Bài tập chương 3

3.1 CÂU HỎI TRẮC NGHIỆM

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
    - *Giải thích:* Pha phân tích là giai đoạn biến các yêu cầu của khách hàng thành đặc tả kỹ thuật chi tiết để làm cơ sở cho pha thiết kế và triển khai.
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
    - *Giải thích:* Mô hình lặp và tăng trưởng phát triển phần mềm theo từng vòng lặp nhỏ, mỗi vòng lặp bổ sung tính năng mới, giúp cải thiện sản phẩm theo thời gian.
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
    - *Giải thích:* Pha bảo trì giúp sửa lỗi, cải tiến và cập nhật tính năng để phần mềm hoạt động ổn định và phù hợp với yêu cầu mới.
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
    - *Giải thích:* Mô hình thác nước có trình tự cố định, thích hợp với dự án có yêu cầu rõ ràng từ đầu và ít thay đổi trong quá trình phát triển.
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
    - *Giải thích:* Mô hình xoắn ốc gồm nhiều vòng lặp, mỗi vòng bao gồm các giai đoạn phân tích rủi ro, lập kế hoạch, phát triển và kiểm thử.
6. Điểm yếu lớn nhất của mô hình xây và sửa là gì?
    - Đáp án: C. Khó kiểm soát chất lượng
    - *Giải thích:* Do thiếu kế hoạch chi tiết và quy trình chuẩn, mô hình này dễ dẫn đến lỗi, chi phí bảo trì cao và khó kiểm soát chất lượng phần mềm.
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

- *Giải thích:* Mô hình bản mẫu nhanh (Prototyping Model) giúp tạo ra các nguyên mẫu nhanh chóng để khách hàng kiểm tra và phản hồi.

8. Pha nào kết thúc vòng đời phát triển phần mềm?
   - Đáp án: A. Pha bảo trì
   - *Giải thích:* Pha bảo trì là giai đoạn cuối cùng, duy trì phần mềm hoạt động cho đến khi không còn được sử dụng.
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
   - *Giải thích:* Mô hình lặp và tăng trưởng chia quá trình phát triển thành các vòng lặp nhỏ, trong khi mô hình thác nước thực hiện theo trình tự cố định.
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
    - *Giải thích:* Mô hình tiến trình linh hoạt (Agile) có khả năng phản hồi nhanh với thay đổi nhờ phát triển theo từng giai đoạn nhỏ, có sự tham gia liên tục của khách hàng.

3.2 CÂU HỎI TRẢ LỜI NGẮN

1. Pha lấy yêu cầu là gì và có vai trò gì trong vòng đời phát triển phần mềm?
   - *Pha lấy yêu cầu* (Requirement Gathering) là giai đoạn đầu tiên trong vòng đời phát triển phần mềm, nơi nhóm phát triển thu thập thông tin từ khách hàng để xác định các yêu cầu của hệ thống.
   - Vai trò:
     - Xác định các yêu cầu chức năng và phi chức năng.
     - Giúp các bên liên quan hiểu rõ về phạm vi dự án.
     - Là cơ sở để thiết kế và phát triển phần mềm.
2. Mô hình thác nước hoạt động như thế nào?
   - Mô hình thác nước (Waterfall Model) hoạt động theo quy trình tuần tự, trong đó mỗi pha phải hoàn thành trước khi chuyển sang pha tiếp theo. Các pha chính gồm: Lấy yêu cầu, Phân tích, Thiết kế, Cài đặt (Lập trình), Kiểm thử, Triển khai, Bảo trì
   - Mô hình này phù hợp với các 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?
   - Điểm khác biệt chính:

        - Mô hình thác nước là quy trình tuần tự, không quay lại các pha trước.
        - Mô hình lặp và tăng trưởng phát triển theo từng phiên bản nhỏ, lặp lại nhiều lần, mỗi lần bổ sung hoặc cải tiến chức năng.
    - Ưu điểm của mô hình lặp và tăng trưởng:
        - Dễ dàng thay đổi và thích nghi với yêu cầu mới.
        - Phát triển phần mềm theo từng bước nhỏ, có thể kiểm thử và nhận phản hồi sớm.

4. Mục tiêu của pha bảo trì là gì?
    - Đảm bảo phần mềm hoạt động ổn định sau khi triển khai.
    - Sửa lỗi phát sinh trong quá trình sử dụng.
    - Cập nhật, nâng cấp tính năng để phù hợp với nhu cầu mới của người dùng.
    - Cải thiện hiệu suất và bảo mật phần mềm.
5. Mô hình xây và sửa có nhược điểm gì?
    - Khó kiểm soát chất lượng: Không có kế hoạch chi tiết, dễ phát sinh lỗi.
    - Chi phí bảo trì cao: Phần mềm thường cần nhiều lần sửa chữa và cải tiến.
    - Thiếu tài liệu và quy trình chuẩn: Gây khó khăn cho bảo trì và mở rộng hệ thống.
    - Không phù hợp với các dự án lớn hoặc phức tạp.
6. Mô hình bản mẫu nhanh là gì?
    - Mô hình bản mẫu nhanh (Rapid Prototyping) là phương pháp phát triển phần mềm bằng cách tạo ra một nguyên mẫu (prototype) nhanh chóng để thu thập phản hồi từ khách hàng.
    - Sau đó, nguyên mẫu được cải tiến nhiều lần trước khi phát triển phần mềm chính thức.
    - Ưu điểm:
        - Giúp khách hàng thấy trước sản phẩm.
        - Giảm rủi ro hiểu sai yêu cầu.
        - Tăng tốc độ phát triển.
7. Pha giải thể là gì?

- Pha giải thể (Disposal Phase) là giai đoạn kết thúc vòng đời của phần mềm khi nó không còn được sử dụng hoặc không còn đáp ứng nhu cầu.
- Các hoạt động chính gồm:
  - Gỡ bỏ phần mềm khỏi hệ thống.
  - Chuyển dữ liệu sang hệ thống mới.
  - Hủy bỏ tài liệu liên quan (nếu cần).

8. Mô hình xoắn ốc là gì?
   - Mô hình xoắn ốc (Spiral Model) là phương pháp phát triển phần mềm kết hợp giữa mô hình thác nước và mô hình lặp, nhấn mạnh vào phân tích rủi ro.
   - Mỗi vòng xoắn trong mô hình tương ứng với một chu kỳ phát triển, bao gồm các pha:
     1. Xác định yêu cầu
     2. Phân tích rủi ro
     3. Thiết kế và phát triển
     4. Đánh giá và kiểm thử
   - Ưu điểm:
     - Giảm thiểu rủi ro nhờ đánh giá sớm.
     - Thích hợp cho các dự án lớn và phức tạp.
9. Tại sao mô hình tiến trình linh hoạt được đánh giá cao?
   - Mô hình tiến trình linh hoạt (Agile) được đánh giá cao vì:
     - Thích nghi nhanh với thay đổi: Linh hoạt trong việc cập nhật yêu cầu mới.
     - Phát triển theo từng giai đoạn ngắn: Có thể kiểm thử và phản hồi sớm.
     - Tăng cường sự hợp tác: Khách hàng, nhóm phát triển và các bên liên quan làm việc chặt chẽ.
     - Nâng cao chất lượng phần mềm: Nhờ kiểm thử liên tục và phản hồi thường xuyên.
10. Điểm khác biệt chính giữa mô hình mã nguồn mở và các mô hình khác là gì?

- Mô hình mã nguồn mở (Open Source Model):
  - Mã nguồn được công khai, bất kỳ ai cũng có thể sử dụng, sửa đổi và đóng góp.

    - Phát triển bởi cộng đồng thay vì một công ty duy nhất.
- Khác với các mô hình khác:
    - Các mô hình truyền thống (như thác nước, xoắn ốc, Agile) thường có quy trình phát triển do một nhóm hoặc công ty kiểm soát.
    - Mô hình mã nguồn mở có tính minh bạch cao, nhận được đóng góp từ nhiều lập trình viên trên toàn thế giới.

3.3. CÂU HỎI THẢO LUẬN NHÓM

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

Tiêu chí	Mô hình thác nước	Mô hình xoắn ốc
Ưu điểm	- Dễ hiểu, rõ ràng, có kế hoạch cụ thể. - Phù hợp với dự án có yêu cầu cố định. - Dễ quản lý tiến độ do có từng pha cụ thể.	- Tập trung vào quản lý rủi ro. - Linh hoạt hơn, có thể thay đổi theo từng vòng lặp. - Phù hợp với dự án lớn, phức tạp.
Nhược điểm	- Khó thay đổi yêu cầu sau khi đã bắt đầu phát triển. - Chỉ kiểm thử sau khi hoàn thành toàn bộ hệ thống. - Không phù hợp với dự án yêu cầu linh hoạt.	- Chi phí cao hơn do phải thực hiện nhiều vòng lặp. - Đòi hỏi chuyên gia để phân tích rủi ro. - Thời gian phát triển dài hơn.

2. Tình huống thực tế có thể áp dụng mô hình lặp và tăng trưởng

- Dự án phát triển website thương mại điện tử
    - Bắt đầu với một phiên bản đơn giản (chỉ hiển thị sản phẩm).
    - Sau đó bổ sung giỏ hàng, thanh toán, đánh giá sản phẩm qua các lần lặp tiếp theo.
- Phát triển ứng dụng di động
    - Ra mắt phiên bản đầu tiên với chức năng cơ bản.
    - Thu thập phản hồi từ người dùng, sau đó cập nhật và mở rộng.

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

- Không có kế hoạch rõ ràng → Dễ gây chồng chéo và mất kiểm soát khi mở rộng hệ thống.

- Khó kiểm soát lỗi và chất lượng → Do sửa chữa liên tục mà không có quy trình rõ ràng.
- Chi phí bảo trì cao → Do không có tài liệu đầy đủ, khó mở rộng và nâng cấp.
- Không phù hợp với phần mềm có yêu cầu bảo mật cao → Vì không có kiểm soát chặt chẽ từ đầu.

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

Tiêu chí	Mô hình bản mẫu nhanh	Mô hình tiến trình linh hoạt
Mục tiêu	Tạo nguyên mẫu nhanh để lấy phản hồi từ khách hàng.	Phát triển phần mềm theo từng giai đoạn nhỏ và thích nghi liên tục.
Phương pháp	Xây dựng một mô hình thử nghiệm, sau đó cải tiến dần.	Phát triển phần mềm theo từng sprint (vòng lặp ngắn).
Ưu điểm	- Giảm rủi ro hiểu sai yêu cầu. - Thấy trước sản phẩm sớm.	- Linh hoạt với thay đổi yêu cầu. - Liên tục cải thiện chất lượng.
Nhược điểm	- Nguyên mẫu ban đầu có thể gây hiểu nhầm về sản phẩm cuối. - Không phù hợp với hệ thống lớn, phức tạp.	- Khó áp dụng nếu khách hàng không có thời gian tham gia thường xuyên. - Yêu cầu kỹ năng cao của nhóm phát triển.

5. Vai trò của quản lý rủi ro trong mô hình xoắn ốc

- Giúp phát hiện rủi ro sớm → Phân tích và giảm thiểu rủi ro trước khi đi vào các pha tiếp theo.
- Tăng độ tin cậy → Nhờ đánh giá định kỳ, phần mềm có độ ổn định cao hơn.
- Giảm chi phí phát triển → Vì rủi ro được giải quyết từ đầu thay vì sửa lỗi về sau.
- Thích hợp với dự án lớn, phức tạp → Đặc biệt là phần mềm quan trọng như hàng không, y tế, tài chính.

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

- Khi yêu cầu ổn định, không thay đổi.
- Khi cần một kế hoạch phát triển có cấu trúc rõ ràng.
- Khi đội ngũ phát triển không quen với mô hình linh hoạt.

- Khi dự án có thời gian cố định và cần kiểm soát chi phí chặt chẽ (ví dụ: hệ thống quản lý tài chính).

7. Những khó khăn khi áp dụng mô hình mã nguồn mở

- Khó kiểm soát chất lượng → Vì nhiều lập trình viên đóng góp với tiêu chuẩn khác nhau.
- Bảo mật → Mã nguồn mở có thể bị khai thác nếu không kiểm soát tốt.
- Thiếu hỗ trợ chính thức → Không có công ty đứng sau bảo trì lâu dài.
- Khó kiếm lợi nhuận trực tiếp → Vì người dùng có thể sử dụng miễn phí.

8. Cách mô hình tiến trình linh hoạt giúp cải thiện chất lượng phần mềm

- Liên tục kiểm thử → Mỗi sprint đều có giai đoạn kiểm thử và phản hồi.
- Phát triển dựa trên phản hồi → Người dùng tham gia vào quá trình phát triển.
- Cải tiến liên tục → Phát hiện lỗi sớm và sửa ngay thay vì chờ đến cuối dự án.
- Tăng tính hợp tác → Nhóm phát triển và khách hàng làm việc chặt chẽ với nhau.

9. Vai trò của pha bảo trì trong vòng đời phát triển phần mềm

- Sửa lỗi → Đảm bảo hệ thống hoạt động ổn định.
- Nâng cấp và mở rộng → Bổ sung tính năng mới theo nhu cầu.
- Tối ưu hóa hiệu suất → Cải thiện tốc độ và khả năng xử lý.
- Bảo mật → Vá lỗ hổng bảo mật để bảo vệ dữ liệu.

10. Đề xuất mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng

- Mô hình phù hợp: Mô hình xoắn ốc
- Lý do:
    - Rủi ro cao → Ngành tài chính đòi hỏi mức độ bảo mật và độ tin cậy cao.
    - Yêu cầu chặt chẽ → Cần tuân thủ các quy định pháp lý và tiêu chuẩn bảo mật.
    - Cần kiểm thử kỹ lưỡng → Tránh sai sót trong giao dịch tài chính.
    - Có thể phát triển theo từng giai đoạn → Mô hình xoắn ốc cho phép đánh giá và tối ưu hóa từng phần trước khi triển khai chính thức.

3.4 CÂU HỎI TÌNH HUỐNG

1. Công ty phát triển theo mô hình thác nước gặp vấn đề khi khách hàng yêu cầu thay đổi sau pha thiết kế.

Cách xử lý:

- Đánh giá mức độ ảnh hưởng: Xác định thay đổi có ảnh hưởng đến toàn bộ dự án không.
- Ước tính chi phí và thời gian: Nếu thay đổi lớn, cần điều chỉnh kế hoạch, hợp đồng và ngân sách.
- Đề xuất giải pháp:
    - Nếu thay đổi nhỏ → Có thể điều chỉnh trong pha sau (ví dụ: pha bảo trì).
    - Nếu thay đổi lớn → Cân nhắc chuyển sang mô hình xoắn ốc hoặc lặp và tăng trưởng.
- Thương lượng với khách hàng: Xác định phạm vi thay đổi và ảnh hưởng đến tiến độ.

2. Dự án theo mô hình lặp và tăng trưởng bị trễ tiến độ do thiếu nhân lực.

Cách xử lý:

- Tái phân bổ nguồn lực: Ưu tiên các tính năng quan trọng, giảm bớt công việc không cần thiết.
- Tuyển dụng bổ sung: Thuê thêm nhân sự (toàn thời gian hoặc freelancer) nếu có thể.
- Điều chỉnh kế hoạch: Kéo dài thời gian hoàn thành hoặc giảm bớt phạm vi từng lần lặp.
- Tự động hóa quy trình: Sử dụng công cụ CI/CD để tăng tốc kiểm thử và triển khai.

3. Trong mô hình tiến trình linh hoạt, khách hàng không đưa ra phản hồi kịp thời.

Cách xử lý:

- Tạo lịch trình cố định cho phản hồi: Yêu cầu khách hàng tham gia các cuộc họp định kỳ.
- Gửi báo cáo tự động: Cung cấp bản demo hoặc báo cáo tiến độ để khách hàng có thể xem nhanh.
- Chủ động đề xuất giải pháp: Nếu khách hàng chậm phản hồi, nhóm phát triển có thể đưa ra phương án thay thế dựa trên kinh nghiệm.
- Thỏa thuận trách nhiệm: Làm rõ trách nhiệm của khách hàng trong việc cung cấp phản hồi đúng hạn.

4. Khách hàng yêu cầu bổ sung tính năng khi phần mềm đã bước vào pha cài đặt.

Mô hình phù hợp: Mô hình xoắn ốc hoặc mô hình tiến trình linh hoạt

- Mô hình xoắn ốc: Cho phép thay đổi ở các vòng lặp tiếp theo mà không ảnh hưởng đến chất lượng.

- Mô hình tiến trình linh hoạt: Nếu nhóm phát triển đang sử dụng Agile/Scrum, có thể bổ sung tính năng vào backlog và triển khai trong sprint tiếp theo.

5. Công ty nhỏ muốn áp dụng mô hình bản mẫu nhanh nhưng gặp khó khăn do thiếu nguồn lực.

Giải pháp:

- Tập trung vào các tính năng cốt lõi: Không cần làm nguyên mẫu cho toàn bộ phần mềm, chỉ cần xây dựng các tính năng quan trọng.
- Sử dụng công cụ thiết kế nhanh: Dùng Figma, Adobe XD hoặc low-code để giảm thời gian phát triển mẫu.
- Hợp tác với khách hàng: Yêu cầu khách hàng hỗ trợ xác định yêu cầu để tránh sửa đổi nhiều lần.
- Thuê ngoài một số phần công việc: Nếu cần, có thể thuê freelancer để hỗ trợ thiết kế giao diện hoặc lập trình mẫu thử.

6. Dự án thương mại điện tử, khách hàng liên tục yêu cầu thay đổi giao diện.

Mô hình phù hợp: Mô hình tiến trình linh hoạt (Agile, Scrum)

- Linh hoạt thay đổi: Scrum cho phép thay đổi yêu cầu giữa các sprint.
- Phản hồi nhanh: Giao diện có thể được cập nhật dựa trên phản hồi của khách hàng mà không ảnh hưởng đến phần backend.
- Tối ưu trải nghiệm người dùng: Cho phép kiểm thử và cải tiến liên tục.

7. Dự án phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau.

Mô hình phù hợp: Mô hình xoắn ốc hoặc mô hình tiến trình linh hoạt kết hợp với phương pháp Scaled Agile (SAFe).

- Mô hình xoắn ốc: Phù hợp với dự án lớn, đảm bảo kiểm soát rủi ro.
- Scaled Agile Framework (SAFe):
    - Chia dự án thành nhiều nhóm nhỏ, mỗi nhóm có sprint riêng.
    - Sử dụng Agile Release Train để đồng bộ tiến độ giữa các nhóm.
- Tích hợp công cụ quản lý từ xa: Jira, Confluence, Slack giúp nhóm làm việc hiệu quả dù ở các quốc gia khác nhau.

8. Đề xuất cách giảm thiểu rủi ro khi áp dụng mô hình xoắn ốc.

Giải pháp:

- Phân tích rủi ro sớm: Xác định và đánh giá rủi ro ngay từ vòng lặp đầu tiên.

- Chia nhỏ dự án thành nhiều giai đoạn: Giúp giảm thiểu tác động nếu có vấn đề xảy ra.
- Thử nghiệm liên tục: Kiểm thử từng phần để phát hiện lỗi sớm.
- Tham vấn chuyên gia: Đối với rủi ro lớn, có thể thuê chuyên gia phân tích bảo mật, hiệu suất.
- Tài liệu hóa rủi ro: Ghi lại rủi ro tiềm ẩn và cách xử lý để tránh lặp lại trong các vòng tiếp theo.

9. Dự án phần mềm ngân hàng cần yêu cầu bảo mật cao.

Mô hình phù hợp: Mô hình xoắn ốc

- Lý do:
    - Quản lý rủi ro tốt: Ngân hàng có yêu cầu bảo mật nghiêm ngặt, mô hình xoắn ốc giúp kiểm soát chặt chẽ.
    - Kiểm thử liên tục: Giúp phát hiện và khắc phục lỗ hổng bảo mật ngay từ giai đoạn đầu.
    - Phát triển từng phần: Cho phép tích hợp các tiêu chuẩn bảo mật theo từng vòng lặp.
    - Phù hợp với quy định pháp lý: Đảm bảo tuân thủ các tiêu chuẩn ngành như PCI-DSS, ISO 27001.

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

Khi nào nên giải thể phần mềm?

- Không còn phù hợp với nhu cầu người dùng.
- Chi phí duy trì cao hơn lợi ích.
- Không thể nâng cấp do công nghệ quá cũ.
- Có phần mềm mới thay thế tốt hơn.

Các bước trong pha giải thể:

1. Thông báo cho khách hàng và người dùng về việc ngừng hỗ trợ.
2. Chuyển dữ liệu quan trọng sang hệ thống mới (nếu có).
3. Gỡ bỏ hệ thống một cách an toàn để tránh rủi ro bảo mật.
4. Lưu trữ tài liệu và mã nguồn để tham khảo trong tương lai.