1. Trình bày ngắn gọn (>2 trang) 4 pha trong framework phát triển yêu cầu phần mềm

Một framework phát triển yêu cầu phần mềm bao gồm nhiều pha quan trọng để xác định, thu thập, và quản lý yêu cầu cho dự án phần mềm. Dưới đây là một tóm tắt ngắn gọn về 4 pha chính trong một framework phát triển yêu cầu phần mềm.

❖ **Pha Xác định Yêu cầu (Requirements Identification):**

- Phía Kỹ thuật: Trong ngữ cảnh kỹ thuật, pha này có thể đòi hỏi xác định cụ thể các hệ thống, công nghệ, và giao diện mà dự án sẽ tương tác hoặc tích hợp. Các tài nguyên cần được xác định, bao gồm cơ sở dữ liệu, API, và hệ thống phụ trợ.
- Phía Kinh doanh: Chuyên gia kinh doanh cần tập trung vào việc xác định các mục tiêu kinh doanh, giá trị kỳ vọng, và cơ hội thị trường. Họ cũng cần xác định các đối thủ cạnh tranh và khách hàng tiềm năng.

❖ **Pha Thu Thập Yêu cầu (Requirements Elicitation):**

- Phía Kỹ thuật: Các kỹ thuật thu thập thông tin có thể liên quan đến việc thăm dò kiến thức về công nghệ cụ thể, thực hiện kiểm tra bảo mật, hoặc thậm chí là việc đánh giá khả năng tích hợp với các hệ thống sẵn có.
- Phía Kinh doanh: Chuyên gia kinh doanh có thể sử dụng phân tích thị trường và phản hồi từ khách hàng để thu thập yêu cầu về sản phẩm và dịch vụ mong muốn. Cuộc thảo luận với các bên liên quan kinh doanh khác có thể cần thiết để hiểu rõ hơn về mục tiêu và chiến lược.

❖ **Pha Phân tích Yêu cầu (Requirements Analysis):**

- Phía Kỹ thuật: Trong môi trường kỹ thuật, phân tích yêu cầu có thể bao gồm việc thiết kế các khung sườn kỹ thuật, xác định các cơ chế bảo mật, và đảm bảo tích hợp sao cho hiệu quả và khả thi.
- Phía Kinh doanh: Chuyên gia kinh doanh có thể sử dụng phân tích SWOT để xác định các yếu điểm, mạnh điểm, cơ hội, và đe dọa liên quan đến mô hình kinh doanh. Họ cũng có thể thực hiện phân tích ROI (Return on Investment) để đánh giá tính khả thi của dự án.

❖ **Pha Quản lý Yêu cầu (Requirements Management):**

- Phía Kỹ thuật: Quản lý yêu cầu từ góc độ kỹ thuật đòi hỏi việc đảm bảo tích hợp liên tục, kiểm soát thay đổi kỹ thuật, và duy trì tài liệu yêu cầu kỹ thuật cập nhật.
- Phía Kinh doanh: Chuyên gia kinh doanh có thể liên quan đến việc quản lý yêu cầu về nguồn lực, ngân sách, và lịch trình. Họ cũng có trách nhiệm

đảm bảo rằng dự án vẫn phù hợp với mục tiêu kinh doanh và chiến lược tổng thể của tổ chức.

Các pha này thường phụ thuộc vào nhau và cần được thực hiện song song để đảm bảo rằng cả hai khía cạnh kỹ thuật và kinh doanh của dự án phát triển phần mềm được quản lý và hiểu rõ. Các pha này cũng có thể thực hiện theo các chu kỳ lặp lại để đảm bảo tính nhất quán và phản ánh đúng nhu cầu của dự án trong suốt quá trình phát triển.

2. Giải thích Figure 4.1 (>1 trang)

Business analyst đóng vai trò trung gian quan trọng trong việc giao tiếp giữa khách hàng và các bên liên quan đến quá trình phát triển phần mềm hoặc dự án. Vai trò này bao gồm nhiều nhiệm vụ và trách nhiệm quan trọng như sau:

a) Thu thập yêu cầu từ khách hàng:
- Cuộc họp hoặc phiên thảo chất vấn: Bắt đầu bằng việc tổ chức cuộc họp hoặc phiên thảo chất vấn với khách hàng để hiểu rõ mục tiêu của họ và vấn đề họ đang gặp phải.
- Sử dụng kỹ thuật ghi chép: Sử dụng kỹ thuật ghi chép để lưu lại thông tin quan trọng và đảm bảo không bỏ sót thông tin quan trọng.
- Sử dụng mẫu câu hỏi: Sử dụng các mẫu câu hỏi cụ thể để hỏi khách hàng về các khía cạnh quan trọng của yêu cầu.

b) Hiểu biết về người dùng và thị trường:
- Phân tích thị trường: Tìm hiểu về thị trường mà sản phẩm hoặc dự án sẽ hoạt động trong đó, bao gồm đối thủ cạnh tranh, xu hướng thị trường, và tiềm năng phát triển.
- Phân tích khách hàng mục tiêu: Xác định và nghiên cứu các đối tượng mục tiêu của sản phẩm hoặc dự án.
- Tiến hành cuộc khảo sát hoặc phỏng vấn: Thu thập ý kiến, thông tin và phản hồi từ các khách hàng tiềm năng.
- Xây dựng hồ sơ người dùng: Tạo ra hồ sơ chi tiết về các loại người dùng và khách hàng tiềm năng.

c) Phân tích và biên dịch yêu cầu:
- Phân tách yêu cầu: Phân tích yêu cầu thành các yếu tố nhỏ hơn để hiểu rõ hơn về mỗi phần của sản phẩm hoặc dự án.
- Xác định mối quan hệ giữa các yêu cầu: Hiểu cách các yêu cầu tương tác và phụ thuộc vào nhau.

- Sử dụng công cụ và kỹ thuật phân tích: Sử dụng các công cụ phân tích như biểu đồ luồng dữ liệu, biểu đồ UML, hoặc biểu đồ use case để mô tả các yêu cầu một cách trực quan.
- Bảo trì và cập nhật yêu cầu: Theo dõi và cập nhật yêu cầu theo thời gian khi có thay đổi hoặc phát sinh yêu cầu mới để đảm bảo tính nhất quán và phù hợp tiếp tục.

d) Giao tiếp với nhóm phát triển:
- Trình bày yêu cầu: Sử dụng ngôn ngữ dễ hiểu để trình bày yêu cầu và sử dụng ví dụ và minh họa để giúp nhóm phát triển hiểu rõ hơn.
- Theo dõi tiến độ: Theo dõi tiến độ công việc và đảm bảo rằng nhóm phát triển đang làm việc theo đúng hướng và kế hoạch.
- Giải quyết câu hỏi và thảo luận: Mở cửa để nhóm phát triển đặt câu hỏi và thảo luận để làm rõ các điểm quan trọng trong yêu cầu.

e) Đảm bảo tính nhất quán và phù hợp:
- Kiểm tra mâu thuẫn và xung đột: Đảm bảo rằng không có mâu thuẫn hoặc xung đột giữa các yêu cầu khác nhau và sử dụng quy tắc đặt tên và ghi chép rõ ràng để đảm bảo tính nhất quán.
- Xác định mục tiêu và phạm vi: Đảm bảo rằng yêu cầu đáp ứng mục tiêu và phạm vi của dự án hoặc sản phẩm.
- Kiểm tra với khách hàng và người dùng: Liên hệ với khách hàng và người dùng cuối để đảm bảo rằng yêu cầu đáp ứng nhu cầu và mong đợi của họ.

f) Hỗ trợ trong quá trình phát triển và kiểm tra:
- Hỗ trợ giải quyết thay đổi và thảo luận: Tham gia vào quá trình giải quyết thay đổi và thảo luận để đảm bảo tính nhất quán và phù hợp của các điều chỉnh này.
- Theo dõi tiến độ: Theo dõi tiến độ công việc và đảm bảo rằng nhóm phát triển đang làm việc theo đúng hướng và kế hoạch.
- Kiểm tra sản phẩm: Tham gia vào quá trình kiểm tra sản phẩm để đảm bảo rằng sản phẩm đáp ứng yêu cầu và chất lượng mong đợi.

Vai trò của nhà phân tích kinh doanh là cầu nối quan trọng giữa khách hàng và nhóm phát triển, đảm bảo rằng yêu cầu được hiểu rõ và thực hiện đúng cách để tạo ra sản phẩm hoặc dự án thành công.

3. Công việc của người làm BA: Knowledge & Skills . Để trở thành BA

a) **Nhiệm vụ của BA**

Những nhiệm vụ của một Business Analyst (BA) mà bạn đã liệt kê là rất chi tiết và phản ánh chính xác vai trò quan trọng của BA trong quá trình phát triển dự án hoặc sản phẩm. Dưới đây là một số điểm quan trọng trong số các nhiệm vụ này:

- **Định nghĩa yêu cầu kinh doanh:** BA là người chịu trách nhiệm đảm bảo rằng mục tiêu kinh doanh của dự án được xác định rõ ràng và được hiểu bởi tất cả các bên liên quan. Việc định nghĩa rõ ràng mục tiêu này là cơ sở quan trọng cho việc xác định các yêu cầu cụ thể.
- **Lập kế hoạch tiếp cận yêu cầu:** BA cần phát triển kế hoạch chi tiết về việc thu thập, phân tích, xác nhận và quản lý yêu cầu. Kế hoạch này cần phù hợp với kế hoạch tổng thể của dự án để đảm bảo tính nhất quán và hiệu quả.
- **Phân tích yêu cầu:** Phân tích yêu cầu là một quá trình quan trọng để hiểu rõ những gì khách hàng thực sự cần và muốn. Điều này bao gồm cả việc tìm kiếm yêu cầu phát sinh và ngụ ý để đảm bảo rằng tất cả các khía cạnh quan trọng đã được xem xét.
- **Ghi chép tài liệu yêu cầu:** Việc viết tài liệu yêu cầu là bước quan trọng để truyền đạt thông tin một cách rõ ràng và có cấu trúc. Tài liệu này thường bao gồm các mô hình, biểu đồ và ví dụ để giải thích yêu cầu một cách trực quan.
- **Truyền đạt yêu cầu:** BA phải có khả năng truyền đạt yêu cầu một cách hiệu quả cho tất cả các bên liên quan, bao gồm cả nhóm phát triển. Điều này có thể bao gồm việc sử dụng các phương tiện truyền thông khác nhau để đảm bảo sự hiểu rõ.
- **Xác nhận yêu cầu chính:** Đảm bảo rằng các yêu cầu đã được xác minh và rằng giải pháp được đề xuất dựa trên yêu cầu có khả năng đáp ứng mục tiêu kinh doanh.
- **Hỗ trợ ưu tiên hóa yêu cầu:** Hỗ trợ quá trình thương thảo và ưu tiên hóa yêu cầu giữa các bên liên quan khác nhau để đảm bảo rằng các yêu cầu quan trọng nhất được ưu tiên cao.
- **Quản lý yêu cầu:** Trong toàn bộ vòng đời của dự án, BA cần theo dõi tình trạng yêu cầu, xác minh tính truy vết của chúng và quản lý các thay đổi đối với yêu cầu khi cần thiết.

Những nhiệm vụ này đóng vai trò quan trọng trong việc đảm bảo rằng dự án hoặc sản phẩm được phát triển đúng hướng và đáp ứng được mục tiêu kinh doanh của khách hàng.

b) **Kỹ năng của BA**

Nhà phân tích (BA - Business Analyst) đóng một vai trò quan trọng trong quá trình phát triển dự án phần mềm và quản lý yêu cầu. Để hoàn thành nhiệm vụ này một cách hiệu quả, họ cần phải có nhiều kỹ năng mềm cũng như kỹ năng kỹ thuật. Dưới đây là một số kỹ năng quan trọng mà một nhà phân tích cần phải phát triển:

- **Kỹ năng lắng nghe:** Lắng nghe là một kỹ năng quan trọng để hiểu rõ yêu cầu của khách hàng và các bên liên quan. Điều này bao gồm việc tập trung vào người đang nói, xử lý thông tin một cách cẩn thận và đặt câu hỏi để làm rõ.
- **Kỹ năng phỏng vấn và đặt câu hỏi:** Nhà phân tích cần phải biết cách đặt câu hỏi hiệu quả để thu thập thông tin và đảm bảo hiểu rõ yêu cầu của người dùng và các bên liên quan.
- **Kỹ năng suy nghĩ nhanh**: Kỹ năng suy nghĩ nhanh giúp nhà phân tích xử lý thông tin mới một cách nhanh chóng và đưa ra câu hỏi hoặc phân tích thông tin một cách hiệu quả.
- **Kỹ năng phân tích:** Nhà phân tích cần phải có khả năng phân tích thông tin từ nhiều nguồn khác nhau và xác định các yêu cầu cụ thể.
- **Kỹ năng tư duy hệ thống:** Kỹ năng này giúp nhà phân tích nhìn vào toàn cảnh và hiểu cách các yêu cầu tương tác với nhau và với hệ thống tổng thể.
- **Kỹ năng học tập:** Với sự phát triển nhanh chóng của công nghệ và ngành công nghiệp, nhà phân tích cần phải có khả năng học tập liên tục để cập nhật kiến thức và kỹ năng.
- **Kỹ năng giao tiếp:** Giao tiếp là quá trình truyền đạt thông tin một cách rõ ràng và hiệu quả. Nhà phân tích cần phải có khả năng giao tiếp bằng cả văn bản và lời nói để diễn giải yêu cầu cho các bên liên quan.
- **Kỹ năng tổ chức:** Nhà phân tích cần phải tổ chức thông tin một cách có hệ thống để dễ dàng quản lý và tra cứu.

- **Kỹ năng mô hình hóa:** Mô hình hóa là quá trình biểu diễn thông tin dưới dạng biểu đồ, sơ đồ hoặc các hình thức khác để giúp người khác hiểu một cách dễ dàng. Nhà phân tích cần phải biết sử dụng các phương tiện này.
- **Kỹ năng giao tiếp giữa cá nhân:** Nhà phân tích cần phải làm việc với nhiều cá nhân và nhóm khác nhau. Họ cần phải biết cách hòa giải và tạo sự đồng thuận trong quá trình làm việc.
- **Tính sáng tạo:** Sáng tạo giúp nhà phân tích đưa ra các giải pháp và yêu cầu mới, đặc biệt khi phải đối mặt với những thách thức phức tạp.

Những kỹ năng này cùng với kiến thức về lĩnh vực kinh doanh và công nghệ làm cho một nhà phân tích trở thành một thành viên quan trọng của dự án phát triển phần mềm.

c) **Kiến thức của BA**

- Bên cạnh việc sở hữu những kỹ năng cụ thể và đặc điểm cá nhân, những người làm nhà phân tích kinh doanh cần phải tích luỹ một kiến thức đa dạng, một phần lớn thông qua những bài học từ thực tế. Họ cần phải nắm vững các phương pháp kỹ thuật yêu cầu tiên tiến và biết cách áp dụng chúng trong ngữ cảnh của các giai đoạn phát triển phần mềm khác nhau. Thêm vào đó, khả năng hướng dẫn và thuyết phục những người không quen thuộc với các tiêu chuẩn và phương thức đã được thiết lập cũng là yếu tố quan trọng. Người phân tích hiệu quả cần phải sử dụng một loạt công cụ kỹ thuật đa dạng và biết cân nhắc khi nào nên và khi nào không nên áp dụng từng phương pháp.
- Các chuyên gia phân tích cần thực hiện các hoạt động liên quan đến phát triển và quản lý yêu cầu trong suốt quá trình thực hiện dự án. Một nhà phân tích có hiểu biết về quản lý dự án, chu kỳ phát triển, quản lý rủi ro và đảm bảo chất lượng có thể giúp ngăn ngừa các vấn đề liên quan đến yêu cầu gây khó khăn cho tiến trình dự án. Trong môi trường phát triển thương mại, những nhà phân tích còn có thể hưởng lợi từ kiến thức về các khái niệm quản lý sản phẩm. Những chuyên gia phân tích cũng cần hiểu biết về kiến trúc hệ thống và môi trường hoạt động, để có thể tham gia vào cuộc trò chuyện kỹ thuật về ưu tiên và yêu cầu ngoài chức năng.
- Kiến thức về lĩnh vực doanh nghiệp, ngành công nghiệp và tổ chức là một tài sản quý báu của một nhà phân tích hiệu quả. Những người làm phân tích hiểu rõ về hoạt động kinh doanh có thể giảm thiểu sự hiểu lầm với người dùng. Những nhà phân tích có kiến thức về tổ chức và ngành công nghiệp thường phát hiện ra những giả thuyết và yêu cầu chưa được nêu ra. Họ có

thể đề xuất cách người dùng có thể cải thiện quy trình kinh doanh của họ hoặc đề xuất các chức năng có giá trị mà không có bên liên quan nào đã suy nghĩ đến. Hiểu biết về lĩnh vực ngành công nghiệp có thể đặc biệt hữu ích trong môi trường thương mại, giúp những nhà phân tích có thể thực hiện các phân tích thị trường và sản phẩm cạnh tranh một cách hiệu quả.

4. Tiến trình phát triển phần mềm Agile & Waterfall (Khảo sát trên internet, >3 trang). BA trong các dự án Agile

Tiến trình phát triển phần mềm Agile và Waterfall là hai phương pháp quan trọng và phổ biến được ứng dụng trong ngành công nghiệp phần mềm. Cả hai phương pháp này đều mang lại lợi ích và hạn chế riêng, và lựa chọn giữa Agile và Waterfall phụ thuộc vào nhiều yếu tố, bao gồm loại dự án, yêu cầu của khách hàng và môi trường làm việc. Cụ thể:

❖ **Phương pháp phát triển phần mềm Agile:**

Phát triển phần mềm Agile là một tiến trình linh hoạt dựa trên việc thiết lập mục tiêu và tương tác thường xuyên với khách hàng. Dưới đây là một số điểm quan trọng về Agile:

- **Ưu điểm của Agile:**
  + Linh hoạt: Agile cho phép thay đổi yêu cầu và phản hồi nhanh chóng trong quá trình phát triển.
  + Tương tác khách hàng: Khách hàng tham gia tích cực trong quá trình phát triển và có cơ hội kiểm tra sản phẩm thường xuyên.
  + Phản hồi thường xuyên: Agile tạo điều kiện cho việc kiểm tra và đánh giá liên tục, giúp phát hiện lỗi sớm và điều chỉnh chúng nhanh chóng.
- **Nguyên tắc Agile:**
  + Ưu tiên người dùng: Tập trung vào việc đáp ứng nhu cầu của người dùng cuối.
  + Phản hồi thường xuyên: Liên tục kiểm tra và đánh giá tiến trình và sản phẩm.
  + Tương tác tốt: Thành viên nhóm làm việc chặt chẽ với khách hàng và với nhau để đảm bảo hiệu suất tốt nhất.
- **Kỹ thuật Agile phổ biến:**
  + Scrum: Sử dụng các sprint ngắn hạn để phát triển sản phẩm và có một cuộc họp Scrum hàng ngày để theo dõi tiến trình.

+ Kanban: Sử dụng bảng Kanban để quản lý công việc và theo dõi quy trình phát triển.

❖ **Phương pháp phát triển phần mềm Waterfall:**

Phát triển phần mềm Waterfall là một tiến trình tuần tự, trong đó mỗi giai đoạn phải hoàn thành trước khi bắt đầu giai đoạn tiếp theo. Dưới đây là một số điểm quan trọng về Waterfall:

- **Ưu điểm của Waterfall:**
  + Cấu trúc rõ ràng: Waterfall có cấu trúc rõ ràng và dễ quản lý.
  + Phân chia công việc: Mỗi giai đoạn phát triển được hoàn thành trước khi chuyển sang giai đoạn tiếp theo, giúp kiểm soát dự án.
- **Nhược điểm của Waterfall:**
  + Không linh hoạt: Khó điều chỉnh yêu cầu sau khi bắt đầu giai đoạn thực hiện.
  + Khó cho phản hồi khách hàng: Khách hàng không có cơ hội kiểm tra sản phẩm cho đến khi hoàn thành.
- **Các giai đoạn chính trong Waterfall:**
  + Thu thập yêu cầu: Xác định yêu cầu dự án.
  + Thiết kế: Xây dựng thiết kế chi tiết của sản phẩm.
  + Thực hiện: Phát triển sản phẩm dựa trên thiết kế.
  + Kiểm tra: Kiểm tra và kiểm tra chất lượng sản phẩm.
  + Triển khai: Đưa sản phẩm vào sử dụng thực tế.
  + Bảo trì: Duy trì và hỗ trợ sản phẩm sau triển khai.

❖ **Sự so sánh giữa Agile và Waterfall:**

- Phản hồi và linh hoạt: Agile cho phép phản hồi và thay đổi linh hoạt trong suốt quá trình phát triển, trong khi Waterfall yêu cầu yêu cầu được xác định rõ ràng từ đầu và khó thay đổi sau khi bắt đầu.
- Tương tác khách hàng: Agile thúc đẩy tương tác chặt chẽ với khách hàng, trong khi Waterfall thường không cho phép khách hàng kiểm tra sản phẩm cho đến khi hoàn thành.
- Quản lý dự án: Waterfall có cấu trúc quản lý dự án rõ ràng hơn, trong khi Agile thường đòi hỏi sự đồng tình và tự quản lý của nhóm.
- Kiểm tra và đảm bảo chất lượng: Waterfall thường có giai đoạn kiểm tra cuối cùng, trong khi Agile thường có kiểm tra và đánh giá thường xuyên.

- Sản phẩm cuối cùng: Waterfall cung cấp sản phẩm hoàn chỉnh ở cuối dự án, trong khi Agile cung cấp các phiên bản sản phẩm có thể sử dụng theo thời gian.

Sự lựa chọn giữa Agile và Waterfall phụ thuộc vào loại dự án, yêu cầu của khách hàng và môi trường làm việc. Một số dự án có thể tận dụng lợi ích của cả hai phương pháp, thực hiện Agile cho phần mềm cơ bản và Waterfall cho phần mềm cốt lõi hoặc ứng dụng lớn. Quan trọng nhất, quyết định nên dựa trên mục tiêu và đặc điểm cụ thể của dự án để đảm bảo sự thành công.

5. Mô hình yêu cầu phần mềm : Liệt kê các biểu đồ có thể sử dụng cho mô hình yêu cầu phần mềm (> 3 trang)

Mô hình yêu cầu phần mềm thường được biểu diễn và trình bày bằng nhiều loại biểu đồ khác nhau để giúp hiểu rõ và truyền đạt thông tin một cách hiệu quả. Dưới đây là danh sách các biểu đồ phổ biến có thể sử dụng cho mô hình yêu cầu phần mềm:

- Biểu đồ Use Case (Sử dụng trường hợp): Biểu đồ Use Case giúp xác định và biểu diễn các tình huống sử dụng hệ thống từ góc độ người dùng cuối. Các thành phần chính bao gồm:
  + Tác nhân ngoại vi (Actors): Đại diện cho các thực thể bên ngoài tương tác với hệ thống, chẳng hạn như người dùng cuối hoặc hệ thống ngoại vi.
  + Use Case (Trường hợp sử dụng): Biểu diễn các tình huống cụ thể mà hệ thống cần hỗ trợ và cung cấp giải pháp cho nhu cầu của người dùng. Mỗi Use Case có thể bao gồm các tương tác khác nhau với hệ thống.
- Biểu đồ Luồng Dữ liệu (Data Flow Diagram - DFD): DFDs biểu diễn cách dữ liệu di chuyển qua hệ thống và xác định các quy trình xử lý dữ liệu. Các thành phần chính bao gồm:
  + Quy trình (Process): Biểu diễn các hoạt động xử lý dữ liệu và thông tin.
  + Lưu trữ dữ liệu (Data Store): Đại diện cho nơi dữ liệu được lưu trữ trong hệ thống.
  + Luồng dữ liệu (Data Flow): Biểu diễn dòng dữ liệu di chuyển giữa các quy trình và lưu trữ dữ liệu.
- Biểu đồ Luồng Công việc (Workflow Diagram): Thường được sử dụng để mô tả chuỗi các hoạt động hoặc công việc trong quy trình làm việc. Biểu đồ này giúp hiểu rõ quy trình làm việc hiện tại và đề xuất các cải tiến.

- Biểu đồ Lớp (Class Diagram): Biểu diễn cấu trúc của hệ thống bằng cách xác định các lớp (classes), thuộc tính và phương thức của chúng. Class Diagrams thường liên quan đến thiết kế hệ thống và mô hình hóa các đối tượng quan trọng.
- Biểu đồ Trình tự (Sequence Diagram): Mô tả các tương tác giữa các đối tượng trong hệ thống theo thời gian. Nó biểu diễn thứ tự và luồng làm việc của các đối tượng trong một tình huống cụ thể.
- Biểu đồ Sự kiện (Event Diagram): Sử dụng để mô tả sự kiện và các xử lý sự kiện trong hệ thống. Điều này giúp nắm bắt và xử lý các sự kiện từ người dùng hoặc hệ thống ngoại vi.
- Biểu đồ Tương tác (Interaction Diagrams): Bao gồm Sequence Diagrams và Communication Diagrams, giúp biểu diễn cách các đối tượng tương tác trong một tình huống cụ thể, với Sequence Diagrams tập trung vào thứ tự thực hiện và Communication Diagrams tập trung vào mối quan hệ giữa các đối tượng.
- Biểu đồ Trạng thái (State Diagram): Mô tả các trạng thái mà một đối tượng hoặc hệ thống có thể tồn tại và cách chuyển đổi giữa các trạng thái này. State Diagrams thường được sử dụng để mô hình hóa hành vi của các đối tượng.
- Biểu đồ Gantt (Gantt Chart): Sử dụng để biểu diễn kế hoạch thời gian cho dự án, xác định thời gian thực hiện các yêu cầu và nhiệm vụ.
- Biểu đồ Cốt lõi (Core Diagrams): Tập hợp các biểu đồ quan trọng như Use Case Diagrams, Class Diagrams và Sequence Diagrams để tập trung vào các khía cạnh quan trọng của hệ thống.
- Biểu đồ Flowchart: Sử dụng hình khối và mũi tên để biểu diễn quá trình làm việc hoặc luồng công việc. Flowcharts thường được sử dụng để đơn giản hóa và hiển thị các quy trình và quá trình công việc.

Mỗi loại biểu đồ có mục tiêu và ứng dụng riêng trong quá trình phát triển phần mềm, và sự lựa chọn phụ thuộc vào mục tiêu cụ thể của dự án và loại thông tin cần truyền đạt.

6. Chọn kỹ thuật mô hình hay biểu diễn thích hợp (tài liệu [0], Tab 12.2, pg. 225--, >5 trang). Cho ví dụ minh họa

Lựa chọn kỹ thuật mô hình hoặc biểu diễn thích hợp phụ thuộc vào mục tiêu và bối cảnh của dự án phần mềm cũng như đối tượng mà bạn đang làm việc. Dưới đây là một số tình huống và gợi ý về việc chọn kỹ thuật thích hợp:

- **Mục tiêu là thể hiện tương tác thời gian thực và luồng thông tin phức tạp:** Sử dụng biểu đồ Activity Diagram để biểu diễn các hoạt động và luồng công việc trong hệ thống. Điều này giúp hiểu rõ các quá trình xử lý và tương tác thời gian thực.
- **Mục tiêu là đo lường hiệu suất và tối ưu hóa:** Sử dụng biểu đồ Performance Diagram để theo dõi và đánh giá hiệu suất của hệ thống, bao gồm thời gian phản hồi và tải công việc.
- **Mục tiêu là trình bày kiến thức chuyên ngành:** Sử dụng biểu đồ Domain Model để mô hình hóa kiến thức chuyên ngành, bao gồm các thực thể, quan hệ và thuộc tính liên quan đến lĩnh vực cụ thể.
- **Mục tiêu là trình bày tương tác giữa các hệ thống:** Sử dụng biểu đồ Component Diagram để biểu diễn các thành phần của hệ thống và cách chúng tương tác với nhau.
- **Mục tiêu là quản lý rủi ro và an ninh:** Sử dụng biểu đồ Use Case để biểu diễn các tình huống đe dọa và cách hệ thống phản ứng để đảm bảo an ninh thông tin.
- **Mục tiêu là tạo sáng tạo và thử nghiệm ý tưởng mới:** Sử dụng biểu đồ Mind Map để hỗ trợ việc tạo ra ý tưởng sáng tạo và khám phá các hướng tiếp cận mới.
- **Mục tiêu là mô phỏng và kiểm tra hệ thống:** Sử dụng biểu đồ Simulation Diagram để mô phỏng hoạt động của hệ thống và kiểm tra tính đúng đắn của nó trước khi triển khai.
- **Mục tiêu là tương tác với khách hàng hoặc người dùng cuối:** Sử dụng biểu đồ User Interface (UI) để thiết kế giao diện người dùng và hiển thị cách người dùng sẽ tương tác với hệ thống.
- **Mục tiêu là trình bày thông tin phức tạp cho đối tượng chuyên gia công nghệ:** Chọn các biểu đồ phức tạp hơn như Activity Diagrams, State Diagrams, và Class Diagrams để truyền đạt thông tin chi tiết và kỹ thuật cho đối tượng chuyên gia.
- **Mục tiêu là tổ chức và quản lý dự án phức tạp:** Sử dụng biểu đồ Cấu trúc tổ chức và biểu đồ Gantt kết hợp để quản lý và tổ chức các tác vụ, nguồn lực và tiến độ dự án một cách hiệu quả.

Hãy nhớ rằng sự linh hoạt là quan trọng, và bạn có thể kết hợp nhiều loại biểu đồ để đảm bảo rằng bạn đang truyền đạt thông tin một cách toàn diện và hiệu quả cho đối tượng và mục tiêu cụ thể của mình trong dự án phần mềm.

7. Trình bày ngắn gọn 14 biểu đồ UML .Biểu đồ UML nào thích hợp cho mô hình yêu cầu phần mềm

Tất cả các loại biểu đồ UML đều có vai trò và mục tiêu riêng biệt trong mô hình hóa và mô tả hệ thống phần mềm. Dưới đây là sự bổ sung thông tin cho từng loại biểu đồ:

- **Biểu đồ Use Case (Use Case Diagram):**
  + Sử dụng để biểu diễn tương tác giữa người dùng (tác nhân) và hệ thống.
  + Hữu ích để xác định các chức năng cần phát triển và hiểu rõ các tình huống sử dụng của hệ thống từ góc độ người dùng cuối.
- **Biểu đồ Lớp (Class Diagram):**
  + Biểu diễn cấu trúc dữ liệu của hệ thống, bao gồm các lớp, thuộc tính và phương thức.
  + Mô tả các đối tượng trong hệ thống và mối quan hệ giữa chúng, như kế thừa, giao tiếp, và liên kết.
- **Biểu đồ Trạng thái (State Diagram):**
  + Dùng để mô tả các trạng thái khác nhau của đối tượng trong hệ thống và cách chuyển đổi giữa chúng.
  + Thường được sử dụng để biểu diễn hành vi của các đối tượng theo thời gian và trong các tình huống khác nhau.
- **Biểu đồ Sequence (Sequence Diagram):**
  + Mô tả tương tác giữa các đối tượng trong hệ thống theo thời gian, đặc biệt là trong các tình huống sử dụng cụ thể.
  + Cho phép xem xét thứ tự các cuộc gọi phương thức giữa các đối tượng và truyền thông tin giữa chúng.
- **Biểu đồ Communication (Communication Diagram):**
  + Tương tự như Sequence Diagram, tập trung vào cách các đối tượng liên lạc với nhau và thứ tự truyền thông tin giữa chúng.
  + Giúp hiểu cách các đối tượng trong hệ thống liên kết và giao tiếp với nhau trong một tình huống cụ thể.
- **Biểu đồ Sự kiện (Activity Diagram):**
  + Mô tả các hoạt động và luồng công việc trong quy trình làm việc hoặc quy trình kinh doanh.
  + Hữu ích để biểu diễn quy trình làm việc hiện tại và đề xuất cải tiến.
- **Biểu đồ Giao diện người dùng (UI Diagram):**

+ Sử dụng để thiết kế giao diện người dùng của hệ thống, bao gồm vị trí các yếu tố như cửa sổ, nút bấm và biểu mẫu.
+ Được dùng để tạo giao diện người dùng thân thiện và tương tác.

– **Biểu đồ Component (Component Diagram):**

+ Biểu diễn cấu trúc vật lý của hệ thống và các thành phần phần mềm của nó.
+ Thường được sử dụng để mô tả cách các thành phần liên kết với nhau trong kiến trúc hệ thống.

– **Biểu đồ Lớp Đối tượng (Object Diagram):**

+ Mô tả trạng thái cụ thể của các đối tượng và mối quan hệ giữa chúng tại một thời điểm cụ thể.
+ Giúp kiểm tra và hiểu rõ trạng thái của hệ thống trong các tình huống cụ thể.

– **Biểu đồ Deployment (Deployment Diagram):**

+ Biểu diễn cách các thành phần phần mềm được triển khai trên các nút mạng hoặc máy chủ vật lý.
+ Hữu ích cho việc lên kế hoạch triển khai và quản lý môi trường triển khai.

– **Biểu đồ Package (Package Diagram):**

+ Mô tả cách các thành phần phần mềm được tổ chức thành các gói (packages).
+ Được sử dụng để quản lý và tổ chức mã nguồn một cách hệ thống.

– **Biểu đồ Lớp Sơ đồ (Composite Structure Diagram):**

+ Mô tả cấu trúc nội tại phức tạp của một đối tượng hoặc lớp.
+ Cho phép mô hình hóa cấu trúc bên trong một đối tượng hoặc lớp.

– **Biểu đồ Được Điều Khiển (Control Flow Diagram):**

+ Sử dụng để biểu diễn dòng điều khiển trong quy trình hoặc thuật toán.
+ Hữu ích để hiểu cách luồng điều khiển chương trình hoạt động.

– **Biểu đồ Hợp Đồng (Object Constraint Language - OCL):**

  + Sử dụng để mô tả ràng buộc và điều kiện của các yêu cầu.
  + Thường đi kèm với các biểu đồ khác để làm rõ ràng và định nghĩa điều kiện cụ thể.

Các loại biểu đồ UML có vai trò quan trọng trong việc hiểu, thiết kế và quản lý hệ thống phần mềm. Lựa chọn loại biểu đồ cụ thể phụ thuộc vào mục tiêu và nhiệm vụ của dự án, cũng như sự phức tạp của hệ thống cần mô hình hóa.

8. Trình bày các đặc trưng phi chức năng hay thuộc tính chất lượng phần mềm.

❖ **Các đặc trưng phi chức năng/ thuộc tính phần mềm:**

– **Các thuộc tính ngoại kiểm:**

+ **Availability**

Tính khả dụng là thước đo thời gian hoạt động theo kế hoạch trong đó các dịch vụ của hệ thống có sẵn để sử dụng và hoạt động đầy đủ. Tính khả dụng bằng tỷ lệ thời gian tăng với tổng up time và down time. Tính khả dụng còn bằng thời gian trung bình giữa các lần hỏng hóc (MTBF) cho hệ thống chia cho tổng MTBF và thời gian sửa chữa trung bình (MTTR) hệ thống sau khi gặp lỗi. Thời gian bảo trì theo lịch trình cũng ảnh hưởng đến tính khả dụng. Tính khả dụng có liên quan chặt chẽ đến độ tin cậy và bị ảnh hưởng mạnh mẽ bởi khả năng bảo trì.

+ **Installability**

Phần mềm không hữu ích cho đến khi nó được cài đặt trên thiết bị hoặc nền tảng thích hợp. Một số ví dụ về cài đặt phần mềm là: tải ứng dụng xuống điện thoại hoặc máy tính bảng; di chuyển phần mềm từ PC sang máy chủ web; cập nhật hệ điều hành; cài đặt một hệ thống thương mại khổng lồ, chẳng hạn như một công cụ hoạch định nguồn lực doanh nghiệp; tải bản cập nhật chương trình cơ sở xuống hộp set-top TV cáp; và cài đặt ứng dụng người dùng cuối lên PC. Khả năng cài đặt mô tả mức độ dễ dàng để thực hiện các thao tác này một cách chính xác. Tăng khả năng cài đặt của hệ thống giúp giảm thời gian, chi phí, gián đoạn người dùng, tần suất lỗi và mức độ kỹ năng cần thiết cho hoạt động cài đặt.

+ **Integrity**

Tính toàn vẹn liên quan đến việc ngăn ngừa mất thông tin và duy trì tính chính xác của dữ liệu được nhập vào hệ thống. Yêu cầu về tính toàn vẹn: không có sai sót: dữ liệu hoặc ở trạng thái tốt và được bảo vệ, hoặc không. Dữ liệu cần được bảo vệ chống lại các mối đe dọa như mất mát hoặc hỏng hóc do tai nạn, các bộ dữ liệu có vẻ giống hệt nhau nhưng không khớp, thiệt hại vật lý đối với phương tiện lưu trữ, vô tình xóa tệp hoặc ghi đè dữ liệu của người dùng. Các cuộc tấn công có chủ ý cố tình làm hỏng hoặc đánh cắp dữ liệu cũng là một rủi ro. Bảo mật đôi khi được coi là một tập hợp con của tính toàn vẹn, bởi vì một số yêu cầu bảo mật nhằm ngăn chặn người dùng trái phép truy cập dữ liệu. Yêu cầu về tính toàn vẹn phải đảm bảo rằng dữ liệu nhận được từ các hệ thống khác khớp với những gì được gửi và ngược lại. Bản thân các
tệp thực thi phần mềm có thể bị tấn công, vì vậy tính toàn vẹn của

chúng cũng phải được bảo vệ. Tính toàn vẹn dữ liệu cũng đề cập đến tính chính xác và định dạng đúng của dữ liệu (Miller 2009). Điều này bao gồm các mối quan tâm như định dạng trường cho ngày, hạn chế các trường theo đúng kiểu dữ liệu hoặc độ dài, đảm bảo rằng các yếu tố dữ liệu có giá trị hợp lệ, kiểm tra mục nhập thích hợp trong một trường khi trường khác có giá trị nhất định, v.v

+ **Interoperability**

  Khả năng tương tác cho biết hệ thống có thể trao đổi dữ liệu và dịch vụ với các hệ thống phần mềm khác dễ dàng như thế nào và nó có thể tích hợp dễ dàng với các thiết bị phần cứng bên ngoài như thế nào. Để đánh giá khả năng tương tác, ta cần biết những ứng dụng nào khác mà người dùng sẽ sử dụng cùng với sản phẩm và dữ liệu nào họ mong đợi trao đổi Các yêu cầu về khả năng tương tác có thể chỉ ra rằng các định dạng trao đổi dữ liệu tiêu chuẩn được sử dụng để tạo điều kiện trao đổi thông tin với các hệ thống phần mềm khác

+ **Performance**

  Hiệu suất thể hiện khả năng đáp ứng của hệ thống đối với các yêu cầu và hành động khác nhau của người dùng, nhưng nó bao gồm nhiều hơn thế. Hiệu suất kém là một khó chịu cho người dùng đang chờ đợi một truy vấn để hiển thị kết quả. Nhưng các vấn đề về hiệu suất cũng có thể đại diện cho những rủi ro nghiêm trọng đối với an toàn, chẳng hạn như khi hệ thống kiểm soát quy trình thời gian thực bị quá tải. Yêu cầu hiệu suất nghiêm ngặt ảnh hưởng mạnh mẽ đến thiết kế phần mềm chiến lược và lựa chọn phần cứng, vì vậy hãy xác định mục tiêu hiệu suất phù hợp với môi trường hoạt động. Đáp ứng các yêu cầu về hiệu suất có thể khó khăn vì chúng phụ thuộc rất nhiều vào các yếu tố bên ngoài như tốc độ của máy tính đang được sử dụng, kết nối mạng và các thành phần phần cứng khác.

+ **Reliability**

  Xác suất phần mềm thực thi mà không bị lỗi trong một khoảng thời gian cụ thể được gọi là độ tin cậy. Các vấn đề về độ tin cậy có thể xảy ra do đầu vào không đúng, lỗi trong chính mã phần mềm, các thành phần không có sẵn khi cần thiết và lỗi phần cứng. Sức tải và tính khả dụng liên quan chặt chẽ đến độ tin cậy. Các cách để chỉ định và đo lường độ tin cậy của phần mềm bao gồm tỷ lệ phần trăm các hoạt động được hoàn thành chính xác, độ dài trung bình thời gian hệ thống chạy trước khi hỏng hóc (thời gian trung bình giữa các lần hỏng hóc hoặc MTBF) và

xác suất lỗi tối đa có thể chấp nhận được trong một khoảng thời gian nhất định. Thiết lập các yêu cầu về độ tin cậy định lượng dựa trên mức độ nghiêm trọng của tác động nếu xảy ra lỗi và liệu chi phí tối đa hóa độ tin cậy có hợp lý hay không. Các hệ thống đòi hỏi độ tin cậy cao cũng nên được thiết kế để có khả năng xác minh cao để dễ dàng tìm ra các lỗi có thể ảnh hưởng đến độ tin cậy

+ **Robustness**
Sức tải là mức độ mà hệ thống tiếp tục hoạt động bình thường khi phải đối mặt với đầu vào không hợp lệ, lỗi trong các thành phần phần mềm hoặc phần cứng được kết nối, tấn công bên ngoài hoặc điều kiện hoạt động bất ngờ. Phần mềm phục hồi từ các tình huống có vấn đề và tha thứ cho những sai lầm của người dùng. Nó phục hồi từ các lỗi nội bộ mà không ảnh hưởng đến trải nghiệm người dùng cuối. Lỗi phần mềm được xử lý theo cách mà người dùng cho là hợp lý, không gây phiền nhiễu. Lỗi phần mềm được xử lý theo cách mà người dùng cho là hợp lý, không gây phiền nhiễu.

+ **Safety**
Các yêu cầu an toàn liên quan đến sự cần thiết phải ngăn chặn một hệ thống thực hiện bất kỳ thương tích nào cho người hoặc thiệt hại về tài sản. Các yêu cầu an toàn có thể được quyết định bởi các quy định của chính phủ hoặc các quy tắc kinh doanh khác và các vấn đề pháp lý hoặc chứng nhận có thể liên quan đến việc đáp ứng các yêu cầu đó. Các yêu cầu an toàn thường được viết dưới dạng các điều kiện hoặc hành động mà hệ thống không được phép xảy ra. Mọi người hiếm khi bị thương do phát nổ bảng tính. Tuy nhiên, các thiết bị phần cứng được điều khiển bằng phần mềm chắc chắn có thể gây nguy hiểm đến tính mạng và chân tay. Ngay cả một số ứng dụng chỉ có phần mềm cũng có thể có các yêu cầu an toàn không rõ ràng.

+ **Security**
Bảo mật liên quan đến việc chặn truy cập trái phép vào các chức năng hoặc dữ liệu của hệ thống, đảm bảo rằng phần mềm được bảo vệ khỏi các cuộc tấn công phần mềm độc hại, v.v. Bảo mật là một vấn đề lớn với phần mềm Internet. Người dùng hệ thống thương mại điện tử muốn thông tin thẻ tín dụng của họ được bảo mật. Người lướt web không muốn thông tin cá nhân hoặc hồ sơ của các trang web họ truy cập được sử dụng không phù hợp. Các công ty muốn bảo vệ trang web của họ chống lại các cuộc tấn công từ chối dịch vụ hoặc hack.

+ **Usability**

  Khả năng sử dụng giải quyết vô số yếu tố cấu thành những gì mọi người mô tả thông tục là thân thiện với người dùng, dễ sử dụng và kỹ thuật của con người. Các nhà phân tích và nhà phát triển không nên nói về phần mềm "thân thiện" mà nên nói về phần mềm được thiết kế để sử dụng hiệu quả và không phô trương. Khả năng sử dụng đo lường nỗ lực cần thiết để chuẩn bị đầu vào cho một hệ thống, vận hành nó và giải thích đầu ra của nó

– **Các thuộc tính nội kiểm:**

+ **Efficiency**

  Hiệu quả có liên quan chặt chẽ đến thuộc tính chất lượng bên ngoài của hiệu suất. Hiệu quả là thước đo của hệ thống sử dụng dung lượng bộ xử lý, dung lượng đĩa, bộ nhớ hoặc băng thông truyền thông tốt như thế nào. Nếu một hệ thống tiêu thụ quá nhiều tài nguyên có sẵn, người dùng sẽ gặp phải hiệu suất bị suy giảm.

+ **Modifiability**

  Khả năng sửa đổi đề cập đến việc các thiết kế và mã phần mềm có thể được hiểu, thay đổi và mở rộng dễ dàng như thế nào. Khả năng sửa đổi bao gồm một số thuật ngữ thuộc tính chất lượng khác liên quan đến các hình thức bảo trì phần mềm khác nhau. Nó liên quan chặt chẽ đến khả năng xác minh. Nếu các nhà phát triển dự đoán thực hiện nhiều cải tiến, họ có thể chọn các phương pháp thiết kế tối đa hóa khả năng sửa đổi của phần mềm. Khả năng sửa đổi cao là rất quan trọng đối với các hệ thống sẽ trải qua sửa đổi thường xuyên, chẳng hạn như những hệ thống được thay đổi bằng cách sử dụng vòng đời gia tăng hoặc lặp đi lặp lại.

+ **Portability**

  Nỗ lực cần thiết để di chuyển phần mềm từ môi trường này sang môi trường khác là thước đo tính di động. Tính di động ngày càng trở nên quan trọng vì các ứng dụng phải chạy trong nhiều môi trường, chẳng hạn như Windows, Mac và Linux; iOS và Android; và PC, máy tính bảng và điện thoại. Yêu cầu về tính di động của dữ liệu cũng rất quan trọng.

+ **Reusability**

  Khả năng tái sử dụng cho thấy công sức cần thiết để chuyển đổi một thành phần phần mềm để sử dụng trong các ứng dụng khác. Phần mềm có thể tái sử dụng phải là mô-đun, được ghi chép đầy đủ, độc lập với một ứng dụng và môi trường hoạt động cụ thể và có khả năng hơi chung chung. Nhiều thành phần của dự án có tiềm năng tái sử dụng, bao gồm

các yêu cầu, kiến trúc, thiết kế, mã, thử nghiệm, quy tắc kinh doanh, mô hình dữ liệu, mô tả lớp người dùng, hồ sơ các bên liên quan và thuật ngữ. Các đặc điểm kỹ thuật kỹ lưỡng về các yêu cầu và thiết kế, tuân thủ nghiêm ngặt các tiêu chuẩn mã hóa, bộ trường hợp kiểm thử hồi quy được duy trì và thư viện tiêu chuẩn duy trì các thành phần có thể tái sử dụng tạo điều kiện tái sử dụng phần mềm

+ **Scalability**

  Các yêu cầu về khả năng mở rộng giải quyết nhấn mạnh về khả năng phát triển của ứng dụng để phù hợp với nhiều người dùng, dữ liệu, máy chủ, vị trí địa lý, giao dịch, lưu lượng mạng, tìm kiếm và các dịch vụ khác mà không ảnh hưởng đến hiệu suất hoặc tính chính xác. Khả năng mở rộng có ý nghĩa cho cả phần cứng và phần mềm. Mở rộng quy mô hệ thống có thể có nghĩa là mua máy tính nhanh hơn, thêm bộ nhớ hoặc dung lượng đĩa, thêm máy chủ, phản chiếu cơ sở dữ liệu hoặc tăng dung lượng mạng. Các phương pháp tiếp cận phần mềm có thể bao gồm phân phối tính toán lên nhiều bộ xử lý, nén dữ liệu, tối ưu hóa thuật toán và các kỹ thuật điều chỉnh hiệu suất khác. Khả năng mở rộng có liên quan đến khả năng sửa đổi và tính mạnh mẽ.

+ **Verifiability**

  Hay còn được gọi là khả năng kiểm thử, khả năng xác minh đề cập đến việc các thành phần phần mềm hoặc sản phẩm tích hợp có thể được đánh giá tốt như thế nào để chứng minh liệu hệ thống có hoạt động như mong đợi hay không. Thiết kế có khả năng xác minh cao là rất quan trọng nếu sản phẩm có các thuật toán và logic phức tạp, hoặc nếu nó chứa các mối quan hệ tương tác chức năng tinh tế. Khả năng xác minh cũng rất quan trọng nếu sản phẩm sẽ được sửa đổi thường xuyên, bởi vì nó sẽ trải qua kiểm tra hồi quy thường xuyên để xác định xem các thay đổi có làm hỏng bất kỳ chức năng hiện có nào hay không. Các hệ thống có khả năng xác minh cao có thể được kiểm tra cả hiệu quả và hiệu quả. Thiết kế phần mềm để xác minh có nghĩa là giúp dễ dàng đặt phần mềm vào trạng thái tiền kiểm mong muốn, cung cấp dữ liệu
  kiểm tra cần thiết và quan sát kết quả kiểm tra.