

Informática e Tecnologias Multimédia

Licenciatura, 1º Ciclo

Plano: Despacho n.º 9184/2020 - 25/09/2020

Ficha da Unidade Curricular: Metodologia de Desenvolvimento de Software

ECTS: 6; Horas - Totais: 162.0, Contacto e Tipologia, TP:28.0; PL:42.0; OT:5.0;

Ano | Semestre: 3 | S1

Tipo: Obrigatória; Interação: Presencial; Código: 814332

Área Científica: Informática

Docente Responsável

Fernando Sérgio Hortas Rodrigues

Professor Adjunto

Docente(s)

Fernando Sérgio Hortas Rodrigues

Professor Adjunto

Objetivos de Aprendizagem

Permitir ao aluno apreenderem o que é e para que serve a Engenharia de Software.

Introduzir os princípios metodológicos ágeis, com recurso à metodologia XP e à framework SCRUM.

Transmitir os conceitos de Análise e Desenho Orientados a Objetos, através da metodologia Unified Process.

Objetivos de Aprendizagem (detalhado)

Com esta unidade curricular pretende-se que o aluno apreenda os objetos de estudo Engenharia de Software e em particular aprenda a efetuar a Análise e Desenho Orientados por Objetos, de um sistema. O aluno deve tornar-se capaz de:

1. Explicar o que é e para que serve a Engenharia de Software;
2. Explicar o que aproxima e distingue metodologias ágeis e tradicionais;
3. Conhecer e descrever as características básicas da framework SCRUM;
4. Conhecer e descrever as características básicas da metodologia de desenvolvimento Extreme

Programming;

5. Conhecer e explicar os principais conceitos e princípios envolvidos no desenvolvimento de software;
6. Utilizar a linguagem de modelação visual UML, na modelação de um sistema;
7. Utilizar a metodologia de desenvolvimento de software Unified Process, no desenvolvimento de sistemas;
8. A efetuar a Análise e o Desenho orientados por Objetos, de uma aplicação;
9. Conhecer e aplicar padrões de desenho elementares, na Análise e Desenho de Sistemas;
10. Saber o que são e para que servem as práticas Devops;
11. Saber o que é e como se pode implementar uma Pipeline CI/CD.

Conteúdos Programáticos

Metodologias ágeis, XP e SCRUM; Metodologias tradicionais; UML - Unified Modeling Language; Análise e Desenho Orientados a Objetos; UP - Unified Process; Cultura Devops; CI/CD Pipeline - Integração Contínua/Entrega Contínua.

Conteúdos Programáticos (detalhado)

1. Categorias de Metodologias de Desenvolvimento de Software
 - 1.1. Categorias e características das metodologias de desenvolvimento de software
 - 1.2. Visão geral de metodologias de desenvolvimento de software populares: Waterfall, Scrum, Extreme Programming, Unified Process.
2. A UML - Unified Modeling Language
 - 2.1. Princípios da Modelação
 - 2.2. Perspetivas de Arquiteturas orientadas a objetos
 - 2.3. Formas de utilização da UML
 - 2.4. Modelo conceptual da UML
 - 2.4.1. Blocos Básicos
 - 2.4.1.1. Entidades: Estruturais; De comportamento; Agregadoras; De notação
 - 2.4.1.2. Relações: De dependência; De associação; De generalização; De realização
 - 2.4.1.3. Diagramas
 - 2.4.2. Regras
 - 2.4.3. Mecanismos Comuns
 - 2.4.3.1. Especificações
 - 2.4.3.2. Adornos
 - 2.4.3.3. Divisões comuns
 - 2.4.3.4. Mecanismos de extensibilidade
 - 2.5. Os diagramas da UML
3. Desenho e Análise Orientados por Objetos
 - 3.1. Arquitetura do Sistema
 - 3.2. Perspetivas e características de uma Arquitetura Orientada a Objetos
 - 3.3. O que é a Análise e o Desenho
 - 3.4. Análise e Desenho orientados por objetos

- 4. Processos Iterativos, Evolutivos e Ágeis
 - 4.1. Características e vantagens do desenvolvimento Iterativo e Evolutivo
 - 4.2. A mudança num processo iterativo
 - 4.3. Timeboxing
 - 4.4. O Processo Unificado (UP)
 - 4.4.1. Fases, Iterações e Milestones
 - 4.4.2. Disciplinas
- 5. Fase de Conceção (Inception)
 - 5.1. Características
 - 5.2. Duração
 - 5.3. Artefactos que se iniciam nesta fase
- 6. Requisitos Evolutivos
 - 6.1. O que são Requisitos
 - 6.2. Categorias de Requisitos
 - 6.3. Requisitos Evolutivos vs. Requisitos do tipo Waterfall
 - 6.4. Como encontrar requisitos
- 7. Casos de Uso (UCs)
 - 7.1. Atores, Cenários e Casos de Uso
 - 7.2. Formatos de UCs (Resumido, Casual, Detalhado)
 - 7.3. Evolução dos UCs nas várias iterações
- 8. Fase de Elaboração (Elaboration)
 - 8.1. As várias iterações da fase de Elaboração
 - 8.2. Os UCs durante as várias iterações
 - 8.3. Artefactos iniciados na fase de elaboração
- 9. Modelos de Domínio
 - 9.1. O que são e para que servem
 - 9.2. Classes conceptuais
 - 9.3. Métodos para encontrar classes conceptuais
 - 9.4. Atributos
 - 9.5. Atributos vs Classes
 - 9.6. Modelar com Classes de Descrição
 - 9.7. Associações
 - 9.8. Métodos para encontrar associações
- 10. Diagramas de Sequência do Sistema (DSSs)
 - 10.1. O que são e para que servem
 - 10.2. Relação entre DSSs e Use Cases
 - 10.3. Nomeação de Eventos de Sistema e Operações
 - 10.4. Contractos de Operação e suas características
 - 10.5. Como criar e escrever Contratos de Operação
 - 10.6. Os contratos de operação no contexto do UP
- 11. Requisitos para o Desenho

11.1. Motivação para a passagem para as atividades de desenho

12. Diagramas de Interação

12.1. Notação dos Diagramas de Sequência e Comunicação

12.2. Notação dos Diagramas de Interação

13. Diagramas de Classe

13.1. O que são e para que servem

13.2. Notação dos diagramas de classe

13.3. Diagramas de Classes de Desenho

13.4. Classificadores

13.5. Como mostrar atributos no diagrama de classes

13.6. Anotações

13.7. Operações e Métodos

13.8. Estereótipos, Profiles e Tags

13.9. Propriedades

13.10. Generalização, classes e métodos abstratos

13.11. Dependências

13.12. Interfaces

13.13. Composição e Agregação

13.14. Constraints

13.15. Associação qualificada

13.16. Classe de associação

13.17. Classe Singleton

13.18. Compartimentos definidos pelo utilizador

13.19. Relação entre diagramas de interação e digramas de classes

14. GRASP - General Responsibility Assignment Software Patterns

14.1. Desenho de objetos com responsabilidades

14.2. UML vs. Princípios de desenho

14.3. Desenho de objetos: Entradas, Atividades e Saídas

14.4. Responsabilidades e Orientação por Responsabilidade no desenho

14.5. Metodologia GRASP para desenho OO

14.6. Relações entre GRASP, Responsabilidades e UML

14.7. Padrões

14.7.1. O que são e para que servem

14.7.2. Aplicação do GRASP ao desenho de objetos

14.7.3. Creator

14.7.4. Expert

14.7.5. Low Coupling

14.7.6. Controller

14.7.7. High Cohesion

15. Tendências atuais na Engenharia de Software

15.1 A cultura Devops

15.2 O pipeline CI/CD

15.2.1 Integração Contínua

15.2.2 Entrega Contínua (vs. Disponibilização Contínua)

Metodologias de avaliação

Avaliação em Época de Frequência:

- Teste escrito (30%)
- Trabalho Prático (70%).

O aluno obtém aprovação à UC, estando dispensado de Exame, de acordo com o disposto nos Pontos 11 e 12, do Artigo 11º, do regulamento Académico do IPT.

Avaliação em Época Normal de Exame:

- Nota do Teste em época de frequência ou nota de Exame escrito (30%)
- Trabalho Prático (70%).

O aluno obtém aprovação à UC, de acordo com o disposto nos Pontos 11 e 12, do Artigo 11º, do regulamento Académico do IPT.

Avaliação em Época de Recurso e outras épocas:

- Exame escrito (30%)
- Trabalho Prático (70%).

O aluno obtém aprovação à UC, de acordo com o disposto nos Pontos 11 e 12, do Artigo 11º, do regulamento Académico do IPT.

Observação 1: Nota mínima Teste e Exames escritos: 7 valores; Nota mínima Trabalho Prático: 10 valores.

A Falta a uma componente de avaliação, em qualquer época, corresponde à atribuição de 0 (zero) valores nessa componente.

Observação 2: Se o docente entender, pode convocar os alunos para discussão dos trabalhos. A falta à discussão resulta na atribuição de zero valores nessa componente e por conseguinte a reprovação na época de avaliação em causa.

Software utilizado em aula

Visual Paradigm

Estágio

Não aplicável.

Bibliografia recomendada

- Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd, Prentice Hall. -
- Booch, G. e Rumbaugh, J. e Jacobson, I. (2005). *The Unified Modeling Language User Guide*.

Coerência dos conteúdos programáticos com os objetivos

No Cap. 1 são tipificadas as várias MDS e são descritas genericamente as metodologias XP e SCRUM. (Objetivos 1 a 5)

No Cap. 2 é apresentada a UML, em detalhe. (Objetivo 6)

No Cap. 3 são apresentados os conceitos gerais de Análise e Desenho OO. (Objetivos 7 e 8)

Do Cap. 4 ao Cap. 13 é apresentada a metodologia UP acompanhada de exemplos concretos. (Objetivos 7 e 8)

No Cap. 14 são introduzidos Padrões de Software GRASP elementares. (Objetivo 9)

No Cap. 15 são apresentados as práticas Devops e a pipeline CI/CD. (Objetivos 10 e 11)

Metodologias de ensino

Aulas teórico-práticas expositivas onde se descrevem os conceitos fundamentais. Aulas práticas de resolução de casos práticos e aplicação dos conceitos a cenários de utilização real.

Coerência das metodologias de ensino com os objetivos

Aulas teórico-práticas expositivas onde se descrevem os conceitos fundamentais. Aulas práticas de resolução de casos práticos e aplicação dos conceitos a cenários de utilização real.

Língua de ensino

Português

Pré-requisitos

Bons conhecimentos de uma linguagem orientada a objetos.

Programas Opcionais recomendados

Não aplicável.

Observações

Objetivos de Desenvolvimento Sustentável:

- 4 - Garantir o acesso à educação inclusiva, de qualidade e equitativa, e promover oportunidades de aprendizagem ao longo da vida para todos;
- 9 - Construir infraestruturas resilientes, promover a industrialização inclusiva e sustentável e fomentar a inovação;
-

Docente responsável
