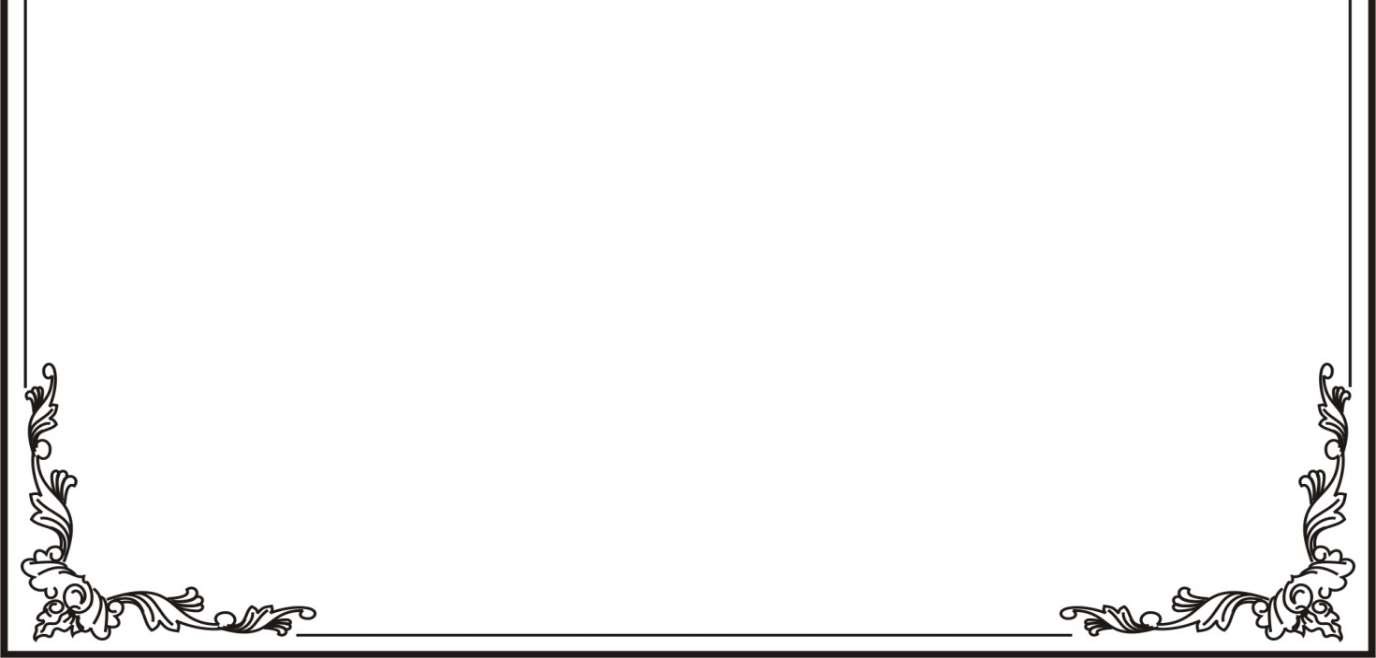
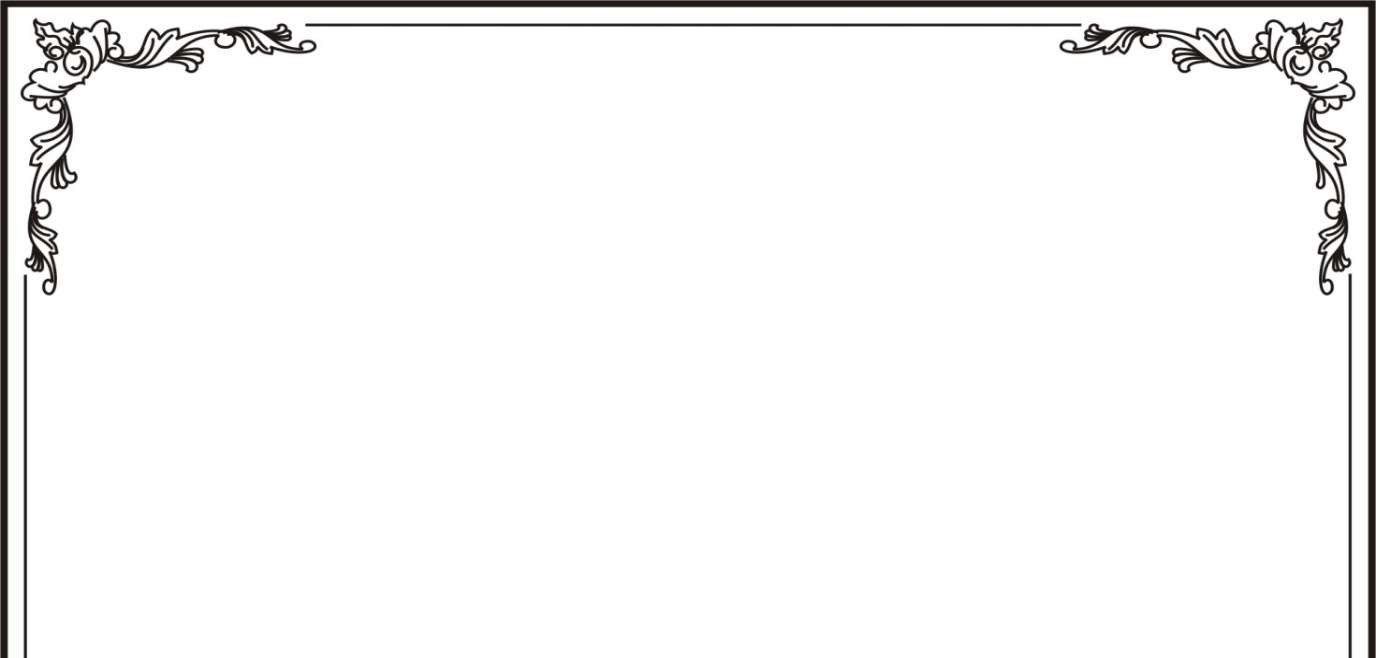
**Giảng viên : Đỗ Thị Bích Ngọc**

**Nhóm bài tập lớn : 10**



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

**Môn học: NHẬP MÔN CÔNG NGHỆ PHẦN MỀM**

**TÀI LIỆU**

**TRẢ LỜI CÂU HỎI**

**Nhóm Lớp : 06**

**Sinh viên :**

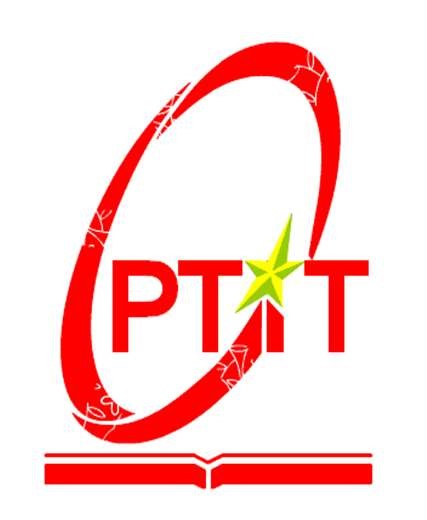
**Nguyễn Mạnh Cường – B19DCCN084**

**Nguyễn Quý Dương – B19DCCN156**

**Nguyễn Trung Kiên – B19DCCN346**

**Đinh Tuấn Tỉnh – B19DCCN585**

Hà Nội – 2022



Mục lục

[**Câu 1. Thế nào là corrective maintenance?** 6](#_Toc99270353)

[**Câu 2. Thế nào là apdaptive maintenance?** 6](#_Toc99270354)

[**Câu 3. Thế nào là perfective maintenance?** 6](#_Toc99270355)

[**Câu 4. Thế nào là refactoring?** 6](#_Toc99270356)

[**Câu 5. Thế nào là “from scratch”?** 6](#_Toc99270357)

[**Câu 6. Thế nào là moving target problem?** 6](#_Toc99270358)

[**Câu 7. Thế nào là regession fault?** 6](#_Toc99270359)

[**Câu 8. Thế nào là một episode?** 7](#_Toc99270360)

[**Câu 9. Thế nào là một iteration?** 7](#_Toc99270361)

[**Câu 10. Thế nào là một increasement?** 7](#_Toc99270362)

[**Câu 11. Thế nào là một artifact?** 7](#_Toc99270363)

[**Câu 12. Thế nào là một portability?** 7](#_Toc99270364)

[**Câu 13. Thế nào là reusebility?** 8](#_Toc99270365)

[**Câu 14. Thế nào là milestone?** 8](#_Toc99270366)

[**Câu 15. Thế nào là một story?** 8](#_Toc99270367)

[**Câu 16. Thế nào là refactoring?** 8](#_Toc99270368)

[**Câu 17. Thế nào là concept exploration?** 8](#_Toc99270369)

[**Câu 18. Thế nào là business model?** 9](#_Toc99270370)

[**Câu 19. Thế nào là traceability?** 9](#_Toc99270371)

[**Câu 20. Thế nào là egoless programming?** 9](#_Toc99270372)

[**Câu 21. Thế nào là PM?** 9](#_Toc99270373)

[**Câu 22. Thế nào là technical leader?** 9](#_Toc99270374)

[**Câu 23. Thế nào là programming secrectary?** 10](#_Toc99270375)

[**Câu 24. Thế nào là backup programmer?** 10](#_Toc99270376)

[**Câu 25: Thế nào là superprogrammer?** 10](#_Toc99270377)

[**Câu 26: Thế nào là một bản thiết kế còn ommision?** 10](#_Toc99270378)

[**Câu 27: Thế nào là một bản thiết kế còn contradiction?** 10](#_Toc99270379)

[**Câu 28: Thế nào là một phần mềm COTS?** 10](#_Toc99270380)

[**Câu 29: Thế nào là SPMP?** 10](#_Toc99270381)

[**Câu 30: Thế nào là alpha release?** 11](#_Toc99270382)

[**Câu 31: Thế nào là beta release?** 11](#_Toc99270383)

[**Câu 32: Thế nào là process?** 11](#_Toc99270384)

[**Câu 33: Thế nào là workflow?** 11](#_Toc99270385)

[**Câu 34: Luật Miller trong CNPM nói gì?** 11](#_Toc99270386)

[**Câu 35: Luật Brooks trong CNPM nói gì?** 11](#_Toc99270387)

[**Cây 36: Luật Dijkstra trong CNPM nói gì?** 11](#_Toc99270388)

[**Câu 37: Verification và Validation (V&V) là gì?** 11](#_Toc99270389)

[**Câu 38: Thế nào là inspection?** 12](#_Toc99270390)

[**Câu 39: Thế nào walkthrough?** 12](#_Toc99270391)

[**Câu 40: Thế nào là một moderator trong nhóm inspection?** 12](#_Toc99270392)

[**Câu 41: Thế nào là một recorder trong nhóm inspection?** 13](#_Toc99270393)

[**Câu 42: Mô hình CMM là gì?** 13](#_Toc99270394)

[**Câu 43: Thế nào là test performance?** 13](#_Toc99270395)

[**Câu 44: Thế nào là test robustness?** 13](#_Toc99270396)

[**Câu 45: Thế nào là cone of uncertainty?** 14](#_Toc99270397)

[**Câu 46: Thế nào là norminal effort?** 14](#_Toc99270398)

[**Câu 47: Thế nào là phần mềm organic?** 14](#_Toc99270399)

[**Câu 48: Thế nào là phần mềm embedded?** 14](#_Toc99270400)

[**Câu 49. Thế nào là phần mềm semi-detached?** 14](#_Toc99270401)

[**Câu 50. Thế nào là TCF?** 15](#_Toc99270402)

[**Câu 51. Thế nào là UFP?** 15](#_Toc99270403)

[**Câu 52. Thế nào là flow trong FFP?** 15](#_Toc99270404)

[**Câu 53. Thế nào là process trong FFP?** 15](#_Toc99270405)

[**Câu 54. Tại sao không có pha kiểm thử?** 16](#_Toc99270406)

[**Câu 55. Tại sao không có pha làm tài liệu?** 16](#_Toc99270407)

[**Câu 56. Tại sao không có pha lập kế hoạch?** 16](#_Toc99270408)

[**Câu 57. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao?** 16](#_Toc99270409)

[**Câu 58. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?** 16](#_Toc99270410)

[**Câu 59. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?** 17](#_Toc99270411)

[**Câu 60. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?** 17](#_Toc99270412)

[**61. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?** 18](#_Toc99270413)

[**Câu 62. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?** 18](#_Toc99270414)

[**Câu 63. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?** 18](#_Toc99270415)

[**Câu 64. Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng thì phù hợp với những dự án có đặc điểm gì?** 19](#_Toc99270416)

[**Câu 65. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu xoắn ốc?** 19](#_Toc99270417)

[**Câu 66. Mô hình vòng đời phần mềm kiểu xoắn ốc thì phù hợp với những dự án có đặc điểm gì?** 19](#_Toc99270418)

[**Câu 67. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu tiến trình linh hoạt?** 19](#_Toc99270419)

[**Câu 68. Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt thì phù hợp với những dự án có đặc điểm gì?** 20](#_Toc99270420)

[**Câu 69. Trong mô hình tiến trình liên hoạt, luôn có đại diện của khác hàng trong nhóm phát triển thì có ưu điểm gì?** 20](#_Toc99270421)

[**Câu 70. Nêu ưu điểm, nhược điểm của mô hình nhóm code bình đẳng?** 20](#_Toc99270422)

[**Câu 71. Mô hình nhóm code bình đẳng thì phù hợp với những dự án có đặc điểm gì?** 21](#_Toc99270423)

[**Câu 72. Nêu ưu điểm, nhược điểm của mô hình nhóm code có chef?** 21](#_Toc99270424)

[**Câu 73. Mô hình nhóm code có chef thì phù hợp với những dự án có đặc điểm gì?** 21](#_Toc99270425)

[**Câu 74. Nêu ưu điểm, nhược điểm của kĩ thuật pair programming?** 21](#_Toc99270426)

[**Câu 75. Kĩ thuật pair programming thì phù hợp với những dự án có đặc điểm gì?** 22](#_Toc99270427)

[**Câu 76. Nêu ưu điểm, nhược điểm của kĩ thuật time boxing?** 22](#_Toc99270428)

[**Câu 77. Nêu ưu điểm, nhược điểm của kĩ thuật stand up meeting?** 22](#_Toc99270429)

[**Câu 78. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC?** 22](#_Toc99270430)

[**Câu 79. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP?** 22](#_Toc99270431)

[**Câu 80. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point?** 23](#_Toc99270432)

[**Câu 81. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO? COCOMO trung gian (Intermediate COCOMO)** 23](#_Toc99270433)

[**Câu 82. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả?** 24](#_Toc99270434)

[**Câu 83. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo?** 24](#_Toc99270435)

[**Câu 84. Nếu nhóm SQA phát hiện ra ít lỗi, thì có thể kết luận nhóm code giỏi hay nhóm SQA kém? Tại sao?** 24](#_Toc99270436)

[**Câu 85. Tại sao nói inspection và walkthrough là hướng tài liệu, mà không phải hướng vào người tham gia?** 24](#_Toc99270437)

[**Câu 86. Quality assurance thì khác gì với testing?** 24](#_Toc99270438)

[**Câu 87. Tại sao nói function point chịu ảnh hưởng chủ quan của các chuyên gia?** 25](#_Toc99270439)

[**Câu 88. COCOMO tính đến nhiều tiêu chí hơn hay là function point? Giải thích?** 25](#_Toc99270440)

[**Câu 89. SW development multiplier của COCOMO thì khác gì TCF của function point?** 25](#_Toc99270441)

[**Câu 90. TCF của function point thì khác gì hằng số b của FFP?** 26](#_Toc99270442)

[**Câu 91. Tại sao nguyên lí Djistra lại đúng?** 26](#_Toc99270443)

[**Câu 92. Tại sao luật Brook lại đúng?** 26](#_Toc99270444)

[**Câu 93. Người ta áp dụng luật Miller trong CNPM như thế nào?** 26](#_Toc99270445)

[**Câu 94. Phát triển phần mềm thì khác gì sản xuất phần mềm?** 26](#_Toc99270446)

[**Câu 95. Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại test gì?** 26](#_Toc99270447)

- 95 câu lí thuyết-

**Câu 1. Thế nào là corrective maintenance?**

Corrective maintenance là loại bỏ các lỗi còn sót lại trong khi vẫn giữ nguyên các thông số kỹ thuật, cũng như nâng cao (hoặc cập nhật phần mềm), bao gồm các thay đổi đối với các thông số kỹ thuật và việc thực hiện các thay đổi đó.

**Câu 2. Thế nào là apdaptive maintenance?**

Adaptive maintenance là thay đổi được thực hiện để đáp ứng với những thay đổi trong môi trường mà sản phẩm hoạt động, chẳng hạn như phần cứng / hệ điều hành mới hoặc các quy định mới của chính phủ.

**Câu 3. Thế nào là perfective maintenance?**

Perfective maintenance là những thay đổi mà khách hàng cho rằng sẽ cải thiện hiệu quả của sản phẩm, chẳng hạn như chức năng bổ sung hoặc giảm thời gian phản hồi.

**Câu 4. Thế nào là refactoring?**

Refactoring 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 lại.

**Câu 5. Thế nào là “from scratch”?**

“from scratch” có nghĩa là làm từ đầu, bắt đầu từ con số 0. Không dựa trên một cơ sở hay là một phần mềm có sẵn nào từ trước mà là làm mới hoàn toàn.

**Câu 6. Thế nào là moving target problem?**

Moving target problem là những yêu cầu thay đổi của khách hàng về sản phẩm. Điều này không chỉ gây khó chịu cho nhóm phát triển, những thay đổi thường xuyên có thể dẫn đến một sản phẩm được xây dựng kém. Ngoài ra, những thay đổi như vậy làm tăng thêm giá thành của sản phẩm.

**Câu 7. Thế nào là regession fault?**

Regession fault là lỗi vô tình được đưa vào một bộ phận của sản phẩm do hậu quả của việc thay đổi, rõ ràng không liên quan đến một phần khác của sản phẩm.

**Câu 8. Thế nào là một episode?**

Một episode là một giai đoạn phát triển của phần mềm, mỗi một giai đoạn thì sẽ trải qua những quy trình sản xuất và thiết kế đặc trưng riêng.

**Câu 9. Thế nào là một iteration?**

Iteration là một khía cạnh tồn tại bên trong của kĩ thuật phần mềm, và các mô hình vòng đời lặp đi lặp lại đã được sử dụng trong hơn 30 năm. Trong thực tế, lặp đi lặp lại và tăng dần được sử dụng kết hợp với nhau. Đó là, một hiện vật được xây dựng từng phần (tăng dần) và mỗi phần gia tăng đi qua nhiều phiên bản (lặp lại).

**Câu 10. Thế nào là một increasement?(sự gia tăng)**

Increasement là một khía cạnh tồn tại bên trong của kỹ thuật phần mềm, bắt đầu ở đầu Phần tăng A và tiếp tục cho đến khi kết thúc phần tăng dần D. Sau đó, sản phẩm phần mềm hoàn chỉnh sẽ được cài đặt trên máy tính của khách hàng.

Phần tăng dần A tương ứng với episode 1: thiết lập phiên bản đầu tiên của phần mềm

Phần tăng B tương ứng với episode 2: chỉ bao gồm quy trình thực hiện

Phần tăng C tương ứng với episode 3: chỉ bao gồm quy trình thiết kế và quy trình thực hiện.

Phần tăng D tương ứng với episode 4: các yêu cầu được thay đổi để nâng cao độ chính xác tổng thể, sau đó các thay đổi tương ứng được thực hiện đối với quy trình phân tích, thiết kế và thực hiện.

**Câu 11. Thế nào là một artifact?**

Một artifact đóng vai trò giúp mô tả chức năng, là môt thành phần của sản phẩm phần mềm, chẳng hạn như tài liệu đặc tả, mã modun hoặc sách hướng dẫn.

**Câu 12. Thế nào là một portability?**

Chi phí ngày càng tăng của phần mềm khiến nó bắt buộc phải tìm thấy một số phương tiện để chứa chi phí. Một cách là đảm bảo rằng toàn bộ sản phẩm có thể được điều chỉnh dễ dàng để chạy trên nhiều loại kết hợp phần cứng-hệ điều hành khác nhau. Sau đó, một số chi phí triển khai sản phẩm có thể được hoàn lại bằng cách bán các phiên bản chạy trên máy tính khác. Tuy nhiên, lý do quan trọng nhất để phát triển phần mềm có thể được triển khai dễ dàng trên các máy tính khác, cứ sau 4 năm hoặc lâu hơn, tổ chức khách hàng mua phần cứng mới và tất cả phần mềm của nó sau đó phải được chuyển đổi để chạy trên phần cứng mới.

**Câu 13. Thế nào là reusebility?**

Tái sử dụng đề cập đến việc sử dụng các thành phần của một sản phẩm để tạo điều kiện phát triển một sản phẩm khác với chức năng khác. Thành phần có thể tái sử dụng không nhất thiết phải là mô-đun, một lớp hoặc một đoạn mã — nó có thể là một thiết kế, một phần của sách hướng dẫn, một tập hợp các bài kiểm tra dữ liệu, hợp đồng hoặc thời hạn và ước tính chi phí.

**Câu 14. Thế nào là milestone?**

Một khía cạnh quan trọng của kế hoạch liên quan đến việc hoàn thành các sản phẩm công việc. Ngày mà một sản phẩm công việc được coi là hoàn thành được coi là một cột mốc quan trọng. Để xác định xem một tác phẩm sản phẩm thực sự đã đạt đến một cột mốc quan trọng, trước tiên sản phẩm đó phải vượt qua một loạt các bài đánh giá được thực hiện bởi các thành viên trong nhóm, quản lý hoặc khách hàng.

**Câu 15. Thế nào là một story?**

Story là tính năng. Lập trình cực đoan là một cách tiếp cận mới gây tranh cãi về phát triển phần mềm dựa trên mô hình lặp đi lặp lại và tăng dần. Bước đầu tiên là bước mà các nhóm phát triển phần mềm xác định các tính năng (stories) khác nhau mà khách hàng muốn sản phẩm hỗ trợ. Đối với mỗi tính năng như vậy, nhóm sẽ thông báo cho khách hàng mất bao lâu để triển khai tính năng đó và chi phí bao nhiêu. Bước đầu tiên này tương ứng với các yêu cầu và quy trình phân tích của quá trình lặp lại và tăng dần.

**Câu 16. Thế nào là refactoring?**

Không có bước thiết kế tổng thể trước khi xây dựng các công trình khác nhau. Thay vào đó, thiết kế được sửa đổi trong khi sản phẩm đang được chế tạo. Thủ tục này được gọi là tái cấu trúc. Bất cứ khi nào một trường hợp thử nghiệm không chạy, mã sẽ được tổ chức lại cho đến khi nhóm hài lòng rằng thiết kế đơn giản, dễ hiểu và chạy tốt tất cả các trường hợp thử nghiệm.

**Câu 17. Thế nào là concept exploration?**

Cuộc điều tra sơ bộ về nhu cầu của khách hàng đôi khi được gọi là thăm dò ý tưởng (concept exploration). Trong các cuộc khảo sát giữa các thành viên của nhóm phát triển và nhóm khách hàng thì chức năng được đề xuất của sản phẩm liên tục được cập nhập lại và phân tích tính khả thi về kỹ thuật và tính khả thi về tài chính.

**Câu 18. Thế nào là business model?**

Business model là một mô tả về các quá trình kinh doanh của một tổ chức. Mô hình kinh doanh cung cấp thông tin về toàn bộ hoạt động kinh doanh của khách hàng. Với những thông tin này các nhà phát triển có thể tư vấn cho khách hàng về những phần nào trong công việc kinh doanh của khách hàng cần tin học hóa. Ngoài ra, nếu nhiệm vụ là mở rộng một sản phẩm phần mềm hiện có, các nhà phát triển phải hiểu doanh nghiệp hiện tại nói chung để xác định cách kết hợp tiện ích mở rộng và tìm hiểu những bộ phận nào, nếu có, của sản phẩm hiện có cần được chỉnh sửa để thêm phần mới.

**Câu 19. Thế nào là traceability?**

Nếu các artifacts yêu cầu có thể được kiểm tra qua những vòng đời của sản phẩm phần mềm, khi đó một tài sản mà họ phải có là khả năng truy xuất nguồn gốc (traceability). Nó phải có khả năng theo dõi mọi mục trong phân tích artifacts trở lại artifacts yêu cầu và tương tự đối với hiện vật thiết kế và hiện vật thực hiện. Nếu các yêu cầu đã được trình bày một cách có phương pháp, được đánh số thích hợp, được tham chiếu chéo và được lập chỉ mục, sau đó các nhà phát triển nên có một chút khó khăn khác nhau trong việc truy tìm các artifacts tiếp theo và đảm bảo rằng chúng thực sự phản ánh đúng yêu cầu của khách hàng.

**Câu 20. Thế nào là egoless programming?**

Egoless programming là phương pháp tổ chức các nhóm lập trình, ở đó mỗi thành viên của nhóm phải hiểu và có trách nhiệm với từng chương trình con trong hệ thống mà không phụ thuộc là ai sẽ lập chương trình nào trong hệ thống đó. Lập trình viên gắn bó cao với mã lệnh của họ. Các modun như là sự mở rộng của chính bản thân, khó phát hiện lỗi.

**Câu 21. Thế nào là PM?**

PM-Project Manager: người quản lí dự án, là người được chỉ định bởi các tổ chức, công ty để lãnh đạo một dự án, người phải chịu trách nhiệm hoàn thành các mục tiêu đề ra từ khi bắt đầu đến khi thực hiện và kết thúc. Trách nhiệm quản lý dự án chính bao gồm xác định và truyền đạt các mục tiêu dự án rõ ràng, hữu ích và có thể đạt được các yêu cầu của dự án(lực lượng lao động, thông tin cần thiết, các thỏa thuận khác nhau và vật liệu hoặc công nghệ) cần thiết để hoàn thành các mục tiêu của dự án.

**Câu 22. Thế nào là technical leader?(người đứng đầu về chuyên môn, kĩ thuật)**

Technical leader đảm nhiệm công việc giám sát một nhóm tập trung vào các vấn đề kỹ thuật, bao gồm phát triển phần mềm, phát hành sản phẩm và các nhiệm vụ kỹ thuật khác. Đôi lúc, technical leader cũng hợp tác với các nhân viên để xác định và sửa chữa các vấn đề kỹ thuật. Họ thường đánh giá các quy trình làm việc của nhóm và thực tiễn tốt nhất và thực hiện các thay đổi để hợp lý hóa các hoạt động.

**Câu 23. Thế nào là programming secrectary?**

Programming secrectary là thành viên có tay nghề cao, được trả lương cao trong đội ngũ lập trình viên. Programming secrectary chịu trách nhiệm duy trì thư viện sản xuất dự án, tài liệu của dự án. Danh sách mã nguồn bao gồm JCL và dữ liệu thử nghiệm. Các lập trình viên giao mã nguồn cho thư ký, người chịu trách nhiệm về việc chuyển đổi nó sang dạng máy có thể đọc được, biên dịch, liên kết, tải, thực thi và chạy các trường hợp thử nghiệm.

**Câu 24. Thế nào là backup programmer?**

Backup programmer là người có thể đảm nhận và thay thế Chief programmer (lập trình viên trưởng) khi lập trình viên trưởng không thể trực tiếp điều hành công việc hoặc có vấn đề về sức khỏe. Vì vậy backup programmer cũng phải có năng lực về mọi mặt, và để giải phóng cho chief programmer chỉ cần tập trung vào thiết kết kiến trúc thì backup programmer đã lập kế hoạch cho mỗi trường hợp kiểm thử hộp đen và các nhiệm vụ khác độc lập với thiết kế quy trình.

**Câu 25. Thế nào là super programmer?**

Super programmer - một lập trình viên có kết quả đầu ra gấp nhiều lần so với một lập trình viên giỏi trung bình.

**Câu 26. 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.

**Câu 27. 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.

**Câu 28. Thế nào là một phần mềm COTS?**

Phần mềm COTS : 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,...).

**Câu 29. Thế nào là SPMP?**

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 biếu chi tiết, ngân sách, phân bổ tài nguyên, ước lượng thời gian và chi phí.

**Câu 30. Thế nào là alpha 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.

**Câu 31. Thế nào là beta release?**

Beta 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).

**Câu 32. Thế nào là process?**

Process(Tiến trình) Là phương pháp, cách thức sản xuất ra phần mềm.

**Câu 33. Thế nào là workflow?**

Workflow: là các định nghĩa của các qui trình đã chuẩn hóa. Và khi mình viết các module cho từng công việc, workflow là một chuỗi công việc phải làm.

**Câu 34. 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)

**Câu 35. 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.

**Câu 36. 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.

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

Verification 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.   
 Validation là một 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.

**Câu 38. Thế nào là inspection?**

Team Inspection thường gồm 4 người. Vd cho Inspect design gồm có: Moderator,   
designer, implementer, tester. là 1 quy trình gồm 5 bước:   
 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.   
 Bước 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.   
 Bước 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.

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

Bước 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).

**Câu 39. Thế nào walkthrough?**

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.

**Câu 40. Thế nào là một moderator trong nhóm inspection?**

Moderator: là người điều hành quá trình review, xác định sử dụng loại hình review nào, thực hiện kiểm tra đầu vào và theo dõi quá trình làm lại để kiểm soát chất lượng đầu vào và đầu ra của quá trình xem xét

Moderator có trách nhiệm đảm bảo rằng tất cả các lỗi được ghi lại, đề xuất cải tiến quy trình và yêu cầu thay đổi đều đã được cập nhật, sửa chữa.

Moderator tập hợp các số liệu cần thiết và đánh giá nếu các tiêu chí đầu ra được đáp ứng để đóng các hoạt động review.

**Câu 41. Thế nào là một recorder trong nhóm inspection?**

Recorder: là người ghi chép lại từng lỗi được đề cập và ghi lại các đề xuất để cải thiện quy trình trong cuộc họp đánh giá.

**Câu 42. Mô hình CMM là gì?**

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).

Các mức tăng trưởng trong mô hình CMM: 5 mức

+ Lv1: Initial Level

+ Lv2: Repeatable Level

+ Lv3: Defined Level

+ Lv4: (Managed Level)

+ Lv5: (Optimizing Level)

**Câu 43. Thế nào là test performance?**

Performance là một khía cạnh khác của sản phẩm phải được kiểm tra. Ví dụ, điều cần thiết là phải biết mức độ mà sản phẩm đáp ứng được các hạn chế của nó liên quan đến các yêu cầu về thời gian hoặc không gian yêu cầu. Đối với một hệ thống máy tính nhúng chẳng hạn như một máy tính tích hợp trong tên lửa phòng không cầm tay, các hạn chế về không gian của hệ thống có thể là chỉ 128 megabyte (MB) bộ nhớ chính có sẵn cho phần mềm. Cho dù phần mềm có xuất sắc đến đâu, nhưng nếu nó cần 256 MB bộ nhớ chính, thì nó không thể sử dụng được.

**Câu 44. Thế nào là test robustness?**

Test robustness : tức là, cố ý gửi dữ liệu đầu vào có lỗi để xác định xem sản phẩm có bị lỗi hay không hoặc khả năng xử lý lỗi của nó có đủ để xử lý dữ liệu lỗi hay không. Nếu sản phẩm được chạy cùng với phần mềm hiện được cài đặt của khách hàng, thì cũng phải thử nghiệm để kiểm tra xem sản phẩm mới có ảnh hưởng đến các hoạt động của máy tính hiện có của khách hàng không. Cuối cùng, phải kiểm tra xem mã nguồn và tất cả các loại tài liệu khác có đầy đủ và nhất quán hay không.

**Câu 45. Thế nào là cone of uncertainty?**

Cone of uncertainty : là một mô hình , nó mô tả phạm vi ước tính chi phí tương đối cho các khoản công việc khác nhau của life-cycle. Ví dụ 1 sản phẩm khi được hoàn thiện và trải qua được hết các test khi đến tay khách hàng giá của nó là 100 triệu,nếu chúng ta ước lượng chi phí ở giữa pha requirements có thể giá của sản phẩm năm trong khoảng 20 – 400 triệu, nếu chúng ta lại ước lượng ở giữa pha analysis có thể giá của sản phẩm nằm trong khoảng 50-200 triệu và nếu chúng ta lại ước lượng ở cuối pha analysis có thể giá của sản phẩm nằm trong khoảng 75 – 150 triệu.

**Câu 46. Thế nào là norminal effort?**

Nominal effort (ước lượng ban đầu )- được xác định bằng một phương trình có dạng được sử dụng trong các mô hình biến đơn tĩnh, sử dụng KDLOC làm thước đo kích thước. Để xác định nominal effort đầu Ei tính bằng man/month, phương trình được sử dụng :

Ei = a \* (KDLOC) b

Giá trị của hằng số a và b phụ thuộc vào loại dự án.

**Câu 47. Thế nào là phần mềm organic?**

Một dự án phần mềm được coi là organic nếu quy mô nhóm yêu cầu 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 đó.

**Câu 48. Thế nào là phần mềm embeded?**

Embedded là một hệ thống nhúng được xem là một trong những thuật ngữ IT thuộc một chỉnh thể với khả năng tự trị được các nhúng trong môi trường hoặc hệ thống mẹ. Hệ thống này là một trong những sự kết hợp của các phần cứng và phần mềm khác.

Phần mềm Embedded Software trở thành phần mềm được viết ra để thực hiện những nhiệm vụ cũng như mục đích cụ thể đối với một thiết bị. Hiện nay, các phần mềm Embedded đang không ngừng tương tác với thế giới thực dựa vào nguyên lý nhân các tín hiệu đã được mã hóa ban đầu và thực hiện các hoạt động dựa vào các dữ liệu truyền vào.

**Câu 49. Thế nào là phần mềm semi-detached?**

Một dự án phần mềm được cho là semi-detached nếu các đặc điểm quan trọng như quy mô nhóm, kinh nghiệm, kiến thức về môi trường lập trình khác nhau nằm giữa môi trường hữu cơ và nhúng. Các dự án được phân loại là semi-detached không thường gặp và khó phát triển hơn so với phần mềm organic và yêu cầu nhiều kinh nghiệm hơn, hướng dẫn tốt hơn và sáng tạo. Vd: Các trình biên dịch hoặc các Hệ thống nhúng (Embedded Systems) khác nhau có thể được coi là semi-detached.

**Câu 50. Thế nào là TCF?**

TCF là hệ số phức tạp kỹ thuật (TCF) được tính toán. Đây là một thước đo của ảnh hưởng của 14 yếu tố kỹ thuật, chẳng hạn như tỷ lệ giao dịch cao, tiêu chí hiệu suất (đối với ví dụ, thông lượng hoặc thời gian phản hồi) và cập nhật trực tuyến; tập hợp đầy đủ các yếu tố là được thể hiện trong bảng dưới. Mỗi yếu tố trong số 14 yếu tố này được gán một giá trị từ 0 (“không xuất hiện hoặc không có ảnh hưởng ”) đến 5 (“ ảnh hưởng mạnh xuyên suốt ”). 14 số kết quả được tính tổng, dựa trên phần trăm mức độ ảnh hưởng (DI). TCF sau đó được tính bởi công thức sau: TCF = 0.6 + (DI/100)

Text

Description automatically generated with medium confidence

**Câu 51. Thế nào là UFP?**

UFP (unadjusted function points) : Các điểm không thích ứng, giá trị của UFP

được tính toán dựa trên số lượng các thành phần trong giao dịch.

**Câu 52. Thế nào là flow trong FFT?**

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.

**Câu 53. Thế nào là process trong FFP?**

Process(Tiến trình): Là phương cách sản xuất ra phần mềm.

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.

**Câu 54. Tại sao không có pha kiểm thử?**

Không có pha kiểm thử là bởi việc kiểm thử phần mềm phải được thực hiện xuyên suốt quá trinh phát triển phần mềm từ khi bắt đầu đến khi kết thúc để có thể hạn chế tối đa lỗi ở những pha cuối cùng vì đây là giai đoạn mà việc kiểm thử trở nên vô cùng khó khăn và tốn kém. Điều này dẫn đến việc bàn giao sản phẩm không đúng hạn.

**Câu 55. Tại sao không có pha làm tài liệu?**

Không có pha làm tài liệu là vì:

- Tài liệu bị hoãn lại thì sẽ không bao giờ hoàn thành.

- Cá nhân chịu trách nhiệm trong pha trước có thể chuyển sang bộ phận khác

- Sản phẩm thường xuyên thay đổi dựa trên yêu cầu của khách hàng khi phát triển nên cần tài liệu để ghi lại

**Câu 56. Tại sao không có pha lập kế hoạch?**

Chúng ta không 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.

**Câu 57. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao?**

Nếu không áp dụng các mô hình vòng đời thì rất khó để phát triển phần mềm. Vì những lý do sau:

* 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 do có nhiều lỗi phát sinh
* Chậm tiến độ bàn giao sản phẩm.

**Câu 58. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?**

Phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm là vì mỗi mô hình đều có những ưu, nhược điểm riêng của nó và 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.

**Câu 59. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?**

* Ưu điểm:
* Dễ dàng triển khai và quy trình dễ hiểu.
* Với những dự án nhỏ, mô hình thác nước hoạt động hiệu quả và cho kết quả rất tốt.
* Vì quá trình phát triển rất chặt chẽ, do đó chất lượng mỗi phần và cả dự án sẽ rất chặt chẽ.
* Yêu cầu đầu vào và đầu ra có sự rõ ràng, nên tiến trình làm việc rất dễ dàng và chất lượng.
* Kết quả được ghi chép dễ dàng.
* Nhược điểm:
* Do pha đầu tiên là phân tích yêu cầu, nên sẽ không thể thay đổi yêu cầu ở các pha tiếp theo trong mô hình.
* Mô hình thác nước không phù hợp để triển khai với những dự án dài và phức tạp, có nhiều sự thay đổi trong yêu cầu.
* Khó di chuyển qua lại giữa các pha. Ví dụ, nếu phần mềm chuẩn bị bước vào pha kiểm thử và có sự thay đổi ở yêu cầu thì sẽ rất khó để quay lại và thay đổi. Việc quay lại sẽ ngốn một khoảng thời gian và công sức rất lớn của team phát triển.
* Rủi ro cao hơn. Do quá trình kiểm thử trong mô hình thác nước xuất hiện gần cuối, các chiến thuật để giảm thiểu rủi ro không được triển khai từ sớm. Trong khi ở những mô hình linh hoạt, quá trình kiểm thử diễn ra song song với quá trình phát triển do đó việc giảm thiểu rủi ro hiệu quả hơn rất nhiều.

**Câu 60. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?**

Mô hình vòng đời phần mềm kiểu thác nước phù hợp với những dự án như:

* Dự án ngắn và đơn giản: Như đã trình bày ở trên, các dự án dài và phức tạp thường đòi hỏi các bước phải được thực hiện đồng thời cùng với kiểm thử để tiết kiệm thời gian, trong khi ở mô hình thác nước thì các bước và thứ tự đã cố định, do đó sẽ tốn nhiều thời gian để sửa chữa và hoàn thành hơn.
* Những yêu cầu cố định, không thay đổi thường xuyên và dễ hiểu.
* Tài nguyên phong phú và các thành viên có chuyên môn kỹ thuật tốt.

**Câu 61. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?**

* Ưu điểm:
* Có 1 số ưu điểm giống với mô hình thác nước hạn chế được tối đa việc sửa sản phẩm thông qua bản mẫu.
* Kết quả của bản mẫu có thể ảnh hưởng đến cả qui trình phát triển phần mềm.
* Trong 1 hệ thống phần mềm thì thành phần giao diện và cơ sở dữ liệu là thuận tiện cho việc ra bản vá.
* Nhược điểm:
* Các pha được tiến hành liên tục mà ko đc viết tài liệu.
* Áp dụng cho các mô hình đòi hỏi sự chính xác cao đúng với nhu cầu của khách hàng.

**Câu 62. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?**

Mô hình vòng đời phần mềm kiểu bản mẫu nhanh phù hợp với những dự án:

* Hệ thống chủ yếu dựa trên giao diện người dùng (GUI)
* Khách hàng, nhất là người sử dụng cuối, không thể xác định rõ ràng yêu cầu.

**Câu 63. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?**

* Ưu điểm:
* Xây dựng và hoàn thiện các bước sản phẩm theo từng bước.
* Nhận được phản hồi của người sử dụng từ những bản phác thảo.
* Thời gian làm tài liệu sẽ ít hơn so với thời gian thiết kế.
* Phần mềm làm việc một cách nhanh chóng trong suốt vòng đời phát triền.
* Mô hình này linh hoạt hơn, ít tốn kém hơn để thay đổi phạm vi và yêu cầu.
* Dễ dàng hơn trong việc kiểm tra và sửa lỗi với sự lặp lại nhỏ hơn.
* Nhược điểm:
* Mỗi giai đoạn lặp lại thì cứng nhắc.
* 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.
* Cần một định nghĩa rõ ràng và đầy đủ của toàn bộ hệ thống trước khi nó có thể được chia nhỏ và được xây dựng từng bước.
* Tổng chi phí là cao hơn so với thác nước.

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

Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng phù hợp với những dự án:

* Yêu cầu của hề thống đã hoàn chỉnh, được xác định rõ ràng và dễ hiểu.
* Yêu cầu chính cần được xác định, và một số chi tiết có thể được đổi mới theo thời gian.
* Áp dụng cho những dự án có yêu cầu đã được mô tả, định nghĩa và hiểu một cách rõ ràng.
* Có nhu cầu về sản phẩm sớm.

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

* Ưu điểm:
* Phân tích đánh giá rủi ro được đẩy lên như một phần thiết yếu trong mỗi "spiral" để tăng mức độ tin cậy của dự án.
* Kết hợp những tính chất tốt nhất của mô hình waterfall và tiến hóa.
* Cho phép thay đổi tùy theo điều kiện thực tế dự án tại mỗi "spiral".
* Đây chính là mô hình tổng quát nhất, tất cả các mô hình khác đều có thể xem là một hiện thựccủa mô hình tổng quát này, hay cũng có thể xem nó là mô hình tổng hợp các mô hình khác.
* Đặc biệt, nó được ứng dụng không chỉ trong phát triển phần mềm mà còn trong phát triển phần cứng.
* Nhược điểm:
* Phức tạp và không phù hợp cho dự án nhỏ với ít rủi ro.
* Cần có kỹ năng tốt về phân tích rủi ro.

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

Mô hình vòng đời phần mềm kiểu xoắn ốc phù hợp với những dự án:

* Dự án lớn có nhiều rủi ro hay sự thành công của dự án không có được sự đảm bảo nhất định;những dự án đòi hỏi nhiều tính toán, xử lý như hệ thống hỗ trợ quyết định.
* Đội ngũ thực hiện dự án có khả năng phân tích rủi ro.

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

* Ưu điểm:
* Giảm thời gian cần thiết để tận dụng một số tính năng của hệ thống.
* Kết quả cuối cùng là phần mềm chất lượng cao trong thời gian ít nhất có thể và sự hài lòng của khách hàng.
* Nhược điểm:
* Phụ thuộc vào kỹ năng của người phát triển phần mềm.
* Tài liệu được thực hiện ở giai đoạn sau.
* Cần một team có kinh nghiệm và trinh độ cao.

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

Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt phù hợp với những dự án:

* Nó có thể được sử dụng với bất kỳ loại hình dự án nào, nhưng nó cần sự tham gia và tính tương tác của khách hàng.
* Ngoài ra, nó có thể được sử dụng khi khách hàng yêu cầu chức năng sẵn sàng trong khoảng thời gian ngắn(3 tuần).

**Câu 69. 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 liên hoạt, luôn có đại diện của khác hàng trong nhóm phát triển xuyên suốt quá trinh phát triển phần mềm điều này giúp cho sản phẩm được xây dựng sẽ bám sát với nhu cầu và nghiệp vụ của khách hàng, giúp cho nhóm phát triển dễ dàng nắm bắt yêu cầu cho mỗi module và triển khai các chức năng chính xác hơn. Ngoài ra khách hàng cũng có thể giám sát được quy trình và độ hiệu quả của sản phẩm.

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

* 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.
* 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.

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

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

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

* 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 trinh viên thường rất dị ứng với công việc giấy tờ, do đó để tìm người lập trinh chỉ đơn thuần làm công việc giấy tờ là rất khó.

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

Mô hình có Chief 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.

**Câu 74. Nêu ưu điểm, 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.

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

Kỹ thuật pair programming là một giải pháp tuyệt vời bất cứ khi nào bạn cần thực hiện một dự án phức tạp, tăng tốc một dự án quan trọng, đào tạo một nhà phát triển mới, trợ giúp hoặc được một nhà phát triển khác giúp đỡ về một vấn đề kỹ thuật, vượt qua nút thắt cổ chai phụ thuộc trong nước rút, chia sẻ kiến ​​thức trong nhóm phát triển hoặc thực hiện một cuộc phỏng vấn kỹ thuật.

**Câu 76. Nêu ưu điểm, nhược điểm của kỹ 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.

- Quản lý 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.

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

● Ưu điểm:

- Giúp kết nối và đáp ứng yêu cầu của khách hàng nhanh nhất có thể.

● Nhược điểm:

- Chỉ phù hợp với các dự án nhỏ.

**Câu 78. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC?**

● 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).

- Chỉ phù hợp với những dự án nhỏ, đơn giản.

**Câu 79. 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.

**Câu 80. 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:

- Nó cần những đánh giá chủ quan với rất nhiều nhận định liên quan.

- Ít dữ liệu nghiên cứu hơn về các điểm chức năng so với LOC.

- Nó được thực hiện sau khi tạo ra các thông số kỹ thuật thiết kế.

- Nó có độ chính xác thấp trong việc đánh giá do có liên quan đến một phán đoán chủ quan.

- Vì đường cong học tập khá dài nên không dễ để đạt được năng lực.

- Đó là một phương pháp tốn thời gian.

**Câu 81. Nêu ưu điểm, 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.

**Câu 82. Tại sao trong mô hình tiến trình linh hoạt, không cần 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.

**Câu 83. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của**

**workflow tiếp theo?**

Bởi vì người đại diện cho nhóm workflow tiếp theo sẽ chuyển những thông tin

công việc từ workflow trước về thông tin công việc của mình.

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

Đây là một đánh giá khách quan và chúng ta không thể kết luận nhóm code giỏi hay nhóm SQA kém được bởi vì SQA kiểm tra phần mềm dựa trên yêu cầu của dự án đặt ra nhằm đáp ứng khách hàng. Tuy không thể chắc rằng tương lai phần mềm có lỗi không nhưng hiện tại nếu phát hiện ít lỗi thì không phải do SQA kém. Mặt khác khi phần mềm ngay lần đầu kiểm tra đã ít lỗi thì chúng ta chỉ có thể kết luận nhóm code làm việc tốt.

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

Inspection là một dạng đánh giá ngang hàng được dựa trên kiểm tra trực quan tài liệu để phát hiện khiếm khuyết (defect).

Walkthrough là trình bày từng bước từng bước bởi tác giả của tài liệu để thu thập thông tin và để thiết lập hiểu biết chung về thành phần của nó.

Chính vì thế inspection và walkthrough là hướng tài liệu, mà không phải hướng vào người tham gia.

**Câu 86. Quality assurance thì khác gì với testing?**

Quality Assurance (viết tắt là QA): Bao gồm các hoạt động đảm bảo việc thực hiện các quy trình, thủ tục và tiêu chuẩn trong ngữ cảnh để xác minh của phần mềm phát triển và yêu cầu dự định.

Testing: Bao gồm các hoạt động đảm bảo việc xác định các bug/error/defects trong 1 phần mềm.

So sánh:

=>Quality assurance:

* Tập hợp con của SDLC.
* Định hướng quy trình.
* Đảm bảo rằng các quy trình và thủ tục được thực hiện để đạt được chất lượng.
* Tập trung vào quá trình để đạt được chất lượng yêu cầu.
* Ngăn ngừa khuyết tật.
* Phương pháp tiếp cận toàn đội.
* Quy trình chủ động.

=>Testing:

* Tập con của QC.
* Sản phẩm được định hướng.
* Xác thực sản phẩm dựa trên thông số kỹ thuật.
* Tập trung vào thử nghiệm thực tế của sản phẩm.
* Tìm và sửa lỗi.
* Quy trình phản ứng.
* Nhóm kiểm tra.

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

Bởi vì:

* Giống như FFP, bảo trì có thể được đo một cách thiếu chính xác.
* Có thể thay đổi.

o Số lượng các tệp, luồng và các tiến trình.

o Số lượng đầu vào, đầu ra, câu hỏi, các tệp chính và các giao diện.

* Theo lý thuyết, có thể thay đổi tất cả các dòng mã với việc thay đổi số lượng các dòng mã.

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

COCOMO tính đến nhiều tiêu chí hơn là function point. Nó đòi hỏi các tiêu chí về các thuộc tính của sản phẩm, phần cứng, nhân sự và dự án. Còn Function Point chỉ cần có tiêu chí về đo lường kích thước chức năng.

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

Hiệu năng của COCOMO chính xác còn TCF của function point thì giống với FFP hiệu năng không cao và đôi khi thiếu chính xác.

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

TCF có hiệu năng, năng suất ảnh hưởng ở mức tổng thể.

Hằng số b có hiệu năng hoặc năng suất thay đổi từ tổ chức này đến tổ chức khác.

**Câu 91. Tại sao nguyên lí Dijkstra lại đúng?**

Bởi vì nó rất dễ để chứng minh một phần mềm có lỗi. Và nó luôn hiển thị lỗi hiện

có và chỉ ra lỗi chưa thấy.

**Câu 92. Tại sao luật Brook lại đúng?**

Bởi khi thêm người vào một dự án sẽ mất thời gian bàn giao công việc và để người mới làm quen với dự án sẽ làm cho dự án càng thêm chậm trễ.

**Câu 93. Người ta áp dụng luật Miller trong CNPM như thế nào?**

Bất cứ lúc nào con người cũng có thể tập trung vào 7 + - 2 phần (hạn ngạch thông tin tinh chỉnh từng bước cho phép kỹ sư phần mềm tập trung vào những "phần" phù hợp nhất ở giai đoạn phát triển hiện tại. Nói cách khác, hãy tập trung vào hầu hết các khía cạnh có liên quan và bỏ qua các chi tiết. Sự tinh chỉnh từng bước được Wirth giới thiệu lần đầu tiên vào năm 1971, áp dụng nó cho mã giả. Nó cũng có thể được áp dụng cho lưu đồ, sơ đồ khối, thông số kỹ thuật chính thức và được sử dụng trong mọi giai đoạn phát triển phần mềm. Bạn có thể mắc sai lầm trong quá trình sàng lọc từng bước. Do đó, bạn cần kiểm tra từng lần tinh chỉnh kế tiếp trước khi chuyển sang cấp độ tiếp theo.

**Câu 94. Phát triển phần mềm thì khác gì sản xuất phần mềm?**

Phát triển phần mềm là việc chuyển nhu cầu của khách hàng hay mục tiêu tiếp thị thành sản phẩm phần mềm. Bao gồm các quá trình của kỹ nghệ phần mềm kết hợp với nghiên cứu mục tiêu tiếp thị. Từ đó phát triển, nâng cấp những sản phẩm phần mềm của máy tính.

Ngược lại Sản xuất phần mềm là trực tiếp nghiên cứu, triển khai và hoàn thiện sản xuất các phần mềm.

**Câu 95. Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại test gì?**

Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại test xác thực dữ liệu.

-Hết-