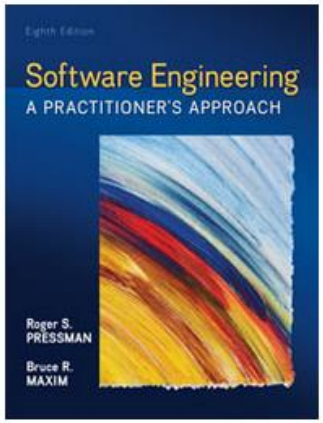
Unidade 3 – Desenvolvimento Ágil de Software – Parte 1

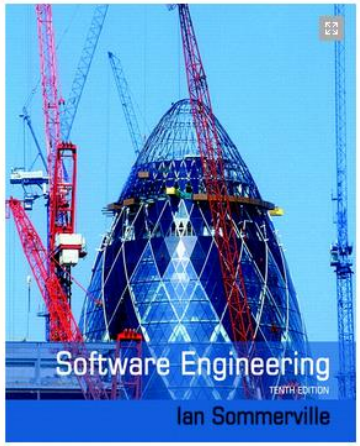
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP

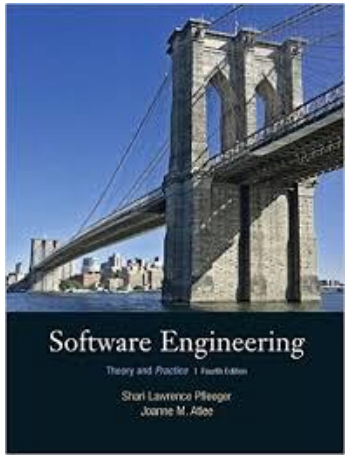
Bibliografia

- **Software Engineering – A Practitioner's Approach – Roger S. Pressman – Eight Edition – 2014**
- **Software Engineering – Ian Sommerville – 10th edition – 2015**
- **Software Engineering – Pfleeger & Atlee – Theory and Practice – 4th edition – Prentice Hall 2009**
- Engenharia de Software – Uma abordagem profissional – Roger Pressman - McGraw Hill, Sétima Edição - 2011
- Engenharia de Software – Ian Sommerville – Nona Edição – Addison Wesley, 2007
- Engenharia de Software – Teoria e Prática - Shari Lawrence Pfleeger – Editora Pearson – 3ª edição



Software Engineering: A Practitioner's Approach, 8/e





Volatilidade dos Requisitos

- Negócios operam com requisitos que mudam rapidamente.
- Há muita dificuldade em se produzir um conjunto estável de requisitos de software.
- Software precisa evoluir rapidamente para refletir as necessidades do usuário.

Modelos de Processos Tradicionais

- Os modelos de processo de software tradicionais são orientados a planos (prescritivos)
- São sistemáticos e podem representar altos custos no desenvolvimento de software.
- Nesses modelos, o cliente apresenta necessidades aos Engenheiros de Requisitos que especificam a aplicação.
- Gera-se documentação do projeto que é entregue à equipe de desenvolvimento. Em seguida, uma outra equipe de testes valida a aplicação.
- Contratos, em geral, com escopo fechado.

RUP®
Rational Unified Process®

Métodos Ágeis

- Em metodologias ágeis, toda equipe de desenvolvimento participa ativamente desde de concepção do software.
- Cliente deve participar ativamente do projeto.
- Escopo do contrato do projeto é mais flexível e com mudanças previsíveis.

Métodos Ágeis

- Foco no código.
- Baseados em abordagem iterativa de desenvolvimento de software.
- Planeja-se a entrega rápida do software em funcionamento.
- Software é evoluído de forma ágil para alcançar os requisitos em constante mudança.

Métodos Ágeis

- Tem por objetivo reduzir o overhead nos processos de software (Ex. limitando a documentação) e permitir uma resposta rápida aos requisitos em constante mudança sem retrabalho excessivo.

Princípios do Método Ágil

Princípios	Descrição
Envolvimento do cliente	Os clientes devem estar intimamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.
Entrega incremental	O software é desenvolvido em incrementos com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Membros da equipe devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Deve-se ter em mente que os requisitos do sistema vão mudar. Por isso, projete o sistema de maneira a acomodar essas mudanças.
Manter a simplicidade	Focalize a simplicidade, tanto do software a ser desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.

Desenvolvimento Ágil

- 2000, líderes da comunidade XP se reuniram em Oregon para debater a relação entre XP e processos semelhantes (Lightweight Methods).
- 2001, montanhas nevadas de Utah, representantes dos métodos leves publicaram um documento chamado "**Manifesto Ágil**".

Manifesto Ágil

- Valoriza-se mais indivíduos e interações que processos e ferramentas;
- Valoriza-se mais softwares que já funcionam que documentação abrangente;
- Valoriza-se mais colaboração do cliente que negociação contratual;
- Valoriza-se mais respostas à mudanças que seguimento de plano.

12 princípios do Manifesto Ágil

- Prioridade de entrega de software ao cliente com valor agregado;
- Mudanças de requisitos;
- Entrega contínua de versões de software, em pequena escala de tempo;
- Desenvolvedores e Pessoas de negócio devem trabalhar em conjunto;
- Equipe deve estar motivada;
- Comunicação face-a-face;
- Software em funcionamento é medida primária de progresso;
- Desenvolvedor e cliente devem estar sempre disponíveis;
- Atenção contínua ao bom projeto e tecnologias;
- Simplicidade;
- Trabalho em equipe;
- Avaliação e ajuste da equipe em intervalos regulares.

Objetivos Ágeis (Questões técnicas)

- Criar um método que reduzisse o custo de mudanças;
- Criar um método que reduzisse o tempo que o cliente recebe a entrega do software (releases);
- Criar um método focado na equipe (colaboração entre participantes);
- Criar um método capaz de fazer estimativas de funcionalidades;
- Criar um método que reduzisse gorduras do processo de software.

Métodos Ágeis

- XP – Extreme Programming;
- SCRUM;
- Crystal;
- DSDM – Método de Desenvolvimento de Sistemas Dinâmicos;
- FDD – Desenvolvimento Dirigido a Funcionalidades;
- LSD – Desenvolvimento de Software Enxuto (Lean);
- AUP – Processo de Desenvolvimento Ágil;
- MSF – Microsoft Solutions Framework.

Desenvolvimento rápido de software

- Desenvolvimento rápido de software

  - ✓ A especificação, o projeto e a implementação são intercaladas.
  - ✓ O sistema desenvolvido como uma série de versões, com os stakeholders envolvidos na avaliação das versões.
  - ✓ Geralmente as interfaces de usuário são desenvolvidas usando uma IDE e um conjunto de ferramentas gráficas.

Aplicabilidade dos métodos ágeis

- Desenvolvimento de produto, quando a empresa de software está desenvolvendo um produto pequeno ou médio para venda.
- Desenvolvimento de sistema personalizado dentro de uma organização, quando existe um compromisso claro do cliente em se envolver no processo de desenvolvimento e quando não existem muitas regras e regulamentos externos que afetam o software.
- Devido ao foco em equipes pequenas e fortemente integradas, existem problemas na escalabilidade de métodos ágeis em sistemas grandes.
- Não há muitas regras e regulamentos externos que afetam o software.

Problemas com métodos ágeis

- Pode ser difícil manter o interesse dos clientes que estão envolvidos no processo.
- Membros da equipe podem não ser adequados ao envolvimento intenso que caracteriza os métodos ágeis.
- Manter a simplicidade requer trabalho extra.
- Os contratos podem ser um problema assim como em outras abordagens que usam o desenvolvimento iterativo.
- Priorizar mudanças pode ser difícil onde existem múltiplos stakeholders.

Métodos ágeis e manutenção de software

- Duas questões muito importantes:

✓ É possível dar suporte aos sistemas que são desenvolvidos usando uma abordagem ágil, tendo em vista a ênfase no processo de minimização da documentação formal?

✓ Os métodos ágeis podem ser usados efetivamente, para evoluir um sistema em resposta a mudanças nos requisitos do cliente?

Desenvolvimento ágil e dirigido a planos

- Desenvolvimento dirigido a planos

  - ✓ Para a engenharia de software, uma abordagem dirigida a planos, é baseada em estágios de desenvolvimento separados, com os produtos a serem produzidos em cada um desses estágios planejados antecipadamente.
  - ✓ Iterações ocorrem dentro das atividades.

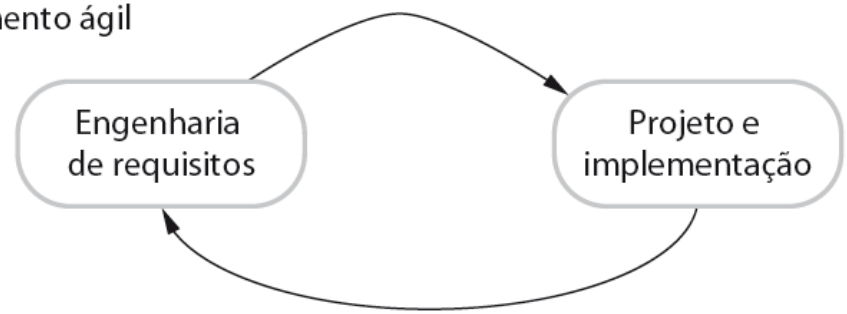
- Desenvolvimento ágil

  - ✓ Especificação, projeto, implementação e teste são intercalados e os produtos do processo de desenvolvimento são decididos através de um processo de negociação, durante o processo de desenvolvimento do software.

Especificações dirigida a planos e ágil

Desenvolvimento baseado em planos

Desenvolvimento ágil



Questões técnicas, humanas e organizacionais

- A maioria dos projetos incluem elementos de processos dirigidos a planos e ágeis. Decidir no equilíbrio depende de:

1. É importante ter uma especificação e projeto bem **detalhados** antes de passar para a implementação? Caso seja, provavelmente você precisa usar uma abordagem dirigida a planos.

2. Uma estratégia de **entrega incremental** onde você entrega o software para os clientes e recebe feedback rápido deles é possível? Caso seja, considere usar métodos ágeis.

Questões técnicas, humanas e organizacionais

3. Qual o tamanho do sistema a ser desenvolvido? Os métodos ágeis são mais efetivos quando o sistema pode ser desenvolvido com uma **equipe pequena** que pode se comunicar informalmente. O que pode não ser possível para sistemas grandes que requerem grandes equipes de desenvolvimento, nesses casos, deve ser usada uma abordagem dirigida a planos.

4. Que tipo de sistema está sendo desenvolvido? Abordagens dirigidas a planos podem ser necessárias para sistemas que requerem muita **análise** antes da implementação (ex. sistema que opere em tempo real com requisitos de temporização complexos).

5. Qual é o tempo de vida esperado para o sistema? Sistemas com **longo tempo de vida** podem precisar de mais documentação de projeto para comunicar as intenções originais dos desenvolvedores do sistema para a equipe de suporte.

Questões técnicas, humanas e organizacionais

6. Como está organizada a equipe de desenvolvimento? Se a **equipe** de desenvolvimento está **distribuída** ou se parte do desenvolvimento está sendo terceirizado você pode precisar desenvolver documentos de projeto para que haja comunicação entre as equipes de desenvolvimento.

7. Existem questões culturais ou organizacionais que podem afetar o desenvolvimento do sistema? As organizações tradicionais de engenharia têm uma **cultura** de desenvolvimento dirigido a planos, o que é padrão em engenharia.

8. Quão bons são os projetistas e os programadores da equipe de desenvolvimento? É dito que os métodos ágeis requerem um nível de **habilidade** mais alto do que as abordagens dirigidas a planos, nas quais os programadores simplesmente traduzem um projeto detalhado em código.

9. O sistema está sujeito a **regulamentação externa**? Se o sistema precisa ser aprovado por um regulador externo (ex. O FAA aprova softwares críticos para a operação de um avião) então provavelmente requisitaram a você a produção de documentação detalhada como parte da documentação de segurança do sistema.

Métodos ágeis em grandes projetos

- Os métodos ágeis provaram-se bem-sucedidos para projetos **pequenos** e **médios** que podem ser desenvolvidos por uma **equipe pequena** e **localizada**.
- É dito que o sucesso desses métodos ocorre devido a melhorias na **comunicação**, as quais são possíveis quando todos estão trabalhando juntos.
- Para grandes projetos, podem haver **múltiplas equipes** de desenvolvimento, talvez trabalhando em localizações diferentes, as vezes, trabalhando em diferentes fusos horários.
- Sistemas de grande porte podem ser restringidos por **regras externas** e **regulamentação** que podem exigir documentação detalhada. Pode haver um conjunto diversificado de stakeholders.

Origem do Método XP

Kent Beck

Ward Cunninghan

- Metodologia criada em 1997
- Aplicada inicialmente em grande projeto da Chrysler (2000 classes e 30.000 métodos em um ano)

Extreme Programming

- Método ágil muito conhecido e amplamente usado;
- 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.

Equipe de Trabalho

- Com metodologias ágeis, toda a equipe participa do projeto;
- Cliente descreve cenários (narrações,estórias) do que está sendo pedido;
- Programadores que participam da equipe também atuam como analistas;
- Essas estórias são priorizadas com relação a custo e esforço de desenvolvimento;
- Cliente se compromete a disponibilizar tempo e dedicação à essa fase de definição do escopo do projeto (duração média de uma semana);
- Cliente deve validar (feedback) o que foi feito durante a semana e novos passos são planejados.

XP
EXTREME PROGRAMMING

- Desenvolvimento modular;
- Semanalmente, cliente deve acompanhar e validar o desenvolvimento de cada módulo do projeto;
- Aplicam-se princípios de Engenharia de Software e Orientação a Objetos;
- Programação em pares;
- Dúvidas relativas ao projeto sempre são tratadas com o cliente;
- Cliente participa efetivamente até a entrega de todos os módulos (process

Características do Método XP

- Programação em resulta em produtividade.
- Programação orientada a testes.
- Código clean.
- Cada membro da equipe tem contato com o código fonte.
- Produção de código manutenível.
- Relação amigável com o cliente, redundando em maior facilidade em cumprir contrato.
- Conflitos resolvidos com comunicação e diplomacia.

Valores da XP

- São princípios e qualidades que nortearão as práticas da XP;

Valores da XP

- Simplicidade;
- Feedback;
- Comunicação;
- Coragem e Respeito.

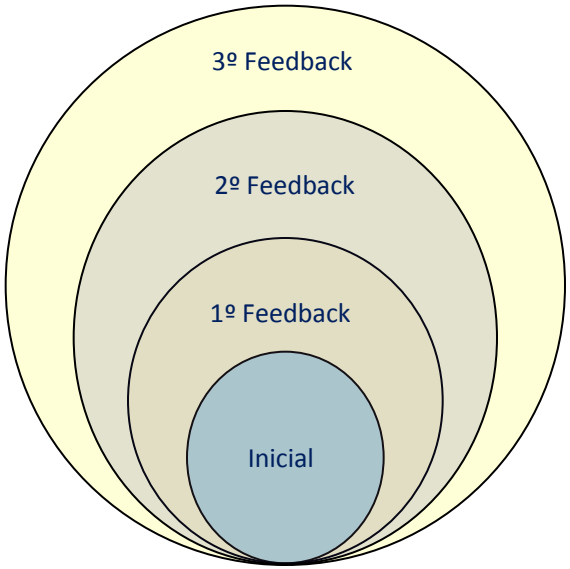
Simplicidade

- Estabelecer a solução mais simples possível para as funcionalidades do sistema;
- Não significa sistemas sem recursos ou sem funcionalidades;
- Foco nas funções essenciais, de forma a tornar o projeto simples;
- Abolir funções raramente executadas ou que nem foram solicitadas pelo cliente;
- Código de software não utilizável é anti-ágil;
- Funcionalidade deve ser simples mas operacional;
- Manter o menor número de classes e métodos possíveis;
- Conhecer bem as API's e frameworks de Linguagens de programação.

AS COISAS SIMPLES DA VIDA

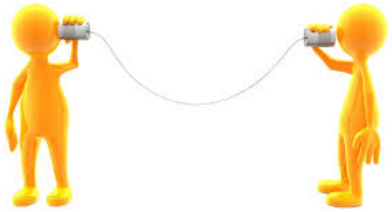
Feedback

- Parte de um outro valor que é a comunicação;
- Tem por objetivos: orientar, evoluir, alinhar e aprender com o sistema;
- Cliente analisa o que foi produzido e pode-se corrigir a rota do desenvolvimento;



Comunicação

- Informações relacionadas ao software devem ser comunicadas;
- Deve ser constante para que a informação circule dentro da equipe e entre o cliente;
- Todas as atividades feitas dentro da equipe carecem de comunicação;
- Importante para a gestão de pessoas dentro do projeto;
- Dimensão bem maior onde o feedback está incluso;
- Deve ser eficiente (tempo) e eficaz (resultado);
- Deve ser direta, bidirecional, concisa, expressiva;
- Linguagem textual (documentos) podem apresentar ambiguidades;
- O que deve ser feito é comunicado face a face;
- Documentos são gerados para funcionalidades prontas;



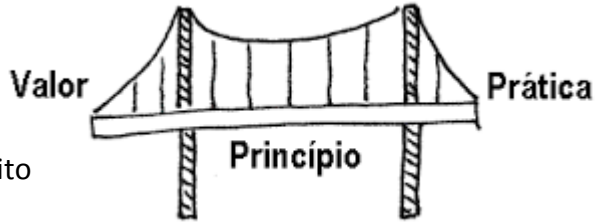
Coragem e Respeito

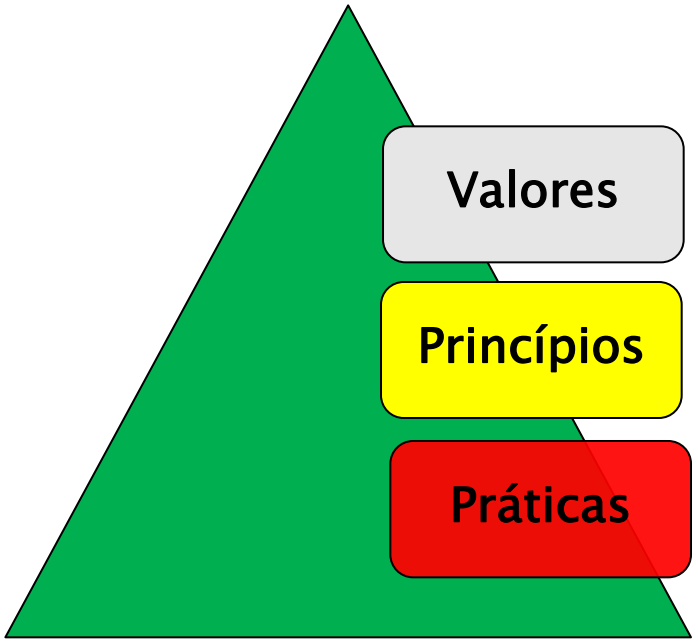
- Respeito podem evitar discussões judiciais;
- Respeito deve ocorrer entre desenvolvedor e cliente, cliente e desenvolvedor, membros da equipe e respeito próprio;
- Respeitos mútuos contribuem para o bom fluxo do projeto;
- Coragem para aplicar as práticas da XP:
  - ✓ Manter os valores da Simplicidade;
  - ✓ Desenvolver em releases;
  - ✓ Cliente deve participar do projeto;
  - ✓ Programação em Par;
  - ✓ Refatorar código;
  - ✓ Testes automatizados;
  - ✓ Utilizar Estórias;
  - ✓ Código compartilhado;
  - ✓ Integração Contínua constante;
  - ✓ Ritmo Sustentável;
  - ✓ Não ter documentação formal;
  - ✓ Contrato de Escopo Variável.

Princípios na XP

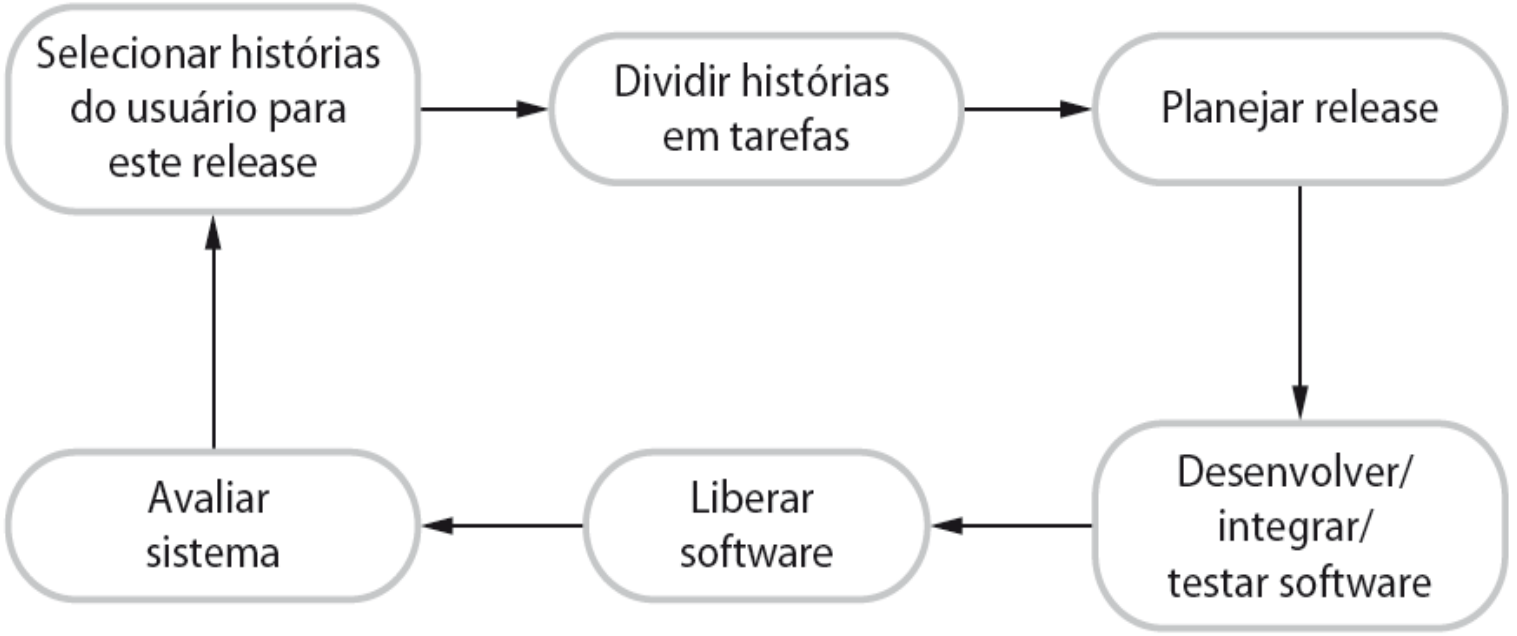
- São uma ponte de Ligação entre “valores” e “práticas” da XP

Simplicidade
Feedback
Comunicação
Coragem e Respeito





O ciclo de um release em Extreme Programming



Práticas do Extreme Programming

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'.
Pequenos *releases*	Em primeiro lugar, desenvolve-se um conjunto mínimo de funcionalidades útil, que fornece o valor do negócio. *Releases* do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro *release*.
Projeto simples	Cada projeto é realizado para atender às necessidades atuais, e nada mais.
Desenvolvimento *test-first*	Um *framework* 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.

Práticas do Extreme Programming

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 *expertise*. 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.

Cenários de requisitos

- Em XP, 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 estórias dos usuários.
- Esses são escritos em cartões e a equipe de desenvolvimento os divide em tarefas de implementação. Essas tarefas são a base das estimativas de cronograma e custo.
- O cliente escolhe as estórias que serão incluídas no próximo release baseando-se nas suas prioridades e nas estimativas de cronograma.

Uma estória de 'prescrição de medicamentos'

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'.

Exemplos de cartões de tarefa para a prescrição de medicamentos

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'.

Refatoração

- A equipe de programação busca possíveis melhorias de software e as faz mesmo quando essas não são uma necessidade imediata.
- O que melhora a inteligibilidade do software e reduz a necessidade de documentação.
- Torna-se mais fácil fazer mudanças porque o código é bem construído e limpo.
- No entanto, algumas mudanças requerem refatoração da arquitetura, o que é muito mais caro.

XP
EXTREME PROGRAMMING

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.

Desenvolvimento test-first

- Escrever testes antes do código esclarece os requisitos que devem ser implementados.
- Os testes são escritos na forma de programas ao invés de dados para que possam ser executados automaticamente.
- Os testes incluem checagem de que foram executados corretamente.
- Geralmente conta com um framework de testes como o **Junit**.
- Todos os testes anteriores e novos são executados automaticamente quando uma nova funcionalidade é adicionada, para checar se a nova funcionalidade não introduziu erros.

A automação de testes

- A automação de testes significa que os testes são escritos como componentes executáveis antes que a tarefa seja implementada.

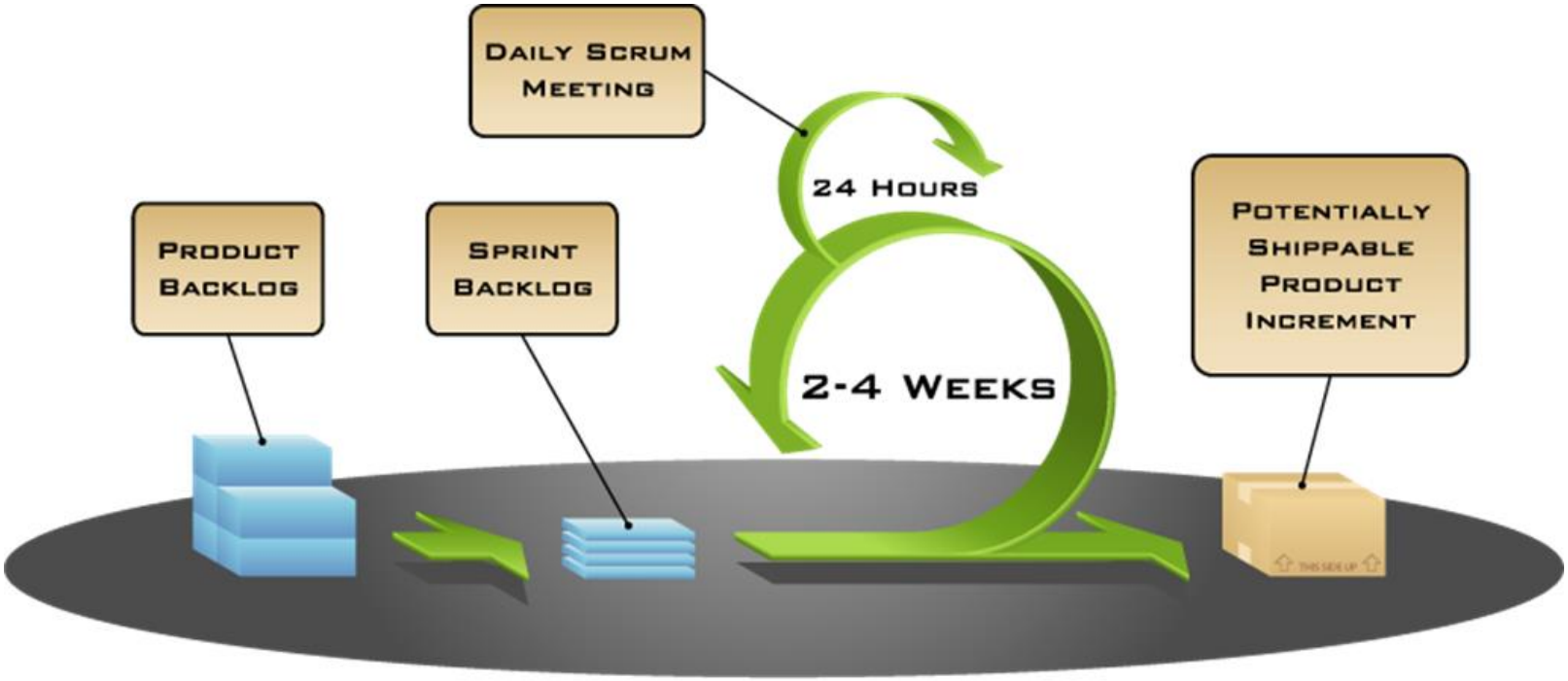
  ✓ Esses componentes de teste devem ser autômatos, devem simular a submissão de entrada para ser testada e devem avaliar se o resultado atende à especificação de saída. Um framework de testes automatizados (ex. **Junit**) é um sistema que facilita a escrita de testes executáveis e a submição de um conjunto de testes para execução.

- Como os testes são automatizados, sempre existe um conjunto de testes que podem ser rapidamente executados.

  ✓ Quando qualquer funcionalidade é adicionada ao sistema os testes podem ser executados e problemas que o novo código possa ter introduzido podem ser percebidos imediatamente.

Scrum

- Concebido por Jeff Sutherland em 1990;
- Metodologia consistente com o manifesto ágil;
- Em cada atividade ocorrem tarefas chamadas **Sprint** (unidade de trabalho);
- **Backlog**: registro de pendências com prioridades dos requisitos ou funcionalidades;
- Reuniões Scrum: curtas (tipicamente 15 minutos) realizadas diariamente por toda a equipe;



Scrum

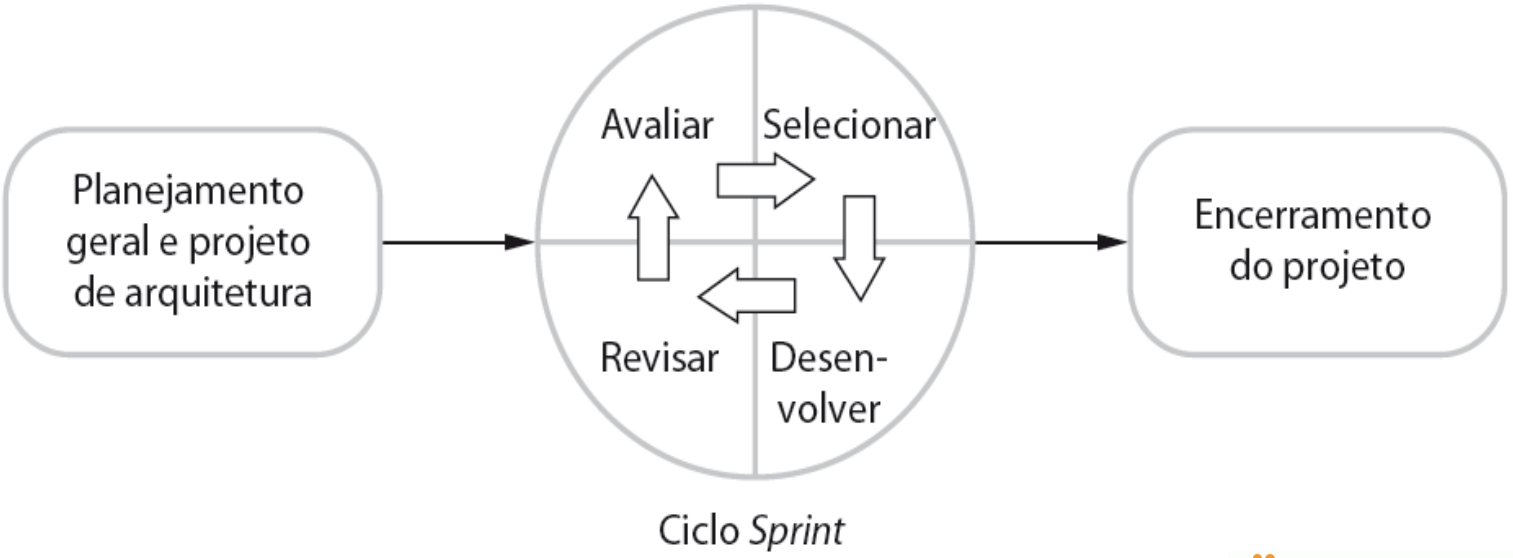
- Existem três fases no Scrum:

1. A fase inicial é uma fase de planejamento em que se estabelece os objetivos gerais do projeto e se projeta a arquitetura do software.

2. Essa é seguida por uma série de ciclos de Sprint, em que cada ciclo desenvolve um incremento do sistema.

3. A fase de encerramento do projeto finaliza o projeto, completa a documentação necessária como manuais de usuário e avalia as lições aprendidas no projeto.

O processo Scrum



O ciclo de Sprint

- Os **Sprints** possuem um deadline definido, geralmente de 2 a 4 semanas.
- Eles correspondem ao desenvolvimento de um release de um sistema em XP.
- O ponto de partida de planejamento é o backlog de produto, que é a lista de trabalho a ser feito no projeto.
- A fase de seleção envolve a seleção das características e funções que serão desenvolvidas durante o Sprint, pela equipe do projeto que trabalha com o cliente.

O ciclo de Sprint

- Assim que isso é definido, a equipe se organiza para desenvolver o software.
- Durante esse estágio a equipe é isolada do cliente e da organização, com todas as comunicações canalizadas por meio do chamado "Scrum Master".
- A função do Scrum Master é proteger a equipe de desenvolvimento de distrações externas.
- Ao final do **Sprint** o trabalho feito é revisto e apresentado aos stakeholders. Em seguida, o próximo ciclo de Sprint se inicia.

Trabalho em equipe no Scrum

- O **Scrum Master** é um facilitador que organiza reuniões diárias, mantêm o backlog do trabalho a ser feito, grava decisões e mede o processo usando o backlog e comunica-se com os clientes e a gerência fora da equipe.

- O **PO** – **Product Owner** desempenha o papel de cliente, criando e priorizando o backlog. Deve ter uma visão do negócio.

- A equipe inteira comparece às reuniões diárias curtas nas quais todos os membros da equipe compartilham informações, descrevem seu progresso desde a última reunião, descrevem os problemas que surgiram e o quê está planejado para o dia seguinte.

  - ✓ Com isso, todos na equipe sabem o quê está acontecendo e, caso ocorra um problema, podem replanejar o trabalho a curto prazo para lidar com a situação.