Processos e Qualidade de Software

Processos Ágeis XP e Scrum

PUC Minas

Instituto de Ciências Exatas
e Informática

Prof. Lesandro Ponciano

Departamento de Engenharia de Software
e Sistemas de Informação (DES)

Questões Motivadoras da Aula

1) Quais as principais características dos processos Scrum e XP?

2) Quais os pontos positivos e negativos de técnicas como testes automatizados, refatoração e requisitos descritos como histórias de usuários?

3) O que é mais eficaz em termos de se obter um conhecimento comum do sistema, programação em pares ou reuniões diárias?

Objetivos da Aula

- Analisar o Processo XP
  - Práticas
  - Ciclos
  - Características do processo
- Analisar o Processo Scrum
  - Características
  - Gerência
  - Ciclo

Programação Extrema (XP)

- *eXtreme Programmming* (XP)
  - Processo que surgiu no ano 2000* e é um dos processos baseados em método ágil mais conhecidos
  - Foi desenvolvido para impulsionar práticas reconhecidamente boas

- Em XP
  - Requisitos são expressos como histórias do usuário
  - Programadores trabalham em pares
  - Programadores desenvolvem testes para cada tarefa antes de escreverem o código
  - Há um curto intervalo de tempo entre entregas (*releases*)

XP é Ágil

1) Desenvolvimento incremental
   - Sustentado por *releases* pequenos e frequentes

2) Cliente participa
   - Representante do cliente participa do desenvolvimento e elabora os testes de aceitação
   - Especificação e priorização de requisitos

3) Pessoas em vez de processos
   - Programação em pares
   - Propriedade coletiva do código
   - Desenvolvimento sem horas excessivas de trabalho

XP é Ágil

4) Aceitação de mudanças
   - Teste
   - Refatoração (*refactoring*)
   - Integração contínua

5) Simplicidade
   - Refatoração
   - Projetos simples sem excessiva antecipação de mudanças

XP é Extremo

- Novas versões do software são construídas várias vezes por dia
  - Sempre que uma nova versão é construída, testes automáticos existentes e testes da nova funcionalidade devem ser executados
  - A versão só é aceita quando passa em todos os testes

Entrega de *Releases*

- *Releases* são entregues ao cliente a cada duas semanas, aproximadamente

- Prazos de *releases* nunca são desrespeitados
  - Se houver problema, o cliente é consultado
  - Nesse caso, alguma funcionalidade pode ser removida do *release*

Práticas do XP

1) Planejamento incremental
2) Pequenos *releases*
3) Projeto simples
4) Desenvolvimento *test-first*
5) Refatoração
6) Programação em pares
7) Propriedade coletiva
8) Integração contínua
9) Ritmo sustentável
10) Cliente no local

Cenários de Requisitos

- Um cliente ou usuário é parte do time de XP e é responsável na tomada de decisões sobre requisitos

- Requisitos do usuário são expressos como cenários ou histórias dos usuários
  - Esses são escritos em cartões
  - Capturam o "quem", "o quê" e "por quê" de um requisito em uma forma concisa e simples

História de Usuários

- São escritas por ou para usuários ou clientes de negócio
- A equipe de desenvolvimento divide as histórias em tarefas de implementação
  - São a base das estimativas de cronograma e custo
- O cliente escolhe as histórias que serão incluídas no próximo *release* levando em conta
  - Suas prioridades
  - Estimativas de cronograma

Exemplo de História

Prescrição de medicamentos

Kate é uma médica que deseja prescrever medicamentos para um paciente de uma clínica. O prontuário do paciente já está sendo exibido em seu computador, assim, ela clica o campo 'medicação' e pode selecionar 'medicação atual', 'nova medicação', ou 'formulário'.

Se ela selecionar 'medicação atual', o sistema pede que ela verifique a dose. Se ela quiser mudar a dose, ela altera esta e em seguida, confirma a prescrição.

Se ela escolher 'nova medicação', o sistema assume que ela sabe qual medicação receitar.
Ela digita as primeiras letras do nome do medicamento. O sistema exibe uma lista de possíveis fármacos que começam com essas letras. Ela escolhe a medicação requerida e o sistema responde, pedindo-lhe para verificar se o medicamento selecionado está correto.
Ela insere a dose e, em seguida, confirma a prescrição.

Se ela escolhe 'formulário', o sistema exibe uma caixa de busca para o formulário aprovado.
Ela pode, então, procurar pelo medicamento requerido. Ela seleciona um medicamento e é solicitado que verifique se a medicação está correta. Ela insere a dose e, em seguida, confirma a prescrição.

O sistema sempre verifica se a dose está dentro da faixa permitida. Caso não esteja, Kate é convidada a alterar a dose.

Após Kate confirmar a prescrição, esta será exibida para verificação. Ela pode escolher 'OK' ou 'Alterar'. Se clicar em 'OK', a prescrição fica gravada nos bancos de dados da auditoria.
Se ela clicar em 'Alterar', reinicia o processo de 'Prescrição de Medicamentos'.

Exemplo de Cartões de Tarefas

Tarefa 1: Alterar dose de medicamentos prescritos

Tarefa 2: Seleção de formulário

Tarefa 3: Verificação de dose

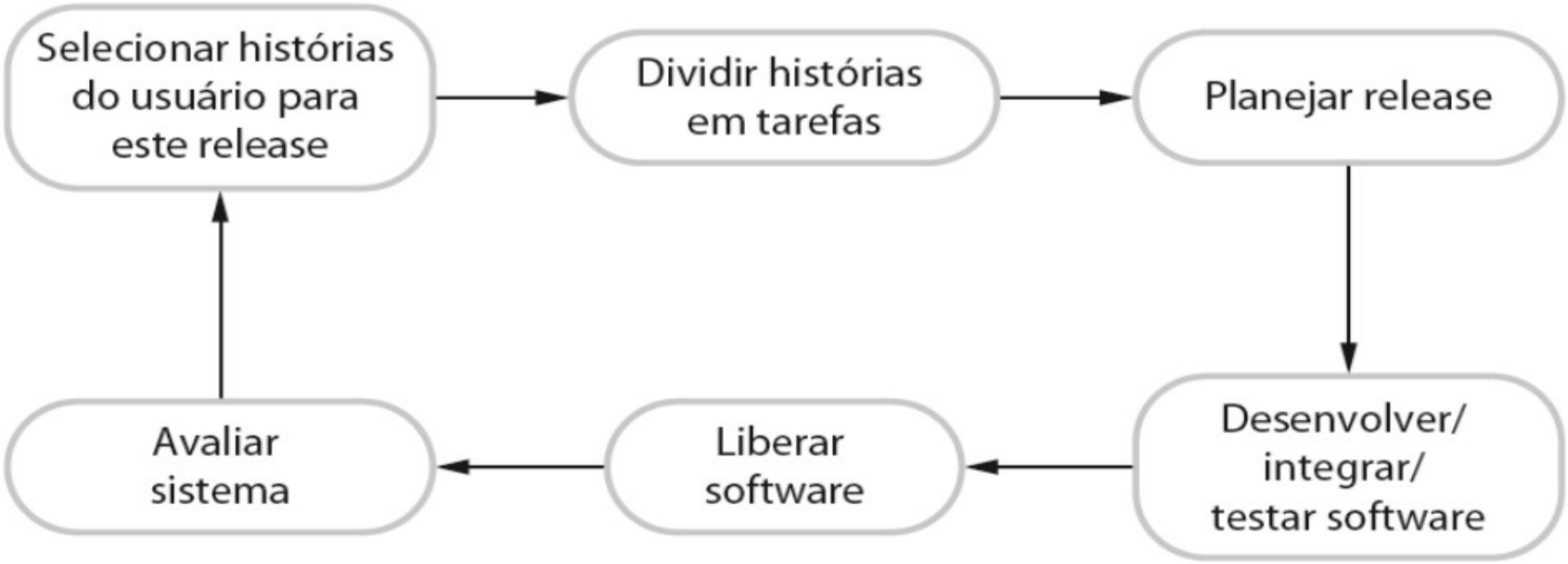
A verificação da dose é uma precaução de segurança para verificar se o médico não receitou uma dose perigosamente pequena ou grande.

Usando o ID do formulário para o nome do medicamento genérico, procure o formulário e obtenha a dose mínima e máxima recomendada.

Verifique a dose mínima e máxima prescrita. Caso esteja fora da faixa, emita uma mensagem de erro dizendo que a dose está muito alta ou muito baixa.

Caso esteja dentro da faixa, habilite o botão 'Confirmar'.

Um Ciclo no XP



XP e Mudanças

- O senso comum em ES diz que deve-se projetar pensado em mudanças
  - Vale a pena gastar tempo antecipando as mudanças
- XP afirma que isso não vale a pena, pois mudanças não podem ser antecipadas de forma confiável
- XP propõe o uso de refatoração para tornar as mudanças mais fáceis quando essas acontecem

Testes em XP

- XP desenvolveu uma abordagem em que o programa é testado depois que cada alteração é feita

- Características da abordagem de teste
  1) Desenvolvimento do teste primeiro (*test driven development*, TDD)
  2) Desenvolvimento de testes incrementais a partir de cenários
  3) Envolvimento do usuário no desenvolvimento de testes e validação
  4) A cada novo *release*, todos os testes do sistema são executados

Caso de Teste

Teste 4: Verificação de dose
Entrada: 1. Um número em mg representando uma única dose da medicação. 2. Um número que representa o número de doses únicas por dia. **Testes:** 1. Teste para entradas em que a dose única é correta, mas a frequência é muito alta. 2. Teste para entradas em que a única dose é muito alta e muito baixa. 3. Teste para entradas em que a dose única x frequência é muito alta e muito baixa. 4. Teste para entradas em que a dose única x frequência é permitida. **Saída:** Mensagem de OK ou erro indicando que a dose está fora da faixa de segurança.

Automação do Teste

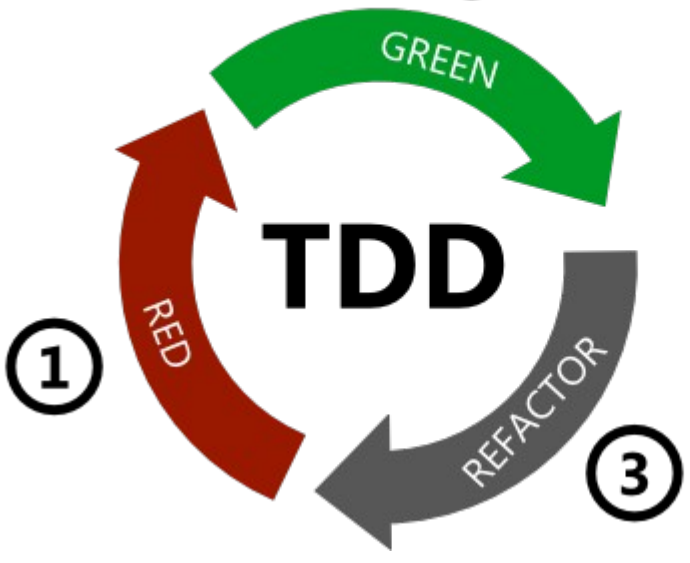
- Testes são escritos como componentes executáveis antes que a tarefa seja implementada
  - Exemplo de framework: Junit,

- Como os testes são automatizados, sempre existe um conjunto de testes que pode ser rapidamente e facilmente executados

- Dificuldades
  - É difícil escrever testes de unidade para o código que implementa a 'lógica de *display*' e o fluxo entre telas
  - É difícil julgar se um conjunto de testes está completo

A Abordagem TDD

2 Escreva código suficiente para passar no teste.

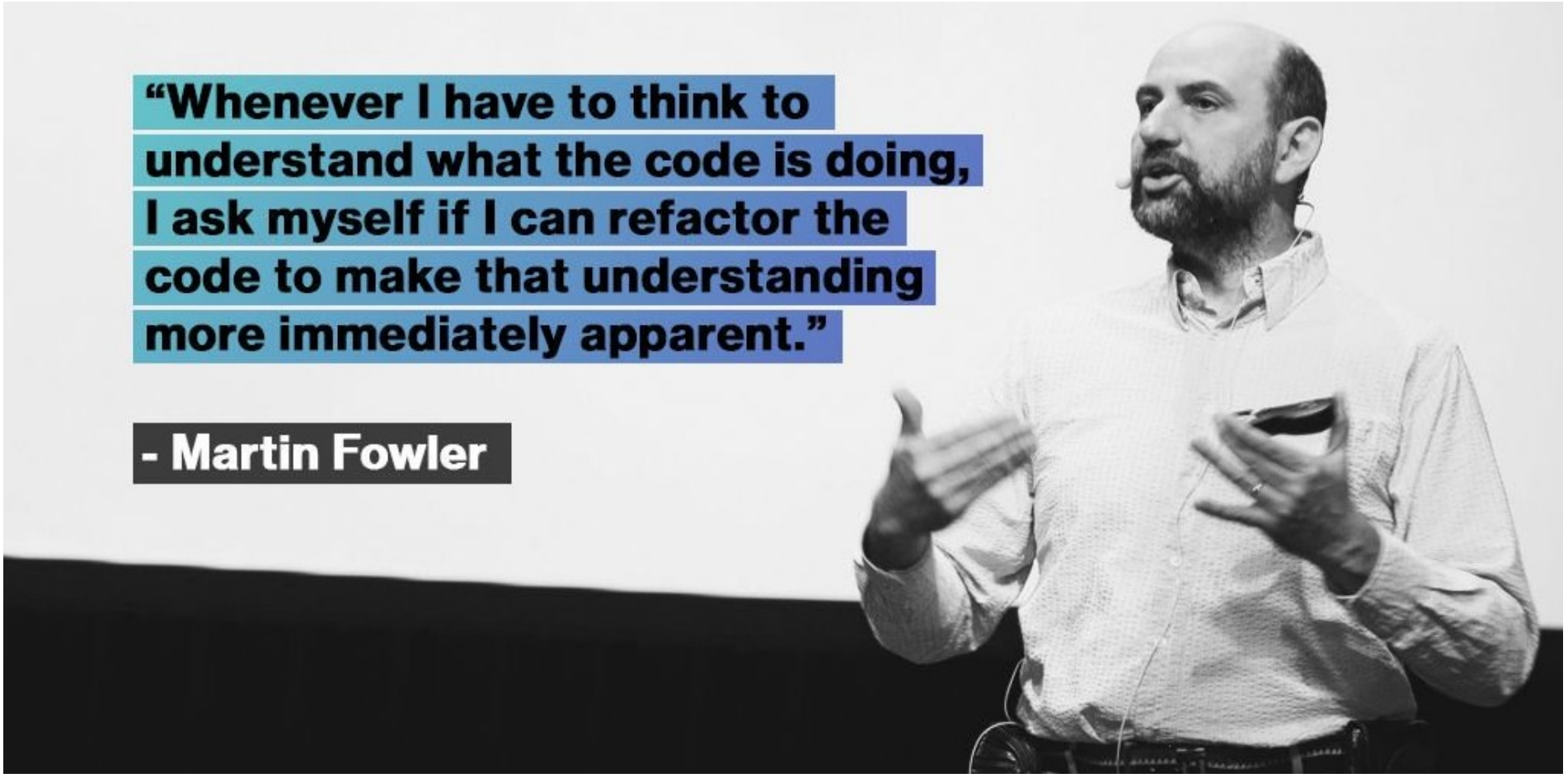


1 Escreva o teste e veja ele falhar

3 Melhore o código sem mudar o seu comportamento

Refatoração

- Melhorias do software
  - Aumentar a inteligibilidade
  - Deve-se garantir que todos os testes automáticos estão passando e que continuam passando após a refatoração
- Reduz a necessidade de documentação
- É mais fácil fazer mudanças
- Refatoração da arquitetura é algo mais caro
  - Pode requerer muitas alterações no código


“Whenever I have to think to understand what the code is doing, I ask myself if I can refactor the code to make that understanding more immediately apparent.”
- Martin Fowler

Exemplos de Refatoração

- Reorganização de uma hierarquia de classes para remover código duplicado
- Organização e renomeação de atributos e métodos para torná-los mais fáceis de entender
- A substituição do código com as chamadas para métodos definidos em uma biblioteca de programas

Programação em Pares

- Programadores trabalham em pares sentando juntos na mesma estação de trabalho para desenvolver código
  - Pares são criados dinamicamente de modo que todos trabalham com todos

- Vantagens
  - Apoia a ideia de propriedade coletiva
  - Funciona como um processo de revisão informal
  - Espalha conhecimento na equipe
  - Ajuda na refatoração

Programação em Pares

- Avaliações mostram que
  - A produtividade do desenvolvimento com programação em pares é similar a de duas pessoas trabalhando independentemente
  - Isso pode não ocorrer com programadores mais experientes
    - Nesse caso há perda significativa de produtividade em relação a programadores trabalhando sozinhos

Processo Scrum

- Teve início no ano de 1990*

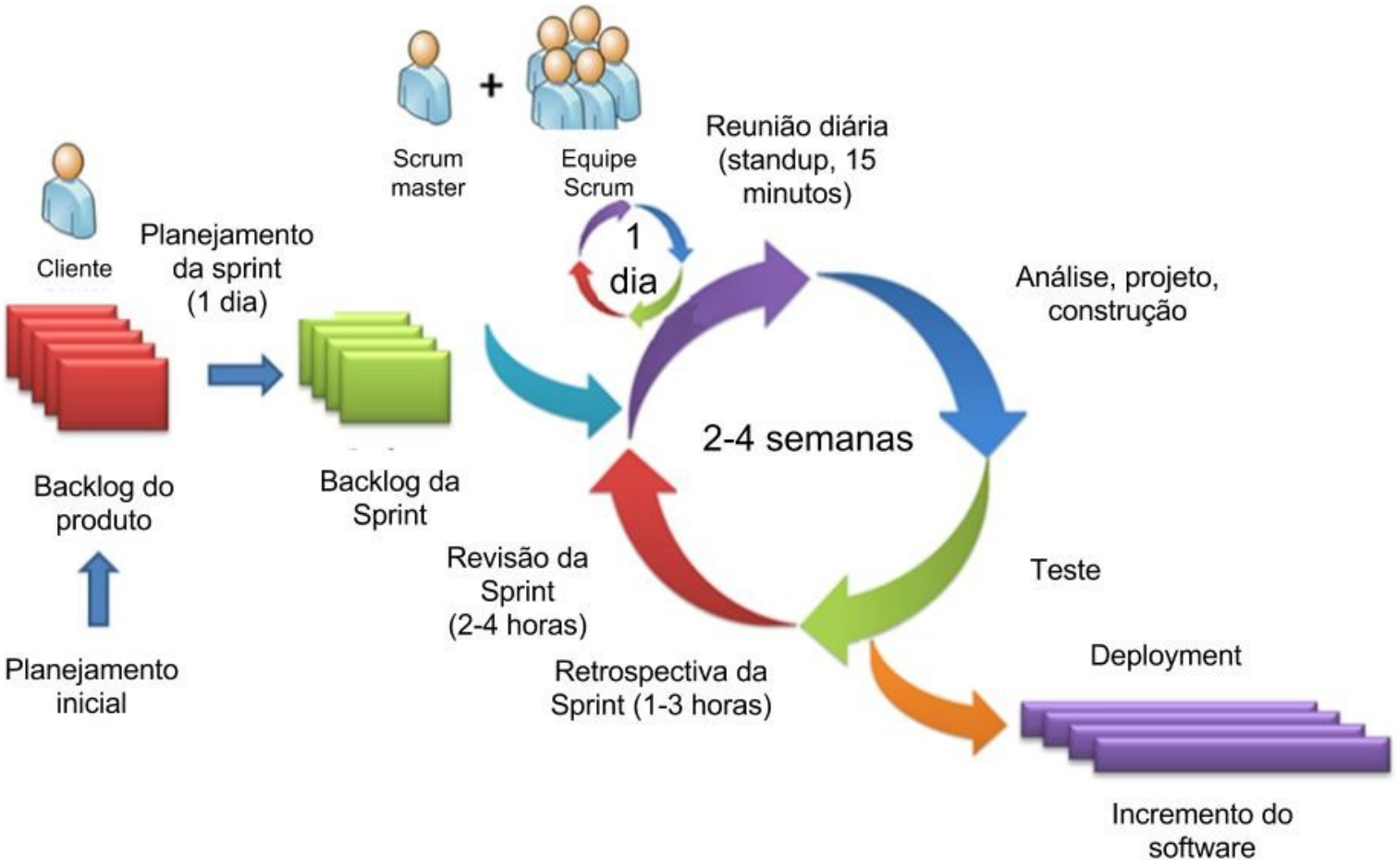
- Maior foco no gerenciamento do desenvolvimento do que em técnicas específicas de engenharia ágil
  - Não prescreve práticas de programação, como TDD, programação em pares, etc.

- Tem três fases
  1) Planejamento geral: objetivos do projeto e da arquitetura
  2) Ciclos de sprint: gerar incrementos do software
  3) Encerramento: documentação, ajuda e manuais dos usuários e síntese de lições aprendidas

Características do Scrum

1) O ponto de partida é o *backlog* do produto
2) Clientes e equipe trabalham juntos para selecionar funcionalidades a serem implementadas no *sprint*
3) Sprint corresponde ao desenvolvimento de um *release* do software; tem tamanho fixo de 2 a 4 semanas
4) Durante o sprint, há reuniões diárias (*daily meetings*) envolvendo a equipe sem contato com o cliente
5) No fim do sprint o trabalho é revisto com o cliente e o próximo sprint é planejado e iniciado

Visão Geral do Scrum

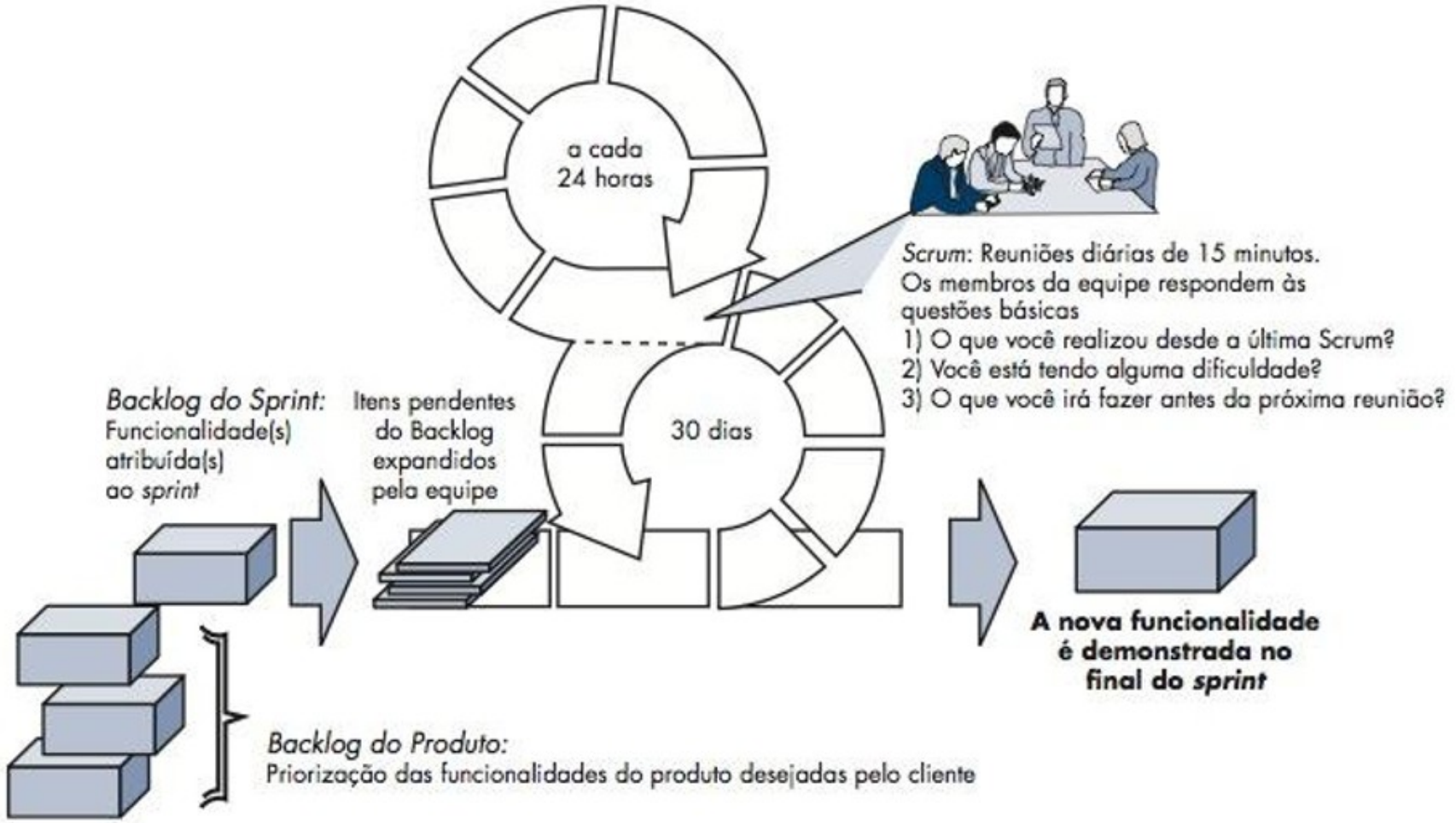


Gerência no Scrum

- A equipe tem poderes para tomar decisões de gerência
  - O termo "gerente de projeto" é evitado
  - O "scrum master" é um facilitador
    - Organiza as reuniões diárias; controla o *backlog;* registra decisões; mede o progresso e se comunica com os clientes

- Reuniões diárias são feitas em pé ("*stand-up*")
  - Toda a equipe participa
  - São rápidas
  - Discute-se o progresso desde a última reunião

Visão Geral


a cada
24 horas
30 dias
Scrum: Reuniões diárias de 15 minutos.
Os membros da equipe respondem às questões básicas
1) O que você realizou desde a última Scrum?
2) Você está tendo alguma dificuldade?
3) O que você irá fazer antes da próxima reunião?
Backlog do Sprint:
Funcionalidade(s) atribuída(s) ao sprint
Itens pendentes do Backlog expandidos pela equipe
A nova funcionalidade é demonstrada no final do sprint
Backlog do Produto:
Priorização das funcionalidades do produto desejadas pelo cliente

Atividade de Fixação

1) O que você acha mais eficaz em termos de obter um conhecimento comum do sistema, programação em pares ou reuniões diárias? Justifique.

2) Discuta dois pontos positivos e dois pontos negativos de
   - Testes automáticos
   - Refatoração
   - Requisitos descritos como histórias de usuários

3) Discuta sobre semelhanças e diferenças entre Scrum e XP.

Referências

- SOMMERVILLE, Ian. Engenharia de Software - 9a edição. Pearson ISBN 9788579361081. (Capítulo 3)

- PRESSMAN, Roger. Engenharia de software. 8. Porto Alegre ISBN 9788580555349. (Capítulo 3)