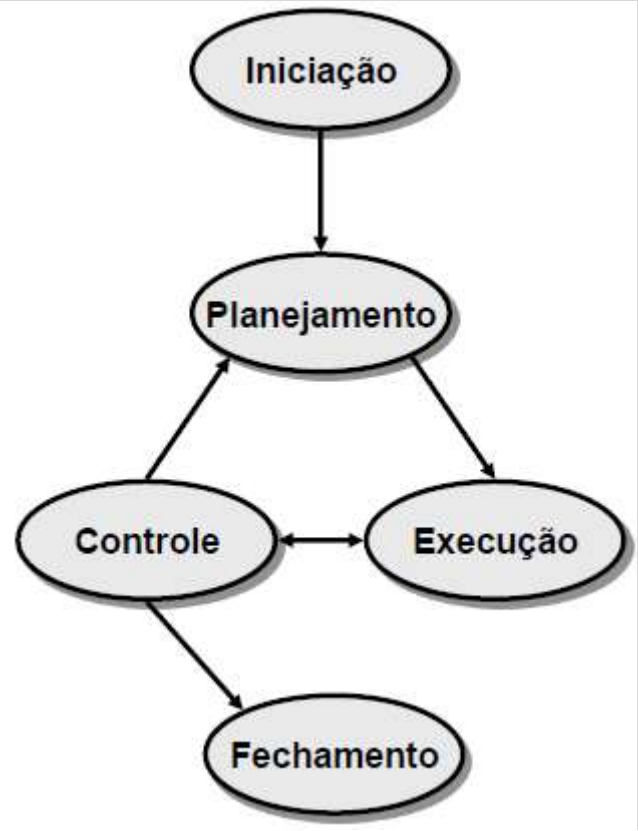
Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina: Engenharia de Software
AD2 2° semestre de 2018.

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!
Os tópicos tratados como pesquisa na AD2 poderão ser questionados na AP2 ou AP3!

1. Quais são os cinco grupos de processos encontrados no gerenciamento de um projeto de software? O que se deve fazer em cada um destes grupos? Desenhe um diagrama mostrando como eles se relacionam? (Valor: 1,5 ponto)

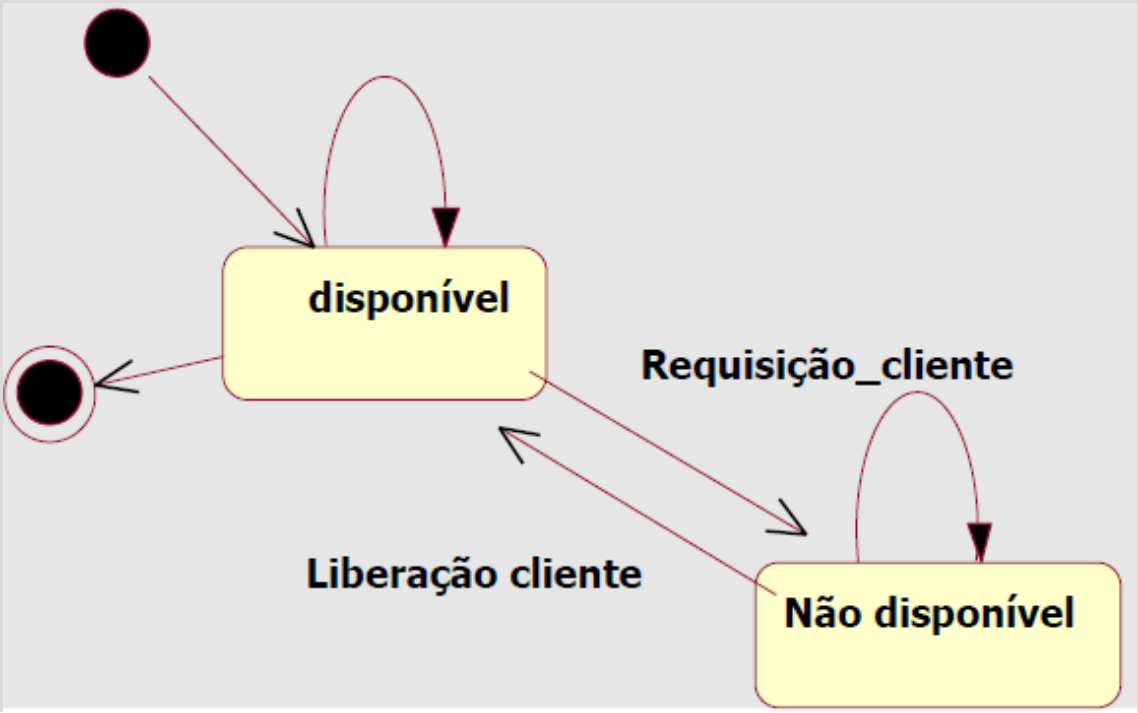
Os cinco grupos de processos em que podemos dividir o gerenciamento de projetos são a iniciação, o planejamento, a execução, o controle e o fechamento. A iniciação é a primeira fase do projeto, em que a alta administração da empresa autoriza o início do projeto e autoriza o gerente de projetos à sua coordenação. No planejamento, o gerente de projetos e a equipe definem um plano para o desenvolvimento do projeto. A execução consiste em colocar o plano em prática e o controle consiste em comparar o plano com o andamento da execução. Por fim, o fechamento é o encerramento administrativo do projeto.



2. Quais são os componentes de um diagrama de estados da UML? Mostre um diagrama de estados e explique como estes componentes se relacionam e o quê cada um representa no diagrama. (Valor 1,5 ponto)

Diagramas de estado são compostos por estados e transições. Um estado representa uma condição em que um objeto de uma classe do projeto pode se encontrar durante seu ciclo de vida no sistema. Existem dois estados especiais, que são o estado inicial (onde começa o ciclo de vida) e o estado final (que pode ser múltiplo, representando o fim do ciclo de vida). Transações são eventos que provocam a troca de estado, sendo representadas como setas entre dois estados do diagrama

Um exemplo de diagrama de estado é apresentado a seguir:

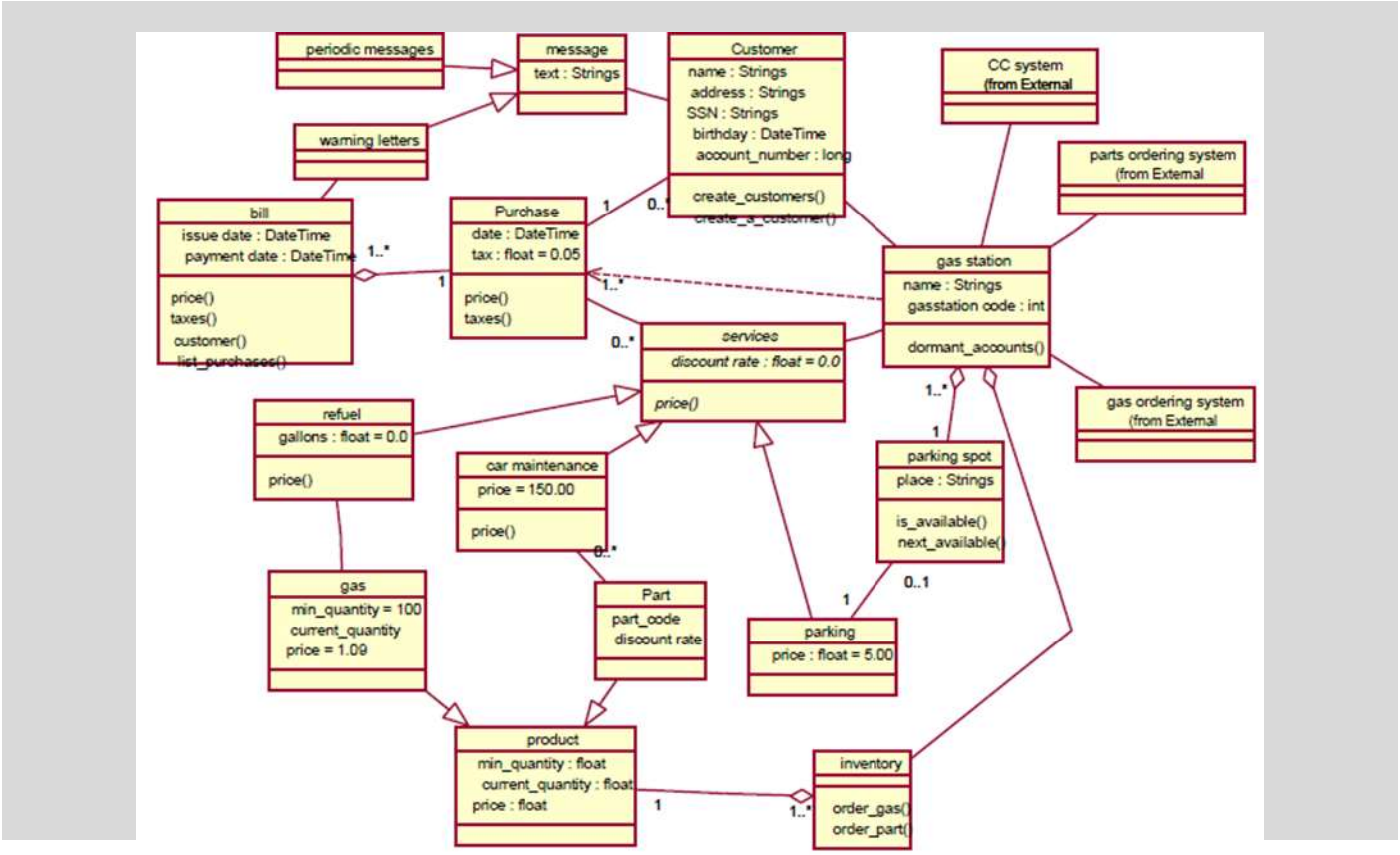


3. O que representa o conceito de acoplamento em software. Como podemos observar acoplamento no software? Como medir acoplamento? Apresente um exemplo de acoplamento em software. (Valor: 1,5 ponto)

Acoplamento indica o grau de dependência entre os componentes (ou módulos) do software. Um software com baixo acoplamento é sempre desejável. Um software com alto grau de acoplamento indica que a dependência entre módulos é grande, aumentando o risco de efeito colateral e, consequentemente, de falhas. Em relação a manutenção, o esforço aumenta (e os riscos de falha também) tendo em vista que alterações realizadas em um determinado módulo tem o potencial de afetar os módulos dependentes

Uma métrica que pode ser usada para medir acoplamento é a CBO (coupling between objects). Basicamente ela mede o numero de relacionamentos existente entre as classes que irão instanciar os objetos.

No diagrama de classes abaixo, a classe Gas Station possui acoplamento alto (8) em comparação com a classe periodic messages (1). Portanto, objetos da classe Gas Station são altamente acoplados, dependendo de 8 outros objetos para desempenhar suas responsabilidades.



4. Explique o que é teste funcional. Apresente três exemplos de critérios de teste que podem ser utilizados para projetar casos de teste para este tipo de teste. (valor: 1,5 ponto)

Os testes funcionais, também chamados testes de caixa fechada, enxergam o software de forma macro, na perspectiva do usuário, sem perceber detalhes de sua estrutura interna. Os testes funcionais baseiam-se na especificação do software para derivar os casos de testes. Exemplos de critérios que podem ser aplicados para testes funcionais são particionamento em classes de equivalência e Análise de Valor Limite

5. Atividade de pesquisa (não vale copiar e colar! Você deve pesquisar e explicar com suas palavras. Indique fontes alternativas que usar!): (valor 2,0 pontos)

O que diz o Código de ética do Profissional de Informática da Sociedade Brasileira de Computação (http://www.sbc.org.br/institucional-3/codigo-de-etica)? Como ele se compara ao Código de ética e de Prática Profissional da Engenharia de Software, segundo as recomendações da ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices (https://www.computer.org/cms/Computer.org/professional-education/pdf/doc.pdf)? Qual a importância de seguir um código de ética em nossos projetos de engenharia de software?

Tanto o código de ética da SBC quanto o da ACM/IEEE-CS fornecem diretrizes para a conduta dos profissionais de informática. O código da SBC consiste em um conjunto de 12 artigos que descrevem os deveres dos profissionais de informática. Já o código da ACM/IEEE-CS consiste em um conjunto de 8

artigos. Uma diferença entre os códigos é que o ACM/IEEE-CS, em sua versão completa, possui um detalhamento dos 8 artigos propostos

6. Atividade de pesquisa II (não vale copiar e colar! Você deve pesquisar e respondem com suas palavras. Indique fontes alternativas que usar!): (valor 2,0 pontos)

   A engenharia de sistemas de software contemporâneos (IoT, Industria 4.0, Sistemas de Sistemas, Sistemas Sensíveis ao contexto, dentre outros) é equivalente a engenharia de sistemas de software convencionais (web, cliente/servidor, dentre outros)? Quais são as similaridades? Quais as diferenças? Quais são seus desafios?

   Os chamados sistemas de software contemporâneos são essencialmente diferentes em suas propriedades. Esses sistemas deixaram de ser formados por blocos monolíticos sendo executados em um único ou poucos dispositivos e passaram a ser compostos por diferentes "coisas" que possuem sua própria inteligência e comportamento.

   Esses novos sistemas requerem o tratamento de propriedades adicionais não comumente encontradas nos sistemas de software tradicionais. Interoperabilidade intensa, descoberta de serviços, escalabilidade, disponibilidade, privacidade e segurança são algumas delas.

   Dessa forma, esse novo tipo de sistema de software requer abordagens de engenharia que possam contemplar sua construção e apoiando o desenvolvimento dessas novas propriedades.

   Referências

   Atzori, L, Iera, A and Morabito, G. "The Internet of Things: A Survey" The International Journal of Computer and Telecommunications Networking (2010)

   Belkeziz, R and Jarir, Z. "A Survey on Internet of Things Coordination" (2016)

   Top 50 IoT projets. Disponível em http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking/

   Design Principles for Industrie 4.0 Scenarios, Hawaii International Conference on System Sciences - Industrie 4.0: Hit or Hype?