# Tópicos de estudo para o exame

Revisto em: 2022-01-28

## Conteúdos:

## Visão geral dos conteúdos da disciplina


A mensagem de MAS: perante o papel cada vez mais decisivo dos sistemas de software no processo de transformação digital das economias e da sociedade, coloca-se uma crescente exigência no processo de desenvolvimento (*software process*). Um dos pontos críticos é a correta determinação e gestão dos requisitos: não pode haver um produto de sucesso perante requisitos mal definidos. Há várias maneiras de abordar a definição de requisitos, com claras vantagens para as abordagens centradas na utilização, apresentado cenários de interação. Com uma clara visão das motivações dos utilizadores e *stakeholders*, a construção do software deve ser evolutiva, ao longo de vários ciclos, em que se constrói e entrega pacotes de funcionalidade relevantes para o promotor.

## A) O que é que está incluído no SDLC?

### O trabalho do Analista na equipa de desenvolvimento

Referências principais: *t.b.c.*

- Explicar o que é o ciclo de vida de desenvolvimento de sistemas (SDLC)
- Descrever as principais atividades/assuntos dentro de cada uma das quatro fases do SDLC
- Descrever o papel e as responsabilidades do Analista no SDLC
- Distinguir as competências de “análise de sistemas” das de “programação de sistemas”, em engenharia de software. Relacionar com os conceitos de “soft skills” e “hard skills”.

### O Unified Process/OpenUP

Referências principais: *t.b.c.*

- Descrever a estrutura do UP/OpenUP (fases e iterações)
- Descrever os objetivos de cada fase do UP/OpenUP
- Identificar as principais atividades de modelação/desenvolvimento associados a cada fase
- O OpenUP pode ser considerado “método ágil”?
- Porque é que o UP se assume como “orientado por casos de utilização, focado na arquitetura, iterativo e incremental”?

### Principais características dos métodos Ágeis

Principais referências: *t.b.c.*

- Identificar características distintivas dos processos sequenciais, como a abordagem *waterfall*.
- Identificar as práticas distintivas dos métodos ágeis (o que há de novo no modelo de processo, comparando com a abordagem "tradicional"?).
- Discuta o argumento: "A abordagem em cascata tende a mascarar os riscos reais de um projeto até que seja tarde demais para fazer algo significativo sobre eles."
- Discuta o argumento: "A abordagem ágil dispensa o planeamento do projeto".
- Identifique vantagens de estruturar um projeto em iterações, produzindo incrementos funcionais com frequência.
- Caracterizar os princípios da gestão do *backlog* em projetos ágeis.
- Dado um "princípio" (do *Agile Manifest*), explicá-lo por palavras próprias, destacando a sua novidade (com relação às abordagens "clássicas") e impacto / benefício.
- Apresentar situações em que, de facto, um método sequencial pode ser o mais adequado.

### O papel da modelação (visual)

Principais referências: *t.b.c.*

- Justifique o uso de modelos na engenharia de sistemas
- Descreva a diferença entre modelos funcionais, modelos estáticos e modelos de comportamento.
- Enumerar as vantagens dos modelos visuais.
- Explicar a organização da UML (classificação dos diagramas)
- Identificar os principais diagramas na UML e seu respetivo "ponto de vista" (perspetiva) de modelação.
- Ler e criar Diagramas de Atividades, Diagramas de Casos de Utilização, Diagramas de Classes, Diagramas de Sequência, Diagramas de Estado, Diagramas de Implementação[1], Diagramas de Pacotes[1] e Diagramas de Componentes[1].

## B) Compreender as necessidades do negócio (modelos e atividades da Análise)

### Práticas de engenharia de requisitos

Principais referências: *t.b.c.*

- Distinguir entre requisitos funcionais e não funcionais
- Distinguir entre abordagens centradas em cenários (utilização) e abordagens centradas no produto para a determinação de requisitos
- Identificar, numa lista, requisitos funcionais e atributos de qualidade.
- Justifique que "a determinação de requisitos é mais que a recolha de requisitos".
- Identifique requisitos bem e mal formulados (aplicando os critérios S.M.A.R.T.)
- Discutir as "verdades incontornáveis" apresentadas por Wiegers, sobre os requisitos de sistemas software [original, cópia disponível no material das TP].

### A modelação do contexto do problema: modelo do domínio/negócio

Principais referências: *t.b.c.*.
Caraterizar os conceitos do domínio de aplicação

[1] Estes diagramas foram apresentados, mas não foram aprofundados. Espera-se que os alunos tenham um conhecimento geral, sem precisar de aprofundar a sua aplicação.

- Desenhe um diagrama de classes simples para capturar os conceitos de um domínio de problema.
- Apresente duas estratégias para descobrir sistematicamente os conceitos candidatos para incluir no modelo de domínio.
- Identificar construções específicas (associadas à implementação) que podem poluir o modelo de domínio (na etapa de análise).

Caraterizar os processos do negócio/organizacionais

- Leia e desenhe diagramas de atividades para descrever os fluxos de trabalho da organização / negócios.
- Identifique o uso adequado de ações, fluxo de controle, fluxo de objetos, eventos e partições com relação a uma determinada descrição de um processo.
- Relacione os "conceitos da área do negócio" (classes no modelo de domínio) com fluxos de objetos nos modelos de atividade.

### Modelação funcional com casos de utilização

Principais referências: *t.b.c.*

- Descrever o processo usado para identificar casos de utilização.
- Ler e criar diagramas de casos de utilização.
- Rever modelos de casos de utilização existentes para detetar problemas semânticos e sintáticos.
- Descrever os elementos essenciais de uma especificação de caso de uso.
- Explicar o uso complementar de diagramas de casos de utilização, diagramas de atividades e narrativas de casos de utilização.
- Explicar o sentido da expressão "desenvolvimento orientado por casos de utilização".
- Explicar os seis "Princípios para a adoção de casos de utilização" propostos por Ivar Jacobson (com relação ao "Use Cases 2.0")
- Compreender a relação entre requisitos e casos de utilização
- Identificar as disciplinas e atividades relacionadas aos requisitos no OpenUP

### Modelação estrutural

Principais referências: *t.b.c.*.

- Distinguir entre a análise de sistemas baseada num abordagem algorítmica *top-down* e baseada nos conceitos do domínio do problema.
- Justifique o uso de modelos estruturais na especificação de sistemas.
- Explicar a relação entre os diagramas de classe e de objetos.
- Rever um modelo de classes quanto a problemas de sintaxe e semânticos, considerando uma descrição do um problema de aplicação.
- Descreva os tipos e funções das diferentes associações no diagrama de classes.
- Identifique o uso adequado da associação, composição e agregação para modelar a relação entre objetos.
- Identifique o uso adequado de classes de associação.

### Modelação de comportamento

Principais referências: *t.b.c.*.

- Explique o papel da modelagem de comportamento no SDLC
- Entenda as regras e diretrizes de estilo para diagramas de sequência, comunicação e estado
- Entenda a complementaridade entre diagramas de sequência e comunicação Diagramas de
- sequência de mapa em código orientado a objeto e reverso.
- Analise criticamente os modelos de diagramas de sequência existentes para descrever a cooperação entre dispositivos ou entidades de software.

## C) Modelos no desenho e implementação

### Vistas de arquitetura

Principais referências: *t.b.c.*.

- Explicar as atividades associadas ao desenvolvimento de arquitetura de software.
- Identifique os elementos abstratos de uma arquitetura de software
- Identifique as camadas e partições numa arquitetura de software por camadas.
- Explique a prática de "arquitetura evolutiva" proposta no OpenUp.
- ~~Rever criticamente um diagrama de pacotes existente para ilustrar uma arquitetura lógica~~
- ~~Rever criticamente um diagrama de componente existente para descrever as partes tangíveis do software~~
- ~~Analise criticamente um diagrama de implementação existente para descrever a instalação de um sistema~~

### Classes e desenho de métodos (perspetiva do programador)

Referências principais: *t.b.c.*

- Relacionar código por objetos com a sua representação em diagramas de classes da UML.
- Modelar a interação entre unidades de software (objectos) como diagramas de sequência.
- Construa um diagrama de classes e um diagrama de sequência considerando um código Java.
- Explicar as implicações no código da navegabilidade modelada no diagrama de classes.

## D) Práticas selecionadas na construção do software

### Garantia de qualidade

Principais referências: *t.b.c.*

- Identifique as atividades de validação e verificação incluídas no SDLC
- Descreva quais são as camadas da pirâmide de teste
- Descreva o assunto/objetivo dos testes de unidade, integração, sistema e de aceitação
- Explique o ciclo de vida do TDD
- Descreva as abordagens "debug-later" e "test-driven", de acordo com J. Grenning.
- Explique como é que as atividades de garantia de qualidade (QA) são inseridas no processo de desenvolvimento, numa abordagem clássica e nos métodos ágeis.
- O que é o "V-model"?
- Relacione os critérios de aceitação da história (*user-story*) com o teste *Agile*.
- Explique as práticas de CI/CD e a sua relevância para a implementação de uma abordagem ágil.

## E) Abordagens complementares

### Histórias (=*user stories*) e métodos ágeis

Principais referências: *t.b.c.*

- Defina histórias (*user stories*) e dê exemplos.
- Explique a metáfora do "*post-it*" (para planeamento e seguimento) comum em projetos ágeis.
- Identifique os elementos-chave de uma "persona".
- Compare histórias e casos de utilização em relação a pontos comuns e diferenças.
- Compare "Persona" com Ator com respeito a semelhanças e diferenças.

- O que é a pontuação de uma história e como é que é determinada?
- Descreva o conceito de velocidade da equipa (como usado no PivotalTracker e SCRUM).
- Explique a abordagem proposta por Jacobson em "Use Cases 2.0".
- Discuta se os casos de utilização e as histórias são abordagens redundantes ou complementares (quando seguir cada uma das abordagens? Em que condições? ...)

### O *framework* SCRUM

Principais referências: *t.b.c.*.

- Explique o objetivo da "Daily Scrum meeting"
- Relacione os conceitos de *sprint* e iteração e discuta a sua duração esperada.
- Explique a método de pontuação das histórias (e critérios aplicados)
- Relacione as práticas previstas no SCRUM e os princípios do "Agile Manifest": em que medida estão alinhados?

## Recursos adicionais

- Na página da disciplina (Moodle), está disponível um exemplo de teste de uma edição anterior.