# MC426 – Engenharia de Software

(1º semestre 2019)

Prof. Breno Bernard Nicolau de França

breno@ic.unicamp.br

www.ic.unicamp.br/~breno

| Dia | Horário |
|---|---|
| **Terça-feira** | 21h-23h (Sala) |
| **Sexta-feira** | 19h-21h (Sala) |
| **Atendimento (com horário marcado)** | Segundas 14h-16h |

## 1. 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.

★ Implantar o sistema desenvolvido em um ambiente de nuvem (IaaS/PaaS).

## 2. 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).

### Avaliação Diagnóstica:

Na primeira semana do curso, será aplicada uma avaliação diagnóstica no intuito de identificar o repertório do aluno no que diz respeito ao conjunto de conhecimentos requeridos que se entende como imprescindíveis para um bom desempenho na disciplina.

Nesta oportunidade, o aluno deverá ser capaz de responder corretamente a um questionário de múltipla escolha com dez questões, envolvendo programação modular, mais especificamente os princípios de abstração, ocultamento de informação, separação de preocupações, e independência funcional.

# 3. Unidades

## 3.1. 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[1] e leitura de artigo[2]. |
| ***Avaliação*** | Lista de questões discursivas. Discussão de possíveis respostas no gabarito. Entrega individual. |

## 3.2. Processos de Software e Gerenciamento de Projetos

| | |
|---|---|
| ***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 BizDevOps. Por fim, o aluno deve ser capaz de definir um cronograma do projeto, apontando prazos estimados, recursos, entregáveis e dependências entre atividades. |
| ***Conteúdo*** | Modelos Cascata, Iterativo/Incremental, e Evolucionários; Métodos Ágeis e Enxutos; Fronteiras BizDevOps; Planejamento e Monitoramento de projetos. |
| ***Procedimento*** | Aula expositiva dialogada, duas leituras de artigos[3][4], exercícios práticos de planejamento. |
| ***Avaliação*** | 1. Síntese crítica sobre as fronteiras BizDevOps em diferentes cenários dos modelos de processos, ágil e lean (Entrega em Dupla). Critérios: completude, correção e capacidade de síntese. |

[1] Este modelo de aula expande o conceito da aula expositiva tradicional, trazendo o aluno para uma discussão do conteúdo exposto, podendo incitar questionamentos, análise crítica e confronto com a realidade. Práticas: convidar alunos a responder questões, formar grupos para discutir e apresentar uma visão sobre um tópico; pedir para alunos levantarem questões; discussão de práticas de como o objeto aparece em cenários reais.

[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.

[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.

[4] Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.

| | |
|---|---|
| | 2. Definir um cronograma com período de 3 meses para o projeto, que precisará ser monitorado e revisado a cada semana. |

## 3.3. Engenharia de Requisitos

| | |
|---|---|
| ***Objetivo*** | O aluno deve ser capaz de aplicar métodos/técnicas para elicitação, análise e especificação de requisitos. |
| ***Conteúdo*** | Elicitação de Requisitos, Análise de Requisitos (UML e BPMN), Especificação de Requisitos de Software |
| ***Procedimento*** | Aula expositiva dialogada, exercícios práticos de entrevistas, *brainstorming*, workshops, e definição de personas. |
| ***Avaliação*** | Entrega da lista de requisitos (épicos) priorizada para o projeto no gitlab. |

## 3.4. 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, Padrões de Projeto Refatoração, Reutilização e Frameworks |
| ***Conteúdo*** | Princípios de projeto, Arquitetura de software, Componentes e Interfaces, Padrões de Projeto, Refatoração, Reutilização de Software |
| ***Procedimento*** | Aula expositiva dialogada, exercícios e laboratórios (modularização, componentes e interfaces, padrões de projeto, refatoração, reutilização e frameworks) |
| ***Avaliação*** | Avaliar arquitetura existente do projeto e redefiní-la, utilizando diagrama de componentes UML, com apoio de estilos, padrões, frameworks e aplicando refatorações. |

## 3.5. Testes de Software

| | |
|---|---|
| ***Objetivo*** | O aluno deve ser capaz de definir 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*** | Aula expositiva dialogada e laboratórios de testes. |

| | |
|---|---|
| ***Avaliação*** | Criação de suíte de testes automatizados, contendo casos de teste para as unidades do sistema desenvolvido no projeto.<br>Adição de novos casos de testes para funções ainda não implementadas utilizando TDD. |

### 3.6. Gerência de Configuração de Software

| | |
|---|---|
| ***Objetivo*** | O aluno deve ser capaz de definir uma estratégia de gerência de configuração e mudanças, operando um sistema de controle de versão. |
| ***Conteúdo*** | Controle de versão, Gestão de mudança, Integração Contínua |
| ***Procedimento*** | Aula expositiva dialogada, laboratórios/exercícios práticos. |
| ***Avaliação*** | Realizar solicitações de mudança (issues) no repositório, com o fluxo de análise, tomada de decisão, versionamento e implementação das mudanças, e execução do fluxo de integração contínua. Apresentar evidências dessas atividades. |

### 3.7. 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*** | Aula expositiva dialogada, laboratórios sobre integração contínua e pipeline de implantação. |
| ***Avaliação*** | Execução de um pipeline de implantação no gitlab com deploy automático funcionando em produção. O pipeline deve conter, no mínimo, três atividades: build, teste unitário, e deploy. |

## 4. Critérios de Avaliação

A avaliação da disciplina realizada com base em três critérios:

1. **Participação**: este critério é individualizado e representa 10% da nota final. A atribuição da nota de participação é proporcional e considerada a frequência, envolvimento nas atividades em sala de aula e laboratório, cumprimento de prazos relativos às entregas (exercícios, projetos, etc.), leituras e vídeos recomendados.

2. **Avaliação de Unidade (A)**: este critério representa 70% da nota final, sendo os pesos das unidades igual a 10% cada. As notas serão atribuídas com base no desempenho do aluno para as atividades de avaliação previstas para cada unidade (seção 3). As datas das avaliações e prazos de entrega estão definidos no cronograma (seção 5).
3. **Projeto**: este critério representa 20% da nota final. Os alunos devem se organizar em equipes de cinco (5) integrantes. Os critérios de avaliação são o estágio final do produto e o nível de utilização das práticas ensinadas em sala.

$$NF = Part \times 0{,}1 + (A1+A2) \times 0{,}1 + (A3+A4+A5+A6+A7) \times 0{,}1 + Proj \times 0{,}2$$

### 4.1. Informações Importantes:

- As datas referentes às entregas, tanto dos projetos quanto dos exercícios, estão disponíveis no cronograma da disciplina.
- A presença é **obrigatória** em todas as aulas (incluindo laboratórios). Frequência inferior a 75% causa reprovação.
- 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.

## 5. Cronograma

As datas definidas a seguir podem sofrer alterações devido a imprevistos e/ou situações adversas. Aulas indicadas com "Lab" serão realizadas no laboratório.

| Data | Tópico |
|---|---|
| 01/03 | Formação de Equipes e Escolha de Projetos |
| | Apresentação da Disciplina (PDD) + Avaliação Diagnóstica |
| 05/03 | Feriado: Não haverá atividades |
| 08/03 | Início dos Projetos |
| | Introdução a Engenharia de Software |
| 12/03 | Processos de Software e Gerenciamento de Projetos |
| 15/03 | Entrega Avaliação A1 |
| | Métodos Ágeis e Lean (Princípios e Práticas) |
| 19/03 | Métodos Ágeis e Lean (BizDevOps e Gestão Ágil) |
| 22/03 | Entrega do Projeto (Sprint 1) |
| | Engenharia de Requisitos (Conceitos e Processo) |

| | |
|---|---|
| 26/03 | Entrega da Avaliação A2 |
| | Engenharia de Requisitos (Elicitação) |
| 29/03 | Engenharia de Requisitos (Especificação) |
| 02/04 | Engenharia de Requisitos (Histórias de Usuários) |
| 05/04 | Entrega do Projeto (Sprint 2) |
| | Engenharia de Requisitos (Casos de Uso) |
| 09/04 | Projeto e Arquitetura de Software (Conceitos e Princípios) |
| 12/04 | Projeto e Arquitetura de Software (Estilos Arquiteturais e Padrões de Projeto) - Lab |
| 16/04 | Projeto e Arquitetura de Software (Estilos Arquiteturais e Padrões de Projeto) - Lab |
| 19/04 | Entrega do Projeto (Sprint 3) e Avaliação A3 |
| | Feriado: Não haverá atividades |
| 23/04 | Atividades em Lab |
| 26/04 | Atividades em Lab |
| 30/04 | Projeto e Arquitetura de Software (Refatoração) - Lab |
| 03/05 | Entrega do Projeto (Sprint 4) |
| | Projeto e Arquitetura de Software (Refatoração) - Lab |
| 07/05 | Projeto e Arquitetura de Software (Reutilização e Frameworks) - Lab |
| 10/05 | Teste de Software (Fundamentos) |
| 14/05 | Teste de Software (Fundamentos) |
| 17/05 | Entrega do Projeto (Sprint 5) e Avaliação A4 |
| | Teste de Software (Teste Funcional) - Lab |
| 21/05 | Avaliação do Curso: Não haverá atividades |
| 24/05 | Teste de Software (Teste Funcional) - Lab |
| 28/05 | Teste de Software (TDD) - Lab |
| 31/05 | Entrega do Projeto (Sprint 6) e Avaliação A5 |
| | Gerência de Configuração de Software (Gestão de Mudança) |
| 04/06 | Gerência de Configuração de Software (Controle de Versão) - Lab |
| 07/06 | Integração Contínua - Lab |

| | |
|---|---|
| 11/06 | Integração Contínua - Lab |
| 14/06 | Entrega do Projeto (Sprint 7) e Avaliação A6 |
| | Liberação de Software (Entrega/Implantação Contínua) |
| 18/06 | Liberação de Software (Pipeline de Implantação) - Lab |
| 21/06 | Feriado: Não haverá atividades |
| 26/06 | Apresentação de Trabalhos |
| 28/06 | Entrega Final do Projeto (Sprint 8) e Avaliação A7 |
| 12/07 | Exame |

## 6. Bibliografia

O curso é baseado nos seguintes livros texto, ou edições mais novas dos mesmos. Qualquer material adicional de leitura será anunciado, em sala, quando necessário.


- 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.


## 7. Sugestões de Temas para o Projeto

- e-Health (sistemas de apoio a saúde)
- IoT (Internet das Coisas)
- Software para Robôs
- Sistemas de apoio a processos produtivos (automação industrial)
- Sistemas baseados em dados abertos
- Sistemas Diretoria Acadêmica (DAC) contatos Marcelo e Alex
    - Projeto 1 - Intranet da DAC com Pedido de Férias (integração Redmine) e Pedido de Capacitação de funcionários

    - Projeto 2 – Melhoria do formulário de atendimento ao usuário (mais que um formulário, um sistema de solicitações em cima do Redmine)
    - Projeto 3 – Processo de compras (maior deles, nem se sabe como faz)
    - Projeto 4 – Atendimento virtual: dá para ser um chatbot (Balcão da DAC)
    - Projeto 5 - Localização de salas de aula no campus
- Qualquer outro tema deve ser discutido e acordado com o professor.