Họ và tên: Lâm Thành Trung  
MSS: N18DCAT096  
Bài tập tuần 2 – Nhập môn công nghệ phần mềm

**Câu hỏi chương 2**  
**Câu hỏi trắc nghiệm  
Câu hỏi 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**

Workflow lấy yêu cầu (Requirement Workflow) là công việc đầu tiên và quan trọng nhất trong quy trình phát triển phần mềm. Mục tiêu chính của workflow này là thu thập, phân tích và tài liệu hóa tất cả các yêu cầu từ khách hàng và các bên liên quan.

**Câu hỏi 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: A. Pha khởi đầu**

Pha khởi đầu (Inception Phase) trong tiến trình thống nhất tập trung vào việc xác định phạm vi dự án, đánh giá tính khả thi, xác định các rủi ro chính và phác thảo kiến trúc hệ thống ban đầu. Mục tiêu là quyết định xem có nên tiếp tục dự án hay không.

**Câu hỏi 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**

Mô hình CMM (Capability Maturity Model) mức 4 là "Được quản lý định lượng" (Quantitatively Managed). Ở mức này, tổ chức sử dụng dữ liệu và các phép đo để kiểm soát và quản lý quy trình phát triển phần mềm một cách định lượng.

**Câu hỏi 4: Các pha trong tiến trình thống nhất bao gồm:**

**Đáp án đúng: B. Khởi đầu, làm rõ, xây dựng, chuyển giao**

Tiến trình thống nhất (Unified Process - UP) chia vòng đời phát triển phần mềm thành bốn pha chính: Pha khởi đầu (Inception), Pha làm rõ (Elaboration), Pha xây dựng (Construction), và Pha chuyển giao (Transition).

**Câu hỏi 5: Trong tiến trình thống nhất workflow nào thực hiện sau cùng?**

**Đáp án đúng: D. Workflow kiểm thử**

Trong tiến trình thống nhất, Workflow kiểm thử (Test Workflow) được thực hiện xuyên suốt các pha, nhưng nó đặc biệt quan trọng và thường được đẩy mạnh vào giai đoạn cuối, gần với pha chuyển giao để đảm bảo chất lượng phần mềm trước khi bàn giao cho khách hàng. Tuy nhiên, nếu xét về workflow *cuối cùng* trong một vòng lặp nhỏ (iteration) của UP, thì Workflow triển khai (Deployment Workflow) sẽ là workflow sau cùng để đưa sản phẩm vào sử dụng. Nhưng trong các lựa chọn đưa ra, Workflow kiểm thử là đáp án hợp lý nhất xét về vị trí tương đối *sau cùng* trong quá trình phát triển.

**Câu hỏi 6: Mô hình CMM mức 1 có đặc điểm gì?**

**Đáp án đúng: C. Quy trình không ổn định, phụ thuộc vào cá nhân**

Mức 1 của CMM là "Sơ khai" (Initial). Ở mức này, quy trình phát triển phần mềm thường không ổn định, mang tính chất ứng phó, và thành công của dự án phụ thuộc nhiều vào năng lực cá nhân của các thành viên hơn là vào quy trình bài bản.

**Câu hỏi 7: Tiến trình thống nhất là một ví dụ của mô hình nào?**

**Đáp án đúng: B. Mô hình lặp và tăng trưởng**

Tiến trình thống nhất (Unified Process) là một mô hình phát triển phần mềm lặp và tăng trưởng (Iterative and Incremental). Nó phát triển phần mềm qua nhiều vòng lặp, mỗi vòng lặp xây dựng thêm các chức năng và cải tiến sản phẩm.

**Câu hỏi 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**

Mức 5 của CMM là "Tối ưu hóa" (Optimizing). Ở mức cao nhất này, tổ chức tập trung vào việc cải tiến quy trình phát triển phần mềm một cách liên tục dựa trên phản hồi và phân tích dữ liệu để nâng cao hiệu quả và chất lượng.

**Câu hỏi 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**

Workflow thiết kế (Design Workflow) tập trung vào việc chuyển đổi các yêu cầu đã thu thập được thành bản thiết kế hệ thống. Hoạt động chính bao gồm thiết kế kiến trúc tổng thể của hệ thống và thiết kế chi tiết các thành phần bên trong.

**Câu hỏi 10: B  
  
  
Câu hỏi trả lời ngắn**

1. Pha khởi đầu trong tiến trình thống nhất là gì?

Pha khởi đầu (Inception Phase) là pha đầu tiên trong Tiến trình Thống nhất (Unified Process - UP). Mục tiêu chính của pha này là:

* Xác định phạm vi dự án: Hiểu rõ dự án sẽ làm gì, không làm gì, và giới hạn của nó.
* Đánh giá tính khả thi: Xem xét liệu dự án có khả thi về mặt kỹ thuật, tài chính, thời gian, và nguồn lực hay không.
* Thu thập yêu cầu sơ bộ: Bước đầu thu thập các yêu cầu quan trọng nhất từ khách hàng và các bên liên quan.
* Lập kế hoạch sơ bộ: Phác thảo kế hoạch dự án ban đầu, bao gồm ước tính chi phí, thời gian, và các mốc quan trọng.
* Đánh giá rủi ro: Nhận diện và đánh giá các rủi ro tiềm ẩn có thể ảnh hưởng đến dự án.
* Xây dựng kiến trúc ban đầu: Phác thảo kiến trúc hệ thống ở mức cao, đủ để đánh giá tính khả thi về mặt kỹ thuật.

Tóm lại, pha khởi đầu giống như việc "khởi động" dự án, đảm bảo rằng dự án có một nền tảng vững chắc trước khi tiến hành các giai đoạn phát triển chi tiết hơn.

2. Mục tiêu của workflow lấy yêu cầu là gì?

Mục tiêu chính của Workflow lấy yêu cầu (Requirement Workflow) là hiểu rõ và ghi lại đầy đủ các nhu cầu của khách hàng và người dùng đối với phần mềm cần phát triển. Cụ thể hơn, mục tiêu của workflow này bao gồm:

* Thu thập yêu cầu: Lắng nghe và thu thập thông tin về nhu cầu từ tất cả các bên liên quan (khách hàng, người dùng cuối, các bộ phận khác trong tổ chức, v.v.) thông qua nhiều phương pháp khác nhau (phỏng vấn, khảo sát, hội thảo, phân tích tài liệu hiện có, v.v.).
* Phân tích yêu cầu: Làm rõ các yêu cầu đã thu thập được, đảm bảo chúng rõ ràng, đầy đủ, nhất quán, và không mâu thuẫn.
* Đặc tả yêu cầu: Ghi lại các yêu cầu dưới dạng tài liệu đặc tả yêu cầu một cách chi tiết, dễ hiểu, và có thể kiểm chứng được. Tài liệu này sẽ là cơ sở để thiết kế, phát triển, và kiểm thử phần mềm.
* Xác nhận yêu cầu: Xác nhận lại các yêu cầu đã đặc tả với khách hàng và các bên liên quan để đảm bảo rằng mọi người đều hiểu và đồng ý với các yêu cầu đó.

Nói ngắn gọn, Workflow lấy yêu cầu đảm bảo rằng chúng ta "xây đúng thứ mà khách hàng muốn" bằng cách hiểu rõ và ghi lại chính xác mong muốn của họ.

3. Tiến trình thống nhất gồm bao nhiêu pha chính?

Tiến trình Thống nhất (Unified Process) bao gồm bốn pha chính (được sắp xếp theo thứ tự thời gian):

1. Pha Khởi đầu (Inception Phase): Như đã giải thích ở câu 1, pha này tập trung vào việc khởi động dự án và xác định phạm vi.
2. Pha Làm rõ (Elaboration Phase): Pha này tập trung vào việc lập kế hoạch chi tiết, xác định các yêu cầu chức năng chính, thiết kế kiến trúc hệ thống chi tiết hơn, và giảm thiểu các rủi ro chính.
3. Pha Xây dựng (Construction Phase): Pha này là giai đoạn phát triển chính của phần mềm, tập trung vào việc xây dựng và kiểm thử các thành phần phần mềm dựa trên thiết kế đã được xác định ở pha làm rõ.
4. Pha Chuyển giao (Transition Phase): Pha cuối cùng này tập trung vào việc chuyển giao phần mềm đã hoàn thiện cho người dùng cuối, bao gồm các hoạt động như kiểm thử chấp nhận người dùng, đào tạo người dùng, triển khai phần mềm, và khắc phục lỗi (nếu có) sau triển khai.

Mỗi pha trong Tiến trình Thống nhất đều có mục tiêu cụ thể và kết thúc bằng một mốc quan trọng (milestone) để đánh giá tiến độ và đưa ra quyết định tiếp tục hay điều chỉnh dự án.

4. Sự khác nhau giữa CMM mức 2 và mức 3 là gì?

Mô hình CMM (Capability Maturity Model) là một mô hình đánh giá mức độ trưởng thành của quy trình phát triển phần mềm của một tổ chức. Mức 2 và mức 3 là hai mức độ trưởng thành khác nhau, với sự khác biệt chính như sau:

* CMM Mức 2: Được quản lý (Managed)
  + Tập trung vào dự án: Ở mức 2, tổ chức đã bắt đầu có các quy trình được quản lý cho từng dự án cụ thể.
  + Quy trình lặp lại: Các quy trình này được lập kế hoạch, thực hiện, đo lường và kiểm soát. Dự án có thể lặp lại các thành công của dự án trước đó.
  + Ví dụ: Quản lý yêu cầu, quản lý dự án, đảm bảo chất lượng, quản lý cấu hình.
  + Tuy nhiên: Các quy trình này thường chỉ áp dụng cho từng dự án và có thể khác nhau giữa các dự án. Chưa có quy trình chuẩn hóa chung cho toàn tổ chức.
* CMM Mức 3: Được định nghĩa (Defined)
  + Tập trung vào tổ chức: Ở mức 3, tổ chức đã có một tập hợp các quy trình chuẩn hóa được định nghĩa và áp dụng cho toàn bộ tổ chức.
  + Quy trình chủ động: Các quy trình này được mô tả rõ ràng, được chuẩn hóa và được cải tiến liên tục. Dự án được thực hiện theo các quy trình đã được định nghĩa này.
  + Ví dụ: Phát triển yêu cầu, thiết kế kỹ thuật, kiểm thử phần mềm, tích hợp hệ thống (được định nghĩa là quy trình chuẩn của tổ chức).
  + Lợi ích: Tính nhất quán cao hơn giữa các dự án, dễ dàng dự đoán và kiểm soát chất lượng, hiệu quả hoạt động của tổ chức được nâng cao.

Tóm lại, sự khác biệt chính là ở mức 2 quy trình được quản lý ở mức dự án, còn ở mức 3 quy trình được chuẩn hóa và áp dụng cho toàn tổ chức. Mức 3 thể hiện sự trưởng thành cao hơn về quy trình so với mức 2.

5. Workflow kiểm thử có nhiệm vụ gì?

Workflow kiểm thử (Test Workflow) có nhiệm vụ đảm bảo chất lượng của phần mềm được phát triển. Nhiệm vụ chính của workflow này bao gồm:

* Lập kế hoạch kiểm thử: Xác định phạm vi kiểm thử, các loại kiểm thử cần thực hiện, nguồn lực cần thiết, và lịch trình kiểm thử.
* Thiết kế kiểm thử: Tạo ra các kịch bản kiểm thử, trường hợp kiểm thử, và dữ liệu kiểm thử để kiểm tra các chức năng và phi chức năng của phần mềm.
* Thực hiện kiểm thử: Tiến hành chạy các trường hợp kiểm thử đã thiết kế, ghi lại kết quả kiểm thử và các lỗi phát hiện được.
* Báo cáo lỗi: Báo cáo chi tiết các lỗi (bug) cho đội phát triển để sửa chữa.
* Kiểm thử lại: Sau khi lỗi được sửa, thực hiện kiểm thử lại để đảm bảo lỗi đã được khắc phục và không gây ra lỗi mới.
* Đánh giá chất lượng: Đánh giá chất lượng tổng thể của phần mềm dựa trên kết quả kiểm thử, đảm bảo phần mềm đáp ứng các yêu cầu và tiêu chuẩn chất lượng đã đặt ra.

Workflow kiểm thử là một quá trình lặp đi lặp lại và thường được thực hiện xuyên suốt vòng đời phát triển phần mềm, không chỉ ở giai đoạn cuối. Mục tiêu là phát hiện lỗi sớm nhất có thể để giảm chi phí sửa chữa và nâng cao chất lượng sản phẩm cuối cùng.

6. Mô hình CMM có bao nhiêu mức?

Mô hình CMM (Capability Maturity Model) có 5 mức độ trưởng thành, được sắp xếp theo thứ tự tăng dần về độ trưởng thành của quy trình phát triển phần mềm:

1. Mức 1: Sơ khai (Initial): Quy trình không ổn định, phụ thuộc vào cá nhân, thành công mang tính chất ngẫu nhiên.
2. Mức 2: Được quản lý (Managed): Quy trình được quản lý ở mức dự án, có kế hoạch, có kiểm soát, nhưng chưa chuẩn hóa toàn tổ chức.
3. Mức 3: Được định nghĩa (Defined): Quy trình được chuẩn hóa và định nghĩa cho toàn tổ chức, được áp dụng cho mọi dự án.
4. Mức 4: Được quản lý định lượng (Quantitatively Managed): Sử dụng các phép đo và dữ liệu để kiểm soát và quản lý quy trình một cách định lượng, có thể dự đoán hiệu suất.
5. Mức 5: Tối ưu hóa (Optimizing): Tập trung vào cải tiến quy trình liên tục dựa trên phản hồi và phân tích dữ liệu, hướng tới sự xuất sắc và đổi mới.

Mỗi mức độ thể hiện một giai đoạn phát triển của tổ chức trong việc cải thiện quy trình phát triển phần mềm của mình. Mức càng cao, quy trình càng trưởng thành và hiệu quả.

7. Khác biệt giữa mô hình thác nước và mô hình lặp là gì?

Mô hình thác nước (Waterfall model) và mô hình lặp (Iterative model) là hai mô hình phát triển phần mềm có cách tiếp cận hoàn toàn khác nhau:

* Mô hình Thác Nước (Waterfall):
  + Tuần tự, tuyến tính: Các giai đoạn phát triển (yêu cầu, thiết kế, thực hiện, kiểm thử, triển khai, bảo trì) được thực hiện tuần tự, lần lượt từ trên xuống như thác nước.
  + Giai đoạn trước phải hoàn thành trước khi sang giai đoạn sau: Không có sự quay lại giai đoạn trước, hoặc rất hạn chế.
  + Phù hợp: Với các dự án có yêu cầu rõ ràng, ổn định, ít thay đổi, và ít rủi ro.
  + Ưu điểm: Đơn giản, dễ quản lý khi yêu cầu rõ ràng.
  + Nhược điểm: Cứng nhắc, khó thích ứng với thay đổi yêu cầu, rủi ro cao nếu yêu cầu ban đầu không chính xác, thời gian phát hiện lỗi muộn (ở giai đoạn kiểm thử cuối cùng).
* Mô hình Lặp (Iterative):
  + Lặp đi lặp lại: Phát triển phần mềm theo các vòng lặp (iterations) ngắn. Mỗi vòng lặp bao gồm các giai đoạn phát triển (yêu cầu, thiết kế, thực hiện, kiểm thử) cho một phần nhỏ của chức năng.
  + Tăng trưởng: Qua mỗi vòng lặp, sản phẩm được xây dựng từng bước, từng bước và có thêm các chức năng mới (tăng trưởng).
  + Phản hồi và điều chỉnh: Sau mỗi vòng lặp, có sự đánh giá, phản hồi từ khách hàng và các bên liên quan, và kế hoạch cho vòng lặp tiếp theo có thể được điều chỉnh dựa trên phản hồi này.
  + Phù hợp: Với các dự án có yêu cầu chưa rõ ràng, có thể thay đổi, nhiều rủi ro, cần sự linh hoạt và phản hồi thường xuyên từ khách hàng.
  + Ưu điểm: Linh hoạt, dễ thích ứng với thay đổi yêu cầu, giảm thiểu rủi ro do phát hiện lỗi sớm, có sản phẩm hoạt động sớm (dù chưa hoàn chỉnh) để trình diễn cho khách hàng.
  + Nhược điểm: Phức tạp hơn trong quản lý so với mô hình thác nước, cần sự phối hợp chặt chẽ giữa các thành viên.

Tóm lại, mô hình thác nước là tuyến tính và cứng nhắc, còn mô hình lặp là linh hoạt và thích ứng với thay đổi. Mô hình lặp phù hợp với môi trường phát triển phần mềm hiện đại, nơi yêu cầu thường xuyên thay đổi và cần sự tương tác liên tục với khách hàng.

8. Tiến trình thống nhất có phải là mô hình lặp không?

Đúng, Tiến trình Thống nhất (Unified Process - UP) là một mô hình phát triển phần mềm lặp và tăng trưởng (Iterative and Incremental).

Như đã giải thích ở câu 7, mô hình lặp phát triển phần mềm qua các vòng lặp ngắn, mỗi vòng lặp tạo ra một phiên bản phần mềm có thể hoạt động được (dù chưa hoàn chỉnh) và có sự phản hồi sau mỗi vòng lặp. Tiến trình Thống nhất hoàn toàn tuân theo nguyên tắc này:

* Vòng lặp (Iterations): UP chia quá trình phát triển thành các vòng lặp (iterations). Mỗi vòng lặp đi qua các workflow (lấy yêu cầu, phân tích, thiết kế, thực hiện, kiểm thử, triển khai).
* Tăng trưởng (Incremental): Qua mỗi vòng lặp, phần mềm được xây dựng thêm các chức năng mới, tăng dần về độ hoàn thiện và đáp ứng đầy đủ các yêu cầu.
* Phản hồi (Feedback): Sau mỗi vòng lặp, có sự đánh giá, phản hồi từ khách hàng và các bên liên quan để điều chỉnh kế hoạch cho các vòng lặp tiếp theo.

Vì vậy, Tiến trình Thống nhất chính là một ví dụ điển hình của mô hình phát triển phần mềm lặp.

9. Mục đích của workflow thiết kế là gì?

Mục đích chính của Workflow thiết kế (Design Workflow) là chuyển đổi các yêu cầu đã được xác định (trong Workflow lấy yêu cầu) thành một bản thiết kế chi tiết cho hệ thống phần mềm. Cụ thể, workflow thiết kế có các mục tiêu sau:

* Thiết kế kiến trúc hệ thống: Xác định cấu trúc tổng thể của hệ thống, các thành phần chính, cách chúng tương tác với nhau, và các công nghệ sẽ sử dụng. Kiến trúc hệ thống là "bản vẽ tổng quan" của phần mềm.
* Thiết kế chi tiết: Thiết kế chi tiết cho từng thành phần, module, lớp, giao diện, cơ sở dữ liệu, thuật toán, v.v. Bản thiết kế chi tiết cung cấp đầy đủ thông tin để lập trình viên có thể bắt đầu viết code.
* Đảm bảo tính khả thi về mặt kỹ thuật: Đảm bảo rằng thiết kế hệ thống là khả thi về mặt kỹ thuật, có thể hiện thực hóa được với công nghệ và nguồn lực hiện có.
* Tối ưu hóa hiệu năng và chất lượng: Thiết kế hệ thống sao cho đáp ứng được các yêu cầu về hiệu năng (tốc độ, khả năng mở rộng, v.v.) và chất lượng (bảo mật, độ tin cậy, khả năng bảo trì, v.v.).
* Truyền đạt thiết kế cho đội phát triển: Tài liệu hóa thiết kế một cách rõ ràng, dễ hiểu để đội phát triển có thể sử dụng làm cơ sở để thực hiện code và kiểm thử.

Nói tóm lại, Workflow thiết kế tạo ra "bản thiết kế xây dựng" cho phần mềm, hướng dẫn cách thức xây dựng phần mềm để đáp ứng các yêu cầu đã được xác định.

10. CMM mức 5 tập trung vào điều gì?

CMM mức 5 (Mức Tối ưu hóa - Optimizing) là mức cao nhất trong mô hình CMM, và nó tập trung vào cải tiến quy trình liên tục (Continuous Process Improvement). Cụ thể, trọng tâm của CMM mức 5 là:

* Cải tiến liên tục: Tổ chức luôn tìm kiếm cơ hội để cải tiến quy trình phát triển phần mềm của mình một cách liên tục, không ngừng nghỉ.
* Đổi mới và sáng tạo: Khuyến khích đổi mới, sáng tạo trong quy trình và công nghệ để nâng cao hiệu quả và chất lượng.
* Phân tích nguyên nhân gốc rễ: Khi có vấn đề xảy ra, tập trung vào việc phân tích nguyên nhân gốc rễ thay vì chỉ xử lý triệu chứng, để ngăn chặn vấn đề tái diễn.
* Học hỏi từ kinh nghiệm: Tổ chức học hỏi từ cả thành công và thất bại, thu thập dữ liệu, phân tích hiệu suất, và sử dụng thông tin này để cải tiến quy trình.
* Sử dụng công nghệ tiên tiến: Chủ động tìm kiếm và áp dụng các công nghệ và phương pháp mới để cải thiện quy trình phát triển phần mềm.
* Văn hóa cải tiến: Xây dựng một văn hóa cải tiến trong toàn tổ chức, khuyến khích mọi thành viên đóng góp ý kiến và tham gia vào quá trình cải tiến quy trình.

Tóm lại, CMM mức 5 không chỉ duy trì quy trình tốt mà còn liên tục tìm cách để làm cho quy trình trở nên tốt hơn nữa, hướng tới sự xuất sắc trong phát triển phần mềm.

**Câu hỏi thảo luận nhóm**Câu 1: Thảo luận về vai trò của từng workflow trong tiến trình phát triển phần mềm.

Trong tiến trình phát triển phần mềm, "workflow" đề cập đến chuỗi các hoạt động và nhiệm vụ được thực hiện để hoàn thành một mục tiêu cụ thể. Các workflow này giúp tổ chức và quản lý quá trình phát triển, đảm bảo rằng mọi bước đều được thực hiện một cách có hệ thống và hiệu quả.

Dưới đây là một số workflow phổ biến và vai trò của chúng:

* Lấy yêu cầu (Requirements Workflow):
  + Vai trò: Thu thập, phân tích và xác định các yêu cầu của người dùng và hệ thống.
  + Mục tiêu: Đảm bảo rằng phần mềm được phát triển đáp ứng đúng nhu cầu của người dùng.
* Phân tích (Analysis Workflow):
  + Vai trò: Phân tích các yêu cầu để hiểu rõ hơn về hệ thống cần xây dựng.
  + Mục tiêu: Tạo ra các mô hình và tài liệu mô tả chi tiết về chức năng và cấu trúc của hệ thống.
* Thiết kế (Design Workflow):
  + Vai trò: Thiết kế kiến trúc và chi tiết của hệ thống, bao gồm giao diện người dùng, cơ sở dữ liệu và các thành phần phần mềm.
  + Mục tiêu: Tạo ra một bản thiết kế chi tiết để hướng dẫn quá trình lập trình.
* Lập trình (Implementation Workflow):
  + Vai trò: Viết mã nguồn và xây dựng các thành phần phần mềm.
  + Mục tiêu: Hiện thực hóa thiết kế thành một hệ thống phần mềm hoạt động.
* Kiểm thử (Test Workflow):
  + Vai trò: Kiểm tra và đánh giá chất lượng của phần mềm để đảm bảo rằng nó hoạt động đúng và đáp ứng các yêu cầu.
  + Mục tiêu: Phát hiện và sửa chữa các lỗi trước khi phần mềm được triển khai.
* Triển khai (Deployment Workflow):
  + Vai trò: Đưa phần mềm vào hoạt động trong môi trường thực tế.
  + Mục tiêu: Cung cấp phần mềm cho người dùng cuối.
* Bảo trì (Maintenance Workflow):
  + Vai trò: Sửa chữa lỗi, cập nhật và cải tiến phần mềm sau khi triển khai.
  + Mục tiêu: Đảm bảo rằng phần mềm tiếp tục hoạt động ổn định và đáp ứng các nhu cầu thay đổi.

Câu 2: Phân biệt mô hình vòng đời thác nước và tiến trình thống nhất.

* Mô hình vòng đời thác nước (Waterfall Model):
  + Là một mô hình tuyến tính, tuần tự, trong đó mỗi giai đoạn phải được hoàn thành trước khi chuyển sang giai đoạn tiếp theo.
  + Các giai đoạn bao gồm: Yêu cầu, thiết kế, lập trình, kiểm thử và bảo trì.
  + Ưu điểm: Đơn giản, dễ hiểu và dễ quản lý.
  + Nhược điểm: Thiếu linh hoạt, khó xử lý các thay đổi yêu cầu và rủi ro cao nếu có lỗi ở giai đoạn đầu.
* Tiến trình thống nhất (Unified Process):
  + Là một mô hình lặp và tăng trưởng, trong đó các giai đoạn được thực hiện lặp đi lặp lại và tăng dần theo thời gian.
  + Các giai đoạn bao gồm: Khởi đầu, xây dựng, chuyển giao và sản xuất.
  + Ưu điểm: Linh hoạt, dễ xử lý các thay đổi yêu cầu, giảm thiểu rủi ro và tập trung vào người dùng.
  + Nhược điểm: Phức tạp hơn, đòi hỏi kỹ năng quản lý cao và có thể tốn nhiều thời gian hơn.

Tóm tắt sự khác biệt:

* Thác nước: Tuyến tính, tuần tự.
* Tiến trình thống nhất: Lặp, tăng trưởng.
* Thác nước: Ít linh hoạt, rủi ro cao.
* Tiến trình thống nhất: Linh hoạt, giảm thiểu rủi ro.

**Câu hỏi tình huống**

1. Một công ty phát triển phần mềm gặp khó khăn khi yêu cầu của khách hàng liên tục thay đổi trong pha xây dựng. Đội phát triển nên làm gì để giải quyết vấn đề này?

Vấn đề: Yêu cầu thay đổi liên tục trong pha xây dựng (Construction Phase) là một thách thức lớn, vì đây là giai đoạn tập trung vào việc hiện thực hóa thiết kế thành sản phẩm. Thay đổi liên tục có thể dẫn đến trễ tiến độ, tăng chi phí, và giảm chất lượng sản phẩm.

Giải pháp đội phát triển nên làm:

* Áp dụng quy trình phát triển linh hoạt (Agile): Chuyển sang các phương pháp Agile như Scrum hoặc Kanban. Agile được thiết kế để đối phó với sự thay đổi. Các sprint ngắn (ví dụ 2-4 tuần) cho phép đội phát triển thích ứng với các thay đổi yêu cầu sau mỗi sprint.
* Thiết lập quy trình quản lý thay đổi:
  + Đánh giá tác động: Khi nhận được yêu cầu thay đổi, cần đánh giá kỹ lưỡng tác động của thay đổi đó đến tiến độ, chi phí, và các yêu cầu khác.
  + Ưu tiên hóa thay đổi: Không phải tất cả các thay đổi đều cần được thực hiện ngay lập tức. Ưu tiên các thay đổi quan trọng và cấp bách nhất.
  + Thương lượng với khách hàng: Thảo luận với khách hàng về tác động của việc thay đổi yêu cầu, có thể cần điều chỉnh lại phạm vi dự án, tiến độ hoặc chi phí.
  + Ghi nhận thay đổi: Ghi lại tất cả các thay đổi yêu cầu, lý do thay đổi, và các quyết định liên quan.
* Tăng cường giao tiếp với khách hàng: Duy trì giao tiếp thường xuyên và minh bạch với khách hàng để hiểu rõ hơn về lý do thay đổi và quản lý kỳ vọng của họ. Tổ chức các buổi demo sản phẩm thường xuyên để khách hàng có thể đưa ra phản hồi sớm và thường xuyên.
* Thiết kế linh hoạt: Cố gắng thiết kế hệ thống sao cho có khả năng mở rộng và thay đổi dễ dàng hơn. Sử dụng các nguyên tắc thiết kế hướng đối tượng, module hóa, và microservices có thể giúp hệ thống linh hoạt hơn.
* Tập trung vào giá trị cốt lõi: Trong trường hợp bị giới hạn về thời gian và nguồn lực do thay đổi yêu cầu, đội phát triển nên tập trung vào việc cung cấp các tính năng cốt lõi và quan trọng nhất trước.

Nguyên tắc quan trọng: Linh hoạt, giao tiếp hiệu quả, và quản lý thay đổi một cách có hệ thống.

2. Trong pha chuyển giao của tiến trình thống nhất, khách hàng yêu cầu bổ sung thêm tính năng mới. Đội phát triển nên xử lý ra sao?

Vấn đề: Yêu cầu bổ sung tính năng mới trong pha chuyển giao (Transition Phase) là rất muộn. Pha này tập trung vào việc bàn giao sản phẩm đã hoàn thiện, không phải là giai đoạn phát triển tính năng mới. Việc bổ sung tính năng mới có thể ảnh hưởng nghiêm trọng đến kế hoạch bàn giao.

Giải pháp đội phát triển nên xử lý:

* Nguyên tắc đầu tiên: Kháng cự việc thay đổi phạm vi ở giai đoạn này. Pha chuyển giao không phải là thời điểm thích hợp để phát triển tính năng mới.
* Đánh giá nghiêm túc yêu cầu:
  + Tính cấp bách: Hỏi khách hàng về mức độ cấp bách và lý do tại sao tính năng này lại cần thiết ngay bây giờ.
  + Giá trị: Đánh giá giá trị mà tính năng mới mang lại so với công sức và rủi ro khi thực hiện nó vào giai đoạn này.
  + Tác động: Đánh giá tác động của việc bổ sung tính năng mới đến tiến độ bàn giao, chất lượng sản phẩm hiện tại, và các tính năng đã phát triển.
* Đề xuất phương án:
  + Từ chối (nếu có thể): Nếu tính năng không thực sự quan trọng cho bản phát hành hiện tại, và việc bổ sung sẽ gây ra rủi ro lớn, đội phát triển nên giải thích rõ ràng cho khách hàng và đề xuất dời việc phát triển tính năng này sang phiên bản sau.
  + Chấp nhận (nếu thực sự cần thiết): Nếu tính năng là thực sự quan trọng và không thể trì hoãn, đội phát triển cần phải:
    - Đàm phán lại tiến độ và chi phí: Việc bổ sung tính năng mới chắc chắn sẽ làm chậm tiến độ và tăng chi phí. Cần thảo luận với khách hàng để điều chỉnh lại kế hoạch và ngân sách.
    - Đánh giá lại rủi ro: Việc thay đổi vào phút chót luôn tiềm ẩn rủi ro về chất lượng và ổn định. Cần đánh giá lại rủi ro và có biện pháp kiểm soát.
    - Lập kế hoạch lại pha chuyển giao: Pha chuyển giao có thể cần được kéo dài hoặc điều chỉnh để tích hợp tính năng mới và thực hiện kiểm thử đầy đủ.
* Giao tiếp minh bạch: Thông báo rõ ràng với khách hàng về các lựa chọn, tác động của việc bổ sung tính năng mới, và các quyết định liên quan.

Nguyên tắc quan trọng: Bảo vệ pha chuyển giao, đánh giá kỹ lưỡng, đàm phán, và giao tiếp rõ ràng.

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?

Vấn đề: Dự án trễ tiến độ do lỗi liên tục trong quá trình kiểm thử (Testing Phase) cho thấy có vấn đề tiềm ẩn trong quy trình phát triển và chất lượng code.

Giải pháp trưởng dự án nên làm:

* Dừng việc phát triển tính năng mới: Tập trung nguồn lực vào việc sửa lỗi và đảm bảo chất lượng sản phẩm hiện tại thay vì tiếp tục phát triển thêm tính năng mới.
* Ưu tiên sửa lỗi: Phân loại và ưu tiên sửa các lỗi theo mức độ nghiêm trọng và ảnh hưởng. Tập trung vào các lỗi "chặn" (blocker) và "nghiêm trọng" (critical) trước.
* Phân tích nguyên nhân gốc rễ (Root Cause Analysis): Không chỉ sửa lỗi, mà cần tìm ra nguyên nhân tại sao lỗi lại phát sinh nhiều như vậy. Có thể là do:
  + Yêu cầu không rõ ràng: Dẫn đến hiểu sai và code sai ngay từ đầu.
  + Thiết kế kém: Kiến trúc hệ thống hoặc thiết kế chi tiết có vấn đề.
  + Chất lượng code kém: Code viết ẩu, không tuân thủ coding standards, thiếu kiểm thử đơn vị (unit test).
  + Quy trình kiểm thử chưa hiệu quả: Kiểm thử không đủ bao phủ, không phát hiện lỗi sớm.
  + Áp lực tiến độ: Áp lực phải hoàn thành nhanh dẫn đến bỏ qua các bước đảm bảo chất lượng.
* Cải thiện quy trình phát triển: Dựa trên phân tích nguyên nhân gốc rễ, thực hiện các biện pháp cải thiện quy trình:
  + Nâng cao chất lượng yêu cầu và thiết kế: Dành thời gian đầu tư vào pha lấy yêu cầu và pha thiết kế, đảm bảo yêu cầu rõ ràng và thiết kế tốt.
  + Cải thiện chất lượng code: Thực hiện code review, áp dụng coding standards, tăng cường kiểm thử đơn vị.
  + Cải thiện quy trình kiểm thử: Tăng cường kiểm thử sớm và thường xuyên (ví dụ, kiểm thử liên tục - Continuous Testing), sử dụng các công cụ kiểm thử tự động.
  + Đào tạo đội phát triển: Đào tạo về các kỹ năng viết code chất lượng, kiểm thử, và quy trình phát triển tốt.
* Đánh giá lại tiến độ và nguồn lực: Việc sửa lỗi sẽ mất thời gian. Cần đánh giá lại tiến độ dự án, có thể cần điều chỉnh lại kế hoạch hoặc bổ sung nguồn lực (nếu có thể).
* Giao tiếp với khách hàng: Thông báo cho khách hàng về tình hình chậm trễ, lý do, và các biện pháp khắc phục. Duy trì sự minh bạch và quản lý kỳ vọng của khách hàng.

Nguyên tắc quan trọng: Tìm nguyên nhân gốc rễ, cải thiện quy trình, ưu tiên chất lượng, và giao tiếp minh bạch.

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?

Vấn đề: Kiến trúc sư phần mềm (Software Architect) đề xuất thay đổi thiết kế ban đầu trong workflow thiết kế (Design Workflow) để cải thiện hiệu suất (Performance). Thay đổi thiết kế có thể có tác động lớn đến toàn bộ hệ thống.

Giải pháp đội phát triển nên xử lý:

* Đánh giá kỹ lưỡng đề xuất thay đổi:
  + Lợi ích: Đánh giá mức độ cải thiện hiệu suất mà thay đổi thiết kế mang lại. Đo lường hiệu suất hiện tại và dự kiến sau thay đổi.
  + Chi phí và rủi ro: Đánh giá chi phí và rủi ro khi thực hiện thay đổi thiết kế:
    - Công sức: Mức độ phức tạp và thời gian cần thiết để thực hiện thay đổi.
    - Tác động đến các thành phần khác: Thay đổi thiết kế có thể ảnh hưởng đến các thành phần khác của hệ thống. Cần đánh giá tác động lan tỏa.
    - Rủi ro về ổn định: Thay đổi lớn có thể gây ra rủi ro về ổn định và phát sinh lỗi mới.
  + Tính khả thi: Đánh giá tính khả thi về mặt kỹ thuật và nguồn lực để thực hiện thay đổi.
* Thảo luận và thống nhất với các bên liên quan:
  + Đội phát triển: Thảo luận với đội phát triển để hiểu rõ hơn về đề xuất, thu thập ý kiến phản hồi, và đảm bảo họ hiểu và đồng ý với thay đổi.
  + Khách hàng (nếu cần): Trong một số trường hợp, thay đổi thiết kế có thể ảnh hưởng đến chức năng hoặc chi phí. Cần thảo luận với khách hàng nếu có tác động đáng kể.
* Quyết định dựa trên phân tích: Dựa trên đánh giá lợi ích, chi phí, rủi ro, và ý kiến các bên liên quan, trưởng dự án và kiến trúc sư phần mềm đưa ra quyết định cuối cùng.
  + Chấp nhận thay đổi: Nếu lợi ích vượt trội chi phí và rủi ro chấp nhận được, và các bên liên quan đồng ý, thì tiến hành thay đổi thiết kế.
  + Từ chối thay đổi: Nếu chi phí và rủi ro quá cao, hoặc lợi ích không đáng kể, thì nên giữ nguyên thiết kế ban đầu.
  + Điều chỉnh thay đổi: Có thể điều chỉnh đề xuất ban đầu để giảm thiểu rủi ro hoặc chi phí, nhưng vẫn đạt được một phần lợi ích về hiệu suất.
* Cập nhật tài liệu thiết kế: Nếu thay đổi thiết kế được chấp nhận, cần cập nhật đầy đủ tài liệu thiết kế để phản ánh thay đổi mới nhất.
* Kiểm thử kỹ lưỡng: Sau khi thực hiện thay đổi thiết kế, cần kiểm thử kỹ lưỡng để đảm bảo thay đổi thực sự cải thiện hiệu suất và không gây ra lỗi mới.

Nguyên tắc quan trọng: Phân tích kỹ lưỡng, thảo luận, quyết định dựa trên dữ liệu, và cập nhật tài liệu.

5. Khách hàng yêu cầu rút ngắn thời gian phát triển dự án mà không thay đổi yêu cầu. Đội phát triển nên phản ứng ra sao?

Vấn đề: Khách hàng yêu cầu rút ngắn thời gian phát triển dự án (Shorten Project Time) mà không muốn giảm phạm vi yêu cầu (Scope). Đây là một yêu cầu thường gặp nhưng thường không thực tế, vì thời gian, phạm vi, và chi phí dự án thường có mối quan hệ ràng buộc.

Giải pháp đội phát triển nên phản ứng:

* Giải thích rõ ràng về nguyên tắc "Tam giác dự án" (Project Triangle): Giải thích cho khách hàng về mối quan hệ giữa ba yếu tố chính của dự án: Phạm vi (Scope), Thời gian (Time), và Chi phí (Cost). Thông thường, khi một yếu tố thay đổi, ít nhất một trong hai yếu tố còn lại cũng phải thay đổi để duy trì cân bằng. Trong trường hợp này, nếu khách hàng muốn giảm thời gian mà không giảm phạm vi, thì chi phí có thể sẽ phải tăng lên, hoặc chất lượng có thể bị ảnh hưởng.
* Đàm phán và tìm phương án khả thi:
  + Giảm phạm vi (Scope Reduction): Đề xuất giảm bớt một số tính năng ít quan trọng hơn (ưu tiên tính năng cốt lõi). Đây thường là cách hiệu quả nhất để rút ngắn thời gian mà vẫn đảm bảo chất lượng.
  + Tăng nguồn lực (Resource Increase): Nếu có thể, tăng thêm nhân sự (lập trình viên, kiểm thử viên) để tăng tốc độ phát triển. Tuy nhiên, việc tăng nguồn lực không phải lúc nào cũng hiệu quả và có thể làm tăng chi phí.
  + Làm việc ngoài giờ (Overtime): Yêu cầu đội phát triển làm thêm giờ. Tuy nhiên, cách này chỉ nên là giải pháp tạm thời, vì làm việc quá sức có thể dẫn đến mệt mỏi, giảm năng suất và chất lượng về lâu dài.
  + Tối ưu hóa quy trình phát triển: Xem xét các cách tối ưu hóa quy trình phát triển để tăng tốc độ, ví dụ như:
    - Sử dụng các công cụ phát triển nhanh (Rapid Application Development - RAD).
    - Áp dụng phương pháp phát triển lặp và tăng trưởng (Iterative and Incremental).
    - Tự động hóa các công việc lặp đi lặp lại (ví dụ, kiểm thử tự động).
* Đánh giá tác động và rủi ro: Nếu rút ngắn thời gian bằng cách nào đó, cần đánh giá tác động đến chất lượng sản phẩm và các rủi ro tiềm ẩn.
* Đặt kỳ vọng thực tế: Giúp khách hàng hiểu được rằng việc rút ngắn thời gian mà không thay đổi yêu cầu có thể không khả thi hoặc có thể ảnh hưởng đến các yếu tố khác (chất lượng, chi phí). Đặt ra kỳ vọng thực tế về những gì có thể đạt được trong thời gian ngắn hơn.
* Giao tiếp trung thực: Duy trì giao tiếp trung thực và minh bạch với khách hàng về những thách thức và giới hạn của việc rút ngắn thời gian.

Nguyên tắc quan trọng: Giáo dục khách hàng về tam giác dự án, đàm phán giải pháp, đánh giá rủi ro, và quản lý kỳ vọng.

6. Một công ty nhỏ muốn áp dụng mô hình CMM nhưng gặp khó khăn do thiếu nguồn lực. Hãy đề xuất giải pháp.

Vấn đề: Công ty nhỏ muốn áp dụng CMM (Capability Maturity Model) để cải thiện quy trình phát triển phần mềm, nhưng gặp khó khăn do hạn chế về nguồn lực (nhân lực, tài chính, thời gian).

Giải pháp đề xuất:

* Tiếp cận theo từng giai đoạn (Phased Approach): Không cần phải áp dụng toàn bộ CMM ngay lập tức. Bắt đầu từ từ và từng bước, tập trung vào các lĩnh vực quan trọng nhất trước.
  + Bắt đầu từ CMM mức 2: Mức 2 là bước khởi đầu quan trọng để thiết lập các quy trình cơ bản được quản lý ở mức dự án. Đây là mức có thể đạt được với nguồn lực hạn chế ban đầu.
  + Tập trung vào một vài quy trình chính: Chọn ra một vài quy trình quan trọng nhất cần cải thiện trước (ví dụ: quản lý yêu cầu, quản lý dự án, kiểm thử). Không cần phải triển khai tất cả các quy trình của CMM cùng một lúc.
* Ưu tiên hóa các mục tiêu cải tiến: Xác định các mục tiêu cải tiến quy trình quan trọng nhất đối với công ty dựa trên các vấn đề hiện tại và mục tiêu kinh doanh. Tập trung nguồn lực vào việc đạt được các mục tiêu này.
* Sử dụng các công cụ và tự động hóa (Automation & Tools): Sử dụng các công cụ phần mềm để hỗ trợ quy trình và tự động hóa các công việc lặp đi lặp lại. Điều này giúp giảm tải công việc thủ công và tiết kiệm nguồn lực. Ví dụ:
  + Công cụ quản lý dự án (Jira, Trello, Asana)
  + Công cụ quản lý yêu cầu (Requirement Management Tools)
  + Công cụ kiểm thử tự động (Selenium, JUnit)
  + Công cụ quản lý cấu hình (Git, SVN)
* Tận dụng nguồn lực bên ngoài (External Resources):
  + Tư vấn và đào tạo: Thuê chuyên gia tư vấn CMM hoặc tham gia các khóa đào tạo để được hướng dẫn và hỗ trợ áp dụng CMM một cách hiệu quả.
  + Sử dụng mẫu quy trình (Process Templates): Sử dụng các mẫu quy trình phát triển phần mềm đã được chuẩn hóa (ví dụ, mẫu quy trình Agile) để tiết kiệm thời gian và công sức xây dựng quy trình từ đầu.
* Đo lường và đánh giá thường xuyên: Theo dõi tiến độ cải tiến quy trình, đo lường hiệu quả của các biện pháp đã thực hiện, và điều chỉnh kế hoạch khi cần thiết. Bắt đầu với các chỉ số đo lường đơn giản và dễ thu thập.
* Xây dựng văn hóa cải tiến dần dần: Thay đổi văn hóa cần thời gian. Bắt đầu bằng việc tạo ra nhận thức về tầm quan trọng của quy trình và khuyến khích mọi người tham gia vào quá trình cải tiến.
* Linh hoạt và thích ứng: Áp dụng CMM một cách linh hoạt và thích ứng với đặc thù của công ty nhỏ. Không cần phải quá cứng nhắc tuân theo mọi quy định của CMM. Quan trọng là tập trung vào việc cải thiện quy trình một cách thực tế và hiệu quả.

Nguyên tắc quan trọng: Tiếp cận từng bước, ưu tiên, sử dụng công cụ, tận dụng nguồn lực bên ngoài, và linh hoạt.

7. Trong workflow lấy yêu cầu, khách hàng cung cấp thông tin không rõ ràng. Đội phát triển cần làm gì?

Vấn đề: Khách hàng cung cấp thông tin yêu cầu không rõ ràng (Ambiguous Requirements) trong Workflow lấy yêu cầu (Requirement Workflow) là một vấn đề phổ biến. Yêu cầu không rõ ràng có thể dẫn đến hiểu sai, phát triển sai chức năng, và phát sinh lỗi sau này.

Giải pháp đội phát triển cần làm:

* Chủ động đặt câu hỏi làm rõ (Clarification Questions): Không nên im lặng hoặc đoán mò. Chủ động đặt câu hỏi cụ thể, chi tiết để làm rõ những điểm chưa rõ ràng trong yêu cầu của khách hàng.
  + Câu hỏi mở: Khuyến khích khách hàng giải thích thêm, ví dụ: "Bạn có thể nói rõ hơn về...?", "Bạn có thể cho ví dụ về...?"
  + Câu hỏi đóng: Để xác nhận hiểu đúng, ví dụ: "Như vậy, ý bạn là... đúng không?", "Để tôi chắc chắn, bạn muốn... phải không?"
* Sử dụng kỹ thuật gợi ý yêu cầu (Requirement Elicitation Techniques):
  + Phỏng vấn (Interviews): Gặp gỡ trực tiếp hoặc trực tuyến với khách hàng để thảo luận và làm rõ yêu cầu.
  + Hội thảo (Workshops): Tổ chức các buổi hội thảo với sự tham gia của nhiều bên liên quan (khách hàng, người dùng cuối, đội phát triển) để cùng nhau thảo luận và làm rõ yêu cầu.
  + Khảo sát (Surveys): Sử dụng bảng hỏi để thu thập thông tin từ một số lượng lớn người dùng (nếu có).
  + Phân tích tài liệu hiện có (Document Analysis): Xem xét các tài liệu hiện có của khách hàng (ví dụ, tài liệu nghiệp vụ, quy trình làm việc) để hiểu rõ hơn về nhu cầu của họ.
  + Quan sát (Observation): Quan sát trực tiếp cách người dùng làm việc hiện tại để hiểu rõ hơn về quy trình nghiệp vụ và nhu cầu của họ.
* Xây dựng mẫu thử nghiệm/prototypes (Prototyping): Xây dựng nhanh các mẫu thử nghiệm hoặc giao diện người dùng sơ bộ (wireframes, mockups) để khách hàng có thể hình dung và đưa ra phản hồi cụ thể hơn về yêu cầu.
* Xác nhận lại yêu cầu bằng văn bản (Written Confirmation): Sau khi làm rõ yêu cầu, ghi lại yêu cầu đã được làm rõ bằng văn bản (ví dụ, tài liệu đặc tả yêu cầu) và gửi lại cho khách hàng để xác nhận lại. Đảm bảo có sự đồng thuận về yêu cầu đã được làm rõ.
* Lặp lại quy trình lấy yêu cầu (Iterate Requirement Elicitation): Quy trình lấy yêu cầu thường là một quá trình lặp đi lặp lại. Nếu yêu cầu vẫn chưa rõ ràng sau lần lấy yêu cầu đầu tiên, cần lặp lại quy trình nhiều lần, sử dụng các kỹ thuật khác nhau, cho đến khi yêu cầu được làm rõ đầy đủ.

Nguyên tắc quan trọng: Chủ động, đặt câu hỏi, sử dụng đa dạng kỹ thuật, xác nhận bằng văn bản, và lặp lại nếu cần.

8. Một dự án gặp rủi ro cao trong pha khởi đầu do thiếu tài liệu yêu cầu rõ ràng. Đội phát triển nên làm gì?

Vấn đề: Dự án gặp rủi ro cao (High Risk) trong pha khởi đầu (Inception Phase) do thiếu tài liệu yêu cầu rõ ràng (Lack of Clear Requirements). Thiếu yêu cầu rõ ràng ở giai đoạn đầu có thể dẫn đến đi sai hướng ngay từ đầu và gây ra nhiều vấn đề nghiêm trọng sau này.

Giải pháp đội phát triển nên làm:

* Ưu tiên cao nhất: Làm rõ yêu cầu. Nhiệm vụ quan trọng nhất trong pha khởi đầu lúc này là tập trung mọi nỗ lực để làm rõ yêu cầu càng sớm càng tốt.
* Tăng cường hoạt động lấy yêu cầu (Intensify Requirement Elicitation Activities): Áp dụng các kỹ thuật gợi ý yêu cầu đã nêu ở câu 7 (phỏng vấn, hội thảo, khảo sát, prototyping, v.v.) một cách tích cực và triệt để hơn. Dành nhiều thời gian và nguồn lực hơn cho hoạt động này trong pha khởi đầu.
* Làm việc chặt chẽ với khách hàng: Tăng cường giao tiếp và hợp tác chặt chẽ với khách hàng để thu thập thông tin và làm rõ yêu cầu. Tổ chức các buổi họp thường xuyên hơn, có thể là hàng ngày hoặc cách ngày, để thảo luận và làm rõ yêu cầu.
* Tạo ra tài liệu yêu cầu sơ bộ (Initial Requirements Documents): Mặc dù yêu cầu chưa rõ ràng hoàn toàn, nhưng vẫn cần cố gắng ghi lại những gì đã biết được về yêu cầu dưới dạng tài liệu sơ bộ. Tài liệu này sẽ là cơ sở để thảo luận và làm rõ thêm với khách hàng.
* Xây dựng mô hình nghiệp vụ (Business Modeling): Nếu yêu cầu liên quan đến quy trình nghiệp vụ, hãy xây dựng mô hình nghiệp vụ (ví dụ, sơ đồ luồng công việc, Use Case diagrams) để hiểu rõ hơn về quy trình và nhu cầu của khách hàng.
* Thực hiện nghiên cứu khả thi (Feasibility Study): Tiến hành nghiên cứu khả thi kỹ thuật và nghiệp vụ để đánh giá xem liệu dự án có khả thi với yêu cầu hiện tại hay không. Nếu yêu cầu quá mơ hồ và rủi ro quá cao, có thể cần phải xem xét lại phạm vi dự án hoặc thậm chí hủy bỏ dự án.
* Quản lý rủi ro chủ động (Proactive Risk Management): Vì rủi ro yêu cầu không rõ ràng đã được xác định là rủi ro cao nhất, cần lập kế hoạch quản lý rủi ro chủ động. Điều này bao gồm:
  + Theo dõi rủi ro: Theo dõi tiến độ làm rõ yêu cầu và đánh giá lại mức độ rủi ro thường xuyên.
  + Biện pháp giảm thiểu rủi ro: Tiếp tục áp dụng các biện pháp làm rõ yêu cầu, tăng cường giao tiếp với khách hàng.
  + Kế hoạch dự phòng: Chuẩn bị kế hoạch dự phòng nếu rủi ro vẫn không được kiểm soát (ví dụ, điều chỉnh phạm vi dự án, gia hạn tiến độ).
* Không vội vàng sang pha tiếp theo: Không nên vội vàng chuyển sang pha làm rõ (Elaboration Phase) nếu yêu cầu vẫn chưa rõ ràng. Cần kéo dài pha khởi đầu và tập trung vào làm rõ yêu cầu cho đến khi rủi ro được giảm xuống mức chấp nhận được.

Nguyên tắc quan trọng: Ưu tiên làm rõ yêu cầu, tăng cường giao tiếp, quản lý rủi ro chủ động, không vội vàng.

9. Dự án phần mềm lớn có nhiều nhóm phát triển ở các địa điểm khác nhau. Làm thế nào để đảm bảo các nhóm phối hợp hiệu quả?

Vấn đề: Dự án phần mềm lớn (Large Project) với nhiều nhóm phát triển (Distributed Teams) ở các địa điểm khác nhau (Geographically Distributed) gặp khó khăn trong việc phối hợp hiệu quả. Khoảng cách địa lý có thể gây ra các rào cản về giao tiếp, phối hợp, và chia sẻ thông tin.

Giải pháp để đảm bảo các nhóm phối hợp hiệu quả:

* Thiết lập quy trình giao tiếp rõ ràng và hiệu quả:
  + Xác định kênh giao tiếp chính: Chọn ra các kênh giao tiếp chính thức (ví dụ, email, chat, video conferencing) và quy định rõ ràng khi nào sử dụng kênh nào.
  + Lịch họp định kỳ: Tổ chức các cuộc họp định kỳ (ví dụ, hàng ngày, hàng tuần) giữa các nhóm để cập nhật tiến độ, thảo luận vấn đề, và phối hợp công việc. Sử dụng video conferencing để tăng cường tính tương tác.
  + Công cụ cộng tác trực tuyến (Collaboration Tools): Sử dụng các công cụ cộng tác trực tuyến như:
    - Project Management Tools: (Jira, Asana, Trello) để quản lý công việc, theo dõi tiến độ, và giao tiếp.
    - Document Sharing Platforms: (Google Drive, SharePoint, Confluence) để chia sẻ và quản lý tài liệu dự án.
    - Instant Messaging & Chat: (Slack, Microsoft Teams) để giao tiếp nhanh và trao đổi thông tin tức thời.
    - Video Conferencing: (Zoom, Google Meet, Microsoft Teams) cho các cuộc họp và thảo luận trực tuyến.
* Chuẩn hóa quy trình phát triển: Áp dụng quy trình phát triển phần mềm chuẩn hóa và thống nhất cho tất cả các nhóm. Điều này giúp đảm bảo các nhóm làm việc theo cùng một cách, dễ dàng phối hợp và tích hợp công việc.
  + Coding Standards: Thống nhất coding standards để đảm bảo chất lượng code và dễ dàng tích hợp code từ các nhóm khác nhau.
  + Branching Strategy (trong hệ thống quản lý phiên bản): Xác định rõ ràng chiến lược phân nhánh (branching strategy) trong hệ thống quản lý phiên bản (ví dụ, Git) để quản lý việc tích hợp code từ nhiều nhóm.
  + Quy trình kiểm thử và tích hợp: Thống nhất quy trình kiểm thử và tích hợp để đảm bảo chất lượng sản phẩm khi tích hợp các thành phần từ các nhóm khác nhau.
* Phân công trách nhiệm rõ ràng (Clear Roles and Responsibilities): Xác định rõ vai trò và trách nhiệm của từng nhóm và từng thành viên trong nhóm. Đảm bảo mỗi nhóm hiểu rõ phạm vi công việc và giao diện làm việc với các nhóm khác.
* Giao tiếp thường xuyên và chủ động (Proactive Communication): Khuyến khích giao tiếp thường xuyên và chủ động giữa các thành viên trong các nhóm khác nhau. Không chỉ giao tiếp trong các cuộc họp chính thức mà còn giao tiếp không chính thức (ví dụ, qua chat) để xây dựng mối quan hệ và giải quyết vấn đề nhanh chóng.
* Chia sẻ thông tin minh bạch (Transparent Information Sharing): Đảm bảo tất cả thông tin dự án quan trọng (yêu cầu, thiết kế, tiến độ, vấn đề, rủi ro) được chia sẻ minh bạch và dễ dàng truy cập cho tất cả các thành viên dự án.
* Xây dựng văn hóa hợp tác (Collaboration Culture): Khuyến khích văn hóa hợp tác, tin tưởng, và hỗ trợ lẫn nhau giữa các nhóm. Tổ chức các hoạt động team building trực tuyến hoặc trực tiếp (nếu có thể) để tăng cường tinh thần đồng đội.
* Lãnh đạo và điều phối hiệu quả (Effective Leadership and Coordination): Trưởng dự án và các trưởng nhóm đóng vai trò quan trọng trong việc điều phối công việc giữa các nhóm, giải quyết xung đột, và đảm bảo dự án đi đúng hướng.

Nguyên tắc quan trọng: Giao tiếp hiệu quả, quy trình chuẩn hóa, trách nhiệm rõ ràng, chia sẻ thông tin, văn hóa hợp tác, và lãnh đạo tốt.

10. Một công ty phát triển phần mềm gặp khó khăn trong việc quản lý quy trình do không có chuẩn hóa. Hãy đề xuất giải pháp.

Vấn đề: Công ty phát triển phần mềm gặp khó khăn trong việc quản lý quy trình (Process Management) do thiếu chuẩn hóa (Lack of Standardization). Quy trình không chuẩn hóa có thể dẫn đến: \* Không nhất quán: Các dự án khác nhau có thể sử dụng các quy trình khác nhau, gây khó khăn trong việc quản lý và so sánh hiệu suất. \* Khó dự đoán: Khó dự đoán thời gian và chi phí dự án do quy trình không ổn định. \* Khó kiểm soát chất lượng: Chất lượng sản phẩm không đồng đều do thiếu quy trình đảm bảo chất lượng. \* Khó mở rộng: Khó mở rộng quy mô hoạt động của công ty vì quy trình không được chuẩn hóa và khó lặp lại.

Giải pháp đề xuất:

* Xây dựng và chuẩn hóa quy trình phát triển phần mềm (Standardize Software Development Process): Đây là giải pháp cốt lõi. Cần bắt đầu xây dựng và chuẩn hóa các quy trình phát triển phần mềm quan trọng nhất, bao gồm:
  + Quy trình quản lý dự án: (lập kế hoạch, theo dõi tiến độ, quản lý rủi ro, quản lý thay đổi).
  + Quy trình lấy yêu cầu: (thu thập, phân tích, đặc tả, xác nhận yêu cầu).
  + Quy trình thiết kế: (thiết kế kiến trúc, thiết kế chi tiết).
  + Quy trình lập trình: (coding standards, code review).
  + Quy trình kiểm thử: (lập kế hoạch kiểm thử, thiết kế kiểm thử, thực hiện kiểm thử, báo cáo lỗi).
  + Quy trình triển khai và bảo trì.
* Tài liệu hóa quy trình (Document Processes): Viết ra và tài liệu hóa các quy trình đã được chuẩn hóa. Tài liệu quy trình cần rõ ràng, dễ hiểu, và dễ truy cập cho tất cả các thành viên công ty.
* Đào tạo và phổ biến quy trình (Training and Communication): Đào tạo cho tất cả các thành viên công ty về các quy trình đã được chuẩn hóa. Đảm bảo mọi người hiểu rõ và tuân thủ theo quy trình.
* Thực hiện từng bước và lặp đi lặp lại (Iterative and Incremental Implementation): Không cần phải chuẩn hóa tất cả các quy trình cùng một lúc. Bắt đầu với các quy trình quan trọng nhất trước. Sau đó, từng bước mở rộng phạm vi chuẩn hóa sang các quy trình khác. Liên tục cải tiến quy trình dựa trên phản hồi và kinh nghiệm thực tế.
* Sử dụng công cụ hỗ trợ quy trình (Process Support Tools): Sử dụng các công cụ phần mềm để hỗ trợ việc thực hiện và quản lý quy trình. Ví dụ:
  + Công cụ quản lý dự án (Jira, Asana, Trello).
  + Công cụ quản lý quy trình nghiệp vụ (Business Process Management - BPM tools).
  + Công cụ quản lý tài liệu (Document Management Systems).
* Chỉ định người chịu trách nhiệm (Assign Process Owners): Chỉ định người chịu trách nhiệm cho từng quy trình. Người này sẽ có trách nhiệm duy trì, cập nhật, và cải tiến quy trình.
* Đo lường và đánh giá hiệu quả quy trình (Process Measurement and Evaluation): Thiết lập các chỉ số đo lường hiệu quả quy trình (Process Performance Metrics) để theo dõi và đánh giá hiệu quả của quy trình đã được chuẩn hóa. Sử dụng dữ liệu đo lường để cải tiến quy trình liên tục.
* Khuyến khích văn hóa tuân thủ quy trình (Process Adherence Culture): Xây dựng văn hóa tuân thủ quy trình trong công ty. Điều này cần sự hỗ trợ từ lãnh đạo và sự tham gia của tất cả các thành viên.

Nguyên tắc quan trọng: Chuẩn hóa quy trình, tài liệu hóa, đào tạo, tiếp cận từng bước, sử dụng công cụ, đo lường và đánh giá, và xây dựng văn hóa tuân thủ.