**ĐỀ CƯƠNG ÔN TẬP**

**MÔN: ĐẢM BẢO CHẤT LƯỢNG VÀ KIỂM THỬ PHẦN MỀM**

# Chương 1

1. **Định nghĩa phần mềm; chất lượng phần mềm; đảm bảo chất lượng phần mềm.**

|  |
| --- |
| **Định nghĩa phần mềm:**  Phần mềm là tập hợp các chương trình máy tính, thủ tục, dữ liệu và tài liệu liên quan phục vụ cho việc xử lý thông tin.  - Chương trình máy tính: bao gồm các mã lệnh.  - Thủ tục: quy trình, cách thức, kỹ thuật phát triển phần mềm.  - Dữ liệu: dữ liệu cần thiết để phần mềm hoạt động hiệu quả.  - Tài liệu:  + Cho nhà phát triển: tài liệu yêu cầu, thiết kế, mô tả chương trình,…  + Cho người dùng: hướng dẫn sử dụng.  + Cho nhân viên bảo trì: thông tin về lỗi, thay đổi hoặc bổ sung chức năng. |
| **Chất lượng phần mềm:**  Chất lượng phần mềm là mức độ mà sản phẩm phần mềm đáp ứng đầy đủ các yêu cầu đã được xác định, từ đó thỏa mãn mong đợi của các bên liên quan. Các bên liên quan gồm:   * **Khách hàng**: Muốn phần mềm đáp ứng đúng nhu cầu. * **Nhà phát triển**: Muốn phần mềm được thiết kế đúng chuẩn và có tài liệu đầy đủ. * **Quản lý**: Muốn các giai đoạn phát triển hoàn thành đúng tiến độ.   => **Phần mềm có chất lượng cao** khi mức độ thỏa mãn mong đợi của các bên liên quan càng lớn. |
| **Lưu ý:**  **Sản phẩm có chất lượng** là sản phẩm đáp ứng tốt các yêu cầu và mong đợi của các bên liên quan, nhưng mức độ "tốt" có thể khác nhau tùy ngữ cảnh và đối tượng:   * **Người làm sản phẩm**: Quan tâm sản phẩm có được thiết kế và phát triển đúng không. * **Khách hàng**: Muốn sản phẩm đáp ứng nhu cầu và mong muốn của họ. * **Người quản lý**: Muốn sản phẩm hoàn thành đúng thời hạn và chi phí hợp lý.   => Một sản phẩm tốt với người này có thể chưa tốt với người khác, tùy thuộc vào mong đợi và hoàn cảnh cụ thể. |
| **Đảm bảo chất lượng phần mềm:**  SQA là một tập hợp các hoạt động có hệ thống nhằm đảm bảo phần mềm được phát triển và bảo trì theo các tiêu chuẩn chất lượng đã định sẵn. Không chỉ tập trung vào việc tìm và sửa lỗi, SQA còn nhấn mạnh ngăn ngừa lỗi từ đầu thông qua quy trình hiệu quả và tuân thủ tiêu chuẩn chất lượng.  **Mục tiêu:** Đảm bảo phần mềm đạt chất lượng cao, đáp ứng nhu cầu người dùng và mang lại giá trị cho doanh nghiệp. SQA cần tính khách quan, độc lập với áp lực kỹ thuật, quản lý hoặc tài chính, đóng vai trò như nền tảng cho sự thành công của dự án phần mềm. |

1. **Liệt kê các yếu tố chất lượng theo mô hình McCall.(slide 18/ chapter 1)**

|  |
| --- |
| Đã tải lên ảnh **1. Hoạt động sản phẩm (Product Operation)** **- Tính đúng đắn (Correctness)**  **- Độ tin cậy (Reliability)**  **- Tính hiệu quả (Efficiency)**  **- Tính toàn vẹn (Integrity)**  **- Khả năng sử dụng (Usability)** 2. **Sửa đổi sản phẩm (Product Revision)** **- Khả năng bảo trì (Maintainability)**  **- Tính linh hoạt (Flexibility)**  **- Khả năng kiểm thử (Testability)** 3. **Chuyển đổi sản phẩm (Product Transition)** **- Tính di động (Portability)**  **- Khả năng tái sử dụng (Reusability)**  **- Khả năng tương tác (Interoperability)** |

1. **Liệt kê các thành phần của hệ thống SQA. (slice 30/ chapter 1)**

|  |
| --- |
| **1. Thành phần tiền dự án - Pre-project**  Mục tiêu: Đảm bảo các thỏa thuận giữa các bên rõ ràng và có kế hoạch cụ thể.   * **Rà soát hợp đồng**: Đảm bảo điều khoản hợp đồng được kiểm tra kỹ trước khi ký. * **Lập kế hoạch phát triển và kế hoạch chất lượng**:   + Kế hoạch phát triển: Lên lịch biểu, phân bổ nguồn lực, lập nhóm, phân công công việc, dự phòng rủi ro.   + Kế hoạch chất lượng: Quy trình review, kiểm thử (test), quản lý cấu hình, và nghiệm thu.   **2. Thành phần đánh giá vòng đời dự án - Project Life Cycle SQA**  Mục tiêu: Đảm bảo chất lượng xuyên suốt vòng đời phần mềm.   * **Reviews**: Đánh giá (Review) là một phần của kiểm thử phần mềm. * **Software testing**: Thành phần chính của SQA. * **Software maintenance**: Dịch vụ bảo trì sau khi phần mềm hoạt động. * **SQA of external participants**: Đảm bảo chất lượng cho bên tham gia bên ngoài.   **3. Thành phần cơ sở hạ tầng chất lượng - Infrastructure**  Mục tiêu: Đầu tư cơ sở hạ tầng tốt để giảm lỗi và nâng cao năng suất.   * **Thủ tục và hướng dẫn công việc**: Quy trình làm việc cụ thể. * **Công cụ hỗ trợ (Supporting devices)**: Các mẫu (templates), checklist. * **Đào tạo (Training)**: Nâng cao kỹ năng và nhận thức. * **Hành động phòng ngừa và khắc phục**: Thu thập thông tin để rút kinh nghiệm. * **Quản lý cấu hình**: Xử lý thay đổi trong hệ thống. * **Kiểm soát tài liệu**: Đảm bảo tài liệu được quản lý chặt chẽ.   **4. Thành phần quản lý chất lượng - Quality Management**  Mục tiêu: Quản lý và đo lường chất lượng phần mềm.   * **Project progress control**: Kiểm soát tài nguyên, lịch biểu, ngân sách, rủi ro. * **Software quality metrics**: Ví dụ:   + Số lỗi/1KLOC để tính mật độ lỗi.   + Số lượng cuộc gọi hỗ trợ/năm để đánh giá vận hành. * **Software quality costs**: Bao gồm chi phí phát triển và thất bại.   **5. Thành phần chuẩn hóa và đánh giá - Standards**  Mục tiêu: Tuân thủ các tiêu chuẩn quốc tế và đánh giá tiến trình.   * **Chuẩn quản lý chất lượng**: ISO, IEEE,...   **6. Thành phần tổ chức và con người - Organizational Base (Human)**  Mục tiêu: Đảm bảo chất lượng đội ngũ thực hiện SQA.   * **Đội ngũ quản lý và đơn vị SQA**:   + SQA trustees (ủy viên).   + SQA committee members (ủy ban).   + SQA forum members (diễn đàn). |
| **Quy trình CAPA (Corrective and Preventive Actions)**   1. **Thu thập thông tin**: Từ lỗi, phản hồi hoặc vấn đề phát sinh. 2. **Phân tích thông tin**: Tìm nguyên nhân và ảnh hưởng. 3. **Đề xuất giải pháp**: Và cách triển khai thực hiện. 4. **Theo dõi kết quả**: Đảm bảo các giải pháp đạt hiệu quả |

1. **Nêu các thành phần liên quan cơ sở hạ tầng trong SQA. (slice 30/ chapter 1)**

|  |
| --- |
| **Thủ tục và hướng dẫn công việc**: Xây dựng và tuân thủ các quy trình, hướng dẫn rõ ràng để thực hiện công việc một cách nhất quán.  **Công cụ hỗ trợ - Supporting Quality Devices**: Sử dụng các mẫu (templates) và danh sách kiểm tra (checklists) để chuẩn hóa các hoạt động.  **Đào tạo - Training**: Đào tạo đội ngũ nhân viên để nâng cao kỹ năng và nhận thức về quản lý chất lượng.  **Hành động phòng ngừa và khắc phục - Preventive and Corrective Actions**: Thu thập thông tin từ các lỗi hoặc vấn đề đã xảy ra để rút kinh nghiệm và đưa ra giải pháp ngăn ngừa, sửa chữa.  **Quản lý cấu hình - Configuration Management**: Theo dõi và quản lý các thay đổi liên quan đến hệ thống phần mềm để tránh sự nhầm lẫn hoặc sai sót.  **Kiểm soát tài liệu - Documentation Control**: Đảm bảo rằng tất cả tài liệu liên quan được lưu trữ, cập nhật và kiểm soát một cách phù hợp. |

1. **Nêu và giải thích 7 nguyên tắc kiểm thử PM. (slide 52 / chapter 1)**

|  |
| --- |
| **1. Kiểm thử cho thấy sự hiện diện của lỗi (Kiểm thử để tìm lỗi**): Kiểm thử giúp tìm lỗi, nhưng không thể khẳng định phần mềm không còn lỗi. Dù kiểm thử kỹ đến đâu, lỗi vẫn có thể ẩn giấu.  **2. Kiểm thử toàn diện là không thể (Không thể kiểm thử mọi thứ):** Kiểm tra hết tất cả trường hợp là không khả thi vì có quá nhiều tổ hợp dữ liệu. Thay vì kiểm thử toàn diện, cần **tập trung vào phân tích rủi ro** để **ưu tiên** những gì cần kiểm thử trước, kiểm thử nhiều nhất và những gì không cần kiểm thử.  **3. Kiểm thử sớm:** Bắt đầu kiểm thử ngay từ giai đoạn đầu của dự án để phát hiện và sửa lỗi sớm, tránh chi phí cao khi sửa lỗi sau này.  **4. Phân cụm lỗi (Lỗi tập trung ở vài chỗ):** Lỗi thường xuất hiện nhiều ở những phần phức tạp hoặc dễ gặp vấn đề. Nên tập trung kiểm tra những khu vực này để tiết kiệm thời gian.  **5. Nghịch lý thuốc trừ sâu:** Nếu dùng đi dùng lại một bộ kiểm tra, sẽ không tìm ra lỗi mới. Cần thay đổi cách kiểm tra và dữ liệu kiểm tra để tìm thêm lỗi.  **6. Kiểm thử phụ thuộc ngữ cảnh (Kiểm thử phụ thuộc vào loại phần mềm):** Kiểm thử được thực hiện **khác nhau** trong các **ngữ cảnh khác nhau**. Không nên áp dụng cùng một bộ kiểm thử cho mọi phần mềm vì các phần mềm khác nhau có các yêu cầu, chức năng và mục đích khác nhau.. Ví dụ: phần mềm máy bay cần kiểm tra độ chính xác cao, trong khi website bán hàng cần kiểm tra tính dễ dùng và bảo mật.  **7. Ngụy biện không có lỗi:** Dù kiểm thử xong không còn lỗi, phần mềm vẫn có thể thất bại nếu không đáp ứng nhu cầu và mong đợi của người dùng. |

# Chương 2

1. Nêu quy trình kiểm thử cơ bản. Nêu các công việc chính trong từng bước của quy trình.

|  |
| --- |
| Quy trình kiểm thử phần mềm cơ bản bao gồm các bước sau:  ●**Phân tích và thiết kế kiểm thử**:  ○Đánh giá khả năng kiểm thử của cơ sở kiểm thử và đối tượng kiểm thử.1 Ví dụ, yêu cầu "phần mềm cần phản hồi đủ nhanh" là không kiểm thử được.Một yêu cầu kiểm thử được tốt hơn là "phần mềm cần phản hồi trong 5 giây khi có 20 người đăng nhập".  ○Xác định và ưu tiên các điều kiện kiểm thử dựa trên phân tích các mục kiểm thử, thông số kỹ thuật, hành vi và cấu trúc.  ●**Thực hiện kiểm thử**: Chuẩn bị môi trường và các công cụ kiểm thử cần thiết.  ●**Đánh giá tiêu chí thoát**: Xác định xem có cần thêm kiểm thử hay không.  ●**Báo cáo kết quả**: Báo cáo các lỗi kiểm thử.  Sau khi thực hiện kiểm thử, tester thường thực hiện **viết báo cáo kiểm thử**.23 Tài liệu cần thiết để viết kế hoạch kiểm thử bao gồm các câu chuyện người dùng, tài liệu kế hoạch dự án, tài liệu giải pháp và kế hoạch kiểm thử. |

1. Nêu và giải thích từng mức kiểm thử (*test levels*) theo ISTQB.

|  |
| --- |
| Theo ISTQB, có 4 mức kiểm thử chính:   * **Kiểm thử thành phần (Component testing):**   + Mức kiểm thử này tập trung vào việc kiểm tra từng thành phần phần mềm riêng lẻ, chẳng hạn như chương trình, hàm, thủ tục, phương thức hoặc đối tượng.   + Mục tiêu chính là xác minh xem từng thành phần hoạt động đúng với thiết kế và đặc tả của nó.   + Ví dụ: Viết trình điều khiển (driver) cho thành phần X chưa sẵn sàng để kiểm thử, trình điều khiển này sẽ trả về giá trị cho các thành phần Y và Z đã sẵn sàng. * **Kiểm thử tích hợp (Integration testing):**   + Kiểm thử tích hợp tập trung vào việc kiểm tra **sự tương tác giữa các thành phần phần mềm** sau khi chúng được tích hợp với nhau.   + Mục tiêu là phát hiện lỗi phát sinh từ sự kết hợp của các thành phần.   + Có ba phương pháp kiểm thử tích hợp tăng dần:     - **Từ trên xuống (top-down):** Kiểm thử từ thành phần cao nhất (ví dụ: giao diện người dùng) xuống các thành phần thấp hơn.     - **Từ dưới lên (bottom-up):** Kiểm thử từ các thành phần thấp nhất lên các thành phần cao hơn.     - **Tăng dần theo chức năng (functional incremental):** Tích hợp và kiểm thử dựa trên các chức năng được ghi trong đặc tả chức năng. * **Kiểm thử hệ thống (System testing):**   + Kiểm thử hệ thống được thực hiện trên **toàn bộ hệ thống**, nhằm xác minh xem hệ thống đáp ứng các yêu cầu đã được chỉ định.   + Cơ sở kiểm thử bao gồm đặc tả yêu cầu, quy trình nghiệp vụ, trường hợp sử dụng và báo cáo phân tích rủi ro.   + Kiểm thử hệ thống nên được thực hiện trong môi trường kiểm thử càng gần với môi trường đích cuối cùng càng tốt.   + Mục tiêu là đảm bảo rằng hệ thống hoạt động như mong đợi và đáp ứng các yêu cầu chức năng và phi chức năng. * **Kiểm thử chấp nhận (Acceptance testing):**   + Kiểm thử chấp nhận thường tập trung vào việc **xác nhận** (validation) xem hệ thống có đáp ứng nhu cầu của người dùng hay không.   + Trách nhiệm kiểm thử chấp nhận thường thuộc về người dùng hoặc khách hàng.   + Mục tiêu chính không phải là tìm kiếm lỗi, mà là **xác nhận rằng hệ thống phù hợp với mục đích sử dụng**.   + Có một số hình thức kiểm thử chấp nhận phổ biến:     - **Kiểm thử chấp nhận của người dùng (User acceptance test):** Xác minh xem hệ thống có phù hợp với mục đích sử dụng bởi người dùng doanh nghiệp hay không.     - **Kiểm thử chấp nhận vận hành (Operational acceptance test):** Bao gồm kiểm tra các chức năng như sao lưu/khôi phục, phục hồi sau thảm họa, quản lý người dùng và các tác vụ bảo trì.     - **Kiểm thử chấp nhận hợp đồng/quy định (Contract/regulation acceptance test):** Thực hiện dựa trên các tiêu chí chấp nhận của hợp đồng và các quy định cần tuân thủ (ví dụ: quy định của chính phủ, pháp lý hoặc an toàn).     - **Kiểm thử alpha/beta:** Được sử dụng cho các hệ thống được phát triển cho thị trường đại chúng (ví dụ: phần mềm thương mại).   Tóm lại, mỗi mức kiểm thử đều có mục tiêu, phạm vi và phương pháp riêng, góp phần vào việc đảm bảo chất lượng phần mềm. |

1. Nêu và giải thích các loại kiểm thử (*test types*) theo ISTQB.

|  |
| --- |
| Theo ISTQB, **kiểu kiểm thử (test type)** tập trung vào một mục tiêu kiểm thử cụ thể. Có bốn loại kiểm thử chính:   * **Kiểm thử chức năng (Functional test):** Kiểm tra các chức năng của hệ thống hoặc thành phần, tập trung vào việc hệ thống "làm gì".   + Chức năng thường được mô tả trong các tài liệu như đặc tả yêu cầu, đặc tả chức năng hoặc trường hợp sử dụng.   + Kiểm thử chức năng có thể được thực hiện ở tất cả các mức kiểm thử.   + Có hai cách tiếp cận để kiểm thử chức năng:     - **Dựa trên yêu cầu (Requirements-based):** Thiết kế kiểm thử dựa trên đặc tả yêu cầu chức năng của hệ thống.     - **Dựa trên quy trình nghiệp vụ (Business process-based):** Sử dụng kiến thức về quy trình nghiệp vụ để mô tả các tình huống kinh doanh thực tế. * **Kiểm thử phi chức năng (Non-functional test):** Kiểm tra các **đặc tính chất lượng phi chức năng** của hệ thống, tập trung vào việc hệ thống "hoạt động tốt như thế nào".   + Ví dụ: Kiểm thử hiệu năng, kiểm thử tải, kiểm thử ứng suất, kiểm thử khả năng sử dụng, kiểm thử khả năng bảo trì, kiểm thử độ tin cậy, kiểm thử khả năng di chuyển,...   + Kiểm thử phi chức năng có thể được thực hiện ở tất cả các mức kiểm thử. * **Kiểm thử cấu trúc (Structural test):** Còn được gọi là kiểm thử "hộp trắng" (white-box), tập trung vào **cấu trúc bên trong** của hệ thống hoặc thành phần.   + Kiểm thử cấu trúc có thể được thực hiện ở tất cả các mức kiểm thử, nhưng thường được áp dụng nhiều nhất ở mức kiểm thử thành phần và kiểm thử tích hợp. * **Kiểm thử liên quan đến thay đổi (Testing related to changes):** Được thực hiện sau khi có thay đổi đối với phần mềm, bao gồm:   + **Kiểm thử xác nhận (Confirmation testing / Re-testing):** Xác nhận rằng lỗi đã được sửa.   + **Kiểm thử hồi quy (Regression testing):** Kiểm tra xem các thay đổi có gây ra lỗi mới hay ảnh hưởng đến chức năng hiện có hay không.   + Các loại kiểm thử cụ thể liên quan đến thay đổi có thể bao gồm tất cả các loại kiểm thử khác.   Tóm lại, việc lựa chọn loại kiểm thử phù hợp phụ thuộc vào mục tiêu kiểm thử cụ thể và giai đoạn phát triển phần mềm. |

1. Phân biệt các khái niệm: kiểm thử xác nhận (*confirmation testing*) và kiểm thử hồi quy (*regression testing*); kiểm thử chức năng và kiểm thử phi chức năng; *walkthrough* và *inspection*; *alpha test* và *beta test*.

|  |
| --- |
| **Phân biệt các khái niệm kiểm thử phần mềm**  **Kiểm thử xác nhận (Confirmation testing) và Kiểm thử hồi quy (Regression testing)**   * **Kiểm thử xác nhận (hay re-testing):** được thực hiện **sau khi sửa lỗi**, nhằm mục đích **xác nhận lỗi đã được khắc phục**. Tester sẽ **chạy lại các test case** đã bị lỗi trước đó để đảm bảo rằng lỗi không còn tồn tại. * **Kiểm thử hồi quy:** được thực hiện **sau khi có bất kỳ thay đổi nào** trong phần mềm (sửa lỗi, thêm tính năng mới,...), nhằm mục đích **đảm bảo rằng các thay đổi không gây ra lỗi mới hoặc ảnh hưởng đến chức năng hiện có**. Tester sẽ chạy lại một tập hợp các test case đã được thực hiện trước đó để kiểm tra sự ổn định của hệ thống sau khi thay đổi.   **Kiểm thử chức năng (Functional test) và Kiểm thử phi chức năng (Non-functional test)**   * **Kiểm thử chức năng:** tập trung vào việc **kiểm tra các chức năng** của hệ thống hoặc thành phần. Tester sẽ kiểm tra xem hệ thống có **"làm những gì nó được yêu cầu làm"** hay không. Ví dụ: Kiểm tra xem nút đăng nhập có hoạt động hay không, kiểm tra xem hệ thống có lưu trữ thông tin người dùng chính xác hay không. * **Kiểm thử phi chức năng:** tập trung vào việc **kiểm tra các đặc tính chất lượng phi chức năng** của hệ thống, chẳng hạn như hiệu năng, bảo mật, khả năng sử dụng, độ tin cậy. Tester sẽ kiểm tra xem hệ thống có **"hoạt động tốt như thế nào"**. Ví dụ: Kiểm tra xem hệ thống có thể xử lý bao nhiêu người dùng đồng thời, kiểm tra xem hệ thống có dễ sử dụng hay không, kiểm tra xem hệ thống có an toàn trước các cuộc tấn công mạng hay không.   **Walkthrough và Inspection**  Cả walkthrough và inspection đều là **các phương pháp đánh giá tĩnh (static testing)**, được thực hiện để **phát hiện lỗi sớm** trong các tài liệu, mã nguồn hoặc các sản phẩm phần mềm khác.   * **Walkthrough:** là một **cuộc họp không chính thức** với mục đích **đánh giá hoặc cung cấp thông tin** về sản phẩm phần mềm. Người viết tài liệu sẽ **trình bày** sản phẩm với các thành viên khác trong nhóm, và nhóm sẽ **đặt câu hỏi** và **đưa ra nhận xét** về sản phẩm. Walkthrough tập trung vào việc **hiểu** và **chia sẻ kiến thức** về sản phẩm. * **Inspection:** là một **quá trình đánh giá chính thức** hơn walkthrough. Inspection có **mục tiêu rõ ràng** là **tìm kiếm lỗi** trong sản phẩm. Một **nhóm kiểm tra được đào tạo bài bản** sẽ **kiểm tra kỹ lưỡng** sản phẩm theo **danh sách kiểm tra** được xác định trước. Inspection tập trung vào việc **phát hiện lỗi** và **cải thiện chất lượng** của sản phẩm.   **Alpha test và Beta test**  Cả alpha test và beta test đều là các hình thức **kiểm thử chấp nhận (acceptance testing)**, được thực hiện **sau khi phần mềm đã được phát triển xong** và **trước khi phát hành chính thức**. Mục tiêu của cả hai loại kiểm thử này là **thu thập phản hồi từ người dùng thực tế** để **cải thiện chất lượng** của phần mềm.   * **Alpha test:** được thực hiện **tại địa điểm của nhà phát triển**. Người dùng tiềm năng và các thành viên của tổ chức phát triển được mời sử dụng hệ thống và cung cấp phản hồi. * **Beta test (kiểm thử thực địa):** được thực hiện **tại địa điểm của khách hàng** hoặc người dùng tiềm năng. Khách hàng hoặc người dùng tiềm năng sử dụng sản phẩm trong môi trường thực tế của họ và cung cấp phản hồi về lỗi, trải nghiệm sử dụng, và đề xuất cải tiến. |

# Chương 3

1. **Quality support devices gồm có những gì. Lợi ích của chúng. (slide 14&16 / chapter 3)**

|  |
| --- |
| **Quality Support Devices (Thiết bị hỗ trợ chất lượng) trong SQA bao gồm:**   * **Biểu mẫu (Templates):** là định dạng được tạo bởi các đơn vị hoặc tổ chức, được áp dụng khi biên soạn báo cáo hoặc một số loại tài liệu khác.Mục đích của biểu mẫu là đảm bảo **tính nhất quán** và **đầy đủ** cho các tài liệu được tạo ra. Ví dụ về biểu mẫu bao gồm Kế hoạch Kiểm thử Phần mềm (STP), Mô tả Kiểm thử Phần mềm (STD), Báo cáo Kiểm thử Phần mềm (STR), Đặc tả Yêu cầu Phần mềm (SRS), v.v. * **Danh sách kiểm tra (Checklists):** là danh sách các mục được tạo riêng cho từng loại tài liệu, cần được hoàn thành trước khi thực hiện một hoạt động.Danh sách kiểm tra giúp **đảm bảo** rằng tất cả các **bước cần thiết** đã được **thực hiện** và **không có bước nào bị bỏ sót**. |
| **Đối với nhóm phát triển:**  - Giúp **dễ dàng** hơn trong việc **chuẩn bị tài liệu**.  - Tài liệu được chuẩn bị **đầy đủ** hơn.  - Giúp **tích hợp** thành viên **mới** vào nhóm **dễ dàng** hơn.  - Giúp **xem xét** tài liệu **dễ dàng** hơn.  **Đối với nhóm bảo trì phần mềm:**  -Giúp **xác định vị trí thông tin dễ dàng** hơn. |

1. **Quản lý cấu hình (Configuration management) là gì. Nêu các hoạt động chính trong quản lý cấu hình. Giải thích hoạt động kiểm soát phiên bản (phần independent development). (slide 43-45/ chapter 3)**

|  |
| --- |
| **Quản lý cấu hình (Configuration management)**  **Quản lý cấu hình (CM)** trong SQA là thành phần được giao nhiệm vụ **quản lý các thay đổi** và cung cấp câu trả lời chính xác cho các câu hỏi liên quan đến các phiên bản và thay đổi của phần mềm.Ví dụ, CM giúp trả lời các câu hỏi như:  ●Ai có thể cung cấp bản sao chính xác của phiên bản 4.1 của phần mềm TMY năm ngoái?  ●Những thay đổi nào đã được giới thiệu trong phiên bản phần mềm mới? |
| **Các hoạt động chính trong quản lý cấu hình :**  Quản lý thay đổi (Change Management): Theo dõi các yêu cầu thay đổi phần mềm từ khách hàng và nhà phát triển. Quá trình này bao gồm việc phân tích chi phí và lợi ích của các thay đổi được đề xuất, phê duyệt những thay đổi đáng giá và theo dõi những thành phần trong hệ thống đã được thay đổi.  Quản lý phiên bản (Version Management): Theo dõi nhiều phiên bản của các thành phần hệ thống và đảm bảo rằng các thay đổi được thực hiện đối với các thành phần bởi các nhà phát triển khác nhau không can thiệp lẫn nhau. Hoạt động này có thể được coi là quá trình quản lý các codeline (dòng mã) và baseline (cột mốc).  Xây dựng hệ thống (System Building): Lắp ráp các thành phần chương trình, dữ liệu và thư viện, sau đó biên dịch chúng để tạo ra một hệ thống thực thi. Nhà phát triển kiểm tra mã vào hệ thống quản lý phiên bản trước khi nó được xây dựng. Việc xây dựng hệ thống có thể dựa vào các thư viện bên ngoài không được bao gồm trong hệ thống quản lý phiên bản.  Quản lý bản phát hành (Release Management): Chuẩn bị phần mềm để phát hành ra bên ngoài và theo dõi các phiên bản hệ thống đã được phát hành cho khách hàng sử dụng. Khi bản phát hành hệ thống được tạo ra, nó phải được ghi lại để đảm bảo rằng nó có thể được tạo lại chính xác trong tương lai. |
| **Giải thích hoạt động kiểm soát phiên bản (phần independent development).**  **Kiểm soát phiên bản (Version Control)** là quá trình theo dõi các phiên bản khác nhau của các thành phần phần mềm hoặc các mục cấu hình và các hệ thống mà các thành phần này được sử dụng. Nó cũng liên quan đến việc đảm bảo rằng các thay đổi được thực hiện bởi các nhà phát triển khác nhau đối với các phiên bản này không can thiệp lẫn nhau.  **Independent Development** trong Version Management System (Hệ thống Quản lý Phiên bản) đảm bảo việc các nhà phát triển có thể làm việc độc lập trên cùng một thành phần phần mềm mà không ảnh hưởng đến nhau.Hệ thống sẽ theo dõi các thành phần đã được kiểm tra để chỉnh sửa và đảm bảo rằng các thay đổi được thực hiện cho một thành phần bởi các nhà phát triển khác nhau không gây xung đột. Điều này giúp cho việc phát triển phần mềm trở nên hiệu quả và linh hoạt hơn. |

1. **Tại sao cần phải kiểm soát tài liệu. (slide 71/ chapter 3)**

|  |
| --- |
| **- Để đảm bảo chất lượng của tài liệu**  - Đảm bảo tính đầy đủ, phù hợp về mặt kỹ thuật và tuân thủ các quy trình, hướng dẫn về cấu trúc tài liệu (sử dụng mẫu, ký hiệu đúng cách).  **- Để đảm bảo tính khả dụng trong tương lai**: Tài liệu cần thiết cho việc bảo trì hệ thống phần mềm, phát triển thêm hoặc xử lý các khiếu nại dự kiến từ khách hàng.  **- Để hỗ trợ điều tra lỗi và phân công trách nhiệm**: Tài liệu là cơ sở để phân tích nguyên nhân lỗi phần mềm, thực hiện hành động khắc phục và xác định trách nhiệm các bên liên quan. |

# Chương 4

1. **Phân biệt process metric và product metric.**

|  |
| --- |
| Trong quản lý chất lượng phần mềm (SQA), Process Metric và Product Metric là hai loại chỉ số quan trọng được sử dụng để đánh giá hiệu quả của quá trình phát triển và chất lượng của sản phẩm phần mềm. |

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Process Metric** | **Product Metric** |
| **Định nghĩa** | Đo lường **hiệu quả của quá trình phát triển phần mềm** | Đo lường **chất lượng của sản phẩm phần mềm** |
| **Mục tiêu** | Cải thiện quy trình phát triển phần mềm | Đảm bảo chất lượng của sản phẩm phần mềm |
| **Giai đoạn áp dụng** | Trong suốt quá trình phát triển phần mềm | Giai đoạn vận hành của sản phẩm phần mềm |
| **Ví dụ** | Tỷ lệ lỗi, thời gian hoàn thành, năng suất phát triển | Độ phức tạp mã nguồn, số lượng lỗi, khả năng sử dụng |

|  |
| --- |
| **Product Metric (Chỉ số sản phẩm):**  -Liên quan đến chất lượng của sản phẩm phần mềm sau khi được phát hành và đưa vào sử dụng.  -Mục tiêu: giúp đánh giá mức độ đáp ứng yêu cầu, độ tin cậy, hiệu năng và khả năng bảo trì của sản phẩm.  **Product Metric (Chỉ số sản phẩm):**  -Liên quan đến chất lượng của sản phẩm phần mềm sau khi được phát hành và đưa vào sử dụng.  -Mục tiêu: giúp đánh giá mức độ đáp ứng yêu cầu, độ tin cậy, hiệu năng và khả năng bảo trì của sản phẩm. |

1. **Hiểu và giải thích được 3-5 công thức cụ thể của 2 loại độ đo (process metric và product metric). (slide 23/36 chapter4)**

|  |
| --- |
| **1. Process Metrics:**  **Code Error Density (CED):**  **Công thức:** CED = NCE / KLOC  **Giải thích:** CED đo lường **mật độ lỗi mã nguồn**, được tính bằng số lượng lỗi mã nguồn (NCE) chia cho số lượng dòng mã nguồn (KLOC). Chỉ số này cho biết mức độ tập trung lỗi trong mã nguồn. Giá trị CED càng cao thì chất lượng mã nguồn càng kém.  **Weighted Code Error Density (WCED):**  **Công thức:** WCED = WCE / KLOC  **Giải thích:** Tương tự như CED, WCED cũng đo lường mật độ lỗi mã nguồn, nhưng có tính đến **mức độ nghiêm trọng của lỗi** thông qua hệ số trọng số (WCE).  **Development Error Density (DED):**  **Công thức:** DED = NDE / KLOC  **Giải thích:** DED đo lường **mật độ lỗi phát triển**, bao gồm cả lỗi thiết kế và lỗi mã nguồn (NDE), chia cho số lượng dòng mã nguồn (KLOC).  **2. Product Metrics:**  **Software System Failure Density (SSFD):**  **Công thức:** SSFD = NYF / KLMC  **Giải thích:** SSFD đo lường **mật độ lỗi hệ thống phần mềm** trong quá trình vận hành. Nó được tính bằng số lượng lỗi hệ thống được phát hiện trong một năm (NYF) chia cho số lượng dòng mã nguồn được bảo trì (KLMC).  **Weighted Software System Failure Density (WSSFD):**  **Công thức:** WSSFD = WYF / KLMC  **Giải thích:** Tương tự như SSFD, WSSFD cũng đo lường mật độ lỗi hệ thống, nhưng có tính đến **mức độ nghiêm trọng của lỗi** (WYF). |

1. **Hãy phân loại các chi phí liên quan đến chất lượng phần mềm theo mô hình cổ điển. Cho ví dụ từng loại. (slide 44/ chapter 4)**

|  |
| --- |
| **1. Costs of Control (Chi phí kiểm soát)**  **Mục tiêu**: Ngăn chặn và phát hiện lỗi để giảm lỗi đến mức chấp nhận được.  **Thành phần**:   * **Prevention Costs (Chi phí phòng ngừa)**: Đầu tư mới, cập nhật hoặc cải tiến cơ sở hạ tầng, không hướng đến dự án cụ thể nào. Gồm phát triển template, checklist, xây dựng hệ thống quản lý cấu hình, đào tạo nhân viên về SQA, rà soát chất lượng. **Ví dụ**: Mở quán ăn, chi phí ban đầu như mua thiết bị, setup bếp. * **Appraisal Costs (Chi phí thẩm định)**: Chi phí phát hiện lỗi trong sản phẩm cụ thể trước khi chuyển giao cho khách hàng. Gồm chi phí review, chi phí testing, đánh giá chất lượng của external participants.   **2. Costs of Failure of Control (Chi phí lỗi xảy ra trong kiểm soát)**  **Mục tiêu**: Xử lý lỗi không được ngăn chặn và phát hiện kịp thời.  **Thành phần**:   * **Internal Failure Costs (Chi phí thất bại nội bộ)**: Chi phí sửa lỗi sau khi kiểm tra phần mềm trong quá trình phát triển. Gồm thiết kế lại, code lại, test lại. * **External Failure Costs (Chi phí thất bại bên ngoài)**: Chi phí sửa lỗi phát hiện bởi khách hàng hoặc đội bảo trì sau khi phần mềm đã được cài đặt. Gồm * **Chi phí công khai**:   + Giải quyết khiếu nại cho khách hàng trong thời gian bảo hành.  + Sửa lỗi trong quá trình vận hành hệ thống.  + Bồi thường thiệt hại cho khách hàng.   * **Chi phí ẩn (gián tiếp)**:   + Giảm doanh số bán hàng do lỗi phần mềm (như thu hồi sản phẩm lỗi).  + Mất danh tiếng công ty, ảnh hưởng nghiêm trọng đến doanh thu. |

# Chương 5

1. **Nêu ít nhất 2 lợi ích của việc sử dụng chuẩn. (slide 7 chapter 5)**

|  |
| --- |
| **Nâng cao năng lực chuyên môn và khả năng hợp tác:**  Sử dụng chuẩn giúp áp dụng phương pháp luận và quy trình chuyên nghiệp.  Tăng cường hiểu biết và hợp tác giữa các thành viên, nhóm dự án, nhà phát triển và các bên liên quan.  CMM/CMMI cải thiện quy trình phát triển, dự đoán chất lượng, thời gian, chi phí và quản lý rủi ro.  Chuẩn IEEE (829, 1012, 1028) cung cấp hướng dẫn kiểm thử chuyên nghiệp, nâng cao chất lượng phần mềm.  **Đảm bảo chất lượng phần mềm và nâng cao sự hài lòng của khách hàng:**  ISO 9000-3 giúp xây dựng hệ thống quản lý chất lượng, đáp ứng yêu cầu khách hàng.  ISO/IEC 12207 cung cấp phương pháp luận trong phát triển và bảo trì phần mềm.  Áp dụng chuẩn giúp giảm lỗi phần mềm, rút ngắn thời gian phát triển, tiết kiệm chi phí và tăng uy tín. |

1. **Phân biệt chuẩn quản lý chất lượng (Quality management standards) và chuẩn tiến trình dự án (SQA project process standards). Nêu ít nhất 2 tên gọi của mỗi loại chuẩn này. (slide 10/ chapter 5)**

|  |
| --- |
|  |
| **2 tên gọi của mỗi loại chuẩn:**  **Tên gọi của mỗi loại chuẩn:**   * **Chuẩn quản lý chất lượng:** ISO 9000-3, CMM (Capability Maturity Model). * **Chuẩn tiến trình dự án:** ISO/IEC 12207, IEEE 1012-1998. |

1. **ISO 9000, ISO 9000-3 là chuẩn gì. (slide 13/ chapter 5)**

|  |
| --- |
| **ISO**: Tổ chức quốc tế xây dựng và ban hành các tiêu chuẩn đảm bảo chất lượng trong nhiều lĩnh vực : sản xuất, kinh doanh, dịch vụ |
| **ISO 9000**  ISO 9000 là một bộ tiêu chuẩn quốc tế về **quản lý chất lượng**, được áp dụng cho mọi loại hình tổ chức bất kể quy mô hay lĩnh vực (sản xuất, dịch vụ, giáo dục), bao gồm cả các trường đại học và cao đẳng. |
| **ISO 9000-3**  **ISO 9000-3:** là một tiêu chuẩn chuyên biệt thuộc bộ ISO 9000, chuyên dùng trong **phát triển và bảo trì phần mềm**, hướng dẫn cách áp dụng các quy định của ISO 9000 vào lĩnh vực phần mềm, bao gồm các hoạt động phát triển, kiểm thử, triển khai và bảo trì. |
| **Cả hai tiêu chuẩn ISO 9001 và ISO 9000-3 được xem xét và cập nhật 5-8 năm / lần** |

1. **CMM là gì. Các mức phát triển của CMM. (slide 17&19 / chapter 5)**

|  |
| --- |
| **CMM là :**  **CMM (Capability Maturity Model)** là một mô hình đánh giá **năng lực trưởng thành** của **quay trình phát triển phần mềm** trong một tổ chức. CMM được phát triển bởi **Software Engineering Institute (SEI)**, được tài trợ bởi **Bộ Quốc phòng Hoa Kỳ (DoD)** vào cuối những năm 1980.  **Mục tiêu** của CMM là giúp các tổ chức:  **Cải thiện quy trình phát triển phần mềm.**  **Nâng cao khả năng dự đoán** về **chất lượng**, **thời gian** và **chi phí** của các dự án phần mềm.  **Tăng cường khả năng quản lý rủi ro** trong các dự án phần mềm. |
| **Các mức phát triển của CMM :**  **Cấp độ 1 - Khởi đầu (Initial):**  - Quy trình "bộc phát", không có kế hoạch rõ ràng.  - Thành công dựa vào nỗ lực cá nhân, không kiểm soát được chất lượng, thời hạn giao nộp, v.v.  **Cấp độ 2 - Lặp lại (Repeatable):**  - Quản lý tiến trình cơ bản để theo dõi chi phí, lịch trình, chức năng.  - Có khả năng lặp lại thành công các dự án tương tự.  - Thành công phụ thuộc vào kỹ năng và kinh nghiệm quản lý cá nhân.  **Cấp độ 3 - Định nghĩa (Defined):**  - Quy trình phát triển phần mềm được định nghĩa rõ ràng và phù hợp với tổ chức.  **Cấp độ 4 - Quản lý (Managed):**  - Thu thập các số liệu chi tiết về quy trình và chất lượng sản phẩm.  - Quy trình và sản phẩm được đo lường và giám sát chặt chẽ.  **Cấp độ 5 - Tối ưu hóa (Optimizing):**  - Cải tiến liên tục dựa trên phản hồi và thử nghiệm công nghệ mới.  - Quy trình phát triển phần mềm được tối ưu hóa không ngừng. |

1. **So sánh ISO và CMM/CMMI. (slide 21-22/chapter 5)**

|  |
| --- |
| **Giống nhau:**  - Đều quan tâm đến **quản lý chất lượng** và **quy trình sản xuất**.  - Đều hướng tới việc nâng cao **hiệu quả hoạt động** của tổ chức.  **Khác nhau:** |

1. **Nêu một số chuẩn về kiểm thử của IEEE. (slide 24-29/ chapter 5)**

|  |
| --- |
| **IEEE 1012**:  - Chuẩn này thiết lập khung chung cho các hoạt động xác minh và thẩm định phần mềm trong suốt vòng đời phát triển phần mềm.  **IEEE 1028**:  - Chuẩn này định nghĩa các thủ tục đánh giá có hệ thống, áp dụng cho các hoạt động đánh giá được thực hiện trong suốt vòng đời phần mềm.  **IEEE Std 829**:  - Chuẩn này cung cấp khuôn mẫu cho việc lập tài liệu kiểm thử phần mềm, bao gồm các tài liệu như kế hoạch kiểm thử, báo cáo kết quả kiểm thử, báo cáo lỗi, v.v.  ----------------------------------------------------------------------------------------------------------------------------------  **IEEE Std 730**:  - Đảm bảo chất lượng phần mềm, nhấn mạnh quy trình kiểm thử.  **IEEE Std 1044**:  - Quy định cách phân loại và báo cáo lỗi trong kiểm thử.  **IEEE Std 1633**:  - Hỗ trợ đánh giá và dự báo độ tin cậy của phần mềm qua kết quả kiểm thử.  **IEEE/EIA Std 12207**:  - Quy định các quy trình kiểm thử trong vòng đời phần mềm như kiểm thử đơn vị, tích hợp, hệ thống, chấp nhận. |

# Chương 6

1. **Các hình thức của kiểm thử tĩnh (*static testing*).**

|  |
| --- |
| Các kỹ thuật kiểm thử tĩnh là những kỹ thuật kiểm tra một thành phần hoặc hệ thống ở cấp độ đặc tả hoặc triển khai mà không cần thực thi phần mềm. **Mục tiêu của kiểm thử tĩnh** là xác định lỗi trong bất kỳ giai đoạn nào của SDLC càng sớm càng tốt, xác minh rằng các thành phần của phần mềm phù hợp với yêu cầu của nó, cung cấp thông tin để theo dõi dự án, cải thiện chất lượng phần mềm và tăng năng suất. Các loại lỗi dễ tìm thấy hơn trong quá trình kiểm thử tĩnh là sai lệch so với tiêu chuẩn, thiếu yêu cầu, lỗi thiết kế, mã không thể bảo trì, đặc tả giao diện không nhất quán,... **Kiểm thử tĩnh** bao gồm các hình thức như đánh giá không chính thức, thảo luận, đánh giá kỹ thuật, kiểm tra, phân tích tĩnh. **Phân tích tĩnh** có thể được chia thành các loại: tiêu chuẩn mã hóa (quy ước mã hóa), số liệu mã và cấu trúc mã. |

1. **Định nghĩa review (IEEE). Mục đích của review. (slide 10 chapter 6)**

|  |
| --- |
| **Định nghĩa:**  Review (theo IEEE) là một quá trình hoặc cuộc họp nhằm xem xét, kiểm tra và đánh giá một tài liệu (như tài liệu thiết kế, mã nguồn, kế hoạch kiểm thử, v.v.) bởi một hoặc nhiều người có kinh nghiệm trong lĩnh vực liên quan. |
| **Mục đích:**  Mục đích của việc review là để phát hiện ra các khuyết điểm, lỗi thông tin, và các vấn đề trong tài liệu, từ đó cải thiện chất lượng và độ chính xác của tài liệu. |
| **Mục tiêu của việc review**:   * **Tìm ra khuyết điểm**: Phát hiện các lỗi, thiếu sót trong tài liệu, giúp cải tiến và đảm bảo sản phẩm đạt chất lượng cao. * **Thông tin**: Đảm bảo thông tin trong tài liệu chính xác và đầy đủ, giúp người sử dụng hiểu rõ các yêu cầu và hướng dẫn. * **Giao tiếp**: Tạo cơ hội trao đổi ý kiến giữa các thành viên trong nhóm, cải thiện sự hiểu biết chung về dự án. * **Giáo dục**: Cung cấp cơ hội học hỏi cho các thành viên trong nhóm về các quy trình, tiêu chuẩn và kỹ thuật tốt nhất.   Việc review giúp nâng cao chất lượng sản phẩm và quá trình phát triển phần mềm, đồng thời tạo điều kiện cho sự hợp tác và cải thiện liên tục trong dự án. |

1. **Tiến trình review gồm những bước nào. Trong từng bước, ai là người chịu trách nhiệm thực hiện. (slide 16 chapter 6)**

|  |
| --- |
| **Tiến trình review gồm những bước:**  **1. Planning – Lập kế hoạch kiểm thử:** Đây là giai đoạn đầu tiên trong quy trình kiểm thử phần mềm, nơi các mục tiêu, phương pháp, và tài nguyên cần thiết cho quá trình kiểm thử được xác định.  **2. Kick-off – Khởi động:** Đây là buổi họp hoặc sự kiện mở đầu cho dự án kiểm thử phần mềm, nơi các bên liên quan (như nhóm phát triển, nhóm kiểm thử) sẽ gặp gỡ để thảo luận về các mục tiêu, kế hoạch và các yêu cầu liên quan đến quá trình kiểm thử.  **3. Preparation – Chuẩn bị:** bao gồm việc chuẩn bị các tài liệu cần thiết, thiết lập môi trường kiểm thử, tạo ra các kế hoạch kiểm thử chi tiết và chuẩn bị dữ liệu kiểm thử để đảm bảo rằng quá trình kiểm thử diễn ra suôn sẻ và hiệu quả.  **4. Review meeting – Cuộc họp đánh giá/ Gặp gỡ để đánh giá:** Đây là cuộc họp mà các nhóm liên quan (như nhóm kiểm thử, nhóm phát triển, hoặc các bên liên quan khác) gặp gỡ để xem xét và đánh giá các kết quả kiểm thử, tài liệu kiểm thử, hoặc kế hoạch kiểm thử. Mục tiêu của cuộc họp là đảm bảo rằng mọi thứ đã được thực hiện đúng cách và đáp ứng yêu cầu chất lượng.  **5. Rework – Chỉnh sửa lại:** nếu các lỗi hoặc vấn đề được phát hiện trong quá trình kiểm thử sẽ được đội ngũ phát triển sửa chữa hoặc điều chỉnh. Sau khi các thay đổi được thực hiện, phần mềm sẽ được kiểm tra lại để đảm bảo rằng các vấn đề đã được khắc phục và không gây ra lỗi mới.  **6. Following-up – theo dõi:** theo dõi tiến độ và kết quả của các hoạt động kiểm thử, đảm bảo rằng các vấn đề đã được giải quyết, các lỗi đã được sửa chữa và kiểm tra lại, và mọi hành động cần thiết để đạt được chất lượng phần mềm đều được thực hiện đúng hạn. |
| **Người chịu trách nhiệm thực hiện từng bước là:**  1. Planning – Lập kế hoạch kiểm thử:  2. Kick-off – Khởi động:  3. Preparation – Chuẩn bị:  4. Review meeting – Cuộc họp đánh giá/ Gặp gỡ để đánh giá:  5. Rework – Chỉnh sửa lại:  6. Following-up – theo dõi: |

1. **Chuẩn mã nguồn (*coding standard*) là gì. Có vai trò như thế nào.**

|  |
| --- |
| **Chuẩn mã nguồn (coding standard)** là tập hợp các quy tắc, hướng dẫn và thực tiễn được sử dụng để tạo ra mã nguồn nhất quán và dễ đọc. Chuẩn mã nguồn đề xuất phong cách lập trình, cách thức và phương pháp cho mỗi khía cạnh của một chương trình. **Các quy ước phổ biến có thể bao gồm các lĩnh vực sau**:   * Tổ chức tệp * Quy ước đặt tên * Thụt đầu dòng, khoảng trắng * Chú thích, khai báo, câu lệnh * Thực tiễn lập trình, nguyên tắc, quy tắc chung   **Vai trò của chuẩn mã nguồn**:   * **Nâng cao khả năng bảo trì phần mềm**: 80% chi phí vòng đời phần mềm là cho việc bảo trì, và những người bảo trì phần mềm có thể thay đổi. Việc tuân thủ nghiêm ngặt các quy ước mã hóa giúp cải thiện khả năng đọc hiểu của phần mềm, cho phép các kỹ sư hiểu mã mới một cách nhanh chóng và kỹ lưỡng hơn. * **Phát hiện lỗi sớm**: Việc sử dụng công cụ phân tích tĩnh để kiểm tra mã nguồn dựa trên các quy ước mã hóa giúp phát hiện lỗi sớm trong quá trình phát triển. * **Nâng cao hiệu quả quy trình review**: Việc sử dụng công cụ giúp tự động hóa việc kiểm tra các quy ước mã hóa, giảm thiểu thời gian và công sức cho việc review thủ công, cho phép người review tập trung vào việc tìm kiếm các lỗi khác.   **Tóm lại, chuẩn mã nguồn đóng vai trò quan trọng trong việc đảm bảo chất lượng, khả năng bảo trì và hiệu quả phát triển phần mềm.** |

1. **Phân tích tĩnh gồm những hoạt động nào.**

|  |
| --- |
| Phân tích tĩnh bao gồm các hoạt động chính sau:   * **Phân tích luồng điều khiển (Control flow analysis):**   + Phân tích luồng điều khiển được sử dụng để khám phá luồng điều khiển phân cấp trong một chương trình.   + **Mục tiêu:** Xác định các lỗi như vòng lặp vô hạn, nhiều điểm nhập vào vòng lặp, mã không thể truy cập (mã chết),...   + **Kỹ thuật:** Sử dụng biểu đồ luồng điều khiển (CFG) để biểu diễn luồng điều khiển của chương trình và tính toán các số liệu mã như số lượng mức lồng nhau hoặc độ phức tạp cyclomatic. * **Phân tích luồng dữ liệu (Data flow analysis):**   + Tập trung vào sự xuất hiện của các biến, theo dõi đường dẫn từ định nghĩa (gán giá trị cho biến) đến các lần sử dụng (đọc giá trị của biến).   + **Mục tiêu:** Phát hiện các lỗi liên quan đến việc sử dụng biến như biến không bao giờ được sử dụng, tham chiếu đến biến chưa được khởi tạo, gán giá trị không chính xác hoặc không hợp lệ cho biến,... * **Phân tích cấu trúc dữ liệu (Data structure analysis):**   + Phân tích cấu trúc dữ liệu logic của chương trình.   + **Mục tiêu:** Phân tích cách tổ chức dữ liệu, các phép biến đổi trên cấu trúc dữ liệu logic, ví dụ như danh sách, hàng đợi, ngăn xếp,...   + **Lợi ích:** Cung cấp thông tin về độ khó trong việc viết chương trình xử lý dữ liệu và thiết kế các trường hợp kiểm thử.   Ngoài ra, phân tích tĩnh cũng bao gồm việc sử dụng các công cụ để kiểm tra mã nguồn dựa trên các **chuẩn mã nguồn (coding standard)**, như đã thảo luận trong đoạn hội thoại trước. Việc này giúp phát hiện sớm các lỗi vi phạm quy ước mã hóa, nâng cao khả năng đọc hiểu và bảo trì mã nguồn. |

1. **Cho biết một vài độ đo mã nguồn.**

|  |
| --- |
| Một số **độ đo mã nguồn (code metrics)** thường được sử dụng trong phân tích tĩnh bao gồm:   * **Tần suất chú thích (Comment frequency):** Đo lường tỷ lệ dòng chú thích so với tổng số dòng mã nguồn. Tần suất chú thích cao cho thấy mã nguồn được chú thích đầy đủ, dễ hiểu và bảo trì hơn. * **Độ sâu lồng nhau (Depth of nesting):** Đo lường số lượng khối mã lồng nhau (ví dụ: các vòng lặp, câu lệnh điều kiện) trong một hàm hoặc phương thức. Độ sâu lồng nhau cao có thể làm giảm khả năng đọc hiểu và bảo trì của mã nguồn. * **Số dòng mã (Number of lines of code):** Đo lường số lượng dòng mã trong một chương trình hoặc module. Số dòng mã cao có thể là dấu hiệu của sự phức tạp và khó bảo trì. * **Độ phức tạp Cyclomatic (Cyclomatic complexity):** Đo lường số lượng đường dẫn độc lập tuyến tính trong một hàm hoặc phương thức. Độ phức tạp Cyclomatic cao cho thấy mã nguồn có nhiều nhánh điều khiển, khó kiểm thử và dễ bị lỗi hơn.   Các **độ đo mã nguồn** này cung cấp thông tin hữu ích về chất lượng và độ phức tạp của mã nguồn. Bằng cách theo dõi và phân tích các độ đo này, các nhà phát triển có thể cải thiện khả năng đọc hiểu, bảo trì và độ tin cậy của phần mềm. |

1. **Đưa ra ít nhất 5 lỗi thường được phát hiện khi phân tích mã nguồn (trừ lỗi syntax).**

|  |
| --- |
| Ngoài các lỗi cú pháp (syntax), phân tích mã nguồn tĩnh có thể phát hiện ra nhiều loại lỗi khác, bao gồm:   * **Vi phạm chuẩn mã nguồn (Coding standard violations):** Các công cụ phân tích tĩnh có thể tự động kiểm tra mã nguồn dựa trên các quy ước mã hóa được định nghĩa trước, giúp phát hiện các lỗi như sai quy ước đặt tên, thụt lồng không đúng, thiếu chú thích,...   + Ví dụ, công cụ có thể phát hiện lỗi thụt lồng không đúng quy định trong đoạn mã if (a == b && c == d) và đề xuất sửa thành if ((a == b) && (c == d)) để tuân thủ quy tắc sử dụng dấu ngoặc đơn trong biểu thức. * **Lỗi luồng điều khiển (Control flow errors):**   + **Vòng lặp vô hạn:** Xảy ra khi điều kiện thoát vòng lặp không bao giờ được đáp ứng, dẫn đến chương trình bị treo.   + **Mã chết (Dead code):** Đoạn mã không thể được thực thi do luồng điều khiển của chương trình. Ví dụ: đoạn mã unreachable(dead) code trong ví dụ phân tích luồng điều khiển.   + **Nhiều điểm nhập vào vòng lặp:** Có thể dẫn đến hành vi không mong muốn của chương trình. * **Lỗi luồng dữ liệu (Data flow errors):**   + **Sử dụng biến chưa được khởi tạo:** Dẫn đến giá trị không xác định và có thể gây ra lỗi trong chương trình. Ví dụ: biến y được sử dụng trước khi được định nghĩa trong ví dụ phân tích luồng dữ liệu.   + **Gán giá trị không hợp lệ cho biến:** Có thể dẫn đến lỗi logic hoặc lỗi runtime.   + **Biến được định nghĩa lại mà không được sử dụng:** Là dấu hiệu của mã nguồn dư thừa hoặc lỗi logic. Ví dụ: biến n trong ví dụ phân tích luồng dữ liệu. * **Rò rỉ bộ nhớ (Memory leaks):** Xảy ra khi chương trình cấp phát bộ nhớ nhưng không giải phóng sau khi sử dụng xong, dẫn đến cạn kiệt tài nguyên hệ thống. Vấn đề này thường gặp trong các ngôn ngữ như C, C++. Ví dụ: quên gọi free(A) sau khi cấp phát bộ nhớ cho biến con trỏ A. * **Lỗi tiềm ẩn (Potential bugs):**   + **Lỗi Null Pointer:** Xảy ra khi chương trình cố gắng truy cập vào một đối tượng null, ví dụ: pPointer->member = 1; mà không kiểm tra null trước.   + **SQL Injection:** Lỗ hổng bảo mật cho phép kẻ tấn công chèn mã SQL độc hại vào câu truy vấn. Ví dụ: đoạn mã string cmdtext = string.Format("SELECT \* FROM [Users] WHERE [Account]='{0}' AND [Password]='{1}' “, userName, password); trong ví dụ có nguy cơ bị tấn công SQL Injection.   + **Vấn đề về hiệu năng:** Các đoạn mã không hiệu quả có thể làm giảm hiệu suất của chương trình. Ví dụ: việc tạo đối tượng trong vòng lặp có thể ảnh hưởng đến hiệu suất ứng dụng.   Đây chỉ là một số ví dụ về lỗi thường được phát hiện khi phân tích mã nguồn. Việc sử dụng công cụ phân tích tĩnh và review mã nguồn thường xuyên là các biện pháp hiệu quả để phát hiện và sửa lỗi sớm trong quá trình phát triển phần mềm, giúp nâng cao chất lượng và độ tin cậy của sản phẩm. |

# Chương 7 (Có bài tập)

1. Các kỹ thuật blackbox và cách áp dụng: **equivalence partitioning, boundary value analysis, decision tables, state transition testing**. Thuận lợi và khó khăn của *black-box*.

|  |
| --- |
| Dưới đây là những thuận lợi và khó khăn của kỹ thuật kiểm thử hộp đen (black-box techniques):  **Ưu điểm:**   * Kỹ thuật black-box testing **hiệu quả hơn trên các đơn vị mã lớn** so với kỹ thuật kiểm thử hộp trắng (glass box testing). * Người kiểm thử **không cần kiến thức về triển khai**, bao gồm các ngôn ngữ lập trình cụ thể. * Người kiểm thử và lập trình viên **độc lập** với nhau. * Các thử nghiệm được thực hiện **từ quan điểm của người dùng**. * Sẽ giúp **phát hiện bất kỳ sự mơ hồ hoặc không nhất quán nào trong các thông số kỹ thuật**. * Các trường hợp thử nghiệm (test case) có thể được **thiết kế ngay sau khi các thông số kỹ thuật được hoàn thành**.   **Nhược điểm:**   * Chỉ một **số lượng nhỏ các đầu vào có thể có** thực sự có thể được kiểm tra. * Có thể **bỏ sót nhiều đường dẫn chương trình** chưa được kiểm tra. * **Nếu không có các thông số kỹ thuật rõ ràng và súc tích, các trường hợp thử nghiệm rất khó thiết kế**. * **Có thể có sự lặp lại không cần thiết** các đầu vào thử nghiệm nếu người kiểm thử không được thông báo về các trường hợp thử nghiệm mà lập trình viên đã thử. |

1. **Các kỹ thuật whitebox và cách áp dụng: statement coverage, decision coverage. Thuận lợi và khó khăn của *white-box*.**

|  |
| --- |
| Dưới đây là những thuận lợi và khó khăn của kỹ thuật kiểm thử hộp trắng (white-box techniques):  Ưu điểm:   * Cho phép kiểm tra trực tiếp các đường dẫn xử lý và thuật toán. * Cung cấp theo dõi phạm vi dòng (line coverage) để đưa ra danh sách các dòng mã chưa được thực thi. * Có khả năng kiểm tra chất lượng của công việc viết mã.   Nhược điểm:   * Cần nguồn lực lớn, cao hơn nhiều so với kỹ thuật kiểm thử hộp đen. * Không thể kiểm tra hiệu năng của phần mềm về mặt khả năng sẵn sàng (availability), độ tin cậy (reliability), khả năng chịu tải (stress), v.v. * Người kiểm thử phải có đủ kỹ năng lập trình để hiểu mã và luồng điều khiển của nó. * Kiểm tra luồng điều khiển có thể rất tốn thời gian do tất cả các mô-đun và đường dẫn cơ sở (basis paths) tạo nên một hệ thống. |

# Chương 8

1. **Quản lý kiểm thử gồm có những hoạt động chính nào.**

|  |
| --- |
| Quản lý kiểm thử bao gồm các hoạt động chính sau:   * Lập kế hoạch kiểm thử: Xác định mục tiêu kiểm thử, chính sách kiểm thử của tổ chức, chiến lược kiểm thử và kế hoạch kiểm thử. Kế hoạch kiểm thử là kế hoạch dự án cho công việc kiểm thử sẽ được thực hiện. Kế hoạch kiểm thử không phải là đặc tả thiết kế kiểm thử, tập hợp các trường hợp kiểm thử hoặc một tập hợp các quy trình kiểm thử. Việc viết kế hoạch kiểm thử giúp định hướng suy nghĩ, đóng vai trò là phương tiện để giao tiếp với các thành viên khác trong nhóm dự án, người kiểm thử, đồng nghiệp, người quản lý và các bên liên quan khác, đồng thời giúp quản lý thay đổi. * Ước tính kiểm thử: Ước tính khối lượng công việc kiểm thử cần thực hiện và thương lượng với ban quản lý. Có thể sử dụng phương pháp dựa trên số liệu hoặc phương pháp dựa trên chuyên gia để ước tính. Phương pháp dựa trên số liệu dựa trên dữ liệu được thu thập từ các dự án trước đó hoặc tương tự, trong khi phương pháp dựa trên chuyên gia liên quan đến việc tham khảo ý kiến ​​của những người sẽ thực hiện công việc và những người khác có chuyên môn về các nhiệm vụ cần thực hiện. * Giám sát và kiểm soát tiến độ kiểm thử: Theo dõi, đo lường, kiểm soát và báo cáo tiến độ kiểm thử, trạng thái chất lượng sản phẩm và kết quả kiểm thử. Giám sát tiến độ kiểm thử cung cấp phản hồi cho nhóm kiểm thử và người quản lý kiểm thử, cung cấp khả năng hiển thị về kết quả kiểm thử, đo lường trạng thái của quá trình kiểm thử so với tiêu chí thoát, đồng thời thu thập dữ liệu để sử dụng trong việc ước tính các nỗ lực kiểm thử trong tương lai. * Quản lý rủi ro: Xác định, đánh giá và giảm thiểu rủi ro liên quan đến kiểm thử. Rủi ro có thể là rủi ro sản phẩm, liên quan đến những gì được tạo ra bởi công việc, hoặc rủi ro dự án, liên quan đến cách thức công việc được thực hiện. * Quản lý sự cố: Theo dõi, báo cáo và giải quyết các sự cố xảy ra trong quá trình kiểm thử. Sự cố là bất kỳ sự kiện nào xảy ra trong quá trình kiểm thử mà yêu cầu điều tra hoặc sửa chữa sau đó. Quản lý sự cố cung cấp cho các lập trình viên, người quản lý và những người khác thông tin chi tiết về hành vi được quan sát và lỗi, cung cấp cho các trưởng nhóm kiểm thử phương tiện theo dõi chất lượng của hệ thống đang được kiểm thử và tiến độ kiểm thử, đồng thời cung cấp ý tưởng để phát triển và cải tiến quy trình kiểm thử. |

1. **Nhiệm vụ của test leader, của tester.**

|  |
| --- |
| 1. Nhiệm vụ của Test Leader:  * **Xây dựng mục tiêu kiểm thử, chính sách kiểm thử của tổ chức, chiến lược kiểm thử và kế hoạch kiểm thử.** * **Ước lượng khối lượng công việc kiểm thử cần thực hiện và đàm phán với ban quản lý.** * **Nhận biết khi nào cần tự động hóa kiểm thử.** * **Dẫn dắt, hướng dẫn và giám sát việc phân tích, thiết kế, triển khai và thực thi các trường hợp kiểm thử, quy trình kiểm thử và bộ kiểm thử.** * **Đảm bảo quản lý cấu hình đầy đủ của phần mềm kiểm thử.** * **Đảm bảo môi trường kiểm thử được thiết lập trước khi thực thi kiểm thử và được quản lý trong quá trình thực thi kiểm thử.** * **Lập lịch trình kiểm thử để thực thi, sau đó theo dõi, đo lường, kiểm soát và báo cáo về tiến độ kiểm thử, trạng thái chất lượng sản phẩm và kết quả kiểm thử.** * **Viết báo cáo tóm tắt kiểm thử.**   Ngoài ra, Test Leader có thể đảm nhận vai trò quản lý dự án, quản lý phát triển hoặc quản lý đảm bảo chất lượng. Khả năng lập kế hoạch, giám sát và kiểm soát công việc kiểm thử là rất quan trọng đối với vị trí này. 2. Nhiệm vụ của Tester:  * **Xem xét và đóng góp vào kế hoạch kiểm thử.** * **Phân tích, xem xét và đánh giá các yêu cầu và thông số kỹ thuật thiết kế.** * **Tham gia hoặc thậm chí là người chính xác định các điều kiện kiểm thử và tạo các trường hợp kiểm thử, thông số kỹ thuật quy trình kiểm thử và dữ liệu kiểm thử, đồng thời có thể tự động hóa hoặc giúp tự động hóa các bài kiểm thử.** * **Thiết lập môi trường kiểm thử hoặc hỗ trợ nhân viên quản trị hệ thống và quản lý mạng.** * **Thực hiện kiểm thử trên tất cả các cấp độ kiểm thử, đánh giá kết quả so với kết quả mong đợi.** * **Giám sát quá trình kiểm thử và môi trường kiểm thử, đồng thời thường xuyên thu thập các chỉ số hiệu suất.** * **Xem xét công việc của nhau.**   Để thực hiện tốt các nhiệm vụ trên, Tester cần có các **kỹ năng chuyên môn và kỹ năng xã hội cơ bản** trong ba lĩnh vực chính: **lĩnh vực ứng dụng hoặc kinh doanh, công nghệ và kiểm thử.** Sự chuyên môn hóa kỹ năng và phân chia vai trò là rất quan trọng trong kiểm thử phần mềm. |

1. **Cần gì để viết một test plan. Cấu trúc của một test plan theo IEEE 829 Standard.**

|  |
| --- |
| Để viết một test plan, bạn cần hiểu rõ mục đích của test plan, nắm bắt được thông tin về dự án và sản phẩm cần kiểm thử, và có kỹ năng phân tích, thiết kế, và quản lý kiểm thử.  **Mục đích của test plan là:**   * Định hướng suy nghĩ cho quá trình kiểm thử. * Giao tiếp với các thành viên trong nhóm dự án, tester, đồng nghiệp, quản lý, và các bên liên quan khác. * Quản lý thay đổi trong quá trình kiểm thử.   **Để viết được test plan, bạn cần thu thập thông tin sau:**   * Yêu cầu kinh doanh (Business Requirements) * Đặc tả chức năng (Functional Specification) * User Stories * Tài liệu kế hoạch dự án (Project Plan Document) * Tài liệu giải pháp (Solution Document)   **Cấu trúc của một test plan theo chuẩn IEEE 829 Standard bao gồm các phần sau:**   1. **Test Plan Identifier:** Định danh của kế hoạch kiểm thử. 2. **Introduction:** Giới thiệu tổng quan về kế hoạch kiểm thử, bao gồm mục đích, phạm vi, và đối tượng của kế hoạch kiểm thử. 3. **Test Items:** Danh sách các hạng mục cần kiểm thử, bao gồm phần mềm, phần cứng, hoặc tài liệu. 4. **Features to be tested:** Danh sách các tính năng cần kiểm thử. 5. **Features not to be tested:** Danh sách các tính năng không cần kiểm thử, kèm theo lý do. 6. **Approach:** Phương pháp tiếp cận kiểm thử, bao gồm các kỹ thuật kiểm thử, mức độ kiểm thử, và môi trường kiểm thử. 7. **Item pass/fail criteria:** Tiêu chí đánh giá kết quả kiểm thử, xác định khi nào một hạng mục được coi là đạt hoặc không đạt. 8. **Suspension criteria and resumption requirements:** Tiêu chí tạm dừng kiểm thử và yêu cầu để tiếp tục kiểm thử. 9. **Test deliverables:** Danh sách các sản phẩm đầu ra của quá trình kiểm thử, bao gồm báo cáo kiểm thử, lỗi, và dữ liệu kiểm thử. 10. **Testing tasks:** Danh sách các nhiệm vụ kiểm thử cần thực hiện, bao gồm thiết kế test case, thực thi test case, và báo cáo kết quả kiểm thử. 11. **Environmental needs:** Yêu cầu về môi trường kiểm thử, bao gồm phần cứng, phần mềm, và mạng. 12. **Responsibilities:** Trách nhiệm của các thành viên trong nhóm kiểm thử. 13. **Staffing and training needs:** Nhu cầu về nhân sự và đào tạo cho quá trình kiểm thử. 14. **Schedule:** Lịch trình thực hiện kiểm thử. 15. **Risks and contingencies:** Rủi ro tiềm ẩn và phương án dự phòng. 16. **Approvals:** Danh sách các bên liên quan cần phê duyệt kế hoạch kiểm thử.   **Tóm lại, để viết một test plan hiệu quả, bạn cần hiểu rõ mục đích, nắm bắt thông tin cần thiết, và tuân thủ cấu trúc chuẩn IEEE 829.** |

1. **Sự cố là gì. Trong testing, có những sự cố gì. Các thành phần của một báo cáo sự cố theo IEEE 829 Standard. (slide 44 & 46/ chapter 8)**

|  |
| --- |
| **Sự cố là gì.**  **Sự cố** là bất kỳ sự kiện nào trong quá trình kiểm tra mà kết quả thực tế không khớp với kết quả dự kiến. |
| **Trong testing, có những sự cố gì:**  Lỗi trong môi trường kiểm thử, lỗi bộ dữ liệu thử nghiệm, kết quả dự kiến sai, hoặc lỗi do tester. Sự cố có thể xảy ra ở cả tài liệu và mã nguồn. |
| **Các thành phần của một báo cáo sự cố theo IEEE 829 Standard.**  **- Định danh**: Mã số duy nhất để nhận diện sự cố.  **- Tóm tắt**: Tóm tắt ngắn gọn về sự cố, cung cấp đủ thông tin để người khác hiểu sự cố đã được phát hiện như thế nào, bao gồm các thông tin liên quan như:  + Sự cố trong trường hợp kiểm thử.  + Quy trình kiểm thử đã thực hiện.  + Nhật ký kiểm thử.  **- Mô tả chi tiết sự cố**:  **+Dữ liệu nhập**: Các dữ liệu đầu vào trong quá trình kiểm thử.  **+ Kết quả mong đợi**: Những gì dự kiến sẽ xảy ra.  **+ Kết quả thực tế**: Những gì thực tế đã xảy ra trong thử nghiệm.  **+ Bất thường**: Các sự khác biệt hoặc lỗi giữa kết quả mong đợi và kết quả thực tế.  **+Thông tin phân loại**:  ++ Ngày và giờ phát hiện sự cố.  ++ Giai đoạn của dự án mà sự cố được tìm thấy.  **+ Bước thực hiện**: Các bước cụ thể trong quy trình kiểm thử dẫn đến sự cố (quan trọng đối với các thử nghiệm dài và phức tạp).  **+ Môi trường thử nghiệm**: Các điều kiện hệ thống hoặc môi trường sử dụng trong thử nghiệm.  **+ Số lần thử lặp lại**: Bao nhiêu lần thử lại để tái tạo sự cố.  **+ Chú thích của tester và người quan sát**: Nhận xét từ người thử nghiệm và người quan sát có kiến thức về sự cố.  **- Ảnh hưởng**: Mô tả thiệt hại thực tế hoặc tiềm ẩn của sự cố, có thể bao gồm:  + Đánh giá mức độ nghiêm trọng (Severity) của sự cố.  + Độ ưu tiên sửa chữa (Priority) của sự cố.  + Các thông tin khác liên quan đến ảnh hưởng của sự cố đối với hệ thống. |

1. **Theo bạn, cần các kỹ năng và kiến thức gì khi muốn trở thành tester.**

|  |
| --- |
| Để trở thành một tester, bạn cần những kỹ năng và kiến thức sau:   * **Kiến thức chuyên môn về ứng dụng hoặc lĩnh vực kinh doanh:** Hiểu biết về lĩnh vực mà phần mềm được sử dụng sẽ giúp tester thiết kế các trường hợp kiểm thử phù hợp và đánh giá kết quả một cách hiệu quả. * **Kiến thức về công nghệ:** Tester cần có kiến thức về các công nghệ liên quan đến phần mềm, bao gồm hệ điều hành, cơ sở dữ liệu, mạng, v.v. Kiến thức này giúp tester thiết lập môi trường kiểm thử và thực hiện các kiểm thử một cách hiệu quả. * **Kiến thức về kiểm thử:** Bao gồm các kỹ thuật kiểm thử, các loại kiểm thử, quy trình kiểm thử, cách viết test case, cách báo cáo lỗi, v.v. Kiến thức này là nền tảng cho việc thực hiện công việc kiểm thử. * **Kỹ năng chuyên môn về kiểm thử:** Kỹ năng này đến từ kinh nghiệm và sự rèn luyện. Kiểm thử nghiệp dư thường không đủ để đảm bảo chất lượng phần mềm, vì vậy cần có sự tham gia của các tester chuyên nghiệp. * **Kỹ năng mềm:** Bao gồm kỹ năng giao tiếp, làm việc nhóm, giải quyết vấn đề, quản lý thời gian, v.v. Tester cần giao tiếp hiệu quả với các thành viên trong nhóm phát triển, báo cáo lỗi một cách rõ ràng, và làm việc nhóm để đạt được mục tiêu chung.   Ngoài ra, tùy vào vai trò và chuyên môn, tester có thể cần các kỹ năng và kiến thức chuyên sâu hơn như:   * **Kiến thức về tự động hóa kiểm thử:** Tester có thể cần phải biết cách sử dụng các công cụ tự động hóa để thực hiện các kiểm thử một cách hiệu quả hơn. * **Kiến thức về quản lý kiểm thử:** Test leader cần có kỹ năng lập kế hoạch, giám sát và kiểm soát công việc kiểm thử, cũng như quản lý rủi ro và sự cố.   Tóm lại, để trở thành một tester giỏi, bạn cần kết hợp kiến thức chuyên môn, kỹ năng kiểm thử và kỹ năng mềm. |

Bài thi gồm 2 phần: tự luận và trắc nghiệm (dạng câu hỏi trả lời ngắn hoặc lựa chọn).

Luyện tập phần trắc nghiệm trong giáo trình kiểm thử “Foundations of Software Testing”.

Ngày thi: theo lịch trên web.

**KHÔNG ĐƯỢC SỬ DỤNG TÀI LIỆU**