Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AD1 1° semestre de 2015.

1) É comum que programadores pensem que os únicos artefatos que interessam no desenvolvimento de um software, mesmo um software de larga escala, são os módulos de código-fonte. Porque esta linha de pensamento está errada? Dica: explique que outros artefatos (além do código-fonte) são importantes e porquê. (valor: 2,0 pontos; máximo: 10 linhas).

Os módulos de código-fonte são fundamentais para o desenvolvimento de software: em muitos casos, eles são o objetivo final de todo um processo de desenvolvimento. No entanto, em projetos de larga escala, precisamos tomar conhecimento do que efetivamente precisa ser desenvolvido e registrar este conhecimento de forma que nos aproximemos, passo a passo, do nível de abstração do código-fonte. Precisamos registrar o conhecimento sobre o problema (desenvolvendo modelos de análise), sobre a solução computacional proposta (através de modelos de projeto), até que finalmente possamos criar o código-fonte. Além disso, precisamos de planos e casos de teste para avaliar se a implementação corresponde à especificação. Os artefatos intermediários ajudam no entendimento do domínio para o qual o software será desenvolvido, ajudando os desenvolvedores a se comunicar com os usuários, clientes e entre si.

2) Explique as diferenças entre as atividades de manutenção corretiva, adaptativa, preventiva e evolutiva. (valor: 2,0 pontos; máximo: 10 linhas)

A manutenção é um processo de correção de erros que ficaram latentes no software após do processo de desenvolvimento. A manutenção pode ser evolutiva (quando as modificações no software dizem respeito à introdução ou modificação das funcionalidades do software), corretiva (quando as modificações têm como objetivo corrigir erros que passaram pela fase de testes), adaptativa (que diz respeito à alteração do software para um novo hardware ou plataforma) ou preventiva (que trata da modificação do software para facilitar futuras atividades de manutenção).

3) Qual é a característica que destaca um modelo de ciclo de vida incremental de um modelo de ciclo de vida clássico, como o modelo em cascata? (valor: 2,0 pontos; máximo: 10 linhas)

A principal diferença entre um ciclo incremental e um ciclo tradicional, em cascata, é a forma com que as atividades são conduzidas. Em um projeto que siga o modelo de

ciclo de vida em cascata, a atividade de análise é realizada em um único, grande bloco, onde todos os requisitos são levantados. Este modelo é útil quando o domínio em que o projeto está sendo desenvolvido é conhecido da equipe, o que torna possível a identificação de todos os requisitos a priori. O mesmo modelo de desenvolvimento em bloco é usado nas demais atividades do processo (projeto, codificação e testes), que são disparadas a medida que as atividades que as precedem são concluídas. Em um ciclo incremental, as atividades são realizadas em diversas etapas: realiza-se um pouco de análise de requisitos, um pouco de projeto (sobre os requisitos recém identificados), um pouco de codificação (com base na parte do projeto recém concluída) e testes. Em seguida, retorna-se para a análise, projeto, codificação e testes, sucessivamente, e em cada etapa trabalhando sobre um conjutno dos requisitos do software. Em cada rodada, um subconjunto dos requisitos é identificado, projetado e implementado.

4) Porque corrigir um erro introduzido na análise e identificado na fase de testes é mais caro do que corrigir do que um erro introduzido durante a codificação de um projeto de software? (valor: 2,0 pontos; máximo: 10 linhas).

O custo de correção de erros aumenta a medida que se avança no processo de desenvolvimento de software. O custo de resolução de um erro introduzido durante a fase de levantamento de requisitos quando identificado ainda durante a fase de análise é relativamente pequeno. O mesmo erro, quando identificado na atividade de testes, possui um custo de correção muito maior porque diversas decisões de projeto e codificação já foram tomadas com base no erro (de análise) e precisam ser corrigidas. Erros introduzidos somente durante a codificação são mais baratos de corrigir, pois um número menor de decisões foram tomadas com base neste erro. Assim, quando mais cedo um erro é corrigido ou quando mais tarde ele é introduzido, mais barata será a sua correção.

5) Quais são os três documentos desenvolvidos na aplicação da análise estruturada? Qual é o papel de cada um destes documentos? (valor: 2,0 pontos; máximo: 10 linhas)

Diagramas de fluxo de dados especificam as funções de um sistema, representando o trânsito das informações relevantes de um sistema e sua transformação pelos processos (funções do sistema). São desenvolvidos em diversos níveis, cada qual apresentando mais detalhes sobre o nível imediatamente superior.

Dicionários de dados descrevem as informações que transitam pelos fluxos de dados ou residem nos repositórios de dados representados nos diagramas de fluxos de dados. O dicionário determina a estrutura e o significado das informações, assim como os limites de valores que podem ser assumidos por elas.

A mini-especificação de processo descreve o algoritmo para o processamento que deve ser realizado nos processos componentes dos DFD. Podem ser escritas em português estruturado (mais usual), linguagens formais, fluxogramas ou qualquer outra notação capaz de representar algoritmos.