Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AD1 2° semestre de 2011.

1) Porque é errado afirmarmos que "até que um programa esteja rodando não há como medir sua qualidade". (valor 2,5 pontos).

Porque é possível realizar revisões sobre documentos anteriores ao código-fonte, como o projeto do sistema ou seus requisitos. Com isto, a qualidade do sistema pode ser avaliada muito antes da etapa de testes.

2) Explique o que você entende por abstração e como este conceito é aplicado no desenvolvimento de software? (valor 2.5 pontos)

Abstração é a descrição dos problemas e conceitos abordados por um projeto de software em um nível de generalização que esconde os detalhes e permite o foco dos analistas nos principais mecanismos que regem o sistema. Ela permite que um sistema complexo seja projetado de forma incremental, permitindo que sua estrutura de alto nível seja compreendida antes que maiores detalhes sejam adicionados.

3) Explique a diferença entre as técnicas *top-down*, *bottom-up* e *middle-out* na Engenharia de Software. (valor 2.5 pontos)

Técnicas top-down partem de elementos mais complexos, descendo a níveis cada vez mais detalhados. Técnicas bottom-up partem de componentes menores, que são agrupados para formar os principais componentes do sistema (de mais alto nível). Técnicas middle-out partem de componentes intermediários, compondo os elementos mais complexos a partir destes e decompondo estes em outros mais simples.

4) Cite três técnicas usualmente utilizadas na identificação de requisitos em um projeto de desenvolvimento de software. (valor 1.0 ponto)

Qualquer três dentre as seguintes: entrevistas estruturadas, seminários de identificação de requisitos (*workshops*), reuniões de brainstorming, storyboading/prototipação e *role playing*.

5) Porque afirmamos que “quanto mais tarde é identificado um erro, geralmente mais cara é a sua correção”? O que se pode fazer para evitar este custo em um projeto de desenvolvimento de software. (valor 1.5 pontos)

Quando mais tarde um erro for identificado, mais cara será a sua correção porque um maior número de artefatos foi criado com base neste erro. Por exemplo, um erro nos requisitos pode se transformar em um ou mais erros de projeto, que vão se transformar em um ou mais erros de codificação, que passarão para a documentação, e assim por diante. Assim, quanto mais tarde um erro é encontrado, mais artefatos precisam ser corrigidos, encarecendo a sua resolução.