



UNIVERSIDADE FEDERAL DE OURO PRETO
DEPARTAMENTO DE COMPUTAÇÃO

PLANO DE ENSINO



Nome do Componente Curricular em português: Engenharia de Software I		Código: BCC322
Nome do Componente Curricular em inglês: Software Engineering I		
Nome e sigla do departamento: Departamento de Computação (DECOM)		Unidade acadêmica: ICEB
Nome do docente: Tiago Garcia de Senna Carneiro		
Carga horária semestral: 60 horas	Carga horária semanal teórica: 4 horas/aula	Carga horária semanal prática: 0 horas/aula
Data de aprovação na assembleia departamental: 28/08/2023		
Ementa: Modelagem clássica; modelagem orientada a objetos; projeto orientado a objetos; desenvolvimento modular; desenvolvimento dirigido por API (Application Programming Interface); qualidade de software; reuso de software; ferramentas para desenvolvimento de software; evolução de Software; a pesquisa e o futuro da Engenharia de Software.		
Conteúdo Programático: <ul style="list-style-type: none">Modelagem Clássica<ul style="list-style-type: none">EspecificaçõesModelo entidade-relacionamentoDiagramas de fluxo de dadosMáquinas de estadoModelagem Orientada a Objetos<ul style="list-style-type: none">Extração e representação de classes.Unified Modeling Language (UML)Diagrama de classesDiagrama de sequênciaDiagrama de colaboraçãoDiagrama de objetosDiagrama de atividadesDiagrama de estadosProjeto Orientado a Objetos (OO)<ul style="list-style-type: none">Conceitos e principais fundamentos de projeto OOPadrões de Projeto de softwareArquitetura de softwareDesenvolvimento modular<ul style="list-style-type: none">Fraco acoplamento, alta coesão, encapsulamento de informação e eficiência		

- Projeto de Software Dirigido por API (Application Programming Interface)
- Reuso das (APIs) amplamente utilizadas na indústria de software
- Projeto para reuso
- Qualidade de software
 - Confiabilidade
 - Manutenibilidade
 - Usabilidade
 - Desempenho
 - Reusabilidade
- Reúso de software
 - Tipos de reúso
 - Geradores de código
 - Desenvolvimento baseado em componentes e frameworks
 - Desenvolvimento baseado em aspectos
 - Engenharia de domínio
 - Linha de Produto de Software.
- Ferramentas
 - Ambientes de programação (Integrated Development Environment – IDE)
 - Ferramentas para modelagem e projeto de software
 - Ferramentas de teste para análise estática e dinâmica de software
 - Ferramentas para o controle de código e desenvolvimento colaborativo (time)
 - Ferramenta de suporte e integração de software
- Evolução de Software
 - Manutenção de software
 - Reengenharia e engenharia reversa
- A Pesquisa e o futuro da Engenharia de Software
 - Cenário atual da pesquisa em Engenharia de Software
 - Engenharia de Software Experimental
 - Problemas em aberto

Objetivos:

Ao final do curso é esperado que o aluno:

Explicar o ciclo de vida de um software por meio de exemplos, ilustrar suas fases e entregáveis que devem ser produzidos;

Selecionar modelos e processos para o desenvolvimento e manutenção de projetos de software para diferentes domínios de aplicação;

Explicar o papel dos modelos de maturação do processo de desenvolvimento de software;

Comparar diferentes modelos de desenvolvimento de software: modelo ágil, modelo incremental, modelo em cascata, entre outros;

Analisar e avaliar um conjunto de ferramentas de suporte ao desenvolvimento de software (gestão, modelagem, test, etc);

Aplicar ferramentas para o desenvolvimento de um produto de software de médio porte;

Aplicar elementos chave e métodos comuns à elicitação e análise de requisitos para o desenvolvimento de um produto de software de médio porte;

Discutir os desafios na manutenção de software legado;

Identificar as principais questões associadas à evolução de software e explicar seus impactos ao ciclo de vida do software;

Discutir os desafios encontrados na manutenção de software legado e a necessidade da engenharia reversa;

Identificar pontos fracos em um projeto de software e mostrar como eles podem ser removidos através da re-engenharia;

Demonstrar por meio do envolvimento em um projeto de software colaborativo, os elementos centrais da criação e gestão de equipes de desenvolvimento de software;

Preparar um plano de projeto para um produto de software de médio porte que inclua: estimativa de tamanho e esforço, cronograma, alocação de recursos, controle de configuração, gestão de mudanças (escopo) e identificação e gestão de riscos.

Distinguir entre verificação e validação de software;

Descrever o papel das ferramentas para a validação de software;

Distinguir entre os diferentes tipos e níveis de teste (módulo, integração, sistema e aceitação);

Criar, avaliar e implementar um plano de testes para um produto de software de médio porte;

Discutir as questões relacionadas ao teste de software orientador por objeto;

Metodologia:

Esta disciplina provê ao estudante conhecimentos práticos e teóricos acerca dos principais métodos, processos, técnicas e padrões para o desenvolvimento de sistemas de computação ou sistemas de informação. É dado foco aos processos ágeis de software dirigidos por testes e refatoração de código. Ela prepara o estudante para atuar na Indústria de Desenvolvimento de Software. Para isso, o aluno é treinado no uso dos principais conceitos e das melhores práticas envolvidas adotadas pela indústria. O foco dessa disciplina não está no desenvolvimento de estruturas de dados ou de algoritmos eficiente. O foco é dado ao desenvolvimento de projetos de software como um todo, desde a sua concepção dirigida por comportamento, e projeto baseado em padrões estabelecidos pela indústria, passando pela integração com software legado, com sistemas gerenciadores de banco de dados, pelo desenvolvimento de aplicações multitarefa e multiusuário, até desenvolvimento

de interfaces gráficas. Para isso, o estudante é motivado a treinar-se no uso das APIs (Application Programming Interfaces) de bibliotecas ou frameworks destinados ao desenvolvimento de sistemas de computação que são amplamente utilizadas pela indústria. A disciplina é lecionada conforme período letivo previsto no calendário acadêmico e é dividida em dois momentos diferentes. Na primeira metade, o professor utiliza o método de ensino construtivista para apresentar o conteúdo teórico da disciplina e avaliar os estudantes. Na segunda e última metade, o método de ensino dirigido por problemas (PBL – Problem Based Learning) é utilizado para o ensino do conteúdo prático da disciplina. Os estudantes passam por uma espécie de residência, semelhante ao período de residência em hospitais pelo qual passam os médicos ao se graduarem. Durante as aulas, os estudantes passam a enfrentar semanalmente problemas realistas e idênticos aos que enfrentará no desempenho da sua profissão. O estudante tem a oportunidade de conhecer e atuar em acordo com diversos papéis existentes no mercado de trabalho: analista, arquiteto, engenheiro, tester, gestor, etc. O professor simula um ambiente empresarial no qual técnicas, métodos e sistemas de computação precisam ser projetados, construídos e avaliados para a solução de um problema realista.

Requisitos de Hardware e Software:

- Computador pessoal, executando sistema operacional Microsoft Windows 32 ou 64 bits
- Ambiente de desenvolvimento: MinGW - <http://www.mingw.org/>
- Framework para desenvolvimento C++: Qt- <https://www.qt.io/>

Atividades avaliativas:

O seguinte método de avaliação é adotado.

- Somente serão aprovados os estudantes que atingirem 60% de aproveitamento;
- 15 % dos pontos serão distribuídos através de resumos, seminários (apresentações síncronas), vídeos (apresentações assíncronas), listas de exercícios, tutoriais ou laboratórios que deverão ser realizados pelos estudantes e entregues na data estipulada pelo professor. Aqueles alunos que NÃO entregarem as listas de exercícios nas datas corretas serão penalizados em 20% da pontuação obtida;
- 25 % dos pontos serão distribuídos em avaliações teóricas;
- 60% dos pontos serão distribuídos através de trabalhos práticos semanais que deverão ser feitos pelos estudantes e entregues na data estipulada pelo professor. Aqueles alunos que NÃO realizarem as entregas nas datas corretas serão penalizados em 20% da pontuação obtida;
- para aqueles alunos que não atingirem 60% de aproveitamento (somando-se os pontos distribuídos nas listas de exercícios, avaliações teóricas e trabalhos práticos) e com frequência mínima de 75%, serão concedidos exames especiais total ou parciais, conforme resolução CEP2880. O valor resultante deverá ser superior a 6.0 para que o aluno seja aprovado, conforme artigo 5 desta resolução.
- O controle de frequência será realizado em sala de aula e pelas entregas das atividades avaliativas, como listas de exercícios, quizzes, provas e trabalhos práticos

Cronograma:

Aulas 1 a 4 - Semana 1

- Apresentação da Disciplina, Objetivo, Conteúdo, Material Didático e Método de Avaliação.
- Ferramentas: "Porque C++?"; Introdução ao ambiente de Desenvolvimento C++ em MinGw. Ferramentas: "Porque Qt?"; Introdução à biblioteca Qt.
- Projeto arquitetural de software: Arquiteturas monolíticas, em camadas, cliente-servidor,

distribuídas e micro serviços.

- Estudo de caso: arquiteturas dos sistemas Linux, Windows DNA e Pilha de Protocolos TCP/IP.
- Lista de exercício 1: Introdução a Engenharia de Software;
- Lista de Exercício 2: “A vocação de C”;
- Lista de Exercício 3: “Introdução a C++”

Aulas 5 a 8 - Semana 2

- Reuso de código;
- Projeto de software dirigido por API, fraco acoplamento entre módulos, alta coesão dos módulos, encapsulamento de informação e eficiência.
- Dinâmica: Projeto de API de uma Classe container.
- Enunciado do Trabalho prático. Parte 1: Desenvolvimento dirigido por API e teste (TDD) de um framework C++.

Aulas 9 a 12 - Semana 3

- Qualidade de Software em C++: regras para nomenclatura, forma canônica das classes, herança, delegação e polimorfismo, teste e integração continua de software segundo o TDD
 - Projeto modular de software: classes concretas, classes abstratas e o conceito de interface;
 - Desenvolvimento modular em C++: Programação Genérica, templates e iteradores em C++.
- Lista de Exercício 4:” Programação genérica em C++”.

Aulas 13 a 16 - Semana 4

- Estudo de caso: "o tratamento de exceções em C++".
- Introdução aos padrões de projeto de software: o padrão Singleton

Aulas 17 a 20 - Semana 5

- Auxílio na resolução do trabalho prático- parte 1: Elicitação de testes segundo a técnica BDD e a implementação de testes segundo o processo TDD

Aulas 21 a 24 - Semana 6

- Auxílio na resolução do trabalho prático- parte 1: padrões de projeto de software factory method e composite; e o idioma handle-body (ou bridge).

Aulas 25 a 28 - Semana 7

- Ferramentas: Controle de versão de software e controle de mudancas no GitHub.
- Enunciado do trabalho prático – Parte 2: artefatos de software - documento de visão de sistema, escopo do projeto, missão e benefício do produto, backlog, storyboard, histórias de usuário, cenários de teste de aceitação.

Aulas 29 a 32 - Semana 8

- Trabalho prático - Avaliação da parte 1: Seminários e entrevistas. Data: (14 e 16)/nov/2023.

Aulas 33 a 36 - Semana 9

- Trabalho prático - Avaliação da parte 1: Seminários e entrevistas. Data: (29 e 30)/nov/2023.

Aulas 37 a 40 - Semana 9

- Conceitos e princípios fundamentais para o desenvolvimento de projetos de software de software.
- Qualidade de software: confiabilidade, manutenabilidade, usabilidade e desempenho;

- Gestão de projetos segundo o Project Management Institute (PMI e PMBook) e o processo SCRUM;
- Projeto, coleta e análise de métricas (de produto, processo e equipe).
- Processos de desenvolvimento de software tradicionais e ágeis: RUP, SCRUM e XP.

Aulas 41 a 44 - Semana 10

- Análise de requisitos orientada a objetos: elicitación de requisitos, identificação de classes, objetos, heranças, relacionamento e interações. Notação UML. Elicitación de critérios de aceitação segundo o BDD.

Aulas 45 a 48 - Semana 11

- Desenvolvimento de aplicações MVC em Qt.
- Parte 2: Desenvolvimento em 3 camadas (Lógica de Negócio, Interface com o Usuário, Acesso a Dados) de uma aplicação CRUD (Create, Read, Update, Delete)

Aulas 49 a 52 - Semana 12

- Refatoração de código e os ciclos incrementais e evolutivos dos processos de desenvolvimento de software (BDD+TDD+SCRUM+PMI).
- APIs amplamente utilizadas na indústria de software: GUI em Qt, SQL em Qt, Threads em Qt e Sockets em Qt.

Aulas 53 a 56 - Semana 14

- Trabalho prático - Avaliação da parte 2: Seminários e entrevistas, em (16 e 18)/jan/2024.

Aulas 57 a 60 - Semana 15

- Trabalho prático - Avaliação da parte 2: Seminários e entrevistas, em (23 e 25)/jan/2024.

Aulas 61 a 64 - Semana 16

- Trabalho prático - Avaliação da parte 2: Seminários e entrevistas, em (30/jan e 01/fev)/2024.

Aulas 65 a 68 - Semana 17

- Trabalho prático - Avaliação da parte 2: Seminários e entrevistas, em (06 e 08)/fev/2024.
- Prova teórica - A ser realizada em 15/fev/2024.

Aulas 69 a 72 - Semana 18

- Exame especial - A ser realizada na forma de uma entrevista em 22/fev/2024.

Bibliografia Básica:

- Valente, M. T. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. 1ª edição. UmLivro. 2020. ISBN 978-6500019506.
- <https://engsoftmoderna.info/>
- Patterson, D. A. Fox, A. Construindo Software como Serviço (SaaS): Uma abordagem Ágil Usando Computação em Nuvem, 1st ed., Strawberry Canyon LLC, 2015. ISBN 9780984881277.
- <http://www.saasbook.info/translations/br> ou <http://www.saasbook.info/first-edition>
- GAMMA, Erich. Design patterns elements of reusable object-oriented software / Erich Gamma... [et al.]. Reading: Addison Wesley 1995. 395 p. (Addison-Wesley Professional Computing Series). ISBN 0201633612 (enc.).
<https://doku.pub/download/padroes-de-projeto-solucoes-reutilizaveis-de-software-orientado-a-objetos-erich-gamma.pdf>

Bibliografia Complementar:

- SOMMERVILLE, Ian. Engenharia de software. 10. ed. São Paulo: Pearson Education do Brasil, 2018. ISBN 9788543024974.
<https://plataforma.bvirtual.com.br/Leitor/Loader/168127/pdf/6?keep=True>
- SUMMERFIELD, Mark. Advanced Qt programming: creating great software with C++ and Qt4. New York: Addison Wesley 2010. 536 p. ISBN 9780321635907. <http://www.qtrac.eu/aqpbook.html>
- FOWLER, Martin. UML distilled: a brief guide to the standard object modeling language. 3rd ed. Boston: Addison Wesley, c2004. xxx, 175 p. (Object technology series). ISBN 0321193687 (Broch.). <https://integrada.minhabiblioteca.com.br/#/books/9788560031382/cfi/6/2!/4/18/52@0:76.8>
- PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. 780 p. ISBN 9788563308337.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML: guia do usuário. 2. ed. Rio de Janeiro (RJ): Elsevier, Campus, c2006. xviii, 474 p. ISBN 8535217843 (broch.).
- Teste de Software / organizador Henrique Cacique Braga. São Paulo. Pearson Education do Brasil. Ed. 1º. 2016. 139 p. ISBN 9788543020211.
<https://plataforma.bvirtual.com.br/Acervo/Publicacao/150962>
- CRISPIN, Lisa; GREGORY, Janet. Agile testing: a practical guide for testers and agile teams. Upper Saddle River, NJ: Addison Wesley, c2009. xli, 533 p. (The Addison-Wesley signature series). ISBN 9780321534460.