Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AP2 2° semestre de 2011.

1) Quais das características abaixo são utilizadas para descrever uma associação em um diagrama de classes? Escreva os números das características selecionadas na folha de resposta. (valor 1.0 ponto)

1. Nome
2. Herança
3. Papel
4. Cardinalidade
5. Método
6. Navegação
7. Caso de uso

RESPOSTA: 1, 3, 4 e 6.

2) Explique o que é uma interface e como ela é representada em um diagrama de classes. (valor 1.0 ponto)

RESPOSTA: uma interface é a declaração de um conjunto de métodos que devem ser implementados pelas classes que desejam atender à interface. Métodos e classes podem fazer referência à interface como um tipo, ainda que não seja possível criar instâncias deste tipo. A realização de uma interface por uma classe é representada por um símbolo especial, uma bola saindo da interface e ligada a ela por uma linha. A classe que implementa a interface é ligada por uma linha ou seta à bolinha.

3) Explique o que é um projeto de software consistente, citando pelo menos duas características de um projeto com consistência. Porque a consistência é importante? (valor 2.0 pontos)

RESPOSTA: a consistência deve ser a primeira e principal força direcionando um projeto de software. Um projeto de software consistente é documentado sempre da mesma maneira, identado da mesma maneira, possui padrões para a resolução dos mesmos problemas, possui padrões para nomear variáveis e funções, entre outras características.

4) Coesão e Acoplamento representam importantes atributos que podem ser observados em um software. Desta forma, o que significam Coesão e Acoplamento e quais os problemas que podem causar quando se apresentam em excesso ou deficiência nos projetos de software. Indique uma métrica que pode ser usada para observar cada um deles. (Valor: 2,0 pontos)

Coesão: Intuitivamente, refere-se à consistência interna dos pedaços de um projeto. O grau de similaridade de métodos pode ser visto como o maior aspecto de coesividade das classes/objetos. Se uma classe/objeto tem diferentes métodos executando diferentes operações para um mesmo conjunto de variáveis de instância, a classe é coesa. A deficiência de coesão (baixa coesão) leva ao aumento das dependências e possibilidade de efeitos colaterais, tendo em vista que um determinado módulo acaba fazendo muito mais que deveria dentro do contexto do software. Seu "excesso", ou seja. alta coesão, é sempre desejável e indica que o produto foi construído com qualidade. Métrica: Perda de Coesão em Métodos (LCOM).

Acoplamento: Intuitivamente, refere-se ao grau de interdependência entre diferentes pedaços de um projeto. Desde que objetos de uma mesma classe possuem as mesmas propriedades, duas classes estão acopladas quando métodos declarados numa classe utilizam métodos ou variáveis de instância de outra classe. Métricas: Acoplamento entre Objetos (CBO), Resposta de uma classe (RFC)

A deficiência (baixo acoplamento) é desejável. Porém, é muito difícil construir um sistema que não tenha algum nível de acoplamento, pois a modularidade implica na existência de alguma dependência entre os módulos para se chegar a solução final. Portanto, algum acoplamento é sempre esperado e espera-se que seja o menor possível. O excesso (alto acoplamento) não é desejável, pois isso implica em muita dependência entre os módulos e qualquer mudança em um modulo especifico pode levar a ocorrer falhas em diferentes partes do software.

5) Na fase de detecção de defeitos do método de inspeção de software é possível utilizar checklists. Explique o que é um checklist, dê um exemplo e cite pelo menos 2 diferenças quando comparado com uma técnica ad-hoc. (valor: 2,0 pontos)

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?

A diferença para revisões ad-hoc se relaciona a cobertura da inspeção, pois é possível direcionar a revisão a partir dos itens do checklist (características de qualidade definidas a priori e cobertura do documento) e o checklist sempre pode ser adaptado ao projeto em que esta sendo utilizado.

6) Alguns critérios podem ser utilizados para projetar casos de teste. Descreva um destes critérios, indicando se ele deve ser usado para teste funcional ou teste estrutural e dê um exemplo de sua utilização (valor: 2,0 pontos)

Os critérios podem ser identificados de acordo com a estratégia que se quer explorar: funcional ou estrutural. Para testes funcionais podemos usar particionamento por classes de equivalência, analise valor limite e grafo de causa efeito. Para testes estruturais podemos explorar alguma característica estrutural da aplicação, como por exemplo, todos os caminhos, todos os nós, todos os usos, potenciais usos, etc.

Particionamento por equivalência: o conjunto de dados de entrada deve ser dividido em subconjuntos de classes válidas e inválidas (classes equivalentes) e para cada um destes subconjuntos um caso de teste deve ser usado. Por exemplo, para o conjunto de dados de entrada representando os CPFs dos clientes, 2 classes de equivalência (uma valida e uma invalida) existem e neste caso serão necessários 2 casos de teste para avaliar a entrada deste dado.

Análise valor limite: os dados de entrada são avaliados de acordo com seus limites extremos. Casos de teste devem ser elaborados para valores no limite e imediatamente abaixo e acima dos limites. Normalmente é utilizado em conjunto com particionamento por classes de equivalência. Por exemplo, para se testar uma funcionalidade que trate a idade do cliente no intervalo [18..60], os casos de teste {17,18,60,61} devem ser utilizados.

Grafo de Causa e efeito: os dados e restrições de entrada são representados a partir de arvores/tabelas de decisão onde a entrada do dado é representada pela raiz da

arvore (grafo – causa) e o resultado pelas folhas (efeito), de acordo com as condições existentes e referentes a combinação dos dados. Ver exemplo no material de aula.

Para os critérios aplicáveis a teste estrutural, a idéia básica está em escolher o critério e partir dele gerar as possíveis combinações de casos de teste que satisfaçam o critério escolhido. Outra opção é utilizar métricas do tipo Complexidade Ciclomática para identificar o numero de casos de testes necessários e a partir daí identificar os caminhos que devem ser satisfeitos no teste. Ver material de aula.