Fundação CECIERJ – Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 2° semestre de 2016.

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 AD2: Explique quais são as características de um bom programa. Porque é importante escrever o programa pensando em sua manutenção do que apenas nas funcionalidades entregues aos usuários? (valor: 2,0 pontos)

Acima de tudo, um bom programa deve ser correto, ou seja, deve fazer o que sua especificação estabelece e deve se comportar de forma previsível quando dados fora do padrão lhe são apresentados. Depois, um bom programa deve ser claro, no sentido de facilitar o seu entendimento por outros programadores além de quem o escreveu. Por fim, um bom programa é moderado no consumo de recursos, tais como memória, bateria e capacidade de processamento.

2. O que a cardinalidade de um relacionamento entre duas classes representa em um projeto de desenvolvimento de software? (1,0 ponto).

A cardinalidade indica o número mínimo e máximo de objetos que participam em cada lado de uma associação entre classes.

3. O que um cenário representa na modelagem dinâmica de um projeto de software usando os diagramas da UML? (1,0 ponto).

Um cenário é um caminho entre os fluxos de um caso de uso. Ele é um caminho da raiz até um nó folha da árvore composta pelos fluxos e subfluxos de um caso de uso.

4. Para as métricas a-e abaixo, indique seu nível de precisão (objetiva, subjetiva), escala (nominal, ordinal, intervalar, razão) e objeto de medição (processo, produto). Apresente 1 (um) exemplo de uso para cada uma delas no contexto dos projetos de software. (Valor: 2,0 pontos; máximo 10 linhas)

a. Acoplamento entre Objetos: Objetiva, razão, produto. Usada para medir o nível de dependência entre classes/objetos.

b. Complexidade Ciclomática: Objetiva, razão, produto. Usada para medir o numero de ciclos fechados no grafo de programa. Associada ao numero de pontos de decisão do algoritmo e a indicação de "esforço de teste" do algoritmo.

c. Nível de Maturidade em Processo de Software: subjetiva, ordinal, processo). Indica o perfil da organização/time de desenvolvimento frente aos diversos níveis de processos que podem ser usados na construção do software

d. Profundidade de Herança: objetiva, razão, produto. Usada para medir a altura da hierarquia em modelos/software orientado a objetos.

e. Confiabilidade: objetiva, razão, produto. Usada para indicar o tempo médio entre falhas de produtos de software.

5. Explique o que é teste estrutural. Qual a diferença para teste funcional? Apresente três exemplos de critérios de teste que podem ser utilizados para projetar casos de teste para este tipo de teste. (valor: 2,0 pontos)

Testes Estruturais se diferenciam dos testes funcionais por verificarem a estrutura interna do software. Usualmente são utilizados para colocar a prova os diferentes caminhos de execução do software em diferentes níveis de granularidade. O objetivo é garantir que todo caminho executável é processado sem falhas (quebras de execução). Diferentes critérios podem ser utilizados: todos-os-ramos; todos-os-nós; todos-os-caminhos, todos-os-arcos; todas-as-variáveis; dentre outros.

6. Defina os casos de teste para estes requisitos usando a técnica de particionamento por classes de equivalência (Valor 2 pontos):

Em uma indústria química, o controle da temperatura e pressão de um determinado processo depende de dois fatores: a temperatura de ebulição do produto sendo processado e de um coeficiente Z. As entradas do programa de controle são a temperatura de ebulição do produto em graus Celsius limitada ao intervalo [90, 200] e o coeficiente Z limitado ao intervalo [0.3, 2.0]. O sistema de controle deve suspender o processo se a temperatura ou a pressão atingirem valores críticos.

O cálculo do valor crítico da temperatura é obtido pela fórmula: tk = te*2.

O valor crítico da pressão pela fórmula:

pk = tk * Z, quando te for menor que 100;

pk = te * Z nos demais casos.

Onde, tk=temperatura crítica, te=temperatura de ebulição, pk=pressão crítica.

Como este software foi construído visando *testabilidade*, é possível entrar com um terceiro e quarto valores correspondentes à temperatura e a pressão em um dado instante. Nesse caso, o software efetua os cálculos e imprime uma mensagem sinalizando se interromperia ou não o processo.

Cálculo dos parâmetros de teste:
Temperatura Crítica: 400ºC (200 x 2)
Pressão Crítica:
Te <100: [54...400]
Te >100: [30...400]

Desta forma teríamos as seguintes situações para o teste:

Parâmetro	Inválida	Pk (te < 100)	Pk (te >100)	Inválida
te	<90	te * 2 * Z	te *Z	> 200

Observe que a combinação de temperatura e pressão compõe o requisito. Assim, as classes válidas e inválidas devem ser entrelaçadas já que o processo deve ser interrompido em qualquer situação que temperatura ou pressão assumam valores críticos. Como estamos trabalhando com valores críticos (muito baixos ou muito altos), Z deve variar de 0.3 a 2.0 (mínimo, máximo) para a definir a faixa aceitável de pressão crítica.
Precisamos de um caso de teste para cada classe inválida (<90, >200) e independentemente da pressão, o processo dever ser interrompido: { 85, --, interromper processo}; {201, --, interromper processo}.
Precisamos de três casos de teste para o intervalo [90,100]: {95, 100, manter processo}; {95, 52, interromper processo}; {95, 401, interromper processo}.
Precisamos de três casos de teste para o intervalo [100, 200]: { 150, 300, manter processo}; {150, 25; interromper processo}; {150, 401, interromper processo}.