Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP1 1° semestre de 2015.

1) Explique porque o principal custo no desenvolvimento de software é o fator humano. Relacione o desenvolvimento de software com outros tipos de projetos e apresente as diferenças entre eles do ponto de vista de custo e materiais utilizados. (valor: 2,0 pontos; máximo: 10 linhas).

Projetos tradicionais, como a construção de um imóvel ou de uma ponte, consomem grandes quantidades de material físico, que são manipuladas por seres humanos para compor o produto final. Em projetos de software, o material manipulado é lógico, um conjunto de arquivos de dados e programas. Sendo assim, o principal custo dos projetos é a mão-de-obra especializada necessária para manipular estes materiais.

2) Explique porque grande parte do custo de um projeto de desenvolvimento de software é incorrido depois que as atividades de codificação e testes foram encerradas. (valor: 2,0 pontos; máximo: 10 linhas)

A manutenção é parte fundamental dos projetos de desenvolvimento de software. Software utilizado deve ser continuamente atualizado para que acompanhe o domínio e as áreas onde é aplicado, de modo a continuar útil. Desta forma, o software deve sofrer manutenções evolutivas. Também não é possível garantir, para todos os casos, que um software está 100% correto após os testes. Sendo assim, ele precisa da manutenção corretiva. Finalmente, considerando que o software será útil por muitos anos, o esforço total de manutenção tende a ser maior que o esforço de desenvolvimento, levando então ao alto custo incorrido depois dos testes.

3) Explique a relação existente entre o dicionário de dados e os fluxos de dados e repositórios em um diagrama de fluxo de dados (DFD). (valor: 2,0 pontos; máximo: 10 linhas)

Na análise estruturada ou essencial, um dicionário de dados documenta, usando uma notação estruturada, os dados que passam pelos fluxos de dados e são armazenados nos repositórios de um DFD.

4) Quais das técnicas abaixo podem ser utilizadas para apoiar o levantamento de requisitos junto aos usuários durante a análise dos requisitos de um software? Escreva os números das técnicas selecionadas na folha de resposta. (valor 1,0 ponto, cada alternativa incorreta selecionada elimina uma alternativa correta)

1. Entrevistas estruturadas
2. Ciclo de vida em cascata
3. Gerenciamento de projetos
4. *Role playing*
5. Reuniões de *brainstorming*
6. Modularização
7. *Storyboading* e prototipação

1, 4, 5, 7.

5) Quais das perguntas abaixo podem ser usadas para identificar casos de uso durante a fase de análise em um processo de desenvolvimento de software? Responda listando os números das alternativas corretas na folha de resposta. (valor: 1,0 ponto, cada resposta errada elimina uma alternativa correta).

1. Quais são as funções executadas por cada ator?
2. Que atores criam, alteram, removem ou consultam informações no sistema?
3. Quem será beneficiado pelo sistema?
4. Que funções do sistema utilizaram estas informações?
5. Alguma pessoa realiza diversos papéis no sistema?
6. Algum ator deve ser informado de alguma mudança externa?
7. Quais são as entidades do modelo de dados do sistema?
8. Que funções suportam ou mantém o sistema?

1, 2, 4, 6, 8.

6) Explique porque a modularização é importante em um projeto de desenvolvimento de software. (valor: 2,0 pontos; máximo: 10 linhas)

Um sistema de software deve ser decomponível em elementos simples, denominados módulos. A modularização auxilia na separação dos objetivos do sistema, sendo cada objetivo representado por um componente. Módulos permitem a implementação da estratégia de “dividir para conquistar”: problemas complexos são divididos em

problemas menores e mais simples, que podem ser resolvidos de forma independente (em módulos), reunindo-se suas soluções para resolver o problema complexo. Cada módulo esconde um “segredo”, que determina seu objetivo no sistema e que pode ser analisado sem que outros módulos sejam estudados em detalhes. Isto permite que os desenvolvedores conheçam parte do sistema e saibam resolver problemas relacionados a ela sem ter que conhecer o sistema como um todo.