## Trắc nghiệm

**Câu 1: B**. Workflow lấy yêu cầu

**Câu 2: B**. Pha làm rõ

**Câu 3: C**. Mức 4

**Câu 4**: **B**. Khởi đầu, làm rõ, xây dựng, chuyển giao

**Câu 5**: **D**. Workflow kiểm thử

**Câu 6: C**. Quy trình không ổn định, phụ thuộc vào cá nhân

**Câu 7**: **B.** Mô hình lặp và tăng trưởng

**Câu 8**: **A**. Quy trình được cải tiến liên tục

Câu 9: **C.** Thiết kế kiến trúc và chi tiết hệ thống

Câu 10: **B**. Capability Maturity Model

## Câu hỏi ngắn

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

Pha khởi đầu trong tiến trình thống nhất (Unified Process) tập trung vào việc phân tích rủi ro và xây dựng kiến trúc ban đầu. Mục tiêu chính là hiểu rõ các yêu cầu cơ bản và đánh giá tính khả thi của dự án.

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

Mục tiêu: Xác định và ghi nhận tất cả các yêu cầu từ phía khách

hàng.

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

- *Pha khởi đầu (Inception):*

Mục tiêu chính là hiểu rõ các yêu cầu cơ bản và đánh giá tính khả thi của dự án.

*-Pha làm rõ (Elaboration):*

Tập trung vào phân tích và thiết kế hệ thống, giải quyết các rủi ro lớn.

*-Pha xây dựng (Construction):*

Triển khai các module phần mềm và kiểm thử chúng.

*-Pha chuyển giao (Transition):*

Phần mềm được triển khai và bàn giao cho khách hàng sử dụng.

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

CMM mức 2:

▪ Quy trình được quản lý ở mức cơ bản.

▪ Các hoạt động như lập kế hoạch, quản lý rủi ro được thực hiện.

CMM mức 3:

▪ Quy trình được định nghĩa rõ ràng và nhất quán trong toàn tổ chức.

▪ Các tiêu chuẩn quy trình được xây dựng và áp dụng.

**5. Workflow kiểm thử có nhiệm vụ gì?**

Thực hiện kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống.

Sửa lỗi phát hiện trong quá trình kiểm thử.

Lập báo cáo kiểm thử.

**6. Mô hình CMM có bao nhiêu mức?**

- Mức 1 – Ban đầu
- Mức 2 – Lặp lại được (Quản lý)
- Mức 3 – Được xác định
- Mức 4 – Được quản lý định lượng
- Mức 5 – Tối ưu hóa

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

Trong mô hình thác nước, các giai đoạn phát triển phần mềm được thực hiện theo thứ tự tuyến tính và không cho phép quay lại giai đoạn trước khi bước hiện tại hoàn thành. Ngược lại, mô hình lặp (iterative) chia quá trình phát triển thành các vòng lặp nhỏ, trong đó mỗi vòng lặp cung cấp một phiên bản cải tiến của sản phẩm, cho phép phản hồi và điều chỉnh liên tục.

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

Đúng, tiến trình thống nhất (Unified Process) là một mô hình lặp – nó phát triển sản phẩm theo cách lặp đi lặp lại và tăng dần, giúp tích hợp phản hồi từ khách hàng vào mỗi vòng lặp để cải thiện sản phẩm.

**9. Mục đích của workflow thiết kế là gì?**

**Xác định kiến trúc hệ thống:** Định nghĩa cấu trúc tổng thể, bao gồm các thành phần và cách chúng tương tác.

**Chuẩn hóa quy trình phát triển:** Đảm bảo tất cả thành viên nhóm tuân theo một hướng thiết kế nhất quán.

**Giảm thiểu rủi ro:** Phát hiện và giải quyết vấn đề thiết kế trước khi triển khai.

**Tối ưu hiệu suất và bảo trì:** Tạo ra thiết kế dễ mở rộng, bảo trì và nâng cấp trong tương lai.

**Tăng cường khả năng tái sử dụng:** Xây dựng các mô-đun có thể sử dụng lại để giảm chi phí phát triển.

**10. CMM mức 5 tập trung vào điều gì?**

- **Cải tiến quy trình liên tục:** Luôn tìm cách tối ưu hóa hiệu suất và chất lượng phần mềm.
- **Đổi mới sáng tạo:** Ứng dụng công nghệ mới để nâng cao hiệu quả phát triển.
- **Dự đoán và phòng ngừa lỗi:** Xây dựng hệ thống phát hiện lỗi sớm, giảm thiểu rủi ro.
- **Tự động hóa quy trình:** Sử dụng công cụ CI/CD, kiểm thử tự động để tối ưu hóa sản xuất.
- **Học hỏi từ kinh nghiệm:** Phân tích dữ liệu từ các dự án trước để cải thiện hiệu suất.

## Câu hỏi thảo luận nhóm

1. Thảo luận về vai trò của từng workflow trong tiến trình phát triển phần mềm.

**Workflow Yêu Cầu (Requirements Workflow)**

**Vai trò:**
Workflow này chịu trách nhiệm thu thập, phân tích và xác định yêu cầu của khách hàng và người dùng cuối. Đây là bước đầu tiên quan trọng trong phát triển phần mềm vì yêu cầu chính là nền tảng để xác định mục tiêu và hướng đi của dự án.

- **Xác định các yêu cầu chức năng và phi chức năng:** Những yêu cầu này sẽ là cơ sở để thiết kế và triển khai phần mềm.
- **Hiểu biết rõ về nhu cầu người dùng:** Workflow này giúp nhóm phát triển hiểu được những gì người dùng cần và mong muốn từ phần mềm.
- **Giảm thiểu rủi ro:** Khi yêu cầu được hiểu rõ ngay từ đầu, việc thay đổi và điều chỉnh trong quá trình phát triển sẽ ít hơn.

**Workflow Phân Tích (Analysis Workflow)**

**Vai trò:**
Workflow phân tích là nơi các yêu cầu đã thu thập được được phân tích sâu hơn để hiểu rõ cách thức hoạt động của hệ thống. Quá trình này giúp chuyển đổi yêu cầu thành mô hình, giúp việc thiết kế và phát triển phần mềm trở nên dễ dàng hơn.

- **Xây dựng mô hình hệ thống:** Phân tích yêu cầu giúp nhóm phát triển xây dựng mô hình, chẳng hạn như mô hình dữ liệu, mô hình chức năng của hệ thống.
- **Xác định các rủi ro tiềm ẩn:** Quá trình phân tích giúp nhận diện các vấn đề có thể xảy ra trong quá trình phát triển và tìm cách giải quyết.

**Workflow Thiết Kế (Design Workflow)**

**Vai trò:**
Workflow thiết kế chuyển các yêu cầu và mô hình phân tích thành các kế hoạch chi tiết để phát triển phần mềm. Thiết kế giúp nhóm phát triển biết được cách thức thực hiện các yêu cầu và chức năng của phần mềm.

- **Thiết kế cấu trúc phần mềm:** Bao gồm việc thiết kế kiến trúc phần mềm, cấu trúc dữ liệu, giao diện người dùng, và các thành phần khác.
- **Tạo bản thiết kế chi tiết:** Các thiết kế này sẽ là cơ sở để lập trình viên xây dựng phần mềm.
- **Lựa chọn công nghệ và công cụ:** Thiết kế cũng bao gồm việc chọn công nghệ phù hợp để phát triển phần mềm, đảm bảo sự tương thích và hiệu quả.

**Workflow Lập Trình (Implementation Workflow)**

**Vai trò:**
Workflow lập trình là giai đoạn thực tế triển khai phần mềm theo thiết kế đã xác định. Lập trình viên viết mã nguồn, phát triển các module, chức năng và hệ thống phần mềm.

- **Chuyển thiết kế thành mã nguồn thực tế:** Đây là giai đoạn quan trọng nhất trong việc biến các thiết kế thành sản phẩm phần mềm thực sự.
- **Tạo các tính năng và chức năng cho phần mềm:** Lập trình viên phát triển các tính năng của phần mềm theo yêu cầu đã được phân tích và thiết kế.
- **Quản lý mã nguồn:** Đảm bảo mã nguồn được kiểm soát chặt chẽ, tránh sự xung đột và đảm bảo tính ổn định của phần mềm.

**Workflow Kiểm Thử (Testing Workflow)**

**Vai trò:**
Workflow kiểm thử kiểm tra phần mềm để đảm bảo nó hoạt động đúng như yêu cầu và không có lỗi. Đây là một bước quan trọng để đảm bảo chất lượng phần mềm.

- **Phát hiện và sửa lỗi:** Mục tiêu chính của kiểm thử là phát hiện các lỗi và sửa chữa chúng trước khi phần mềm được đưa vào sử dụng.
- **Đảm bảo yêu cầu được đáp ứng:** Kiểm thử giúp đảm bảo phần mềm thực hiện đúng các chức năng đã được yêu cầu trong giai đoạn phân tích.
- **Kiểm thử tính ổn định và hiệu suất:** Kiểm thử không chỉ kiểm tra tính chính xác mà còn đánh giá hiệu suất và khả năng chịu tải của phần mềm.

**Workflow Triển Khai (Deployment Workflow)**

**Vai trò:**
Workflow triển khai là giai đoạn đưa phần mềm vào môi trường thực tế để người dùng có thể sử dụng. Đây là bước cuối cùng trong quá trình phát triển phần mềm.

- **Cài đặt và cấu hình phần mềm:** Đảm bảo phần mềm hoạt động đúng trên hệ thống và thiết bị của người dùng.
- **Hỗ trợ người dùng:** Cung cấp tài liệu hướng dẫn và hỗ trợ kỹ thuật cho người dùng khi họ bắt đầu sử dụng phần mềm.
- **Cập nhật và bảo trì:** Sau khi triển khai, phần mềm có thể cần được bảo trì và cập nhật để khắc phục sự cố hoặc cải thiện tính năng.

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

Mô hình **vòng đời thác nước** và **tiến trình thống nhất** đều là các phương pháp phát triển phần mềm, nhưng chúng có cách tiếp cận và tổ chức công việc khác nhau.

**Mô hình vòng đời thác nước (Waterfall Model)**

**Khái niệm:**
Mô hình thác nước là một phương pháp phát triển phần mềm tuyến tính, trong đó các giai đoạn phát triển được thực hiện lần lượt theo một trình tự cố định, từ giai đoạn yêu cầu, thiết kế, lập trình, kiểm thử cho đến bảo trì. Mỗi giai đoạn chỉ bắt đầu khi giai đoạn trước đó hoàn tất.

**Ưu điểm:**

- **Dễ hiểu và quản lý:** Vì mỗi giai đoạn đều có mốc thời gian và mục tiêu rõ ràng, dễ dàng theo dõi tiến độ.
- **Rõ ràng về yêu cầu:** Yêu cầu được xác định rõ ràng từ đầu, giúp giảm thiểu sự thay đổi trong suốt quá trình phát triển.
- **Phù hợp với các dự án có yêu cầu cố định:** Mô hình này rất hữu ích cho những dự án có yêu cầu rõ ràng và không thay đổi nhiều.

**Nhược điểm:**

- **Khó thay đổi:** Khi một giai đoạn đã hoàn thành, việc quay lại và thay đổi là rất khó khăn và tốn kém.
- **Phát hiện lỗi muộn:** Lỗi chỉ được phát hiện sau khi giai đoạn kiểm thử hoàn tất, gây khó khăn trong việc sửa chữa.
- **Không linh hoạt:** Phương pháp này không dễ dàng thích nghi với những yêu cầu thay đổi trong quá trình phát triển.

**Tiến trình thống nhất (Unified Process)**

**Khái niệm:**
Tiến trình thống nhất (Unified Process, viết tắt là UP) là một phương pháp phát triển phần mềm theo hướng đối tượng, dựa trên các chu trình lặp và tăng trưởng. Phương pháp này chia quá trình phát triển phần mềm thành nhiều giai đoạn lặp đi lặp lại (iteration), mỗi giai đoạn có thể hoàn thiện một phần nhỏ của phần mềm.

**Ưu điểm:**

- **Linh hoạt cao:** Phương pháp này cho phép thay đổi yêu cầu trong suốt quá trình phát triển mà không gây gián đoạn lớn.

- **Giảm thiểu rủi ro:** Vì có các giai đoạn kiểm thử liên tục trong các vòng lặp, các lỗi có thể được phát hiện sớm hơn.
- **Tập trung vào người dùng:** Phương pháp chú trọng vào việc phát triển phần mềm có thể dễ dàng điều chỉnh theo nhu cầu của người dùng trong mỗi vòng lặp.

**Nhược điểm:**

- **Phức tạp hơn:** Do có nhiều vòng lặp và quá trình kiểm thử liên tục, quản lý dự án có thể trở nên phức tạp.
- **Cần nguồn lực lớn:** Để duy trì các vòng lặp và đảm bảo sự tương tác giữa các nhóm, đòi hỏi đội ngũ phát triển và kiểm thử có sự phối hợp chặt chẽ.
- **Yêu cầu tài liệu chi tiết:** Các dự án sử dụng UP cần có tài liệu rõ ràng về từng giai đoạn phát triển và thay đổi trong quá trình thực hiện.

**So sánh tổng quan**

- **Mô hình vòng đời thác nước** là tuyến tính và dễ quản lý khi yêu cầu rõ ràng ngay từ đầu. Tuy nhiên, nó thiếu tính linh hoạt và khả năng thích ứng với thay đổi trong quá trình phát triển.
- **Tiến trình thống nhất** linh hoạt hơn nhiều, cho phép thay đổi trong suốt quá trình phát triển và tập trung vào việc kiểm thử và phát triển phần mềm theo từng phần nhỏ. Tuy nhiên, nó có thể phức tạp hơn trong quản lý và đòi hỏi tài nguyên lớn hơn.

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

**Thảo luận về Ưu và Nhược điểm của Mô hình Lặp và Tăng Trưởng**

Mô hình **Lặp (Iterative Model)** và **Tăng Trưởng (Incremental Model)** là hai phương pháp phát triển phần mềm linh hoạt. Dưới đây là các ưu và nhược điểm của từng mô hình.

**Mô hình Lặp (Iterative Model)**

**Khái niệm:**
Mô hình lặp phát triển phần mềm qua các vòng lặp, mỗi vòng sẽ cải tiến và mở rộng phiên bản trước đó.

**Ưu điểm:**

· Phát hiện lỗi sớm: Phần mềm được kiểm thử liên tục qua các vòng lặp.

· Linh hoạt: Dễ dàng thay đổi yêu cầu nếu khách hàng có nhu cầu.

· Cải thiện dần chất lượng: Mỗi lần lặp giúp phần mềm trở nên hoàn thiện hơn.

· Giảm rủi ro: Các lỗi lớn có thể được phát hiện và khắc phục trước khi phần mềm hoàn chỉnh.

**Nhược điểm:**

· Chi phí cao: Cần tài nguyên cho nhiều vòng lặp phát triển.

· Quản lý phức tạp: Cần sự giám sát và phối hợp liên tục giữa các nhóm.

· Không phù hợp với dự án nhỏ: Mô hình này có thể tốn thời gian và nguồn lực hơn so với mô hình tuyến tính.

**Mô hình Tăng Trưởng (Incremental Model)**

**Khái niệm:**
Phần mềm được phát triển theo từng phần nhỏ, mỗi phần có thể hoạt động độc lập và được tích hợp vào sản phẩm hoàn chỉnh theo từng giai đoạn.

**Ưu điểm:**

· Cung cấp sản phẩm sớm: Có thể triển khai phần mềm ngay từ các bản đầu tiên.

· Dễ dàng mở rộng: Từng phần có thể được cải tiến mà không ảnh hưởng đến hệ thống chính.

· Dễ bảo trì: Phần mềm chia thành các module nhỏ, dễ quản lý.

· Giảm rủi ro: Các phần mềm còn lại vẫn có thể hoạt động nếu một phần gặp lỗi.

**Nhược điểm:**

· Khó tích hợp: Nếu không thiết kế cẩn thận, việc kết hợp các phần có thể gặp vấn đề.

· Yêu cầu chính xác từ ban đầu: Nếu thiết kế ban đầu sai, sẽ ảnh hưởng đến các phần sau.

· Quản lý phức tạp: Cần sự phối hợp tốt giữa các nhóm để đảm bảo các phần hoạt động cùng nhau.

**So sánh Mô hình Lặp và Tăng Trưởng**

- **Phát triển**: Mô hình lặp phát triển toàn bộ phần mềm qua các vòng lặp. Mô hình tăng trưởng phát triển theo từng phần nhỏ có thể hoạt động độc lập.
- **Sản phẩm ban đầu**: Mô hình lặp có thể cho ra một nguyên mẫu chưa hoàn chỉnh, trong khi mô hình tăng trưởng cho ra các phần có thể sử dụng được ngay.
- **Linh hoạt**: Mô hình lặp có tính linh hoạt cao, dễ thay đổi yêu cầu trong mỗi vòng lặp. Mô hình tăng trưởng linh hoạt hơn khi yêu cầu ban đầu rõ ràng.
- **Rủi ro**: Mô hình lặp có rủi ro thấp hơn do kiểm thử thường xuyên, trong khi mô hình tăng trưởng có thể gặp vấn đề khi tích hợp các phần.
- **Thời gian ra thị trường**: Mô hình lặp mất nhiều thời gian hơn do cần nhiều vòng lặp, trong khi mô hình tăng trưởng có thể phát hành sản phẩm sớm, từng phần một.
- **Chi phí**: Mô hình lặp tốn chi phí cao hơn do cần kiểm thử và cải tiến nhiều lần, trong khi mô hình tăng trưởng có thể tiết kiệm chi phí nếu yêu cầu ban đầu rõ ràng.

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

**Cung cấp khung đánh giá chuẩn hóa**: CMM đưa ra cách tiếp cận có hệ thống để đánh giá mức độ trưởng thành của quy trình phát triển phần mềm, cho phép các tổ chức tự đánh giá và so sánh với các tiêu chuẩn ngành.

**Lộ trình cải tiến rõ ràng**: Mô hình định nghĩa năm mức trưởng thành (từ Ban đầu đến Tối ưu hóa), tạo lộ trình phát triển dễ hiểu với các mục tiêu cụ thể cho mỗi cấp độ.

**Giảm rủi ro dự án**: Thông qua việc chuẩn hóa quy trình, CMM giúp giảm thiểu rủi ro thất bại dự án, cải thiện khả năng dự đoán kết quả và đảm bảo chất lượng.

**Đáp ứng yêu cầu hợp đồng**: Nhiều hợp đồng, đặc biệt từ các cơ quan chính phủ, yêu cầu chứng nhận CMM nhất định, khiến việc áp dụng trở thành yêu cầu kinh doanh.

**Tăng cường khả năng cạnh tranh**: Chứng nhận CMM cao trở thành lợi thế cạnh tranh, chứng minh cam kết về chất lượng và quy trình của tổ chức.

**Tập trung vào quy trình thay vì sản phẩm**: CMM nhấn mạnh việc cải thiện quy trình, dẫn đến chất lượng sản phẩm tốt hơn một cách tự nhiên và bền vững.

**Khả năng áp dụng rộng rãi**: Mặc dù bắt đầu từ phát triển phần mềm, CMM (và phiên bản cải tiến CMMI) có thể áp dụng cho nhiều lĩnh vực khác, từ quản lý dịch vụ đến quản lý dự án.

**Khuyến khích học tập liên tục**: Các cấp độ cao hơn của CMM yêu cầu tổ chức tập trung vào cải tiến liên tục, thúc đẩy văn hóa học hỏi và phát triển.

5. Thảo luận về các khó khăn khi áp dụng mô hình CMM trong thực tế.

Mô hình CMM (Capability Maturity Model – Mô hình trưởng thành năng lực) là một khung quản lý được thiết kế để giúp các tổ chức cải thiện quy trình phát triển phần mềm và quản lý dự án. Tuy nhiên, việc áp dụng mô hình này trong thực tế thường gặp phải nhiều khó khăn. Một số thách thức chính:

**Bất đồng từ nhân viên và văn hóa tổ chức**: CMM yêu cầu thay đổi cách làm việc, từ quy trình đến tư duy. Nhân viên có thể phản đối vì họ quen với cách làm việc cũ hoặc cảm thấy bị áp đặt thêm công việc. Nếu văn hóa tổ chức không khuyến khích cải tiến liên tục, việc triển khai CMM sẽ rất khó khăn.

**Chi phí và nguồn lực lớn**: Việc áp dụng CMM đòi hỏi đầu tư đáng kể vào đào tạo, tư vấn, và thời gian để đánh giá, điều chỉnh quy trình. Với các tổ chức nhỏ hoặc thiếu ngân sách, đây có thể là rào cản lớn.

**Thời gian triển khai dài**: CMM không phải là giải pháp tức thời. Để tiến từ cấp độ thấp lên cấp độ cao (ví dụ, từ cấp 1 - Initial lên cấp 5 - Optimizing), cần nhiều năm cải tiến liên tục. Điều này có thể làm nản lòng các tổ chức muốn thấy kết quả nhanh chóng.

**Khó khăn trong việc đo lường hiệu quả**: CMM tập trung vào cải thiện quy trình, nhưng việc định lượng lợi ích cụ thể (như tăng năng suất hay giảm lỗi) không phải lúc nào cũng dễ dàng. Điều này khiến một số nhà quản lý nghi ngờ về giá trị thực tế của mô hình.

**Áp dụng không linh hoạt**: CMM được thiết kế với cấu trúc khá cứng nhắc, thường phù hợp hơn với các tổ chức lớn hoặc dự án phức tạp. Các công ty nhỏ hoặc các nhóm phát triển nhanh (như trong mô hình Agile) có thể thấy nó không phù hợp với phong cách làm việc của họ.

**Thiếu sự hỗ trợ từ lãnh đạo**: Thành công của CMM phụ thuộc nhiều vào sự cam kết từ cấp quản lý cao nhất. Nếu lãnh đạo không hiểu rõ hoặc không ưu tiên mô hình này, việc triển khai sẽ dễ dàng thất bại.

6. Đề xuất các giải pháp để cải tiến quy trình phát triển phần mềm.

Mô hình CMM (Capability Maturity Model) là một khung quản lý chất lượng phần mềm phổ biến, giúp tổ chức nâng cao năng lực phát triển phần mềm. Tuy nhiên, việc áp dụng mô hình này trong thực tế gặp nhiều khó khăn:

- **Chi phí cao**: Việc triển khai CMM đòi hỏi nguồn lực tài chính đáng kể để đào tạo nhân viên, thuê chuyên gia đánh giá và cải tiến quy trình.
- **Thay đổi văn hóa tổ chức**: CMM yêu cầu doanh nghiệp phải thay đổi cách thức làm việc, điều này có thể gặp phải sự phản đối từ nhân viên.
- **Mất nhiều thời gian**: Quá trình nâng cấp mức độ của CMM không thể diễn ra ngay lập tức mà cần có lộ trình dài hạn.
- **Thiếu nhân sự có chuyên môn**: CMM đòi hỏi đội ngũ nhân viên có hiểu biết sâu về mô hình, nhưng nhiều tổ chức lại thiếu nhân sự có kinh nghiệm.
- **Khó khăn trong đo lường hiệu quả**: Việc đánh giá tác động của CMM đối với hiệu suất phát triển phần mềm không phải lúc nào cũng rõ ràng.

7. Phân tích ưu điểm của việc áp dụng tiến trình thống nhất trong các dự án lớn.

- **Tự động hóa quy trình**: Sử dụng các công cụ DevOps và CI/CD để giảm thiểu lỗi thủ công, cải thiện tốc độ phát triển.
- **Áp dụng Agile và Scrum**: Linh hoạt trong phát triển phần mềm giúp tăng khả năng thích ứng với thay đổi của thị trường.
- **Đào tạo nhân viên**: Tăng cường đào tạo nội bộ để nâng cao kỹ năng cho đội ngũ phát triển phần mềm.
- **Tăng cường kiểm thử phần mềm**: Áp dụng kiểm thử tự động, kiểm thử liên tục để đảm bảo chất lượng sản phẩm.
- **Cải thiện tài liệu hóa**: Xây dựng hệ thống tài liệu chi tiết để hỗ trợ bảo trì và mở rộng phần mềm trong tương lai.

8. Thảo luận về sự cần thiết của việc kiểm thử trong từng pha của tiến trình thống nhất.

Tiến trình thống nhất (Unified Process - UP) là một phương pháp phát triển phần mềm có cấu trúc rõ ràng, phù hợp với các dự án lớn. Những ưu điểm chính bao gồm:

- **Quản lý tốt rủi ro**: UP chia dự án thành các giai đoạn rõ ràng như Khởi tạo, Lập kế hoạch, Xây dựng, Chuyển giao, giúp kiểm soát rủi ro hiệu quả.
- **Cải thiện chất lượng phần mềm**: Việc thực hiện kiểm thử và đánh giá liên tục giúp phát hiện lỗi sớm, giảm chi phí sửa lỗi.
- **Tăng cường sự cộng tác**: UP tạo điều kiện để các bên liên quan có thể phối hợp chặt chẽ, đảm bảo sản phẩm cuối cùng đáp ứng yêu cầu khách hàng.
- **Tái sử dụng mã nguồn**: UP khuyến khích việc xây dựng các thành phần phần mềm có thể tái sử dụng, giúp tiết kiệm thời gian và nguồn lực.

- **Tính linh hoạt cao**: Mặc dù UP có cấu trúc chặt chẽ, nhưng vẫn có khả năng điều chỉnh phù hợp với đặc thù từng dự án.

9. So sánh giữa mô hình CMM mức 4 và mức 5.

| Tiêu chí | Mức 4 (Quản lý định lượng - Managed) | Mức 5 (Tối ưu hóa - Optimizing) |
|---|---|---|
| Mục tiêu | Đo lường và kiểm soát chất lượng dự án bằng dữ liệu định lượng. | Liên tục cải tiến quy trình dựa trên phân tích dữ liệu và phản hồi. |
| Cách quản lý | Dựa trên số liệu thống kê để kiểm soát hiệu suất và dự đoán kết quả. | Cải tiến quy trình liên tục, tập trung vào đổi mới và phòng ngừa sai sót. |
| Công cụ | Phân tích dữ liệu, đo lường hiệu suất, kiểm soát quy trình. | Công nghệ tiên tiến, quản lý rủi ro, điều chỉnh quy trình linh hoạt. |
| Đặc điểm | Có chuẩn hóa quy trình nhưng vẫn có thể có lỗi do chưa tối ưu hoàn toàn. | Loại bỏ lỗi ngay từ đầu, hướng đến hiệu suất cao nhất. |
| Ứng dụng thực tế | Các tổ chức có quy trình phần mềm chặt chẽ nhưng chưa tập trung cao vào cải tiến liên tục. | Các tổ chức tiên tiến luôn đổi mới, cải tiến quy trình liên tục để đạt hiệu suất tối ưu. |

10.Đề xuất cách tổ chức hoạt động nhóm trong workflow lấy yêu cầu

**Xác định vai trò trong nhóm**

- **BA (Business Analyst)**: Thu thập, phân tích yêu cầu từ khách hàng.
- **PM (Project Manager)**: Điều phối hoạt động nhóm, đảm bảo tiến độ.
- **Dev Lead**: Đánh giá yêu cầu, đảm bảo khả thi về mặt kỹ thuật.
- **Tester**: Xác nhận yêu cầu đầy đủ, tránh sai sót.

**Quy trình hoạt động**

- **Giai đoạn 1: Tiếp nhận yêu cầu**
    - Tổ chức họp với khách hàng để lấy yêu cầu sơ bộ.

    - Ghi nhận tất cả mong muốn của khách hàng (có thể dùng User Story).
- **Giai đoạn 2: Phân tích và làm rõ yêu cầu**
    - Xác định phạm vi dự án, chức năng chính.
    - Sử dụng mô hình như Use Case, Wireframe để mô tả.
    - Họp nội bộ nhóm để thống nhất yêu cầu.
- **Giai đoạn 3: Xác nhận yêu cầu với khách hàng**
    - Gửi tài liệu mô tả yêu cầu (SRS - Software Requirement Specification).
    - Lấy phản hồi và điều chỉnh nếu cần.
- **Giai đoạn 4: Chính thức hóa yêu cầu**
    - Khách hàng ký xác nhận.
    - Lưu trữ yêu cầu trong hệ thống quản lý (JIRA, Confluence, v.v.).

**Công cụ hỗ trợ**

- Google Docs, Notion, Confluence (Quản lý tài liệu).
- JIRA, Trello (Quản lý nhiệm vụ).
- Miro, Figma (Vẽ wireframe, mô hình hóa yêu cầu).

**Nguyên tắc làm việc nhóm**

- **Giao tiếp rõ ràng**: Luôn xác nhận lại yêu cầu.
- **Làm việc minh bạch**: Ghi nhận tất cả thay đổi trong yêu cầu.
- **Tương tác liên tục**: Họp ngắn (daily standup) để cập nhật tiến độ.

## 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?

**Áp dụng phương pháp phát triển phần mềm linh hoạt (Agile)**

Phương pháp Agile giúp đội ngũ phát triển thích nghi nhanh chóng với sự thay đổi yêu cầu từ khách hàng. Các nguyên lý của Agile, như phát triển theo các vòng lặp ngắn (sprint), giao tiếp thường xuyên với khách hàng và phản hồi liên tục, giúp dễ dàng điều chỉnh các yêu cầu khi cần thiết.

**Cách làm:**

- **Tổ chức các cuộc họp sprint:** Các cuộc họp ngắn giúp đội ngũ phát triển cập nhật tiến độ và trao đổi với khách hàng về bất kỳ thay đổi nào trong yêu cầu.

- **Phản hồi nhanh:** Đảm bảo rằng khách hàng có thể đưa ra phản hồi và các thay đổi nhanh chóng được áp dụng vào phần mềm.

**Đảm bảo tài liệu yêu cầu chi tiết và rõ ràng**

Để giảm thiểu sự thay đổi không cần thiết, nhóm phát triển nên làm việc chặt chẽ với khách hàng từ giai đoạn yêu cầu ban đầu để xác định rõ mục tiêu và chức năng của phần mềm. Nếu có sự thay đổi yêu cầu, cần đảm bảo rằng các thay đổi này được ghi nhận đầy đủ và rõ ràng trong tài liệu yêu cầu.

**Cách làm:**

- **Định nghĩa rõ ràng các yêu cầu:** Khi có yêu cầu thay đổi, đội ngũ cần làm việc trực tiếp với khách hàng để xác nhận lại mục tiêu và ưu tiên của thay đổi đó.
- **Ghi nhận yêu cầu thay đổi:** Mỗi yêu cầu thay đổi cần được ghi nhận chi tiết và rõ ràng, tránh nhầm lẫn và đảm bảo rằng nhóm phát triển hiểu rõ yêu cầu.

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?

Khi khách hàng yêu cầu bổ sung thêm tính năng mới trong pha chuyển giao của tiến trình thống nhất, đội phát triển cần xử lý cẩn thận để đảm bảo rằng yêu cầu này được tích hợp mà không ảnh hưởng đến chất lượng hoặc tiến độ của dự án.

**Đánh giá yêu cầu thay đổi**

Trước tiên, đội phát triển cần hiểu rõ yêu cầu mới của khách hàng, bao gồm mục tiêu, phạm vi và tác động của tính năng mới đối với phần mềm hiện tại.

- **Xác định yêu cầu chi tiết:** Cần làm việc trực tiếp với khách hàng để làm rõ yêu cầu mới, đảm bảo rằng đội ngũ hiểu đúng những gì khách hàng muốn.
- **Đánh giá tính khả thi:** Đội ngũ phát triển cần đánh giá tính khả thi của việc tích hợp tính năng mới vào hệ thống hiện tại, bao gồm thời gian, chi phí và các tài nguyên cần thiết.

**Đánh giá tác động đến tiến độ và ngân sách**

Việc bổ sung tính năng mới có thể ảnh hưởng đến tiến độ và chi phí của dự án, đặc biệt là khi đã ở pha chuyển giao.

- **Xác định tác động đến kế hoạch:** Đánh giá xem yêu cầu bổ sung này có thể làm trì hoãn tiến độ hay cần bổ sung thêm thời gian kiểm thử và triển khai.
- **Ước tính chi phí:** Tính toán chi phí bổ sung cho việc phát triển và kiểm thử tính năng mới, bao gồm tài nguyên và công sức.

**Thảo luận với khách hàng về các ưu tiên**

Đội ngũ phát triển cần thảo luận với khách hàng về mức độ ưu tiên của tính năng mới và quyết định liệu có thể trì hoãn yêu cầu này đến giai đoạn bảo trì hoặc phiên bản sau hay không.

- **Làm rõ mức độ quan trọng:** Nếu tính năng mới là yêu cầu bắt buộc và không thể trì hoãn, đội ngũ sẽ cần tập trung vào việc phát triển tính năng đó ngay lập tức.
- **Thỏa thuận với khách hàng:** Trong trường hợp tính năng không quá cấp bách, đội ngũ có thể đề xuất trì hoãn để phát triển nó trong các phiên bản tương lai, tránh ảnh hưởng đến kế hoạch chuyển giao.

**Lập kế hoạch thay đổi**

Nếu quyết định tích hợp tính năng mới ngay trong pha chuyển giao, đội ngũ phát triển cần lập một kế hoạch thay đổi chi tiết, xác định các bước cần thực hiện và đảm bảo rằng tính năng mới sẽ được phát triển và kiểm thử đúng tiến độ.

- **Cập nhật tài liệu và kế hoạch dự án:** Các thay đổi trong kế hoạch dự án và tài liệu kỹ thuật phải được thực hiện để đảm bảo tính minh bạch và dễ theo dõi.
- **Xác định rõ các giai đoạn phát triển:** Chia nhỏ việc phát triển tính năng mới thành các phần nhỏ để dễ dàng kiểm soát và theo dõi tiến độ.

**Kiểm thử và đảm bảo chất lượng**

Việc kiểm thử trở nên quan trọng khi có yêu cầu bổ sung tính năng mới. Đội ngũ phát triển cần đảm bảo rằng tính năng mới được tích hợp một cách mượt mà và không gây lỗi hệ thống.

- **Thực hiện kiểm thử tích hợp:** Kiểm tra tính tương thích của tính năng mới với các chức năng hiện có trong hệ thống.
- **Kiểm thử hồi quy:** Đảm bảo rằng tính năng mới không ảnh hưởng đến các tính năng đã triển khai trước đó.

**Cập nhật và triển khai**

Sau khi tính năng mới được phát triển và kiểm thử, đội ngũ phát triển sẽ triển khai tính năng vào hệ thống chính thức.

- **Cập nhật hệ thống:** Đảm bảo rằng tất cả các thay đổi được triển khai đầy đủ vào hệ thống mà không làm gián đoạn dịch vụ.
- **Cung cấp hướng dẫn sử dụng:** Nếu cần thiết, đội ngũ sẽ cung cấp tài liệu hoặc hướng dẫn bổ sung cho người dùng để họ có thể sử dụng tính năng mới.

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?

**Đánh giá và phân tích nguyên nhân lỗi**

Trước tiên, bạn cần xác định nguyên nhân gây ra lỗi trong quá trình kiểm thử. Điều này giúp tìm ra giải pháp phù hợp và hiệu quả.

- **Phân tích lỗi:** Kiểm tra và phân loại các lỗi phát sinh trong quá trình kiểm thử. Xác định xem lỗi là do mã nguồn, thiếu sót trong yêu cầu, hay do các vấn đề liên quan đến môi trường kiểm thử.
- **Đánh giá mức độ nghiêm trọng của lỗi:** Xem xét các lỗi là vấn đề nhỏ có thể sửa ngay, hay là các lỗi lớn ảnh hưởng đến nhiều phần của hệ thống.
- **Kiểm tra quy trình kiểm thử:** Đảm bảo quy trình kiểm thử được thực hiện đúng và có đủ các kịch bản kiểm thử để kiểm tra các tình huống khác nhau.

**Điều chỉnh lại kế hoạch dự án**

Sau khi hiểu rõ nguyên nhân gây ra lỗi và mức độ nghiêm trọng của chúng, bạn cần điều chỉnh kế hoạch dự án để có thể xử lý tình huống này mà không gây thêm áp lực lên đội ngũ phát triển.

- **Xác định lại ưu tiên:** Nếu có lỗi nghiêm trọng hoặc ảnh hưởng đến các chức năng quan trọng của hệ thống, bạn cần ưu tiên sửa chữa những lỗi này trước. Các tính năng không quan trọng có thể được trì hoãn.
- **Điều chỉnh tiến độ:** Cập nhật lại tiến độ dự án với đội ngũ và khách hàng, và xác định lại các mốc thời gian dựa trên việc sửa lỗi.
- **Tăng cường kiểm thử:** Đảm bảo rằng các vòng kiểm thử tiếp theo sẽ tập trung vào các yếu tố dễ bị bỏ sót và có thể gây ảnh hưởng lớn đến hệ thống.

**Tăng cường sự phối hợp giữa các nhóm**

Để giải quyết lỗi phát sinh nhanh chóng, cần tăng cường sự phối hợp giữa các nhóm phát triển, kiểm thử và quản lý dự án.

- **Tăng cường giao tiếp:** Đảm bảo sự giao tiếp liên tục giữa các nhóm phát triển và kiểm thử để nhanh chóng nhận diện và giải quyết lỗi.
- **Phối hợp chặt chẽ:** Các nhà phát triển cần tham gia sâu vào quá trình kiểm thử, hỗ trợ nhóm kiểm thử tìm ra nguyên nhân và sửa lỗi kịp thời.

**Tạo thêm các vòng kiểm thử và cải thiện tự động hóa**

Nếu lỗi phát sinh nhiều do việc kiểm thử không đủ hiệu quả, bạn có thể áp dụng các biện pháp cải thiện quy trình kiểm thử.

- **Kiểm thử tự động:** Nếu chưa có, triển khai kiểm thử tự động cho các tính năng cốt lõi của hệ thống. Điều này sẽ giúp phát hiện lỗi sớm và tiết kiệm thời gian kiểm thử.
- **Tăng cường kiểm thử hồi quy:** Đảm bảo rằng các tính năng đã phát triển trước đây không bị ảnh hưởng bởi các thay đổi mới và lỗi được phát hiện sớm.
- **Thực hiện kiểm thử theo mô hình Risk-Based Testing:** Tập trung vào các khu vực có nguy cơ cao nhất và có ảnh hưởng lớn đến hệ thống.

**Đánh giá và tái cấu trúc mã nguồn (nếu cần)**

Nếu các lỗi là do mã nguồn không ổn định hoặc thiếu tính module hóa, bạn có thể cần phải xem xét lại cấu trúc mã nguồn của dự án.

- **Rà soát mã nguồn:** Kiểm tra xem có phần nào trong mã nguồn cần được tái cấu trúc hoặc tối ưu lại để tránh phát sinh lỗi.
- **Thực hiện các bản sửa lỗi:** Đảm bảo rằng các lỗi phát sinh từ các vấn đề trong mã nguồn được sửa chữa kịp thời và không ảnh hưởng đến các phần khác của phần mềm.

**Tăng cường quản lý rủi ro**

Quản lý rủi ro là yếu tố quan trọng trong việc kiểm soát các sự cố phát sinh trong dự án.

- **Đánh giá lại các rủi ro:** Dựa trên các lỗi phát sinh, bạn cần cập nhật lại danh sách rủi ro và các biện pháp phòng ngừa.
- **Chia sẻ rủi ro với khách hàng:** Nếu cần, hãy thảo luận với khách hàng về các thay đổi trong tiến độ và cung cấp thông tin về các biện pháp bạn đang thực hiện để giải quyết các vấn đề.

**Cập nhật báo cáo và giao tiếp với khách hàng**

Sau khi thực hiện các bước trên, cần thông báo rõ ràng với khách hàng về tình hình tiến độ và các giải pháp bạn đang thực hiện.

- **Thông báo tiến độ:** Cập nhật thông tin tiến độ với khách hàng, cho biết nguyên nhân gây ra trễ tiến độ và các bước sửa chữa.
- **Đảm bảo sự hài lòng của khách hàng:** Đảm bảo khách hàng hiểu và đồng ý với kế hoạch điều chỉnh và tiếp tục duy trì sự liên lạc thường xuyên.

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?

Dưới đây là các bước mà đội phát triển nên thực hiện:

**1. Hiểu rõ lý do và mục tiêu của thay đổi:**

Đội phát triển nên yêu cầu kiến trúc sư giải thích chi tiết lý do cần thay đổi (ví dụ: điểm nghẽn hiệu suất, khả năng mở rộng, hay yêu cầu mới từ khách hàng). Hiểu rõ vấn đề sẽ giúp đội đánh giá xem thay đổi có thực sự cần thiết hay không.

**2. Đánh giá tác động của thay đổi:**

· Phân tích mức độ ảnh hưởng đến các thành phần hiện tại

· Xác định những module/thành phần nào sẽ cần được sửa đổi

· Đánh giá tác động đến tiến độ, nguồn lực và ngân sách dự án

**3. Thảo luận và thống nhất trong đội:**

Tổ chức một buổi họp với các thành viên chủ chốt (kiến trúc sư, nhà phát triển, quản lý dự án) để thảo luận về đề xuất. Đội nên đặt câu hỏi:

· Lợi ích của cải thiện hiệu suất có vượt trội hơn chi phí và rủi ro không?

· Có giải pháp thay thế nào (như tối ưu hóa mã hiện tại) mà ít xáo trộn hơn không?

· Mục tiêu là đạt được sự đồng thuận hoặc ít nhất là hiểu rõ quan điểm của mỗi bên.

**4. Thử nghiệm và kiểm chứng:**

Nếu khả thi, đội có thể triển khai một prototype hoặc proof-of-concept (PoC) để kiểm tra thiết kế mới trong môi trường thử nghiệm. Điều này giúp đo lường hiệu suất thực tế và phát hiện các vấn đề tiềm ẩn trước khi áp dụng toàn diện.

**5. Thực hiện và theo dõi:**

Nếu thay đổi được phê duyệt, đội nên thực hiện từng bước, sử dụng kiểm soát phiên bản (như Git) để theo dõi và dễ dàng quay lại nếu cần. Sau khi triển khai, hiệu suất cần được đo lường lại để xác nhận cải thiện.

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?

**1. Hiểu rõ yêu cầu và lý do của khách hàng**

Bước hành động: Tổ chức một cuộc họp với khách hàng để thảo luận chi tiết về yêu cầu rút ngắn thời gian. Hỏi khách hàng về lý do cụ thể đằng sau yêu cầu này (ví dụ: thay đổi trong chiến lược kinh doanh, áp lực từ thị trường, hoặc các yếu tố bên ngoài khác).

**2. Đánh giá lại kế hoạch dự án hiện tại**

Bước hành động: Xem xét kỹ lưỡng tiến độ dự án, các công việc quan trọng (critical path), và các tài nguyên hiện có. Xác định xem có thể tối ưu hóa quy trình, thực hiện các công việc song song, hoặc áp dụng các phương pháp phát triển linh hoạt (như Agile) để tăng tốc độ mà không làm giảm chất lượng.

**3. Đánh giá khả năng rút ngắn thời gian**

Bước hành động: Đánh giá xem việc rút ngắn thời gian có khả thi hay không bằng cách xem xét các ràng buộc về tài nguyên, công nghệ, và nhân lực. Nếu không thể rút ngắn thời gian mà không ảnh hưởng đến chất lượng, đội cần chuẩn bị các phương án thay thế.

**4. Thảo luận với khách hàng về các phương án**

Bước hành động: Tổ chức một cuộc họp với khách hàng để trình bày các phương án khả thi:

Phương án 1: Rút ngắn thời gian bằng cách tăng tài nguyên (ví dụ: thêm nhân sự, làm việc ngoài giờ), nhưng điều này có thể làm tăng chi phí.

Phương án 2: Điều chỉnh phạm vi dự án bằng cách loại bỏ hoặc hoãn các tính năng không quan trọng để tập trung vào các yêu cầu cốt lõi.

Phương án 3: Chấp nhận rủi ro về chất lượng hoặc các vấn đề tiềm ẩn nếu buộc phải rút ngắn thời gian mà không thay đổi yêu cầu hoặc tài nguyên.

**5. Đưa ra kế hoạch mới và đảm bảo sự đồng thuận**

Bước hành động: Sau khi thảo luận, đội phát triển cần lập một kế hoạch mới với lịch trình cập nhật, bao gồm các điều chỉnh về tài nguyên, phạm vi, hoặc rủi ro. Đảm bảo rằng khách hàng đồng ý với kế hoạch này bằng cách ghi nhận sự đồng thuận bằng văn bản.

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

- **Đánh giá hiện trạng và ưu tiên quy trình trọng tâm:**
  Bắt đầu bằng việc đánh giá mức độ hiện tại của các quy trình nội bộ để xác định những khâu then chốt cần cải tiến. Tập trung cải tiến ở những quy trình mang lại giá trị lớn cho doanh nghiệp, thay vì cố gắng triển khai toàn bộ mô hình ngay lập tức.
- **Triển khai cải tiến theo từng bước (incremental):**
  Thay vì áp dụng đồng loát toàn bộ mô hình CMM, công ty có thể lựa chọn triển khai theo từng giai đoạn. Bắt đầu với các yếu tố cơ bản (ví dụ: quản lý dự án, kiểm soát chất lượng) và sau đó dần dần mở rộng ra các quy trình khác.
- **Tích hợp phương pháp Agile:**
  Do nguồn lực hạn chế, công ty có thể kết hợp các nguyên tắc của Agile với CMM để tạo ra một hệ thống quản lý quy trình linh hoạt, giảm bớt gánh nặng về tài liệu và thủ tục mà vẫn đảm bảo chất lượng sản phẩm.
- **Đào tạo và sử dụng công cụ hỗ trợ:**
  Đầu tư vào đào tạo cho nhân viên về các quy trình chủ chốt và ứng dụng các công cụ tự động hóa, giúp giảm thiểu thời gian và công sức cần thiết để tuân thủ quy trình.
- **Hỗ trợ từ bên ngoài:**
  Nếu có thể, công ty có thể hợp tác với các chuyên gia tư vấn hoặc các đối tác có kinh nghiệm trong việc áp dụng CMM để được hướng dẫn cụ thể, tối ưu hóa chi phí và thời gian triển khai

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

**Thực hiện làm rõ thông tin:**
Tổ chức các cuộc họp, phỏng vấn hay workshop trực tiếp với khách hàng để đặt câu

hỏi và thu thập thông tin chi tiết. Việc này giúp loại bỏ những hiểu lầm và xác định rõ ràng mong đợi của khách hàng.

**Sử dụng kỹ thuật trực quan:**
Áp dụng các kỹ thuật như mô phỏng, thiết kế wireframe, prototype hoặc bản vẽ sơ bộ để trực quan hóa yêu cầu. Khi khách hàng thấy được hình ảnh hay mẫu cụ thể, họ sẽ dễ dàng cung cấp phản hồi và làm rõ ý tưởng của mình.

**Ghi chép và xác nhận:**
Ghi chép lại các yêu cầu đã được thảo luận và gửi cho khách hàng để xác nhận. Điều này đảm bảo rằng cả hai bên đều có cùng một nhận thức về các yêu cầu cần thực hiện.

**Thực hiện vòng phản hồi ngắn:**
Áp dụng mô hình phát triển theo chu kỳ (iterative) để nhanh chóng đưa ra sản phẩm mẫu và nhận phản hồi từ khách hàng. Qua đó, yêu cầu được liên tục làm rõ và cải thiện

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?

**Tổ chức lại quá trình thu thập yêu cầu:**
Trước khi bắt đầu dự án, tiến hành các buổi làm rõ yêu cầu với khách hàng, tập trung vào việc thu thập càng nhiều thông tin càng tốt. Tạo bản nháp tài liệu yêu cầu để khách hàng xem xét và chỉnh sửa.

**Phân tích rủi ro và lập kế hoạch dự phòng:**
Xác định các điểm rủi ro phát sinh từ sự mơ hồ của yêu cầu và xây dựng kế hoạch giảm thiểu, chẳng hạn như thiết lập các mốc kiểm tra lại và cập nhật yêu cầu theo định kỳ.

**Áp dụng phương pháp phát triển linh hoạt (Agile):**
Nếu các yêu cầu không thể được làm rõ hoàn toàn ngay từ đầu, phương pháp Agile cho phép đội phát triển linh hoạt điều chỉnh theo các phản hồi liên tục từ khách hàng trong suốt quá trình dự án.

**Tăng cường giao tiếp và quản lý thay đổi:**
Đảm bảo có kênh giao tiếp thường xuyên giữa đội phát triển và khách hàng, giúp kịp thời phát hiện và xử lý các thay đổi hoặc sai lệch trong yêu cầu. Sử dụng các công cụ quản lý thay đổi để theo dõi và kiểm soát các phiên bản của tài liệu yêu cầu.

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?

1. **Thiết lập quy trình làm việc chung**
    - Áp dụng mô hình Agile (Scrum, SAFe) hoặc DevOps để đồng bộ quy trình.
    - Xây dựng tài liệu hướng dẫn chi tiết về coding standard, review code, và quản lý issue.
2. **Sử dụng công cụ hỗ trợ**
    - Quản lý mã nguồn: Git (GitHub, GitLab, Bitbucket).
    - Quản lý nhiệm vụ: JIRA, Trello, Asana.
    - Giao tiếp: Slack, Microsoft Teams, Zoom.
3. **Tổ chức họp định kỳ**
    - Daily Standup: Cập nhật tiến độ, giải quyết vấn đề nhanh chóng.
    - Sprint Review/Demo: Chia sẻ kết quả phát triển giữa các nhóm.
    - Retrospective: Đánh giá hiệu suất và cải tiến quy trình.
4. **Xây dựng môi trường CI/CD**
    - Thiết lập hệ thống tích hợp liên tục (Jenkins, GitHub Actions).
    - Triển khai môi trường test chung để đảm bảo tính tương thích.
5. **Chuẩn hóa giao tiếp và báo cáo**
    - Mỗi nhóm cần có đại diện tham gia vào các cuộc họp liên nhóm.
    - Tạo dashboard theo dõi tiến độ chung.

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.

**Xây dựng mô hình quản lý phù hợp**

- Áp dụng mô hình CMMI, ISO 9001 hoặc Agile để chuẩn hóa quy trình.
- Định nghĩa rõ ràng từng giai đoạn trong vòng đời phát triển phần mềm (SDLC).

**Tạo tài liệu quy trình chuẩn**

- Viết hướng dẫn về các bước phát triển, review code, kiểm thử, và triển khai.
- Sử dụng Wiki nội bộ (Confluence, Notion) để lưu trữ tài liệu.

**Ứng dụng công cụ quản lý**

- Quản lý dự án: JIRA, ClickUp.
- Quản lý mã nguồn: GitHub, GitLab.
- Quản lý kiểm thử: TestRail, Postman.

**Tổ chức đào tạo nội bộ**

- Hướng dẫn nhân viên về quy trình mới, công cụ sử dụng.
- Tổ chức workshop, training để đảm bảo hiểu rõ quy trình.

**Thiết lập quy trình kiểm soát chất lượng**

- Áp dụng code review bắt buộc trước khi merge code.
- Thực hiện kiểm thử tự động và kiểm thử thủ công theo tiêu chuẩn.

**Theo dõi và cải tiến liên tục**

- Lấy phản hồi từ nhân viên để cải tiến quy trình.
- Kiểm tra định kỳ và điều chỉnh khi cần thiết.