Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AD2 2° semestre de 2011.

Atenção: Como a AD é individual, caso seja constatado que provas de alunos distintos são cópias uma das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem sim, ser buscadas por grupos de alunos, mas a redação final das respostas para as questões da prova tem que ser individual!

1. Utilizando a notação UML e considerando o documento de requisitos para o Sistema de Inventário Patrimonial (em anexo), apresente **(8,0):**

    a. o diagrama de classes para o sistema (2,0 pontos)

O diagrama de classes depende da perspectiva (ponto de vista) utilizado pelo engenheiro de software. Entretanto, os seguintes conceitos principais deveriam estar representados de acordo com alguma abstração utilizada:

- Usuário
- Colaborador
- Administrador
- Responsável
- Item patrimonial
- Setor patrimonial
- Notificação de saída
- Notificação de transferência
- Origem de Verba
- Local da organização

Uma boa maneira de identificar classes em um documento é ler atentamente e identificar substantivos no documento. Em geral, estes quando relacionados ao domínio do problema podem ser classes do seu sistema. **Atenção:** não são todos os substantivos que se tornam classes conceituais do sistema. Outra dica é tentar representar os atores como classes, porém é preciso ter atenção pois, assim como acontece com os substantivos, pode ocorrer a existência de ator que não vai ser representado no diagrama de classes. Em geral, tente verificar se a classe criada, correspondente a um ator, tem algum relacionamento com outra classe do sistema,

senão pode haver um equívoco. Pois este ator pode ser um elemento externo ao sistema e não deveria estar representado no diagrama de classes conceitual. Prestem atenção para as classes de associação que podem existir. Em geral, elas surgem em relacionamentos entre dois tipos de classes que requerem não só informações sobre as classes participantes, mas também sobre a própria relação. Um exemplo é a relação entre empresa e funcionário, o seu relacionamento contrato tem informações próprias como data de início, data fim, etc. que diz respeito a relação entre o funcionário e a empresa.

b. os diagramas de seqüência para os casos de uso UC03, UC10 e UC13 (1,5 ponto)

Novamente, a estrutura final do diagrama de seqüência depende de como o engenheiro definiu os conceitos (classes). Dicas: as classes existentes nos diagramas de seqüência TÊM que corresponder às classes do diagrama de classes. Se um objeto envia mensagem para outro objeto, provavelmente vai existir uma relação entre eles nos diagramas de classe, a não ser que este objeto passe uma mensagem para um objeto recebido como parâmetro em um método. Neste caso, pode-se não ter uma associação, mas uma relação de dependência, que não é usualmente representada nos diagramas de classe. Ainda, a mensagem enviada tem que compor a relação de serviços do objeto (classe) que receber a mensagem.

c. Que técnicas de inspeção podem ser usadas para revisar estes diagramas? Que critérios você usou para indicar estas técnicas e que tipos de defeito elas podem ajudar a encontrar? Dê um exemplo aplicando uma das técnicas indicadas nos diagramas que você criou (1,5 ponto)

As técnicas de inspeção neste caso poderiam ser ad-hoc (leitura de acordo com a experiência do desenvolvedor), checklist ou técnica de leitura (neste caso, seria uma técnica para inspeção de modelos de projeto. Ex.: OORTs). O exemplo de aplicação de uma técnica de inspeção deveria contemplar situações de inconsistência entre diagramas, por exemplo, as dicas fornecidas para a questão 1.b. Diferentes tipos de defeito podem ser encontrados (ex.: omissão, fato incorreto, inconsistência, ambigüidade e informação estranha)

d. Apresente o conjunto de casos de testes para realizar o teste do UC02 – Cadastrar Item Patrimonial, indicando qual abordagem foi utilizada para gerar esta informação (1,5 pontos)

As Regras RN1, RN2, RN3 e RN4 direcionam o planejamento dos testes. Portanto, valores válidos e inválidos para o cadastramento de um novo item devem ser fornecidos. Além disso, o item a ser utilizado para teste deve ter seus valores variando conforme as regras e fluxos alternativos.

e. Utilize as métricas NOC, DIT e CBO para identificar as classes que poderiam ser mais propensas à falha no seu projeto. Indique as classes e o motivo desta indicação (1,5 ponto)

Esta questão depende do diagrama de classes capturado. Dependendo da perspectiva de projeto utilizada, a complexidade estrutural da solução pode se alterara, fazendo com que uma ou outra classe possa apresentar características de propensão a falhas. Normalmente, classes com alto valor para CBO têm mais chance de falhar, pois possuem muitas relações de dependência com as outras classes/objetos do projeto.

2. Explique o que é uma inversão de comando na codificação de um projeto de software (valor: 1,0 ponto; máximo: 5 linhas).

A inversão de comando ocorre quando uma rotina recebe um parâmetro de controle, como um flag, por exemplo, que determina como a rotina deve agir. Dizemos que é uma inversão de comando porque um agente externo, que chama a rotina, determina como a rotina deve atuar, quebrando sua independência. A inversão de comando geralmente dificulta o entendimento da rotina e do sistema.

3. Ao discutirmos os princípios de projeto de software, vimos um ciclo onde requisitos geram a demanda por mais software, que entra em operação, provoca mudanças no ambiente de trabalho, mudanças estas que acabam gerando novos requisitos. Explique porque este ciclo ocorre e como normalmente é encerrado. (Valor: 1,0 ponto; máximo: 10 linhas)

Um software normalmente é construído de acordo com os requisitos de seus futuros usuários. No entanto, quando o software entra em operação, ele altera o ambiente de trabalho, mudando a forma com que seus usuários trabalham. Desta forma, o software geralmente precisa ser adaptado para a nova realidade, da qual ele participou na formação. Esta é uma das razões pela qual um software usado está em constante alteração, sendo ajustado a medida que os processos que ele suporte evoluem. O ciclo é encerrado quando o sistema não atende mais as necessidades dos seus usuários e a manutenção e/ou evolução demanda muito esforço e custo, tornando-se, neste caso, mas viável a construção de um novo produto.