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**

<u>Câu hỏi ngắn</u>

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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.
7. 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.
8. 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.
9. Dự án phần mềm lớn có nhiều nhóm phát triển ở các địa điểm khác nhau. Làm thế nào để đảm bảo các nhóm phối hợp hiệu quả?

   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.
10. 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.