Chương 1: PHẠM VI CỦA CÔNG NGHỆ PHẦN MỀM

Phần 1: Câu hỏi trắc nghiệm

Câu hỏi 1: Phần mềm bao gồm các loại nào dưới đây?
Đáp án: D. Cả A, B và C (Phần mềm hệ thống, Phần mềm ứng dụng, Phần mềm nhúng)

Câu hỏi 2: Công nghệ phần mềm là gì?
Đáp án: C. Ứng dụng các phương pháp khoa học để phát triển phần mềm

Câu hỏi 3: Quy trình phát triển phần mềm gồm mấy giai đoạn chính?
Đáp án: B. 4

Câu hỏi 4: Hoạt động nào dưới đây thuộc quy trình bảo trì phần mềm?
Đáp án: C. Cập nhật phần mềm để phù hợp với thay đổi môi trường

Câu hỏi 5: Chi phí bảo trì phần mềm chiếm bao nhiêu phần trăm tổng chi phí vòng đời phần mềm?
Đáp án: C. 60%

Câu hỏi 6: Nguyên nhân chính gây ra việc vượt chi phí khi phát triển phần mềm là gì?
Đáp án: D. Cả A và C (Thiếu nhân lực, Thay đổi công nghệ)

Câu hỏi 7: Yêu cầu nào dưới đây không phải là yêu cầu phi chức năng?
Đáp án: D. Chức năng đăng nhập

Câu hỏi 8: Khi nào phần mềm được coi là hoàn thành?
Đáp án: 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?
Đáp án: D. Tất cả đều đúng

Câu hỏi 10: Phần mềm có thể được chia thành bao nhiêu loại chính?
Đáp án: B. 3

Phần 2: Câu hỏi ngắn:

1. **Phần mềm là gì?**
   → Phần mềm là tập hợp các chương trình, dữ liệu và tài liệu liên quan được sử dụng để điều khiển máy tính hoặc thực hiện các tác vụ cụ thể**.**
2. **Công nghệ phần mềm là gì?**
   → Công nghệ phần mềm là ngành khoa học và kỹ thuật nghiên cứu các phương pháp, quy trình và công cụ để phát triển, vận hành và bảo trì phần mềm hiệu quả.
3. **Các loại phần mềm chính là gì?**
   → Ba loại phần mềm chính:

- Phần mềm hệ thống (System Software)
- Phần mềm ứng dụng (Application Software)
- Phần mềm nhúng (Embedded Software)

4. **Tại sao công nghệ phần mềm lại quan trọng?**
   → Công nghệ phần mềm giúp tạo ra phần mềm chất lượng cao, giảm lỗi, tối ưu hiệu suất, tiết kiệm chi phí và nâng cao trải nghiệm người dùng. Nó cũng giúp quản lý vòng đời phần mềm một cách có hệ thống.

5. **Quy trình phát triển phần mềm gồm những giai đoạn nào?**
   → Các giai đoạn chính của quy trình phát triển phần mềm:
   - Phân tích yêu cầu
   - Thiết kế
   - Lập trình và triển khai
   - Kiểm thử
   - Bảo trì và nâng cấp

6. **Khía cạnh kinh tế của công nghệ phần mềm là gì?**
   → Khía cạnh kinh tế liên quan đến chi phí phát triển, vận hành, bảo trì phần mềm và tối ưu hóa nguồn lực để đạt hiệu quả tài chính cao nhất.

7. **Khía cạnh công nghệ của công nghệ phần mềm là gì?**
   → Liên quan đến việc áp dụng các công cụ, ngôn ngữ lập trình, kiến trúc hệ thống, mô hình phát triển và phương pháp quản lý để xây dựng phần mềm.

8. **Khía cạnh bảo trì của công nghệ phần mềm là gì?**
   → Liên quan đến việc cập nhật, sửa lỗi, nâng cấp tính năng và đảm bảo phần mềm hoạt động ổn định trong suốt vòng đời của nó.

9. **Các nguyên nhân chính gây trễ thời hạn khi phát triển phần mềm là gì?**
   → Một số nguyên nhân chính:
   - Xác định yêu cầu không rõ ràng
   - Thay đổi yêu cầu trong quá trình phát triển
   - Thiếu nhân lực hoặc công nghệ không phù hợp
   - Quản lý dự án kém

10. **Bảo trì phần mềm bao gồm những hoạt động nào?**
    → Các hoạt động bảo trì phần mềm:

- Sửa lỗi (Corrective Maintenance): Khắc phục lỗi xuất hiện trong quá trình sử dụng

- Bảo trì thích ứng (Adaptive Maintenance): Điều chỉnh phần mềm để phù hợp với môi trường mới
- Bảo trì hoàn thiện (Perfective Maintenance): Cải thiện hiệu suất, tối ưu phần mềm
- Bảo trì phòng ngừa (Preventive Maintenance): Ngăn ngừa lỗi xảy ra trong tương lai

Phần 3: Câu hỏi thảo luận nhóm

1. **phân biệt phần mềm hệ thống và phần mềm ứng dụng.**
    - Phần mềm hệ thống là phần mềm giúp máy tính hoạt động, kiểu như nền tảng để các phần mềm khác chạy được.
        - *Ví dụ:* Windows, macOS, Linux – mấy cái này giúp máy tính khởi động và vận hành.
    - Phần mềm ứng dụng là mấy phần mềm phục vụ nhu cầu cụ thể của người dùng.
        - *Ví dụ:* Zalo để nhắn tin, Microsoft Word để soạn thảo văn bản.
2. **Vai trò của công nghệ phần mềm trong tài chính?**
    - Giúp giao dịch nhanh hơn, chính xác hơn, bảo mật cao hơn.
    - Hỗ trợ quản lý tài chính, kế toán, giao dịch ngân hàng.
    - *Ví dụ:* Mấy app ngân hàng như Vietcombank, MB Bank cho phép chuyển khoản ngay tại nhà, thay vì ra ngân hàng chờ cả buổi.
3. **Thách thức trong bảo trì phần mềm?**
    - Hệ thống cũ khó nâng cấp, chồng chéo mã nguồn.
    - Chi phí bảo trì cao, nhất là với hệ thống lớn.
    - *Ví dụ:* Một công ty vẫn dùng phần mềm kế toán từ 10 năm trước, giờ muốn nâng cấp thì mất rất nhiều thời gian và tiền bạc.
4. **Tại sao phần mềm thương mại điện tử cần bảo trì thường xuyên?**
    - Cập nhật bảo mật, tránh bị hack.
    - Cải thiện hiệu suất, tránh tình trạng load chậm khi có quá nhiều người dùng.
    - *Ví dụ:* Shopee, Lazada hay cập nhật để tối ưu tìm kiếm sản phẩm và nâng cấp hệ thống thanh toán.
5. **Vấn đề khi khách hàng liên tục thay đổi yêu cầu?**
    - Dự án kéo dài, tốn kém chi phí hơn dự tính.

- Gây xáo trộn trong quá trình phát triển, dễ phát sinh lỗi.
- *Ví dụ:* Một doanh nghiệp đặt làm phần mềm quản lý nhân sự, nhưng cứ đổi yêu cầu về giao diện liên tục, khiến team dev phát triển hoài không xong.

6. **So sánh chi phí phát triển và bảo trì phần mềm.**
    - Chi phí phát triển là chi phí bỏ ra ban đầu để tạo ra phần mềm.
    - Chi phí bảo trì là số tiền phải chi để giữ phần mềm hoạt động tốt, thường cao hơn nhiều so với chi phí phát triển.
    - *Ví dụ:* Phát triển một phần mềm CRM mất 1 tỷ, nhưng mỗi năm mất 500 triệu để cập nhật, sửa lỗi, nâng cấp.
7. **Phân biệt yêu cầu chức năng và phi chức năng.**
    - Yêu cầu chức năng là những thứ phần mềm bắt buộc phải có.
        - *Ví dụ:* Ứng dụng ngân hàng phải có tính năng đăng nhập, chuyển tiền.
    - Yêu cầu phi chức năng là những yếu tố ảnh hưởng đến trải nghiệm sử dụng như tốc độ, bảo mật.
        - *Ví dụ:* App ngân hàng phải xử lý giao dịch nhanh, không bị giật lag.
8. **Các mô hình quy trình phát triển phần mềm phổ biến?**
    - Mô hình thác nước: Làm từng bước theo thứ tự, khó quay lại sửa nếu có lỗi.
        - *Ví dụ:* Phát triển phần mềm điều khiển máy bay, phải đúng ngay từ đầu.
    - Mô hình Agile: Làm theo từng giai đoạn nhỏ, linh hoạt thay đổi nếu cần.
        - *Ví dụ:* Ứng dụng Grab liên tục cập nhật tính năng dựa trên phản hồi người dùng.
9. **Cách giảm thiểu lỗi phần mềm sau khi bàn giao?**
    - Kiểm thử kỹ trước khi phát hành.
    - Viết tài liệu hướng dẫn chi tiết.
    - Hỗ trợ khách hàng trong giai đoạn đầu.
    - *Ví dụ:* Trước khi Apple ra mắt iOS mới, họ phải kiểm tra rất nhiều để tránh lỗi nghiêm trọng ảnh hưởng đến hàng triệu người dùng.
10. **Vai trò của đội kiểm thử (QA) trong phát triển phần mềm?**

- Phát hiện lỗi trước khi phát hành, giúp phần mềm chạy ổn định.
- Đảm bảo phần mềm đáp ứng đúng yêu cầu khách hàng.

- *Ví dụ:* Trước khi Facebook cập nhật một tính năng mới, đội QA sẽ kiểm thử xem có lỗi gì không, tránh ảnh hưởng đến hàng tỷ người dùng.

Phần 4: Câu hỏi tình huống

Tình huống 1: Kiểm tra nguyên nhân lỗi, đề xuất bản vá, cập nhật phần mềm sau khi thảo luận với khách hàng.

Tình huống 2: Đánh giá mức độ ảnh hưởng, lập kế hoạch bổ sung tính năng, thương lượng với khách hàng về chi phí và thời gian.

Tình huống 3: Tổ chức lại tài liệu yêu cầu, cải thiện giao tiếp với khách hàng, phân công nhiệm vụ rõ ràng.

Tình huống 4: Thu thập phản hồi người dùng, cải thiện UI/UX, tổ chức đào tạo hoặc hướng dẫn sử dụng.

Tình huống 5: Xác định nguyên nhân vượt ngân sách, điều chỉnh quy trình quản lý dự án, kiểm soát phạm vi công việc chặt chẽ hơn.

Tình huống 6: Nếu lỗi không ảnh hưởng lớn, có thể hoãn sửa để tối ưu chi phí và tiến độ.

Tình huống 7: Thương lượng với khách hàng, đánh giá khả năng tăng nguồn lực hoặc điều chỉnh phạm vi dự án.

Tình huống 8: Đề xuất sử dụng công nghệ phù hợp hơn, thuê chuyên gia hoặc đào tạo nội bộ để nâng cao năng lực đội ngũ.

Tình huống 9: Ưu tiên khắc phục lỗi bảo mật ngay lập tức, thông báo khách hàng và cập nhật phần mềm.

Tình huống 10: Phân tích sự thay đổi quy trình sản xuất, cập nhật phần mềm sao cho không ảnh hưởng hoạt động của khách hàng.

Chương 2: TIẾN TRÌNH PHẦN MỀM

Phần 1: Trắc nghiệm

1. B. Workflow lấy yêu cầu
2. A. Pha khởi đầu
3. C. Mức 4
4. B. Khởi đầu, làm rõ, xây dựng, chuyển giao
5. D. Workflow kiểm thử

6. B. Quy trình được kiểm soát và đo lường
7. C. Mô hình hàm mũ
8. A. Quy trình được cải tiến liên tục
9. C. Thiết kế kiến trúc và chi tiết hệ thống
10. B. Capability Maturity Model

Phần 2: Câu hỏi tự luận

1. Pha khởi đầu tập trung xác định phạm vi, mục tiêu và rủi ro ban đầu của dự án.
2. Workflow lấy yêu cầu nhằm thu thập, phân tích và đặc tả yêu cầu của khách hàng.
3. Tiến trình thống nhất gồm 4 pha chính: Khởi đầu, Làm rõ, Xây dựng, Chuyển giao.
4. CMM mức 2 tập trung vào quản lý dự án cơ bản, trong khi CMM mức 3 tập trung vào chuẩn hóa quy trình.
5. Workflow kiểm thử có nhiệm vụ kiểm tra, đánh giá chất lượng phần mềm và phát hiện lỗi.
6. Mô hình CMM có 5 mức độ: Khởi đầu, Quản lý, Xác định, Định lượng, Tối ưu hóa.
7. Mô hình thác nước có quy trình tuyến tính, còn mô hình lặp thực hiện phát triển theo vòng lặp, cải tiến liên tục.
8. Tiến trình thống nhất không phải là mô hình lặp, nhưng nó có tính lặp trong các pha phát triển.
9. Workflow thiết kế nhằm tạo ra kiến trúc phần mềm và chi tiết hệ thống.
10. CMM mức 5 tập trung vào tối ưu hóa và cải tiến quy trình liên tục.

Phần 3: Câu hỏi thảo luận

1. Mô tả vai trò của từng workflow như lấy yêu cầu, thiết kế, kiểm thử, triển khai.
2. Mô hình vòng đời thác nước là tuyến tính, tiến trình thống nhất có tính lặp và linh hoạt hơn.
3. Mô hình lặp cải thiện liên tục, mô hình tăng trưởng mở rộng dần theo thời gian.
4. CMM giúp nâng cao chất lượng bằng cách tiêu chuẩn hóa quy trình phát triển phần mềm.
5. Doanh nghiệp nhỏ khó áp dụng CMM do tài nguyên hạn chế và chi phí cao.

6. Cải tiến bằng cách áp dụng công cụ đo lường, tối ưu hóa quy trình, sử dụng phản hồi người dùng.
7. Tiến trình thống nhất phù hợp với dự án lớn vì tính linh hoạt và kiểm soát rủi ro tốt hơn.
8. Kiểm thử quan trọng ở tất cả các pha để đảm bảo chất lượng phần mềm và giảm lỗi.
9. CMM mức 4 tập trung vào định lượng, mức 5 tập trung vào tối ưu hóa quy trình.
10. Tổ chức nhóm workflow lấy yêu cầu bằng cách phân chia vai trò rõ ràng, sử dụng công cụ hỗ trợ thu thập yêu cầu.

Phần 4: Câu hỏi tình huống

1. Quản lý thay đổi chặt chẽ, thảo luận với khách hàng để kiểm soát phạm vi yêu cầu, áp dụng phương pháp phát triển linh hoạt.
2. Đánh giá tác động của tính năng mới, thương lượng về thời gian và chi phí bổ sung, lập kế hoạch cập nhật phần mềm.
3. Cải thiện quy trình kiểm thử, sử dụng kiểm thử tự động để giảm lỗi, tối ưu hóa quy trình sửa lỗi.
4. Đánh giá lại thiết kế, thực hiện tối ưu mã nguồn và thuật toán, kiểm tra tác động trước khi thay đổi.
5. Phân tích rủi ro của việc rút ngắn thời gian, thương lượng về phạm vi yêu cầu hoặc tăng nguồn lực.
6. Điều chỉnh phạm vi áp dụng CMM phù hợp với khả năng công ty, tập trung vào các quy trình quan trọng trước.
7. Hướng dẫn khách hàng cung cấp yêu cầu rõ ràng hơn, sử dụng mẫu thu thập yêu cầu chuẩn hóa.
8. Xây dựng tài liệu yêu cầu chi tiết, tổ chức họp thường xuyên với khách hàng để làm rõ yêu cầu.
9. Xây dựng quy trình giao tiếp rõ ràng giữa các nhóm, sử dụng công cụ quản lý dự án chung.
10. Chuẩn hóa quy trình bằng cách áp dụng mô hình phát triển phù hợp (ví dụ: Agile hoặc CMMI).

Chương 3: MỘT SỐ MỐ HÌNH VÒNG ĐỜI PHÁT TRIỂN PHẦN MỀM

Phần 1: Câu hỏi 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: C. Pha phân tích

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: B. Một chu kỳ lặp của toàn bộ quy trình phát triển

Câu hỏi 6: **Điểm yếu lớn nhất của mô hình xây và 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

Phần 2: 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 thu thập, phân tích và làm rõ các yêu cầu từ khách hàng hoặc người dùng.

Vai trò:

- Xác định rõ các chức năng, tính năng cần thiết.

- Giúp tránh nhầm lẫn và thay đổi không cần thiết trong quá trình phát triển.
- Là nền tảng để các giai đoạn tiếp theo như phân tích, thiết kế và phát triển.

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

- Mô hình thác nước (Waterfall Model) là mô hình phát triển phần mềm tuần tự gồm các giai đoạn cố định: Lấy yêu cầu - Phân tích - Thiết kế - Cài đặt - Kiểm thử - Triển khai - Bảo trì
- Phát triển tuyến tính: Sau khi hoàn thành một giai đoạn sẽ chuyển sang giai đoạn tiếp theo mà không quay lại các bước trước đó.

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

- Mô hình lặp và tăng trưởng phát triển phần mềm theo từng đợt nhỏ, cho phép cập nhật và cải thiện dần theo phản hồi của người dùng.
- Mô hình thác nước là quy trình tuần tự, mỗi giai đoạn phải hoàn tất trước khi bước sang giai đoạn tiếp theo.
- Điểm khác biệt: Mô hình lặp và tăng trưởng linh hoạt hơn, giúp giảm rủi ro do có thể điều chỉnh theo yêu cầu mới, trong khi mô hình thác nước sẽ không quay lại các giai đoạn trước đó khiến cho việc thích nghi với những thay đổi của khách hàng.

4.Mục tiêu của pha bảo trì là gì?

- Sửa lỗi và cải thiện phần mềm sau khi triển khai.
- Cập nhật phần mềm để đáp ứng yêu cầu mới hoặc thay đổi công nghệ.
- Tối ưu hiệu suất, bảo mật và khả năng mở rộng của hệ thống.

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

- Khó kiểm soát chất lượng vì không có kế hoạch rõ ràng.
- Dễ dẫn đến sản phẩm không tối ưu do thiếu tài liệu và quy trình kiểm thử chuẩn.
- Bảo trì khó khăn, chi phí bảo trì có thể cao nếu không quản lý tốt.

6.Mô hình bản mẫu nhanh là gì?

- Là mô hình tập trung vào việc tạo nguyên mẫu (prototype) nhanh chóng để khách hàng đánh giá.
- Dựa trên phản hồi, nguyên mẫu sẽ được điều chỉnh và cải tiến dần cho đến khi đạt yêu cầu.

- Giúp giảm rủi ro do khách hàng có thể kiểm tra sản phẩm ngay từ đầu.

7.Pha giải thể là gì?

Pha giải thể (Retirement Phase) là giai đoạn phần mềm không còn được sử dụng (hết giá trị sử dụng) và bị thay thế hoặc ngừng hỗ trợ.

Các hoạt động của pha giải thể bao gồm:

- Sao lưu dữ liệu quan trọng.
- Gỡ bỏ phần mềm khỏi hệ thống.

Hỗ trợ chuyển đổi sang hệ thống mới nếu cần

8.Mô hình xoắn ốc là gì?

- Là mô hình phát triển phần mềm kết hợp giữa lặp và quản lý rủi ro.
- Chia quá trình phát triển thành nhiều vòng xoắn, mỗi vòng gồm: Xác định mục tiêu - Phân tích rủi ro - Phát triển và kiểm thử - Đánh giá và lập kế hoạch cho vòng tiếp theo
- Phù hợp với các dự án phức tạp, có nhiều rủi ro.

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

- Thích nghi nhanh với thay đổi yêu cầu từ khách hàng.
- Phát triển theo từng giai đoạn nhỏ giúp kiểm soát rủi ro.
- Khuyến khích sự hợp tác giữa nhóm phát triển và khách hàng.
- Tăng cường khả năng phản hồi và cải thiện liên tục.

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

- Mô hình mã nguồn mở: Phần mềm được phát triển bởi cộng đồng, mã nguồn công khai cho mọi người tham gia đóng góp.
- Điểm khác biệt chính: Mô hình mã nguồn mở có sự tham gia rộng rãi từ cộng đồng, linh hoạt và không tốn chi phí bản quyền. Trong khi đó những mô hình khác thường phải trả phí bản quyền. Tốc độ phát triển và khả năng thay đổi của mô hình mã nguồn mở thường cao hơn các mô hình khác.

Phần 3: Câu hỏi thảo luận nhóm

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

Mô hình thác nước:

Ưu điểm:

- Cấu trúc đơn giản, dễ hiểu và quản lý
- Các giai đoạn được xác định rõ ràng với đầu vào, đầu ra cụ thể
- Tài liệu đầy đủ sau mỗi giai đoạn
- Phù hợp với các dự án có yêu cầu ổn định, ít thay đổi
- Dễ dàng ước tính thời gian và chi phí

Nhược điểm:

- Thiếu linh hoạt, khó điều chỉnh khi yêu cầu thay đổi
- Khách hàng chỉ thấy được sản phẩm khi hoàn thành
- Rủi ro cao nếu phát hiện lỗi ở giai đoạn cuối
- Thời gian phát triển dài
- Không phù hợp với dự án phức tạp, có nhiều thay đổi

Mô hình xoắn ốc:

Ưu điểm:

- Tích hợp quản lý rủi ro vào từng chu kỳ phát triển
- Linh hoạt, dễ dàng điều chỉnh theo phản hồi
- Cho phép tạo nguyên mẫu liên tục và cải tiến
- Khách hàng tham gia đánh giá sau mỗi chu kỳ
- Phù hợp với dự án phức tạp, có nhiều yếu tố không chắc chắn

Nhược điểm:

- Phức tạp, đòi hỏi kỹ năng quản lý dự án cao
- Khó ước tính chính xác thời gian và chi phí
- Có thể kéo dài nếu không kiểm soát tốt các vòng lặp
- Đòi hỏi chuyên gia đánh giá rủi ro
- Chi phí cao hơn do quá trình lặp và xử lý rủi ro

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

Mô hình lặp và tăng trưởng có thể phù hợp với các tình huống:

1. Phát triển ứng dụng di động:

- Thị trường app di động thay đổi nhanh chóng
- Phản hồi người dùng liên tục giúp cải thiện UX/UI
- Phát hành phiên bản tăng trưởng theo từng giai đoạn

2. Nâng cấp hệ thống thương mại điện tử:

- Hệ thống đang hoạt động cần nâng cấp liên tục
- Tính năng mới được thêm vào theo độ ưu tiên
- Mỗi lần lặp tập trung vào một nhóm chức năng (thanh toán, tìm kiếm, giỏ hàng)

3. Phát triển phần mềm doanh nghiệp:

- Yêu cầu phức tạp và thường xuyên thay đổi
- Cần phản hồi từ nhiều bên liên quan
- Triển khai từng phân hệ một cách độc lập và tích hợp dần

4. Chuyển đổi hệ thống legacy:

- Chuyển đổi từng phần của hệ thống cũ
- Kiểm thử kỹ lưỡng từng phần trước khi chuyển đổi phần tiếp theo
- Giảm thiểu rủi ro và gián đoạn dịch vụ

Mô hình này đặc biệt hiệu quả khi sản phẩm cần được đưa ra thị trường nhanh chóng, sau đó cải tiến dựa trên phản hồi thực tế, hoặc khi yêu cầu không thể xác định đầy đủ ngay từ đầu.

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

Mô hình xây và sửa (build-and-fix) không phù hợp với dự án lớn vì:

- Thiếu kế hoạch tổng thể: Dự án lớn đòi hỏi tầm nhìn và kiến trúc tổng thể, trong khi xây và sửa tập trung vào giải quyết vấn đề tức thời.

- Liên tục sửa chữa mà không có thiết kế ban đầu tốt dẫn đến code cồng kềnh, khó bảo trì.

- Chi phí sửa lỗi cao: Trong dự án lớn, chi phí sửa lỗi tăng theo cấp số nhân khi quy mô phần mềm tăng.

- Khó phối hợp nhóm: Dự án lớn có nhiều nhóm làm việc đồng thời, cần quy trình rõ ràng để phối hợp.

- Tài liệu không đầy đủ: Mô hình xây và sửa thường không chú trọng tài liệu, gây khó khăn cho bảo trì dự án lớn.

- Khó kiểm soát tiến độ: Không có các cột mốc và đánh giá định kỳ, khó đánh giá tiến độ dự án.

- Rủi ro cao về chất lượng: Thiếu quy trình đảm bảo chất lượng chính thức dẫn đến sản phẩm không ổn định.

- Khó mở rộng: Khi dự án phát triển, cấu trúc không được thiết kế từ đầu sẽ khó mở rộng.

- Khó ước tính: Việc lặp đi lặp lại quá trình sửa chữa khiến không thể ước tính chính xác thời gian và ngân sách.

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

Mô hình bản mẫu nhanh (Rapid Prototyping):

- Tập trung vào việc tạo nhanh một bản mẫu có thể hoạt động

- Chủ yếu dùng để làm rõ yêu cầu và khám phá giải pháp

- Bản mẫu thường được loại bỏ sau khi thu thập đủ thông tin

- Tương tác với khách hàng ở giai đoạn đầu để xác nhận yêu cầu

- Quá trình phát triển thực tế vẫn thường theo quy trình truyền thống

Mô hình tiến trình linh hoạt (Agile):

- Phát triển lặp đi lặp lại với chu kỳ ngắn (sprint)

- Sản phẩm làm việc được sau mỗi chu kỳ, có giá trị thực tế

- Code được cải tiến liên tục, không loại bỏ

- Khách hàng tham gia xuyên suốt quá trình

- Ưu tiên phần mềm hoạt động hơn tài liệu đầy đủ

- Chấp nhận thay đổi, thậm chí ở giai đoạn cuối

Khác biệt chính:

Mục đích: Bản mẫu nhanh tập trung vào khám phá yêu cầu, Agile tập trung vào phát triển sản phẩm có giá trị.

Kết quả: Bản mẫu thường bị loại bỏ, Agile tạo ra phần mềm làm việc được.

Phạm vi: Bản mẫu thường chỉ là một phần của quy trình lớn hơn, Agile là một phương pháp phát triển hoàn chỉnh.

Thời gian: Bản mẫu thường ngắn hạn, Agile áp dụng xuyên suốt dự án.

Kiến trúc: Bản mẫu ít quan tâm đến kiến trúc, Agile coi trọng thiết kế đơn giản và tái cấu trúc.

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

Quản lý rủi ro là trọng tâm của mô hình xoắn ốc. Các vai trò cụ thể bao gồm:

- Định hướng phát triển: Rủi ro được sử dụng để quyết định ưu tiên phát triển tính năng nào trước, giảm thiểu những rủi ro lớn nhất càng sớm càng tốt.

- Xác định quy trình phù hợp: Mỗi chu kỳ chọn quy trình phát triển dựa trên phân tích rủi ro (nguyên mẫu, thác nước, v.v).

- Mỗi vòng xoắn được thực hiện để giải quyết các rủi ro đã xác định trước đó.

- Tiêu chí đánh giá: Thành công của mỗi chu kỳ được đánh giá dựa trên việc giảm thiểu rủi ro.

- Định nghĩa mức chi tiết: Mức độ chi tiết trong phân tích, thiết kế và triển khai phụ thuộc vào rủi ro còn lại.

- Quyết định tiếp tục hay dừng: Phân tích rủi ro giúp quyết định có nên tiếp tục dự án hay không.

- Truyền thông với các bên liên quan: Rủi ro là ngôn ngữ chung giúp các bên hiểu lý do cho quyết định kỹ thuật.

- Tối ưu hóa nguồn lực: Nguồn lực được phân bổ vào những lĩnh vực có rủi ro cao nhất.

- Cải tiến liên tục: Qua mỗi vòng xoắn, kinh nghiệm quản lý rủi ro được tích lũy và cải thiện.

Mô hình xoắn ốc đặc biệt phù hợp với các dự án có nhiều yếu tố không chắc chắn, chi phí thất bại cao, hoặc yêu cầu độ tin cậy cao.

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

Nên sử dụng mô hình thác nước trong các trường hợp:

- Yêu cầu ổn định và rõ ràng: Khi yêu cầu được xác định đầy đủ từ đầu và ít thay đổi.

- Dự án có tính pháp lý hoặc quy định cao: Ngành như y tế, hàng không, quốc phòng đòi hỏi tài liệu đầy đủ và tuân thủ quy định nghiêm ngặt.

- Hệ thống quan trọng về an toàn: Các hệ thống mà lỗi có thể gây hậu quả nghiêm trọng (thiết bị y tế, điều khiển nhà máy).

- Nhóm phát triển phân tán địa lý: Khi nhóm làm việc ở nhiều nơi khác nhau, cần quy trình và tài liệu rõ ràng.

- Hợp đồng cố định: Khi khách hàng yêu cầu cam kết phạm vi, thời gian và chi phí cố định từ đầu.

- Công nghệ ổn định: Khi sử dụng công nghệ đã được chứng minh, ít rủi ro kỹ thuật.

- Dự án quy mô nhỏ, ngắn hạn: Dự án đơn giản có thể hoàn thành trong thời gian ngắn.

- Thiếu sự tham gia liên tục từ khách hàng: Khi khách hàng không thể tham gia thường xuyên vào quá trình phát triển.

- Đội ngũ thiếu kinh nghiệm với phương pháp linh hoạt: Khi nhóm quen với quy trình truyền thống và cần thời gian để chuyển đổi.

- Khả năng tiên đoán cao: Khi có thể dự đoán trước các thách thức và giải pháp.

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

Áp dụng mô hình mã nguồn mở trong phát triển phần mềm gặp nhiều thách thức:

- Quản lý cộng đồng: Khó điều phối và duy trì động lực cho người đóng góp tình nguyện từ nhiều nền tảng văn hóa và múi giờ khác nhau.

- Đảm bảo chất lượng: Thiếu cơ chế kiểm soát chất lượng chính thức có thể dẫn đến code không đồng nhất.

- Tính bền vững: Dự án có thể mất đà nếu các nhà phát triển chính rời đi, thiếu nguồn tài trợ liên tục.

- Tích hợp đóng góp: Đánh giá và tích hợp đóng góp từ nhiều nguồn đòi hỏi thời gian và nguồn lực.

- Lộ trình phát triển: Khó duy trì lộ trình rõ ràng khi có nhiều ý kiến về hướng đi của dự án.

- Bảo mật: Mã nguồn mở có thể bị khai thác lỗ hổng dễ dàng hơn vì mã nguồn công khai.

- Sở hữu trí tuệ: Quản lý giấy phép và đảm bảo không xâm phạm bản quyền là thách thức phức tạp.

- Tương thích ngược: Duy trì tương thích với phiên bản cũ khi cộng đồng muốn đổi mới.

- Tài liệu: Thường thiếu tài liệu chất lượng cao vì ít người thích viết tài liệu.

- Mô hình kinh doanh: Khó xây dựng mô hình kinh doanh bền vững dựa trên phần mềm miễn phí.

- Hỗ trợ người dùng: Cung cấp hỗ trợ chuyên nghiệp và kịp thời là thách thức lớn.

- Kỳ vọng của người dùng: Người dùng có thể có kỳ vọng cao về hỗ trợ và tính năng dù không trả phí.

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

Mô hình tiến trình linh hoạt (Agile) cải thiện chất lượng phần mềm thông qua:

- Phản hồi liên tục: Chu kỳ phát triển ngắn cho phép thu thập phản hồi sớm và thường xuyên, giúp phát hiện và sửa lỗi nhanh chóng.

- Kiểm thử tích hợp: Kiểm thử liên tục và tích hợp liên tục giúp phát hiện lỗi sớm khi chi phí sửa chữa còn thấp.

- Tái cấu trúc thường xuyên: Việc liên tục cải thiện code giúp duy trì chất lượng kỹ thuật và giảm nợ kỹ thuật.

- Phát triển hướng kiểm thử : Viết test trước khi viết code giúp làm rõ yêu cầu và đảm bảo độ bao phủ kiểm thử cao.

- Lập trình /đánh giá code: Hợp tác chặt chẽ giữa các lập trình viên giúp phát hiện lỗi sớm và chia sẻ kiến thức.

- Tập trung vào giá trị kinh doanh: Ưu tiên các tính năng có giá trị cao nhất, đảm bảo phần mềm đáp ứng nhu cầu thực tế.

- Đội ngũ có quyền quyết định kỹ thuật, tăng trách nhiệm và cam kết với chất lượng.

- Họp, trao đổi với khách hàng thường xuyên: Giúp phát hiện vấn đề nhanh chóng và phối hợp giải quyết.

- Nhìn lại quy trình (Retrospective): Cải tiến liên tục quy trình làm việc dựa trên kinh nghiệm thực tế.

- Tiêu chí chất lượng rõ ràng cho mỗi tính năng trước khi được coi là hoàn thành.

- Liên tục tích hợp phản hồi người dùng: Đảm bảo sản phẩm đáp ứng nhu cầu thực tế, không chỉ đáp ứng đặc tả.

- Quản lý kỹ thuật: Xác định và giải quyết kỹ thuật một cách chủ động.

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

Pha bảo trì trong vòng đời phát triển phần mềm có vai trò then chốt:

- Kéo dài tuổi thọ sản phẩm: Bảo trì giúp phần mềm hoạt động hiệu quả lâu dài, tối đa hóa giá trị đầu tư.

- Đảm bảo độ tin cậy: Sửa lỗi và nâng cao tính ổn định của hệ thống qua thời gian.

- Thích ứng với thay đổi: Cập nhật phần mềm để đáp ứng thay đổi về môi trường, pháp lý, công nghệ.

- Tối ưu hóa hiệu suất: Cải thiện hiệu suất hệ thống dựa trên dữ liệu sử dụng thực tế.

- Cải thiện bảo mật: Phát hiện và khắc phục các lỗ hổng bảo mật mới xuất hiện.

- Phản hồi người dùng: Giải quyết vấn đề và cải thiện dựa trên phản hồi của người dùng.

- Chi phí vòng đời: Bảo trì chiếm 60-80% tổng chi phí vòng đời phần mềm, là giai đoạn tốn kém nhất.

- Cung cấp thông tin quý giá cho phiên bản mới của phần mềm.

- Phân loại bảo trì:

- Bảo trì sửa chữa: Sửa lỗi
- Bảo trì thích ứng: Điều chỉnh theo môi trường thay đổi
- Bảo trì hoàn thiện: Cải thiện hiệu suất, tính năng
- Bảo trì phòng ngừa: Ngăn chặn vấn đề tiềm ẩn

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

Đề xuất sử dụng mô hình xoắn ốc kết hợp với các yếu tố của RUP (Rational Unified Process) cho dự án phát triển phần mềm ngân hàng.

Lý do:

- Quản lý rủi ro: Hệ thống ngân hàng đòi hỏi độ tin cậy cao và không thể chấp nhận lỗi nghiêm trọng. Mô hình xoắn ốc đặt quản lý rủi ro là trọng tâm.

- Yêu cầu phức tạp: Phần mềm ngân hàng có nhiều yêu cầu nghiệp vụ phức tạp, cần được phân tích kỹ lưỡng và xác nhận qua nhiều chu kỳ.

- Tuân thủ quy định: Ngành ngân hàng chịu sự quản lý chặt chẽ, cần tài liệu đầy đủ và quy trình rõ ràng mà RUP cung cấp.

- Tích hợp liên tục: Hệ thống ngân hàng thường cần tích hợp với nhiều hệ thống khác, đòi hỏi kiểm thử tích hợp liên tục.

- Bảo mật cao: Mô hình xoắn ốc cho phép đánh giá và giải quyết rủi ro bảo mật trong từng chu kỳ.

- Phát triển từng phần: Có thể triển khai từng phần chức năng (module) một cách độc lập (ví dụ: thanh toán, kế toán, báo cáo).

- Kiểm thử nghiêm ngặt: Mô hình xoắn ốc và RUP đều nhấn mạnh kiểm thử toàn diện qua nhiều giai đoạn.

- Kiến trúc vững chắc: RUP chú trọng thiết kế kiến trúc từ sớm, quan trọng cho hệ thống có tính sẵn sàng cao.

- Khả năng mở rộng: Cho phép hệ thống phát triển theo thời gian khi có sản phẩm và dịch vụ tài chính mới.

- Cân bằng giữa linh hoạt và kiểm soát: Kết hợp được sự linh hoạt để thích ứng với thay đổi và kiểm soát chặt chẽ yêu cầu về chất lượng, tuân thủ.

Phần 4: Câu hỏi tình huống

Tình huống 1: Một công ty phát triển phần mềm theo mô hình thác nước gặp vấn đề khi khách hàng yêu cầu thay đổi sau khi hoàn thành pha thiết kế. Đội phát triển nên xử lý như thế nào?

- Đánh giá tác động: Phân tích mức độ ảnh hưởng của thay đổi đối với thiết kế hiện tại, chi phí và thời gian

- Lập tài liệu quản lý thay đổi: Ghi nhận chính thức yêu cầu thay đổi và tác động của nó

- Thương lượng với khách hàng: Thảo luận về chi phí, thời gian bổ sung và điều chỉnh hợp đồng nếu cần

- Thực hiện quy trình quản lý thay đổi: Cập nhật tài liệu thiết kế và thông báo cho tất cả các bên liên quan

- Áp dụng thay đổi có kiểm soát: Thực hiện thay đổi trong khi đảm bảo chất lượng và tính nhất quán

- Xem xét áp dụng một số yếu tố của phương pháp linh hoạt cho các giai đoạn tiếp theo để xử lý thay đổi hiệu quả hơn

Tình huống 2: Dự án phát triển phần mềm theo mô hình lặp và tăng trưởng liên tục bị trễ tiến độ do thiếu nhân lực. Là quản lý dự án, bạn sẽ làm gì?

- Ưu tiên lại các tính năng: Xác định các tính năng thiết yếu cho phiên bản hiện tại và xem xét dời các tính năng ít quan trọng sang phiên bản sau

- Tái phân bổ nguồn lực: Điều chuyển nhân sự từ các nhiệm vụ ít quan trọng sang những nhiệm vụ trọng yếu

- Thuê thêm nhân sự tạm thời: Xem xét thuê các chuyên gia hoặc freelancer để hỗ trợ trong thời gian ngắn

- Điều chỉnh phạm vi: Đàm phán với khách hàng để giảm phạm vi của phiên bản hiện tại

- Tăng cường tự động hóa: Tận dụng các công cụ tự động để giảm công việc thủ công

- Rút ngắn chu kỳ lặp: Giảm thời gian chu kỳ để đánh giá tiến độ thường xuyên hơn

- Cải thiện quy trình: Xác định và loại bỏ các quy trình không hiệu quả gây lãng phí thời gian

Tình huống 3: Trong quá trình phát triển phần mềm theo mô hình tiến trình linh hoạt, khách hàng không đưa ra phản hồi kịp thời. Đội phát triển nên xử lý ra sao?

- Xác định người có thẩm quyền ra quyết định từ phía khách hàng khi người chính không có mặt

- Thiết lập lịch trình rõ ràng: Thỏa thuận trước về thời điểm cần phản hồi và hậu quả nếu không nhận được phản hồi

- Triển khai nguyên tắc "mặc định đồng ý": Quy định rằng nếu không có phản hồi trong thời hạn, phương án đề xuất sẽ được thực hiện

- Tạo danh sách quyết định trì hoãn: Ưu tiên công việc không phụ thuộc vào phản hồi của khách hàng

- Tổ chức các cuộc họp demo ngắn và thường xuyên: Làm cho các cuộc họp dễ tham gia hơn và tập trung vào các quyết định cụ thể

- Sử dụng nguyên mẫu trực quan: Cung cấp mô hình trực quan thay vì mô tả bằng văn bản để rút ngắn thời gian phản hồi

- Nâng cấp kênh liên lạc: Sử dụng các công cụ giao tiếp phù hợp với lịch trình của khách hang

Tình huống 4: Khách hàng yêu cầu bổ sung một số tính năng mới khi phần mềm đã bước vào pha cài đặt. Nên áp dụng mô hình nào để xử lý tốt nhất yêu cầu này?

Mô hình phù hợp nhất là mô hình lặp và tăng trưởng hoặc mô hình tiến trình linh hoạt vì:

- Cho phép thêm tính năng mới vào các chu kỳ phát triển tiếp theo

- Có khả năng điều chỉnh ưu tiên và lập kế hoạch linh hoạt
- Cung cấp cơ chế để đánh giá tác động và tái lập kế hoạch
- Cho phép phân chia tính năng mới thành các phần nhỏ có thể quản lý
- Hỗ trợ triển khai từng phần theo thứ tự ưu tiên

Các bước thực hiện:

- Phân tích tác động của tính năng mới đối với hệ thống hiện tại
- Đánh giá độ ưu tiên của các tính năng mới
- Lập kế hoạch cho chu kỳ phát triển mới bao gồm các tính năng này
- Điều chỉnh kiến trúc hiện tại nếu cần
- Tích hợp các tính năng mới vào quy trình kiểm thử

Tình huống 5: Một công ty nhỏ muốn áp dụng mô hình bản mẫu nhanh nhưng gặp khó khăn do thiếu nguồn lực. Hãy đề xuất giải pháp.

- Sử dụng công cụ low-code/no-code: Tận dụng các nền tảng như Figma, Adobe XD, Bubble.io để tạo nguyên mẫu mà không cần nhiều lập trình viên

- Áp dụng phạm vi hẹp: Tập trung tạo nguyên mẫu cho các tính năng quan trọng nhất thay vì toàn bộ hệ thống

- Kỹ thuật "Wizard of Oz": Sử dụng giao diện giả lập mà không cần xây dựng logic phía sau hoàn chỉnh

- Tận dụng các thư viện và bộ khung có sẵn: Sử dụng các template và component có sẵn để giảm thời gian phát triển

- Thuê ngoài có chọn lọc: Thuê freelancer cho các công việc cụ thể trong quá trình tạo nguyên mẫu

- Phân chia nguyên mẫu: Phát triển nguyên mẫu theo các phần nhỏ thay vì một nguyên mẫu lớn

- Kết hợp với mô hình tiến trình linh hoạt tinh gọn: Áp dụng các nguyên tắc Agile tinh gọn với chu kỳ ngắn

Tình huống 6: Trong dự án phần mềm thương mại điện tử, khách hàng liên tục yêu cầu thay đổi giao diện. Mô hình nào sẽ phù hợp nhất?

Mô hình phù hợp nhất là mô hình tiến trình linh hoạt (Agile), vì:

- Chấp nhận thay đổi là nguyên tắc cốt lõi của Agile
- Chu kỳ phát triển ngắn cho phép điều chỉnh thường xuyên
- Đánh giá thường xuyên giúp khách hàng thấy kết quả và đưa ra phản hồi sớm
- Giao tiếp liên tục giảm thiểu hiểu lầm về yêu cầu của khách hàng
- Kiến trúc front-end linh hoạt cho phép thay đổi giao diện dễ dàng
- Tự động hóa kiểm thử giúp đảm bảo thay đổi giao diện không ảnh hưởng đến chức năng

Tình huống 7: Dự án phát triển phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau nên áp dụng mô hình nào?

Mô hình phù hợp nhất là kết hợp mô hình SAFe (Scaled Agile Framework) với DevOps vì:

- SAFe cung cấp khung làm việc cho việc mở rộng Agile ở quy mô doanh nghiệp
- Hỗ trợ nhiều nhóm phát triển làm việc đồng bộ với nhau
- Cung cấp cách quản lý danh mục công việc thống nhất
- DevOps giúp tự động hóa quy trình tích hợp và triển khai
- Tạo ra chu kỳ phát triển rõ ràng, khung thời gian nhất quán
- Định nghĩa rõ ràng về vai trò và trách nhiệm trên quy mô toàn cầu
- Nhấn mạnh tài liệu và tiêu chuẩn phát triển thống nhất
- Hỗ trợ giao tiếp bất đồng bộ giữa các múi giờ khác nhau
- Sử dụng cơ sở hạ tầng đám mây để hỗ trợ phát triển phân tán

Tình huống 8: Phân tích rủi ro là hoạt động chính trong mô hình xoắn ốc. Đề xuất cách giảm thiểu rủi ro khi áp dụng mô hình này.

Để giảm thiểu rủi ro trong mô hình xoắn ốc:

- Xây dựng hệ thống phân loại rủi ro: Phân loại rủi ro theo mức độ nghiêm trọng, khả năng xảy ra và tác động
- Sử dụng ma trận rủi ro: Lập ma trận trực quan để ưu tiên các rủi ro cần xử lý trước

- Thực hiện phân tích định lượng: Áp dụng các phương pháp định lượng như Monte Carlo để đánh giá tác động
- Lập kế hoạch dự phòng: Chuẩn bị phương án thay thế cho các rủi ro lớn
- Dự trữ nguồn lực: Dành dụm thời gian và ngân sách để xử lý các rủi ro phát sinh
- Đặt cột mốc đánh giá rủi ro: Xác định điểm kiểm tra để đánh giá lại các rủi ro trong mỗi vòng xoắn
- Tạo mẫu thử các thành phần rủi ro cao: Xây dựng nguyên mẫu cho các phần có rủi ro cao trước
- Tăng cường giao tiếp: Đảm bảo mọi bên liên quan hiểu rõ về rủi ro
- Sử dụng chỉ số theo dõi rủi ro: Thiết lập các chỉ số cụ thể để theo dõi diễn biến của rủi ro
- Áp dụng kiểm thử liên tục: Triển khai kiểm thử tự động để phát hiện sớm các vấn đề

Tình huống 9: Một dự án phát triển phần mềm ngân hàng cần yêu cầu bảo mật cao. Nên áp dụng mô hình nào để đảm bảo yêu cầu này?

Mô hình phù hợp nhất là mô hình xoắn ốc kết hợp với V-Model vì:

- Mô hình xoắn ốc đặt trọng tâm vào phân tích và giảm thiểu rủi ro, đặc biệt là rủi ro bảo mật
- V-Model nhấn mạnh kiểm thử tương ứng với mỗi giai đoạn phát triển
- Cho phép tài liệu hóa chi tiết, đáp ứng yêu cầu tuân thủ của ngành ngân hàng
- Hỗ trợ kiểm định và xác nhận từng bước
- Tích hợp quy trình đánh giá bảo mật vào mỗi chu kỳ phát triển
- Cho phép giải quyết vấn đề bảo mật từ giai đoạn đầu thiết kế
- Yêu cầu phê duyệt chính thức ở mỗi giai đoạn, tăng cường kiểm soát
- Hỗ trợ kiểm thử xâm nhập và đánh giá lỗ hổng có hệ thống

Các biện pháp tăng cường:

- Tích hợp DevSecOps vào quy trình
- Áp dụng nguyên tắc "bảo mật theo thiết kế"
- Thực hiện đánh giá bảo mật của bên thứ ba

- Tuân thủ các tiêu chuẩn bảo mật ngân hàng (PCI DSS, ISO 27001)

Tình huống 10: Khi nào nên kết thúc vòng đời phần mềm và thực hiện pha giải thể?

Nên kết thúc vòng đời phần mềm và thực hiện pha giải thể khi:

- Chi phí bảo trì vượt quá lợi ích: Khi chi phí duy trì phần mềm cao hơn giá trị mang lại
- Công nghệ lỗi thời: Nền tảng công nghệ không còn được hỗ trợ hoặc cập nhật
- Khó đáp ứng yêu cầu mới: Kiến trúc hiện tại không thể mở rộng để đáp ứng nhu cầu mới
- Xuất hiện giải pháp thay thế tốt hơn: Có sản phẩm thay thế hiệu quả hơn về chi phí hoặc tính năng
- Thay đổi chiến lược kinh doanh: Sản phẩm không còn phù hợp với định hướng kinh doanh
- Vấn đề bảo mật nghiêm trọng: Xuất hiện lỗ hổng bảo mật không thể khắc phục
- Giảm sút người dùng: Số lượng người dùng giảm đáng kể
- Ngân sách bị cắt giảm: Không còn đủ ngân sách để duy trì hệ thống