**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

Bài tập cá nhân nhóm 11

Sinh viên thực hiện:

Nguyễn Văn Hiền



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

Tuần 2

**Part 1: Câu hỏi trắc nghiệm**

1. Workflow nào trong tiến trình phát triển phần mềm chịu trách nhiệm thu thập yêu cầu từ khách hàng?

* Đáp án đúng: B. Workflow lấy yêu cầu
* Giải thích: Workflow lấy yêu cầu là bước đầu tiên trong quy trình phát triển phần mềm, giúp nhóm phát triển hiểu rõ yêu cầu từ khách hàng.
* Ví dụ: Trong quá trình phát triển ứng dụng ngân hàng số, đội phát triển gặp gỡ khách hàng để thu thập yêu cầu như tích hợp thanh toán QR, xác thực bằng OTP, v.v.

2. Pha nào trong tiến trình thống nhất (Unified Process) tập trung vào việc phân tích rủi ro và xây dựng kiến trúc ban đầu?

* Đáp án đúng: B. Pha làm rõ
* Giải thích: Pha làm rõ (Elaboration Phase) giúp xác định kiến trúc hệ thống và phân tích rủi ro trước khi phát triển phần mềm.
* Ví dụ: Khi thiết kế hệ thống đặt vé máy bay trực tuyến, đội phát triển xác định trước các rủi ro như xử lý hàng triệu giao dịch đồng thời, bảo mật dữ liệu khách hàng.

3. Mô hình CMM mức nào yêu cầu quy trình phát triển phần mềm phải được quản lý định lượng?

* Đáp án đúng: C. Mức 4
* Giải thích: Ở mức 4 (Managed Level) của mô hình CMM, mọi quy trình phát triển phần mềm đều phải được đo lường và kiểm soát bằng các chỉ số định lượng.
* Ví dụ: Trong một công ty phần mềm lớn như Google, các chỉ số về tốc độ xử lý dữ liệu, tỷ lệ lỗi phần mềm, thời gian phản hồi đều được đo lường để đảm bảo hiệu suất tối ưu.

4. Các pha trong tiến trình thống nhất gồm:

* Đáp án đúng: B. Khởi đầu, làm rõ, xây dựng, chuyển giao
* Giải thích: Tiến trình thống nhất (Unified Process) chia thành 4 pha chính:
  1. Khởi đầu (Inception) – Xác định mục tiêu dự án.
  2. Làm rõ (Elaboration) – Phân tích rủi ro, xây dựng kiến trúc.
  3. Xây dựng (Construction) – Phát triển phần mềm.
  4. Chuyển giao (Transition) – Bàn giao sản phẩm cho khách hàng.
* Ví dụ: Khi phát triển ứng dụng thương mại điện tử, đội phát triển bắt đầu bằng xác định tính năng (khởi đầu), sau đó làm rõ kiến trúc hệ thống, lập trình, kiểm thử (xây dựng), và cuối cùng triển khai cho người dùng (chuyển giao).

5. Trong tiến trình thống nhất, workflow nào thực hiện sau cùng?

* Đáp án đúng: D. Workflow kiểm thử
* Giải thích: Workflow kiểm thử (Testing Workflow) diễn ra cuối cùng để đảm bảo phần mềm hoạt động đúng và không có lỗi.
* Ví dụ: Trước khi phát hành bản cập nhật iOS mới, Apple phải kiểm thử trên nhiều thiết bị khác nhau, đảm bảo không có lỗi nghiêm trọng xảy ra.

6. Mô hình CMM mức 1 có đặc điểm gì?

* Đáp án đúng: C. Quy trình không ổn định, phụ thuộc vào cá nhân
* Giải thích: Ở mức 1 (Initial Level) của CMM, quy trình phát triển phần mềm chưa được chuẩn hóa, phụ thuộc nhiều vào từng cá nhân và thiếu tính ổn định.
* Ví dụ: Một công ty khởi nghiệp phát triển phần mềm không có quy trình rõ ràng, mỗi lập trình viên làm việc theo cách riêng, dẫn đến chất lượng sản phẩm không đồng đều.

7. Tiến trình thống nhất là một ví dụ của mô hình nào?

* Đáp án đúng: B. Mô hình lặp và tăng trưởng
* Giải thích: Tiến trình thống nhất phát triển phần mềm theo từng vòng lặp, cải tiến qua mỗi lần lặp, giúp phần mềm phát triển linh hoạt hơn.
* Ví dụ: Microsoft phát triển Windows 10 bằng cách liên tục cập nhật và cải thiện thay vì làm một phiên bản hoàn toàn mới.

8. Trong mô hình CMM mức 5, quy trình phát triển phần mềm có đặc điểm gì?

* Đáp án đúng: A. Quy trình được cải tiến liên tục
* Giải thích: Ở mức 5 (Optimized Level), quy trình phát triển phần mềm liên tục được cải tiến để tối ưu hiệu suất và chất lượng.
* Ví dụ: Amazon sử dụng dữ liệu từ hàng triệu giao dịch mỗi ngày để tối ưu hệ thống, giảm thời gian tải trang và tăng hiệu suất xử lý đơn hàng.

9. Workflow thiết kế bao gồm việc thực hiện hoạt động nào?

* Đáp án đúng: C. Thiết kế kiến trúc và chi tiết hệ thống
* Giải thích: Workflow thiết kế tập trung vào việc xây dựng kiến trúc phần mềm, tạo sơ đồ hệ thống và thiết kế chi tiết các thành phần.
* Ví dụ: Khi phát triển ứng dụng ngân hàng, giai đoạn thiết kế sẽ bao gồm xác định kiến trúc backend, bảo mật dữ liệu giao dịch, thiết kế API.

10. CMM viết tắt của cụm từ nào?

* Đáp án đúng: B. Capability Maturity Model
* Giải thích: CMM (Capability Maturity Model) là một mô hình đánh giá mức độ trưởng thành của quy trình phát triển phần mềm.
* Ví dụ: Một công ty phần mềm muốn đạt CMM mức 3 phải chuẩn hóa quy trình, lập tài liệu rõ ràng và đảm bảo chất lượng sản phẩm.

**Part 2: Câu hỏi trả lời ngắn**

Câu hỏi 1: Pha khởi đầu trong tiến trình thống nhất là gì?

* Pha khởi đầu (*Inception Phase*) tập trung vào việc xác định phạm vi dự án, các yêu cầu cơ bản và tính khả thi.
* Ví dụ: Khi một công ty phần mềm bắt đầu phát triển một ứng dụng thương mại điện tử, họ sẽ nghiên cứu thị trường, xác định nhóm khách hàng mục tiêu và thu thập yêu cầu từ các bên liên quan.

Câu hỏi 2: Mục tiêu của workflow lấy yêu cầu là gì?

* Trả lời: Thu thập, phân tích và làm rõ yêu cầu từ khách hàng, đảm bảo hệ thống phần mềm đáp ứng đúng nhu cầu.
* Ví dụ: Một công ty phát triển phần mềm đặt lịch hẹn bác sĩ sẽ tiến hành phỏng vấn bệnh viện và bệnh nhân để hiểu rõ các yêu cầu về đặt lịch và quản lý hồ sơ y tế.

Câu hỏi 3: Tiến trình thống nhất gồm bao nhiêu pha chính?

* 4 pha: Khởi đầu (Inception), Làm rõ (Elaboration), Xây dựng (Construction), Chuyển giao (Transition).
* Ví dụ: Khi phát triển một ứng dụng ngân hàng, nhóm phát triển sẽ trải qua tất cả 4 pha này để đảm bảo sản phẩm cuối cùng hoàn thiện và có thể triển khai.

Câu hỏi 4: Sự khác nhau giữa CMM mức 2 và mức 3 là gì?

* Trả lời:
  + Mức 2: Quy trình cơ bản được quản lý nhưng chưa có tiêu chuẩn hóa toàn bộ.
  + Mức 3: Quy trình được tiêu chuẩn hóa trên toàn tổ chức.
* Ví dụ: Ở mức 2, một nhóm phát triển phần mềm có thể theo quy trình riêng của họ, nhưng ở mức 3, công ty áp dụng một quy trình chung cho tất cả các dự án.

Câu hỏi 5: Workflow kiểm thử có nhiệm vụ gì?

* Đánh giá chất lượng phần mềm, tìm lỗi và xác nhận hệ thống đáp ứng yêu cầu.
* Ví dụ: Trước khi ra mắt một ứng dụng ngân hàng di động, nhóm QA sẽ kiểm tra chức năng đăng nhập, giao dịch và bảo mật để đảm bảo không có lỗi nghiêm trọng.

Câu hỏi 6: Mô hình CMM có bao nhiêu mức?

* 5 mức:
  1. Initial (Ban đầu) – Không có quy trình cố định
  2. Managed (Quản lý) – Quy trình được quản lý
  3. Defined (Định nghĩa) – Quy trình được tiêu chuẩn hóa
  4. Quantitatively Managed (Đo lường định lượng) – Kiểm soát bằng dữ liệu
  5. Optimizing (Tối ưu hóa) – Liên tục cải tiến
* Ví dụ: Một công ty khởi nghiệp có thể ở mức 1, nhưng một công ty phần mềm lớn như Microsoft có thể ở mức 5.

Câu hỏi 7: Khác biệt giữa mô hình thác nước và mô hình lặp là gì?

* Trả lời:
  + Thác nước (Waterfall): Phát triển phần mềm theo từng bước tuần tự, không quay lại bước trước.
  + Mô hình lặp (Iterative): Phát triển theo vòng lặp, có thể cải tiến dần theo phản hồi.
* Ví dụ:
  + Thác nước: Khi xây dựng phần mềm kế toán, nhóm phát triển làm lần lượt từ yêu cầu → thiết kế → lập trình → kiểm thử → triển khai.
  + Lặp: Khi phát triển một ứng dụng AI, nhóm liên tục cập nhật mô hình dựa trên dữ liệu mới.

Câu hỏi 8: Tiến trình thống nhất có phải là mô hình lặp không?

* Có, vì tiến trình thống nhất phát triển phần mềm theo nhiều vòng lặp, cải tiến liên tục qua từng giai đoạn.
* Ví dụ: Khi phát triển một hệ thống đặt vé máy bay, nhóm có thể xây dựng trước tính năng đặt vé, sau đó lặp lại để bổ sung thanh toán online và quản lý hành lý.

Câu hỏi 9: Mục đích của workflow thiết kế là gì?

* Xác định kiến trúc hệ thống, tạo mô hình thiết kế và đảm bảo phần mềm có thể mở rộng và bảo trì dễ dàng.
* Ví dụ: Khi thiết kế ứng dụng nhắn tin, nhóm kỹ sư sẽ quyết định việc sử dụng kiến trúc microservices hay monolithic.

Câu hỏi 10: CMM mức 5 tập trung vào điều gì?

* Tối ưu hóa quy trình liên tục để nâng cao hiệu suất và chất lượng sản phẩm.
* Ví dụ: Google sử dụng phân tích dữ liệu để tối ưu hóa quy trình phát triển AI, đảm bảo mô hình AI hoạt động ngày càng chính xác hơn.

**Part 3: Câu hỏi thảo luận nhóm**

Câu hỏi 3: Thảo luận về các ưu và nhược điểm của mô hình lặp và tăng trưởng.

Mô hình lặp và tăng trưởng (Iterative and Incremental Model)

Mô hình này phát triển phần mềm thông qua các vòng lặp, trong đó mỗi lần lặp sẽ bổ sung thêm tính năng hoặc cải thiện hệ thống hiện có. Mô hình này phổ biến trong các phương pháp phát triển hiện đại như Agile, Scrum, và Rational Unified Process (RUP).

Ưu điểm:

1. Giảm rủi ro
   * Vì phần mềm được xây dựng dần dần, các vấn đề có thể được phát hiện sớm và điều chỉnh kịp thời.
   * Nếu một vòng lặp có lỗi hoặc chưa đáp ứng nhu cầu, nhóm có thể thay đổi mà không ảnh hưởng đến toàn bộ hệ thống.
2. Phản hồi nhanh từ khách hàng
   * Khách hàng có thể thấy sản phẩm ngay từ những phiên bản đầu tiên và đóng góp ý kiến để cải tiến.
   * Tránh việc xây dựng một hệ thống hoàn chỉnh rồi mới nhận ra rằng nó không phù hợp.
3. Dễ dàng thích ứng với thay đổi
   * Nếu yêu cầu thay đổi, chỉ cần điều chỉnh vòng lặp sau thay vì thay đổi toàn bộ dự án như trong mô hình thác nước (*Waterfall*).
   * Thích hợp với những dự án có yêu cầu chưa rõ ràng ngay từ đầu.
4. Kiểm soát tiến độ và ngân sách tốt hơn
   * Nhờ vào các vòng lặp nhỏ, nhóm có thể ước lượng chi phí và thời gian tốt hơn.
   * Nhà quản lý có thể dừng dự án sau một số vòng lặp nếu thấy không khả thi, giảm lãng phí tài nguyên.

Nhược điểm:

1. Có thể dẫn đến chi phí cao hơn
   * Vì phần mềm được cải tiến liên tục, có thể mất nhiều thời gian và công sức hơn so với việc lập kế hoạch chi tiết ngay từ đầu.
   * Một số vòng lặp có thể cần viết lại mã nguồn hoặc thiết kế lại hệ thống.
2. Dễ bị mất kiểm soát nếu không quản lý tốt
   * Nếu không có kế hoạch chặt chẽ, nhóm có thể tiếp tục thêm vòng lặp mà không có điểm dừng cụ thể.
   * Dự án có thể kéo dài hơn dự kiến do liên tục thay đổi.
3. Yêu cầu đội ngũ phát triển có kỹ năng cao
   * Cần những lập trình viên và nhà phân tích giỏi để xác định các yêu cầu, đánh giá rủi ro và thiết kế linh hoạt.
   * Nếu đội ngũ không có kinh nghiệm, việc quản lý các vòng lặp có thể trở nên hỗn loạn.

Ví dụ thực tế:

* Facebook sử dụng mô hình lặp để cải tiến giao diện và tính năng, thử nghiệm các bản cập nhật nhỏ trước khi triển khai rộng rãi.
* Google Chrome liên tục cập nhật theo chu kỳ ngắn để cải thiện hiệu suất và bảo mật.

Câu hỏi 4: Vì sao mô hình CMM được sử dụng rộng rãi trong quản lý chất lượng phần mềm?

Mô hình CMM (Capability Maturity Model là một mô hình giúp các tổ chức đánh giá và cải thiện quy trình phát triển phần mềm, đảm bảo chất lượng và hiệu suất.

Lý do mô hình CMM được sử dụng rộng rãi:

1. Chuẩn hóa quy trình phát triển phần mềm
   * CMM cung cấp một khung làm việc rõ ràng để quản lý phần mềm theo từng cấp độ trưởng thành (5 mức từ *Initial* đến *Optimizing*).
   * Giúp các công ty thiết lập các quy trình chuẩn, tránh việc làm việc theo cách ngẫu nhiên hoặc thiếu hệ thống.
2. Cải thiện chất lượng sản phẩm
   * CMM tập trung vào việc kiểm soát và đo lường chất lượng phần mềm, giảm thiểu lỗi và rủi ro.
   * Mức 4 và 5 của CMM giúp tổ chức sử dụng dữ liệu để tối ưu hóa hiệu suất.
3. Giảm chi phí phát triển phần mềm
   * Khi quy trình được chuẩn hóa, việc phát hiện và sửa lỗi sớm giúp giảm chi phí sửa chữa sau này.
   * CMM giúp tăng hiệu suất làm việc, giảm thời gian phát triển và kiểm thử.
4. Tăng cường khả năng cạnh tranh
   * Các công ty phần mềm tuân theo CMM thường có lợi thế khi đấu thầu dự án lớn vì họ chứng minh được năng lực và chất lượng sản phẩm.
   * Các công ty đạt mức CMM cao thường được đánh giá cao hơn trên thị trường quốc tế.
5. Cải thiện khả năng quản lý rủi ro
   * CMM yêu cầu đánh giá rủi ro và cải tiến liên tục, giúp doanh nghiệp lường trước các vấn đề tiềm ẩn.
   * Các tổ chức có thể dự báo tốt hơn về thời gian và nguồn lực cần thiết cho mỗi dự án.

Ví dụ thực tế:

* IBM và Microsoft áp dụng CMM để kiểm soát chất lượng phần mềm và tối ưu hóa quy trình phát triển.
* Boeing sử dụng CMM trong phát triển hệ thống phần mềm hàng không, đảm bảo tiêu chuẩn an toàn cao nhất.
* Các công ty outsource tại Ấn Độ như Infosys, TCS đạt mức CMM-5 giúp họ đảm bảo chất lượng dịch vụ khi làm việc với khách hàng quốc tế.

**Part 4: Câu hỏi tình huống**

Câu 3: Dự án phát triển phần mềm bị trễ tiến độ do lỗi phát sinh liên tục trong quá trình kiểm thử. Là trưởng dự án, bạn sẽ làm gì?

Hướng xử lý:

1. Phân tích nguyên nhân gốc rễ (Root Cause Analysis - RCA):
   * Xác định xem lỗi xuất phát từ yêu cầu không rõ ràng, thiết kế kém, code chưa tối ưu, hay kiểm thử chưa đầy đủ.
   * Thu thập thông tin từ team QA, Dev, BA để đánh giá nguyên nhân chính.
2. Ưu tiên và phân loại lỗi (Prioritize Bugs & Issues):
   * Phân loại lỗi theo mức độ nghiêm trọng: blocker, critical, major, minor.
   * Tập trung sửa các lỗi blocker & critical trước để giảm rủi ro.
3. Xây dựng chiến lược sửa lỗi hiệu quả:
   * Áp dụng mô hình Shift-Left Testing (đưa kiểm thử vào sớm trong quy trình phát triển).
   * Bổ sung kiểm thử tự động (Automated Testing) nếu có thể.
   * Cải thiện quy trình Code Review và Continuous Integration/Continuous Deployment (CI/CD).
4. Điều chỉnh kế hoạch và nhân sự:
   * Nếu cần, phân bổ lại nguồn lực hoặc bổ sung nhân sự để đẩy nhanh việc sửa lỗi.
   * Sắp xếp lại thứ tự công việc để đảm bảo tiến độ chung.
5. Giao tiếp với Stakeholders:
   * Báo cáo tình hình với khách hàng hoặc ban lãnh đạo để có phương án điều chỉnh deadline hợp lý.
   * Cam kết kế hoạch khắc phục và đưa ra timeline cập nhật.

Câu 4: Trong workflow thiết kế, kiến trúc sư phần mềm muốn thay đổi thiết kế ban đầu để cải thiện hiệu suất. Đội phát triển nên xử lý thế nào?

Hướng xử lý:

1. Đánh giá tác động của thay đổi:
   * Xác định xem thay đổi này có ảnh hưởng đến tiến độ, tài nguyên hay chất lượng sản phẩm không.
   * Phân tích tác động đối với các module liên quan.
2. Thảo luận và thống nhất với team:
   * Tổ chức họp giữa kiến trúc sư, trưởng nhóm phát triển, QA và Product Owner để xem xét đề xuất.
   * Nếu thay đổi có lợi nhưng cần thêm thời gian, thương lượng lại timeline với stakeholders.
3. Xác định phương án triển khai:
   * Nếu thay đổi nhỏ, có thể áp dụng ngay trong sprint hiện tại.
   * Nếu thay đổi lớn, cần lập kế hoạch cụ thể (refactor code, cập nhật tài liệu, kiểm thử lại).
4. Áp dụng nguyên tắc thiết kế phần mềm linh hoạt:
   * Nếu dự án đang theo Agile/Scrum, cần đảm bảo việc thay đổi không làm ảnh hưởng đến sprint hiện tại.
   * Nếu sử dụng kiến trúc Microservices, có thể áp dụng thay đổi theo từng module nhỏ để giảm rủi ro.
5. Thử nghiệm và triển khai:
   * Xây dựng Proof of Concept (PoC) nếu cần.
   * Kiểm thử hiệu suất sau khi áp dụng thay đổi.
   * Đảm bảo backward compatibility nếu thay đổi ảnh hưởng đến hệ thống hiện tại.