# **Phần 1: Lý thuyết**

1. **Tổng quan về công nghệ phần mềm**

* **Cho biết 3 thách thức của công nghệ phần mềm, giải thích tại sao.**

**Thiếu Nhân Lực Phát Triển:** Nhu cầu về các nhà phát triển phần mềm có kỹ năng thường vượt quá nguồn nhân lực có sẵn, dẫn đến thách thức trong việc tuyển dụng và giữ chân các chuyên gia có trình độ.

**Thời Gian Để Xây Dựng Ứng Dụng:** Cuộc khảo sát của Reveal đã phát hiện rằng gần một nửa (43,4%) thời gian của một nhà phát triển được dành cho việc lập trình một ứng dụng. Cách tốt nhất để giảm thời gian lập trình là tích hợp các nền tảng phát triển low-code/no-code giúp các nhóm phát triển tạo ra các ứng dụng bằng các công cụ hình ảnh đòi hỏi ít hoặc không cần mã lập trình trước.

**Bắt Kịp Sự Đổi Mới Trong Công Cụ Phát Triển:** Các phương pháp mới, ngôn ngữ lập trình, công cụ và công nghệ đòi hỏi sự chú ý của các nhà phát triển. Tuy nhiên, việc áp dụng một ngôn ngữ/công nghệ/công cụ mới mỗi khi có sự đổi mới trong không gian phát triển phần mềm đặt ra một thách thức.

* Nếu bạn chuyển sang sử dụng quá sớm, bạn có thể nhanh chóng nhận ra rằng sự đổi mới không đáp ứng được kỳ vọng của bạn, và việc thất bại này có thể làm trở ngại cho các dự án của bạn.
* Nếu bạn chờ quá lâu, bạn có thể thấy mình bị tụt lại so với đối thủ của mình, những người đã thành thạo với các đổi mới mới.

**Khó Khăn Trong Việc Tích Hợp Bên Thứ Ba:**Tích hợp API của bên thứ ba thường cần thiết để làm cho sản phẩm của bạn trở nên hữu ích và hấp dẫn hơn đối với người dùng của bạn.Tích hợp mang lại giá trị lớn vì chúng tiết kiệm thời gian và tiền bạc phát triển nội bộ, nhưng bạn cần đảm bảo rằng API hoạt động như mong đợi và không ảnh hưởng đến các chức năng hiện có của sản phẩm của bạn.

**Mối Đe Dọa An Ninh**: Lạm dụng API, chẳng hạn như việc phá vỡ và mất dữ liệu, có thể gây tổn thương nghiêm trọng đến uy tín của một công ty, chưa kể đến những hậu quả có thể gây ra cho khách hàng. Các phương pháp xâm nhập dữ liệu ngày càng phức tạp, điều này có nghĩa là tích hợp kém chất lượng có rủi ro rất cao. Giữ một kết nối an toàn với hệ thống khác đòi hỏi kiểm soát và cải thiện liên tục. Bạn nên đánh giá ứng dụng bên thứ ba mà bạn muốn tích hợp vào sản phẩm của mình. Hãy đảm bảo rằng API

**Kỳ Vọng Tăng Cao Của Khách Hàng:** Khách hàng trực tiếp chịu ảnh hưởng từ sự tiến triển và thành công của một dự án, đó là lý do tại sao các nhà phát triển và các quản lý sản phẩm của họ phải đảm bảo rằng kỳ vọng của khách hàng được đáp ứng.

Một phần tư số người tham gia khảo sát của Reveal (26,2%) gặp khó khăn trong việc quản lý khối lượng công việc của họ, và 26% cảm thấy rằng kỳ vọng của khách hàng quá cao.

Trong môi trường áp lực cao này, các nhà phát triển phải tìm cách hoàn thành công việc một cách hiệu quả mà vẫn đảm bảo sự hài lòng của khách hàng.

Các công cụ tăng năng suất làm việc tại nơi làm việc cho phép minh bạch, sự hiển thị và sự hợp tác giữa các nhà phát triển và khách hàng, đồng bộ tất cả mọi người về tiến độ, thời hạn và ngân sách.

**Quản Lý Khối Lượng Công Việc và Dự Án:** Nhiều người vẫn tiếp tục làm việc từ xa hoặc trong môi trường lai hóa trong thời gian dài. Các khó khăn mà họ gặp phải trong việc quản lý khối lượng công việc và dự án có thể được khắc phục bằng một công cụ tăng năng suất làm việc tại nơi làm việc kỹ thuật số.

Các nền tảng này là một phần không thể thiếu của quá trình biến đổi kỹ thuật số thành công vì chúng cung cấp cho các nhóm phân tán các công cụ để hoàn thành các nhiệm vụ của họ bằng cách quản lý khối lượng công việc và dự án một cách hiệu quả và hiệu quả. Các công cụ tăng năng suất làm việc tại nơi làm việc kỹ thuật số tích hợp quản lý dự án và nội dung, chat, và thậm chí cả phân tích dữ liệu trong một nền tảng phần mềm có thể tăng năng suất, cải thiện giao tiếp, thúc đẩy sáng tạo và thúc đẩy sự phát triển kinh doanh.

Cuộc khảo sát về các thách thức trong phát triển phần mềm đã phát hiện ra rằng hơn một nửa (54,4%) các nhà phát triển và các nhà chuyên môn IT muốn sử dụng một công cụ nơi mọi người có thể cùng nhau làm việc và giải quyết vấn đề. 47,5% còn muốn tự động hóa các luồng công việc và quy trình, và 43,7% ưa thích loại bỏ việc chia sẻ tập tin bằng cách thủ công.

* **Để đánh giá một phần mềm, những yếu tố nào có trong tiêu chuẩn đánh giá phần mềm? Nêu lý do tại sao?**

-Maintainability (duy trì) : Phần mềm nên được thiết viết theo cách sau cho nó có thể phát triển để đáp ứng nhu cầu thay đổi của khách hàng. Đây là một thuộc tính quan trọng vì thay đổi phầm mềm là yêu cầu tất yếu của môi trường kinh doanh đang thay đổi

**Lý do:** Môi trường kinh doanh thay đổi liên tục, yêu cầu phần mềm cũng thay đổi theo. Nếu phần mềm không thể tiến hóa để đáp ứng những thay đổi này, nó sẽ trở nên lỗi thời và không hiệu quả. Bằng cách duy trì phần mềm, người phát triển có thể thêm tính năng mới, sửa lỗi và cập nhật để đáp ứng yêu cầu mới của khách hàng.

-Dependability and security (độ tin cậy và bảo mật): độ tin cậy của phần mềm bao gồm một loạt các đặc điểm bao gồm độ tin cậy, bảo mật và an toàn. Phần mềm đáng tin cậy không được gây ra thiệt hại vật chất hoặc kinh tế trong trường hợp hệ thống bị lỗi.Người dùng độc hại sẽ không thể truy cập hoặc làm hỏng hệ thống.

**Lý do**: Người dùng mong muốn sự tin cậy và an toàn khi sử dụng phần mềm. Nếu một phần mềm không đáng tin cậy, có thể dẫn đến sự cố nghiêm trọng và thậm chí làm hại đến người dùng hoặc doanh nghiệp. Bảo mật là một yếu tố quan trọng, đặc biệt trong các ứng dụng chứa thông tin nhạy cảm như thông tin tài khoản ngân hàng hoặc thông tin cá nhân.

-Efficiency (tính hiệu quả): Phầm mềm không được sử dụng lãng phí tài nguyên hệ thống như bộ nhớ và chu trình xử lý. Do đó, hiệu quả bao gồm khả năng đáp ứng, thời gian xử lý, sử dụng bộ nhớ,…

**Lý do**: Sử dụng tài nguyên hệ thống một cách hiệu quả giúp tăng hiệu suất và giảm chi phí vận hành. Phần mềm hiệu quả cũng cung cấp trải nghiệm người dùng tốt hơn, với thời gian phản hồi nhanh và ít lỗi hơn. Nếu phần mềm sử dụng quá nhiều tài nguyên, nó có thể làm chậm hệ thống hoặc tăng chi phí vận hành, gây phiền toái cho người dùng.

Acceptability (Khả năng chấp nhận được): Phần mềm phải được chấp nhận đối với loại người dùng mà nó được thiết kế. Điều này có nghĩa là nó phải dễ hiểu, sử dụng được và tương thích với các hệ thống khác mà họ sử dụng

**Lý do**: Phần mềm chỉ có ý nghĩa khi được chấp nhận và sử dụng bởi người dùng. Nếu nó không dễ sử dụng, không tương thích hoặc không đáp ứng được nhu cầu của người dùng, họ có thể từ chối sử dụng và tìm kiếm các giải pháp khác. Khả năng chấp nhận được đảm bảo rằng phần mềm đáp ứng được mong đợi của người dùng và tạo ra một trải nghiệm tích cực.

* **Tính Hiệu quả của phần mềm, là một trong 4 thuộc tính thiết yếu để đánh giá một phần mềm, có nghĩa là gì?**

**Như câu trên**

* **Tại sao hệ thống phần mềm có thể bị lỗi theo thời gian?**

**A diagram of a curve

Description automatically generated**

+ Công nghệ thay đổi : Các thay đổi trong công nghệ có thể dẫn đến việc phần mềm sử dụng các công nghệ lỗi thời hoặc có lỗ hổng bảo mật. Khi phải sửa chữa các lỗ hổng này, có thể xuất hiện thêm các lỗi mới hoặc mở ra các lỗ hổng khác. Việc thường xuyên cập nhật và duy trì bảo mật của hệ thống là cần thiết để giảm thiểu rủi ro.

+ Về các phương pháp tấn công : Hacker và các kẻ tấn công mạng ngày càng tinh vi và sáng tạo trong việc tìm ra các lỗ hổng và khai thác chúng.

+ Sự mở rộng và thay đổi: Khi hệ thống mở rộng hoặc thay đổi, như thêm tính năng mới, có thể tạo ra các lỗ hổng hoặc lỗi trong mã nguồn hiện tại. Việc thử nghiệm và kiểm tra kỹ lưỡng trước khi triển khai các thay đổi là quan trọng để tránh sự cố.

+ Thay đổi trong môi trường: Sự thay đổi trong môi trường hoạt động của phần mềm, như cập nhật hệ điều hành hoặc phần cứng, có thể gây ra các lỗi không mong muốn.

+ Không tương thích: Các cập nhật phần mềm, thay đổi hệ thống hoặc cài đặt mới có thể không tương thích với các phần mềm hoặc cấu hình hiện có, gây ra lỗi hoặc xung đột.

**Để phần mềm không bị lỗi theo thời gian thì chúng ta cần phải làm gì (ít nhất 2 công việc)? Tại sao?**

+ Kiểm thử và kiểm tra chất lượng: Thực hiện các bước kiểm thử và kiểm tra chất lượng phần mềm một cách kỹ lưỡng trước khi triển khai là cực kỳ quan trọng. Kiểm thử sẽ giúp đảm bảo rằng phần mềm hoạt động như mong đợi và giảm thiểu các lỗi xuất hiện sau này trong quá trình sử dụng.

+ Duy trì và nâng cấp định kỳ: Duy trì phần mềm định kỳ, bao gồm việc cập nhật, sửa lỗi và nâng cấp, là cực kỳ quan trọng để đảm bảo rằng phần mềm luôn duy trì được hiệu suất và độ tin cậy cao trong môi trường hoạt động. Việc duy trì định kỳ giúp ngăn chặn việc tích tụ lỗi và cải thiện tính ổn định của hệ thống phần mềm theo thời gian.

* **Cho biết bốn tầng của Công nghệ phần mềm. Nhiệm vụ của mỗi tầng trong CNPM**

**A diagram of a diagram of a diagram

Description automatically generated**

* A”quality” focus : đảm bảo rằng spham phần mềm đáp ứng được các tiêu chí chất lượng và yêu cầu của người dùng đồng thời tuân thủ các tiêu chuẩn và chất lượng
* Process : xác định và thực hiện các quy trình làm việc hiệu quả nhằm đảm bảo sự đồng nhất và sự hiệu quả trong quá trình phát triển phần mềm ,tự lập kế hoạch đến triển khai.
* Methods : Sử dụng các phương pháp, kỹ thuật, và quy trình tiên tiến nhất để tối ưu hóa quá trình phát triển và tạo ra sản phẩm chất lượng.
* Tools : Cung cấp các công cụ và môi trường phát triển cần thiết để hỗ trợ quá trình phát triển phần mềm, bao gồm các công cụ quản lý dự án, biên soạn mã nguồn, kiểm thử, và quản lý phiên bản.

1. **Tìm hiểu về hoạt động hỗ trợ (Umbrella Activities)**

* **Hoạt động hỗ trợ là gì?**

Một chuỗi các bước quy tắc hay quy định gì đó sẽ cho phép team phát triển phần mềm để duy trì tiến độ dự án, chất lượng sản phẩm,kiểm soát những thay đổi và rủi ro để hoàn thành nhiệm vụ phát triển.

Việc hỗ trợ sẽ phát triển qua các giai đoạn của quá trình phát triển phần mềm. Mỗi giai đoạn, công việc hỗ trợ sẽ khác nhau

Áp dụng cho toàn bộ dự án phần mềm và hỗ trợ nhóm phát triển phần mềm. quản lý và theo dõi tiến độ, chất lượng, những thay đổi và rủi ro.

* **Có những loại hoạt động hỗ trợ nào?**

**Một số loại hoạt động hỗ trợ phổ biến :**

-Project tracking and control : theo dõi và kiểm soát tiến độ của dự án, bao gồm việc theo dõi các công việc, tiến độ hoàn thành, và xử lý các thay đổi.

-Technical reviews: Thực hiện các cuộc đánh giá kỹ thuật để kiểm tra và đảm bảo chất lượng của mã nguồn, thiết kế và tài liệu kỹ thuật.

-Quality Assurance : Thực hiện các hoạt động để đảm bảo chất lượng của sản phẩm phần mềm, bao gồm kiểm tra, kiểm thử và đảm bảo tuân thủ các tiêu chuẩn và quy trình.

-Configure Management : Quản lý và kiểm soát các phiên bản của mã nguồn, tài liệu và các thành phần khác của hệ thống phần mềm.

-Documentation : Tạo và duy trì các tài liệu liên quan đến dự án phần mềm, bao gồm tài liệu yêu cầu, tài liệu thiết kế, tài liệu hướng dẫn sử dụng và tài liệu kỹ thuật.

-Re-usability : Thúc đẩy việc tái sử dụng mã nguồn, thành phần và các phần mềm có sẵn để tối ưu hóa hiệu suất và tiết kiệm thời gian trong quá trình phát triển.

-Measurement : Thu thập và phân tích dữ liệu để đo lường và đánh giá hiệu suất của dự án phần mềm, bao gồm cả tiến độ, chất lượng và năng suất.

-Risk management : Xác định, đánh giá và quản lý các rủi ro có thể ảnh hưởng đến dự án, bao gồm cả việc phát triển các kế hoạch phòng tránh và ứng phó.

* **Những loại hỗ trợ nào cần thiết nhất khi đơn vị thực hiện phát triển một hệ thống lớn mà có thời gian thực hiện ngắn.**

+ Project tracking and control (Theo dõi và kiểm soát dự án): Trong dự án có thời gian ngắn, việc chặt chẽ theo dõi và kiểm soát tiến độ là cực kỳ quan trọng. Điều này giúp đảm bảo rằng các mục tiêu được đặt ra được đáp ứng, và các vấn đề có thể được xử lý kịp thời để đảm bảo dự án không trễ hạn.

+ Quality Assurance (Bảo đảm chất lượng): Trong dự án với thời gian ngắn, việc đảm bảo chất lượng của sản phẩm là quan trọng để tránh việc phát hiện các lỗi sau khi triển khai. Kiểm thử và đánh giá chất lượng phải được thực hiện một cách kỹ lưỡng để đảm bảo tính ổn định và đáng tin cậy của hệ thống.

+ Risk management (Quản lý rủi ro): Trong dự án với thời gian ngắn, các rủi ro có thể có một ảnh hưởng lớn đến tiến độ và chất lượng của dự án. Việc xác định, đánh giá và quản lý các rủi ro sẽ giúp đảm bảo rằng các biện pháp phòng tránh và ứng phó được áp dụng kịp thời để giảm thiểu tác động tiêu cực.

+ Configuration Management (Quản lý cấu hình): Trong một dự án lớn, việc quản lý phiên bản của mã nguồn và các thành phần khác của hệ thống là rất quan trọng. Điều này giúp đảm bảo rằng các thay đổi được theo dõi và kiểm soát chặt chẽ, và không gây ra xung đột hoặc mất mát dữ liệu không mong muốn.

1. **Quy trình phát triển phần mềm**

* **Quy trình phát triển phần mềm là gì**

Quy trình phát triển phần mềm là chuỗi các hoạt động và quy trình thiết kế, phát triển và triển khai phần mềm hiệu quả và có tổ chức. Nó bao gồm các bước cụ thể và quy định về cách thức hoạt động của các nhóm, vai trò và công cụ trong suốt quá trình phát triển. Đây là phần quan trọng của quản lý dự án phần mềm, đảm bảo sản phẩm cuối cùng đáp ứng được yêu cầu của khách hàng và đạt chất lượng mong đợi.

***Quy trình phát triển phần mềm bao gồm các yếu tố như:***

1. Mô hình quy trình: Đây là cách tổ chức và sắp xếp các bước trong quy trình phát triển, ví dụ như mô hình thác nước (waterfall), mô hình xoắn ốc (spiral), hay mô hình agile.
2. Phương pháp thực hiện: Đây là các phương pháp, kỹ thuật được sử dụng để thực hiện các hoạt động trong mỗi giai đoạn của quy trình, như thiết kế hướng đối tượng (object-oriented design), phát triển dựa trên kiểm thử (test-driven development), kiểm tra hình thức (formal inspection),...
3. Vai trò và trách nhiệm của các thành viên: Xác định rõ vai trò, trách nhiệm của từng người tham gia trong dự án, như lập trình viên, kiểm thử viên, kiến ​​trúc sư hệ thống, v.v.
4. Luồng công việc (workflows): Định rõ các bước cần thực hiện từ yêu cầu đến thiết kế, lập trình, kiểm thử và triển khai, giúp quản lý và theo dõi tiến độ của dự án.

* **Cho biết 2 đặc trưng chính của quy trình phát triển phần mềm theo mô hình thác nước (hoặc agile, hoặc RUP, hoặc tăng trưởng, chữ V, …) giải thích để thấy đó là đặc trưng**
* **Cho biết 2 đặc trưng chính của phương pháp thác nước, giải thích lý do**

**Tính cấu trúc rõ ràng (Clear Structure):**

Phương pháp thác nước có một cấu trúc được xác định rõ ràng và đơn giản. Các giai đoạn (như thiết kế, phát triển, kiểm thử) được xác định trước và phải được hoàn thành tuần tự.

Lý do: Cấu trúc rõ ràng giúp tạo ra một bức tranh tổng quan về dự án, giúp các bên liên quan hiểu rõ về quy trình và các bước cần thực hiện. Điều này tạo ra sự tổ chức và tiến độ dễ quản lý trong dự án, giúp đảm bảo rằng mọi người đều biết mình cần làm gì và khi nào cần hoàn thành.

**Tính dễ hiểu và triển khai (Ease of Understanding and Implementation):**

Phương pháp thác nước dễ hiểu và triển khai. Mỗi giai đoạn của dự án có mục tiêu cụ thể và các bước cần làm được xác định rõ ràng.

Lý do: Sự rõ ràng và dễ hiểu giúp đảm bảo rằng mọi người trong nhóm dự án đều hiểu rõ về công việc của họ và các bước cần thực hiện. Điều này giúp tránh được sự mơ hồ và hiểu nhầm, từ đó tăng cơ hội thành công của dự án.

* **Cho biết 2 đặc trưng chính của phương pháp RUP, giải thích lý do**

**Tính chất linh hoạt (Flexibility):** RUP không yêu cầu một quy trình cố định mà có thể linh hoạt thích ứng với nhu cầu cụ thể của dự án. Quy trình này có thể được tinh chỉnh và điều chỉnh để phản ánh các yêu cầu và điều kiện cụ thể của dự án, bao gồm kích thước, phức tạp, và môi trường làm việc.

**Sự tập trung vào khả năng thích ứng (Adaptability):** RUP đề cao việc thích ứng với sự thay đổi trong quy trình phát triển phần mềm. Nó cung cấp một khung làm việc có thể điều chỉnh để đáp ứng các yêu cầu thay đổi của dự án và môi trường kinh doanh. Sự tập trung vào việc thích ứng giúp RUP tồn tại trong các dự án phần mềm đa dạng và đòi hỏi sự linh hoạt cao.

* **Cho biết 2 đặc trưng chính của phương pháp tang trưởng, giải thích lý do**

**Tính linh hoạt và sự thích nghi (Flexibility and Adaptability):** Phương pháp tang trưởng linh hoạt và có khả năng thích ứng với biến đổi của thị trường và phản hồi từ người dùng. Điều này cho phép nhóm phát triển thích ứng nhanh chóng với yêu cầu mới và thay đổi trong môi trường kinh doanh, giúp tối ưu hóa hiệu suất và tăng trưởng của sản phẩm hoặc dịch vụ.

**Tập trung vào dữ liệu và thử nghiệm (Data-driven and Experimentation):** Phương pháp này dựa trên việc sử dụng dữ liệu và thử nghiệm để đo lường và tối ưu hóa hiệu suất của các chiến lược tang trưởng. Sử dụng dữ liệu và thử nghiệm giúp đưa ra các quyết định dựa trên số liệu cụ thể và giảm thiểu các giả định, từ đó tăng khả năng thành công của các chiến lược tang trưởng.

* **Cho biết 2 đặc trưng chính của phương pháp chữ V, giải thích lý do?**

**Tính phản hồi (Feedback):** Phương pháp chữ V tập trung vào việc tích hợp phản hồi từ kiểm thử sớm và thường xuyên vào quá trình phát triển, giúp nhóm phát triển nhanh chóng nhận biết và giải quyết các vấn đề, từ đó cải thiện chất lượng sản phẩm và giảm thiểu rủi ro.

**Tính đảm bảo chất lượng (Quality Assurance):** Phương pháp chữ V đảm bảo rằng quá trình kiểm thử và kiểm tra được thực hiện một cách kỹ lưỡng để đảm bảo chất lượng sản phẩm, giúp giảm thiểu lỗi và rủi ro liên quan đến chất lượng sản phẩm, từ đó tăng cơ hội thành công của dự án.

* **Cho biết 2 đặc trưng chính của phương pháp agile, giải thích lý do**

**Tính lặp (Iterative):**

Agile thực hiện dự án trong các phân đoạn lặp lại, được gọi là Iteration hoặc Sprint. Mỗi phân đoạn này thường có thời gian ngắn và nhóm phát triển hoàn thành một phần nhỏ của sản phẩm trong mỗi lần lặp.

Lý do: Tính lặp cho phép nhóm dự án tạo ra các phiên bản phần mềm có thể sử dụng được một cách nhanh chóng, giúp giảm thiểu rủi ro và kiểm soát chất lượng. Bằng cách này, khách hàng có thể xem xét và đánh giá sản phẩm ở mỗi giai đoạn, từ đó có thể cung cấp phản hồi sớm, giúp điều chỉnh hướng phát triển của dự án2.Tính thích ứng (hay thích nghi – adaptive).

**Tính thích ứng (Adaptive):**

Phương pháp Agile linh hoạt và thích ứng với sự thay đổi trong quá trình phát triển. Do các phân đoạn chỉ kéo dài trong thời gian ngắn và lập kế hoạch có thể được điều chỉnh liên tục, nên Agile có khả năng đáp ứng nhanh chóng và linh hoạt với các yêu cầu mới hoặc thay đổi trong dự án.

Lý do: Tính thích ứng cho phép nhóm dự án thích ứng với các yêu cầu mới từ khách hàng hoặc môi trường kinh doanh, giúp đảm bảo rằng sản phẩm được phát triển là phản ánh chính xác những gì khách hàng cần và mong đợi. Điều này giúp tăng cường sự hài lòng của khách hàng và giảm thiểu rủi ro liên quan đến sự thay đổi muộn màng.

* **Mô tả một dự án mà phù hợp triển khai với mô hình tăng trưởng, đặc tả dự án, giải thích lý do**

Một dự án phù hợp với mô hình tăng trưởng có thể là việc phát triển một ứng dụng di động kỹ thuật số cho việc quản lý sức khỏe và tinh thần cá nhân.

Đặc tả dự án: Một công ty công nghệ quyết định phát triển một ứng dụng di động kỹ thuật số để giúp người dùng quản lý sức khỏe và tinh thần cá nhân. Ứng dụng sẽ cung cấp các tính năng như theo dõi hoạt động vận động, giấc ngủ, lịch trình hàng ngày, quản lý căng thẳng và tinh thần, cung cấp gợi ý cho chế độ dinh dưỡng và tập luyện, và cung cấp các công cụ hỗ trợ tâm lý như mindfulness và meditation.

Lý do là:

1. Phản hồi từ người dùng: Mô hình tăng trưởng cho phép ứng dụng điều chỉnh dựa trên phản hồi và yêu cầu từ người dùng, cải thiện trải nghiệm và tính năng của ứng dụng.
2. Linh hoạt trong phát triển: Dự án có thể phát triển theo các giai đoạn nhỏ và linh hoạt, thích ứng với yêu cầu mới và thay đổi từ thị trường một cách nhanh chóng.
3. Đồng bộ hóa chiến lược tiếp thị và phát triển sản phẩm: Mô hình tăng trưởng giúp tối ưu hóa chiến lược tiếp thị và phát triển sản phẩm để tăng cường sự nhận biết và sử dụng ứng dụng từ phía người dùng và thị trường.

* **Mô tả một dự án mà phù hợp triển khai với mô hình RUP, đặc tả dự án, giải thích lý do**

Một dự án phát triển một hệ thống quản lý tài sản cho một tổ chức tài chính là một ví dụ phù hợp để triển khai với mô hình Rational Unified Process (RUP).

Đặc tả dự án: Một tổ chức tài chính quyết định triển khai một hệ thống quản lý tài sản mới để quản lý các tài sản tài chính của họ, bao gồm cổ phiếu, trái phiếu, quỹ đầu tư, và tài sản tài chính khác. Hệ thống sẽ cung cấp các tính năng như quản lý danh mục tài sản, theo dõi vị trí và giá trị của các tài sản, phân tích rủi ro và hiệu suất, và tạo ra các báo cáo và thông báo liên quan đến tài sản.

Lý do mô hình RUP phù hợp:

-Phát triển theo giai đoạn và lặp lại: Mô hình RUP cung cấp cơ chế phát triển theo các giai đoạn rõ ràng và lặp lại, phù hợp với việc tiếp tục cải thiện và điều chỉnh hệ thống theo yêu cầu và phản hồi từ khách hàng và người dùng.

-Tập trung vào chất lượng và kiểm soát rủi ro: Mô hình RUP chú trọng vào việc kiểm soát chất lượng và quản lý rủi ro, quan trọng đối với môi trường tài chính nơi độ chính xác và đáng tin cậy của dữ liệu là rất quan trọng.

-Tính linh hoạt và tùy chỉnh: RUP cho phép dự án linh hoạt thích ứng với các yêu cầu thay đổi, đảm bảo rằng hệ thống phát triển luôn đáp ứng được nhu cầu cụ thể của tổ chức và thị trường.

* **Mô tả một dự án mà phù hợp triển khai với mô hình agile, đặc tả dự án, giải thích lý do**

Dự án: Phát triển một ứng dụng di động dành cho việc quản lý thời gian và công việc cá nhân.

Đặc tả dự án: Một công ty phần mềm quyết định phát triển một ứng dụng di động nhằm giúp người dùng quản lý thời gian và công việc cá nhân một cách hiệu quả. Ứng dụng sẽ có các tính năng như lập kế hoạch công việc, quản lý deadline, ghi chú, nhắc nhở, và thống kê hoạt động. Giao diện sẽ được thiết kế đơn giản và dễ sử dụng, phù hợp với cả người dùng mới và người dùng có kinh nghiệm.

Lý do mô hình Agile phù hợp:

Tính linh hoạt: Mô hình Agile cho phép phát triển ứng dụng theo các chu kỳ ngắn, linh hoạt và lặp lại. Điều này cho phép nhóm phát triển nhanh chóng thích ứng với sự thay đổi trong yêu cầu của khách hàng hoặc thị trường, cung cấp các bản cập nhật và tính năng mới định kỳ.

Tăng tương tác với khách hàng: Agile khuyến khích sự tương tác thường xuyên với khách hàng trong quá trình phát triển. Việc này giúp đảm bảo rằng ứng dụng đáp ứng được mong đợi của người dùng và có thể điều chỉnh nhanh chóng khi cần thiết.

Phản hồi từ người dùng: Khách hàng có thể tham gia đóng góp ý kiến và đề xuất tính năng mới trong quá trình phát triển, giúp sản phẩm cuối cùng đáp ứng được nhu cầu thực tế và mong muốn của người dùng.

Kiểm soát rủi ro: Mô hình Agile cho phép đánh giá và giảm thiểu rủi ro ngay từ các giai đoạn sớm của dự án, giúp nhận diện và giải quyết vấn đề một cách linh hoạt và kịp thời.

* **Quy trình phát triển phần mềm RUP khác với mô hình thác nước chỗ nào? Cho ví dụ một hệ thống phù hợp với quy trình RUP**

|  |  |  |
| --- | --- | --- |
|  | **RUP (Rational Unified Process )** | **Thác nước (Waterfall)** |
| **Tính linh hoạt và tập trung vào quy trình** | Tập trung vào việc thiết kế một quy trình phát triển phần mềm linh hoạt và có thể tùy chỉnh dựa trên nhu cầu cụ thể của dự án | thường áp dụng một quy trình cố định và chuẩn mực cho tất cả các dự án |
| **Sự tập trung vào cung cấp giá trị sớm** | khuyến khích việc tạo ra các phiên bản phần mềm có giá trị sớm và thường xuyên để thu hút phản hồi từ khách hàng và người dùng. | thường đi theo một quy trình tuần tự và chỉ cung cấp sản phẩm cuối cùng vào cuối giai đoạn phát triển |
| **Kiểm soát chi phí và thời gian** | RUP thường có một cấu trúc quản lý dự án chặt chẽ hơn, giúp kiểm soát chi phí và thời gian phát triển. | thường không linh hoạt trong việc điều chỉnh và thay đổi, dẫn đến khả năng kiểm soát chi phí và thời gian không hiệu quả. |

Ví dụ

Một hệ thống phù hợp với quy trình RUP có thể là một ứng dụng di động cho việc đặt vé sự kiện. Trong quá trình phát triển, dự án này có thể đối mặt với nhiều yêu cầu thay đổi từ người dùng, như thêm tính năng đặt vé nhóm hoặc cải thiện giao diện người dùng. Sự linh hoạt của RUP cho phép nhóm nhanh chóng thích ứng và cung cấp các bản phát triển sớm để thu hút phản hồi từ người dùng, giúp sản phẩm luôn được cải thiện và đáp ứng nhu cầu người dùng một cách hiệu quả.

* **Trong Scrum thì chu kỳ Spring thường kéo dài khoảng bao lâu, nhóm thường dưới bao nhiêu người, cuộc họp thường xuyên ngắn trả lời 3 câu hỏi gì?**

Trong Scrum, chu kỳ Sprint thường kéo dài từ 1 đến 4 tuần. Thực tế là không có một con số chính xác cho số lượng Sprint "lý tưởng" trong một dự án, và nó thực sự phụ thuộc vào phạm vi và tỷ lệ phức tạp của dự án.

Số lượng thành viên trong nhóm Nhóm phát triển tốt nhất thường có số lượng thành viên từ 3 đến 9, theo quy định của Scrum. Số lượng thành viên quá ít có thể dẫn đến sản phẩm nhỏ và thiếu các kỹ năng cần thiết, trong khi số lượng quá nhiều có thể làm tăng phức tạp trong quản lý và giao tiếp.

Cuộc họp thường xuyên: Được gọi là Daily Scrum hoặc Daily Standup, thường diễn ra hàng ngày.

Trong cuộc họp này, mỗi thành viên trong nhóm trả lời ba câu hỏi:

a. Hôm qua tôi/bạn đã làm gì để giúp nhóm đạt được mục tiêu Sprint?

b. Bạn/Tôi sẽ làm gì trong ngày hôm nay để giúp nhóm phát triển đạt được mục tiêu Sprint?

c. Có bất kỳ vấn đề hoặc trở ngại nào đang cản trở bạn trong quá trình làm việc không?

* **Quy trình nào có phân tích rủi ro tường minh. Mức độ lớn của dự án khi áp dụng mô hình này?**

Quy trình phát triển phần mềm có phân tích rủi ro tường minh thường được thực hiện trong mô hình Agile như Scrum. Agile có thể áp dụng cho cả dự án nhỏ và lớn, nhưng việc thực hiện Agile trên dự án lớn có thể đòi hỏi các điều chỉnh và chiến lược quản lý rủi ro phù hợp.

1. **Đặc tả yêu cầu (Requirements Specification)**

* **Đặc tả yêu cầu (Requirements Specification) là gì?**

Đặc tả yêu cầu (Requirements Specification) là quá trình viết xuống các yêu cầu của người dùng và hệ thống trong một tài liệu yêu cầu. Yêu cầu của người dùng cần phải được viết sao cho dễ hiểu đối với người dùng cuối và khách hàng không có kiến thức kỹ thuật. Trong khi đó, yêu cầu của hệ thống cần phải chi tiết hơn và có thể bao gồm thông tin kỹ thuật hơn. Các yêu cầu này có thể là một phần của một hợp đồng cho việc phát triển hệ thống, vì vậy quan trọng là chúng cần phải hoàn chỉnh nhất có thể.

* **Có mấy đặc tả yêu cầu? Ai là người sử dụng nó? Người đó sử dụng để làm gì ( vì dụ Các Tester và các Kỹ sư Bảo trì Phần mềm cần sử dụng tài liệu yêu cầu để làm gì?)**

**Có hai loại đặc tả yêu cầu chính:**

Đặc tả yêu cầu của người dùng (User Requirements Specification - URS): Đây là tài liệu mô tả về các tính năng, dịch vụ mà hệ thống cung cấp dưới dạng câu chữ tự nhiên và hình vẽ. Được viết dành cho khách hàng hoặc người sử dụng cuối.

Đặc tả yêu cầu hệ thống (System Requirements Specification - SRS): Đây là tài liệu cấu trúc và chi tiết mô tả về các chức năng, dịch vụ và các ràng buộc vận hành của hệ thống. Thường là một phần của hợp đồng giữa khách hàng và nhà thầu.

***Người đọc đặc tả yêu cầu:***

User requirements :

* Quản lý khách hàng (Client managers): Sử dụng để hiểu và phản hồi về các yêu cầu từ phía người dùng cuối.
* Người dùng cuối hệ thống (System end-users): Sử dụng để hiểu tính năng và dịch vụ của hệ thống để có thể sử dụng nó một cách hiệu quả.
* Kỹ sư của khách hàng (Client engineers): Sử dụng để đảm bảo rằng yêu cầu của khách hàng được hiểu đúng và triển khai đúng cách.
* Quản lý của nhà thầu (Contractor managers): Sử dụng để hiểu và quản lý quá trình triển khai của hệ thống.
* Kiến trúc sư hệ thống (System architects): Sử dụng để hiểu cách hệ thống phải hoạt động để thiết kế kiến trúc phù hợp.

System requirements:

* Người dùng cuối hệ thống (System end-users): Sử dụng để hiểu cách hệ thống phải hoạt động và các ràng buộc vận hành của nó.
* Kỹ sư của khách hàng (Client engineers): Sử dụng để đảm bảo rằng hệ thống được triển khai đúng như yêu cầu của khách hàng.
* Kiến trúc sư hệ thống (System architects): Sử dụng để thiết kế và triển khai hệ thống theo các yêu cầu cụ thể.
* Software developers : Để xây dựng phần mềm dựa trên các yêu cầu cụ thể.

Các Tester: Các Tester sử dụng tài liệu yêu cầu (URS và SRS) để hiểu các tính năng và chức năng của hệ thống cần được kiểm tra. Họ sử dụng URS để đảm bảo rằng các tính năng được phát triển đáp ứng đúng yêu cầu của người dùng. SRS giúp họ hiểu các yêu cầu kỹ thuật và chi tiết hơn để lên kế hoạch và thực hiện các bước kiểm thử.

Các Kỹ sư Bảo trì Phần mềm: Các Kỹ sư Bảo trì Phần mềm sử dụng tài liệu yêu cầu (đặc biệt là SRS) để hiểu cách hệ thống được thiết kế và triển khai. Điều này giúp họ nắm rõ cấu trúc của hệ thống để có thể thực hiện bảo trì, cập nhật và sửa chữa các lỗi sau khi hệ thống đã được triển khai vào sản xuất.

* **Yêu cầu chức năng và phi chức năng khác nhau chỗ nào? Có yêu câu phi chức năng nào mà tạo ra yêu cầu chức năng của hệ thống**

1. **Yêu cầu Chức Năng (Functional Requirements):**
   * Mô tả các chức năng hoặc dịch vụ cụ thể mà hệ thống cần cung cấp.
   * Tập trung vào việc mô tả cách hệ thống phản ứng với các đầu vào cụ thể và cách hoạt động trong các tình huống cụ thể.
   * Có thể bao gồm cả những gì hệ thống nên làm và những gì không nên làm.
   * Tùy thuộc vào loại phần mềm, người dùng dự kiến và loại hệ thống mà phần mềm được sử dụng.
   * Mô tả các dịch vụ của hệ thống chi tiết.
2. **Yêu cầu Phi Chức Năng (Non-functional Requirements):**
   * Định nghĩa các ràng buộc về thuộc tính hoặc hạn chế của hệ thống, chẳng hạn như độ tin cậy, thời gian phản hồi, và yêu cầu lưu trữ.
   * Thường áp dụng cho toàn bộ hệ thống thay vì từng tính năng hoặc dịch vụ riêng lẻ.
   * Có thể áp dụng các yêu cầu về quy trình, như yêu cầu về một công cụ phát triển cụ thể hoặc một phương pháp phát triển.
   * Có thể quan trọng hơn so với yêu cầu chức năng, vì nếu không đáp ứng được, hệ thống có thể trở nên không hữu ích.
   * Có thể ảnh hưởng đến kiến trúc tổng thể của hệ thống.

Một yêu cầu phi chức năng có thể tạo ra các yêu cầu chức năng liên quan bằng cách xác định các dịch vụ hệ thống cần thiết, và cũng có thể tạo ra các yêu cầu hạn chế các yêu cầu chức năng hiện có bằng cách ràng buộc chúng.

***Một ví dụ cụ thể về yêu cầu phi chức năng tạo ra yêu cầu chức năng là:***

Yêu cầu Phi Chức Năng: Hệ thống phải bảo mật và ngăn chặn truy cập không hợp lệ.

Yêu cầu Chức Năng tạo ra từ yêu cầu phi chức năng: Hệ thống phải cung cấp tính năng xác thực người dùng trước khi cho phép họ tạo mới hoặc xem thông tin đơn hàng.

* **Cấu trúc của một tài liệu yêu cầu gồm những nội dung gì? Trong những nội dung đó có những một vài nội dung nào quan trọng hơn những nội dung khác đúng không? Nêu lý do tại sao**

**Cấu trúc của một tài liệu yêu cầu**

Preface (mở đầu)

Introduction (giới thiệu)

Glossary (thuật ngữ)

User requirements definition (xác định yêu cầu người dùng)

System architecture (kiến trúc hệ thống)

System requirements specification (Đặc tả yêu cầu hệ thống)

System models (mô hình hệ thống)

System evolution (tiến hóa hệ thống)

Appendices (phụ luc)

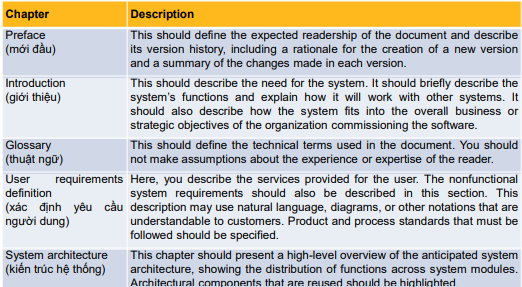
Index (chỉ mục)

**Trong số các phần này, một số phần quan trọng hơn những phần khác, bao gồm:**

**User requirements definition:** Bạn mô tả các dịch vụ (services) được cung cấp cho người dùng. Các yêu cầu hệ thống phi chức năng (nonfunctional system requirements) cũng cần được mô tả trong phần này. Mô tả này có thể sử dụng ngôn ngữ tự nhiên, sơ đồ hoặc các ký hiệu khác mà khách hàng có thể hiểu được. Cần phải xác định rõ các tiêu chuẩn về sản phẩm và quy trình phải tuân theo.

**System requirements specification:** Vì nó mô tả chi tiết các yêu cầu chức năng và phi chức năng của hệ thống. Giao tiếp với các hệ thống khác có thể được xác định.

**System architecture:** Vì nó trình bày tổng quan cấp cao về kiến trúc hệ thống dự kiến, hiển thị sự phân bổ chức năng trên các mô-đun hệ thống . Các thành phần kiến trúc được tái sử dụng cần được làm nổi bật



A yellow and orange text on a white background

Description automatically generated

* **Hãy đưa ra ít nhất 5 gợi ý (tips) để viết một tài liệu đặc tả tốt, nêu lý do tại sao.**

**Hiểu rõ yêu cầu của khách hàng và người dùng cuối:** đảm bảo hiểu rõ những gì khách hàng và người dùng cuối mong đợi từ hệ thống. Điều này giúp đảm bảo rằng tất cả các yêu cầu được phản ánh chính xác và đáp ứng nhu cầu thực tế của họ.

**Sử dụng ngôn ngữ rõ ràng và dễ hiểu:** Viết tài liệu một cách rõ ràng và dễ hiểu giúp tránh hiểu lầm và giúp các bên liên quan dễ dàng hiểu và đồng ý với yêu cầu. Sử dụng từ ngữ đơn giản, tránh các thuật ngữ kỹ thuật phức tạp có thể làm cho tài liệu trở nên khó hiểu.

**Thiết lập một cấu trúc rõ ràng và logic:** Sắp xếp tài liệu theo một cấu trúc logic giúp người đọc dễ dàng theo dõi và hiểu thông tin.

**Đảm bảo sự đầy đủ và chi tiết:** Mỗi yêu cầu cần phải được mô tả đầy đủ và chi tiết, bao gồm mọi thông tin cần thiết để phát triển và kiểm tra tính đúng đắn của hệ thống. Thiếu sót hoặc mơ hồ trong tài liệu yêu cầu có thể dẫn đến hiểu lầm hoặc thiếu sót trong việc triển khai hệ thống.

**Xác thực và phản hồi:** Xác thực và thu thập phản hồi để đảm bảo rằng tất cả các yêu cầu được hiểu và chấp nhận. Phản hồi từ các bên liên quan cung cấp cơ hội để điều chỉnh và cải thiện tài liệu trước khi triển khai hệ thống.

* **Phê chuẩn yêu cầu (Requirement Validation) là gì? Tại sao cần phải có việc phê chuẩn yêu cầu?**

Phê chuẩn yêu cầu (Requirement Validation) là quá trình xác nhận và đảm bảo rằng các yêu cầu đã được thu thập và mô tả đúng đắn và đầy đủ theo nhu cầu thực tế của khách hàng và người sử dụng cuối. Trong quá trình này, các bên liên quan đánh giá và chấp nhận các yêu cầu được đề xuất trước khi bắt đầu công việc phát triển.

***Việc phê chuẩn yêu cầu là cần thiết vì:***

Xác nhận sự hiểu biết chính xác của yêu cầu: Đảm bảo rằng tất cả các bên liên quan hiểu và đồng ý với các yêu cầu được mô tả trong tài liệu. Việc này giúp tránh hiểu lầm và bất đồng quan điểm sau này.

Giảm thiểu rủi ro và chi phí sau này: Chi phí sữa những yêu cầu sai rất nhiều và tốn thời gian gấp 100 lần so với việc làm lại từ đầu cho nên việc phê chuẩn rất quan trọng

Tăng cơ hội thành công cho dự án: Việc có các yêu cầu được phê chuẩn giúp tăng khả năng thành công của dự án bằng cách đảm bảo rằng hệ thống được phát triển đáp ứng đúng nhu cầu và mong đợi của khách hàng và người sử dụng cuối.

Tạo nền tảng cho việc kiểm soát và quản lý yêu cầu: Khi các yêu cầu đã được phê chuẩn, chúng trở thành căn cứ cho việc kiểm soát và quản lý trong quá trình phát triển và triển khai hệ thống.

* **Trình bày các phương pháp để phê chuẩn yêu cầu (Requirement Validation)**

Requirements Reviews: Tổ chức các phiên đánh giá định kỳ để xem xét và đánh giá yêu cầu. Cả nhân viên của khách hàng và nhà thầu tham gia vào đánh giá này. Có thể tổ chức đánh giá chính thức hoặc không chính thức.

Prototyping:Tạo ra một phiên bản sơ bộ của hệ thống để kiểm tra và minh họa yêu cầu. Việc này giúp đảm bảo rằng yêu cầu được hiểu đúng và có thể thực hiện được.

Test-case Generation:Phát triển các trường hợp kiểm thử dựa trên yêu cầu để kiểm tra tính kiểm soát được và tính toàn vẹn của hệ thống.

* **Ai là người có thể thực hiện công việc phê chuẩn yêu cầu? Tại sao**

Khách hàng/Cơ quan quản lý dự án: Khách hàng chính hoặc cơ quan quản lý dự án có thể thực hiện phê chuẩn yêu cầu để đảm bảo rằng hệ thống được phát triển đáp ứng đúng nhu cầu và mong muốn của họ.

Nhóm Phát triển (Developers): Nhóm phát triển, bao gồm các nhà phát triển phần mềm, kiểm thử viên và những người liên quan khác, có trách nhiệm đảm bảo rằng các yêu cầu đã được xác nhận và hiểu đúng trước khi bắt đầu quá trình phát triển.

Người Dùng Cuối (End Users): Người dùng cuối của hệ thống, người sẽ sử dụng và tương tác trực tiếp với nó, cũng có thể tham gia vào việc phê chuẩn yêu cầu để đảm bảo rằng hệ thống đáp ứng đúng nhu cầu và mong muốn của họ.

* **Tại sao phải quản lý việc thay đổi yêu cầu (Requirements Change Management)? Những dự án nào nên có quản lý thay đổi yêu cầu và những dự án nào không cần? Nêu lý do tại sao**

***Lý do tại sao việc quản lý thay đổi yêu cầu là cần thiết:***

Đảm bảo tính đáng tin cậy của hệ thống: Việc quản lý thay đổi yêu cầu giúp đảm bảo rằng các thay đổi được thực hiện một cách có hệ thống và được kiểm soát. Điều này giúp tránh các thay đổi ngẫu nhiên hoặc không kiểm soát có thể ảnh hưởng đến tính đáng tin cậy của hệ thống.

Đáp ứng nhu cầu thay đổi của khách hàng: Do môi trường kinh doanh thay đổi liên tục, khách hàng có thể có những yêu cầu mới hoặc thay đổi về hệ thống. Quản lý thay đổi yêu cầu giúp đáp ứng những nhu cầu này một cách linh hoạt và hiệu quả.

Tránh rủi ro và xung đột: Việc thay đổi yêu cầu một cách không kiểm soát có thể dẫn đến rủi ro và xung đột trong quá trình phát triển phần mềm. Quản lý thay đổi yêu cầu giúp đánh giá và quản lý những rủi ro này một cách có hệ thống.

Duy trì tính nhất quán và hiệu suất của dự án: Khi có nhiều thay đổi yêu cầu xảy ra, việc không kiểm soát có thể dẫn đến sự mất nhất quán và giảm hiệu suất trong quá trình phát triển. Quản lý thay đổi yêu cầu giúp duy trì sự nhất quán và tiến độ của dự án

***Dự án nào nên có quản lý thay đổi yêu cầu:***

Dự án có tính phức tạp: Các dự án phức tạp thường liên quan đến nhiều bên liên quan và yêu cầu phải thay đổi linh hoạt theo thời gian.

Yêu cầu không rõ ràng từ đầu: Trong trường hợp không thể định rõ yêu cầu từ đầu, việc quản lý thay đổi yêu cầu sẽ giúp dự án thích ứng với sự thay đổi của thông tin và yêu cầu mới.

Khách hàng yêu cầu thay đổi liên tục: Đảm bảo sự linh hoạt và phản hồi nhanh chóng đối với các yêu cầu mới từ khách hàng.

Dự án với chu kỳ phát triển ngắn: quản lý thay đổi yêu cầu giúp đảm bảo rằng các thay đổi được áp dụng một cách hiệu quả và không gây ảnh hưởng đến tiến trình phát triển.

***Dự án nào không cần quản lý thay đổi yêu cầu:***

Dự án có yêu cầu cố định từ đầu: Trong trường hợp các yêu cầu được định rõ và không có dấu hiệu thay đổi lớn, việc quản lý thay đổi yêu cầu có thể không cần thiết.

Dự án với quy mô nhỏ và ít phức tạp: Trong các dự án nhỏ với quy mô đơn giản và yêu cầu rõ ràng, việc thay đổi yêu cầu có thể được quản lý thông qua giao tiếp trực tiếp giữa các bên liên quan mà không cần một quy trình chính thức.

Dự án có yêu cầu được đóng băng sớm: Trong trường hợp yêu cầu được đóng băng sớm và không có dấu hiệu thay đổi.

1. **Mô hình hóa hệ thống (System Modeling)**

* **Mô hình hóa hệ thống (System Modeling) là gì? Tại sao mô hình hóa hệ thống ở nhiều góc độ khác nhau?**

-Là quy trình phát triển các mô hình trừu tượng của một hệ thống, trong đó mỗi mô hình biểu diễn một góc nhìn hoặc phối cảnh khác nhau của hệ thống đó.

-Mỗi mô hình sẽ biểu diễn một góc nhìn về hệ thống đó hoặc một khía cạnh nào đó của hệ thống

-Mô hình hóa hệ thống có nghĩa là biểu diễn một hệ thống bằng cách sử dụng một số loiaj ký hiệu đồ họa, hiện nay hầu như luôn dựa trên các ký hiệu trong UML.

Ví dụ : Muốn mô tả các chức năng có thể có trong hệ thống thì chúng ta có thể mô hình hóa bằng sơ đồ Use case.

- Mục tiêu của các mô hình

* Giúp cho người phân tích hiểu được chức năng của một hệ thống
* Được sử dụng để giao tiếp với khách hang

**Tại sao mô hình hóa ở nhiều góc độ khác nhau :**

**Góc độ Bên Ngoài (External Perspective):**Tập trung vào việc mô hình hóa môi trường hoặc ngữ cảnh xung quanh hệ thống.Mô hình hóa giúp hiểu cách mà hệ thống tương tác và phản ứng với các yếu tố bên ngoài như người dùng, các hệ thống khác, hoặc yêu cầu pháp lý.Điều này giúp đảm bảo rằng hệ thống được phát triển để phù hợp với môi trường và yêu cầu xung quanh.

**Góc độ Tương Tác (Interaction Perspective):**Mô hình hóa các tương tác giữa hệ thống và môi trường của nó hoặc giữa các thành phần bên trong hệ thống.Tập trung vào việc mô tả cách mà thông tin và dữ liệu được trao đổi và xử lý giữa các thành phần của hệ thống hoặc với môi trường bên ngoài.

**Góc độ Cấu Trúc (Structural Perspective):**Tập trung vào việc mô hình hóa cấu trúc tổ chức của hệ thống hoặc cấu trúc của dữ liệu được xử lý bởi hệ thống.Mô hình hóa giúp hiểu cách mà các thành phần của hệ thống tổ chức và tương tác với nhau, cũng như cách mà dữ liệu được tổ chức và lưu trữ.

**Góc độ Hành Vi (Behavioral Perspective):**Tập trung vào việc mô hình hóa hành vi động của hệ thống và cách mà nó phản ứng với các sự kiện.Mô hình hóa giúp hiểu cách mà hệ thống hoạt động trong thực tế và làm thế nào để nó phản ứng với các yêu cầu, sự kiện hoặc thay đổi trong môi trường.

* **Mô hình ngữ cảnh (Context models) của hệ thống là gì? Tại sao phải có mô hình ngữ cảnh?**

**A diagram of a medical system

Description automatically generated**

-Các mô hình ngữ cảnh được sử dụng để minh họa bối cảnh hoạt động của một hệ thống – chúng hiển thị những gì nằm ngoài ranh giới hệ thống.

-Các mối quan tâm về mặt xã hội và tổ chức có thể ảnh hưởng đến quyết định về vị trí xác định ranh giới hệ thống.

***Tại sao phải có mô hình ngữ cảnh:***

Hiểu môi trường hoạt động: Giúp hiểu rõ hơn về môi trường hoạt động của hệ thống và các tương tác với các yếu tố bên ngoài.

Xác định và quản lý rủi ro: Giúp xác định và quản lý rủi ro liên quan đến các yếu tố môi trường và tương tác với chúng.

Xác định yêu cầu rõ ràng: Đưa ra yêu cầu một cách rõ ràng hơn bằng cách nhìn nhận các tương tác với các phần của hệ thống hoặc môi trường xung quanh.

Phản hồi chính xác: Cung cấp thông tin phản hồi chính xác hơn từ môi trường, giúp quyết định và hành động có chất lượng hơn.

Tích hợp và tương tác hệ thống: Hỗ trợ thiết kế và triển khai hệ thống để tích hợp và tương tác tốt hơn với các hệ thống và dịch vụ khác.

Tăng độ tin cậy và bảo mật: Giúp xác định và giảm thiểu rủi ro bảo mật và đảm bảo tính đáng tin cậy của hệ thống trong môi trường xung quanh.

* **Mô hình tương tác của của hệ thống là gì gì? Tại sao phải có mô hình tương tác? Cho ví dụ một mô hình tương tác và cho biết ai có nhu cầu dùng và dùng để làm gì?**

- Mô hình hóa tương tác người dùng rất là quan vì nó hỗ trợ việc nhận diện các yêu cầu người dùng.

˗ Mô hình hóa tương tác của một hệ thống với hệ thống (system-to-system) làm nổi rõ các vấn đề về mặt giao tiếp có thể phát sinh giữa hai hệ thống.

˗ Mô hình hóa tương tác thành phần (component) giúp ta hiểu rõ liệu một cấu trúc hệ thống đưa ra có đáp ứng được các yêu cầu về hiệu năng và độ tin cậy hay không.

˗ Có thể sử dụng biểu đồ use case và biểu đồ tuần tự để mô hình hóa sự tương tác.

***Lí do tại sao cần có mô hình tương tác:***

Nhận diện yêu cầu người dùng: Mô hình tương tác người dùng giúp nhận diện và hiểu rõ các yêu cầu của người dùng đối với hệ thống. Điều này giúp đảm bảo rằng hệ thống được phát triển đáp ứng đúng nhu cầu và mong muốn của người dùng.

Xác định vấn đề giao tiếp: Mô hình tương tác giữa các hệ thống (system-to-system) làm nổi bật các vấn đề về mặt giao tiếp có thể phát sinh giữa hai hệ thống. Điều này giúp phát hiện và giải quyết các vấn đề kỹ thuật hoặc tương tác giữa các hệ thống.

Đảm bảo hiệu năng và độ tin cậy: Mô hình tương tác thành phần giúp đánh giá xem cấu trúc của hệ thống có đáp ứng được các yêu cầu về hiệu năng và độ tin cậy hay không. Điều này giúp tối ưu hóa thiết kế và triển khai của hệ thống.

***Ví dụ:***

Một ví dụ về mô hình tương tác có thể là biểu đồ tuần tự, mô tả các tương tác giữa các thành phần của hệ thống trong một chuỗi thời gian. Dưới đây là một ví dụ về biểu đồ tuần tự đơn giản và người dùng có nhu cầu sử dụng và mục đích sử dụng của họ. Ví dụ về biểu đồ tuần tự: Mô hình tương tác giữa người dùng và hệ thống đăng nhập vào một ứng dụng di động.

***Người có nhu cầu dùng:***

Nhà thiết kế hệ thống: Họ sử dụng mô hình này để hiểu cách người dùng tương tác với hệ thống và thiết kế giao diện người dùng phù hợp.

Nhà phát triển phần mềm: Họ sử dụng mô hình này để hiểu cách các thành phần của hệ thống tương tác với nhau và triển khai logic xử lý tương ứng.

Người quản lý dự án: Họ sử dụng mô hình này để theo dõi tiến độ phát triển và đảm bảo rằng giao diện người dùng đáp ứng đúng yêu cầu của người dùng.

* **Mô hình hành vi của hệ thống là gì? Tại sao phải có mô hình hành vi? Cho ví dụ một mô hình hành vi và cho biết mô hình này ai có nhu cầu dùng và dùng để làm gì?**

- Là các mô hình hành vi động (dynamic behavior) của một hệ thống khi nó đang thực thi.

˗ Chỉ ra cái gì xảy ra hoặc cái gì được giả định là xảy ra khi một hệ thống trả lời một tác động (stimuli) từ môi trường.

˗ Có hai loại tác động được đề cập đến:

* Dữ liệu (Data): Một số dữ liệu đến mà hệ thống phải xử lý.
* Sự kiện (Events): Một số sự kiện xảy ra làm kích hoạt việc xử lý của hệ thốn

***Lý do tại sao cần có mô hình hành vi:***

Dự đoán và kiểm soát hành vi: Mô hình hành vi giúp dự đoán và kiểm soát cách hệ thống phản ứng và xử lý các tác động từ môi trường hoặc từ các thành phần khác trong hệ thống. Điều này giúp đảm bảo rằng hệ thống hoạt động đúng cách trong mọi tình huống.

Hiểu rõ cách hoạt động của hệ thống: Mô hình hành vi giúp hiểu rõ cách các thành phần trong hệ thống tương tác với nhau và với môi trường xung quanh trong quá trình thực thi.

Thiết kế và phát triển: Mô hình hành vi là công cụ hữu ích cho việc thiết kế và phát triển hệ thống, giúp đảm bảo rằng các chức năng và tính năng được triển khai đúng cách và hoạt động như mong đợi.  
***Ví dụ :*** biểu đồ trạng thái (State diagram). Biểu đồ này mô tả các trạng thái khác nhau mà một đối tượng hoặc hệ thống có thể đạt được và các sự kiện hoặc dữ liệu mà có thể làm thay đổi trạng thái của đối tượng hoặc hệ thống. Ví dụ trạng thái đơn hang trên Shoppee.

***Người có nhu cầu dùng :***

**- Nhà thiết kế hệ thống**: Họ có thể sử dụng biểu đồ trạng thái để thiết kế cách mà máy bán hàng tự động phản ứng với các sự kiện khác nhau và điều kiện trong quá trình giao dịch.

**- Nhà phát triển phần mềm**: Họ có thể sử dụng mô hình này để hiểu cách hệ thống phản ứng với các tác động từ người dùng và triển khai logic hoạt động tương ứng trong mã nguồn.

**- Người quản lý dự án**: Họ có thể sử dụng mô hình này để theo dõi quá trình phát triển và đảm bảo rằng hệ thống đáp ứng đúng yêu cầu và mong đợi của khách hàng.

**Phần 2: Bài tập**

1. Cho đặc tả 1 tình huống. Dựa vào tình huống đặc tả, đưa ra:
   1. 4 yêu cầu của người dùng về hệ thống trong tình huống
   2. 4 yêu cầu chức năng của hệ thống trong tình huống **(lưu chỉ rõ thuộc loại nào)**
2. Cho 1 đặc tả về một phần của hệ thống phần mềm nào đó. Dựa vào đặc tả này, đưa ra
   1. Mô hình ngữ cảnh của hệ thống phần mềm
   2. Mô hình hóa một quy trình có trong hệ thống bằng mô hình DFD
   3. Mô hình usecase của của hệ thống phần mềm
   4. Mô hình Class ở mức sơ bộ
3. Cho một đặc tả về 1 tình huống/chức năng trong một hệ thống nào đó. Dựa vào đặc tả này, đưa ra:
   1. Viết lại đặc tả trên theo mẫu đặc tả usecase
   2. Đưa ra mô hình tuần tự (squence diagram)
   3. Đưa ra mô hình hoạt động (activity diagram)
   4. Đưa ra mô hình trạng thái (State diagram)

**Hãy đưa ra ít nhất 5 gợi ý (tips) để viết một tài liệu đặc tả tốt, nêu lý do tại sao.**

**Tại sao phải quản lý việc thay đổi yêu cầu (Requirements Change Management)? Những dự án nào nên có quản lý thay đổi yêu cầu và những dự án nào không cần? Nêu lý do tại sao**

**3/ cuối**