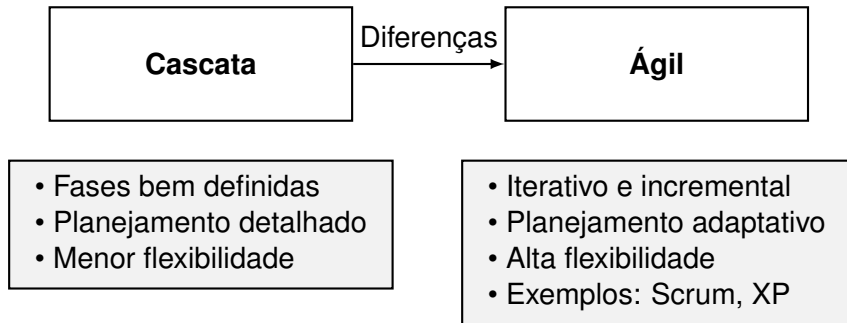


Modelo Ágil - Exemplos

Prof. Dr. William Simão de Deus

william.deus@ifpr.edu.br
Instituto Federal do Paraná (IFPR)
Campus Pinhais
Gestão da Tecnologia da Informação – 2025.01



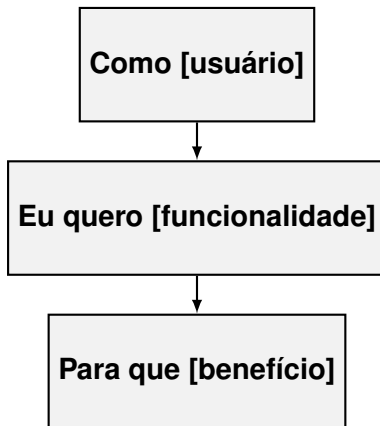
- Na XP (Extreme Programming), os requisitos são expressos em cenários chamados de "histórias do usuário"

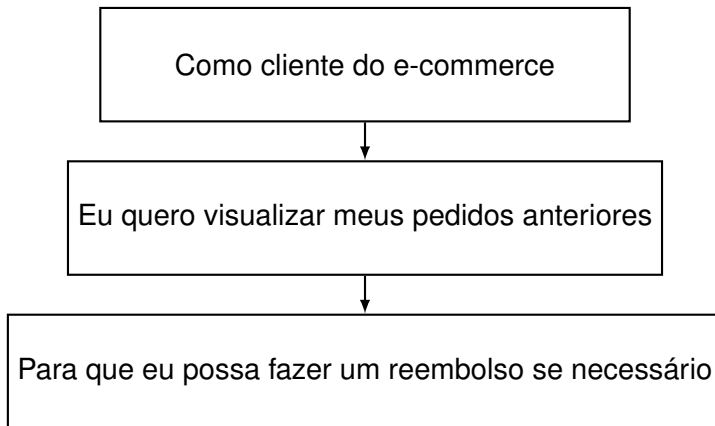
- Na XP (Extreme Programming), os requisitos são expressos em cenários chamados de "histórias do usuário"
- As histórias do usuário são implementadas diretamente como uma série de tarefas

- Na XP (Extreme Programming), os requisitos são expressos em cenários chamados de "histórias do usuário"
- As histórias do usuário são implementadas diretamente como uma série de tarefas
- Os programadores trabalham em pares

- Na XP (Extreme Programming), os requisitos são expressos em cenários chamados de "histórias do usuário"
- As histórias do usuário são implementadas diretamente como uma série de tarefas
- Os programadores trabalham em pares
- Testes são desenvolvidos para cada tarefa antes de escrever o código

- Na XP (Extreme Programming), os requisitos são expressos em cenários chamados de "histórias do usuário"
- As histórias do usuário são implementadas diretamente como uma série de tarefas
- Os programadores trabalham em pares
- Testes são desenvolvidos para cada tarefa antes de escrever o código
- Todos os testes devem ser executados com sucesso quando o novo código é integrado ao sistema





- **Colaboração constante:** Dois programadores trabalham juntos em um único computador, compartilhando a responsabilidade pelo código
- **Divisão de papéis:** Um programador escreve o código (motorista) enquanto o outro revisa continuamente (navegador)
- **Revezamento de papéis:** Os programadores trocam de papéis regularmente, garantindo que ambos estejam envolvidos ativamente no processo

- **Qualidade do código:** O feedback contínuo do parceiro ajuda a identificar erros rapidamente, aumentando a qualidade do código produzido
- **Design e soluções melhores:** A colaboração direta leva a decisões de design mais cuidadosas e soluções criativas para os problemas
- **Compartilhamento de conhecimento:** A prática promove a troca de conhecimentos e habilidades entre os programadores, melhorando o nível técnico da equipe

(SOMMERVILLE, 2011)

- *Não estou convencido de que a Programação Extrema seja um método ágil prático para a maioria das empresas, mas sua contribuição mais importante é, provavelmente, o conjunto de práticas de desenvolvimento ágil que introduziu na comunidade.*

- História de usuários: decompor as histórias e definir o esforço.
Capacidade de priorizar tarefas
- Refatoração: apesar de ser uma prática importante, muitas vezes isso não é realizado em projetos de software
 - Códigos redundantes
 - Funções semelhantes
 - Dependências
 - Códigos sem utilização

- Desenvolvimento com testes a priori: desenvolver casos de testes ao invés de apenas codificar o software
- Programação em Pares: revisão informal já que cada linha de código é examinada por ao menos 2 pessoas

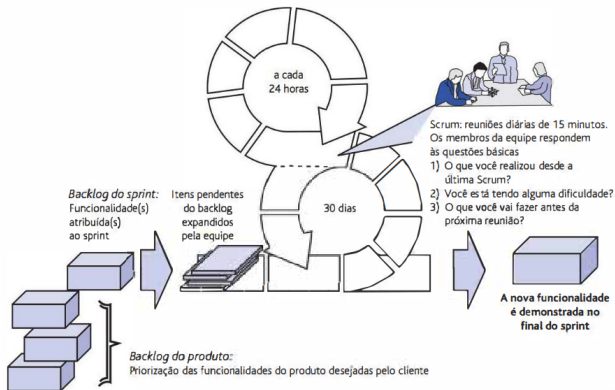


FIGURA 5.3 Fluxo do processo Scrum.

Figura: Fonte: (PRESSMAN, 2005)

- Scrum é um framework ágil para gerenciamento e desenvolvimento de produtos

- Scrum é um framework ágil para gerenciamento e desenvolvimento de produtos
- Baseado na teoria empírica de controle de processo, que enfatiza a transparência, comunicação e adaptação

- Scrum é um framework ágil para gerenciamento e desenvolvimento de produtos
- Baseado na teoria empírica de controle de processo, que enfatiza a transparência, comunicação e adaptação
- Utiliza iterações curtas e entregas frequentes (chamadas de sprints) para promover o progresso contínuo.

- **Product Owner:** Responsável por maximizar o valor do produto e gerenciar o backlog do produto

- **Product Owner:** Responsável por maximizar o valor do produto e gerenciar o backlog do produto
- **Scrum Master:** Garante que o Scrum seja entendido e seguido pela equipe, removendo impedimentos e facilitando o processo

- **Product Owner:** Responsável por maximizar o valor do produto e gerenciar o backlog do produto
- **Scrum Master:** Garante que o Scrum seja entendido e seguido pela equipe, removendo impedimentos e facilitando o processo
- **Equipe de Desenvolvimento:** Equipe multifuncional que cria o incremento de produto no final de cada sprint.

- **Product Backlog:** Lista priorizada de tudo que é necessário no produto. Mantido pelo Product Owner

- **Product Backlog:** Lista priorizada de tudo que é necessário no produto. Mantido pelo Product Owner
- **Sprint Backlog:** Itens selecionados do Product Backlog para a sprint atual, junto com um plano para entregar o incremento

- **Product Backlog:** Lista priorizada de tudo que é necessário no produto. Mantido pelo Product Owner
- **Sprint Backlog:** Itens selecionados do Product Backlog para a sprint atual, junto com um plano para entregar o incremento
- **Incremento:** O produto entregue ao final de cada sprint, que deve ser funcional e potencialmente utilizável

- **Sprint:** Período de tempo fixo (geralmente 2 a 4 semanas) durante o qual o trabalho é realizado

- **Sprint:** Período de tempo fixo (geralmente 2 a 4 semanas) durante o qual o trabalho é realizado
- **Sprint Planning:** Reunião no início da sprint para definir o que será entregue e como o trabalho será feito

- **Sprint:** Período de tempo fixo (geralmente 2 a 4 semanas) durante o qual o trabalho é realizado
- **Sprint Planning:** Reunião no início da sprint para definir o que será entregue e como o trabalho será feito
- **Daily Scrum:** Reunião diária de 15 minutos para sincronizar as atividades da equipe e planejar o próximo dia

- **Sprint Review:** Reunião no final da sprint para inspecionar o incremento e adaptar o backlog do produto

- **Sprint Review:** Reunião no final da sprint para inspecionar o incremento e adaptar o backlog do produto
- **Sprint Retrospective:** Reunião no final da sprint para identificar melhorias no processo e no trabalho em equipe

- **Adaptabilidade:** Capacidade de reagir a mudanças rapidamente
- **Entrega frequente:** Incrementos regulares garantem que o produto esteja sempre evoluindo
- **Colaboração:** Times autônomos e multifuncionais promovem a colaboração eficaz
- **Foco no valor:** Prioridade dada a itens que trazem maior valor ao cliente

- Criado por Alistair Cockburn em 1997 (WAZLAWICK, 2019)
- Voltado para equipes pequenas
- Segundo (WAZLAWICK, 2019), uma equipe geralmente é composta por:
 - Um designer líder
 - Dois a sete programadores

(WAZLAWICK, 2019):

- Uso de **radiadores de informação**, como quadros e murais à vista de todos

(WAZLAWICK, 2019):

- Uso de **radiadores de informação**, como quadros e murais à vista de todos
- Acesso fácil a **especialistas de domínio**

(WAZLAWICK, 2019):

- Uso de **radiadores de informação**, como quadros e murais à vista de todos
- Acesso fácil a **especialistas de domínio**
- **Eliminação de distrações** no ambiente de trabalho

(WAZLAWICK, 2019):

- Uso de **radiadores de informação**, como quadros e murais à vista de todos
- Acesso fácil a **especialistas de domínio**
- **Eliminação de distrações** no ambiente de trabalho
- **Cronograma de desenvolvimento** baseado na técnica de timeboxing

(WAZLAWICK, 2019):

- Uso de **radiadores de informação**, como quadros e murais à vista de todos
- Acesso fácil a **especialistas de domínio**
- **Eliminação de distrações** no ambiente de trabalho
- **Cronograma de desenvolvimento** baseado na técnica de timeboxing
- Ajuste do método conforme necessário durante o processo

- O ciclo de vida é organizado em três níveis (WAZLAWICK, 2019):
 - 1 **Iteração:** Estimação, desenvolvimento e celebração; dura poucas semanas
 - 2 **Entrega:** Formada por várias iterações, entrega funcionalidades úteis ao cliente no máximo em dois meses
 - 3 **Projeto:** Conjunto de todas as entregas

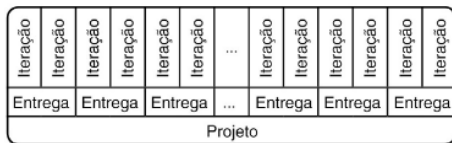


Figura 4.8 Estrutura do ciclo de vida do *Crystal Clear*.

Figura: Fonte: (WAZLAWICK, 2019)

De acordo com (WAZLAWICK, 2019), os três principais pilares do Crystal Clear são:

- No máximo em dois meses, a equipe entrega funcionalidades úteis ao cliente
- Discussão frequente do rumo do projeto e comunicação sobre descobertas
- Comunicação osmótica: a equipe deve trabalhar em uma única sala para que uns possam ouvir a conversa dos outros e participar dela quando julgarem conveniente
 - Considera-se uma boa prática interferir no trabalho dos outros
 - Os programadores devem trabalhar individualmente, mas bem próximos uns dos outros

De acordo com (WAZLAWICK, 2019), os quatro pilares recomendados do Crystal Clear são:

- **Segurança pessoal:** certeza de que os participantes do projeto poderão falar sem medo de repreensões
- **Foco:** Dois ou três tópicos de alta prioridade, sem receber novas atribuições
- **Acesso fácil a especialistas:** especialistas devem estar disponíveis para colaborar com a equipe de desenvolvimento
- **Ambiente tecnologicamente rico:** o ambiente deve permitir testes automáticos, gerenciamento de configuração e integração frequente

Dúvidas?

Conteúdo produzido com base nos materiais da profa. Eliana.

 PRESSMAN, R. S. Software engineering: a practitioner's approach. *Pressman and Associates*, 2005.

 SOMMERVILLE, I. *Engenharia de Software - 9 ed.* [S.l.]: Pearson, 2011.

 WAZLAWICK, R. *Engenharia de software: conceitos e práticas.* [S.l.]: Elsevier Editora Ltda., 2019.