống con trong quá trình kiểm thử. Kiểm thử hệ thống có thể được thực hiện bởi một nhóm kiểm thử độc lập, và có liên quan đến các đặc tả của hệ thống.
- **Kiểm thử tích hợp hệ thống** (system integration testing) tập trung kiểm thử các điểm kết nối giữa hệ thống đang kiểm thử với các hệ thống khác hoặc các dịch vụ bên ngoài. Mức kiểm thử này yêu cầu môi trường kiểm thử phù hợp, tốt nhất là gần giống với môi trường vận hành thực tế.
- **Kiểm thử chấp nhận** (acceptance testing) tập trung vào việc xác nhận (validation) và chứng minh mức độ sẵn sàng để triển khai của hệ thống, tức là hệ thống đã đáp ứng mọi nhu cầu nghiệp vụ của người dùng. Lý tưởng nhất, kiểm thử chấp nhận nên được thực hiện bởi người dùng mục tiêu (nhóm người mà hệ thống được thiết kế để nhắm đến). Các hình thức chính của kiểm thử chấp nhận bao gồm: kiểm thử chấp nhận người dùng (UAT), kiểm thử chấp nhận vận hành, kiểm thử chấp nhận theo hợp đồng, kiểm thử chấp nhận theo quy định, kiểm thử

alpha, và kiểm thử beta.

Các mức kiểm thử được phân biệt dựa trên danh sách các thuộc tính (không đầy đủ) sau, nhằm tránh sự chồng chéo giữa các hoạt động kiểm thử:

- Đối tượng kiểm thử
- Mục tiêu kiểm thử
- Cơ sở kiểm thử
- Lỗi và sự cố
- Cách tiếp cận để thực hiện và trách nhiệm các bên

2.2.2. Các Loại Kiểm thử

Có rất nhiều loại kiểm thử khác nhau và có thể được áp dụng trong các dự án. Đề cương này đề cập đến bốn loại kiểm thử sau:

Kiểm thử chức năng (functional testing) đánh giá các chức năng mà một thành phần hoặc hệ thống cần thực hiện. Các chức năng cho thấy hệ thống phải "làm gì". Mục tiêu chính của kiểm thử chức năng là đánh giá tính đầy đủ của chức năng, tính đúng đắn của chức năng, và tính phù hợp của chức năng.

Kiểm thử phi chức năng (non-functional testing) đánh giá các thuộc tính khác chức năng của thành phần hoặc hệ thống. Đây là việc đánh giá hệ thống hoạt động "tốt như thế nào". Mục tiêu chính là đánh giá các đặc tính chất lượng phi chức năng. Tiêu chuẩn ISO/IEC 25010 phân loại các đặc tính chất lượng phi chức năng như sau:

- Hiệu năng hoạt động
- Khả năng tương thích
- Tính khả dụng, còn gọi là khả năng tương tác
- Độ tin cậy
- Khả năng bảo trì
- Tính di động, còn gọi là tính linh hoạt
- Tính an toàn

Trong một số trường hợp, kiểm thử phi chức năng nên được bắt đầu sớm trong SDLC (ví dụ: như một phần của rà soát hoặc kiểm thử thành phần). Nhiều kiểm thử phi chức năng được xây dựng dựa trên kiểm thử chức năng, sử dụng cùng các test case nhưng kiểm tra thêm các ràng buộc phi chức năng (ví dụ: kiểm tra một chức năng có thể hoàn thành trong một khoảng thời gian nhất định, hay có thể chạy trên nền tảng mới hay không). Việc phát hiện muộn các lỗi phi chức năng có thể gây rủi ro nghiêm trọng cho sự thành công của dự án. Một số kiểm thử phi chức năng đòi hỏi môi trường kiểm thử rất đặc thù, ví dụ như môi trường để kiểm thử tính khả dụng của phần mềm.

Kiểm thử hộp đen (Black-box testing) (xem mục 4.2) là việc kiểm thử dựa trên đặc tả, trong đó các test case được thiết kế dựa vào tài liệu, không liên quan đến cấu trúc bên trong của đối tượng kiểm thử. Mục tiêu chính của kiểm thử hộp đen là để kiểm tra hành vi của hệ thống so với đặc tả.

Kiểm thử hộp trắng (White-box testing) (xem mục 4.3) là việc kiểm thử dựa trên cấu trúc, trong đó các test case được thiết kế dựa trên kiến thức về cấu trúc bên trong hoặc chính mã nguồn của hệ thống (ví dụ: mã nguồn, kiến trúc, luồng điều khiển, và luồng dữ liệu). Mục tiêu chính của kiểm thử hộp trắng là đảm bảo test case hiện có đã bao phủ các cấu trúc bên trong (của đối tượng kiểm thử) ở một mức chấp nhận được.

Tất cả bốn loại kiểm thử nêu trên đều có thể được thực hiện ở mọi mức kiểm thử, mặc dù trọng tâm của chúng sẽ khác nhau ở từng mức. Các kỹ thuật kiểm thử khác nhau có thể được áp dụng để giúp xác định các khía cạnh kiểm thử và test case cho tất cả các loại kiểm thử này.

2.2.3. Kiểm thử Xác nhận và Kiểm thử Hồi quy

Các thay đổi thường được thực hiện đối với một thành phần hoặc hệ thống nhằm mục đích nâng cấp hoặc sửa lỗi. Do đó, các hoạt động kiểm thử cũng cần thực hiện kiểm thử xác nhận (confirmation testing) và kiểm thử hồi quy (regression testing).

Kiểm thử xác nhận xác nhận rằng lỗi đã được sửa thành công. Tùy theo mức rủi ro, có thể thực hiện kiểm thử trên phiên bản đã sửa theo nhiều cách, bao gồm:

- Chạy lại tất cả các test case không đạt trước đó do lỗi, hoặc đồng thời
- Bổ sung thêm các test case mới để bao phủ các thay đổi vừa thực hiện để sửa lỗi

Tuy nhiên, khi thời gian hoặc chi phí hạn chế, kiểm thử xác nhận có thể chỉ giới hạn ở việc thực hiện lại các bước tái hiện lỗi trước đó và kiểm tra rằng lỗi không còn xảy ra nữa.

Kiểm thử hồi quy xác nhận rằng việc thực hiện những thay đổi (bao gồm cả việc sửa lỗi đã được kiểm thử xác nhận) không gây ra các tác động tiêu cực nào. Những tác động này có thể ảnh hưởng đến chính thành phần được thay đổi, các thành phần khác trong cùng hệ thống, hoặc thậm chí các hệ thống khác có liên kết. Kiểm thử hồi quy không chỉ giới hạn ở đối tượng kiểm thử mà còn có thể liên quan đến môi trường. Nên thực hiện phân tích ảnh hưởng trước để xác định phạm vi kiểm thử hồi quy cần thiết, tức là xác định những phần nào của phần mềm có thể bị ảnh hưởng.

Các bộ test case hồi quy thường được thực thi nhiều lần và số lượng test case thường tăng lên sau mỗi chu kỳ phát triển hoặc mỗi lần phát hành, vì vậy kiểm thử hồi quy rất phù hợp để tự động hóa. Việc tự động hóa kiểm thử nên được bắt đầu sớm trong dự án. Khi áp dụng CI, như trong DevOps (xem mục 2.1.4), có một thực hành tốt là nên tích hợp các test case hồi quy tự động vào quy trình phân phối CD. Tùy tình huống, điều này có thể bao gồm kiểm thử hồi quy ở nhiều mức kiểm thử khác nhau từ kiểm thử thành phần đến kiểm thử hệ thống.

Kiểm thử xác nhận và/hoặc kiểm thử hồi quy đều cần thiết ở tất cả các mức kiểm thử nếu có lỗi được sửa và/hoặc có thay đổi được thực hiện tại các mức đó.

2.3. Kiểm thử Bảo trì

Có nhiều loại bảo trì khác nhau, bao gồm: sửa lỗi, cập nhật phần mềm tương ứng với các thay đổi của môi trường, hoặc cải thiện hiệu năng hay khả năng bảo trì (xem ISO/IEC 14764 để biết thêm chi tiết). Do đó, hoạt động bảo trì có thể bao gồm các lần phát hành/triển khai được lên kế hoạch và không được lên kế hoạch trước, như các bản vá lỗi khẩn cấp.

Việc phân tích ảnh hưởng có thể được thực hiện trước khi tiến hành thay đổi, nhằm hỗ trợ quyết định có nên thực hiện thay đổi đó hay không, dựa trên các hệ quả tiềm ẩn đối với các phần khác của hệ thống. Việc kiểm thử các thay đổi trên một hệ thống đang vận hành, bao gồm cả việc đánh giá sự thành công của việc triển khai các thay đổi và kiểm tra các lỗi hồi quy có thể xảy ra trong các khu vực không bị thay đổi của hệ thống (thường chiếm phần lớn trong hệ thống).

Phạm vi kiểm thử bảo trì thường phụ thuộc vào:

- Mức rủi ro của các thay đổi
- Kích thước của hệ thống hiện tại
- Quy mô của thay đổi

Các yếu tố dẫn đến hoạt động bảo trì và kiểm thử bảo trì có thể được phân loại như sau:

- Các thay đổi trong phần mềm, như các cải tiến đã lên kế hoạch (theo từng bản phát hành), các thay đổi để sửa lỗi, hoặc các bản vá lỗi khẩn cấp.
- Nâng cấp hoặc chuyển đổi môi trường vận hành, như chuyển từ nền tảng này sang nền tảng khác. Điều này có thể yêu cầu kiểm thử liên quan đến môi trường mới cũng như phần mềm đã được thay đổi, hoặc kiểm thử chuyển đổi dữ liệu khi dữ liệu từ ứng dụng khác được chuyển vào hệ thống đang được bảo trì.
- Ngừng sử dụng, như khi một ứng dụng kết thúc vòng đời của nó. Khi hệ thống được ngừng sử dụng, có thể cần kiểm thử việc lưu trữ dữ liệu, nếu có yêu cầu lưu trữ trong thời gian dài. Ngoài ra, cũng có thể cần đánh giá các quy trình khôi phục và truy xuất dữ liệu sau khi lưu trữ, trong trường hợp cần sử dụng lại dữ liệu trong thời gian lưu trữ.

3. Kiểm thử tĩnh – 80 phút

Từ khóa

bất thường, rà soát chính thức (formal review), rà soát không chính thức (informal review), đánh giá chuyên sâu (inspection), kiểm thử động, kiểm thử tĩnh, phân tích tĩnh, rà soát (review), rà soát kỹ thuật, rà soát có hướng dẫn (walkthrough)

Mục tiêu Học tập của Chương 3:

3.1 Cơ bản về Kiểm thử tĩnh

FL-3.1.1 (K1) Nhận biết các loại sản phẩm công việc có thể được rà soát bằng kiểm thử tĩnh

FL-3.1.2 (K2) Giải thích giá trị của kiểm thử tĩnh

FL-3.1.3 (K2) So sánh và phân biệt kiểm thử tĩnh với kiểm thử động

3.2 Phản hồi và Quy trình Rà soát

FL-3.2.1 (K1) Xác định lợi ích của việc phản hồi sớm và thường xuyên từ các bên liên quan

FL-3.2.2 (K2) Tóm tắt các hoạt động trong quy trình rà soát

FL-3.2.3 (K1) Ghi nhớ các vai trò chính và trách nhiệm được giao khi thực hiện rà soát

FL-3.2.4 (K2) So sánh và phân biệt các loại rà soát khác nhau

FL-3.2.5 (K1) Ghi nhớ các yếu tố góp phần giúp rà soát thành công

3.1. Cơ bản về Kiểm thử tĩnh

Khác với kiểm thử động, trong kiểm thử tĩnh, phần mềm đang được kiểm thử không cần phải được thực thi (chạy). Mã nguồn, đặc tả quy trình, đặc tả kiến trúc hệ thống, hoặc các sản phẩm công việc khác đều có thể được đánh giá thông qua việc kiểm tra thủ công (ví dụ: rà soát) hoặc bằng công cụ hỗ trợ (ví dụ: phân tích tĩnh). Mục tiêu kiểm thử của kiểm thử tĩnh bao gồm cải thiện chất lượng, phát hiện lỗi, và đánh giá một số đặc tính chất lượng như tính dễ đọc, tính đầy đủ, tính đúng đắn, khả năng kiểm thử, và tính nhất quán. Kiểm thử tĩnh có thể được áp dụng trong việc xác minh và thẩm định.

Tester, các đại diện nghiệp vụ (ví dụ: PO, BA, v.v.), và lập trình viên cùng làm việc với nhau trong các hoạt động example mapping (là một kỹ thuật dùng ví dụ cụ thể để làm rõ hơn các user story, quy tắc nghiệp vụ, và các trường hợp cần xem xét), viết user story theo phương pháp cộng tác, và các buổi tinh chỉnh backlog để đảm bảo user story và các sản phẩm liên quan đáp ứng các tiêu chí đã xác định, ví dụ như Definition of Ready (xem mục 5.1.3). Các kỹ thuật rà soát có thể được áp dụng để đảm bảo user story đầy đủ, dễ hiểu, và kèm theo các tiêu chí chấp nhận có thể kiểm thử được. Bằng cách đặt đúng câu hỏi, tester có thể khám phá, chất vấn, và giúp hoàn thiện các user story được đề xuất.

Phân tích tĩnh có thể giúp phát hiện vấn đề trước khi kiểm thử động được thực hiện, đồng thời thường đòi hỏi ít công sức hơn vì không cần viết test case, và thường được thực hiện bằng công cụ (xem chương 6). Phân tích tĩnh thường được tích hợp trong các chương trình CI (xem mục 2.1.4). Mặc dù chủ yếu được dùng để phát hiện các lỗi cụ thể trong mã nguồn, phân tích tĩnh cũng được sử dụng để đánh giá khả năng bảo trì và bảo mật. Các công cụ kiểm tra chính tả và công cụ đánh giá độ dễ đọc cũng là một ví dụ của công cụ phân tích tĩnh.

3.1.1. Các Sản phẩm Công việc Có thể được Kiểm tra bằng Kiểm thử tĩnh

Hầu như bất kỳ sản phẩm công việc nào cũng có thể được rà soát bằng kiểm thử tĩnh. Ví dụ: tài liệu đặc tả yêu cầu, mã nguồn chương trình, kế hoạch kiểm thử, test case, các mục trong product backlog, test charter, tài liệu dự án, hợp đồng, và các mô hình (như mô hình dữ liệu, mô hình kiến trúc, v.v.).

Bất kỳ sản phẩm công việc nào mà có thể đọc và hiểu được đều có thể là đối tượng của hoạt động rà soát. Tuy nhiên, đối với phân tích tĩnh, các sản phẩm công việc cần có cấu trúc để có thể được kiểm tra dựa trên đó (ví dụ: các mô hình, mã nguồn, hoặc văn bản có cú pháp chính thức như JSON).

Các sản phẩm công việc không phù hợp cho kiểm thử tĩnh bao gồm những tài liệu khó diễn giải đối với con người và không nên được phân tích bằng công cụ (ví dụ: mã nguồn chạy được của bên thứ ba vì lý do pháp lý).

3.1.2. Giá trị của Kiểm thử tĩnh

Kiểm thử tĩnh có thể giúp phát hiện lỗi ngay trong những giai đoạn sớm nhất trong vòng đời phát triển phần mềm, qua đó đáp ứng nguyên tắc kiểm thử sớm (xem mục 1.3). Nó cũng có thể giúp phát hiện các lỗi mà kiểm thử động không thể phát hiện được (ví dụ: các đoạn mã không bao giờ được thực thi, không tuân thủ các giải pháp thiết kế đã được chuẩn hóa (design pattern), hoặc lỗi trong các sản phẩm công việc không thực thi như tài liệu mô tả).

Kiểm thử tĩnh cung cấp khả năng đánh giá chất lượng và tạo sự tin tưởng vào các sản phẩm công việc. Bằng cách xác minh các yêu cầu đã được ghi chép, các bên liên quan cũng có thể đảm bảo rằng các yêu cầu này đã phản ánh đúng nhu cầu thực tế của họ hay chưa. Vì kiểm thử tĩnh có thể được thực hiện sớm trong SDLC, giúp các bên liên quan có thể thống nhất cách hiểu. Đồng thời, việc giao tiếp giữa các bên liên quan cũng được cải thiện. Vì lý do này, nên có sự tham gia của nhiều bên liên quan khác nhau trong kiểm thử tĩnh.

Mặc dù việc thực hiện các hoạt động rà soát có thể tốn kém, nhưng tổng chi phí của dự án thường sẽ thấp hơn so với khi không thực hiện rà soát, vì sẽ cần ít thời gian và công sức hơn để sửa lỗi ở các giai đoạn về sau của dự án.

Một số lỗi cụ thể trong mã nguồn có thể được phát hiện hiệu quả hơn bằng phân tích tĩnh so với kiểm thử động, thường dẫn đến cả việc giảm số lượng lỗi trong mã nguồn và giảm tổng công sức phát triển phần mềm.

3.1.3. Sự Khác nhau giữa Kiểm thử tĩnh và Kiểm thử động

Kiểm thử tĩnh và kiểm thử động là hai phương pháp bổ trợ cho nhau. Chúng có một số mục tiêu tương tự, chẳng hạn như hỗ trợ phát hiện lỗi trong các sản phẩm công việc (xem mục 1.1.1), tuy nhiên cũng có một số khác biệt như sau:

- Kiểm thử tĩnh và kiểm thử động (thông qua việc phân tích các sự cố) đều có thể dẫn đến việc phát hiện lỗi, tuy nhiên có những loại lỗi chỉ có thể được phát hiện bằng kiểm thử tĩnh hoặc chỉ bằng kiểm thử động
- Kiểm thử tĩnh phát hiện lỗi trực tiếp, trong khi kiểm thử động phát hiện sự cố, từ đó những lỗi liên quan có thể được xác định thông qua các hoạt động phân tích tiếp theo
- Kiểm thử tĩnh có thể dễ dàng phát hiện các lỗi trong những phần mã nguồn hiếm khi được thực thi hoặc khó tiếp cận bằng kiểm thử động
- Kiểm thử tĩnh có thể áp dụng được cho cả các sản phẩm công việc không có mã thực thi (như tài liệu), trong khi kiểm thử động chỉ có thể áp dụng được cho các sản phẩm công việc có mã thực thi (như mã nguồn chương trình).
- Kiểm thử tĩnh có thể được sử dụng để đánh giá các đặc tính chất lượng không phụ thuộc vào việc thực thi mã nguồn (ví dụ: khả năng bảo trì), trong khi kiểm thử động có thể được sử dụng để đánh giá các đặc tính chất lượng phụ thuộc vào việc thực thi mã nguồn (ví dụ: hiệu năng)

Các lỗi điển hình dễ phát hiện hơn và/hoặc ít tốn kém hơn khi thực hiện kiểm thử tĩnh bao gồm:

- Lỗi trong yêu cầu (ví dụ: không nhất quán, mơ hồ, mâu thuẫn, thiếu sót, không chính xác, trùng lặp)
- Lỗi trong thiết kế (ví dụ: cấu trúc cơ sở dữ liệu không hiệu quả, việc mô-đun hoá kém)
- Một số loại lỗi lập trình (ví dụ: biến chưa được khai báo, biến chưa được khởi tạo giá trị, đoạn mã không thể thực thi được hoặc bị trùng lặp, độ phức tạp mã nguồn quá cao)
- Sai lệch so với tiêu chuẩn (ví dụ: không tuân thủ quy ước đặt tên trong tiêu chuẩn lập trình)
- Đặc tả giao diện không chính xác (ví dụ: số lượng, kiểu dữ liệu, hoặc thứ tự các tham số không khớp)
- Một số loại lỗ hổng bảo mật cụ thể (ví dụ: buffer overflow – tràn bộ đệm)
- Thiếu sót hoặc bao phủ không đủ so với yêu cầu (ví dụ: thiếu test cho một tiêu chí chấp nhận)

3.2. Phản hồi và Quy trình Rà soát

3.2.1. Lợi ích của Phản hồi Sớm và Thường xuyên từ Các Bên liên quan

Phản hồi sớm và thường xuyên cho phép truyền đạt sớm các vấn đề tiềm ẩn về chất lượng. Nếu mức độ tham gia của các bên liên quan trong vòng đời phát triển phần mềm thấp, sản phẩm đang được phát triển có thể không đáp ứng được tầm nhìn ban đầu hoặc hiện tại của các bên liên quan. Việc không cung cấp đúng những gì các bên liên quan mong muốn, có thể dẫn đến việc phải làm lại tốn kém, trễ tiến độ, đổ lỗi lẫn nhau, và thậm chí có thể dẫn đến thất bại toàn bộ dự án.

Phản hồi thường xuyên từ các bên liên quan trong suốt quá trình phát triển phần mềm, có thể giúp ngăn ngừa những hiểu sai về yêu cầu và đảm bảo rằng các thay đổi yêu cầu được hiểu đúng và triển khai sớm hơn. Điều này giúp đội ngũ phát triển hiểu rõ hơn về những gì họ đang xây dựng. Đồng thời, cho phép họ tập trung vào những tính năng mang lại giá trị cao nhất cho các bên liên quan, và các chức năng có tác động tích cực nhất đến những rủi ro đã được xác định.

3.2.2. Các Hoạt động trong Quy trình Rà roát

Tiêu chuẩn ISO/IEC 20246 định nghĩa một quy trình rà soát tổng quát, cung cấp một khuôn khổ có cấu trúc nhưng linh hoạt, từ đó có thể tùy biến (điều chỉnh) thành quy trình rà soát cụ thể phù hợp với từng tình huống. Nếu quá trình rà soát được yêu cầu tính chính thức cao hơn, thì sẽ cần thực hiện nhiều nhiệm vụ và hoạt động hơn.

Kích thước của nhiều loại sản phẩm công việc thường quá lớn để có thể hoàn thành trong một lần rà soát. Do đó, quá trình rà soát có thể được thực hiện nhiều lần để hoàn thành việc rà soát toàn bộ sản phẩm công việc đó.

Các hoạt động trong quy trình rà soát bao gồm:

- **Lập kế hoạch.** Trong giai đoạn lập kế hoạch, phạm vi rà soát sẽ được xác định, bao gồm: mục đích của việc rà soát, sản phẩm công việc cần được xem xét, các đặc tính chất lượng cần được đánh giá, các khu vực cần tập trung, tiêu chí kết thúc rà soát, các thông tin hỗ trợ như tiêu chuẩn áp dụng, nguồn lực cần thiết và khung thời gian thực hiện rà soát.
- **Khởi tạo rà soát.** Trong giai đoạn khởi tạo rà soát, mục tiêu là đảm bảo rằng tất cả những người tham gia và mọi yếu tố liên quan đều đã sẵn sàng để bắt đầu quá trình rà soát. Điều này bao gồm việc đảm bảo rằng mỗi người tham gia đều có quyền truy cập (tiếp cận) sản phẩm công việc cần được xem xét, hiểu rõ vai trò và trách nhiệm của mình, đồng thời nhận được đầy đủ các thông tin và tài liệu cần thiết để thực hiện việc rà soát.
- **Rà soát cá nhân.** Mỗi người rà soát thực hiện việc rà soát độc lập để đánh giá chất lượng sản phẩm công việc, đồng thời xác định các bất thường, khuyến nghị, và câu hỏi bằng cách áp dụng một hoặc nhiều kỹ thuật rà soát (ví dụ: rà soát dựa trên checklist, rà soát theo kịch bản). Tiêu chuẩn ISO/IEC 20246 cung cấp thêm chi tiết về các kỹ thuật rà soát khác nhau. Người rà soát phải ghi lại tất cả các bất thường, khuyến nghị, và câu hỏi mà họ phát hiện.
- **Trao đổi và phân tích.** Vì các bất thường được xác định trong quá trình rà soát không nhất thiết đều là lỗi, nên tất cả các bất thường này cần được phân tích và thảo luận. Đối với từng bất thường, cần đưa ra quyết định về trạng thái của nó, người chịu trách nhiệm, và các hành động cần thực hiện. Thông thường việc này được thực hiện trong một cuộc họp rà soát, trong đó các thành viên sẽ xác định mức độ chất lượng của sản phẩm công việc đã được rà soát, và quyết định các hành động cần thiết tiếp theo. Có thể cần thực hiện một lần rà soát bổ sung để hoàn tất các hành động đã đề ra.
- **Khắc phục và báo cáo.** Cần tạo báo cáo lỗi riêng cho từng lỗi để có thể theo dõi các hành động khắc phục. Khi các tiêu chí kết thúc rà soát đã được đáp ứng, sản phẩm công việc đó có thể được chấp nhận. Kết quả của quá trình rà soát sẽ được báo cáo lại.

3.2.3. Vai trò và Trách nhiệm trong các Hoạt động Rà soát

Các hoạt động rà soát liên quan đến nhiều bên liên quan, những người này có thể đảm nhận một hoặc nhiều vai trò khác nhau. Các vai trò chính và trách nhiệm tương ứng bao gồm:

- Quản lý – quyết định nội dung cần được rà soát và cung cấp các nguồn lực cần thiết, chẳng hạn như nhân sự và thời gian cho hoạt động rà soát
- Tác giả – tạo ra và thực hiện sửa chữa sản phẩm công việc đang được rà soát
- Điều phối viên – đảm bảo việc tổ chức và vận hành hiệu quả các cuộc họp rà soát, bao gồm điều phối thảo luận, quản lý thời gian, và duy trì môi trường rà soát an toàn để mọi người có thể tự do phát biểu
- Thư ký – tổng hợp các bất thường được phát hiện những bởi người rà soát và ghi lại các thông tin thảo luận trong buổi rà soát, như các quyết định và các bất thường mới phát sinh trong cuộc họp phân tích rà soát
- Người rà soát – thực hiện việc rà soát. Người rà soát có thể là thành viên trong dự án, các chuyên gia trong lĩnh vực, hoặc bất kỳ bên liên quan nào khác
- Trưởng nhóm rà soát – chịu trách nhiệm tổng thể cho hoạt động rà soát, bao gồm việc quyết định ai sẽ tham gia và tổ chức thời gian, địa điểm diễn ra buổi rà soát

Ngoài ra, còn có thể có các vai trò khác được mô tả trong tiêu chuẩn ISO/IEC 20246.

3.2.4. Các Loại Rà soát

Có nhiều loại rà soát khác nhau, từ rà soát không chính thức đến rà soát chính thức. Mức độ chính thức cần thiết phụ thuộc vào các yếu tố như: SDLC đang được áp dụng, mức độ trưởng thành của quy trình phát triển phần mềm, mức độ quan trọng và độ phức tạp của sản phẩm công việc cần được rà soát, các yêu cầu pháp lý hoặc quy định liên quan, và nhu cầu đảm bảo khả năng đánh giá tuân thủ. Cùng một sản phẩm công việc có thể được thực hiện qua nhiều loại rà soát khác nhau, ví dụ: bắt đầu bằng một lần rà soát không chính thức, sau đó thì rà soát chính thức.

Việc lựa chọn loại rà soát phù hợp là yếu tố then chốt để đạt được các mục tiêu rà soát mong muốn (xem mục 3.2.5). Việc lựa chọn này không chỉ dựa trên mục tiêu, mà còn phụ thuộc vào các yếu tố như nhu cầu dự án, nguồn lực sẵn có, loại sản phẩm công việc cần rà soát và rủi ro liên quan, lĩnh vực nghiệp vụ và văn hóa công ty.

Một số loại rà soát phổ biến gồm:

- **Rà soát không chính thức.** Rà soát không chính thức (informal review) không tuân theo một quy trình cố định và không yêu cầu ghi nhận kết quả rà soát một cách chính thức. Mục tiêu chính là phát hiện các bất thường.
- **Walkthrough.** Walkthrough (rà soát có hướng dẫn) do tác giả chủ trì, có thể phục vụ nhiều mục tiêu như: đánh giá chất lượng và xây dựng sự tin tưởng đối với sản phẩm công việc, đào tạo người rà soát, nhằm đạt được sự đồng thuận, tạo ra ý tưởng mới, thúc đẩy và hỗ trợ tác giả hoàn thiện hơn, cũng như phát hiện các bất thường. Người rà soát có thể thực hiện rà soát cá nhân trước buổi walkthrough, nhưng điều này không bắt buộc.
- **Rà soát kỹ thuật.** Rà soát kỹ thuật (technical review) được thực hiện bởi những người rà soát có trình độ kỹ thuật phù hợp và do một điều phối viên dẫn dắt. Mục tiêu của rà soát kỹ thuật là đạt được sự đồng thuận và đưa ra quyết định liên quan đến một vấn đề kỹ thuật, đồng thời phát hiện các bất thường, đánh giá chất lượng và xây dựng sự tin tưởng đối với sản phẩm công việc, tạo ra ý tưởng mới, cũng như thúc đẩy và hỗ trợ tác giả hoàn thiện hơn.
- **Đánh giá chuyên sâu.** Vì kiểm tra chuyên sâu (inspection) là loại rà soát chính thức nhất, nên nó tuân theo đầy đủ quy trình tổng quát (xem mục 3.2.2). Mục tiêu chính là tối đa hoá việc phát hiện các bất thường. Các mục tiêu khác bao gồm đánh giá chất lượng, xây dựng sự tin tưởng đối với sản phẩm công việc, và thúc đẩy cũng như hỗ trợ tác giả hoàn thiện hơn. Các chỉ số được thu thập và sử dụng để cải tiến SDLC và cả chính quy trình đánh giá chuyên sâu. Trong đánh giá chuyên sâu, tác giả không được nắm giữ vai trò là trưởng nhóm rà soát hay thư ký.

3.2.5. Các Yếu tố Thành công của Hoạt động Rà soát

Một số yếu tố quyết định sự thành công của các hoạt động rà soát, bao gồm:

- Xác định mục tiêu và các tiêu chí kết thúc rõ ràng và có thể đo lường được. Không được xem việc đánh giá các thành viên tham gia là một mục tiêu rà soát
- Lựa chọn loại rà soát phù hợp để đạt được các mục tiêu đã đề ra, đồng thời phù hợp với loại sản phẩm công việc cần được rà soát, các thành viên tham gia, nhu cầu dự án và bối cảnh cụ thể
- Thực hiện rà soát theo từng phần nhỏ, nhằm giúp người rà soát không bị mất tập trung trong quá trình rà soát cá nhân và/hoặc trong cuộc họp trao đổi và phân tích (nếu có)
- Cung cấp phản hồi từ những hoạt động rà soát cho các bên liên quan và chính tác giả để họ có thể cải thiện sản phẩm và các hoạt động của mình trong tương lai (xem mục 3.2.1)
- Sắp xếp đủ thời gian cho các thành viên tham gia để họ chuẩn bị tốt hơn cho hoạt động rà soát
- Cần có sự hỗ trợ từ nhóm quản lý đối với quy trình rà soát
- Biến hoạt động rà soát trở thành một phần văn hóa của tổ chức, nhằm thúc đẩy việc học hỏi và cải tiến quy trình rà soát

- Cung cấp đào tạo đầy đủ cho tất cả các thành viên tham gia để họ hiểu cách làm và thực hiện đúng vai trò của mình
- Điều phối hiệu quả các cuộc họp trao đổi và phân tích trong quá trình rà soát

4. Phân tích và thiết kế test case – 390 phút

Từ khóa

bao phủ câu lệnh, độ bao phủ, độ bao phủ nhánh, đoán lỗi, phát triển hướng kiểm thử chấp nhận (ATDD), phân tích giá trị biên, phân vùng tương đương, phương pháp kiểm thử dựa trên sự cộng tác, kiểm thử dựa trên bảng quyết định, kiểm thử dựa trên checklist, kiểm thử dựa vào chuyển đổi trạng thái, kiểm thử khám phá, kỹ thuật kiểm thử, kỹ thuật kiểm thử hộp đen, kỹ thuật kiểm thử hộp trắng, kỹ thuật kiểm thử dựa trên kinh nghiệm, hạng mục bao phủ, tiêu chí chấp nhận

Mục tiêu Học tập của Chương 4:

4.1 Tổng quan về Kỹ thuật Kiểm thử

FL-4.1.1 (K2) Phân biệt các kỹ thuật kiểm thử hộp đen, các kỹ thuật kiểm thử hộp trắng, và các kỹ thuật kiểm thử dựa trên kinh nghiệm

4.2 Các Kỹ thuật Kiểm thử Hộp đen (black-box testing)

FL-4.2.1 (K3) Áp dụng phân vùng tương đương để thiết kế test case

FL-4.2.2 (K3) Áp dụng phân tích giá trị biên để thiết kế test case

FL-4.2.3 (K3) Áp dụng kiểm thử dựa vào bảng quyết định để thiết kế test case

FL-4.2.4 (K3) Áp dụng kiểm thử dựa vào chuyển đổi trạng thái để thiết kế test case

4.3 Các Kỹ thuật Kiểm thử Hộp trắng (white-box testing)

FL-4.3.1 (K2) Giải thích kiểm thử câu lệnh

FL-4.3.2 (K2) Giải thích kiểm thử nhánh

FL-4.3.3 (K2) Giải thích giá trị của kiểm thử hộp trắng

4.4 Kỹ thuật Kiểm thử dựa trên Kinh nghiệm

FL-4.4.1 (K2) Giải thích đoán lỗi

FL-4.4.2 (K2) Giải thích kiểm thử khám phá

FL-4.4.3 (K2) Giải thích kiểm thử dựa trên checklist

4.5. Các Phương pháp Kiểm thử Dựa trên sự Cộng tác

FL-4.5.1 (K2) Giải thích cách viết user story thông qua sự cộng tác với lập trình viên và đại diện nghiệp vụ

FL-4.5.2 (K2) Phân loại các lựa chọn khác nhau để viết tiêu chí chấp nhận (acceptance criteria)

FL-4.5.3 (K3) Áp dụng phát triển hướng kiểm thử chấp nhận (ATDD) để thiết kế test case

4.1. Tổng quan về Kỹ thuật Kiểm thử

Các kỹ thuật kiểm thử hỗ trợ tester trong hoạt động phân tích kiểm thử (kiểm thử cái gì) và thiết kế test case (kiểm thử như thế nào). Các kỹ thuật kiểm thử giúp xây dựng một tập hợp test case tương đối nhỏ nhưng vẫn bao phủ đủ, theo một cách có hệ thống. Các kỹ thuật kiểm thử cũng hỗ trợ tester xác định các khía cạnh kiểm thử (test condition), nhận diện các hạng mục bao phủ, và xác định dữ liệu kiểm thử trong quá trình phân tích yêu cầu và thiết kế test case. Thông tin chi tiết hơn về các kỹ thuật kiểm thử có thể được tìm thấy trong tiêu chuẩn ISO/IEC/IEEE 29119-4, cũng như trong các tài liệu Beizer 1990, Craig 2002, Copeland 2004, Koomen 2006, Jorgensen 2014, Ammann 2016, và Forgács 2019.

Trong đề cương này, các kỹ thuật kiểm thử được phân loại thành các nhóm: kỹ thuật kiểm thử hộp đen, kỹ thuật kiểm thử hộp trắng, và kỹ thuật kiểm thử dựa trên kinh nghiệm.

Các kỹ thuật kiểm thử hộp đen (còn được gọi là kỹ thuật dựa trên đặc tả) dựa trên việc phân tích hành vi được mô tả của đối tượng kiểm thử mà không tham chiếu đến cấu trúc bên trong của đối tượng đó. Do đó, các test case độc lập với cách phần mềm được lập trình. Vì vậy, nếu chương trình có thay đổi nhưng hành vi yêu cầu vẫn giữ nguyên, thì các test case vẫn còn giá trị sử dụng.

Các kỹ thuật kiểm thử hộp trắng (còn được gọi là kỹ thuật dựa trên cấu trúc) dựa trên việc phân tích cấu trúc bên trong và cách xử lý của đối tượng kiểm thử. Do các test case phụ thuộc vào cách phần mềm được thiết kế, test case chỉ có thể được tạo sau khi đã có thiết kế hoặc mã nguồn chương trình.

Các kỹ thuật kiểm thử dựa trên kinh nghiệm tận dụng hiệu quả kiến thức và kinh nghiệm của tester trong việc thiết kế và thực thi test case. Hiệu quả của các kỹ thuật này phụ thuộc rất lớn vào kỹ năng và kinh nghiệm của tester. Các kỹ thuật kiểm thử dựa trên kinh nghiệm có thể giúp phát hiện những lỗi mà kỹ thuật kiểm thử hộp đen và hộp trắng có thể bỏ sót. Vì vậy, các kỹ thuật này mang tính bổ trợ cho kỹ thuật kiểm thử hộp đen và hộp trắng.

4.2. Kỹ thuật Kiểm thử Hộp đen

Các kỹ thuật kiểm thử hộp đen phổ biến được trình bày trong các phần sau gồm:

- Phân vùng tương đương (Equivalence Partitioning)
- Phân tích giá trị biên (Boundary Value Analysis)
- Kiểm thử dựa vào bảng quyết định (Decision Table Testing)
- Kiểm thử dựa vào chuyển đổi trạng thái (State Transition Testing)

4.2.1. Phân vùng Tương đương

Phân vùng tương đương (EP) chia dữ liệu thành các vùng (gọi là các vùng tương đương hoặc lớp tương đương) dựa trên giả định rằng tất cả các phần tử trong cùng một phân vùng sẽ được đối tượng kiểm thử xử lý theo một cách giống nhau. Cơ sở lý thuyết của kỹ thuật này là, nếu kiểm tra một giá trị bất kỳ trong một vùng tương đương phát hiện ra lỗi, thì lỗi đó cũng sẽ được phát hiện bởi các test case kiểm tra các giá trị khác trong cùng phân vùng. Do đó, chỉ cần một test case cho mỗi vùng tương đương là đủ.

Có thể xác định các phân vùng tương đương cho bất kỳ phần tử dữ liệu nào liên quan đến đối tượng kiểm thử, chẳng hạn như dữ liệu đầu vào, dữ liệu đầu ra, các mục cấu hình, giá trị bên trong phần mềm, giá trị liên quan đến thời gian, và tham số giao diện. Các vùng tương đương có thể là liên tục hoặc rời rạc, có thứ tự hoặc không có thứ tự, hữu hạn hoặc vô hạn. Các vùng tương đương không được chồng lấn nhau và phải là cá tập không rỗng (nghĩa là có ít nhất một giá trị).

Với các đối tượng kiểm thử đơn giản, việc xác định phân vùng tương đương có thể khá dễ dàng. Tuy nhiên, trong thực tế, việc hiểu cách đối tượng kiểm thử xử lý các giá trị khác nhau thường phức tạp. Vì vậy, việc phân vùng cần được thực hiện một cách cẩn thận.

Vùng tương đương chứa các giá trị hợp lệ được gọi là phân vùng hợp lệ. Vùng tương đương chứa các giá trị không hợp lệ được gọi là phân vùng không hợp lệ. Cách định nghĩa về giá trị hợp lệ và không hợp lệ có thể khác nhau trong các nhóm dự án và tổ chức khác nhau. Ví dụ, giá trị hợp lệ có

thể được hiểu là những giá trị mà đối tượng kiểm thử sẽ chấp nhận xử lý, hoặc là những giá trị mà đặc tả đã định nghĩa cách xử lý. Giá trị không hợp lệ có thể được hiểu là những giá trị sẽ bị bỏ qua hoặc bị từ chối xử lý bởi đối tượng kiểm thử, hoặc là những giá trị không được định nghĩa cách xử lý trong đặc tả của đối tượng kiểm thử.

Trong kỹ thuật EP, các hạng mục bao phủ chính là các vùng tương đương. Để đạt được độ bao phủ 100% với kỹ thuật này, các test case phải bao phủ tất cả các vùng tương đương đã xác định (bao gồm cả phân vùng không hợp lệ), tức là mỗi vùng tương đương phải được kiểm tra ít nhất một lần. Độ bao phủ được đo bằng số lượng vùng tương đương được kiểm thử bởi các test case, chia cho tổng số vùng tương đương đã xác định, và được biểu diễn dưới dạng phần trăm (%).

Nhiều hạng mục kiểm thử có một tập hợp nhiều tập vùng tương đương (ví dụ: có nhiều tham số đầu vào), điều này có nghĩa là một test case sẽ bao phủ các vùng tương đương từ nhiều tập khác nhau. Tiêu chí bao phủ đơn giản nhất trong trường hợp có nhiều tập vùng tương đương là bao phủ Each Choice (Ammann 2016). Tiêu chí này yêu cầu các test case phải bao phủ mỗi vùng tương đương trong từng tập phân vùng ít nhất một lần. Bao phủ Each Choice không xét đến các tổ hợp các phân vùng tương đương.

4.2.2. Phân tích Giá trị Biên

Phân tích giá trị biên (BVA) là kỹ thuật kiểm thử dựa trên việc kiểm tra các giá trị tại ranh giới (biên) của các phân vùng tương đương. Do đó, BVA chỉ có thể áp dụng cho các vùng tương đương có tính thứ tự. Giá trị nhỏ nhất và lớn nhất của một vùng tương đương chính là các giá trị biên của nó. Trong kỹ thuật BVA, nếu hai phần tử thuộc cùng một vùng tương đương, thì tất cả các phần tử nằm giữa chúng cũng phải thuộc về cùng phân vùng đó.

BVA tập trung vào các giá trị biên vì lập trình viên thường dễ mắc lỗi tại những điểm này. Các lỗi điển hình được phát hiện bởi BVA thường xảy ra khi các ranh giới được đặt sai (như lệch lên trên hoặc xuống dưới so với vị trí dự kiến) hoặc bị bỏ sót hoàn toàn.

Trong đề cương này, BVA được trình bày với hai biến thể: BVA 2 giá trị và BVA 3 giá trị. Hai biến thể này khác nhau về số lượng hạng mục bao phủ trên mỗi biên cần được kiểm tra để đạt 100% độ bao phủ.

BVA 2 giá trị (Craig 2002, Myers 2011), với mỗi biên có hai hạng mục bao phủ: giá trị biên của phân vùng đó và giá trị biên gần nhất của phân vùng liền kề.

Để đạt được 100% độ bao phủ BVA 2 giá trị, các test case phải bao phủ tất cả các hạng mục bao phủ, tức là tất cả các giá trị biên đã xác định. Độ bao phủ được đo bằng số lượng giá trị biên đã được kiểm tra bởi các test case, chia cho tổng số giá trị biên đã xác định, và được biểu diễn dưới dạng phần trăm (%).

BVA 3 giá trị (Koomen 2006, O'Regan 2019), với mỗi biên có ba hạng mục bao phủ: giá trị biên của phân vùng đó và hai giá trị lân cận của nó. Do đó, với kỹ thuật BVA 3 giá trị, một số hạng mục bao phủ có thể không phải là giá trị biên.

Để đạt 100% độ bao phủ BVA 3 giá trị, các test case phải bao phủ tất cả các hạng mục bao phủ, tức là tất cả các giá trị biên và giá trị lân cận của chúng.

Độ bao phủ được đo bằng số lượng các giá trị biên và giá trị lân cận đã được kiểm tra bởi các test case, chia cho tổng số các giá trị biên và giá trị lân cận đã xác định, và được biểu diễn dưới dạng phần trăm (%).

BVA 3 giá trị chặt chẽ hơn BVA 2 giá trị vì nó có thể giúp phát hiện những lỗi mà BVA 2 giá trị có thể bỏ sót. Ví dụ, nếu điều kiện "if (x ≤ 10) …" được lập trình sai thành "if (x = 10) …", thì các giá trị kiểm thử được xác định từ kỹ thuật BVA 2 giá trị (x = 10, x = 11) sẽ không phát hiện được lỗi này. Tuy nhiên, giá trị x = 9 (từ BVA 3 giá trị) có khả năng phát hiện lỗi này.

4.2.3. Kiểm thử Dựa vào Bảng Quyết định

Bảng quyết định được sử dụng để kiểm thử chương trình mà có các yêu cầu quy định các tổ hợp điều kiện khác nhau sẽ tạo ra các kết quả khác nhau. Bảng quyết định là một cách hiệu quả để biểu diễn lôgic phức tạp, chẳng hạn như các quy tắc nghiệp vụ.

Trong lúc tạo bảng quyết định, tester cần phải xác định được các điều kiện (condition) và các hành động tương ứng của hệ thống (action). Từ đó, sử dụng chúng để tạo thành các hàng trong bảng. Mỗi cột trong bảng tương ứng với một quyết định nghiệp vụ, nó thể hiện một tổ hợp điều kiện duy nhất cùng với các hành động liên quan. Trong bảng quyết định dạng hạn chế (limited-entry), tất cả các giá trị của điều kiện và hành động (trừ những giá trị không liên quan hoặc không khả thi; xem bên dưới) được biểu diễn dưới dạng Boolean (đúng/sai). Ngược lại, trong bảng quyết định dạng mở rộng (extended-entry), một số hoặc tất cả các điều kiện và hành động có thể nhận nhiều loại giá trị (ví dụ: các khoảng giá trị, vùng tương đương, hoặc các giá trị rời rạc).

Ký hiệu được sử dụng cho điều kiện như sau: “T” (true) nghĩa là điều kiện được thỏa mãn. “F” (false) là điều kiện không được thỏa mãn. “–” là giá trị của điều kiện không ảnh hưởng đến kết quả hành động. Và “N/A” là điều kiện không áp dụng trong tổ hợp đó. Đối với hành động: “X” là hành động xảy ra. Ô để trống là hành động không xảy ra. Các ký hiệu khác cũng có thể được sử dụng.

Một bảng quyết định đầy đủ có đủ số cột để bao phủ mọi tổ hợp điều kiện. Bảng có thể được đơn giản hóa bằng cách loại bỏ các cột chứa những tổ hợp điều kiện không khả thi. Ngoài ra, bảng cũng có thể được tối giản bằng cách gộp các cột, mà trong đó một số điều kiện không ảnh hưởng đến kết quả, thành một cột duy nhất. Các thuật toán tối giản bảng quyết định không nằm trong phạm vi của đề cương này.

Trong kiểm thử dựa vào bảng quyết định, các hạng mục bao phủ chính là các cột chứa những tổ hợp điều kiện khả thi. Để đạt 100% độ bao phủ, bộ test case phải kiểm tra tất cả các cột trong bảng. Độ bao phủ được đo bằng số lượng cột đã được kiểm tra bởi các test case, chia cho tổng số cột khả thi, và được biểu diễn dưới dạng phần trăm (%).

Ưu điểm của kỹ thuật kiểm thử dựa vào bảng quyết định là cung cấp một cách tiếp cận có hệ thống để xác định tất cả các tổ hợp điều kiện, bao gồm cả những tổ hợp có thể dễ bị bỏ sót. Nó cũng giúp phát hiện những thiếu sót hoặc mâu thuẫn trong yêu cầu. Tuy nhiên, khi có nhiều điều kiện, việc kiểm tra tất cả các tổ hợp có thể tốn nhiều thời gian, vì số lượng kết hợp tăng theo cấp số nhân so với số lượng điều kiện. Trong trường hợp này, để giảm số lượng kết hợp cần kiểm tra, chúng ta có thể sử dụng bảng quyết định đã được tối giản hoặc áp dụng phương pháp kiểm thử dựa vào rủi ro.

4.2.4. Kiểm thử Dựa vào Chuyển đổi Trạng thái

Sơ đồ trạng thái giúp mô hình hóa hành vi của hệ thống bằng cách thể hiện các trạng thái có thể có và các chuyển đổi trạng thái hợp lệ. Một chuyển đổi trạng thái được kích hoạt bởi một sự kiện (event), có thể được bổ sung bởi một điều kiện bảo vệ (guard condition). Các chuyển đổi trạng thái được giả định là xảy ra tức thời và đôi khi có thể dẫn đến việc phần mềm thực hiện một hành động nào đó (action). Cú pháp ghi nhãn cho các chuyển đổi trạng thái thường dùng là: “event [guard condition] / action”. Điều kiện bảo vệ và hành động có thể được lược bỏ nếu không tồn tại hoặc không liên quan đối với tester.

Bảng trạng thái là một mô hình tương đương với sơ đồ trạng thái. Trong bảng trạng thái, hàng biểu diễn các trạng thái, và cột biểu diễn các sự kiện (kèm theo điều kiện bảo vệ nếu có). Các ô trong bảng biểu diễn các chuyển đổi trạng thái, bao gồm trạng thái đích đến và các hành động tương ứng (nếu được mô tả). Khác với sơ đồ trạng thái, bảng trạng thái thể hiện rõ các chuyển đổi trạng thái không hợp lệ, được biểu diễn bằng các ô trống.

Test case dựa trên sơ đồ trạng thái hoặc bảng trạng thái thường được biểu diễn dưới dạng một chuỗi các sự kiện, dẫn đến một chuỗi thay đổi trạng thái (và hành động, nếu có). Một test case có thể, và thường sẽ, bao phủ nhiều chuyển đổi trạng thái.

Có nhiều tiêu chí bao phủ trong kiểm thử chuyển đổi trạng thái. Đề cương này đề cập đến ba tiêu chí sau:

Với **bao phủ tất cả trạng thái**, hạng mục bao phủ là các trạng thái. Để đạt 100% độ bao phủ này, bộ test case phải đảm bảo tất cả các trạng thái đều được thực thi ít nhất một lần. Độ bao phủ được đo bằng số trạng thái đã được thực thi, chia cho tổng số trạng thái, và được biểu diễn dưới dạng phần trăm.

Với **bao phủ chuyển đổi trạng thái hợp lệ** (còn gọi là bao phủ 0-switch), hạng mục bao phủ là các chuyển đổi trạng thái hợp lệ riêng lẻ. Để đạt 100% độ bao phủ này, bộ test case phải thực thi tất cả các chuyển đổi trạng thái hợp lệ. Độ bao phủ được đo bằng số chuyển đổi trạng thái hợp lệ đã được thực thi bởi các test case, chia cho tổng số chuyển đổi trạng thái hợp lệ, và được biểu diễn dưới dạng phần trăm.

Với **bao phủ tất cả chuyển đổi trạng thái**, hạng mục bao phủ là tất cả các chuyển đổi trạng thái được thể hiện trong bảng trạng thái. Để đạt 100% độ bao phủ này, bộ test case phải thực thi tất cả các chuyển đổi trạng thái hợp lệ và cả các chuyển đổi trạng thái không hợp lệ. Chỉ nên có một chuyển đổi trạng thái không hợp lệ trong mỗi test case để tránh hiện tượng che giấu lỗi, tức là tình huống một lỗi xảy ra làm che khuất việc phát hiện lỗi khác. Độ bao phủ được đo bằng số chuyển đổi trạng thái hợp lệ và không hợp lệ đã được thực thi hoặc được cố gắng thực thi, chia cho tổng số chuyển đổi trạng thái hợp lệ và không hợp lệ, và được biểu diễn dưới dạng phần trăm.

Bao phủ tất cả trạng thái là tiêu chí yếu hơn so với bao phủ chuyển đổi trạng thái hợp lệ, vì thường có thể đạt được mà không cần thực thi mọi chuyển đổi trạng thái. Bao phủ chuyển đổi trạng thái hợp lệ là tiêu chí được sử dụng phổ biến nhất. Việc đạt được đầy đủ (100%) bao phủ chuyển đổi trạng thái hợp lệ đảm bảo đạt được bao phủ đầy đủ tất cả trạng thái. Việc đạt được bao phủ 100% mọi chuyển đổi trạng thái đảm bảo đạt được cả bao phủ mọi chuyển đổi trạng thái hợp lệ và trạng thái, và nên được coi đây là yêu cầu tối thiểu đối với các phần mềm quan trọng đối với nhiệm vụ và cần tính an toàn cao.

4.3. Kỹ thuật Kiểm thử Hộp trắng

Do tính phổ biến và đơn giản, phần này tập trung vào hai kỹ thuật kiểm thử hộp trắng liên quan đến mã nguồn:

- Kiểm thử câu lệnh (statement testing)
- Kiểm thử nhánh (branch testing)

Ngoài hai kỹ thuật trên, còn có nhiều kỹ thuật kiểm thử hộp trắng khác chặt chẽ hơn, được sử dụng trong các phần mềm cần tính an toàn cao, quan trọng đối với nhiệm vụ, hoặc hệ thống có tính toàn vẹn cao, nhằm đạt được mức độ bao phủ mã nguồn toàn diện hơn. Cũng có các kỹ thuật kiểm thử hộp trắng được sử dụng ở các mức kiểm thử cao hơn (ví dụ: kiểm thử API), hoặc sử dụng các loại bao phủ không liên quan trực tiếp đến mã nguồn (ví dụ: độ bao phủ neuron trong kiểm thử mạng nơ-ron). Những kỹ thuật này không được đề cập trong đề cương này.

4.3.1. Kiểm thử Câu lệnh và Độ bao phủ Câu lệnh

Trong kiểm thử câu lệnh, các hạng mục bao phủ là các câu lệnh có thể thực thi được trong mã nguồn. Mục đích của kỹ thuật này là thiết kế test case để thực thi các câu lệnh trong mã nguồn, cho đến khi đạt được mức độ bao phủ chấp nhận được. Bao phủ câu lệnh được đo bằng số lượng câu lệnh được thực thi bởi các test case hiện có, chia cho tổng số câu lệnh có thể thực thi trong mã nguồn, và được biểu diễn dưới dạng phần trăm.

Khi đạt được 100% độ bao phủ câu lệnh, điều này đảm bảo rằng tất cả các câu lệnh có thể thực thi trong mã nguồn đã được chạy qua ít nhất một lần. Cụ thể, điều này có nghĩa là nếu câu lệnh có lỗi được thực thi, những lỗi này có thể gây ra sự cố, từ đó chứng minh sự tồn tại của lỗi trong mã nguồn. Tuy nhiên, việc thực thi mọi câu lệnh không phải lúc nào cũng phát hiện được tất cả các lỗi trong mã nguồn. Ví dụ, có thể không phát hiện được các lỗi phụ thuộc vào dữ liệu (chẳng hạn như lỗi chia cho 0 chỉ xảy ra khi mẫu số bằng 0). Ngoài ra, việc đạt 100% bao phủ câu lệnh cũng không đảm bảo rằng toàn bộ lôgic quyết định đã được kiểm thử, vì có thể tất cả các nhánh (xem mục 4.3.2) trong mã nguồn chưa được thực thi.

4.3.2. Kiểm thử Nhánh và Độ bao phủ Nhánh

Nhánh là sự chuyển luồng điều khiển giữa hai nút trong đồ thị luồng điều khiển. Đồ thị này biểu diễn các chuỗi xử lý có thể xảy ra trong quá trình thực thi các câu lệnh trong mã nguồn của đối tượng kiểm thử. Mỗi sự chuyển luồng điều khiển có thể là vô điều kiện (tức là các đoạn mã chạy tuần tự) hoặc có điều kiện (tức là kết quả của một quyết định).

Trong kiểm thử nhánh, các hạng mục bao phủ là các nhánh. Mục tiêu của kỹ thuật này là thiết kế các test case để thực thi các nhánh trong mã nguồn cho đến khi đạt được mức bao phủ chấp nhận được. Độ bao phủ được đo bằng số lượng nhánh được thực thi bởi các test case hiện có, chia cho tổng số nhánh, và được biểu diễn dưới dạng phần trăm.

Khi đạt được 100% độ bao phủ nhánh, tất cả các nhánh trong mã nguồn, cả có điều kiện và vô điều kiện, đều được thực thi bởi các test case. Các nhánh có điều kiện thường tương ứng với kết quả đúng/sai của một quyết định “if...then”, một nhánh trong câu lệnh switch/case, hoặc quyết định kết thúc hay tiếp tục trong một vòng lặp (do, while, for, v.v...). Tuy nhiên, việc thực thi mọi nhánh bằng test case không phải lúc nào cũng phát hiện được lỗi. Ví dụ, có thể không phát hiện được các lỗi chỉ xảy ra khi mã nguồn được thực thi theo một luồng cụ thể nào đó.

Đạt độ bao phủ nhánh cũng đồng thời đạt độ bao phủ câu lệnh. Điều này có nghĩa là bất kỳ tập test case nào đạt 100% độ bao phủ nhánh, thì nó cũng sẽ đạt 100% độ bao phủ câu lệnh (nhưng ngược lại thì không chắc).

4.3.3. Giá trị của Kiểm thử Hộp trắng

Một điểm mạnh cơ bản mà tất cả các kỹ thuật kiểm thử hộp trắng đều có là toàn bộ chương trình phần mềm đều được xem xét trong quá trình kiểm thử. Điều này giúp phát hiện lỗi ngay cả khi đặc tả phần mềm còn mơ hồ, bị lỗi thời, hoặc chưa mô tả đầy đủ. Tuy nhiên, một điểm yếu tương ứng là nếu chương trình phần mềm không thực hiện một hoặc nhiều yêu cầu, thì việc kiểm thử hộp trắng có thể không phát hiện được các lỗi do thiếu sót này (Watson 1996).

Các kỹ thuật kiểm thử hộp trắng cũng có thể được sử dụng trong kiểm thử tĩnh (ví dụ: khi chạy thử code trên giấy – dry run). Chúng đặc biệt phù hợp để rà soát mã nguồn chưa sẵn sàng để thực thi (Hetzel 1988), mã giả (pseudocode), và các lôgic ở mức cao hoặc lôgic từ trên xuống (top-down) có thể được mô hình hóa bằng đồ thị luồng điều khiển.

Nếu chỉ thực hiện kiểm thử hộp đen, sẽ không cung cấp được thước đo về mức độ bao phủ mã nguồn thực tế. Các thước đo bao phủ trong kiểm thử hộp trắng cung cấp một cách đo lường khách quan về độ bao phủ, đồng thời cung cấp thông tin cần thiết để bổ sung thêm test case nhằm tăng độ bao phủ này, từ đó nâng cao mức độ tin cậy của mã nguồn.

4.4. Kỹ thuật Kiểm thử dựa trên Kinh nghiệm

Các kỹ thuật kiểm thử dựa trên kinh nghiệm phổ biến được trình bày trong các phần sau gồm:

- Đoán lỗi (error guessing)
- Kiểm thử khám phá (exploratory testing)
- Kiểm thử dựa trên checklist (checklist-based testing)

4.4.1. Đoán lỗi (error guessing)

Đoán lỗi là một kỹ thuật kiểm thử dùng để dự đoán sự xuất hiện của sai sót, lỗi, và các sự cố, dựa vào kiến thức và kinh nghiệm của của tester gồm:

- Cách hệ thống phần mềm đã hoạt động trong quá khứ
- Các loại sai sót mà lập trình viên thường mắc phải và các loại lỗi phát sinh từ những sai sót đó
- Các loại sự cố đã xảy ra trong những ứng dụng, hệ thống tương tự

Nói chung, sai sót, lỗi, và sự cố có thể liên quan đến: dữ liệu đầu vào (ví dụ: đầu vào đúng nhưng không được chấp nhận, tham số sai hoặc bị thiếu), dữ liệu đầu ra (ví dụ: sai định dạng, sai kết quả), lôgic nghiệp vụ (ví dụ: thiếu trường hợp, sử dụng sai toán tử), tính toán (ví dụ: sai toán hạng, sai phép tính), giao diện giữa các thành phần (ví dụ: tham số không khớp, kiểu dữ liệu không tương thích), dữ liệu (ví dụ: khởi tạo sai, sai kiểu dữ liệu).

Fault attacks (tấn công lỗi) là một cách triển khai thực tế của kỹ thuật đoán lỗi. Kỹ thuật này yêu cầu

tester tạo ra hoặc thu thập danh sách các sai sót, lỗi, hoặc sự cố có thể xảy ra, sau đó thiết kế test case nhằm phát hiện lỗi hoặc sự cố liên quan đến các sai sót đó. Danh sách này có thể được xây dựng dựa trên kinh nghiệm, dữ liệu về lỗi và sự cố hiện có, hoặc kiến thức chung về nguyên nhân gây lỗi phần mềm.

Xem thêm (Whittaker 2002, Whittaker 2003, Andrews 2006) để có thêm thông tin về đoán lỗi và fault attacks.

4.4.2. Kiểm thử Khám phá (exploratory testing)

Trong kiểm thử khám phá, việc thiết kế, thực thi test case, và đánh giá kết quả được thực hiện đồng thời, cùng lúc đó tester cũng sẽ tìm hiểu thêm về đối tượng kiểm thử. Hoạt động kiểm thử được sử dụng để giúp tester hiểu rõ hơn về đối tượng kiểm thử, khám phá nó sâu hơn bằng các test case có mục tiêu, và nghĩ ra các test case cho những khu vực chưa được kiểm thử.

Kiểm thử khám phá đôi khi được thực hiện theo phương pháp kiểm thử theo phiên (session-based) để cấu trúc hoá các hoạt động kiểm thử. Với cách tiếp cận này, kiểm thử khám phá được thực hiện trong một khoảng thời gian có giới hạn (ví dụ: 45 phút). Tester sử dụng một thẻ kiểm thử (test charter) chứa các mục tiêu kiểm thử giúp định hướng việc kiểm thử. Phiên kiểm thử thường được theo sau bởi một buổi tổng kết, bao gồm việc thảo luận giữa tester và các bên liên quan khác có quan tâm đến kết quả của phiên kiểm thử. Trong cách tiếp cận này, các mục tiêu kiểm thử có thể được xem như các khía cạnh kiểm thử ở mức cao. Các hạng mục bao phủ sẽ được xác định và thực thi trong suốt phiên kiểm thử. Tester có thể sử dụng các phiếu ghi phiên kiểm thử, để ghi lại các bước đã thực hiện và các phát hiện trong quá trình kiểm thử.

Kiểm thử khám phá sẽ hữu ích khi có ít hoặc không đủ tài liệu đặc tả, hoặc khi gặp áp lực lớn về thời gian kiểm thử. Kiểm thử khám phá cũng hữu ích để bổ trợ cho các kỹ thuật kiểm thử chính quy hơn (như các kỹ thuật kiểm thử hộp đen và hộp trắng). Kiểm thử khám phá sẽ hiệu quả hơn nếu tester có kinh nghiệm, có kiến thức nghiệp vụ liên quan đến ứng dụng, và sở hữu các kỹ năng quan trọng như kỹ năng phân tích, sự tò mò, và tính sáng tạo (xem mục 1.5.1).

Kiểm thử khám phá có thể được thực hiện kết hợp với các kỹ thuật kiểm thử khác (ví dụ: phân vùng tương đương). Thông tin chi tiết hơn về kiểm thử khám phá có thể tham khảo trong (Kaner 1999, Whittaker 2009, Hendrickson 2013).

4.4.3. Kiểm thử Dựa trên Checklist

Trong kiểm thử dựa trên checklist, tester thiết kế, triển khai, và thực thi các test case để bao phủ các khía cạnh kiểm thử được mô tả trong một checklist (danh sách các mục cần kiểm tra). Checklist có thể được xây dựng dựa trên kinh nghiệm, kiến thức về những gì quan trọng đối với người dùng, hoặc sự hiểu biết về lý do và cách phần mềm bị lỗi. Checklist có thể được xây dựng để hỗ trợ nhiều loại kiểm thử khác nhau, bao gồm cả kiểm thử chức năng và phi chức năng (xem thêm: 10 nguyên tắc kinh nghiệm cho kiểm thử khả dụng (Nielsen 1994)).

Một số mục trong checklist có thể dần trở nên kém hiệu quả theo thời gian, vì các lập trình viên sẽ học cách tránh lặp lại các sai sót cũ. Đồng thời, cũng cần bổ sung thêm các mục mới để phản ánh những lỗi nghiêm trọng vừa được phát hiện. Vì vậy, checklist cần được cập nhật thường xuyên dựa trên phân tích lỗi. Tuy nhiên, cần tránh để checklist trở nên quá dài (Gawande 2009).

Trong trường hợp không có bộ test case chi tiết, kiểm thử dựa trên checklist có thể cung cấp hướng dẫn và một mức độ nhất quán nhất định cho hoạt động kiểm thử. Do checklist là ở mức khái quát, hoạt động kiểm thử thực tế có thể có một số khác biệt, dẫn đến khả năng đạt độ bao phủ cao hơn nhưng tính lặp lại sẽ thấp hơn.

4.5. Các Phương pháp Kiểm thử Dựa trên sự Cộng tác

Các kỹ thuật kiểm thử đã nêu ở những phần trên (xem các mục 4.2, 4.3, và 4.4) đều có mục tiêu riêng liên quan đến việc phát hiện lỗi. Trong khi đó, các phương pháp tiếp cận kiểm thử dựa trên sự cộng tác tập trung vào việc phòng tránh lỗi thông qua sự hợp tác và giao tiếp giữa các bên liên quan.

4.5.1. Viết User Story theo Phương pháp Cộng tác

Mỗi user story đại diện cho một tính năng mang lại giá trị cho người dùng hoặc người mua hệ thống/ phần mềm. User story có ba yếu tố quan trọng (Jeffries 2000), gọi chung là nguyên tắc “3 C’s”:

- Card (thẻ) – phương tiện mô tả user story (ví dụ: thẻ giấy, một mục trong bảng điện tử)
- Conversation (thảo luận) – giải thích cách phần mềm sẽ được sử dụng (có thể được ghi lại hoặc trao đổi bằng lời nói)
- Confirmation (xác nhận) – tiêu chí chấp nhận (xem mục 4.5.2)

Định dạng phổ biến nhất của một user story là: “Là [vai trò], tôi muốn [mục tiêu cần đạt được], để tôi có thể [giá trị nghiệp vụ mang lại cho vai trò đó]”, theo sau đó là các tiêu chí chấp nhận.

Việc cùng nhau xây dựng user story có thể sử dụng các kỹ thuật như động não brainstorming (động não) và mind mapping (lập sơ đồ tư duy). Sự cộng tác này giúp nhóm đạt được một tầm nhìn chung về những gì cần được phát triển, bằng cách xem xét đồng thời từ ba góc nhìn: nghiệp vụ, phát triển, và kiểm thử.

Một user story tốt nên tuân theo nguyên tắc INVEST: Independent – độc lập, Negotiable – có thể thương lượng, Valuable – có giá trị, Estimable – có thể ước lượng, Small – nhỏ gọn, và Testable – có thể kiểm thử được. Nếu một bên liên quan (như tester hay lập trình viên) không biết cách kiểm thử một user story nào đó, điều này có thể cho thấy user story đó chưa đủ rõ ràng, hoặc chưa phản ánh giá trị đối với họ, hoặc đơn giản là họ cần được hỗ trợ trong việc kiểm thử user story đó (Wake 2003).

4.5.2. Tiêu chí Chấp nhận

Tiêu chí chấp nhận (acceptance criteria - AC) của một user story là các điều kiện mà phần chương trình được lập trình cho user story đó phải đáp ứng để được các bên liên quan chấp nhận. Theo cách nhìn này, tiêu chí chấp nhận có thể được xem như các khía cạnh kiểm thử cần được thực thi bởi các test case. Tiêu chí chấp nhận thường là kết quả của phần thảo luận trong nguyên tắc 3C's (xem mục 4.5.1).

Tiêu chí chấp nhận được sử dụng để:

- Xác định phạm vi của user story
- Đạt được sự thống nhất giữa các bên liên quan
- Mô tả các kịch bản bao gồm cả hợp lệ và không hợp lệ
- Làm cơ sở cho việc kiểm thử chấp nhận user story (xem mục 4.5.3)
- Giúp lập kế hoạch và ước lượng chính xác

Có nhiều cách biểu diễn tiêu chí chấp nhận trong một user story. Hai dạng phổ biến nhất là:

- Theo hướng kịch bản (ví dụ: dạng Given/When/Then trong BDD, xem mục 2.1.3)
- Theo hướng quy tắc (ví dụ: danh sách các mục dạng gạch đầu dòng hoặc dạng bảng ánh xạ đầu vào – đầu ra)

Hầu hết các tiêu chí chấp nhận có thể được mô tả theo một trong hai định dạng trên. Tuy nhiên, các nhóm phát triển phần mềm có thể sử dụng một định dạng tùy chỉnh khác, miễn là các tiêu chí chấp nhận được mô tả và trình bày rõ ràng.

4.5.3. Phát triển hướng Kiểm thử Chấp nhận (ATDD)

ATDD là một phương pháp phát triển phần mềm theo hướng kiểm thử trước (test-first) (xem mục 2.1.3). Các test case được tạo ra trước khi lập trình cho một user story. Các test case này được xây dựng bởi các thành viên trong nhóm với các góc nhìn khác nhau, ví dụ: khách hàng, lập trình viên, và tester (Adzic 2009). Test case có thể được thực thi theo cách thủ công hoặc tự động.

Bước đầu tiên là buổi làm việc nhóm về đặc tả yêu cầu, trong đó user story và các tiêu chí chấp nhận của nó (nếu chưa được xác định) sẽ được phân tích, thảo luận, và ghi lại bởi các thành viên trong nhóm. Các vấn đề như thiếu sót, mơ hồ, hoặc lỗi trong user story sẽ được giải quyết ngay trong quá trình này. Bước tiếp theo là tạo test case. Việc này có thể được thực hiện bởi cả nhóm hoặc chỉ mình tester. Các test case được xây dựng dựa trên tiêu chí chấp nhận. Và chúng có thể được xem như là các ví dụ về cách phần mềm hoạt động. Điều này giúp nhóm tiến hành lập trình các user story một cách chính xác hơn.

Bởi vì “ví dụ” và “test case” thực chất là như nhau, nên hai thuật ngữ này thường được sử dụng thay thế cho nhau. Trong quá trình thiết kế test case, chúng ta có thể áp dụng các kỹ thuật kiểm thử đã được mô tả ở các mục 4.2, 4.3, và 4.4.

Thông thường, các test case đầu tiên là để kiểm tra các trường hợp hợp lệ (positive case), xác nhận hành vi đúng theo yêu cầu mà không xảy ra tình huống bất thường hoặc điều kiện lỗi, bao gồm một chuỗi các hoạt động được thực hiện khi mọi thứ diễn ra như mong đợi. Sau khi hoàn thành các trường hợp hợp lệ, nhóm sẽ kiểm thử các trường hợp không hợp lệ (negative testing). Cuối cùng, nhóm sẽ kiểm tra các đặc tính chất lượng phi chức năng (ví dụ: hiệu năng, khả năng sử dụng). Test case nên được diễn đạt theo cách dễ hiểu đối với tất cả các bên liên quan. Thông thường, test case bao gồm các câu mô tả bằng ngôn ngữ tự nhiên, gồm các điều kiện tiên quyết nếu có (precondition), dữ liệu đầu vào, và các yêu cầu sau khi thực thi test case (postcondition).

Các test case phải bao phủ mọi yêu cầu và khía cạnh của user story và không được vượt ra ngoài phạm vi của story. Tuy nhiên, các tiêu chí chấp nhận có thể chi tiết hóa một số vấn đề được mô tả trong user story. Ngoài ra, các test case không được trùng lặp trong việc mô tả cùng một đặc tính của user story.

Khi được trình bày theo định dạng hỗ trợ bởi một công cụ tự động hóa kiểm thử, các lập trình viên có thể tự động hóa những test case này bằng cách viết thêm mã nguồn bổ trợ song song với việc lập trình tính năng được mô tả trong user story. Khi đó, những test case (ở mức chấp nhận) này sẽ trở thành những yêu cầu có thể thực thi được.

5. Quản lý các hoạt động kiểm thử – 335 phút

Từ khóa

báo cáo lỗi, báo cáo kết thúc kiểm thử, báo cáo tiến độ kiểm thử, chiến lược kiểm thử, giảm thiểu rủi ro, giám sát kiểm thử, giám sát rủi ro, kế hoạch kiểm thử, kiểm soát kiểm thử, kiểm soát rủi ro, kiểm thử dựa vào rủi ro, lập kế hoạch kiểm thử, mức rủi ro, phương pháp tiếp cận kiểm thử, phân tích rủi ro, quản lý lỗi, quản lý rủi ro, nhận diện rủi ro, rủi ro, rủi ro dự án, rủi ro sản phẩm, tiêu chí bắt đầu, tiêu chí kết thúc, testing quadrants (các góc phần tư kiểm thử), test pyramid (tháp kiểm thử)

Mục tiêu Học tập của Chương 5:

5.1 Lập kế hoạch Kiểm thử

FL-5.1.1 (K2 Minh hoạ mục đích và nội dung của kế hoạch kiểm thử

FL-5.1.2 (K1) Nhận biết cách tester mang lại giá trị trong việc lập kế hoạch trong từng vòng lặp phát triển (iteration) và bản phát hành (release)

FL-5.1.3 (K2) So sánh và đối chiếu tiêu chí bắt đầu và tiêu chí kết thúc kiểm thử

FL-5.1.4 (K3) Sử dụng các kỹ thuật ước lượng để tính toán công sức kiểm thử cần thiết

FL-5.1.5 (K3) Áp dụng ưu tiên hóa test case

FL-5.1.6 (K1) Ghi nhớ khái niệm tháp kiểm thử (test pyramid)

FL-5.1.7 (K2) Tóm tắt các góc phần tư kiểm thử (testing quadrants) và mối quan hệ của chúng với các cấp độ kiểm thử và các loại kiểm thử

5.2 Quản lý Rủi ro

FL-5.2.1 (K1) Xác định mức rủi ro dựa vào khả năng xảy ra và mức độ ảnh hưởng

FL-5.2.2 (K2) Phân biệt rủi ro dự án và rủi ro sản phẩm

FL-5.2.3 (K2) Giải thích cách phân tích rủi ro sản phẩm có thể ảnh hưởng đến mức độ kiểm thử và phạm vi kiểm thử

FL-5.2.4 (K2) Giải thích các biện pháp có thể thực hiện để ứng phó với rủi ro sản phẩm đã được phân tích

5.3 Giám sát Kiểm thử, Kiểm soát Kiểm thử và Hoàn tất Kiểm thử

FL-5.3.1 (K1) Ghi nhớ các chỉ số được sử dụng trong kiểm thử

FL-5.3.2 (K2) Tóm tắt mục đích, nội dung và đối tượng của báo cáo kiểm thử

FL-5.3.3 (K1) Minh họa cách truyền đạt trạng thái của quá trình kiểm thử

5.4 Quản lý Cấu hình

FL-5.4.1 (K2) Tóm tắt cách quản lý cấu hình hỗ trợ kiểm thử

5.5 Quản lý Lỗi

FL-5.5.1 (K3) Chuẩn bị báo cáo lỗi

5.1. Lập Kế hoạch Kiểm thử

5.1.1. Mục đích và Nội dung của Kế hoạch Kiểm thử

Kế hoạch kiểm thử mô tả các mục tiêu kiểm thử, nguồn lực, và các quy trình của một dự án kiểm thử. Kế hoạch kiểm thử sẽ:

- Ghi lại cách thức và lịch trình để đạt được các mục tiêu kiểm thử
- Giúp đảm bảo rằng các hoạt động kiểm thử được thực hiện sẽ đáp ứng các tiêu chí đã đề ra
- Đóng vai trò như một phương tiện giao tiếp với các thành viên trong nhóm và các bên liên quan khác
- Chứng minh rằng quá trình kiểm thử sẽ tuân thủ các chính sách kiểm thử và chiến lược kiểm thử hiện có trong công ty (hoặc giải thích lý do về các sai lệch nếu có)

Việc lập kế hoạch kiểm thử sẽ định hướng tư duy của các tester và buộc các tester phải đối mặt với những thách thức trong tương lai liên quan đến rủi ro, lịch trình, con người, công cụ, chi phí, công sức bỏ ra, v.v. Quá trình lập kế hoạch kiểm thử giúp xem xét một cách có hệ thống các nguồn lực cần thiết để đạt được các mục tiêu kiểm thử.

Nội dung điển hình của một kế hoạch kiểm thử bao gồm:

- Bối cảnh kiểm thử (ví dụ: phạm vi kiểm thử, mục tiêu kiểm thử, cơ sở kiểm thử)
- Các giả định và ràng buộc của dự án kiểm thử
- Các bên liên quan (ví dụ: vai trò, trách nhiệm, mức độ liên quan đến các hoạt động kiểm thử, nhu cầu tuyển dụng và đào tạo)
- Truyền thông (ví dụ: hình thức và tần suất truyền thông, mẫu tài liệu)
- Danh sách rủi ro (ví dụ: rủi ro sản phẩm, rủi ro dự án)
- Phương pháp tiếp cận kiểm thử (ví dụ: mức kiểm thử, loại kiểm thử, kỹ thuật kiểm thử, các sản phẩm bàn giao của kiểm thử, tiêu chí bắt đầu và tiêu chí kết thúc, mức độ độc lập của kiểm thử, các chỉ số cần thu thập, yêu cầu về dữ liệu kiểm thử, yêu cầu về môi trường kiểm thử, các sai lệch so với chính sách kiểm thử và chiến lược kiểm thử)
- Ngân sách và lịch làm việc

5.1.2. Đóng góp của Tester trong việc Lập Kế hoạch cho Iteration và Release

Trong các mô hình SDLC lặp, thường có hai loại lập kế hoạch: lập kế hoạch cho release (bản phát hành) và lập kế hoạch cho iteration (vòng lặp phát triển).

Lập kế hoạch cho release hướng đến phiên bản sản phẩm dự kiến phát hành, xác định và cập nhật danh mục công việc của sản phẩm (product backlog), đồng thời có thể bao gồm việc tinh chỉnh các user story lớn thành các user story nhỏ hơn. Kế hoạch release cũng đóng vai trò là cơ sở để xác định phương pháp tiếp cận kiểm thử và lập kế hoạch kiểm thử xuyên suốt các vòng lặp phát triển của nó. Tester tham gia lập kế hoạch release cùng viết các user story và tiêu chí chấp nhận có thể kiểm thử được (xem mục 4.5), tham gia phân tích rủi ro dự án và rủi ro chất lượng (xem mục 5.2), ước lượng nguồn lực kiểm thử liên quan đến các user story (xem mục 5.1.4), xác định phương pháp tiếp cận kiểm thử, và lập kế hoạch kiểm thử cho release đó.

Lập kế hoạch cho iteration hướng đến từng iteration và tập trung vào đầu mục công việc của các iteration. Tester tham gia lập kế hoạch iteration sẽ cùng phân tích chi tiết rủi ro của các user story, xác định khả năng kiểm thử của từng user story, phân tách user story thành các nhiệm vụ riêng biệt (đặc biệt là các nhiệm vụ kiểm thử), ước lượng công sức kiểm thử cho mọi nhiệm vụ kiểm thử, và xác định cũng như tinh chỉnh các khía cạnh chức năng và phi chức năng của đối tượng kiểm thử.

5.1.3. Tiêu chí Bắt đầu và Tiêu chí Kết thúc

Tiêu chí bắt đầu xác định các điều kiện tiên quyết để tiến hành một hoạt động nhất định. Nếu các tiêu chí bắt đầu không được đáp ứng, có khả năng hoạt động đó sẽ trở nên khó khăn hơn, tốn nhiều thời gian hơn, tốn kém và rủi ro hơn. Tiêu chí kết thúc xác định những điều phải đạt được để tuyên bố rằng một hoạt động đã hoàn thành. Tiêu chí bắt đầu và tiêu chí kết thúc nên được xác định riêng cho từng mức kiểm thử, và sẽ khác nhau tùy theo mục tiêu kiểm thử.

Tiêu chí bắt đầu điển hình bao gồm: Sự sẵn sàng của mọi nguồn lực (ví dụ: con người, công cụ, môi trường, dữ liệu kiểm thử, ngân sách, thời gian), sự sẵn sàng của sản phẩm kiểm thử (ví dụ: cơ sở kiểm thử, yêu cầu có thể kiểm thử được, user story, test case), và mức chất lượng ban đầu của đối tượng kiểm thử (ví dụ: tất cả các test case để đánh giá nhanh (smoke test) đã thực thi và đạt).

Tiêu chí kết thúc điển hình bao gồm: Các thước đo mức độ bao quát (ví dụ: mức độ bao phủ đạt được, số lượng lỗi chưa được giải quyết, mật độ lỗi, số lượng test case không đạt), và các tiêu chí "có/không" (ví dụ: đã thực hiện mọi test case theo kế hoạch, đã thực hiện kiểm thử tĩnh, đã báo cáo tất cả các lỗi được phát hiện, đã tự động hóa tất cả các test case hồi quy).

Việc hết thời gian hoặc ngân sách cũng có thể được xem là một tiêu chí kết thúc hợp lệ. Ngay cả khi chưa đáp ứng đầy đủ các tiêu chí kết thúc khác, việc dừng kiểm thử trong những trường hợp như vậy vẫn có thể được chấp nhận, nếu các bên liên quan đã xem xét và chấp nhận rủi ro khi đưa sản phẩm ra môi trường thực tế mà không cần kiểm thử thêm.

Trong phát triển phần mềm Agile, tiêu chí kết thúc thường được gọi là Định nghĩa Hoàn thành (Definition of Done - DoD), là các chỉ số khách quan của nhóm để quyết định một hạng mục có thể được phát hành. Các tiêu chí bắt đầu mà một user story phải đáp ứng để tiến hành các hoạt động phát triển và/hoặc kiểm thử được gọi là Định nghĩa Sẵn sàng (Definition of Ready - DoR).

5.1.4. Các Kỹ thuật Ước lượng

Ước lượng công việc kiểm thử là việc dự đoán khối lượng công việc liên quan đến các hoạt động kiểm thử cần thiết để đạt được các mục tiêu kiểm thử của dự án kiểm thử. Điều quan trọng là phải làm rõ với các bên liên quan rằng ước lượng này được dựa trên một số giả định và luôn có sai số trong ước lượng. Ước lượng cho các nhiệm vụ nhỏ thường chính xác hơn so với các nhiệm vụ lớn. Do đó, khi ước lượng một nhiệm vụ lớn, nên phân tách ra thành các nhiệm vụ nhỏ hơn, sau đó ước lượng cho các nhiệm vụ nhỏ này.

Trong đề cương này, bốn kỹ thuật ước lượng sau đây được mô tả.

Ước lượng dựa trên tỷ lệ. Trong kỹ thuật dựa trên số liệu này, các số liệu được thu thập từ những dự án trước đây trong tổ chức, qua đó có thể xác định các tỷ lệ "chuẩn" cho những dự án tương tự. Các tỷ lệ này được rút ra từ các dự án của chính tổ chức (ví dụ: lấy từ dữ liệu lịch sử) thường là nguồn đáng tin cậy nhất để sử dụng trong quá trình ước lượng. Những tỷ lệ chuẩn này sau đó có thể được dùng để ước lượng công việc kiểm thử cho dự án mới. Ví dụ, nếu trong dự án trước đó, tỷ lệ công sức phát triển so với kiểm thử là 3:2, và trong dự án hiện tại công sức phát triển dự kiến là 600 ngày, thì công sức kiểm thử có thể được ước lượng là 400 ngày.

Kỹ thuật ước lượng ngoại suy. Trong kỹ thuật dựa trên số liệu này, các phép đo được thực hiện càng sớm càng tốt trong dự án hiện tại để thu thập dữ liệu. Khi đã có đủ lượng dữ liệu quan sát, công sức cần thiết cho phần công việc còn lại có thể được ước lượng bằng cách ngoại suy từ dữ liệu này (thường bằng cách áp dụng một mô hình toán học). Phương pháp này rất phù hợp trong các mô hình phát triển phần mềm lặp. Ví dụ, nhóm phát triển phần mềm có thể ngoại suy công sức kiểm thử trong chu kỳ phát triển sắp tới bằng cách lấy giá trị trung bình công sức từ ba chu kỳ phát triển gần nhất.

Wideband Delphi. Trong kỹ thuật ước lượng dựa vào chuyên gia này, các chuyên gia đưa ra ước lượng dựa trên kinh nghiệm của mình. Mỗi chuyên gia, làm việc độc lập, đưa ra con số ước lượng cần thiết. Các kết quả sau đó được thu thập lại và nếu có bất kỳ ước lượng nào nằm ngoài phạm vi chênh lệch đã thống nhất trước, các chuyên gia sẽ thảo luận về ước lượng hiện tại của họ. Sau đó, mỗi chuyên gia sẽ được yêu cầu đưa ra một ước lượng mới dựa trên kết quả thảo luận, một lần nữa việc đưa ra ước lượng của mỗi chuyên gia là riêng tư. Quá trình này được lặp lại cho đến khi đạt được sự đồng thuận. Planning Poker là một biến thể của Wideband Delphi, thường được sử dụng trong các mô hình phát triển phần mềm Agile. Trong Planning Poker, các ước lượng thường được thực hiện bằng cách sử dụng các lá bài có số biểu thị độ lớn của công sức bỏ ra.

Ước lượng ba điểm. Trong kỹ thuật ước lượng dựa vào chuyên gia này, mỗi chuyên gia cần đưa ra 3 ước lượng: ước lượng lạc quan nhất (a), ước lượng có khả năng xảy ra nhất (m), và ước lượng bi quan nhất (b). Ước lượng cuối cùng (E) là trung bình cộng có tính trọng số của 3 giá trị này. Trong phiên bản phổ biến nhất của kỹ thuật này, giá trị ước lượng được tính như sau: E = (a + 4*m + b) / 6. Ưu điểm của kỹ thuật này là cho phép các chuyên gia tính được sai số ước lượng: SD = (b – a) / 6. Ví dụ, nếu các ước lượng (tính theo giờ làm việc) là: a = 6, m = 9, và b = 18, thì ước lượng cuối cùng là 10 ± 2 giờ (tức nằm trong khoảng từ 8 đến 12 giờ làm việc), vì E = (6 + 4*9 + 18) / 6 = 10 và SD = (18 – 6) / 6 = 2.

Xem thêm (Kan 2003, Koomen 2006, Westfall 2009) để biết thêm các kỹ thuật ước lượng kiểm thử khác.

5.1.5. Ưu tiên hóa Test Case

Sau khi các test case và trình tự thực hiện chúng đã được xác định và tập hợp lại thành các bộ test case, những bộ test case này có thể được sắp xếp vào một lịch trình thực thi kiểm thử, xác định thứ tự mà chúng sẽ được thực thi. Khi ưu tiên hóa test case, chúng ta có thể xem xét nhiều yếu tố khác nhau. Các chiến lược ưu tiên hoá test case phổ biến nhất bao gồm:

- Ưu tiên hoá dựa vào rủi ro, trong chiến lược này thứ tự thực hiện test case được xác định dựa trên kết quả phân tích rủi ro (xem mục 5.2.3). Các test case bao phủ những rủi ro cao nhất sẽ được thực thi trước.
- Ưu tiên hoá dựa vào độ bao phủ, trong chiến lược này thứ tự thực hiện test case được xác định dựa vào độ bao phủ (ví dụ: bao phủ câu lệnh). Các test case đạt được mức độ bao phủ cao nhất sẽ được thực thi trước. Trong một biến thể khác gọi là ưu tiên hoá dựa vào bao phủ bổ sung, test case đạt được mức độ bao phủ cao nhất sẽ được thực thi trước; mỗi test case tiếp theo là test case mang lại mức độ bao phủ bổ sung cao nhất.
- Ưu tiên hoá dựa vào yêu cầu, trong chiến lược này thứ tự thực hiện test case được xác định dựa trên mức độ ưu tiên của yêu cầu tương ứng liên kết với các test case đó. Mức độ ưu tiên của yêu cầu do các bên liên quan xác định. Các test case liên quan đến những yêu cầu quan trọng nhất sẽ được thực thi trước.

Trong điều kiện lý tưởng, các test case sẽ được sắp xếp để thực thi theo mức độ ưu tiên của chúng, sử dụng một trong các chiến lược ưu tiên nêu trên. Tuy nhiên, cách tiếp cận này có thể không áp dụng được nếu các test case hoặc các tính năng được kiểm thử có sự phụ thuộc lẫn nhau. Nếu một test case có mức độ ưu tiên cao phụ thuộc vào một test case có mức độ ưu tiên thấp hơn, thì test case có mức ưu tiên thấp hơn vẫn phải được thực thi trước.

Thứ tự thực thi kiểm thử cũng phải xem xét tính sẵn có của nguồn lực kiểm thử. Ví dụ, các công cụ kiểm thử, môi trường kiểm thử, hoặc nhân sự có thể chỉ sẵn sàng trong một khoảng thời gian nhất định.

5.1.6. Tháp kiểm thử (Test Pyramid)

Test pyramid (tháp kiểm thử) là mô hình thể hiện các loại test case khác nhau có thể có mức độ chi tiết khác nhau. Mô hình test pyramid hỗ trợ nhóm trong việc tự động hóa kiểm thử và phân bổ nguồn lực kiểm thử, bằng cách chỉ ra các mục tiêu kiểm thử khác nhau được hỗ trợ bởi các mức độ tự động hóa kiểm thử khác nhau.

Các tầng của tháp đại diện cho các nhóm test case. Tầng càng cao, độ chi tiết của việc kiểm thử càng thấp, mức độ tách biệt của kiểm thử càng thấp (tức là test case càng phụ thuộc vào nhiều phần tử khác của hệ thống và thực thi nhiều thành phần hệ thống hơn), và thời gian thực thi test case càng lâu. Các test case ở tầng đáy có kích thước nhỏ, độc lập, chạy nhanh, và kiểm tra một phần chức năng nhỏ, vì vậy thường cần rất nhiều test case như vậy để đạt được độ bao phủ hợp lý. Tầng trên cùng đại diện cho các kiểm thử phức tạp, mức kiểm thử cao, kiểm thử end-to-end. Các test case ở mức cao này thường chậm hơn so với các kiểm thử ở tầng dưới, và chúng thường kiểm tra một phần chức năng lớn, vì vậy chỉ cần một số lượng nhỏ để đạt được mức độ bao phủ hợp lý. Số lượng và cách đặt tên các tầng trong mô hình test pyramid có thể khác nhau. Ví dụ, mô hình tháp kiểm thử ban đầu (Cohn 2009) định nghĩa 3 tầng: “unit tests”, “service tests” và “UI tests”. Một mô hình phổ biến khác định nghĩa các tầng gồm: kiểm thử đơn vị (unit tests, còn gọi là component tests - kiểm thử thành phần), kiểm thử mức tích hợp thành phần (component integration tests), và kiểm thử end-to-end. Các mức kiểm thử khác (xem mục 2.2.1) cũng có thể được sử dụng.

5.1.7. Các Góc phần tư Kiểm thử (Testing Quadrants)

Các testing quadrant (góc phần tư kiểm thử), được định nghĩa bởi Brian Marick (Marick 2003, Crispin 2008), gom nhóm các mức kiểm thử với các loại kiểm thử phù hợp, các hoạt động kiểm thử, kỹ thuật kiểm thử, và các sản phẩm công việc trong phát triển phần mềm Agile. Testing quadrant hỗ trợ việc quản lý kiểm thử bằng cách trực quan hóa các yếu tố trên, nhằm đảm bảo rằng tất cả các loại kiểm thử và mức kiểm thử phù hợp đều được đưa vào vòng đời phát triển phần mềm (SDLC), đồng thời cho thấy rằng một số loại kiểm thử sẽ phù hợp hơn trong một vài mức kiểm thử này so với các mức kiểm thử khác. Testing quadrant cũng cung cấp một cách để phân biệt và mô tả các loại kiểm thử cho tất cả các bên liên quan, bao gồm lập trình viên, tester, và đại diện nghiệp vụ.

Trong mô hình testing quadrant này, các test case có thể hướng về nghiệp vụ hoặc hướng về công nghệ. Test case cũng có thể nhằm hỗ trợ nhóm phát triển (tức là định hướng và dẫn dắt quá trình phát triển) hoặc để đánh giá/phê bình sản phẩm (tức là đo lường hành vi của sản phẩm so với mong đợi của các bên liên quan). Sự kết hợp của hai góc nhìn này tạo ra bốn góc phần tư (quadrant):

- Góc phần tư Q1 (hướng về công nghệ, hỗ trợ nhóm). Góc phần tư này bao gồm các test case ở mức kiểm thử thành phần và kiểm thử tích hợp thành phần. Các test case này nên được tự động hóa và tích hợp vào quy trình CI (Continuous Integration).
- Góc phần tư Q2 (hướng về nghiệp vụ, hỗ trợ nhóm). Góc phần tư này bao gồm các test case kiểm thử chức năng, examples (ví dụ), kiểm thử dựa vào user story, các nguyên mẫu để đánh giá trải nghiệm người dùng, kiểm thử API, và các mô phỏng. Các test case này kiểm tra các tiêu chí chấp nhận và có thể được thực hiện thủ công hoặc tự động.
- Góc phần tư Q3 (hướng về nghiệp vụ, đánh giá sản phẩm). Góc phần tư này bao gồm kiểm thử khám phá (exploratory testing), kiểm thử tính khả dụng (usability testing), và kiểm thử chấp nhận người dùng (UAT). Các test case này hướng tới người dùng và thường được thực hiện thủ công.
- Góc phần tư Q4 (hướng về công nghệ, đánh giá sản phẩm). Góc phần tư này bao gồm các test case để đánh giá nhanh (smoke test) và các test case kiểm thử phi chức năng (ngoại trừ kiểm thử khả dụng). Các test case này thường được tự động hóa hoặc cần công cụ hỗ trợ kiểm thử.

5.2. Quản lý Rủi ro

Các tổ chức phải đối mặt với nhiều yếu tố bên trong và bên ngoài, khiến việc xác định liệu họ có đạt được mục tiêu hay không, và khi nào đạt được, trở nên không chắc chắn (ISO 31000). Quản lý rủi ro giúp các tổ chức tăng khả năng đạt được mục tiêu, nâng cao chất lượng sản phẩm, và xây dựng niềm tin cũng như mức độ tín nhiệm của các bên liên quan.

Các hoạt động chính của quản lý rủi ro bao gồm:

- Phân tích rủi ro (gồm nhận diện rủi ro và đánh giá rủi ro; xem mục 5.2.3)
- Kiểm soát rủi ro (gồm giảm thiểu rủi ro và giám sát rủi ro; xem mục 5.2.4)

Cách tiếp cận kiểm thử, trong đó các hoạt động kiểm thử được lựa chọn, ưu tiên, và quản lý dựa trên việc phân tích rủi ro và kiểm soát rủi ro, được gọi là kiểm thử dựa vào rủi ro (risk-based testing).

5.2.1. Định nghĩa Rủi ro và Các Thuộc tính của Rủi ro

Rủi ro là một sự kiện, mối nguy, mối đe dọa, hoặc tình huống tiềm ẩn mà khi xảy ra sẽ gây ra tác động bất lợi. Một rủi ro có thể được xác định dựa trên hai yếu tố:

- Khả năng xảy ra rủi ro – xác suất rủi ro xảy ra (lớn hơn 0 và nhỏ hơn 1)
- Mức độ ảnh hưởng của rủi ro (gây hại) – các hậu quả khi rủi ro xảy ra

Hai yếu tố này thể hiện mức rủi ro, là thước đo rủi ro. Mức rủi ro càng cao thì việc xử lý rủi ro đó càng quan trọng.

5.2.2. Rủi ro Dự án và Rủi ro Sản phẩm

Trong kiểm thử phần mềm, chúng ta thường quan tâm đến hai loại rủi ro: rủi ro dự án và rủi ro sản phẩm.

Rủi ro dự án liên quan đến việc quản lý và kiểm soát các hoạt động trong dự án. Các rủi ro dự án bao gồm:

- Vấn đề tổ chức (ví dụ: chậm trễ trong việc bàn giao các sản phẩm công việc, ước lượng không chính xác, cắt giảm chi phí)
- Vấn đề con người (ví dụ: thiếu kỹ năng, mâu thuẫn, vấn đề giao tiếp, thiếu nhân sự)
- Vấn đề kỹ thuật (ví dụ: phạm vi công việc bị mở rộng ngoài tầm kiểm soát, công cụ hỗ trợ kém)
- Vấn đề nhà cung cấp (ví dụ: bên thứ ba không cung cấp dịch vụ/ giao hàng đúng hạn, bên thứ ba bị phá sản)

Khi xảy ra, các rủi ro dự án có thể ảnh hưởng đến tiến độ, ngân sách, hoặc phạm vi dự án, từ đó tác động đến khả năng đạt được mục tiêu của dự án.

Rủi ro sản phẩm liên quan đến các đặc tính chất lượng của sản phẩm (ví dụ như được mô tả trong mô hình chất lượng ISO 25010). Các ví dụ về rủi ro sản phẩm bao gồm: chương trình bị thiếu hoặc sai chức năng, tính toán sai, lỗi khi chạy, kiến trúc kém, thuật toán không hiệu quả, thời gian phản hồi không phù hợp, trải nghiệm người dùng kém, các lỗ hổng bảo mật. Khi xảy ra, rủi ro sản phẩm có thể dẫn đến nhiều hậu quả tiêu cực khác nhau, bao gồm:

- Người dùng không hài lòng
- Mất doanh thu, mất niềm tin, tổn hại danh tiếng
- Gây thiệt hại cho bên thứ ba
- Chi phí bảo trì cao, gây quá tải bộ phận chăm sóc khách hàng
- Các hình phạt pháp lý, hình sự
- Trong trường hợp nghiêm trọng, có thể gây thiệt hại về thể chất cho con người như thương tích hoặc thậm chí tử vong

5.2.3. Phân tích Rủi ro Sản phẩm

Từ góc độ kiểm thử, mục tiêu của việc phân tích rủi ro sản phẩm là nâng cao nhận thức về rủi ro sản phẩm và tập trung nguồn lực kiểm thử theo cách giúp giảm thiểu mức độ rủi ro sản phẩm còn lại. Lý tưởng nhất, việc phân tích rủi ro sản phẩm nên được bắt đầu sớm từ những giai đoạn đầu của SDLC.

Phân tích rủi ro sản phẩm bao gồm các hoạt động nhận diện rủi ro và đánh giá rủi ro. Nhận diện rủi ro là liệt kê danh sách các rủi ro đầy đủ. Các bên liên quan có thể nhận diện rủi ro bằng nhiều kỹ thuật và công cụ khác nhau, ví dụ: brainstorming (động não), các phiên làm việc nhóm, phỏng vấn, hoặc sơ đồ nguyên nhân – kết quả. Đánh giá rủi ro bao gồm: phân loại các rủi ro đã xác định, xác định khả năng xảy ra, mức độ ảnh hưởng, và mức rủi ro, ưu tiên hóa, và đề xuất cách xử lý chúng. Việc phân loại rủi ro giúp hỗ trợ việc xác định các hành động giảm thiểu rủi ro, vì thông thường các rủi ro thuộc cùng một nhóm có thể được xử lý bằng cách tiếp cận tương tự.

Đánh giá rủi ro có thể sử dụng cách tiếp cận định lượng, định tính, hoặc kết hợp cả hai. Trong cách tiếp cận định lượng, mức rủi ro được tính toán bằng cách nhân khả năng xảy ra rủi ro với mức độ ảnh hưởng của rủi ro. Trong cách tiếp cận định tính, mức rủi ro có thể được xác định bằng cách sử dụng ma trận rủi ro.

Phân tích rủi ro sản phẩm có thể ảnh hưởng đến mức độ chi tiết và phạm vi kiểm thử. Kết quả của nó được sử dụng để:

- Xác định phạm vi kiểm thử cần thực hiện
- Xác định các mức kiểm thử cụ thể và đề xuất các loại kiểm thử cần thực hiện
- Xác định các kỹ thuật kiểm thử sẽ được sử dụng và mức độ bao phủ cần đạt được

- Ước lượng nguồn lực kiểm thử cần thiết cho từng công việc
- Ưu tiên các hoạt động kiểm thử nhằm phát hiện các lỗi nghiêm trọng sớm nhất có thể
- Xác định xem có thể áp dụng thêm hoạt động nào khác ngoài kiểm thử để giảm thiểu rủi ro hay không

5.2.4. Kiểm soát Rủi ro Sản phẩm

Kiểm soát rủi ro sản phẩm bao gồm tất cả các biện pháp được thực hiện để ứng phó các rủi ro sản phẩm đã được xác định và đánh giá. Kiểm soát rủi ro sản phẩm bao gồm giảm thiểu rủi ro và giám sát rủi ro. Giảm thiểu rủi ro là việc triển khai các hành động được đề xuất trong quá trình đánh giá rủi ro nhằm giảm mức rủi ro. Mục tiêu của việc giám sát rủi ro là đảm bảo rằng các biện pháp giảm thiểu rủi ro mang lại hiệu quả, thu thập thêm thông tin để cải thiện việc đánh giá rủi ro, và xác định các rủi ro mới phát sinh.

Liên quan đến việc kiểm soát rủi ro sản phẩm, sau khi một rủi ro đã được phân tích, có thể có nhiều lựa chọn ứng phó với rủi ro đó, ví dụ: giảm thiểu rủi ro bằng kiểm thử, chấp nhận rủi ro, chuyển giao rủi ro, hoặc lập kế hoạch dự phòng (Veenendaal 2012). Các hành động có thể được thực hiện để giảm thiểu rủi ro sản phẩm thông qua kiểm thử bao gồm:

- Lựa chọn tester có kinh nghiệm và kỹ năng phù hợp với từng loại rủi ro
- Áp dụng mức kiểm thử độc lập phù hợp
- Thực hiện rà soát và phân tích tĩnh
- Áp dụng các kỹ thuật kiểm thử và mức độ bao phủ phù hợp
- Áp dụng các loại kiểm thử phù hợp để giải quyết (các rủi ro liên quan đến) những đặc tính chất lượng bị ảnh hưởng.
- Thực hiện kiểm thử động, bao gồm cả kiểm thử hồi quy

5.3. Giám sát Kiểm thử, Kiểm soát Kiểm thử và Kết thúc Kiểm thử

Giám sát kiểm thử liên quan đến việc thu thập thông tin về các hoạt động kiểm thử. Thông tin này được sử dụng để đánh giá tiến độ kiểm thử và đo lường xem các tiêu chí kết thúc hoặc các nhiệm vụ kiểm thử liên quan đến các tiêu chí kết thúc đã được đáp ứng hay chưa, chẳng hạn như đã đáp ứng mục tiêu về độ bao phủ rủi ro sản phẩm, yêu cầu, hoặc các tiêu chí chấp nhận.

Kiểm soát kiểm thử sử dụng những thông tin từ hoạt động giám sát kiểm thử để đưa ra các chỉ đạo điều chỉnh, hướng dẫn, và các hành động khắc phục cần thiết, nhằm đạt được hiệu quả và hiệu suất kiểm thử tốt nhất.

Ví dụ về các chỉ đạo kiểm soát bao gồm:

- Thay đổi mức độ ưu tiên của các test case sau khi một rủi ro xảy ra, trở thành sự cố
- Đánh giá lại việc một đối tượng kiểm thử có đáp ứng tiêu chí bắt đầu hoặc tiêu chí kết thúc hay không, sau khi đã thực hiện thay đổi
- Điều chỉnh lịch kiểm thử để xử lý việc chậm trễ trong việc cung cấp môi trường kiểm thử
- Bổ sung thêm nguồn lực vào thời điểm và những nơi cần thiết

Kết thúc kiểm thử thu thập dữ liệu từ các hoạt động kiểm thử đã hoàn thành nhằm tổng hợp kinh nghiệm, các sản phẩm kiểm thử, và các thông tin liên quan khác. Các hoạt động kết thúc kiểm thử diễn ra tại các mốc quan trọng của dự án, chẳng hạn như khi một mức kiểm thử đã hoàn tất, một vòng lặp phát triển trong Agile kết thúc, dự án kiểm thử hoàn thành (hoặc bị hủy), hệ thống phần mềm đã được phát hành, hoặc một bản phát hành bảo trì đã được hoàn tất.

5.3.1. Các Chỉ số được Sử dụng trong Kiểm thử

Các chỉ số kiểm thử (test metric) được thu thập để thể hiện tiến độ thực tế so với kế hoạch kiểm thử đã đề ra và ngân sách đã phân bổ, chất lượng hiện tại của đối tượng kiểm thử, cũng như hiệu quả của các hoạt động kiểm thử so với mục tiêu kiểm thử, hoặc mục tiêu của một vòng lặp phát triển. Thông qua các hoạt động giám sát kiểm thử, nhiều loại chỉ số khác nhau được thu thập nhằm hỗ trợ cho việc kiểm soát kiểm thử và kết thúc kiểm thử.

Các chỉ số kiểm thử phổ biến bao gồm:

- Chỉ số về tiến độ dự án (ví dụ: mức độ hoàn thành công việc, mức độ sử dụng nguồn lực, công sức kiểm thử)
- Chỉ số về tiến độ kiểm thử (ví dụ: tiến độ viết test case, tiến độ chuẩn bị môi trường kiểm thử, số lượng test case đã chạy/chưa chạy, đạt/không đạt, thời gian thực thi kiểm thử)
- Chỉ số về chất lượng sản phẩm (ví dụ: khả năng sẵn sàng, thời gian phản hồi, thời gian trung bình giữa các sự cố trên môi trường thực tế)
- Chỉ số về lỗi (ví dụ: số lượng và mức độ ưu tiên của lỗi được phát hiện/sửa, mật độ lỗi, tỷ lệ phát hiện lỗi)
- Chỉ số về rủi ro (ví dụ: mức rủi ro còn lại)
- Chỉ số về độ bao phủ (ví dụ: độ bao phủ yêu cầu, độ bao phủ mã nguồn)
- Chỉ số về chi phí (ví dụ: chi phí kiểm thử, chi phí chất lượng của tổ chức)

5.3.2. Mục đích, Nội dung và Đối tượng của Báo cáo Kiểm thử

Việc báo cáo kiểm thử là quá trình tổng hợp và truyền đạt thông tin về các hoạt động kiểm thử trong suốt quá trình và sau khi hoạt động kiểm thử kết thúc. Báo cáo tiến độ kiểm thử hỗ trợ việc kiểm soát kiểm thử liên tục và phải cung cấp thông tin đầy đủ để có thể điều chỉnh lịch thực thi kiểm thử, nguồn lực, hoặc kế hoạch kiểm thử khi cần thiết, do có sự sai lệch so với kế hoạch, hoặc do bối cảnh kiểm thử đã bị thay đổi. Báo cáo kết thúc kiểm thử tổng hợp thông tin về một hoạt động kiểm thử cụ thể (ví dụ: một mức kiểm thử, một chu kỳ kiểm thử, một vòng lặp phát triển trong Agile) và có thể cung cấp thông tin đầu vào cho các hoạt động kiểm thử tiếp theo.

Trong quá trình giám sát kiểm thử và kiểm soát kiểm thử, nhóm kiểm thử tạo ra các báo cáo tiến độ kiểm thử để cung cấp thông tin cho các bên liên quan, giúp họ nắm được tình hình đang diễn ra. Các báo cáo tiến độ kiểm thử thường được tạo định kỳ (ví dụ: hàng ngày, hàng tuần, v.v...) và bao gồm các thông tin như:

- Giai đoạn kiểm thử
- Tiến độ kiểm thử (ví dụ: nhanh hơn hoặc chậm hơn so với kế hoạch), bao gồm các sai lệch đáng chú ý
- Các trở ngại trong quá trình thực hiện kiểm thử và cách khắc phục tạm thời
- Các chỉ số kiểm thử (xem các ví dụ trong mục 5.3.1)
- Các rủi ro mới và các rủi ro đã thay đổi trong giai đoạn kiểm thử
- Các hoạt động kiểm thử được lên kế hoạch cho giai đoạn tiếp theo

Báo cáo kết thúc kiểm thử được lập trong quá trình kết thúc kiểm thử, khi một dự án, cấp độ kiểm thử hoặc loại kiểm thử hoàn tất và, lý tưởng nhất, các tiêu chí kết thúc của nó đã được đáp ứng. Báo cáo này sử dụng thông tin từ các báo cáo tiến độ kiểm thử và thông tin khác. Báo cáo kết thúc kiểm thử thường bao gồm:

- Thông tin tóm tắt về quá trình kiểm thử đã diễn ra
- Đánh giá các hoạt động kiểm thử và chất lượng sản phẩm dựa trên kế hoạch kiểm thử ban đầu (tức là mục tiêu kiểm thử và các tiêu chí kết thúc)
- Các sai lệch so với kế hoạch kiểm thử (ví dụ: khác biệt về lịch trình, thời lượng, và nguồn lực)

- Các trở ngại trong quá trình kiểm thử và các cách khắc phục tạm thời
- Các chỉ số kiểm thử dựa trên báo cáo tiến độ kiểm thử
- Các rủi ro chưa được giảm thiểu, và các lỗi chưa được sửa
- Những bài học kinh nghiệm liên quan đến các hoạt động kiểm thử

Các bên nhận báo cáo khác nhau sẽ cần những thông tin khác nhau trong báo cáo và ảnh hưởng đến mức độ chính thức cũng như tần suất báo cáo kiểm thử. Báo cáo tiến độ kiểm thử gửi đến các thành viên khác trong cùng nhóm sẽ thường xuyên hơn và ít trang trọng hơn, trong khi đó, báo cáo kết thúc kiểm thử sẽ tuân theo một biểu mẫu được quy định trước và chỉ được thực hiện một lần.

Tiêu chuẩn ISO/IEC/IEEE 29119-3 mô tả các biểu mẫu và ví dụ cho báo cáo tiến độ kiểm thử (được gọi là báo cáo trạng thái kiểm thử) và báo cáo kết thúc kiểm thử.

5.3.3. Truyền đạt Trạng thái Kiểm thử

Phương thức tốt nhất để truyền đạt trạng thái kiểm thử sẽ khác nhau tùy thuộc vào mối quan tâm của quản lý kiểm thử, chiến lược kiểm thử của tổ chức, các tiêu chuẩn quy định, hoặc tùy thuộc vào chính nhóm đó, trong trường hợp các nhóm tự quản lý (xem mục 1.5.2). Các lựa chọn bao gồm:

- Giao tiếp bằng lời nói với các thành viên trong nhóm và các bên liên quan khác
- Bảng điều khiển (ví dụ: bảng điều khiển CI/CD, bảng công việc, và biểu đồ burn-down)
- Các kênh giao tiếp điện tử (ví dụ: email, chat)
- Tài liệu trực tuyến
- Báo cáo kiểm thử chính thức (xem mục 5.3.2)

Có thể sử dụng một hoặc nhiều trong số các lựa chọn trên. Các hình thức trao đổi mang tính chính thức hơn có thể phù hợp hơn với các nhóm phân tán, do việc trao đổi mặt-đối-mặt trực tiếp không phải lúc nào cũng khả thi do khoảng cách địa lý, hoặc khác múi giờ. Thông thường, các bên liên quan khác nhau sẽ quan tâm đến các loại thông tin khác nhau, do đó việc truyền đạt trạng thái kiểm thử cũng cần được điều chỉnh phù hợp.

5.4. Quản lý Cấu hình

Trong kiểm thử, quản lý cấu hình (CM - configuration management) cung cấp một phương thức có hệ thống để xác định, kiểm soát và theo dõi các sản phẩm công việc như kế hoạch kiểm thử, chiến lược kiểm thử, điều kiện kiểm thử, test case, test script, kết quả kiểm thử, nhật ký kiểm thử và báo cáo kiểm thử dưới dạng các hạng mục cấu hình.

Đối với một hạng mục cấu hình phức tạp (ví dụ: môi trường kiểm thử), CM ghi lại các hạng mục mà nó bao gồm, mối quan hệ giữa chúng, và các phiên bản của chúng. Khi một hạng mục cấu hình đã được phê duyệt để kiểm thử, nó trở thành một phiên bản chuẩn và chỉ có thể được thay đổi thông qua một quy trình kiểm soát thay đổi chính thức.

Quản lý cấu hình lưu lại thông tin về các hạng mục cấu hình đã thay đổi khi một phiên bản chuẩn mới được tạo. Có thể quay lại một phiên bản chuẩn trước đó khi cần, để tái hiện lại kết quả kiểm thử trước đây.

Để hỗ trợ kiểm thử một cách phù hợp, CM đảm bảo các điều sau:

- Tất cả các hạng mục cấu hình, bao gồm các hạng mục kiểm thử (các phần riêng lẻ của đối tượng kiểm thử), được gán định danh duy nhất, kiểm soát phiên bản, theo dõi thay đổi, và liên kết với các hạng mục cấu hình khác để đảm bảo khả năng truy vết (truy xuất nguồn gốc) trong toàn bộ quá trình kiểm thử
- Tất cả các tài liệu và hạng mục phần mềm đã được xác định đều được tham chiếu rõ ràng giữa các testware (sản phẩm kiểm thử)

Tích hợp liên tục (CI), phân phối liên tục (CD), triển khai liên tục (CD), và các hoạt động kiểm thử liên quan thường được triển khai như một phần của quy trình phân phối DevOps tự động (xem mục 2.1.4), trong đó việc quản lý cấu hình tự động thường đã được tích hợp sẵn.

5.5. Quản lý Lỗi

Vì một trong những mục tiêu chính của kiểm thử là phát hiện lỗi, nên quy trình quản lý lỗi được thiết lập là điều rất cần thiết. Mặc dù ở đây chúng ta sử dụng thuật ngữ "lỗi" (defects), nhưng các bất thường (anomalies) được báo cáo có thể thực sự là lỗi hoặc cũng có thể là một vấn đề khác (ví dụ: dương tính giả – false positive, hoặc là một yêu cầu cải tiến) — điều này sẽ được xác định trong quá trình xử lý các báo cáo lỗi. Các bất thường có thể được báo cáo trong bất kỳ giai đoạn nào của quá trình phát triển phần mềm (SDLC) và hình thức báo cáo phụ thuộc vào SDLC đang sử dụng. Ít nhất, quy trình quản lý lỗi gồm một luồng xử lý để giải quyết từng lỗi hoặc bất thường từ khi phát hiện cho đến khi nó được đóng lại, cùng với các quy tắc phân loại chúng. Luồng xử lý này thường bao gồm các hoạt động: ghi nhận các bất thường được báo cáo, phân tích và phân loại chúng, quyết định phản hồi phù hợp như sửa lỗi hoặc giữ nguyên, và cuối cùng là đóng báo cáo lỗi. Quy trình này phải được tất cả các bên liên quan tuân thủ thực hiện. Những lỗi được phát hiện trong các hoạt động kiểm thử tĩnh (đặc biệt là phân tích tĩnh) cũng nên được xử lý theo cách tương tự.

Các báo cáo lỗi thường có các mục tiêu sau:

- Cung cấp đầy đủ thông tin cho những người chịu trách nhiệm xử lý và khắc phục lỗi để họ giải quyết vấn đề
- Cung cấp phương tiện để theo dõi chất lượng của sản phẩm công việc
- Cung cấp ý tưởng để cải tiến quy trình phát triển và kiểm thử

Một báo cáo lỗi được ghi nhận trong quá trình kiểm thử động thường bao gồm:

- Mã định danh duy nhất (ví dụ: Bug ID)
- Tiêu đề tóm tắt ngắn gọn về bất thường được báo cáo
- Ngày phát hiện bất thường, tổ chức, và người báo cáo, bao gồm vai trò của họ
- Thông tin xác định đối tượng kiểm thử và môi trường kiểm thử
- Bối cảnh của lỗi (ví dụ: test case đang thực thi, hoạt động kiểm thử đang thực hiện, lỗi được phát hiện ở giai đoạn nào trong SDLC, và các thông tin liên quan khác như kỹ thuật kiểm thử, checklist nào, hoặc dữ liệu kiểm thử đang sử dụng)
- Mô tả chi tiết nhằm hỗ trợ việc tái hiện và khắc phục, bao gồm các bước thực hiện giúp phát hiện bất thường, cùng với nhật ký kiểm thử, bản sao lưu cơ sở dữ liệu, ảnh chụp màn hình, hoặc bản ghi hình liên quan
- Kết quả mong đợi và kết quả thực tế
- Mức độ ảnh hưởng (mức độ nghiêm trọng) của lỗi đối với lợi ích của các bên liên quan hoặc yêu cầu
- Mức độ ưu tiên sửa lỗi
- Trạng thái của lỗi (ví dụ: open (mở), deferred (hoãn xử lý), duplicate (trùng lặp), waiting to be fixed (chờ sửa), awaiting confirmation testing (chờ kiểm thử xác nhận), re-opened (mở lại), closed (đã đóng), rejected (bị từ chối))
- Tài liệu tham chiếu (ví dụ: test case liên quan)

Một số dữ liệu trên có thể được tự động được cung cấp khi sử dụng các công cụ quản lý lỗi (ví dụ: mã định danh - Bug ID, thời gian tạo, người báo cáo, và trạng thái ban đầu của lỗi). Các mẫu tài liệu về báo cáo lỗi và ví dụ về báo cáo lỗi có thể được tìm thấy trong tiêu chuẩn ISO/IEC/IEEE 29119-3, trong đó báo cáo lỗi được gọi là báo cáo sự cố.

6. Công cụ Kiểm thử – 20 phút

Từ khóa

tự động hóa kiểm thử

Mục tiêu học tập của Chương 6:

6.1 Công cụ Hỗ trợ trong Kiểm thử

FL-6.1.1 (K2) Giải thích cách các loại công cụ khác nhau hỗ trợ hoạt động kiểm thử

6.2 Lợi ích và Rủi ro của việc Tự động hóa Kiểm thử

FL-6.2.1 (K1) Ghi nhớ các lợi ích và rủi ro của tự động hóa kiểm thử

6.1. Công cụ Hỗ trợ trong Kiểm thử

Các công cụ kiểm thử hỗ trợ và tạo điều kiện thuận lợi cho nhiều hoạt động kiểm thử. Một số ví dụ bao gồm, nhưng không giới hạn:

- Công cụ quản lý kiểm thử – giúp tăng hiệu quả của quy trình kiểm thử bằng cách hỗ trợ quản lý: vòng đời phát triển phần mềm (SDLC), yêu cầu, test case, lỗi, và cấu hình
- Công cụ kiểm thử tĩnh – hỗ trợ tester trong việc thực hiện rà soát và phân tích tĩnh
- Công cụ thiết kế và triển khai kiểm thử – hỗ trợ việc tạo test case, dữ liệu kiểm thử, và trình tự thực hiện kiểm thử
- Công cụ thực thi kiểm thử và đo độ bao phủ – hỗ trợ thực thi kiểm thử tự động và đo lường độ bao phủ
- Công cụ kiểm thử phi chức năng – cho phép tester thực hiện các kiểm thử phi chức năng vốn khó hoặc không thể thực hiện thủ công
- Công cụ DevOps – hỗ trợ quy trình phân phối DevOps, theo dõi quy trình công việc, tự động hóa quá trình build phần mềm và CI/CD
- Công cụ cộng tác – hỗ trợ giao tiếp giữa các bên liên quan
- Các công cụ hỗ trợ khả năng mở rộng và chuẩn hóa triển khai phần mềm (ví dụ: máy ảo, công cụ container hóa)
- Bất kỳ công cụ nào khác mà hỗ trợ các hoạt động kiểm thử (ví dụ: bảng tính cũng được xem là một công cụ kiểm thử trong ngữ cảnh kiểm thử)

6.2. Lợi ích và rủi ro của tự động hóa kiểm thử

Chỉ mua hoặc triển khai một công cụ sẽ không đảm bảo thành công. Mỗi công cụ mới đều cần nhiều nỗ lực để mang lại lợi ích thực sự và lâu dài (ví dụ: triển khai công cụ, bảo trì, và đào tạo). Bên cạnh đó, quá trình tự động hoá kiểm thử phần mềm cũng tồn tại một số rủi ro cần được phân tích và giảm thiểu.

Các lợi ích tiềm năng của việc tự động hóa kiểm thử bao gồm:

- Tiết kiệm thời gian nhờ giảm các công việc thủ công lặp đi lặp lại (ví dụ: chạy kiểm thử hồi quy, nhập lại cùng một dữ liệu kiểm thử, so sánh kết quả mong đợi với kết quả thực tế, và đánh giá dựa theo các tiêu chuẩn lập trình)
- Giảm thiểu lỗi do con người thông qua tính nhất quán và khả năng lặp lại cao hơn (ví dụ: test case được xây dựng nhất quán từ yêu cầu, dữ liệu kiểm thử được tạo một cách có hệ thống, và các test case được công cụ thực thi theo cùng một thứ tự với cùng tần suất)
- Đánh giá khách quan hơn (ví dụ: độ bao phủ) và cung cấp các số liệu mà con người khó có thể xác định trong quá trình kiểm thử
- Dễ dàng truy cập thông tin kiểm thử để hỗ trợ quản lý và báo cáo kiểm thử (ví dụ: số liệu thống kê, biểu đồ, và dữ liệu tổng hợp về tiến độ kiểm thử, tỉ lệ sự cố, và thời gian thực thi)
- Giảm thời gian thực thi các hoạt động kiểm thử, giúp phát hiện lỗi sớm hơn, phản hồi nhanh hơn và rút ngắn thời gian đưa sản phẩm ra thị trường
- Giúp tester có thêm thời gian để thiết kế các test case mới, chuyên sâu hơn và hiệu quả hơn

Các rủi ro tiềm ẩn khi áp dụng tự động hóa kiểm thử bao gồm:

- Kỳ vọng không thực tế về lợi ích về công cụ (bao gồm chức năng và mức độ dễ sử dụng)
- Ước lượng không chính xác về thời gian, chi phí, và công sức cần thiết để triển khai công cụ, bảo trì test script, và thay đổi quy trình kiểm thử thủ công hiện tại để triển khai công cụ mới.

- Sử dụng công cụ kiểm thử trong khi kiểm thử thủ công phù hợp hơn
- Phụ thuộc quá nhiều vào công cụ, ví dụ như bỏ qua vai trò của tư duy phản biện của con người
- Phụ thuộc vào nhà cung cấp công cụ, có thể dẫn đến rủi ro như: nhà cung cấp ngừng hoạt động, ngừng phát triển công cụ đó, bán công cụ đó cho bên khác, hoặc cung cấp hỗ trợ kém (ví dụ: phản hồi chậm, nâng cấp hạn chế, sửa lỗi không kịp thời)
- Sử dụng phần mềm mã nguồn mở có thể bị ngừng phát triển, dẫn đến không còn được cập nhật; hoặc các thành phần bên trong phần mềm đó cần phải cập nhật thường xuyên để tiếp tục sử dụng
- Công cụ kiểm thử tự động không tương thích với nền tảng phát triển
- Lựa chọn một công cụ không phù hợp, không đáp ứng các yêu cầu theo quy định và/hoặc tiêu chuẩn an toàn

7. Tài liệu Tham khảo

Tiêu chuẩn

ISO/IEC/IEEE 29119-1 (2022) Software and systems engineering – Software testing – Part 1: General Concepts

ISO/IEC/IEEE 29119-2 (2021) Software and systems engineering – Software testing – Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2021) Software and systems engineering – Software testing – Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2021) Software and systems engineering – Software testing – Part 4: Test techniques

ISO/IEC 25010, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 20246 (2017) Software and systems engineering – Work product reviews

ISO/IEC/IEEE 14764:2022 – Software engineering – Software life cycle processes – Maintenance ISO 31000 (2018) Risk management – Principles and guidelines

Sách

Adzic, G. (2009) Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing, Neuri Limited

Ammann, P. and Offutt, J. (2016) Introduction to Software Testing (2e), Cambridge University Press

Andrews, M. and Whittaker, J. (2006) How to Break Web Software: Functional and Security Testing of Web Applications and Web Services, Addison-Wesley Professional

Beck, K. (2003) Test Driven Development: By Example, Addison-Wesley

Beizer, B. (1990) Software Testing Techniques (2e), Van Nostrand Reinhold: Boston MA Boehm, B. (1981) Software Engineering Economics, Prentice Hall, Englewood Cliffs, NJ

Buxton, J.N. and Randell B., eds (1970), Software Engineering Techniques. Report on a conference sponsored by the NATO Science Committee, Rome, Italy, 27–31 October 1969, p. 16

Chelimsky, D. et al. (2010) The Rspec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, The Pragmatic Bookshelf: Raleigh, NC

Cohn, M. (2009) Succeeding with Agile: Software Development Using Scrum, Addison-Wesley Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood MA Craig, R. and Jaskiel, S. (2002) Systematic Software Testing, Artech House: Norwood MA

Crispin, L. and Gregory, J. (2008) Agile Testing: A Practical Guide for Testers and Agile Teams, Pearson Education: Boston MA

Forgács, I., and Kovács, A. (2019) Practical Test Design: Selection of traditional and automated test design techniques, BCS, The Chartered Institute for IT

Gawande A. (2009) The Checklist Manifesto: How to Get Things Right, New York, NY: Metropolitan Books

Gärtner, M. (2011), ATDD by Example: A Practical Guide to Acceptance Test-Driven Development, Pearson Education: Boston MA

Gilb, T., Graham, D. (1993) Software Inspection, Addison Wesley

Hendrickson, E. (2013) Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing, The Pragmatic Programmers

Hetzel, B. (1988) The Complete Guide to Software Testing, 2nd ed., John Wiley and Sons

Jeffries, R., Anderson, A., Hendrickson, C. (2000) Extreme Programming Installed, Addison-Wesley Professional

Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL Kan, S. (2003) Metrics and Models in Software Quality Engineering, 2nd ed., Addison-Wesley Kaner, C., Falk, J., and Nguyen, H.Q. (1999) Testing Computer Software, 2nd ed., Wiley

Kaner, C., Bach, J., and Pettichord, B. (2011) Lessons Learned in Software Testing: A Context-Driven Approach, 1st ed., Wiley

Kim, G., Humble, J., Debois, P. and Willis, J. (2016) The DevOps Handbook, Portland, OR

Koomen, T., van der Aalst, L., Broekman, B. and Vroon, M. (2006) TMap Next for result-driven testing, UTN Publishers, The Netherlands

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY O'Regan, G. (2019) Concise Guide to Software Testing, Springer Nature Switzerland Pressman, R.S. (2019) Software Engineering. A Practitioner's Approach, 9th ed., McGraw Hill

Roman, A. (2018) Thinking-Driven Testing. The Most Reasonable Approach to Quality Control, Springer Nature Switzerland

Van Veenendaal, E (ed.) (2012) Practical Risk-Based Testing, The PRISMA Approach, UTN Publishers: The Netherlands

Watson, A.H., Wallace, D.R. and McCabe, T.J. (1996) Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, U.S. Dept. of Commerce, Technology Administration, NIST

Westfall, L. (2009) The Certified Software Quality Engineer Handbook, ASQ Quality Press Whittaker, J. (2002) How to Break Software: A Practical Guide to Testing, Pearson

Whittaker, J. (2009) Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design, Addison Wesley

Whittaker, J. and Thompson, H. (2003) How to Break Software Security, Addison Wesley Wiegers, K. (2001) Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional

Bài viết và Trang web

Brykczynski, B. (1999) "A survey of software inspection checklists," *ACM SIGSOFT Software Engineering Notes, 24(1), pp. 82-89*

Enders, A. (1975) "An Analysis of Errors and Their Causes in System Programs," *IEEE Transactions on Software Engineering 1(2), pp. 140-149*

Manna, Z., Waldinger, R. (1978) "The logic of computer programming," *IEEE Transactions on Software Engineering 4(3), pp. 199-229*

Marick, B. (2003) Exploration through Example, http://www.exampler.com/old-blog/2003/08/21.1.html#agile-testing-project-1

Nielsen, J. (1994) "Enhancing the explanatory power of usability heuristics," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating Interdependence, ACM Press, pp. 152–158*

Salman. I. (2016) "Cognitive biases in software quality and testing," *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16), ACM, pp. 823-826.*

Wake, B. (2003) "INVEST in Good Stories, and SMART Tasks," https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

8. Phụ lục A – Mục tiêu học tập/Mức độ nhận thức

Các mục tiêu học tập dưới đây được định nghĩa và áp dụng cho đề cương này. Mỗi chủ đề trong Đề cương sẽ được kiểm tra theo mục tiêu học tập tương ứng. Các mục tiêu học tập bắt đầu bằng một động từ hành động tương ứng với mức độ nhận thức như được liệt kê bên dưới.

Mức 1: Ghi nhớ (K1) – ứng viên có thể ghi nhớ, nhận biết, và nhớ lại một thuật ngữ hoặc khái niệm.

Động từ hành động: xác định, nhớ lại, ghi nhớ, nhận biết.

Vị dụ:

- "Xác định các mục tiêu kiểm thử điển hình."
- "Nhớ lại các khái niệm về test pyramid."
- "Nhận biết cách một tester tạo giá trị trong việc lập kế hoạch sprint và release."

Mức độ 2: Hiểu (K2) – ứng viên có thể lựa chọn lý do hoặc giải thích cho các phát biểu liên quan đến chủ đề, đồng thời có thể tóm tắt, so sánh, phân loại và đưa ra ví dụ cho khái niệm kiểm thử.

Động từ hành động: phân loại, so sánh, đối chiếu, phân biệt, làm rõ sự khác nhau, minh họa, giải thích, đưa ví dụ, diễn giải, tóm tắt.

Examples:

- "Phân loại các lựa chọn khác nhau để viết tiêu chí chấp nhận."
- "So sánh các vai trò khác nhau trong kiểm thử" (tìm điểm giống và khác).
- "Phân biệt giữa rủi ro dự án và rủi ro sản phẩm" (giúp làm rõ các khái niệm).
- "Minh họa mục đích và nội dung của một kế hoạch kiểm thử."
- "Giải thích tác động của bối cảnh lên quy trình kiểm thử."
- "Tóm tắt các hoạt động của quy trình rà soát."

Mức độ: Áp dụng (K3) – ứng viên có thể thực hiện một quy trình khi đối mặt với một nhiệm vụ quen thuộc, hoặc lựa chọn đúng quy trình và áp dụng nó vào một bối cảnh cụ thể.

Động từ hành động: áp dụng, triển khai, chuẩn bị, sử dụng.

Ví dụ:

- "Áp dụng kỹ thuật ưu tiên hóa test case" (nên tham chiếu đến một quy trình, kỹ thuật, phương pháp hoặc thuật toán cụ thể).
- "Chuẩn bị báo cáo lỗi."
- "Sử dụng kỹ thuật phân tích giá trị biên để thiết kế test case."

Tài liệu tham khảo về các mức độ nhận thức:

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

9. Phụ lục B – Ma trận truy vết Giá trị Mang lại cho Doanh nghiệp so với Mục tiêu Học tập

Phần này liệt kê số lượng các Mục tiêu Học tập ở mức Nền tảng liên quan đến các Giá trị Mang lại cho Doanh nghiệp, cũng như mối liên kết truy vết giữa các Giá trị Mang lại cho Doanh nghiệp mức Nền tảng và các Mục tiêu Học tập mức Nền tảng.

Giá trị mang lại cho doanh nghiệp: Mức Nền tảng		FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
BO1	Hiểu kiểm thử là gì và vì sao nó mang lại giá trị	6													
BO2	Hiểu các khái niệm cơ bản của kiểm thử phần mềm		22												
BO3	Xác định cách tiếp cận và các hoạt động kiểm thử cần thực hiện tùy theo ngữ cảnh kiểm thử			6											
BO4	Đánh giá và cải thiện chất lượng tài liệu				9										
BO5	Nâng cao hiệu quả và hiệu suất của kiểm thử					20									
BO6	Đảm bảo quy trình kiểm thử phù hợp với các mô hình phát triển phần mềm						6								
BO7	Hiểu các nguyên tắc quản lý kiểm thử							6							
BO8	Viết và truyền đạt báo cáo lỗi rõ ràng và dễ hiểu								1						
BO9	Hiểu được các yếu tố ảnh hưởng đến mức độ ưu tiên và công sức kiểm thử									7					
BO10	Làm việc như một phần của nhóm đa chức năng										8				
BO11	Hiểu các rủi ro và lợi ích liên quan đến tự động hóa kiểm thử											1			
BO12	Hiểu được ảnh hưởng của rủi ro đến kiểm thử												5		
BO13	Understand the impact of risk on testing													4	
BO14	Báo cáo hiệu quả về tiến độ kiểm thử và chất lượng sản phẩm														4

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
Chương 1	**Nền tảng cơ bản của kiểm thử**															
1.1	**Kiểm thử là gì?**															
1.1.1	Xác định các mục tiêu kiểm thử điển hình	K1	X													
1.1.2	Phân biệt kiểm thử và gỡ lỗi	K2		X												
1.2	**Tại sao cần kiểm thử?**															
1.2.1	Minh họa lý do kiểm thử là cần thiết	K2	X													
1.2.2	Ghi nhớ mối quan hệ giữa kiểm thử và đảm bảo chất lượng	K1		X												
1.2.3	Phân biệt giữa nguyên nhân gốc rễ, sai sót, lỗi phần mềm, và sự cố	K2		X												
1.3	**Các nguyên tắc kiểm thử**															
1.3.1	Giải thích bảy nguyên tắc kiểm thử	K2		X												
1.4	**Hoạt động kiểm thử, sản phẩm kiểm thử, và vai trò kiểm thử**															
1.4.1	Giải thích các hoạt động kiểm thử khác nhau và các công việc liên quan	K2			X											
1.4.2	Giải thích ảnh hưởng của ngữ cảnh đến quy trình kiểm thử	K2			X			X								
1.4.3	Phân biệt các sản phẩm kiểm thử hỗ trợ cho các hoạt động kiểm thử	K2			X											
1.4.4	Giải thích giá trị của khả năng truy xuất nguồn gốc	K2				X	X									
1.4.5	So sánh các vai trò khác nhau trong kiểm thử	K2										X				

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
1.5	**Kỹ năng cốt lõi và thực hành tốt trong kiểm thử**															
1.5.1	Đưa ra ví dụ về các kỹ năng chung cần thiết cho kiểm thử	K2												X		
1.5.2	Ghi nhớ lợi ích của cách tiếp cận toàn đội	K1										X				
1.5.3	Phân biệt lợi ích và hạn chế của tính độc lập trong kiểm thử	K2			X											
Chương 2	**Kiểm thử trong suốt vòng đời phát triển phần mềm**															
2.1	**Kiểm thử theo từng ngữ cảnh của vòng đời phát triển phần mềm**															
2.1.1	Giải thích ảnh hưởng đến kiểm thử của mô hình phát triển phần mềm được lựa chọn	K2						X								
2.1.2	Ghi nhớ các thực hành kiểm thử tốt áp dụng cho mọi mô hình phát triển phần mềm	K1						X								
2.1.3	Ghi nhớ các ví dụ về các phương pháp phát triển theo hướng kiểm thử trước	K1					X									
2.1.4	Tóm tắt cách DevOps có thể ảnh hưởng đến kiểm thử	K2					X	X			X	X				
2.1.5	Giải thích khái niệm shift left	K2					X	X								
2.1.6	Giải thích cách sử dụng retrospective như một cơ chế cải tiến quy trình	K2					X					X				
2.2	**Các mức kiểm thử và các loại kiểm thử**															
2.2.1	Phân biệt các mức kiểm thử khác nhau	K2		X	X											
2.2.2	Phân biệt các loại kiểm thử khác nhau	K2		X												
2.2.3	Phân biệt kiểm thử xác nhận và kiểm thử hồi quy	K2		X												
2.3	**Kiểm thử bảo trì**															
2.3.1	Tóm tắt kiểm thử bảo trì và các yếu tố kích hoạt	K2		X					X							

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
Chương 3	**Kiểm thử tĩnh**															
3.1	**Cơ bản về kiểm thử tĩnh**															
3.1.1	Nhận biết các loại sản phẩm công việc có thể được kiểm tra bằng kiểm thử tĩnh	K1				X	X									
3.1.2	Giải thích các giá trị của kiểm thử tĩnh	K2	X			X	X									
3.1.3	So sánh và đối chiếu giữa kiểm thử tĩnh và kiểm thử động	K2				X	X									
3.2	**Phản hồi và quy trình rà soát**															
3.2.1	Xác định lợi ích của việc nhận phản hồi sớm và thường xuyên từ các bên liên quan	K1	X			X						X				
3.2.2	Tóm tắt các hoạt động trong quy trình rà soát	K2			X	X										
3.2.3	Ghi nhớ các trách nhiệm được giao cho các vai trò chính khi thực hiện rà soát	K1				X								X		
3.2.4	So sánh và đối chiếu các loại rà soát khác nhau	K2		X												
3.2.5	Ghi nhớ các yếu tố góp phần vào một buổi rà soát thành công	K1					X							X		
Chương 4	**Phân tích và thiết kế test case**															
4.1	**Tổng quan về các kỹ thuật kiểm thử**															
4.1.1	Phân biệt các kỹ thuật kiểm thử hộp đen, kỹ thuật kiểm thử hộp trắng, và kỹ thuật kiểm thử dựa vào kinh nghiệm	K2		X												
4.2	**Các kỹ thuật kiểm thử hộp đen**															
4.2.1	Áp dụng phân vùng tương đương để thiết kế test case	K3					X									
4.2.2	Áp dụng phân tích giá trị biên để thiết kế test case	K3					X									

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
4.2.3	Áp dụng kiểm thử bảng quyết định để thiết kế test case	K3					X									
4.2.4	Áp dụng kiểm thử chuyển đổi trạng thái để thiết kế test case	K3					X									
4.3	**Các kỹ thuật kiểm thử hộp trắng**															
4.3.1	Giải thích kiểm thử câu lệnh	K2		X												
4.3.2	Giải thích kiểm thử nhánh	K2		X												
4.3.3	Giải thích giá trị của kiểm thử hộp trắng	K2	X	X												
4.4	**Các kỹ thuật kiểm thử dựa vào kinh nghiệm**															
4.4.1	Giải thích đoán lỗi	K2		X												
4.4.2	Giải thích kiểm thử khám phá	K2		X												
4.4.3	Giải thích kiểm thử dựa vào checklist	K2		X												
4.5	**Các cách tiếp cận kiểm thử dựa trên hợp tác**															
4.5.1	Giải thích cách viết user story thông qua sự hợp tác với developer và đại diện nghiệp vụ	K2				X						X				
4.5.2	Phân loại một số cách viết tiêu chí chấp nhận	K2										X				
4.5.3	Sử dụng phát triển hướng kiểm thử chấp nhận (ATDD) để thiết kế test case	K3					X									
Chương 5	**Quản lý các hoạt động kiểm thử**															
5.1	**Lập kế hoạch kiểm thử**															
5.1.1	Minh hoạ mục đích và nội dung của kế hoạch kiểm thử	K2		X					X							
5.1.2	Nhận biết cách tester mang lại giá trị trong việc lập kế hoạch trong từng iteration và release	K1	X									X		X		

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
5.1.3	So sánh và đối chiếu tiêu chí bắt đầu và tiêu chí kết thúc kiểm thử	K2				X		X								X
5.1.4	Sử dụng các kỹ thuật ước lượng để tính toán công sức kiểm thử cần thiết	K3							X		X					
5.1.5	Áp dụng ưu tiên hóa test case	K3							X		X					
5.1.6	Ghi nhớ khái niệm tháp kiểm thử (test pyramid)	K1		X												
5.1.7	Tóm tắt các testing quadrant và mối quan hệ của chúng với các cấp độ kiểm thử và các loại kiểm thử	K2		X							X					
5.2	**Quản lý rủi ro**															
5.2.1	Xác định mức rủi ro dựa vào khả năng xảy ra và mức độ ảnh hưởng	K1							X						X	
5.2.2	Phân biệt rủi ro dự án và rủi ro sản phẩm	K2		X											X	
5.2.3	Giải thích cách phân tích rủi ro sản phẩm có thể ảnh hưởng đến mức độ kiểm thử và phạm vi kiểm thử	K2					X				X				X	
5.2.4	Giải thích các biện pháp có thể thực hiện để ứng phó với rủi ro sản phẩm đã được phân tích	K2		X			X								X	
5.3	**Giám sát kiểm thử, kiểm soát kiểm thử và hoàn tất kiểm thử**															
5.3.1	Ghi nhớ các chỉ số được sử dụng trong kiểm thử	K1									X					X
5.3.2	Tóm tắt mục đích, nội dung và đối tượng của báo cáo kiểm thử	K2					X				X					X
5.3.3	Minh hoạ cách truyền đạt trạng thái kiểm thử	K2												X		X
5.4	**Quản lý cấu hình**															
5.4.1	Tóm tắt cách quản lý cấu hình hỗ trợ kiểm thử	K2					X		X							

Chương/ mục/ tiểu mục	Mục tiêu học tập	K-level	GIÁ TRỊ DOANH NGHIỆP													
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
5.5	**Quản lý lỗi**															
5.5.1	Chuẩn bị báo cáo lỗi	K3		X						X						
Chương 6	**Công cụ kiểm thử**															
6.1	**Hỗ trợ kiểm thử bằng công cụ**															
6.1.1	Giải thích cách các loại công cụ khác nhau hỗ trợ hoạt động kiểm thử	K2					X									
6.2	**Lợi ích và Rủi ro của việc Tự động hóa Kiểm thử**															
6.2.1	Ghi nhớ các lợi ích và rủi ro của tự động hóa kiểm thử	K1					X						X			

10. Phụ lục C – Ghi chú Phát hành

Đề cương ISTQB® Mức độ Nền tảng phiên bản v4.0.1 là một bản đính chính của Đề cương ISTQB® Mức độ Nền tảng phiên bản v4.0. Bản đính chính này bao gồm các thay đổi sau đây.

Thay đổi cách diễn đạt mục tiêu học tập, nhằm đồng bộ với các thuật ngữ trong bảng thuật ngữ:

- FL-1.4.1: Tóm tắt các hoạt động và nhiệm vụ kiểm thử khác nhau → Giải thích các hoạt động kiểm thử khác nhau và các nhiệm vụ liên quan
- FL-2.1.5: Giải thích phương pháp tiếp cận shift-left → Giải thích khái niệm shift left
- FL-3.1.1: Nhận biết các loại sản phẩm có thể được xem xét bằng các kỹ thuật kiểm thử tĩnh khác nhau → Nhận biết các loại sản phẩm công việc có thể được xem xét bằng kiểm thử tĩnh
- FL-3.1.3 So sánh và đối chiếu kiểm thử tĩnh và kiểm thử động → So sánh và đối chiếu giữa kiểm thử tĩnh và kiểm thử động
- FL-4.1.1: Phân biệt kỹ thuật kiểm thử hộp đen, hộp trắng, và dựa trên kinh nghiệm → Phân biệt các kỹ thuật kiểm thử hộp đen, kỹ thuật kiểm thử hộp trắng, và kỹ thuật kiểm thử dựa trên kinh nghiệm
- FL-5.2.3: Giải thích cách phân tích rủi ro sản phẩm có thể ảnh hưởng đến mức độ kỹ lưỡng và phạm vi kiểm thử → Giải thích cách phân tích rủi ro sản phẩm có thể ảnh hưởng đến mức độ kỹ lưỡng và phạm vi kiểm thử

Các thay đổi về nội dung từ ngữ nhằm đồng bộ với các thuật ngữ trong bảng thuật ngữ (tạo tác, tài liệu → sản phẩm công việc, mức rủi ro → mức rủi ro, mục tiêu, mục tiêu của kiểm thử, mục tiêu của dự án kiểm thử → mục tiêu kiểm thử, giám sát và kiểm soát kiểm thử → giám sát kiểm thử và kiểm soát kiểm thử, tài liệu kiểm thử → sản phẩm kiểm thử, mô hình phát triển lặp và tăng dần → mô hình phát triển lặp và mô hình phát triển tăng dần, các thành phần môi trường kiểm thử → các hạng mục môi trường kiểm thử, các đặc tính chất lượng phần mềm → đặc tính chất lượng, báo cáo tiến độ và hoàn thành kiểm thử → báo cáo tiến độ kiểm thử và báo cáo hoàn thành kiểm thử, tính độc lập của kiểm thử → sự độc lập của hoạt động kiểm thử, kiểm thử thành phần và tích hợp thành phần → kiểm thử thành phần và kiểm thử tích hợp thành phần, hiệu năng → hiệu quả hiệu năng, kiểm thử chấp nhận theo hợp đồng và quy định → kiểm thử chấp nhận theo hợp đồng và kiểm thử chấp nhận theo quy định, tiêu chí bắt đầu/kết thúc → tiêu chí bắt đầu hoặc tiêu chí kết thúc, chính sách kiểm thử của tổ chức → chính sách kiểm thử, shift-left, phương pháp tiếp cận shift-left, chiến lược shift-left → shift left, các loại kiểm thử → loại kiểm thử, kiểm soát hoạt động kiểm thử → kiểm soát kiểm thử, giai đoạn kiểm thử → hoạt động kiểm thử, báo cáo về tiến độ kiểm thử → báo cáo tiến độ kiểm thử, báo cáo kiểm thử cho một dự án đã hoàn thành → báo cáo hoàn thành kiểm thử, dương tính giả → kết quả dương tính giả, bước → bước kiểm thử, phạm vi của hoạt động kiểm thử → phạm vi kiểm thử, công cụ thiết kế và triển khai kiểm thử → công cụ thiết kế kiểm thử và triển khai kiểm thử, kiểm thử tĩnh và động → kiểm thử tĩnh và kiểm thử động)

Cập nhật đối với ISO 25010. Một phiên bản mới của tiêu chuẩn ISO 25010 đã được công bố vào năm 2023. Phiên bản này đổi tên "khả năng tích hợp" (usability) thành "khả năng tương tác" (interaction capability), "khả năng chuyển đổi" (portability) thành "Tính linh hoạt" (flexibility), đồng thời bổ sung một đặc tính mới là “an toàn” (safety). Chúng tôi vẫn giữ nguyên tên gọi của các đặc tính ban đầu, nhưng bổ sung thêm các tên gọi mới cho khả năng tích hợp và khả năng chuyển đổi trong mục 2.2.2

Bổ sung ba từ khóa mới (qui trình kiểm thử và khả năng truy xuất nguồn gốc trong Chương 1, chiến lược kiểm thử trong Chương 5)

Các chỉnh sửa trong nội dung văn bản

- Trong mục 1.1.2, “nguyên nhân” và “nguyên nhân gốc rễ” được thay thế bằng “lỗi”.
- Trong mục 1.4.1, phần mô tả các hoạt động được làm rõ hơn và tránh gây hiểu nhầm.
- Trong mục 1.4.3, “kịch bản kiểm thử tự động” được đổi thành “kịch bản kiểm thử tự động và thủ công”.
- Trong mục 1.4.4, từ “được tìm thấy” trong cụm “lỗi được tìm thấy” được loại bỏ

- Trong mục 1.2.2, “QC” được thay thế bằng “kiểm thử”, vì phần này so sánh QA với kiểm thử, không phải với QC
- Trong mục 2.1.3, cụm “các test case sau đó được tự động chuyển đổi” được thay bằng “các test case sau đó nên được tự động chuyển đổi” trong ngữ cảnh của BDD
- Trong mục 2.1.5, cụm “quan điểm của kiểm thử” được đổi thành “quan điểm của tester”
- Trong mục 2.1.6, cụm “các cuộc họp sau dự án” được loại bỏ
- Trong mục 2.2.2, mô tả về nền tảng kiểm thử được thay đổi từ “tài liệu bên ngoài đối tượng kiểm thử” thành “tài liệu không liên quan đến cấu trúc bên trong của đối tượng kiểm thử”, nhằm thể hiện rõ hơn sự khác biệt giữa kiểm thử hộp đen và kiểm thử hộp trắng. Đồng thời, ví dụ về kiểm thử hệ thống cũng được loại bỏ khỏi phần minh họa cho việc bắt đầu sớm trong SDLC
- Trong mục 3.1, các đại diện nghiệp vụ được mô tả cụ thể hơn
- Trong mục 3.1.2, từ “cụ thể” được bổ sung vào câu “Một số lỗi trong mã nguồn có thể được phát hiện bằng phân tích tĩnh hiệu quả hơn so với kiểm thử động”. Nội dung trước đó có thể khiến người đọc hiểu rằng điều này áp dụng cho mọi loại lỗi mã nguồn có thể xảy ra
- Trong mục 3.2.2, cụm “một vài lần” được thay bằng “nhiều lần”, vì đối với các tài liệu lớn, việc xem xét chỉ một vài lần là không đủ
- Trong mục 4.2.1, “đối tượng kiểm thử” được thay bằng “hạng mục kiểm thử”, vì đây là thuật ngữ phù hợp trong ngữ cảnh áp dụng các kỹ thuật kiểm thử
- Trong mục 4.2.1, bổ sung nội dung rằng các phân vùng tương đương không hợp lệ nên được kiểm thử riêng lẻ để tránh hiện tượng che giấu lỗi
- Trong mục 4.2.4, “các trạng thái được đi qua” được thay bằng “các trạng thái được thực thi”, vì “thực thi” là thuật ngữ phù hợp trong ngữ cảnh bao phủ các phần tử của mô hình bằng các test case
- Trong mục 4.2.4, “lược đồ thay đổi trạng thái” được thay bằng “lược đồ trạng thái”, vì đây là tên gọi phổ biến của mô hình này trong khoa học máy tính, đồng thời để đảm bảo tính nhất quán với đề cương Kiểm thử dựa trên Mô hình
- Trong mục 5.1.1, từ “các ràng buộc” ở gạch đầu dòng đầu tiên được loại bỏ, vì các ràng buộc là nội dung trọng tâm của gạch đầu dòng thứ hai
- Trong mục 5.1.3, “tiêu chí hoàn thành” được sử dụng trong ngữ cảnh tiêu chí nhị phân “có/không”, chứ không phải như từ đồng nghĩa của “tiêu chí kết thúc”, nên thuật ngữ phù hợp đã được thay đổi
- Trong mục 5.1.6, mối quan hệ giữa các tầng của trong test pyramid và các mức độ cô lập trong kiểm thử đã được chỉnh sửa lại cho chính xác (tầng càng cao thì mức độ cô lập kiểm thử càng thấp). Đồng thời, cụm “độ bao phủ phù hợp” được thay bằng “mức độ bao phủ phù hợp”
- Trong mục 5.5, “các bất thường” được thay bằng “lỗi hoặc bất thường”
- Trong mục 6.2, “tỉ lệ lỗi” được thay bằng “tỉ lệ sự cố”, và cụm “quá phức tạp để con người suy luận” được thay bằng “con người có có thể xác định”
- Ngoài ra, nhiều lỗi đánh máy đã được sửa và một số thuật ngữ đã được thống nhất xuyên suốt toàn bộ đề cương

Ngoài ra, nhiều lỗi đánh máy đã được sửa và một số thuật ngữ đã được thống nhất xuyên suốt toàn bộ đề cương.

GHI CHÚ PHÁT HÀNH CHO PHIÊN BẢN 4.0

Đề cương Nền tảng phiên bản v4.0 là bản cập nhật lớn dựa trên đề cương Mức độ Nền tảng phiên bản v3.1.1 và đề cương Agile Tester 2014. Vì lý do này, sẽ không có ghi chú phát hành chi tiết cho từng chương và từng mục. Tuy nhiên, phần dưới đây cung cấp phần tóm tắt các thay đổi chính. Ngoài ra, trong tài liệu Ghi chú Phát hành riêng, ISTQB® cung cấp khả năng truy vết giữa các Mục tiêu Học tập trong phiên bản Đề cương Mức độ Nền tảng phiên bản v3.1.1, phiên bản Đề cương Agile Tester 2014, và các mục tiêu học tập trong đề cương Mức độ Nền tảng phiên bản v4.0 mới, cho thấy những mục tiêu học tập nào đã được bổ sung, cập nhật, hoặc loại bỏ.

Tại thời điểm đề cương này được biên soạn (2022–2023), đã có hơn một triệu người tại hơn 100 quốc gia đã tham gia kỳ thi ISTQB® Mức độ Nền tảng, và hơn 800.000 tester được chứng nhận trên toàn thế giới. Với giả định rằng tất cả những người này đều đã đọc qua Đề cương Nền tảng để có thể vượt qua kỳ thi, điều này khiến Đề cương Nền tảng nhiều khả năng trở thành tài liệu về kiểm thử phần mềm được đọc nhiều nhất từ trước đến nay. Bản cập nhật lớn này được thực hiện nhằm tôn trọng di sản đó, đồng thời nâng cao nhận thức của hàng trăm nghìn người hơn nữa về mức chất lượng mà tổ chức ISTQB® mang lại cho cộng đồng kiểm thử toàn cầu.

Trong phiên bản này, tất cả các mục tiêu học tập đã được chỉnh sửa để trở nên độc lập hơn, đồng thời tạo ra khả năng truy vết một-một giữa các mục tiêu học tập và các mục trong đề cương, nhờ đó tránh tình trạng có nội dung nhưng không có mục tiêu học tập tương ứng. Mục tiêu là giúp phiên bản này dễ đọc hơn, dễ hiểu hơn, dễ học hơn, và dễ dịch hơn, đồng thời tập trung vào việc gia tăng tính hữu ích trong thực tiễn và cân bằng giữa kiến thức và kỹ năng.

Phiên bản phát hành chính này đã thực hiện các thay đổi sau:

- Giảm quy mô tổng thể của đề cương. Đề cương không phải là sách giáo khoa, mà là tài liệu dùng để phác thảo các thành phần cơ bản của một khóa học nhập môn kiểm thử phần mềm, bao gồm các chủ đề cần được đề cập và mức độ cần bao phủ. Vì vậy, cụ thể:
    - Trong hầu hết các trường hợp, các ví dụ được loại bỏ khỏi nội dung văn bản. Việc cung cấp ví dụ cũng như các bài thực hành trong quá trình đào tạo là nhiệm vụ của nhà cung cấp đào tạo
    - “Danh sách kiểm tra biên soạn đề cương” đã được áp dụng, trong đó đề xuất độ dài tối đa của nội dung cho các mục tiêu học tập ở từng cấp độ K-level (K1 = tối đa 10 dòng, K2 = tối đa 15 dòng, K3 = tối đa 25 dòng)
- Giảm số lượng mục tiêu học tập (LO) so với đề cương Mức độ Nền tảng phiên bản v3.1.1 và Agile Tester 2014:
    - 14 LO cấp độ K1, so với 21 LO trong bản Nền tảng v3.1.1 (15) và Agile Tester 2014 (6)
    - 42 LO cấp độ K2, so với 53 LO trong bản Nền tảng v3.1.1 (40) và Agile Tester 2014 (13)
    - 8 LO cấp độ K3, so với 15 LO trong bản Nền tảng v3.1.1 (7) và Agile Tester 2014 (8)
- Bổ sung thêm nhiều tài liệu tham khảo đến các sách, và bài viết kinh điển, và/hoặc có uy tín về kiểm thử phần mềm, cùng các chủ đề liên quan
    - Phần về kỹ năng kiểm thử được mở rộng và cải thiện
    - Bổ sung phần mô tả cách tiếp cận toàn đội ngũ (K1)
    - Phần về sự độc lập của hoạt động kiểm thử được chuyển từ Chương 5 sang Chương 1
- Các thay đổi lớn trong Chương 2 (Kiểm thử xuyên suốt SDLC)
    - Các mục 2.1.1 và 2.1.2 được viết lại và cải thiện; các mục tiêu học tập tương ứng cũng được điều chỉnh
    - Tăng cường tập trung vào các thực hành như: phương pháp phát triển theo hướng kiểm thử trước (K1), shift left (K2), và retrospective (K2)
    - Bổ sung mục mới về kiểm thử trong bối cảnh DevOps (K2)
    - Mức kiểm thử tích hợp được tách thành hai mức kiểm thử riêng biệt: kiểm thử tích hợp thành phần và kiểm thử tích hợp hệ thống

- Các thay đổi lớn trong Chương 3 (Kiểm thử tĩnh)
    - Phần về kỹ thuật rà soát, cùng với các muc tiêp học tập mức K3 (áp dụng các kỹ thuật rà soát), đã được loại bỏ
- Các thay đổi lớn trong Chương 4 (Phân tích và Thiết kế Kiểm thử)
    - Kiểm thử dựa vào use case được loại bỏ (nhưng vẫn còn trong đề cương Mức độ Nâng cao CTAL Test Analyst).
    - Tăng cường tập trung vào phương pháp kiểm thử dựa trên sự cộng tác: bổ sung mục tiêu học tập mức K3 mới về việc áp dụng ATDD để thiết kế test case, và hai mục tiêu học tập mức K2 mới về user story và các tiêu chí chấp nhận
    - Kiểm thử quyết định và bao phủ được thay thế bằng kiểm thử nhánh và bao phủ (thứ nhất, độ bao phủ nhánh được sử dụng phổ biến hơn trong thực tế; thứ hai, các tiêu chuẩn khác nhau có định nghĩa “quyết định” khác nhau, trong khi “nhánh” nhất quán hơn; thứ ba, điều này khắc phục một lỗi tinh vi nhưng nghiêm trọng của đề cương Nền tảng 2018 cũ khi cho rằng “100% bao phủ quyết định kéo theo bao phủ 100% câu lệnh” — điều này không đúng đối với các chương trình không có quyết định)
    - Phần về giá trị của kiểm thử hộp trắng đã được cải thiện
- Các thay đổi lớn trong Chương 5 (Quản lý Các Hoạt động Kiểm thử)
    - Phần về chiến lược/phương pháp tiếp cận kiểm thử đã được loại bỏ
    - Bổ sung mục tiêu học tập mức K3 mới về các kỹ thuật ước lượng công sức kiểm thử
    - Tăng cường tập trung vào các khái niệm và công cụ quen thuộc liên quan đến Agile trong quản lý kiểm thử: lập kế hoạch iteration và release (K1), test pyramid (K1), và testing quadrant (K2)
    - Phần về quản lý rủi ro được cấu trúc lại rõ ràng hơn thông qua mô tả bốn hoạt động chính: nhận diện rủi ro, đánh giá rủi ro, giảm thiểu rủi ro, và giám sát rủi ro
- Các thay đổi lớn trong Chương 6 (Công cụ Kiểm thử)
    - Nội dung về tự động hóa kiểm thử đã được tinh gọn do được xem là quá nâng cao đối với cấp độ nền tảng — các phần về lựa chọn công cụ, thực hiện dự án thử nghiệm, và triển khai công cụ vào tổ chức đã được loại bỏ