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 – AD2 1° semestre de 2010.

Nome –

Observações importantes:

1- Prova com consulta.

2- ADs enviadas pelo correio devem ser postadas cinco dias antes da data final de entrega estabelecida no calendário de entrega de ADs.

3- Como a avaliação à distância é individual, caso sejam constatadas semelhanças entre provas de alunos distintos, será atribuída a nota ZERO a TODAS as provas envolvidas. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser distinta. ALÉM DISSO, às questões desta AD respondidas de maneira muito semelhantes às respostas oriundas dos gabaritos já publicados de ADs de períodos anteriores, será atribuída a nota ZERO, incluindo também cópias diretas e sem sentido de tópicos dos slides das aulas.

Questão 1 [4 pontos]

Em relação à Reutilização de Software, responda:

a) [1 ponto] Enuncie quatro obstáculos que se estabelecem durante a implantação de um programa de reutilização em uma organização e discuta uma possível solução para cada um deles.
b) [1 ponto] Pesquise e discuta brevemente pelo menos três casos de sucesso de programas de reutilização na indústria de software brasileira e/ou internacional (apresente as devidas referências ou fontes para cada um dos casos).
c) [1 ponto] Da área de Qualidade de Software, sabe-se que o modelo MPS, no contexto do programa de Melhoria de Processo do Software Brasileiro (MPS.BR), se baseia nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços

correlatos. Uma vez que uma das metas da Reutilização é agregar qualidade a produtos de software, o modelo MPS apresenta um processo chamado Gerência de Reutilização (GRU), que visa gerenciar o ciclo de vida dos ativos reutilizáveis que compõem estes produtos, estabelecendo, para isso, um conjunto de resultados a serem alcançados pela organização. Apresente e discorra sobre cada um destes resultados (DICA: verificar o Guia de Implementação – Parte 3: Fundamentação para Implementação do Nível E do MR-MPS[1]).

d) [1 ponto] Uma das abordagens mais populares da Reutilização é o Desenvolvimento Baseado em Componentes (DBC). Cite e explique pelo menos cinco características que formam o conceito de componentes quando aplicado no contexto da Engenharia de Software.

Resposta:

a) Os quatro obstáculos selecionados são:

i) **Estrutura organizacional inapropriada**: uma vez que implantar a Reutilização de Software não se trata apenas de uma decisão técnica, mas sim estratégica, um programa que a contemple deve lidar com o fato de que as estruturas organizacionais devem considerar diferentes necessidades que surgem quando a reutilização explícita e em larga escala está sendo adotada. Assim, visando contornar este problema, inicialmente, uma equipe separada pode ser definida para desenvolver, manter e certificar artefatos reutilizáveis, por exemplo, assessorando as equipes de desenvolvimento;

ii) **Carência de incentivos gerenciais**: a falta de incentivos da organização "proíbe" os gerentes de deixarem os engenheiros de software despenderem tempo para o desenvolvimento para e com reutilização, uma vez que o sucesso é freqüentemente mensurado apenas pelo tempo necessário para executar um projeto. Dessa forma, para contornar este problema, o programa de reutilização deve contemplar minimamente todas as proposições de valor dos *stakeholders* da organização por meio de uma definição bem clara dos benefícios e dificuldades envolvidos;

iii) **Limitações econômicas**: a reutilização permite a economia de recursos ao longo do tempo, mas não sem custos, os quais envolvem o custo de fazer algo reutilizável, o custo de reutilizar e os custos de definir e implementar um processo de reutilização. Ou seja, a reutilização requer investimentos em infraestrutura, metodologia, treinamento e ferramentas, com retornos significativos a médio ou longo prazo. Nesse sentido, desenvolver artefatos para reutilização pode ser mais caro do que desenvolvê-los sem reutilização. Para minimizar este problema, modelos de retorno de investimento devem justificar os investimentos considerando o contexto da organização, de maneira que as decisões contemplem alguns casos, por exemplo, como será tratado um componente que será utilizado apenas uma vez;

[1] Maiores informações sobre o modelo MPS e sobre os processos do Modelo de Referência MR-MPS estão disponíveis em http://www.softex.br/mpsbr/_guias/default.asp.

iv) **Dificuldade de busca e recuperação de artefatos reutilizáveis**: a fim de reutilizar efetivamente os artefatos desenvolvidos para este fim e/ou oriundos de outros contextos, deve haver formas eficientes de buscá-los e recuperá-los. A publicação destes em diretórios compartilhados pode ser inapropriada, uma vez que não permite a realização de buscas com base em parâmetros de documentação, estratégia de classificação mais flexível (e.g., baseada em facetas), avaliação de consumidores etc. A fim de resolver este problema, é importante que um mecanismo de armazenamento e recuperação, isto é, um repositório, seja desenvolvido de acordo com as necessidades e com o *layout* da organização ao prover meios para controlar o ciclo de vida dos ativos.

b) Os três casos de sucesso de programas de reutilização selecionados são[2]:

i) **IBM**: em 1991, o *Reuse Technology Support Center* foi estabelecido para coordenar e gerenciar as atividades de reutilização na IBM. Uma instituição importante estabelecida foi o *Reusable Parts Technology Center*, em Boeblingen, na Alemanha, com a missão de desenvolver partes de software reutilizáveis bem como de avançar no estado da prática da Reutilização de Software. Esta instituição remonta 1981 e atendeu os seguintes passos desde esta data: (1) a necessidade de uma central que gerenciasse as partes de software; (2) o tratamento da modelagem (*design*) de produtos baseados em reutilização; (3) a incorporação da reusabilidade enquanto características de produtos de software por meio dos tipos abstratos de dados; (4) a evolução para uma abordagem de construção de blocos; e, finalmente, (5) o desenvolvimento de um ambiente de reutilização. Assim, em agosto de 1991, a primeira release da biblioteca de classes para C++ foi disponibilizada na IBM (*IBM's Collection Class Library*), e ao final de 1991, 57 projetos com 340 programadores já estava utilizando os blocos de construção em C++. Em junho de 1993, a *Collection Class Library* foi disponibilizada como parte do compilador IBM C Set++, para C++;

ii) **HP**: os esforços para a reutilização foram iniciados no começo dos anos 1980, com a meta de reduzir o *time-to-market*, e melhorar a produtividade e a consistência dos produtos de software. Dez anos depois, a HP começou o *Corporate Reuse Program* de forma multifacetada a fim de reunir informação sobre reutilização dentro da própria HP e de outras companhias. Os responsáveis pelo programa de reutilização da HP detectaram que liderança e suporte gerencial, mudança organizacional e a criação de uma "mentalidade" de reutilização eram elementos fundamentais para o sucesso. Além disso, foi adotado um modelo incremental para a adoção da reutilização, de modo que adotar reutilização significa aumentar, de maneira incremental, o investimento e a experiência na companhia e, conseqüentemente, focar em reutilizar e aprender mais sobre o processo, o que aumenta o nível de reutilização;

[2] Maiores informações sobre os três casos podem ser obtidas em: ALMEIDA, E.S., ALVARO, A., GARCIA, V.C., MASCENA, J.C.C.P., BURÉGIO, V.A.A., NASCIMENTO, L., LUCRÉDIO, D., MEIRA, S.L., 2007, *C.R.U.I.S.E. – Component Reuse in Software Engineering*. Recife, PE, Brasil, RISE/CESAR.

iii) **Motorola**: no início dos anos 1990, a Motorola decidiu ampliar a qualidade e a produtividade de software com foco na reutilização, e uma das grandes barreiras foi mudar o ambiente de desenvolvimento de software de uma "mentalidade" focada no hardware para outra focada no software. Uma vez que o processo foi estabelecido, a Motorola iniciou um processo de reutilização de software de três fases: (1) *Grass-Roots Beginning*: criação de uma força tarefa para reutilização a fim de investigar e fazer recomendações sobre a reutilização de software na organização, lidando com a necessidade de educação e motivação, métodos, tecnologia e implementação – esta força tarefa foi substituída por um grupo de trabalho em reutilização formado por 15 engenheiros, focados em educar os stakeholders ao novo processo (alguns participantes eram voluntários e não trabalhavam *full-time*, o que gerou certas dificuldades); (2) *Senior-Management Involvement*: os gerentes medianos estavam relutantes quanto aos custos iniciais e quanto ao vagaroso retorno de investimentos (três anos ou mais), o que levou à responsabilidade da gerência sênior de iniciar a reutilização – George Fisher, CEO da Motorola, assumiu esta missão, que consistia em tornar a gerência mais envolvida com o estado da prática do software, com o desafio de promover mudanças culturais (mais do que técnicas); e (3) *Future Markets and Tools*: algumas equipes perceberam que a reutilização prometia mais do que economia interna, abrindo oportunidades para novos mercados para componentes e ferramentas de apoio – assim, a Morotola começou o desenvolvimento de uma infraestrutura (*reuse toolkit*) que permitia capturar modelos, aplicar reengenharia e promover o desenvolvimento baseado em componentes, cujo protótipo foi iniciado em 1996 (a literatura não apresenta qualquer resultado relacionado).

c) Os cinco resultados a serem alcançados são:

i) **GRU 1 – Uma estratégia de gerenciamento de ativos é documentada, contemplando a definição de ativo reutilizável, além dos critérios para aceitação, certificação, classificação, descontinuidade e avaliação de ativos reutilizáveis**: a organização necessita delimitar o escopo de utilização dos ativos reutilizáveis, isto é, definir que ativos serão passíveis de reutilização e os pontos nas atividades dos processos em que se dará essa reutilização. Essa escolha caracteriza como o repositório se organizará, favorecendo os procedimentos de busca e seleção, definindo a arquitetura da biblioteca, além de toda infraestrutura para sua utilização. Um conjunto de critérios é estabelecido para qualificar um ativo reutilizável desde a possibilidade de este fazer parte da biblioteca até o momento em que este não se faz mais necessário. Periodicamente, os critérios são revistos, observando as próprias necessidades da execução do processo GRU;
ii) **GRU 2 – Um mecanismo de armazenamento e recuperação de ativos reutilizáveis é implantado**: um mecanismo de armazenamento e recuperação de ativos reutilizáveis é definido, levando em consideração as suas informações de documentação, compatível com as necessidades da

organização em catalogá-los e recuperá-los. É interessante permitir que os engenheiros de software externem a necessidade de catalogação de certos tipos de ativos. Uma vez definido esse mecanismo, este é implantado e disponibilizado para os engenheiros de software da organização através da biblioteca;

iii) **GRU3 - Os dados de utilização dos ativos reutilizáveis são registrados**: a informação de que consumidor utiliza determinada versão de um ativo reutilizável é registrada, para estabelecer o elo produtor-consumidor. Isso é importante para notificações acerca do ativo, além de tornar possível a observação da sua utilização, isto é, informações que caracterizem tendência ou comportamento, podendo auxiliar o gerente de ativos reutilizáveis na realização das atividades de manutenção da biblioteca (e.g., descontinuar ativos), notificando o produtor e os consumidores;

iv) **GRU4 - Os ativos reutilizáveis são periodicamente mantidos, segundo os critérios definidos, e suas modificações são controladas ao longo do seu ciclo de vida**: os ativos reutilizáveis estão sob o processo gerência de configuração e de garantia de qualidade, o que favorece a sua manutenção e a garantia da aderência aos padrões estabelecidos para os critérios definidos;

v) **GRU5 - Os usuários de ativos reutilizáveis são notificados sobre problemas detectados, modificações realizadas, novas versões disponibilizadas e descontinuidade de ativos**: os dados de utilização de ativos reutilizáveis são aproveitados pelo gerente de ativos reutilizáveis para notificar os consumidores sobre alterações que ocorreram nos ativos reutilizados. Quando as novas versões de ativos reutilizáveis se tornam disponíveis na biblioteca, os seus consumidores são notificados: se a versão for oriunda de uma evolução do ativo ou de uma correção, a substituição pode ou não ser opcional, respectivamente.

d) As cinco características selecionadas são:

i) **Autocontido**: característica dos componentes de poderem ser reutilizáveis sem a necessidade de incluir ou depender de outros componentes. Caso haja alguma dependência, todo o conjunto deve ser abstraído como um componente reutilizável;

ii) **Identificação**: componentes devem ser facilmente identificados, ou seja, devem estar armazenados em um único lugar, ao invés de espalhados e misturados com outros artefatos de software ou documentação;

iii) **Funcionalidade**: componentes têm uma funcionalidade clara e específica que realizam e/ou descrevem. Assim, componentes podem realizar funções ou podem ser simplesmente descrições de funcionalidades (e.g., artefatos do ciclo de vida);

iv) **Interface**: componentes devem possuir interfaces claras, que indicam como estes podem ser reutilizados e conectados a outros componentes, devendo-se ocultar os detalhes que não são necessários para a reutilização per si (encapsulamento);

v) **Documentação**: A existência de documentação é indispensável para a reutilização. O tipo de componente e a sua complexidade indicarão a conveniência do tipo de documentação.

Questão 2 [3 pontos]

O padrão *Polymorphism* visa tratar alternativas em função do tipo da classe e também criar componentes de software substituíveis.

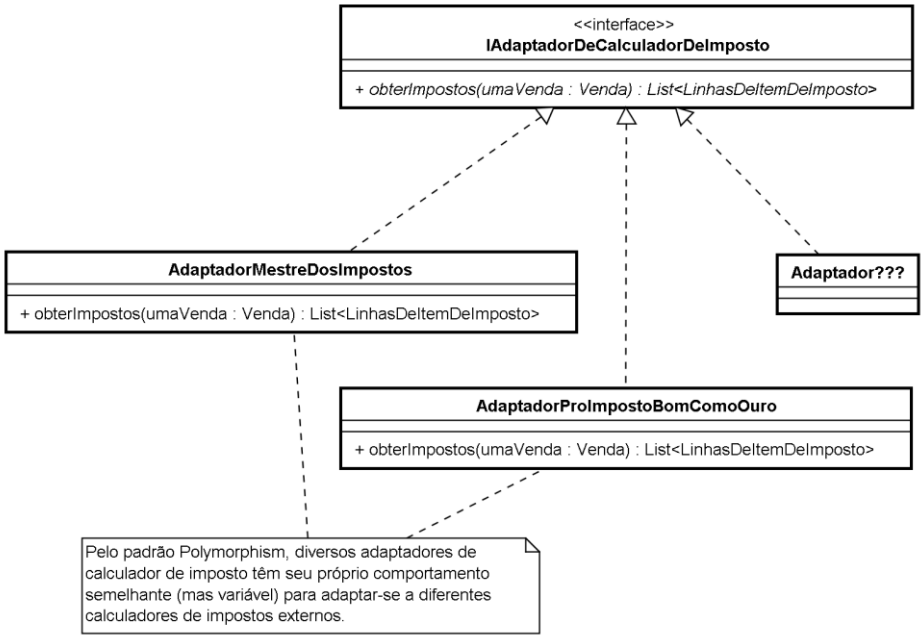
a) [1 ponto] Descreva um exemplo, **diferente daquele visto em aula**, de uma situação onde esse padrão é útil.
b) [1 ponto] Quais são os benefícios alcançados com o uso desse padrão?
c) [1 ponto] Desenhe um modelo de classes exibindo como o padrão pode ser utilizado na situação descrita.

Resposta:

a) No aplicativo de PDV, podem existir vários calculadores de impostos terceirizados, que precisam ser apoiados (e.g., *Tax-Master* ou Mestre-dos-Impostos, e *Good-As-Gold-Tax-Pro* ou Pro-Imposto-Bom-Como-Ouro), e o sistema precisa se integrar com diversos deles. Cada calculador de imposto tem uma interface diferente e, portanto, há um comportamento similar, mas variado, a ser adaptado para cada uma dessas interfaces fixas ou APIs. Um produto pode apoiar um protocolo de soquete TCP puro, outro pode oferecer uma interface SOAP e um terceiro pode apresentar uma interface RMI Java.

b) Facilidade de manutenção e facilidade de inserção de um novo tipo de autorização.

c)



Como o comportamento das adaptações do calculador varia de acordo com o tipo de calculador, por meio do padrão *Polymorphism*, deve-se atribuir a responsabilidade da adaptação aos próprios objetos calculadores externos ou o adaptador para o calculador. Ao se enviar uma mensagem para o objeto local, será feita uma chamada ao calculador externo em sua API nativa. Cada método *obterImpostos* recebe o objeto *Venda* como parâmetro para que o calculador possa analisar a venda. A implementação de cada método *obterImpostos* será diferente: *AdaptadorMestreDosImpostos* vai adaptar a solicitação à API do *Tax-Master*, e assim por diante.

Questão 3 [3 pontos]

Em relação à Arquitetura de Software, responda:

a) [1 ponto] Apresente as diferentes visões requeridas por uma arquitetura e discuta o propósito de cada uma delas.
b) [1 ponto] Como essas visões auxiliam na redução da distância entre a descrição arquitetural e a implementação?
c) [1 ponto] Os estilos arquiteturais são utilizados em qual dessas visões?

Resposta:

a) As quatro visões da arquitetura operam em conjunto por meio da **visão de cenários**, que utiliza cenários como guia ao projeto da arquitetura (e.g., lidar com requisitos amplos e complexos, identificar interfaces críticas, ajudar os projetistas a se concentrarem em questões concretas e determinar as prioridades do sistema). Essas visões são apresentadas a seguir:

    i) **Visão lógica**: sua principal contribuição é oferecer um retrato estático das classes fundamentais e seus relacionamentos. Dá enfoque aos aspectos funcionais do sistema. A visão lógica é composta por diagramas de classes que apresentem as principais abstrações do sistema: as principais classes e pacotes, e os relacionamentos entre eles;
    ii) **Visão de desenvolvimento**: seu principal objetivo é mostrar como o código é organizado em pacotes e bibliotecas. Os pacotes são considerados conjuntos de componentes (i.e., unidade de código fonte que serve como um bloco de construção física do sistema) logicamente relacionados, de forma que cada um dos componentes esteja em um único pacote. A visão de desenvolvimento pode incluir novos pacotes para tratarem a funcionalidade de baixo nível;
    iii) **Visão de processos**: demonstra as atividades e tarefas realizadas pelo sistema. Focaliza a decomposição do sistema em processos, de forma a exibir a alocação de componentes executáveis a processos e atualizar o diagrama de componentes para apresentar os processos aos quais os componentes são alocados. Outro ponto importante é que a visão de processos dá um enfoque a atributos não funcionais do sistema como disponibilidade, desempenho e confiabilidade;

iv) **Visão física**: mostra os processadores, dispositivos e ligações no ambiente de operação. Mapeia os processos em unidades de processamento, considerando que um processo é uma linha de controle executada em uma unidade de processamento e que sistemas grandes ou distribuídos podem ser quebrados em diversos processos. Considera tanto requisitos (e.g., capacidade de resposta, desempenho e tolerância a falhas) como peculiaridades de implementação (e.g., necessidade de processadores específicos). Diagramas de implantação (*deployment diagrams*) mostram as unidades de processamento e seus processos.

b) Através da elaboração de modelos que representam uma base mais concreta para o sistema a ser implementado, se comparados à descrição e aos requisitos identificados na forma de (longos) documentos textuais. Estes modelos consideram as diferentes visões (lógica, desenvolvimento, processos e física) que servem como insumo para as etapas de desenvolvimento e implantação do sistema.

c) Uma vez que os estilos arquiteturais são definidos como coleções de componentes, descrições de suas interações (conectores) e suas restrições, permitindo a reutilização de um conjunto de características desejadas (requisitos funcionais e não funcionais), eles são utilizados na *visão de desenvolvimento*.