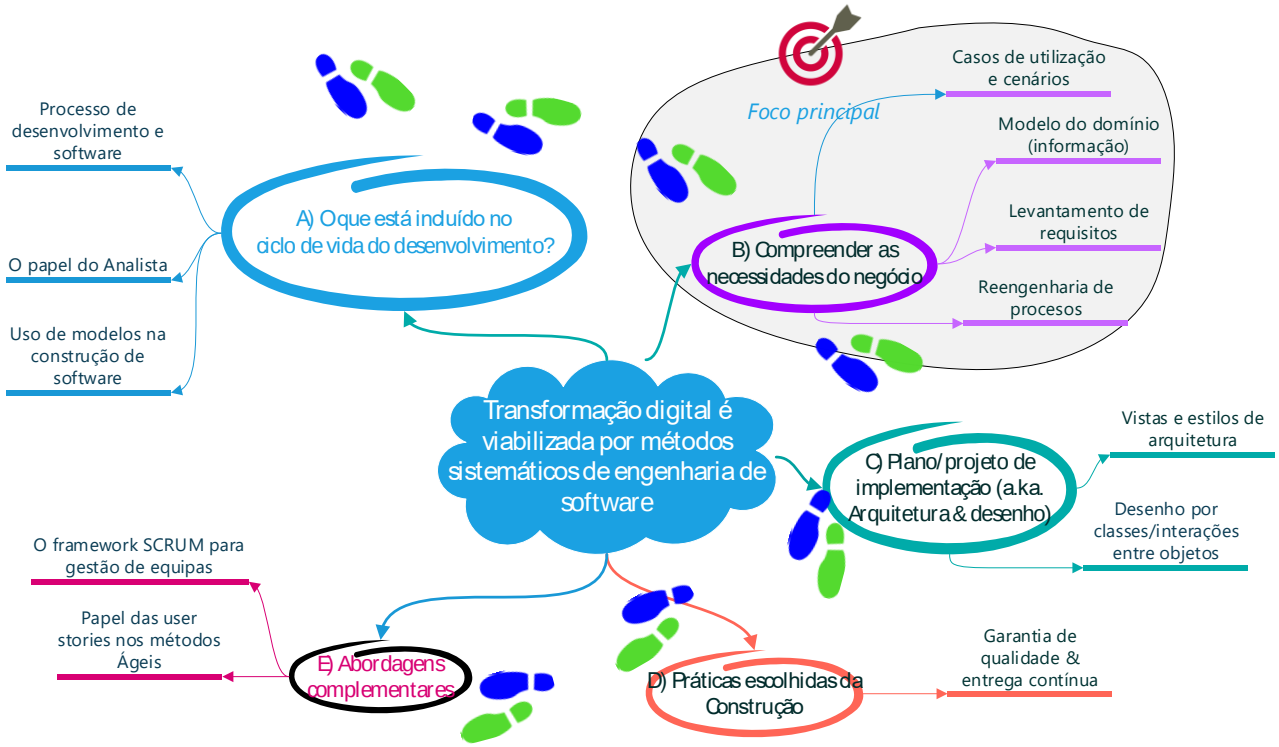
Tópicos de estudo para o exame

Revisto em: 2023-01-08

Conteúdos:

Visão geral dos conteúdos da disciplina **2**
A) O que é que está incluído no SDLC? **2**
O trabalho do Analista na equipa de desenvolvimento 2
Processo de software 2
O Unified Process/OpenUP 2
Principais características dos métodos Ágeis 3
O papel da modelação (visual) 3
B) Compreender as necessidades do negócio (modelos e atividades da Análise) **3**
Práticas de engenharia de requisitos 3
A modelação do contexto do problema: modelo do domínio/negócio 4
Modelação funcional com casos de utilização 4
Modelação estrutural 4
Modelação de comportamento 4
C) Modelos no desenho e implementação **5**
Vistas de arquitetura 5
Classes e desenho de métodos (perspetiva do programador) 5
D) Práticas selecionadas na construção do software **5**
Garantia de qualidade 5
E) Abordagens complementares **5**
Gestão do projeto: planeamento e monitorização do progresso 5
Histórias (=*user stories*) e métodos ágeis 6
O *framework* SCRUM 6

Visão geral dos conteúdos da disciplina



A mensagem principal de MAS está relacionada com o papel dos métodos de desenvolvimento de sistemas baseados em software: perante o papel cada vez mais decisivo dos sistemas de software no processo de transformação digital das economias e da sociedade, coloca-se cada vez maior 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 selecionados.

A) O que é que está incluído no SDLC?

O trabalho do Analista na equipa de desenvolvimento

Referências principais: *TP01, TP02*

- 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. Nota: há modelos que representam o SDLC com cinco etapas (incluindo a Manutenção).
- Descrever o papel e as responsabilidades do Analista no SDLC.

Processo de software

Referências principais: *TP02, TP09*

- Distinguir projetos sequenciais de projetos evolutivos.
- Identificar características distintivas dos processos sequenciais, como a abordagem *waterfall*.
- Discuta o argumento que "A abordagem sequencial tende a mascarar os riscos reais de um projeto até que seja tarde demais para fazer algo significativo sobre eles."
- Apresentar situações em que, de facto, um método sequencial pode ser adequado.

O Unified Process/OpenUP

Referências principais: *TP02, TP09*

- Descrever a estrutura das OpenUP (fases e iterações)

- Descrever os objetivos de cada fase
- Identificar as principais atividades de modelação/desenvolvimento associados a cada fase
- O OpenUP pode ser considerado "método ágil"?
- Porque é que o Unified Process se assume como "orientado por casos de utilização, focado na arquitetura, iterativo e incremental"?
- Distinguir o OpenUP da UML.

Principais características dos métodos Ágeis

Principais referências: TP09, *TP10*

- Identificar as práticas distintivas dos métodos ágeis (o que há de novo no modelo de processo, comparando com a abordagem "tradicional"?).
- Identifique vantagens de estruturar um projeto em iterações, produzindo incrementos 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.

O papel da modelação (visual)

Principais referências: *TP01*

- Justifique o uso de modelos na engenharia de sistemas. Como é que os modelos ajudam a gerir a complexidade, segundo G. Booch?
- Descreva a diferença entre modelos funcionais, modelos estáticos e modelos de comportamento.
- Enumerar as vantagens dos modelos visuais, como com a utilização da UML.
- Explicar a organização da UML (classificação dos diagramas)
- Identificar os principais diagramas na UML e seu respetivo "ponto de vista" (perspetiva) de modelação
- Interpretar (e criar) Diagramas de Atividades, Diagramas de Casos de Utilização, Diagramas de Classes, Diagramas de Sequência, Diagramas de Estado, Diagramas de Implementação, Diagramas de Pacotes e Diagramas de Componentes.
- Explicar o sentido da expressão "é importante procurar a simplicidade dos modelos se queremos criar diagramas que vão realmente ajudar à otimização dos processos."

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

Práticas de engenharia de requisitos

Principais referências: TP02, TP03

- Definir o conceito de requisito.
- Definir o conceito de regra de negócio e relacioná-lo com o de requisito.
- Distinguir entre requisitos funcionais e não funcionais
- Distinguir entre abordagens centradas em cenários (de utilização) e abordagens centradas no produto (listas de capacidades) na determinação de requisitos.
- Identificar, numa lista, requisitos funcionais e atributos de qualidade (especialmente para as categorias fURPS).
- 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.)
- Explique o sentido da expressão "requisitos em contexto" associada à técnica dos Casos de

Utilização.

- Porque é que a "a determinação de requisitos, e muito do trabalho dos projetos de software e de sistemas em geral, é principalmente um desafio de interação humana." (K. Wiegers)

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

Principais referências: *t.b.c.*
Caraterizar os conceitos do domínio de 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: *TP02, TP03.*

- 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: *TP05,*

- 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: *TP06*

- 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 de comunicação
- 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: *TP11*

- ~~Explicar as atividades associadas ao desenvolvimento de arquitetura de software.~~
- Identifique os três tipos de estruturas abstratas de uma arquitetura de software estudadas.
- Identifique as camadas e partições numa arquitetura de software por camadas. Explicar o "padrão" da organização por camadas.
- Explique a prática de "arquitetura evolutiva" proposta no OpenUp.
- Interpretar diagramas relacionados com vistas de arquitetura e o respetivo "ponto de vista" do modelo (diagrama de pacotes, diagrama de componentes, diagrama de implementação)
- Referir/identificar exemplos de requisitos com e sem impacto nas decisões de arquitetura.
- Referir/identificar algumas das **razões** pelas quais se deve investir na definição de uma arquitetura, cedo no desenvolvimento do projeto.

Classes e desenho de métodos (perspetiva do programador)

Referências principais: *TP05, TP07*

- ~~Explicar os princípios de baixo acoplamento e alta coesão no desenho por objetos.~~
- ~~Enumerar/identificar situações de código que implicam acoplamento entre classes.~~
- Relacionar o diagrama de classes/ diagrama de sequência com as construções visíveis em código Java.

D) Práticas selecionadas na construção do software

Garantia de qualidade

Principais referências: *TP12*

- 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

Gestão do projeto: planeamento e monitorização do progresso

Principais referências: *TP08*

- A que nos referimos com a expressão Promotor (*sponsor*) do projeto?
- Quais as três dimensões incluídas na avaliação de viabilidade de um projeto?
- O que é um *Workpackage Breakdown Structure* (WBS)?
- Identifique algumas caraterísticas essenciais de um quadro Kanban (*Kanban board*).
- Explique o princípio de "timeboxing" aplicado ao desenvolvimento de um projeto por incrementos.

Histórias (=*user stories*) e métodos ágeis

Principais referências: *TP10*

- 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).
- Discutir 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: *TP13*

- 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?
- O SCRUM é um processo de software? Os métodos ágeis recorrem à SCRUM?