



UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS DE RUSSAS

Engenharia de Software

Processos Ágeis: XP (eXtreme Programming)

Profa. Dra. Anna Beatriz Marques

Extreme Programming (XP)

- » O Extreme Programming (XP) usa uma abordagem 'extrema' ao desenvolvimento iterativo.
- » Novas versões podem ser construídas várias vezes por dia;
- » Incrementos são entregues aos clientes a cada 2 semanas;
- » Todos os testes devem ser realizados em todas as versões e cada versão só é aceita se os testes forem concluídos com sucesso.



1. Valores do XP

Comunicação

- Considerar procedimentos formais e informais >> comunicação mais ágil

Simplicidade

- Adotar práticas que reduzam a complexidade do sistema

Feedback

- Obter feedback rápido sobre qualidade do código e estado do desenvolvimento

Coragem

- Aumentar a confiança do programador para simplificar um código funcional, investir em testes, pedir ajuda aos mais experientes, focar no código como documentação

Communication (comunicação)

- Várias práticas do XP promovem uma maior comunicação entre os membros da equipe
- A comunicação não é limitada por procedimentos formais.
- Usa-se o melhor meio possível, que pode ser:
 - Uma conversa ou reunião informal
 - Um e-mail, um bate-papo, um telefonema
 - O próprio código
- Preferência à comunicação mais ágil

Simplicity (simplicidade)

- XP incentiva ao extremo práticas que reduzam a complexidade do sistema
- A solução adotada deve ser sempre a mais simples que alcance os objetivos esperados
 - 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

Feedback (retroalimentação)

- Várias práticas de XP garantem um rápido feedback sobre várias etapas/partes do processo
 - Feedback sobre qualidade do código (testes de unidade, programação em pares, posse coletiva)
 - Feedback sobre estado do desenvolvimento (estórias do usuário-final, integração contínua, jogo do planejamento)
- Permite maior agilidade
 - Erros detectados e corrigidos imediatamente
 - Requisitos e prazos reavaliados mais cedo
 - Permite estimativas mais precisas

Courage (coragem)

- Testes, integração contínua, programação em pares e outras práticas de XP aumentam a confiança do programador e ajudam-no a ter coragem para:
 - melhorar o código que está funcionando para torná-lo mais simples
 - investir tempo no desenvolvimento de testes
 - mexer no design em estágio avançado
 - pedir ajuda aos que sabem mais
 - abandonar processos formais e ter o projeto e a documentação em forma de código

2. Princípios do XP

Retorno rápido

- Constante validação do cliente sobre o trabalho dos desenvolvedores

Simplicidade

- A solução deve atender às histórias da iteração corrente, sem pensar em necessidades futuras

Mudanças incrementais

- A solução não será perfeita na primeira iteração, sendo melhorada com pequenas modificações

Aceitar mudanças

- Mudanças ocorrerão no projeto conforme o crescimento do entendimento dos envolvidos

Trabalho de qualidade

- Importância do teste antes da codificação

Rapid Feedback (retorno rápido)

- O retorno entre os desenvolvedores é rápido
 - Cliente sabe se o produto que está sendo desenvolvido atende às suas necessidades
- Modele um pouco, mostre ao cliente e então modele novamente.
 - Garante que o seu modelo será preciso enquanto seu conhecimento do projeto aumenta

Assuma Simplicity (simplicidade)

- Deixe o seu modelo tão simples quanto possível e assuma que a solução mais simples é a melhor
- O design do sistema deve ser feito para a iteração corrente. Não deve ser feito design sobre uma possível necessidade futura.

Incremental Change (mudanças incrementais)

- O modelo não será perfeito na primeira tentativa, ele irá mudar de acordo com o desenvolvimento do projeto
- Os problemas devem ser solucionados com um conjunto de pequenas modificações

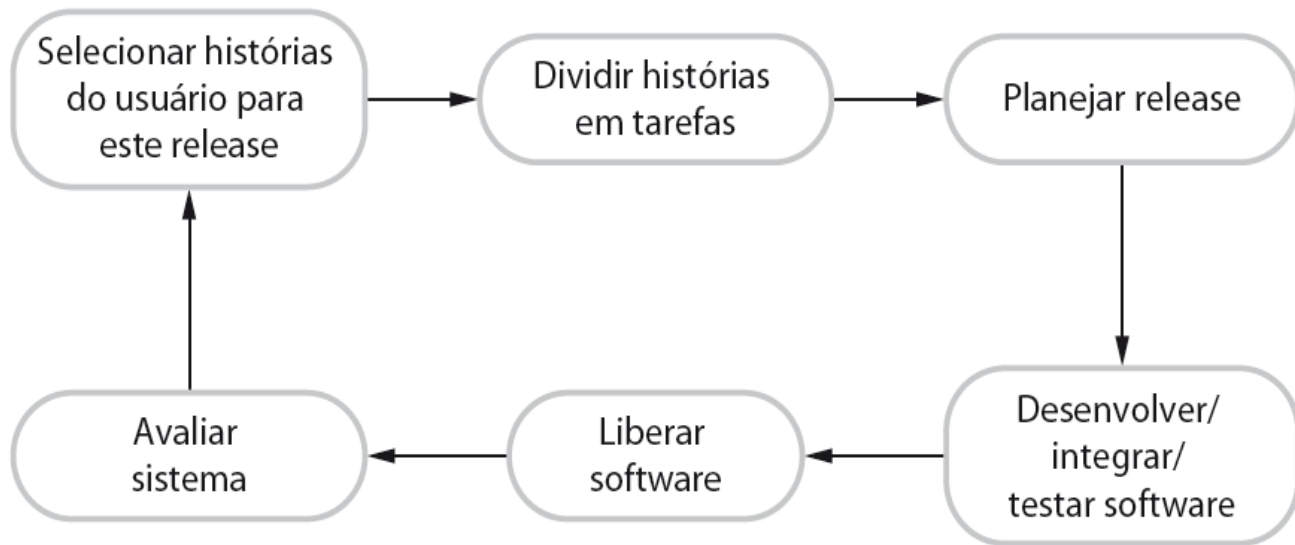
Embrace Change (aceitar mudanças)

- Mudanças ocorrerão no projeto de acordo com o crescimento do entendimento do mesmo
- Aceite as mudanças e tenha coragem para reconstruir

Quality work (trabalho de qualidade)

- A qualidade do trabalho nunca deve ser comprometida
- XP eleva a importância da codificação e do teste antes da programação – test-first programming

3. Atividades





Release - 6 Semanas


Release 1

Release 2

Release 3

Release 4

RELEASE = incremento
do produto entregue ao
cliente.

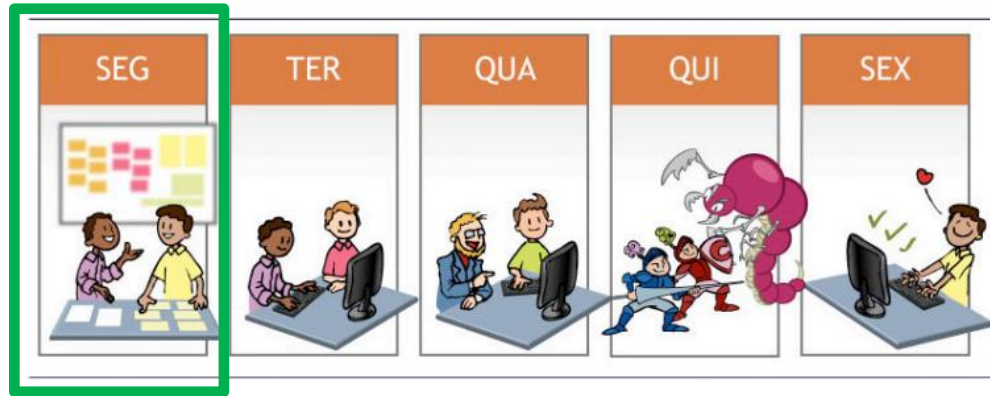


Release - 6 Semanas



ITERAÇÃO = evolução do incremento até atender aos requisitos estabelecidos.

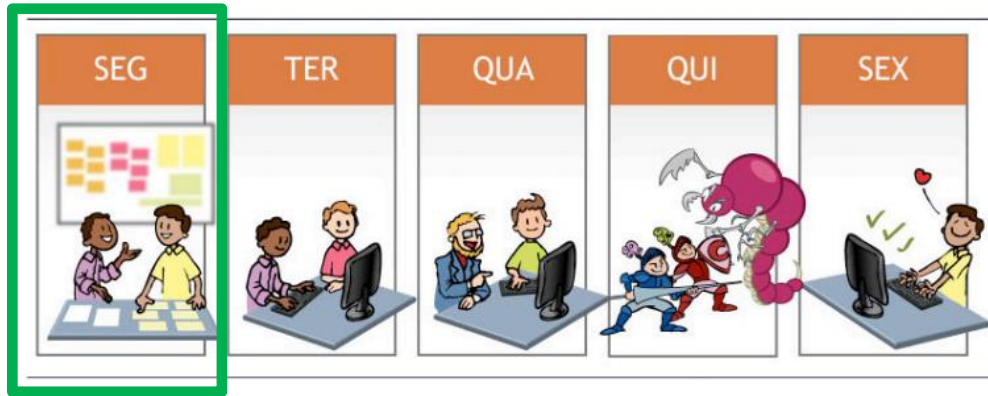
Iteração – Ciclo semanal



Jogo do Planejamento



Iteração – Ciclo semanal



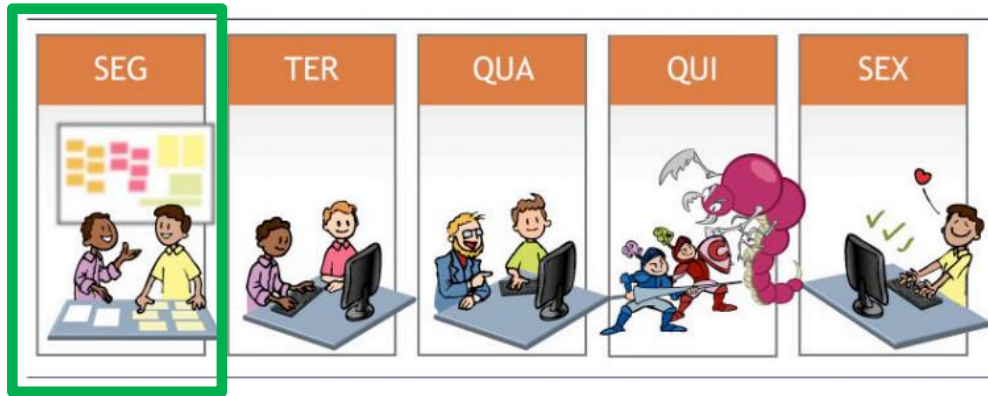
Cliente escreve histórias



Descrição:

Como vendedor eu gostaria de verificar se um livro está disponível no estoque para venda.

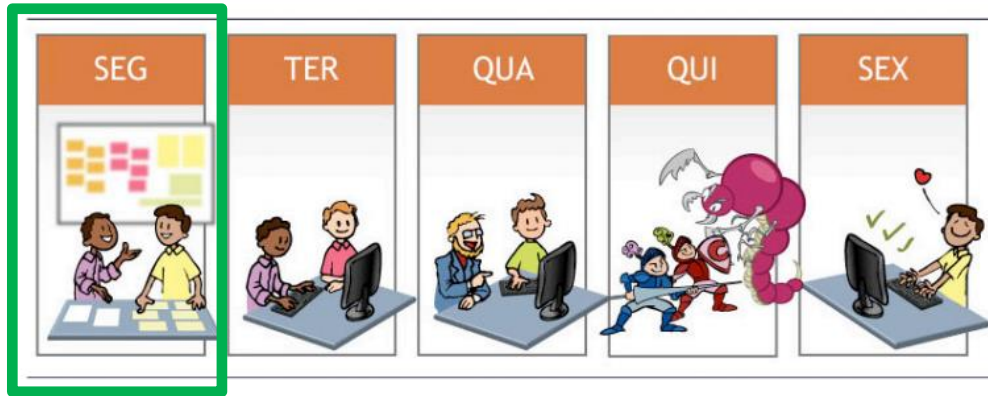
Iteração – Ciclo semanal



Desenvolvedores estimam



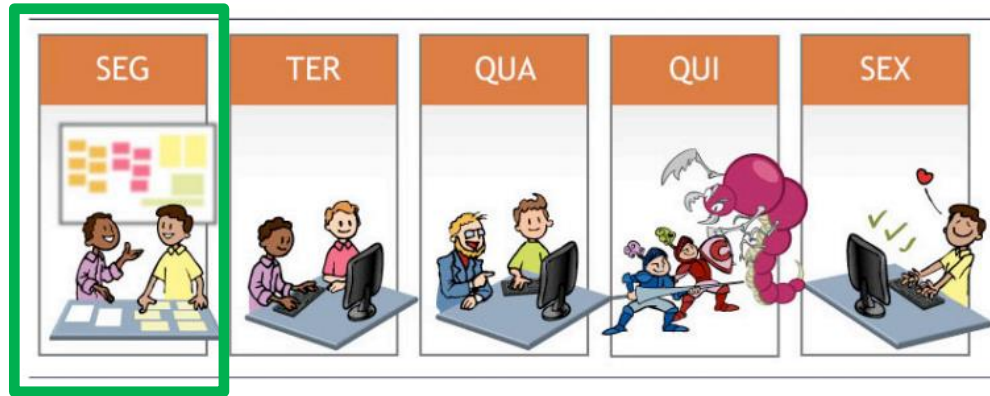
Iteração – Ciclo semanal



Cliente prioriza



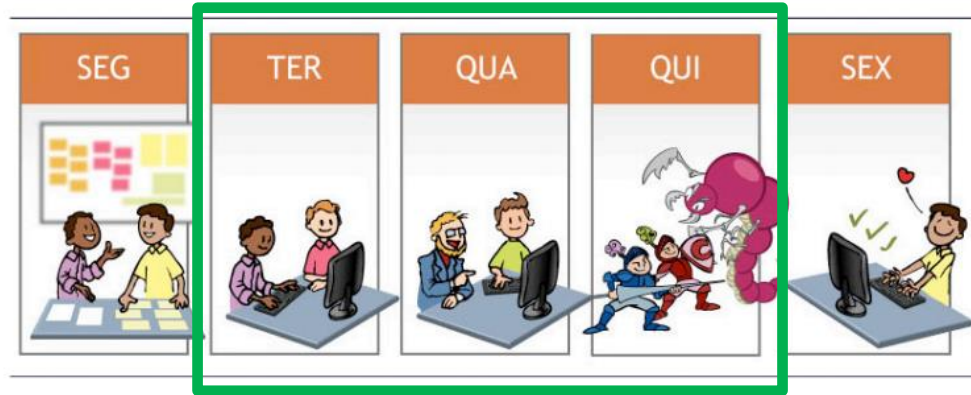
Iteração – Ciclo semanal



Quadro de funcionalidades



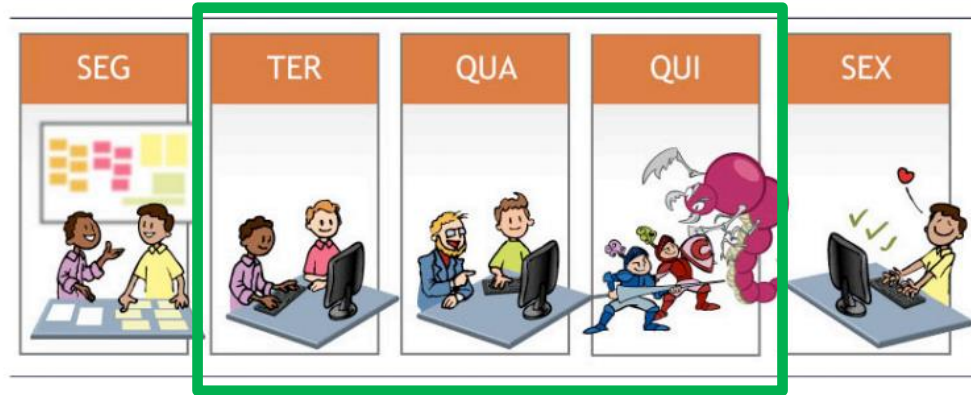
Iteração – Ciclo semanal



Reunião diária



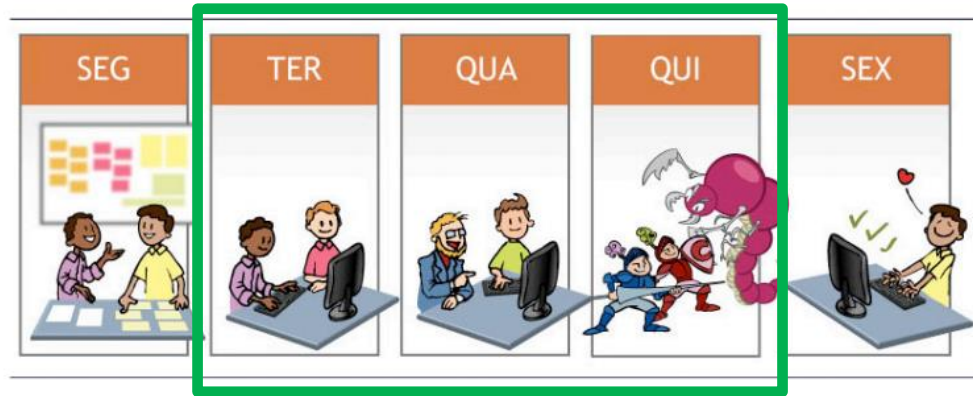
Iteração – Ciclo semanal



Cronograma



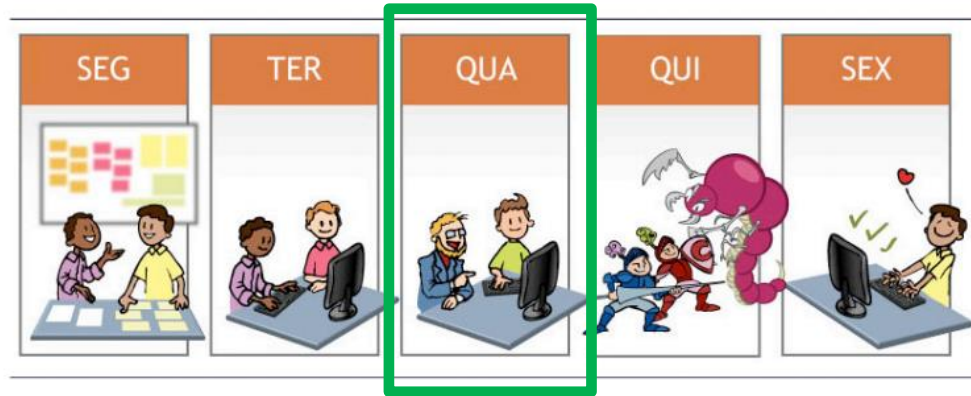
Iteração – Ciclo semanal



Desenvolvimento da aplicação (em par)



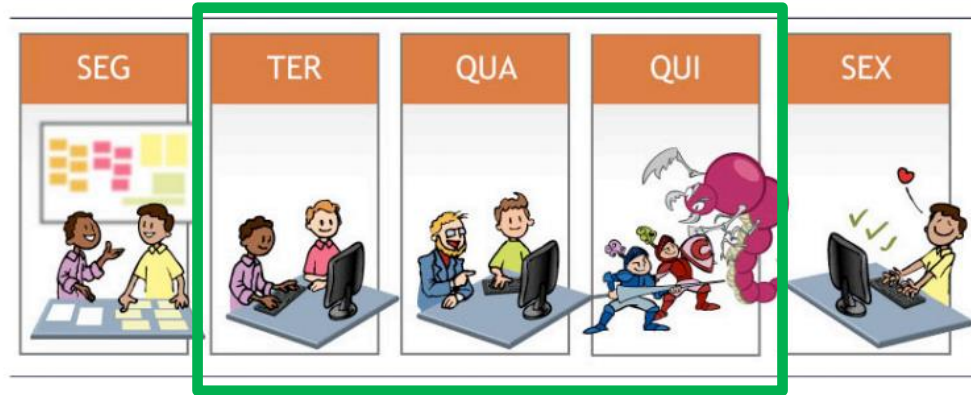
Iteração – Ciclo semanal



Acompanhamento do cliente



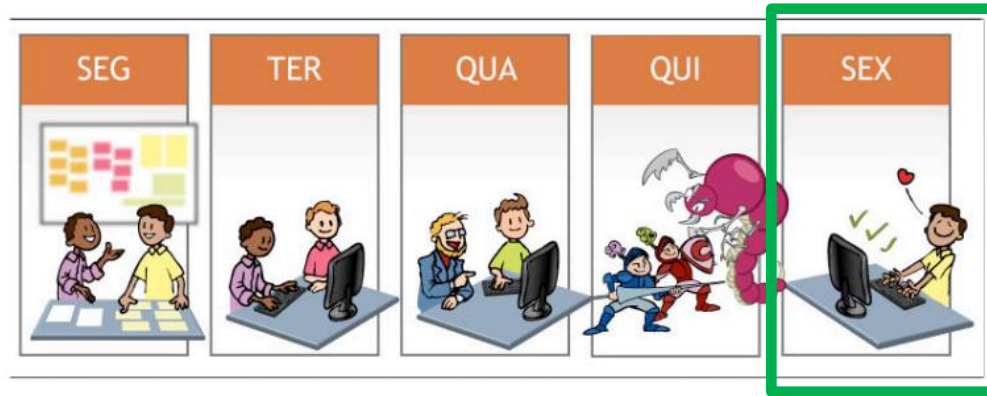
Iteração – Ciclo semanal



Funcionalidades terminam



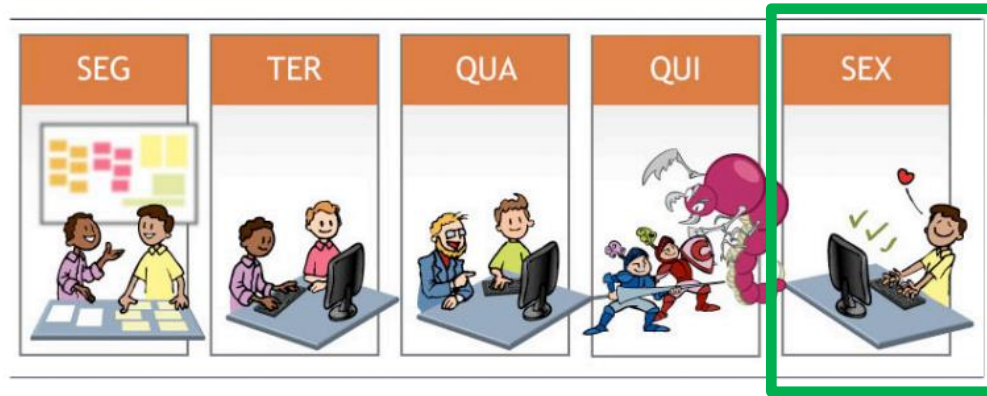
Iteração – Ciclo semanal



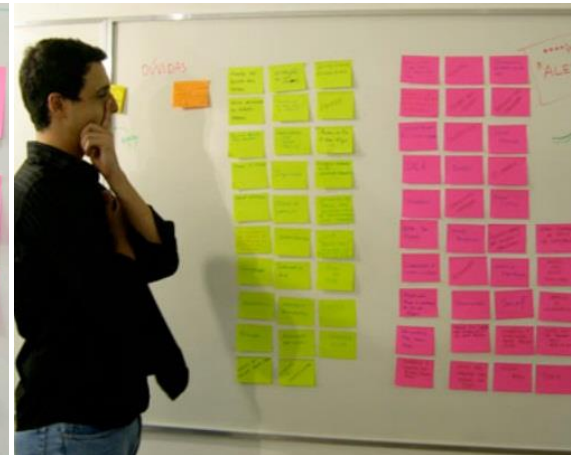
Encerramento da iteração



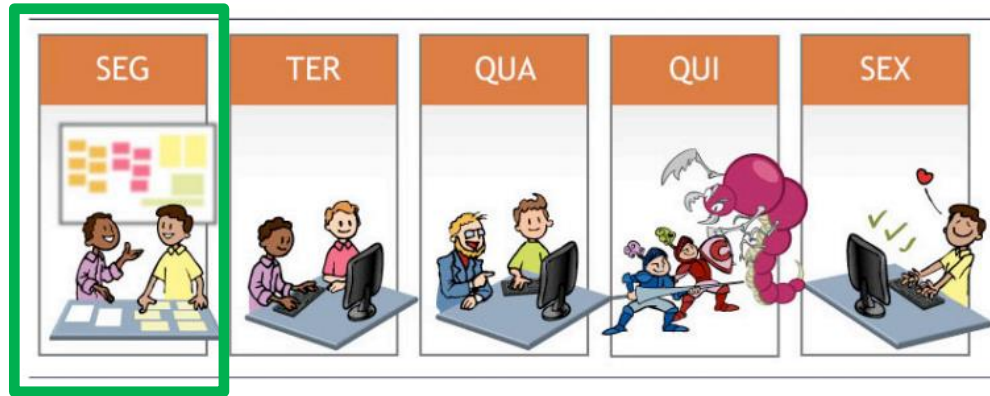
Iteração – Ciclo semanal



Retrospectiva da iteração



Iteração – Ciclo semanal



Inicia nova iteração

4. Práticas do XP

Princípio ou prática	Descrição
Planejamento incremental	Os requisitos são gravados em cartões de história e as histórias que serão incluídas em um release são determinadas pelo tempo disponível e sua relativa prioridade. Os desenvolvedores dividem essas histórias em 'Tarefas'. Veja os quadros 3.1 e 3.2.
Pequenos <i>releases</i>	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. <i>Releases</i> do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro <i>release</i> .
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento <i>test-first</i>	Um <i>framework</i> de testes iniciais automatizados é usado para escrever os testes para uma nova funcionalidade antes que a funcionalidade em si seja implementada.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente assim que encontrarem melhorias de código. Isso mantém o código simples e manutenível.

4. Práticas do XP

Princípio ou prática	Descrição
Programação em pares	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e prestando apoio para um bom trabalho sempre.
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não se desenvolvam ilhas de <i>expertise</i> . Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo. Após essa integração, todos os testes de unidade do sistema devem passar.
Ritmo sustentável	Grandes quantidades de horas-extra não são consideradas aceitáveis, pois o resultado final, muitas vezes, é a redução da qualidade do código e da produtividade a médio prazo.
Cliente no local	Um representante do usuário final do sistema (o cliente) deve estar disponível todo o tempo à equipe de XP. Em um processo de Extreme Programming, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

4. Práticas XP

- Whole Team – Equipe
- Plannig Game – Jogo do planejamento
- Customer Tests – Testes de aceitação
- Small releases – Versões pequenas
- Simple Design – Projeto simples
- Pair programming – Programação em pares
- Test-driven Development – Desenvolvimento orientado a testes (TDD)

4. Práticas XP

- Refactoring – Refinamento do projeto
- Continuous Integration – Integração contínua
- Collective Ownership – Posse coletiva
- Coding Standards – Padrões de codificação
- Metaphor – Metáfora
- Sustainable Pace – Ritmo saudável

A equipe (Whole Team)

- Todos em um projeto XP são parte de uma equipe.
- A equipe deve incluir um representante do cliente:
 - estabelece os requisitos do projeto
 - define as prioridades
 - controla o rumo do projeto
- Outros papéis assumidos pelos integrantes da equipe:
 - programadores
 - testadores (que ajudam o cliente com testes de aceitação)
 - analistas (que ajudam o cliente a definir requisitos)
 - gerente (garante os recursos necessários)
 - coach (orienta a equipe, controla a aplicação de XP)
 - tracker (coleta métricas)

Jogo do Planejamento (Planning Game)

- Dois passos chaves:
 - Planejamento de um release
- ◇ Cliente propõe funcionalidades desejadas (estórias)
- ◇ Programadores avaliam a dificuldade de implementá-las
 - Planejamento de uma iteração
- ◇ Cliente define as funcionalidades prioritárias para a iteração;
- ◇ Programadores as quebram em tarefas e avaliam o seu custo (tempo de implementação)

Teste de aceitação (Customer Tests)

- Testes de aceitação são elaborados pelo cliente
 - São testes automáticos
 - Quando rodarem com sucesso, funcionalidade foi implementada
 - Devem ser rodados novamente em cada iteração
 - Oferecem feedback: pode-se saber, a qualquer momento, quanto do sistema já foi implementado e quanto falta.

Versões Pequenas (Small Releases)

- Disponibiliza, a cada iteração, software 100% funcional
 - Benefícios do desenvolvimento disponíveis imediatamente
 - Menor risco (se o projeto não terminar, parte existe e funciona)
 - Cliente pode medir com precisão quanto já foi feito
 - Feedback do cliente permitirá que problemas sejam detectados cedo e facilita a comunicação entre o cliente e os desenvolvedores
- Lançamento pode ser destinado a
 - usuário-cliente (que pode testá-lo, avaliá-lo, oferecer feedback)
 - usuário-final (sempre que possível)

Design simples (Simple Design)

- Design está presente em todas as etapas no XP
 - Projeto começa simples e se mantém simples através de testes e refinamento do design (refactory).
 - Não é permitido que se implemente nenhuma função adicional que não será usada na atual iteração
- Implementação ideal é aquela que
 - Roda todos os testes
 - Expressa todas as idéias que você deseja expressar
 - Não contém código duplicado
 - Tem o mínimo de classes e métodos

Programação em pares (Pair programming)

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

TDD (Test-driven Development)

- "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

Refatoração (Refactoring)

- Não existe uma etapa isolada de projeto em XP
- O código é o projeto!
 - O projeto é melhorado continuamente através de refactory
 - Mudança proposital de código que está funcionando
 - Objetivos: melhorar o design, simplificar o código, remover código duplicado, aumentar a coesão, reduzir o acoplamento
 - Realizado o tempo todo, durante o desenvolvimento

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 (viabiliza lançamentos pequenos e frequentes)
 - Estimula design simples, tarefas curtas, agilidade
 - Oferece feedback sobre todo o sistema
 - Permite encontrar problemas de design rapidamente

Posse coletiva (Collective Ownership)

- Em um projeto XP qualquer dupla de programadores pode melhorar o sistema a qualquer momento.
- Todo o código em XP pertence a um único dono: a equipe
 - Todo o código recebe a atenção de todos os participantes resultando em maior comunicação
 - Maior qualidade (menos duplicação, maior coesão)
 - Menos riscos e menos dependência de indivíduos
- Todos compartilham a responsabilidade pelas alterações

Padrões de codificação (Coding Standards)

- O código escrito em projetos XP segue um padrão de codificação, definido pela equipe
 - Padrão para nomes de métodos, classes, variáveis
 - Organização do código (chaves, etc.)
- Código com estrutura familiar facilita e estimula
 - Posse coletiva
 - Comunicação mais eficiente
 - Simplicidade
 - Programação em pares
 - Refinamento do design

Metáfora (Metaphor)

- Pode ser uma analogia com algum outro sistema (computacional, natural, abstrato) que facilite a comunicação entre os membros da equipe e cliente
- Facilita a escolha dos nomes de métodos, classes, campos de dados, etc.
 - Serve de base para estabelecimento de padrões de codificação

Ritmo saudável (Sustainable Place)

- Projetos com cronogramas apertados que sugam todas as energias dos programadores não são projetos XP
 - "Semanas de 80 horas" levam à baixa produtividade
 - Produtividade baixa leva a código ruim, relaxamento da disciplina (testes, refactoring, simplicidade), dificulta a comunicação, aumenta a irritação e o stress da equipe
 - Tempo "ganho" será perdido depois
- Eventuais horas extras são aceitáveis quando produtividade é maximizada ao longo prazo

Referências

- **Sommerville, I. (2011). Engenharia de Software, 9 edição. Pearson Prentice Hall.**
- **Pressman, R., & Maxim, B. (2016). Engenharia de Software – Uma abordagem profissional-8ª Edição. McGraw Hill Brasil.**
- **Sutherland, J., van Solingen, R., & Rustenburg, E. (2011). The power of Scrum. CreateSpace.**
- **Beck, K. Extreme Programming Explained: Embrace Change, 2000. Addison-Wesley.**



OBRIGADA!

