**Đáp án đề cương công nghệ phần mềm**

**Câu 1** Công nghệ phần mềm là gì? Liệt kê các lĩnh vực trong công nghệ phần mềm. (chapter 1)

**Có nhiều định nghĩa khác nhau về công nghệ phần mềm, trong đó:**

* Theo tiêu chuẩn IEEE 610-1990: công nghệ phần mềm là sự áp dụng một cách tiếp cận có hệ thống, có kỷ luật, và định lượng được cho việc phát triển, sử dụng và bảo trì phần mềm.
* Theo CMU/SEI-90-TR-003 (Canegie Mellon University/Software Engineering Institute-90-Team Respone-003- Viện nghiên cứu công nghệ phần mêm thuộc đại học Canegie Mellon: công nghệ phần mềm là hình thức kĩ thuật áp dụng các quy tắc của khoa học máy tính và toán học để đạt được giải pháp có chi phí hợp lí cho các vấn đề phần mềm.
* Theo Ian Sommerville: Công nghệ phần mềm là một ngành kỹ thuật mà trọng tâm là phát triển hệ thống phần mềm có chất lượng cao với chi phí hợp lí.
* Theo sách Essensial of SE tái bản lần 3: Công nghệ phần mềm là một lĩnh vực rộng lớn có liên quan đến các phương diện của quá trình phát triển và hỗ trợ một hệ thống phần mềm, bao gồm:
  + Các quy trình kỹ thuật và nghiệp vụ
  + Các phương pháp và kỹ thuật đặc biệt
  + Các đặc tính và số liệu của sản phẩm
  + Con người, kỹ năng, làm việc nhóm
  + Công cụ và huấn luyện
  + Sự quản lý và phối hợp giữa các dự án

**Các lĩnh vực trong công nghệ phần mềm:**

Hiện nay, các lĩnh vực trong công nghệ phần mềm là rất rộng lớn, không thể phủ nhận vai trò của các phần mềm đối với cuộc sống hiện nay. Về tài chính, có hệ thống phần mềm quản lý ngân hàng; phương tiện vận chuyển, hay các thiết bị gia đình được điều khiển bởi các thiết bị được cài đặt phần mềm có sẵn, sức khỏe của chúng ta được chăm sóc tốt hơn nhờ các thiết bị y tế,…

**Câu 2** Cho biết yêu cầu phần mềm, yêu cầu chức năng, yêu cầu phi chức năng là gì? Cho ví dụ một yêu cầu chức năng và một yêu cầu phi chức năng. (chapter 1)

**Yêu cầu phần mềm:**

Theo IEEE Standard Glossary of SE Technology (1997) định nghĩa một yêu cầu là:

* + (1): Yêu cầu là một điều kiện hoặc khả năng của sản phẩm phần mềm cần thiết cho người dùng để giải quyết một vấn đề hoặc đạt được một mục tiêu.
  + (2): yêu cầu là một điều kiện hoặc khả năng cần phải thỏa mãn hoặc cần có của một hệ thống hoặc một thành phần hệ thống nhằm đáp ứng một hợp đồng, một tiêu chuẩn, một đặc tả hoặc 1 tài liệu bắt buộc khác.
  + (3): một diễn giải được tài liệu hóa của một điều kiện hoặc khả năng nào đó theo (1), (2).

Định nghĩa của IEEE bao hàm cả 2 quan điểm về yêu cầu từ người dùng (quan sát hành vi của hệ thống từ bên ngoài) và từ người phát triển (quan sát hệ thống từ bên trong).

Theo chuyên gia về yêu cần Alan Davis: “yêu cầu là một nhu cầu của người dùng hoặc một đặc tính, một chức năng, một thuộc tính cần thiết của một hệ thống **có thể được hiểu từ một vị trí bên ngoài hệ thống đó.”** Định nghĩa này nhấn mạnh cái mà sản phẩm phải cung cấp thay vì sản phẩm sẽ được làm như thế nào từ goc nhìn của người thiết kế và thi công.

Định nghĩa sau đi từ nhu cầu người dùng tới các đặc tính của hệ thống của Sommerville and Sawyer 1997:

*Yêu cầu là một đặc tả về cái phải được thi công.* Chúng là các mô tả về hành vi của hệ thống phải như thế nào, hoặc về một thuộc tính của hệ thống phải như thế nào. Yêu cầu có thể là một ràng buộc về quy trình phát triển của hệ thống.

**Yêu cầu chức năng:** những gì mà chương trình cần phải làm (các chức năng mà phần mềm cung cấp), chúng thường quan hệ các use-case hay những quy tắc nghiệp vụ.

**Ví dụ:** trong hệ thống quản lý thông tin sinh viên: người dùng có thể tìm kiếm thông tin sinh viên, in phiếu điểm của sinh viên…

**Yêu cầu phi chức năng:** những ràng buộc về tiêu chuẩn, thời gian, quy trình phát triển, … chủ yếu là những yêu cầu về chất lượng.

**Ví dụ:** trong hệ thống quản lý thông tin sinh viên: yêu cầu của người sử dụng là dễ sử dụng và thân thiện.

**Câu 3** Trình bày ba loại yêu cầu phi chức năng tiêu biểu. (chapter 2)

**Yêu cầu sản phẩm:**

* Yêu cầu khả năng sử dụng: một bộ quy định người dùng có thể sử dụng một cái gì đó một cách hiệu quả.
* Yêu cầu tính hiệu quả:
  + Yêu cầu hiệu suất: sản phẩm có thể hoạt động liên tục trong một thời gian dài, thời gian thực hiện các tác vụ nhanh.
  + Yêu cầu không gian: người dùng được cấp phát vùng lưu trữ riêng để có thể lưu trữ tài liệu lâu dài.
* Yêu cầu tính tin cậy: đảm bảo sản phẩm sẽ hoạt động tốt trong một thời gian dài.
* Yêu cầu tính di động: các ứng dụng hoặc thành phần của sản phẩm dễ dàng và nhanh chóng chuyển đến môi trường mới khi cần.

**Yêu cầu tổ chức:**

* Yêu cầu phân phối: sản phẩm được phân phối và sử dụng bởi nhiều bộ phận khác nhau.
* Yêu cầu thực hiện: thực hiện đầy đủ chức năng mà tổ chức cần có.
* Yêu cầu tiêu chuẩn: tuân theo các tiêu chuẩn mà tổ chức đưa ra

**Yêu cầu ngoài:**

* Yêu cầu đạo đức: các tiêu chuẩn về đạo đức cần được tuân thủ.
* Yêu cầu khả năng tương tác: đảm bảo các ứng dụng, thành phần của sản phẩm có thể vận hành với các ứng dụng, thành phần cụ thể khác (có thể truyền, nhận và sử dụng dữ liệu với các ứng dụng, thành phần cụ thể khác…)
* Yêu cầu lập pháp:
  + Yêu cầu tính an toàn: an toàn trong thiết kế, phát triển, sử dụng, bảo trì phần mềm.
  + Yêu cầu tính bảo mật: thông tin về người dùng, khách hàng sẽ không được tiết lộ.

**Câu 4** Độ phức tạp của phần mềm theo chiều rộng và chiều sâu được xác định thông qua những yếu tố nào? (chapter 2)

**Theo chiều rộng:**

* Các chức năng chính
* Các tính năng trong mỗi chức năng chính
* Quan hệ giữa tính năng và chức năng
* Giao diện cho hệ thống bên ngoài
* Số người sử dụng đồng thời và tổng số người sử dụng

**Theo chiều sâu:**

* Số lượng các mắt xích (các mối quan hệ, ràng buộc): các mối liên kết có thể được thông qua việc chia sẻ dữ liệu hoặc giao thức chuyển giao hoặc cả hai; các mối quan hệ có thể là phân cấp, tuần tự, vòng lặp, đệ quy hoặc 1 hình thức khác
* Ngân sách để thực hiện

**Câu 5** Cho biết các vấn đề kỹ thuật và phi kỹ thuật trong phát triển các hệ thống lớn. (chapter 2)

**Các vấn đề kỹ thuật:**

* Vấn đề và sự phân tích thiết kế: cách phân chia một vấn đề lớn, phức tạp và các giải pháp của nó thành các phần nhỏ. Chìa khóa để giải quyết vấn đề trên có thể thông qua các loại hành động sau:
  + Sự phân tích
  + Mô đun hóa
  + Sự phân ly
  + Sự tăng lên của các vòng lặp
* Công nghệ và công cụ: Một hệ thống lớn, phức tạp yêu cầu có nhiều hơn một người tham gia phát triển. Lúc đó mỗi người sẽ có sự lựa chọn về công nghệ và công cụ riêng của mình, điều đó dẫn đến sự rối loạn. Vì vậy cần chọn lọc ra một công nghệ và công cụ chung. Công nghệ và công cụ cần chọn lọc có thể liên quan đến những thứ sau
  + Cơ sở dữ liệu
  + Mạng
  + Nền tảng, ngôn ngữ phát triển
  + Cấu hình
* Quy trình và phương pháp: do có nhiều bộ phận cần được xem xét hơn trong một hệ thống so với một chương trình đơn giản được phát triển bởi một người dành cho một số hữu hạn người dùng nên cần có sự phối hợp giữa quy trình và phương pháp (xem thêm trong sách để hiểu rõ hơn)
  + Sự lựa chọn quy trình
  + Sự lựa chọn phương pháp
  + Sự lựa chọn công cụ hỗ trợ quy trình

**Các vấn đề phi kỹ thuật**

* Nỗ lực đánh giá và lập lịch
  + Cần xem xét và đánh giá nhiều thành phần hơn
  + Cần phối hợp nhiều thành phần trong giới hạn về yêu cầu tiên quyết và yêu cầu chung.
  + Cần xem xét nhiều hơn tiềm năng vể rủi ro và sự biến động.
* Phân công và liên lạc
  + Nhiều người tham gia đồng nghĩa các kĩ năng cũng tăng lên
  + Sự giao tiếp, liên lạc giữa các thành viên tăng lên
  + Nhiều lỗi và nhiều sự sửa đổi hơn

**Câu 6** Cho biết các yếu tố cần xem xét để xác định nhân lực cần có cho các công việc khi chuyển giao phần mềm cho khách hàng.

* Pre-release education and preparation
  + Number of expected users
  + Number of known problems and expected quality
  + Amount of user and support personel for training
* Post-release user and customer support
  + Call center and problem resolutions
  + Major problem fixes and code changes
  + Functional modification and enchancements.

**Câu 7** Cho biết tám nguyên tắc về đạo đức nghề nghiệp trong công nghệ phần mềm của IEEE-CS và ACM. (chapter 3)

1. Làm việc phải phù hợp với lợi ích chung
2. Làm việc đem lại lợi ích tốt nhất cho khách hàng
3. Đảm bảo sản phẩm đáp ứng được tiêu chuẩn chuyên nghiệp nhất
4. Phải duy trì tính toàn vẹn và độc lập trong xét đoán chuyên môn của kỹ sư phần mềm
5. Các nhà lãnh đạo và người lãnh đạo ngành CNPM phải tán thành và đẩy mạnh cách tiếp cận đạo đức để quản lý phát triển phần mềm và bảo trì.
6. Phải tăng tính toàn vẹn và uy tín nghề nghiệp nhất quán với lợi ích chung
7. Phải công bằng với sự hỗ trợ của đồng nghiệp
8. Cần học tập suốt đời để trau dồi đạo đức nghề nghiệp

**Câu 8** Cho biết 15 nguyên tắc xây dựng phần mềm của Davis. (chapter 3)

1. Chất lượng là số 1
2. Có thể tạo ra phần mềm chất lượng cao
3. Cố gắng đưa sản phẩm tới người dùng sớm nhất
4. Xác định rõ yêu cầu trước khi viết yêu cầu
5. Đánh giá những lựa chọn thiết kế
6. Sử dụng mô hình, quy trình thích hợp
7. Sử dụng các ngôn ngữ (lập trình) khác nhau cho các giai đoạn khác nhau
8. Giảm thiểu khoảng cách về trí tuệ
9. Chuyên môn quan trọng công cụ có sẵn
10. Trước hết phần mềm phải thực hiện đúng sau đó mới tiến hành nâng cấp
11. Kiểm tra code
12. Việc quản lý tốt quan trọng hơn việc có công nghệ tốt
13. Con người là chìa khóa thành công
14. Luôn luôn cẩn thận
15. Có trách nhiệm

**Câu 9** Vẽ biểu đồ minh họa và giải thích các hoạt động cho mô hình tiến trình phát triển phần mềm thác nước (waterfall model).

Mô hình thác nước mô tả quy trình phát triển phần mềm giống như một dòng chảy  với các pha được thực hiện theo trật tự nghiêm ngặt và không có sự quay lui hay nhảy vượt pha là: phân tích yêu cầu, thiết kế, viết code, kiểm thử, hợp nhất và đóng gói sản phẩm.

**Biểu đồ:**

Requirements =>

Design =>

Code =>

Test =>

Integrate and Package

**Ưu điểm:**

* Các giai đoạn được định nghĩa với đầu vào, đầu ra rõ ràng
* Sản phẩm phần mềm được hình thành thông qua chuỗi các hoạt động theo trình tự rõ ràng

**Nhược điểm:**

* Đòi hỏi tất cả yêu cầu phải được xác định rõ ràng ngay từ đầu dự án
* Người dùng không thể tham gia trong suốt thời gian của giai đoạn trung gian từ thiết kế đến kiểm thử.
* Chứa đựng nhiều rủi ro mà chỉ có thể phát hiện ở giai đoạn cuối nên chi phí sửa chữa khi có lỗi có thể rất cao

**Ứng dụng:**

* Yêu cầu được định nghĩa rất rõ ràng, chi tiết, và hầu như không có thay đổi
* Đội ngũ thực hiện quen thuộc và hiểu rõ tất cả yêu cầu của dự án, có kinh nghiệm với các công nghệ được dùng để phát triển phần mềm.
* Dự án xác định hầu như không có rủi ro.

**Câu 10** Vẽ biểu đồ minh họa và giải thích mô hình tiến trình phát triển phần mềm xoắn ốc (spiral model)

Mô hình đặt trọng tâm xem xét rủi ro và xem xét kế hoạch để giải quyết chúng thông qua nhiều chu kì con nối tiếp được lặp liên tiếp dựa trên mô hình của mô hình lặp. Trong mô hình này, việc phân tích và giải quyết những vấn đề có rủi ro cao tập trung vào thiết kế từng khía cạnh cụ thể chứ không dựa vào việc xử lí vấn đề một cách chung chung.

**Biểu đồ:**

Lập kế hoạch => phân tích rủi ro => kỹ nghệ => đánh giá khách hàng

* Lập kế hoạch: xác định mục tiêu, giải pháp, ràng buộc
* Phân tích rủi ro: phân tích các phương án, xác định, giải quyết rủi ro
* Kỹ nghệ: phát triển sản phẩm ở mức tiếp theo
* Đánh giá khách hàng: khẳng định kết quả của kỹ nghệ
  + Với mỗi lần lặp xoắn ốc=> phiên bản hoàn thiện dần
  + Nếu phân tích rủi ro ko chắc chắn trong các yêu cầu => việc làm bản mẫu sẽ sử dụng ở phần kỹ nghệ, các mô hình, mô phỏng khác cũng được làm rõ hơn vấn đề, làm rõ yêu cầu.
  + Khách hàng đánh giá=>thay đổi=> vòng xoáy mới
  + Tại mỗi vòng, cao điểm là việc quyết định tiến hành hay không tiến hành.
  + Kỹ nghệ được thực hiện bằng cách tiếp cận vòng đời và làm bản mẫu

**Ưu điểm:**

* Phân tích đánh giá rủi ro được đẩy lên như một phần thiết yếu trong mỗi vòng xoắn để tăng mức độ tin cậy của dự án
* Kết hợp những tính chất tốt nhất của mô hình thác nước và tiến hóa
* Cho phép thay đổi tùy theo điều kiện thực tế của dự án tại mỗi vòng xoắn

**Nhược điểm:**

* Phức tạp và không phù hợp với mô hình nhỏ và ít rủi ro
* Cần có kĩ năng tốt về phân tích rủi ro

**Ứng dụng:**

* Đội ngũ thực hiện có khả năng phân tích rủi ro
* 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 lí như hệ thống hỗ trợ.

**Câu 11** Cho biết các vấn đề tồn tại của các tiến trình phát triển phần mềm truyền thống. (chapter 5)

* Thời gian phát triển dài
* Không có khả năng đối phó với những thay đổi trong các yêu cầu
* Các yêu cầu phải được hiểu rõ trước khi bắt đầu dự án
* Phụ thuộc quá nhiều vào nỗ lực của nhà phát triển
* Phương pháp phức tạp
* Sự lãng phí, chồng chéo kết quả thu được

**Câu 12** Cho biết các đặc trưng của các phương pháp/quy trình phát triển phần mềm linh hoạt/nhanh (Agile processes). (chapter 5)

* Các phiên bản ngắn và sự lặp đi lặp lại:
  + Phân chia công việc thành các phần nhỏ
  + cho khách hàng càng nhiều càng tốt
* Thiết kế gia tăng
  + Không nên cố gắng hoàn thành bản thiết kế khi chưa hiểu rõ về hệ thống
  + Cố gắng trì hoãn quyết định thiết kế càng lâu càng tốt và cải thiện thiết kế đang tồn tại khi đạt được sự hiểu biết nhất định về nó
* Sự tham gia của người dùng:
  + Nhận được càng nhiều phản hồi từ người dùng càng tốt
* Tối thiểu hóa tài liệu
* Giao tiếp không tuân theo quy định: giúp cho việc giao tiếp trở nên thân mật hơn
* Thay đổi: đảm bảo có sẵn kế hoạch cho sự thay đổi.

**Câu 13** Cho biết các thực hành trong lập trình cực độ. (chapter 5- XP)

1. Lập kế hoạch
   1. Xác định tính năng có trong phiên bản tiếp theo
   2. Ưu tiên nghiệp vụ và các dự đoán kỹ thuật
2. Phiên bản ngắn
   1. Có được một hệ thống làm việc nhanh chóng
   2. Phát hành các phiên bản mới trong chu kì ngắn (2-4 tuần)
   3. Xây dựng kế hoạch chi tiết mới dựa trên phản hồi của khách hàng
3. Ẩn dụ
   1. Dùng một hệ thống các thuật ngữ về đối tượng doanh nghiệp trong việc discuss các vấn đề và các giải pháp cho chúng
4. Thiết kế đơn giản
   1. Loại bỏ sự phức tạp không cần thiết
   2. Thiết kế có thể được thay đổi trong phiên bản tiếp theo
5. Phát triển kiểm thử
   1. Liên tục và tự động
   2. Viết bản kiểm thử trước khi viết code
6. Cải tiến thiết kế
   1. Loại bỏ sự trùng lặp, cải thiện giap tiếp, đơn giản hóa hoặc thêm sự linh hoạt cần thiết
7. Lập trình cặp đôi
   1. Tất cả quá trình phát triển phần mềm đều được thực hiện bởi 2 người trên cùng một máy tính. Điều này có nghĩa là 2 người cùng giải quyết 1 công việc, người này lập trình thì người kia làm giám sát và ngược lại. Khi đó việc trao đổi kinh nghiệm và giải quyết vấn đề sẽ được thực hiện một cách nhanh chóng. Ngoài ra mã nguồn được xem lại liên tục và được chữa lỗi ngay trong quá trình phát triển.
8. Quyền sở hữu tập thể
   1. Bất kì người nào trong đội dự án cũng có thể thay đổi mã trong quá trình làm việc và phải đảm bảo thực hiện thử nghiệm lại toàn bộ sau khi hoàn tất công việc sửa đổi. Điều này sẽ loại bỏ các vấn đề như là sai lệch cấu trúc chương trình,… có thể xảy ra khi một cá nhân mã hóa độc lập.
9. Tương tác liên tục
   1. Các nhóm XP chia công việc ra thành các bước nhỏ và tích hợp mã của họ một vài lần trong một ngày. Do vậy, các vấn đề sẽ được xem xét ngay sau khi thực hiện và có thể dễ dàng sửa chữa khi gặp sự cố. Quá trình này đảm bảo cho mọi người luôn làm việc với phiên bản mới nhất của hệ thống.
10. Tốc độ bền vững
    1. Việc [phát triển phần mềm](https://vi.wikipedia.org/wiki/Ph%C3%A1t_tri%E1%BB%83n_ph%E1%BA%A7n_m%E1%BB%81m) là một công việc sáng tạo, và họ sẽ không thể sáng tạo được nếu họ kiệt sức. Việc giới hạn số giờ làm việc trong tuần sẽ đảm bảo được sức khoẻ của các thành viên và tăng cường chất lượng sản phẩm.
11. Khách hàng trực diện
    1. Các lập trình viên phải luôn tiếp xúc với khách hàng để xác định rõ nhu cầu bất kể nỗ lực tốn bao nhiêu. Các nhà lập trình XP không nên suy đoán các vấn đề cụ thể của một chức năng mà phải hỏi trực tiếp khách hàng.
12. Tiêu chuẩn mã hóa
    1. Đây là một loạt các quy ước về mã hoá để các thành viên của dự án theo đó làm. Khi đó mọi người có thể xem xét lẫn nhau và có thể bàn giao được cho nhau.

**Câu 14** Liệt kê và mô tả các bước cơ bản trong quá trình xây dựng yêu cầu phần mềm. (chapter 6)

* Làm rõ yêu cầu: giao tiếp với khách hàng và người dùng để xác định yêu cầu của họ
* Phân tích yêu cầu: xem xét các yêu cầu được đặt ra có ở tình trạng rõ ràng, không hoàn chỉnh, đa nghĩa hoặc mâu thuẫn hay không, và giải quyết các vấn đề đó
* Định nghĩa, tạo bản mẫu, xét duyệt yêu cầu:
* Đặc tả yêu cầu: mô tả những gì mà phần mềm sẽ làm và làm như thế
* Thỏa thuận và công nhận

*Prototyping (tạo bản mẫu) là quá trình nhanh chóng thực hiện cùng với một mô hình làm việc để mà kiểm tra các mặt khác nhau của thiết kế, làm rõ ý tưởng hoặc các đặc tính và lấy lại sớm phản hồi của người dùng*

**Câu 15** Cho biết ba công việc chính cần thực hiện trước khi xây dựng yêu cầu - phần

mềm. (chapter 6)

* Lập kế hoạch cho kĩ nghệ phần mềm
  + Xác định quy trình được sử dụng
  + Các nguồn lực cần có
  + Lịch trình hoàn thành các công việc yêu cầu
* Thỏa thuận về nguồn lực, tiến trình, lịch trình cho các công việc yêu cầu
* Sử dụng và tổ chức thực hiện các thỏa thuận về nguồn lực và quy trình.

**Câu 16** Mô tả các mục cần xem xét khi xác định mức độ ưu tiên của các yêu cầu. (chapter 6)

* Nhu cầu của người dùng hoặc khách hàng hiện tại
* Cạnh tranh và điều kiện hiện tại của thị trường
* Dự đoán về nhu cầu mới của khách hàng trong tương lai
* Lợi thế về kinh doanh
* Các vấn đề quan trọng của sản phẩm hiện tại

**Câu 17** Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc đường ống và bộ lọc (Pipe and filter) (chapter 7)

Bao gồm một loạt các quá trình kết nối liên tiếp nhau, đầu vào của quá trình này là đầu ra của quá trình kia.

**Cấu trúc:**

**Bộ lọc:**

* Vừa là nơi tiếp nhận vừa là nơi nhận dữ liệu
* Hoạt động độc lập, không phụ thuộc nơi nhận và gửi dữ liệu
* Làm phong phú thêm dữ liệu từ dữ liệu được lưu trữ hoặc các giá trị đã được tính toán
* Lọc dữ liệu đầu vào, đưa ra dữ liệu không phù hợp, sắp xếp dữ liệu

**Đường ống:**

* Là kênh kết nối 2 chiều có nhiệm vụ vận chuyển dữ liệu giữa 2 bộ lọc khác nhau
* Đường ống ở đây có thể là kết nối giữa luồn dữ liệu trong một tiến trình, giữa 2 chương trình trên một máy tính, giữa hai quá trình trên hệ thống phân tán Internet.

**Đặc điểm:**

* Thích hợp cho các hệ thống có nhiệm vụ chủ yếu là biến đổi dữ liệu
* Sử dụng một mạng lưới các phép biến đổi để đạt được kết quả mong muốn
* có khả năng xử lí song song trong môi trường đa nhiệm
* Có tính linh hoạt thông qua sự trao đổi giữa các bộ lọc, sự tái tổ hợp
* Tạo mẫu nhanh
* Khó khăn trong việc xử lí lỗi
* Không hiệu quả trong môi trường đơn nhiệm.

**Câu 18** Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc Khách-Chủ (Client-Server). (chapter 7)

Là loại kiến trúc mà trong đó các thành phần (chủ-khách) tương tác với nhau thông qua các giao thức cơ bản về mạng hoặc thông qua các cuộc gọi thủ tục từ xa.

**Cấu trúc:**

Server: một máy chủ hay một ứng dụng có khả năng cung cấp dịch vụ cho các máy khách.

Client: phần mềm, ứng dụng cung cấp giao diện người dùng để truy cập, sử dụng dịch vụ từ server.

**Đặc điểm:**

* Phân tán dữ liệu trực tiếp, tường minh
* Sử dụng hiệu quả mạng, dùng chung tài nguyên
* Dễ thêm máy chủ và nâng cấp máy chủ
* Không chia sẻ mô hình dữ liệu -> các hệ thống con dùng dữ liệu khác nhau. Trao đổi dữ liệu khó khăn, không hiệu quả
* Quản lí dư thừa trên các máy chủ
* Không có lưu trữ tập trung tên và dịch vụ -> khó tìm một dịch vụ, cùng với một máy chủ sẵn sàng

**Câu 19** Cho biết cấu trúc và các đặc điểm của kiểu kiến trúc dữ liệu dùng chung (Shared-Data). (chapter 7 –silde)

**Cấu trúc:** có 2 thành phần:

* Kho lưu trữ dữ liệu: cung cấp dữ liệu đáng tin cậy
* Người truy cập dữ liệu: truy cập dữ liệu từ kho, thực hiện tính toán, có thể đưa ra kết quả trở lại
* Việc giao tiếp giữa những người truy cập chỉ có thể thông qua kho lưu trữ

**Đặc điểm:** có 2 loại biến thể:

* Bảng đen: nếu dữ liệu được đăng lên kho lưu trữ, tất cả người truy cập sẽ nhận được thông báo
* Kho lưu trữ: người dùng phải truy cập vào kho để nhận dữ liệu mới nếu có.
* Chỉ có thể đọc/ghi

**Câu 20** Liêt kê và mô tả các bước trong thiết kế cơ sở dữ liệu. (chapter 7 page 155)

* Mô hình hóa dữ liệu: mô hình liên kết thực thể
* Thiết kế cơ sở dữ liệu logic: biểu diễn thực thể bằng bảng, xác định thuộc tính được chọn làm khóa chính, khóa ngoại, thiết lập mối quan hệ giữa các thực thể
* Thiết kế cở sở dữ liệu vật lí: xác định loại dữ liệu cho mỗi thuộc tính, xác định thuộc tính cần chỉ rõ (thường là thuộc tính được chọn làm khóa chính)
* Triển khai và duy trì

**Câu 21** Mô tả ba trong số các loại biểu đồ UML trong phân tích và thiết kế hướng đối tượng.

UML là viết tắt của Unified Modeling Language được sử dụng trong kỹ thuật phần mềm hướng đối tượng, được sử dụng để mô hình hóa một cấu trúc, hành vi và thậm chí là cả quy trình nghiệp vụ.

* Biểu đồ lớp:
  + Cho ta thấy các lớp trong một hệ thống, các thuộc tính, hoạt động và mối quan hệ giữa các lớp (được sử dụng nhiều nhất trong các loại mô hình UML), mối quan hệ giữa các lớp được thể hiện trên các mũi tên liên kết.
  + Cấu trúc của một lớp gồm: tên lớp, các thuộc tính, các hoạt động và các phương pháp thực hiện
* Biểu đồ ca sử dụng:
  + Cung cấp cái nhìn tổng quan về các bên liên quan trong một hệ thống
  + Các chức năng cần thiết cho người dùng và cách tương tác giữa người dùng với các chức năng đó
* Biểu đồ tuần tự:
  + Mô tả sự tương tác của các đối tượng theo trình tự về thời gian
  + Có sự liên kết chặt chẽ với biểu đồ lớp
  + Mỗi biểu đồ tuần tự mô tả một tình huống xử lý

**Câu 22** Trình bày cách tính độ đo dung lượng (Volume metric) của Halstread để phân tích độ phức tạp của chương trình nguồn (theo kích cỡ). (chapter 7 - slide)

n1 = số toán tử phân biệt

n2 = số toán hạng phân biệt

N1 = tổng số toán tử

N2 = tổng số toán hạng

Các độ đo:

n = n1 + n2 : số từ vựng

N = N1 + N2 : độ dài chương trình

Độ lớn của chương trình: V = Nlog2n (diễn dịch là số bits tối thiểu)

Cấp độ chương trình L = V\*/V, V\* là biểu diễn thu gọn nhất của giải thuật đang xét. L được tính xấp xỉ L’= (2/n1)(n2/N2)

Công sức viết chương trình E =V/L

Ví dụ:

1. Procedure sort(var x:array, n:integer)
2. Var I,j :integer;
3. Begin
4. For i:=2 to n do
5. For j:=1 to I do
6. If x[i]<x[j] then begin
7. Save:=x[i];
8. X[i]:=x[j];
9. X[j]:=save;
10. End
11. End;

Operator: procedure 1, sort() 1, var 2, : 3, array 1, ; 6, integer 2, “,” 2, begin end 2, for do 2, if then 1, := 5, < 1,[] 6

Operand: x 7, n 2, I 6, j 5, save 3, “2” 1, “1” 1

n1 = 14, n2 = 7, N1=35, N2 =25

n= 21, N = 60, V=264, L=0.044, L’=0.040, E =6000

**Câu 23** Trình bày các cấp độ ghép nối (coupling) giữa các module. (chapter 8)

Độ gắn kết chỉ mức độ phụ thuộc giữa các module, các mức độ gắn kết là

* Gắn kết nội dung: module này làm thay đổi dữ liệu của module khác
* Gắn kết chung: chia sẻ dữ liệu
* Gắn kết ngoài: 2 module trao đổi dữ liệu thông qua một phương tiện, vd như file
* Gắn kết điều khiển: module chuyển điều khiển thực hiện cho module khác
* Gắn kết nhãn: 2 module chia sẻ cùng một cấu trúc dữ liệu
* Gắn kết dữ liệu: dữ liệu được chuyển từ một module này sang module khác

**Câu 24** Trình bày các cấp độ kết dính (cohension) giữa các phần tử trong một module. (chapter 8)

Tính kết dính chỉ ra mức độ gắn kết giữa các module, các cấp độ kết dính:

* Kết dính ngẫu nhiên: các thành phần không liên quan với nhau
* Kết dính logic: module làm nhiều chức năng có liên quan logic với nhau, ví dụ module chứa tất cả các thành phần input
* Kết dính theo thời gian: các thành phần khác nhau cùng được khởi tạo tại 1 thời điểm
* Kết dính thủ tục: nhiều thành phần độc lập thực hiện theo trình tự
* Kết dính giao tiếp: nhiều thành phần độc lập thực hiện trên cùng một dữ liệu
* Kết dính tuần tự: nhiều thành phần độc lập thực hiện theo trình tự, cái này làm đầu vào cho cái kia
* Kết dính chức năng: các thành phần gắn kết nhau tạo nên chức năng đơn nhất của module

**Câu 25** Trình bày 8 quy tắc thiết kế giao diện người dùng của Shneiderman và Plaisant.

1. Đảm bảo tính thống nhất: yêu cầu hành động nhất quán trong các tình huống tương tự nhau (thao tác, thuật ngữ, thông báo,…)
2. Cho phép người dùng sử dụng các phím tắt: hỗ trợ phím tắt là yêu cầu hết sức cần thiết để nâng cao hiệu quả của hệ thống
3. Cung cấp phản hồi: với một hệ thống tương tác thì phản hồi gần như là bắt buộc, người dùng muốn biết tác động của họ tới hệ thống như thế nào
4. Chia nhỏ hành động để thấy được kết quả ngắn hạn: thứ tự của các hành động cần được tổ chức thành các nhóm với một khởi đầu, giữa, và kết thúc. Ví dụ khi tải một tệp tin, chương trình thông báo đã tải được bn %, thời gian dự kiến hoàn tất làm người dùng an tâm và có kế hoạch cho việc khác
5. Dễ dàng khắc phục lỗi: không có hệ thống nào không phát sinh lỗi nên việc tạo ra cách thức khắc phục lỗi phát sinh là cần thiết
6. Cho phép đảo ngược hành động: tác dụng của nút “undo”
7. Con người làm chủ: hãy để con người yêu cầu và máy đáp ứng hơn là bắt họ tìm cách phản ứng lại những hành động do chương trình đặt ra
8. Giảm tải bộ nhớ ngắn hạn: đơn giản hóa màn hình, dữ liệu, sự kiện trong một khoảng thời gian ngắn hạn.

**Câu 26** Cho biết các lỗi lập trình thường gặp. (chapter 9)

Lỗi rò rỉ bộ nhớ: bộ nhớ được cấp phát cho chương trình mà không được giải phóng

Giải phóng tài nguyên đã được giải phóng: lập trình viên cố gắng giải phóng cho tài nguyên đã được giải phóng trước đó.

Đích null: truy cập đến vị trí null (do khởi tạo không đúng hoặc thiếu sự khởi tạo dẫn đến lỗi)

Thiếu địa chỉ duy nhất: trong nhiều trường hợp chúng ta trỏ đến địa chỉ đã có trong khi chúng ta cần địa chỉ khác

Lỗi chỉ số mảng vượt quá giới hạn: chỉ số mảng vượt quá kích thước đã cấp phát

Ngoại lệ toán học: ví dụ 4/0

Thiếu một: khi lập trình với mảng chúng ta nên bắt đầu từ 0 thay vì 1

Lỗi đồng bộ hóa: xảy ra khi nhiều luồng xử lí dùng chung 1 tài nguyên: cùng chờ tài nguyên do của luồng xử lí kia, cùng thay đổi tài nguyên 1 lúc, cố giữ tài nguyên

Tràn bộ đệm: xảy ra khi cấp phát bộ nhớ không đủ dẫn đến tràn sang vùng nhớ khác

**Câu 27** Liệt kê 10 thông lệ lập trình.

1. Làm việc từ những cái căn bản nhất: hiểu rõ mọi thứ ở mức khái niệm là chìa khóa để thành công
2. Luôn đặt ra nhưng câu hỏi (tại sao? Như thế nào?) với mọi đoạn code mình viết ra
3. Học được nhiều hơn từ việc giúp đỡ người khác: việc hiểu những vấn đề của người khác trong ngữ cảnh của họ, tìm hiểu và cung cấp những giải pháp cho các vấn đề đó giúp lập trình viên học được rất nhiều
4. Viết những dòng code đơn giản, dễ hiểu, có logic
5. Dành nhiều thời gian để phân tích vấn đề, như vậy sẽ tiết kiệm thời gian sửa lỗi
6. Hãy là người đầu tiên phân tích và xem xét những dòng code của chính mình
7. Đừng hốt hoảng khi thấy sự thay đổi chóng mặt của thế giới công nghệ: mọi thay đổi đều dựa trên các vấn đề cốt lõi, vì vậy hãy nắm vững các vấn đề cốt lõi
8. Giải pháp tạm thời sẽ không có giá trị lâu dài: chỉ thực thi một công việc khi đã nắm rõ đầu vào, đầu ra của giải pháp
9. Hãy đọc thật nhiều tài liệu: giúp bạn tạo ra những nền tảng vững chắc từ đó bạn sẽ lập trình theo cách tốt nhất
10. Bạn có thể học từ những dòng code của người khác: việc đọc và tham khảo những dữ án mã nguồn mở đáng tin cậy giúp bạn lập trình tốt hơn

**Câu 28** Liệt kê các chuẩn lập trình Java. (chapter 8-slide)

Các quy ước về việc viết code hay còn gọi là chuẩn lập trình trong java hay tất cả các ngôn ngữ lập trình khác có vai trò rất quan trọng vì:

* 80% chi phí cho vòng đời của một phần mềm là dành cho việc bảo trì
* Hầu như tất cả các phần mềm đều được bảo trì bởi chính người viết ra phần mềm đó
* Cải thiện tính dễ đọc của phần mềm, cho phép lập trình viên có thể hiểu code một cách nhanh chóng và rõ ràng

**Các chuẩn lập trình Java:**

**Quy ước đặt tên**

* Tên gói phải ở viết thường

mypackage, edu.iitk.maths

* Nhập tên nên là danh từ bắt đầu với chữ hoa

Day, DateOfBirth, EventHandler

* Tên biến nên là danh từ viết chữ thường

tên, số lượng

* Biến với phạm vi lớn nên có tên dài; biến với phạm vi nhỏ tên ngắn
* Vòng lặp loop nên khai báo biến i, j, k ...
* Hằng số liên tục phải được tất cả chữ hoa

PI, MAX\_INTERACTIONS

* Tên phương pháp nên là động từ bắt đầu với chữ thường

getValue ()

* Biến của lớp riêng nên có hậu tố \_

tin int value\_

* Tiền tố is nên được sử dụng cho các phương pháp Boolean

isStatus,

* Tránh tên biến Boolean tiêu cực

isNotCorrect

* Sử dụng compute cho các phương pháp tính toán một cái gì đó

computeMean ()

* Sử dụng find cho các phương pháp tìm một cái gì đó

findMin ()

* Các lớp ngoại lệ nên được hậu tố với ngoại lệ

OutOfBoundException

* Các tập tin

Các file nguồn cần phải có phần mở rộng java

Mỗi tập tin nên chứa một lớp bên ngoài và tên nên được đặt giống như tập tin

Chiều dài dòng nên được ít hơn 80, nếu còn tiếp tục xuống dòng khác

* Nên tránh dùng các kí tự đặc biệt

Báo cáo

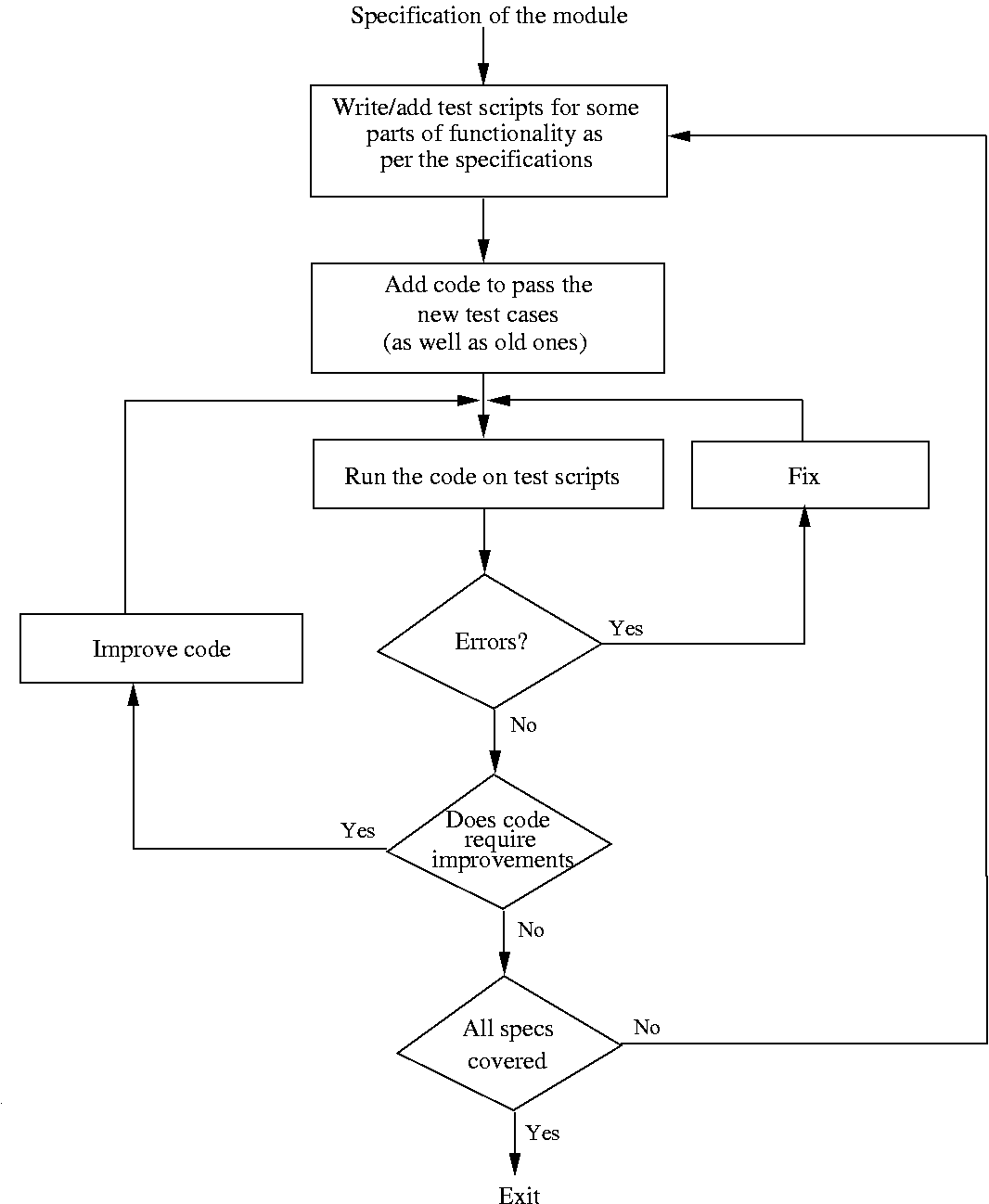
* Biến phải được khởi tạo, nơi đã kê khai trong phạm vi nhỏ nhất có thể
* Khai báo các biến liên quan với nhau; biến không liên quan cần được khai báo riêng
* Biến riêng của lớp không bao giờ được công bố công khai
* Biến lặp chỉ nên được khởi tạo trước vòng lặp
* Tránh sử dụng break và continue trong vòng
* Tránh sử dụng các báo cáo có thể thực hiện được trong câu điều kiện
* Tránh sử dụng cấu trúc do…while
* Tránh biểu thức điều kiện phức tạp, có thể sử dụng các biến Boolean thay thế tạm thời
* Tránh báo cáo thực thi trong điều kiện

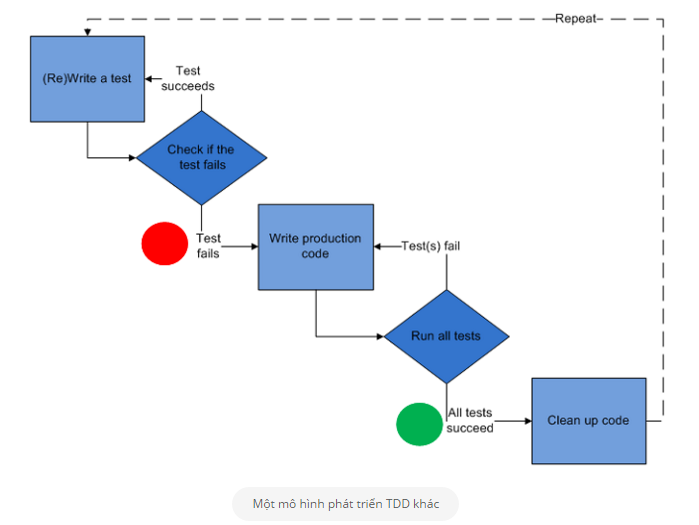
Ý kiến bình luận ​​và cách bố trí

* Dòng ý kiến bình luận cho một khối code nên được đặt thẳng hàng với khối code đó
* Nên có bình luận cho tất cả các biến chính và giải thích chúng tượng trưng cho cái gì
* Một khối comment nên bắt đầu với một dòng với / \* và kết thúc bằng một dòng có dấu \* /
* Những bình luận tiếp theo ​​sau khi báo cáo nên ngắn và trên cùng một dòng

**Câu 29** Vẽ và mô tả biểu đồ biểu diễn tiến trình lập trình hướng kiểm thử (test driven development). (chapter 8-slide && chapter 10)

Mô hình TDD (có nhiều loại mô hình khác nhau)





TDD là một quy trình viết mã mà lập trình viên sẽ thực hiện thông qua các bước nhỏ và tiến độ được đảm bảo liên tục bằng cách viết và chạy các bài test tự động. Quá trình lập trình TDD cực kì chú trọng vào các bước liên tục sau:

1. Khởi tạo 1 testcase cho hàm mới, đảm bảo rằng test sẽ fail
2. Chuyển qua viết code sơ khai nhất cho hàm đó để testcase có thể pass
3. Tối ưu hóa đoạn code vừa viết sao cho đảm bảo test vẫn pass và tối ưu nhất cho việc lập trình kế tiếp
4. Lặp lại cho các hàm khác từ bước 1

**3 quy tắc khi áp dụng TDD:**

1. Không cho phép viết bất kì một mẫ chương trình nào cho tới khi nó làm một test bị fail trở nên pass
2. Không cho phép viết nhiều hơn một unit test mà nếu chỉ cần 1 unit test đã đủ để fail. Hãy chuyển sang viết code function để pass test đó trước
3. Không viết nhiều hơn một mã chương trình mà nó đã đủ làm 1 test bị fail chuyển sang pass

**Các lỗi thường gặp:**

* Không quan tâm đến các test bị fail
* Quên đi thao tác tối ưu sau khi viết code cho test pass
* Đặt tên các test khó hiểu và tối nghĩa
* Viết một test với kịch bản quá phức tạp
* Không bắt đầu từ các tes đơn giản nhất

**Câu 30** Liệt kê 10 mục cần kiểm tra khi thanh tra chương trình (http://www.the-software-experts.com/e\_dta-sw-test-inspection.php)

* Một báo cáo về mục tiêu của việc thanh tra ( bắt buộc )
* Các sản phẩm phần mềm được thanh tra ( bắt buộc )
* Thủ tục thanh tra tài liệu ( bắt buộc )
* Mẫu báo cáo thanh tra ( bắt buộc )
* Danh sách những vấn đề hoặc điều bất thường hiện tại ( bắt buộc )
* Danh sách các mục cần thanh tra (đánh dấu tick)( bắt buộc )
* Bất kỳ quy định , tiêu chuẩn , hướng dẫn , kế hoạch và thủ tục khác với sản phẩm phần mềm được khảo sát ( nên)
* Thông số kỹ thuật phần cứng của sản phẩm( nếu có)
* Dữ liệu về hiệu suất phần cứng ( nếu có)
* Phân loại điều bất thường ( nên)

**Câu 31** Cho biết các cách phân lớp của các kỹ thuật kiểm thử phần mềm. (chapter 9 slide (slide 8), chapte10-book)

* Ai tiến hành kiểm thử?
  + Lập trình viên
  + Kiểm thử viên
  + Người dùng
* Kiểm thử những gì?
  + Kiểm thử đơn vị
  + Kiểm thử chức năng
  + Kiểm thử tích hợp và kiểm thử hệ thống
  + Kiểm thử giao diện người dùng
* Tại sao phải tiến hành kiểm thử
  + Kiểm thử chấp nhận
  + Kiểm thử cấu hình
  + Kiểm thử thích nghi
  + Kiểm thử hiệu quả
  + Kiểm thử khả năng chịu tải
  + Kiểm thử giao diện người dùng
* Kiểm thử như thế nào
  + Dựa vào trực giác
  + Dựa vào đặc tả (kiểm thử hộp trắng)
  + Dựa vào code (kiểm thử hộp đen)
  + Ca kiểm thử đang dùng (kiểm thử hồi quy)
  + Lỗi
    - Error guessing
    - Error-prone analysis

**Câu 32** Cho biết 4 cấp độ kiểm thử phần mềm.(chapter 9-slide)

* Kiểm thử đơn vị
  + Mục đích là đảm bảo thông tin được xử lý và xuất (ra khỏi Unit) là chính xác, trong mối tương quan với giữa liệu nhập và chức năng của Unit
  + Một đơn vị là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được, như các hàm, thủ tục, lớp, phương thức.
  + Việc kiểm thử được tiến hành dễ dàng vì đơn vị kiểm thử thường có kích thước nhỏ, hoạt động đơn giản
  + Thường do lập trình viên thực hiện
* Kiểm thử tích hợp
  + Kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành.
  + Mục đích là phát hiện lỗi giao tiếp xảy ra giữa các Unit, tích hợp các Unit đơn lẻ thành các hệ thống con và hệ thống hoàn chinh, chuẩn bị cho kiểm thử ở mức hệ thống.
  + Nên kiểm thử tích hợp dần dần từng Unit để giảm số lượng ca kiểm thử và sai sót
  + Có 4 loại kiểm thử tích hợp:
    - Kiểm thử cấu trúc: tương tự kiểm thử hộp trắng, đảm bảo các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn như các câu lệnh và nhánh bên trong
    - Kiểm thử chức năng: tương tự kiểm thử hộp đen, chỉ chú trọng đến chức năng của chương trình mà không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kĩ thuật
    - Kiểm thử hiệu năng: kiểm thử việc vận hành của hệ thống
    - Kiểm thử khả năng chịu tải: kiểm thử các giới hạn của hệ thống
* Kiểm thử hệ thống
  + Mục đích là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không
  + Tốn nhiều công sức và thời gian, nhiều trường hợp cần đến công cụ, phần mềm hỗ trợ, phần cứng đặc thù.
  + ở mức độ hệ thống, người kiểm thử cũng tìm kiếm các lỗi nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn bộ hệ thống.
  + thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án
* Kiểm thử chấp nhận sản phẩm
  + Được khách hàng thực hiện hoặc ủy quyền cho một nhóm thứ 3
  + Mục đích là chứng minh phần mềm thỏa mãn tất cả yêu cầu của khác hàng và khách hàng chấp nhận sản phẩm
  + Có 2 loại kiểm thử là:
    - Kiểm thử alpha: người dùng kiểm thử tại nơi phát triển phần mềm, lập trình viên ghi nhận các lỗi hoặc phản hồi, lên kế hoạch sửa chữa
    - Kiểm thử beta: phần mềm sẽ được gửi cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ được gửi ngược lại cho lập trình viên để sửa chữa
  + Nếu khách hàng không tham gia vào quá trình phát triển phần mềm thì kết quả kiểm thử sẽ sai lệch rất lớn mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó, nguyên nhân có thể do hiểu sai yêu cầu hoặc sự mong chờ của khách hàng.

**Câu 33** Trình bày kỹ thuật kiểm thử phần mềm phân lớp tương đương (Equivalence Class Partitioning). (chapter 10)

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Phương pháp này cố gắng xác định ra một ca kiểm thử mà làm lộ ra một lớp lỗi, do đó làm giảm tổng số các trường hợp kiểm thử phải được xây dựng.

Thiết kế ca kiểm thử cho phân lớp tương đương dựa trên sự đánh giá về các lớp tương đương với một điều kiện vào. Lớp tương đương biểu thị cho tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào.

Một cách xác định tập con này là để nhận ra rằng 1 ca kiểm thử được lựa chọn tốt cũng nên có 2 đặc tính khác:

1. Giảm thiểu số lượng các ca kiểm thử khác mà phải được phát triển để hoàn thành mục tiêu đã định của kiểm thử “hợp lý”.
2. Bao phủ một tập rất lớn các ca kiểm thử có thể khác. Tức là, nó nói cho chúng ta một thứ gì đó về sự có mặt hay vắng mặt của những lỗi qua tập giá trị đầu vào cụ thể.

Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

(1). Xác định các lớp tương đương và

(2). Xác định các ca kiểm thử.

##### Xác định các lớp tương đương

Các lớp tương đương được xác định bằng bằng cách lấy mỗi trạng thái đầu vào (thường là 1 câu hay 1 cụm từ trong đặc tả) và phân chia nó thành 2 hay nhiều nhóm (có thể sử dụng bảng 2.3 để liệt kê các lớp tương đương).

###### Hình 2.3 Một mẫu cho việc liệt kê các lớp tương đương

|  |  |  |
| --- | --- | --- |
| Điều kiện bên ngoài | Các lớp tương đương hợp lệ | Các lớp tương đương không hợp lệ |
|  |  |  |

Chú ý là hai kiểu lớp tương đương được xác định: lớp tương đương hợp lệ mô tả các đầu vào hợp lệ của chương trình, và lớp tương đương không hợp lệ mô tả tất cả các trạng thái có thể khác của điều kiện (ví dụ, các giá trị đầu vào không đúng). Với 1 đầu vào hay điều kiện bên ngoài đã cho, việc xác định các lớp tương đương hầu như là 1 quy trình mang tính kinh nghiệm. Để xác định các lớp tương đương c có thể áp dụng tập các nguyên tắc dưới đây:

1. Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.
2. Nếu 1 trạng thái đầu vào xác định số giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương bất hợp lệ.
3. Nếu 1 trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau, xác định 1 lớp tương đương hợp lệ cho mỗi loại và 1 lớp tương đương không hợp lệ.
4. Nếu 1 trạng thái đầu vào chỉ định một tình huống “chắc chắn – must be”, xác định 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ.

Nếu có bất kỳ lý do nào để tin rằng chương trình không xử lý các phần tử trong cùng 1 lớp là như nhau, thì hãy chia lớp tương đương đó thành các lớp tương đương nhỏ hơn.

##### Xác định các ca kiểm thử

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các ca kiểm thử. Quá trình này như sau:

1. Gán 1 số duy nhất cho mỗi lớp tương đương.
2. Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi (hợp nhất thành) các ca kiểm thử, viết 1 ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó chưa được bao phủ càng tốt.
3. Cho đến khi các ca kiểm thử của bạn đã bao phủ tất cả các lớp tương đương không hợp lệ, viết 1 ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.
4. Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác.

Mặc dù việc phân lớp tương đương là rất tốt khi lựa chọn ngẫu nhiên các ca kiểm thử, nhưng nó vẫn có những thiếu sót. Ví dụ, nó bỏ qua các kiểu test – case có lợi nào đó. Hai phương pháp tiếp theo, phân tích giá trị biên và đồ thị nguyên nhân – kết quả , bao phủ được nhiều những thiếu sót này.

**Câu 34** Trình bày kỹ thuật kiểm thử phần mềm phân tích giá trị biên (Boundary Value Analysis). (chapter 10)

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỷ lệ phần trăm cao hơn các ca kiểm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra. Phân tích các giá trị biên là phương pháp thiết kế ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh:

1. Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình, mà nó yêu cầu là 1 hay nhiều phần tử được lựa chọn như vậy mà mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm tra.
2. Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào (không gian đầu vào), các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

Phân tích giá trị biên yêu cầu óc sáng tạo và lượng chuyên môn hóa nhất định và nó là một quá trình mang tính kinh nghiệm rất cao. Tuy nhiên, có một số quy tắc chung như sau:

1. Nếu 1 trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết các ca kiểm thử cho các giá trị cuối của giới hạn, và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vừa ra ngoài phạm vi.
2. Nếu 1 trạng thái đầu vào định rõ số lượng giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên, một giá trị dưới những giá trị này.
3. Sử dụng quy tắc 1 cho mỗi trạng thái đầu vào. Ví dụ, nếu 1 chương trình tính toán sự khấu trừ FICA hàng tháng và nếu mức tối thiểu là 0.00$, và tối đa là 1,165.25$, hãy viết các ca kiểm thử mà khấu trừ 0.00$ và 1,165.25, khấu trừ âm và khấu trừ lớn hơn 1,165.25$. Chú ý là việc xem xét giới hạn của không gian kết quả là quan trọng vì không phải lúc nào các biên của miền đầu vào cũng mô tả cùng một tập sự kiện như biên của giới hạn đầu ra (ví dụ, xét chương trình con tính SIN). Ngoài ra, không phải lúc nào cũng có thể tạo ra 1 kết quả bên ngoài giới hạn đầu ra, nhưng tuy nhiên rất đáng để xem xét tiềm ẩn đó.
4. Sử dụng nguyên tắc 2 cho mỗi trạng thái đầu ra.
5. Nếu đầu vào hay đầu ra của 1 chương trình là tập được sắp thứ tự ( ví dụ,1 file tuần tự hay 1 danh sách định tuyến hay 1 bảng) tập trung chú ý vào các phần tử đầu tiên và cuối cùng của tập hợp.
6. Sử dụng sự khéo léo của bạn để tìm các điều kiện biên.

**Câu 35** Trình bày kỹ thuật kiểm thử phần mềm phân tích đường dẫn (chapter 10)

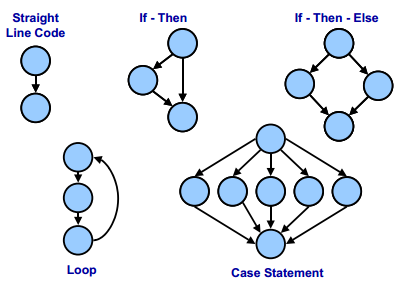
Kiểm thử phân tích đường dẫn là một kỹ thuật kiểm thử hộp trắng do Tom McCabe đề xuất. Phương pháp đường dẫn cơ sở cho phép người thiết kế trường hợp kiểm thử thực hiện phép đo độ phức tạp logic của thiết kế thủ tục và sử dụng phép đo này như một chỉ dẫn cho việc thiết kế một tập cơ sở các đường dẫn thực hiện. Những trường hợp kiểm thử được suy diễn để thực hiện tập cơ sở. Các trường hợp kiểm thử đó được đảm bảo để thực hiện mỗi lệnh trong chương trình ít nhất một lần trong quá trình kiểm thử.

Kiểm thử đường dẫn có 2 nhiệm vụ:

* Phân tích số lượng đường dẫn đang có trong hệ thống hoặc chương trình
* Xác định xem có bao nhiêu đường dẫn nên có trong phép kiểm thử

Kỹ thuật kiểm thử này gồm 4 bước:

* Vẽ đồ thị lưu trình
  + Các cấu trúc đồ thị lưu trình cơ bản:



* + Trên một đồ thị lưu trình chúng ta có thể thấy:
    - Các mũi tên hoặc cung tròn biểu diễn dòng chảy của sự điều chỉnh
    - Vòng tròn hoặc các nút biểu diễn các hành động
    - Các khu vực bị giới hạn bởi các đường viền và các nút gọi là các vùng
    - Một nút điều kiện là nút có chứa một điều kiện
* Xác định độ phức tạp Cyclomatic
  + Độ phức tạp cyclomatic là thước đo phần mềm dùng để chỉ ra sự phức tạp của một chương trình.
  + Xác định độ phức tạp cyclomatic cho biết số lượng đường dẫn độc lập trong một tập cơ sở của chương trình và cho chúng ta biết giới hạn trên số lượng kiểm thử bắt buộc để đảm bảo rằng tất cả các câu lệnh được thực hiện ít nhất một lần.
  + Một đường dẫn độc lập là một đường dẫn bất kỳ trong chương trình đưa ra ít nhất một tập lệnh xử lý hoặc điều kiện mới.
  + Có 3 công thức tính độ phức tạp cyclomatic:
    - V = R, R là số vùng của đồ thị lưu trình
    - V= P + 1, P là số đỉnh điều kiện có trong đồ thị lưu trình
    - V = E – N + 2, E là số cạnh, N là số đỉnh của đồ thị lưu trình.
* Xác định tập cơ sở các đường dẫn độc lập tuyến tính:
  + Xác định tất cả các đường dẫn độc lập có trong đồ thị
* Chuẩn bị các trường hợp kiểm thử có khả năng thực hiện mỗi đường dẫn trong tập cơ sở
  + Thiết kế các trường hợp kiểm thử cho mỗi đường dẫn độc lập trong tập cơ sở đã chọn.

**Câu 36** Liệt kê 10 sản phẩm được quản lý cấu hình. (chapter 10-slide)

1. Đặc tả yêu cầu phần mềm
2. Tài liệu về thiết kế
3. Các file mã nguồn
4. Kịch bản kiểm thử
5. Các đoạn mã có thể thực thi được
6. Các bảng cơ sở dữ liệu
7. Dữ liệu khởi tạo
8. Cuộc gọi vấn đề người dùng
9. Các vấn đề cần sửa chữa
10. Tài liệu người dùng

**Câu 37** Cho biết vai trò của quản lý cấu hình và các chức năng cơ bản của quản lý cấu hình. (http://salmanawan.com/what-is-software-configuration-management-and-its-functions/)

**Vai trò:** Mục đích của quản lí cấu hình là để thiết lập và bảo đảm tính toàn vẹn của các sản phẩm trung gian cũng như các sản phẩm sau cùng của một dự án phần mềm, xuyên suốt chu kỳ sống của dự án đó.

**Các chức năng cơ bản của quản lý cấu hình:**

* Xác định các sản phẩm: các sản phẩm có thể là các tập tin mã nguồn, thư viện của bên thứ ba, các thành phần của bên thứ ba, các phương tiện truyền thông, kịch bản cơ sở dữ liệu…
* Kiểm soát các sản phẩm: đảm bảo kiểm soát việc bổ sung, cập nhật, xóa các sản phẩm này để các thành phần quan trọng của hệ thống không bị mất trong trường hợp cần thiết. Việc kiểm soát cũng liên quan đến cấp giấy phép và quyền cho các cá nhân, hanh động khác nhau có thể thực hiện được trên các sản phẩm này.
* Ghi lại các hành động: trong trường hợp phát sinh lỗi, các tác động trên mỗi sản phẩm phải được tìm thấy, việc ghi lại các hành động này có thể được lưu trong các file log hay việc quản lí một cách bao quát dòng thời gian của sự thay đổi
* Báo cáo về thay đổi: chỉ có quá trình ghi lại là không đủ, bộ quản lý cấu hình phải có khả năng tìm kiếm thông qua các chi tiết được ghi lại khi có một cái gì đó cần quay lại trạng thái trước đó bất cứ lúc nào.
* Thẩm tra tình trạng hệ thống bảo đảm đầy đủ, chính xác và ổn định của hệ thống phần mềm tại một thời điểm nhất định. Đây giống như một hệ thống cảnh bảo sớm cho các nhà quản lý phần mềm, do đó họ biết được bất kỳ sự khác biệt nào trong các sản phẩm cấu thành

**Câu 38** Cho biết mô hình lưu trữ và truy cập các thành phần (Parts Storage and Access Model) trong quản lý cấu hình và các tính năng chính của mô hình này. (chapter 11)

Mô hình lưu trữ và truy cập các thành phần xác định các cơ sở hạ tang vật lí cần thiết cho việc lưu trữ và kiểm soát việc truy cập của tất cả bộ tạo phần mềm.

Có hai chức năng cơ bản:

* Tạo thành: các hàm tạo thành cung cấp khả năng để bước đầu tạo ra một bộ tạo mới và lưu trữ nó cho việc quản lí cấu hình
* Xóa bỏ: các hàm xóa phá hủy các bộ tạo và loại bỏ nó

Chức năng truy cập:

* Xem: các hàm xem cho phép người dùng lấy ra các bộ tạo với mục đích là đọc
* Chỉnh sửa: cho phép người dùng lấy ra các bộ tạo với mục đích là làm thay đổi bộ tạo
* Phục hồi: cho phép các bộ tạo đã bị thay đổi được lưu trữ lại với các thay đổi mới.

Chức năng phục vụ và kiểm soát: (đọc thêm sách trang 250)

* Import
* export
* list
* set
* increment
* change
* gather
* merge
* promte
* compare

**Câu 39** Cho biết các công việc cần thực hiện trong hỗ trợ khắc phục khiếm khuyết phần mềm và cho biết xu hướng thay đổi của số lượng khiếm khuyết theo thời gian sau khi chuyển giao phần mềm. (chapter 10-slide)

**Các công việc cần thực hiện:**

* Báo cáo và theo dõi nguyên nhân, khiếm khuyết
* Tiến hành khắc phục phầm từ nguyên nhân
* Làm việc với các vấn đề có liên quan đến khiếm khuyết đó.
* Cung cấp các bản vá lỗi

**Việc hỗ trợ khắc phục khiếm khuyết phần mềm yêu cầu một số vấn đề chi tiết:**

* Trình bày số lượng vấn đề và tốc độ đến của vấn đề (số lượng vấn đề xảy ra)
* Dự toán và lên kế hoạch cho các nguồn lực hỗ trợ cần thiết
* Đào tạo và xây dựng đội ngũ hỗ trợ

**Xu hướng thay đổi của số lượng khiếm khuyết theo thời gian sau khi chuyển giao phần mềm:**

* Ngay sau khi phát hành, sẽ có nhiều vấn đề được phát hiện và báo cáo bởi người dùng.
* Càng về sau số lượng vấn đề được phát hiện càng giảm tuy nhiên bản chất của vấn đề sẽ trở nên khó chẩn đoán hơn.

**Câu 40** Cho biết các công việc thực hiện trong giai đoạn trước khi kết thúc việc hỗ trợ phần mềm, hay còn gọi là giai đoạn mặt trời lặn (product sunset period). (chapter 12)

* Ngừng nâng cấp bất kì tính năng mới nào của sản phẩm
* Chỉ sửa các vấn đề có tính nghiêm trọng cao của sản phẩm cũ
* Công bố sản phẩm mới thay thế
* Khuyến khích hàng hàng hiện tại và khách hàng mới chuyển sang sử dụng sản phẩm mới
* Thông báo cho tất cả người dùng ngày chấm dứt việc hỗ trợ trên sản phẩm cũ
* Cung cấp tên của các nhà cung cấp khác, những người tiếp tục hỗ trợ sản phẩm cũ cho những khách hàng chọn sử dụng sản phẩm cũ
* Chấm dứt việc hỗ trợ khách hàng theo kế hoạch và thu hồi sản phẩm phần mềm từ thị trường.

**Câu 41** Trình bày các mục cần có trong một mẫu yêu cầu thay đổi trong giai đoạn bảo trì phần mềm. (chapter 12)

* Số hiệu của yêu cầu thay đổi
* Tên của bên yêu cầu
* Mức độ ưu tiên của bên yêu cầu: cao, thấp, trung bình
* Ngày đề nghị
* Tình trạng của yêu cầu
  + Ngày chấp nhận
  + Ngày bác bỏ
  + Ngày bắt đầu
  + Ngày hoàn thành
* Mô tả tóm tắt về yêu cầu thay đổi
* Các lĩnh vực bị tác động bởi thay đổi
* Dự đoán công sức bỏ ra
* Số hiệu phiên bản bào hành mà yêu cầu thay đổi bao hàm trong đó

**Câu 42** Cho biết các giai đoạn và các mục tiêu của quy trình quản lý dự án phần mềm. (chapter 13)

**Các giai đoạn:**

* Lập kế hoạch:
  + Bước đầu tiên của việc lập kế hoạch là hiểu rõ các yêu cầu của dự án, nó có thể được coi như là một dự án nhỏ. Sau đó là 4 bước còn lại.
  + Ước tính
    - Sự nỗ lực làm việc
    - Lịch trình và các nguồn lực cần thiết
  + Xác định rõ ràng và thiết lập các mục tiêu có thể đo lường được của dự án
  + Xác định các phân bổ nguồn lực của dự án
    - Con người
    - Quy trình
    - Công cụ
    - Cơ sở hạ tang vật lí
  + Xác định và phân tích các rủi ro có thể xảy ra
* Tổ chức: khi một kế hoạch hoàn chỉnh hoặc một phần của kế hoạch được xác định rõ ràng thì giai đoạn tổ chức có thể bắt đầu.
  + Cơ cấu thiết lập cần được thiết kế
  + Tuyển dụng nhân lực cần cho việc bắt đầu và được hoàn thành cùng với việc mua lại các nguồn lực khác
  + Phải hoàn thành bất kì hoạt động đào tạo và huấn luyện cần thiết
  + Các cơ chế theo dõi phải được thiết lập
    - Theo dõi và giảm nhẹ rủi ro
    - Giám sát mục tiêu của dự án
* Giám sát: khi dự án đươc triển khai đưa vào hoạt động nó vẫn cần được giám sát để bảo đảm rằng nó đang đi đúng hướng. 3 thành phần chính của việc giám sát là:
  + Thu thập thông tin về tình trạng của dự án
  + Phân tích và đánh giá các thông tin thu thập được
  + Cách trình bày và giao tiếp tình trạng của dự án (dựa vào các loại biểu đồ, mô hình)
* Điều chỉnh: rất khó có một dự án phần mềm nào không xảy ra lỗi trong quá trình phát triển. Vì vậy khi tình trạng dự án chỉ ra các vấn đề tiềm tàng, chúng ta không phải lo lắng vì việc thực hiện thay đổi. Có 3 lĩnh vực chính của sự điều chỉnh là:
  + Các nguồn lực
  + Lịch trình
  + Nội dung dự án

**Các mục tiêu:**

* Kết quả cuối cùng của dự án phải đáp ứng được các nhu cầu của khách hàng
* Tất cả các thuộc tính mong muốn và cần thiết của sản phẩm phải được đáp ứng (chất lượng, tính năng, chi phí, hiệu suất)
* Tất cả các mốc mục tiêu và hầu hết lịch trình phải được đáp ứng
* Các thành viên trong nhóm hoạt động có hiệu quả và có thái độ tinh thần tốt
* Các công cụ và tài nguyên được làm sẵn và sử dụng hiệu quả.

**Câu 43** Cho biết các bước của giai đoạn lập kế hoạch trong quy trình quản lý dự án phần mềm. (chapter 13)

* Lập kế hoạch:
  + Bước đầu tiên của việc lập kế hoạch là hiểu rõ các yêu cầu của dự án, nó có thể được coi như là một dự án nhỏ. Sau đó là 4 bước còn lại.
  + Ước tính
    - Sự nỗ lực làm việc
    - Lịch trình và các nguồn lực cần thiết
  + Xác định rõ ràng và thiết lập các mục tiêu có thể đo lường được của dự án
  + Xác định các phân bổ nguồn lực của dự án
    - Con người
    - Quy trình
    - Công cụ
    - Cơ sở hạ tang vật lí
  + Xác định và phân tích các rủi ro có thể xảy ra

**Câu 44** Cho biết các công việc được thực hiện trong giai đoạn tổ chức trong quy trình quản lý dự án phần mềm (chapter 13)

* Tổ chức: khi một kế hoạch hoàn chỉnh hoặc một phần của kế hoạch được xác định rõ ràng thì giai đoạn tổ chức có thể bắt đầu.
  + Cơ cấu thiết lập cần được thiết kế
  + Tuyển dụng nhân lực cần cho việc bắt đầu và được hoàn thành cùng với việc mua lại các nguồn lực khác
  + Phải hoàn thành bất kì hoạt động đào tạo và huấn luyện cần thiết
  + Các cơ chế theo dõi phải được thiết lập
    - Theo dõi và giảm nhẹ rủi ro
    - Giám sát mục tiêu của dự án

**Câu 45** Trình bày chương trình khung để xây dựng cấu trúc phân rã công việc (Work breakdown Structure) trong việc ước lượng công sức hoàn thành dự án. (chapter 13)

Là một phương pháp được sử dụng để chia các mục tiêu chính của dự án thành những nhiệm vụ nhỏ hơn nhằm đạt được mục tiêu đó.

Các chương trình khung:

* Kiểm tra và xác định các chuyển giao bên ngoài của dự án
* Xác định các bước và công việc cần thiết để tạo ra bất kì chuyển giao nào, bao gồm các công việc cần thiết để tạo ra bất kì chuyển giao nội bộ
* Trình tự công việc, chỉ ra tiềm năng cho xử lí song song
* Cung cấp một ước lượng về việc hoàn thành mỗi công việc
* Cung cấp ước tính năng suất tốt nhất của mỗi cá nhân được giao cho mỗi công việc
* Tính toán thời gian cần thiết để hoàn thành mỗi công việc
* Đối với mỗi chuyển giao bên ngoài, đặt ra các mốc thời gian của tất cả công việc cần thiết để sản xuất ra chuyển giao đó và gắn nhãn cho các nguồn lực sẽ được giao những công việc đó.