Versão relativa à edição de 2023.

(sujeito a alterações para 2023/24)

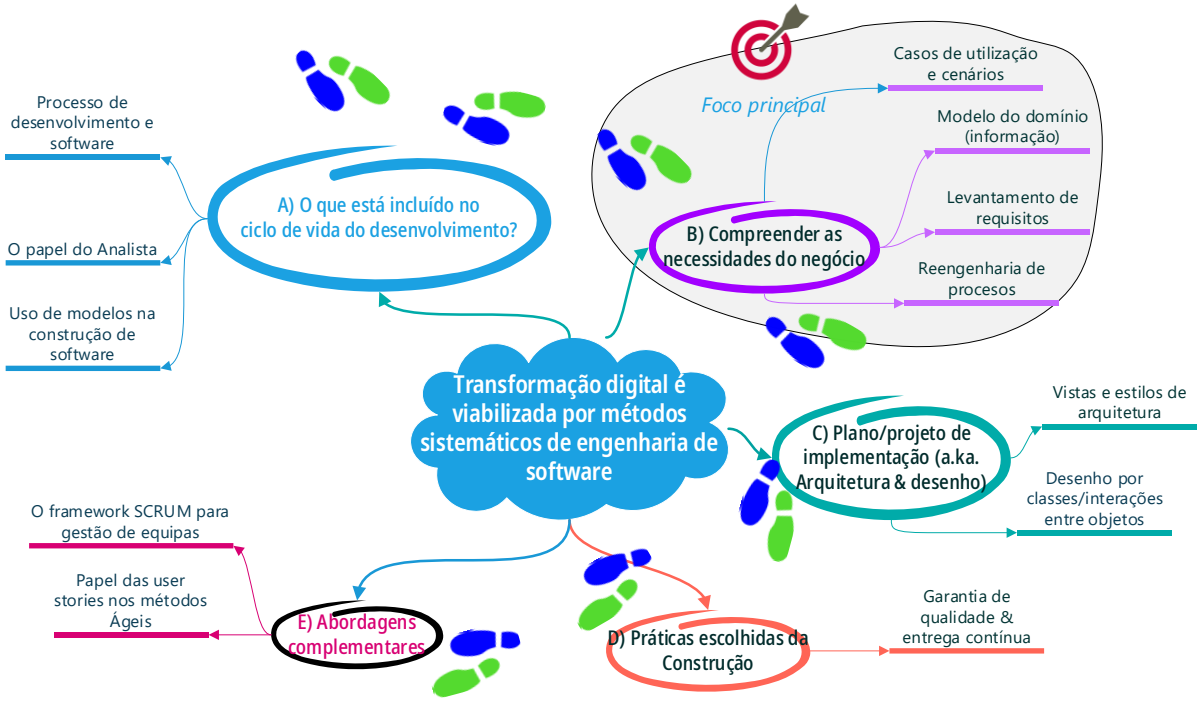
Tópicos de estudo para o exame

Utilizar este documento de apoio como uma revisão dos assuntos a serem estudados. Resulta essencialmente de compilar os objetivos de aprendizagem que já constam nas apresentações das aulas TP.

** DRAFT ** Revisto em: 2023-06-06

Conteúdos:

Visão geral dos conteúdos da disciplina



A mensagem central de AS: 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 (o *software process*). Um dos pontos críticos é a correta determinação 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, através de análise de cenários (de interação). Com uma visão clara das motivações dos utilizadores e *stakeholders*, a construção do software deve ser evolutiva, ao longo de vários ciclos (incrementos), em que se constrói e entrega pacotes de funcionalidade relevantes para o promotor/cliente. No desenvolvimento incremental, podemos procurar um equilíbrio entre as técnicas de modelação mais abrangentes (e.g.: use cases) e as técnicas de especificação mais leves (e.g.: *user stories*).
Os modelos (e.g.: construídos na UML) são uma ferramenta para facilitar a comunicação e a colaboração na equipa, usados de forma transversal no *software process.*

A) O que é que está incluído no SDLC?

O SDLC e o trabalho do Analista

- Explicar o que é o ciclo de vida de desenvolvimento de sistemas (SDLC)
- Descrever as principais atividades/assuntos dentro de cada uma das fases do SDLC (há autores que incluem 4 fases no SDLC, outros que incluem 5 fases com a Manutenção).
- Definir o temo "processo de software" (*software process*)
- Distinguir atividades de análise do domínio (de aplicação) de atividades de especificação do software.
- 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".

Processo de software e o Unified Process/OpenUP

- Descrever a estrutura do UP/OpenUP (fases e objetivos; iterações)

- Descrever os objetivos e principais atividades de cada fase do UP/OpenUP
- 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"?
- 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"?).
- Distinguir projetos (de desenvolvimento de software) sequenciais de projetos evolutivos.

Modelação visual e a UML

- 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)
- Caraterizar o "ponto de vista" (perspetiva) de modelação de cada diagrama da UML usado nas aulas Práticas.
- Relacionar os diagramas UML com o momento em que são aplicados, ao longo do projeto de desenvolvimento.
- Identificar os elementos comuns (dos diagramas) da UML e exemplificar a sua utilização.
- Interpretar 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].

  Este tópico engloba, naturalmente, o domínio da semântica dos elementos de modelação e suas relações, nos diagramas referidos.

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

Práticas de engenharia de requisitos

- Distinguir entre requisitos funcionais e não funcionais
- Apresentar técnicas de recolha de requisitos e recomendá-las para diferentes tipos de projeto.
- 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.)
- Identifique requisitos bem e mal formulados (aplicando os critérios do ISO-IEEE 29148)
- Discutir as "verdades incontornáveis" apresentadas por Wiegers, sobre os requisitos de sistemas software [original, cópia disponível no material das TP].
- Identificar/exemplificar regras de negócio (distinguindo-as do conceito de requisitos).
- Qual a abordagem proposta no OpenUp para a documentação de requisitos de um produto de software (*outcomes* relacionados)?
- Comentar a afirmação "o processo de determinação de requisitos (*requirements elicitation*) é primeiramente um desafio de interação humana".

[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.~~ e devem ser estudados.

Modelação funcional com casos de utilização

- 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”)
- Explicar a relação entre requisitos e os casos de utilização
- Identificar as disciplinas e atividades relacionadas aos requisitos no OpenUP
- Relacionar o caso de utilização (entidade de modelação) com os cenários (formas de percorrer o caso de uso).
- objetos.
- Identifique o uso adequado de classes de associação.

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

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.
- Como é que o empacotamento dos casos de uso (resultado da análise), pode contribuir para a identificação de potenciais módulos, na arquitetura?

C) Modelos no desenvolvimento

Orientação aos objetos no SDLC

- Discutir as diferenças e complementaridades entre os conceitos de orientação aos objetos na Análise (OOA), no Desenho (OOD) e na programação (OOP).
- Apresentar a distribuição de responsabilidades e a colaboração mecanismos de base na orientação aos objetos.

Modelação estrutural

- Distinguir entre a análise de sistemas baseada numa abordagem algorítmica *top-down* e baseada nos conceitos do domínio do problema.

- 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

Modelos de comportamento

- Explique o papel da modelação de comportamento no SDLC
- Explicar a complementaridade entre diagramas de sequência e de comunicação
- Relacionar a ideia essencial (na orientação aos objetos) de distribuição de responsabilidades com o diagrama de sequência (como é que ajuda?).
- Relacionar representações nos diagramas de sequência com código por objetos e vice-versa.
- Representar o ciclo de vida de uma entidade num diagrama de estados.
- Relacionar elementos presentes num D. Sequência com as entidades de um D. de Classes.
- Como é que o desenvolvimento de um modelo de colaboração entre objetos pode ser conduzido a partir dos casos de utilização? (*use case realizations*)

Vistas de arquitetura

- Explicar as atividades associadas ao desenvolvimento de arquitetura de software, no openUP.
- Explicar a relação entre requisitos e a arquitetura (como é que aqueles influenciam esta). Exemplificar requisitos com impacto na arquitetura.
- Explique a prática de "arquitetura evolutiva" proposta no OpenUp.
- Identifique as camadas e partições numa arquitetura de software por camadas.
- Usar diagramas de sequência para descrever a cooperação entre módulos/elementos de uma arquitetura.
- Identifique os três tipos de estruturas principais, constituintes de uma arquitetura (segundo L. Bass *et al*). Relacionar essas categorias com os diagramas de UML mais relevantes para as documentar.
- Discutir razões técnicas e não técnicas que justificam o desenvolvimento de uma (boa) arquitetura para o sistema a construir.

Desenho do software (perspetiva do programador)

- Explicar como os casos de utilização podem ser usados para orientar as atividades de desenho (na perspetiva do livro do Larman).
- Explicar os princípios do baixo acoplamento e alta coesão em OO. Discutir implicações do *coupling* e *cohesion*.
- Comparar, num dado desenho por objetos, a ocorrência de maior/menor *coupling* e *cohesion*.
- Relacionar código por objetos com a sua representação em diagramas de classes da UML.
- Modelar a interação entre unidades de software (objetos) 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

- 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

Escrever o Teste (Red), Fazer o teste passar (Green), refatorar o código (Refactor)

- 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 o sentido da afirmação “especificações executáveis”.
- Justifique a necessidade de testes “*developer facing*” e “*custommer facing*”.

Integração contínua/Entrega Contínua

- Identificar os passos típicos de um ciclo de CI
- Distinguir entre C. Integration, C. Deployment e C. Delivery
- Relacionar o CI/CD com a natureza iterativa e incremental dos métodos ágeis de desenvolvimento, ou seja, como é que o CI/CD ajuda na concretização de metodologias incrementais?
- Explicar as tarefas e *outcomes* englobados numa “*build*”
- Explique o sentido da prática “*continuous testing*”.

E) Práticas dos métodos ágeis

Principais características dos métodos ágeis de desenvolvimento

- 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.”
- Explique o sentido da frase: “O desenvolvimento iterativo centra-se em ciclos curtos e orientados para a geração de valor”
- 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.
- Quais os objetivos das organizações com a adoção de metodologias de desenvolvimento “iterativas e incrementais”?

Histórias (=*user stories*) e métodos ágeis

- Defina histórias (*user stories* - *US*) e dê exemplos.
- Como é que o conceito de US se relaciona [ou não] com o de requisito (da análise)?
- Compare histórias e casos de utilização em relação a pontos comuns e diferenças. Em que medida podem ser usados de forma complementar? histórias são simples, CaU mais complexo (UML)
- Compare “Persona” com Ator com respeito a semelhanças e diferenças. Qual a utilidade das Personas no desenvolvimento das US?
- 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” para combinar a técnica dos *Use cases* com a flexibilidade das *user stories*.
- Discuta se os casos de utilização e as histórias são abordagens redundantes ou complementares (quando seguir cada uma das abordagens? Em que condições? ...)
- Exemplifique a definição de critérios de aceitação para uma US.

- Como é que um *story map* organiza espacialmente o *backlog*, ou seja, como se usa o *story map* no contexto dos métodos ágeis?

O *framework* SCRUM

- 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)
- Identificar os papéis numa equipa de Scrum e as principais "cerimónias”
- Relacione as práticas previstas no SCRUM e os princípios do “Agile Manifest”: em que medida estão alinhados?
- Pode-se considerar o SCRUM como a “*silver bullet*” (solução universal) para o sucesso dos projetos de desenvolvimento?
- Identifique alguns desafios/bloqueios à implementação eficaz do SCRUM.