Câu 1: Hãy giải thích mục đích chính của kiểm chứng(verification) và thẩm định (validation) của quy trình phần mềm.

Verification: SP - thiết kế (implementing)

Validation: SP - SRS (thiết kế)

- **Verification.** là quá trình tìm lỗi implementing so bới **thiết kế**. Nói cách khác, đây là quá trình xác định đội ngũ có **làm sản phẩm đúng hay không**.
  - **In:** $\mathbb{D} \to \mathbb{I}$
  - **Out:** $\mathbb{I} \vdash \mathbb{D}$
    Nếu `fail`: Kèm theo counter-eg.
- **Validation.** là quá trình kiểm tra sản phẩm liệu có đúng với **đặc tả yêu cầu**. Nói cách khác, đây là quá trình xác định đội ngũ có **làm đúng sản phẩm hay không**.
  - **In:** $\mathbb{SRS} \to \mathbb{I}$
  - **Out:** $\mathbb{I} \vdash \mathbb{SRS}$

Nếu quá trình verification **không** diễn ra trước validation, sẽ không thể xác định được ngọn nguồn của lỗi. Cụ thể, nếu như validation diễn ra trước, khi thất bại, nguyên nhân không rõ là do lập trình sai thiết kế hay thiết kế sai so với yêu cầu. Trái lại, nếu verification diễn ra trước, khi thất bại, nguyên nhân đã hoàn toàn xác định là do lập trình sai thiết kế.

Có hai phương thức để kiểm tra kết quả có đúng yêu cầu đề ra hay không:

- **Static V&V.** Kiểm thử mà không cần execute code.
- **Dynamic V&V.** Kiểm thử có execute code. Đây cũng chính là **Software Testing**.

Câu 2: Ca sử dụng (use case) là gì? Hãy cho một ví dụ về đặc tả ca sử dụng (dùng template trong bài giảng)

Mỗi ca sử dụng là một kịch bản mô tả cách hệ thống hoạt động trong từng tình huống để đáp ứng yêu cầu của người dùng, đại diện cho các hành động được thực hiện bởi một hoặc nhiều tác nhân trong việc theo đuổi một mục tiêu cụ thể

Các tác nhân tham gia ca sử dụng có thể là người hoặc các hệ thống khác.

Đặc tả ca sử dụng : vẽ bảng ca sử dụng

Câu 3: Hãy giải thích những nguyên lý của phương pháp agile dẫn tới sự phát triển và triển khai phần mềm nhanh chóng. Theo em khi nào không nên áp dụng phương pháp agile vào phát triển một hệ thống phần mềm?

Hợp tác chặt chẽ giữa các thành viên của nhóm: Agile thúc đẩy việc làm việc trong môi trường đội nhóm, với sự hợp tác chặt chẽ giữa các thành viên và khách hàng. Các thành viên của nhóm cần phải được đào tạo đầy đủ và có kỹ năng để làm việc với nhau.

Phản hồi nhanh chóng: Agile khuyến khích việc cung cấp phản hồi nhanh chóng giữa khách hàng và nhóm phát triển phần mềm. Thông qua việc cung cấp các phiên bản sản phẩm thường xuyên và liên tục, khách hàng có thể đánh giá và đưa ra phản hồi để nhóm phát triển có thể điều chỉnh và cải tiến sản phẩm nhanh chóng.

Tập trung vào giá trị: Agile tập trung vào việc cung cấp giá trị cho khách hàng. Nhóm phát triển phần mềm cần phải hiểu rõ nhu cầu của khách hàng và tập trung vào việc cung cấp các tính năng có giá trị nhất cho khách hàng trước tiên.

Điều chỉnh linh hoạt: Agile cho phép nhóm phát triển phần mềm điều chỉnh bất kỳ thay đổi nào trong quá trình phát triển. Nhóm có thể thích nghi với các yêu cầu mới và điều chỉnh sản phẩm của mình để đáp ứng nhu cầu của khách hàng.

Phát triển liên tục: Agile khuyến khích việc phát triển sản phẩm liên tục bằng cách cung cấp các phiên bản sản phẩm thường xuyên và định kỳ. Việc phát triển liên tục giúp giảm thiểu rủi ro và hạn chế các vấn đề phát sinh trong quá trình phát triển.

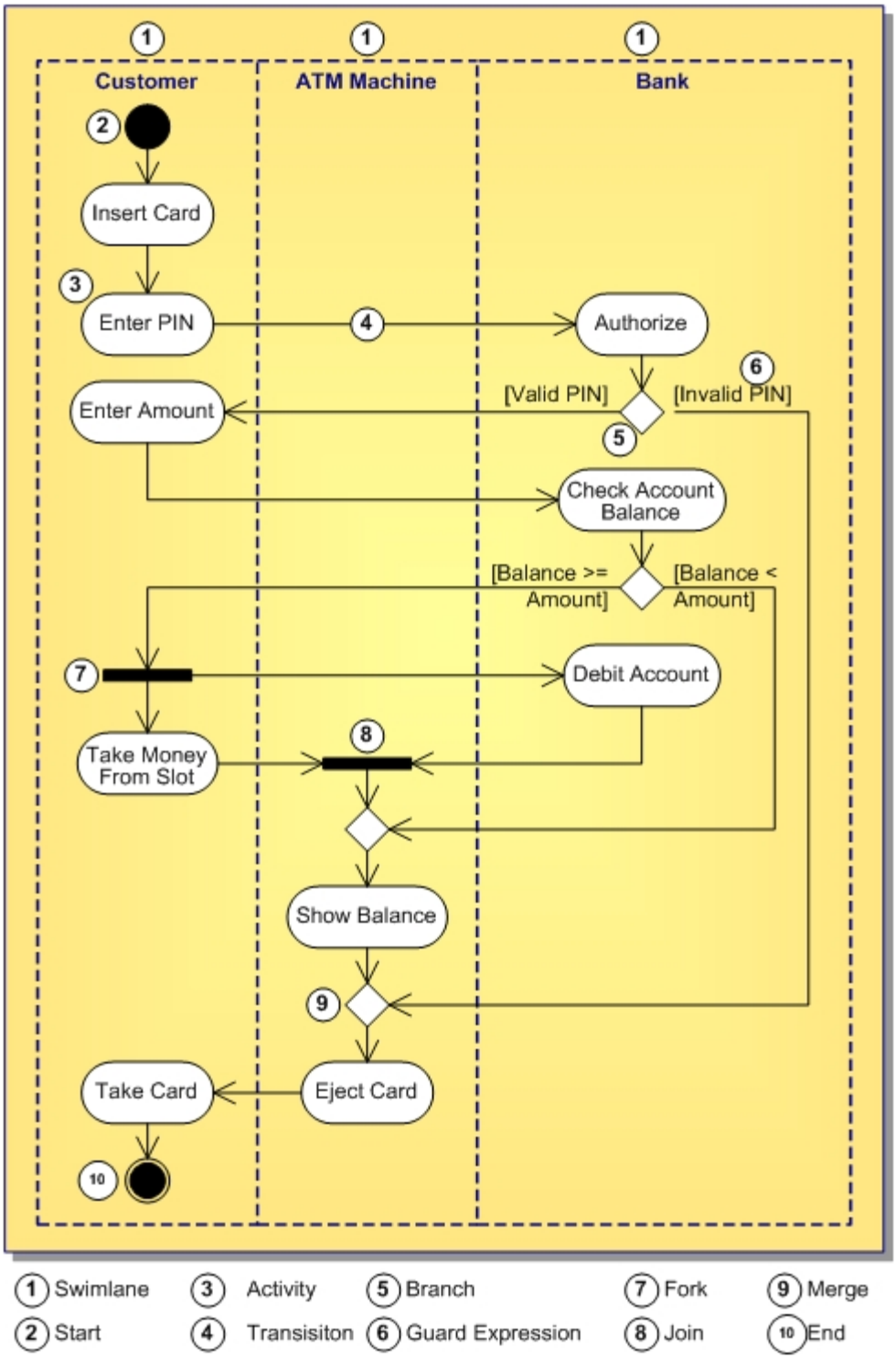
Tính đồng nhất trong phát triển: Agile yêu cầu các thành viên trong nhóm phát triển phần mềm cần phải có mục tiêu chung và đồng nhất trong việc phát triển sản phẩm. Điều này giúp tăng tính hiệu quả của các hoạt động phát triển và đảm bảo rằng sản phẩm đáp ứng được các yêu cầu của khách hàng.

Đánh giá định kỳ: Agile khuyến khích việc đánh giá định kỳ để đảm bảo rằng sản phẩm được phát triển đúng hướng và đáp ứng được nhu cầu của khách hàng. Nhóm phát triển phần mềm cần phải đánh giá tiến độ và chất lượng của sản phẩm định kỳ để đưa ra các điều chỉnh và cải tiến cần thiết.

Câu 4: Dựa vào kinh nghiệm và hiểu biết của em về cây rút tiền ATM, hãy vẽ

- Biểu đồ ca sử dụng
- Sơ đồ hoạt động (activity diagram)
- Biểu đồ tuần tự
- Biểu đồ lớp

biểu diễn việc xử lý dữ liệu cho chức năng rút tiền từ cây ATM



Câu 5: Trình bày những bước cơ bản nhất để làm ra phần mềm, nêu vắn tắt nội dung thực hiện, đầu vào, đầu ra và ý nghĩa

- Lập kế hoạch dự án
    - Thu thập yêu cầu sơ bộ
    - Ước lượng sớm

    - Lập các kế hoạch
- Phân tích và đặc tả yêu cầu (Trả lời câu hỏi What?)
    - Thu thập yêu cầu
    - Phân tích yêu cầu
    - Đặc tả yêu cầu : UML + ngôn ngữ tự nhiên
    - Thẩm định: Đảm bảo các yêu cầu sẽ xác định đúng hệ thống mà khách hàng mong đợi (quan trọng)
    - Quản lý thay đổi yêu cầu (quá trình liên tục trong suốt vòng đời dự án)

Đầu vào: Yêu cầu của khách hàng

Đầu ra: SRS (tài liệu kỹ thuật quan trọng nhất của dự án

Yêu cầu người dùng: viết cho kh, sd ngôn ngữ tự nhiên

Yêu cầu hệ thống: đặc tả chi tiết (nội dung hợp đồng)

Ý nghĩa: xác định các dịch vụ hệ thống mà khách hàng yêu cầu, cùng với các ràng buộc để phát triển và vận hành các dịch vụ đó

- Thiết kế (Trả lời câu hỏi How?)

Đầu vào: SRS

Đầu ra: bản thiết kế kiến trúc

    - Thiết kế mức hệ thống: SA (TK Kiến trúc) và DB (TK database)
    - Thiết kế mức chi tiết: UI/UX (Kịch bản người dùng và màn hình hiển thị) và TK chi tiết

Định nghĩa. "Kiến trúc phần mềm của một hệ thống là tập hợp của các cấu trúc cần thiết để lập luận về hệ thống, bao gồm các software elements, relations, properties of both"

- Software Element: Cấu thành bởi 1 hoặc 1 nhóm UC, hoặc 1 dịch vụ, 1 component.
- Mỗi quan hệ giữa các element
- Tính chất của các mối quan hệ cũng như các element đó.

Tầm quan trọng:

- Hỗ trợ cho giao tiếp giữa các bên liên quan (stakeholders)
- Xác định các ràng buộc cho việc hiện thực hóa
- Dự đoán chất lượng hệ thống
- Nâng cao độ chính xác của việc dự đoán chi phí và thời gian xây dựng hệ thống

Nguyên lý thiết kế:

- Phân tách các khía cạnh quan tâm (Separation of concerns): chia ứng dụng thành các phần càng ít sự chồng chéo về chức năng càng tốt

- Trách nhiệm đơn: Mỗi thành phần chỉ thực hiện một chức năng
- Hiểu biết tối thiểu: Các thành phần không cần biết chi tiết bên trong của các thành phần khác
- Không lặp lại: Mỗi một chức năng chỉ được hiện thực hóa bởi một thành phần
- Hạn chế thiết kế trước: chỉ thiết kế khi cần và có đủ thông tin

Các bước chính.

- Xác định mục tiêu: Phạm vi và thời gian thực hiện
- Xác định các hoạt cảnh sử dụng chính (một hoạt cảnh là sự tổng quát của nhiều ca sử dụng (use case) tương tự)
- Xác định tổng quan về ứng dụng (kiểu ứng dụng, các ràng buộc triển khai, kiểu kiến trúc, công nghệ liên quan
- Xác định các vấn đề chính: Thuộc tính chất lượng: performance, extensibility,… & Khía cạnh quan tâm chung: authentication, authorization, caching, exception handling,…
- Xác định các giải pháp chính: bản thiết kế kiến trúc ở mức cao

Thiết kế UI/UX

Thiết kế chi tiết

- Implementing

  Đầu vào: Thiết kế

  Đầu ra: Sản phẩm

  Nguyên tắc: Dùng lại tối đa. Tại sao?

- Kiểm thử

  Dynamic V&V

- Vận hành và bảo trì

Câu 6: Sử dụng phân tích giá trị biên để sinh các ca kiểm thử

Câu 7: Đặc trưng PM

- Không mòn cũ nhưng thoái hóa theo thời gian
- Không được lắp ráp từ mẫu có sẵn
- Phức tạp, khó hiểu, vô hình
- Luôn luôn thay đổi (thay đổi là bản chất của PM)
- Được phát triển theo nhóm

Câu 8: Đánh giá PM

- Tính năng
    - Tính phù hợp: khả năng có thể cung cấp một tập các chức năng thích hợp cho công việc cụ thể phục vụ mục đích của người sử dụng
    - Tính chính xác: có thể cung cấp các kết quả hay hiệu quả đúng đắn hoặc chấp nhận được với độ chính xác cần thiết
    - Tính an toàn: khả năng bảo vệ thông tin và dữ liệu của sản phẩm phần mềm, sao cho người, hệ thống không được phép thì không thể truy cập, đọc hay chỉnh sửa chúng
    - Tính tương tác (Khả năng hợp tác làm việc): khả năng tương tác với một số HT cụ thể
- Tính tin cậy
    - Tính hoàn thiện: khả năng tránh các kết quả sai
    - Khả năng chịu lỗi: khả năng hoạt động ổn định tại một mức độ cả trong trường hợp có lỗi xảy ra
    - Khả năng phục hồi: có thể tái thiết lại hoạt động tại một mức xác định và khôi phục lại những dữ liệu có liên quan trực tiếp đến lỗi
- Tính khả dụng
    - Dễ hiểu: Users có thể hiểu được xem PM có hợp với họ không và sử dụng thế nào cho những công việc cụ thể
    - Dễ học
    - Có thể sử dụng được
    - Tính hấp dẫn
- Tính hiệu quả
    - Đáp ứng thời gian: khả năng đưa ra kết quả, một thời gian xử lý và một tốc độ thông lượng hợp lý khi nó thực hiện công việc của mình, dưới một điều kiện làm việc xác định
    - Sử dụng tài nguyên: khả năng của phần mềm có thể sử dụng một lượng, một loại tài nguyên hợp lý để thực hiện công việc trong những điều kiện cụ thể
- Khả năng bảo trì
    - Có thể phân tích được: có thể được chẩn đoán để tìm những thiếu sót hay những nguyên nhân gây lỗi hoặc để xác định những phần cần sửa
    - Có thể thay đổi được: có thể chấp nhận một số thay đổi cụ thể trong quá trình triển khai
    - Tính ổn định: khả năng tránh những tác động không mong muốn khi chỉnh sửa phần mềm
    - Có thể kiểm tra được: khả năng cho phép đánh giá được phần mềm chỉnh sửa
- Tính khả chuyển
    - Khả năng thích nghi: có thể thích nghi với nhiều môi trường khác nhau mà không cần phải thay đổi
    - Có thể cài đặt được: phần mềm có thể cài đặt được trên những môi trường cụ thể
    - Khả năng cùng tồn tại: có thể cùng tồn tại với những PM độc lập khác trong một môi trường chung, cùng chia sẻ những tài nguyên chung

- Khả năng thay thế: có thể dùng thay thế cho một phần mềm khác, với cùng mục đích và trong cùng môi trường

Câu 9: Tiến hóa PM?

Câu 10: Em hiểu gì về SE?

Dưới góc nhìn vĩ mô, SE là ngành công nghiệp thực hiện nghiên cứu, phát triển, ứng dụng và chuyển giao các sản phẩm phần mềm. Trong đó, các viện nghiên cứu, trường ĐH nghiên cứu và đưa ra giải pháp; Các công ty thực hiện phát triển, ứng dụng và chuyển giao; Bộ phận R&D là bên trung gian giữa hai bên, cụ thể, họ là những người có khả năng phát biểu bài toán doanh nghiệp đang gặp phải và tìm được đến người có thể giải nó, có được giải pháp và đưa vào sản phẩm, tạo ra giá trị từ giải pháp đó. R&D ở viện NC và doanh nghiệp là không tương đồng. Một bên tập trung vào nghiên cứu khoa học, trong khi một bên tập trung phát triển cách ứng dụng thực tiễn của nghiên cứu và tạo ra sản phẩm phục vụ thị trường.

R&D ở viện NC không chịu áp lực từ thị trường & thương mại, có thể theo đuổi các dự án nghiên cứu dài hạn hoặc rủi ro.

Trái lại, R&D ở doanh nghiệp có nhiều tài nguyên hơn, được tận dụng nguồn lực chuyên môn từ các bộ phận khác cũng như có tài trợ từ công ty mẹ.

.

Dưới góc độ học thuật, SE là một lĩnh vực nghiên cứu về phát triển phần mềm:

SE bao gồm nhiều phương pháp luận và chiến lược để phát triển phần mềm: OOSD (Object-Oriented System Design), SOA (Service-Oriented Architecture), Agile, Waterfall, ...

Từ các CL, PPL, các kỹ thuật cụ thể hình thành và áp dụng vào các pha PTPM: Kỹ thuật phân tích yêu cầu (Requirements analysis), thiết kế (Design), lập trình (Programming), kiểm thử (Testing), triển khai và bảo trì (Deployment and Maintenance), ...

Cuối cùng, nhằm hỗ trợ cho các kỹ thuật PTTM, CASE tool (Computer-Aided Software Engineering) được nghiên cứu và và sử dụng: Công cụ hỗ trợ phân tích và thiết kế (như Enterprise Architect, Rational Rose), quản lý dự án (như Microsoft Project, JIRA), kiểm thử (như Selenium, Appium), ...

Mục tiêu của SE: Bằng cách áp dụng các nguyên lý khoa học và kỹ thuật, ta tạo ra phần mềm chất lượng cao, trong thời gian ngắn với chi phí phù hợp.

Tại sao đường cong lý tưởng được gọi là "lý tưởng"?

Tại sao số lượng lỗi ở đường cong lý tưởng không thể bằng 0?

Do có bàn tay của con người thì không bao giờ có chuyện số lượng lỗi đạt tới con số 0.

Software Engineering là gì?

Đánh giá phần mềm theo tiêu chí nào?

Tại sao **không** sử dụng phương pháp mô hình bản mẫu để phát triển phần mềm?

Mô hình bản mẫu mặc dù tốt khi chưa rõ yêu cầu khách hàng, tuy nhiên bên cạnh mặt lợi cũng có những khiếm khuyết:

- Chi phí để làm bản mẫu không phải nhỏ, và cũng khó biết khi nào là điểm dừng (ngừng làm bản mẫu).
- Yếu tố quan trọng nhất của mô hình bản mẫu là sự tham gia của khách hàng. Tuy nhiên, trên thực tế, việc có được sự tham gia của khách hàng không hề dễ.

Tại sao các mô hình phát triển phần mềm khác cũng có nhược điểm như vậy thì vẫn được sử dụng mà prototype thì không?

Tại sao mô hình thác nước có chi phí lớn?

Tại sao mô hình thác nước chỉ hợp dự án vừa và nhỏ, rõ yêu cầu ngay từ đầu?

Tại sao mô hình thác nước chậm có sản phẩm?

Tại sao mô hình thác nước đảm bảo CLC?

a

Sự khác biệt giữa mô hình thác nước và mô hình chữ V.

Mô hình chữ V về bản chất là cải tiến của mô hình thác nước bằng việc break quy trình Testing ra thành 4 level:

- Unit testing
- Itegration Testing
- System Tesing
- Acceptance Testing

Lý do cho việc cải tiến này là bởi quá trình Testing trong mô hình thác nước diễn ra tương đối muộn, dẫn đến bộ test mang tính chung chung và dễ pass. Trong khi đó, mô hình chữ V sinh test ngay từ trong những giai đoạn tương ứng, và bởi đó bộ test có tính cụ thể hơn, dễ bắt lỗi hơn.

Bài toán đặc tả hình thức là gì, cách xử lý?

Bài toán xử lý xung đột tài nguyên, chuyển thiết kế sang dạng mô hình toán học, các thuộc tính cần chứng minh sang dạng toán học tương ứng và cuối cùng dùng tool để kiểm tra.

Kiến trúc phần mềm là gì? Tầm quan trọng?

Kiến trúc phần mềm là một tập hợp bao gồm 3 thành phần chính:

1. Các element (hay component hoặc service hoàn chỉnh - có thể coi như blackbox), cấu thành bởi 1 hoặc 1 số UC.
2. Mối quan hệ có thể có giữa 2 element (VD như 1 element called from another one)

Tại sao CBSD cho ra chất lượng cao và dễ scale? Nhược điểm của CBSD là gì?

- **CLC:** Do bản thân các component được phát triển cẩn thận độc lập. Do đã được sử dụng phổ biến trong các ứng dụng trong khoảng thời gian dài, đây cũng chính là minh chứng cho chất lượng của các component, bởi ngay cả việc sinh test cũng không đầy đủ bằng.
- **Dễ scale:** Các component độc lập lắp ráp với nhau, dễ thay thế.

Cái khó của component-based software là việc đưa ra quyết định nên chọn service/component nào. Để làm được việc đó, phải có kỹ năng đánh giá chất lượng.

Tại sao gọi là "Implementing" mà không phải "Coding"?

Nguyên tắc của Implementing là gì và tại sao?

Nguyên tắc của Implementing:

- Reuse tối đa.
- Coding là lựa chọn cuối cùng.

Lý

Mối quan hệ giữa Testing và Test?

Phân biệt Verification và Validation? Cái nào trước, và tại sao?

Sự khác nhau giữa Static và Dynamic V&V.