Qualidade

(capítulo 3 de Engenharia de Software, Sommerville)

Profº Mauricio Antonio Ferste – mauricio.ferste@docente.pr.senac.br

Tópicos apresentados

- Métodos ágeis
- Desenvolvimento ágil e dirigido a planos
- Extreme Programming
- Gerenciamento ágil de projetos
- Escalamento de métodos ágeis

Desenvolvimento rápido de software

- Atualmente, a entrega e o desenvolvimento rápidos têm sido geralmente, o requisito mais importante nos sistemas de software:

  - ✓ Os negócios operam com requisitos que mudam rapidamente e é praticamente impossível produzir um conjunto estável de requisitos de software.

  - ✓ O software precisa evoluir rapidamente para refletir as necessidades de negócio em constante mudança.

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.

Métodos ágeis

- A insatisfação com o overhead que envolve os métodos de projeto de software dos anos de 1980 e 1990 levou a criação de métodos ágeis. Esses métodos:

  - ✓ Têm foco no código ao invés de no projeto.
  - ✓ São baseados em uma abordagem iterativa de desenvolvimento de software.
  - ✓ São planejados para entregar rapidamente o software em funcionamento e evoluí-lo rapidamente para alcançar os requisitos em constante mudança.

- O objetivo dos métodos ágeis é reduzir o overhead nos processos de software (ex. limitando a documentação) e permitir uma resposta rápida aos requisitos em constante mudança sem retrabalho excessivo.

Manifesto ágil

- Estamos descobrindo melhores formas de desenvolver softwares e ajudar outros a fazê-lo também. Através desse trabalho, valorizamos mais:

  - ✓ Indivíduos e interações, ao invés de processos e ferramentas.
  - ✓ Softwares que já funcionam ao invés de documentação abrangente.
  - ✓ Colaboração do cliente ao invés de negociação contratual.
  - ✓ Resposta a mudanças ao invés de seguir um plano.

- O que significa que existe valor nos itens a direita, mas que valorizamos mais os itens a esquerda.

Os princípios dos métodos ágeis

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.

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 metodos ágeis em sistemas grandes.

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.
- Priorizar mudanças pode ser difícil onde existem múltiplos stakeholders.
- Manter a simplicidade requer trabalho extra.
- Os contratos podem ser um problema assim como em outras abordagens que usam o desenvolvimento iterativo.

Métodos ágeis e manutenção de software

- A maioria das organizações gasta mais na manutenção de softwares existentes do que no desenvolvimento de softwares novos. Devido a isso, para que os métodos ágeis obtenham sucesso, os softwares devem receber tanta manutenção quanto o desenvolvimento original.

- 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?

- Podem ocorrer problemas no caso do tempo original de

Desenvolvimento ágil e dirigido a planos

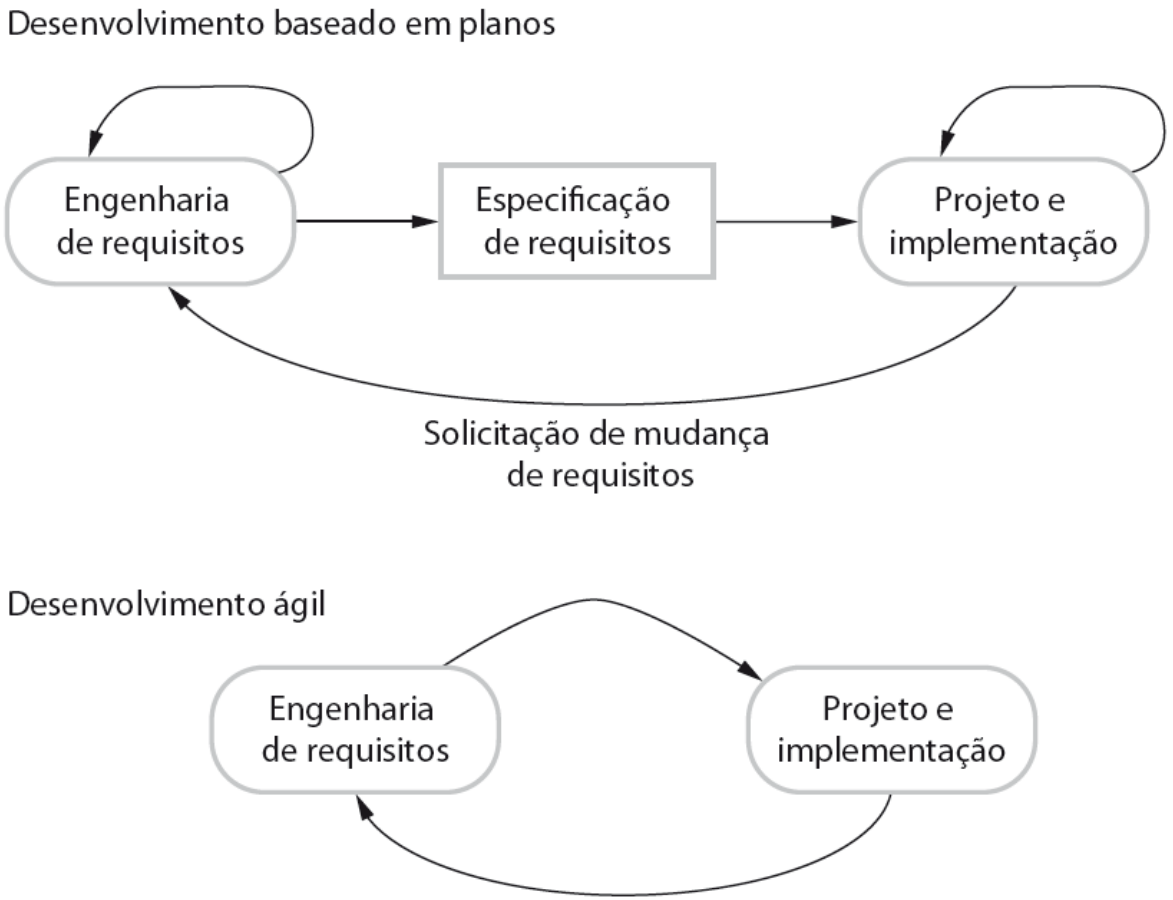
- Desenvolvimento dirigido a planos

  - ✓ 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.
  - ✓ O desenvolvimento incremental é possível no modelo cascata - dirigido a planos.
  - ✓ 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



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

Questões técnicas, humanas e organizacionais

6. Quais tecnologias estão disponíveis para manter o desenvolvimento do sistema? Métodos ágeis dependem de boas ferramentas para acompanhar um sistema em evolução.

7. 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.

8. 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.

Questões técnicas, humanas e organizacionais

9. O 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.

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

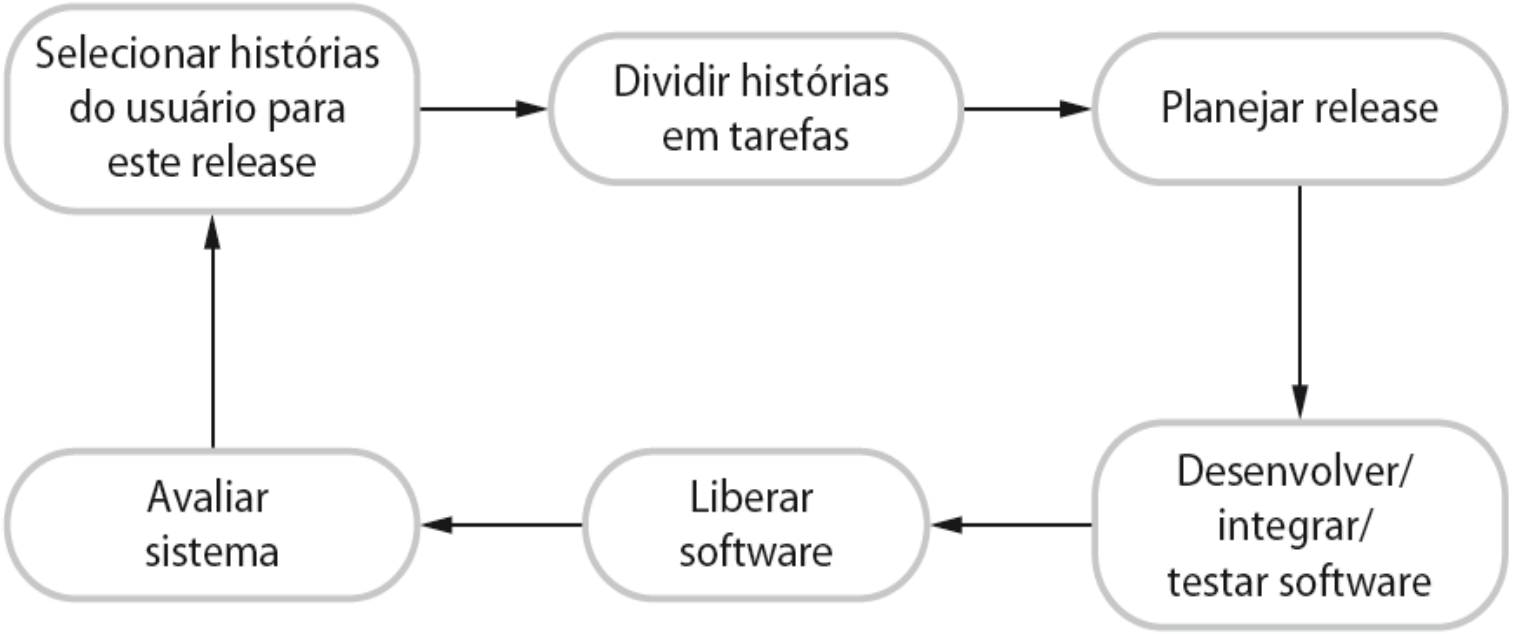
Extreme Programming

- Talvez seja o método ágil mais 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.

Princípios os métodos ágeis e do XP

- O desenvolvimento incremental é mantido através de releases de sistema pequenos e frequentes.
- O envolvimento do cliente significa compromisso do cliente com a equipe em tempo integral.
- 'Pessoas e não processos’ por meio de programação em pares, propriedade coletiva do código e um processo que evita longas horas de trabalho.
- Mudanças suportadas através de releases regulares de sistema.
- Manter a simplicidade através de constante refatoração de código.

O ciclo de um release em Extreme Programming



Práticas do Extreme Programming (a)

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 *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 (a)

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'.

XP e mudanças

- O senso comum da engenharia de software diz que se deve projetar pensando em mudanças.
- Vale a pena gastar tempo e esforço antecipando as mudanças já que, posteriormente, esse esforço reduz custos no ciclo de vida.
- No entanto, o XP afirma que isso não vale a pena já que as mudanças não podem ser antecipadas de forma confiável.
- Ao invés disso, propõe melhorias constantes do código (refatoração) para tornar as mudanças mais fáceis quando essas precisam ser implementadas.

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.

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.

Pontos Importantes

- Os métodos ágeis são métodos de desenvolvimento incremental centrados no desenvolvimento rápido, frequentes releases de software, redução de overheads de processo e produção de código de alta qualidade. Eles envolvem o cliente diretamente no processo de desenvolvimento.

- A decisão de quando usar uma abordagem ao desenvolvimento ágil ou dirigida a planos deve depender do tipo de software que está sendo desenvolvido, da capacidades da equipe de desenvolvimento e da cultura da compania desenvolvedora do sistema.

- O Extreme Programming é um método ágil bem conhecido que integra uma série de boas práticas de programação como por exemplo releases de software frequentes, melhorias contínuas de software e participação do cliente na equipe de desenvolvimento.

Testes em XP

- Em XP, os testes são fundamentais, XP desenvolveu uma abordagem em que o programa é testado depois de que cada alteração é feita.

- Características de testes em XP:

1. Desenvolvimento *test-first*.

2. Desenvolvimento de testes incrementais a partir de cenários.

3. Envolvimento do usuário no desenvolvimento de testes e validação.

4. Cada vez que um novo release é construído, são usados *frameworks* de testes automatizados para executarem todos

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.

Envolvimento do cliente

- A função do cliente no processo de testes é ajudar a desenvolver testes de aceitação para as estórias que serão implementadas no próximo release do sistema.

- O cliente, parte da equipe, escreve testes conforme o desenvolvimento prossegue. Todo código novo é validado para garantia de que seja o que o cliente precisa.

- No entanto, a pessoa que assume a função de cliente tem tempo limitado disponível e não pode trabalhar em tempo integral com a equipe de desenvolvimento.

- Eles podem pensar que prover os requisitos seja contribuição suficiente e se tornarem relutantes em se envolverem no processo de testes.

Descrição de caso de teste para verificação de dose

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.

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 saida. 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 e facilmente executados.

  - ✓ Quando qualquer funcionalidade é adicionada ao sistema os testes podem ser executados e problemas que o novo

Dificuldades dos testes em XP

- Os programadores preferem programar a testar e as vezes eles usam atalhos quando escrevem esses testes. Por exemplo, eles podem escrever testes incompletos que não avaliam todas as possíveis exceções que podem ocorrer.

- Alguns testes podem ser muito difíceis de serem escritos de forma incremental. Por exemplo, em uma interface de usuário complexa, geralmente é difícil escrever testes de unidade para o código que implementa a 'lógica de display' e o fluxo de trabalho entre telas.

- É difícil julgar se um conjunto de testes está completo.

- Embora você tenha vários testes de sistema, o conjunto dos testes pode não prover uma cobertura completa.

Programação em pares

- Em XP, programadores trabalham em pares sentando junto para desenvolver código.
- Isso ajuda a desenvolver propriedade coletiva do código e espalha o conhecimento na equipe.
- Serve como um processo de revisão informal pois cada linha do código é observada por mais de uma pessoa.
- Encoraja a refatoração pois toda a equipe pode se beneficiar dessa atividade.
- Avaliações sugerem que a produtividade do desenvolvimento com programação em pares é similar a de duas pessoas trabalhando independentemente.

Programação em pares

- Na programação em pares os programadores sentam-se juntos na mesma estação de trabalho para desenvolver softwares.

- Os pares são criados dinamicamente para que todos os membros da equipe trabalhem com cada um dos outros membros durante o processo de desenvolvimento.

- O compartilhamento de conhecimento que acontece durante a programação em pares é muito importante por reduzir os riscos gerais de um projeto quando um membro da equipe vai embora.

- A programação em pares não é necessariamente ineficiente e existem evidências de que o trabalho em pares é mais eficiente do que 2 programadores trabalhando separadamente.

Vantagens da programação em pares

1. Apóia a idéia da propriedade coletiva e responsabilidade pelo sistema.

   ✓ Os indivíduos não são responsabilizados por problemas no código. Ao invés disso, a equipe tem responsabilidade coletiva na solução desses problemas.

2. Funciona como um processo de revisão informal porque cada linha de código é observada por pelo menos duas pessoas.

3. Ajuda a apoiar a refatoração, que é um processo de melhoria do software.

   ✓ Em processos nos quais a programação em pares e a propriedade coletiva são usados, outros se beneficiam imediatamente da refatoração, o que provavelmete fará

Gerenciamento ágil de projetos

- A principal responsabilidade de gerentes de projeto de software é gerenciar o projeto para que o software seja entregue em tempo e dentro do orçamento planejado para o projeto.

- A abordagem padrão para o gerenciamento de projeto é dirigida a planos.

- Os gerentes estruturam um plano para o projeto mostrando o que deve ser entregue, quando deve ser entregue e quem irá trabalhar no desenvolvimento dos entregáveis (“deliverables”).

- O gerenciamento ágil de projetos requer uma abordagem diferente, adaptada ao desenvolvimento incremental e aos pontos fortes particulares dos métodos ágeis.

Scrum

- A abordagem Scrum é um método ágil genérico mas seu foco é na gerência de desenvolvimento iterativo ao invés de práticas ágeis específicas.

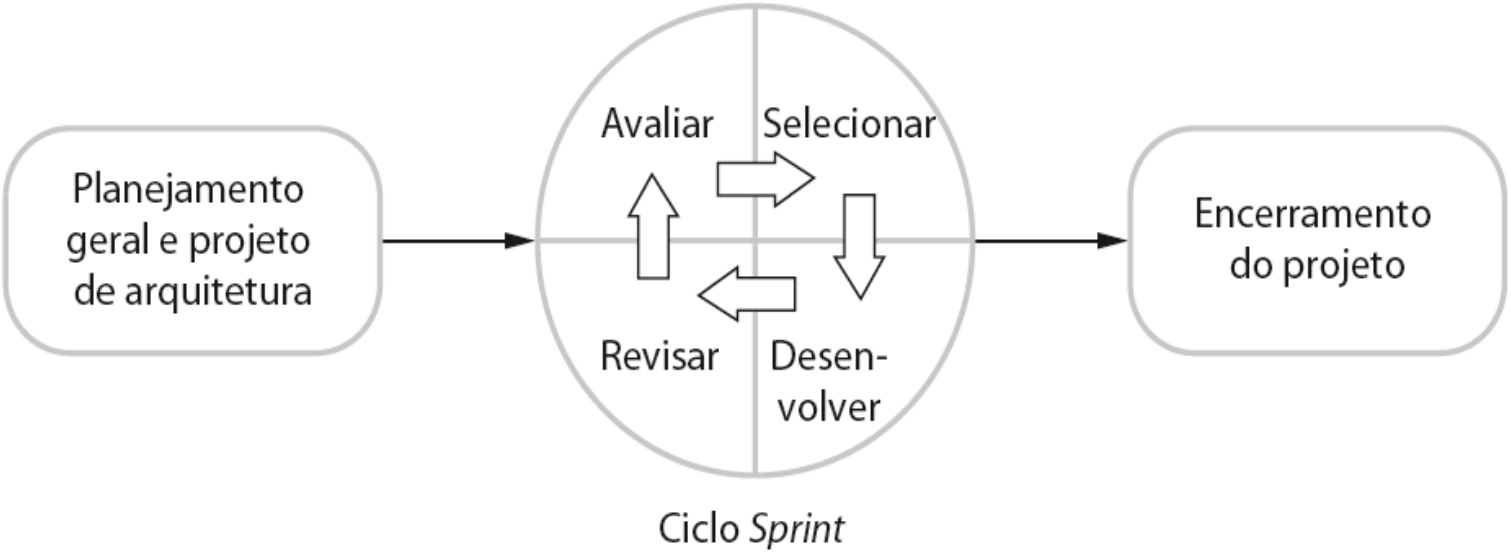
- Existem três fases no Scrum:

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 frames de ajuda do sistema e manuais de usuário e avalia as lições aprendidas no projeto.

O processo Scrum



Ciclo *Sprint*

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 orgainzação, 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. Assim o próximo ciclo de Sprint começa.

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, mede o processo usando o backlog e comunica-se com os clientes e a gerência fora da equipe.

- 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.

Benefícios do Scrum

- O produto é dividido em um conjunto de partes gerenciáveis e intelígiveis.
- Requisitos instáveis não impedem o progresso.
- Toda a equipe tem visão de tudo e consequentemente a comunicação da equipe é melhorada.
- Os clientes recebem a entrega dos incrementos no tempo certo, além do feedback de como o produto funciona.
- Se estabelece a confiança entre os clientes e os desenvolvedores e se cria uma cultura positiva na qual todos acham que o projeto dará certo.

Escalamento de métodos ágeis

- 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.

- A escalamento dos métodos ágeis envolve mudá-los para que lidem com projetos maiores e mais longos onde existem múltiplas equipes de desenvolvimento, talvez trabalhando em localizações diferentes.

Desenvolvimento de sistemas de grande porte

- Geralmente, os sistemas de grande porte são coleções de sistemas separados que se comunicam, e nos quais as equipes desenvolvem cada sistema separadamente. Frequentemente essas equipes trabalham em locais diferentes, as vezes em fuso-horários diferentes.

- Os sistemas de grande porte são '*brownfield systems*', o que significa que incluem e interagem com vários sistemas existentes. Vários dos requisitos de sistema se preocupam com essa interação o que não permite flexibilidade e desenvolvimento incremental.

- Vários sistemas são integrados para criar um sistema, e uma fração significante do desenvolvimento é voltada para a configuração do sistema ao invés do desenvolvimento do código original.

Desenvolvimento de sistemas de grande porte

- Os sistemas de grande porte e seus processos de desenvolvimento geralmente são restringidos por regras externas e regulamentações que limitam a forma como podem ser desenvolvidos.

- Os sistemas de grande porte tem um tempo de aquisição e desenvolvimento longo. Durante esse período, é difícil manter equipes coesas, que conhecem o sistema já que inevitavelmente as pessoas podem sair para outros trabalhos e projetos.

- Geralmente, os sistemas de grande porte tem um conjunto diversificado de stakeholders. É praticamente impossível envolver todos eles no processo de desenvolvimento.

Perspectiva scaling out e scaling up

- ‘Scaling up’ se preocupa em usar métodos ágeis para desenvolver sistemas de software de grande porte que não podem ser desenvolvidos por uma equipe pequena.

- ‘Scaling out’ se preocupa em como os métodos ágeis podem ser introduzidos em uma grande organização com vários anos de experiência de desenvolvimento de software.

- A escalar métodos ágeis é essencial manter os fundamentos ágeis

  - ✓ Planejamento flexível, releases de sistema freguentes, integração contínua, desenvolvimento dirigido a testes e boa comunicação entre os membros da equipe.

Escalamento para sistemas de grande porte

- Para o desenvolvimento de sistemas de grande não é possível focar apenas no código do sistema. De início, é necessário fazer mais designs e documentação do sistema.

- Os mecanismos de comunicação entre as equipes precisam ser desenvolvidos e usados. O que deve envolver telefones comuns e vídeo-conferências e reuniões virtuais curtas e frequentes entre os membros da equipe, nas quais as equipes se informam mutuamente acerca do progresso do trabalho.

- A integração contínua, na qual o sistema todo é construído cada vez que qualquer desenvolvedor aplica uma mudança, é praticamente impossível. No entanto, é essencial manter builds frequentes e releases regulares do sistema.

Scaling out em grandes empresas

- Gerentes de projeto que não possuem experiência em métodos ágeis podem ser relutantes em aceitar o risco de uma nova abordagem.

- Geralmente as grandes organizações possuem procedimentos e padrões de qualidade que espera-se que sejam seguidos por todos os projetos e, devido a sua natureza burocratica, são incompatíveis com os métodos ágeis.

- Os métodos ágeis parecem funcionar melhor quando os membros da equipe possuem um nível de competência relativamente alto. No entanto, dentro de grandes organizações, geralmente ocorre uma grande variação de competências e habilidades.

- Pode haver resistência cultural aos métodos ágeis, especialmente nessas organizações com um longo histórico de

Pontos Importantes

- Um ponto particularmente forte da programação extrema é o desenvolvimento de testes automatizados antes de se criar um atributo do programa.

- Todos os testes devem ser executados com sucesso quando um incremento é integrado ao sistema.

- O método Scrum é um método ágil que provê um framework de gerenciamento de projeto. É baseado em um conjunto de Sprints, que são períodos fixos de tempo em que um incremento de sistema é desenvolvido.

- Escalamento de métodos ágeis para sistemas de grande porte é difícil. Tais sistemas precisam de mais projeto inicial e alguma documentação.