Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP1 2° semestre de 2015.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

1. Pergunta da AD1: Porque é errado afirmarmos que "até que um programa esteja rodando não há como medir sua qualidade". (valor 2,0 pontos; máximo 10 linhas).

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. É comum que programadores pensem que os únicos artefatos que interessam no desenvolvimento de um software são os módulos de código-fonte. Explique porque esta afirmativa está incorreta, ao menos em projetos de larga escala. (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 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 (modelos de análise), sobre a solução computacional proposta (modelos de projeto), até que finalmente possamos criar o código-fonte. Os artefatos intermediários ajudam no entendimento do domínio onde

o software será desenvolvido, ajudando os desenvolvedores a se comunicar com os usuários, clientes e entre si.

3. Dos cinco problemas abaixo, indique quais são observados com frequência quando um modelo de ciclo de vida em cascata é usado no desenvolvimento de um sistema. Escreva os números dos problemas selecionados na folha de resposta. (valor 2,0 pontos)

- ✓ Dificuldade de manter a sequência de tarefas proposta pelo modelo;
- ✓ Incapacidade de concluir a etapa de análise de requisitos, devido a mudanças nas demandas dos clientes, travando as demais atividades do projeto;
- × Alto custo de desenvolvimento, pois os produtos das etapas de desenvolvimento são descartados e cada nova versão do software é construída desde o início;
- ✓ A primeira versão utilizável do software só estará disponível após o término de todas as fases do projeto;
- × Componentes utilizados para acelerar as atividades de projeto e codificação podem apresentar problemas, gerando ruído nas fases seguintes do projeto.

4. Cite dois artefatos construídos durante a atividade de análise de requisitos. Explique o papel e a importância de cada artefato. (valor: 2,0 pontos; máximo: 10 linhas)

Na análise estruturada, são construídos diagramas de fluxos de dados (DFDs, que mostram o ciclo de processamento das informações que entram e saem do sistema), o dicionário de dados (que descreve os termos utilizados no domínio), as mini-especificações de processos (que descrevem, de forma estruturada, uma unidade de processamento de um DFD) e o diagrama entidade-relacionamento (que apresenta os relacionamentos entre os dados manipulados pela aplicação). Na análise orientada a objetos são construídos os diagramas de casos de uso, que representam os diálogos entre os atores que utilizam o sistema e o próprio sistema, descrevendo assim as funcionalidades que o sistema deve oferecer a seus usuários. Qualquer combinação de dois dos cinco elementos acima responde corretamente a pergunta.

5. Explique porque o projeto de software deve se basear na dualidade "interface x implementação". Dica: pense nas técnicas de ocultamento de informação. (valor 2,0 pontos; máximo: 10 linhas)

Uma interface é um conjunto de serviços oferecidos por um módulo a seus clientes e funciona como um contrato entre estas duas partes. A implementação, por outro lado,

define como a interface é codificada em um módulo e diz respeito somente ao módulo. Os clientes não devem ser dependentes da implementação e não devem entender detalhes de como ela é feita para usar as funcionalidades do módulo.

Técnicas de ocultamento de informação sugerem que o projeto de um sistema deve se basear nas interfaces oferecidas pelos módulos, desconsiderando os detalhes de como estas interfaces são implementadas. Assim, as implementações podem ser substituídas sem afetar o resto do projeto.