

Prof. esp. Thalles Canela

- **Graduado:** Sistemas de Informação - Wyden Facimp
- **Pós-graduado:** Segurança em redes de computadores - Wyden Facimp
- **Professor:** Todo núcleo de T.I. (Graduação e Pós) - Wyden Facimp
- **Diretor:** SCS
- **Gerente de Projetos:** Motoca Systems

Redes sociais:

- **Linkedin:** <https://www.linkedin.com/in/thalles-canela/>
- **YouTube:** <https://www.youtube.com/aXR6CyberSecurity>
- **Facebook:** <https://www.facebook.com/axr6PenTest>
- **Instagram:** https://www.instagram.com/thalles_canela
- **Github:** <https://github.com/ThallesCanela>
- **Github:** <https://github.com/aXR6>
- **Twitter:** <https://twitter.com/Axr6S>

Introdução à Engenharia de Software

- O que é Engenharia de Software?
- Importância no mundo atual
- Objetivos e abordagens




Introdução à Engenharia de Software

- A Engenharia de Software refere-se à aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para o desenvolvimento, operação e manutenção do software.
 - Em um mundo cada vez mais digitalizado, a necessidade de criar software confiável, eficiente e seguro é fundamental.
-



Introdução à Engenharia de Software


- **Exemplo:**
 - Pense na construção de uma casa. A arquitetura (projeto), os materiais e a mão de obra (codificação) devem estar em sintonia para criar uma residência segura e funcional. Da mesma forma, a Engenharia de Software busca harmonizar todos os aspectos do desenvolvimento.
-



Fases da Engenharia de Software

Cada fase tem um papel crucial.

- **Requisitos:** é onde as necessidades do cliente são entendidas.
 - **Design:** trata da arquitetura do software.
 - **Implementação:** envolve codificação.
 - **Verificação:** garante que o software atenda aos padrões de qualidade
 - **Manutenção:** lida com atualizações e correções.
-



Fases da Engenharia de Software

Cada fase tem um papel crucial.

- Ao criar um aplicativo de previsão do tempo, primeiro entenda o que o usuário quer ver (**Requisitos**), depois decida a aparência e a funcionalidade (**Design**), escreva o código (**Implementação**), teste-o (**Verificação**) e, finalmente, atualize-o conforme necessário (**Manutenção**).
-



Modelos de Ciclo de Vida de Software

- Cascata
 - Incremental
 - Ágil
 - Espiral
-




Modelos de Ciclo de Vida de Software

- Esses modelos determinam a abordagem para o desenvolvimento do software.
 - Por exemplo, Cascata é uma abordagem sequencial, enquanto Ágil é mais flexível e adaptável às mudanças.
-



Modelos de Ciclo de Vida de Software

- **Exemplo:**
 - Uma startup pode optar pelo modelo Ágil devido às constantes mudanças nas necessidades dos clientes, enquanto uma grande empresa com requisitos bem definidos pode escolher o modelo de Cascata.
-




Modelo Cascata

- Linear e sequencial
 - Cada fase depende da anterior
 - Pouca flexibilidade para mudanças
-



Modelo Cascata

- O modelo Cascata é um dos primeiros modelos de desenvolvimento, caracterizado por uma sequência linear e fixa de fases. Uma vez que uma fase é concluída, geralmente não é possível voltar atrás sem começar tudo de novo.
-




Modelo Cascata - Onde se aplica

- Projetos pequenos ou quando os requisitos são muito claros e improváveis de mudar.
-



Modelo Cascata - Objetivo

- Minimizar a sobreposição entre fases e ter uma visão clara do progresso.
-



Modelo Cascata - Recomendação de leitura

- Royce, W. W. "Managing the Development of Large Software Systems." Proceedings of IEEE WESCON.
-




Modelo Iterativo e Incremental

- Desenvolvimento em ciclos (iterações)
 - Cada iteração resulta em um incremento do software
 - Mais flexível que o Cascata
-




Modelo Iterativo e Incremental

- O modelo Iterativo e Incremental divide o desenvolvimento em ciclos, onde cada ciclo entrega uma parte funcional do software. É mais adaptável a mudanças do que o Cascata.
-




Modelo Iterativo e Incremental - Onde se aplica

- Projetos que não têm todos os requisitos definidos desde o início ou que podem se beneficiar de feedback contínuo.
-



Modelo Iterativo e Incremental - Objetivo

- Permitir a adaptação a mudanças e proporcionar entregas parciais ao longo do projeto.
-



Modelo Iterativo e Incremental - Recomendação de leitura

- Larman, C., & Basili, V. R. "Iterative and incremental development: A brief history." Computer.
-




Modelo Ágil

- Adaptação rápida a mudanças
 - Feedback contínuo
 - Foco no valor para o cliente
-



Modelo Ágil

- O modelo Ágil se concentra em entregas rápidas, feedback constante e adaptabilidade. Ele se baseia em iterações curtas e incorpora o feedback do cliente em cada ciclo.
-



Modelo Ágil - Onde se aplica

- Projetos em ambientes dinâmicos, onde os requisitos podem mudar frequentemente e a colaboração constante com o cliente é essencial.
-



Modelo Ágil - Objetivo

- Maximizar o valor entregue ao cliente e adaptar-se rapidamente às mudanças.
-



Modelo Ágil - Recomendação de leitura

- Beck, K. et al. "Manifesto for Agile Software Development."






Modelo Espiral

- Combinação de abordagens
 - Foco em avaliação e gestão de riscos
 - Iterações em forma de espiral
-



Modelo Espiral

- O modelo Espiral é uma combinação dos modelos Cascata e Iterativo, com uma ênfase particular na avaliação e gestão de riscos a cada ciclo.
-



Modelo Espiral - Onde se aplica

- Projetos complexos e de grande porte, onde os riscos precisam ser cuidadosamente avaliados e gerenciados.
-



Modelo Espiral - Objetivo

- Prevenir e gerenciar riscos, garantindo que sejam identificados e tratados antes que se tornem problemas críticos.
-



Modelo Espiral - Recomendação de leitura

- Boehm, B. W. "A spiral model of software development and enhancement." ACM SIGSOFT Software Engineering Notes.
-



Práticas e Técnicas Fundamentais

- **Programação Orientada a Objetos**
 - **Versionamento de Código**
 - **Teste de Software**
 - **Integração Contínua**
-



Práticas e Técnicas Fundamentais

- Estas são algumas das práticas fundamentais adotadas na indústria para garantir software de alta qualidade. Por exemplo, o versionamento de código permite que várias pessoas colaborem no mesmo projeto sem conflitos.
-



Práticas e Técnicas Fundamentais

- **Exemplo:**
 - Ao desenvolver um jogo, os desenvolvedores podem usar programação orientada a objetos para modelar personagens, enquanto os testes garantem que o jogo funciona como esperado em diferentes cenários.
-



Ferramentas e Ambientes de Desenvolvimento

- **IDEs:** Eclipse, IntelliJ, Visual Studio
 - **Controle de Versão:** Git, SVN
 - **Ferramentas de CI/CD:** Jenkins, Travis CI
-




Ferramentas e Ambientes de Desenvolvimento

- Ferramentas facilitam o desenvolvimento, manutenção e entrega do software. IDEs ajudam no desenvolvimento, ferramentas de controle de versão rastreiam mudanças e ferramentas de CI/CD ajudam na entrega contínua do software.
-



Ferramentas e Ambientes de Desenvolvimento

- **Exemplo:**
 - Um desenvolvedor web pode usar o Visual Studio Code para escrever código, Git para rastrear alterações e Jenkins para implementação automática.
-



Dada uma situação onde você é encarregado de desenvolver um sistema de gerenciamento de biblioteca para uma universidade:

- Liste os possíveis requisitos do sistema.
 - Escolha um modelo de ciclo de vida e justifique sua escolha.
 - Nomeie algumas práticas e técnicas que você implementaria.
 - Que ferramentas você consideraria usar durante o desenvolvimento?
-