ARQUITETURA DE SOFTWARE

João Choma Neto

joao.choma@unicesumar.edu.br

https://github.com/JoaoChoma/arquitetura-software

Unicesumar – Maringá

MODELOS DE PROCESSO DE SOFTWARE

Modelos de processo de software

É uma **descrição simplificada** do processo e representa um processo sob **determinada perspectiva**.

São utilizados para explicar diferentes abordagens do desenvolvimento de software.

Modelos genéricos:

- Modelo Cascata,
- Modelo Evolucionário,
- Engenharia de Software Baseada em Componente.

Modelo Cascata

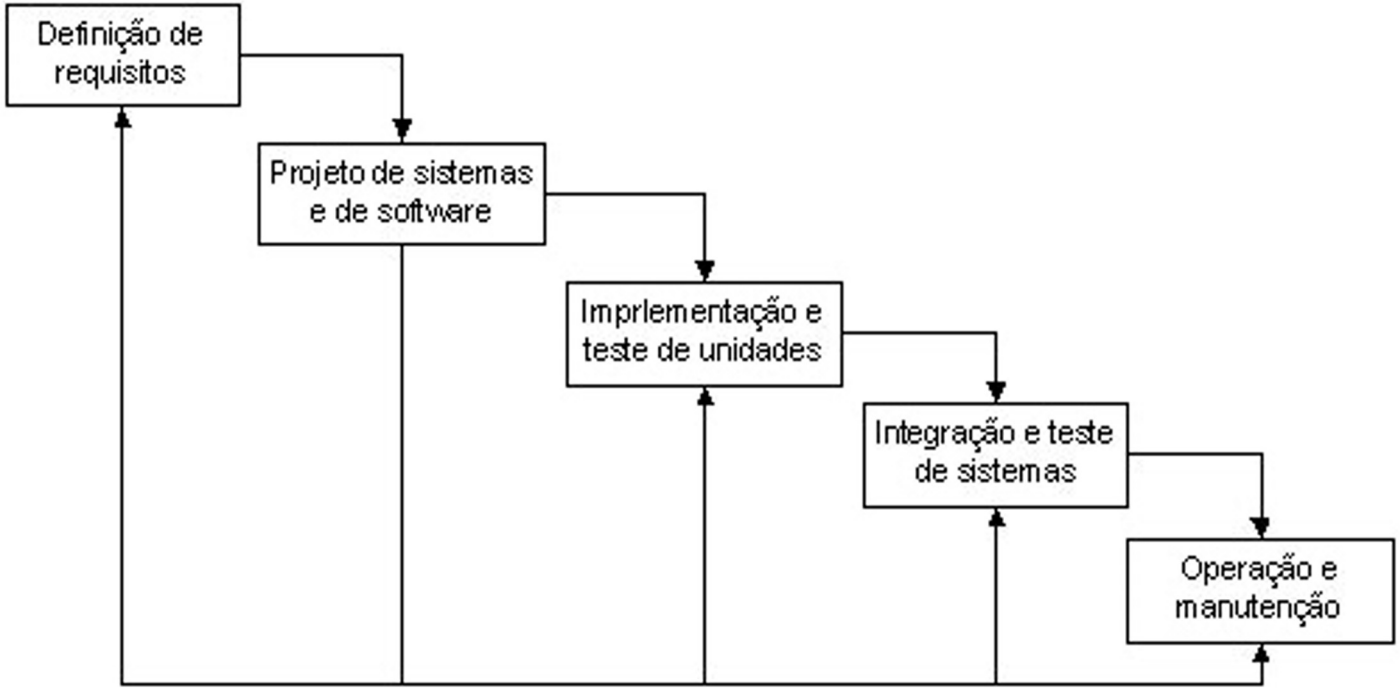
Primeiro modelo a **organizar** as atividades de desenvolvimento de software.

Atividades:

(1) Análise e definição de requisitos,
(2) Projeto de sistema,
(3) Implementação e teste de unidade,
(4) Integração e teste de sistema,
(5) Operação e manutenção.

A fase seguinte não deve começar antes que a fase anterior tenha terminado.

Modelo Cascata



Modelo Cascata

Desvantagens (problemas):

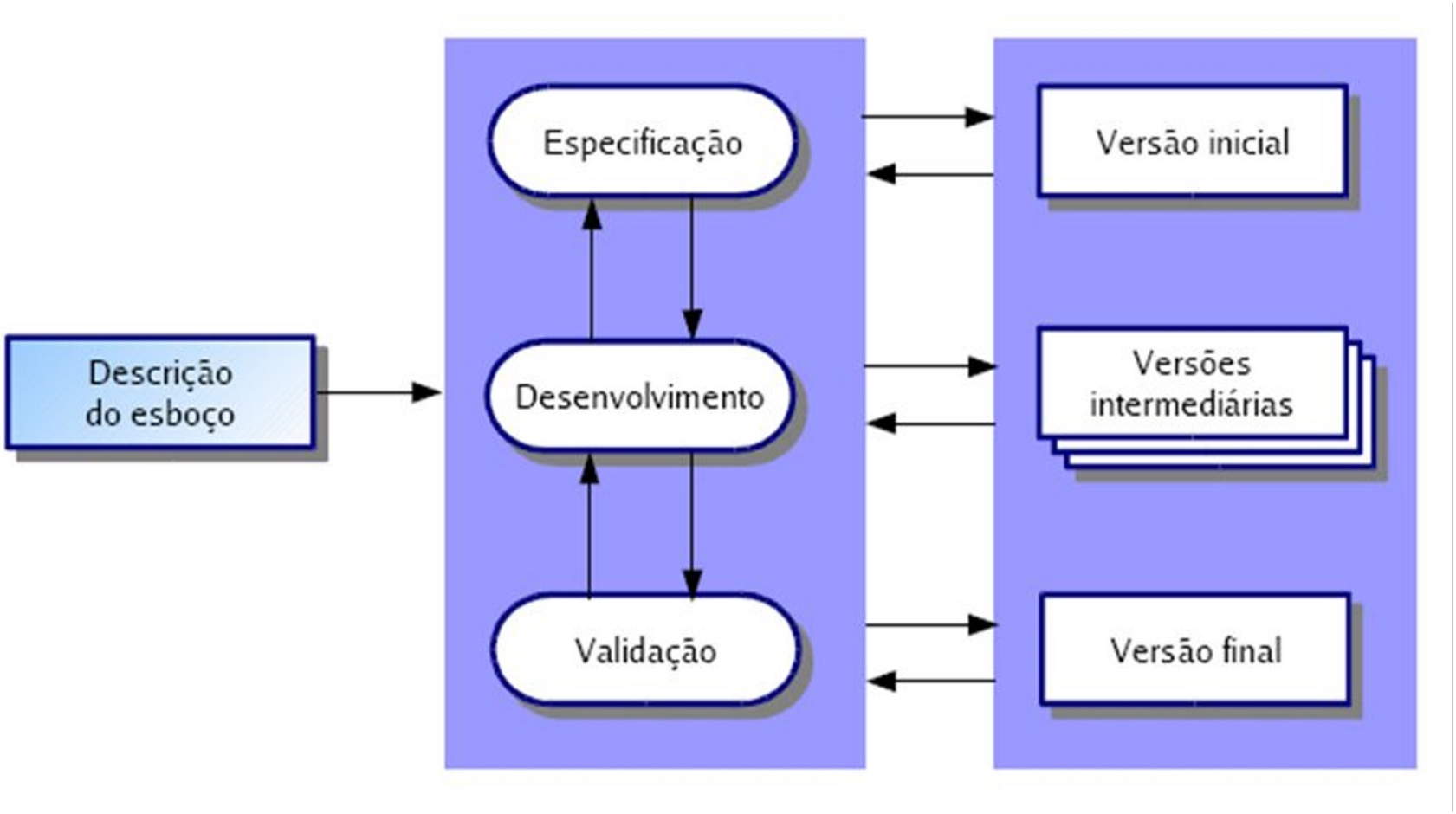
1. O resultado de cada fase envolve um ou mais documentos que são aprovados, gerando **muita documentação**.
2. A fase seguinte não deve iniciar até que a fase precedente tenha sido concluída.
3. Particionamento inflexível do projeto em estágios.
4. Apropriado somente quando os requisitos são bem compreendidos e as mudanças são raras.

Modelo Evolucionário

Desenvolvimento de uma implementação inicial, exposição do resultado aos comentários do usuário e refinamento do resultado por meio de várias versões.

As atividades de especificação, desenvolvimento e validação são intercaladas.

Modelo Evolucionário



Modelo Evolucionário

Vantagens:

- Os sistemas atendem às necessidades imediatas dos clientes.
- A especificação pode ser desenvolvida de forma incremental (usuários compreendem melhor o problema).

Modelo Evolucionário

Desvantagens:

- O progresso é medido por meio dos produtos entregues.
- A mudança contínua tende a corromper a estrutura do software.
- A incorporação de mudanças torna-se cada vez mais difícil e onerosa.

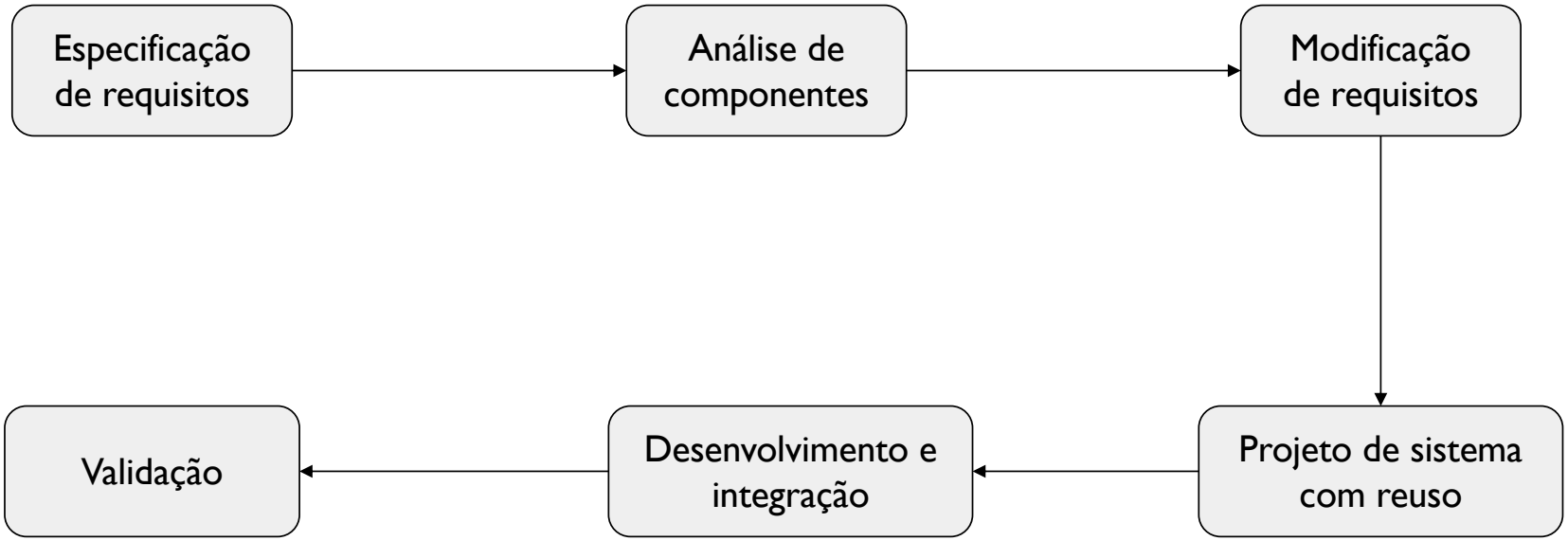
Engenharia de software baseada em componentes

Os sistemas são "desenvolvidos" a partir de componentes existentes ou sistemas COTS (Commercial-of-the-shelf).

Atividades:

(1) Análise de componentes,
(2) Modificação de requisitos,
(3) Projeto de sistema com reuso,
(4) Desenvolvimento e integração.

Engenharia de software baseada em componentes



Engenharia de software baseada em componentes

Vantagens:

- Redução da quantidade de software a ser desenvolvida, reduzindo os custos e riscos.
- Proporciona a entrega mais rápida do software.

Engenharia de software baseada em componentes

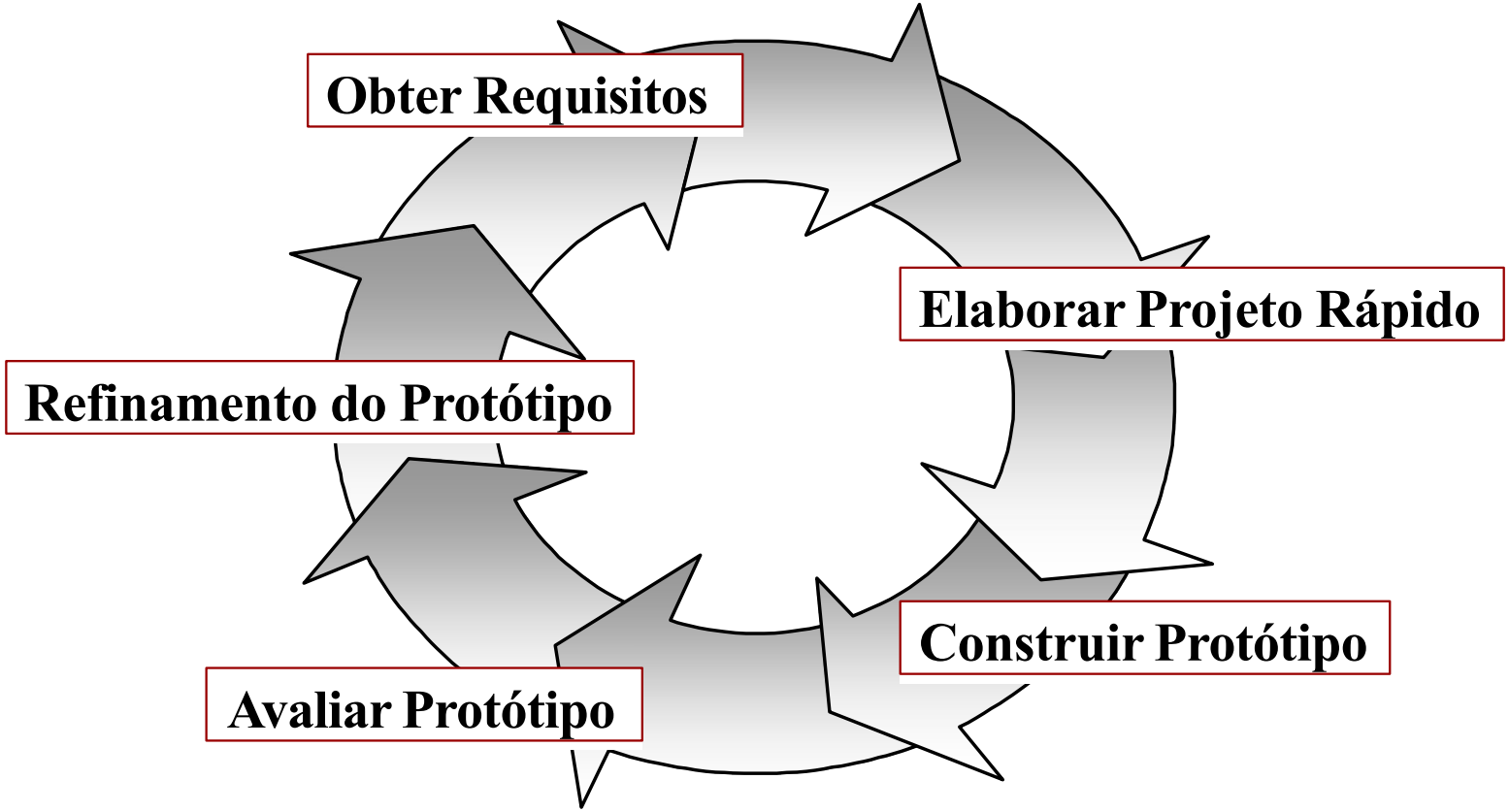
Desvantagens:

- As adequações nos requisitos são inevitáveis, podendo resultar num software que não atende às reais necessidades dos usuários.
- A evolução do software depende de novas versões dos componentes reutilizáveis, não estando sob o controle da organização que utiliza os componentes.

O Modelo de Prototipação

- o objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.
- possibilita que o desenvolvedor crie um modelo (protótipo)do software que deve ser construído
- apropriado para quando o cliente não definiu detalhadamente os requisitos.

O Paradigma de Prototipação
para obtenção dos requisitos



O Modelo de Prototipação

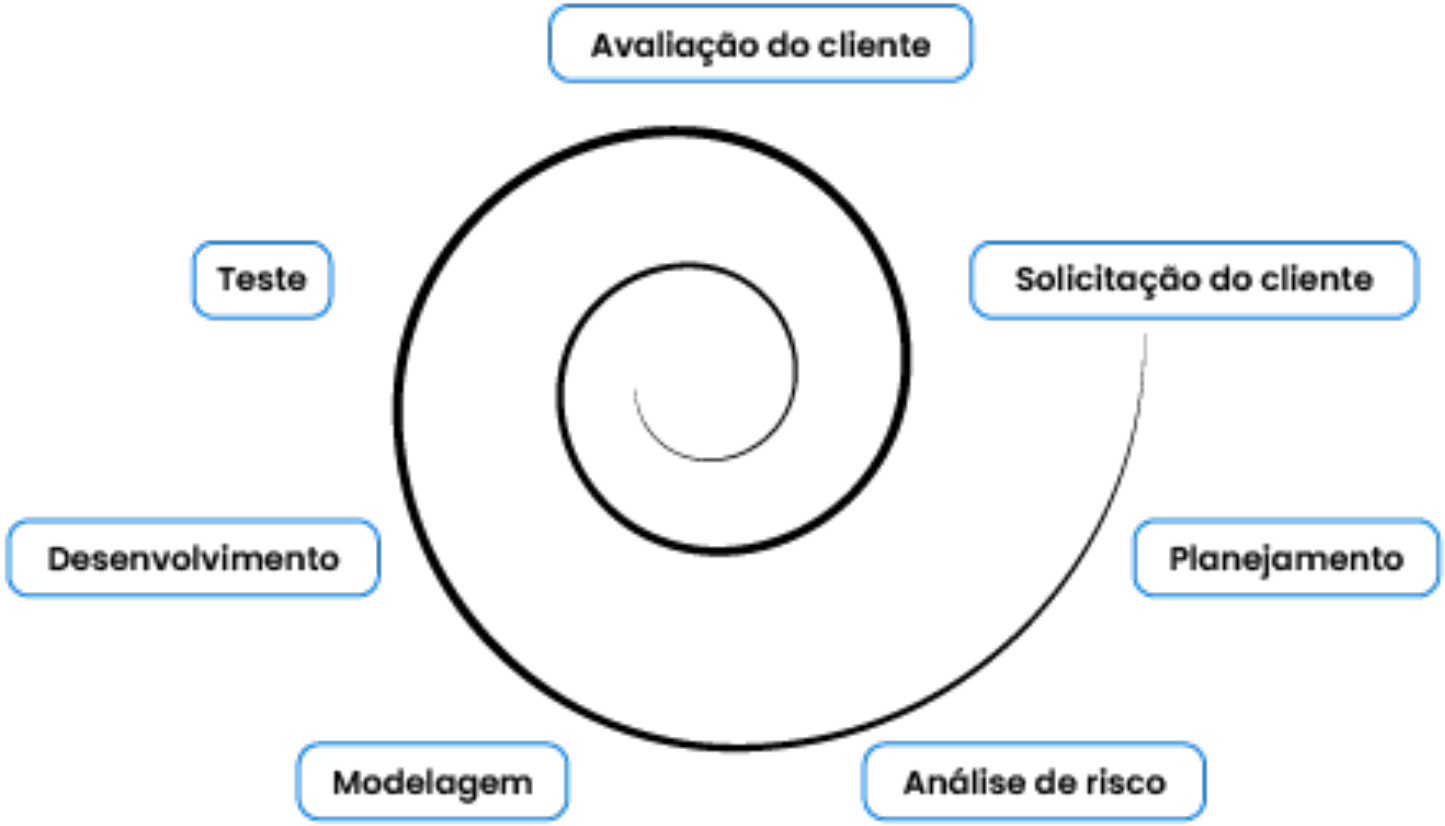
- o objetivo é entender os requisitos do usuário e, assim, obter uma melhor definição dos requisitos do sistema.
- possibilita que o desenvolvedor crie um modelo (protótipo)do software que deve ser construído
- apropriado para quando o cliente não definiu detalhadamente os requisitos.

Modelos Evolutivos de Processo

- modelos evolutivos são iterativos
- possibilitam o desenvolvimento de versões cada vez mais completas do software

O Modelo Espiral

- O modelo espiral acopla a natureza iterativa da prototipação com os aspectos controlados e sistemáticos do modelo cascata.
- O modelo espiral é dividido em uma série de *atividades de trabalho* ou *regiões de tarefa*.
- Existem tipicamente de 3 a 6 regiões de tarefa


Avaliação do cliente
Solicitação do cliente
Planejamento
Análise de risco
Modelagem
Desenvolvimento
Teste

METODOLOGIA ÁGIL

- A metodologia ágil é uma abordagem de desenvolvimento de software que se baseia em princípios e valores que valorizam a colaboração, a flexibilidade, a entrega contínua de valor e a adaptação às mudanças

METODOLOGIA ÁGIL

- As metodologias ágeis têm como expectativa uma resposta mais rápida às necessidades dos clientes e às mudanças nos requisitos

PRINCÍPIOS

- **Colaboração e Comunicação:** As metodologias ágeis enfatizam a colaboração próxima entre os membros da equipe de desenvolvimento, bem como a comunicação regular com os clientes e partes interessadas. Isso ajuda a garantir que todos tenham uma compreensão clara dos objetivos e requisitos do projeto.

PRINCÍPIOS

- **Entrega Contínua de Valor:** Em vez de esperar até o final de um longo ciclo de desenvolvimento para entregar um produto, as metodologias ágeis promovem a entrega contínua de incrementos de valor para os clientes. Isso significa que partes utilizáveis do software são entregues em intervalos regulares.

PRINCÍPIOS

- **Flexibilidade e Adaptação:** As metodologias ágeis reconhecem que os requisitos e as prioridades podem mudar ao longo do tempo. Elas permitem que as equipes se adaptem a essas mudanças de forma eficaz, ajustando o trabalho conforme necessário.

PRINCÍPIOS

- **Iteração e Feedback:** As metodologias ágeis frequentemente usam ciclos curtos de desenvolvimento, chamados de "iterações" ou "sprints", nos quais uma parte do software é desenvolvida e depois revisada. Isso permite que a equipe receba feedback regular e faça melhorias contínuas.

PRINCÍPIOS

- **Pessoas mais que Processos e Ferramentas:** Embora processos e ferramentas sejam importantes, as metodologias ágeis valorizam mais as pessoas e suas interações. Acredita-se que equipes motivadas e colaborativas são fundamentais para o sucesso.

PRINCÍPIOS

- **Trabalho em Equipe Auto-organizada:** As equipes ágeis são frequentemente auto-organizadas, o que significa que têm um grau de autonomia para tomar decisões relacionadas ao projeto. Isso promove a responsabilidade e a motivação da equipe.

METODOLOGIA ÁGIL

- Frameworks e abordagens ágeis mais conhecidos incluem Scrum, Kanban, Extreme Programming (XP)

MANIFESTO ÁGIL

- https://agilemanifesto.org/iso/ptbr/manifesto.html

ARQUITETURA DE SOFTWARE

João Choma Neto

joao.choma@unicesumar.edu.br

https://github.com/JoaoChoma/arquitetura-software

Unicesumar – Maringá