**UNIVERSIDADE FEDERAL DE OURO PRETO**
**DEPARTAMENTO DE COMPUTAÇÃO**

# PLANO DE ENSINO

| | | |
|---|---|---|
| **Nome do Componente Curricular em português:** Engenharia de Software I<br>**Nome do Componente Curricular em inglês:** Software Engineering I | | **Código:** BCC322 |
| **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:** 8 horas/aula | **Carga horária semanal prática:** 0 horas/aula |
| **Data de aprovação na assembleia departamental:** ___ / ___ / _____ | | |

**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:**

- Modelagem Clássica
  - Especificações
  - Modelo entidade-relacionamento
  - Diagramas de fluxo de dados
  - Máquinas de estado
- Modelagem Orientada a Objetos
  - Extração e representação de classes.
  - Unified Modeling Language (UML)
  - Diagrama de classes
  - Diagrama de sequência
  - Diagrama de colaboração
  - Diagrama de objetos
  - Diagrama de atividades
  - Diagrama de estados
- Projeto Orientado a Objetos (OO)
  - Conceitos e principais fundamentos de projeto OO
  - Padrões de Projeto de software
  - Arquitetura de software
- Desenvolvimento modular
  - 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
    - Manutenabilidade
    - 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;

Comprar 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 diferente 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 atotadas 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 em 8 semanas, conforme previsto no calendário acadêmico, e é dividida em dois momentos diferentes. No primeiro bimestre, composto por 4 semanas, o professor utiliza o método de ensino construtivista para apresentar o conteúdo teórico da disciplina e avaliar os estudantes. Nas últimas 4 semanas, 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 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 avalições 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 será concedido um único exame especial. O valor resultante deverá ser superior a 6.0 para que o aluno seja aprovado.

**Cronograma:**

Aulas 1 a 4

- Apresentação da Disciplina, Obejtivo, 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 monoliticas, em camadas, cliente-servidor e distribuídas.
- Estudo de caso: arquiteturas dos sistemas Linux, Windows DNA, Pilha de Protocolos TCP/IP, Java RMI, Whatsapp, TerraLib e TerraME".
- 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

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

Aulas 17 a 20
- Auxilio na resolução do trabalho prático- parte 1: Elictação de testes segundo a técnica BDD e a implentação de testes segundo o processo TDD

Aulas 21 a 24
- Auxilio na resolução do trabalho prático- parte 1: padroes de projeto de sofware method factory e composite; e o idioma handle-body (ou bridge).

Aulas 25 a 28
- 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, estórias de usuário, cenários de teste de aceitação..

Aulas 29 a 32
- Trabalho prático - Avaliação da parte 1: Seminários e entrevistas.

Aulas 33 a 36
- 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 Institure (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 38 a 40
- Análise de requisitos orientada a objetos: elicitação de requisitos, identificação de classes, objetos, heranças, relacionamento e interações. Notação UML. Elicitação de critérios de aceitação segundo o BDD.

Aulas 41 a 44

- Desenvolvimento de applicações MVC em Qt.

Aulas 45 a 48
- Parte 2: Desenvolvimento em 3 camadas Lógica de Nógocio, Interface com o Usuário, Acesso a Dados) de uma aplicação CRUD (Create, Read, Update, Delete)

Aulas 49 a 52
- Refatoração de código e os ciclos incrementais e evolutivos do processos de desenvolvimento software (BDD+TDD+SCRUM+PMI).

Aulas 53 a 56
- APIs amplamente utilizadas na indústria de software: GUI em Qt, SQL em Qt, Threads em Qt e Soquetes em Qt.

Aulas 57 a 62
- Trabalho prático - Avaliação da parte 2: Seminários e entrevistas.

Aulas 63 a 64
- Prova teórica.

Aulas 65 e 66
- Exame especial

**Bibliografia Básica:**


- 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

- 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-soluoes-reutilizaveis-de-software-orientado-a-objetos-erich-gammapdf-6lkvge8em804 e https://integrada.minhabiblioteca.com.br/#/books/9788577800469/cfi/0!/4/4@0.00:0.00

- 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.


**Bibliografia Complementar:**


- 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. 3nd 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

- MCMAHON, Paul E. Integrating CMMI and agile development: case studies and proven techniques for faster performance improvement. Upper Saddle River, NJ: Addison Wesley, 2010. xxxi, 325 p. ISBN 9780321714107.

- 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.).