Estudo para o teste TP: elementos de autodiagnóstico

Atualizado em: 17/06/2016 15:10.

Os tópicos apresentados servem como **elementos de autodiagnóstico** para a preparação do teste TP (não devem ser tomados como perguntas candidatas ou de exemplo).

# Modelos expressos em UML

* Interpretar/ler modelos expressos com diagramas de **Casos de Utilização** (DCaU).
* Dado um diagrama de Casos de Utilização, acrescentar ou corrigir elementos do modelo a partir de um texto com requisitos adicionais.
* Interpretar/ler modelos expressos com diagramas de **Atividades** (DAtiv).
* Dado um diagrama de Atividades, acrescentar ou corrigir elementos do modelo a partir de um texto com requisitos adicionais.
* Interpretar/ler modelos expressos com diagramas de **Classes** (DC).
* Dado um diagrama de Classes, acrescentar ou corrigir elementos do modelo a partir de um texto com requisitos adicionais.
* Interpretar/ler modelos expressos com diagramas de **Sequência** (DSeq).
* Dado um diagrama de Sequência, acrescentar ou corrigir elementos do modelo a partir de um texto com requisitos adicionais.
* Interpretar/ler modelos expressos com diagramas de **Instalação** (DInst).
* Dado um diagrama de Instalação, acrescentar ou corrigir elementos do modelo a partir de um texto com requisitos adicionais.
* Interpretar/ler modelos expressos com diagramas de **Componentes**.
* Interpretar/ler modelos expressos com diagramas de **Pacotes**.
* Interpretar/ler modelos expressos com diagramas de **Estado**.
* Recomendar e justificar, para diferentes situações de modelação, os diagramas UML mais adequados (quando usar cada diagrama).
* Exemplificar a aplicação de diagramas UML do mesmo tipo, mas com objetivos/abstrações diferentes, em diferentes momentos do processo de desenvolvimento.
* Relacionar diferentes diagramas com as categorias da vista “4+1”.
* Recomendar de forma fundamentada a utilização de modelos visuais na análise e desenvolvimento de sistemas de software.

# Disciplinas de análise e construção

## Análise de requistos

* Definir o conceito de domínio.
* Definir o conceito de sistema de informação.
* Descrever processos organizacionais (*workflows*) com modelos visuais.
* Identificar categorias (classes) e instâncias (objetos) num texto com descrição de requisitos de um domínio.
* Interpretar modelos dos conceitos do domínio (expressos com DC).
* Refinar um modelo de domínio dado, a partir de textos com requisitos adicionais em linguagem natural.
* Analisar criticamente uma especificação em texto estruturado de um caso de utilização (e.g.: nível de detalhe, foco na história da interação, etc.)
* Apresentar uma especificação em texto estruturado de um caso de utilização num modelo visual alternativo.
* Distinguir entre requisitos funcionais e requisitos não funcionais.
* Relacionar os diagramas da UML com as disciplinas da engenharia de requisitos (onde é que a UML pode ser usada com vantagens).
* Aplicar modelos da UML para especificar a integração/colaboração entre sistemas, descrita em linguagem natural.
* Caraterizar a complementaridade entre perspetivas de análise dos vários modelos da UML.

## Arquitetura e Desenho

* Interpretar e construir modelos estruturais de código em Java, expressos com DC.
* Interpretar e construir modelos dinâmicos de código em Java, expressos com DSeq.
* Explicar a utilização dos Diagramas de Sequência de Sistemas na transição da Análise para o Desenho.
* Caracterizar os princípios de coesão (*coesion*) e interligação (*coupling*) na avaliação de qualidade do desenho de código.
* Explicar o conceito de uma arquitetura de software por camadas.
* Distinguir entre arquitetura lógica e física e reconhecer modelos correspondentes.
* Interpretar e construir modelos de instalação para aplicações típicas dos projetos desenvolvidos em grupo (*web* e *mobile*).

# Processo de desenvolvimento

* Definir o conceito de modelo de processo de desenvolvimento de software.
* Caraterizar o trabalho dos papéis associados à aplicação de um processo (analista, arquiteto, programador, etc.)
* Caracterizar o modelo de processo de desenvolvimento sequencial (*Waterfall*).
* Caracterizar o modelo de processo de desenvolvimento ágil.
* Caraterizar o conceito de iteração e boas práticas associadas.
* Relacionar o modelo de desenvolvimento (sequencial ou iterativo) com o impacto na gestão de risco do projeto.
* Reconhecer e interpretar representações visuais do modelo de processo do Unified Process/OpenUP.
* Caraterizar as etapas preconizadas no Unified Process/OpenUP.
* Identificar os *milestones* a atingir em cada etapa do OpenUP.
* Caraterizar o princípio “desenvolvimento orientado por casos de utilização”.
* Caraterizar o princípio “desenvolvimento incremental” e “entrega de valor frequente (para o cliente)”.
* Justificar o OpenUP como um exemplo de um método ágil.