MC426 – Engenharia de Software

(2º semestre 2023)

Prof. Breno Bernard Nicolau de França

bfranca@unicamp.br

www.ic.unicamp.br/~breno

PED: Guilherme Pedro Santos Jardim <g203834@dac.unicamp.br>

Aulas Presenciais	
Terça-feira e Quinta-feira	8h às 10h (CB13)
Atendimento agendado	Pelo menos 24h de antecedência

1. Informações Importantes

- Todo conteúdo será disponibilizado via *Google Classroom*.
- Dúvidas serão sanadas durante as aulas. Adicionalmente, o aluno pode solicitar atendimento agendado previamente.
- Meios de comunicação com professor, além da sala de aula: e-mail, Google Meet e Google Classroom.
- As datas referentes às entregas de avaliações estão disponíveis no cronograma da disciplina (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	Aula expositiva dialogada e Leituras do capítulo 1 do livro texto Exercício e discussão em sala. Pós-aula: Leitura de artigo[1]
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.
Conteúdo	Modelos Cascata, Iterativo/Incremental, e Evolucionários; Métodos Ágeis e Enxutos; Fronteiras BizDev e DevOps.
Procedimento	Aula expositiva dialogada e Leituras do capítulo 2 do livro texto. Exercício e discussão em sala. Pós-aula: Leitura de artigo[2]
Avaliação	**A1**: Especificação (diagrama esquemático e descrição textual associada) de um processo de desenvolvimento idealizado para o projeto a ser desenvolvido na disciplina, considerando características dos diferentes modelos de processos, incluindo ágil e enxuto. Critérios: completude e correção do processo. Entrega **Individual**.

[1] 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.

[2] 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.

4.3. Gerência de Configuração de Software e Integração Contínua

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. Ainda, deve ser capaz de definir e implementar ciclos de integração contínua.
Conteúdo	Controle de versão e Gestão de mudança; Integração Contínua
Procedimento	Aula expositiva dialogada e Leitura do Apêndice A do livro texto Exercícios práticos de Gitflow Exercícios práticos de Integração Contínua 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. Configurar um pipeline de integração contínua contendo build e testes (*ad-hoc*) automatizados. 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	Aula expositiva dialogada e Leitura do capítulo 3 do livro texto Lista de exercícios, exercícios práticos de entrevistas, *brainstorming*, definição de histórias de usuários. Pós-aula: Andamento das práticas no projeto
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	Aula expositiva dialogada e Leitura dos cap. 5, 6 e 7 do livro texto Exercícios práticos (modularização, componentes e interfaces, estilos arquiteturais, padrões de projeto, refatoração) Pós-Aula: Andamento das práticas no projeto
Avaliação	**A4**: Definir uma arquitetura para o projeto, utilizando diagrama de componentes C4, com apoio de estilos arquiteturais e frameworks, refletida no código; e a aplicação de padrões de projeto. Ainda, identificação de anomalias de projeto e código e aplicação de devidas refatorações. Avaliação em equipe e entrega via repositório.

4.6. Testes e Liberação 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. Ainda, o aluno deve ser capaz de definir e implementar processos de liberação de software, bem como instrumentar um pipeline de implantação baseado em containers.
Conteúdo	Fundamentos de Teste de Software, Técnica de Teste Funcional, Testes Automatizados, Liberação de Software, Pipeline de Entrega.
Procedimento	Aula expositiva dialogada e Leitura dos capítulos 8 e 10 do livro texto Exercícios práticos de testes e Entrega Contínua 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 novos casos de testes para funções (não) implementadas. Inclusão dos testes automatizados criados no pipeline de entrega.

5. Critérios de Avaliação

A avaliação da disciplina realizada com base em três critérios:

1. **Avaliação de Unidade (A)**: este critério representa 70% da nota final, sendo os pesos das avaliações assim distribuídos: A1 igual a 10%; A2, A3, A4 e A5 iguais a 15% cada. As notas serão atribuídas com base no desempenho para as atividades de avaliação previstas para cada unidade.
   As avaliações 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 (Git) receberão nota zero nas respectivas avaliações.
2. **Projeto**: O projeto é utilizado como ferramenta pedagógica para aplicação contextualizada dos conceitos apresentados nas unidades e representa 30% 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[3] (5) funcionalidades integradas (funcionalidades isoladas não serão consideradas) e não-triviais prontas para uso. Ainda, as entregas do projeto serão realizadas integralmente pelos repositórios dos projetos (GitHub) e, neste contexto, os artefatos lá depositados serão objetos de avaliação, bem como as informações relativas às contribuições de cada membro. Alunos sem registros de atividades no repositório receberão nota zero no projeto.

$$NF = (A1 \times 0{,}1) + ((A2+A3+A4+A5) \times 0{,}15) + (Proj \times 0{,}2)$$

3. **Exame**: O critério para o aluno de exame é ter nota final maior que 2,5 e menor ou igual a 5,0. O exame consiste em refazer as atividades avaliativas já corrigidas e que o aluno não obteve sucesso. Então, é sugerido que o aluno refaça apenas atividades em que a nota foi inferior a 5,0. Após entregar as reavaliações, a média será recalculada conforme a equação definida anteriormente e, se o aluno atingir pelo menos 5,0 na média final, este será aprovado e terá 5,0 como teto da nota final. Assim, não há possibilidade de um aluno de exame ter uma nota final maior que 5,0 quando aprovado.

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

6. Cronograma

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

Data	Tópico
01/08	Apresentação da Disciplina e Organização das Equipes para o Projeto
03/08	U1: Introdução a Engenharia de Software
08/08	U2: Processos de Software
10/08	U2: Métodos Ágeis e Enxutos
15/08	U2: Métodos Ágeis e Enxutos
17/08	U2: BizDev e DevOps
22/08	U3: Gerência de Configuração de Software (Intro e Git)
24/08	U3: Gerência de Configuração de Software (Gitflow)
03/09	Entrega da Avaliação A1
29/08	U3: Gerência de Configuração de Software
31/08	U3: Testes Automatizados e Integração Contínua
05/09	U3: Testes Automatizados e Integração Contínua
07/09	Não haverá atividade - Feriado
12/09	U4: Engenharia de Requisitos: Conceitos e Processo, Elicitação de Requisitos
14/09	U4: Engenharia de Requisitos: Elicitação e Análise
19/09	U4: Engenharia de Requisitos: Análise e Especificação
20/09	Entrega da Avaliação A2
21/09	U4: Engenharia de Requisitos: Histórias de Usuários
26/09	Não haverá atividade - Conferência
28/09	Não haverá atividade - Conferência
03/10	U5: Projeto e Arquitetura de Software (Conceitos)
05/10	U5: Projeto e Arquitetura de Software (Descrição Arquitetural)
08/10	Entrega da Avaliação A3
10/10	U5: Projeto e Arquitetura de Software (Estilos Arquiteturais)

12/10	Não haverá atividade - Feriado
17/10	Não haverá atividade - Avaliação de Curso
19/10	U5: Projeto e Arquitetura de Software (Frameworks e Padrões de Projeto)
24/10	U5: Projeto e Arquitetura de Software (Padrões de Projeto)
26/10	U5: Projeto e Arquitetura de Software (Dívida Técnica e Refatoração)
31/10	U5: Projeto e Arquitetura de Software (Dívida Técnica e Refatoração)
02/11	Não haverá atividade - Feriado
07/11	U6: Teste de Software (Fundamentos)
09/11	U6: Teste de Software (Fundamentos)
14/11	U6: Teste de Software (Teste Funcional)
16/11	U6: Teste de Software (Teste Funcional)
19/11	Entrega da Avaliação A4
21/11	U6: Teste de Software (Teste Funcional)
23/11	U6: Liberação de Software (Entrega/Implantação Contínua)
28/11	Avaliação Final dos Projetos
30/11	Avaliação Final dos Projetos
03/12	Entrega da Avaliação A5
14/12	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.
- 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/
    - https://www.kaggle.com/datasets
- 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.
- Qualquer outro tema **deve** ser discutido e acordado com o professor.