HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

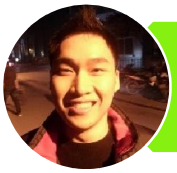
D11CN6

GIẢI NGÂN HÀNG ĐỀ THI

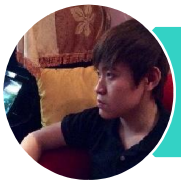
CÔNG NGHỆ PHẦN MỀM

BƯỚM VÀNG MINI GROUP


Tiến Đạt Nguyễn


Ngọc Sơn Đặng


Hưng Hán Việt


Vinh Huy Trần


Tiến Đạt Ngô

LỜI MỞ ĐẦU

Đây là bản bài làm của nhóm tác giả, trong quá trình làm có thể xay ra sai sót do quá trình dịch tài liệu hoặc nguồn tài liệu sử dụng để làm chưa chính xác. Mong các bạn góp ý và xây dựng để bài làm được hoàn thiện hơn

Mọi phản hồi mong các bạn gửi về địa chỉ Facebook của một trong các thành viên nhóm tác giả:

1. https://www.facebook.com/tiendat1025
2. https://www.facebook.com/igcsemybridge
3. https://www.facebook.com/gooner.forever93
4. https://www.facebook.com/mattroidem0593
5. https://www.facebook.com/tranvinhhuy

Bài làm mang tính xây dựng tài liệu ôn thi cho môn. Không sử dụng với mục đích khác.

Xin cảm ơn các bạn
NHÓM TÁC GIẢ

Câu hỏi lí thuyết Kỹ nghệ phần mềm

1. Phần mềm là gì? Nêu đặc trưng của nó. Có những loại ngôn ngữ nào để phát triển phần mềm?
2. Phân loại phần mềm và nội dung cơ bản mỗi loại.
3. Định nghĩa kỹ nghệ phần mềm? Những yếu tố chủ chốt trong kỹ nghệ phần mềm là gì?
4. Tiến trình phần mềm là gì? Mô hình tiến trình là gì? Hãy trình bày mô hình của một số tiến trình cơ bản.
5. Các bước tổng quát của tiến trình phần mềm gồm những giai đoạn nào? Nêu các hoạt động của tiến trình phần mềm và tài liệu mà nó sinh ra?
6. Chất lượng phần mềm là gì? Các tiêu chí của chất lượng phần mềm.
7. Có các dạng bảo trì nào? Nêu và phân biệt.
8. Thế nào là refactoring?
9. Thế nào là "from scratch"?
10. Thế nào là môt episode?
11. Thế nào là một artifact?
12. Thế nào là portability của phần mềm?
13. Thế nào là reusebility của phần mềm?
14. Thế nào là một bản thiết kế còn ommision?
15. Thế nào là một bản thiết kế còn contradiction?
16. Thế nào là một phần mềm COTS?
17. SPMP là viết tắt của từ gì? Ý nghĩa?
18. alpha release là gì? Khác gì với beta release?
19. beta release là gì? Khác gì với alpha release?
20. process là gì? Khác gì với workflow?
21. workflow là gì? Khác gì với process?
22. Tại sao không có pha kiểm thử?
23. Tại sao không có pha làm tài liệu?
24. Tại sao không có pha lập kế hoạch?
25. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao?
26. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?
27. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?
28. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?
29. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?
30. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?
31. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?
32. Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng thì phù hợp với những dự án có đặc điểm gì?
33. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu xoắn ốc?
34. Mô hình vòng đời phần mềm kiểu xoắn ốc thì phù hợp với những dự án có đặc điểm gì?
35. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa?
36. Mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa thì phù hợp với những dự án có đặc điểm gì?
37. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu mã nguồn mở?
38. Mô hình vòng đời phần mềm kiểu mã nguồn mở thì phù hợp với những dự án có đặc điểm gì?
39. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu tiến trình linh hoạt?

40. Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt thì phù hợp với những dự án có đặc điểm gì?
41. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả?
42. Trong mô hình tiến trình liên hoạt, luôn có đại diện của khác hàng trong nhóm phát triển thì cớ ưu điểm gì?
43. Nêu ưu điểm, nhược điểm của mô hình nhóm code bình đẳng?
44. Mô hình nhóm code bình đẳng thì phù hợp với những dự án có đặc điểm gì?
45. Nêu ưu điểm, nhược điểm của mô hình nhóm code có chef?
46. Mô hình nhóm code có chef thì phù hợp với những dự án có đặc điểm gì?
47. Nêu ưu điểm, nhược điểm của kĩ thuật pair programming?
48. Nêu ưu điểm, nhược điểm của kĩ thuật time boxing?
49. Nêu ưu điểm, nhược điểm của kĩ thuật stand up meeting?
50. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC?
51. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP?
52. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point?
53. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO?
54. Luật Miller trong CNPM nói gì?
55. Luật Brooks trong CNPM nói gì?
56. Luật Dijkstra trong CNPM nói gì?
57. Ý nghĩa của scenario và ngoại lệ?
58. Ý nghĩa của sơ đồ tuần tự?
59. Ý nghĩa của sơ đồ lớp?
60. Thế nào là lớp giao diện? Lớp này thường tương tác với các lớp nào?
61. Thế nào là lớp điều khiển? Lớp này thường tương tác với các lớp nào?
62. Thế nào là lớp thực thể? Lớp này thường quan hệ với các lớp nào?
63. Scenario và sơ đồ tuần tự có liên hệ gì với nhau?
64. Scenario và sơ đồ lớp có quan hệ gì với nhau?
65. Việc trích lớp và xây dựng các lớp là việc của pha thiết kế, tại sao người ta lại bắt đầu ngay trong pha phân tích?
66. Kĩ thuật trích danh từ được dùng để trích các lớp nào? Có thể dùng để trích các lớp biên và lớp điều khiển được không?
67. Trình bày kĩ thuật trích lớp điều khiển? Số lượng lớp điều khiển nhiều hay ít thì tốt?
68. Ý nghĩa của sơ đồ trạng thái hữu hạn? Nó biểu diễn trạng thái của hệ thống, của lớp hay của phương thức?
69. Ý nghĩa của thẻ CRC? Dùng thẻ CRC cho lớp biên và lớp thực thể có được không? Có cần không?
70. Thiết kế kiến trúc thì cần các sơ đồ nào của UML?
71. Thiết kế chi tiết thì cần các sơ đồ nào của UML?
72. Làm thế nào để trích các lớp? Cần dùng các sơ đồ nào của UML?
73. Làm thế nào để trích các phương thức của lớp? Cần dùng các sơ đồ/công cụ nào của UML?
74. Nếu cho các phương thức add/update/delete đối tượng vào lớp thực thể tương ứng thì có được không? Tại sao?
75. Sơ đồ tuần tự và thẻ CRC có quan hệ gì với nhau?
76. Sơ đồ lớp và sơ đồ cộng tác có gì khác nhau?
77. Mỗi trạng thái của sơ đồ trạng thái thường ứng với một lớp hay một phương thức, tại sao?
78. Trình bày nguyên lí A của phần gán phương thức cho lớp? Nguyên lí này thường dùng cho các lớp loại nào?
79. Trình bày nguyên lí B của phần gán phương thức cho lớp? Nguyên lí này thường dùng cho các lớp loại nào?
80. Trình bày nguyên lí C của phần gán phương thức cho lớp? Nguyên lí này thường dùng cho

các lớp loại nào?
81. Thẩm định và xác minh là gì (V&V)? Tầm quan trọng của chúng?
82. Có những loại V&V nào? Mô tả nội dung mỗi loại.
83. inspection là gì? Khác gì với walkthrough?
84. walkthrough là gì? Khác gì với inspection?
85. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo?
86. Người ta nói « nhóm SQA tạo ra chất lượng cho phần mềm » đúng hay sai? Tại sao?
87. Scenario và test case có liên quan gì đến nhau?
88. Nêu các phần chính phải có của một test case?
89. Người ta có thể bắt đầu viết test case cho hệ thống bắt đầu từ bước nào?
90. Phần mềm còn lỗi hay không khi thực hiện hết các ca kiểm thử được thiết kế? Khi nào dừng kiểm thử?
91. Trình bày phương pháp kiểm thử hộp trắng: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.
92. Trình bày phương pháp kiểm thử hộp đen: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.
93. Kiểm thử đơn vị đối tượng là gì? Ai thực hiện. Các phương pháp và kỹ thuật nào được sử dụng? Kiểm tra những loại lỗi nào?
94. Các chiến lược nào sử dụng trong kiểm thử tích hợp? Ưu điểm và hạn chế mỗi loại?
95. Giải thích khái nhiệm stub và driver? Chúng được sử dụng ở đâu và vì sao?
96. Kiểm thử hệ thống nhằm kiểm tra cái gì? Ai thực hiện? Các phương pháp?
97. Trình bày các kiểm thử được thực hiện trong kiểm thử hệ thống?
98. Kiểm thử chấp nhận là gì? Trong đó có những kiểm thử nào được thực hiện? Phân biệt.
99. Mô hình CMM là gì? Có những mức tăng trưởng nào trong mô hình CMM? Nội dung của mỗi mức?
100. Làm thế nào để một tổ chức đạt được các mức tăng trưởng trong CMM? Đâu là giải pháp và thước đo về các mức tăng trưởng?
101. Nêu các chuẩn quốc tế về phần mềm. Trình bày sự khác nhau giữa mô hình CMM và các chuẩn đó.

1. Phần mềm là gì? Nêu đặc trưng của nó. Có những loại ngôn ngữ nào để phát triển phần mềm?

- Phần mềm là sản phẩm do các nhà phát triển phần mềm thiết kế và xây dựng
- Đặc trưng: Phần mềm không hao mòn, pm được phát triển chức ko phải là sản xuất, phần mềm rất phức tạp chi phí thay đổi rất lớn.
- Các ngôn ngữ: java .c,c++….

2. Phân loại phần mềm và nội dung cơ bản mỗi loại.

Phần mềm hệ thống: (hệ điều hành, phần mềm vận hành thiết bị,,,) giúp vận hành phần cứng

Phần mềm ứng dụng: cung cấp công cụ hỗ trợ lập trình viên trong khi viết ct và phần mềm bằng các ngôn ngữ khác nhau.

Các loại khác : virus được viết để chạy với mục đích xấu.

3. Định nghĩa kỹ nghệ phần mềm? Những yếu tố chủ chốt trong kỹ nghệ phần mềm là gì?

KNPM: là sự áp dụng một cách tiếp cận có hệ thống, có kỉ luật, va định lượng được cho việc phát triển. hoạt động và bảo trì phần mềm.

KNPM bao trùm kiến thức, các công cụ và các phương pháp cho việc định nghĩa yêu cầu phần mềm, và thực hiện các tác vụ thiết kế, xây dựng kiểm thử, bảo trì phần mềm.

4. Tiến trình phần mềm là gì? Mô hình tiến trình là gì? Hãy trình bày mô hình của một số tiến
trình cơ bản.

Tiến trình phần mềm là 1 tập các hành động mà mục đích của nó là xây dựng và phát triển phần mềm.

5. Các bước tổng quát của tiến trình phần mềm gồm những giai đoạn nào? Nêu các hoạt động của tiến trình phần mềm và tài liệu mà nó sinh ra?

- Đặc tả yêu cầu: (requirements specification): chỉ ra những đòi hỏi cho cả các yêu cầu chức năng và phi chức năng.
- Phát triển phần mềm(development): tạo ra phần mềm thỏa mãn các yêu cầu được chỉ ra trong đặc tả yêu cầu.

- Kiểm thử phần mềm(validation/testing): để bảo đảm phần mềm sản xuất ra đáp ứng những đòi hỏi được chỉ ra trong đặc tả yêu cầu.

6. Chất lượng phần mềm là gì? Các tiêu chí của chất lượng phần mềm.

Chất lượng phần mềm là sự đáp ứng các yêu cầu chức năng, sự hoàn thiện và các chuẩn(đặc tả) được phát triển, các đặc trưng mong chờ từ mọi phần mềm chuyên nghiệp(ngầm định)

Các tiêu chí:

1. tính bảo trì

2. độ tin cậy

3. tính hiệu quả

4. tính khả dụng

7. Có các dạng bảo trì nào? Nêu và phân biệt.

4 loại:

- bảo trì để tu chỉnh: là bảo trì khắc phục những khiếm khuyết co trong phần mềm.
- bảo trì để thích ngi: là tu chình phần mềm theo thay đổi của môi trường bên ngoài nhằm duy trì, thích nghi và quản lí phần mềm theo vòng đời của nó.
- bảo trì để hoàn thiện : là việc tu chỉnh phần mềm theo các yêu cầu ngày càng hoàn thiện hơn, đầy đủ hơn, hợp lí hơn.
- bảo trì để phòng ngừa: là công việc tu chình chương trình có tính đến tương lai của phần mềm đó sẽ mở rộng và thay đổi ntn.

8. Thế nào là refactoring?

Refatoring là cải tiến và làm tốt hơn chất lượng của mã nguồn trong một ứng dụng. Nó không làm thay đổi các chức năng chính, chức năng chung của ứng dụng, nhưng nó làm cho ứng dụng dễ bảo trì hơn dễ phát triển hơn trong tương lai.

9. Thế nào là "from scratch"?

Là bắt đầu xây dựng phần mềm từ mốc không có gì

Không dựa trên 1 cơ sở hay là 1 phần mềm có sẵn nào từ trước mà là làm mới hoàn toàn

10. Thế nào là môt episode?

1 episode là 1 lần lặp lại các pha

Ví dụ: khi mình thưc hiện xong từ phân tích tới cài đặt là 1 ep. nhưng đúng lúc ấy, lại phát hiện ra bản thiết kế k chính xác chẳng hạn

thì mình lại phải thiết kế lại hoặc bổ sung thiết kế và cài đặt lại thì từ thiết kế lại tới cài đặt lại lại là 1 ep. cứ tương tự thì phần mềm sẽ đc hoàn thành và bàn giao tới tay khách hàng sau nhiều ep

11. Thế nào là 1 artifact ?

Artifact (đồ tạo tác) là một trong nhiều loại sản phẩm hữu hình được tạo ra trong quá trình phát triển phần mềm. Một số Artifact (ví dụ: use case, sơ đồ lớp, các mô hình UML khác, yêu cầu và các tài liệu thiết kế) giúp mô tả chức năng, kiến trúc và thiết kế cho phần mềm. Mỗi artifact khác nhau thì liên quan với qui trình phát triển riêng của nó - chẳng hạn như kế hoạch dự án, các business case, và đánh giá rủi ro.

12. Thế nào là portability của phần mềm ?

Portability (tính khả chuyển) của phần mềm là mức độ dễ dàng trong việc chỉnh sửa toàn bộ phần mềm để có thể chạy được trên một hệ thống, môi trường, phần cứng khác mà không phải code lại từ đầu (from scratch).

13. Thế nào là reusebility của phần mềm ?

Reusebitity (tính sử dụng lại) của phần mềm là khả năng sử dụng lại các thành phần của 1 sản phẩm để tạo điều kiện cho phát triển 1 sản phẩm khác với những chức năng khác.

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

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

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

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.

16. Thế nào là một phần mềm COTS (Commercial-off-the-self Software) ?

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..)

17. SPMP là viết tắt của từ gì ? Ý nghĩa ?

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.

18. Alpha release là gì? Khác gì với beta release?

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.

19. Beta release là gì? Khác gì với alpha release?

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)

20. process là gì? Khác gì với workflow?

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

Câu 21 : workflow là gì? Khác gì với process?

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

Cấu 22: Tại sao không có pha kiểm thử?
Vì chương trình 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 giai đoạn cuối và phát hiện ra lỗi, thì thường bàn giao sản phẩm không đúng hạn

Câu 23: Tại sao không có pha làm tài liệu?
Mọi pha phải đc viết tài liệu trc khi bắt đầu 1 pha mới vì

+ Tài liệu bị hoãn lại thì se k bao giờ hàn thành

+ Cá nhân chịu trách nhiệm trong pha trc có thể chuyển sang bộ phận khác

+ SP thường xuyên thay đổi khi phát triển nên cần tài liệu để ghi lại điều này

Câu 24. Tại sao không có pha lập kế hoạch?
Chúng ta k thể lập kế hoạch vào đầu dự án vì 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.
25. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềmkhông? Tại sao?
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

26. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?

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

27. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?

* Ưu điểm:

- Dễ pân công công việc, phân bố chi phí, giám sát công việc.

- Kiến trúc hệ thống hàng đợi ổn định.

* 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. Giả sử, pha phân tích và xác định yêu cầu đã hoàn tất và chuyển sang pha kế tiếp, nhưng lúc này lại có sự thay đổi yêu cầu của người sử dụng; thì chỉ còn cách là phải thực hiện lại từ đầu.

- Chỉ tiếp xúc với khách hằng ở pha đầu tiên nên phần mềm ko đáp ứng được hết các yêu cầu của khách hang.

- Chi phí phát triển dự án tương đối lớn.

- Khả năng thất bại cao.

28. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?

- Mô hình waterfall 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.
- Waterfall 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
- Đối với những khách hàng lớn mà phong cách làm việc của họ chủ yếu theo mô hình truyền thống (waterfall) hoặc những khách hàng lo ngại có nhiều thay đổi trong dự án
- Nên áp dụng waterfall với những dự án có fixed scope và fixed-price contract
- Không gấp rút thời gian

29. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?

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.

+ nhóm thiết kế sử dụng đặc tả để thiết kế mà không cập nhật ý kiến khách hàng.

=> có thể gây nên sai sót, gây tổn thất.

30. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?

Thiết kế nhanh chỉ tập trung vào việc biểu diễn các khía cạnh của phần mềm thấy được đối với người dùng

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.
Phạm vi hoạt động cố định và giá cố định.

31. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?

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.

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

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

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

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 và ít rủi ro.
- 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 hay là mẫu.
- Đòi hỏi năng lực quản lý

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

Hợp với hệ thống lớn có thể phân chia thành nhiều thành phần, ít rủ ro.

35. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa?

Ưu điểm:

Một điểm lợi của cách làm này là những người phát triển có thể sớm nhìn thấy sự hoạt động của phần mềm và có thể hiệu chỉnh các yêu cầu, có thể là ngay trong quá trình các thành phần được xây dựng. Mô hình này có thể áp dụng ngay cả trong trường hợp các đặc tả ban đầu không hoàn thiện.

Nhược điểm: không có thay đổi trong đặc tả

36. Mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa thì phù hợp với những dự án có đặc điểm gì?

Hợp với hệ thống lớn có thể phân chia thành nhiều thành phần, ít rủ ro.

Phần mềm lấy đặc trưng của nhu cầu khách hàng được liệt kê theo thứ tự ưu tiên

Phầm mềm được đóng gói.

37. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu mã nguồn mở?

Ưu điểm:

- tính tái sử dụng: bạn có thể tạo các thành phần (đối tượng) một lần và dùng chúng nhiều lần sau đó.
- khả năng tái sử dụng đối tượng có tác dụng giảm thiểu lỗi và các khó khăn trong việc bảo trì, giúp tăng tốc độ thiết kế và phát triển phần mềm.
- Phương pháp hướng đối tượng giúp chúng ta xử lý các vấn đề phức tạp trong phát triển phần mềm và tạo ra các thế hệ phần mềm có khả năng thích ứng và bền chắc.
- Dễ dàng mở rộng, nâng cấp

Nhược điểm:

- có những hạn chế như báo cáo thất bại (các báo cáo về hành vi không chính xác quan sát được), người dùng không có quyền truy cập vào mã nguồn, vì vậy họ không thể gửi báo cáo lỗi (báo cáo đưa ra chỗ nào mã nguồn không chính xác và làm thế nào để sửa chữa nó).
- Rất khó để hình thành việc phát triển mã nguồn mở của một sản phẩm phần mềm để sử dụng trong một tổ chức thương mại.
- Mô hình có thể dẫn đến sự phát triển tùy tiện của các pha,và làm tùy tiện không có kỷ luật

38. Mô hình vòng đời phần mềm kiểu mã nguồn mở thì phù hợp với những dự án có đặc điểm gì?

- Phù hợp với các hệ thống lớn: Phương pháp hướng đối tượng không chia bài toán thành các bài toán nhỏ mà tập trung vào việc xác định các đối tượng, dữ liệu và hành động gắn với đối tượng và mối quan hệ giữa các đối tượng.
- Hỗ trợ sử dụng lại mã nguồn

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

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 XP 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.

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

quy trình mau lẹ dường như là một cách tiếp cận hữu ích để xây dựng các sản phẩm phần mềm quy mô nhỏ khi yêu cầu của khách hàng không rõ ràng, hay thay đổi

41. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả?

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.

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

Trong mô hình tiến trình linh hoạt (Agile Process) 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.

43. Ưu nhược điểm của mô hình nhóm code bình đẳng (Democratic Team) :

- **Ưu điểm** : Do mô hình này 1 người luôn khuyến khích người khác tìm ra lỗi của mình nên việc tìm ra lỗi trong pm sẽ nhanh hơn, chất lượng phần mềm được cải thiện.
- **Nhược điểm** :

+ Quản lí thường khó chấp nhận kiểu lập trình Ego Programming này

+ Những người có kinh nghiệm thường không thoải mái khi để những người non kinh nghiệm kiểm tra code của mình

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

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

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.

45. Ưu nhược điểm của mô hình nhóm code có chief (Chief Programmer Team) :

- **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 Chief Programmer 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.

+ Backup programmer là người cũng phải có khả năng như Chief tuy nhiên lại phải ở vị trí dự bị, do đó cũng rất khó kiếm ra người như vậy

+ Programming Secretary (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.

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

Mô hình có Chief thì 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.

47. Ưu nhược điểm của kĩ thuật Pair-programming :

- **Ưu điểm** :

+ Thời gian hoàn thành công việc nhanh, thời gian cài đặt và tích hợp chỉ trong vài giờ

+ Khi một người rời khỏi vị trí công việc, người còn lại vẫn có đủ hiểu biết để tiếp tục với người ghép cặp mới, tránh tình trạng phải cài đặt lại từ đầu.

+ Làm việc trong 1 cặp sẽ giúp người non kinh nghiệm học hỏi được nhiều từ người cùng cặp giàu kinh nghiệm hơn.

+ Tất cả máy tính của các cặp sẽ được đặt ở giữa một phòng lớn, do đó tăng tính sở hữu code theo nhóm – đặc trưng tích cực của egoless team (đội bình đẳng)

- Nhược điểm :

+ Đòi hỏi những khoảng thời gian lớn làm việc liên tục

+ Không hiệu quả trong trường hợp có những cá nhân ngại ngùng, kiêu ngạo hoặc cả 2 người đều thiếu kinh nghiệm

48. Ưu nhược điểm của kĩ thuật time boxing :

- Ưu điểm :

+ Thúc đẩy năng suất và hiệu quả của quá trình phát triển phần mềm

+ Giảm thời gian phát triển phần mềm

+ Lập trình viên tập trung cao, ý thức về khoảng thời gian mình có

+ Giúp quản lí tốt rủi ro và độ phức tạp

- Nhược điểm :

+ Nếu không thể hoàn thành toàn bộ công việc trong khoảng thời gian được định sẵn thì công việc sẽ được cắt giảm bớt do timeboxing chỉ yêu cầu thời gian cố định chứ không có chỉ tiêu cố định nên chỉ thích hợp với các dự án nhỏ

49. Ưu nhược điểm của kĩ thuật stand-up meeting :

- Ưu điểm :

+ Cũng giống timeboxing, stand-up meeting giúp kết nối và đáp ứng yêu cầu của khách hàng nhanh nhất có thể

- Nhược điểm :

+ Giống timeboxing, chỉ phù hợp với các dự án nhỏ.

50. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC (Lines of code) ?

- Ưu điểm : đơn giản, trực quan

- Nhược điểm :

+ Phụ thuộc vào ngôn ngữ lập trình

+ Phụ thuộc vào kinh nghiệm, phong cách của lập trình viên

+ Phụ thuộc vào cách tính (thế nào là 1 dòng lệnh)

=> Ít được sử dụng

+ Thời điểm sớm nhất có thể áp dụng pp này là cuối pha cài đặt => khó khả thi, chỉ phù hợp với những dự án nhỏ, đơn giản

51. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP ?

- Ưu điểm :

+ Chính xác hơn LOC, ổn định hơn (nhiều tham số hơn)

+ Áp dụng ở thời điểm sớm hơn LOC (ở cuối pha thiết kế)

- Nhược điểm :

+ Phụ thuộc vào hằng số b

+ Vẫn có biến thiên (như số lượng file)

52. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point ?

- Ưu điểm :

+ Độ chính xác cao hơn, ổn định hơn FFP

+ Áp dụng ở cuối pha thiết kế

- Nhược điểm : Đánh giá dựa vào chuyên gia (mang sự chủ quan)

53. Nêu ưu, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO ?

Gồm 3 loại COCOMO :

***) COCOMO Cơ bản (Basic)** :

- Ưu điểm : Áp dụng tốt để tính nhanh và sơ bộ chi phí cho các dự án nhỏ và vừa

- Nhược điểm : Bị giới hạn chức năng do không cân nhắc đến các yếu tố như ràng buộc phần cứng, chất lượng nhân sự, kinh nghiệm, trình độ kĩ thuật và công cụ sử dụng

***) COCOMO Trung gian (Intermediate)**

- Ưu điểm : Được áp dụng cho gần như toàn bộ dự án ở những cài đặt đơn giản và thô sơ. Nó cũng có thể được áp dụng để tính chi phí ở mức các thành phần của phần mềm

- Nhược điểm : Sản phẩm có quá nhiều thành phần sẽ khó tính được bằng Intermediate COCOMO

***) COCOMO Chi tiết (Detailed)**

- Ưu điểm : Dễ hiểu, trực quan, đặc biệt hữu ích cho người ước lượng hiểu được ảnh hưởng của những yếu tố khác nhau đến chi phí sản phẩm

- Nhược điểm : Thành công của pp phụ thuộc vào mô hình áp dụng với yêu cầu của khách hàng.

54. Luật Miller trong CNPM nói gì ?

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)

55. Luật Brooks trong CNPM nói gì ?

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

56. Luật Dijkstra trong CNPM nói gì ?

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

57. Ý nghĩa của scenario và ngoại lệ ?

- Từ scenario ta làm rõ được chức năng của use case và sự tương tác giữa người dùng/ người quản trị với hệ thống.
- Từ scenario ta cũng xác định được ngoại lệ và cách xử lí ngoại lệ của hệ thống.

58. Ý nghĩa của sơ đồ tuần tự ?

Sơ đồ tuần tự thể hiện sự tương tác giữa các lớp trong hệ thống với nhau và với người dùng.

59. Ý nghĩa của sơ đồ lớp ?

- Mối quan hệ giữa các đối tượng trong hệ thống sẽ được thể hiện qua mối quan hệ giữa các lớp.
- Mô tả 1 hướng nhìn tĩnh về 1 hệ thống bằng các lớp, các thuộc tính, phương thức của lớp và mối liên hệ giữa chúng.

60. Thế nào là lớp giao diện ? Lớp này thường tương tác với các lớp nào ?

- Lớp giao diện là lớp có chức năng tương tác giữa môi trường bên ngoài và phần mềm. Lớp giao diện thường là giao diện vào ra với người dùng
- Lớp giao diện tương tác với lớp điều khiển để trao đổi thông tin giữa người dùng và hệ thống.

61. Thế nào là lớp điều khiển, lớp này tương tác với những lớp nào?

Lớp điều khiển (còn gọi là lớp control)

- Dùng để mô hình các tính toán và thuật toán phức tạp trong hệ thống.
  Có thể chỉ dùng 1 lớp điều khiển cho các hệ thống đơn giản, mỗi phương thức là 1 hàm xử lý, tính toán độc lập.
- Lớp điều khiển tương tác với các lớp thực thể (model) và lớp biên (view) làm nhiệm vụ tính toán và xử lý thông tin.

62. Thế nào là lớp thực thể, lớp này tương tác với những lớp nào?

Lớp thực thể (còn gọi là lớp model)

- Dùng để biểu diễn dữ liệu để xử lý, trao đổi giữa các đối tượng trong hệ thống. Thường chỉ có các thuộc tính và các phương thức truy nhập (get/set).
- Lớp thực thể tương tác với lớp điều khiển (control) để cung cấp và lưu trữ thông tin.

63. Scenario và sơ đồ tuần tự có liên hệ gì đến nhau?

- Sơ đồ tuần tự minh họa các đối tượng tương tác với nhau ra sao.
- Một sơ đồ tuần tự nêu lên sự tương tác trong một scenario: Sự tương tác xảy ra tại một thời điểm nào đó trong quá trình thực thi hệ thống.

- Ngược lại, scenario chính là nội dung của sự tương tác của các đối tượng được thể hiện trong sơ đồ tuần tự.

64. Scenario và sơ đồ lớp có quan hệ gì đến nhau?

- Một nhóm đối tượng có chung 1 số thuộc tính và phương thức sẽ tạo thành 1 lớp. Mối tương tác giữa các đối tượng trong hệ thống được biểu hiện qua mối tương tác giữa các lớp.

- Sơ đồ lớp được tạo thành từ các lớp (bao gồm các thuộc tính và phương thức), và các mối quan hệ của nó.
- Mỗi lớp trong 1 sơ đồ sẽ bao gồm đầy đủ các thuộc tính và các phương thức cùng mối quan hệ của chúng đã được thể hiện trong scenario.
- Ngược lại, scenario đưa ra những thuộc tính, phương thức mà sơ đồ cần thể hiện.

65. Việc trích lớp và xây dựng các lớp là của pha thiết kế, tại sao người ta làm nó ngay trong pha phân tích?

Là do bước đầu tiên trong quá trình phân tích là xác định các lớp. Bởi vì mỗi lớp là một kiểu của mô-đun, việc phân tích mô-đun được thực hiện trong suốt quá trình phân tích. "Because a class is a type of module, the modular decomposition has been performed during the analysis workflow ???"

66. Kỹ thuật trích danh từ dùng để trích các lớp nào? Có thể dùng để trích lớp biên và lớp điều khiển được không?

Kỹ thuật này được dùng để trích các lớp thực thể, không thể dùng để trích các lớp biên và lớp điều khiển vì trong phần kịch bản của 1 use case chỉ mô tả tương tác giữa người dùng và hệ thống mà không nói đến cách thức hệ thống xử lý, điều khiển các chức năng.

67. Trình bày kỹ thuật trích lớp điều khiển? Số lượng lớp điều khiển nhiều hay ít thì tốt?

Kỹ thuật trích các lớp điều khiển:

- Toàn bộ hệ thống dùng chung một lớp điều khiển.
- Mỗi modul dùng riêng một lớp điều khiển.

68. Ý nghĩa của sơ đồ trạng thái hữu hạn? Nó biểu diễn trạng thái của cả hệ thống, của lớp hay của phương thức?

Ý nghĩa: Mô tả các sự kiện và hoạt động của từng hệ thống, lớp hoặc phương thức, cụ thể trong từng thao tác và gắn với phương thức/ trừ trường hợp xảy ra với từng đối tượng.

Được sử dụng cho cả hệ thống, các lớp và phương thức.

69. Ý nghĩa thẻ CRC, dùng thẻ CRC với lớp biên và lớp điều khiển được không? Có cần thiết không?

Ý nghĩa thẻ CRC : Được dùng để mô hình hóa quan hệ giữa các lớp

- C : class. Biểu diễn tên lớp.
- R : responsibility. Trách nhiệm của lớp.
- C : collaboration. Quan hệ của lớp.

70. Thiết kế kiến trúc cần sơ đồ quan hệ các lớp (Communication Diagram)

71. Thiết kế chi tiết cần Sơ đồ cần dùng: Sequence Diagram (tuần tự), Class Diagram (Sơ đồ lớp dạng ô vuông có thể điền thuộc tính và phương thức)

72. Làm thế nào để trích các lớp? Cần dùng các sơ đồ nào của UML?

Kỹ thuật để trích các lớp :

- Mô tả hoạt động của ứng dụng trong một đoạn văn.
- Trích các danh từ xuất hiện trong đoạn văn đó, coi như là một ứng cử viên cho lớp thực thể.
- Xét duyệt từng danh từ và đề xuất nó là lớp thực thể hay là thuộc tính của lớp thực thể.
- Sơ đồ cần dùng : Sơ đồ quan hệ các lớp.

73. Làm thế nào để trích các phương thức của lớp? Cần dùng các sơ đồ/ công cụ nào của UML?

Kỹ thuật trích các phương thức của lớp :

- Dựa trên các lớp đã xác định, với mỗi lớp, những danh từ nào mô tả thông tin của lớp đó -> tìm ra các thuộc tính.
- Loại bỏ những thuộc tính không cần thiết.
- Những thông tin nào là thông tin riêng của lớp (các thuộc tính private), những thông tin nào có thể được chia sẻ trong mối liên hệ với các lớp khác (các thuộc tính protected và public).
- Sau đó, xem xét các động từ đi kèm với các danh từ biểu diễn trong kịch bản để xem động từ đó có trở thành phương thức hay không. Trong pha phân tích chỉ xác định được một số phương thức dễ nhìn thấy.

- Với pha thiết kế: sau khi xây dựng xong sơ đồ tuần tự với các message, cần xem các message để xác nhận hành động tương ứng với message đó thuộc lớp nào.
- Các phương thức cần thiết để chuyển đổi các trạng thái trong sơ đồ trạng thái của một lớp.
- Xác định xem với mỗi lớp, có cần hàm tạo và hàm hủy hay không.
- Xác định chi tiết các giá trị trả về và các tham số liên quan với mỗi phương thức.

Sơ đồ cần dùng: sơ đồ tuần tự, sơ đồ lớp.

74. Nếu cho các phương thức add/update/delete vào các đối tượng tương ứng thì có được không? Tại sao?

Được, khi đó lớp điều khiển sẽ là lớp gọi hàm, lớp thực thể là nơi định nghĩa hàm.

75.

76. Sơ đồ lớp và sơ đồ cộng tác có gì khác nhau?

- Sơ đồ lớp chỉ ra cấu trúc bên trong của các lớp, bao gồm thuộc tính, phương thức, các lớp quan hệ với nhau theo nhiều dạng thức : kế thừa, liên kết…
- Sơ đồ cộng tác chỉ ra sự tác động giữa các đối tượng với nhau, xác định các đối tượng và quan hệ giữa chúng, thể hiện sự tương tác qua những mũi tên theo một dòng chảy thông điệp giữa các đối tượng.

77. Mỗi trạng thái của sơ đồ trạng thái ứng với một lớp hay một phương thức? Tại sao?

Ứng với 1 phương thức vì:

- Trạng thái là một kết quả của các hoạt động trước đó đã được đối tượng thực hiện và nó thường xác định qua giá trị của các thuộc tính cũng như các nối kết của đối tượng với các đối tượng khác. Một lớp có thể có 1 thuộc tính đặc biệt xác định trạng thái, hoặc trạng thái cũng có thể được xác định qua giá trị của các thuộc tính "bình thường" trong đối tượng.
- Mỗi sự chuyển trạng thái được ánh xạ thành 1 phương thức của lớp, mỗi hành động tác động vào trạng thái được ánh xạ vào phương thức tương ứng.

78. Trình bày nguyên lý A của phần gán phương thức cho lớp, nguyên lý này thường dùng cho lớp loại nào?

Nguyên lý A:

- Che dấu thông tin: Các thuộc tính của lớp phải để ở dạng private -> cần các phương thức get/set tương ứng cho các đối tượng khác truy nhập vào các thuộc tính này.
- Áp dụng cho các lớp thực thể: các thuộc tính để private, mỗi thuộc tính có một cặp get/set tương ứng.

79. Trình bày nguyên lý B của phần gán phương thức cho lớp, nguyên lý này thường dùng cho lớp loại nào?

Nguyên lý B:

- Nếu có nhiều đối tượng X gọi đến một hành động k của đối tượng Y thì phương thức để thực hiện hành động k nên gán cho lớp của đối tượng Y, không nên gán cho lớp của đối tượng X.
-

80. Trình bày nguyên lý C của phần gán phương thức cho lớp, nguyên lý này thường dùng cho lớp loại nào?

Nguyên lý C:

- Thiết kế hướng trách nhiệm: Nếu một hành động mà không thể gán thành phương thức cho lớp khác, thì lớp của đối tượng cần thực hiện hành động đó phải chứa phương thức tương ứng của hành động đó.

81. Thẩm định và xác minh là gì (V&V)? Tầm quan trọng của chúng?

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.

82. Có những loại V&V nào? Mô tả nội dung mỗi loại

- Vì V&V thực hiện thẩm định vào cuối mỗi WorkFlow, trong khi yêu cầu kiểm tra được thực hiện thường xuyên trong quá trình phát triển sản phẩm và hoạt động bảo trì là cần thiết. Cho nên người ta thường sử dụng “testing” thay cho V&V.
- Có 2 loại testing là : execution-based testing(test thực thi) and non-execution based testing(test không cần thực thi).

- execution-based testing: Sử dụng đầu vào định sẵn, với nền tảng, môi trường định sẵn. Thực hiện suy diễn từ đầu vào ra kết quả thực thi. (vd: viết test - case chạy code)
- non-execution based testing: kiểm tra phần mềm không sử dụng test-case. (vd: đọc lại source code phần mềm, phân tích logic toán học trong chương trình). Gồm có 2 loại là WalkThrough và Inspection.

83. inspection là gì? Khác gì với walkthrough?

- Inspection:

Team Inspection thường gồm 4 người. Vd cho Inspect design gồm có: Moderator, designer, implementer, tester.

là 1 quy trình gồm 5 bước:

1/ (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.

2/ (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.

3/ 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.

4/ (Rework) các thành viên sửa toàn bộ lỗi được chỉ ra trong báo cáo.

5/ (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).

- Điểm khác so vs Walkthrough: 83 + 84;

84. walkthrough là gì? Khác gì với inspection?

- 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.
- Điểm khác so vs Inspection (83 + 84)

?85. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo?
- Bởi vì người đại diện cho nhóm WF tiếp theo sẽ chuyển những thông tin công việc từ WF trước về thông tin công việc của mình.

86. Người ta nói « nhóm SQA tạo ra chất lượng cho phần mềm » đúng hay sai? Tại sao?
- Sai, SQA không tạo ra chất lượng phần mềm, họ chỉ bảo đảm chất lượng phần mềm. Còn người tạo nên chất lượng phần mềm là Developer.(câu này thầy Hùng đã khẳng định)

?87. Scenario và test case có liên quan gì đến nhau?
+ Hiểu rõ thêm về chức năng của hệ thống mà ta xây dựng
+ từ scenario ta làm rõ được chức năng của use case và sự tương tác giữa người dùng người quản trị với hệ thống
+ Từ scenario ta cũng xác định được các ngoại lệ và cách xử lí ngoại lệ của hệ thống.

?88. Nêu các phần chính phải có của một test case?
- Có 2 loại test case: test biên cho dữ liệu thông thường và test Cơ sở Dữ liệu cho CSDL.
- Kỹ thuật test biên: Phải có 2 phần: Các thao tác và kết quả mong đợi.
Vd:

Các thao tác	**Kết quả mong đợi**
Trong lớp SearchMemberAccFrm:	
Quản lý Không nhập UserName, click OK 1 lần	Hiển thị thông báo lỗi ‘Chưa nhập UserName’

- Kỹ thuật test Cơ sở dữ liệu: phải có 4 phần: Cơ sở dữ liệu ban đầu, Các thao tác, kết quả mong đợi và Cơ sở dữ liệu sau khi đã được cập nhật.

?89. Người ta có thể bắt đầu viết test case cho hệ thống bắt đầu từ bước nào?
Bất cứ Khi nào mà hoàn thành 1 phần của phần mềm và nó có thể chạy được.

90. Phần mềm còn lỗi hay không khi thực hiện hết các ca kiểm thử được thiết kế? Khi nào dừng kiểm thử?
Bất kỳchuẩn chất lượng mong muốn nào mà phần mềm phải thỏa mãn hầu sẵn sàng cho việc phân phối đến khách hàng. Có thể bao gồm các thứ sau :
Các yêu cầu mà phần mềm phải được kiểm thửdưới các môi trường xáx định.
Số lỗi tối thiểu ởcấp an ninh và ưu tiên khác nhau, số phủ kiểm thử tối thiểu,...
Stakeholder sign-off and consensus

91. Trình bày phương pháp kiểm thử hộp trắng: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.
- Kiểm thử hộp trắng (test to code, glass-box, white-box, structural, logic-driven, and path-oriented testing) là phương pháp chọn testcase dựa trên việc kiểm tra code, không phải kiểm tra bản đặc tả.
- Kỹ thuật được sử dụng:

+ Statement Coverage: Phương pháp này chạy 1 loạt các testcase song song với việc đảm bảo mỗi câu (statement) được chạy ít nhất 1 lần. Yêu cầu cần có 1 CASE tool để theo dõi câu lệnh nào đó được thực hiện bao nhiêu lần trong test. Điểm yếu của phương pháp này: nó không bảo đảm đầu ra của mọi nhánh (branch) đều được kiểm tra.

+ Branch Coverage: Chạy 1 loạt các test để bảo đảm tất cả các nhánh đều được kiểm tra ít nhất 1 lần. Và cần có 1 tool để biết được nhánh nào đã kiểm tra (hoặc chưa).

+ Path Coverage: Phương pháp hiệu quả nhất, cho phép kiểm tra tất cả các đường đi (path) của chương trình. Tuy nhiên (giả sử trong 1 chương trình có nhiều vòng lặp) thì số đường đi là rất lớn. Do vậy, vấn đề giảm bớt số đường đi và phát hiện đủ lỗi đang được nghiên cứu.

92. Trình bày phương pháp kiểm thử hộp đen: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.

- Kiểm thử hộp đen (test to specifications, black-box, behavioral, datadriven, functional, and input/output-driven testing.) là phương pháp chọn test case dựa trên tài liệu đặc tả và bỏ qua code.
- kỹ thuật được sử dụng:

+ Equivalence Testing and Boundary Value Analysis:

+ Equivalent Class (lớp tương đương): (pg521): Tất cả các testcase trong lớp tương đương đều có độ tốt như nhau. Do đó với mỗi lớp tương đương chỉ cần 1 testcase là đủ.

VD: quản lý hàng hóa có STT từ 1 đến 2000, ta có 3 khoản Equi Class: (<1); [1, 1000], (>1000) và mỗi khoảng chỉ cần 1 testcase.

+ Boundary Value Analysis (phân tích giá trị biên): Để mở rộng khả năng tìm lỗi, hỗ trợ cho Equivalent Class.

vd: một tham số đầu vào có một giới hạn biên x, thì
phải test ít nhất 4 trường hợp:

- 1: giá trị đầu vào bất kì cách xa x
- 2: giá trị đầu vào ngay trên x
- 3: giá trị đầu vào ngay dưới x
- 4: giá trị đầu vào đúng bằng x

+ Functional Testing: Những dữ liệu kiểm tra sẽ phụ thuộc vào chức năng của code artifact. Sau khi xác định được chức năng của code artifact (vd: code artifact có chức năng xác định có phải số nguyên tố hay không?), tập dữ liệu kiểm tra được sử dụng để kiểm tra các chức năng 1 cách độc lập nhau.

93. Kiểm thử đơn vị đối tượng là gì? Ai thực hiện. Các phương pháp và kỹ thuật nào được sử dụng? Kiểm tra những loại lỗi nào?

- Kiểm thử đơn vị (Unit testing) là việc kiểm thử của lập trình viên sau khi hoàn thành xong 1 code artifact. Sau khi kiểm thử đơn vị hoàn tất, code artifact được gửi cho SQA để tiếp tục kiểm tra.
- Có 2 kỹ thuật cho phép kiểm tra: Hộp trắng & Hộp đen. Như đã mô tả ở trên.

94. Các chiến lược nào sử dụng trong kiểm thử tích hợp? Ưu điểm và hạn chế mỗi loại?

- Kiểm thử tích hợp được sử dụng để kiểm tra 1 code artifact khi kết hợp vào những phần đã được tích hợp rồi xem có lỗi hay không.

- Kiểm tra code artifact sử dụng unit testing như ở câu 93, sau đó kiểm tra xem sau khi tích hợp vào, sản phẩm mới có bị lỗi/sai sót hay không?
(pg 535)

95. Giải thích khái nhiệm stub và driver? Chúng được sử dụng ở đâu và vì sao?

- Stub: Trong quá trình tích hợp các artifact với nhau, có những artifact(giả sử A) cần những artfact khác (vd: B,C,D) để có thể chạy được. Khi đó, B,C,D được gọi là Stub.
- Driver: Trong ví dụ trên, để kiểm tra artifact A ta cần 1 Driver và 3 Stub là B,C,D. (Giả sử) nếu B có thể tự chạy được mà k cần artifact nào, Có thể kiểm tra B với 1 Driver và 0 Stub.
- Chúng được sử dụng trong quá trình tích hợp các artifact với nhau, quá trình test tích hợp để tạo ra sản phẩm hoàn chỉnh.

96. Kiểm thử hệ thống nhằm kiểm tra cái gì? Ai thực hiện? Các phương pháp?

- (Product Testing): Diễn ra sau khi quá trình kiểm tra tích hợp được hoàn tất. Mục đích bảo đảm phần mềm hoàn tất. Thực hiện bởi SQA Group. Được thực hiện trên 2 loại phần mềm Cost Of The Shelf software và Custom Software.
- Với COTS Software, sẽ có 2 giai đoạn là alpha và beta. nhiều khách hàng sẽ được chọn để thử nghiệm và gửi feedback lại.
- Với Custom Software, SQL Group phải bảo đảm rằng Acceptance Test (bảo đảm đúng theo đặc tả) phải hiệu quả. Nếu không developer sẽ bị mất uy tín.
- Kiểm tra bao gồm các phương pháp:
    + Black Box
    + Robustness of Software.
    + Phải thỏa mãn tất cả các ràng buộc của phần mềm.
    + Kiểm tra toàn bộ code + tài liệu gửi đến cho khách hàng.

97. Trình bày các kiểm thử được thực hiện trong kiểm thử hệ thống?

- Giống câu trên.

98. Kiểm thử chấp nhận là gì? Trong đó có những kiểm thử nào được thực hiện? Phân biệt.

- (Acceptance Test) là kiểm tra xem phần mềm đã hoạt động đúng như đặc tả đưa ra bởi developer hay chưa. Acceptance Test được thực hiện trên dữ liệu thực tế. Được thực hiện bởi 1 trong 3 nhóm đối tượng: đại diện khách hàng và nhóm SQA, hoặc nhóm SQA do khách hàng thuê.
- Có các kiểu kiểm thử sau:
    + Kiểm thử đúng (Correctness testing)
    + Hiệu năng (Performance) (tùy chọn)
    + Robustness
    + Tài liệu (Documentation)

99. Mô hình CMM là gì? Có những mức tăng trưởng nào trong mô hình CMM? Nội dung của mỗi mức?

- (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).
- Các mức tăng trưởng trong mô hình CMM: 5 mức

    + Lv1 : Initial Level: không có công nghệ phần mềm được áp dụng ở đây. Mọi thứ đều thực hiện 1 cách rất cơ bản. Tuy nhiên do thiếu kinh nghiệm nên nhìn chung giá thành và thời gian sẽ bị đội lên.

    + Lv2: Repeatable Level: Ứng dụng những kỹ năng công nghệ phần mềm cơ bản vào công việc. Có sự quản lý về thời gian và giá thành cho dự án. Khi có sự cố nhờ có sự tính toán nên có thể kiểm soát được tình hình.

    + Lv3: Defined Level: Quy trình phần mềm đã có đầy đủ tài liệu. Kỹ thuật quản lý cũng như hỗ trợ kỹ thuật được định nghĩa rõ ràng. Và có sự cố gắng tối đa để cải thiện quy trình dự án. Trong level này, giới thiệu công nghệ mới chỉ làm cho công việc bị gián đoạn và k có lợi.

    + Lv4: (Managed Level) Từ mức này rất ít công ty có thể đạt đến. Đặt ra chất lượng và năng suất mục tiêu cho từng dự án. 2 yếu tố trên được xác định

thường xuyên và thực hiện các biện pháp cần thiết để bảo đảm mục tiêu. Sử dụng quản lý chất lượng thống kê (Statistic Quality Control).

+ Lv5: (Optimizing Level): Tiếp tục cải thiện quy trình phần mềm. kỹ thuật quản lý quy trình và chất lượng thống kê (Statistical Quality & Process Control Techniques) được ứng dụng để định hướng tổ chức. Kinh nghiệm cho dự án trước phục vụ cho dự án sau. Và có sự tiếp thu phản hồi ý kiến khách hàng, cải thiện năng suất & chất lượng.

100. Làm thế nào để một tổ chức đạt được các mức tăng trưởng trong CMM? Đâu là giải
pháp và thước đo về các mức tăng trưởng?

- Lv1: Không cần làm gì cũng được rồi.
- Lv2:
    + Quản lý lấy yêu cầu
    + Kế hoạch cho dự án phần mềm
    + Theo dõi và có tầm nhìn xa với dự án phần mềm.
    + Quản lý hợp đồng phần mềm.
    + SQA - bảo đảm chất lượng phần mềm.
    + Software Configuration Management.
- Lv3:
    + Tập trung vào quy trình của tổ chức
    + Định nghĩa các quy trình của tổ chức.
    + Chương trình huấn luyện
    + Quản lý phần mềm tích hợp.
    + Công nghệ dự án phần mềm.
    + Phối hợp liên nhóm
    + Peer Reviews.
- Lv4:
    + Quản lý định lượng quy trình.
    + Quản lý chất lượng phần mềm.
- Lv5:
    + Ngăn chặn lỗi.
    + Quản lý thay đổi công nghệ
    + Quản lý thay đổi quy trình

101. Nêu các chuẩn quốc tế về phần mềm. Trình bày sự khác nhau giữa mô hình CMM và
các chuẩn đó.

- ISO 9001, ISO 9000-3, ISO/IEC 15504, SPICE
- Điểm khác nhau:

+ ISO 9000: Công việc làm tài liệu phải sử dụng cả hình ảnh lẫn ngôn từ để làm rõ ý nghĩa. Không bảo đảm chất lượng phần mềm tốt, nhưng giảm được rủi ro phần mềm chất lượng tồi.