USP – UNIVERSIDADE DO ESTADO DE SÃO PAULO

Métodos Ágeis

•Alunos: Rogério Guaraci dos Santos - rgsantos@ime.usp.br
Giulian Dalton Luz - gdaltonl@ime.usp.br

Manifesto Ágil - Princípios

- ***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*.

WebSite: http://www.agilemanifesto.org/

- Métodos ágeis (AM) é uma coleção de metodologias baseada na prática para modelagem efetiva de sistemas baseados em software. É uma filosofia onde muitas metodologias se encaixam.
- As metodologias ágeis aplicam uma coleção de práticas, guiadas por princípios e valores que podem ser aplicados por profissionais de software no dia a dia.

O que são os Modelos Ágeis?

- Um modelo ágil é um modelo bom o suficiente, nada mais, o que implica que ele exibe as seguintes características:
  1. **Ele** atende seu propósito
  2. **Ele** é inteligível.
  3. **Ele** é suficientemente preciso.
  4. **Ele** é suficientemente consistente.
  5. **Ele** é suficientemente detalhado.
  6. **Ele** provê um valor positivo.
  7. **Ele** é tão simples quanto possível.

O que é (e não é) métodos ágeis?

1. É uma atitude, não um processo prescritivo.
2. É um suplemento aos métodos existentes, ele não é uma metodologia completa.
3. É uma forma efetiva de se trabalhar em conjunto para atingir as necessidades das partes interessadas no projeto.
4. É uma coisa que funciona na prática, não é teoria acadêmica.

O que é (e não é) métodos ágeis? (cont.)

5. É para o desenvolvedor médio, mas não é um substituto de pessoas competentes.
6. Não é um ataque à documentação, pelo contrário aconselha a criação de documentos que tem valor.
7. Não é um ataque às ferramentas CASE.

SCRUM

Processo de Desenvolvimento de Software

- **Scrum** é um processo para construir software incrementalmente em ambientes complexos, onde os requisitos não são claros ou mudam com muita freqüência.

- Em Rugby, **Scrum** é um time de oito integrantes que trabalham em conjunto para levar a bola adiante no campo. Ou seja: times trabalhando como uma unidade altamente integrada com cada membro desempenhando um papel bem definido e o time inteiro focando num único objetivo.

- O objetivo do **Scrum** é fornecer um processo conveniente para projetos e desenvolvimento orientado a objetos.
- A metodologia é baseada em princípios semelhantes aos de XP: equipes pequenas, requisitos pouco estáveis ou desconhecidos, e iterações curtas para promover visibilidade para o desenvolvimento.

- No entanto, as dimensões em **Scrum** diferem de XP.
- **Scrum** divide o desenvolvimento em Sprints de 30 dias. Equipes pequenas, de até 7 pessoas, são formadas por projetistas, programadores, engenheiros e gerentes de qualidade. Estas equipes trabalham em cima de funcionalidade (os requisitos, em outras palavras) definidas no início de cada Sprint. A equipe toda é responsável pelo desenvolvimento desta funcionalidade

- Todo dia, é feita uma reunião de 15 minutos onde o time expões à gerência o que será feito no próximo dia, e nestas reuniões os gerentes podem levantar os fatores de impedimento, e o progresso geral do desenvolvimento.

Fases – Sprint Reuniões Diárias

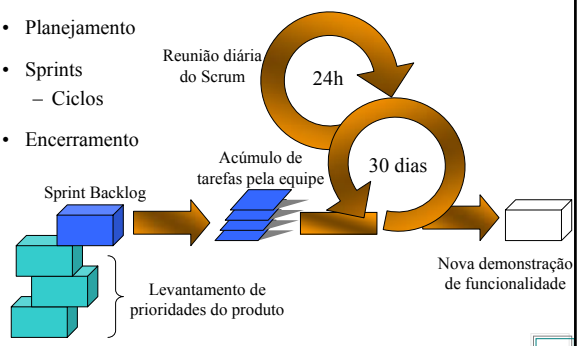
- Todos respondem às perguntas:
  - O que você realizou desde a última reunião?
  - Quais problemas você enfrentou?
  - Em que você trabalhará até a próxima reunião?
- Benefícios:
  - Maior integração entre os membros da equipe
  - Rápida solução de problemas
    - Promovem o compartilhamento de conhecimento
  - Progresso medido continuamente
    - Minimização de riscos

•**Scrum** é interessante porque fornece um mecanismo de informação de status que é atualizado continuamente, e porque utiliza a divisão de tarefas dentro da equipe de forma explicita.

Scrum e XP são complementares pois Scrum provê práticas ágeis de gerenciamento enquanto XP provê práticas integradas de engenharia de software.

Fases

- Planejamento
- Sprints
  - Ciclos
- Encerramento



Fases de encerramento

- Iniciada quando todos os aspectos são satisfatórios (tempo, competitividade, requisitos, qualidade, custo)
- Atividades:
  - Testes de integração
  - Testes de sistema
  - Documentação do usuário
  - Preparação de material de treinamento
  - Preparação de material de marketing

Qualidade, Gerenciamento e Testes

- **Passos e papéis bem definidos**
- **Gerenciamento de riscos**
- **Revisões freqüentes / diárias**
- **Definição de padrões**
- **Realização de testes**
- **Elaboração de documentação**

Controles

Backlog
***Release*/Melhoria**
Mudanças
Problemas
Soluções

Crystal

Processo de Desenvolvimento de Software

- **Crystal/Clear** faz parte, na realidade, de um conjunto de metodologias criado por Cockburn. As premissas apresentadas para a existência deste conjunto são:

- Todo projeto tem necessidades, convenções e uma metodologia diferentes.
- O funcionamento do projeto é influenciado por fatores humanos, e há melhora neste quando os indivíduos produzem melhor.
- Comunicação melhor e lançamentos freqüentes reduzem a necessidade de construir produtos intermediários do processo

- **Crystal/Clear** é uma metodologia direcionada a projetos pequenos, com equipes de até 6 desenvolvedores. Assim como com SCRUM, os membros da equipe tem especialidades distintas. Existe uma forte ênfase na comunicação entre os membros do grupo.

  Existem outras metodologias Crystal para grupos maiores.

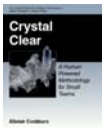
- Toda a especificação e projeto são feitos informalmente, utilizando quadros publicamente visíveis. Os requisitos são elaborados utilizando casos de uso, um conceito similar às estórias de usuário em XP, onde são enunciados os requisitos como tarefas e um processo para sua execução.

- As entregas das novas versões de software são feitos em incrementos regulares de um mês, e existem alguns subprodutos do processo que são responsabilidade de membros específicos do projeto.

- Grande parte da metodologia é pouco definida, e segundo o autor, isto é proposital; a idéia de **Crystal/Clear** é permitir que cada organização implemente as atividades que lhe parecem adequadas, fornecendo um mínimo de suporte útil do ponto de vista de comunicação e documentos

A família *Crystal* de Métodos

- Criada por Alistair Cockburn
- http://alistair.cockburn.us/crystal
- Editor da série *Agile Software Development* da Addison-Wesley.



Feature Driven Development
Desenvolvimento orientado a funcionalidades

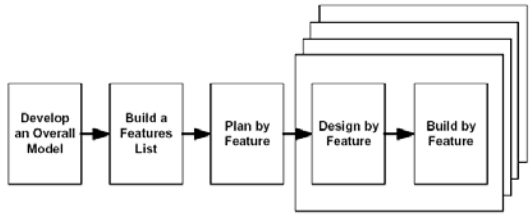
Stephen Palmer & John Felsing **2002**

FDD - Características

- Método ágil e adaptativo;
- Foco nas fases de desenho e construção
- Interage com outras metodologias
- Não exige nenhum processo específico de modelagem
- Possui desenvolvimento iterativo
- Enfatiza aspectos de qualidade durante o processo e inclui entregas freqüentes e tangíveis
- Suporta desenvolvimento ágil com rápidas adaptações às mudanças de requisitos e necessidades do mercado

FDD - Processos

- O FDD consiste de 5 processos principais:



FDD – Processos (Cont.)

- **Desenvolver um modelo compreensível (Develop an overall model)**
- **Construir uma lista de funcionalidades (Build a features list)**
- **Planejar por funcionalidade (Plan By Feature)**
- **Projetar por funcionalidade (Design by feature)**
- **Construir por funcionalidade (Build by feature)**

FDD - Cargos e Responsabilidades

- **Principais**
  - ✓ Gerente de projeto (Project Manager)
  - ✓ Arquiteto líder (Chief architect)
  - ✓ Gerente de desenvolvimento (Development Manager)
  - ✓ Programador líder (Chief programmer)
  - ✓ Proprietário de classe (Class owner)
  - ✓ Especialista do domínio (Domain experts)
  - ✓ Gerente do domínio (Domain manager)

FDD - Cargos e Responsabilidades (Cont.)

- **De apoio**
  - ✓ Gerente de versão (Release manager)
  - ✓ Guru de linguagem (Language lawyer/language guru)
  - ✓ Engenheiro de construção (Build engineer)
  - ✓ "Ferramenteiro" (Toolsmith)
  - ✓ Administrador de sistemas (System Administrator)
- **Adicionais**
  - ✓ Testadores (Testers)
  - ✓ Instaladores (Deployers)
  - ✓ Escritores técnicos (Technical writes)

FDD - Boas Práticas

- **Modelagem de objetos de domínio (**Domain Object Modeling)
  - ✓ Exploração e explicação do problema do domínio
  - ✓ Resulta em um arcabouço
- **Desenvolver por funcionalidade** (Developing by feature)
  - ✓ Desenvolvimento e acompanhamento do progresso através de da lista de funcionalidades.
- **Proprietários de classes individuais** (Individual class ownership)
  - ✓ Cada classe possui um único desenvolvedor responsável

FDD - Boas Práticas (Cont.)

- **Equipe de funcionalidades (Feature teams)**
  - ✓ Formação de equipes pequenas e dinâmicas.
  - ✓ Inspeção (Inspection)
  - ✓ Uso dos melhores métodos conhecidos de detecção de erros.
- **Construções freqüentes (Regular Builds)**
  - ✓ Garantir que existe um sistema sempre disponível e demonstrável.
- **Administração de Configuração (Configuration Manager)**
  - ✓ Habilita acompanhamento do histórico do código-fonte..

Dynamic Systems Development Method (DSDM)

Método dinâmico de desenvolvimento de sistemas

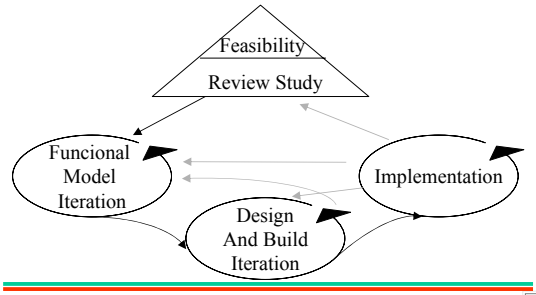
DSDM Consortium - 1994
Jennifer Stapleton - 1997

DSDM - Características

- Progenitor do XP
- Arcabouço para desenvolvimento rápido de aplicações (RAD)
- Fixa tempo e recursos ajustando a quantia de funcionalidades
- Pequenas equipes
- Suporta mudanças nos requisitos durante o ciclo de vida

DSDM - Fases

- O DSDM consiste de 5 fases:



DSDM – Fases (Cont.)

- **Estudo das possibilidades** (Feasibility study)
- **Estudo dos negócios** (Business study)
- **Iteração do modelo funcional** (Functional model iteration)
- **Iteração de projeto e construção** (Design and build iteration)
- **Implementação final** (Final implementation)

DSDM - Cargos e Responsabilidades

- **Desenvolvedores** (Developers)
- **Desenvolvedores Sêniores** (Senior Developers)
- **Coordenador Técnico** (Technical Coordinator
- **Usuário Embaixador** (Ambassador User)
- **Usuário Consultor** (Adviser User)
- **Visionário** (Visionary)
- **Executivo responsável** (Executive Sponsor)
- **Especialista do domínio** (Domain experts)
- **Gerente do domínio** (Domain manager)

DSDM - Práticas

- Usuário sempre envolvido
- Equipe do DSDM autorizada a tomar decisões
- Foco na frequente entrega de produtos
- Adaptação ao negócio é o critério para entregas
  "Construa o produto certo antes de você construí-lo corretamente"
- Desenvolvimento iterativo e incremental
- Mudanças são reversíveis utilizando pequenas iterações
- Requisitos são acompanhados em alto nível
- Testes integrados ao ciclo de vida

Adaptive Software Development
Desenvolvimento Adaptável de Software

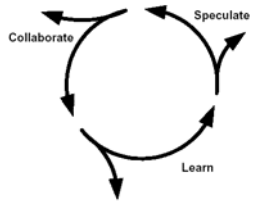
James A. Highsmith III – 2000

ASD - Características

- Iterativo e incremental
- Sistemas grandes e complexos
- Arcabouço para evitar o caos
- Cliente sempre presente
- Desenvolvimento de aplicações em conjunto (Joint Application development – JAD)

ASD - Fases

- O ASD possui ciclos de 3 fases:



ASD – Fases (Cont.)

- **Especular (Speculate)**
  - ✓ Fixa prazos e objetivos
  - ✓ Define um plano baseado em componentes
- **Colaborar (Collaborate)**
  - ✓ Construção concorrente de vários componentes
- **Aprender (Learn)**
  - ✓ Repetitivas revisões de qualidade e foco na demostrânção das funcionalidades desenvolvidas (Learning loop)
  - ✓ Presença do cliente e especialistas do domínio
- ❑ Os ciclos duram de 4 a 8 semanas

ASD - Propriedades

- **Orientado a missões (Misson-driven)**
  - ✓ Atividades são justificadas através de uma missão, que pode mudar ao longo do projeto
- **Baseado em componentes (Component-based)**
  - ✓ Construir o sistema em pequenos pedaços
- **Iterativo (Iterative)**
  - ✓ Desenvolvimento em cascata (Waterfall) só funciona em ambientes bem definidos e compreendidos. O ASD possui foco em refazer do que fazer corretamente já na primeira vez

ASD – Propriedades (Cont.)

- **Prazos pré-fixados (Time-boxed)**
  - ✓ Força os participantes do projeto a definir severamente decisões do projeto logo cedo.
- **Tolerância a mudanças (Change-tolerant)**
  - ✓ As mudanças são freqüentes
  - ✓ É sempre melhor estar pronto a adaptá-las do que controlá-las
  - ✓ Constante avaliação de quais componentes podem mudar
- **Orientado a riscos (Risk driver)**
  - ✓ Itens de alto risco são desenvolvidos primeiro

ASD - Cargos e Responsabilidades

- ❑ Este método não descreve cargos em detalhes
  - **Executivo responsável (Executive Sponsor)**
- ❑ Participantes de uma sessão (workshop) do desenvolvimento de aplicações em conjunto (JAD)
  - **Facilitador (Facilitator)**
    - ✓ Liderar e planejar as sessões
  - **Escriba (Scribe)**
    - Efetuar anotações
  - **Cliente (Customer)**
    - ✓ Sempre presente
  - **Gerente de Projetos (Project Manager)**
  - **Desenvolvedores (Developers)**

Outras Metodologias Ágeis

- **Extreme Programming**
- **Rational Unified Process (RUP)**
  - ✓ Considerado uma metodologia ágil por alguns autores
- **Open Source Software Development**
- **Agile Modeling**
- **Pragmatic Programming**

Links Relacionados

- http://www.agilemanifesto.org/
- http://www.dcc.unicamp.br/~ra022247/Arquivos/scrum.pdf
- http://www.poli.usp.br/pro/procsoft/tproepusp04.pdf
- http://alistair.cockburn.us/crystal
- http://www.featuredrivendevelopment.com/
- http://www.dsdm.org/
- http://www.adaptivesd.com/
- http://www.rspa.com/spi/process-agile.html
- http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf
- **www.ime.usp.br/~gdaltonl/ageis.htm**