Métodos Ágeis

Motivação

- A insatisfação com os *overheads* envolvidos em métodos tradicionais de desenvolvimento levou à criação dos métodos ágeis. Esses métodos:
  - Focam no código ao invés de modelos ou documentos;
  - Baseiam-se em uma abordagem iterativa e incremental;
  - Visam entregar software funcionando rapidamente e evoluir esse software também rapidamente, a fim de satisfazer requisitos em constante mudança
- Métodos ágeis são mais apropriados para sistemas de negócios de tamanhos pequeno ou médio

O Manifesto Ágil

- Assinado por 17 desenvolvedores com reconhecimento mundial em Utah em fevereiro/2001.
- Declaração de 4 valores
- http://www.agilemanifesto.org/

O Manifesto Ágil

"Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a entender que:

- ***Indivíduos e interações*** são mais importantes que *processos e ferramentas*.
- ***Software funcionando*** é mais importante do que *documentação completa e detalhada*.
- ***Colaboração com o cliente*** é mais importante do que *negociação de contratos*.
- **Adaptação a mudanças** é mais importante do que *seguir o plano inicial*.

Ou seja, mesmo tendo valor os itens à direita, valorizamos mais os itens à esquerda. "

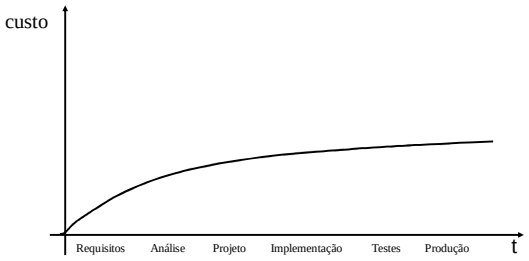
Mudanças sempre ocorrem

- Clientes não sabem o que querem no início
- Desenvolvedores não sabem qual a melhor maneira de fazer o software no início
- Medo da mudança trava o desenvolvimento

Avanços

- Nos últimos 30 anos...
  - Melhoria nos processos
    - Iterativo Incremental
  - Melhorias nas ferramentas
    - IDEs, automação...
  - Maior abstração no desenvolvimento
    - Orientação a Objetos
    - Orientação a Aspectos
    - Orientação a ...

Mudar não é tão caro



Princípios dos Métodos Ágeis

- Envolvimento do cliente
- Entrega incremental
- Pessoas, não processos
- Aceite as mudanças
- Mantenha a simplicidade

Benefícios dos Métodos Ágeis

- Clientes, quando ativamente envolvidos no desenvolvimento, experimentam uma "Síndrome de Estocolmo" benéfica
- Lidam bem com mudanças de requisitos
- Em geral, a equipe de desenvolvimento gosta de processos mais focados no código e menos em planos e modelos
- Produzem software funcional desde as primeiras iterações

Problemas com Métodos Ágeis

- Pode ser difícil manter os clientes tão ativamente envolvidos quanto exigido pelos métodos
- Membros da equipe podem não se prestar ao envolvimento intenso que caracteriza os métodos ágeis
- Manter a simplicidade requer trabalho extra
- Contratos podem ser um problema. Como cobrar por um escopo aberto?

Alguns Métodos Ágeis

- **eXtreme Programming**
- **Scrum**
- FDD
- Lean Software Development
- Cristal Family

XP - eXtreme Programming

O surgimento do XP

- Em meados de 1990, ***Kent Beck*** procurou formas mais simples e eficientes de desenvolver software
  - Identificou o que tornava simples e o que dificultava o desenvolvimento de software
- Em Março de 1996, ele iniciou um projeto com novos conceitos que resultaram na metodologia XP - *eXtreme Programming*

O que é eXtreme Programming?

"Trata-se de uma metodologia ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança"

Kent Beck, criador de XP

O que é eXtreme Programming?

- Um dos métodos ágeis mais populares e amplamente utilizados
- Adota uma abordagem "extrema" para o desenvolvimento iterativo e incremental:
  - Novas versões podem ser integradas várias vezes por dia;
  - Incrementos são entregues aos clientes mais ou menos a cada duas semanas;
  - Todos os testes devem ser executáveis e executados para cada versão integrada. Um *build* só é aceito se os testes passarem.

Programando ao Extremo

- Levar todas as boas práticas ao Extremo
  - Se testar é bom, vamos testar toda hora!!
  - Se projetar é bom, vamos fazer disso parte do trabalho diário de cada pessoa!
  - Se integrar é bom, vamos integrar a maior quantidade de vezes possível!
  - Se iterações curtas é bom, vamos deixar as iterações realmente curtas!

XP inclui...

- Comunicação
- Coragem
- Feedback
- Simplicidade
- Respeito

Comunicação

- Soluções de problemas normalmente já são conhecidas...
- O problema é a solução chegar a quem precisa
- Comunicação face a face é a maneira mais rápida de "espalhar" conhecimento

Coragem

- Extreme Programming é novo e diferente
- Atitude é mais importante que processo
- Coragem para dizer a verdade, para recomeçar do zero, para inovar

Feedback

- Feedback ←→ Comunicação
- Feedback aumenta aprendizado e produtividade

Simplicidade

- Regra geral
- Pensar sempre: " O que pode ser feito que seja o mais simples que funciona?"
- Simplicidade normalmente gera mais valor

Respeito

- Coragem, Feedback, Simplicidade, Comunicação...
- Respeito é necessário para tirar o máximo desses valores
- Respeito pelo próximo
  - Feedback, Comunicação
- Respeito por si mesmo
  - Coragem, Simplicidade

Práticas de XP

Prática	Descrição
Incremental planning	Requisitos são escritos em cartões de estórias e as estórias a serem incluídas em um *release* são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores quebram essas estórias em tarefas de desenvolvimento.
Small releases	O conjunto mínimo de funcionalidades úteis que fornece valor ao negócio é desenvolvido em primeiro lugar. Versões do sistema são frequentes e gradualmente adicionam funcionalidade ao primeiro release.
Simple design	O "design" suficiente para atender às exigências atuais é realizado e nada mais.
Test-first development	Um framework de teste de unidade automatizado é utilizado para escrever os testes para uma nova de funcionalidade antes que a funcionalidade seja implementada.
Refactoring	Todos os programadores devem refatorar o código continuamente, logo que melhorias possíveis para o código são descobertas. Isso mantém o código simples e fácil de manter.

Práticas de XP

Prática	Descrição
Pair programming	Os desenvolvedores trabalham em pares, verificando o trabalho dos outros e oferecendo o apoio para sempre fazer um bom trabalho.
Collective ownership	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de modo que não exista nenhuma "ilha de conhecimento" e todos os desenvolvedores são responsáveis por todo o código. Qualquer um pode alterar qualquer parte do código.
Continuous integration	Logo que o trabalho numa tarefa está completo, é integrado ao sistema como um todo. Depois de tal integração, todos os testes de unidade no sistema devem passar.
Sustainable pace	Grandes quantidades de horas extras não são consideradas aceitáveis, pois podem reduzir a qualidade do código e a produtividade da equipe a médio prazo.
On-site customer	Um representante do cliente do sistema deve estar disponível em tempo integral para a equipe de XP. O cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema para a equipe de desenvolvimento.

Requisitos em XP

- Requisitos são expressos por meio de estórias de usuários
- Estórias são escritas em cartões e quebradas em tarefas de implementação.
  - Essas tarefas são a base para determinar o cronograma e para estimativas de custo
  - Uma estória está mais para um "lembrete" do que para uma descrição detalhada dos requisitos
- O cliente escolhe as estórias que serão incluídas no próximo *release* com base em suas prioridades e nas estimativas do cronograma

CARTÃO DE USER STORY | Projeto: *Everest* | Iteração: *02* | Prioridade: *08*

Título: *Cadastro de usuário*

Descrição:

Acessando-se o menu principal, administradores do sistema têm privilégio de acesso ao módulo de cadastro de usuário.

O módulo consiste num formulário simples que permite o cadastramento de informações como: nome, sobrenome, endereço, e-mail, telefone, estado e cidade.

Após o cadastramento, é exibida uma página com a listagem de usuários.

O evento de cadastramento é registrado em log de atividades.

Analisado em *01 / 08 / 2005*
Por: *Luiz e Bruno*

Planejado em *05 / 08 / 2005*
Por: *Leandro*

Estimado em *02* story points

Iniciado em *07 / 08 / 2005*
Por: *Leandro*

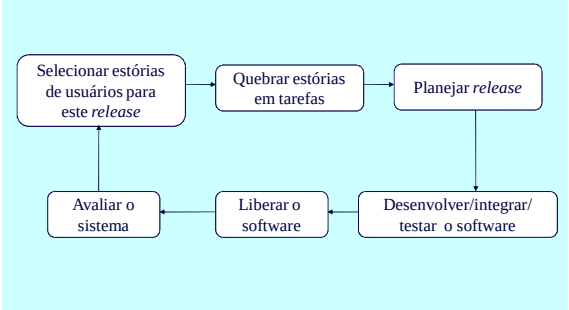
Terminado em *09 / 08 / 2005*

Realizado em *03* story points

Planejamento das iterações

- Divida o projeto em etapas de 1 ou 2 semanas;
- Considere prazos fixos e escolha um dia da semana para integrações e entregas;
- Planeje reuniões diárias para acompanhar a evolução do projeto ("stand-up", de preferência uma reunião matinal);
- As iterações serão as unidades de referência para fazer estimativas;
- Planeje a entrega de um "produto" a cada iteração.

O Ciclo de Vida de XP



Papéis no XP

Big Boss / XpManager

Deve fazer:

- Agendar reuniões;
- Escrever atas;
- Manter o XP Tracker informado dos acontecimentos das reuniões

Não deve fazer:

- Deixar que problemas externos interfiram no desenvolvimento
- Dizer quando as coisas devem acontecer
- Dizer às pessoas o que fazer
- Cobrar das pessoas

Papéis no XP

Xp Gold Owner (Cliente - O proprietário do ouro)

É quem paga pelo sistema, geralmente o dono da empresa.

Xp Goal Donor

Deve fazer:

- Escrever User Stories
- Definir prioridades
- Definir testes de aceitação
- Validar testes de aceitação
- Esclarecer dúvidas

Não deve fazer:

- Implementar código
- Definir quanto tempo uma tarefa leva para ser feita

Papéis no XP

Coordenadores

Xp Coach

Responsável pela negociação com o cliente quanto ao escopo e pela coordenação do *Planning Game*.

métricas

Xp Tracker

Deve fazer:

- Coletar métricas
- Manter todos informados do que está acontecendo
- Definir testes de aceitação
- Tomar atitudes diante de problemas
- Sugerir sessões de CRC (Class, Responsabilities, Collaboration)

Papéis no XP

Programador
(Driver/Partner)

Deve fazer:

- Estimar prazos para User Stories
- Implementar código de produção
- Trabalhar em par
- Fazer refactoring
- Corrigir bugs

Não deve fazer:

- Criar/Alterar novas funcionalidades
- Escrever testes de aceitação

XP e Mudanças

- A "sabedoria convencional" da engenharia de software prega que se deve projetar para mudanças
  - Supõe que antecipar mudanças antes que ocorram reduz custos em estágios posteriores do desenvolvimento
- XP, porém, assume que esse esforço não vale a pena, já que é muito difícil antecipar as mudanças
- Ao invés disso, o método propõe o uso de refatoração para facilitar a incorporação de mudanças, quando elas forem necessárias.

Testes em XP

- Desenvolvimento de "testes antes"
- Desenvolvimento incremental de testes a partir das estórias de usuários
- O usuário está envolvido no desenvolvimento de testes de aceitação
- Conjuntos de testes automáticos são executados para todo o sistema cada vez que um novo *release* é produzido

Benefícios dos "Testes Antes"

- Escrever os testes antes clarifica os requisitos a serem implementados
  - Funciona, ao mesmo tempo, como uma especificação da funcionalidade e um projeto detalhado
- Os testes são programas ao invés de especificações e podem ser executados automaticamente.
- Cada teste construído funciona como teste de regressão nas iterações seguintes.
  - Se uma modificação quebra o código existente, esse problema é detectado imediatamente

Programação em Pares

- Em XP, programadores trabalham em pares, sentando-se juntos para desenvolver código
- Isso auxilia na propriedade coletiva do código e espalha conhecimento por todo o time
- Funciona como um processo informal de revisão
  - Mais de uma pessoa olha para cada linha de código
- Há estudos que sugerem que a produtividade da programação em pares é similar à de duas pessoas trabalhando independentemente

XP e Princípios Ágeis

- Desenvolvimento incremental é apoiado por releases frequentes e pequenos
- Envolvimento do cliente é completo, já que ele é parte da equipe de desenvolvimento
- Programação em pares, propriedade coletiva do código e ritmo sustentável apóiam o princípio de Pessoas e não Processos
- Mudanças são aceitas através de diversas práticas (releases curtos, refatoração, desenvolvimento de "testes antes", etc.)
- Simplicidade é mantida através de projeto simples e refatoração

SCRUM

O que é o SCRUM?

- Um processo de desenvolvimento iterativo e incremental que pode ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa.
- Ken Schwaber e Mike Beedle desenvolveram a metodologia na década de 90 baseando-se em sua própria experiência no desenvolvimento de sistemas e processos

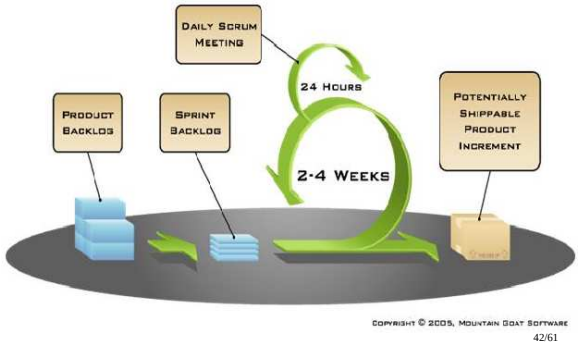
SCRUM

- O Scrum não é um processo previsível, ele não define o que fazer em toda circunstância. KEN SCHWABER (2004).
  - É um framework e um conjunto de práticas
  - Bastante objetivo
  - Papéis e Responsabilidade bem definidas
  - Fácil adaptação
  - Curva de aprendizado baixa

Características

- Equipes auto-organizadas
- Produto progride em uma série de "sprints" de duas semanas a um mês
- Requisitos são capturados como itens em uma lista de um "product backlog"

O ciclo de vida do Scrum



Artefatos do Scrum: *Product Backlog*

- Lista priorizada dos requisitos do projeto com o tempo estimado para torná-los em funcionalidade
- Estimativas em dias, sendo mais precisas para itens mais prioritários
- Mudanças podem ser feitas de acordo com as necessidades que aparecem
- Nunca está completo, sendo que o utilizado no plano de projeto é meramente uma estimativa inicial dos requisitos
- O responsável é o Product Owner

Artefatos do Scrum: *Product Backlog*

- Qualquer membro do time pode adicionar ou remover itens
  - Com consentimento do Product Owner
- São requisitos funcionais ou não-funcionais, ou qualquer tópico de discussão
- Os itens com maior prioridade serão selecionados para o próximo Sprint
- Quanto menos prioritários, mais abstratos são os itens
- Podem existir muito mais requisitos que não foram ainda listados ou nem pensados
- É atualizado pelo Product Owner sempre que necessário
  - Permite a adaptabilidade do processo

Artefatos do Scrum: *Sprint Backlog*

- Lista de tarefas que define o trabalho do time durante o Sprint
- Cada tarefa identifica o responsável que irá trabalhar sobre ela e o restante do tempo estimado para terminá-la em dias
- Tarefas devem estar organizadas para que estejam em 4 a 16 horas de trabalho
- Apenas o time pode modificá-lo

Papéis no Scrum: *Product Owner*

- Responsável por representar os interesses de todos os *stakeholders*
- Tem o poder de tomar (para o time) decisões pelo Cliente e Usuários
- Define fundamentos iniciais do projeto, objetivos com relação ao ROI e planos de release
- A lista de requisitos é o Product Backlog
- Certifica se as atividades com maior valor para o negócio são criadas primeiro
  - Priorização freqüente das funcionalidades antes de cada iteração

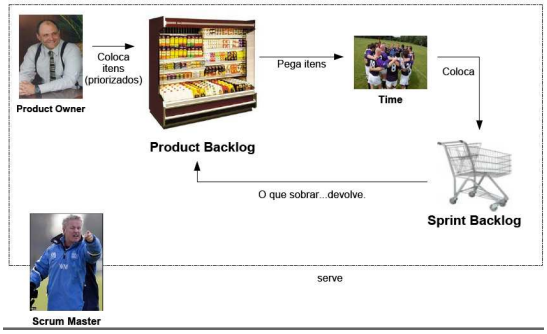
Papéis no Scrum: *ScrumMaster*

- Responsável pelo sucesso do Scrum
- Ensina o Scrum para os envolvidos com o projeto
- Implementa o Scrum na empresa de forma adaptada a sua cultura, para continuamente gerar benefícios
- Certifica se cada pessoa envolvida está seguindo seus papéis e as regras do Scrum
- Certifica que pessoas não responsáveis não interfiram no processo

Papéis no Scrum: Time

- Responsável por escolher as funcionalidades a serem desenvolvidas e cada interação e desenvolvê-las
- O time se auto-gerencia, se auto-organiza
- Todos os membros do time são coletivamente responsáveis pelo sucesso de cada iteração

Papéis x Ciclo de Vida



Regras do Scrum

- O ScrumMaster deve se certificar de que cada envolvido no projeto siga suas regras
- As regras permitem a execução correta do Scrum
- Mudanças das regras devem originar do time
  - Scrum Master deve ser convencido de que todos envolvidos entenderam suficientemente as regras do Scrum para o correto discernimento
  - Discussões desnecessárias são perdas de tempo de produção da equipe

Regras do Scrum: Sprint Planning Meeting

- A reunião de planejamento do Sprint deve ocorrer dentro de 8 horas com duas partes de 4 horas
- Primeiro seguimento:
  - Product Owner deve preparar o Product Backlog antes da reunião
  - Seleção dos itens do Product Backlog que o time se compromete em torná-los incrementos potencialmente implementáveis

Regras do Scrum: Sprint Planning Meeting

- Segundo seguimento:
  - Ocorre imediatamente após o primeiro
  - Product Owner deve estar disponível para o que o time faça perguntas sobre o Product Backlog
  - O time deve decidir sozinho como os itens selecionados serão implementados

Regras do Scrum: Reunião Diária do Scrum

- Reunião de no máximo 15 minutos, a menos que o time seja grande o suficiente para precisar de mais tempo
- Deve ser feita no mesmo lugar onde o time trabalha
- Resulta em melhores resultados se realizada no inicio do dia de trabalho
- Todos os membros do time devem participar desta reunião

Regras do Scrum: Reunião Diária do Scrum

- Perguntas fundamentais
  - 1. O que fiz desde a última reunião?
  - 2. O que farei até a próxima reunião?
  - 3. Existe algum obstáculo?

Regras do Scrum: Sprint

- Não deve ser maior do que 30 dias consecutivos
- Sem considerar outros fatores, este é o tempo necessário para produzir algo de interesse para o Product Owner e os stakeholders
- O time pode pesquisar e requisitar ajuda externa

Regras do Scrum: Sprint

- Responsabilidades do time durante o Sprint:
  - Participar das reuniões diárias do Scrum
  - Manter o Sprint Backlog atualizado
  - Disponibilizar o Sprint Backlog publicamente
- O time tem o compromisso de implementar todos os itens selecionados

Regras do Scrum: Reunião de Revisão

- Reunião ao final do Sprint de no máximo 4 horas sob responsabilidade do ScrumMaster
- O time não deve gastar mais de 1 hora na preparação desta reunião
- Objetivo: mostrar o Product Owner e stakeholders as funcionalidades que foram feitas
- Artefatos não devem ser apresentados, pois não são funcionalidades

Regras do Scrum: Reunião de Revisão

- No final da reunião
  - Cada stakeholder fala suas impressões e sugere mudanças com suas respectivas prioridades
  - Possíveis modificações no Product Backlog são discutidas entre o Product Owner e o time
  - Scrum Master anuncia a data e o local da próxima reunião de revisão do Sprint ao Product Owner e a todos stakeholders

Regras do Scrum: Reunião Retrospectiva do Sprint

- Ocorre logo após a reunião de revisão
- Não deve ser maior do que 3 horas.
- Participam desta reunião: time, o ScrumMaster e, opcionalmente, o Product Owner
- Os membros do time devem responder a duas questões:
  - O que aconteceu de bom durante o último Sprint?
  - O que pode ser melhorado para o próximo Sprint?

Regras do Scrum: Reunião Retrospectiva do Sprint

- ScrumMaster escreve as respostas e prioriza na ordem que deseja discutir as potenciais melhorias
- ScrumMaster nesta reunião tem o papel de facilitar que o time encontre melhores formas de aplicar o Scrum

Considerações Finais

- Métodos Ágeis são métodos de desenvolvimento iterativos e incrementais
  - Visam reduzir o *overhead* de desenvolvimento, produzindo software de qualidade mais rapidamente
- Mudança é algo normal
  - Aceitar, não fugir