Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 1° semestre de 2012.

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) Considere o código abaixo, escrito em uma linguagem de programação genérica. Assumindo que o código esteja sintaticamente e semanticamente correto, que vetores com pelo menos um ponto são passados para ele, que princípio da codificação está sendo ferido por ele. Justifique sua resposta. (valor: 2,0 pontos; máximo: 10 linhas)

```
public void calculaLimite (double[] dados, boolean calculaMinimo)
{
    double min = dados[0];
    double max = dados[0];

    for (int i = 1; i < dados.size(); i++)
    {
            if (dados[i] > max)
                max = dados[i];
            if (dados[i] < min)
                min = dados[i];
    }

    if (calculaMinimo)
          return mín;

    return max;
}
```

É o princípio da “Inversão de Comando”, que determina que os parâmetros de uma rotina devem conter apenas operandos, não opções de execução da rotina. Um operando representa um objeto ou valor que a rotina utiliza para calcular seus resultados ou realizar seu processamento. Uma opção indica um modo de operação da rotina, ou seja, uma informação que determina a função cumprida pela rotina. No programa acima, o parâmetro *calculaMinimo* é uma opção.

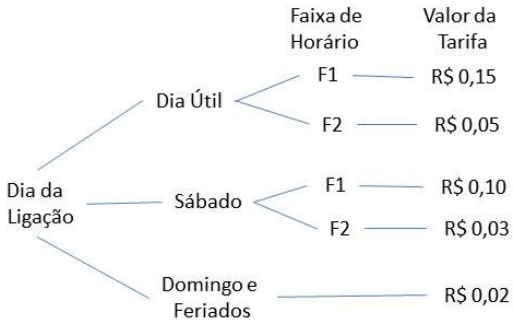
2) Explique o que é o escopo de um projeto de desenvolvimento de software, do ponto de vista da gerência destes projetos. (valor: 2,0 pontos; máximo: 10 linhas)

O escopo determina o trabalho que deve ser realizado no contexto de um projeto. Em projetos de software, estes os processos de gerenciamento de escopo determinam as funcionalidades que serão consideradas na construção do produto, ou seja, define as tarefas necessárias para que o projeto seja realizado com sucesso.

3) Utilize a estratégia de grafo de causa e efeito (apresente árvore e tabela de decisão) para representar os casos de teste para o seguinte requisito de software (2,0 pontos):

“ Uma companhia telefônica realiza a cobrança das ligações comerciais considerando as categorias de tarifas nos horários F1 (entre 8:00 e 18:00 hrs) e F2 (18:01 as 07:59 hrs). Nos dias úteis as ligações para as categorias F1 e F2 são tarifadas em R$ 0,15 e R$ 0,05, respectivamente. Para os sábados as ligações para as categorias F1 e F2 são tarifadas em R$ 0,10 e R$ 0,03 respectivamente. Aos domingos e feriados, as tarifas das ligações para F1 e F2 são tarifadas no mesmo valor de R$ 0,02.”

As datas devem corresponder a dia útil, feriado, sábado e domingo, podendo ser quaisquer dias. Lembre-se que são 2 casos de teste para Domingos e Feriados, já que é preciso garantir que o requisito “funcione” para ambas as situações. Neste caso, o horário pode ser qualquer, por isso está marcado com ---.



Dia da Ligação	Horário da Ligação	RESULTADO: Valor da Tarifa
6/6/2012	9:50	0,15
6/6/2012	22:10	0,05
9/6/2012	8:30	0,10
9/6/2012	21:00	0,03
10/6/2012	---	0,02
7/6/2012	---	0,02

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 exemplo de uso para cada uma delas no contexto do software. (Valor: 2,0 pontos; máximo 10 linhas)

Esforço (Homem/Hora)

Objetiva, razão, processo. Utilizada para medir a "quantidade de trabalho" realizada por hora e com isso apoiar a estimativa de esforço do projeto.

Nível de Experiência do Desenvolvedor
Subjetiva, ordinal ou intervalar (Likert), processo. Utilizada para classificar a experiência percebida de um desenvolvedor. Pode ser aplicada para apoiar a tomada de decisão frente a organização de equipe de projeto.

Número de Defeitos
Objetiva, razão, processo ou produto. Indica o número de defeitos presentes e/ou identificados nos artefatos do software. Pode ser utilizada para calcular quantidade de retrabalho nos projetos ou como indicador de qualidade.

Acoplamento entre Objetos (CBO)
Objetiva, razão, produto. Mede o numero de relacionamentos (dependências) existente entre classes em um projeto Orientado a Objetos. Utilizada para avaliar a complexidade estrutural do projeto.

Nível de Maturidade de Processo
Subjetiva, ordinal ou intervalar (Likert). Sugere o nível de maturidade (experiência no desenvolvimento) de uma organização de software. Utilizada para avaliar os processos usualmente empregados pela organização no desenvolvimento de seus projetos.

5) Na fase de detecção de defeitos do processo de inspeção de software, a forma mais simples de realizar a detecção é a partir da utilização de técnica de inspeção *ad-hoc*. Entretanto, para aumentar a eficiência, inspetores podem usar técnicas mais robustas, tais como *checklists ou técnicas de leitura*. Explique o que é um *checklist* e uma *técnica de leitura* e cite pelo menos 1 diferença entre elas. (Valor: 2,0 pontos; máximo 10 linhas)

Um checklist representa uma lista de questões relacionadas aos atributos de qualidade que ser identificar em um determinado artefato de software. Estas questões descrevem as características esperadas para o artefato em questão. Normalmente, a resposta a cada pergunta é dada em uma das alternativas *sim/não* e usualmente espera-se que ao final da aplicação do *checklist* todas as respostas sejam *sim*, mostrando que, de acordo com o *checklist* nenhum problema foi encontrado. Caso contrário, uma resposta *não* deve levar a descrição de uma discrepância que pode se identificar como um defeito na reunião de inspeção.

Exemplo de possíveis questões para uma *checklist* aplicável a inspeção da especificação de requisitos:

1. Os requisitos exibem a distinção clara entre funções e dados?
2. Os requisitos definem todas as informações a ser apresentada aos usuários?
3. Os requisitos descrevem as respostas do sistema ao usuário devido à condições de erro?

Uma técnica de leitura consiste num conjunto de instruções explicitas que orientam o inspetor a como ler um determinado artefato de software. Técnicas de leitura podem ser baseadas em perspectiva. Neste caso, a leitura do artefato pode ser realizada sob o ponto de vista de um determinado usuário do documento (testador, projetista ou cliente). Estas técnicas transformam a atividade de inspeção proativa, fazendo com que além da lista de defeitos sejam também construídos artefatos de apoio (casos de teste na perspectiva do testador, diagramas de classes na perspectiva do projetista ou casos de uso na perspectiva do cliente). Técnicas de leitura podem também ser orientadas a defeitos, como aquelas aplicadas modelos UML.

Uma diferença marcante entre técnicas de leitura e *checklists* está no conjunto de instruções. Enquanto os *checklists* apresentam questionamentos com base nos critérios de qualidade a serem inspecionados, as técnicas de leitura instruem a como ler e encontrar os defeitos. Outra diferença está na produção de artefatos auxiliares pelas técnicas de leitura, que podem ajudar no processo de desenvolvimento, além da lista de defeitos.