Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina de Arquitetura e Projeto de Sistemas II
Gabarito da AP3 – 1° semestre de 2010

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.

Questão 1 (2 pontos)

Em relação a padrões GRASP,

(a) Explique com suas palavras a diferença entre os padrões Creator e Low Coupling.

Resposta: O padrão Creator apoia na decisão de qual classe deve ser responsável por instanciar um objeto. De modo geral, a heurística indica que a classe responsável pela instanciação do objeto já deve ter acoplamento com a classe que será utilizada para a instanciação. Já o padrão Low Coupling visa manter baixo o grau de acoplamento do sistema como um todo. Esse padrão pode ser visto como um padrão de avaliação de outros padrões. Quanto um padrão, como, por exemplo, Creator, chega a duas ou mais soluções possíveis, o padrão Low Coupling pode ser utilizado para escolher a melhor das soluções.

(b) Explique com suas palavras as deficiências do padrão Pure Fabrication.

Resposta: O padrão Pure Fabrication visa criar uma classe fictícia para evitar o aumento de acoplamento entre classes já existentes. Contudo, essa classe fictícia normalmente não faz parte da realidade, e se utilizada em excesso pode fazer com que o sistema se torne progressivamente mais difícil de entender. Além disso, por ser uma classe que não contém atributos, pode enfraquecer as características de orientação a objetos do sistema e o tornar um sistema orientado a eventos.

Questão 2 (3 pontos)

Em relação às heurísticas de POO,

(a) Cite com suas palavras quatro heurísticas de POO (uma frase por heurística).

Resposta: "modele sempre o mundo real", "não confunda classe com papel", "faça perguntas ao diagrama de classe para verificar a corretude da estrutura" e "a classe todo pode conhecer as classes partes, mas a classe parte não pode conhecer a classe todo".

(b) Escolha duas das heurísticas citadas no item 2.a e explique detalhadamente, com suas palavras, a razão da heurística.

Resposta: "não confunda classe com papel" – em várias situações, múltiplas classes são criadas onde as listas de atributos e métodos dessas classes são idênticas. Nesses casos, possivelmente está havendo uma confusão entre o que é uma classe e o que é somente um papel. Um papel se resume em um relacionamento entre classes, onde uma mesma classe é utilizada sob diferentes perspectivas. Assim, no lugar de serem criadas classes adicionais, são criados novos relacionamentos entre classes existentes.

"a classe todo conhecer as classes partes, mas a classe parte não pode conhecer a classe todo" – nesse caso, a heurística diz que ao criar uma estrutura todo-parte, do ponto de vista de acoplamento e coesão não há problema em ter a classe todo com uma associação direcionada (navegabilidade) para a classe parte. Contudo, para permitir reutilização, é muito importante que a classe parte não tenha nenhuma dependência para a classe todo. Caso contrário, ao tentar reutilizar a classe parte em outro contexto, a classe todo teria que ser reutilizada em conjunto.

(c) Escolha uma das heurísticas detalhadas no item 2.b e exemplifique, com suas palavras e com um exemplo diferente do visto em aula, uma situação onde a heurística poderia ser adotada.

Resposta: "não confunda classe com papel" – imaginando um sistema onde um cliente tem endereço residencial e endereço profissional, uma possível modelagem ignorando essa heurística poderia prever a existência de três classes: Cliente, EnderecoResidencial e EnderecoProfissional, onde Cliente estaria associado com as demais classes. Contudo, ao considerar essa heurística, é possível perceber que o fato de ser residencial ou profissional é somente um papel do endereço. Assim, somente duas classes seriam criadas: Cliente e Endereco. Além disso, seriam criadas duas associações entre Cliente e Endereço: enderecoResitencial e enderecoProfissional, representando os dois papéis de endereço no sistema.

Questão 3 (2 pontos)

O que torna arquitetura de software um tópico relevante hoje em dia? Quais são os benefícios de se considerar a arquitetura de software durante o desenvolvimento de um sistema?

Resposta: Podemos citar uma série de justificativas sobre a relevância da arquitetura de software no cenário atual da Engenharia de Software. Uma das principais está no fato de que produtos de software são entidades que se encontram em constante estado de mudança, sobretudo no contexto de desenvolvimento atual, que contempla novas formas de interação como desenvolvimento distribuído de software. Dessa forma, com a necessidade de evolução, os produtos de software se tornam predispostos a defeitos, atrasos na entrega e custos acima do esperado. Ou seja, seu escopo, complexidade e manutenção são cada vez mais significativos e exigem que profissionais da área se comuniquem por meio de componentes de software, uma vez que percebemos que o sucesso de um sistema computacional depende muito mais das características do software produzido do que do hardware sobre o qual irá rodar. Somando-se a isso, técnicas de abstração como decomposição modular, linguagens de programação de alto nível e tipos de dados abstratos já não são mais suficientes para lidar com os requisitos envolvidos nos domínios de aplicação atuais. Por outro lado, a arquitetura de software visa incorporar a disciplina de reutilização e agregar suas vantagens, evitando retrabalho ou trabalho desnecessário, além de permitir aos engenheiros de software tomarem decisões sobre alternativas de projeto. Enfim, o foco recente em arquitetura de software se deve, sobretudo, à economia e à reutilização.

Dentre os benefícios de se considerar a arquitetura de software durante o desenvolvimento de um sistema, estão: (i) favorecer a gerência da complexidade; (ii) facilitar a comunicação entre os stakeholders envolvidos no desenvolvimento de software (desenvolvedores, clientes, gerentes, etc.); (iii) criar possibilidades de reutilização; (iv) viabilizar a evolução de sistemas; (v) reconhecer estruturas comuns entre sistemas; (vi) manter o controle da produção de software, já que os detalhes relativos a esforços e desenvolvimento e impacto de mudanças tornam-se mais claros; e (vii) permitir novas oportunidades para análise (ex. consistência, atributos de qualidade, atendimento a estilos arquiteturais).

Questão 4 (3 pontos)

Com relação ao desenvolvimento de software em geral:

(a) Qual a importância da modelagem para o desenvolvimento de software?

Resposta: Modelos são úteis para o entendimento de problemas, a difusão do conhecimento entre os desenvolvedores e a possibilidade de testar hipóteses antes de realizá-las. Eles são criados devido à limitação do ser humano em lidar com a complexidade.

(b) Quais são as dificuldades da Engenharia de Requisitos?

Resposta: Algumas das dificuldades são: a comunicação com o usuário, mudanças constantes, previsão de situações futuras, entendimento completo do problema e diversas formas de representação, dentre outras.

(c) Como o desenvolvimento orientado a objetos se difere do desenvolvimento tradicional?

Resposta: O desenvolvimento orientado a objetos se caracteriza por ser centrado em dados e processos, diferentemente do desenvolvimento tradicional que é centrado em processos ou dados de forma excludente. É também caracterizado por um desenvolvimento interativo incremental, diferentemente do desenvolvimento tradicional linear-sequencial.