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 2008.

Nome –

Observações:

1. Prova com consulta.

Atenção: 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.

Questão 1 [1 ponto]

Considerando o processo de Gerência de Reutilização, aponte suas quatro etapas, juntamente com os objetivos e atividades de cada uma delas, de forma a explicar como essas etapas se conectam para prover o processo supracitado.

R: As quatro etapas (ou subprocessos) do processo de Gerência de Reutilização são: *planejamento de reutilização, criação de artefatos, gerência de artefatos* e *utilização de artefatos*, sendo que essas três últimas são encapsuladas em uma etapa maior, a *execução* do processo de Gerência de Reutilização.

O planejamento de reutilização tem por objetivo definir uma estratégia de reutilização e um plano para sua implementação dentro da organização. Essa etapa interage com as demais etapas, considerando-as encapsuladas como uma etapa maior (*execução* do processo de Gerência de Reutilização), ao definir e estruturar objetivos, estratégias, processos e recursos relativos ao programa de reutilização. Por outro lado, a etapa de planejamento recebe como *feedback* novos requisitos, lições, processos e artefatos advindos da etapa maior de execução. As atividades envolvidas são: (i) estabelecer a estratégia para criação, gerência e utilização de artefatos reutilizáveis; (ii) integração da reutilização ao processo de desenvolvimento; e (iii) controle e evolução do processo.

A criação de artefatos visa produzir software e produtos associados para a reutilização, consistindo no chamado *desenvolvimento para reutilização*. Ao se basear nos objetivos,

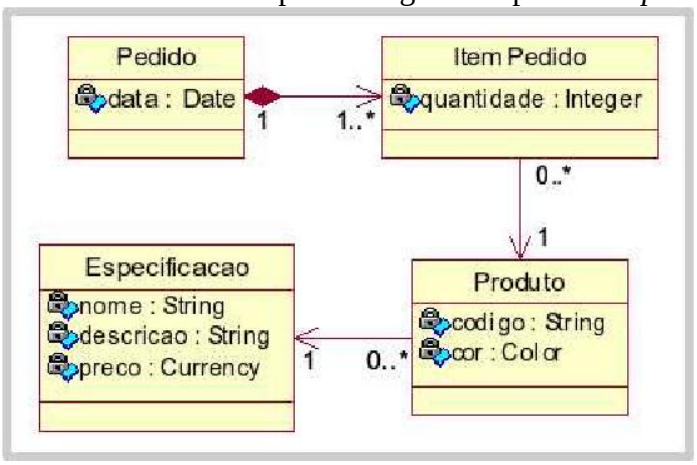
estratégias, processos e recursos providos pela etapa de planejamento, essa etapa gera artefatos que representam o insumo para a próxima etapa (gerência de artefatos), e recebe como *feedback* desta etapa (gerência) as lições aprendidas, a fim de que suas atividades estejam em processo de constante aperfeiçoamento. Além disso, a etapa de criação também recebe requisitos provenientes da etapa de utilização, indicando a necessidade de manutenção e/ou desenvolvimento de novos artefatos. Essas atividades são: (i) análise e modelagem do domínio (*engenharia de domínio*); (ii) desenvolvimento de uma infra-estrutura de reutilização; e (iii) evolução do processo.

A gerência de artefatos busca coletar, avaliar, descrever e organizar artefatos reutilizáveis para garantir sua disponibilização às etapas (subprocessos) de criação e utilização de artefatos, fornecendo lições aprendidas para a primeira e artefatos e descrições para a segunda, respectivamente. Além disso, a gerência de artefatos recebe os artefatos produzidos pela etapa de criação e é alimentada com um *feedback* acerca de lições aprendidas, relativo à etapa de utilização. Suas principais atividades correspondem a: (i) aquisição, aceitação, classificação, catalogação e certificação de artefatos reutilizáveis; (ii) coleta de métricas e administração do repositório; e (iii) manutenção e evolução de artefatos.

Por fim, a utilização de artefatos almeja compor o sistema a partir de artefatos reutilizáveis, ou seja, viabilizar o *desenvolvimento com reutilização*. Essa etapa recebe artefatos e descrições advindos da etapa de gerência e alimenta a execução do processo de Gerência de Reutilização, fornecendo lições aprendidas para a etapa de gerência e requisitos para a etapa de criação. Para isso, suas atividades abrangem: (i) identificação, compreensão, avaliação, seleção, adaptação e integração de artefatos; e (ii) *feedback* ao *planejamento*, *criação* e *gerência* de artefatos.

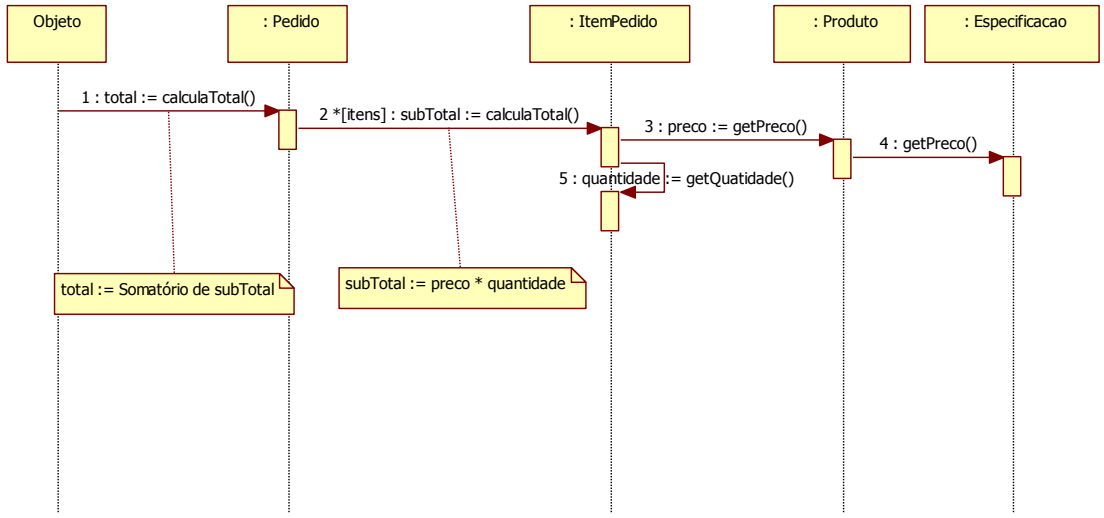
Questão 2 [1 ponto]

Desenvolva o modelo abaixo (através de modelo de seqüência) para inserir a responsabilidade de cálculo do total do pedido segundo o padrão *Expert*:



R: Conforme o padrão *Expert*, em um sistema de PDV, o responsável pelo cálculo do total de um pedido deveria ser a própria classe `Pedido`, uma vez que deve-se atribuir a

responsabilidade ao especialista, ou seja, à classe que tem a informação necessária para satisfazer a responsabilidade (no caso, o cálculo do total do pedido). Isso pode ser visualizado pelo diagrama de seqüência abaixo:



Questão 3 [1 ponto]

A utilização do conceito de interfaces ajuda na substituição de componentes. Entretanto, existem alguns problemas relacionados:

(a) Como poderia ser feito *refactoring* de um sistema que precisa ter partes substituídas, mas que não foi projetado para isso (não usa componentes)?
(b) Como poderia ser feita a modificação de um serviço declarado público na interface?

Obs.) Tente estruturar uma sistematização simplificada para cada uma das duas questões.

R: (a) Dado que as partes desse sistema não se comunicam por meio de interfaces (não foram utilizados componentes em seu projeto), o *refactoring* iniciaria pela identificação das partes principais do sistema e introdução de interfaces entre essas partes. A partir desse momento, cada uma dessas partes do sistema poderia ser aprimorada individualmente, testada e substituída sem que as demais partes sejam afetadas. Em uma situação ideal, essas partes poderiam se tornar componentes, facilitando a manutenção futura.

(b) Se um serviço é declarado como publico na interface de uma classe, outras classes podem estar fazendo uso do mesmo. Sendo assim, é necessário inicialmente comunicar aos possíveis usuários do serviço que uma modificação precisa ser feita. Nesse período, tanto a versão antiga quanto a nova do serviço devem estar disponíveis, para possibilitar uma migração gradativa dos clientes. Somente quando o período de migração se encerrar que a versão antiga do serviço pode ser removida, restando somente a nova versão do mesmo.

Questão 4 [1 ponto]

(a) Defina componentes, explicando suas características fundamentais.
(b) Qual a diferença entre um processo convencional e um processo ideal com reutilização e DBC?

R: (a) Componentes são elementos que podem ser desenvolvidos independentemente e entregues como unidades, que podem ser compostos com outros componentes, e que explicitam as suas funcionalidades por meio de interfaces providas e requeridas. Suas características fundamentais são: (i) unidade de composição auto-contida (encapsulamento), reunindo um conjunto de características e operações implementadas da forma mais genérica possível; (ii) interfaces bem definidas (contrato), provendo e requerendo informações do mundo exterior por meio de um protocolo de comunicação bem definido; (iii) grau de maturidade e documentação (qualidade), contendo um conjunto de artefatos que o descrevem a fim de reduzir riscos devido ao seu propósito de reutilização; (iv) especificado em diferentes níveis (abstração), abrangendo não apenas artefatos de código, mas também artefatos em nível de análise, projeto e testes; e (v) obedece a restrições (arquitetura/plataforma) previamente identificadas e documentadas.

(b) A diferença consiste na inserção de uma fase a mais no processo tradicional (análise – projeto arquitetural – projeto detalhado – codificação – testes), entre projeto arquitetural e projeto detalhado: a busca por componentes. O objetivo é incorporar reutilização ao longo do processo de desenvolvimento por meio do desenvolvimento baseado em componentes. A busca de componentes consiste na customização de componentes e/ou na adaptação do projeto arquitetural, uma vez que nem sempre os componentes desejados são encontrados ou mesmo aqueles encontrados não são exatamente o que se esperava.

Questão 5 [1 ponto]

O que são interfaces? Quais os seus tipos? Explique-os e exemplifique-os considerando um componente de cálculo de conversão de moedas.

R: Interfaces consistem em um conjunto de assinaturas de operações que podem ser invocadas por um cliente (desde que cada operação tenha sua semântica especificada) e visam determinar como o componente pode ser reutilizado e interconectado com outros componentes.

As interfaces podem ser de negócio (providas ou requeridas) ou de configuração. As interfaces providas (*provided interfaces*) servem para fornecer serviços de um determinado componente para outros componentes de necessitam desse serviço. As interfaces requeridas (*required interfaces*) representam os serviços que um componente necessita para efetuar sua função (cumprir seu contrato) e são provenientes de outros componentes. Por fim, as interfaces de configuração (*configuration interfaces*) visam possibilitar a variabilidade do componente segundo alguns parâmetros pré-estabelecidos,

que devem ser ajustados para que a funcionalidade desejada sobre o componente seja alcançada.

Como exemplo, um componente de cálculo de conversão de moedas apresentaria: como interface provida, uma operação que resultasse no valor calculado após a conversão; como interface requerida, operações que fornecem uma tabela de conversão de valores padrão; e como interface de configuração, os diferentes tipos de moedas suportados pelo componente.

Questão 6 [1 ponto]

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

R: 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 7 [1 ponto]

Baseado no capítulo 2 do livro do Mendes, enumere uma vantagem e uma desvantagem para cada um dos estilos arquiteturais estudados (*Pipes & Filters*, em camadas, processos distribuídos e orientado a objetos).

R: QUESTÃO ANULADA. A nota dessa questão será proporcional às demais notas da AD. Ou seja, após calcular a nota N das demais questões (onde o máximo é 9), a nota total da AD será (10/9 * N).

Questão 8 [1 ponto]

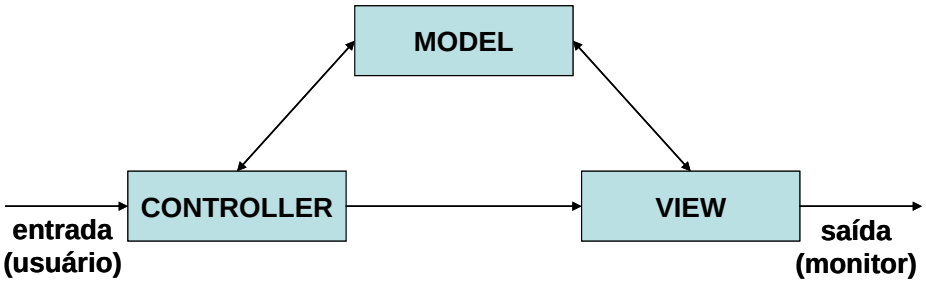
Pesquise acerca da arquitetura de interface com o usuário MVC (*Model-View-Controller*) e explique seus princípios e principais componentes, ilustrando o relacionamento entre eles. Além disso, cite um *framework* utilizado na indústria de software que implemente a arquitetura MVC. Dica: consultar a seção 7.3 do livro do Mendes.

R: A arquitetura MVC tem como base o paradigma de multiagentes. Um modelo arquitetural com base em agentes organiza um sistema interativo em termos de um conjunto de unidades computacionais, denominadas de *agentes*[1]. Nesse caso, a organização do sistema é feita em termos de um conjunto de agentes cooperativos.

Na arquitetura MVC, um agente é visto sob três perspectivas funcionais. Assim, com base nessas perspectivas, a arquitetura MVC divide o sistema nos componentes *Model* (modelo), *View* (visão) e *Controller* (controle). O componente *model* define a competência do agente, ou seja, seu núcleo funcional. O componente *view* define o comportamento perceptivo do agente para a saída, fornecendo suporte à apresentação. O componente *controller* denota o comportamento perceptivo do agente para entrada. Ele é encarregado de tratar as entradas e comandos de usuário, bem como gerenciar toda a informação trocada entre o usuário e o componente *model*. Na arquitetura MVC, o componente *model* se comunica diretamente com os componentes *view* e *controller*, conforme apresentado na figura abaixo. Como exemplo de um ciclo de interação típico, uma informação de entrada do usuário é seguida de uma notificação feita pelo componente *controller* para o componente *model* a fim de fazer uma atualização. Além disso, o componente *model* realiza as modificações necessárias e notifica seus dependentes (através de um *broadcast*). Outra característica importante da arquitetura MVC está no fato de que considerações pertinentes à aplicação (componente *model*) são separadas daquelas pertinentes à interface (componente *controller* e componente *view*).

Um dos *frameworks* que implementa a arquitetura MVC, bastante utilizado pela indústria, é o *Struts*.

[1] Um agente tem um estado, possui conhecimento bem como é capaz de inicializar e reagir a eventos. Os agentes que se comunicam diretamente com o usuário são comumente chamados de interação (ou *interactor*). Um objeto de interação fornece a representação perceptiva de seu estado interno. Aqui, considera-se a noção do conceito de agentes como uma perspectiva da definição mais geral utilizada em inteligência artificial distribuída. Em inteligência artificial, os agentes podem ter características cognitivas e reativas. Entretanto, é importante observar que isso depende principalmente de suas capacidades de representação de conhecimento e raciocínio.



Questão 9 [1 ponto]

O que você entende por arquitetura de referência? Utilize um exemplo para ilustrar sua resposta.

R: Uma arquitetura de referência é um padrão de arquitetura que descreve todos os sistemas num domínio. Se uma arquitetura pode ser vista como um projeto reutilizável, então uma arquitetura de referência constitui um projeto reutilizável para uma classe de sistemas em um domínio específico. Dessa forma, uma arquitetura de referência compreende os seguintes elementos: (i) *estilo arquitetural*: serve para definir os tipos de componentes e conectores, bem como permite a configuração de uma aplicação (note que uma arquitetura de referência pode combinar estilos arquiteturais); (ii) *componentes típicos*: envolve os componentes típicos e variantes deles encontrados no estilo; e (iii) *interface com outros domínios*: necessidade de descrever a interface com outros domínios (isso pode ocorrer se uma aplicação requer que algum serviço ou facilidade seja fornecido por algum outro domínio).

Como exemplos, podemos citar o próprio MVC, uma arquitetura de referência relativa o modelo de referência[2] GUI (*Graphical User Interface*), e CORBA (*Common Object Request Broker Architecture*), uma arquitetura de referência para o modelo de referência sistemas distribuídos. CORBA é a arquitetura padrão criada pelo OMG (*Object Management Group*) para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos. Em face da diversidade de hardware e software que encontramos atualmente, CORBA atua de modo que os objetos (componentes dos produtos de software) possam se comunicar de forma transparente ao usuário, mesmo que para isso seja necessário interoperar com outro software, em outro sistema operacional e em outra ferramenta de desenvolvimento. CORBA é um dos modelos mais populares de objetos distribuídos, juntamente com o DCOM (*Distributed Component Object Model*), formato proprietário da Microsoft (outra arquitetura de referência para o modelo de referência sistemas distribuídos).

Questão 10 [1 ponto]

[2] Um modelo de referência é uma divisão de funcionalidades juntamente com o fluxo de dados entre as partes. É uma decomposição padrão de um problema conhecido em partes que cooperativamente resolvem o problema. Advindos da experiência, os modelos de referência são características de um domínio maduro e descrevem em termos gerais como as partes se inter-relacionam para alcançar seu propósito coletivo.

Discuta cada uma das DSSAs (*Domain Specific Software Architectures*) apresentadas no curso (*sistemas de informação, sistemas de tempo real, sistemas inteligentes adaptativos* e *sistemas para comunicação*) e os benefícios específicos de seu uso.

R: Arquiteturas em Sistemas de Informação correspondem às DSSAs que envolvem suporte a armazenamento, recuperação e processamento de dados, com o objetivo de prover o gerenciamento da informação. Como exemplo, pode-se citar arquiteturas que apóiam sistemas bancários, sistemas comerciais e sistemas de catalogação em bibliotecas, dentre outros. Nessa DSSA, dois conjuntos de atributos de qualidade devem ser levados em consideração: (i) desempenho, segurança, integridade e disponibilidade; e (ii) usabilidade e manutenibilidade. Uma das principais arquiteturas que suportam essa DSSA é a arquitetura cliente-servidor, que consiste no processamento distribuído entre dois elementos: *cliente*, que requisita serviços, e *servidor*, que realiza os serviços solicitados pelos clientes. Além disso, a arquitetura cliente-servidor exige a comunicação entre os dois elementos, necessitando, para isso, de uma rede entre os computadores e de um protocolo de comunicação. Um dos grandes benefícios dessa DSSA está na melhoria dos níveis de produtividade e na facilidade de tomada de decisão em uma organização, uma vez que permite aos funcionários maior acesso aos dados.

Arquiteturas em Sistemas de Tempo Real envolvem as DSSAs que devem apoiar respostas aos estímulos recebidos dentro de um estrito intervalo de tempo. Para isso, devem tratar a disponibilidade de componentes redundantes, a fim de prover suporte de tolerância a falhas e a necessidade de se fazer diagnósticos para a detecção de falhas e a reconfiguração do sistema com a substituição de componentes defeituosos. Dentre os benefícios agregados à utilização dessa DSSA estão o reúso de requisitos, de projeto, de componentes de software e de métodos, o que agrega, conseqüentemente, maior confiabilidade à classe de sistema de tempo real.

Arquiteturas em Sistemas Inteligentes Adaptativos se referem a DSSAs que devem atribuir maior atenção a características como habilidades de percepção, raciocínio e ação, uma vez que propiciarão a geração de sistemas inseridos em ambientes dinâmicos e complexos sob condições de incerteza e mudanças. Como requisitos funcionais, deve-se considerar: (i) percepção concorrente; (ii) sensibilidade a prioridades e a condições limites de tempo; (iii) controle global do comportamento; e (iv) raciocínio e ação. Um dos principais benefícios relacionados ao uso dessa DSSA contempla a idéia de maior modularidade, uma vez que o domínio dessa DSSA pode ser particionado em subdomínios envolvendo aplicações de robôs autônomos (que requer capacidade de navegação e planejamento de ações) e sistemas de monitoramento (exigindo capacidade de detecção de erros e classificação de padrões).

Arquiteturas em Sistemas para Comunicação é uma DSSA que foca na natureza distribuída dos sistemas atuais e deve suportar como mecanismo de comunicação o encapsulamento e a transparência de localização, com o uso da arquitetura em camadas. Nesse caso, as redes são estruturadas em camadas e protocolos, onde cada camada agrupa um conjunto de tarefas em um determinado nível de abstração e a quantidade e funcionalidade de cada camada varia de rede para rede. Como vantagens dessa DSSA,

encontram-se a redução da complexidade do sistema e a possibilidade de maior interoperabilidade.