IPG Politécnico da Guarda Escola Superior de Tecnologia e Gestão	**GUIA DE FUNCIONAMENTO DA UNIDADE CURRICULAR**	MODELO PED.008.02

Curso	**Engenharia Informática**			*Ano letivo*			2018/2019
Unidade Curricular	**Engenharia de Software II**			ECTS			6
Regime	**Obrigatório**						
Ano	**3º**	*Semestre*	**1º sem**	*Horas de trabalho globais*			
Docente	**Maria Clara Silveira**			*Total*	168	*Contacto*	75
Coordenador da área disciplinar	**José Carlos Fonseca**						

GFUC previsto

1. OBJETIVOS DE APRENDIZAGEM

Após a conclusão da UC, os alunos deverão ser capazes de:

1. Projetar, executar e gerir o ciclo de vida do desenvolvimento de um sistema de software recorrendo a diferentes abordagens;
2. Elaborar a documentação técnica de um projeto usando a linguagem UML (*Unified Modeling Language*) e recorrendo a ferramentas CASE;
3. Aplicar padrões de casos de uso no levantamento e análise de requisitos;
4. Validar as funcionalidades do sistema face aos requisitos especificados.

2. CONTEÚDOS PROGRAMÁTICOS

1. Engenharia de Software e a problemática do desenvolvimento de software. Boas práticas no desenvolvimento de software.
2. O processo de desenvolvimento de software: ciclo de vida de software na perspetiva orientada para objetos: *Rational Unified Process*; MDA -*Model-Driven Architecture*: definição, abordagem, ferramentas, casos de sucesso; Melhoria do processo: CMMI - *Capability Maturity Model Integration*.
3. Especificação de requisitos de software. Gestão de requisitos. Modelo em espiral para a engenharia de requisitos.
4. Padrões de Software: introdução ao conceito de padrão; estudo e aplicação de padrões para casos de uso.

5. UML - *Unified Modeling Language*: casos de uso, modelação da estrutura, do comportamento e da arquitetura.
6. Ciclo de vida de utilização de Ferramentas CASE. Integração de ferramentas.
7. Métricas no processo de desenvolvimento de Software.
8. Verificação e validação do software.

3. DEMONSTRAÇÃO DA COERÊNCIA DOS CONTEÚDOS PROGRAMÁTICOS COM OS OBJETIVOS DA UC

1. Os Conteúdos 1, 2, 3, 4, 5, 6, 7 e 8 estão coerentes com o Objetivo 1, pois focam as diferentes abordagens, nomeadamente *Rational Unified Process*, *Model-Driven Architecture*, e CMMI.
2. Os Conteúdos 3, 4, 5 e 6 estão coerentes com o Objetivo 2, pois são apresentados as diversas formas de documentar os requisitos, lecionada a linguagem UML, bem como as ferramentas necessárias.
3. Os Conteúdos 3 e 4 estão coerentes com o Objetivo 3, dado que se ministram os padrões para especificação de requisitos na abordagem "*Patterns for Effective Use Cases".*
4. O Conteúdo 8 está coerente com o Objetivo 4, dado que é usado o Modelo em V que faz o paralelismo entre as entregas do processo de desenvolvimento de software e as entregas do processo de testes.

4. BIBLIOGRAFIA PRINCIPAL

Obrigatória:

1. Textos de apoio e diapositivos das aulas fornecidos pelo docente e disponibilizados na plataforma de e-learning.
2. Adolph, Steve; Bramble, Paul. Patterns for Effective Use Cases, Addison-Wesley Pearson Education, 2003.

3. Nunes, Mauro; O'Neill, Henrique. Fundamental de UML, 5ª Ed., FCA Editora, 2007.
4. Sommerville, Ian. Software Engineering (10th edition). Addison-Wesley Pearson Education, 2015 (Na biblioteca do IPG existe até à 8ª edição).

Recomendada:

5. Booch, Grady; Jacobson, Ivar; Rumbaugh, James. The Unified Modeling Language User Guide; Addison –Wesly; 1999.
6. Jacobson, Ivar; Booch, Grady; Rumbaugh, James; The Unified Software Development Process; Addison –Wesly; 1999.
7. ONeil, H., Nunes, M., Ramos, P. Exercícios de UML, FCA, 2010.
8. Sommerville, Ian. Software Engineering (10th edition). Slides disponíveis em: http://iansommerville.com/software-engineering-book/slides/.
9. Unified Modeling Language™, Resource Page, www.uml.org/.
10. Ivar Jacobson, Bud Lawson, Paul McMahon, Michael Goedicke; Software Engineering Essentialized, www.software-engineering-essentialized.com/; 2017.

5. METODOLOGIAS DE ENSINO (REGRAS DE AVALIAÇÃO)

Metodologias de ensino:

1. Lição expositiva
2. Trabalho de grupo
3. Trabalho de projeto
4. Estudo de casos
5. Resolução de problemas

Regras de avaliação:

Avaliação contínua: O estudante terá um bónus de um valor relativo à presença nas aulas.

1. Trabalho de projeto (em grupo): modelação em UML e protótipo (em conjunto com Programação para a Internet): **10 valores**. O acompanhamento e entregas parciais dos trabalhos é realizado nas aulas de tutoria. Todos os documentos entregues devem ficar alojados na plataforma GitHub.

    a. Entrega intermédia (30%) em que na avaliação são tidas em conta todas as entregas parciais até esta data**.**

    b. Entrega final (70%) com apresentação obrigatória. Inclui o protótipo realizado em conjunto com Programação para a Internet (na avaliação são tidas em conta todas as entregas parciais ao longo do semestre).

2. Prova de frequência: **10 valores** (nota mínima de 6 em 20 valores).

3. A avaliação contínua será feita com base em vários trabalhos, que incluem o desenvolvimento de módulos de um projeto (aplicação web) na UC Programação para a Internet. Para tal, cada aluno será integrado num projeto definido em comum nas unidades curriculares de Engenharia de Software II e de Programação para a Internet. A apresentação dos trabalhos será feita em conjunto para as duas unidades curriculares já referidas. A avaliação em cada uma das unidades curriculares é feita de forma independente, com base no trabalho entregue e na informação obtida nas tutorias, onde é feita a orientação, o controlo, supervisão e a avaliação do trabalho desenvolvido e do seu progresso. Os alunos que já aprovaram à UC Programação para a Internet, terão que desenvolver e entregar um protótipo da aplicação (plataforma de desenvolvimento à escolha do aluno e aprovada pelo docente).

4. Trabalhadores estudantes sem obrigatoriedade de comparecer às aulas, mas têm que apresentar, em horário a combinar com o docente, trabalhos parciais todas as semanas.

Avaliação por exame final na Época Normal, Época de Recurso, Época Especial:

1. Prova parte 1: 10 valores (mínimo de 6 em 20 valores). O aluno poderá ficar dispensado da primeira parte se obteve nota superior à mínima.

2. Parte 2 (prática): 10 valores (mínimo de 6 em 20 valores). O aluno poderá ficar dispensado da segunda parte se entregou e apresentou os trabalhos.

6. DEMONSTRAÇÃO DA COERÊNCIA DAS METODOLOGIAS DE ENSINO COM OS OBJETIVOS DA UNIDADE CURRICULAR

1. Lição expositiva está coerente com os objetivos devido à necessidade de apresentar os conteúdos teóricos aos estudantes, nomeadamente os vários modelos de processo, os padrões, as métricas e os níveis de maturidade

2. Trabalho de grupo está coerente com os objetivos visto que projetar, executar e documentar todo o processo de desenvolvimento de software necessita da colaboração e interação de elementos com diferentes conhecimentos e eventualmente diferentes personalidades

3. Trabalho de projeto está coerente com os objetivos visto que o trabalho final abrange todas as etapas do processo de desenvolvimento de software, pelo que obriga à aplicação de todos os conceitos abordados ao longo do semestre a um caso prático

4. Estudo de casos permite analisar documentação de outros projetos, analisar casos de sucesso de empresas que usam determinadas abordagens (por exemplo MDA), conhecer empresas certificadas CMMI, entre outros

5. Resolução de problemas está coerente com os objetivos pois a aplicação de conteúdos teóricos a exercícios práticos de inspiração realista, relacionados com os conteúdos (modelar sistemas, aplicar padrões, desenhar testes) ajuda a consolidar a matéria, realçando o saber fazer.