



UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO



Engenharia de Software I

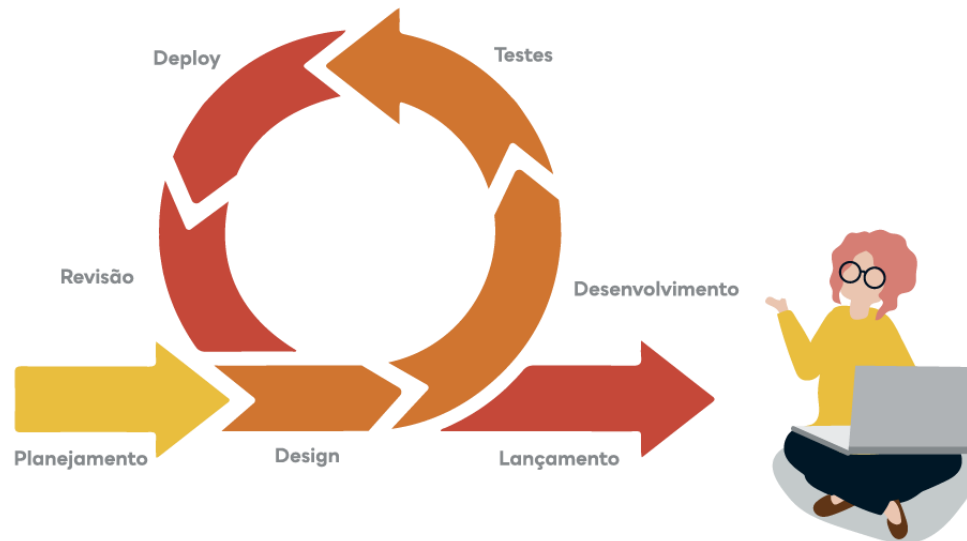
Processos Ágeis

Professora Pâmela Carvalho

10/01/23

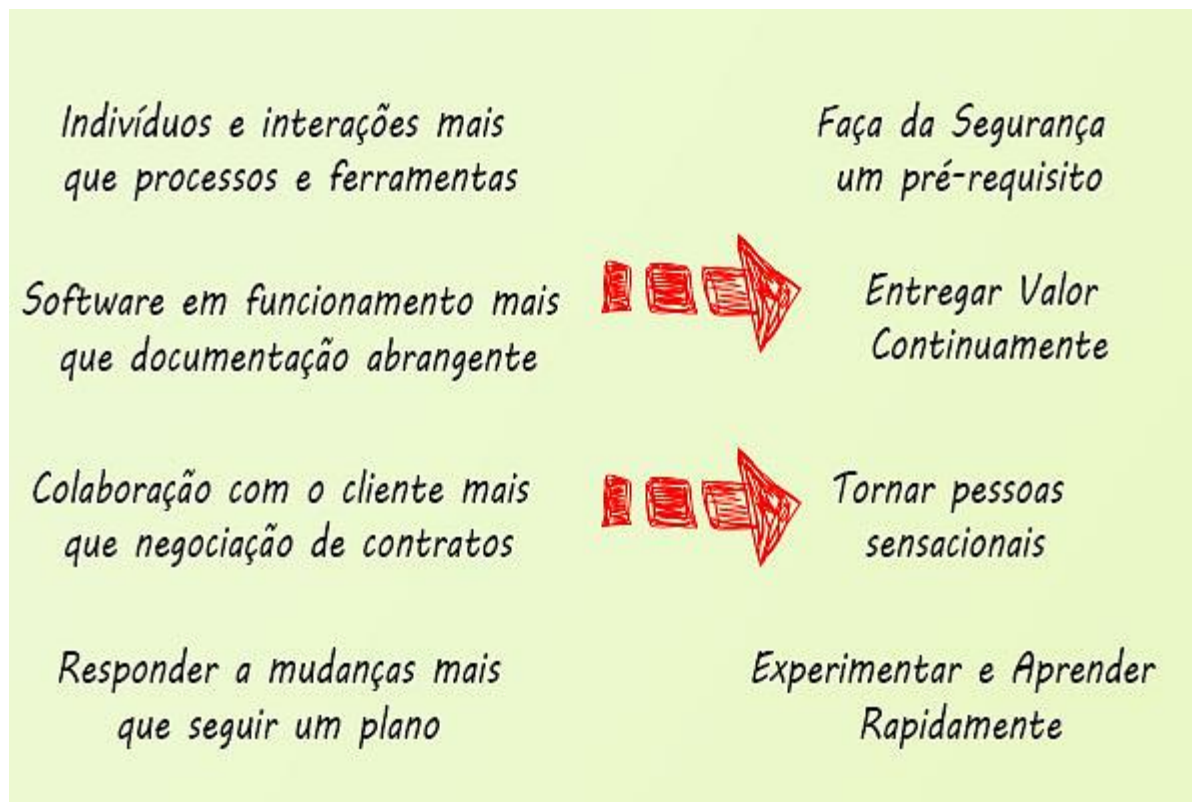
Como surgiram as metodologias ágeis

- Em 2001 Kent Beck, e mais 16 reconhecidos desenvolvedores, se reuniram para estipularem o “***Manifesto for Agile Software Development***” (Manifesto para Desenvolvimento Ágil de Software).



Idéias defendidas pelo manifesto ágil

“Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:



Princípios básicos do Manifesto Ágil



Os 12 princípios ágeis



1. Priorize o Cliente



2. Mudanças são bem-vindas



3. Entregas que geram valor



4. Cooperação entre squads e stakeholders



5. Confie e apoie



6. Converse face a face



7. Priorize o que funciona



8. Desenvolvimento incremental



9. Excelência técnica



10. Mantenha a simplicidade

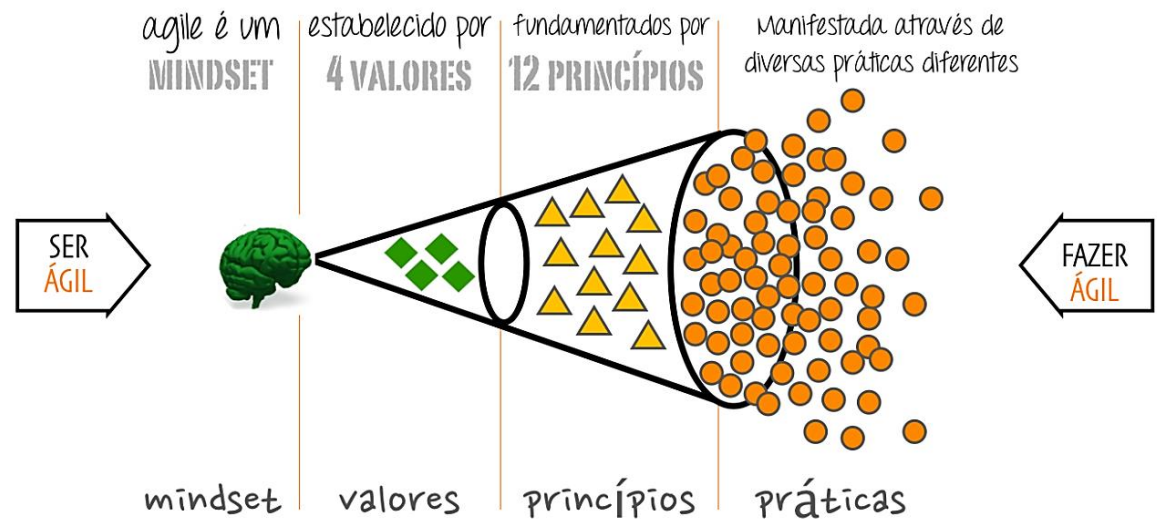


11. Times autônomos e auto-gerenciáveis



12. Retrospectiva e planejamento

Princípios básicos do Manifesto Ágil (1)



- Simplicidade acima de tudo;
- Rápida adaptação incremental às mudanças;
- Desenvolvimento do software preza pela excelência técnica;
- Projetos de sucesso surgem através de indivíduos motivados, e com uma relação de confiança entre eles;
- Desenvolvedores cooperam constantemente e trabalham junto com os usuários/clientes;

Princípios básicos do Manifesto Ágil (2)

- Atender o usuário/cliente, entregando rapidamente e continuamente produtos funcionais em curto espaço de tempo (normalmente a cada 2 semanas);
- Software funcionando é a principal medida de progresso;
- Mudanças no escopo, ou nos requisitos, do projeto não é motivo de chateação;
- A equipe de desenvolvimento se auto-organiza, fazendo ajustes constantes em melhorias.

Metodologia Ágil

Um dos pontos de destaque na Metodologia Ágil é a liberdade dada para as equipes de desenvolvimento.

“A equipe seleciona quanto trabalho acredita que pode realizar dentro da iteração, e a equipe se compromete com o trabalho. Nada desmotiva tanto uma equipe quanto alguém de fora assumir compromissos por ela. Nada motiva tanto uma equipe quanto a aceitação das responsabilidades de cumprir os compromissos que ela própria estabeleceu”. (Ken Schwaber)

Metodologias Ágeis

XP (Extreme Programming)

SCRUM

XP

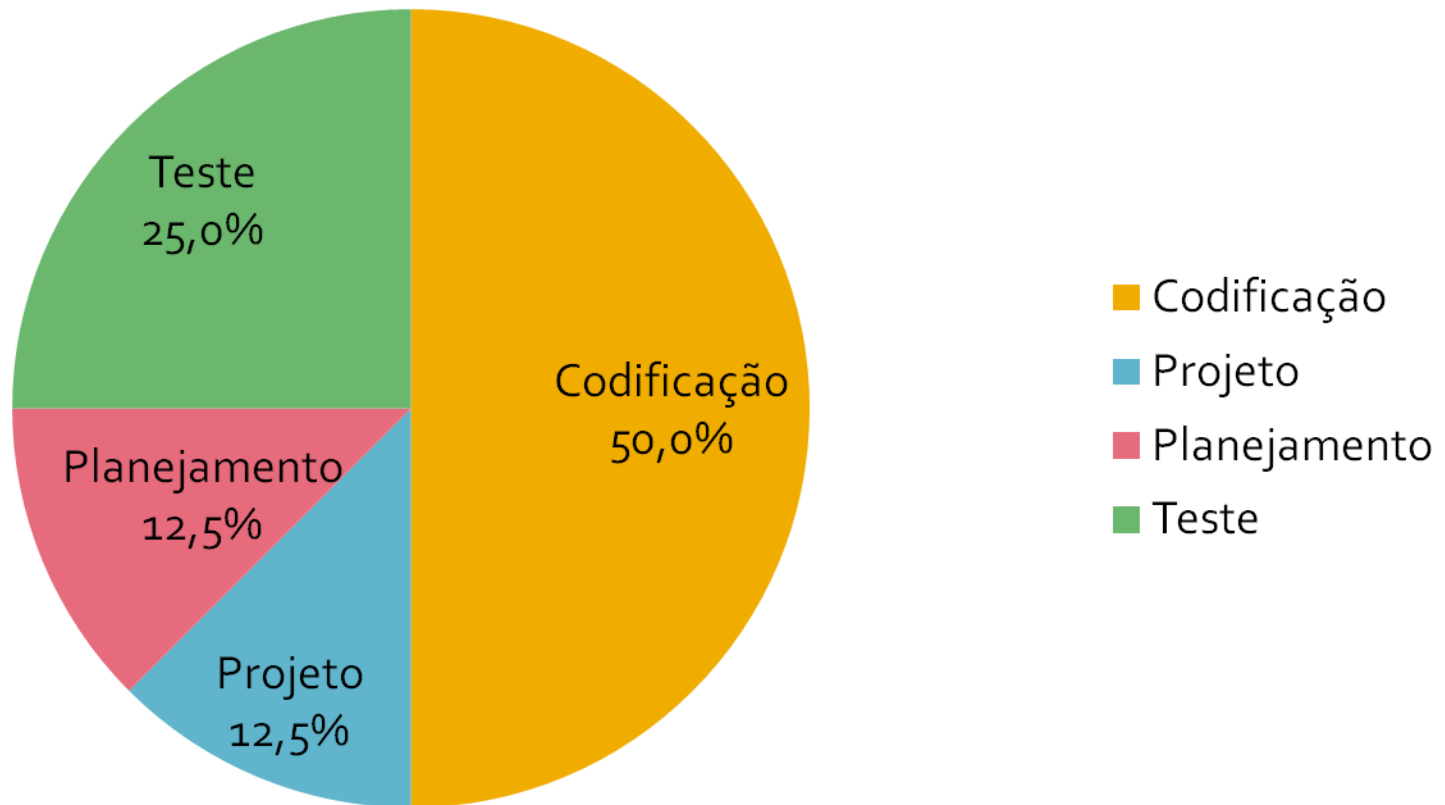


XP (Extreme Programming)

- A Programação Extrema é uma das metodologias ágeis mais conhecidas. Foi criada por Kent Beck.
- Baseada em cinco **valores**, alguns **princípios** e várias **práticas** que ocorrem no contexto de quatro atividades. Ela se destina a times de até dez programadores, projetos de curto e médio prazo.



Extreme Programming



XP (Extreme Programming)

OS 5 VALORES DE XP SÃO:

- **Comunicação** – para um projeto de sucesso é necessário muita interação entre os membros da equipe, programadores, cliente, treinador. Para desenvolver um produto, o time precisa ter muita qualidade nos canais de comunicação. Conversas presenciais são sempre melhores do que telefonemas, e-mails, cartas ou fax.
- **Feedback** – as respostas às decisões tomadas devem ser rápidas e visíveis. Todos devem ter, o tempo todo, **consciência** do que está acontecendo.
- **Coragem** – alterar um código em produção, sem causar bugs, com agilidade, exige muita coragem e responsabilidade.
- **Simplicidade** – para atender rapidamente às necessidades do cliente, quase sempre um dos valores mais importantes é simplicidade. Normalmente o que o cliente quer é muito mais simples do que aquilo que os programadores constroem.
- **Respeito** – todos têm sua importância dentro da equipe e devem ser respeitados e valorizados. Isso mantém o trabalho energizado.

XP (Extreme Programming)

EM XP EXISTEM 4 PAPÉIS PRINCIPAIS:

- **Programador** - foco central da metodologia, sem hierarquia.
- **Treinador** (ou *coach*) - pessoa com mais experiência no time, responsável por lembrar os outros das regras do jogo (que são as práticas e os valores de XP). O treinador não precisa necessariamente ser o melhor programador da equipe e sim o que mais entende da metodologia XP.
- **Acompanhador** (ou *tracker*) - responsável por trazer para o time dados, gráficos, informações que mostrem o andamento do projeto e ajudem a equipe a tomar decisões de implementação, arquitetura e design. Algumas vezes o próprio coach faz papel de tracker. Outras o time escolhe sozinho quem exercerá este papel.
- **Cliente** – em XP o cliente faz parte da equipe. Deve estar sempre presente e pronto para responder às dúvidas dos programadores.

XP (Extreme Programming)

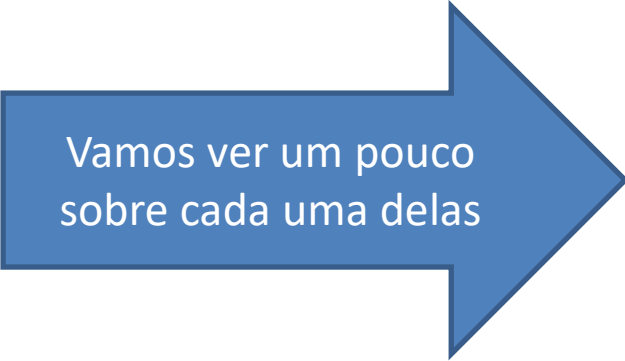
Existe um grande ênfase ao trabalho em duplas, aonde um analista mais experiente trabalha com um novato. Enquanto o mais jovem trabalha na programação o mais antigo vai revisando o código.

Dessa forma ao mesmo tempo desenvolve-se a equipe, e melhora-se automaticamente a qualidade do código fonte gerado.



XP (Extreme Programming)



- Algumas práticas recomendadas pelo XP:
 - Testes
 - Refatoração
 - Programação Pareada
 - Propriedade Coletiva
 - Integração Contínua
 - Semana de 40 horas
 - Cliente Sempre Presente
 - Padronizações



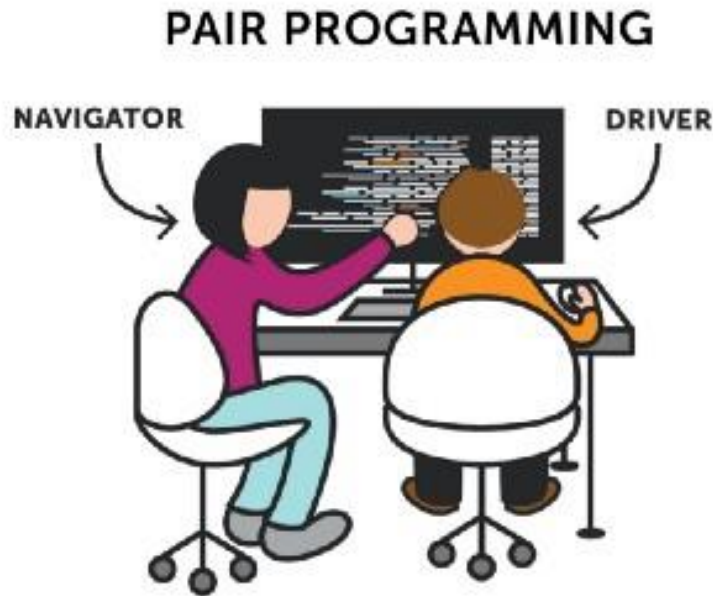
Vamos ver um pouco
sobre cada uma delas

XP (Extreme Programming)



-  **TESTES** - todo desenvolvimento inclui testes. Kent Beck diz que código sem teste não existe. Os testes devem ser escritos de preferência antes do desenvolvimento (TDD - *test driven development*) e sempre devem rodar de forma automatizada.
-  **REFATORAÇÃO** - é um conjunto de técnicas para modificar o código do sistema sem alterar nenhuma funcionalidade. O objetivo é simplificar, melhorar o design, limpar, enfim, deixar o código mais fácil de entender e dar manutenção.

XP (Extreme Programming)



PROGRAMAÇÃO PAREADA - em XP dois programadores trabalham juntos em um único programa. Enquanto um programador digita, o outro observa, revisa, pensa em melhorias, alternativas.

XP (Extreme Programming)




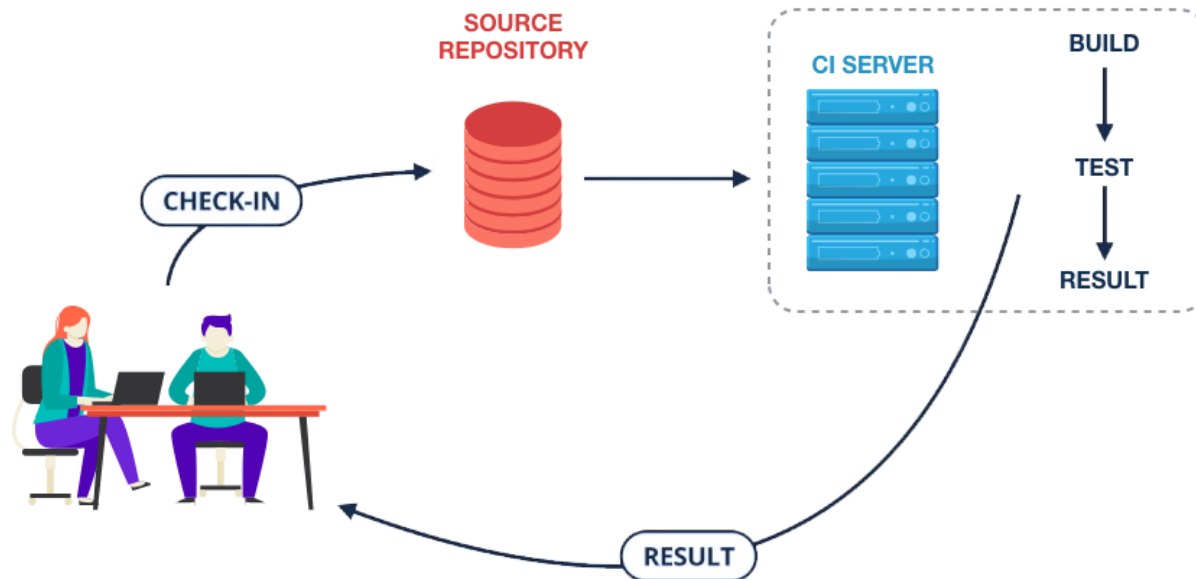
PROPRIEDADE COLETIVA - O código fonte não pertence a um único programador. Todos da equipe são responsáveis. Todos alteram código de todos (mas sempre rodando os testes para se certificar que nada foi quebrado).



SEMANA DE 40 HORAS - programar é uma atividade intensa e que não rende se o programador não estiver descansado e disposto. Por isso, não recomenda-se passar mais do que 40 horas de trabalho por semana. Isso é essencial para a saúde do time.

XP (Extreme Programming)

 **INTEGRAÇÃO CONTÍNUA** - depois de testada, cada nova funcionalidade deve ser imediatamente sincronizada entre todos os desenvolvedores. Quanto mais frequente for essa integração, menores são as chances de conflitos de arquivos que vários programadores alteram simultaneamente.



XP (Extreme Programming)



Cliente Sempre Presente - o cliente não é alguém de fora, mas sim um membro da equipe. Ele deve estar sempre disponível e pronto para atender às dúvidas dos desenvolvedores.



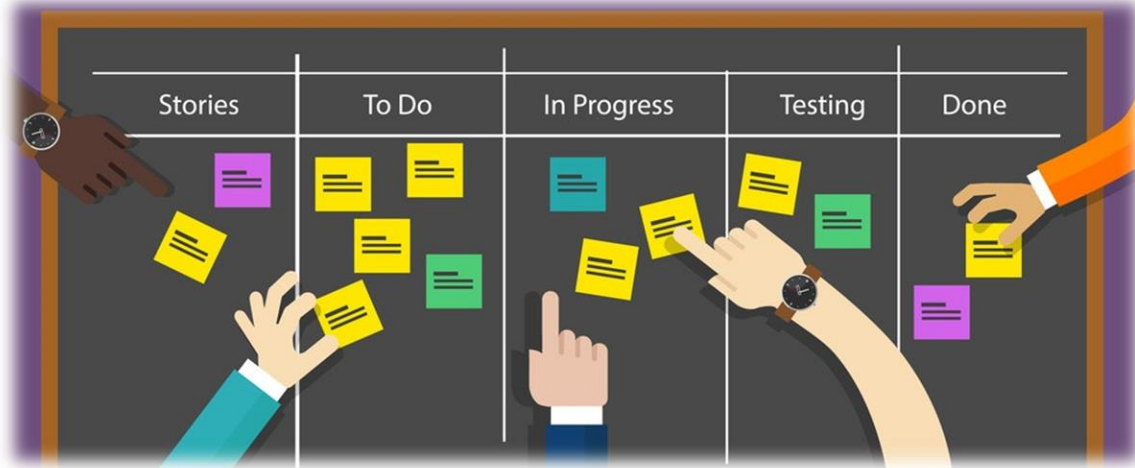
Padronizações - se todo o time seguir padrões pré-acordados de codificação, mais fácil será manter e entender o que já está feito. O uso de padrões é uma das formas de reforçar o valor da comunicação.

SCRUM



Scrum

- Uma alternativa para utilizar métodos ágeis na gerência de projetos.
- Pode ser aplicável a qualquer tipo de projeto.
- É simples.
 - Processo, artefatos e regras são poucos e fáceis de entender
 - A simplicidade pode ser decepcionante aos acostumados com metodologias clássicas.



Scrum

- **Não é um método prescritivo.**
 - Não define previamente o que deve ser feito em cada situação.
 - Projetos complexos não permitem prever todos os eventos.
- **Aplica o senso comum**
 - Combinação de experiência, treinamento, confiança e inteligência de toda a equipe
 - Senso comum em vez do senso de uma única pessoa é uma das razões do sucesso do Scrum



Ênfases

- **Comunicação.**
- **Trabalho em equipe.**
- **Flexibilidade.**
- **Fornecer software funcionando .**
 - Incrementalmente.

Principais Padrões

- *Backlog*
- Equipes
- *Sprints*
- Encontros Scrum
- Revisões Scrum/Demos

Backlog

- Lista de todas as funcionalidades desejadas.
- É gerada incrementalmente:
 - Começa pelo básico, o extra aparece com o tempo
- Pode conter:
 - Tarefas diretas, casos de uso e histórias.
- A lista é priorizada pelo dono do projeto
 - Cliente, departamento de marketing, ...



O *Backlog* Inicial

- Deve conter características que agreguem algum valor de negócio ao produto.
- Novos requisitos aparecem quando o cliente vê o produto.
- A arquitetura do sistema surge enquanto o projeto surge e é refatorado.

Equipes

- Sem nível hierárquico nem papéis.
 - Mas com várias especialidades
- Estão todos no mesmo barco.
- Geralmente equipes pequenas (até 10).
 - Existem casos com equipes maiores.
- Comunicação é essencial.
 - Encontro Scrum diário

OS 3 PAPÉIS DO SCRUM



Papéis no Scrum



Product Owner (P.O.)

Gerencia o product backlog e conecta a equipe de desenvolvimento com as solicitações de clientes



Scrum Master

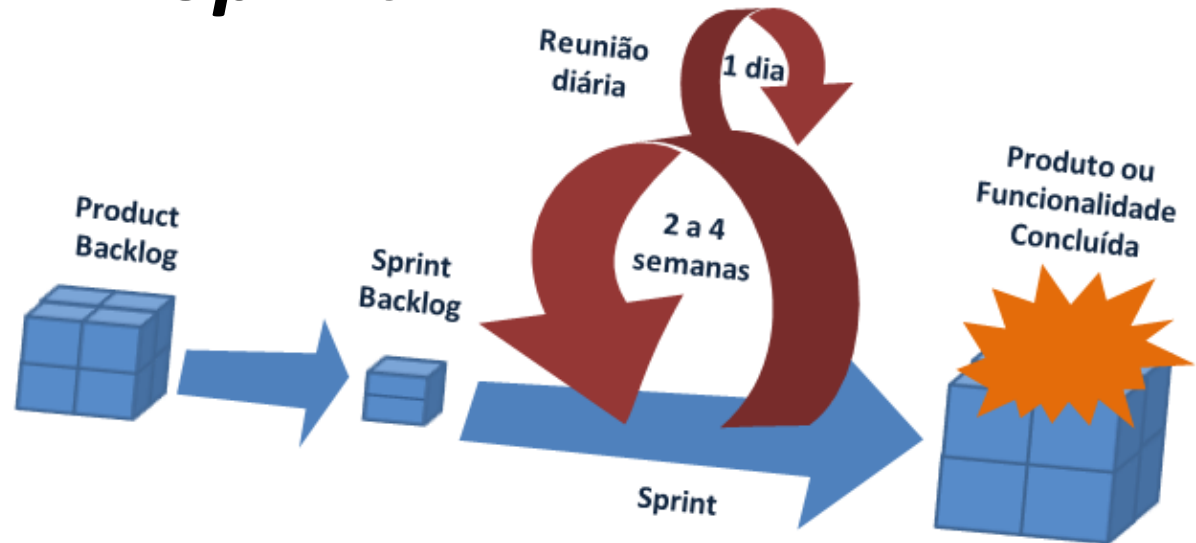
Líder da equipe, vai guiar o time pelas melhores práticas do Scrum. Faz a ponte entre o P.O e o time de dev.



Time de Dev

A equipe de desenvolvimento. Profissionais que executam os incrementos no produto

Sprint

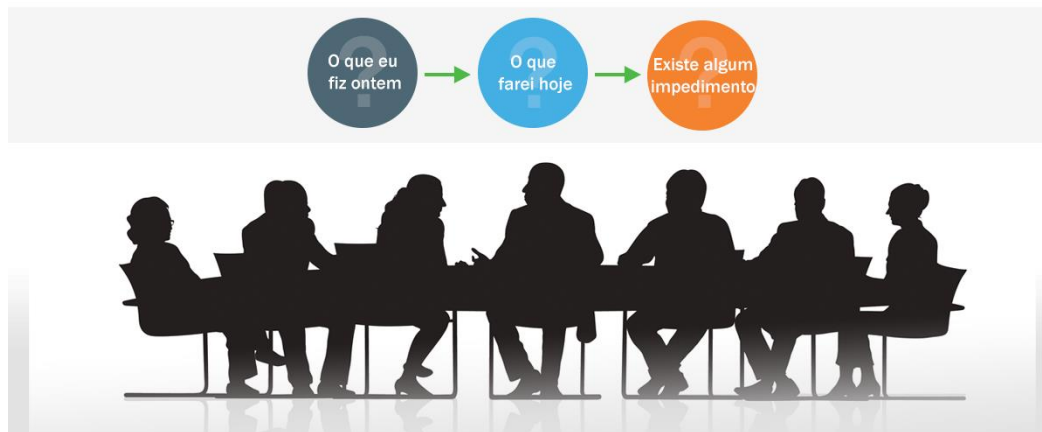


- Unidades básicas de tempo (até 30 dias).
- Começa com um encontro *Sprint*:
 - Tarefas do *Backlog* são priorizadas;
 - A equipe seleciona tarefas que podem ser completadas durante o próximo *Sprint*;
 - As mesmas podem ser quebradas para o *Backlog* do *Sprint*;
 - Cada tarefa recebe um responsável na equipe;
 - Não há mudança nas tarefas durante o *Sprint*.

Encontro Scrum

- Pequenos encontros diários da equipe:
 - Geralmente pela manhã;
 - Todos da equipe devem participar e falar.
- Questões que aparecem devem ser resolvidas durante o dia e não na reunião.
- Os encontros iniciais geralmente são mais longos.

REUNIÃO DIÁRIA



Encontro Scrum

- Questões que devem ser respondidas por cada membro da equipe:
 - 1) O quê você fez ontem?
 - 2) O quê você vai fazer hoje?
 - 3) Quais os problemas encontrados?
- Ajuda a manter as promessas.
- Evita: Como um projeto atrasa um ano?
 - Um dia por vez ...
 - Qualquer deslize pode ser corrigido de imediato.



Local do Encontro

- Sempre no mesmo local e hora.
- Pode ser o local de desenvolvimento.
- Pessoas sentadas ao redor de uma mesa.
- A sala já deve estar arrumada antes.
- Punições (atrasos/faltas).
- Todos devem participar.
- Pode ser em pé.
- Sala bem equipada, quadro branco, etc.



Revisão do *Sprint*

- No final de cada *Sprint* é feita uma reunião com todos os interessados.
- Geralmente
 - Na forma de demonstração.
 - Informal (preparação rápida, sem projetor,...).
 - Deve ser o resultado natural de um *Sprint*.
- O projeto é comparado com os objetivos iniciais do *Sprint*.



Fases

- Planejamento
- Sprints
 - Reuniões Diárias
 - Revisão
- Encerramento

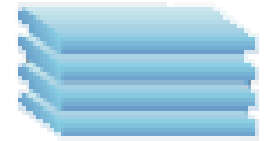


Planejamento

- Relativamente curto
- Projeto da arquitetura do sistema
- Estimativas de datas e custos
- Criação do *backlog*.
 - Participação de clientes e outros departamentos
 - Levantamento dos requisitos e atribuição de prioridades
- Definição de equipes e seus líderes
- Definição de pacotes a serem desenvolvidos

2

ATIVIDADES



Sprint

- O time recebe uma parte do *backlog* para desenvolvimento.
 - O *backlog* não sofrerá modificações durante o Sprint.
- Duração de 1 a 4 semanas.
- Sempre apresenta um executável ao final.



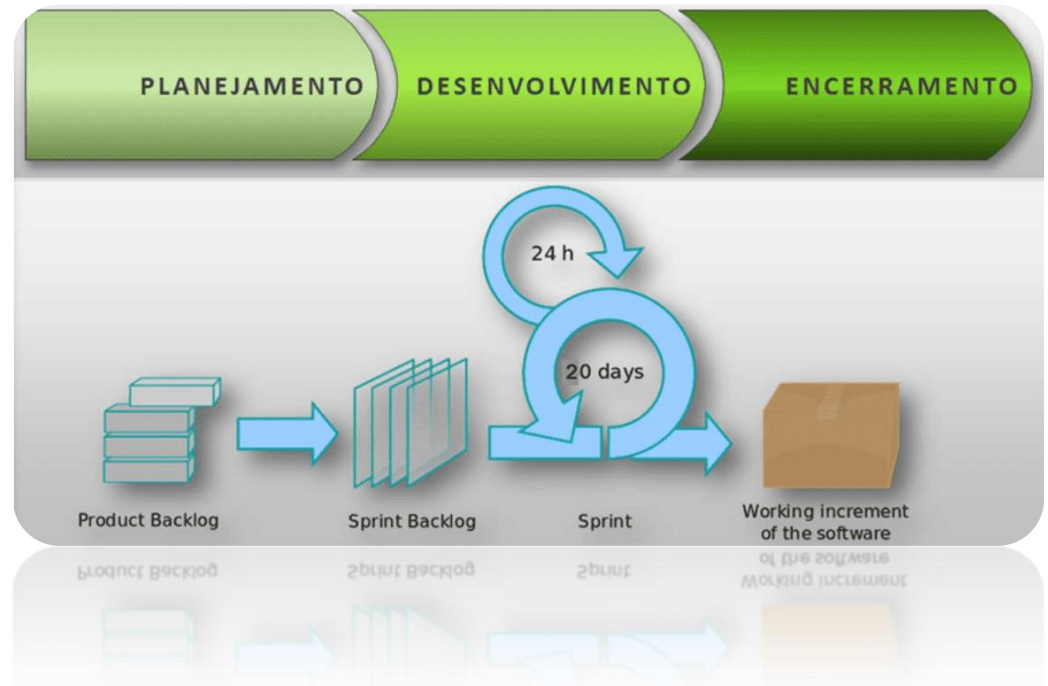
Sprint - Revisão

- Deve obedecer à data de entrega
 - Permitida a diminuição de funcionalidades
- Apresentação do produto ao cliente
 - Sugestões de mudanças são incorporadas ao *backlog*
- Benefícios:
 - Apresentar resultados concretos ao cliente
 - Integrar e testar uma boa parte do software
 - Motivação da equipe



Encerramento

- Finalização do projeto.
- Atividades:
 - Testes de integração
 - Testes de sistema
 - Documentação do usuário
 - Preparação de material de treinamento
 - Preparação de material de marketing



Amanhã (11/01) haverá atividade para
compor a 1ª nota

Fórum

A Prova 1ª avaliação será no dia 18/01/23.

Até a próxima aula...