MC426 – Engenharia de Software

(2º semestre 2021)

Prof. Breno Bernard Nicolau de França

bfranca@unicamp.br

www.ic.unicamp.br/~breno

Aulas Virtuais Online	
Terça-feira e Quinta-feira	08h às 10h
Atendimento agendado	Pelo menos 24h de antecedência

1. Modelo de Aulas

No contexto da pandemia de coronavírus, a disciplina adotará um modelo de **sala de aula invertida**. Nesse modelo, o aluno tem o primeiro contato com o conteúdo por meio de vídeos e textos antes da aula (pré-aula: momento assíncrono) e, em sala virtual online (aula: momento síncrono), o aluno realiza atividades com base naquele conteúdo.

Outras informações importantes:

- Todo conteúdo será disponibilizado via *Google Classroom* e as salas virtuais ocorrerão na ferramenta *Google Meet*, com disponibilização prévia do link no *Google Classroom*.
- Dúvidas serão sanadas durante as aulas. Adicionalmente, o aluno pode solicitar atendimento agendado previamente.
- Meios de comunicação com professor: e-mail, Google Meet e Google Classroom.
- A frequência dos alunos **não será contabilizada** caso atividades síncronas sejam realizadas. Ainda, a nota do aluno não depende da frequência e nenhum aluno será reprovado por frequência. Entretanto, o processo de aprendizado conta com a **participação do aluno nos momentos síncronos e assíncronos**.
- As datas referentes às entregas de avaliações estão disponíveis no cronograma da disciplina, na Seção 6. O horário para entrega é sempre às **23:59 do dia marcado**.
- Casos de plágio (cópia de texto, imagem ou ideia) entre os trabalhos ou de conteúdos externos serão tratados com rigor. A nota da avaliação em questão será anulada sem possibilidade de reposição e o caso será encaminhado à coordenação do curso.

2. Objetivos Terminais

Ao final do curso, o aluno deve ser capaz de:

★ Compreender a necessidade de métodos e práticas adequadas para o desenvolvimento de sistemas de software (*in-the-large*).

★ Desenvolver um sistema de software utilizando um ciclo de vida (fim-a-fim), incluindo métodos, práticas e ferramentas adequados para as atividades de requisitos, projeto e testes de software.

3. Conhecimentos Requeridos

➢ Abstrações de elementos do mundo real em Tipos Abstratos de Dados e Objetos;

➢ Programação modular em linguagem com suporte a definição de interfaces e componentes (módulos).

4. Unidades

4.1. Introdução a Engenharia de Software

Objetivo	O aluno deve ser capaz de **compreender** a natureza do software, bem como os conceitos de qualidade de software, produto e processo de software, e a visão sociotécnica da disciplina de Engenharia de Software.
Conteúdo	Natureza do software, Qualidade de software, Processo de desenvolvimento, Visão sociotécnica da Engenharia de Software.
Procedimento	Pré-aula: Vídeo-aula gravada[1] e Leituras do capítulo 1 do livro texto Aula: Exercício e discussão em sala virtual online. Pós-aula: Leitura de artigo[2]
Avaliação	N/A

4.2. Processo de Software e Métodos Ágeis

Objetivo	O aluno deve ser capaz de **analisar** os diferentes modelos de processos de software (cascata, iterativo/incremental e evolucionários) e os diferentes métodos ágeis e enxutos de desenvolvimento de software. Ainda, o aluno deverá ser capaz de compreender as fronteiras BizDev e DevOps.

[1] Vídeos com exposição do conteúdo da unidade apresentado pelo professor da disciplina, disponibilizado com antecedência (momento assíncrono) antes da aula e, em sala de aula online (momento síncrono), o aluno realiza atividades com base naquele conteúdo.

[2] Cukierman, H. L., Teixeira, C., & Prikladnicki, R. (2007). Um olhar sociotécnico sobre a engenharia de software. Revista de Informática Teórica e Aplicada, 14(2), 199-219.

Conteúdo	Modelos Cascata, Iterativo/Incremental, e Evolucionários; Métodos Ágeis e Enxutos; Fronteiras BizDev e DevOps.
Procedimento	Pré-aula: Vídeo-aula gravada e Leituras do capítulo 2 do livro texto. Aula: Exercício e discussão em sala virtual online. Pós-aula: Leitura de artigo[3]
Avaliação	**A1**: Síntese crítica sobre o histórico dos diferentes modelos de processos, incluindo ágil e enxuto, e como a motivação para DevOps acontece. Critérios: completude, correção e capacidade de síntese. Entrega em **Dupla**.

4.3. Gerência de Configuração de Software

Objetivo	O aluno deve ser capaz de **definir e implementar na prática** uma estratégia de gerência de configuração e mudanças, operando um sistema de controle de versão.
Conteúdo	Controle de versão e Gestão de mudança
Procedimento	Pré-aula: Vídeo-aula gravada e Leitura do Apêndice A do livro texto Aula: Exercícios práticos de Gitflow Pós-aula: Andamento das práticas no projeto
Avaliação	**A2**: Realizar um fluxo gitflow com: (1) solicitações de mudança (issues) no repositório, (2) implementação das mudanças, e (3) versionamento. Apresentar evidências (issues criadas e comentários, commits com alterações, merge) dessas atividades. Entrega em equipe.

4.4. Engenharia de Requisitos

Objetivo	O aluno deve ser capaz de **aplicar** métodos/técnicas para elicitação, análise e especificação de requisitos.
Conteúdo	Elicitação e Análise de Requisitos, Especificação de Requisitos de Software, Histórias de Usuários
Procedimento	Pré-aula: Vídeo-aula gravada e Leitura do capítulo 3 do livro texto Aula: Lista de exercícios, exercícios práticos de entrevistas, *brainstorming*, definição de histórias de usuários. Pós-aula: Andamento das práticas no projeto

[3] de França, B. B. N., Jeronimo Junior, H., & Travassos, G. H. (2016, September). Characterizing DevOps by hearing multiple voices. In Proc. of the 30th Brazilian Symposium on Software Engineering (pp. 53-62). ACM.

Avaliação	**A3**: Para o cenário do projeto, entregar uma lista de requisitos (épicos e histórias) no formato de *issues* priorizada no Gitlab, rotuladas por **Epic** ou **User Story**, conforme o caso. Ainda, as histórias devem ser linkadas com o épico das quais foram derivadas. Entrega em equipe.

4.5. Projeto e Arquitetura de Software

Objetivo	O aluno deve ser capaz de **aplicar** os princípios de projeto de software, desenvolver a arquitetura de um sistema de software, utilizando soluções de reutilização e refatoração
Conteúdo	Princípios de projeto, Arquitetura de software, Componentes e Interfaces, Padrões de Arquitetura e Projeto, Refatoração, Reutilização de Software
Procedimento	Pré-aula: Vídeo-aula gravada e Leitura dos cap. 5, 6 e 7 do livro texto Aula: exercícios práticos (modularização, componentes e interfaces, estilos arquiteturais, padrões de projeto, refatoração) Pós-Aula: Andamento das práticas no projeto
Avaliação	Avaliação em equipe e entrega via Gitlab. 1. **A4.1**: Definir uma arquitetura para o projeto, utilizando diagrama de componentes UML ou C4, com apoio de estilos arquiteturais e frameworks; e Aplicação de padrões de projeto. 2. **A4.2**: Identificação de anomalias e aplicação de refatoração.

4.6. Testes de Software

Objetivo	O aluno deve ser capaz de **definir e aplicar** uma estratégia de testes com base nos requisitos de um sistema de software, bem como aplicar técnicas de testes funcional em nível de unidade e integração.
Conteúdo	Fundamentos de Teste de Software, Técnica de Teste Funcional, Desenvolvimento Orientado a Testes (TDD)
Procedimento	Pré-aula: Vídeo-aula gravada e Leitura do capítulo 8 do livro texto Aula: exercícios práticos de testes Pós-Aula: Andamento das práticas no projeto
Avaliação	Avaliação em equipe e entrega via Gitlab.

	A5: Criação de uma suíte de testes unitários automatizados para o projeto utilizando. Preferencialmente, adição de casos de testes para funções ainda não implementadas utilizando TDD.

4.7. Integração Contínua e Liberação de Software

Objetivo	O aluno deve ser capaz de **definir e implementar** ciclos de integração contínua, definir processos de liberação de software, bem como instrumentar um pipeline de implantação baseado em containers.
Conteúdo	Integração Contínua, Liberação de Software, Pipeline de Implantação.
Procedimento	Pré-aula: Vídeo-aula gravada e Leitura do capítulo 8 do livro texto Aula: Exercícios práticos de integração contínua com pipeline Pós-Aula: Andamento das práticas no projeto
Avaliação	**A6**: Implementação de um pipeline de implantação no gitlab com build automatizado, e testes unitários automatizados. Entrega em equipe.

5. Critérios de Avaliação

A avaliação da disciplina realizada com base em três critérios:

1. **Avaliação de Unidade (A)**: este critério representa 80% da nota final, sendo os pesos das avaliações assim distribuídos: A1, A2 e A6 igual a 10% cada; A3, A4.1, A4.2 e A5 igual a 12,5% cada. As notas serão atribuídas com base no desempenho para as atividades de avaliação previstas para cada unidade (seção 4).
   As avaliações (de A2 a A6) serão todas realizadas no contexto do projeto, mas terão suas notas atribuídas sem considerar outros aspectos do projeto que não aqueles da atividade específica da avaliação. As datas das avaliações e prazos de entrega estão definidos no cronograma (Seção 6). Alunos sem registros de atividades no Google Sala de Aula **e** no repositório do projeto receberão nota zero.
2. **Projeto**: O projeto é utilizado como ferramenta pedagógica para aplicação contextualizada dos conceitos apresentados nas unidades e representa 20% da nota final. Os alunos devem se organizar em equipes de até cinco (5) integrantes. O critério da avaliação é o estado funcional do produto, demonstrado pelas entregas parciais e final. Para nota máxima, espera-se uma aplicação com pelo menos cinco[4] (5) funcionalidades não-triviais prontas para uso. Ainda, as entregas do projeto serão realizadas integralmente pelos repositórios dos projetos (Gitlab) e, neste contexto, os artefatos lá depositados serão objetos de avaliação, bem como as informações

[4] A quantidade de funcionalidades pode variar dependendo da complexidade da aplicação em desenvolvimento. Exceções devem ser discutidas com o professor da disciplina.

relativas às contribuições de cada membro. Alunos sem registros de atividades no repositório receberão nota zero.

$$NF = ((A1+A2+A6) \times 0,1) + ((A3+A4.1+A4.2+A5) \times 0,125) + (Proj \times 0,2)$$

3. **Exame**: O critério para o aluno de exame é ter nota final maior de 2,5 e menor ou igual a 5,0. O aluno deverá refazer apenas atividades em que a nota foi inferior a 5,0. Ao final, a nota máxima após o exame é, no máximo, igual a 5,0.

6. Cronograma

As datas definidas a seguir podem sofrer alterações devido a imprevistos e/ou situações adversas.

Data	**Tópico**
10/08	Apresentação da Disciplina Online
12/08	U1: Introdução a Engenharia de Software
17/08	Atendimento e Organização das Equipes para o Projeto
19/08	U2: Processos de Software
24/08	U2: Métodos Ágeis e Lean (Princípios e Práticas), Fronteiras BizDev e DevOps
26/08	U3: Gerência de Configuração de Software (Git e Gitlab Básico)
30/08	Entrega Avaliação A1
31/08	U3: Gerência de Configuração de Software (Gitflow)
02/09	U3: Gerência de Configuração de Software (Git Avançado)
07/09	Não haverá atividades (Feriado)
09/09	U4: Engenharia de Requisitos: Conceitos e Processo; Elicitação de Requisitos
12/09	Entrega Avaliação A2
14/09	U4: Engenharia de Requisitos: Conceitos e Processo; Elicitação de Requisitos
16/09	U4: Engenharia de Requisitos: Especificação e Histórias de Usuários
21/09	U4: Engenharia de Requisitos: Especificação e Histórias de Usuários
23/09	U4: Engenharia de Requisitos: Design Thinking
28/09	Atendimento e Acompanhamento dos Projetos
30/09	U5: Projeto e Arquitetura de Software (Conceitos e Ciclos de Vida)
04/10	Entrega Avaliação A3

05/10	U5: Projeto e Arquitetura de Software (Descrição, Estilos Arquiteturais)
07/10	U5: Projeto e Arquitetura de Software (Frameworks e Padrões de Projeto)
12/10	Não haverá atividades (Feriado)
14/10	U5: Projeto e Arquitetura de Software (Dívida Técnica e Refatoração)
18/10	Entrega Avaliação A4.1
19/10	U5: Projeto e Arquitetura de Software (Dívida Técnica e Refatoração)
21/10	Atendimento e Acompanhamento dos Projetos
26/10	U6: Teste de Software (Fundamentos)
27/10	Entrega Avaliação A4.2
28/10	U6: Teste de Software (TDD)
02/11	Não haverá atividades (Feriado)
04/11	U6: Teste de Software (TDD)
09/11	U6: Teste de Software (Teste Funcional)
11/11	U6: Teste de Software (Teste Funcional)
16/11	Atendimento e Acompanhamento dos Projetos
18/11	U7: Integração Contínua e Liberação de Software (Entrega/Implantação Contínua)
22/10	Entrega Avaliação A5
23/11	U7: Integração Contínua e Liberação de Software (Entrega/Implantação Contínua)
25/11	Atendimento e Acompanhamento dos Projetos
30/11	Atendimento e Acompanhamento dos Projetos
01/12	Entrega Avaliação A6
02/12	Avaliação Final dos Projetos
07/12	Avaliação Final dos Projetos
20/12	Entrega do Exame Final

7. Bibliografia

O curso utiliza um livro texto e outros livros adicionais. Qualquer material adicional de leitura será anunciado quando necessário.

Livro texto:
Marco Tulio Valente. Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. Leanpub, 2020.
Edição online gratuita: https://engsoftmoderna.info/

Adicionais:

- Sommerville, I. (2016). *Software Engineering*, 10th edition. Pearson.
- Prikladnicki, Rafael, Renato Willi, and Fabiano Milani. Métodos ágeis para desenvolvimento de software. Bookman Editora, 2014.
- Fowler, Martin. Refactoring: improving the design of existing code. Addison-Wesley Professional, 2018.
- Suryanarayana, Girish, Ganesh Samarthyam, and Tushar Sharma. Refactoring for software design smells: managing technical debt. Morgan Kaufmann, 2014.
- Delamaro, M., Jino, M., & Maldonado, J. (2017). *Introdução ao teste de software*. Elsevier Brasil.
- Beck, Kent. Test-driven development: by example. Addison-Wesley Professional, 2003.
- Humble, Jez, and David Farley. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.

8. Sugestões de Temas para o Projeto

- Sistemas baseados em dados abertos, exemplos de fontes:
  - http://www.dados.gov.br/
  - http://www.portaltransparencia.gov.br/
  - http://www.governoaberto.sp.gov.br/
  - https://ourworldindata.org/
- Sistemas ou aplicações para entendimento ou solução de problemas da sociedade, exemplos de temas incluem:
  - Segurança pública;
  - Direitos humanos;
  - Condições de trabalho;
  - Fiscalização de recursos de parlamentares/executivos;
  - Transporte público;
  - Saúde pública;
  - Urbanização;
  - Meio ambiente;
  - entre outros.
- Plataformas para Divulgação Científica (e integrações);
- Qualquer outro tema deve ser discutido e acordado com o professor.