Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 1° semestre de 2014.

Nome –

Assinatura –

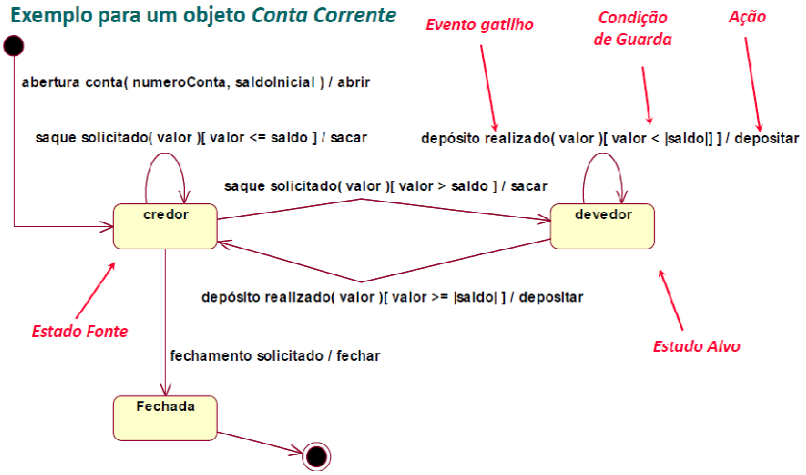
Observações:

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) O que representa uma Condição de Guarda em um diagrama de estados? Ela deve ser opcional ou obrigatória? Dê um exemplo. (Valor: 1,0 ponto; máximo: 10 linhas)

Uma condição de guarda indica a expressão que deve ser avaliada quando o objeto receber um evento gatilho ("trigger"). Se for verdadeira, a transição deve ocorrer, portanto, sua existência implica que ela é obrigatória.



2) Dizemos que um projeto de software deve ser gerenciado em cinco grandes etapas: inicialização, planejamento, execução, controle e fechamento. Uma organização de software decidiu inverter a ordem das etapas, fazendo com que o controle ocorresse antes da execução. Você, como engenheiro de software, faria isso? Se sim, em que situações? Se não, por quê? (Valor: 2,0 pontos; máximo: 10 linhas)

Não é usual inverter esta ordem. O **controle** envolve a monitoração e medição do progresso para garantir que os objetivos do projeto serão atingidos. Portanto, o controle pode e deve ser feito ao longo do desenvolvimento.

3) Marque Verdadeiro (V) ou Falso (F) para as afirmações abaixo (valor até 2,0 pontos – 0,2 por marcação correta. Atenção: cada 2 erros/2 brancos/ 1 erro, 1 branco eliminam 1 acerto!):

A. As técnicas de inspeção *ad-hoc*, se realizadas por inspetores experientes, sempre garantem a cobertura do documento como um todo.

B. Inspeções de Software encontram defeitos e, por isso, provocam um impacto negativo na produtividade, qualidade e estabilidade dos projetos de software.

C. Inspeções de Software permitem capturar mais de 60% dos defeitos.

D. O perfil de experiência do inspetor não afeta a eficiência da inspeção *ad-hoc*.

E. O custo para correção de um defeito aumenta ao longo do ciclo de vida do software.

F. Os melhores testadores para o software são os próprios desenvolvedores.

G. Defeitos são usualmente gerados na comunicação e na transformação das informações ao longo do projeto do software.

H. Prevenir Defeitos é melhor que Remover Defeitos.

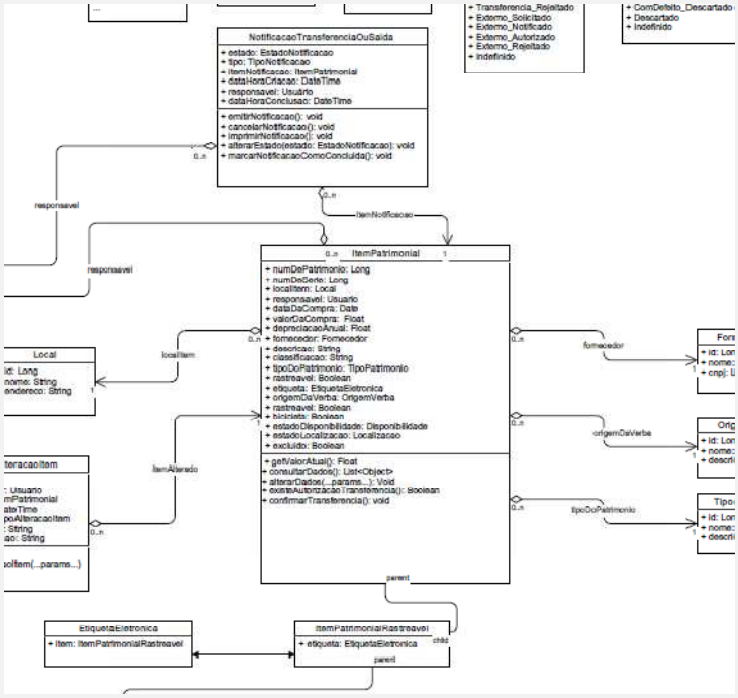
I. Os testes provam que o software está livre de defeitos.

J. Os custos e resultados de um *Walkthrough* são iguais aos de uma inspeção de software.

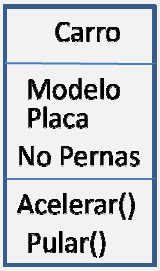
R: **F, F, V, F, V, F, V, V, F, F**

4) Defina o que representam os conceitos de acoplamento e coesão em software. Quais os riscos que devem ser considerados quando cada uma destas características é observada no software. Por exemplo, considere o que pode acontecer quando se apresentam em excesso ou deficiência. Indique uma métrica que pode ser usada para observar cada conceito e exemplifique, através de um diagrama de classes, classes que apresentam problemas de coesão conceitual e acoplamento. (Valor: 2,0 pontos; máximo 10 linhas)

Acoplamento diz respeito as dependências existentes entre unidades do software (módulos, classes, objetos, etc.). Quanto maior o acoplamento, maior a dependência. Coesão diz respeito a aderência da unidade com suas próprias propriedades, ou seja, a percepção de responsabilidade individual indicando que uma unidade de software (módulo, classe, objeto) não deveria assumir responsabilidades que não são suas. Quando estes conceitos são ignorados ou aparecem em excesso o projeto passa a enfrentar problemas de aumenta da chance de falhas, manutenção e reutilização. è preciso ter um equilíbrio entre coesão e acoplamento. Métricas que podem ser usadas, respectivamente, para observar acoplamento e coesão são CBO (acoplamento entre objetos) e LCOM (perda de coesão em métodos).



Classe com alto acoplamento



Classe com problema de coesão conceitual (note que o atributo No Pernas e o serviço Pular() conceitualmente não fazem parte do domínio Carro)

5) Utilize a abordagem de *Particionamento por Equivalência* (apresente as classes válidas e inválidas) para representar os casos de teste para o seguinte requisito (Valor: 2,0 pontos):

> "Uma companhia telefônica realiza a tarifação das ligações comerciais considerando as categorias de horário F1, F2 e F3. Nos dias úteis as ligações para as categorias F1, F2 e F3 são tarifadas em R$ 0,04, R$0,03 e R$ 0,05, respectivamente. Para os finais de semana e feriados, as tarifas das ligações para F1, F2 e F3 são R$ 0,06, R$0,05 e R$ 0,04 respectivamente. Caso a ligação seja realizada para um numero telefônico da mesma companhia, as tarifas tem redução de 30% nos dias úteis, de 50% nos finais de semana e de 55% nos feriados."

Condições de Entrada:
Dia útil: Classe Válida (Sim); Classe Inválida (Não)
Feriado: Classe Válida (Sim); Classe Inválida (Não)
Finais de Semana: Classe Válida (Sim); Classe Inválida (Não)

Número Mesma Companhia: Classe Válida (Sim); Classe Inválida (Não)

Note que Dia útil e Finais de Semana são excludentes, ou seja ou é dia útil ou final de semana. Por outro, um Feriado pode ser num dia útil ou num final de semana.

Assumindo que a tarifação é feita por ligação e não por tempo de ligação; teríamos os seguintes casos de teste:

T_{F1}= {(Dia útil, Outra Companhia, 0,04); (Dia útil, Mesma companhia, 0,7*0,04); (Final de Semana, Outra Companhia, 0,06); (Final de Semana, Mesma companhia, 0,5*0,06); (Feriado, Outra Companhia, 0,06); (Feriado, Mesma companhia, 0,45*0,06)}

T_{F2} = {(Dia útil, Outra Companhia, 0,03); (Dia útil, Mesma companhia, 0,7*0,03); (Final de Semana, Outra Companhia, 0,05); (Final de Semana, Mesma companhia, 0,5*0,05); (Feriado, Outra Companhia, 0,05); (Feriado, Mesma companhia, 0,45*0,05)}

T_{F3}= {(Dia útil, Outra Companhia, 0,05); (Dia útil, Mesma companhia, 0,7*0,05); (Final de Semana, Outra Companhia, 0,04); (Final de Semana, Mesma companhia, 0,5*0,04); (Feriado, Outra Companhia, 0,04); (Feriado, Mesma companhia, 0,45*0,04)}

6) Defina ***falta***, ***erro***, ***falha*** e ***defeito***. Quais técnicas podem ser usadas para identifica-los? (Valor: 1,0 ponto, 10 linhas).

Falta: Deficiência mecânica ou algorítmica que se ativada pode levar a uma falha. É introduzido no software quando um desenvolvedor comete um engano. Um engano num software profissional pode causar várias faltas. De forma inversa, uma falta pode ser causada em conseqüência de vários enganos.

Erro: item de informação ou estado de execução inconsistente. Indica o quanto um resultado está incorreto.

Falha: Evento notável onde o sistema viola suas especificações. Comportamento incorreto do software como conseqüência de uma falta.

Defeito: Termo genérico para falta, falha ou erro.

Inspeções de software identificam faltas, Testes de software revelam falhas.