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

Engenharia de Requisitos e Análise de Sistema

- **Exemplo:** Desenvolvimento de um aplicativo de entrega de alimentos.
- **Ferramenta:** JIRA para gerenciamento de requisitos.
- **Fundamentação:** A engenharia de requisitos serve como alicerce para todo o projeto, garantindo que o software atenda às necessidades dos usuários.

Etapas da Engenharia de Requisitos

- **Elicitação de Requisitos:** Processo de coleta das necessidades e desejos dos stakeholders.
  - **Exemplo:** Entrevistas, brainstorming e observações.
- **Análise:** Verificação da viabilidade dos requisitos coletados e remoção de ambiguidades.
  - **Exemplo:** Um requisito que sugere a criação de um recurso que não é tecnicamente viável.
- **Especificação:** Documentação detalhada e formal dos requisitos.
  - **Exemplo:** Documento de Especificação de Requisitos de Software (SRS).
- **Validação:** Certificar-se de que os requisitos são corretos, completos e atendem às necessidades do usuário.
  - **Exemplo:** Revisões por pares, prototipagem.

Ferramentas e Técnicas

- **JIRA:** Uma ferramenta popular que permite rastrear, priorizar e discutir tarefas e requisitos de uma equipe.
- **Casos de Uso:** Descrições de como os usuários interagem com o sistema.
- **Diagramas UML:** Ajudam na visualização de requisitos, especialmente requisitos de sistema.

Desafios na Engenharia de Requisitos

- **Mudanças frequentes:** Requisitos podem mudar à medida que o projeto avança ou novas informações são descobertas.
- **Ambiguidades:** Requisitos mal definidos ou vagos podem levar a problemas de implementação.
- **Comunicação:** Garantir que todas as partes interessadas estejam na mesma página pode ser um desafio, especialmente em equipes grandes ou distribuídas.

Exemplo Prático - Aplicativo de Entrega de Alimentos

- **Elicitação:** Entrevistas com clientes potenciais revelaram a necessidade de um sistema de avaliação para restaurantes e entregadores.
- **Análise:** Avaliar a viabilidade técnica de ter um sistema de localização em tempo real para os entregadores.
- **Especificação:** Criação de um SRS que detalha todas as funcionalidades, como busca de restaurantes, sistema de pagamento, avaliações, entre outros.
- **Validação:** Usando protótipos para confirmar se as funcionalidades atendem às expectativas dos usuários.

Projeto do Sistema

- **Exemplo:** Design da arquitetura de um aplicativo multiplataforma.
- **Ferramenta:** UML para modelagem.
- **Fundamentação:** Uma arquitetura bem definida assegura escalabilidade, manutenibilidade e desempenho.

O que é Projeto do Sistema?

- **Definição:** Etapa em que é definida a arquitetura do software, são selecionados os padrões de projeto, e é estabelecida a estrutura geral do sistema.
- **Importância:** Uma arquitetura bem projetada garante um sistema robusto, escalável e de fácil manutenção. É a base para a fase de implementação.

Exemplo - Design da Arquitetura de um Aplicativo Multiplataforma

- **Explicação:** Ao projetar um aplicativo multiplataforma, como um de streaming de música, considerações importantes incluem:
  - Se a base de código será unificada (usando frameworks como Flutter ou React Native) ou nativa para cada sistema operacional.
  - Como os dados serão sincronizados entre plataformas.
  - Como garantir uma experiência de usuário consistente em todos os dispositivos.

Ferramentas para Projeto de Sistema

- **UML (Linguagem de Modelagem Unificada):** Padrão para esboçar e visualizar o design do seu software.
- Exemplo: Diagrama de Classes para mostrar a estrutura do aplicativo e as relações entre suas partes.
- **Software de Design de Interface, como Figma ou Adobe XD:** Auxilia no design de interfaces de usuário para garantir uma experiência fluida e intuitiva.
- **Ferramentas de Design de Banco de Dados, como MySQL Workbench:** Permitem projetar a estrutura do banco de dados, garantindo eficiência na armazenagem e recuperação de dados.

Fundamentação

- Citando Sommerville (2011), o projeto do sistema é uma fase crucial, pois:
  - É onde as decisões críticas são tomadas para garantir a eficácia do software.
  - Um projeto inadequado pode levar a problemas significativos na fase de implementação e manutenção.
  - Esta fase garante que o software seja escalável, eficiente e manutenível.

Considerações Finais

- A fase de projeto é uma etapa estratégica e técnica.
- As decisões tomadas aqui terão um impacto direto na qualidade e eficácia do software final.

Implementação e Testes

- **Exemplo:** Codificação do sistema de login do aplicativo.
- **Ferramenta:** GitHub para versionamento e Jenkins para integração contínua.
- **Fundamentação:** Esta fase dá vida ao projeto, transformando os requisitos e projetos em um produto real.

Implementação:

- **Definição:** É a fase onde o design e a arquitetura do sistema são transformados em código funcional.
- **Exemplo:** Considere o desenvolvimento do sistema de login de um aplicativo. Esta etapa envolve a codificação real do formulário de login, integração com bancos de dados para verificação de credenciais, entre outros.
- **Ferramenta em destaque:**
  - **GitHub:** Uma plataforma de hospedagem de código-fonte que facilita a colaboração entre desenvolvedores, o rastreamento de problemas e a integração com outras ferramentas.

Testes:

- **Definição:** É a etapa de verificar e validar o software para garantir que ele esteja livre de erros, atenda aos requisitos e ofereça uma experiência do usuário de alta qualidade.
- **Tipos de Testes:**
  - **Testes unitários:** Testam partes individuais (unidades) do software.
  - **Testes de integração:** Testam a combinação de unidades para garantir que elas funcionem juntas.
  - **Testes de sistema:** Testam o sistema como um todo.
- **Exemplo:** Usando nosso sistema de login, os testes unitários podem verificar se a entrada de senha está sendo corretamente criptografada. Testes de integração podem verificar se o sistema de login se integra corretamente ao banco de dados. Já os testes de sistema podem simular cenários de usuários reais tentando fazer login.

Testes:

- **Ferramenta em destaque:**
  - **Jenkins:** Uma ferramenta de integração contínua que permite automatizar a construção e teste de projetos, garantindo que mudanças recentes não introduzam erros.

Fundamentação:

- A fase de implementação é crítica porque é aqui que as ideias se materializam em produtos. A fase de testes é igualmente crucial, pois um software bem codificado mas mal testado pode resultar em um produto de baixa qualidade ou em falhas graves, impactando negativamente a experiência do usuário e a reputação da organização.

Manutenção

- **Exemplo:** Atualização do aplicativo para incluir novos restaurantes.
- **Ferramenta:** Git para controle de versão e Docker para implementação de mudanças.
- **Fundamentação:** Mesmo após o lançamento, o software deve evoluir para atender às necessidades em constante mudança dos usuários.

Definição:

- A fase pós-lançamento do software onde atualizações, correções e melhorias são implementadas para assegurar a relevância e eficácia do produto.

Tipos de Manutenção:

- **Corretiva:** Resolução de bugs e falhas identificados após o lançamento.
  - **Exemplo:** Um usuário reporta que o aplicativo fecha inesperadamente ao acessar uma funcionalidade específica. A equipe corrige esse bug.
- **Adaptativa:** Alterações feitas para adaptar o software a novos ambientes ou requisitos.
  - **Exemplo:** Atualizar o aplicativo para ser compatível com a última versão do sistema operacional.
- **Aperfeiçoamento (ou Evolutiva):** Implementação de novas funcionalidades ou melhorias nas já existentes.
  - **Exemplo:** Adicionar um novo recurso de recomendação personalizada para o usuário no aplicativo.
- **Preventiva:** Ações tomadas para evitar problemas futuros, otimizando código ou atualizando bibliotecas.
  - **Exemplo:** Atualizar uma biblioteca de terceiros que será descontinuada em breve.

Ferramentas:

- **Git:** Ferramenta de controle de versão que permite rastrear mudanças e colaborar com equipes.
- **Docker:** Plataforma que facilita a implementação de aplicações em um ambiente isolado chamado container. Pode ser útil para testar novas atualizações em um ambiente controlado.

Importância:

- A manutenção garante que o software permaneça atualizado, confiável e seguro.
- Ela permite que o software evolua com as mudanças nas necessidades dos usuários e no ambiente tecnológico.

Desafios da Manutenção:

- Compreender código antigo ou mal documentado.
- Manter a compatibilidade com versões anteriores.
- Gerenciar riscos ao introduzir mudanças em sistemas em funcionamento.

Caso de Estudo

- **Título:** Spotify: Um Estudo de Caso
- **Descrição:** Apresentar brevemente como o Spotify gerencia suas fases de desenvolvimento, desde a coleta de requisitos até a manutenção.

Introdução ao Spotify

- **Breve descrição:** O Spotify é uma plataforma de streaming de música que tem milhões de usuários e uma ampla gama de funcionalidades. Como eles gerenciam as fases de desenvolvimento?

Engenharia de Requisitos e Análise de Sistema no Spotify

- **Exemplo:** Introdução do recurso "Descobertas da Semana".
- **Como funciona:** O Spotify coletou feedback dos usuários e identificou a necessidade de recomendações personalizadas semanais.
- **Desafio:** Como fornece recomendações personalizadas para milhões de usuários com base em seus gostos musicais?

Projeto do Sistema no Spotify

- **Exemplo:** Design do algoritmo de recomendação.
- **Como funciona:** Utilizando data mining e machine learning, o Spotify desenvolveu um sistema que analisa o histórico de escuta do usuário e compara com outros usuários para gerar recomendações.
- **Desafio:** Garantir que as recomendações sejam relevantes e variadas.

Implementação e Testes no Spotify

- **Exemplo:** Codificação e teste do recurso "Descobertas da Semana".
- **Como funciona:** Depois de projetar o algoritmo, a equipe de desenvolvimento do Spotify implementa a funcionalidade. Ela é então testada internamente e com um grupo selecionado de usuários (beta testers) antes de ser lançada para todos.
- **Desafio:** Garantir que o recurso funcione sem problemas para todos os usuários, independentemente da região ou do dispositivo.

Manutenção no Spotify

- **Exemplo:** Atualizações no recurso "Descobertas da Semana".
- **Como funciona:** Com base no feedback dos usuários e nas métricas de uso, o Spotify faz ajustes e melhorias no recurso. Por exemplo, eles podem ajustar o algoritmo para melhorar a precisão das recomendações.
- **Desafio:** Continuar a fornecer recomendações de alta qualidade à medida que a base de usuários cresce e seus gostos musicais evoluem.

Reflexão sobre o Estudo de Caso

- **Pergunta para a turma:** Como vocês acham que o Spotify poderia melhorar ainda mais o recurso "Descobertas da Semana"?
- **Objetivo:** Engajar os alunos a pensar criticamente sobre como as fases de desenvolvimento são aplicadas em um caso real e incentivá-los a pensar em inovações.

Conclusão e Reflexão

- Resumo dos pontos principais discutidos.
- Importância de cada fase no ciclo de vida do software.

Recapitulação

- **Título:** Pontos-chave da Aula

- **Itens:**

- Engenharia de Requisitos: Alicerce do software.
- Projeto do Sistema: Blueprint funcional.
- Implementação e Testes: Onde o código ganha vida.
- Manutenção: Evolução após lançamento.

A Importância das Fases

- **Título:** Por que cada fase é crucial?

- **Itens:**

- **Engenharia de Requisitos:** Evita retrabalho e garante que o software atende às necessidades.
- **Projeto:** Garante uma estrutura sólida para o software, facilitando expansões futuras.
- **Implementação e Testes:** Assegura a funcionalidade correta e robustez do software.
- **Manutenção:** A tecnologia e as necessidades dos usuários estão em constante evolução. Softwares também devem evoluir.

Reflexão

- **Título:** A Integração entre as Fases

- **Texto:**
- "Desenvolver software não é apenas sobre escrever código. É sobre entender as necessidades, planejar de forma eficaz, executar com precisão e adaptar-se às mudanças. Cada fase no desenvolvimento de software está intrinsecamente ligada à anterior e à subsequente, formando um ciclo contínuo de evolução e melhoria."

Pensamento Final

- **Título:** O Software como uma Entidade Viva

- **Texto:**
- "Assim como um ser vivo, o software passa por várias fases de crescimento e adaptação. Compreender e respeitar cada fase é a chave para criar soluções eficazes, robustas e duradouras que atendam e superem as expectativas dos usuários."

Referências Bibliográficas

- Sommerville, I. (2011). Engenharia de software. Addison-Wesley.
- Pressman, R. S., & Maxim, B. R. (2014). Engenharia de Software: Uma Abordagem Profissional. AMGH.
- Artigo: "Understanding software development process based on software engineering ontology" - Journal of Systems and Software.
- Site científico: IEEE Digital Library.
- Revista: ACM Computing Surveys.