L4N18T1B19DCCN060 - none

information technology (Học viện Công nghệ Bưu chính Viễn thông)

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ PHẦN MỀM

T1. TRẢ LỜI CÂU HỎI

Giảng viên:	**Đỗ Thị Bích Ngọc**
Nhóm lớp:	**4**
Nhóm btl:	**18**
Thành viên trong nhóm:	**Phan Vương Bảo**
	Nguyễn Phi Hùng
	Hà Thị Ngọc Anh
	Nguyễn Việt Anh

Hà Nội, tháng 3 năm 2022

Trả lời câu hỏi

1. Thế nào là corrective maintenance?

Trả lời:

Corrective Maintenance (bảo trì sửa chữa hay bảo trì phục hồi, bảo trì khắc phục) là những hoạt động bảo trì được thực hiện nhằm khôi phục tình trạng của máy móc, thiết bị hay cả dây chuyền sản xuất về các trạng thái hoạt động bình thường.

2. Thế nào là adaptive maintenance?

Trả lời:

Adaptive Maintenance (bảo trì thích ứng) là việc thực hiện các thay đổi trong một phần của hệ thống, đã bị ảnh hưởng bởi sự thay đổi xảy ra trong một số phần khác của hệ thống. Bảo trì thích ứng bao gồm việc phần mềm thích ứng với những thay đổi của môi trường như phần cứng hoặc hệ điều hành . Thuật ngữ môi trường trong ngữ cảnh này dùng để chỉ các điều kiện và ảnh hưởng tác động (từ bên ngoài) lên hệ thống. Ví dụ, các quy tắc kinh doanh, mô hình làm việc và chính sách của chính phủ có tác động đáng kể đến hệ thống phần mềm.

3. Thế nào là perfective maintenance?

Trả lời:

Perfective Maintenance (bảo trì hoàn thiện) chủ yếu giải quyết việc thực hiện các yêu cầu mới hoặc thay đổi của người dùng. Bảo trì hoàn thiện liên quan đến việc thực hiện các cải tiến chức năng cho hệ thống bên cạnh các hoạt động nhằm tăng hiệu suất của hệ thống ngay cả khi các thay đổi không được đề xuất do lỗi. Điều này bao gồm việc nâng cao cả chức năng và hiệu quả của mã và thay đổi các chức năng của hệ thống theo nhu cầu thay đổi của người dùng.

4. Thế nào là refactoring?

Trả lời:

Refactoring là thay đổi ở cấu trúc bên trong mà không làm thay đổi hành vi với bên ngoài của hệ thống.

5. Thế nào là "from scratch"?

Trả lời:

From Scratch là ngay từ đầu, mà không sử dụng bất cứ thứ gì đã tồn tại.

6. Thế nào là moving target problem?

Trả lời:

Moving Target Problem là một thuật ngữ phổbiến để chỉ sự thay đổi trong lĩnh vực chăm sóc sức khỏe có thể khiến những phát hiện về đánh giá kỹ thuật y tế trở nên lỗi thời , đôi khi thậm chí trước khi chúng có thể được thực hiện.

7. Thế nào là regression fault?

Trả lời:

Regression Fault (lỗi hồi quy) là một lỗi đã vô tình thành một phần của sản phẩm do hậu quả của việc tạo ra một sự thay đổi rõ ràng không liên quan đến một phần khác của sản phẩm.

8. Thế nào là một episode?

Trả lời:

Episode là một chu trình hoàn chỉnh từ khâu thiết kế, phân tích đến khâu cài đặt.

9. Thế nào là một iteration?

Trả lời:

Iteration là khi có một phiên bản đầu tiên của vật phẩm, sửa lại và tạo ra các phiên bản khác. Mục đích là các phiên bản tiếp theo sẽ gần với mục tiêu hơn và phiên bản cuối cùng là một phiên bản đạt yêu cầu.

10. Thế nào là một increasement?

Trả lời:

Increasement là một chu kì tăng trưởng từ đánh giá, thiết lập, xây dựng đến chuyển tiếp.

11. Thế nào là một artifact?

Trả lời:

Artifact là một thành phần của một sản phẩm phần mềm, chẳng hạn như tài liệu thông số kỹ thuật, mô-đun mã hoặc hướng dẫn sử dụng.

12. Thế nào là portability?

Trả lời:

Giả sử một sản phẩm P được biên dịch bởi trình biên dịch C và sau đó chạy trên máy tính nguồn, cụ thể là, cấu hình phần cứng H trong hệ điều hành O. Cần một sản phẩm P' có chức năng tương đương với P nhưng phải được biên dịch bởi trình biên dịch C' và chạy trên máy tính đích, cụ thể là cấu hình phần cứng H' theo hệ điều hành O'. Nếu chi phí chuyển đổi P thành P' ít hơn đáng kể so với chi phí mã hóa P' từ đầu, thì P được cho là Portability.

13. Thế nào là reusebility?

Trả lời:

Có hai loại, Opportunistic Reuse and Deliberate Reuse. Nếu các nhà phát triển của một sản phẩm mới nhận ra rằng một thành phần của một sản phẩm được phát triển trước đó có thể được sử dụng lại trong sản phẩm mới, thì đây là Opportunistic Reuse. Mặt khác, việc sử dụng các thành phần phần mềm được xây dựng dành riêng cho việc tái sử dụng trong tương lai gọi là Deliberate Reuse.

14. Thế nào là milestone?

Trả lời:

Milestone là ngày mà thiết kế hoàn thành và vượt qua đánh giá, ngày mà hoàn thành một sản phẩm.

15. Thế nào là một story?

Trả lời:

Story là một tài liêu đơn giản về yêu cầu sản phẩm, có thể của người dùng hoặc nhà sản xuất.

16. Thế nào là refactoring?

Trả lời:

Không có bước thiết kế tổng thể trước khi các bản dựng khác nhau được xây dựng. Thay vào đó, dấu hiệu đã được sửa đổi trong khi sản phẩm đang được xây dựng gọi là Refactoring.

17. Thế nào là concept exploration?

Trả lời:

Điều tra sơ bộ về nhu cầu của khách hàng được gọi là Concept Exploration.

18. Thế nào là business model?

Trả lời:

Business Model là mô tả về các quy trình kinh doanh của khách hàng.

19. Thế nào là traceability?

Trả lời:

Nếu các yêu cầu tạo tác sẽ được kiểm soát trong vòng đời của sản phẩm phần mềm, thì một tài sản mà họ phải có là Traceability.

20. Thế nào là egoless programming?

Trả lời:

Khái niệm cơ bản bên dưới the democratic team là egoless programming. Weinberg chỉ ra rằng các lập trình viên có thể được gắn mã cao vào mã của họ.

21. Thế nào là PM?

Trả lời:

PM là thực hiện phần mềm trong một môi trường mô phỏng hay thực tế bằng cách sử dụng các yếu tố đầu vào được lựa chọn theo cách nào đó.

22. Thế nào là technical leader?

Trả lời:

Technical Leader là người đứng đầu về kỹ thuật và phụ trách các vấn đề liên quan đến kỹ thuật.

23. Thế nào là programming secretary?

Trả lời:

Programming Secretary là một thành viên trung tâm có tay nghề cao, giàu có, trung tâm của một đội ngũ Lập trình viên trưởng. Programming Secretary chịu trách nhiệm duy trì thư viện sản xuất dự án, tài liệu của dự án.

24. Thế nào là backup programmer?

Trả lời:

Backup programer là những lập trình viên dự bị dùng để thay thế những lập trình viên trưởng, khi lập trình viên trưởng bị ốm, tai nạn hoặc thay đổi việc. Ngoài ra, Backup programer còn có năng lực như lập trình viên trưởng và hiểu biết nhiều về các dự án.

25. Thế nào là super programmer?

Trả lời:
Còn được gọi là lập trình viên hoặc nhà phát triển phần mềm, programmer là một cá nhân viết mã hoặc tạo phần mềm để kiếm sống. Trong khi mỗi công việc khác nhau, hầu hết các programmer chịu trách nhiệm tạo ra các chương trình phần mềm hoặc các phần của một chương trình, gỡ lỗi các vấn đề, hoặc thêm vào một chương trình.

26. Thế nào là một bản thiết kế còn ommision?

Trả lời:
Bản thiết kế còn ommision là bản thiết kế còn nhiều sự mập mờ, thiếu rõ ràng.

27. Thế nào là một bản thiết kế còn contradiction?

Trả lời:
Bản thiết kế còn contradiction là bản thiết kế có sự mâu thuẫn dẫn đến phần mềm sau này có thể không hoạt động được ở chức năng đó. Ví dụ : Theo thiết kế của 1 hệ thống, van M17 sẽ được đóng khi áp suất vượt quá 35 psi. Tuy nhiên có một trạng thái khác được thiết kế là nếu áp suất vượt quá 35 psi thì còi sẽ báo động trong 30s sau đó van M17 mới tự động đóng. Do đó dẫn đến mâu thuẫn khiến hệ thống không chạy được chức năng đó.

28. Thế nào là một phần mềm COTS?

Trả lời:
Là sản phẩm phần mềm được nhà phát triển sản xuất bán với với giá thành thấp, đáp ứng nhu cầu của rất nhiều người, thu lại lợi nhuận nhờ việc bán ra với số lượng lớn (vd : Microsoft..).

29. Thế nào là SPMP?

Trả lời:

- SPMP là viết tắt của Software Project Management Plan – Bản kế hoạch quản lí dự án phần mềm.
- SPMP được tạo ra ở pha phân tích, ghi rõ mô hình vòng đời sử dụng, cấu trúc tổ chức của đội phát triển, nhiệm vụ của dự án, phản ánh từng workflow riêng biệt của tiến trình phát triển, phân công việc của mỗi người trong đội và deadline hoàn thành, các CASE tools và kĩ thuật được sử dụng, thời gian biểu chi tiết, ngân sách, phân bổ tài nguyên, ước lượng thời gian và chi phí.

30. Thế nào là alpha release?

Trả lời:
Alpha release là phiên bản được kiểm thử hoạt động chức năng thực tế hoặc giả lập do một số ít người dùng/khách hàng được chỉ định hoặc một nhóm test thực hiện tại nơi sản xuất phần mềm. Alpha release được thực hiện ngay sau kiểm thử hệ thống (Product Testing). Alpha release thường áp dụng cho các sản phẩm COTS (MS Office, Windows,..) là một hình thức kiểm thử chấp nhận nội bộ, trước khi phần mềm được tiến hành kiểm thử beta.

31. Thế nào là beta release?

Trả lời:

Được thực hiện sau alpha release, được phát hành tới một số nhóm khách hàng bên ngoài nhóm phát triển phần mềm để tăng phạm vi phản hồi từ người sử dụng tương lai lớn nhất. Beta release gần như phiên bản cuối cùng (final version).

32. Thế nào là process?

Trả lời:

- Process(Tiến trình):Là phương cách sản xuất ra phần mềm.
- Process khác với workflow là:
    + Bao gồm nhiều quy trình trong việc tạo ra 1 phần mềm + Có thể có nhiều mô hình khác nhau trong 1 process.
    + Process có thể sử dụng mô hình workflow để làm ra phần mềm.

33. Thế nào là workflow?

Trả lời:

- Định nghĩa đơn giản nhất của workflow: là các định nghĩa của các qui trình đã chuẩn hóa. Và khi mình viết các module cho từng công việc, workflow là 1 chuỗi công việc phải làm.
- Là 1 cách thực hiện cụ thể của process.
- Quá trình (process) :
    + bao gồm nhiều quy trình trong việc tạo ra 1 phần mềm
    + có thể có nhiều mô hình khác nhau trong 1 process
    + Process có thể sử dụng mô hình workflow để làm ra phần mềm

34. Luật Miller trong CNPM nói gì?

Trả lời:

Luật Miller : Tại một thời điểm bất kì, một người chỉ có thể có khả năng tập trung vào khoảng 7 chunk (đơn vị thông tin).

35. Luật Brooks trong CNPM nói gì?

Trả lời:

Luật Brooks : Thêm một người vào trong một dự án đã bị trễ sẽ làm cho nó trở nên chậm trễ hơn.

36. Luật Dijkstra trong CNPM nói gì?

Trả lời:

Luật Dijkstra : Rất dễ để chứng minh một phần mềm có lỗi, nhưng không thể chứng minh được một phần mềm không có lỗi.

37. Verification và Validation (V&V) là gì?

Trả lời:

- Verification & Validation:
    + Thẩm định là việc xác định xem luồng công việc (workflow) đã được thực hiện đúng hay chưa, và việc thẩm định sẽ được thực hiện ở cuối mỗi Workflow.
    + Xác minh là 1 quy trình đánh giá chuyên sâu được thực hiện trước khi đưa sản phẩm tới tay khách hàng.

- Mục đích của việc này là xác định xem sản phẩm đã thỏa mãn hoàn toàn bản đặc tả chưa.

38. Thế nào là inspection?

Trả lời:

- Inspection: Team Inspection thường gồm 4 người. Vd cho Inspect design gồm có: Moderator, designer, implementer, tester. Là 1 quy trình gồm 5 bước:
    + (Overview)Nhìn lại tổng quan toàn bộ tài liệu điều tra (lấy yêu cầu, đặc tả, thiết kế, code hoặc kế hoạch), tài liệu được phát cho các thành viên.
    + (Preparation) các thành viên hiểu chi tiết tài liệu. Liệt kê danh sách các loại lỗi, sắp xếp theo tần suất xuất hiện. Việc này giúp mọi người tập trung vùng có nhiều lỗi.
    + 1 người sẽ lướt qua tài liệu, và các thành viên điều tra sẽ xác định lỗi mà không sửa chúng. Và moderator của inspection team trong 1 ngày sẽ có báo cáo, bảo đảm quy trình được thực hiện đúng.
    + (Rework) các thành viên sửa toàn bộ lỗi được chỉ ra trong báo cáo.
    + (Follow Up) Moderator phải kiểm tra toàn bộ, bảo đảm không còn lỗi trong báo cáo và không có lỗi phát sinh. Nếu 5% tài liệu trên phải rework (bước 4) lại thì team phải điều tra lại (reinspection).

39. Thế nào walkthrough?

Trả lời:

- Walkthrough: Gồm có 4-6 người, 1 người viết đặc tả (specification), 1 người quản lý workflow phân tích, đại diện khách hàng,1 người thuộc team thiết kế, 1 đại diện thuộc SQA (leader). Có 2 kiểu walkthrough : Document Driven và participants Driven.
    + Document Driven: 1 người (presenter) sẽ kiểm tra toàn bộ nội dung, và những người còn lại có trách nhiệm bình luận, ngắt quãng những chỗ nghi ngờ có lỗi. (cách hiệu quả hơn để tìm lỗi)
    + Participant Driven: mỗi thành viên chuẩn bị 1 danh sách gồm 2 cột: các chi tiết chưa rõ ràng hoặc chi tiết nghi ngờ dính lỗi. Đại diện team phân tích phải phân tích được đâu là lỗi và làm rõ cho người hỏi.

40. Thế nào là một moderator trong nhóm inspection?

Trả lời:

- Người điều tiết (moderator) là người lãnh đạo thanh tra và họ chịu trách nhiệm lập kế hoạch kiểm tra cũng như phối hợp nó. Hơn nữa, người điều hành đảm bảo rằng mặt hàng được kiểm tra đã đáp ứng các tiêu chí nhập cảnh để sẵn sàng kiểm tra.

41. Thế nào là một recorder trong nhóm inspection?

Trả lời:

- Người ghi chép (recorder) là người tham gia chịu trách nhiệm ghi lại các khiếm khuyết được tìm thấy trong quá trình kiểm tra. Mục đích chính của họ là cung cấp nhật ký các khiếm khuyết tiềm ẩn trong cuộc họp thanh tra.

42. Mô hình CMM là gì?

Trả lời:

- CMM (capability maturity models) : Là các chiến lược có thể sử dụng để cải thiện quy trình phần mềm mà không phụ thuộc vào bất cứ một mô hình vòng đời phần mềm (life-cycle model) nào cả.
- CMM được chia thành các loại sau:
    + CMMs for software (SW–CMM)
    + Management of human resources (P–CMM; the P stands for “people”)
    + Systems engineering (SE–CMM)
    + Integrated product development (IPD–CMM)
    + Software acquisition (SA–CMM).

43. Thế nào là test performance?

Trả lời:

- Một kỹ thuật kiểm tra phi chức năng được thực hiện để xác định các thông số hệ thống về khả năng đáp ứng và ổn định theo khối lượng công việc khác nhau. Kiểm tra hiệu suất đo lường các thuộc tính chất lượng của hệ thống, chẳng hạn như khả năng mở rộng, độ tin cậy và sử dụng tài nguyên.

44. Thế nào là test robustness?

Trả lời:

- Là bất kỳ phương pháp đảm bảo chất lượng nào tập trung vào việc kiểm tra độ bền của phần mềm. Kiểm tra độ bền cũng đã được sử dụng để mô tả quá trình xác minh độ bền (tức là tính chính xác) của các trường hợp thử nghiệm trong quá trình thử nghiệm.
- ANSI và IEEE đã định nghĩa độ mạnh mẽ là mức độ mà một hệ thống hoặc thành phần có thể hoạt động chính xác khi có đầu vào không hợp lệ hoặc điều kiện môi trường căng thẳng.

45. Thế nào là coin of uncertainty?

Trả lời:

Trong quản lý dự án, Cone of uncertainty mô tả sự tiến hóa của số lượng sự không chắc chắn trường hợp tốt nhất trong một dự án. Khi bắt đầu một dự án, tương đối ít được biết về sản phẩm hoặc kết quả công việc, và do đó ước tính phải chịu sự không chắc chắn lớn.

46. Thế nào là norminal effort?

Trả lời:

Nominal effort là nỗ lực quản lý và kỹ thuật trong việc phát triển một sản phẩm phần mềm có quy mô nhất định trong dự án lý tưởng. 1 dự án lý tưởng là một dự án giả định “lý tưởng” trong một môi trường nhất định của một tổ chức (hoặc đơn vị kinh doanh). Nó là một dự án chạy dưới sự tối ưu điều kiện, nghĩa là, một dự án mà tất cả các đặc điểm môi trường có tác động đến nỗ lực của dự án ở mức “tốt nhất” (“hoàn hảo”) khi bắt đầu dự án. Lưu ý rằng "tốt nhất" đề cập đến các cấp độ thực tế có thể có trong một bối cảnh; không đến mức tốt nhất có thể tưởng tượng

47. Thế nào là phần mềm organic?
Trả lời:
Organic – Một dự án phần mềm có quy mô nhóm cần thiết là đủ nhỏ, vấn đề được hiểu rõ và đã được giải quyết trong quá khứ và các thành viên trong nhóm cũng có kinh nghiệm về vấn đề này.
48. Thế nào là phần mềm embeded?
Trả lời:
Phần mềm nhúng hay embedded software là lập trình chuyên biệt trong các thiết bị không phải PC, là một phần của ứng dụng khác nằm trên chip để điều khiển các chức năng cụ thể của thiết bị. Không giống như phần mềm ứng dụng, có thể được cài đặt trên nhiều hệ thống máy tính và được sửa đổi để cung cấp các mức chức năng khác nhau, phần mềm nhúng có các yêu cầu và khả năng phần cứng cố định. Nó được tạo riêng cho thiết bị cụ thể mà nó chạy, với các hạn chế về xử lý và bộ nhớ gắn trực tiếp với thông số kỹ thuật của thiết bị đó.
49. Thế nào là phần mềm semi-detached?
Trả lời:
Phần mềm semi-detached là 1 kiểu phần mềm tích hợp mà độ khó của các yêu cầu ở giữa organic và embedded. 1 phần mềm tích hợp được làm bởi 1 đội bao gồm cả người có kinh nghiệm và không có kinh nghiệm.
50. Thế nào là TCF?
Trả lời:
TCF là 1 đơn vị cho biết độ phức tạp kỹ thuật của phần mềm.
51. Thế nào là UFP?
Trả lời:
UFP là 1 đơn vị đo lường cho biết kích thước chức năng của phần mềm chưa được điều chỉnh.
52. Thế nào là flow trong FFT?
Trả lời:
Flow là giao diện dữ liệu giữa sản phẩm và môi trường như màn hình, báo cáo,…
53. Thế nào là process trong FFP?
Trả lời:
Process là 1 chức năng được định nghĩa logic hoặc các thao tác số học liên quan đến dữ liệu như sắp xếp, xác thực, cập nhật,…
54. Tại sao không có pha kiểm thử?
Trả lời:
Vì chương trình đã được kiểm tra tại cuối mỗi workflow và khi hoàn thành sản phẩm rồi. Nếu không kiểm thử tính đúng đắn của phần mềm ở từng giai đoạn mà chỉ kiểm ở pha kiểm thử và phát hiện ra lỗi, thì thường sẽ tốn thời gian để sửa lỗi và bàn giao sản phẩm không đúng hạn.
55. Tại sao không có pha làm tài liệu?

Trả lời:
Vì mọi pha phải đc viết tài liệu trc khi bắt đầu 1 pha mới nếu không tài liệu bị hoãn lại thì sẽ không bao giờ hoàn thành và cá nhân chịu trách nhiệm trong pha trước có thể chuyển sang bộ phận khác, bên cạnh đó sản phẩm thường xuyên thay đổi khi phát triển nên cần k có 1 pha ghi tài liệu để cố định.
56. Tại sao không có pha lập kế hoạch?
Trả lời:
Vì chúng ta k thể lập kế hoạch vào đầu dự án do chúng ta chưa biết chính xác những gì mà chúng ta sẽ xây dựng. Chúng ta chỉ có thể lập kế hoạch sơ bộ cho pha yêu cầu và pha phân tích khi bắt đầu mỗi dự án. Kế hoạch quản lý dự án phần mềm chỉ đc đưa ra khi các chi tiết kỹ thuật mà khách hàng đưa ra đã được hoàn tất Vì chỉ có bản kế hoạch tạm thời về quản lý dự án, mọi kế hoạch chỉ là ước lượng, quá trình có thể bị thay đổi do nhiều tác nhân khác nhau trong lúc thực hiện dự án.
57. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềmkhông? Tại sao?
Trả lời:
Nếu không áp dụng các mô hình vòng đời thì rất khó để phát triển phần mềm vì:
+ Khó kiểm soát đc phần mềm(nhiều lỗi tiềm ẩn, khả năng gắn kết các module kém) .
+ Khả năng tái sử dụng module kém .
+ Chi phí để sản xuất 1 phần mềm cao.
58. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?
Trả lời:
Vì các mô hình vòng đời có các ưu điểm, nhược điểm khác nhau phù hợp với những điều kiện phát triển phần mềm khác nhau. Quy mô của các dự án phát triển phần mềm khác nhau nên đòi hỏi các mô hình vòng đời khác nhau phù hợp với kinh phí phát triển dự án đó.
59. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?
Trả lời:
- Ưu điểm:
+ Dễ phân công công việc, phân bố chi phí, giám sát công việc.’
+ Kiến trúc hệ thống hàng đợi ổn định.
+ tài liệu được cung cấp ở mỗi giai đoạn và yêu cầu rằng tất cả các sản phẩm của mỗi giai đoạn (bao gồm cả tài liệu) phải được kiểm tra tỉ mỉ bởi bộ phận SQA.
- Nhược điểm:
+ Mối quan hệ giữa các giai đoạn không được thể hiện - Hệ thống phải được kết thúc ở từng giai đoạn do vậy rất khó thực hiện được đầy đủ những yêu cầu của khách hàng... vì trong mô hình này rất khó khăn trong việc thay đổi các pha đã được thực hiện thì chỉ còn cách là phải thực hiện lại từ đầu.

60. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?
Trả lời:
- Mô hình thác nước chỉ nên sử dụng khi mà đội dự án đã có kinh nghiệm làm việc, bởi mô hình này đòi hỏi sự chính xác ngay từ đầu.
- Hợp với những dự án mà khách hàng xác định được yêu cầu cụ thể, chính xác ngay từ đầu và ít có khả năng thay đổi .
- Nên áp dụng với những dự án có phạm vi cố định và chi phí cố định.
- Không gấp rút thời gian.

61. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?
Trả lời:
- Phần mềm phát triển theo dạng tuyến tính, tiến hành từ làm bản mẫu nhanh đến khi sản phẩm được giao; vòng lặp gần như không cần thiết trong mô hình này. Đây cũng là ưu điểm và cũng là nhược điểm của mô hình này.
- Ưu điểm : nhanh,tốc độ phát triển sản phẩm diễn ra nhanh chóng
- Nhược điểm: thành viên trong nhóm phát triển sử dụng bản mẫu nhanh k tham khảo ý kiến khách hàng => có thể gây nên sai sót, không đúng yêu cầu khách hàng, gây tổn thất.

62. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?
Trả lời:
Thiết kế nhanh chỉ tập trung vào việc biểu diễn các khía cạnh của phần mềm thấy được đối với người dùng, phù hợp với những dự án đã thỏa thuận xong xuôi, xác định được yêu cầu cụ thể, chính xác ngay từ đầu và ít có khả năng thay đổi, có phạm vi hoạt động cố định và giá cố định.

63. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?
Trả lời:
- Ưu điểm:
  + Mỗi giai đoạn khách hàng có được sản phẩm thực hiện 1 phần công việc theo yêu cầu. Ngay từ phân phối ban đầu khách hàng đã có thể thấy được sự lợi ích của sản phẩm
  + Giảm thiểu được chi phí bảo trì
  + Các phần được thực hiện nhanh, tính bằng tuần lễ
  + Giảm shock tâm lý khi dùng sản phẩm mới hoàn toàn, Dần dần giúp khách hàng có thời gian thích nghi với sản phẩm
  + Không đòi hỏi khách hàng có kinh phí lớn, có thể dừng phát triển bất cứ lúc nào
- Khó khăn:
  + Khó khi kết hợp build vào cấu trúc hiện thời là làm sao không phá hủy cấu trúc đã xây dựng. Như vậy, yêu cầu cấu trúc phải mở.

+ Mặc dù uyển chuyển hơn các mô hình như thác nước và bản mẫu vì dễ thay đổi theo ý kiến khách hàng, nhưng có thể làm suy biến thành mô hình bản mẫu và khó điều phối chung.

64. Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Mô hình này đã được sử dụng hiệu quả cho một số phần mềm vừa và nhỏ,có dung lượng công việc từ 9 người-tháng cho đến 100 người-năm. Mô hình này thực sự có lợi khi yêu cầu của khách hàng không rõ ràng và hay thay đổi.

65. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu xoắn ốc?

Trả lời:

- Ưu điểm :

+ Phân tích rủi ro dự án được đầy lên làm một phần thiết yếu trong quy trình xoắn ốc để tăng độ tin cậy của dự án

+ Xây dựng dự án có sự kết hợp các mô hình khác vào phát triển (Thác nước, mô hình mẫu…)

+ Cho phép thay đổi tuỳ theo yêu cầu cho mỗi vòng xoắn ốc

+ Nó được xem như là một mô hình tổng hợp của các mô hình khác. Không chỉ áp dụng cho phần mềm mà còn phải cho cả phần cứng

+ Một rủi ro nào đó không được giải quyết thì chấm dứt dự án

+ Các vòng tròn được lặp để đáp ưng được những thay đổi của người dùng

+ Kiểm soát rủi ro ở từng giai đoạn phát triển.

+ Đánh giá tri phí chính xác hơn các phương pháp khác

- Nhược điểm:

+ Phức tạp và không thích hợp với các dự án nhỏ.

+ Cần có kỹ năng tốt về phân tích rủi ro.

+ Yêu cầu thay đổi thường xuyên dẫn đến lặp vô hạn

+ Chưa được dùng rộng dãi như mô hình thác nước.

+ Đòi hỏi năng lực quản lý.

66. Mô hình vòng đời phần mềm kiểu xoắn ốc thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Hợp với hệ thống lớn có thể phân chia thành nhiều thành phần và nhiều rủi ro

67. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu tiến trình linh hoạt?

Trả lời:

- Ưu điểm:

+Tận dụng tối đa hiệu quả của việc lập trình nhóm, lập trình cặp đôi.

+ Giảm thiểu rủi ro đến mức tối thiểu, cung cấp cho khách hàng sản phẩm mà vừa lòng họ nhất.

+ Khả năng ứng phó một cách linh hoạt, nhanh chóng trước những thay đổi yêu cầu của khách hàng ở bất kỳ giai đoạn nào của dự án

+ Khả năng cung cấp phiên bản release của phần mềm nhanh chóng ngay khi khách hàng cần đến.

- Nhược điểm:

+ Khó có thể áp dụng trên một đề án có số lượng nhân viên khổng lồ. + Không có hiệu quả đối với những khách hàng ở xa đội ngũ phát triển.

68. Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Hợp với các phần mềm mà yêu cầu của khách hàng không rõ rang và chỉ vận hành được với các dự án có quy mô nhỏ.

69. Trong mô hình tiến trình liên hoạt, luôn có đại diện của khác hàng trong nhóm phát triển thì có ưu điểm gì?

Trả lời:

Trong mô hình tiến trình linh hoạt luôn có đại diện khách hàng trong nhóm phát triển thì ưu điểm là khách hàng và nhóm phát triển liên tục trao đổi trực tiếp giúp cho những chi tiêu khách hàng đặt ra được đáp ứng tốt nhất, những sự không hài lòng được sửa chữa ngay lập tức, giúp giảm thời gian phát triển và nâng cao chất lượng phần mềm.

70. Nêu ưu điểm, nhược điểm của mô hình nhóm code bình đẳng?

Trả lời:

- Ưu điểm :

+ Do mô hình này 1 người luôn khuyến khích người khác tìm ra lỗi của mình nên việc tìm ra lỗi trong phần mềm sẽ nhanh hơn, chất lượng phần mềm được cải thiện.

+ Phù hợp với xử lý các vấn đề khó

- Nhược điểm :

+ Những người có kinh nghiệm thường không thoải mái khi để cho những người mới làm kiểm tra code của mình.

+ Do không có sự cạnh tranh, thăng tiến nên thường thích hợp ở những môi trường nghiên cứu

+ Không thể bị áp đặt từ bên ngoài.

71. Mô hình nhóm code bình đẳng thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Mô hình nhóm code bình đẳng thì phù hợp với những dự án kiểu nghiên cứu, giải quyết những vấn đề khó do tính bình đẳng và hỗ trợ hết sức, không có cạnh tranh, thăng tiến, không có leader thực sự trong nhóm

72. Nêu ưu điểm, nhược điểm của mô hình nhóm code có chef?

Trả lời:

- Ưu điểm : Cần ít nhân lực (khoảng 6 người), mỗi người có 1 công việc được phân công rõ ràng => Giảm chi phí, thời gian, đảm bảo chất lượng sản phẩm.

- Nhược điểm :

+ Không thực tế do lập trình viên chính là người phải vừa giỏi việc lập trình, vừa giỏi công việc quản lí do vậy rất khó để kiếm được người có khả năng như vậy.

+ Lập trình viên dự bị là người cũng phải có khả năng như người chính tuy nhiên lại phải ở vị trí dự bị, do đó cũng rất khó kiếm ra người như vậy.

+ Thư kí lập trình cũng khó tìm vì những lập trình viên thường rất dị ứng với công việc giấy tờ, do đó để tìm người lập trình chỉ đơn thuần làm công việc giấy tờ là rất khó.

73. Mô hình nhóm code có chef thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Mô hình nhóm code có chef phù hợp với những dự án nghiên cứu hoặc trong những trường hợp có những vấn đề khó cần sự giải quyết hợp lực của 1 nhóm tương tác lẫn nhau để đưa ra giải pháp.

74. Nêu ưu điểm, nhược điểm của kĩ thuật pair programming?

Trả lời:

- Ưu điểm:
  + Khi một thành viên rời nhóm, thì thành viên mới dễ dàng gia nhập vì có thể học với người cùng cặp.
  + Phát huy ưu điểm của lập trình bình đẳng.
  + Các lập trình viên có thể học hỏi kinh nghiệm lẫn nhau. Khi 1 lập trình viên gặp 1 vấn đề thì người còn lại có thể hỗ trợ giải quyết vấn đề đó.
- Nhược điểm:
  + Đòi hỏi trình độ của 2 thành viên.
  + Cần có kỹ năng teamwork tốt, để tránh xung đột khi làm việc.

75. Kĩ thuật pair programming thì phù hợp với những dự án có đặc điểm gì?

Trả lời:

Kỹ thuật programming phù hợp với những dự án có task phức tạp, khi fixing.

76. Nêu ưu điểm, nhược điểm của kĩ thuật time boxing?

Trả lời:

- Ưu điểm:
  + Giúp lập trình viên tập trung cao và có phân phối thời gian làm việc hiệu quả
  + Quản lí tốt rủi ro và độ phức tạp.
  + Thúc đẩy năng xuất và hiệu quả của quá trình phát triển phần mềm.
- Nhược điểm: time boxing yêu cầu thời gian cố định nên nếu không hoàn thành công việc đúng thời gian đề ra thì sẽ ảnh hưởng đến tiến độ công việc vì vậy thích hợp với dự án nhỏ.

77. Nêu ưu điểm, nhược điểm của kĩ thuật stand up meeting?

Trả lời:

- Ưu điểm:
  + Không cần set up phòng, chỗ ngồi, tiết kiệm thời gian
  + Biết được tình trạng các thành viên trong đội sau mỗi ngày.

- Nhược điểm:
  + Thích hợp với những dự án nhỏ.

78. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC?
Trả lời:
- Ưu điểm: dễ tính toán.
- Nhược điểm: ngôn ngữ khác nhau có độ dài mã lệnh khác nhau. Không phải mọi thứ viết ra đều chuyển giao cho khách hàng. Chưa thể hiện rõ công sức của lập trình viên và độ khó của dự án.

79. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP?
Trả lời:
- Ưu điểm: Đi sâu hơn vào cấu trúc, chức năng hệ thống
- Nhược điểm:
  + Phụ thuộc vào hằng số b
  + Kích thước các file khác nhau nên vẫn chưa hoàn toàn chính xác.

80. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point?
Trả lời:
- Ưu điểm: chính xác hơn FFP và LOC.
- Nhược điểm: cần thêm nhiều người có kinh nghiệm làm.

81. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO?
Trả lời:
- Ưu điểm: độ chính xác cao hơn
- Nhược điểm: phức tạp

82. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả?
Trả lời:
Tiến trình linh hoạt (Agile Process) không có pha đặc tả do pha phân tích trong tiến trình linh hoạt không được chú trọng. Việc phần mềm chạy được quan trọng hơn là những tài liệu chi tiết. Đồng thời luôn có đại diện khách hàng trong nhóm phát triển nên những tiêu chí của khách hàng sẽ được trao đổi, thực hiện, thay đổi trong lúc cài đặt.

83. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo?
Trả lời:
Trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo vì sản phẩm của workflow này là đầu vào của workflow tiếp theo nên nếu test workflow này không phát hiện ra lỗi nhưng sản phẩm ấy không dùng được trong pha tiếp theo thì nó vẫn có lỗi. Đây là một cách để kiểm thử xem nó dùng được ở pha tiếp theo hay không.

84. Nếu nhóm SQA phát hiện ra ít lỗi, thì có thể kết luận nhóm code giỏi hay nhóm SQA kém?Tại sao?
Trả lời:

Nếu nhóm SQA phát hiện ra ít lỗi, thì chưa thể kết luận được bởi nó còn phụ thuộc vào mức độ lỗi nặng hay không và tổng số lỗi của đội phát triển phần mềm tạo ra. Ví dụ nhóm phát triển phần mềm có 5 lỗi còn nhóm kiểm thử tìm được 5 lỗi thì nhóm kiểm thử vẫn làm việc tốt.

85. Tại sao nói inspection và walkthrough là hướng tài liệu, mà không phải hướng vào người tham gia?

Trả lời:

Inspection và walkthrough là hướng tài liệu, mà không phải hướng vào người tham gia bởi vì những lỗi được phát hiện chưa chắc đã sai trong thực tế.

86. Quality assurance thì khác gì với testing?

Trả lời:

Sự khác biệt giữa QA và testing chất lượng là đảm bảo chất lượng là về các hoạt động được thiết kế để đảm bảo dự án phù hợp với kỳ vọng của các bên liên quan, trong khi testing là một quá trình để khám phá một hệ thống để tìm khuyết điểm.

87. Tại sao nói function point chịu ảnh hưởng chủ quan của các chuyên gia?

Trả lời:

Function point chịu ảnh hưởng chủ quan của các chuyên gia bởi vì phải ước lượng input, output, maf, inq, inf dễ, trung bình khó và độ phức tạp kỹ thuật TCF.

88. COCOMO tính đến nhiều tiêu chí hơn hay là function point? Giải thích?

Trả lời:

COCOMO tính nhiều tiêu chí hơn function point bởi vì:

- FP phải phân loại mỗi thành phần của phần mềm(Input, output, inq, maf, inf) thuộc loại đơn giản, trung bình, phức tạp. Sau đó toán độ phức tạp kỹ thuật đối với mỗi nhân tố trong 14 nhân tố gán giá trị từ 0 đến 5. Rồi tính số điểm chức năng bằng công thức
- COCOMO ước lượng kích thước phần mềm theo KDSI, xét xem phần mềm thuộc dạng organic, semi-detached, embedded. Tính giờ công chuẩn rồi nhân với hệ số chi phí với 6 mức độ(very low, low, normal, high, very high, extra high). Giờ công cuối cùng làm đầu vào cho ước lượng chi phí, lập kế hoạch phân bổ nhân sự, chi phí khấu hao, bảo trì hàng năm.

89. SW development multiplier của COCOMO thì khác gì TCF của function point?

Trả lời:

TCF là độ phức tạp kỹ thuật được tính dựa trên 14 nhân tố gán giá trị từ 0 đến 5, SW development multiplier của COCOMA là tích 15 hệ số chi phí với 6 mức độ

90. TCF của function point thì khác gì hằng số b của FFP?

Trả lời:

Hằng số b là hiệu năng của công ty, TCF là độ phức tạp về kỹ thuật: TCF = 0.65 + 0.01*DI (DI là mức độ ảnh hưởng tổng thể)

91. Tại sao nguyên lí Djistra lại đúng?

Trả lời:

Nguyên lí Djistra: Kiểm thử một chương trình rất dễ chỉ ra chương trình có lỗi, nhưng rất khó để chứng tỏ rằng chương trình không còn lỗi.
Bởi vì: muốn kiểm thử chương trình có lỗi thì chỉ cần kiểm thử một trường hợp có lỗi. Còn để kết luận một chương trình không có lỗi thì phải kiểm thử hết tất cả các trường hợp mà thường thì kiểm thử tất cả các trường hợp rất khó vì số lượng test case rất nhiều nên rất khó để thử được hết. Như vậy rất khó để chứng tỏ rằng chương trình không có lỗi
92. Tại sao luật Brook lại đúng?
Trả lời:
Luật Brook: Khi đưa thêm người mới vào dự án đang nguy cơ bị trễ, thì không giải quyết được vấn đề trễ, thậm chí còn làm dự án bị trễ thêm!
Bởi vì khi thêm người mới vào thì cần phải chia lại công việc trong nhóm, thêm kênh giao tiếp khiến cho lượng công việc này đội lên đôi khi nhiều hơn công việc ban đầu.
93. Người ta áp dụng luật Miller trong CNPM như thế nào?
Trả lời:
Luật Miller: Tại mỗi thời điểm, người ta chỉ có thể tập trung vào tối đa khoảng 7 vấn đề
→ Áp dụng vào CNPM: để xử lí các vấn đề lớn, sử dụng phương pháp làm mịn từng bước:
- Tập trung xử lí các việc quan trọng trước
- Các việc ít quan trọng hơn xử lí sau
94. Phát triển phần mềm thì khác gì sản xuất phần mềm?
Trả lời:
- Phát triển phần mềm là quá trình tạo ra phần mềm.
- Sản suất phần mềm thường là sản xuất hàng loạt để bán ra thị trường.

95. Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại test gì?
Trả lời:
Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại kiểm thử chức năng.