

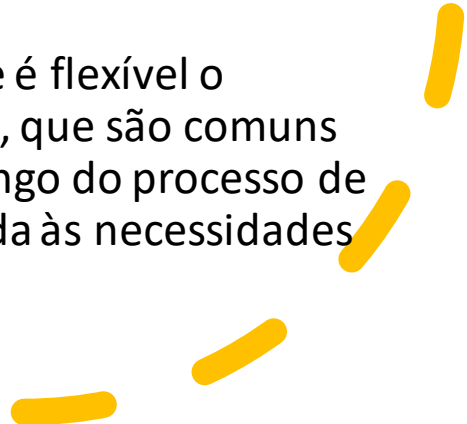
# Prof. esp. Thalles Canela

- **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

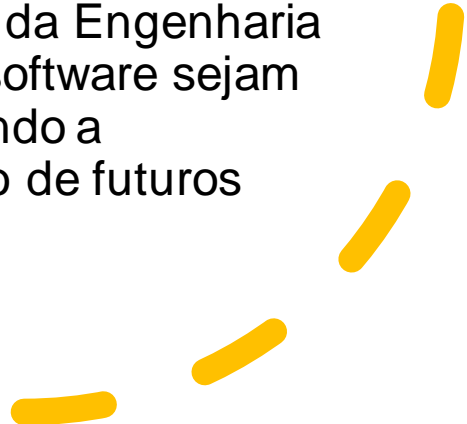
## Redes sociais:

- **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](https://www.instagram.com/thalles_canela)
- **Github:** <https://github.com/ThallesCanela>
- **Github:** <https://github.com/aXR6>
- **Twitter:** <https://twitter.com/Axr6S>

# Introdução ao curso e à Engenharia de Software

- **Organização e Estrutura:** A Engenharia de Software fornece uma estrutura clara e organizada para o processo de desenvolvimento. Ela define fases, atividades e metodologias que orientam os desenvolvedores, garantindo que todas as etapas necessárias sejam realizadas de forma coerente.
  - **Melhoria na Qualidade:** A Engenharia de Software promove práticas que visam a qualidade do software, incluindo a realização de testes rigorosos, revisões de código, identificação e correção de defeitos. Isso resulta em sistemas mais confiáveis, com menos problemas e maior satisfação do usuário.
  - **Economia de Tempo e Recursos:** Seguir processos de Engenharia de Software ajuda a evitar retrabalhos e mudanças de última hora, economizando tempo e recursos. As metodologias de gerenciamento de projetos, por exemplo, permitem o controle eficaz dos prazos e recursos.
  - **Adaptação a Mudanças:** A Engenharia de Software é flexível o suficiente para acomodar mudanças nos requisitos, que são comuns em projetos de software. Ela permite ajustes ao longo do processo de desenvolvimento, garantindo que o software atenda às necessidades atuais.
- 

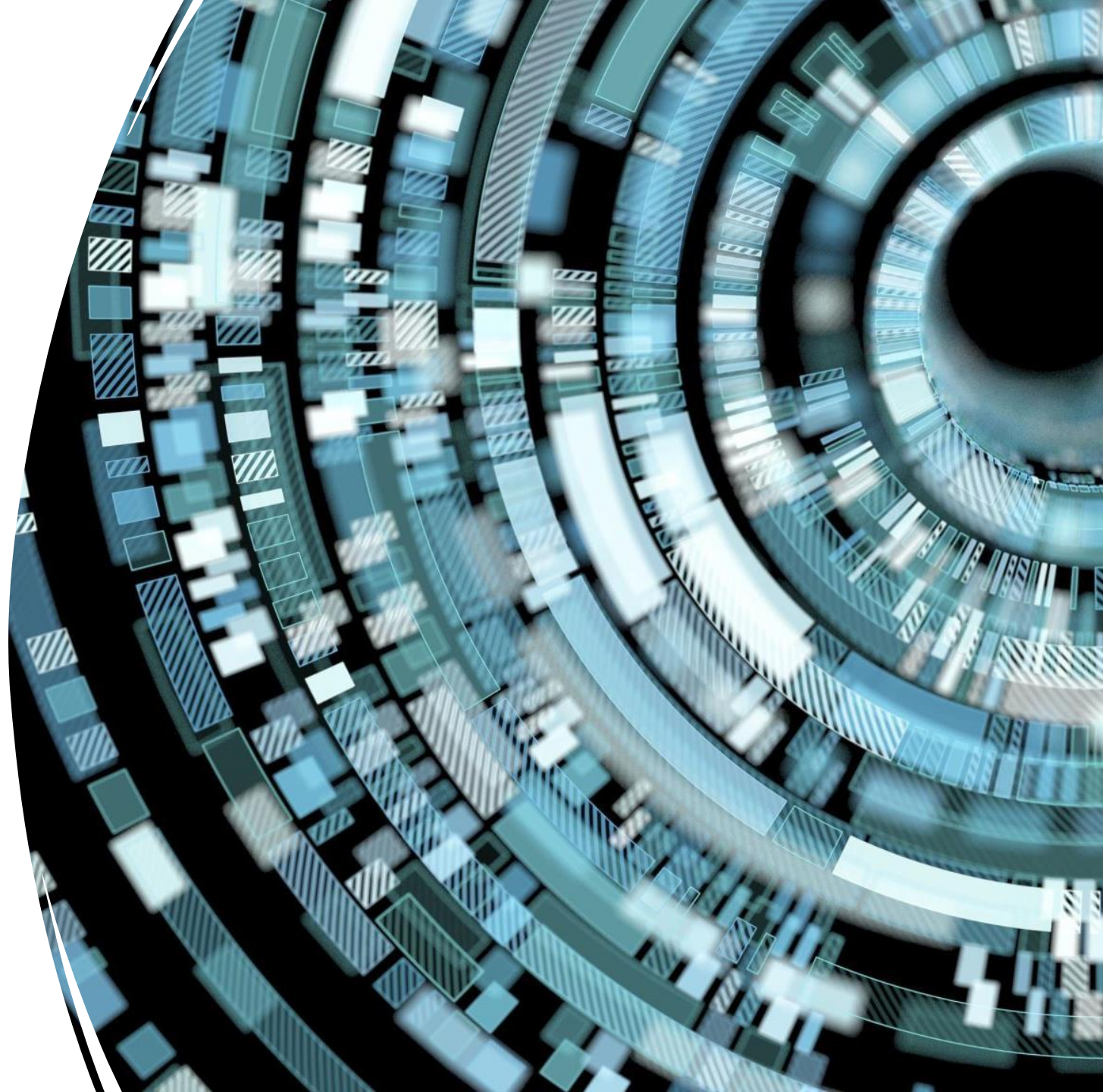
# Introdução ao curso e à Engenharia de Software

- **Redução de Riscos:** A identificação e o gerenciamento de riscos são partes integrantes da Engenharia de Software. Isso ajuda a prever problemas potenciais e a adotar medidas preventivas para minimizar impactos negativos.
  - **Comunicação Eficiente:** A Engenharia de Software incentiva uma comunicação clara e eficiente entre as equipes de desenvolvimento, stakeholders e usuários finais. Isso ajuda a garantir que todos compreendam os requisitos e expectativas, evitando mal-entendidos.
  - **Documentação Adequada:** A Engenharia de Software promove a criação de documentação detalhada, que é valiosa para o desenvolvimento contínuo, manutenção e entendimento do software, mesmo após longos períodos.
  - **Escalabilidade e Reutilização:** A abordagem da Engenharia de Software permite que os componentes do software sejam projetados com reutilização em mente, facilitando a escalabilidade do sistema e o desenvolvimento de futuros projetos.
- 

# Engenharia de Software

---

- Aplicação de princípios de engenharia no desenvolvimento, operação e manutenção de sistemas de software.

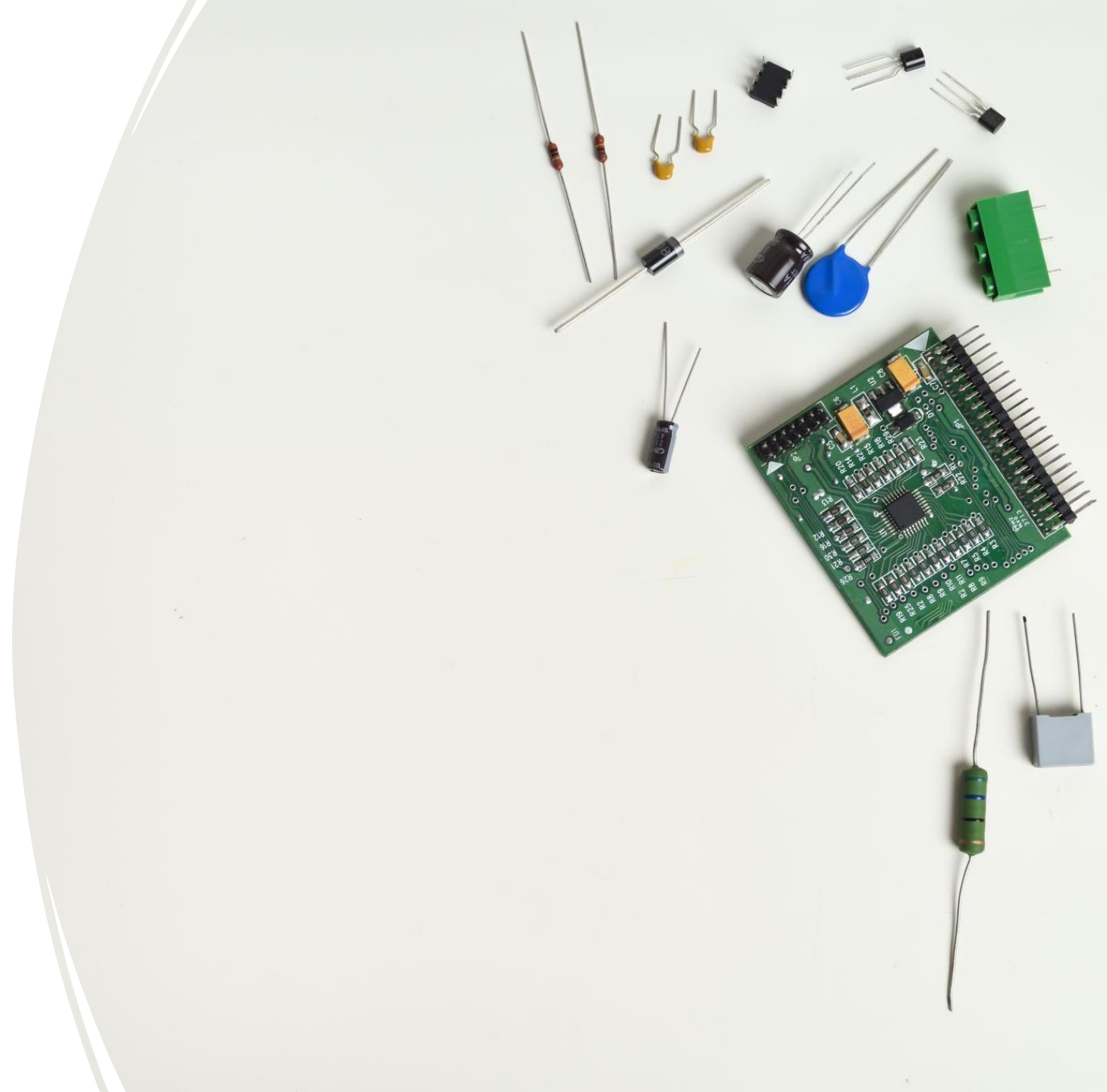




# Processo de Fabricação do Hardware x Processo de Desenvolvimento do Software

---

- Comparação dos Processos:
- Hardware: Fabricação física.
- Software: Criação intangível.



# Desafios Únicos do Desenvolvimento de Software

Intangibilidade

Flexibilidade

Replicação

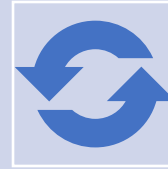
## Intangibilidade:

- Intangibilidade é uma característica fundamental do software. Diferente do hardware, que é tangível e pode ser tocado, o software é imaterial, não tem forma física. Ele é composto por códigos, algoritmos, dados e instruções que operam em máquinas, mas não pode ser tocado ou manipulado diretamente no sentido tradicional.

# Intangibilidade:



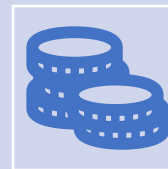
**Fácil Reprodução:** Como o software é intangível, ele pode ser facilmente copiado e distribuído, o que é uma grande vantagem para a disseminação de aplicativos e sistemas.



**Manutenção e Atualização:** Alterar ou atualizar um software é mais flexível, pois não envolve mudanças físicas. Pode-se corrigir erros e implementar melhorias de maneira relativamente mais ágil.



**Facilidade de Distribuição:** A distribuição online de software é uma realidade, permitindo que as pessoas acessem e instalem programas sem a necessidade de mídias físicas.



**Custo Marginal Zero:** Após o desenvolvimento inicial, a replicação e distribuição de cópias adicionais de software têm um custo marginal próximo a zero, o que é uma característica econômica importante.



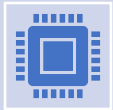


# Flexibilidade:

---

- A flexibilidade é outra característica essencial do software. A natureza intangível permite que o software seja altamente configurável e adaptável às necessidades do usuário e às mudanças no ambiente.

# Flexibilidade:



A flexibilidade é outra característica essencial do software. A natureza intangível permite que o software seja altamente configurável e adaptável às necessidades do usuário e às mudanças no ambiente. Alguns aspectos da flexibilidade incluem:



**Customização:** É possível personalizar o software para atender a requisitos específicos de diferentes usuários, o que é especialmente relevante em aplicações empresariais.



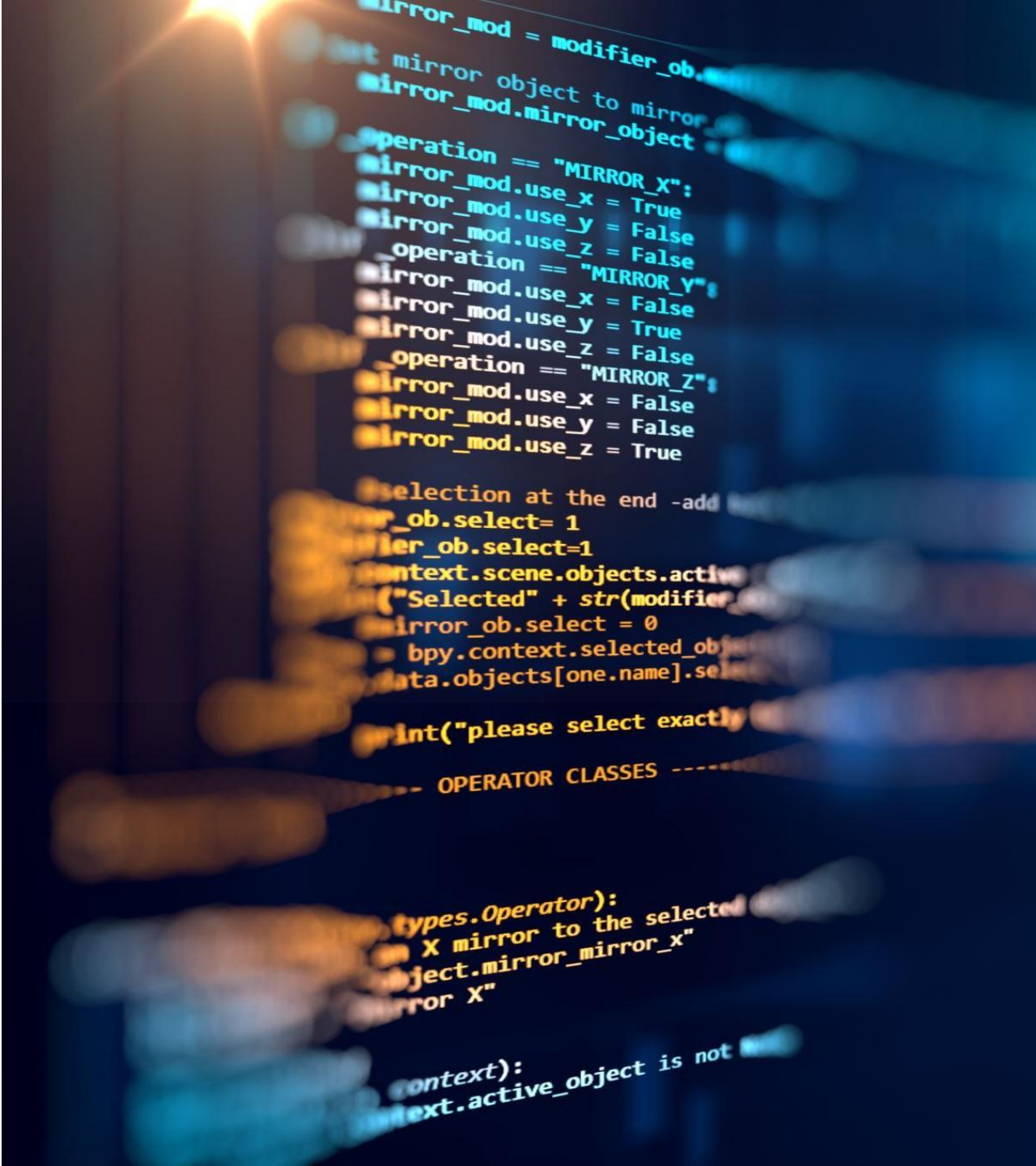
**Atualizações:** Como mencionado anteriormente, o software pode ser facilmente atualizado para adicionar novos recursos, corrigir bugs ou melhorar o desempenho.



**Escalabilidade:** O software pode ser escalado para atender a demandas crescentes, seja aumentando a capacidade de processamento, gerenciando mais usuários ou lidando com dados em expansão.

# Replicação:

- A replicação refere-se à capacidade de criar cópias idênticas do software, permitindo sua distribuição em larga escala. Isso é especialmente relevante na era digital, onde o software pode ser entregue pela internet ou por outras formas eletrônicas.





# Replicação:

- Benefícios: A replicação torna a distribuição eficiente e econômica. O software pode alcançar um grande número de usuários em todo o mundo em um curto espaço de tempo.
- Desafios: Com a replicação, também surge a necessidade de gerenciar versões, garantir a integridade das cópias distribuídas e lidar com problemas de pirataria e segurança.



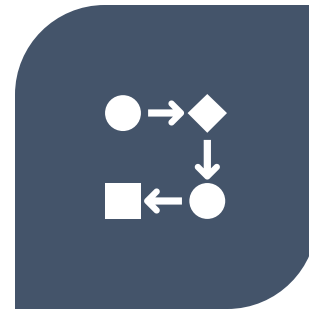
# Modelos de Processo de Software



CASCATA



INCREMENTAL



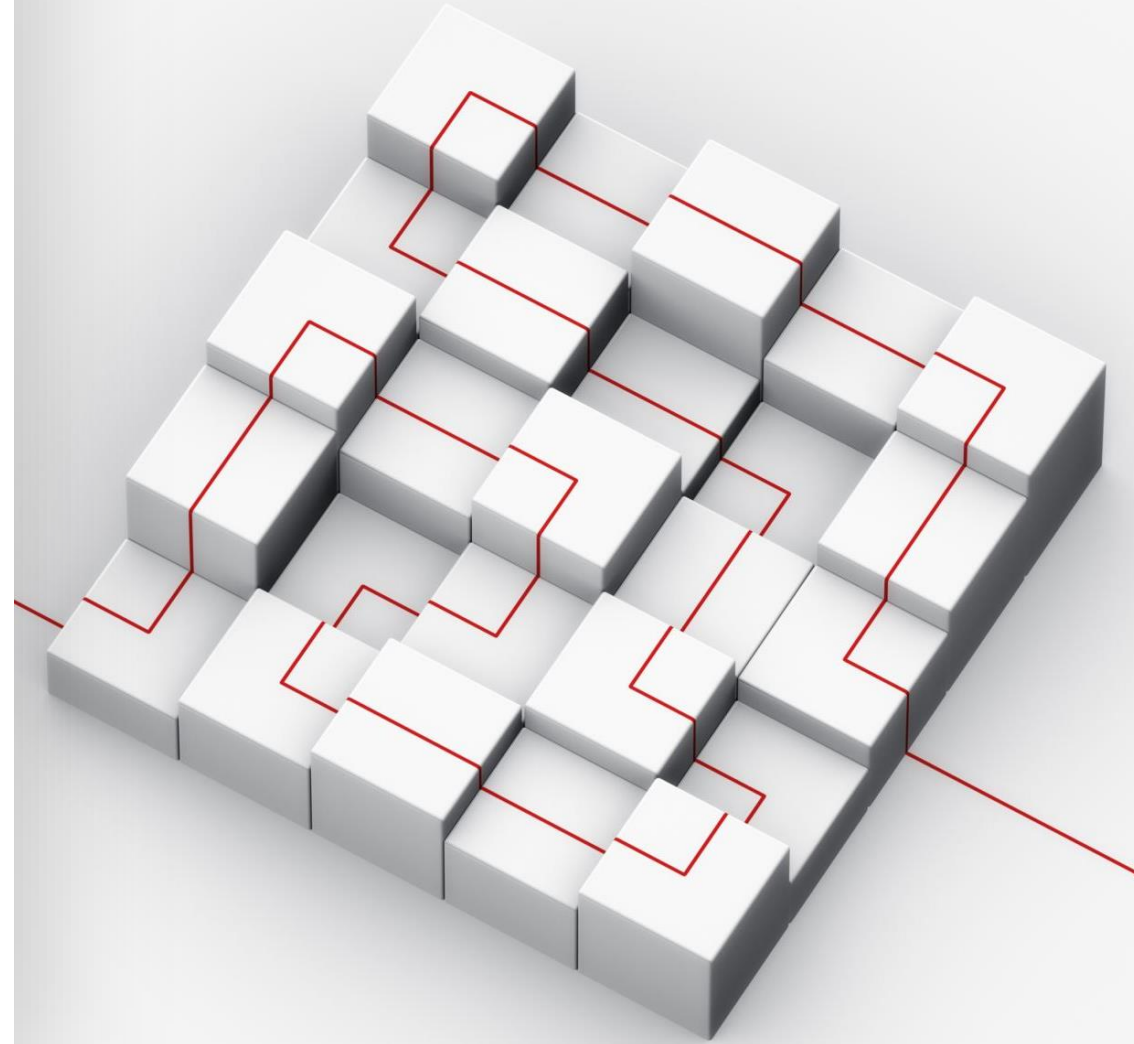
ESPIRAL



AGILE (ÁGIL)

# Modelo Cascata

- O modelo em cascata é um dos modelos mais antigos e amplamente utilizados para o desenvolvimento de software. Ele segue uma abordagem sequencial, onde cada fase do desenvolvimento é executada de forma linear, com uma fase dependendo da conclusão da fase anterior.





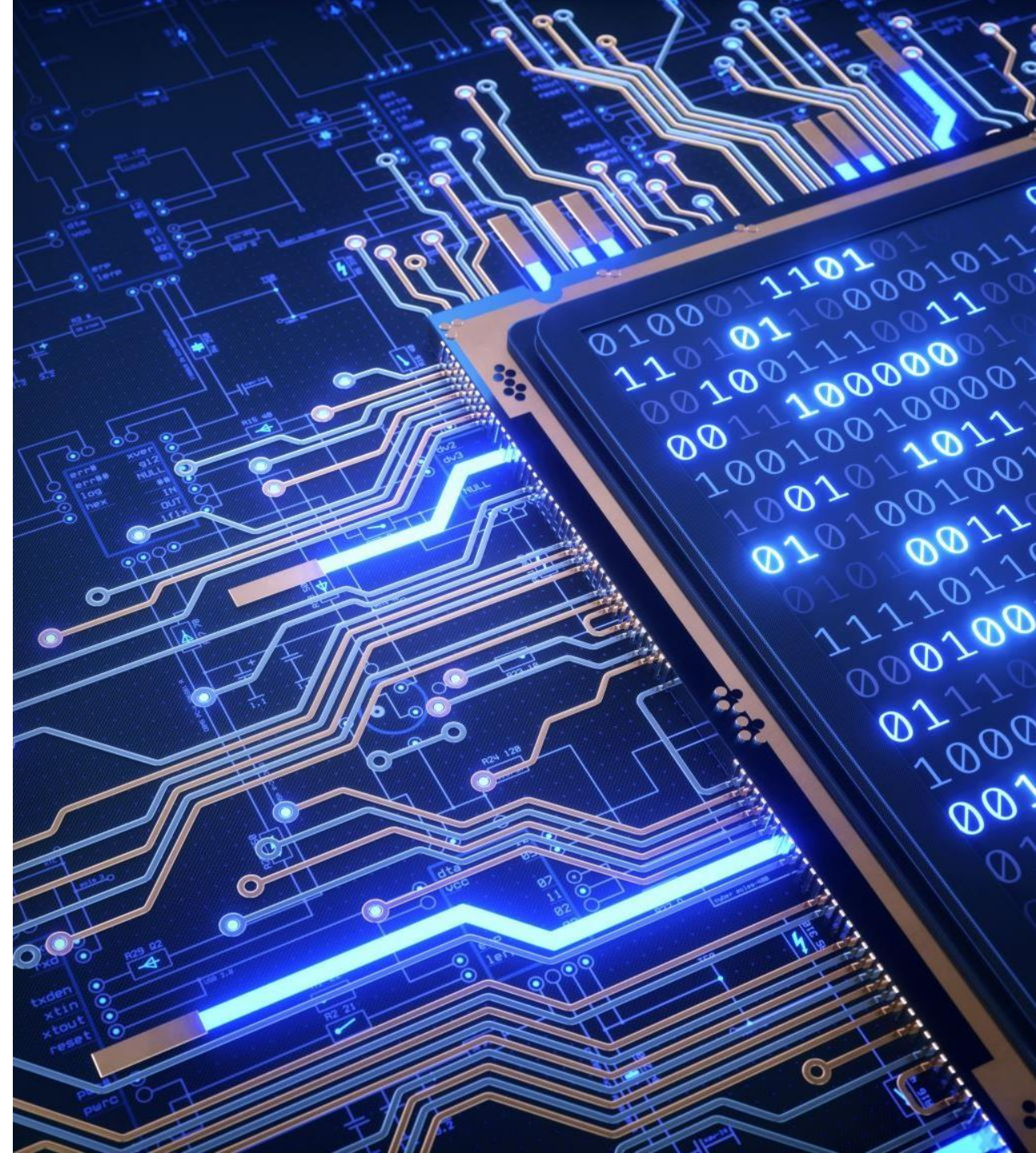
# Características:

- Sequencial: As fases do projeto são executadas em uma sequência linear e raramente se sobrepõem.
- Fases Bem Definidas: Cada fase tem objetivos claros e produz entregáveis específicos.
- Requisitos Fixos: Os requisitos do sistema são definidos no início e esperam-se poucas mudanças.
- Ênfase na Documentação: Cada fase gera documentação detalhada para guiar as fases subsequentes.



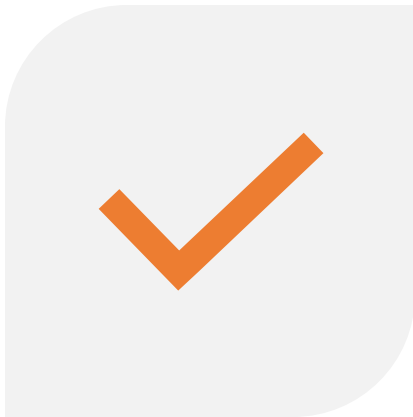
# Etapas do Modelo Cascata:

- **Requisitos:** Definição detalhada dos requisitos do sistema a serem desenvolvidos. Essa é uma das fases mais críticas, pois os requisitos fixados aqui guiarão o restante do projeto.
- **Design:** Baseado nos requisitos, é elaborado o design do sistema, incluindo a arquitetura, estruturas de dados e interfaces.
- **Implementação:** A codificação real do software é realizada, traduzindo o design em código executável.
- **Testes:** Testes rigorosos são realizados para verificar se o software atende aos requisitos especificados.
- **Integração:** Os componentes individuais do software são integrados para formar o sistema completo.
- **Manutenção:** Atividades de manutenção, correção de defeitos e atualizações podem ocorrer nesta fase.

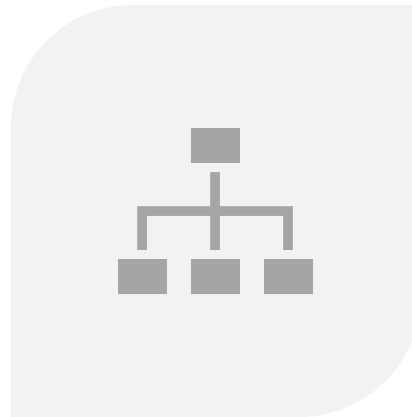




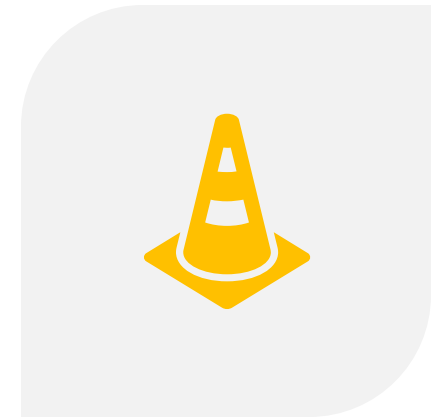
# Vantagens do Modelo Cascata:



ADEQUADO PARA PROJETOS COM  
REQUISITOS BEM DEFINIDOS E  
ESTÁVEIS.



FÁCIL DE GERENCIAR E ENTENDER,  
DEVIDO À NATUREZA LINEAR.



FORTE ÊNFASE NA DOCUMENTAÇÃO,  
O QUE É VALIOSO PARA A  
MANUTENÇÃO A LONGO PRAZO.

# Desvantagens do Modelo Cascata:



Pode ser inflexível quando os requisitos mudam, requerendo revisões significativas.



Pode levar a atrasos significativos se um problema for identificado em uma fase posterior.



Pode não ser adequado para projetos onde a necessidade do cliente evolui rapidamente.



Quando usar o Modelo Cascata:



O modelo cascata é mais apropriado para projetos onde os requisitos são bem definidos e não se espera uma mudança significativa. É especialmente útil quando a documentação detalhada é essencial, como em projetos que precisam atender a regulamentações específicas.

# Modelo Incremental

- O modelo incremental é uma abordagem que divide o desenvolvimento de um sistema em pequenas partes incrementais, chamadas de incrementos. Cada incremento representa uma versão funcional e executável do sistema que incorpora novas funcionalidades ou melhorias em relação ao incremento anterior.



# Características:

- **Iterativo:** O desenvolvimento é dividido em iterações, cada uma resultando em um incremento funcional.
- **Ciclos:** Cada ciclo consiste em uma fase de planejamento, implementação, teste e avaliação.
- **Entrega Contínua:** A cada ciclo, uma parte funcional do sistema é entregue e pode ser utilizada pelos usuários.
- **Feedback Constante:** Os usuários podem fornecer feedback após cada incremento, o que permite ajustes contínuos.





# Etapas do Modelo Incremental:

- **Requisitos Iniciais:** Os requisitos iniciais são definidos, e os principais componentes e funcionalidades do sistema são identificados.
- **Desenvolvimento dos Incrementos:** Cada incremento é desenvolvido, testado e integrado. Cada incremento adiciona novas funcionalidades ao sistema.
- **Avaliação e Feedback:** Após a entrega de cada incremento, os usuários avaliam e fornecem feedback, que é utilizado para aprimorar os incrementos subsequentes.
- **Integração:** Os incrementos são integrados para formar o sistema completo. A integração ocorre ao longo de todo o processo, mas é especialmente importante nesta fase.



---

## Vantagens do Modelo Incremental:

- **Feedback Contínuo:** Os usuários têm a oportunidade de utilizar o software em estágios iniciais e influenciar o desenvolvimento.
- **Entrega Gradual:** Partes funcionais do sistema são entregues em intervalos regulares, o que pode ser útil para cumprir prazos ou atender a necessidades imediatas.
- **Adaptação às Mudanças:** Mudanças nos requisitos podem ser incorporadas em incrementos futuros, aumentando a flexibilidade.



# Desvantagens do Modelo Incremental:



Requisitos Bem Definidos: Ainda é necessário ter uma base sólida de requisitos iniciais para começar o desenvolvimento.



Gerenciamento de Integração: A integração de múltiplos incrementos pode ser complexa e requer um cuidadoso gerenciamento.



Quando usar o Modelo Incremental:



O modelo incremental é adequado para projetos onde os requisitos não estão completamente definidos no início e quando uma abordagem de entrega gradual é preferida. Também é útil quando se deseja obter feedback constante dos usuários para melhorar o sistema. Este modelo é especialmente útil para sistemas maiores e complexos, onde a capacidade de realizar entregas intermediárias é uma vantagem.



# Modelo Espiral

- O modelo espiral é uma abordagem de desenvolvimento de software que combina elementos de abordagens sequenciais e iterativas, buscando controlar riscos e permitir a adaptação às mudanças ao longo do processo. Ele segue um ciclo de atividades que se assemelha a uma espiral, onde cada ciclo representa uma iteração.





## Características:

- **Iterativo:** O desenvolvimento ocorre em ciclos, permitindo revisões frequentes e ajustes ao longo do tempo.
- **Análise de Riscos:** Cada ciclo inclui uma análise de riscos, que ajuda a identificar e mitigar potenciais problemas antes que eles se tornem críticos.
- **Feedback Contínuo:** As iterações permitem a obtenção de feedback dos usuários, clientes ou partes interessadas.

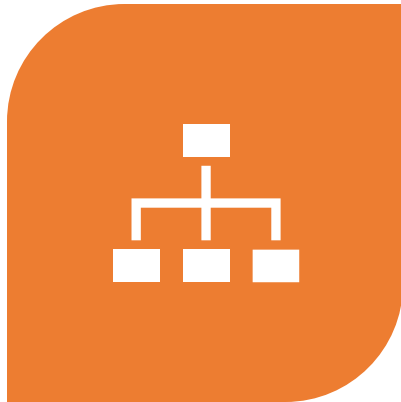
# Etapas do Modelo Espiral:

- **Planejamento:** Definição dos objetivos, identificação dos riscos e criação de um plano para a iteração.
- **Análise de Riscos:** Avaliação dos riscos identificados e planejamento de ações para mitigá-los.
- **Engenharia:** Desenvolvimento do software, com foco nas funcionalidades prioritárias.
- **Avaliação:** Revisão do progresso, avaliação do software desenvolvido e coleta de feedback.
- **Planejamento da Próxima Iteração:** Com base na avaliação e no feedback, planejamento da próxima iteração, ajustando objetivos e riscos.



# Vantagens do Modelo Espiral:

---



FLEXIBILIDADE: A ABORDAGEM ITERATIVA PERMITE A ADAPTAÇÃO A MUDANÇAS DE REQUISITOS OU SITUAÇÕES.



GESTÃO DE RISCOS: A ANÁLISE CONSTANTE DE RISCOS AJUDA A PREVER PROBLEMAS E A TOMAR MEDIDAS PROATIVAS.



FEEDBACK CONTÍNUO: AS ITERAÇÕES PERMITEM AJUSTES COM BASE NAS NECESSIDADES REAIS DOS USUÁRIOS.

# Desvantagens do Modelo Espiral:



**Complexidade:** A abordagem exige uma gestão cuidadosa das iterações e riscos, o que pode ser mais complexo em comparação com modelos mais lineares.



**Custo:** A análise contínua de riscos e as revisões podem aumentar o custo do desenvolvimento.



**Quando usar o Modelo Espiral:**



O modelo espiral é apropriado para projetos onde os requisitos são incertos ou podem evoluir ao longo do tempo. Também é útil quando a gestão de riscos é uma preocupação importante ou quando há necessidade de coletar feedback de forma contínua.

## Modelo Agile (Ágil): Abordagem Adaptativa e Iterativa para Desenvolvimento de Software

- O modelo ágil é uma abordagem moderna e flexível para o desenvolvimento de software, que valoriza a colaboração, adaptação a mudanças, entrega contínua de valor e a interação constante com os clientes e usuários. Ele enfatiza a criação de um ambiente colaborativo e a produção de software funcional em curtos períodos chamados de iterações ou sprints.



# Características:

Adaptação a Mudanças:  
Acomodar mudanças nos  
requisitos, prioridades e  
cenários em todas as fases  
do projeto.

Iterativo e Incremental: O  
desenvolvimento ocorre em  
ciclos curtos, produzindo  
entregas incrementais de  
funcionalidades.

Colaboração Intensa:  
Interação frequente entre  
desenvolvedores, clientes,  
usuários e demais partes  
interessadas.

Foco no Valor: Priorização  
constante de funcionalidades  
que trazem o maior valor ao  
cliente.

# Práticas Ágeis Fundamentais:

- **Scrum:** Um framework que organiza o trabalho em sprints, define papéis (Scrum Master, Product Owner, Time de Desenvolvimento) e promove reuniões regulares (Daily Standup, Sprint Planning, Sprint Review, Retrospectiva).
- **Kanban:** Uma abordagem visual que permite o gerenciamento do fluxo de trabalho, com ênfase na redução de gargalos e melhoria contínua.
- **User Stories:** Pequenas descrições das funcionalidades a partir da perspectiva do usuário, ajudando a manter o foco nas necessidades reais.
- **Testes Contínuos:** Integração de testes automatizados para garantir a qualidade do software desde o início.

# Vantagens do Modelo Ágil:

**Flexibilidade:** Adapta-se rapidamente a mudanças, permitindo que o produto evolua conforme a necessidade do cliente.

**Entrega Contínua:** Entregas frequentes de software funcional, possibilitando o feedback precoce dos usuários.

**Colaboração Melhorada:** A colaboração constante entre a equipe e os stakeholders resulta em um melhor entendimento das necessidades e expectativas.



# Desafios do Modelo Ágil:



Necessidade de Engajamento: A colaboração contínua exige um alto nível de envolvimento dos clientes e das partes interessadas.



Aprendizado Contínuo: A equipe deve estar disposta a aprender, adaptar-se e melhorar constantemente suas práticas.



Quando usar o Modelo Ágil:



O modelo ágil é mais adequado para projetos em que os requisitos são voláteis, a inovação é necessária, a entrega rápida é valorizada e há a disponibilidade de envolvimento contínuo das partes interessadas.

# Exercícios e Discussão sobre Metodologias Ágeis



ATIVIDADES:



EXERCÍCIOS PRÁTICOS  
RELACIONADOS AO  
DESENVOLVIMENTO ÁGIL.



DISCUSSÃO EM GRUPO SOBRE AS  
VANTAGENS E DESAFIOS DAS  
METODOLOGIAS ÁGEIS.

# Por que a Agilidade é Importante?



Mudanças frequentes nas necessidades dos clientes.



Adaptação a um mercado em constante evolução.