**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ**

**THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN 2**





**BÁO CÁO**

**MÔN HỌC: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM**

**BÀI TẬP CHƯƠNG 3: MỘT SỐ MÔ HÌNH VÒNG ĐỜI   
PHÁT TRIỂN PHẦN MỀM**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:** | **Gv. Châu Văn Vân** |
| **Nhóm sinh viên thực hiện:** | **5** |
| **Lớp:** | **D22CQAT01** |
| **Thành viên nhóm:** | **Phạm Đình Nam - N21DCCN054**  **Vũ Huy Hùng – N20DCCN020** |
|  | **Hồ Xuân Thiện – N22DCAT051**  **Lê Anh Dũng – N22DCAT011**  **Nguyễn Duyên Duy – N22DCAT013** |

***TP.HCM, 3/2025***

LỜI CẢM ƠN

Em chân thành cảm ơn Thầy Châu Văn Vân, người đã tận tình hướng dẫn và giúp đỡ em trong quá trình học môn Nhập Môn Công Nghệ Phần Mềm. Kiến thức và kỹ năng thu được từ môn học này cũng sẽ hỗ trợ em trong việc làm việc nhóm và quản lý dự án, xây dựng phần mềm trong tương lai.

Em cũng muốn bày tỏ lòng biết ơn đến Ban lãnh đạo và các giảng viên tại Học Viện Công Nghệ Bưu Chính Viễn Thông, đã tạo điều kiện và cung cấp cơ sở vật chất để chúng em có môi trường học tập và rèn luyện.

Dù em nhận thức rằng đề tài của mình có những hạn chế và thiếu sót do kiến thức và kinh nghiệm còn hạn chế, và thời gian nghiên cứu có hạn, nhưng em hy vọng nhận được sự đóng góp ý kiến của thầy để có thêm kiến thức và kinh nghiệm, từ đó hoàn thiện đề tài một cách tốt nhất.

Em xin trân trọng cảm ơn!

*Thành phố Hồ Chí Minh, tháng 3 năm 2025*

# BẢNG PHÂN CÔNG CÔNG VIỆC

SINH VIÊN: **Nguyễn Duyên Duy**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên** | **Phụ trách công việc** | **Thời gian** |
| 1 | Phạm Đình Nam  (Nhóm trưởng) | Phân công nhiệm vụ.  Thực hiện:   * Câu 1.1, 1.2 * Câu 2.1, 2.2 * Câu 3 * Câu 4.1, 4.2 | Phân công nhiệm vụ:  Ngày 3/3/2025.  Thực hiện:  Từ ngày 3/3.  Đến ngày 5/3.  Tổng hợp câu 3:  Ngày 6/3/2025  \*Note: Câu 3 là câu thảo luận nhóm nên cần tổng hợp lại. |
| 2 | Vũ Huy Hùng | Thực hiện:   * Câu 1.3, 1.4 * Câu 2.3, 2.4 * Câu 3 * Câu 4.3, 4.4 |
| 3 | Nguyễn Duyên Duy | Thực hiện:   * Câu 1.5, 1.6 * Câu 2.5, 2.6 * Câu 3 * Câu 4.5, 4.6 |
| 4 | Hồ Xuân Thiện | Thực hiện:   * Câu 1.7, 1.8 * Câu 2.7, 2.8 * Câu 3 * Câu 4.7, 4.8 |
| 5 | Lê Anh Dũng | Thực hiện:   * Câu 1.9, 1.10 * Câu 2.9, 2.10 * Câu 3 * Câu 4.9, 4.10 |

**MỤC LỤC**

[BẢNG PHÂN CÔNG CÔNG VIỆC 3](#_Toc192101166)

[PHẦN 1: CÂU HỎI TRẮC NGHIỆM 6](#_Toc192101167)

[Câu 1.1: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? 6](#_Toc192101168)

[Câu 1.2: Mô hình vòng đời nào phát triển phần mềm bằng cách tạo các phiên bản nhỏ và tăng dần tính năng? 6](#_Toc192101169)

[Câu 1.3: Pha bảo trì trong vòng đời phát triển phần mềm bao gồm hoạt động nào? 6](#_Toc192101170)

[Câu 1.4: Mô hình thác nước phù hợp nhất với loại dự án nào? 6](#_Toc192101171)

[Câu 1.5: Trong mô hình xoắn ốc, mỗi vòng xoắn tương ứng với: 6](#_Toc192101172)

[Câu 1.6: Điểm yếu lớn nhất của mô hình xây và sửa là gì? 6](#_Toc192101173)

[Câu 1.7:Mô hình nào tập trung vào việc tạo các nguyên mẫu nhanh để thu thập phản hồi từ khách hàng? 6](#_Toc192101174)

[Câu 1.8:Pha nào kết thúc vòng đời phát triển phần mềm? 7](#_Toc192101175)

[Câu 1.9: Điểm khác biệt chính giữa mô hình lặp và tăng trưởng với mô hình thác nước là gì? 7](#_Toc192101176)

[Câu 1.10: Mô hình nào có khả năng thích nghi tốt nhất với sự thay đổi của yêu cầu khách hàng? 7](#_Toc192101177)

[PHẦN 2: CÂU TRẢ LỜI NGẮN 7](#_Toc192101178)

[Câu 2.1: Pha lấy yêu cầu là gì và có vai trò gì trong vòng đời phát triển phần mềm? 7](#_Toc192101179)

[Câu 2.2: Mô hình thác nước hoạt động như thế nào? 8](#_Toc192101180)

[Câu 2.3: Mô hình lặp và tăng trưởng khác gì so với mô hình thác nước? 9](#_Toc192101181)

[Câu 2.4: Mục tiêu của pha bảo trì là gì? 10](#_Toc192101182)

[Câu 2.5:Mô hình xây và sửa có nhược điểm gì? 10](#_Toc192101183)

[Câu 2.6:Mô hình bản mẫu nhanh là gì? 10](#_Toc192101184)

[Câu 2.7:Pha giải thể là gì? 11](#_Toc192101185)

[Câu 2.8:Mô hình xoắn ốc là gì? 12](#_Toc192101186)

[Câu 2.9: Tại sao mô hình tiến trình linh hoạt được đánh giá cao? 13](#_Toc192101187)

[Câu 2.10: Điểm khác biệt chính giữa mô hình mã nguồn mở và các mô hình khác là gì? 13](#_Toc192101188)

[PHẦN 3: CÂU HỎI THẢO LUẬN NHÓM 13](#_Toc192101189)

[Câu 3.1: So sánh ưu và nhược điểm của mô hình thác nước và mô hình xoắn ốc. 13](#_Toc192101190)

[Câu 3.2: Thảo luận về tình huống thực tế có thể áp dụng mô hình lặp và tăng trưởng. 14](#_Toc192101191)

[Câu 3.3: Tại sao mô hình xây và sửa không phù hợp với các dự án lớn? 15](#_Toc192101192)

[Câu 3.4: So sánh giữa mô hình bản mẫu nhanh và mô hình tiến trình linh hoạt. 15](#_Toc192101193)

[Câu 3.5:Phân tích vai trò của quản lý rủi ro trong mô hình xoắn ốc. 16](#_Toc192101194)

[Câu 3.6:Khi nào nên sử dụng mô hình thác nước thay vì mô hình tiến trình linh hoạt? 17](#_Toc192101195)

[Câu 3.7:Thảo luận về những khó khăn khi áp dụng mô hình mã nguồn mở. 17](#_Toc192101196)

[Câu 3.8:Phân tích cách mô hình tiến trình linh hoạt giúp cải thiện chất lượng phần mềm. 18](#_Toc192101197)

[Câu 3.9:Thảo luận về vai trò của pha bảo trì trong vòng đời phát triển phần mềm. 19](#_Toc192101198)

[Câu 3.10:Đề xuất mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng và giải thích lý do. 20](#_Toc192101199)

[PHẦN 4: CÂU HỎI TÌNH HUỐNG 22](#_Toc192101200)

[Câu 4.1: 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? 22](#_Toc192101201)

[Câu 4.2: 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? 22](#_Toc192101202)

[Câu 4.3: 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? 23](#_Toc192101203)

[Câu 4.4: 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? 24](#_Toc192101204)

[Câu 4.5: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. 24](#_Toc192101205)

[Câu 4.6: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 25](#_Toc192101206)

[Câu 4.7:Dự án phát triển phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau nên áp dụng mô hình nào? 26](#_Toc192101207)

[Câu 4.8: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. 27](#_Toc192101208)

[Câu 4.9: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? 28](#_Toc192101209)

[Câu 4.10:Khi nào nên kết thúc vòng đời phần mềm và thực hiện pha giải thể? 29](#_Toc192101210)

# PHẦN 1: CÂU HỎI TRẮC NGHIỆM

## Câu 1.1: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?

A. Pha thiết kế

B. Pha lấy yêu cầu

**C. Pha phân tích**

D. Pha cài đặt

Đáp án C.

## Câu 1.2: Mô hình vòng đời nào phát triển phần mềm bằng cách tạo các phiên bản nhỏ và tăng dần tính năng?

A. Mô hình thác nước

**B. Mô hình lặp và tăng trưởng**

C. Mô hình bản mẫu nhanh

D. Mô hình tiến trình linh hoạt

Đáp án B.

## Câu 1.3: Pha bảo trì trong vòng đời phát triển phần mềm bao gồm hoạt động nào?

A. Viết mã nguồn

**B. Sửa lỗi và cập nhật tính năng mới**

C. Gỡ bỏ phần mềm

D. Phân tích yêu cầu

## Câu 1.4: Mô hình thác nước phù hợp nhất với loại dự án nào?

A. Dự án nhỏ, đơn giản

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

C. Dự án yêu cầu linh hoạt cao

D. Dự án mã nguồn mở

## Câu 1.5: Trong mô hình xoắn ốc, mỗi vòng xoắn tương ứng với:

A. Một pha kiểm thử

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

C. Một phiên bản phần mềm nhỏ

D. Một lần phân tích rủi ro

Đáp án: B

## Câu 1.6: Điểm yếu lớn nhất của mô hình xây và sửa là gì?

A. Khó phát triển nhanh

B. Tốn nhiều chi phí bảo trì

**C. Khó kiểm soát chất lượng**

D. Không phù hợp với dự án nhỏ

Đáp án: C

## Câu 1.7:Mô hình nào tập trung vào việc tạo các nguyên mẫu nhanh để thu thập phản hồi từ khách hàng?

A. Mô hình lặp và tăng trưởng

**B. Mô hình bản mẫu nhanh**

C. Mô hình xoắn ốc

D. Mô hình tiến trình linh hoạt

## Câu 1.8:Pha nào kết thúc vòng đời phát triển phần mềm?

A. Pha bảo trì

B. Pha cài đặt

**C. Pha giải thể**

D. Pha thiết kế

## Câu 1.9: Điểm khác biệt chính giữa mô hình lặp và tăng trưởng với mô hình thác nước là gì?

A. Mô hình thác nước lặp lại nhiều lần

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

C. Mô hình lặp và tăng trưởng không có giai đoạn kiểm thử

D. Mô hình thác nước không có giai đoạn bảo trì

Đáp án: B

## Câu 1.10: Mô hình nào có khả năng thích nghi tốt nhất với sự thay đổi của yêu cầu khách hàng?

A. Mô hình thác nước

B. Mô hình xoắn ốc

C. Mô hình xây và sửa

**D. Mô hình tiến trình linh hoạt**

Đáp án: D

# PHẦN 2: CÂU TRẢ LỜI NGẮN

## Câu 2.1: Pha lấy yêu cầu là gì và có vai trò gì trong vòng đời phát triển phần mềm?

Pha lấy yêu cầu (Requirements Elicitation):

Pha lấy yêu cầu là giai đoạn đầu tiên và quan trọng trong quy trình phát triển phần mềm, trong đó nhóm phát triển thu thập, xác định và làm rõ các yêu cầu của dự án từ khách hàng và các bên liên quan.

Vai trò của pha lấy yêu cầu:

* Xác định mục tiêu và phạm vi của dự án
* Hiểu rõ nhu cầu thực tế của người dùng
* Giảm thiểu rủi ro và sai sót trong quá trình phát triển
* Tạo nền tảng cho các giai đoạn tiếp theo như thiết kế và phát triển
* Giúp đảm bảo sản phẩm cuối cùng đáp ứng đúng mong đợi của khách hàng

Các kỹ thuật thường được sử dụng trong pha này bao gồm:

* Phỏng vấn trực tiếp
* Khảo sát
* Quan sát người dùng
* Phân tích tài liệu hiện có
* Thảo luận nhóm
* Xây dựng nguyên mẫu

## Câu 2.2: Mô hình thác nước hoạt động như thế nào?

Mô hình thác nước

Mô hình thác nước là một phương pháp tuần tự trong phát triển phần mềm, trong đó các giai đoạn được thực hiện một cách tuần tự và chỉ chuyển sang giai đoạn tiếp theo khi giai đoạn trước đã hoàn toàn hoàn thành.

Các giai đoạn chính:

1. **Thu thập và phân tích yêu cầu**
   * Xác định chi tiết các yêu cầu của hệ thống
   * Tạo tài liệu yêu cầu chi tiết
2. **Thiết kế hệ thống**
   * Thiết kế kiến trúc phần mềm
   * Xác định cấu trúc dữ liệu
   * Lựa chọn công nghệ
3. **Lập trình/Cài đặt**
   * Viết mã nguồn dựa trên thiết kế
   * Chuyển đổi các yêu cầu thành code
4. **Kiểm thử**
   * Kiểm tra chất lượng phần mềm
   * Phát hiện và sửa lỗi
   * Đảm bảo phần mềm hoạt động đúng yêu cầu
5. **Triển khai**
   * Cài đặt phần mềm tại môi trường người dùng
   * Hỗ trợ người dùng sử dụng
6. **Bảo trì**
   * Sửa chữa lỗi
   * Cập nhật các tính năng mới

Đặc điểm:

* Tuần tự và rõ ràng
* Dễ quản lý và hiểu
* Phù hợp với các dự án nhỏ có yêu cầu ổn định
* Khó thích ứng với các thay đổi

Hạn chế:

* Thiếu tính linh hoạt
* Khó điều chỉnh khi yêu cầu thay đổi
* Kiểm thử muộn
* Rủi ro cao nếu có sai sót ở các giai đoạn đầu

## Câu 2.3: Mô hình lặp và tăng trưởng khác gì so với mô hình thác nước?

|  |  |  |
| --- | --- | --- |
|  | **Mô hình Thác nước** | **Mô hình lặp và tăng trưởng** |
| Tài liệu | Cần tài liệu chi tiết là bắt buộc. | Cần tài liệu chi tiết nhưng không quá nhiều. |
| Lập kế hoạch | Cần lập kế hoạch chi tiết từ giai đoạn đầu. | Cũng cần lập kế hoạch chi tiết từ giai đoạn đầu. |
| Rủi ro | Mức độ rủi ro cao. | Mức độ rủi ro thấp. |
| Thời gian chờ để có phần mềm chạy được | Thời gian chờ lâu. | Thời gian chờ ngắn. |
| Xử lý dự án lớn | Không phù hợp cho các dự án lớn. | Cũng không phù hợp cho các dự án lớn. |
| Khả năng thay đổi | Khó thay đổi. | Dễ thay đổi. |
| Chi phí | Chi phí thấp. | Chi phí cũng thấp. |
| Kiểm thử | Kiểm thử chỉ diễn ra sau khi hoàn thành giai đoạn mã hóa. | Kiểm thử diễn ra sau mỗi vòng lặp. |
| Quay lại giai đoạn trước | Không thể quay lại giai đoạn trước. | Có thể quay lại giai đoạn trước. |
| Quy mô nhóm phát triển | Cần đội ngũ lớn. | Không cần đội ngũ lớn. |
| Sự chồng chéo giữa các giai đoạn | Không có sự chồng chéo giữa các giai đoạn. | Có sự chồng chéo giữa các giai đoạn. |
| Chu kỳ phát triển | Chỉ có một chu kỳ phát triển. | Có nhiều chu kỳ phát triển. |
| Sự tham gia của khách hàng | Khách hàng chỉ tham gia vào giai đoạn đầu của quá trình phát triển. | Khách hàng tham gia vào các giai đoạn trung gian. |
| Loại khung phát triển | Sử dụng khung phát triển tuyến tính. | Sử dụng khung phát triển tuyến tính kết hợp lặp. |
| Kiểm soát của khách hàng | Khách hàng có ít quyền kiểm soát đối với quản trị viên. | Khách hàng có nhiều quyền kiểm soát hơn so với mô hình Thác nước. |
| Tính tái sử dụng | Khả năng tái sử dụng thấp nhất. | Có thể tái sử dụng ở một mức độ nhất định. |

## Câu 2.4: Mục tiêu của pha bảo trì là gì?

Đảm bào phần mềm tiếp tục hoạt động ổn định và đáp ứng các yêu cầu mới.

## Câu 2.5:Mô hình xây và sửa có nhược điểm gì?

Mô hình **Xây và Sửa** (*Build and Fix Model*) là một cách tiếp cận đơn giản trong phát triển phần mềm, trong đó phần mềm được viết ra, thử nghiệm, sửa lỗi và tiếp tục cải tiến mà không có một quy trình rõ ràng.

* **Nhược điểm của mô hình này:**

1. **Khó kiểm soát chất lượng**
   * Do không có quy trình phát triển rõ ràng nên dễ xuất hiện nhiều lỗi.
   * Thiếu kiểm thử hệ thống dẫn đến sản phẩm không ổn định.
2. **Tốn nhiều chi phí bảo trì**
   * Việc sửa chữa liên tục khiến chi phí phát triển và bảo trì tăng cao.
   * Không có tài liệu thiết kế, gây khó khăn khi bảo trì và mở rộng.
3. **Không phù hợp với dự án lớn**
   * Chỉ phù hợp với các phần mềm nhỏ, nguyên mẫu (*prototype*).
   * Đối với phần mềm phức tạp, việc quản lý tiến độ và yêu cầu sẽ rất khó khăn.
4. **Thiếu tài liệu và quy trình**
   * Không có quy trình lập kế hoạch, phân tích và kiểm thử bài bản.
   * Khi có sự thay đổi nhân sự, rất khó để tiếp tục phát triển phần mềm.

## Câu 2.6:Mô hình bản mẫu nhanh là gì?

**Mô hình Bản mẫu nhanh** (*Rapid Prototyping Model*) là một phương pháp phát triển phần mềm tập trung vào việc xây dựng nhanh chóng một **bản mẫu (prototype)** để khách hàng đánh giá và phản hồi, sau đó tiếp tục cải tiến dựa trên phản hồi đó.

* **Quy trình của mô hình bản mẫu nhanh:**

1. **Xác định yêu cầu ban đầu** → Thu thập các yêu cầu chính từ khách hàng.
2. **Xây dựng bản mẫu** → Tạo một phiên bản đơn giản của phần mềm để khách hàng dùng thử.
3. **Đánh giá và phản hồi** → Khách hàng sử dụng bản mẫu và đưa ra nhận xét.
4. **Cải tiến và hoàn thiện** → Điều chỉnh và phát triển tiếp dựa trên phản hồi của khách hàng.
5. **Triển khai sản phẩm cuối cùng** → Khi bản mẫu đạt yêu cầu, phần mềm chính thức được phát triển và triển khai.

* **Ưu điểm của mô hình Bản mẫu nhanh:**-Giúp khách hàng hiểu rõ sản phẩm trước khi phát triển chính thức.  
  -Giảm rủi ro về yêu cầu không rõ ràng hoặc thay đổi sau này.  
  -Cải thiện sự hợp tác giữa khách hàng và đội ngũ phát triển.
* **Nhược điểm:** -Tốn thời gian nếu khách hàng liên tục thay đổi yêu cầu.  
   -Dễ dẫn đến tình trạng mở rộng phạm vi (*scope creep*), làm chậm tiến độ dự án.  
  - Không phù hợp với các hệ thống yêu cầu độ ổn định cao ngay từ đầu.
* **Ứng dụng:** Mô hình này thường được sử dụng trong các dự án phần mềm có giao diện phức tạp hoặc khi yêu cầu chưa rõ ràng.

## Câu 2.7:Pha giải thể là gì?

Pha giải thể (Disposal Phase) là gì?

**Pha giải thể** là giai đoạn cuối cùng trong vòng đời phát triển phần mềm (SDLC - Software Development Life Cycle). Ở giai đoạn này, phần mềm không còn được sử dụng và cần được **ngừng hoạt động, loại bỏ hoặc thay thế**.

Mục tiêu của pha giải thể

* **Dừng hoạt động phần mềm** một cách an toàn, tránh gây gián đoạn cho hệ thống và người dùng.
* **Bảo vệ và xử lý dữ liệu** trước khi hệ thống bị gỡ bỏ (sao lưu hoặc di chuyển dữ liệu sang hệ thống mới).
* **Loại bỏ phần cứng, tài nguyên liên quan** nếu cần thiết.
* **Đánh giá tổng kết** để rút kinh nghiệm cho các dự án tương lai.

Các hoạt động chính trong pha giải thể

1. **Xác định lý do ngừng sử dụng phần mềm**
   * Phần mềm lỗi thời, không còn đáp ứng nhu cầu.
   * Thay thế bằng hệ thống mới.
   * Không còn được hỗ trợ hoặc bảo trì.
2. **Chuyển đổi dữ liệu và hệ thống**
   * Xuất và lưu trữ dữ liệu quan trọng.
   * Di chuyển dữ liệu sang phần mềm mới (nếu có).
3. **Gỡ bỏ phần mềm**
   * Gỡ cài đặt ứng dụng và các dịch vụ liên quan.
   * Vô hiệu hóa quyền truy cập.
4. **Đánh giá hậu giải thể**
   * Phân tích xem hệ thống cũ có ảnh hưởng gì đến hoạt động hiện tại.
   * Đưa ra các bài học kinh nghiệm cho dự án sau.

## Câu 2.8:Mô hình xoắn ốc là gì?

Mô hình xoắn ốc (Spiral Model) là gì?

**Mô hình xoắn ốc** là một mô hình phát triển phần mềm kết hợp giữa **mô hình thác nước (Waterfall Model)** và **mô hình lặp (Iterative Model)**. Nó được đề xuất bởi **Barry Boehm** vào năm 1986 và tập trung vào **quản lý rủi ro** trong quá trình phát triển phần mềm.

Đặc điểm chính của mô hình xoắn ốc

1. **Phát triển theo chu kỳ (vòng lặp)**
   * Quá trình phát triển được chia thành nhiều vòng lặp (spiral iterations).
   * Mỗi vòng lặp bao gồm các bước từ phân tích, thiết kế, phát triển đến đánh giá.
2. **Tập trung vào quản lý rủi ro**
   * Mô hình giúp xác định và giảm thiểu rủi ro ngay từ đầu, đảm bảo sản phẩm cuối cùng ổn định và đáp ứng yêu cầu.
3. **Linh hoạt trong thay đổi yêu cầu**
   * Vì mỗi vòng xoắn có thể điều chỉnh lại yêu cầu, mô hình này phù hợp với các dự án lớn, phức tạp.

Các pha trong mô hình xoắn ốc

Mô hình xoắn ốc được chia thành 4 pha chính trong mỗi vòng lặp:

1. **Xác định yêu cầu và lập kế hoạch**
   * Xác định mục tiêu, yêu cầu của sản phẩm trong giai đoạn hiện tại.
   * Đưa ra phương án phát triển và tài nguyên cần thiết.
2. **Phân tích rủi ro và đánh giá**
   * Nhận diện các rủi ro có thể xảy ra trong dự án.
   * Lập kế hoạch giảm thiểu rủi ro hoặc thay đổi chiến lược phát triển nếu cần.
3. **Thiết kế, phát triển và kiểm thử**
   * Tiến hành thiết kế kiến trúc, lập trình và kiểm thử phần mềm.
   * Đưa ra phiên bản phần mềm có thể chạy được.
4. **Đánh giá và lập kế hoạch cho vòng xoắn tiếp theo**
   * Xác định các vấn đề còn tồn đọng.
   * Tiếp tục điều chỉnh và cải tiến cho vòng xoắn tiếp theo.

## Câu 2.9: Tại sao mô hình tiến trình linh hoạt được đánh giá cao?

Mô hình tiến trình linh hoạt (AGILE MODEL - PAIR PROGRAMMING) được đánh giá cao vì:

* Thích nghi tốt với thay đổi – Mô hình tiến trình linh hoạt cho phép điều chỉnh linh hoạt theo phản hồi khách hàng, giúp sản phẩm phù hợp với nhu cầu thực tế.
* Nâng cao chất lượng phần mềm – Kiểm thử liên tục, cải tiến từng giai đoạn giúp phát hiện lỗi sớm và tối ưu sản phẩm.

## Câu 2.10: Điểm khác biệt chính giữa mô hình mã nguồn mở và các mô hình khác là gì?

* Trong mô hình mã nguồn mở, Mã nguồn được công khai, cho phép bất kỳ ai cũng có thể xem, sửa đổi và phân phối lại.. Điều này thúc đẩy sự cộng tác, minh bạch và khả năng tùy chỉnh từ cộng đồng.
* Ngược lại, các mô hình khác, mã nguồn sẽ bí mật, chỉ nhà phát triển hoặc công ty sở hữu mới có quyền truy cập và chỉnh sửa. Người dùng cuối thường chỉ nhận được sản phẩm hoàn thiện mà không có khả năng can thiệp vào mã nguồn.

# PHẦN 3: CÂU HỎI THẢO LUẬN NHÓM

## Câu 3.1: So sánh ưu và nhược điểm của mô hình thác nước và mô hình xoắn ốc.

So sánh Mô hình Thác nước và Mô hình Xoắn ốc

Mô hình Thác nước (Waterfall Model)

Ưu điểm:

* Cấu trúc rõ ràng và dễ hiểu
* Phù hợp với các dự án có yêu cầu ổn định
* Quản lý dễ dàng do các giai đoạn được xác định rõ ràng
* Phù hợp với các dự án nhỏ và có phạm vi xác định chính xác
* Tài liệu được lập chi tiết ở mỗi giai đoạn

Nhược điểm:

* Thiếu tính linh hoạt
* Khó điều chỉnh khi yêu cầu thay đổi
* Kiểm thử diễn ra muộn trong chu trình
* Rủi ro cao nếu có sai sót ở các giai đoạn đầu
* Không phù hợp với các dự án phức tạp hoặc động

Mô hình Xoắn ốc (Spiral Model)

Ưu điểm:

* Linh hoạt và thích ứng cao
* Quản lý rủi ro tốt thông qua phân tích liên tục
* Cho phép thay đổi yêu cầu giữa các chu kỳ
* Kiểm thử sớm và liên tục
* Phù hợp với các dự án lớn và phức tạp
* Tích hợp quản lý rủi ro vào quy trình phát triển

Nhược điểm:

* Phức tạp và khó quản lý
* Chi phí cao do các chu kỳ lặp liên tục
* Yêu cầu chuyên gia có kỹ năng cao
* Khó xác định thời gian và chi phí dự án
* Phụ thuộc nhiều vào chuyên môn của nhóm phát triển

## Câu 3.2: Thảo luận về tình huống thực tế có thể áp dụng mô hình lặp và tăng trưởng.

Ứng dụng Mô hình Lặp và Tăng trưởng trong Thực tế

Tình huống 1: Phát triển Ứng dụng Di động

**Ví dụ:** Ứng dụng ngân hàng trực tuyến

Đặc điểm:

* Yêu cầu thay đổi liên tục
* Cần cập nhật tính năng thường xuyên
* An ninh và trải nghiệm người dùng quan trọng

Áp dụng Mô hình Lặp và Tăng trưởng:

* Chu kỳ phát triển ngắn (2-4 tuần)
* Phát triển từng tính năng:
  1. Đăng nhập
  2. Tra cứu số dư
  3. Chuyển khoản
  4. Quản lý thẻ
* Kiểm thử và phản hồi liên tục
* Cải tiến dần theo góp ý người dùng

Tình huống 2: Hệ thống Quản lý Doanh nghiệp

**Ví dụ:** Phần mềm ERP cho doanh nghiệp sản xuất

Đặc điểm:

* Phức tạp với nhiều module
* Yêu cầu tích hợp nhiều chức năng
* Cần linh hoạt với quy trình kinh doanh

Áp dụng Mô hình Lặp và Tăng trưởng:

* Phát triển từng module độc lập:
  1. Quản lý nhân sự
  2. Kế toán
  3. Quản lý kho
  4. Báo cáo kinh doanh
* Mỗi module được phát triển, kiểm thử và tích hợp riêng
* Điều chỉnh liên tục dựa trên phản hồi người dùng

Lợi ích Chính

* Giảm thiểu rủi ro
* Tăng tính linh hoạt
* Đáp ứng nhanh yêu cầu thay đổi
* Cải thiện chất lượng sản phẩm
* Tăng sự hài lòng của khách hàng

## Câu 3.3: Tại sao mô hình xây và sửa không phù hợp với các dự án lớn?

Mô hình xây và sửa không có quy trình phát triển rõ ràng, dẫn đến khó kiểm soát khi dự án mở rộng. Việc thiếu kế hoạch chi tiết khiến mã nguồn dễ trở nên lộn xộn, khó bảo trì và dễ phát sinh lỗi khi bổ sung tính năng mới.

## Câu 3.4: So sánh giữa mô hình bản mẫu nhanh và mô hình tiến trình linh hoạt.

|  |  |  |
| --- | --- | --- |
|  | **Mô hình Bản mẫu nhanh (Rapid Prototype Model)** | **Mô hình Tiến trình linh hoạt (Agile)** |
| Mục tiêu | Tạo ra một bản mẫu nhanh để khách hàng đánh giá trước khi phát triển sản phẩm hoàn chỉnh. | Phát triển phần mềm theo từng giai đoạn nhỏ, phản hồi nhanh và điều chỉnh liên tục. |
| Cách tiếp cận | Xây dựng một bản mẫu (prototype), nhận phản hồi từ khách hàng, sau đó chỉnh sửa dần cho đến khi đạt yêu cầu. | Chia dự án thành các vòng lặp (iteration), mỗi vòng lặp phát triển một phần hoàn chỉnh của phần mềm. |
| Tốc độ phát triển | Nhanh chóng tạo bản mẫu, nhưng có thể mất thêm thời gian để hoàn thiện sản phẩm cuối cùng. | Phát triển theo từng bước nhỏ nhưng có thể cung cấp sản phẩm sớm hơn. |
| Khả năng thay đổi yêu cầu | Có thể thay đổi dễ dàng ở giai đoạn đầu khi phát triển bản mẫu, nhưng nếu thay đổi nhiều sau khi phát triển chính thức thì có thể gây khó khăn. | Linh hoạt với thay đổi ở mọi giai đoạn phát triển. |
| Kiểm thử & Đánh giá | Bản mẫu được kiểm tra nhiều lần để điều chỉnh theo yêu cầu khách hàng trước khi phát triển chính thức. | Mỗi vòng lặp đều có kiểm thử và đánh giá để đảm bảo chất lượng liên tục. |
| Chi phí | Chi phí ban đầu thấp nhưng có thể tăng nếu bản mẫu phải chỉnh sửa nhiều lần. | Kiểm soát chi phí tốt hơn do phát triển theo từng phần nhỏ. |
| Tính phù hợp | Phù hợp với các dự án cần xác định rõ yêu cầu trước khi bắt tay vào phát triển. | Thích hợp cho các dự án có yêu cầu thay đổi thường xuyên. |

## Câu 3.5:Phân tích vai trò của quản lý rủi ro trong mô hình xoắn ốc.

**Mô hình xoắn ốc (Spiral Model)** là một phương pháp phát triển phần mềm tập trung vào **quản lý rủi ro**, giúp giảm thiểu những vấn đề tiềm ẩn trong quá trình phát triển.

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

* **Nhận diện rủi ro sớm**
* Trong mỗi vòng lặp, các rủi ro tiềm ẩn (kỹ thuật, tài chính, yêu cầu khách hàng…) đều được phân tích.
* Giúp dự đoán trước các vấn đề có thể xảy ra.
* **Đưa ra biện pháp kiểm soát rủi ro**
* Nếu một rủi ro được phát hiện, nhóm phát triển sẽ có thời gian để **đề xuất phương án giảm thiểu**.
* Ví dụ: Nếu công nghệ mới chưa được kiểm chứng, có thể tạo **nguyên mẫu (prototype)** để kiểm tra trước khi triển khai chính thức.
* **Lặp lại để giảm thiểu rủi ro dần dần**
* Mỗi vòng xoắn trong mô hình giúp phát hiện và giảm bớt rủi ro trước khi chuyển sang giai đoạn tiếp theo.
* Điều này giúp tránh các vấn đề lớn khi dự án đã đi quá xa.
* **Tối ưu hóa chi phí và chất lượng phần mềm**
* Nhờ việc quản lý rủi ro tốt, dự án không bị gián đoạn do các vấn đề lớn, giảm thiểu việc sửa lỗi tốn kém ở giai đoạn sau.

## Câu 3.6:Khi nào nên sử dụng mô hình thác nước thay vì mô hình tiến trình linh hoạt?

* **Yêu cầu rõ ràng, ít thay đổi** → Tránh phải điều chỉnh lại sau này.
* **Dự án ngắn hạn** → Quản lý dễ dàng từ đầu đến cuối.
* **Ngân sách & thời gian cố định** → Giúp kiểm soát tốt chi phí & tiến độ.
* **Đội ngũ ít kinh nghiệm với Agile** → Dễ triển khai hơn với các nhóm mới.
* **Dự án cần tuân thủ chặt chẽ** → Phù hợp với phần mềm **y tế, quân sự, tài chính**, nơi cần tài liệu chi tiết và kiểm tra nghiêm ngặt.

## Câu 3.7:Thảo luận về những khó khăn khi áp dụng mô hình mã nguồn mở.

Áp dụng mô hình mã nguồn mở mang lại nhiều lợi ích như chi phí thấp, khả năng tùy chỉnh linh hoạt, và cộng đồng hỗ trợ mạnh. Nhưng nó cũng đi kèm với không ít khó khăn. Dưới đây là một số thách thức thường gặp:

1. **Thiếu hỗ trợ chính thức** – Không có đội ngũ hỗ trợ chuyên trách như phần mềm thương mại. Nếu gặp lỗi nghiêm trọng, bạn phải tự tìm cách sửa hoặc dựa vào cộng đồng.
2. **Rủi ro về bảo mật** – Mặc dù mã nguồn mở có lợi thế là nhiều người kiểm tra, nhưng cũng có thể bị khai thác nếu không cập nhật kịp thời. Bạn cần có kế hoạch bảo trì và kiểm tra bảo mật thường xuyên.
3. **Khả năng tương thích và tích hợp** – Một số phần mềm mã nguồn mở không hoạt động tốt với hệ thống hiện tại hoặc các công nghệ thương mại. Việc tùy chỉnh để tích hợp có thể tốn công sức.
4. **Khó kiểm soát chất lượng** – Mã nguồn mở có thể có nhiều người đóng góp với các tiêu chuẩn khác nhau. Một số dự án không có quy trình kiểm thử chặt chẽ, dẫn đến lỗi hoặc mã kém tối ưu.
5. **Vấn đề bản quyền và giấy phép** – Có nhiều loại giấy phép mã nguồn mở khác nhau (GPL, MIT, Apache…), nếu không nắm rõ, bạn có thể vi phạm điều khoản sử dụng mà không biết.
6. **Thiếu tài liệu đầy đủ** – Không phải dự án nào cũng có tài liệu chi tiết. Nếu tài liệu kém hoặc không cập nhật, việc triển khai và bảo trì sẽ khó khăn

## Câu 3.8:Phân tích cách mô hình tiến trình linh hoạt giúp cải thiện chất lượng phần mềm.

Mô hình tiến trình linh hoạt (Agile) giúp nâng cao chất lượng phần mềm thông qua các đặc điểm sau:

* Phát triển lặp và cải tiến liên tục

Phần mềm được phát triển theo từng vòng lặp nhỏ (iteration).

Mỗi vòng lặp tạo ra một sản phẩm có thể hoạt động, giúp phát hiện lỗi sớm và điều chỉnh nhanh.

Việc kiểm thử liên tục trong mỗi vòng lặp giúp giảm lỗi và cải thiện chất lượng tổng thể.

* Hợp tác chặt chẽ giữa các bên liên quan

Khách hàng, lập trình viên, tester và các bên liên quan làm việc chặt chẽ để đảm bảo phần mềm đáp ứng đúng yêu cầu.

Phản hồi của khách hàng được thu thập và tích hợp ngay vào quá trình phát triển.

* Phản hồi nhanh và linh hoạt với thay đổi

Agile cho phép thay đổi yêu cầu ngay cả khi dự án đang phát triển.

Điều này giúp phần mềm luôn phù hợp với nhu cầu thực tế thay vì bị giới hạn bởi kế hoạch cố định.

* Kiểm thử liên tục và tự động hóa

Agile khuyến khích kiểm thử sớm và liên tục để phát hiện lỗi ngay từ đầu.

Các phương pháp như **Test-Driven Development (TDD)** và **Continuous Integration (CI)** giúp duy trì chất lượng phần mềm.

* Tăng cường tính trách nhiệm của nhóm phát triển

Agile yêu cầu nhóm làm việc tự quản lý và chủ động cải tiến quy trình.

Các buổi họp như **Daily Stand-up** giúp nhóm phát hiện vấn đề sớm và điều chỉnh kịp thời.

* Giảm rủi ro và tối ưu hiệu suất

Việc phát triển theo từng giai đoạn nhỏ giúp giảm thiểu rủi ro khi có vấn đề xảy ra.

Khách hàng luôn có phiên bản hoạt động được, hạn chế trường hợp sản phẩm cuối cùng không đáp ứng yêu cầu.

## 

## Câu 3.9:Thảo luận về vai trò của pha bảo trì trong vòng đời phát triển phần mềm.

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

Pha bảo trì (Maintenance Phase) là một trong những giai đoạn quan trọng trong vòng đời phát triển phần mềm (Software Development Life Cycle - SDLC). Sau khi phần mềm được triển khai và đưa vào sử dụng, pha bảo trì đảm bảo phần mềm hoạt động ổn định, đáp ứng nhu cầu thay đổi và cải thiện hiệu suất theo thời gian. Dưới đây là các vai trò chính của pha bảo trì:

**Sửa lỗi (Corrective Maintenance)**

* Phát hiện và khắc phục các lỗi (bugs) xuất hiện trong quá trình sử dụng.
* Xử lý các lỗi về logic, giao diện, hoặc hiệu suất mà chưa được phát hiện trong giai đoạn kiểm thử.

**Cải tiến và nâng cấp (Adaptive Maintenance)**

* Điều chỉnh phần mềm để phù hợp với sự thay đổi của môi trường hoạt động, chẳng hạn như cập nhật hệ điều hành, phần cứng hoặc công nghệ mới.
* Đáp ứng các yêu cầu mới từ người dùng, chẳng hạn như thay đổi quy trình kinh doanh hoặc mở rộng quy mô hệ thống.

**Tối ưu hóa hiệu suất (Perfective Maintenance)**

* Cải thiện hiệu suất phần mềm để chạy nhanh hơn, giảm mức tiêu thụ tài nguyên.
* Tinh chỉnh giao diện và trải nghiệm người dùng để tăng tính thân thiện.

**Phòng ngừa sự cố (Preventive Maintenance)**

* Dự đoán và sửa chữa các vấn đề tiềm ẩn trước khi chúng ảnh hưởng đến hệ thống.
* Cập nhật bảo mật để bảo vệ phần mềm khỏi các cuộc tấn công hoặc lỗ hổng bảo mật mới.

**Đảm bảo tuổi thọ và tính bền vững của phần mềm**

* Phần mềm cần được duy trì để đáp ứng nhu cầu lâu dài của người dùng và doanh nghiệp.
* Một phần mềm không được bảo trì có thể trở nên lỗi thời, không thể sử dụng được trong môi trường công nghệ thay đổi nhanh chóng.

**Tiết kiệm chi phí và nâng cao hiệu quả**

* Việc bảo trì định kỳ giúp giảm thiểu chi phí sửa chữa lỗi nghiêm trọng trong tương lai.
* Hạn chế thời gian hệ thống bị gián đoạn, từ đó nâng cao hiệu suất làm việc của doanh nghiệp.

## Câu 3.10:Đề xuất mô hình vòng đời phù hợp cho dự án phát triển phần mềm ngân hàng và giải thích lý do.

**Mô hình đề xuất: Mô hình V (V-Model) hoặc Mô hình Spiral (Mô hình xoắn ốc)**

Dự án phát triển phần mềm ngân hàng yêu cầu tính bảo mật cao, độ chính xác tuyệt đối và khả năng thích ứng với quy định pháp lý. Do đó, mô hình phát triển phần mềm phù hợp phải đảm bảo kiểm thử nghiêm ngặt, giảm thiểu rủi ro và có khả năng thích ứng với các thay đổi. Hai mô hình phù hợp nhất là mô hình V (V-Model) và mô hình Spiral (mô hình xoắn ốc).

**1. Mô hình V (V-Model)**

Mô tả:

* Mô hình V là một biến thể của mô hình thác nước (Waterfall) nhưng nhấn mạnh vào kiểm thử ở mỗi giai đoạn phát triển.
* Mỗi bước trong quá trình phát triển đều có một bước kiểm thử tương ứng.

Lý do chọn mô hình V cho phần mềm ngân hàng:

✅ Tính chính xác cao: Do ngân hàng yêu cầu độ chính xác tuyệt đối, mô hình V đảm bảo kiểm thử nghiêm ngặt ở từng giai đoạn.  
✅ Bảo mật tốt: Mỗi bước phát triển đều có kiểm thử tương ứng, giúp phát hiện lỗi sớm, giảm rủi ro bảo mật.  
✅ Độ tin cậy cao: Phần mềm ngân hàng không thể có lỗi nghiêm trọng khi triển khai, nên mô hình này giúp đảm bảo chất lượng ngay từ đầu.  
✅ Phù hợp với hệ thống có yêu cầu rõ ràng: Các yêu cầu của phần mềm ngân hàng thường được xác định ngay từ đầu, nên mô hình V phù hợp để triển khai.

Nhược điểm:

❌ Không linh hoạt nếu yêu cầu thay đổi sau khi bắt đầu phát triển.

📌 Khi nào nên chọn mô hình V?

* Khi yêu cầu phần mềm được xác định rõ ngay từ đầu.
* Khi cần đảm bảo chất lượng cao với quy trình kiểm thử nghiêm ngặt.

**2. Mô hình Spiral (Xoắn ốc)**

Mô tả:

* Mô hình Spiral kết hợp giữa mô hình thác nước và phát triển lặp.
* Phát triển phần mềm qua nhiều vòng lặp (iterations), mỗi vòng bao gồm lập kế hoạch, phân tích rủi ro, phát triển và kiểm thử.

Lý do chọn mô hình Spiral cho phần mềm ngân hàng:

✅ Quản lý rủi ro tốt: Mô hình này có bước phân tích rủi ro ở mỗi vòng lặp, rất quan trọng đối với phần mềm ngân hàng.  
✅ Linh hoạt với thay đổi: Nếu quy định pháp lý thay đổi hoặc có yêu cầu mới, mô hình Spiral dễ thích nghi hơn mô hình V.  
✅ Tích hợp kiểm thử liên tục: Kiểm thử và đánh giá rủi ro diễn ra liên tục, giúp giảm thiểu lỗi trước khi triển khai chính thức.  
✅ Phù hợp với các hệ thống phức tạp: Hệ thống ngân hàng thường lớn và phức tạp, mô hình Spiral giúp phát triển từng phần và kiểm tra từng giai đoạn.

Nhược điểm:

❌ Yêu cầu tài nguyên cao hơn do phải thực hiện nhiều vòng lặp.  
❌ Cần đội ngũ có kinh nghiệm để quản lý rủi ro hiệu quả.

📌 Khi nào nên chọn mô hình Spiral?

* Khi phần mềm ngân hàng có quy mô lớn và có thể thay đổi trong quá trình phát triển.
* Khi cần đánh giá rủi ro liên tục để đảm bảo an toàn cho hệ thống.

## 

# PHẦN 4: CÂU HỎI TÌNH HUỐNG

## Câu 4.1: 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?

Thay Đổi Yêu Cầu Trong Mô Hình Thác Nước

Diễn Giải Tình Huống

* Đã hoàn thành pha thiết kế
* Khách hàng yêu cầu thay đổi
* Mô hình thác nước khó điều chỉnh

Các Giải Pháp Khả Thi:

1. **Đánh Giá Mức Độ Thay Đổi**

* Phân tích mức độ ảnh hưởng của yêu cầu mới
* Xác định phạm vi thay đổi
* Ước tính chi phí và thời gian điều chỉnh

1. **Quản Lý Thay Đổi Chính Thức**

* Lập văn bản yêu cầu thay đổi (Change Request)
* Thảo luận chi tiết với khách hàng
* Xác định chi phí bổ sung
* Ký phụ lục hợp đồng

1. **Phương Án Kỹ Thuật**

* Nếu thay đổi nhỏ: Điều chỉnh trong giai đoạn hiện tại
* Nếu thay đổi lớn: Quay lại pha phân tích yêu cầu
* Cập nhật tài liệu thiết kế
* Đánh giá lại toàn bộ kiến trúc hệ thống

1. **Đàm Phán Với Khách Hàng**

* Giải thích chi phí và rủi ro của việc thay đổi
* Đề xuất các phương án linh hoạt
* Thống nhất phương án phù hợp nhất

Nguyên Tắc Quan Trọng

* Giao tiếp minh bạch
* Quản lý kỳ vọng của khách hàng
* Linh hoạt trong giới hạn cho phép

## Câu 4.2: 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?

Vấn Đề Tiến Độ Và Nhân Lực Trong Mô Hình Lặp Và Tăng Trưởng

Diễn Giải Tình Huống

* Dự án chậm tiến độ
* Thiếu nhân lực
* Áp dụng mô hình lặp và tăng trưởng

Các Giải Pháp Chiến Lược:

1. **Tối Ưu Hóa Nguồn Nhân Lực**

* Phân tích năng lực từng thành viên
* Tái phân bổ nguồn lực
* Ưu tiên nhiệm vụ then chốt

1. **Giải Pháp Nhân Sự**

* Tuyển dụng gấp nhân viên mới
* Thuê chuyên gia ngoài (Freelancer)
* Huy động nguồn lực từ các dự án khác

1. **Điều Chỉnh Phạm Vi Dự Án**

* Đàm phán với khách hàng để giảm phạm vi
* Chia nhỏ dự án thành các giai đoạn
* Hoàn thiện các tính năng quan trọng nhất

1. **Quản Lý SCRUM Linh Hoạt**

* Giảm độ dài sprint
* Tăng cường họp nhóm ngắn
* Theo dõi sát tiến độ
* Sử dụng công cụ quản lý dự án hiệu quả

1. **Công Nghệ Hỗ Trợ**

* Sử dụng công cụ tự động hóa
* Áp dụng DevOps
* Tăng năng suất phát triển

Nguyên Tắc Quản Lý

* Linh hoạt và thích ứng
* Ưu tiên giá trị cốt lõi
* Giao tiếp minh bạch

## Câu 4.3: 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?

Trong mô hình tiến trình linh hoạt (Agile), phản hồi kịp thời từ khách hàng rất quan trọng để đảm bảo phần mềm đáp ứng đúng yêu cầu. Nếu khách hàng không đưa ra phản hồi kịp thời, đội phát triển có thể xử lý bằng cách:

1. Chủ động liên lạc với khách hàng

* Sử dụng email, cuộc gọi hoặc họp trực tuyến để nhắc nhở và yêu cầu phản hồi.
* Xác định rõ khách hàng gặp khó khăn gì khi phản hồi (bận rộn, thiếu thông tin, chưa hiểu yêu cầu?).
* Đưa ra câu hỏi cụ thể thay vì chờ đợi phản hồi chung chung.

1. Định rõ trách nhiệm và cam kết phản hồi

* Thống nhất với khách hàng về thời gian phản hồi ngay từ đầu dự án.
* Nếu khách hàng chậm trễ, có thể đưa ra mốc thời gian cuối cùng để đảm bảo tiến độ dự án.
* Yêu cầu khách hàng chỉ định một người đại diện có thể phản hồi thay mặt họ.

1. Tiếp tục phát triển dựa trên giả định hợp lý

* Nếu không thể chờ đợi, đội phát triển có thể tạm thời đưa ra quyết định dựa trên dữ liệu trước đó và kinh nghiệm của nhóm.
* Tài liệu hóa các giả định này để có thể điều chỉnh sau khi nhận được phản hồi.

1. Giảm thiểu rủi ro bằng nguyên tắc phát triển linh hoạt

* Sử dụng nguyên tắc MVP (Minimum Viable Product): phát triển các tính năng cốt lõi trước để giảm tác động nếu cần thay đổi.
* Chia nhỏ công việc thành các sprint ngắn để có thể thích ứng với thay đổi nếu khách hàng phản hồi trễ.

1. Báo cáo tác động của việc chậm phản hồi

* Giải thích cho khách hàng rằng phản hồi chậm có thể gây ảnh hưởng đến tiến độ và chi phí dự án.
* Cung cấp báo cáo về các rủi ro có thể xảy ra nếu không có phản hồi kịp thời.

## Câu 4.4: 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?

Khi phần mềm đã ở pha cài đặt (deployment) mà khách hàng yêu cầu bổ sung tính năng mới, mô hình Agile (cụ thể là Scrum hoặc Kanban) là lựa chọn phù hợp nhất để xử lý tình huống này. Lý do:

* Linh hoạt: Agile cho phép đội phát triển tiếp nhận yêu cầu mới bất cứ lúc nào, thêm chúng vào Product Backlog và ưu tiên xử lý trong các Sprint tiếp theo.
* Phản hồi nhanh: Quy trình lặp (iterative) của Agile giúp triển khai tính năng mới mà không làm gián đoạn hệ thống đã cài đặt.
* Minh bạch: Khách hàng có thể theo dõi tiến độ qua các buổi demo (Sprint Review).

## Câu 4.5: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.

* **Tận dụng công cụ và phần mềm mã nguồn mở:** Sử dụng các công cụ và phần mềm mã nguồn mở để giảm chi phí phát triển. Các công cụ này có thể hỗ trợ trong việc tạo mẫu thử nhanh chóng mà không cần đầu tư quá nhiều vào phần mềm thương mại đắt tiền.
* **Phát triển mẫu thử nhỏ gọn:** Hạn chế phạm vi của mẫu thử ban đầu để chỉ tập trung vào những tính năng cốt lõi và quan trọng nhất. Điều này giúp tiết kiệm thời gian và nguồn lực, đồng thời cho phép nhận phản hồi nhanh chóng từ người dùng.
* **Thuê ngoài và hợp tác:** Hợp tác với các đối tác bên ngoài hoặc thuê ngoài các dịch vụ phát triển mẫu thử. Điều này giúp công ty tận dụng được nguồn lực từ bên ngoài mà không cần phải tuyển dụng thêm nhân viên cố định.
* **Sử dụng các framework và template:** Sử dụng các framework và template có sẵn để tiết kiệm thời gian phát triển mẫu thử. Các công cụ này giúp xây dựng mẫu thử nhanh chóng và dễ dàng hơn.
* **Phân bổ nguồn lực hợp lý:** Xác định và phân bổ nguồn lực một cách hợp lý, tập trung vào những giai đoạn quan trọng nhất của quá trình phát triển mẫu thử. Điều này giúp tối ưu hóa sử dụng nguồn lực sẵn có.
* **Lập kế hoạch chi tiết:** Lập kế hoạch chi tiết và theo dõi tiến độ công việc để đảm bảo rằng việc phát triển mẫu thử được thực hiện đúng tiến độ và không gặp trở ngại do thiếu nguồn lực.
* **Tìm nguồn tài trợ và hỗ trợ:** Tìm kiếm các nguồn tài trợ hoặc hỗ trợ từ các tổ chức, quỹ đầu tư hoặc chương trình hỗ trợ doanh nghiệp nhỏ để có thêm nguồn lực phát triển mẫu thử.

## Câu 4.6: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

Trong dự án phần mềm thương mại điện tử, khi khách hàng liên tục yêu cầu thay đổi giao diện, mô hình phát triển phần mềm **Agile** sẽ phù hợp nhất. Cụ thể, các phương pháp Agile như **Scrum** hoặc **Kanban** sẽ giúp quản lý các yêu cầu thay đổi một cách linh hoạt và hiệu quả.

**Lý do Agile phù hợp:**

1. **Linh hoạt với thay đổi**: Agile được thiết kế để đáp ứng các yêu cầu thay đổi thường xuyên từ phía khách hàng. Các yêu cầu thay đổi giao diện có thể được tích hợp vào các sprint ngắn (thường là 1-2 tuần) mà không làm gián đoạn tiến độ dự án.
2. **Phản hồi nhanh chóng**: Với các sprint ngắn, khách hàng có thể xem và phản hồi về các thay đổi giao diện nhanh chóng, giúp đảm bảo sản phẩm cuối cùng đáp ứng đúng nhu cầu của họ.
3. **Tập trung vào giá trị người dùng**: Agile tập trung vào việc cung cấp giá trị cho người dùng cuối thông qua các bản cập nhật nhỏ và thường xuyên. Điều này đặc biệt quan trọng trong thương mại điện tử, nơi trải nghiệm người dùng (UX/UI) là yếu tố then chốt.
4. **Giao tiếp liên tục**: Agile khuyến khích giao tiếp thường xuyên giữa nhóm phát triển và khách hàng, giúp đảm bảo rằng mọi thay đổi đều được hiểu rõ và thực hiện đúng cách.

**Các bước triển khai Agile trong trường hợp này:**

1. **Chia nhỏ yêu cầu**: Phân tích các yêu cầu thay đổi giao diện thành các task nhỏ và ưu tiên chúng dựa trên giá trị và độ khẩn cấp.
2. **Lập kế hoạch sprint**: Đưa các task vào các sprint ngắn, thường là 1-2 tuần.
3. **Demo và phản hồi**: Sau mỗi sprint, demo sản phẩm cho khách hàng và nhận phản hồi để điều chỉnh trong các sprint tiếp theo.
4. **Tích hợp liên tục**: Đảm bảo rằng các thay đổi giao diện được tích hợp liên tục vào sản phẩm chính để tránh tích tụ công việc và xung đột.

**Lưu ý:**

* **Quản lý kỳ vọng**: Cần làm rõ với khách hàng rằng mặc dù Agile linh hoạt, nhưng không phải mọi thay đổi đều có thể được thực hiện ngay lập tức mà cần có sự ưu tiên và lập kế hoạch.
* **Tài liệu hóa**: Dù Agile ưu tiên phần mềm hơn tài liệu, nhưng vẫn cần ghi chép lại các yêu cầu thay đổi để tránh hiểu lầm và đảm bảo tính minh bạch.

## Câu 4.7: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 cho dự án phần mềm lớn với nhiều nhóm phát triển ở các quốc gia khác nhau

Đối với một dự án **phát triển phần mềm lớn, có nhiều nhóm phát triển ở các quốc gia khác nhau**, mô hình phù hợp nhất là:

💡 Mô hình tiến trình linh hoạt mở rộng (Scaled Agile Framework - SAFe) hoặc Mô hình phát triển toàn cầu (Global Software Development - GSD).

1. Scaled Agile Framework (SAFe)

**Lý do chọn SAFe:**

* **Hỗ trợ làm việc nhóm từ xa:** SAFe giúp quản lý và phối hợp nhiều nhóm phát triển tại các quốc gia khác nhau.
* **Tính linh hoạt cao:** Dễ thích ứng với thay đổi yêu cầu của khách hàng và thị trường.
* **Cấu trúc quản lý rõ ràng:** Chia dự án thành các nhóm nhỏ nhưng vẫn có sự kết nối, giúp tăng hiệu suất làm việc.
* **Làm việc theo Sprint:** Các nhóm có thể phát triển song song theo các chu kỳ ngắn, đảm bảo tiến độ.

2. Mô hình phát triển toàn cầu (Global Software Development - GSD)

**Lý do chọn GSD:**

* **Phù hợp với các dự án đa quốc gia:** Quản lý đội ngũ phát triển phần mềm trên phạm vi toàn cầu.
* **Giảm chi phí và tận dụng nguồn nhân lực toàn cầu:** Có thể phân bổ công việc theo múi giờ, tăng hiệu suất làm việc 24/7.
* **Cần công cụ quản lý mạnh mẽ:** Đòi hỏi các công cụ như Jira, Trello, Slack, GitHub để quản lý và giao tiếp hiệu quả.
* **Thách thức lớn về giao tiếp:** Cần có kế hoạch phối hợp chặt chẽ giữa các nhóm để tránh xung đột thời gian và khác biệt văn hóa.

Tại sao không nên dùng các mô hình khác?

* **Mô hình Thác nước (Waterfall):** Không phù hợp vì cứng nhắc, khó thích ứng với thay đổi.
* **Mô hình Xoắn ốc (Spiral):** Mặc dù quản lý rủi ro tốt, nhưng không tối ưu cho làm việc nhóm từ xa vì chi phí cao và mất nhiều thời gian.
* **Mô hình Bản mẫu nhanh (Rapid Prototyping):** Thích hợp cho các dự án nhỏ, không phù hợp với quy mô lớn.

## Câu 4.8: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.

Cách giảm thiểu rủi ro khi áp dụng mô hình xoắn ốc

Mô hình xoắn ốc tập trung mạnh vào phân tích và quản lý rủi ro, do đó, để tối ưu hóa hiệu quả của mô hình này, ta cần thực hiện các biện pháp giảm thiểu rủi ro ngay từ đầu. Dưới đây là một số cách tiếp cận quan trọng:

1. Xác định và phân loại rủi ro sớm

* Phân tích rủi ro ngay từ đầu mỗi vòng xoắn để xác định các nguy cơ tiềm ẩn.
* Phân loại rủi ro thành các nhóm như:
  + Rủi ro kỹ thuật (công nghệ không phù hợp, lỗi phần mềm).
  + Rủi ro tài chính (chi phí vượt ngân sách).
  + Rủi ro quản lý (thiếu nhân sự, tiến độ chậm).
  + Rủi ro về yêu cầu (khách hàng thay đổi yêu cầu liên tục).

2. Phát triển nguyên mẫu (Prototype) để giảm rủi ro kỹ thuật

* Xây dựng nguyên mẫu sớm để kiểm tra tính khả thi và hiệu suất của phần mềm.
* Nguyên mẫu giúp phát hiện sớm các vấn đề về giao diện, tính năng và hiệu năng.
* Giúp khách hàng đánh giá sản phẩm trước khi phát triển toàn bộ.

3. Kiểm thử liên tục và đánh giá định kỳ

* Thực hiện kiểm thử phần mềm ngay từ các giai đoạn đầu để phát hiện lỗi sớm.
* Áp dụng Test-Driven Development (TDD) hoặc Continuous Integration (CI) để kiểm soát chất lượng.
* Đánh giá lại rủi ro sau mỗi vòng xoắn để điều chỉnh kế hoạch kịp thời.

4. Quản lý thay đổi yêu cầu hiệu quả

* Áp dụng quy trình kiểm soát thay đổi để tránh việc yêu cầu thay đổi quá nhiều làm chậm tiến độ.
* Sử dụng các công cụ như JIRA, Trello để theo dõi và quản lý thay đổi.
* Ưu tiên các yêu cầu quan trọng nhất trong mỗi vòng xoắn để tối ưu hóa thời gian phát triển.

5. Quản lý tài nguyên và chi phí chặt chẽ

* Dự báo chi phí cho từng vòng xoắn để tránh vượt ngân sách.
* Theo dõi tiến độ thường xuyên để đảm bảo dự án không bị kéo dài ngoài dự kiến.
* Định kỳ đánh giá lại kế hoạch tài chính để điều chỉnh khi cần thiết.

6. Cải thiện giao tiếp và phối hợp nhóm

* Thiết lập các cuộc họp định kỳ giữa các nhóm phát triển để cập nhật tiến độ và giải quyết vấn đề.
* Đảm bảo giao tiếp hiệu quả giữa khách hàng, nhóm phát triển và quản lý dự án.
* Sử dụng công cụ hỗ trợ như Slack, Microsoft Teams để tăng cường tương tác.

## 

## Câu 4.9: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 cho dự án phát triển phần mềm ngân hàng với yêu cầu bảo mật cao**

Đối với một dự án phần mềm ngân hàng, bảo mật là yếu tố quan trọng hàng đầu vì hệ thống xử lý các giao dịch tài chính và thông tin nhạy cảm của khách hàng. Do đó, cần một mô hình phát triển phần mềm đảm bảo kiểm thử nghiêm ngặt, quản lý rủi ro chặt chẽ và khả năng thích ứng với các mối đe dọa bảo mật mới.

**Mô hình đề xuất: Mô hình Spiral (Mô hình xoắn ốc)**

Lý do chọn mô hình Spiral:

✅ Quản lý rủi ro bảo mật liên tục

* Trong mỗi vòng lặp của mô hình Spiral, có một bước phân tích rủi ro, giúp đánh giá và giảm thiểu các mối đe dọa bảo mật ngay từ giai đoạn đầu.
* Điều này đặc biệt quan trọng đối với phần mềm ngân hàng, nơi các cuộc tấn công mạng và lỗ hổng bảo mật có thể gây thiệt hại nghiêm trọng.

✅ Tích hợp kiểm thử bảo mật ở nhiều giai đoạn

* Kiểm thử bảo mật không chỉ được thực hiện ở cuối quá trình phát triển mà xuất hiện trong từng vòng lặp, giúp phát hiện và xử lý sớm các vấn đề bảo mật.

✅ Dễ thích ứng với các quy định bảo mật mới

* Ngành ngân hàng phải tuân thủ nhiều quy định pháp lý như PCI-DSS, GDPR, ISO 27001.
* Mô hình Spiral linh hoạt trong việc thay đổi thiết kế hệ thống để đáp ứng các yêu cầu pháp lý mới.

✅ Phù hợp với hệ thống ngân hàng phức tạp

* Phần mềm ngân hàng thường lớn, có nhiều tính năng và tích hợp với nhiều hệ thống khác (ví dụ: thanh toán quốc tế, xác thực đa yếu tố).
* Mô hình Spiral giúp phát triển từng phần và đảm bảo từng phần đạt tiêu chuẩn bảo mật trước khi tiếp tục mở rộng.

✅ Cải tiến bảo mật liên tục

* Vì mô hình Spiral có tính lặp lại, nó giúp cập nhật các biện pháp bảo mật thường xuyên để chống lại các mối đe dọa bảo mật mới.

## 

## Câu 4.10:Khi nào nên kết thúc vòng đời phần mềm và thực hiện pha giải thể?

Pha giải thể (Retirement/Disposal Phase) là giai đoạn cuối cùng trong vòng đời phát triển phần mềm (Software Development Life Cycle - SDLC). Nó diễn ra khi phần mềm không còn đáp ứng nhu cầu hoặc trở nên lỗi thời, và cần được thay thế hoặc ngừng sử dụng. Việc quyết định khi nào kết thúc vòng đời phần mềm và thực hiện pha giải thể phụ thuộc vào nhiều yếu tố, bao gồm kỹ thuật, kinh doanh và bảo mật.

**1. Phần mềm không còn đáp ứng yêu cầu kinh doanh**

* Doanh nghiệp thay đổi quy trình làm việc hoặc chiến lược kinh doanh, khiến phần mềm trở nên không phù hợp.
* Công nghệ mới ra đời cung cấp các giải pháp hiệu quả hơn, giảm chi phí vận hành.

**2. Hệ thống đã lỗi thời về công nghệ**

* Phần mềm được xây dựng trên nền tảng cũ không còn được hỗ trợ (ví dụ: hệ điều hành, framework, ngôn ngữ lập trình).
* Không thể tích hợp với các hệ thống hiện đại hoặc nâng cấp để đáp ứng các yêu cầu mới.

**3. Chi phí bảo trì quá cao**

* Khi chi phí bảo trì phần mềm vượt quá lợi ích mà nó mang lại.
* Khắc phục lỗi, cập nhật bảo mật hoặc tối ưu hiệu suất trở nên quá tốn kém so với việc phát triển một hệ thống mới.

**4. Rủi ro bảo mật tăng cao**

* Phần mềm chứa các lỗ hổng bảo mật nghiêm trọng không thể khắc phục do công nghệ cũ.
* Nguy cơ bị tấn công mạng hoặc vi phạm quy định bảo mật (như GDPR, ISO 27001) tăng cao.

**5. Người dùng giảm mạnh hoặc không còn sử dụng**

* Nếu phần mềm không còn người dùng do sự thay đổi của thị trường hoặc nhu cầu sử dụng.
* Các tổ chức hoặc khách hàng đã chuyển sang sử dụng sản phẩm khác tốt hơn.

**6. Hệ thống kế thừa đã sẵn sàng thay thế**

* Một phiên bản phần mềm mới đã được phát triển với nhiều cải tiến.
* Hệ thống mới có thể đảm bảo tính bảo mật, hiệu suất và tính khả dụng cao hơn.

**2. Các bước thực hiện pha giải thể phần mềm**

**🔹 Bước 1: Đánh giá tác động của việc giải thể**

* Xác định ảnh hưởng của việc ngừng sử dụng phần mềm đối với người dùng, dữ liệu và hệ thống liên quan.
* Lập kế hoạch chuyển đổi sang hệ thống mới (nếu có).

**🔹 Bước 2: Sao lưu và chuyển giao dữ liệu**

* Lưu trữ dữ liệu quan trọng trước khi ngừng hệ thống.
* Xuất dữ liệu sang hệ thống mới hoặc lưu trữ theo quy định pháp lý.

**🔹 Bước 3: Thông báo cho người dùng**

* Thông báo trước cho khách hàng, nhân viên và các bên liên quan về kế hoạch ngừng hoạt động.
* Cung cấp hướng dẫn chuyển đổi hoặc thay thế.

**🔹 Bước 4: Ngừng hệ thống một cách an toàn**

* Tắt phần mềm và hủy các quyền truy cập không cần thiết.
* Xóa dữ liệu nhạy cảm hoặc mã nguồn theo quy trình bảo mật.

**🔹 Bước 5: Đánh giá và tài liệu hóa quá trình giải thể**

* Lưu trữ thông tin về quá trình ngừng hoạt động để tham khảo trong tương lai.
* Đánh giá bài học kinh nghiệm từ quá trình phát triển và giải thể phần mềm.