**CHƯƠNG 1**

**Câu hỏi trắc nghiệm**

**Câu hỏi 1: Phần mềm bao gồm các loại nào dưới đây?**

A. Phần mềm hệ thống

B. Phần mềm ứng dụng

C. Phần mềm nhúng

D. Cả A, B và C

**Câu hỏi 2: Công nghệ phần mềm là gì?**

A. Việc viết mã nguồn cho phần mềm

B. Phát triển phần mềm mà không có lỗi

C. Ứng dụng các phương pháp khoa học để phát triển phần mềm

D. Chỉ bảo trì phần mềm

**Câu hỏi 3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính?**

A. 3

B. 4

C. 5

D. 6

**Câu hỏi 4: Hoạt động nào dưới đây thuộc quy trình bảo trì phần mềm?**

A. Lập kế hoạch

B. Triển khai phần mềm

C. Cập nhật phần mềm để phù hợp với thay đổi môi trường

D. Phân tích yêu cầu

**Câu hỏi 5: Chi phí bảo trì phần mềm chiếm bao nhiêu phần trăm tổng chi phí vòng đời** **phần mềm?**

A. 46%

B. 56%

C. 76%

D. 86%

**Câu hỏi 6: Nguyên nhân chính gây ra việc vượt chi phí khi phát triển phần mềm là gì?**

A. Thiếu nhân lực

B. Không xác định rõ yêu cầu

C. Thay đổi công nghệ

D. Cả A và C

**Câu hỏi 7: Yêu cầu nào dưới đây không phải là yêu cầu phi chức năng?**

A. Hiệu suất xử lý

B. Tính bảo mật

C. Khả năng mở rộng

D. Chức năng đăng nhập

**Câu hỏi 8: Khi nào phần mềm được coi là hoàn thành?**

A. Khi hoàn thành việc viết mã nguồn

B. Khi được bàn giao cho khách hàng và không còn lỗi

C. Khi được triển khai trên hệ thống của khách hàng

D. Khi được khách hàng chấp nhận và đưa vào sử dụng

**Câu hỏi 9: Vấn đề phổ biến nào thường gặp khi phát triển phần mềm?**

A. Thiếu công cụ hỗ trợ

B. Vượt chi phí, trễ thời hạn và lỗi sau khi bàn giao

C. Không có đội kiểm thử

D. Tất cả đều đúng

**Câu hỏi 10: Phần mềm có thể được chia thành bao nhiêu loại chính?**

A. 2

B. 3

C. 4

D. 5

**Câu hỏi ngắn:**

**1. Phần mềm là gì?**

Phần mềm là tập hợp các hướng dẫn, chương trình được viết để máy tính thực thi nhằm thực hiện các chức năng hoặc nhiệm vụ cụ thể.

**2. Công nghệ phần mềm?**

Công nghệ phần mềm là lĩnh vực nghiên cứu, phát triển và áp dụng các phương pháp có hệ thống để xây dựng phần mềm chất lượng cao.

**3. Các loại phần mềm chính là gì?**

* Phần mềm hệ thống: Hệ điều hành, trình điều khiển thiết bị.
* Phần mềm ứng dụng: Các chương trình phục vụ công việc hoặc nhu cầu của ngừoi dùng cuối như Microsoft Office, trình duyệt Web.
* Phần mềm nhúng: Phần mềm điều khiển các thiết bị phần cứng như máy giặt, máy điều hoà.

**4. Tại sao công nghệ phần mềm lại quan trọng?**

Công nghệ phần mềm giúp tạo ra các ứng dụng phục vụ nhu cầu con người, tự động hóa quy trình, tối ưu hóa công việc và cải thiện chất lượng cuộc sống. Phần mềm hỗ trợ trong nhiều lĩnh vực như y tế, giáo dục, tài chính, giao thông và giải trí. Nếu không có công nghệ phần mềm, hầu hết các hệ thống hiện đại sẽ không thể vận hành hiệu quả.

**5. Quy trình phát triển phần mềm gồm những giai đoạn nào?**

* Lấy yêu cầu: Thu thập và phân tích các yêu cầu từ khách hàng.
* Thiết kế: Lên kế hoạch và cấu trúc hệ thống phần mềm.
* Lập trình: Chuyển đổi thiết kế thành mã nguồn thực thi.
* Kiểm thử: Đảm bảo phần mềm hoạt động đúng chức năng.
* Triển khai: Cài đặt và bàn giao phần mềm cho khách hàng.
* Bảo trì: Khắc phụ lỗi và nâng cấp phần mềm sau khi triển khai.

**Câu 6. Khía cạnh kinh tế của công nghệ phần mềm là gì?**

Công nghệ phần mềm không chỉ là một lĩnh vực kỹ thuật mà còn có tác động lớn đến nền kinh tế, ảnh hưởng trực tiếp đến chi phí, lợi ích và hiệu quả tài chính của doanh nghiệp. Ngày nay, phần mềm là yếu tố cốt lõi trong nhiều ngành công nghiệp như tài chính, y tế, giáo dục, đóng vai trò quan trọng trong việc tự động hóa quy trình, nâng cao năng suất và tối ưu hóa chi phí vận hành. Dưới đây là những khía cạnh kinh tế quan trọng của công nghệ phần mềm:

1. Chi phí phát triển phần mềm

* Việc phát triển phần mềm đòi hỏi nguồn lực lớn, bao gồm nhân sự (lập trình viên, kiểm thử viên, quản lý dự án), công cụ phát triển, cơ sở hạ tầng và phần mềm hỗ trợ.
* Các dự án phần mềm lớn thường có chi phí rất cao, do đó việc áp dụng công nghệ phần mềm giúp kiểm soát chi phí tốt hơn thông qua các phương pháp phát triển linh hoạt như Agile, DevOps.

2. Chi phí bảo trì và vận hành

* Chi phí bảo trì phần mềm thường chiếm 50-70% tổng chi phí vòng đời sản phẩm, bao gồm sửa lỗi, cập nhật, mở rộng tính năng và tối ưu hóa hiệu suất.
* Nếu phần mềm được thiết kế tốt ngay từ đầu, doanh nghiệp có thể giảm thiểu chi phí bảo trì và tăng tính ổn định của hệ thống.

3. Giá trị kinh tế và lợi ích kinh doanh

* Sự phát triển của phần mềm giúp tăng năng suất lao động bằng cách tự động hóa quy trình, giảm sự phụ thuộc vào nhân công và cải thiện tốc độ xử lý công việc.
* Doanh nghiệp có thể thương mại hóa phần mềm theo nhiều mô hình khác nhau như bản quyền thương mại (COTS), phần mềm dịch vụ (SaaS), mã nguồn mở (Open-source), giúp tạo ra doanh thu ổn định.

4. Kinh tế quy mô trong phần mềm

* Không giống như sản phẩm vật lý, phần mềm có thể được sao chép và phân phối với chi phí gần như bằng không. Một phần mềm sau khi phát triển có thể được bán cho hàng triệu khách hàng mà không tốn thêm chi phí sản xuất, giúp tối ưu hóa lợi nhuận.
* Các tập đoàn lớn như Microsoft, Google, Amazon đã tận dụng yếu tố này để phát triển hệ sinh thái phần mềm mạnh mẽ, mang lại lợi nhuận khổng lồ.

5. Rủi ro tài chính trong phát triển phần mềm

* Các dự án phần mềm có thể gặp rủi ro như vượt ngân sách, chậm tiến độ, hoặc không đáp ứng được nhu cầu thị trường. Điều này có thể gây tổn thất tài chính lớn, đặc biệt đối với các doanh nghiệp khởi nghiệp.
* Việc áp dụng các phương pháp quản lý dự án như Scrum, Kanban giúp doanh nghiệp giảm thiểu rủi ro và tối ưu hóa nguồn lực.

6. Chi phí cơ hội

* Nếu một phần mềm không được phát triển đúng tiến độ hoặc chất lượng không đảm bảo, doanh nghiệp có thể mất cơ hội kinh doanh, đánh mất thị phần vào tay đối thủ.
* Trong bối cảnh cạnh tranh khốc liệt, việc đầu tư vào công nghệ phần mềm không chỉ là chiến lược tiết kiệm chi phí mà còn là yếu tố quyết định sự thành công của doanh nghiệp.

**Câu 7. Khía cạnh công nghệ của công nghệ phần mềm là gì?**

Công nghệ phần mềm là một lĩnh vực không ngừng phát triển, đòi hỏi các hệ thống phần mềm phải liên tục thích ứng với các xu hướng công nghệ mới, sự gia tăng khối lượng dữ liệu và nhu cầu kết nối hệ thống phức tạp. Dưới đây là những khía cạnh công nghệ quan trọng của công nghệ phần mềm:

1. Sự phát triển nhanh chóng của công nghệ

* Công nghệ phần mềm luôn phải cải tiến liên tục để đáp ứng các yêu cầu mới về hiệu suất, bảo mật và khả năng mở rộng.
* Các xu hướng như trí tuệ nhân tạo (AI), điện toán đám mây (Cloud Computing), chuỗi khối (Blockchain) và Internet vạn vật (IoT) đã làm thay đổi cách phần mềm được thiết kế và triển khai.
* Việc tự động hóa quy trình phát triển phần mềm (CI/CD) giúp giảm thời gian phát hành sản phẩm và nâng cao chất lượng phần mềm.

2. Kiến trúc phần mềm hiện đại

* Sự phát triển của kiến trúc microservices giúp phần mềm dễ bảo trì, mở rộng và cập nhật mà không ảnh hưởng đến toàn bộ hệ thống.
* Các ứng dụng serverless trên nền tảng AWS Lambda, Azure Functions giúp giảm chi phí vận hành và tăng hiệu quả xử lý.
* Containerization (Docker, Kubernetes) cho phép triển khai phần mềm linh hoạt hơn trên nhiều môi trường khác nhau.

3. Kết nối các hệ thống phức tạp

* Phần mềm ngày nay không hoạt động độc lập mà cần kết nối với nhiều hệ thống khác nhau, từ hệ thống doanh nghiệp (ERP, CRM), dịch vụ đám mây, đến thiết bị IoT.
* API (Application Programming Interface) đóng vai trò quan trọng trong việc tích hợp các hệ thống, cho phép dữ liệu luân chuyển một cách nhanh chóng và bảo mật.
* Các giao thức truyền thông như RESTful API, GraphQL, WebSocket giúp tối ưu hóa hiệu suất và khả năng mở rộng của hệ thống phần mềm.

4. Xử lý và phân tích khối lượng dữ liệu lớn

* Với sự bùng nổ của dữ liệu, phần mềm hiện đại cần khả năng thu thập, xử lý và phân tích dữ liệu theo thời gian thực.
* Các công nghệ như Big Data, Data Lake, Machine Learning giúp phần mềm khai thác dữ liệu hiệu quả, hỗ trợ doanh nghiệp ra quyết định dựa trên dữ liệu.
* Hệ quản trị cơ sở dữ liệu hiện đại như NoSQL (MongoDB, Cassandra), SQL phân tán (CockroachDB, Amazon Aurora) giúp xử lý dữ liệu lớn với hiệu suất cao.

5. Bảo mật trong công nghệ phần mềm

* Khi phần mềm ngày càng kết nối với nhiều hệ thống, nguy cơ bảo mật cũng gia tăng, đòi hỏi phải tích hợp các giải pháp bảo mật ứng dụng (AppSec), mã hóa dữ liệu, xác thực đa yếu tố (MFA) và Zero Trust Security.
* DevSecOps được áp dụng để tích hợp bảo mật vào toàn bộ vòng đời phát triển phần mềm, giúp phát hiện và khắc phục lỗ hổng sớm hơn.

**Câu 8. Khía cạnh bảo trì của công nghệ phần mềm là gì?**

Sau khi phần mềm được triển khai, việc bảo trì là một hoạt động quan trọng và tốn kém, nhằm đảm bảo phần mềm hoạt động ổn định, bảo mật và đáp ứng các yêu cầu thay đổi của người dùng. Theo thống kê, bảo trì phần mềm chiếm khoảng 60% tổng chi phí vòng đời của một hệ thống phần mềm, cho thấy đây là một khía cạnh quan trọng cần được quản lý hiệu quả.

a. Bảo trì sửa lỗi (Corrective Maintenance)

* Mục tiêu: Khắc phục lỗi xuất hiện trong quá trình sử dụng phần mềm.
* Lỗi có thể đến từ code, dữ liệu, giao diện hoặc hiệu suất hệ thống.
* Ví dụ: Fix lỗi crash ứng dụng, sửa lỗi hiển thị trên giao diện người dùng.

b. Bảo trì thích nghi (Adaptive Maintenance)

* Mục tiêu: Điều chỉnh phần mềm để thích nghi với sự thay đổi của môi trường bên ngoài.
* Thường liên quan đến cập nhật hệ điều hành, cơ sở dữ liệu, phần cứng hoặc tích hợp với hệ thống mới.
* Ví dụ: Cập nhật phần mềm để tương thích với phiên bản mới của Windows hoặc Android.

c. Bảo trì hoàn thiện (Perfective Maintenance)

* Mục tiêu: Cải thiện hiệu suất, tối ưu hóa hoặc bổ sung tính năng mới mà người dùng yêu cầu.
* Giúp tăng trải nghiệm người dùng và nâng cao hiệu suất của phần mềm.
* Ví dụ: Nâng cấp giao diện UI/UX, tối ưu thuật toán để chạy nhanh hơn.

d. Bảo trì phòng ngừa (Preventive Maintenance)

* Mục tiêu: Dự đoán và ngăn chặn lỗi xảy ra trong tương lai, giúp phần mềm hoạt động ổn định lâu dài.
* Thường liên quan đến tối ưu hóa mã nguồn, cải thiện bảo mật, kiểm tra hệ thống định kỳ.
* Ví dụ: Refactoring (tái cấu trúc) code để dễ bảo trì hơn, cập nhật vá lỗi bảo mật.

**Câu 9. Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì?**

Trong quá trình phát triển phần mềm, việc hoàn thành dự án đúng tiến độ là một thách thức lớn. Thực tế, nhiều dự án bị trễ tiến độ so với kế hoạch ban đầu do ước lượng thời gian không chính xác và nhiều yếu tố khách quan lẫn chủ quan tác động. Dưới đây là các nguyên nhân phổ biến dẫn đến tình trạng này:

1. Yêu cầu thay đổi liên tục từ phía khách hàng

* Khách hàng có thể thay đổi yêu cầu giữa chừng, đòi hỏi nhóm phát triển phải điều chỉnh thiết kế, tính năng hoặc kiến trúc hệ thống.
* Việc thay đổi này tác động đến kế hoạch ban đầu, khiến nhóm phát triển phải làm lại một số phần hoặc thay đổi toàn bộ chiến lược.
* Nếu không có quy trình quản lý yêu cầu chặt chẽ, dự án dễ rơi vào tình trạng "scope creep" – tức là phạm vi dự án mở rộng liên tục mà không có kế hoạch rõ ràng.

2. Ước lượng thời gian không chính xác

* Ước lượng không chính xác là nguyên nhân phổ biến nhất dẫn đến trễ hạn.
* Một số yếu tố gây sai lệch trong ước lượng thời gian:
  + Đánh giá thấp độ phức tạp của phần mềm.
  + Không tính toán đúng thời gian phát sinh lỗi và sửa lỗi.
  + Không dự trù thời gian cho việc kiểm thử và triển khai.
* Nếu ước lượng sai ngay từ đầu, các giai đoạn sau dễ bị chồng chéo, dẫn đến chậm tiến độ.

3. Thiếu nhân lực hoặc sự phối hợp kém trong nhóm phát triển

* Thiếu nhân sự hoặc chất lượng nhân sự không đáp ứng yêu cầu khiến công việc bị đình trệ.
* Phân công công việc không hợp lý có thể dẫn đến một số thành viên quá tải trong khi người khác không có việc làm.
* Sự phối hợp kém giữa lập trình viên, kiểm thử viên, quản lý dự án và khách hàng có thể dẫn đến hiểu sai yêu cầu, phát triển sai hướng, gây mất thời gian chỉnh sửa.

4. Thiếu tài liệu và quy trình phát triển không rõ ràng

* Nếu không có tài liệu thiết kế phần mềm đầy đủ, lập trình viên có thể hiểu sai yêu cầu, dẫn đến việc phát triển sai tính năng.
* Quy trình phát triển không rõ ràng sẽ khiến nhóm làm việc thiếu tổ chức, khó kiểm soát tiến độ.
* Thiếu kiểm thử liên tục có thể dẫn đến phát hiện lỗi muộn, kéo dài thời gian sửa lỗi.

5. Công nghệ mới hoặc không phù hợp

* Việc áp dụng công nghệ mới mà nhóm phát triển chưa có kinh nghiệm có thể làm chậm tiến độ do thời gian học tập và thử nghiệm.
* Nếu lựa chọn công nghệ không phù hợp, có thể dẫn đến tình trạng tắc nghẽn hiệu suất hoặc khó mở rộng, khiến nhóm phải quay lại thay đổi công nghệ giữa chừng.

6. Vấn đề tài chính và thay đổi trong tổ chức

* Nếu dự án gặp vấn đề về ngân sách, có thể dẫn đến cắt giảm nhân sự hoặc tài nguyên, ảnh hưởng đến tiến độ.
* Thay đổi lãnh đạo hoặc tái cơ cấu công ty cũng có thể làm gián đoạn dự án, khiến quá trình phát triển bị trì hoãn.

7. Không có kế hoạch dự phòng rủi ro

* Nếu không có kế hoạch đối phó với rủi ro, khi gặp sự cố như nhân viên nghỉ việc, lỗi phần mềm nghiêm trọng, hoặc yêu cầu thay đổi lớn, dự án sẽ bị đình trệ.
* Một số nhóm phát triển không dự trù thời gian buffer để xử lý vấn đề phát sinh, dẫn đến việc chậm tiến độ.

Cách giảm thiểu tình trạng trễ hạn

* Áp dụng mô hình phát triển phù hợp: Agile, Scrum giúp linh hoạt xử lý yêu cầu thay đổi.
* Sử dụng phương pháp ước lượng chính xác: Kết hợp Planning Poker, PERT, Function Points để dự báo thời gian tốt hơn.
* Quản lý yêu cầu chặt chẽ: Xây dựng tài liệu yêu cầu (SRS), quản lý bằng công cụ như JIRA, Trello để theo dõi tiến độ.
* Cải thiện kỹ năng quản lý dự án: Lập kế hoạch chi tiết, phân bổ công việc hợp lý và giám sát tiến độ thường xuyên.
* Kiểm thử sớm và liên tục: Áp dụng CI/CD, Test Automation để phát hiện lỗi sớm và tránh sửa lỗi muộn.

**Câu 10.Bảo trì phần mềm bao gồm những hoạt động nào?**

1. Bảo trì sửa lỗi (Corrective Maintenance)

* Nhằm khắc phục các lỗi phát sinh trong quá trình sử dụng phần mềm.
* Các lỗi có thể bao gồm:
  + Lỗi chức năng: Phần mềm không hoạt động như mong đợi.
  + Lỗi giao diện: Sai sót về hiển thị hoặc trải nghiệm người dùng.
  + Lỗi bảo mật: Lỗ hổng bảo mật cần được vá để tránh rủi ro tấn công.
  + Lỗi hiệu suất: Hệ thống chậm, quá tải hoặc tiêu tốn tài nguyên bất thường.
* Ví dụ: Một ứng dụng ngân hàng gặp lỗi khi xử lý giao dịch trực tuyến, nhóm bảo trì sẽ cập nhật mã nguồn để khắc phục.

2. Bảo trì thích ứng (Adaptive Maintenance)

* Nhằm điều chỉnh phần mềm để thích ứng với những thay đổi của môi trường hệ thống.
* Các thay đổi có thể đến từ:
  + Hệ điều hành mới: Phần mềm cần cập nhật để hoạt động trên Windows 11 hoặc phiên bản iOS mới nhất.
  + Phần cứng mới: Hỗ trợ cho vi xử lý mới hoặc thiết bị di động mới.
  + Công nghệ mới: Tích hợp trí tuệ nhân tạo (AI), điện toán đám mây (Cloud) hoặc công nghệ bảo mật tiên tiến.
* Ví dụ: Một phần mềm kế toán cần cập nhật để tuân theo quy định tài chính mới của chính phủ.

3. Bảo trì hoàn thiện (Perfective Maintenance)

* Nhằm nâng cao hiệu suất, tính ổn định và trải nghiệm người dùng.
* Hoạt động này không sửa lỗi mà tập trung vào cải thiện hệ thống dựa trên phản hồi của người dùng và yêu cầu của doanh nghiệp.
* Các hoạt động chính:
  + Tối ưu hóa mã nguồn để tăng tốc độ xử lý.
  + Cải tiến giao diện người dùng (UI/UX) để thân thiện hơn.
  + Bổ sung tính năng mới để đáp ứng nhu cầu khách hàng.
* Ví dụ: Cập nhật thuật toán tìm kiếm trên một ứng dụng thương mại điện tử để giúp khách hàng tìm sản phẩm nhanh hơn.

4. Bảo trì phòng ngừa (Preventive Maintenance)

* Nhằm dự đoán và ngăn chặn các sự cố có thể xảy ra trong tương lai.
* Các hoạt động chính:
  + Phân tích mã nguồn để phát hiện lỗi tiềm ẩn.
  + Sao lưu dữ liệu định kỳ để đề phòng mất dữ liệu.
  + Cập nhật hệ thống bảo mật để ngăn chặn nguy cơ tấn công mạng.
* Ví dụ: Một công ty triển khai các bản vá bảo mật hàng tháng để ngăn chặn lỗ hổng trước khi hacker khai thác.

**Câu hỏi thảo luận nhóm**

**1. Phân biệt phần mềm hệ thống và phần mềm ứng dụng.**

* **Phần mềm hệ thống:** Quản lý và điều khiển phần cứng máy tính, tạo nền tảng cho các ứng dụng hoạt động. Ví dụ: Hệ điều hành (Windows, Linux), trình điều khiển thiết bị.
* **Phần mềm ứng dụng**: Được phát triển để phục vụ nhu cầu cụ thể của người dùng, như soạn thảo văn bản, duyệt web, chơi game. Ví dụ: Microsoft Word, Google Chrome.

**2. Thảo luận về vai trò của công nghệ phần mềm trong lĩnh vực tài chính.**

* Hỗ trợ giao dịch trực tuyến, ngân hàng điện tử.
* Cung cấp hệ thống quản lý tài chính, kế toán.
* Đảm bảo bảo mật dữ liệu và giao dịch an toàn.
* Phân tích dữ liệu tài chính, hỗ trợ ra quyết định.
* Tự động hóa quy trình như quản lý rủi ro, kiểm soát gian lận.

**3. Nêu các thách thức thường gặp trong bảo trì phần mềm.**

* **Tính tương thích**: Phần mềm cần được cập nhật để tương thích với hệ thống mới.
* **Bảo mật**: Liên tục vá lỗi bảo mật để chống lại các cuộc tấn công mạng.
* **Hiệu suất**: Phải tối ưu hóa để đảm bảo chạy mượt mà khi dữ liệu và số lượng người dùng tăng.
* **Yêu cầu thay đổi**: Người dùng yêu cầu thêm tính năng mới hoặc sửa đổi giao diện.
* **Tài liệu không đầy đủ**: Việc bảo trì gặp khó khăn nếu tài liệu thiết kế ban đầu không rõ ràng.

**4. Vì sao phần mềm thương mại điện tử cần được bảo trì thường xuyên?**

* **Cập nhật bảo mật**: Ngăn chặn hacker đánh cắp dữ liệu khách hàng.
* **Cải thiện hiệu suất**: Đảm bảo tốc độ tải trang nhanh, đáp ứng nhu cầu mua sắm.
* **Tương thích với công nghệ mới**: Hỗ trợ nhiều nền tảng thanh toán và thiết bị.
* **Cập nhật tính năng mới**: Cải thiện trải nghiệm khách hàng, tăng doanh thu.
* **Khắc phục lỗi hệ thống**: Đảm bảo quá trình mua hàng không bị gián đoạn.

**5. Phân tích những vấn đề khi yêu cầu khách hàng liên tục thay đổi trong quá trình phát triền phần mềm.**

* **Tăng chi phí và thời gian**: Thay đổi yêu cầu dẫn đến việc viết lại mã nguồn, kiểm thử lại, kéo dài tiến độ dự án.
* **Khó khăn trong quản lý dự án**: Dễ làm nhóm phát triển mất phương hướng, khó kiểm soát phạm vi dự án.
* **Xung đột giữa các bên liên quan**: Nhà phát triển, khách hàng, và nhà đầu tư có thể có ý kiến khác nhau.
* **Ảnh hưởng đến chất lượng phần mềm**: Các thay đổi liên tục có thể gây lỗi hoặc làm phần mềm trở nên phức tạp.
* **Giải pháp**: Sử dụng phương pháp phát triển linh hoạt (Agile), quản lý yêu cầu chặt chẽ ngay từ đầu.

**Câu 6. So sánh chi phí phát triển và chi phí bảo trì phần mềm.**

Chi phí phát triển và chi phí bảo trì phần mềm đều đóng vai trò quan trọng trong vòng đời của một sản phẩm phần mềm. Tuy nhiên, chi phí bảo trì thường cao hơn nhiều so với chi phí phát triển ban đầu, chiếm từ 60% đến 80% tổng chi phí vòng đời của phần mềm.

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Chi phí phát triển phần mềm | Chi phí bảo trì phần mềm |
| Khái niệm | Chi phí để thiết kế, lập trình, kiểm thử và triển khai phần mềm ban đầu. | Chi phí để sửa lỗi, nâng cấp, tối ưu hóa và đảm bảo phần mềm hoạt động lâu dài. |
| Tỷ lệ trong vòng đời phần mềm | Chiếm khoảng 20% - 40% tổng chi phí vòng đời phần mềm. | Chiếm khoảng 60% - 80% tổng chi phí vòng đời phần mềm. |
| Mục tiêu chính | Xây dựng phần mềm từ đầu với đầy đủ tính năng đáp ứng yêu cầu. | Duy trì, cập nhật và cải tiến phần mềm theo thời gian. |
| |  | | --- | | Hoạt động chính |  |  | | --- | |  | | - Phân tích yêu cầu, thiết kế hệ thống.  - Viết mã nguồn, kiểm thử, triển khai. | - Sửa lỗi (bảo trì sửa lỗi).  - Cập nhật để phù hợp với công nghệ mới (bảo trì thích ứng).  - Cải thiện hiệu suất, thêm tính năng mới (bảo trì hoàn thiện).  - Ngăn ngừa sự cố tiềm ẩn (bảo trì phòng ngừa). |
| Độ phức tạp | Dễ kiểm soát hơn vì các yêu cầu thường được xác định trước. | Khó kiểm soát vì yêu cầu có thể thay đổi liên tục và xuất hiện các vấn đề không lường trước. |
| Tác động đến doanh nghiệp | Cần đầu tư ban đầu lớn nhưng có thể lên kế hoạch trước. | Nếu không bảo trì tốt, phần mềm có thể lỗi thời, gây mất dữ liệu, giảm hiệu suất hoặc mất khách hàng. |

Ví dụ : Phần mềm quản lý khách sạn

* Chi phí phát triển ban đầu: Công ty A thuê một nhóm lập trình viên để xây dựng hệ thống quản lý khách sạn với chi phí khoảng 100.000 USD.
* Chi phí bảo trì hàng năm:
  + Sửa lỗi phần mềm: 10.000 USD/năm
  + Cập nhật để tương thích với phần cứng và hệ điều hành mới: 20.000 USD/năm
  + Thêm tính năng mới (như tích hợp AI để dự báo đặt phòng): 30.000 USD/năm
  + Bảo mật và sao lưu dữ liệu: 15.000 USD/năm  
    => Tổng chi phí bảo trì hàng năm: 75.000 USD  
    => Sau 5 năm, chi phí bảo trì đã lên đến 375.000 USD, gấp 3,75 lần chi phí phát triển ban đầu.

**Câu 7. Phân biệt các loại yêu cầu trong phát triển phần mềm (chức năng và phi chức năng).**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Yêu cầu chức năng (Functional Requirements) | Yêu cầu phi chức năng (Non-functional Requirements) |
| Khái niệm | Mô tả hệ thống phải làm gì, các chức năng và tính năng mà phần mềm cần có để đáp ứng yêu cầu của người dùng. | Mô tả hệ thống hoạt động như thế nào, tập trung vào hiệu suất, bảo mật, khả năng mở rộng và trải nghiệm người dùng. |
| Mục tiêu | Định nghĩa các thao tác, dịch vụ và chức năng mà hệ thống phải cung cấp. | Định nghĩa các tiêu chí chất lượng, hiệu suất, bảo mật và các ràng buộc hệ thống. |
| Đối tượng chính | Người dùng cuối, khách hàng, nhà phát triển. | Kiến trúc sư hệ thống, quản lý dự án, đội bảo mật. |
| Ví dụ | - Đăng ký tài khoản người dùng.  - Đăng nhập và xác thực tài khoản.  - Tìm kiếm sản phẩm theo tên, danh mục.  - Thanh toán trực tuyến bằng thẻ tín dụng. | - Hệ thống phải có thời gian phản hồi dưới 2 giây cho mỗi yêu cầu.  - Dữ liệu người dùng phải được mã hóa để đảm bảo bảo mật.  - Ứng dụng phải hỗ trợ 100.000 người dùng truy cập đồng thời.  - Giao diện phải thân thiện, dễ sử dụng trên di động và máy tính. |
| Tác động đến phần mềm | Nếu không đáp ứng yêu cầu chức năng, hệ thống không thể hoạt động đúng. | Nếu không đáp ứng yêu cầu phi chức năng, hệ thống vẫn có thể chạy nhưng kém hiệu quả, chậm, không bảo mật. |

Ví dụ : Website thương mại điện tử

* Yêu cầu chức năng:
  + Người dùng có thể tìm kiếm sản phẩm theo danh mục.
  + Hệ thống hỗ trợ thanh toán bằng nhiều phương thức (thẻ, ví điện tử, chuyển khoản).
  + Người dùng có thể đánh giá và bình luận về sản phẩm.
* Yêu cầu phi chức năng:
  + Website phải tải trang dưới 3 giây ngay cả khi có 10.000 lượt truy cập đồng thời.
  + Hệ thống phải có khả năng tự động mở rộng tài nguyên khi số lượng truy cập tăng đột biến.
  + Giao diện phải tương thích với cả máy tính, điện thoại và máy tính bảng.

**Câu 8. Thảo luận về các mô hình quy trình phát triển phần mềm phổ biến.**

*1. Mô hình thác nước (Waterfall Model)*Mô hình này chia quá trình phát triển phần mềm thành các giai đoạn tuần tự, từ phân tích yêu cầu đến thiết kế, triển khai, kiểm thử và bảo trì. Mỗi giai đoạn phải hoàn thành trước khi chuyển sang giai đoạn tiếp theo.

Các giai đoạn:

1. Phân tích yêu cầu
2. Thiết kế hệ thống
3. Triển khai (lập trình)
4. Kiểm thử
5. Bảo trì

Ưu điểm:  
✔ Dễ hiểu và dễ quản lý do có quy trình rõ ràng.  
✔ Phù hợp với các dự án có yêu cầu ổn định, ít thay đổi.  
✔ Dễ dàng lên kế hoạch và ước lượng chi phí.

Nhược điểm:  
✘ Khó thích ứng với các thay đổi yêu cầu sau khi đã hoàn thành một giai đoạn.  
✘ Nếu xảy ra lỗi trong giai đoạn đầu, chi phí sửa chữa sẽ rất cao.  
✘ Thời gian phát triển dài, người dùng phải chờ đến khi hoàn thành mới có sản phẩm để sử dụng.

Ví dụ: Dùng cho phát triển phần mềm hệ thống nhúng, phần mềm quân sự, nơi yêu cầu phải rõ ràng ngay từ đầu.

*2. Mô hình xoắn ốc (Spiral Model)*  
Mô hình này kết hợp lặp lại (iterative) và quản lý rủi ro, với mỗi vòng xoắn bao gồm các bước: lập kế hoạch, phân tích rủi ro, thực hiện và đánh giá.

Ưu điểm:  
✔ Quản lý rủi ro tốt, thích hợp cho các dự án phức tạp.  
✔ Linh hoạt trong việc thay đổi yêu cầu trong từng vòng lặp.  
✔ Phù hợp với phần mềm lớn, có nhiều rủi ro chưa xác định.

Nhược điểm:  
✘ Chi phí cao do cần nhiều tài nguyên để phân tích rủi ro.  
✘ Quá trình phát triển phức tạp, khó quản lý.

Ví dụ: Áp dụng cho dự án phần mềm ngân hàng, phần mềm hàng không, nơi yêu cầu an toàn cao và có nhiều rủi ro.

*3. Mô hình phát triển linh hoạt (Agile Model)*  
Mô hình này tập trung vào phát triển phần mềm theo từng vòng lặp nhỏ (iteration), với sự tham gia liên tục của khách hàng để điều chỉnh sản phẩm.

Ưu điểm:  
✔ Thích ứng nhanh với sự thay đổi của yêu cầu.  
✔ Tạo ra sản phẩm sớm và có thể phát hành từng phiên bản nhỏ.  
✔ Giúp cải thiện sự hợp tác giữa các thành viên trong nhóm.

Nhược điểm:  
✘ Đòi hỏi đội ngũ phát triển có kỹ năng cao.  
✘ Khó dự đoán chi phí và thời gian hoàn thành chính xác.  
✘ Không phù hợp với dự án có yêu cầu rõ ràng ngay từ đầu.

Ví dụ: Áp dụng cho các ứng dụng web, mobile app như Facebook, Zalo, Grab, nơi yêu cầu thay đổi thường xuyên.

4. Mô hình phát triển lặp và gia tăng (Incremental & Iterative Model)

* Mô hình gia tăng (Incremental): Chia phần mềm thành các module nhỏ, mỗi phần được phát triển và phát hành theo từng giai đoạn.
* Mô hình lặp (Iterative): Xây dựng phần mềm theo từng phiên bản, mỗi phiên bản cải tiến so với phiên bản trước.

Ưu điểm:  
✔ Cho phép phát hành sớm các tính năng quan trọng.  
✔ Giảm thiểu rủi ro nhờ cải thiện dần qua từng vòng lặp.  
✔ Khách hàng có thể trải nghiệm và phản hồi ngay từ các giai đoạn đầu.

Nhược điểm:  
✘ Quá trình phát triển có thể kéo dài do liên tục phải cập nhật.  
✘ Cần có kế hoạch rõ ràng để tránh việc các phiên bản sau chồng chéo nhau.

Ví dụ: Phát triển phần mềm game, website thương mại điện tử như Shopee, Tiki, Lazada.

5. Mô hình DevOps  
DevOps là mô hình kết hợp giữa phát triển phần mềm (Development) và vận hành hệ thống (Operations), giúp tự động hóa và tối ưu quy trình triển khai.

Ưu điểm:  
✔ Rút ngắn thời gian phát hành phần mềm.  
✔ Cải thiện độ ổn định và hiệu suất của hệ thống.  
✔ Giúp phát hiện lỗi sớm và giảm thiểu rủi ro khi triển khai.

Nhược điểm:  
✘ Cần công cụ và đội ngũ có chuyên môn cao.  
✘ Yêu cầu tích hợp liên tục (CI/CD), không phù hợp với dự án nhỏ.

Ví dụ: Dùng trong các công ty công nghệ lớn như Google, Amazon, Netflix, nơi yêu cầu triển khai nhanh và ổn định

**Câu 9. Đề xuất giải pháp giảm thiểu lỗi phần mềm sau khi bàn giao.**

Sau khi bàn giao, phần mềm vẫn có thể gặp lỗi do nhiều nguyên nhân như yêu cầu chưa rõ ràng, lỗi lập trình, hoặc thay đổi trong môi trường triển khai. Dưới đây là một số giải pháp để giảm thiểu lỗi phần mềm sau khi bàn giao:

1. Kiểm thử kỹ lưỡng trước khi bàn giao:

* Thực hiện kiểm thử đơn vị (Unit Test), kiểm thử tích hợp (Integration Test), kiểm thử hệ thống (System Test) trước khi phát hành.
* Sử dụng kiểm thử tự động (Automated Testing) để phát hiện lỗi sớm.
* Áp dụng kiểm thử hồi quy (Regression Testing) khi có thay đổi.

Ví dụ: Ứng dụng ngân hàng điện tử (Mobile Banking App): Trước khi phát hành, ngân hàng thực hiện kiểm thử bảo mật (Security Testing) để ngăn chặn lỗi có thể gây rò rỉ thông tin tài khoản khách hàng.

2. Đào tạo người dùng và cung cấp tài liệu hướng dẫn:

* Hướng dẫn người dùng cách sử dụng phần mềm đúng cách.
* Cung cấp tài liệu hướng dẫn, video tutorial giúp người dùng hiểu rõ chức năng.

Ví dụ: Hệ thống ERP của doanh nghiệp: Sau khi triển khai, công ty tổ chức các buổi đào tạo cho nhân viên để họ sử dụng hệ thống hiệu quả, giảm thiểu lỗi thao tác.

3. Xây dựng quy trình bảo trì và hỗ trợ sau bàn giao:

* Thiết lập kênh hỗ trợ khách hàng để tiếp nhận và xử lý lỗi kịp thời.
* Áp dụng bảo trì phòng ngừa (Preventive Maintenance) để phát hiện và sửa lỗi trước khi chúng ảnh hưởng đến hệ thống.

Ví dụ: Google Chrome: Khi người dùng báo lỗi trình duyệt, Google có nhóm hỗ trợ phản hồi nhanh và phát hành bản vá lỗi trong vòng vài ngày.

4. Sử dụng công cụ giám sát và log lỗi:

* Cài đặt hệ thống giám sát (Monitoring System) để theo dõi hiệu suất và lỗi trong thời gian thực.
* Ghi lại nhật ký lỗi (Logging) để dễ dàng phân tích và xử lý sự cố.

Ví dụ: Facebook: Sử dụng AI và hệ thống log lỗi tự động để phát hiện các bài đăng vi phạm chính sách, giúp giảm lỗi hệ thống và trải nghiệm người dùng mượt mà hơn.

5. Kiểm tra và cập nhật phần mềm định kỳ:

* Phát hành các bản cập nhật định kỳ để sửa lỗi và cải thiện hiệu suất.
* Áp dụng bảo trì điều chỉnh (Adaptive Maintenance) nếu có thay đổi về môi trường sử dụng.

Ví dụ: Hệ điều hành Windows: Microsoft phát hành bản cập nhật bảo mật hàng tháng (Patch Tuesday) để vá lỗi và cải thiện hiệu suất hệ thống.

**Câu 10.Vai trò của đội kiểm thử trong quy trình phát triển phần mềm**.

Đội kiểm thử phần mềm (QA - Quality Assurance) đóng vai trò quan trọng trong việc đảm bảo chất lượng sản phẩm trước khi phát hành. Họ giúp phát hiện lỗi, đảm bảo phần mềm đáp ứng yêu cầu và mang lại trải nghiệm tốt nhất cho người dùng.

1. Phát hiện và ngăn chặn lỗi trước khi triển khai:

* Kiểm thử giúp phát hiện lỗi logic, lỗi chức năng, lỗi giao diện, lỗi bảo mật trước khi phần mềm đến tay người dùng.
* Giảm chi phí sửa lỗi bằng cách phát hiện lỗi sớm trong vòng đời phát triển phần mềm.

Ví dụ: Một lỗi nhỏ trong ứng dụng ngân hàng có thể dẫn đến mất tiền của khách hàng. Đội kiểm thử giúp đảm bảo tính chính xác trước khi phát hành.

2. Đảm bảo phần mềm đáp ứng đúng yêu cầu:

* Đội kiểm thử so sánh phần mềm với tài liệu đặc tả yêu cầu (SRS - Software Requirement Specification) để đảm bảo phần mềm hoạt động đúng như mong muốn.
* Kiểm tra cả yêu cầu chức năng (các tính năng chính) và yêu cầu phi chức năng (hiệu suất, bảo mật, giao diện).

Ví dụ: Trong một website thương mại điện tử, đội kiểm thử sẽ kiểm tra xem quy trình đặt hàng, thanh toán, gửi email xác nhận có hoạt động chính xác không.

3. Cải thiện trải nghiệm người dùng (UX/UI Testing):

* Đảm bảo giao diện dễ sử dụng, trực quan, không có lỗi hiển thị trên các thiết bị khác nhau.
* Kiểm tra tính khả dụng (Usability Testing) để phát hiện các vấn đề gây khó chịu cho người dùng.

Ví dụ: Một ứng dụng di động bị lỗi font chữ hoặc nút bấm quá nhỏ sẽ gây khó khăn cho người dùng. Đội kiểm thử giúp phát hiện và đề xuất cải thiện.

4. Đảm bảo hiệu suất và khả năng mở rộng:

* Thực hiện kiểm thử tải (Load Testing), kiểm thử áp lực (Stress Testing), kiểm thử hiệu suất (Performance Testing) để đảm bảo hệ thống hoạt động mượt mà.
* Đánh giá khả năng mở rộng khi có số lượng người dùng lớn.

Ví dụ: Shopee, Lazada cần đảm bảo hệ thống không bị sập vào ngày sale lớn (11.11, 12.12) khi hàng triệu người truy cập cùng lúc.

5. Đảm bảo bảo mật phần mềm:

* Kiểm tra các lỗ hổng bảo mật (Penetration Testing) để ngăn chặn tấn công từ hacker.
* Đảm bảo dữ liệu người dùng được mã hóa an toàn.

Ví dụ: Ứng dụng thanh toán điện tử cần kiểm thử kỹ lưỡng để tránh lỗi SQL Injection, XSS, CSRF có thể khiến hacker đánh cắp thông tin thẻ tín dụng.

6. Hỗ trợ đội phát triển khắc phục lỗi nhanh chóng:

* Đội kiểm thử ghi lại lỗi trong hệ thống quản lý lỗi (JIRA, Bugzilla) và phối hợp với lập trình viên để sửa lỗi.
* Thực hiện kiểm thử hồi quy (Regression Testing) để đảm bảo lỗi đã sửa không ảnh hưởng đến các chức năng khác.

Ví dụ: Khi phát triển một tính năng mới trong Facebook, đội kiểm thử kiểm tra xem tính năng này có làm ảnh hưởng đến news feed, tin nhắn, video call hay không.

**Câu hỏi tình huống**

*Câu hỏi tình huống 1: Một công ty phát triển phần mềm quản lý tài chính đã hoàn thành dự án và bàn giao cho khách hàng. Tuy nhiên, sau 2 tháng sử dụng, khách hàng phát hiện ra nhiều lỗi phát sinh khi phần mềm xử lý các giao dịch có giá trị lớn. Hãy đề xuất giải pháp xử lý tình huống này.*

Khi khách hàng phản ánh lỗi trong phần mềm quản lý tài chính, công ty cần nhanh chóng tiếp nhận thông tin, xác định nguyên nhân và mức độ ảnh hưởng. Trước hết, đội ngũ kỹ thuật phải kiểm tra lại thuật toán xử lý giao dịch, đặc biệt với các giao dịch có giá trị lớn, để tìm ra lỗi và điều chỉnh. Sau đó, phần mềm cần được kiểm thử kỹ lưỡng trước khi cập nhật cho khách hàng, đồng thời hướng dẫn họ thực hiện nâng cấp. Để tránh tình trạng này tái diễn, công ty nên cải thiện quy trình kiểm thử, bổ sung các tình huống kiểm tra thực tế, nâng cao năng lực đội ngũ phát triển và thiết lập chính sách hỗ trợ sau bàn giao. Quan trọng hơn, công ty cần chủ động xin lỗi khách hàng, hỗ trợ khắc phục nhanh chóng và cam kết đảm bảo chất lượng dịch vụ để duy trì uy tín và sự tin tưởng.

*Câu hỏi tình huống 2: Trong quá trình phát triển phần mềm quản lý bệnh viện, khách hàng yêu cầu bổ sung thêm tính năng quản lý kho thuốc khi dự án đã đi vào giai đoạn kiểm thử. Là trưởng nhóm phát triển, bạn sẽ xử lý yêu cầu này như thế nào?*

Khi khách hàng yêu cầu bổ sung tính năng quản lý kho thuốc trong giai đoạn kiểm thử, với vai trò trưởng nhóm phát triển, em sẽ xử lý tình huống này theo các bước sau: Đầu tiên, ta cần phải tiếp nhận yêu cầu từ khách hàng và phải làm rõ phạm vi tính năng mới mà khách hàng yêu cầu, bao gồm các chức nắng cần có, mức độ ưu tiên của chức năng, thời gian mong muốn và tác động của nó đối với hệ thống hiện tại. Tiếp theo, ta phải phối hợp với đội ngũ phân tích mức độ ảnh hưởng của của tính năng này với tiến độ của dự án, kiến trúc phần mềm và cả tài nguyên chúng ta đang có. Nếu việc bổ sung không gây ra gián đoạn lớn cho dự án thì sẽ tiến hành tích hợp ngay trong giai đoạn kiểm thử. Tuy nhiên, nếu tính năng quá phức tạp, ta cần phải thảo luận với khách hàng đưa ra các phương án khác như đưa tính năng vào phiên bản cập nhật sau khi triển hoặc chia thành nhiều giai đoạn tùy thuộc vào tính năng khách hàng yêu cầu để giảm thiểu rủi ro cho dự án. Sau khi đã thống nhất phương án thực hiện, tiến hành điều phối nhóm phát triển để thiết kể, triển khai và kiểm thử tính năng mới cunngx như phối hợp với khách hàng để kiểm tra và xác nhận tính năng đã đáp ứng đúng yêu cầu của khách hàng không trước khi tích hợp vào hệ thống. Cuối cùng, cập nhật lại tài liệu dự án và quy trình kiểm thử để đảm bảo tính nhất quán trong toàn bộ hệ thống.

*Câu hỏi tình huống 3: Một nhóm phát triển phần mềm gặp phải vấn đề trễ tiến độ do nhiều thành viên không hiểu rõ yêu cầu của khách hàng. Là trưởng dự án, bạn sẽ làm gì để giải quyết vấn đề này và đảm bảo tiến độ dự án?*

Khi nhóm phát triển phần mềm bị trễ tiến độ do nhiều thành viên không hiểu rõ yêu cầu của khách hàng, để giải quyết vấn đề này và đảm bảo tiến độ dự án, trước tiên, cần tổ chức một cuộc họp nội bộ để xác định cụ thể những điểm chưa rõ trong yêu cầu cũng như mức độ ảnh hưởng của chúng đến tiến độ chung. Sau đó, cần liên hệ trực tiếp với khách hàng để làm rõ các yêu cầu chưa thống nhất, đồng thời ghi nhận mọi thay đổi hoặc điều chỉnh cần thiết. Nếu cần thiết, có thể đề xuất tổ chức các buổi trao đổi giữa nhóm phát triển và khách hàng để đảm bảo tất cả thành viên đều hiểu đúng về yêu cầu dự án. Tiếp theo, cần cập nhật lại tài liệu yêu cầu một cách rõ ràng, dễ hiểu và cung cấp hướng dẫn cụ thể cho từng thành viên trong nhóm. Đồng thời, việc xem xét lại kế hoạch dự án, ưu tiên các nhiệm vụ quan trọng và phân công công việc hợp lý cũng là điều cần thiết để bù đắp phần tiến độ đã bị trễ. Ngoài ra, cần thiết lập cơ chế giao tiếp hiệu quả hơn trong nhóm, chẳng hạn như họp ngắn hằng ngày để cập nhật tiến độ và giải quyết các vướng mắc kịp thời. Nếu cần thiết, có thể tổ chức các buổi đào tạo bổ sung để đảm bảo các thành viên nắm vững yêu cầu và thực hiện đúng nhiệm vụ. Cuối cùng, việc theo dõi sát sao tiến độ, điều chỉnh linh hoạt khi cần thiết và phối hợp chặt chẽ với khách hàng sẽ giúp đảm bảo sản phẩm đáp ứng đúng yêu cầu mà không làm ảnh hưởng đến chất lượng hoặc thời gian bàn giao.

*Câu hỏi tình huống 4: Sau khi triển khai phần mềm quản lý thư viện, người dùng phản hồi rằng giao diện khó sử dụng và không thân thiện. Đội phát triển cần làm gì để cải thiện trải nghiệm người dùng?*

Khi nhận được phản hồi về việc giao diện phần mềm quản lý thư viện khó sử dụng và không thân thiện, đội phát triển cần nhanh chóng tìm cách cải thiện trải nghiệm người dùng. Trước tiên, cần thu thập ý kiến chi tiết từ người dùng thông qua khảo sát, phỏng vấn hoặc quan sát trực tiếp quá trình họ sử dụng phần mềm để xác định rõ những điểm gây khó khăn. Sau đó, phân tích các vấn đề này để tìm ra nguyên nhân, có thể liên quan đến bố cục giao diện, màu sắc, phông chữ hoặc thao tác sử dụng phức tạp. Dựa trên những thông tin thu thập được, đội phát triển có thể điều chỉnh lại thiết kế theo hướng trực quan, đơn giản và dễ sử dụng hơn. Việc áp dụng các nguyên tắc UX/UI như tối ưu điều hướng, sử dụng biểu tượng trực quan, bổ sung hướng dẫn sử dụng hoặc tính năng trợ giúp cũng sẽ giúp nâng cao trải nghiệm người dùng. Trước khi triển khai phiên bản cập nhật, cần tiến hành kiểm thử với một nhóm người dùng thực tế để đảm bảo những thay đổi thực sự mang lại hiệu quả. Sau khi cập nhật phần mềm, đội ngũ hỗ trợ kỹ thuật nên tiếp tục theo dõi phản hồi để có những điều chỉnh kịp thời, giúp phần mềm ngày càng hoàn thiện hơn.

*Câu hỏi tình huống 5: Một dự án phát triển phần mềm đã vượt quá ngân sách dự kiến do thời gian hoàn thành lâu hơn kế hoạch. Là quản lý dự án, bạn sẽ đề xuất những giải pháp nào để hạn chế việc vượt ngân sách trong tương lai?*

Trả lời: Khi một dự án phần mềm vượt quá ngân sách do thời gian hoàn thành kéo dài hơn kế hoạch, cần có những giải pháp cụ thể để hạn chế tình trạng này trong tương lai. Trước hết, việc lập kế hoạch chi tiết và thực tế ngay từ đầu là rất quan trọng. Cần đánh giá kỹ lưỡng phạm vi dự án, ước lượng chính xác nguồn lực, thời gian và chi phí, đồng thời dự trù ngân sách cho các rủi ro có thể phát sinh. Tiếp theo, áp dụng các phương pháp quản lý dự án hiệu quả như Agile hoặc Scrum giúp theo dõi tiến độ liên tục, phản ứng linh hoạt trước các thay đổi và tối ưu hóa quy trình làm việc. Bên cạnh đó, cần giám sát chặt chẽ tiến độ và ngân sách trong suốt quá trình triển khai, sử dụng các công cụ quản lý để phát hiện sớm các dấu hiệu chậm trễ hoặc chi tiêu vượt mức, từ đó đưa ra các điều chỉnh kịp thời. Ngoài ra, việc tối ưu hóa nguồn lực, đảm bảo đội ngũ làm việc hiệu quả và giảm thiểu lãng phí cũng góp phần kiểm soát ngân sách. Cuối cùng, cần thường xuyên trao đổi với khách hàng để kiểm soát phạm vi yêu cầu, tránh phát sinh thêm những thay đổi lớn làm kéo dài thời gian và tăng chi phí. Bằng cách áp dụng những biện pháp này, dự án có thể được thực hiện đúng tiến độ và trong giới hạn ngân sách cho phép.

*Câu hỏi tình huống 6: Trong quá trình bảo trì phần mềm quản lý khách sạn, một nhân viên phát hiện ra một lỗi nhỏ không ảnh hưởng lớn đến hoạt động. Tuy nhiên, chi phí để sửa lỗi này khá cao. Bạn sẽ quyết định sửa lỗi hay không? Vì sao?*

Việc quyết định sửa lỗi hay không cần dựa trên mức độ ảnh hưởng của lỗi đối với hệ thống và chi phí khắc phục. Nếu lỗi không gây tác động lớn đến hoạt động của phần mềm và không ảnh hưởng đến trải nghiệm người dùng, có thể tạm thời ghi nhận và theo dõi thay vì sửa ngay lập tức, đặc biệt khi chi phí khắc phục quá cao. Tuy nhiên, cũng cần xem xét khả năng lỗi này có thể dẫn đến những vấn đề nghiêm trọng hơn trong tương lai hay không. Nếu có nguy cơ ảnh hưởng đến dữ liệu, bảo mật hoặc hiệu suất hệ thống, thì việc sửa lỗi sớm có thể giúp tránh được chi phí lớn hơn sau này. Ngoài ra, nếu phần mềm đang trong giai đoạn bảo trì có kế hoạch, có thể đợi đến lần cập nhật tiếp theo để sửa lỗi nhằm tối ưu chi phí. Vì vậy, quyết định sửa hay không cần được cân nhắc kỹ lưỡng dựa trên mức độ ảnh hưởng, tính cấp thiết và ngân sách hiện có.

*Câu hỏi tình huống 7: Khách hàng yêu cầu đội phát triển phải hoàn thành dự án sớm hơn 1 tháng so với kế hoạch ban đầu. Đội phát triển đang gặp khó khăn về nhân lực và tài nguyên. Bạn sẽ xử lý yêu cầu này như thế nào?*

Khi khách hàng yêu cầu hoàn thành dự án sớm hơn một tháng trong khi đội phát triển đang gặp khó khăn về nhân lực và tài nguyên, cần có phương án xử lý hợp lý để đảm bảo chất lượng dự án mà vẫn đáp ứng yêu cầu khách hàng. Trước tiên, cần trao đổi trực tiếp với khách hàng để hiểu rõ lý do của yêu cầu rút ngắn thời gian và đánh giá mức độ linh hoạt trong thời gian bàn giao. Nếu có thể, đề xuất một giải pháp trung gian như rút ngắn thời gian nhưng không quá một tháng để giảm áp lực cho đội ngũ. Tiếp theo, phân tích tiến độ hiện tại, xác định các nhiệm vụ quan trọng và xem xét điều chỉnh quy trình làm việc để tối ưu hóa thời gian. Nếu cần thiết, có thể cân nhắc tăng cường nhân lực bằng cách huy động nội bộ hoặc thuê thêm nhân sự tạm thời. Đồng thời, áp dụng các phương pháp làm việc hiệu quả như Agile để tăng tốc độ phát triển và kiểm thử liên tục. Nếu việc hoàn thành sớm có nguy cơ ảnh hưởng đến chất lượng phần mềm, cần thông báo rõ ràng với khách hàng về những rủi ro tiềm ẩn và đưa ra phương án như chia nhỏ dự án để bàn giao từng phần. Việc minh bạch trong trao đổi và đưa ra giải pháp hợp lý sẽ giúp cân bằng giữa yêu cầu khách hàng và khả năng thực hiện của đội phát triển.

*Câu hỏi tình huống 8: Một công ty phần mềm nhỏ nhận được dự án phát triển ứng dụng di động. Do hạn chế về nguồn lực và kinh nghiệm, công ty đã liên tục thay đổi công nghệ sử dụng trong dự án. Điều này khiến dự án bị kéo dài và chi phí tăng cao. Bạn sẽ đưa ra giải pháp gì để khắc phục?*

Để khắc phục tình trạng kéo dài tiến độ và tăng chi phí do liên tục thay đổi công nghệ trong dự án phát triển ứng dụng di động, công ty cần có chiến lược rõ ràng ngay từ đầu. Trước tiên, cần đánh giá lại yêu cầu dự án và xác định công nghệ phù hợp dựa trên nguồn lực, kinh nghiệm sẵn có và tính ổn định lâu dài. Thay vì liên tục thay đổi, công ty nên chọn một công nghệ phù hợp và cam kết duy trì trong suốt quá trình phát triển để tránh lãng phí thời gian và tài nguyên. Ngoài ra, việc nâng cao năng lực đội ngũ bằng cách tổ chức các buổi đào tạo hoặc thuê chuyên gia tư vấn sẽ giúp giảm bớt khó khăn khi áp dụng công nghệ mới. Đồng thời, công ty cần áp dụng phương pháp quản lý dự án hiệu quả, như Agile hoặc Scrum, để kiểm soát tiến độ và điều chỉnh linh hoạt mà không làm gián đoạn quá trình phát triển. Việc lập kế hoạch rõ ràng, tránh thay đổi không cần thiết và tối ưu hóa nguồn lực sẽ giúp dự án hoàn thành đúng tiến độ và tiết kiệm chi phí.

*Câu hỏi tình huống 9: Sau khi bàn giao phần mềm cho khách hàng, đội phát triển phát hiện ra một lỗi bảo mật nghiêm trọng có thể bị hacker khai thác. Là người phụ trách dự án, bạn sẽ giải quyết tình huống này như thế nào?*

Khi phát hiện một lỗi bảo mật nghiêm trọng sau khi bàn giao phần mềm cho khách hàng, cần hành động nhanh chóng và có kế hoạch xử lý rõ ràng để giảm thiểu rủi ro. Trước tiên, tôi sẽ đánh giá mức độ nghiêm trọng của lỗi, xác định phạm vi ảnh hưởng và nguy cơ bị khai thác. Nếu lỗi có thể gây thiệt hại nghiêm trọng, tôi sẽ đề xuất tạm thời vô hiệu hóa tính năng liên quan hoặc cung cấp hướng dẫn giảm thiểu rủi ro cho khách hàng trong khi đội phát triển khẩn trương triển khai bản vá. Đồng thời, cần thông báo minh bạch cho khách hàng về sự cố, phương án khắc phục và thời gian dự kiến hoàn thành để đảm bảo sự tin tưởng. Sau khi sửa lỗi, tôi sẽ yêu cầu kiểm thử bảo mật kỹ lưỡng để tránh lặp lại tình trạng tương tự. Cuối cùng, cần xem xét lại quy trình phát triển và kiểm thử bảo mật để cải thiện, đảm bảo rằng các vấn đề bảo mật được phát hiện sớm hơn trong các dự án tiếp theo.

*Câu hỏi tình huống 10: Dự án phát triển hệ thống quản lý sản xuất đã được triển khai thành công tại nhà máy. Tuy nhiên, do thay đổi quy trình sản xuất, khách hàng yêu cầu sửa đổi phần mềm để phù hợp với quy trình mới. Đội phát triển cần làm gì để đáp ứng yêu cầu này mà không làm ảnh hưởng đến hoạt động sản xuất của khách hàng?*

Để đáp ứng yêu cầu sửa đổi phần mềm theo quy trình sản xuất mới mà không ảnh hưởng đến hoạt động của khách hàng, đội phát triển cần lập kế hoạch triển khai một cách cẩn trọng. Trước tiên, cần làm việc chặt chẽ với khách hàng để hiểu rõ những thay đổi trong quy trình sản xuất và xác định các tính năng cần chỉnh sửa hoặc bổ sung. Sau đó, đội phát triển sẽ phân tích tác động của các thay đổi này đối với hệ thống hiện tại để đảm bảo không gây lỗi hoặc gián đoạn hoạt động. Một môi trường kiểm thử độc lập nên được thiết lập để thử nghiệm các tính năng mới trước khi triển khai chính thức. Ngoài ra, cần thực hiện triển khai theo từng giai đoạn hoặc vào thời điểm ít ảnh hưởng nhất đến hoạt động sản xuất, chẳng hạn ngoài giờ làm việc hoặc cuối tuần. Cuối cùng, đội phát triển nên cung cấp tài liệu hướng dẫn và hỗ trợ đào tạo để đảm bảo nhân viên nhà máy có thể sử dụng hệ thống mới một cách hiệu quả ngay sau khi cập nhật.

**CHƯƠNG 2**

**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****:* B. Workflow lấy yêu cầu**

**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:* **B. Pha làm rõ**

**Câu hỏi 3**: Mô hình CMM mức nào yêu cầu quy trình phát triển phần mềm phải

được quản lý định lượng?

*Đáp án:* **C. Mức 4**

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

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

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

*Đáp án:* **D. Workflow kiểm thử**

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

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

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

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

**Câu hỏi 8**: Trong mô hình CMM mức 5, quy trình phát triển phần mềm

có đặc điểm gì?

*Đáp án:* **A. Quy trình được cải tiến liên tục**

**Câu hỏi 9**: Workflow thiết kế bao gồm việc thực hiện hoạt động nào?

*Đáp án:* **C. Thiết kế kiến trúc và chi tiết hệ thống**

**Câu hỏi 10**: CMM viết tắt của cụm từ nào?

*Đáp án:* **B. Capability Maturity Model**

**Câu hỏi ngắn**

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

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 khởi đầu trong tiến trình thống nhất giúp nhóm phát triển phần mềm có cái nhìn rõ ràng về mục tiêu, yêu cầu và tính khả thi của dự án. Pha này cũng giúp xác định các yếu tố rủi ro và lập kế hoạch cho các giai đoạn tiếp theo, như thiết kế, triển khai và kiểm thử.

1. **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.

Kết quả: Tài liệu yêu cầu phần mềm được duyệt bởi khách hàng.

1. **Tiến trình thống nhất gồm bao nhiêu pha chính?**

4 pha: Pha khởi đầu (Inception), Pha làm rõ (Elaboration), Pha xây dựng (Construction), Pha chuyển giao (Transition)

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

CMM mức 2 – Managed (Quản lý):

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 – Defined (Định nghĩa):

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.

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

Mục tiêu: Đảm bảo rằng phần mềm hoạt động đúng như mong đợi.

Kết quả: Phần mềm đạt tiêu chuẩn chất lượng và sẵn sàng triển khai.

1. **Mô hình CMM có bao nhiêu mức?**

5 mức: 1 – Initial (Ban đầu), 2 – Managed (Quản lý), 3 – Defined (Định nghĩa), 4 – Quantitatively Managed (Quản lý định lượng), 5 – Optimizing (Tối ưu hóa)

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

Mô hình thác nước (Waterfall Model):

Đặc điểm: Phát triển tuyến tính, từng pha hoàn thành trước khi chuyển sang pha tiếp theo.

Ưu điểm: Dễ quản lý và theo dõi tiến độ.

Nhược điểm: Khó thích nghi khi yêu cầu thay đổi.

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

Đặc điểm: Phát triển phần mềm theo từng đợt lặp lại, mỗi đợt cải tiến dần phần mềm.

Ưu điểm: Giảm thiểu rủi ro và thích nghi tốt với thay đổi yêu cầu.

Nhược điểm: Tốn nhiều công sức quản lý và kiểm thử.

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

Có, tiến trình thống nhất (Unified Process) là một ví dụ của mô hình lặp và tăng trưởng.

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

Mục tiêu: Thiết kế chi tiết các thành phần phần mềm dựa trên kết quả phân tích.

Kết quả: Tài liệu thiết kế chi tiết và sơ đồ UML.

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

Mục tiêu là đạt được sự hoàn hảo 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 quy trình phát triển phần mềm, mỗi workflow đóng vai trò quan trọng nhằm đảm bảo hệ thống được xây dựng một cách hiệu quả, đáp ứng đúng yêu cầu và đạt chất lượng cao. Theo nhóm em, vai trò cụ thể của từng workflow là:

1. Workflow lấy yêu cầu: Đây là giai đoạn nền tảng, giúp xác định rõ nhu cầu của khách hàng hoặc người dùng cuối. Việc thu thập yêu cầu đầy đủ và chính xác sẽ đảm bảo phần mềm được phát triển theo đúng mục
2. Workflow phân tích: Sau khi xác định yêu cầu, bước phân tích giúp làm rõ các chức năng của hệ thống, xác định các ràng buộc kỹ thuật và đánh giá tính khả thi của dự án. Workflow này đóng vai trò quan trọng trong việc nhận diện rủi ro và đề xuất giải pháp tối ưu, tạo tiền đề cho thiết kế và triển khai hiệu quả.
3. Workflow thiết kế: Giai đoạn thiết kế xác định kiến trúc tổng thể của hệ thống, bao gồm thiết kế giao diện, mô hình dữ liệu, cấu trúc xử lý và các thành phần kỹ thuật liên quan. Một thiết kế tốt không chỉ đảm bảo hiệu suất và tính ổn định của phần mềm mà còn giúp hệ thống dễ bảo trì và mở rộng trong tương lai.
4. Workflow cài đặt: Đây là giai đoạn hiện thực hóa thiết kế thông qua việc lập trình. Các lập trình viên phát triển phần mềm theo đúng tiêu chuẩn kỹ thuật, đảm bảo mã nguồn tối ưu, dễ hiểu và có khả năng bảo trì.
5. Workflow kiểm thử: Trước khi đưa vào vận hành, phần mềm cần trải qua quá trình kiểm thử nghiêm ngặt nhằm phát hiện và khắc phục lỗi. Kiểm thử bao gồm nhiều cấp độ như kiểm thử chức năng, kiểm thử hiệu năng, kiểm thử bảo mật và kiểm thử khả năng sử dụng. Workflow này đảm bảo hệ thống hoạt động ổn định, đáp ứng yêu cầu kỹ thuật và mang lại trải nghiệm tốt cho người dùng.

Mỗi workflow trong quy trình phát triển phần mềm đều có vai trò không thể thiếu, giúp tối ưu hóa quy trình làm việc, nâng cao chất lượng sản phẩm và đảm bảo dự án được triển khai thành công.

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

1. Mô hình Vòng đời Thác nước (Waterfall Model): Mô hình Vòng đời Thác nước là phương pháp phát triển phần mềm tuyến tính, trong đó mỗi giai đoạn phải hoàn thành trước khi chuyển sang giai đoạn tiếp theo.

- Đặc điểm: Quy trình phát triển theo thứ tự chặt chẽ từ phân tích, thiết kế, cài đặt, kiểm thử đến bảo trì, không có sự quay lại hoặc lặp lại giữa các pha, phù hợp với các dự án có yêu cầu rõ ràng, ít thay đổi.

- Ưu điểm: Dễ quản lý do có quy trình cụ thể và tài liệu chi tiết, phù hợp với dự án nhỏ, yêu cầu rõ ràng ngay từ đầu.

- Nhược điểm: Thiếu linh hoạt, khó thay đổi khi yêu cầu khách hàng thay đổi, phát hiện lỗi muộn, chi phí sửa đổi cao.

2. Tiến trình Thống nhất (Unified Process - UP): Tiến trình Thống nhất là mô hình phát triển phần mềm lặp và gia tăng, tập trung vào việc giải quyết rủi ro sớm trong dự án.

- Các pha trong Tiến trình Thống nhất:

+ Pha Khởi đầu (Inception): Xác định yêu cầu cốt lõi, đánh giá tính khả thi.

+ Pha Làm rõ (Elaboration): Phân tích và thiết kế hệ thống, giải quyết rủi ro.

+ Pha Xây dựng (Construction): Lập trình, kiểm thử và phát triển phần mềm.

+ Pha Chuyển giao (Transition): Bàn giao, triển khai phần mềm cho khách hàng.

* Đặc điểm: Cho phép lặp lại các pha, có thể điều chỉnh dựa trên phản hồi, xác định và giải quyết rủi ro từ sớm, phù hợp với dự án phức tạp, yêu cầu thay đổi linh hoạt.
* Ưu điểm: Giảm thiểu rủi ro nhờ phát hiện sớm các vấn đề, tích hợp kiểm thử sớm, giúp phát hiện lỗi kịp thời, phù hợp với dự án có yêu cầu thay đổi liên tục.
* Nhược điểm: Yêu cầu tài nguyên và kỹ năng quản lý cao, quá trình phát triển phức tạp, khó thực hiện đối với nhóm nhỏ.

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

Mô hình lặp là một phương pháp tiếp cận mạnh mẽ trong phát triển phần mềm, nhấn mạnh vào cải tiến liên tục và khả năng thích ứng. Mô hình này cho phép các nhóm phát triển phần mềm theo từng bước nhỏ, dễ quản lý, có thể mang lại một số lợi thế:

* Tính linh hoạt được nâng cao: Một trong những lợi ích chính của mô hình lặp là tính linh hoạt của nó. Khi các yêu cầu phát triển, các nhóm có thể điều chỉnh kế hoạch và ưu tiên của mình dựa trên phản hồi từ các bên liên quan. Khả năng thích ứng này rất quan trọng trong môi trường phát triển nhanh như hiện nay, nơi nhu cầu của người dùng có thể thay đổi nhanh chóng.
* Phát hiện sớm các vấn đề: Bằng cách chia nhỏ dự án thành các lần lặp, các nhóm có thể xác định và giải quyết các vấn đề ngay từ đầu trong quá trình phát triển. Mỗi lần lặp bao gồm thử nghiệm và phản hồi, giúp phát hiện lỗi trước khi chúng trở thành vấn đề lớn hơn. Cách tiếp cận chủ động này có thể tiết kiệm thời gian và tài nguyên trong thời gian dài.
* Cải thiện sự tham gia của người dùng: Mô hình lặp lại khuyến khích tương tác thường xuyên với người dùng, cho phép họ cung cấp phản hồi về mỗi lần gia tăng. Sự tham gia này không chỉ giúp tinh chỉnh sản phẩm mà còn đảm bảo rằng đầu ra cuối cùng phù hợp chặt chẽ với kỳ vọng của người dùng. Người dùng cảm thấy tham gia nhiều hơn vào quá trình phát triển, điều này có thể dẫn đến sự hài lòng cao hơn với sản phẩm cuối cùng.
* Chuyển giao từng phần: Với mô hình lặp lại, các nhóm có thể cung cấp các thành phần chức năng của phần mềm vào cuối mỗi lần lặp lại. Việc cung cấp gia tăng này cho phép người dùng bắt đầu sử dụng các phần của hệ thống sớm hơn, cung cấp giá trị ngay từ đầu vòng đời dự án. Nó cũng cho phép các nhóm thu thập phản hồi của người dùng về các thành phần này, có thể cung cấp thông tin cho các lần lặp lại trong tương lai.
* Quản lý rủi ro: Phương pháp lặp lại giúp quản lý rủi ro hiệu quả. Bằng cách đánh giá dự án vào cuối mỗi lần lặp lại, các nhóm có thể đánh giá lại các chiến lược của mình và thực hiện các điều chỉnh cần thiết. Đánh giá liên tục này giúp giảm thiểu rủi ro liên quan đến các yêu cầu thay đổi hoặc các thách thức không lường trước được.
* Cải tiến liên tục: Mỗi lần lặp lại đóng vai trò là một cơ hội học tập. Các nhóm có thể phản ánh về quy trình và kết quả của mình, xác định các lĩnh vực cần cải thiện. Văn hóa cải tiến liên tục này thúc đẩy sự đổi mới và nâng cao chất lượng chung của phần mềm đang được phát triển.

Tuy nhiên cũng tồn tại một số nhược điểm của mô hình lặp và gia tăng:

* Phạm vi mở rộng : Tính linh hoạt của mô hình lặp có thể dẫn đến phạm vi mở rộng nếu những thay đổi không được quản lý hiệu quả.
* Tốn nhiều tài nguyên : Việc lặp lại thường xuyên có thể đòi hỏi nhiều tài nguyên và thời gian hơn, đặc biệt là nếu liên quan đến việc thử nghiệm mở rộng

Dù có một số hạn chế, mô hình lặp vẫn là một phương pháp phổ biến trong phát triển phần mềm hiện đại, giúp đảm bảo chất lượng và tối ưu trải nghiệm người dùng.

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

* CMM (Capability Maturity Model) là một mô hình đánh giá và trưởng thành năng lực, được phát triển bởi Viện Kỹ nghệ Phần mềm (SEI) của Đại học Carnegie Mellon.
* Mô hình này cung cấp một khung tham chiếu để đánh giá mức độ trưởng thành của quy trình phát triển phần mềm của một tổ chức.
* CMM giúp các tổ chức xác định các điểm mạnh và điểm yếu trong quy trình của mình, từ đó đưa ra các biện pháp cải tiến phù hợp.
* Việc áp dụng CMM giúp các tổ chức nâng cao chất lượng phần mềm, giảm thiểu rủi ro và tăng cường hiệu quả hoạt động.
* CMM cũng giúp các tổ chức tạo ra một môi trường làm việc chuyên nghiệp và có tổ chức hơn.

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

* \*\*Tốn kém chi phí và thời gian:\*\* Việc đánh giá và triển khai CMM đòi hỏi đầu tư đáng kể về thời gian và nguồn lực.
* **Yêu cầu sự thay đổi văn hóa:** CMM yêu cầu các tổ chức thay đổi cách làm việc truyền thống và áp dụng các quy trình mới, điều này có thể gặp phải sự kháng cự từ nhân viên.
* **Khó khăn trong việc đo lường:** Việc đo lường hiệu quả của các cải tiến theo CMM có thể gặp khó khăn, đặc biệt là trong các tổ chức nhỏ.
* **Cần sự cam kết từ lãnh đạo:** Việc áp dụng CMM thành công đòi hỏi sự cam kết và hỗ trợ mạnh mẽ từ lãnh đạo cấp cao.
* **Không phù hợp với mọi tổ chức:** CMM được thiết kế cho các tổ chức lớn và có quy trình phức tạp, có thể không phù hợp với các tổ chức nhỏ hoặc các dự án đơn giản.

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

* **Áp dụng phương pháp Agile:** Agile là một phương pháp phát triển phần mềm linh hoạt, giúp tăng cường sự hợp tác, phản hồi nhanh và khả năng thích ứng với thay đổi.
* **Tự động hóa quy trình:** Tự động hóa các tác vụ lặp đi lặp lại giúp giảm thiểu sai sót và tăng hiệu quả.
* **Tăng cường giao tiếp và hợp tác:** Tạo môi trường làm việc cởi mở, khuyến khích giao tiếp và chia sẻ thông tin giữa các thành viên trong nhóm.
* **Đào tạo và phát triển nhân viên:** Đầu tư vào đào tạo và phát triển kỹ năng cho nhân viên để họ có thể áp dụng các quy trình và công nghệ mới.
* **Sử dụng công cụ hỗ trợ:** Sử dụng các công cụ quản lý dự án, kiểm soát phiên bản và kiểm thử tự động để nâng cao hiệu quả.
* **Đo lường và đánh giá thường xuyên:** Thiết lập các chỉ số đo lường hiệu quả và thực hiện đánh giá thường xuyên để xác định các lĩnh vực cần cải tiến.
* **Học hỏi từ kinh nghiệm:** Tổ chức các buổi họp rút kinh nghiệm sau mỗi dự án để chia sẻ bài học và cải tiến quy trình.

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

Việc áp dụng tiến trình thông nhất (Unified Process) trong các dự án lớn mang lại nhiều ưu điểm, như:

* **Giảm rủi ro:** Phân chia công việc thành các giai đoạn nhỏ, cho phép phát hiện và xử lý vấn đề sớm.
* **Hỗ trợ làm việc nhóm:** Tăng cường phối hợp giữa các thành viên nhờ quy trình chuẩn hóa.
* **Tái sử dụng:** Khuyến khích sử dụng các thành phần đã có sẵn, tiết kiệm thời gian và chi phí.
* **Tập trung vào yêu cầu:** Đảm bảo hiểu rõ và đáp ứng tốt nhu cầu của khách hàng thông qua các vòng lặp phát triển.
* **Tính cấu trúc rõ ràng:** Tiến trình được chia thành các pha (khởi tạo, chuẩn bị, xây dựng, triển khai), giúp quản lý dự án dễ dàng hơn.

**8 Thảo luận về sự cần thiết của việc kiểm thử trong tiến trình chuẩn bị thông nhất.**  
Việc kiểm thử trong pha chuẩn bị của tiến trình thông nhất rất cần thiết vì:

* **Đảm bảo chất lượng sớm:** Phát hiện lỗi hoặc vấn đề ngay từ đầu, tránh chi phí sửa chữa lớn ở các pha sau.
* **Lập kế hoạch hiệu quả:** Kiểm thử giúp xác định các rủi ro tiềm ẩn, từ đó xây dựng kế hoạch kiểm thử chi tiết cho các pha sau.
* **Tăng độ tin cậy:** Giúp đội phát triển tự tin hơn khi chuyển sang pha xây dựng với cơ sở vững chắc.
* **Xác minh yêu cầu:** Đảm bảo rằng các yêu cầu ban đầu được hiểu đúng và có thể triển khai được.

**9 So sánh mô hình CMMI mức 4 và mức 5**

* **CMMI mức 4 (Quản lý định lượng):**
  + Quản lý hiệu suất dự án thông qua các chỉ số cụ thể, nhưng vẫn có thể có biến động nhỏ.
  + Tập trung vào đo lường và kiểm soát quy trình bằng dữ liệu định lượng.
  + Yêu cầu tổ chức có khả năng thu thập và phân tích dữ liệu để cải tiến liên tục.
* **CMMI mức 5 (Tối ưu hóa):**
  + Cao hơn mức 4, tập trung vào tối ưu hóa quy trình thông qua cải tiến liên tục dựa trên dữ liệu.
  + Có khả năng dự đoán và ngăn ngừa vấn đề trước khi xảy ra nhờ phân tích sâu hơn.
  + Yêu cầu tổ chức đã đạt được tính ổn định cao và áp dụng các phương pháp tiên tiến.  
    Sự khác biệt chính nằm ở mức độ tự động hóa, dự đoán và tối ưu hóa quy trình, với mức 5 vượt trội hơn về tính tiên tiến và hiệu quả.

**10 Đề xuất cách tối ưu hoạt động nhóm trong workflow lập yêu cầu.**

 **Rõ ràng hóa vai trò:** Phân công nhiệm vụ cụ thể cho từng thành viên, đảm bảo ai cũng hiểu trách nhiệm của mình.

 **Sử dụng công cụ quản lý:** Áp dụng các phần mềm như Jira, Trello hoặc Confluence để theo dõi tiến độ và giao tiếp hiệu quả.

 **Tổ chức họp định kỳ:** Thực hiện các buổi họp ngắn (stand-up meeting) để cập nhật tình hình, giải quyết vấn đề nhanh chóng.

 **Đào tạo và giao tiếp:** Đảm bảo tất cả thành viên hiểu rõ quy trình và yêu cầu, giảm thiểu hiểu lầm.

 **Phân tích phản hồi:** Thu thập ý kiến từ khách hàng và đội ngũ để cải tiến cách làm việc, tăng hiệu suất.

**Câu hỏi tình huống**

1. **Mỗi công ty phát triển phần mềm đều khó khăn khi yêu cầu của khách hàng liên tục thay đổi trong pha xây dựng. Dối phát triển nên làm gì để giải quyết vấn đề này?**  
   Để giải quyết vấn đề này, đội phát triển nên áp dụng phương pháp linh hoạt như Agile, tập trung vào việc giao tiếp thường xuyên với khách hàng, sử dụng các cuộc họp định kỳ (như sprint review) để cập nhật yêu cầu, và duy trì tính linh hoạt trong quy trình phát triển. Điều này giúp điều chỉnh nhanh chóng khi có thay đổi.
2. **Trong pha chuẩn bị của tiến trình thông nhất, khách hàng yêu cầu bổ sung thêm tính năng mới. Đội phát triển nên xử lý ra sao?**  
   Đội phát triển nên đánh giá tác động của tính năng mới đến thời gian, chi phí và phạm vi dự án. Sau đó, thảo luận với khách hàng để ưu tiên tính năng, có thể điều chỉnh kế hoạch hoặc đưa ra lộ trình bổ sung trong các giai đoạn sau nếu cần.
3. **Dự án phát triển phần mềm bị trì hoãn do lỗi phát sinh liên tục trong quá trình kiểm thử. Làm trường dự án, bạn sẽ làm gì?**  
   Là trưởng dự án, bạn nên phân tích nguyên nhân gốc rễ của các lỗi, cải thiện quy trình kiểm thử (như tăng cường kiểm thử tự động hoặc kiểm thử sớm hơn), phân bổ lại nguồn lực nếu cần, và phối hợp chặt chẽ với đội phát triển để giảm thiểu lỗi trong tương lai.
4. **Trong workflow thiệt 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?**  
   Đội phát triển nên đánh giá lợi ích và rủi ro của thay đổi thiết kế, tham khảo ý kiến các bên liên quan (bao gồm khách hàng nếu cần), và nếu thay đổi hợp lý, lập kế hoạch cập nhật mà không làm gián đoạn tiến độ tổng thể.
5. **Khách hàng yêu cầu rút ngắn thời gian phát triển dự án mà không thay đổi yêu cầu. Đội phát triển nên phân ứng ra sao?**  
   Đội phát triển cần làm rõ tính khả thi, có thể đề xuất tăng cường nguồn lực, áp dụng các công cụ tự động hóa, hoặc ưu tiên các tính năng quan trọng nhất. Tuy nhiên, cần cảnh báo khách hàng về rủi ro chất lượng nếu thời gian bị rút ngắn quá mức.
6. **Mỗi công ty muốn áp dụng mô hình CMMI nhưng gặp khó khăn do thiếu nguồn lực. Hãy đề xuất giải pháp.**  
   Giải pháp có thể bao gồm bắt đầu từ cấp độ CMMI cơ bản (Level 1 hoặc 2), đào tạo nhân sự, sử dụng tư vấn chuyên nghiệp, và triển khai dần dần theo khả năng tài chính và nhân lực của công ty.
7. **Trong workflow lập yêu cầu, khách hàng liên tục cập thông tin không rõ ràng. Đội phát triển có vấn đề gì?**  
   Vấn đề chính là sự thiếu nhất quán và hiểu biết rõ ràng về yêu cầu, dẫn đến khó khăn trong thiết kế và phát triển. Đội cần tổ chức các buổi làm việc trực tiếp hoặc sử dụng tài liệu chi tiết hơn để xác nhận yêu cầu với khách hàng.
8. **Mất dự án gấp rút ra cao trong pha kiểm tra do thiếu tài liệu yêu cầu rõ ràng. Đội phát triển nên làm gì?**  
   Đội nên ưu tiên làm rõ yêu cầu với khách hàng ngay lập tức, sử dụng các công cụ quản lý yêu cầu (như Jira hoặc Confluence), và tăng cường kiểm tra để đảm bảo không xảy ra lỗi nghiêm trọng trước khi phát hành.
9. **Dự án phần mềm tồn cơ nhiều lỗi phát triển ở các giai đoạn khác nhau. Làm thế nào để đảm bảo các lỗi phổ biến không tái diễn?**  
   Để đảm bảo không tái diễn lỗi, đội có thể áp dụng quy trình kiểm tra nghiêm ngặt hơn, sử dụng các công cụ kiểm tra tự động, tổ chức các buổi phân tích sau dự án (post-mortem) để rút kinh nghiệm, và cải tiến quy trình phát triển.
10. **Mỗi 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.**  
    Giải pháp bao gồm xây dựng một quy trình chuẩn hóa (như sử dụng mô hình Agile, Scrum, hoặc CMMI), đào tạo nhân viên về quy trình, sử dụng phần mềm quản lý dự án (như Trello, Asana), và thường xuyên đánh giá, cải tiến quy trình.

CHƯƠNG 3

**Trắc nghiệm**

**Câu hỏi 1**: Pha nào trong mô hình lý thuyết vòng đời phát triển phần mềm chịu trách nhiệm chuyển đổi yêu cầu thành đặc tả kỹ thuật?

Đáp án: **B. Pha lấy yêu cầu**

**Câu hỏi 2**: Mô hình vòng đời nào phát triển phần mềm bằng cách tạo các phiên bản nhỏ và tăng dần tính năng?

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

**Câu hỏi 3**: Pha bảo trì trong vòng đời phát triển phần mềm bao gồm hoạt động nào?

Đáp án: B. Sửa lỗi và cập nhật tính năng mới

**Câu hỏi 4**: Mô hình thác nước phù hợp nhất với loại dự án nào?

Đáp án: **B. Dự án có yêu cầu rõ ràng và ít thay đổi**

**Câu hỏi 5**: Trong mô hình xoắn ốc, mỗi vòng xoắn tương ứng với:

Đáp án: **D. Một lần phân tích rủi ro**

Câu hỏi 6: Điểm yếu lớn nhất của mô hình xây à sửa là gì?

Đáp án: **C. Khó kiểm soát chất lượng**

**Câu hỏi 7**: Mô hình nào tập trung vào việc tạo các nguyên mẫu nhanh để thu thập phản hồi từ khách hàng?

Đáp án: **B. Mô hình bản mẫu nhanh**

**Câu hỏi 8**: Pha nào kết thúc vòng đời phát triển phần mềm?

Đáp án: **C. Pha giải thể**

Câu hỏi 9: Điểm khác biệt chính giữa mô hình lặp và tăng trưởng với mô hình thác nước là gì?

Đáp án: **B. Mô hình lặp và tăng trưởng phát triển theo từng đợt nhỏ**

**Câu hỏi 10**: Mô hình nào có khả năng thích nghi tốt nhất với sự thay đổi của yêu cầu khách hàng?

Đáp án: **D. Mô hình tiến trình linh hoạt**

**Câu hỏi ngắn**

1. Pha lấy yêu cầu là gì và có vai trò gì trong vòng đời phát triển phần mềm?

* **Pha lấy yêu cầu** là giai đoạn đầu tiên trong vòng đời phát triển phần mềm (SDLC). Nó tập trung vào việc thu thập, phân tích và xác định các yêu cầu từ khách hàng hoặc người dùng cuối.
* **Vai trò:**
* Xác định rõ những gì phần mềm cần làm.
* Tránh hiểu sai hoặc thiếu sót yêu cầu.
* Làm cơ sở cho thiết kế và phát triển phần mềm sau này.

2. Mô hình thác nước hoạt động như thế nào?

* Mô hình thác nước (Waterfall) là phương pháp phát triển phần mềm tuần tự, gồm các giai đoạn:

1. **Lấy yêu cầu** → 2. **Phân tích** → 3. **Thiết kế** → 4. **Lập trình** → 5. **Kiểm thử** → 6. **Triển khai** → 7. **Bảo trì**

* Mỗi giai đoạn hoàn thành trước khi chuyển sang giai đoạn tiếp theo.
* Phù hợp với dự án có yêu cầu rõ ràng và ít thay đổi.

3. Mô hình lặp và tăng trưởng khác gì so với mô hình thác nước?

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Mô hình thác nước (Waterfall Model) | Mô hình lặp và tăng trưởng |
| Cách phát triển | Theo từng bước, tuần tự | Phát triển theo từng đợt nhỏ |
| Khả năng thay đổi | Khó thay đổi yêu cầu | Dễ dàng thay đổi yêu cầu |
| Kiểm thử | Chỉ thực hiện ở cuối | Kiểm thử ở mỗi vòng lặp |
| Rủi ro | Rủi ro cao nếu yêu cầu sai ngày từ đầu | Giảm rủi ro nhờ phản hồi liên tục |

4. Mục tiêu của pha bảo trì là gì?

Đảm bảo phần mềm tiếp tục hoạt động ổn định và đáp ứng các yêu cầu mới. Sửa lỗi phần mềm, Cải tiến phần mềm theo yêu cầu và điều chỉnh phần mềm để tương thích với môi trường mới.

5. Mô hình xây và sửa có nhược điểm gì?

* Thiếu kế hoạch và tài liệu chi tiết.
* Khó bảo trì và nâng cấp.
* Khó kiểm soát chất lượng.
* Dễ tốn kém chi phí bảo trì.
* Không phù hợp với dự án lớn, phức tạp.

6. Mô hình bản mẫu nhanh là gì?

* Là phương pháp phát triển phần mềm bằng cách tạo nguyên mẫu (prototype) nhanh chóng để thu thập phản hồi từ khách hàng.
* Sau khi nhận phản hồi, nguyên mẫu được chỉnh sửa và phát triển thành sản phẩm cuối cùng.
* Giúp giảm rủi ro do hiểu sai yêu cầu.

7. Pha giải thể là gì?

* Pha giải thể là giai đoạn cuối cùng của vòng đời phần mềm khi phần mềm không còn được sử dụng.
* Hoạt động chính:
* Gỡ bỏ phần mềm khỏi hệ thống.
* Lưu trữ tài liệu và dữ liệu cần thiết.
* Đảm bảo dữ liệu được bảo mật khi giải thể phần mềm.

8. Mô hình xoắn ốc là gì?

* Là mô hình kết hợp giữa mô hình thác nước và lặp, mỗi vòng xoắn là một pha phát triển.
* Gồm 4 pha trong mỗi vòng xoắn:

1. Xác định yêu cầu
2. Phân tích rủi ro
3. Phát triển và kiểm thử
4. Đánh giá và lập kế hoạch cho vòng tiếp theo

* Phù hợp với dự án lớn, phức tạp và yêu cầu linh hoạt

9. Tại sao mô hình tiến trình linh hoạt được đánh giá cao?

* Dễ thích nghi với thay đổi yêu cầu khách hàng.
* Phát triển nhanh hơn nhờ quy trình lặp.
* Liên tục kiểm thử và phản hồi, giảm rủi ro.
* Tăng tương tác giữa nhóm phát triển và khách hàng.

10. Điểm khác biệt chính giữa mô hình mã nguồn mở và các mô hình khác là gì?

* **Mở cho cộng đồng:** Mọi người có thể xem, sửa đổi và đóng góp vào mã nguồn.
* **Phát triển phi tập trung:** Không thuộc sở hữu của một công ty duy nhất.
* **Cập nhật và cải tiến liên tục** nhờ sự đóng góp của nhiều lập trình viên trên toàn thế giới.
* **Thường miễn phí** hoặc có giấy phép mở.

Câu hỏi thảo luận nhóm

**Câu 1.So sánh ưu và nhược điểm của mô hình thác nước và mô hình xoắn ốc.**

1. Mô hình Thác nước (Waterfall Model)

Ưu điểm

* Dễ hiểu, dễ quản lý:
* Mô hình thác nước có trình tự rõ ràng theo từng giai đoạn: Phân tích yêu cầu → Thiết kế → Triển khai → Kiểm thử → Bảo trì.
* Thích hợp cho các dự án có yêu cầu cố định, ít thay đổi.
* Tài liệu đầy đủ:
* Mỗi giai đoạn đều có tài liệu chi tiết, giúp việc bảo trì và mở rộng dễ dàng.
* Kiểm soát tốt tiến độ:
* Vì từng giai đoạn hoàn tất trước khi sang bước tiếp theo nên dễ theo dõi tiến độ và quản lý thời gian.

Nhược điểm

* Không linh hoạt khi có thay đổi:
* Nếu yêu cầu thay đổi sau khi đã hoàn thành một giai đoạn, việc quay lại để chỉnh sửa rất tốn kém.
* Không phù hợp với dự án phức tạp:
* Nếu dự án có nhiều rủi ro hoặc chưa xác định đầy đủ yêu cầu ngay từ đầu, mô hình này dễ thất bại.
* Khó khăn trong việc kiểm thử sớm:
* Kiểm thử chỉ diễn ra sau khi hoàn thành toàn bộ quá trình phát triển, dẫn đến việc phát hiện lỗi chậm.

2. Mô hình Xoắn ốc (Spiral Model)

Ưu điểm

* Linh hoạt, phù hợp với dự án lớn, phức tạp:
* Mô hình cho phép quay lại các bước trước đó để chỉnh sửa khi cần, giúp thích ứng với những thay đổi trong yêu cầu.
* Giảm thiểu rủi ro:
* Tập trung vào đánh giá rủi ro ở mỗi vòng xoắn, giúp phát hiện và giảm thiểu các vấn đề sớm.
* Có thể kiểm thử sớm:
* Kiểm thử được thực hiện ngay từ các vòng xoắn đầu tiên, giúp cải thiện chất lượng sản phẩm.
* Thích hợp cho phát triển phần mềm dài hạn:
* Phù hợp với các dự án có yêu cầu chưa rõ ràng ngay từ đầu hoặc cần cập nhật liên tục.

Nhược điểm

* Chi phí cao, mất nhiều thời gian:
* Do phải thực hiện nhiều vòng lặp, mô hình này có thể làm tăng thời gian và chi phí phát triển.
* Quản lý phức tạp:
* Việc theo dõi tiến độ và quản lý tài liệu phức tạp hơn so với mô hình thác nước.
* Phụ thuộc vào khả năng phân tích rủi ro:
* Nếu nhóm phát triển không có kinh nghiệm trong việc đánh giá rủi ro, mô hình này có thể không hiệu quả.

**Câu 2.Thảo luận về tình huống thực tế có thể áp dụng mô hình lặp và tăng trưởng.**

1. Phát triển ứng dụng thương mại điện tử

Tình huống

Giả sử một công ty khởi nghiệp muốn xây dựng một website bán hàng trực tuyến nhưng không có đủ thời gian và ngân sách để phát triển toàn bộ hệ thống ngay từ đầu.

Áp dụng mô hình Lặp và Tăng trưởng

* Giai đoạn 1: Phát triển phiên bản đầu tiên với các tính năng cơ bản như đăng ký tài khoản, hiển thị danh mục sản phẩm, giỏ hang
* Giai đoạn 2: Thu thập phản hồi từ người dùng và bổ sung các tính năng như thanh toán trực tuyến, theo dõi đơn hàng.
* Giai đoạn 3: Tích hợp hệ thống đánh giá sản phẩm, chat hỗ trợ khách hàng, tối ưu hiệu suất.
* Giai đoạn 4: Phát triển ứng dụng di động dựa trên phiên bản web đã hoàn thiện.

Lợi ích:

* Có thể nhanh chóng đưa sản phẩm ra thị trường, tạo doanh thu sớm.
* Dễ dàng điều chỉnh và mở rộng theo nhu cầu thực tế.
* Giảm rủi ro so với phát triển toàn bộ hệ thống một lần.

2. Phát triển phần mềm quản lý bệnh viện

Tình huống

Một bệnh viện muốn triển khai một hệ thống quản lý hồ sơ bệnh nhân, nhưng hệ thống phải đáp ứng nhiều phòng ban khác nhau như khoa khám bệnh, khoa xét nghiệm, khoa dược...

Áp dụng mô hình Lặp và Tăng trưởng

* Giai đoạn 1: Phát triển chức năng quản lý hồ sơ bệnh nhân cơ bản.
* Giai đoạn 2: Tích hợp hệ thống đặt lịch hẹn trực tuyến.
* Giai đoạn 3: Kết nối với hệ thống xét nghiệm và chẩn đoán hình ảnh.
* Giai đoạn 4: Hoàn thiện hệ thống quản lý thuốc và đơn thuốc điện tử.

Lợi ích:

* Bệnh viện có thể sử dụng hệ thống ngay từ các giai đoạn đầu.
* Các phòng ban có thể thích ứng dần với hệ thống mới.
* Giảm thiểu rủi ro và chi phí nếu có thay đổi trong yêu cầu.

**Câu 3.Tại sao mô hình xây và sửa không phù hợp với các dự án lớn?**

Tại sao mô hình Xây và Sửa (Build and Fix) không phù hợp với các dự án lớn?

Mô hình Xây và Sửa (Build and Fix) là cách tiếp cận đơn giản nhất trong phát triển phần mềm: lập trình viên viết mã, thử nghiệm, sửa lỗi và lặp lại quy trình này mà không có kế hoạch cụ thể. Mặc dù mô hình này có thể phù hợp với các dự án nhỏ, nhưng nó không thích hợp cho các dự án lớn do những lý do sau:

1. Thiếu kế hoạch và tài liệu chi tiết

* Dự án lớn thường có yêu cầu phức tạp, liên quan đến nhiều nhóm phát triển, khách hàng và bên liên quan.
* Mô hình Xây và Sửa không có giai đoạn phân tích yêu cầu, thiết kế hay tài liệu đầy đủ, dẫn đến khó khăn khi mở rộng và bảo trì hệ thống.

Ví dụ: Nếu một công ty phần mềm lớn phát triển hệ thống ngân hàng mà không có kế hoạch chi tiết, việc sửa lỗi trong tương lai có thể mất rất nhiều thời gian và chi phí.

2. Tốn kém chi phí và thời gian về lâu dài

* Ban đầu, mô hình này có vẻ nhanh chóng vì không cần lập kế hoạch, nhưng việc sửa lỗi liên tục và không có cấu trúc rõ ràng sẽ làm tăng chi phí phát triển.
* Các dự án lớn có thể mất nhiều năm để hoàn thành, nếu cứ sửa chữa mà không có lộ trình cụ thể, chi phí có thể vượt xa ngân sách dự kiến.

Ví dụ: Một công ty phát triển một hệ thống ERP (quản lý doanh nghiệp) mà không có kiến trúc rõ ràng sẽ dễ gặp lỗi khi tích hợp các bộ phận như kế toán, nhân sự, kho hàng, khiến chi phí sửa đổi ngày càng cao.

3. Khó kiểm soát chất lượng và bảo trì

* Do không có tài liệu và quy trình chuẩn, mỗi lần sửa lỗi có thể tạo ra lỗi mới mà không ai dự đoán được.
* Các dự án lớn cần cập nhật và bảo trì trong thời gian dài, nhưng mô hình Xây và Sửa không cung cấp tài liệu chi tiết, làm cho việc nâng cấp trở nên khó khăn.

Ví dụ: Nếu một công ty công nghệ muốn nâng cấp một hệ thống quản lý bệnh viện mà trước đó được phát triển theo mô hình Xây và Sửa, thiếu tài liệu và cấu trúc rõ ràng sẽ khiến việc bảo trì trở thành ác mộng.

4. Không phù hợp với làm việc nhóm và dự án có quy mô lớn

* Các dự án lớn thường có nhiều nhóm làm việc song song, nhưng mô hình Xây và Sửa không có quy trình chuẩn để phân chia công việc và phối hợp giữa các nhóm.
* Điều này dễ dẫn đến xung đột mã nguồn, trùng lặp công việc hoặc lỗi không đồng bộ giữa các phần khác nhau của phần mềm.

Ví dụ: Khi phát triển một hệ thống đặt vé máy bay trực tuyến, nếu không có quy trình rõ ràng, nhóm làm giao diện và nhóm làm hệ thống đặt vé có thể không đồng bộ, gây lỗi nghiêm trọng.

5. Dễ gặp rủi ro và không có khả năng dự đoán tiến độ

* Không có kế hoạch cụ thể nên rất khó ước tính thời gian hoàn thành dự án.
* Nếu yêu cầu thay đổi hoặc phát sinh lỗi lớn, nhóm phát triển không có cách nào để quay lại phiên bản trước mà không làm lại từ đầu.

Ví dụ: Nếu một công ty khởi nghiệp phát triển một ứng dụng fintech mà không có kế hoạch rõ ràng, khi quy mô người dùng tăng lên, hệ thống có thể không đáp ứng nổi, dẫn đến việc phải xây dựng lại từ đầu.

**Câu 4.So sánh giữa mô hình bản mẫu nhanh và mô hình tiến trình linh hoạt.**

|  |  |  |
| --- | --- | --- |
| Tiêu chí | Mô hình Bản Mẫu Nhanh (Rapid Prototyping) | Mô hình Tiến Trình Linh Hoạt (Agile Process Model) |
| Mục tiêu chính | Xây dựng bản mẫu nhanh để khách hàng đánh giá và phản hồi trước khi phát triển phần mềm hoàn chỉnh. | Chia dự án thành nhiều chu kỳ nhỏ (iteration/sprint) để có thể liên tục thích nghi với thay đổi trong quá trình phát triển. |
| Cách tiếp cận | Tạo ra bản mẫu sơ bộ, thu thập phản hồi từ khách hàng, sau đó mới bắt đầu phát triển sản phẩm chính thức. | Phát triển theo từng chu kỳ ngắn, với mỗi chu kỳ là một phần nhỏ của sản phẩm, liên tục cập nhật và cải tiến theo phản hồi của khách hàng. |
| Quy trình thực hiện | - Xác định yêu cầu sơ bộ.  - Xây dựng bản mẫu nhanh.  - Thu thập phản hồi từ khách hàng.  - Cải thiện bản mẫu dựa trên phản hồi.  - Phát triển phần mềm chính thức. | - Xác định yêu cầu cho từng giai đoạn ngắn (sprint).  - Phát triển một phần của sản phẩm.  - Kiểm thử, nhận phản hồi từ khách hàng.  - Tiếp tục phát triển trong sprint tiếp theo. |
| Khả năng thay đổi yêu cầu | Cho phép thay đổi trong giai đoạn tạo bản mẫu, nhưng sau khi xác định yêu cầu cuối cùng thì khó thay đổi hơn. | Linh hoạt cao, có thể thay đổi yêu cầu ở bất kỳ giai đoạn nào. |
| Sự tham gia của khách hàng | Khách hàng chỉ tham gia trong giai đoạn đánh giá bản mẫu. | Khách hàng tham gia xuyên suốt toàn bộ quá trình phát triển. |
| Tốc độ phát triển | Nhanh chóng tạo ra bản mẫu, nhưng tổng thời gian phát triển có thể kéo dài nếu có nhiều lần chỉnh sửa. | Tốc độ phát triển nhanh do phần mềm được xây dựng và cập nhật theo từng sprint nhỏ. |
| Chất lượng sản phẩm cuối cùng | Nếu bản mẫu không được thiết kế tốt, có thể dẫn đến lỗi hoặc sản phẩm không đạt kỳ vọng. | Chất lượng cao hơn do sản phẩm được cải tiến liên tục sau mỗi sprint. |
| Ứng dụng thực tế | Phù hợp với dự án chưa rõ ràng yêu cầu, cần kiểm tra ý tưởng trước khi phát triển chính thức, ví dụ: phát triển giao diện website, ứng dụng thử nghiệm (MVP). | Phù hợp với dự án phức tạp, yêu cầu thay đổi liên tục, như phần mềm doanh nghiệp, ứng dụng thương mại điện tử, phần mềm dịch vụ. |

**Câu 5.Phân tích vai trò của quản lý rủi ro trong mô hình xoắn ốc.**

Mô hình xoắn ốc (Spiral Model) là một mô hình phát triển phần mềm tập trung mạnh vào quản lý rủi ro. Nó kết hợp các yếu tố của mô hình thác nước và mô hình lặp để giảm thiểu rủi ro trong suốt vòng đời phát triển phần mềm.

Quản lý rủi ro trong mô hình xoắn ốc

Mô hình xoắn ốc bao gồm bốn giai đoạn chính trong mỗi vòng lặp:

1. Xác định yêu cầu và mục tiêu
   * Thu thập yêu cầu từ khách hàng.
   * Xác định các mục tiêu và ràng buộc của hệ thống.
2. Phân tích và đánh giá rủi ro
   * Xác định các rủi ro tiềm ẩn (thay đổi yêu cầu, lỗi công nghệ, chi phí vượt ngân sách, tiến độ chậm, v.v.).
   * Lập kế hoạch giảm thiểu rủi ro (tạo bản mẫu, sử dụng công nghệ đáng tin cậy hơn, dự phòng tài nguyên).
   * Nếu rủi ro quá cao, dự án có thể bị hủy hoặc điều chỉnh hướng đi.
3. Phát triển và kiểm thử
   * Lập trình, kiểm thử và triển khai một phần của hệ thống.
   * Kiểm tra hiệu quả của các biện pháp giảm thiểu rủi ro.
4. Đánh giá và lập kế hoạch cho vòng tiếp theo
   * Đánh giá sản phẩm tạm thời, phản hồi từ khách hàng.
   * Tiếp tục chu kỳ xoắn ốc với các tính năng mới hoặc điều chỉnh cần thiết.

Vai trò quan trọng của quản lý rủi ro trong mô hình xoắn ốc:

|  |  |
| --- | --- |
| Vai trò | Ý nghĩa |
| Xác định rủi ro sớm | Giúp phát hiện sớm các nguy cơ có thể ảnh hưởng đến tiến độ và chất lượng sản phẩm. |
| Giảm thiểu chi phí sửa đổi | Nếu rủi ro được phát hiện và khắc phục sớm, chi phí sửa lỗi sẽ thấp hơn so với phát hiện trễ trong giai đoạn triển khai. |
| Cải thiện chất lượng sản phẩm | Kiểm soát rủi ro giúp đảm bảo phần mềm đạt tiêu chuẩn cao và ít lỗi hơn. |
| |  | | --- | | Tối ưu hóa tài nguyên |  |  | | --- | |  | | Dự đoán và lên kế hoạch cho các tình huống rủi ro giúp sử dụng hiệu quả nhân lực, tài chính và thời gian. |
| Linh hoạt với thay đổi | Nếu có thay đổi yêu cầu hoặc công nghệ, mô hình có thể điều chỉnh kế hoạch mà không làm ảnh hưởng lớn đến tiến độ. |
| Tăng sự hài lòng của khách hàng | Khách hàng được tham gia vào quá trình phát triển, có thể phản hồi và điều chỉnh sản phẩm theo mong muốn. |

**Câu 6.Khi nào nên sử dụng mô hình thác nước thay vì mô hình tiến trình linh hoạt ?**

Mô hình thác nước (Waterfall) phù hợp hơn so với mô hình tiến trình linh hoạt (Agile) trong các trường hợp sau:

1. Yêu cầu rõ ràng, ổn định và không thay đổi nhiều
   * Nếu dự án có yêu cầu được xác định từ đầu và ít có khả năng thay đổi, mô hình thác nước giúp đảm bảo sự nhất quán và tránh phát sinh chi phí thay đổi.
2. Dự án có quy trình chặt chẽ và cần tài liệu chi tiết
   * Các dự án yêu cầu tài liệu chi tiết cho từng giai đoạn (phân tích, thiết kế, phát triển, kiểm thử, triển khai) thường phù hợp với mô hình thác nước.
3. Thời gian và ngân sách được xác định trước
   * Khi dự án có thời gian và ngân sách cố định, mô hình thác nước giúp kiểm soát chi phí và tiến độ chặt chẽ hơn so với Agile.
4. Dự án có tính chất tuyến tính, không cần phản hồi liên tục
   * Nếu các giai đoạn của dự án diễn ra theo trình tự cố định, ít có sự lặp lại hoặc thay đổi giữa chừng, thì mô hình thác nước là một lựa chọn tốt.
5. Dự án có quy mô lớn, liên quan đến nhiều bên liên quan
   * Những dự án lớn như phát triển phần mềm cho chính phủ, y tế, tài chính thường yêu cầu quy trình nghiêm ngặt và kiểm soát rủi ro chặt chẽ, nên mô hình thác nước sẽ phù hợp.
6. Dự án có yêu cầu về tuân thủ và kiểm tra chất lượng nghiêm ngặt
   * Các dự án trong lĩnh vực hàng không, quân sự, y tế… thường phải tuân theo các tiêu chuẩn nghiêm ngặt (ISO, CMMI, FDA, v.v.), do đó cần một mô hình có tài liệu đầy đủ như Waterfall.

**Câu 7.Thảo luận về những khó khăn khi áp dụng mô hình mã nguồn mở.**

Mô hình mã nguồn mở mang lại nhiều lợi ích như chi phí thấp, tính minh bạch và sự đóng góp từ cộng đồng. Tuy nhiên, khi áp dụng mô hình này, doanh nghiệp và tổ chức cũng phải đối mặt với nhiều thách thức, bao gồm:

1. Vấn đề bảo mật

* Rủi ro lỗ hổng bảo mật: Vì mã nguồn mở công khai, tin tặc có thể dễ dàng tìm kiếm lỗ hổng để khai thác. Nếu không được kiểm tra thường xuyên, hệ thống có thể trở thành mục tiêu tấn công.
* Thiếu hỗ trợ bảo mật chuyên nghiệp: Không phải tất cả dự án mã nguồn mở đều có đội ngũ bảo trì bảo mật kịp thời như các giải pháp thương mại.

2. Thiếu sự hỗ trợ chính thức

* Không giống như phần mềm thương mại có hỗ trợ khách hàng 24/7, phần mềm mã nguồn mở thường chỉ nhận được hỗ trợ từ cộng đồng hoặc nhà phát triển tình nguyện, có thể không đáp ứng nhu cầu doanh nghiệp kịp thời.
* Một số dự án có thể bị bỏ rơi nếu nhà phát triển ngừng duy trì.

3. Khó khăn trong triển khai và bảo trì

* Tích hợp với hệ thống hiện có: Một số phần mềm mã nguồn mở có thể không tương thích hoàn toàn với hệ thống nội bộ của doanh nghiệp.
* Cần đội ngũ kỹ thuật có chuyên môn: Doanh nghiệp cần nhân sự am hiểu về phần mềm để triển khai, tùy chỉnh và bảo trì, điều này có thể làm tăng chi phí gián tiếp.

4. Vấn đề về giấy phép và pháp lý

* Rủi ro vi phạm giấy phép: Một số giấy phép mã nguồn mở (như GPL) yêu cầu công khai mã nguồn khi sử dụng trong sản phẩm thương mại, điều này có thể không phù hợp với doanh nghiệp.
* Bất cập trong quyền sở hữu trí tuệ: Nếu sử dụng mã từ nhiều nguồn, doanh nghiệp có thể gặp rắc rối về quyền sở hữu và bản quyền.

5. Khó khăn trong kiểm soát chất lượng

* Thiếu tiêu chuẩn nhất quán: Vì được phát triển bởi nhiều cá nhân/tổ chức khác nhau, mã nguồn có thể không được kiểm soát chất lượng đồng đều.
* Không có lộ trình phát triển rõ ràng: Một số dự án mã nguồn mở phát triển theo hướng tự phát, không có kế hoạch dài hạn, gây khó khăn trong việc dự đoán tương lai của sản phẩm.

**Câu 8.Phân tích cách mô hình tiến trình linh hoạt giúp cải thiện chất lượng phần mềm.**

Mô hình tiến trình linh hoạt (Agile) giúp cải thiện chất lượng phần mềm thông qua các nguyên tắc như phát triển lặp, kiểm thử liên tục, phản hồi khách hàng và thích ứng linh hoạt. Dưới đây là các cách cụ thể mà Agile nâng cao chất lượng phần mềm:

1. Phát triển lặp (Iterative Development) giúp phát hiện lỗi sớm

* Agile chia dự án thành các vòng lặp nhỏ (iteration/sprint), mỗi vòng thường kéo dài từ 1-4 tuần.
* Ở mỗi vòng, phần mềm được kiểm thử và phản hồi liên tục, giúp phát hiện lỗi sớm và giảm thiểu chi phí sửa lỗi so với việc phát hiện lỗi ở giai đoạn cuối.

2. Kiểm thử liên tục (Continuous Testing) đảm bảo phần mềm ổn định

* Agile khuyến khích kiểm thử tự động (Automated Testing) để đảm bảo phần mềm hoạt động đúng sau mỗi lần thay đổi.
* Mô hình Test-Driven Development (TDD) giúp lập trình viên viết mã dựa trên các bài kiểm thử trước, giúp giảm lỗi logic ngay từ đầu.
* Kiểm thử hồi quy (Regression Testing) giúp đảm bảo các tính năng cũ không bị ảnh hưởng khi có thay đổi mới.

3. Phản hồi nhanh từ khách hàng giúp sản phẩm sát với nhu cầu thực tế

* Agile nhấn mạnh sự tham gia thường xuyên của khách hàng trong quá trình phát triển.
* Mỗi sprint đều có buổi demo sản phẩm, giúp khách hàng đánh giá và phản hồi sớm, giảm nguy cơ sản phẩm không đáp ứng yêu cầu thực tế.

4. Cải tiến liên tục giúp tối ưu hóa chất lượng

* Agile áp dụng mô hình Kaizen (cải tiến liên tục), trong đó nhóm phát triển luôn đánh giá và cải thiện quy trình làm việc thông qua các cuộc họp retrospective sau mỗi sprint.
* Tích hợp liên tục (Continuous Integration - CI) giúp phát hiện và sửa lỗi nhanh chóng, đảm bảo chất lượng mã nguồn luôn ở mức cao.

5. Tăng cường giao tiếp và cộng tác giúp giảm lỗi

* Agile khuyến khích giao tiếp thường xuyên giữa các thành viên trong nhóm (Scrum meetings, Daily Standups), giúp đảm bảo mọi người hiểu rõ yêu cầu và tránh lỗi do hiểu sai.
* Pair Programming (lập trình đôi) giúp giảm thiểu lỗi lập trình do có sự giám sát từ đồng đội.

6. Dễ thích nghi với thay đổi, tránh lỗi do yêu cầu không rõ ràng

* Agile linh hoạt với thay đổi yêu cầu từ khách hàng, giúp tránh lỗi phát sinh do yêu cầu không rõ ràng hoặc thay đổi muộn.
* Backlog ưu tiên các tính năng quan trọng trước, đảm bảo phần mềm phát triển theo đúng nhu cầu thực tế.

**Câu 9.Thảo luận về vai trò của pha bảo trì trong vòng đời phát triển phần mềm.**

Pha bảo trì (Maintenance Phase) là giai đoạn cuối cùng trong vòng đời phát triển phần mềm (SDLC - Software Development Life Cycle). Mặc dù phần mềm đã được triển khai và đưa vào sử dụng, nhưng nó vẫn cần được cập nhật, sửa lỗi và tối ưu hóa để đảm bảo hoạt động hiệu quả, đáp ứng yêu cầu thay đổi của người dùng và công nghệ.

1. Các loại bảo trì phần mềm

Pha bảo trì bao gồm nhiều hoạt động khác nhau, được chia thành bốn loại chính:

1.1. Bảo trì sửa lỗi (Corrective Maintenance)

* Mục tiêu: Sửa lỗi phát sinh sau khi triển khai phần mềm.
* Nguyên nhân: Lỗi logic, lỗi lập trình, lỗi bảo mật hoặc lỗi phát sinh từ môi trường hệ thống.
* Ví dụ: Vá lỗ hổng bảo mật hoặc sửa lỗi crash khi người dùng nhập dữ liệu sai định dạng.

1.2. Bảo trì thích ứng (Adaptive Maintenance)

* Mục tiêu: Điều chỉnh phần mềm để phù hợp với thay đổi trong môi trường hệ thống hoặc yêu cầu mới.
* Nguyên nhân: Nâng cấp hệ điều hành, thay đổi phần cứng, tích hợp với hệ thống khác.
* Ví dụ: Cập nhật ứng dụng để tương thích với phiên bản mới của Windows hoặc Android.

1.3. Bảo trì hoàn thiện (Perfective Maintenance)

* Mục tiêu: Cải thiện hiệu suất, giao diện hoặc tính năng của phần mềm mà không làm thay đổi chức năng cốt lõi.
* Nguyên nhân: Phản hồi từ người dùng, tối ưu hóa thuật toán hoặc nâng cao trải nghiệm người dùng.
* Ví dụ: Tăng tốc độ tải trang trong một ứng dụng web hoặc cải thiện giao diện người dùng.

1.4. Bảo trì phòng ngừa (Preventive Maintenance)

* Mục tiêu: Ngăn ngừa lỗi trong tương lai bằng cách tối ưu hóa mã nguồn và cơ sở hạ tầng phần mềm.
* Nguyên nhân: Mã nguồn cũ kém hiệu quả, lỗ hổng bảo mật tiềm ẩn.
* Ví dụ: Refactor lại code để tăng hiệu suất hoặc cập nhật thư viện bảo mật để tránh rủi ro trong tương lai.

2. Tầm quan trọng của pha bảo trì

2.1. Đảm bảo tính ổn định và hiệu suất của phần mềm

* Phần mềm cần được duy trì để đảm bảo hoạt động trơn tru và không bị gián đoạn khi hệ thống, phần cứng hoặc môi trường vận hành thay đổi.
* Giúp tối ưu hiệu suất, giảm độ trễ và cải thiện khả năng mở rộng.

2.2. Đáp ứng nhu cầu thay đổi của người dùng và doanh nghiệp

* Yêu cầu của khách hàng và doanh nghiệp thay đổi liên tục, bảo trì giúp phần mềm cập nhật tính năng mới để phù hợp với nhu cầu thực tế.
* Giữ chân người dùng và tăng giá trị sản phẩm.

2.3. Đảm bảo an toàn và bảo mật hệ thống

* Lỗ hổng bảo mật mới liên tục xuất hiện, bảo trì giúp cập nhật bản vá kịp thời, bảo vệ hệ thống khỏi tấn công mạng.
* Đảm bảo tuân thủ các tiêu chuẩn bảo mật và quy định pháp lý.

2.4. Giảm chi phí sửa chữa trong tương lai

* Phát hiện và khắc phục lỗi sớm giúp tránh các sự cố nghiêm trọng, giảm chi phí sửa chữa và downtime hệ thống.
* Cải thiện chất lượng mã nguồn, giảm nguy cơ lỗi phát sinh do code cũ.

2.5. Kéo dài vòng đời phần mềm

* Phần mềm được bảo trì thường xuyên sẽ có vòng đời dài hơn, tránh phải phát triển lại từ đầu khi công nghệ thay đổi.
* Tận dụng được nguồn lực đầu tư ban đầu, tối ưu ROI (Return on Investment).

3. Thách thức trong pha bảo trì

3.1. Chi phí bảo trì cao

* Trong nhiều trường hợp, chi phí bảo trì có thể cao hơn chi phí phát triển ban đầu, đặc biệt với phần mềm lớn và phức tạp.

3.2. Khó khăn trong quản lý mã nguồn cũ

* Phần mềm phát triển lâu năm có thể sử dụng công nghệ lỗi thời, khó bảo trì và nâng cấp.

3.3. Ảnh hưởng đến hoạt động kinh doanh

* Việc cập nhật phần mềm có thể gây gián đoạn dịch vụ nếu không được thực hiện đúng cách.

**10.Đề xuất mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng và giải thích lý do.**

- Mô hình vòng đời phù hợp khác: Mô hình Spiral (Mô hình xoắn ốc)

- Mô hình Spiral (xoắn ốc) phù hợp cho phần mềm ngân hàng vì nó kết hợp quy trình có kế hoạch rõ ràng của mô hình thác nước với tính linh hoạt và kiểm soát rủi ro của mô hình Agile.

Lý do chính:  
- Quản lý rủi ro tốt: Mỗi vòng lặp của mô hình xoắn ốc đều có bước phân tích rủi ro, rất quan trọng với phần mềm ngân hàng.  
- Linh hoạt với yêu cầu thay đổi: Nếu ngân hàng cần thay đổi quy trình hoặc chức năng, mô hình có thể điều chỉnh dễ dàng.

- Kiểm thử liên tục: Sau mỗi vòng lặp, hệ thống đều được kiểm thử, giúp phát hiện lỗi sớm và giảm chi phí sửa lỗi.  
- Triển khai từng phần: Có thể phát hành phiên bản nhỏ của phần mềm để thử nghiệm trước khi triển khai toàn bộ.

=> Mô hình Spiral là lựa chọn tốt cho phần mềm ngân hàng vì nó kết hợp giữa kiểm soát rủi ro, tính linh hoạt và kiểm thử liên tục. Nếu ngân hàng muốn có một hệ thống dễ mở rộng và cập nhật mà vẫn đảm bảo bảo mật và hiệu suất.

**Câu hỏi tình huống**

1. Mọi công ty phát triển phần mềm có cần phải xây dựng quy trình phát triển phần mềm không? Tại sao?

Có, mọi công ty phát triển phần mềm cần xây dựng quy trình phát triển phần mềm. Lý do là để đảm bảo chất lượng sản phẩm, quản lý hiệu quả thời gian và nguồn lực, giảm thiểu rủi ro, và đáp ứng nhu cầu của khách hàng một cách nhất quán và chuyên nghiệp.

1. Trong quá trình giao tiếp với khách hàng, bạn có cần phải bổ sung thêm thông tin mới không? Tại sao?

Có, trong quá trình giao tiếp với khách hàng, cần bổ sung thêm thông tin mới. Điều này giúp hiểu rõ hơn về nhu cầu của khách hàng, xây dựng niềm tin, và đảm bảo rằng sản phẩm hoặc dịch vụ đáp ứng đúng kỳ vọng của họ.

1. Dự án phát triển phần mềm bị trì hoãn do đội ngũ phát triển thiếu sót trong quá trình kiểm thử. Làm sao để tránh điều này, bạn sẽ làm gì?

Để tránh tình trạng này, cần:

* + Xây dựng kế hoạch kiểm thử chi tiết và rõ ràng từ đầu.
  + Phân công nhiệm vụ kiểm thử cho đội ngũ có chuyên môn phù hợp.
  + Sử dụng công cụ tự động hóa kiểm thử để tăng hiệu quả.
  + Thường xuyên theo dõi và đánh giá tiến độ kiểm thử trong suốt dự án.

1. Trong workflow hiện tại, kiến trúc sư phần mềm phải thiết kế bản vẽ để chuẩn bị cho giai đoạn tiếp theo. Dự án phát triển nên xử lý thế nào?

Dự án nên đảm bảo rằng:

* + Kiến trúc sư có đủ thời gian và tài nguyên để hoàn thành bản vẽ chất lượng.
  + Có sự phối hợp chặt chẽ giữa các đội ngũ để bản vẽ được phê duyệt nhanh chóng.
  + Workflow cần được tối ưu hóa để giảm thiểu chậm trễ và đảm bảo tính liền mạch.

1. Khách hàng yêu cầu rút ngắn thời gian phát triển dự án mà không thay đổi yêu cầu. Dự án phát triển nên phản ứng như thế nào?

Dự án nên:

* + Đánh giá lại toàn bộ quy trình để tìm cách tối ưu hóa mà không ảnh hưởng chất lượng.
  + Thảo luận với khách hàng để cân nhắc ưu tiên các tính năng quan trọng nhất.
  + Nếu không khả thi, giải thích rõ ràng với khách hàng về rủi ro và giới hạn.

1. Mọi công ty muốn áp dụng mô hình CMMI phải bắt đầu từ đâu và làm thế nào để đạt được các cấp độ cao hơn?

Để áp dụng mô hình CMMI:

* + Bắt đầu bằng việc đánh giá hiện trạng quy trình của công ty.
  + Xây dựng kế hoạch cải tiến dựa trên các tiêu chuẩn CMMI.
  + Đào tạo nhân viên, áp dụng các quy trình mới, và thường xuyên kiểm tra, cải tiến để đạt các cấp độ cao hơn như 2, 3, 4, 5.

1. Trong workflow làm việc, khách hàng cùng tham gia trong từng bước rõ ràng. Dự án phát triển cần làm gì?

Dự án cần:

* + Xây dựng quy trình giao tiếp minh bạch để khách hàng dễ dàng theo dõi.
  + Đảm bảo mọi phản hồi của khách hàng được xử lý nhanh chóng và tích hợp vào sản phẩm.
  + Sử dụng công cụ quản lý dự án để duy trì sự phối hợp hiệu quả.

1. Mọi dự án phải cao trong phần đầu dự án để thử nghiệm yêu cầu rõ ràng. Dự án phát triển cần làm gì?

Dự án cần:

* + Thực hiện phân tích yêu cầu kỹ lưỡng ở giai đoạn đầu.
  + Sử dụng các phương pháp như phác thảo, mô phỏng, hoặc prototype để xác nhận yêu cầu.
  + Đảm bảo đội ngũ hiểu rõ yêu cầu trước khi bắt đầu phát triển.

1. 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?

Để đảm bảo phối hợp hiệu quả:

* + Sử dụng công cụ quản lý dự án và giao tiếp như Jira, Slack, hoặc Microsoft Teams.
  + Thiết lập lịch họp định kỳ và quy trình chia sẻ thông tin rõ ràng.
  + Đào tạo đội ngũ về cách làm việc nhóm từ xa và giải quyết xung đột.

1. Mọi công ty phát triển phần mềm cần trong việc quản lý dự án để không có sự chồng chéo. Hãy giải thích rõ.

Việc quản lý dự án chặt chẽ giúp:

* Phân công nhiệm vụ rõ ràng, tránh trùng lặp công việc giữa các nhóm.
* Sử dụng công cụ quản lý dự án để theo dõi tiến độ và tài nguyên.
* Thường xuyên kiểm tra và điều chỉnh để đảm bảo mọi hoạt động diễn ra suôn sẻ, không gây lãng phí thời gian hoặc nguồn lực.

**Review chapter 1**

1. Định nghĩa Công nghệ phần mềm: Phần mềm là tập hợp các hướng dẫn, chương trình được viết để máy tính thực thi nhằm thực hiện các chức năng hoặc nhiệm vụ cụ thể.

Gồm 3 loại phần mềm: Hệ thống, ứng dụng và nhúng.

Mục tiêu: Tạo ra phần mềm đúng yêu cầu khách hàng, đảm bảo phần mềm có thể bảo trì và nâng cấp dễ dàng, giảm thiểu thời gian và chi phí phát triển.

1. Quy trình phát triển phần mềm: Lấy yêu cầu, thiết kế, lập trình, kiểm thử, triển khai, bảo trì.
2. Phạm vi phát triển phần mềm: Bảo trì, biuer đồ chi phí tương đối của các giai đoạn trong vòng đời ohaanf mềm
3. Các vấn đề thường gặp khi phát triển phần mềm: Vấn đề vượt chi phí, vấn đề trễ thời gian, vấn đề vẫn còn lỗi sau khi bàn giao. Tùy thuộc vào vấn đề gặp phải gây ra hậu quả xác định.
4. Các lỗi dẫn đến các vấn đề trong phát triển phần mềm: Không đồng bộ, thiếu quyết đoán, yêu cầu kém, phạm vi mở rộng không kiểm soát, thiếu giao tiếp

**Review chapter 2**

**Phần 1: Giới thiệu về**

**các mô hình quy trình phần mềm**

* **Mô hình thác nước (Waterfall Model)**: Mô hình này là một mô hình tuyến tính, tuần tự, trong đó mỗi giai đoạn của quy trình phát triển phần mềm phải được hoàn thành trước khi giai đoạn tiếp theo bắt đầu. Mô hình này đơn giản và dễ hiểu, nhưng nó không linh hoạt và khó thích ứng với các thay đổi.
* **Mô hình chữ V (V-Model)**: Mô hình này là một biến thể của mô hình thác nước, trong đó giai đoạn kiểm thử được nhấn mạnh hơn. Mô hình chữ V liên kết mỗi giai đoạn phát triển với một giai đoạn kiểm thử tương ứng.
* **Mô hình nguyên mẫu (Prototyping Model)**: Mô hình này cho phép tạo ra một nguyên mẫu của phần mềm trước khi phát triển phần mềm hoàn chỉnh. Nguyên mẫu này được sử dụng để thu thập phản hồi từ người dùng và để xác định các yêu cầu của phần mềm.
* **Mô hình xoắn ốc (Spiral Model)**: Mô hình này là một mô hình lặp, trong đó mỗi vòng lặp bao gồm các giai đoạn lập kế hoạch, phân tích rủi ro, phát triển và đánh giá. Mô hình xoắn ốc phù hợp cho các dự án lớn và phức tạp, trong đó rủi ro là một yếu tố quan trọng.

**Phần 2: Các khía cạnh thực tế của phát triển phần mềm**

* **Thay đổi yêu cầu**: Yêu cầu của phần mềm có thể thay đổi trong quá trình phát triển. Điều này có thể là do người dùng thay đổi nhu cầu của họ, hoặc do các yếu tố bên ngoài khác.
* **Rủi ro**: Phát triển phần mềm là một hoạt động rủi ro. Có nhiều yếu tố có thể dẫn đến thất bại của dự án, chẳng hạn như yêu cầu không rõ ràng, ước tính chi phí và thời gian không chính xác, và các vấn đề kỹ thuật.
* **Quản lý dự án**: Quản lý dự án hiệu quả là rất quan trọng để đảm bảo thành công của dự án phát triển phần mềm. Điều này bao gồm việc lập kế hoạch, tổ chức, lãnh đạo và kiểm soát dự án.

**Phần 3: Các phương pháp phát triển phần mềm linh hoạt (Agile)**

* **Lập trình cực đoan (Extreme Programming - XP)**: XP là một phương pháp phát triển phần mềm linh hoạt, trong đó nhấn mạnh vào việc giao tiếp, phản hồi và đơn giản.
* **Scrum**: Scrum là một khung làm việc linh hoạt, trong đó dự án được chia thành các sprint ngắn. Mỗi sprint có một mục tiêu cụ thể và một khoảng thời gian cố định.
* **Kanban**: Kanban là một phương pháp quản lý công việc trực quan, trong đó công việc được chia thành các giai đoạn và được hiển thị trên một bảng Kanban.

**Kết luận**

Chương 2 cung cấp một cái nhìn tổng quan về các mô hình quy trình phần mềm khác nhau và các khía cạnh thực tế của phát triển phần mềm. Việc lựa chọn mô hình quy trình phần mềm phù hợp và áp dụng các phương pháp quản lý dự án hiệu quả là rất quan trọng để đảm bảo thành công của dự án phát triển phần mềm.

**Review chapter 3**

**Tổng quan về nội dung chương**

Chương "The Software Process" tập trung vào việc giới thiệu và giải thích quy trình phát triển phần mềm, nhấn mạnh tầm quan trọng của việc quản lý và cải tiến quy trình này. Nội dung được chia thành nhiều phần, bao gồm:

1. **Khái niệm cơ bản về quy trình phần mềm:** Quy trình phần mềm được định nghĩa là cách thức tổ chức các hoạt động, kỹ thuật, công cụ, và con người để sản xuất phần mềm. Chương chỉ ra sự khác biệt giữa các tổ chức trong cách tiếp cận quy trình, từ việc coi phần mềm là tự tài liệu hóa (self-documenting) đến các tổ chức yêu cầu tài liệu chi tiết và kiểm tra kỹ lưỡng.
2. **Quy trình Thống nhất (Unified Process):** Đây là một phương pháp phát triển phần mềm hướng đối tượng (object-oriented) chính, được mô tả chi tiết với năm luồng công việc cốt lõi (core workflows): yêu cầu (requirements), phân tích (analysis), thiết kế (design), thực hiện (implementation), và kiểm thử (testing). Chương cũng giải thích bốn pha (phases) của Unified Process: khởi tạo (inception), chi tiết hóa (elaboration), xây dựng (construction), và chuyển giao (transition).
3. **Tính chất lặp lại và tăng dần (Iteration and Incrementation):** Phương pháp này nhấn mạnh rằng phát triển phần mềm thường được thực hiện qua nhiều lần lặp lại và tăng dần, sử dụng Ngôn ngữ Mô hình Thống nhất (UML - Unified Modeling Language) để mô phỏng các khía cạnh của sản phẩm phần mềm, giúp giao tiếp hiệu quả hơn giữa các thành viên đội ngũ phát triển.
4. **Cải tiến quy trình phần mềm:** Chương đề cập đến các sáng kiến cải tiến quy trình phần mềm ở cấp quốc gia và quốc tế, như Mô hình Độ trưởng thành Khả năng (Capability Maturity Model - CMM) của Viện Kỹ thuật Phần mềm (Software Engineering Institute - SEI), cùng các tiêu chuẩn như ISO 9000 và ISO/IEC 15504 (SPICE).
5. **Các thách thức và giải pháp:** Chương nêu bật các vấn đề thường gặp trong quy trình phát triển phần mềm, như sự mơ hồ, mâu thuẫn, hoặc thiếu sót trong yêu cầu, cũng như việc bảo trì sau khi giao hàng (postdelivery maintenance) và việc nghỉ hưu của phần mềm (retirement).

**Ý nghĩa và giá trị**

* **Tầm quan trọng của quy trình có cấu trúc:** Quy trình phần mềm không chỉ là một chuỗi các bước đơn giản, mà cần được quản lý chặt chẽ để đảm bảo chất lượng, giảm thiểu rủi ro, và tối ưu hóa hiệu suất. Unified Process, với tính lặp lại và tăng dần, là một phương pháp linh hoạt và hiệu quả để đối phó với sự phức tạp của các dự án phần mềm lớn.
* **Vai trò của UML và mô hình hóa:** Sử dụng UML giúp cải thiện giao tiếp và giảm thiểu hiểu lầm giữa các bên liên quan, đặc biệt là giữa khách hàng và đội phát triển, nhờ vào khả năng thể hiện trực quan các khái niệm phức tạp.
* **Cải tiến liên tục:** Các mô hình như CMM và tiêu chuẩn ISO nhấn mạnh rằng cải tiến quy trình phần mềm là cần thiết để nâng cao chất lượng sản phẩm, giảm chi phí, và tăng năng suất. Dữ liệu từ các tổ chức như Hughes Aircraft, Raytheon, và Motorola cho thấy lợi ích kinh tế rõ rệt khi nâng cao mức độ trưởng thành của quy trình.
* **Xử lý các thách thức thực tế:** Chương giải quyết các vấn đề thực tế như sự thay đổi yêu cầu của khách hàng (mowing-target problem), thiếu tài liệu, và rủi ro kỹ thuật, cung cấp nền tảng để tìm kiếm giải pháp trong các phần khác của sách.

**Điểm mạnh và hạn chế**

* **Điểm mạnh:**
  + Nội dung được trình bày chi tiết, có cấu trúc rõ ràng, với các ví dụ thực tế (như lỗi phần mềm trong các dự án không gian) và các "Just in Case You Wanted to Know" Box giúp làm sáng tỏ các khái niệm phức tạp.
  + Chương cung cấp cái nhìn toàn diện về cả lý thuyết (như các luồng công việc và pha của Unified Process) và thực tiễn (như cải tiến quy trình qua CMM và ISO).
  + Các tài liệu tham khảo phong phú, bao gồm các nghiên cứu và tiêu chuẩn quốc tế, giúp người đọc mở rộng kiến thức.
* **Hạn chế:**
  + Một số khái niệm, như chi tiết về UML, chỉ được đề cập sơ qua và yêu cầu tham khảo các chương khác, có thể khiến người đọc mới cảm thấy thiếu liền mạch.
  + Nội dung tập trung nhiều vào các dự án phần mềm lớn, có thể không hoàn toàn phù hợp với các dự án nhỏ hoặc vừa, dù tác giả đã đề cập đến việc điều chỉnh Unified Process cho quy mô khác nhau.

**Kết luận**

Chương The Software Process cung cấp nền tảng lý thuyết vững chắc về Unified Process và các mô hình cải tiến mà còn nhấn mạnh giá trị thực tiễn của việc áp dụng các phương pháp này trong thế giới thực.