

**ĐỀ TÀI**

TRẢ LỜI CÂU HỎI NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

# Nhóm: 4

**Họ và tên nhóm sinh viên thực hiện:**

**Đỗ Thị Thu Thảo – MSV: B19DCAT182**

**Ngô Thị Thơm – MSV: B19DCAT194**

**Nguyễn Minh Phương – MSV: B19DCAT141**

**Trần Trung Đức – MSV: B19DCAT049**

**Giảng viên hướng dẫn: Đỗ Thị Bích Ngọc Ngành: Công nghệ thông tin**



**ĐỀ TÀI**

TRẢ LỜI CÂU HỎI NHẬP MÔN CÔNG NGHỆ PHẦN MỀM

# Nhóm: 4

**Họ và tên nhóm sinh viên thực hiện:**

**Đỗ Thị Thu Thảo – MSV: B19DCAT182**

**Ngô Thị Thơm – MSV: B19DCAT194**

**Nguyễn Minh Phương – MSV: B19DCAT141**

**Trần Trung Đức – MSV: B19DCAT049**

**Giảng viên hướng dẫn: Đỗ Thị Bích Ngọc Ngành: Công nghệ thông tin**

**PHẦN A. GIỚI THIỆU**

**LỜI CẢM ƠN**

Chúng em xin gửi lời tri ân sâu sắc đến cô Đỗ Thị Bích Ngọc – Giảng viên Chuyên ngành Công nghệ thông tin, trường Học viện Công nghệ Bưu chính Viễn thông đã nhiệt tình hướng dẫn chúng em trong suốt thời gian làm báo cáo. Cô đã tạo rất nhiều điều kiện thuận lợi, định hướng, chỉ bảo tận tình và cho những lời khuyên quý báu giúp chúng em giải quyết những khó khăn trong việc tiếp cận những kiến thức mới. Đồng thời, cô đã giúp chúng em hiểu được những kiến thức quan trọng, giúp chúng em tiếp cận với cách tư duy, giải quyết và trình bày để có thể tìm hiểu và hoàn thành báo cáo đúng thời hạn.

Chúng em cũng xin chân thành cảm ơn các bạn trong nhóm đã luôn tạo điều kiện, quan tâm, nhiệt tình trong suốt quá trình học tập và hoàn thành báo cáo.

Chúng em xin chân thành cảm ơn!

*Hà Nội, tháng 3 năm 2022*

### Nhóm học viên

**Nhóm 4**

**LỜI NÓI ĐẦU**

Nhập môn Công nghệ phần mềm là môn học cung cấp những kiến thức cơ bản về tất cả mọi hoạt động liên quan đến phát triển phần mềm và kiến thức cơ bản về UML trong phát triển phần mềm. Nội dung bao gồm các kiểu hệ thống thông tin, các mô hình phát triển phần mềm, lập kế hoạch và quản lý dự án; các pha phát triển phần mềm từ xác định yêu cầu, phân tích, thiết kế đến lập trình – tích hợp; các kiến thức cơ bản về mô hình phần mềm với UML.

Bài báo cáo này trả lời những câu hỏi trọng tâm về môn học Nhập môn Công nghệ phần mềm.

### MỤC LỤC

[1. Thế nào là corrective maintenance? 1](#_Toc99285577)

[2. Thế nào là adaptive maintenance? 1](#_Toc99285578)

[3. Thế nào là perfective maintenance? 1](#_Toc99285579)

[4. Thế nào là refactoring? 1](#_Toc99285580)

[5. Thế nào là "from scratch"? 1](#_Toc99285581)

[6. Thế nào là moving target problem? 1](#_Toc99285582)

[7. Thế nào là regression fault? 1](#_Toc99285583)

[8. Thế nào là một episode? 1](#_Toc99285584)

[9. Thế nào là một iteration? 1](#_Toc99285585)

[10. Thế nào là một increasement? 2](#_Toc99285586)

[11. Thế nào là một artifact? 2](#_Toc99285587)

[12. Thế nào là portability? 2](#_Toc99285588)

[13. Thế nào là reusability? 2](#_Toc99285589)

[14. Thế nào là milestone? 2](#_Toc99285590)

[15. Thế nào là một story? 2](#_Toc99285591)

[16. Thế nào là refactoring? 2](#_Toc99285592)

[17. Thế nào là concept exploration? 2](#_Toc99285593)

[18. Thế nào là business model? 3](#_Toc99285594)

[19. Thế nào là traceability? 3](#_Toc99285595)

[20. Thế nào là egoless programming? 3](#_Toc99285596)

[21. Thế nào là PM? 3](#_Toc99285597)

[22. Thế nào là technical leader? 3](#_Toc99285598)

[23. Thế nào là programming secretary? 3](#_Toc99285599)

[24. Thế nào là backup programmer? 3](#_Toc99285600)

[25. Thế nào là supper programmer? 4](#_Toc99285601)

[26. Thế nào là một bản thiết kế còn ommision? 4](#_Toc99285602)

[27. Thế nào là một bản thiết kế còn contradiction? 4](#_Toc99285603)

[28. Thế nào là một phần mềm COTS? 4](#_Toc99285604)

[29. Thế nào là SPMP? 4](#_Toc99285605)

[30. Thế nào là alpha release? 4](#_Toc99285606)

[31. Thế nào là beta release? 4](#_Toc99285607)

[32. Thế nào là process? 4](#_Toc99285608)

[33. Thế nào là workflow? 5](#_Toc99285609)

[34. Luật Miller trong CNPM nói gì? 5](#_Toc99285610)

[35. Luật Brooks trong CNPM nói gì? 5](#_Toc99285611)

[36. Luật Dijkstra trong CNPM nói gì? 5](#_Toc99285612)

[37. Verification và Validation (V&V) là gì? 5](#_Toc99285613)

[38. Thế nào là inspection? 5](#_Toc99285614)

[39. Thế nào walkthrough? 6](#_Toc99285615)

[40. Thế nào là một moderator trong nhóm inspection? 7](#_Toc99285616)

[41. Thế nào là một recorder trong nhóm inspection? 7](#_Toc99285617)

[42. Mô hình CMM là gì? 7](#_Toc99285618)

[43. Thế nào là test performance? 8](#_Toc99285619)

[44. Thế nào là test robustness? 8](#_Toc99285620)

[45. Thế nào là coin of uncertainty? 8](#_Toc99285621)

[46. Thế nào là norminal effort? 8](#_Toc99285622)

[47. Thế nào là phần mềm organic? 8](#_Toc99285623)

[48. Thế nào là phần mềm embeded? 9](#_Toc99285624)

[49. Thế nào là phần mềm Semi-Detached? 9](#_Toc99285625)

[50. Thế nào là TCF? 9](#_Toc99285626)

[51. Thế nào là UFP? 9](#_Toc99285627)

[52. Thế nào là flow trong FFT? 10](#_Toc99285628)

[53. Thế nào là process trong FFP? 10](#_Toc99285629)

[54. Tại sao không có pha kiểm thử? 10](#_Toc99285630)

[55. Tại sao không có pha làm tài liệu? 10](#_Toc99285631)

[56. Tại sao không có pha lập kế hoạch? 10](#_Toc99285632)

[57. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao? 10](#_Toc99285633)

[58. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm? 10](#_Toc99285634)

[59. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước? 11](#_Toc99285635)

[60. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì? 11](#_Toc99285636)

[61. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh? 11](#_Toc99285637)

[62. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì? 12](#_Toc99285638)

[63. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng? 12](#_Toc99285639)

[64. Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng thì phù hợp với những dự án có đặc điểm gì? 12](#_Toc99285640)

[65. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu xoắn ốc? 12](#_Toc99285641)

[66. Mô hình vòng đời phần mềm kiểu xoắn ốc thì phù hợp với những dự án có đặc điểm gì? 13](#_Toc99285642)

[67. Nêu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu tiến trình linh hoạt? 13](#_Toc99285643)

[68. Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt thì phù hợp với những dự án có đặc điểm gì? 13](#_Toc99285644)

[69. Trong mô hình tiến trình liên hoạt, luôn có đại diện của khác hàng trong nhóm phát triển thì có ưu điểm gì? 13](#_Toc99285645)

[70. Nêu ưu điểm, nhược điểm của mô hình nhóm code bình đẳng? 14](#_Toc99285646)

[71. Mô hình nhóm code bình đẳng thì phù hợp với những dự án có đặc điểm gì? 14](#_Toc99285647)

[72. Nêu ưu điểm, nhược điểm của mô hình nhóm code có chef? 14](#_Toc99285648)

[73. Mô hình nhóm code có chef thì phù hợp với những dự án có đặc điểm gì? 14](#_Toc99285649)

[74. Nêu ưu điểm, nhược điểm của kĩ thuật pair programming? 14](#_Toc99285650)

[75. Kĩ thuật pair programming thì phù hợp với những dự án có đặc điểm gì? 15](#_Toc99285651)

[76. Nêu ưu điểm, nhược điểm của kĩ thuật time boxing? 15](#_Toc99285652)

[77. Nêu ưu điểm, nhược điểm của kĩ thuật stand up meeting? 15](#_Toc99285653)

[78. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC? 15](#_Toc99285654)

[79. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP? 15](#_Toc99285655)

[80. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point? 16](#_Toc99285656)

[81. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO? 16](#_Toc99285657)

[82. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả? 16](#_Toc99285658)

[83. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo? 17](#_Toc99285659)

[84. Nếu nhóm SQA phát hiện ra ít lỗi, thì có thể kết luận nhóm code giỏi hay nhóm SQA kém? Tại sao? 17](#_Toc99285660)

[85. Tại sao nói inspection và walkthrough là hướng tài liệu, mà không phải hướng vào người tham gia? 17](#_Toc99285661)

[86. Quality assurance thì khác gì với testing? 17](#_Toc99285662)

[87. Tại sao nói function point chịu ảnh hưởng chủ quan của các chuyên gia? 17](#_Toc99285663)

[88. COCOMO tính đến nhiều tiêu chí hơn hay là function point? Giải thích? 17](#_Toc99285664)

[89. SW development multiplier của COCOMO thì khác gì TCF của function point? 17](#_Toc99285665)

[90. TCF của function point thì khác gì hằng số b của FFP? 18](#_Toc99285666)

[91. Tại sao nguyên lí Djistra lại đúng? 18](#_Toc99285667)

[92. Tại sao luật Brook lại đúng? 18](#_Toc99285668)

[93. Người ta áp dụng luật Miller trong CNPM như thế nào? 18](#_Toc99285669)

[94. Phát triển phần mềm thì khác gì sản xuất phần mềm? 18](#_Toc99285670)

[95. Test trường hợp sai kiểu dữ liệu đầu vào thì thuộc thể loại test gì? 18](#_Toc99285671)

**PHẦN B. NỘI DUNG TRẢ LỜI**

### 1. Thế nào là corrective maintenance?

Bảo Trì Hiệu Chỉnh (Corrective Maintenance) là các hoạt động khắc phục sự cố khi một phần mềm không thực hiện đúng chức năng của nó. Bảo trì hiệu chỉnh tập trung vào fix lỗi, đó là mọi lỗi cần được sửa chữa, cho dù là lỗi phân tích, lỗi thiết kế, lỗi mã hóa, lỗi tài liệu hoặc bất kỳ loại lỗi nào khác.

### 2. Thế nào là adaptive maintenance?

Bảo Trì Thích Nghi (Adaptive Maintenance) là các hoạt động tiến hoá hệ thống nhằm đạt nhu cầu người dùng và doanh nghiệp. Được gây ra bởi: Nhu cầu nội bộ, cạnh tranh bên ngoài, những yêu cầu ngoài. Do đó cần thiết đưa ra những yêu cầu mới cho hệ thống. Như vậy chúng ta nên xử lý giống như phát triển mới theo hướng tiếp cận và phương pháp.

### 3. Thế nào là perfective maintenance?

Bảo Trì Dự Đoán (Predictive Maintenance) là hoạt động theo dõi năng suất và tình trạng của phần mềm trong quá trình vận hành bình thường. Hoạt động này giúp có thể dự đoán khi nào các lỗi thiết bị có thể xảy ra. Từ đó, giảm khả năng xảy ra hư hỏng của thiết bị, cải thiện chất lượng chương trình sẵn sàng hoạt động, giúp giảm chi phí trong sử dụng hệ thống, tăng khả năng bảo trì hệ thống và gần hơn nhu cầu người dùng.

### 4. Thế nào là refactoring?

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

### 5. Thế nào là "from scratch"?

“From scratch" là bắt đầu xây dựng phần mềm từ đầu, không dựa trên một cơ sở hay một phần mềm có sẵn nào từ trước mà là làm mới hoàn toàn.

### 6. Thế nào là moving target problem?

Moving target problem là thay đổi các yêu cầu phần mềm sau khi giai đoạn triển khai phần mềm đã được bắt đầu. Moving target problem cũng góp phần phần gây tổn hại về chất lượng phần mềm, hiệu suất phần mềm và trì hoãn việc hoàn thành sản phẩm bằng cách đẩy vào các tính năng lớn vào phút cuối.

### 7. Thế nào là regression fault?

Hồi quy lỗi (regression fault) là một loại lỗi phần mềm. Sự cố này do một số yếu tố gây ra như nâng cấp hệ thống, cải tiến tính năng và sửa các lỗi trước đó thông qua bản vá. Có rất nhiều sự phụ thuộc mà hầu như bất kỳ thay đổi nào cũng gây ra một hoặc nhiều lỗi hồi quy. Khi đó, điều duy nhất có thể làm là thiết kế lại toàn bộ sản phẩm phần mềm và thực hiện lại nó.

### 8. Thế nào là một episode?

Một episode là một lần lặp lại các pha. Một phần mềm sẽ hoàn thành và bàn giao tới tay khách hàng sau nhiều episode.

### 9. Thế nào là một iteration?

Lặp lại (iteration) là một khía cạnh nội tại của kỹ thuật phần mềm,và các mô hình vòng đời lặp đi lặp lại đã được sử dụng trong hơn 30 năm [Larman và Basili, 2003].

Ví dụ, mô hình thác nước, được đưa ra lần đầu tiên vào năm 1970, là mô hình lặp lại (nhưng không tăng dần).

### 10. Thế nào là một increasement?

Một increasement đc sinh ra trong mô hình lặp và tăng trưởng sẽ được bổ sung thêm chức năng hơn so với sản phẩm phát hành ở vòng lặp trước. Quá trình này sẽ diễn ra đến khi đạt được một hệ thống hoàn chỉnh.

### 11. Thế nào là một 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?

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 reusability?

Reusability (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 milestone?

Milestone là "Cột mốc" quan trọng của dự án. Các mốc này là những chỉ dấu bắt đầu hoặc kết thúc một dự án hoặc đánh dấu sự hoàn thành của một giai đoạn công việc chính. Nó có nhiệm vụ chia nhỏ một dự án lớn thành các phần có thể quản lý được và giúp quá trình quản lý dự án một cách dễ dàng hơn.

### 15. Thế nào là một story?

Bước đầu tiên là nhóm phát triển phần mềm xác định các tính năng (story) 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 là bao nhiêu. Bước đầu tiên này tương ứng với các yêu cầu và quy trình phân tích của mô hình lặp lại và tăng dần

### 16. Thế nào là refactoring?

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

### 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 họp tiếp theo giữa các thành viên của nhóm phát triển và nhóm khách hàng, chức năng của sản phẩm được đề xuất sẽ được cập nhật liên tục và được phân tích về tính khả thi về mặt kỹ thuật và biện minh tài chính.

### 18. Thế nào là business model?

Mô hình kinh doanh (business model) mô tả cách một tổ chức tạo ra và mang lại giá trị cho khách hàng. Nó mô tả các sản phẩm hoặc dịch vụ mà một công ty cung cấp và cách một công ty nhận lại từ những sản phẩm hoặc dịch vụ - một mô hình doanh thu.

### 19. Thế nào là traceability?

Khả năng xác định nguồn gốc (Traceability): là khả năng theo dõi các hạng mục công việc trong suốt vòng đời phát triển. Nó được sử dụng để theo dõi những gì đang diễn ra trong vòng đời phát triển và hiển thị những gì đã xảy ra. Đạt được sự tuân thủ quy định là một mục đích chung để truy xuất nguồn gốc trong kỹ thuật phần mềm.

### 20. Thế nào là egoless programming?

Lập trình bản ngã (egoless programming): là một phong cách lập trình máy tính trong đó các yếu tố cá nhân được giảm thiểu để chất lượng có thể được cải thiện. Mục đích là sản xuất phần mềm là sản phẩm của cả nhóm chứ không phải của một hoặc một vài cá nhân. Trong đó, các lập trình viên gắn bó cao với mã lệnh của họ, các mô-đun như là sự mở rộng của chính bản thân

### 21. Thế nào là PM?

Người quản lý dự án – Project Manager (PM) 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. 90% công việc Project Manager liên quan đến giao tiếp như hoạch định, giải quyết sự cố, quản lý rủi ro, giao tiếp với khách hàng. 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.

### 22. Thế nào là technical leader?

Technical leader (Trưởng nhóm kĩ thuật) là những người chịu trách nhiệm lãnh đạo một đội nhóm phát triển và xây dựng tầm nhìn về kỹ thuật, cùng các mục tiêu phù hợp với doanh nghiệp. Sau đó biến nó thành hiện thực với sự góp sức của toàn đội. Họ sẽ tham gia nhiều hơn vào các cuộc thảo luận và quyết định về kiến ​​trúc code. Đồng thời Tech Lead cũng sẽ quan sát, quản lý chất lượng của codebase và sự phát triển kỹ thuật của team. Đôi khi họ đảm nhận luôn vai trò là một kiến trúc sư phần mềm, trưởng nhóm hoặc quản lý kỹ sư phần mềm. Kết hợp cùng với các phòng ban khác để duy trì mối quan hệ và đạt được mục tiêu đã đề ra.

### 23. Thế nào là programming secretary?

Thư ký lập trình (programming secretary) là thành viên chủ chốt, có tay kĩ năng cao, được trả lương cao. Thư ký lập trình chịu trách nhiệm duy trì về tài liệu cho toàn bộ dự án như: Liệt kê danh sách mã nguồn, ngôn ngữ điều khiển công việc (JCL), dữ liệu test, biên dịch code, kiểm tra code convention, chạy các test case

### 24. Thế nào là backup programmer?

Là lập trình viên dự phòng khi có sự cố bất ngờ xảy ra, nắm rõ dự án trong quản lý và code, lập kế hoạch test hộp đen (black-box) và các công việc độc lập với tiến trình thiết kế.

### 25. Thế nào là supper programmer?

- Supper programmer một lập trình viên có kết quả đầu ra gấp bốn hoặc năm lần so với một lập trình viên giỏi trung bình.

- Ngoài ra,còn là một nhà quản lý và lãnh đạo tuyệt vời, và các kỹ năng, sự nhiệt tình là lý do tạo nên thành công của dự án.

### 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.

### 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

### 28. Thế nào là một phần mềm COTS?

- Commercial off-the-shelf (COTS) là phần mềm

Thuật ngữ trước đó cho loại phần mềm này là phần mềm được bọc co lại vì hộp con-taining CD hoặc diskettes, hướng dẫn sử dụng, và thỏa thuận cấp phép hầu như luôn luôn được thu nhỏ bọc. Ngày nay, phần mềm COTS thường được tải xuống qua World Wide Web - không có hộp để thu nhỏ bọc. Vì lý do này, phần mềm COTS ngày nay một số lần được gọi là clickware. Phần mềm COTS được phát triển cho "thị trường"; nghĩa là, phần mềm không được nhắm mục tiêu đến một khách hàng hoặc người dùng cụ thể cho đến khi nó được phát triển và có sẵn để mua.

### 29. Thế nào là SPMP?

- SPMP - The software product management plan - kế hoạch quản lý quá trình sản xuất phần mềm.

- Một khi những gì sẽ được phát triển được biết chính xác, SPMP được soạn thảo. Điều này bao gồm ngân sách, yêu cầu nhân sự và lịch trình chi tiết. Sớm nhất chúng ta có thể lập kế hoạch quản lý dự án là khi tài liệu đặc điểm kỹ thuật đã được khách hàng phê duyệt, nghĩa là vào cuối giai đoạn phân tích. Cho đến thời điểm đó, việc lập kế hoạch phải là sơ bộ và một phần.

- Trong suốt dự án, ban quản lý cần theo dõi SPMP và theo dõi bất kỳ sai lệch nào so với kế hoạch.

### 30. Thế nào là alpha release?

- Trong bất kì một phần mềm nào đó, ngay sau khi thử nghiệm sản phẩm hoàn tất, các phiên bản của sản phẩm hoàn chỉnh được cung cấp cho các khách hàng tương lai được chọn để thử nghiệm tại chỗ.

- Phiên bản đầu tiên như vậy được gọi là bản phát hành alpha – alpha release.

### 31. Thế nào là beta release?

- Bản phát hành alpha được sửa chữa được gọi là bản phát hành beta – beta release ; nói chung, bản phát hành beta được dự định gần với phiên bản cuối cùng.

### 32. Thế nào là process?

- Tiến trình phần mềm là “phương cách” sản xuất ra phần mềm.

- Tiến trình phần mềm nghiên cứu các cách kết hợp:

 +  Mô hình vòng đời

+ Các công cụ CASE

+  Các cá nhân xây dựng phần mềm

+ Các công nghệ

    • Tiến trình phần mềm = Khía cạnh kỹ thuật + Khía cạnh quản lý

- Các tổ chức khác nhau có những tiến trình phần mềm khác nhau.

### 33. Thế nào là workflow?

- Mỗi luồng công việc bao gồm một số bước và để thực hiện quy trình làm việc đó, các bước của quy trình làm việc được thực hiện nhiều lần

- Là 1 cách thực hiện cụ thể của process.

### 34. Luật Miller trong CNPM nói gì?

- Theo luật Miller, ở bất cứ thời điểm nào, chúng ta chỉ có thể tập trung vào khoảng 7 chunks ( tương ứng 7đơn vị thông tin)

-  Để xử lý lượng thông tin lớn hơn yêu cầu sử dụng bước làm mịn theo kiểu bậc thang :

+  Tập trung vào các khía cạnh quan trọng nhất hiện thời

+ Các khía cạnh trì hoãn thường ít quan trọng hơn

+ Cuối cùng mọi khía cạnh đều được xử lý nhưng theo thứ tự mức độ quan trọng hiện thời

### 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.

### 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.

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

-  Xác minh (Verification) : quá trình xác định xem một quy trình làm việc đã được thực hiện chính xác hay chưa; điều này diễn ra ở cuối mỗi workflow

-  Thẩm định (Validation) : quá trình đánh giá chuyên sâu diễn ra ngay trước khi sản phẩm được giao cho khách hàng. Mục đích của nó là  xác định xem toàn bộ sản phẩm đáp ứng các yêu cầu đưa ra hay không

-  Thuật ngữ “xác minh” cũng được sử dụng cho kiểm thử không có sự thực thi.

### 38. Thế nào là inspection?

    Kiểm tra kỹ lưỡng (Inspection)

    •  Quá trình kiểm tra kỹ lưỡng gồm 5 bước hình thức

o Overview: Tổng quan về tài liệu cần được kiểm tra (yêu cầu, đặc điểm kỹ thuật, thiết kế, mã hoặc kế hoạch) được đưa ra bởi một trong những cá nhân chịu trách nhiệm sản xuất tài liệu đó. Cuối phiên tổng quan, tài liệu được phân phối cho những người tham gia.

o Preparation: Chuẩn bị, với mục đích thống kê các loại lỗi.

Những người tham gia cố gắng hiểu tài liệu một cách chi tiết. Danh sách các loại lỗi được tìm thấy trong các cuộc kiểm tra gần đây, với các loại lỗi được xếp hạng theo tần số. Những danh sách này giúp các thành viên trong nhóm tập trung vào các lĩnh vực xảy ra nhiều lỗi nhất.

o Kiểm tra kỹ lưỡng. Người điều hành phải xuất trình một báo cáo bằng văn bản về việc kiểm tra để đảm bảo theo dõi tỉ mỉ.

o Rework: Sửa lỗi

Trong quá trình làm lại, cá nhân chịu trách nhiệm về tài liệu giải quyết tất cả các lỗi và vấn đề được ghi nhận trong báo cáo bằng văn bản.

o Follow-up:  Theo dõi

 Tất cả các bản sửa lỗi phải được kiểm tra để đảm bảo rằng không có lỗi mới nào được giới thiệu. Nếu hơn 5% tài liệu được kiểm tra đã được làm lại, thì nhóm nghiên cứu phải tập hợp lại để tái tạo 100%.

    • Việc kiểm tra nên thực hiện bởi 1 nhóm 4 thành viên

o Người điều tiết (Moderator)  - quản lý, trưởng đoàn kiểm tra

o Một thành viên trong đội thực đang thi luồng công việc hiện hành

o Một thành viên của đội sẽ thực hiện luồng công việc tiếp theo

o Một thành viên của nhóm SQA

    • Những vai trò đặc biệt được đảm nhiệm bởi

o Người điều tiết (Moderator)

o Người đọc (Reader)

o Người ghi chép (Recorder)

### 39. Thế nào walkthrough?

Rà soát lướt qua (Walkthrough)

    •  Đội rà soát lướt qua bao gồm từ 4 đến 6 thành viên

    • Nó bao gồm đại diện của

o Đội chịu trách nhiệm cho luồng công việc hiện thời

o Đội chịu trách nhiệm cho luồng công việc tiếp thep

o Nhóm SQA

    • Có 2 cách tiến hành Walkthrough:

+ Document Driven: 1 người (presenter) sẽ kiểm tra toàn bộ nội dung, và những người còn lại có trách nhiệm bình luận, ngắt quãng những chỗ nghi ngờ có lỗi. (cách hiệu quả hơn để tìm lỗi)

+  Participant Driven: mỗi thành viên chuẩn bị 1 danh sách gồm 2 cột: các chi tiết chưa rõ ràng hoặc chi tiết nghi ngờ dính lỗi. Đại diện team phân tích phải phân tích được đâu là lỗi và làm rõ cho người hỏi.

### 40. Thế nào là một moderator trong nhóm inspection?

Người điều tiết (moderator) là người lãnh đạo, trưởng nhóm, trưởng đoàn kiểm tra và họ chịu trách nhiệm lập kế hoạch kiểm tra.

### 41. Thế nào là một recorder trong nhóm inspection?

- Người ghi chép (recorder) là người tham gia chịu trách nhiệm ghi lại các lỗi được tìm thấy trong quá trình kiểm tra. Các lỗi được ghi lại theo mức độ nghiêm trọng (lớn hay nhỏ; một ví dụ về lỗi lớn là lỗi gây ra chấm dứt sớm hoặc làm hỏng cơ sở dữ liệu) và loại lỗi.

### 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.

+  Tập các chiến lược cải tiến tiến trình phần mềm

- SW-CMM cho phần mềm

- P-CMM quản lý nguồn nhân lực (“people”)

- SE-CMM cho kỹ thuật hệ thống

- IPD-CMM cho phát triển sản phẩm tích hợp

- SA-CMM để mua lại phần mềm

+ Các chiến lược được thống nhất thành CMMI (Capability maturity model integration) năm 1997

- SW-CMM: 5 mức:

+ Maturity level 1: Initial Level

• Phương pháp đặc biệt

- Toàn bộ quá trình là không thể đoán trước

- Quản lý bao gồm các phản ứng đối với các cuộc khủng hoảng

• Hầu hết các tổ chức phần mềm trên toàn thế giới đều ở cấp độ 1

+  Maturity level 2: Rêpatable Level

• Quản lý phần mềm cơ bản

 - Các quyết định quản lý cần được thực hiện trên cơ sở kinh nghiệm trước đó với Sản phẩm tương tự

- Các phép đo (“số liệu”) được thực hiện

- Chúng có thể được sử dụng để đưa ra các dự đoán về chi phí và thời gian trong dự án tiếp theo

- Các vấn đề được xác định, hành động khắc phục ngay lập tức được thực hiện

+ Maturity level 3: Defined Level

• Quy trình phần mềm được ghi lại đầy đủ

- Các khía cạnh quản lý và kỹ thuật được xác định rõ ràng

- Không ngừng nỗ lực nâng cao chất lượng, năng suất

- Đánh giá được thực hiện để cải thiện chất lượng phần mềm

- Các công cụ CASE hiện có thể áp dụng (và không phải ở cấp độ 1 hoặc 2)

+ Maturity level 4: Managed Level

• Các mục tiêu về chất lượng và năng suất được đặt ra cho từng dự án

- Chất lượng, năng suất được giám sát liên tục

- Kiểm soát chất lượng thống kê được thực hiện

+ Maturity level 5: Optimizing Level

• Sự cải tiến quá trình liên tục

- Chất lượng thống kê và kiểm soát quá trình

- Phản hồi kiến thức từ mỗi dự án cho kế tiếp

- KL:

### 43. Thế nào là test performance?

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

### 44. Thế nào là test robustness?

- Là bất kỳ phương pháp đảm bảo chất lượng nào tập trung vào việc kiểm tra độ bền của phần mềm. Kiểm tra độ bền cũng đã được sử dụng để mô tả quá trình xác minh độ bền (tức là tính chính xác) của các trường hợp thử nghiệm trong quá trình thử nghiệm.

-  ANSI và IEEE đã định nghĩa độ mạnh mẽ là mức độ mà một hệ thống hoặc thành phần có thể hoạt động chính xác khi có đầu vào không hợp lệ hoặc điều kiện môi trường căng thẳng.

### 45. Thế nào là coin of uncertainty?

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

### 46. Thế nào là norminal effort?

Effort có nghĩa là tổng thời gian ước tính cho đến khi hoàn thành 1 task nào đó trong công việc

### 47. Thế nào là phần mềm organic?

Organic – Một dự án phần mềm có quy mô nhóm cần thiết là đủ nhỏ, vấn đề được hiểu rõ và đã được giải quyết trong quá khứ và các thành viên trong nhóm cũng có kinh nghiệm về vấn đề này.

### 48. Thế nào là phần mềm embeded?

Phần mềm nhúng hay embedded software là lập trình chuyên biệt trong các thiết bị không phải PC, là một phần của ứng dụng khác nằm trên chip để điều khiển các chức năng cụ thể của thiết bị. Không giống như phần mềm ứng dụng, có thể được cài đặt trên nhiều hệ thống máy tính và được sửa đổi để cung cấp các mức chức năng khác nhau, phần mềm nhúng có các yêu cầu và khả năng phần cứng cố định. Nó được tạo riêng cho thiết bị cụ thể mà nó chạy, với các hạn chế về xử lý và bộ nhớ gắn trực tiếp với thông số kỹ thuật của thiết bị đó.

### 49. Thế nào là phần mềm Semi-Detached?

    • Một dự án phần mềm được gọi 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 Organic và Embedded.

    • Các dự án được gọi là Semi-Detached tương đối ít quen thuộc hơn và khó phát triển hơn so với các dự án Organic và đòi hỏi nhiều kinh nghiệm hơn, sự hướng dẫn tốt hơn và yêu cầu tính sáng tạo .

    • VD: Các trình biên dịch hoặc các Embedded khác nhau có thể được coi là Semi-Detached.

### 50. Thế nào là TCF?

    • TCF (technical complexity factor) là thứ 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… và cập nhật trực tuyến.

    • Mỗi yếu tố trong số 14 yếu tố này được gán một giá trị từ 0 (“không hiện tại hoặc không có dòng chảy”) đến 5 (“dòng chảy mạnh và xuyên suốt”). 14 giá trị được tính tổng mang lại tổng mức độ ảnh hưởng (DI). Sau đó TCF được tính bởi:

TCF = 0.65 + 0.01 x DI

     Vì DI có thể thay đổi từ 0 đến 70 nên TCF thay đổi từ 0,65 đến 1,35.

### 51. Thế nào là UFP?

    • UFP (Unadjusted Function Points) là các điểm chức năng chưa được điều chỉnh.

    • FP (Function Points) là một thành phần của phát triển phần mềm giúp ước tính gần đúng chi phí phát triển sớm trong quá trình. Nó là một quá trình xác định các chức năng cần thiết và độ phức tạp của chúng trong một phần mềm để ước tính kích thước và phạm vi của phần mềm sau khi hoàn thành.

    • Cách tính FP được cho bởi phương trình:

FP = 4 x Input + 5 x Output + 4 x Inquiry + 10 x Master file + 7 x Interface

    • Công thức trên là đơn giản hóa của phép tính ba bước như sau:

        ◦ Mỗi thành phần của sản phẩm (Inp, Out, Inq, Maf và Inf) phải được phân loại là đơn giản, trung bình hoặc phức tạp

        ◦ Mỗi thành phần được gán một số điểm chức năng tùy thuộc vào cấp độ của nó

        ◦ Sau đó các FP được gán cho mỗi thành phần được tổng hợp lại, tạo ra các UFP

### 52. Thế nào là flow trong FFT?

    • Flow (Dòng): giao diện dữ liệu giữa sản phẩm và môi trường như màn hình, báo cáo…

### 53. Thế nào là process trong FFP?

    • Process (Xử lý): nếu bàn về chức năng thì process là một định nghĩa logic hay toán học dùng để thao tác trên dữ liệu.

### 54. 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.

### 55. Tại sao không có pha làm tài liệu?

    • Mọi pha phải được viết tài liệu trước khi bắt đầu 1 pha mới 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 khi phát triển nên cần tài liệu để ghi lại điều này.

### 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 được đưa ra khi các chi tiết kỹ thuật mà khách hàng đưa ra đã được hoàn tất.

    • Vì chỉ có bản kế hoạch tạm thời về quản lý dự án, mọi kế hoạch chỉ là ước lượng, quá trình có thể bị thay đổi do nhiều tác nhân khác nhau trong lúc thực hiện dự án.

### 57. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? 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.

### 58. 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.

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

    • Ưu điểm:

        ◦ Dễ phân 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.

### 60. 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

### 61. 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 triền sản phẩm diễn ra nhanh chóng.

    • Nhược điểm:

        ◦ Thành viên trong nhóm phát triển sử dụng bản mẫu nhanh không 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.

### 62. 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

### 63. 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

    • Nhược điểm:

        ◦ Khó khi kết hợp build vào cấu trúc hiện thời là làm sao không phá hủy cấu trúc đã xây dựng.

            ▪ Như vậy, yêu cầu cấu trúc phải mở.

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

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

    • Mô hình này đã được sử dụng hiệu quả cho một số phần mềm vừa và nhỏ, có dung lượng công việc từ 9 người-tháng cho đến 100 người-năm.

    • Mô hình này thực sự có lợi khi yêu cầu của khách hàng không rõ ràng và hay thay đổi.

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

    • 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.

### 66. 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.

### 67. 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 trên một đề án có số lượng nhân viên khổng lồ.

        ◦ Không có hiệu quả đối với những khách hàng ở xa đội ngũ phát triển.

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

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.

### 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 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.

### 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

### 71. 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 đằng và hỗ trợ hết sức, không có cạnh tranh, thăng tiến, không có leader thực sự trong nhóm.

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

    • 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.

### 73. 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.

### 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.

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

    • Dùng pair programing khi dự án phức tạp hoặc PR quá lớn.

### 76. Nêu ưu điểm, 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.

### 77. Nêu ưu điểm, 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.

### 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).

### 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 (như số lượng file).

### 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:

    • Đánh giá dựa vào chuyên gia (mang sự chủ quan).

### 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 phương pháp phụ thuộc vào mô hình áp dụng với yêu cầu của khách hàng.

### 82. 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.

### 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.

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

Nếu nhóm SQA phát hiện ra ít lỗi thì có thể kết luận là nhóm SQA kém vì theo nguyên lí Dijkstra thì rất dễ để chứng minh phần mềm còn lỗi.

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

Nói Inspection và Walkthrough là hướng tài liệu vì cả hai đều vận dụng kĩ năng và kinh nghiệm của các thành viên để bổ sung và hoàn thiện tài liệu, bản thảo phần mềm chứ không nhằm nâng cao trình độ và kĩ năng của các thành viên tham gia.

### 86. Quality assurance thì khác gì với testing?

    • Sự khác biệt giữa Quality Assurance và Testing:

    • Hoạt động của QA:

    • Kiểm tra chất lượng.

    • Xác định quy trình.

    • Nhận diện và lựa chọn công cụ.

    • Đào tạo về quy trình và  các tiêu chuẩn về chất lượng.

    • Hoạt động của Testing:

    • Hướng dẫn.

    • Thử nghiệm.

    • Kiểm tra.

    • Đánh giá, kiểm soát bắt buộc.

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

Khi thiết kế ứng dụng phần mềm, cũng giống như một tảng băng trôi, phần lớn chìm ở dưới còn nổi trên bề mặt chỉ là một phần nhỏ. Nhìn bề ngoài thì có vẻ rất đơn giản nhưng thực tế bên trong phải thực hiện hàng nghìn câu lệnh, hàng nghìn tiến trình. Khi đánh giá ứng dụng phần mềm chúng ta phải hiểu những gì là bên trong và những gì là bên ngoài. Bởi vậy, function point chịu ảnh hưởng chủ quan của các chuyên gia.

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

COCOMO tính đến nhiều tiêu chí hơn function point vì COCOMO có 3 loại phù hợp với chức năng và tiêu chí của các dự án hơn function point.

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

SW development multiplier của COCOMO chi tiết hơn và tập trung vào nhiều yếu tố hơn TCF của function point.

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

TCF của function point không phụ thuộc vào các yếu tố khách quan như hằng số b của FFP

### 91. Tại sao nguyên lí Djistra lại đúng?

Nguyên lí Dijkstra đúng vì quá trình kiểm thử phần mềm là quá trình chứng minh phần mềm còn lỗi, nên với một người kiểm thử phần mềm thì phần mềm luôn có lỗi.

### 92. Tại sao luật Brook lại đúng?

Nguyên lí Brooks đúng vì việc thêm người vào dự án sẽ kèm theo rất nhiều việc: phổ biến dự án, giải thích những thứ đã làm được, những thứ chưa làm được, những thứ nên và không nên cho người mới… Điều này sẽ làm cho dự án bị kéo dài và làm cho dự án bị trễ.

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

Trong CNPM, người ta luôn sắp xếp các thông tin trong danh sách các mục không lớn hơn 9, tốt nhất là khoảng 5 tập hợp thông tin vì theo định luật Miller thì mỗi người còn thể tập trung nhiều nhất vào 7 thông tin tại một thời điểm nên việc đưa quá nhiều thông tin trong danh sách các mục sẽ làm cho người dùng khó làm quen với sản phẩm.

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

Phát triển phần mềm là tạo ra và chịu trách nhiệm phát triển, bảo trì, khắc phục các sự cố của phần mềm. Còn sản xuất phần mềm là tạo ra phần mềm theo đơn hàng và khi phù hợp với nhu cầu khách hàng thì sẽ bàn giao và không cần phải chịu bất cứ trách nhiệm gì cho sản phẩm.

### 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 là loại test data.

Máy chủ DNS có thể quản lý hoặc miền chính (primary zone) hay miền thứ cấp (secondary zone) hay cả hai. Khi đặt cấu hình cho máy chủ DNS có hai kiểu vùng khác nhau: Forward Lookup Zone và Reverse Lookup Zone

- Khi người quản trị muốn kiểm tra dịch vụ có thể sử dụng lệnh nslookup