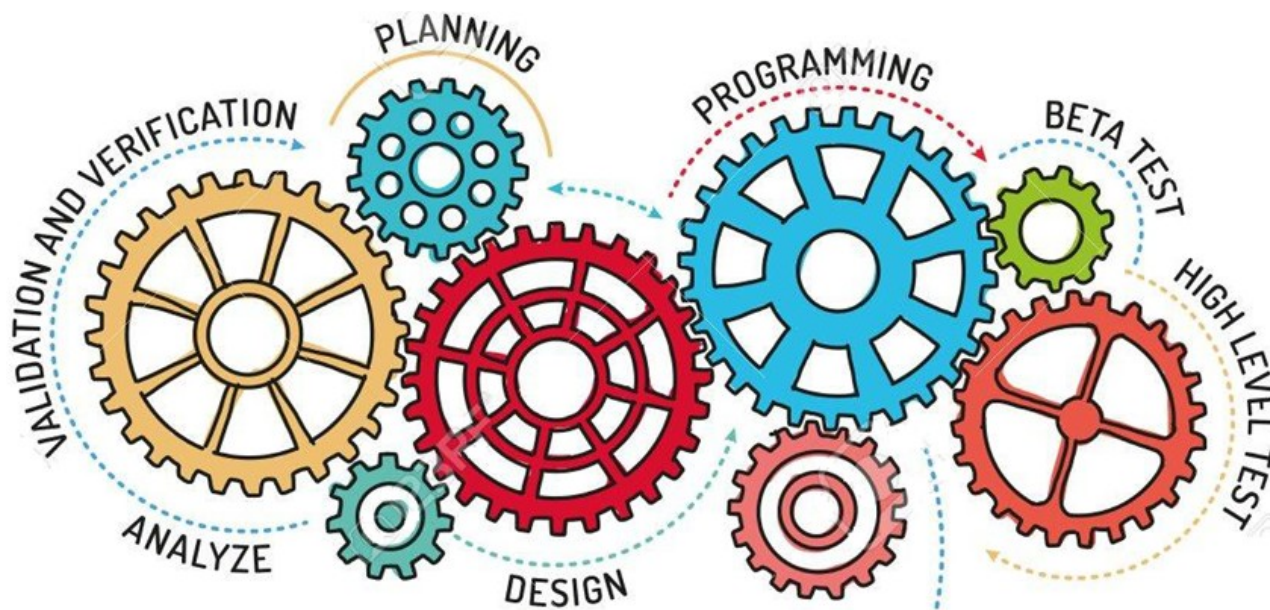


Engenharia de Software



Scrum

- Modelo agil de gestão de projetos;
- Conceito mais importante chama-se **sprint** (ou ciclo);
- Origem na industria automobilística;
- Livro de Schwaber e Beedle (2001) explica de forma completa e sistematica;



Perfis Importantes no Scrum

Product Owner



Scrum Master



Scrum Team



Perfis Importantes no Scrum

Product Owner



- Responsável pelo projeto em si;
- Indicar quais requisitos são os mais importantes em cada ciclo;
- Responsável por conhecer e avaliar as necessidades do cliente;

Perfis Importantes no Scrum

Scrum Master

- Não é gerente;
- Não é líder.



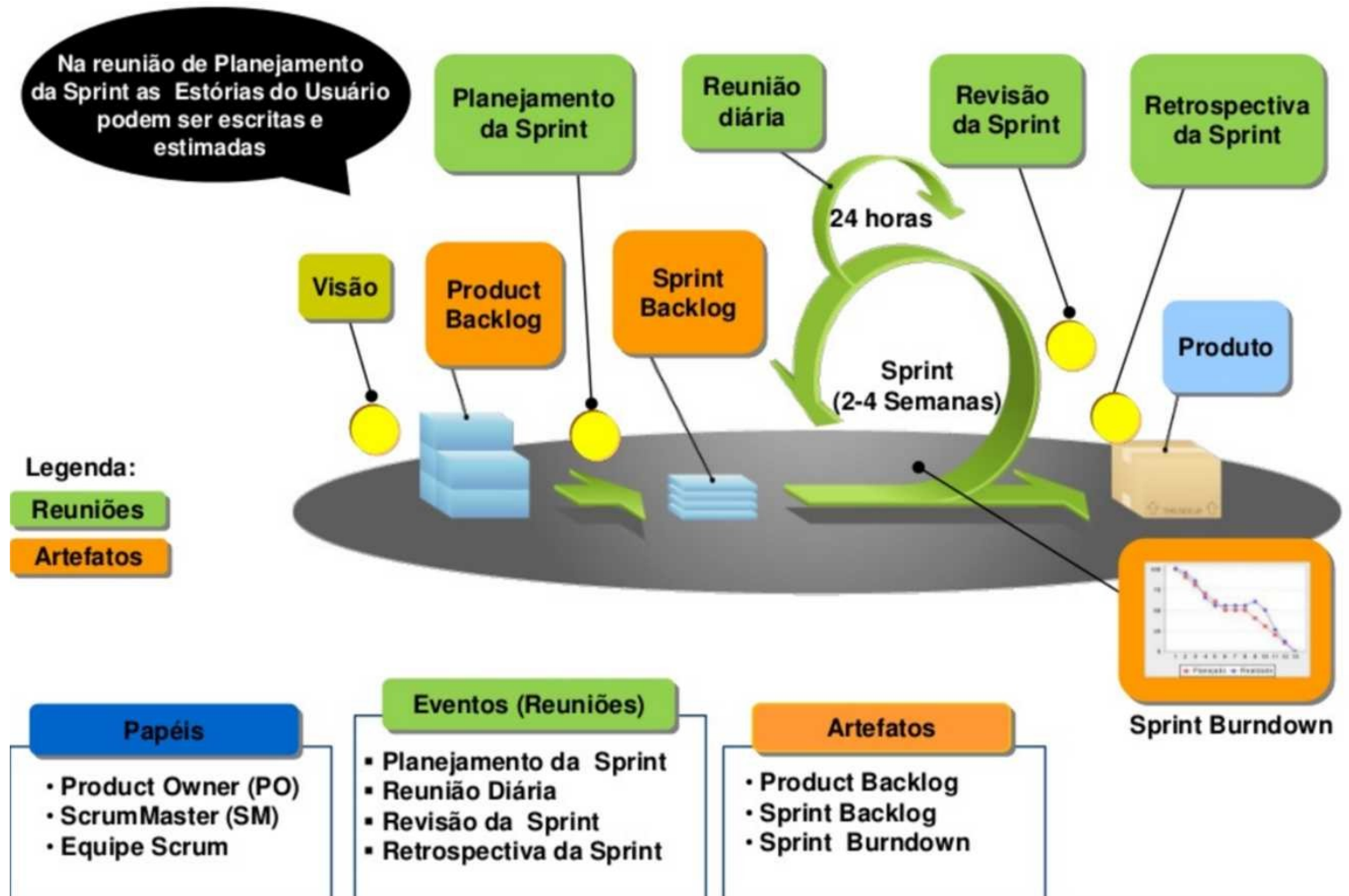
- É um facilitador;
- Conhece bem o modelo;
- Solucionador de conflitos;

Perfis Importantes no Scrum

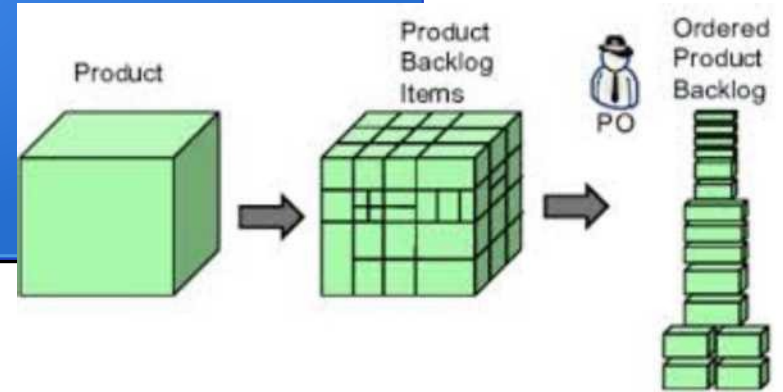
- Equipe de desenvolvimento;
- Não necessariamente dividida em papéis (analista, designer...)
- Todos interagem para desenvolver o produto em conjunto;
- Recomendado equipes de 6 a 10 pessoas.



Visão Geral do Scrum



Product Backlog



- Lista contendo as funcionalidades a serem implementadas em cada projeto (requisitos ou histórias de usuário);
- Não precisa ser completo (do Manifesto Ágil, **adaptação** em vez de planejamento);
- Tentar obter do cliente o maior número possível de informações sobre suas necessidades.

Product Backlog

Exemplos:

ID	Nome	Imp	PH	Como demonstrar	Notas
1	Depósito	30	5	Logar, abrir página de depósito, depositar R\$ 10,00, ir para a página de saldo e verificar que ele aumentou em R\$ 10,00	Precisa de um diagrama de sequência UML.
2	Ver extrato	10	8	Logar, clicar em "Transações". Fazer um depósito. Voltar para "Transações", ver que o depósito apareceu.	Usar paginação para evitar consultas grandes ao BD.

- **Imp:** Importancia da historia de usuario (maior = mais importante);
- **PH:** Estimativa de esforço necessario para transformar a historia em software; Valor dado em Pontos de Historia;
- **Como demonstrar:** considerar a historia efetivamente implementada.

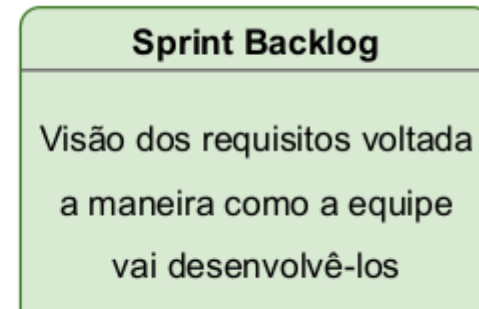
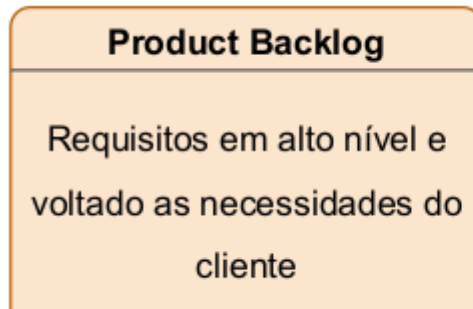
Planning Poker



- Definido pela primeira vez por James Grenning em 2002;
- Obtém estimativas por meio de um jogo de cartas;
- Realizadas rodadas para obter a estimativa de um cartão que possui uma história ou tarefa a desenvolver;
- PO é responsável por tirar todas as possíveis dúvidas evitando assim o retrabalho.

Sprint

- Ciclo de desenvolvimento de poucas semanas de duração (2 a 4 semanas);
- No início é feito um sprint planning meeting
 - Prioriza os elementos do product backlog e transfere para o sprint backlog.
- Equipe se compromete em desenvolver as atividades do sprint backlog;
- Product Owner se compromete a não trazer novas funcionalidades durante o mesmo sprint;



Quadro de Andamento de Atividades

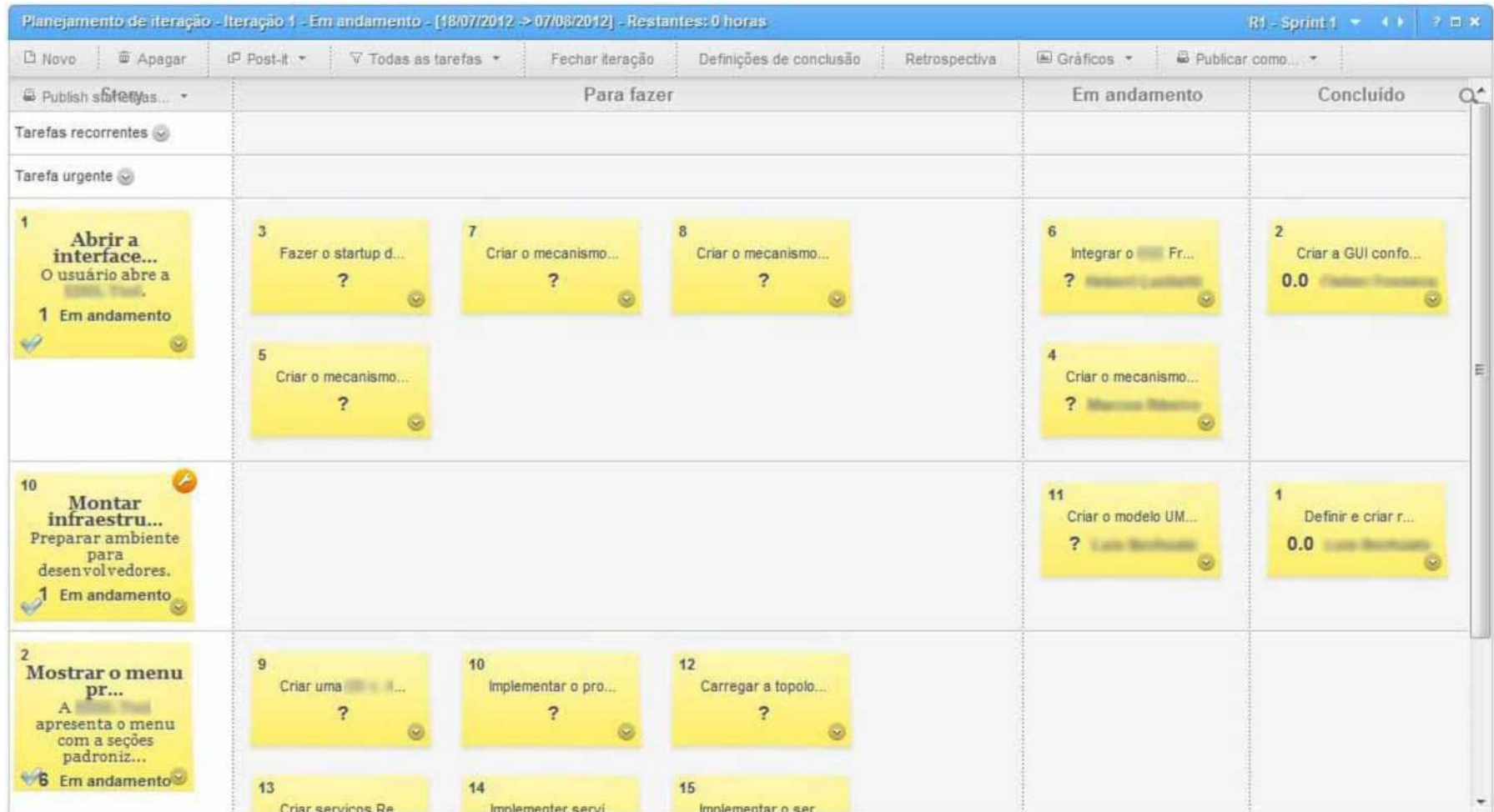
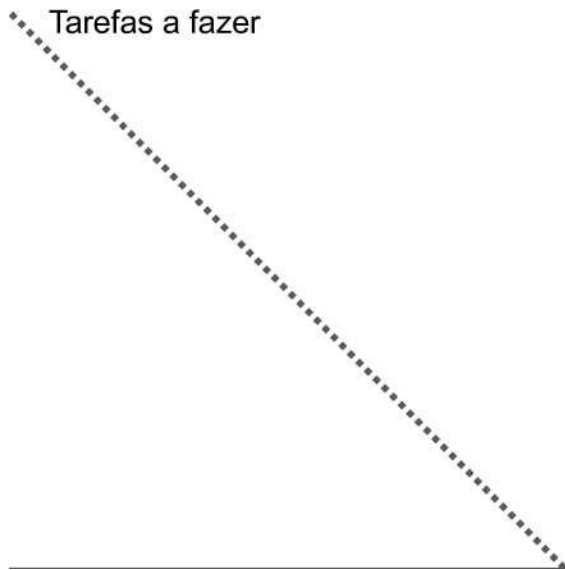


Diagrama Sprint Burndown

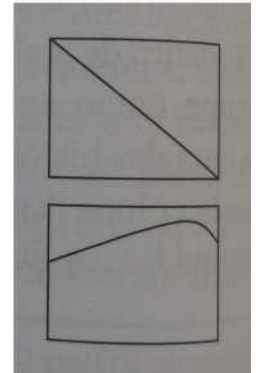
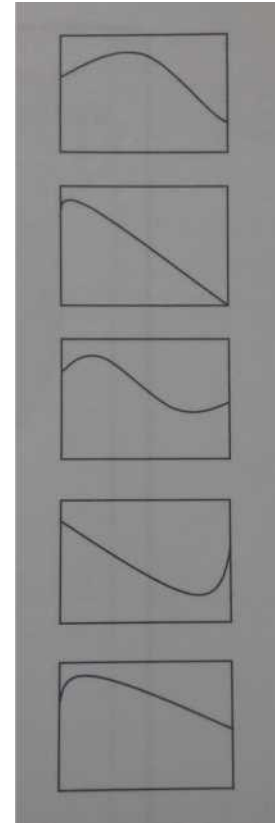
Número de
Tarefas

Tarefas a fazer



Tempo

Diagrama Ideal



Sprint

Final da sprint, equipe deve realizar:

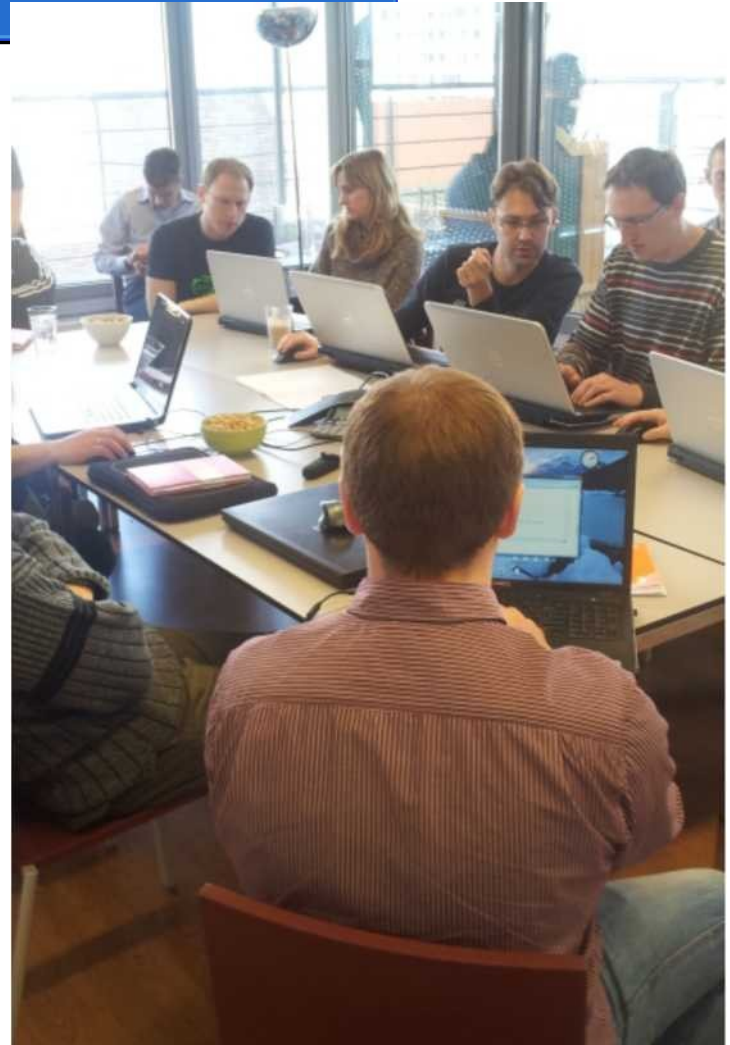
- Sprint Review Meeting
- Sprint Retrospective

Sprint Review Meeting

- Verificar o que foi feito e, então, partir para uma nova sprint

Sprint Retrospective

- Avaliar a equipe e os processos (impedimentos, problemas, dificuldades, ideias novas...)

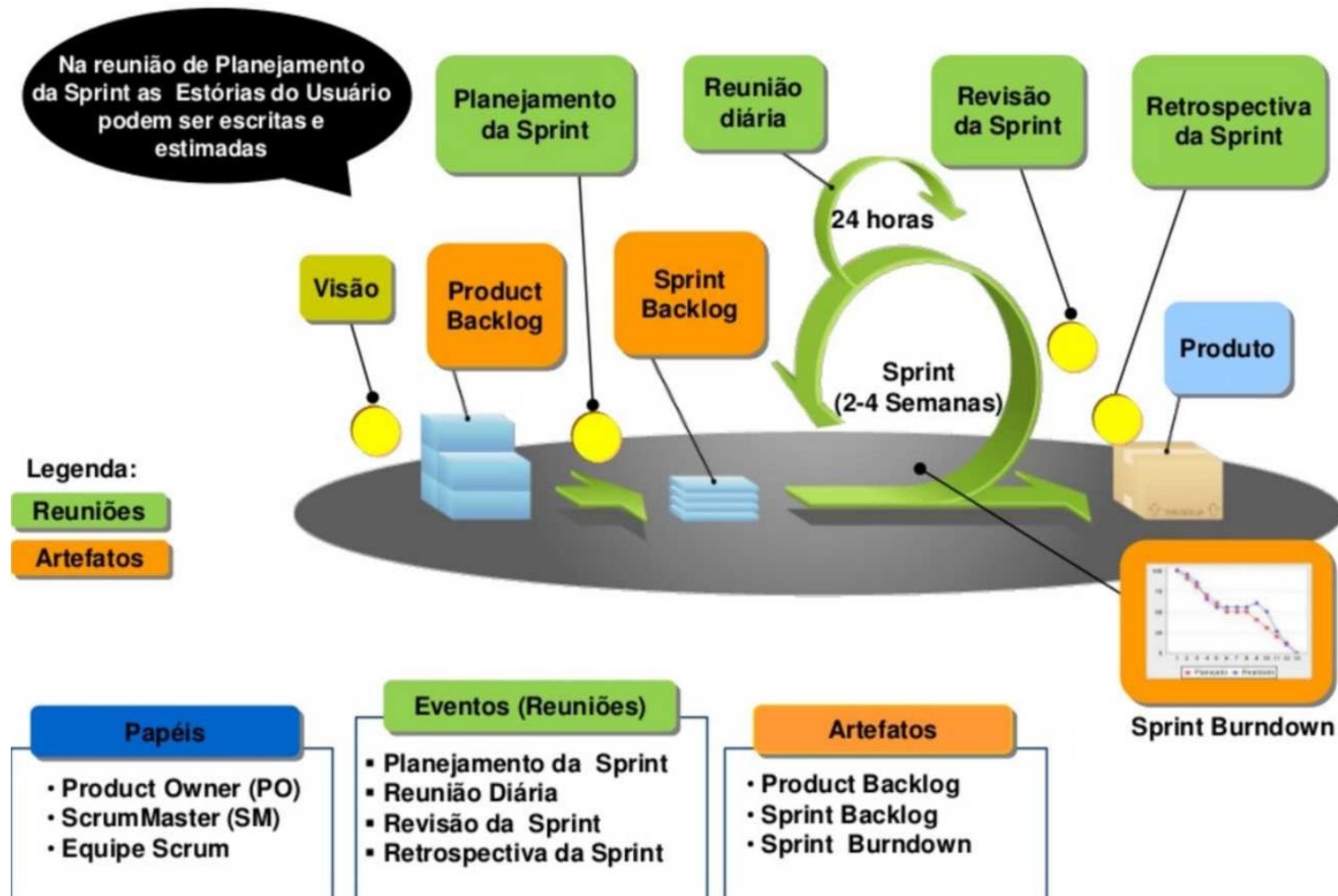


Daily Scrum

- Modelo sugere reuniões diárias chamada **Daily Scrum**;
- Objetivo:
 - Falar o que fez no dia anterior;
 - O que vai fazer no dia seguinte;
 - O que impede de prosseguir.
- Reuniões rápidas e em **pé** em frente ao quadro de anotações;
- Boa maneira de dissipar o cansaço.



Visão Geral do Scrum



Extreme Programming

- Também conhecido como XP;
- Surgiu nos Estados Unidos no final da década de 1990;
- Inicialmente adequada a equipes pequenas e medias;
- Codificação e a principal tarefa;
- Baseada em diversos valores, princípios e regras;
- Principais valores do XP:
- Simplicidade;
- Respeito;
- Comunicação;
- Feedback;
- Coragem.

Simplicidade

- Concentrar nas atividades **efetivamente necessárias** e não naquelas que poderiam ser;
- Assuma a solução **mais simples** como a **melhor**;
- Use as tecnologias, algoritmos e técnicas mais simples que permitirão atender aos requisitos do usuário final;
- **Design, processo e código** podem ser simplificados a qualquer momento.

Respeito

- Respeito entre os membros da equipe, assim como entre a equipe e o cliente;

Comunicação

- XP prioriza comunicação de **boa qualidade** preferindo encontros **presenciais**. Quanto mais **pessoal** e **expressiva**, melhor;
- Encontro presenciais > videoconferencias > telefonemas > e-mails;
- De preferência a comunicação mais agil.

Feedback

- Buscar obter feedback o quanto antes para evitar eventuais **falhas** de comunicação e aumento do **custo** da correção;
- Cliente sabe se o produto que esta sendo desenvolvido **atende** as suas necessidades;

Coragem

- Coragem de abraçar as **inevitáveis mudanças** em vez de simplesmente ignorá-las por estarem fora do contrato formal ou por serem difíceis de acomodar;
- **Testes, integração contínua, programação em pares** e outras praticas de XP **aumentam** a confiança do programador e ajudam-no a ter coragem para:
 - Melhorar o código que esta funcionando;
 - Investir tempo no desenvolvimento de testes;
 - Pedir ajuda aos que sabem mais.

Princípios Básicos do XP

- A partir dos valores, os princípios básicos do XP são definidos:
 - Feedback Rápido;
 - Presumir Simplicidade;
 - Mudanças Incrementais;
 - Abraçar Mudanças;
 - Trabalho de Alta Qualidade.
- Priorização das funcionalidades mais importantes.

Princípios Básicos do XP

- **Feedback Rápido**
 - Modele um pouco, mostre ao cliente e então modele novamente
- **Presumir Simplicidade**
 - Deixe o modelo tão simples quanto possível
- **Mudanças Incrementais**
 - Os problemas devem ser solucionados com um conjunto de pequenas modificações
- **Abraçar Mudanças**
 - Aceite as mudanças e tenha coragem para reconstruir
- **Trabalho de Alta Qualidade**
 - A qualidade do trabalho nunca deve ser comprometida

Atividades do XP

Escutar / Planejar

- Projetar
- Codificar
- Testar



Praticas XP

- Jogo de Planejamento
- Metáfora
- Equipe Coesa
- Reuniões em Pé
- Design Simples
- Versões Pequenas
- Ritmo Sustentável
- Posse Coletiva
- Programação em Pares
- Padrões de Codificação
- Testes de Aceitação
- Desenvolvimento orientado a testes (TDD)
- Refatoração
- Integração Contínua

* As práticas do XP não são consenso entre os desenvolvedores;

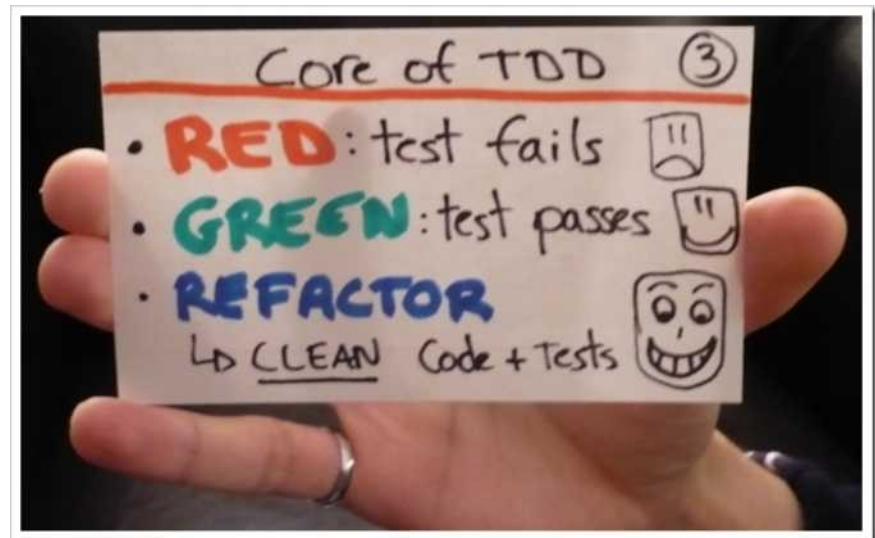
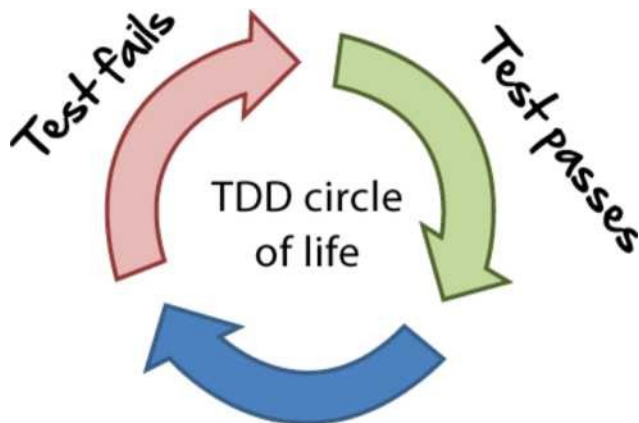
Programação em Pares

- Todo o desenvolvimento em XP feito em pares
 - . Um computador, um teclado, d(programadores . Um piloto, um co-piloto . Papeis sao alternados frequentemente . Pares são trocados periodicamente
- Benefícios
 - . Melhor qualidade do design, código testes
 - . Revisão constante do código . Nivelamento da equipe . Maior comunicação



TDD (Test-Driven Development)

- Desenvolvimento orientado a Testes;
- *“Test first, then code”*;
- Programadores XP escrevem testes primeiro, escrevem código e rodam testes para validar o código escrito;
- Cada unidade de código só tem valor se seu teste funcionar 100%;
- Testes são a documentação executável do sistema;

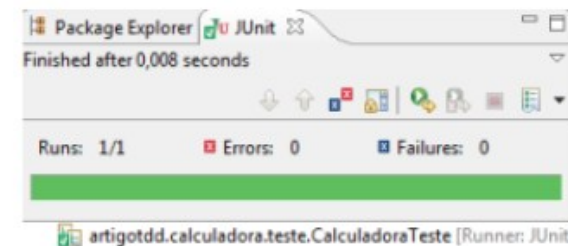
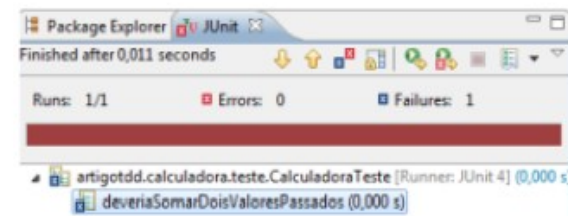
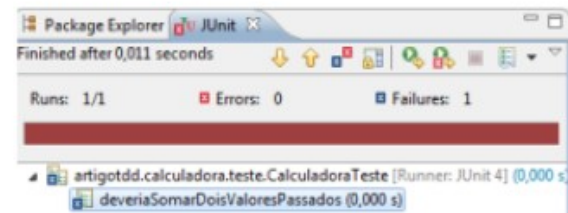


TDD

```
public class CalculadoraTeste {  
    @Test  
    public void deveriaSomarDoisValoresPassados() throws Exception {  
        int valorA = 1;  
        int valorB = 2;  
        Calculadora calculadora = new Calculadora();  
        int soma = calculadora.soma(valorA, valorB);  
  
        assertEquals(3, soma);  
    }  
}
```

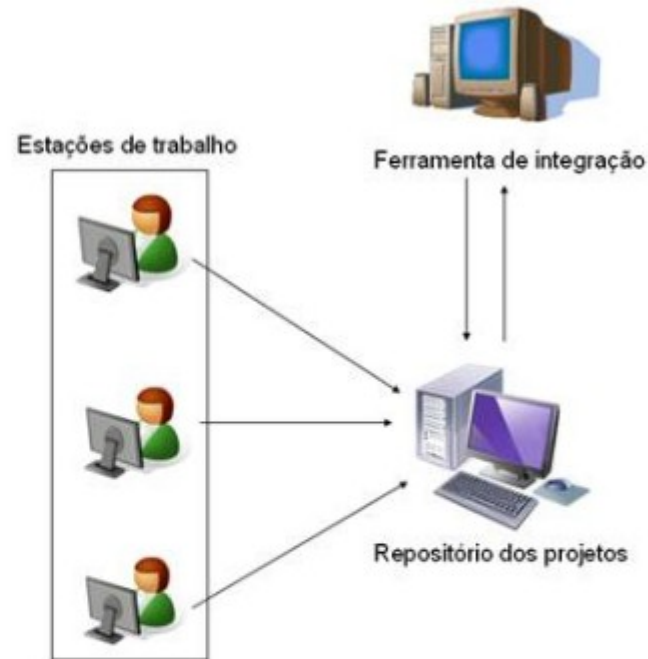
```
public class Calculadora {  
  
    public int soma(int valorA, int valorB) {  
        return 0;  
    }  
}
```

```
public class Calculadora {  
  
    public int soma(int valorA, int valorB) {  
        return valorA + valorB;  
    }  
}
```



Integração Contínua

- Projetos XP mantêm o sistema integrado o tempo todo
- Integração de todo o sistema pode ocorrer várias vezes ao dia (pelo menos uma vez ao dia)
- Todos os testes (unidade e integração) devem ser executados
- Benefícios:
 - Expõe o estado atual do desenvolvimento;
 - Oferece *feedback* sobre todo o sistema;
 - Permite encontrar problemas de design;



Refatoração

- processo de modificar um sistema de software para melhorar a **estrutura interna do código** sem alterar seu comportamento externo

- Antes

```
/** Salva o produto no banco de dados. */  
public void save() {  
    // Verifica propriedades  
    if (this.getName() == null) {  
        throw new Exception("Falta nome");  
    }  
    else if (this.getDescription() == null) {  
        throw new Exception("Falta a descrição");  
    }  
  
    this.getDatabase().save(this);  
}
```

- Depois

```
/** Salva o produto no banco de dados. */  
public void save() {  
    this.checkProperties();  
    this.getDatabase().save(this);  
}  
  
/** Verifica as propriedades do produto. */  
private void checkProperties() {  
    if (this.getName() == null) {  
        throw new Exception("Falta nome do produto.");  
    }  
    else if (this.getDescription() == null) {  
        throw new Exception("Falta a descrição do produto.");  
    }  
}
```

Dificuldades

- Vencer barreiras culturais;
- Deixar alguém mexer no seu código;
- Trabalhar em pares e ter coragem de admitir que não sabe;
- Vencer hábitos antigos:
 - Manter as coisas simples;
 - Jogar fora código desnecessário;
 - Escrever testes antes de codificar;
 - Refatoração com frequência