Fundação CECIERJ – Vice-Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 1° semestre de 2017.

Nome –

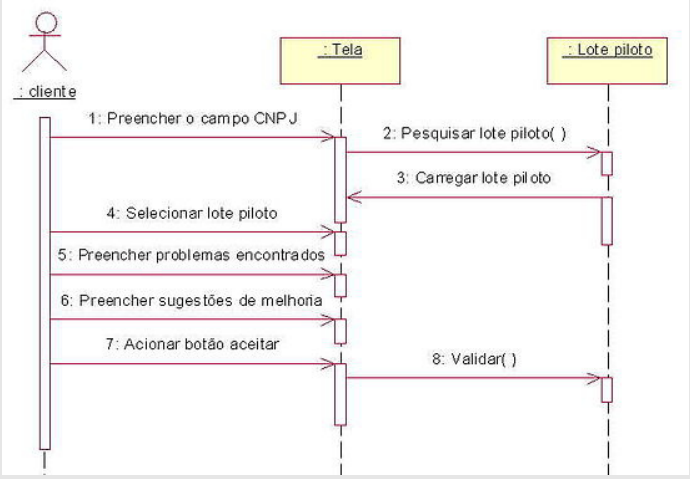
Assinatura –

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. Quais são os componentes de um diagrama de sequência da UML? Explique como estes componentes se relacionam e o quê cada um representa no diagrama. Apresente um exemplo contendo todos os componentes relacionados na sua resposta (Valor 2,0 pontos; máximo: 10 linhas)

Diagramas de sequência determinam os objetos responsáveis pela realização de um cenário de caso de uso e as mensagens que são trocadas entre eles. O diagrama apresenta a ordem com que as mensagens são trocadas no tempo. Ele é composto por objetos, que são apresentados na primeira linha do diagrama na forma de retângulos com uma "linha do tempo" que se estende até a base do diagrama, e sequências de mensagens, representadas como setas que conectam as "linhas do tempo" dos objetos. As mensagens podem ter condições de guarda e são associadas aos métodos do objeto destino da seta. A figura abaixo apresenta um diagrama de sequência:



2. Porque a duplicação de código, tipicamente realizada por *copy & paste*, quando realizada em excesso, é prejudicial ao trabalho de codificação de software. Dê um exemplo de problema que pode ocorrer (valor: 2,0 pontos; máximo: 10 linhas)

Criar código duplicado indica um erro de decomposição do programa. Se o mesmo código existe em duas rotinas diferentes e possui um tamanho considerável (mais do que umas poucas linhas de código), ele deve ser retirado das duas rotinas originais e movido para uma rotina em separado, que deve ser chamada pelas rotinas originais. Esta separação reduz a memória ocupada pelo programa, facilita a correção de erros, a otimização do código e a manutenção do sistema. Um exemplo de problema que ocorre com copy&paste é a identificação de um erro em um código copiado, devendo este erro ser resolvido em todas as cópias do código; se o código tivesse sido isolado em uma rotina e chamado onde necessário, apenas o código desta rotina precisaria ser alterado para resolver o erro.

3. Para as métricas a-e abaixo, indique seu nível de precisão (objetiva, subjetiva), escala (nominal, ordinal, intervalar, razão) e objeto de medição (processo, produto). Informe uma situação de projeto de software que a variação dos seus valores pode ajudar a observar. (Valor: 2,0 pontos; máximo 10 linhas)

Perda de Coesão em Métodos (Objetiva, razão, produto). Quanto maior a perda de coesão, maior o risco de problemas (falhas, efeitos colaterais, etc.)

Resposta para uma Classe (objetiva, razão, produto). O aumento no numero de respostas indica que o nível de depencia com a classe também aumenta, e uma falha nesta classe pode implicar em problemas em diferentes partes do software.

Métodos Ponderados por Classe (objetiva, razão, produto). Quanto maior o valor da medida, maior a complexida da classe. O aumento da complexidade implica em aumento de esforço para sua construção e teste, com eventual aumento de risco de falha do software.

Nível de Conhecimento do Domínio (subjetiva, pode assumir nominal, ordinal ou intervalar, processo ou produto). Indica a percepção de conhecimento sobre o dominio do problema apresentado pelo desenvolvedor. Quanto maior o nível de conhecimento do domínio, mais adequada a participação do desenvolvedor nas atividades, principalmente, de requisitos, projeto e testes.

Satisfação do Usuário (subjetiva, pode assumir nominal, ordinal ou intervalar). Indica a percepção de conforto e satisfação do usuário com o software. Quanto maior a satisfação, maior a chance de sucesso do software.

4. Explique o que é teste funcional. Qual a diferença para teste baseado em erros? Identifique três critérios de teste que podem ser utilizados para projetar casos de teste funcional. Escolha um deles e explique como usá-lo para projetar os casos de teste (valor: 2,0 pontos)

Teste funcional, ou teste de caixa-preta ou caixa fechada, visa avaliar o comportamento do software com foco em suas funcionalidades, sem se preocupar em sua arquitetura ou organização interna. Neste cenário de trabalho, são preparados casos de teste que intencionam identificar dados de entrada e condição de execução que garantam observar um comportamento especificado e esperado na saída. Diferentemente, testes

baseados em erro intencional introduzir defeitos no software e preparar casos de testes para identificar estes defeitos. Três critérios para testes funcional são particionamento por equivalência, analise valor limite e grafo de causa efeito. Particionamento por equivalência trata o domínio de entrada (ou saída) em classes de equivalência (validas e invalidas) e com base nestas classes escolhe-se (aleatoriamente) um valor de cada classe para ser submetido ao software. Portanto, o numero de casos de testes esta associados ao numero de classes equivalentes identificadas. Análise por valor limite considera os limites dos intervalos (extremos) para identificação dos casos de teste. Desta forma, ao se projetar os casos de teste observa-se os limites mínimo e máximo e se preparam casos de teste abaixo do limite mínimo,no limite minino, no limite máximo e acima do limite máximo. Grafos de causa efeito intencionam capturar a logica de decisão utilizada para construir o requisito e combinar as diferentes condições de funcionando para identificar os casos de teste que permitem executar todas as tomadas de decisão possíveis previstas pela funcionalidade retratada pelo código.

5. No método de inspeção de software, a forma mais simples de realizar a identificação de defeitos é utilizar uma técnica *ad-hoc*. Entretanto, para aumentar a eficiência, inspetores podem usar *checklists* ou *técnicas de leitura*. O que é e como se apresenta um *checklist*? O que é e como se apresenta uma *técnica de leitura*? Cite duas ações que devem ser realizadas para introduzir *checklists* ou *técnicas de leitura* no método de inspeção em um projeto de software (Valor: 2,0 pontos; máximo 10 linhas)

   Um Checklist é composto por conjunto de itens de qualidade que devem ser verificados no artefato.A medida que os itens são verificados, uma marca (check) vai sendo colocada ao lado do item indicando se foi possível ou não observar tal característica. A não observância indica a presença de um defeito e a experiência do inspetor afeta sua aplicação . Uma técnica de leitura apresenta um conjunto de instruções concretas que visam guiar (apoiar) a leitura (inspeção) do artefato. Por apresentar instruções claras sobre como proceder para realizar a avaliação, técnicas de leitura são menos dependentes da experiência do inspetor. O uso destas técnicas demanda, dentre outras questões, sua configuração e ajuste para o contexto organizacional e no treinamento dos inspetores.

Boa Prova!