Modulo I
Metodologias Ágeis
Panorama

Prof. Ismael H F Santos

Bibliografia

- *Vinicius Manhaes Teles, Extreme Programming, Novatec Editora*
- *Agile Software Development*
- *Scrum and XP from the Trenches*
- *Martin Fowler, Analysis Patterns - Reusable Object Models, Addison-Wesley,1997*
- *Martin Fowler, Refatoração - Aperfeiçoando o projeto de código existente, Ed Bookman*

Ementa

- Introdução
- Processo Unificado
- Manifesto Ágil
  - XP
  - DSDM
  - SCRUM
  - FDD
  - Lean Software
- CONCLUSÃO

MA-Overview

O Desafio do Desenvolvimento de Software

- Ainda vivemos em crise?
  - **Crise do Software** = Conjunto de problemas enfrentados ao longo do desenvolvimento.
  - Problemas na Definição, Construção, Implantação, Manutenção.
- Foco no objetivo principal do desenvolvimento:
  - Desenvolver o produto que atenda as necessidades do cliente e seja entregue no prazo, com o custo e o nível de qualidade desejado.

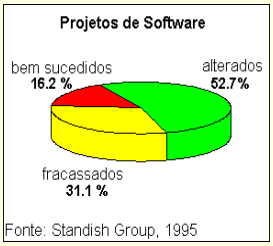
Caos no Desenvovlimento SW

Ref: The Chaos Report (Standish Group, 1994)

- Objetivo:
  - ???
- Meio:
  - Não é feito projeto
  - Decisões de curtíssimo prazo.
- Problemas:
  - Dificuldade de corrigir defeitos quando o sistema cresce.
  - Longa fase de debug/teste depois do sistema estar "completo" (debug/teste é impossível de orçar)

O Desafio do Desenvolvimento de Software

- Olhando o cenário a nossa volta
  - The CHAOS Report



The CHAOS Report, 1995, Standish Group

Existe uma bala de prata?

"There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity."

Frederick Brooks, 1986

- O que dizer da Orientação a Objetos, da UML, etc?
  - Atacam tarefas acidentais
  - O problema é como tratar as tarefas essenciais

Melhorando o Software pela Melhoria do Processo

- Não existe uma solução mágica e única, mas sim um conjunto de práticas reconhecidamente eficientes.
  - Desenvolvimento Incremental, Refinamento de Requisitos e Prototipação Rápida, **BONS PROJETISTAS**...
- Melhorar a qualidade do software implica na melhoria do processo pelo qual o mesmo é produzido.
  - Assumir práticas de sucesso
  - Garantir que estas práticas serão seguidas durante o desenvolvimento
  - Ser fácil de seguir
  - Evoluir com o aprendizado do grupo
- Na indústria atual, dois extremos foram definidos:
  - Processos Monumentais X *Hacking*

Rigor - Ref: Pressman (1980), CMM (1987)

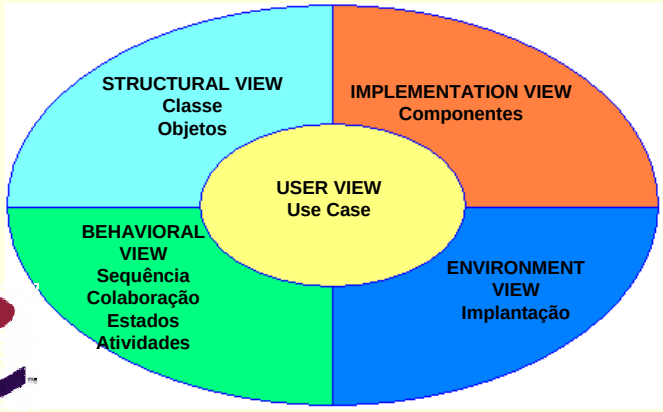
- Objetivo:
  - Previsibilidade,
  - Comando e Controle
- Abordagem:
  - Planejamento detalhado (“Engenharia” de software),
  - Fases seqüenciais de processo (cascata, “cascatinha”)
  - Artefatos de uma fase para a seguinte (“Fábrica de Software”)
- Problemas:
  - burocracia → mais tarefas para um resultado
  - não adaptabilidade → realidade (prazo, escopo, processo, pessoas) difere do planejado/documentado

Orientação a Objetos

- A Decepção da UML
  - Análise essencial dizia o QUE fazer, COMO fazer e QUANDO
  - Quando surge a UML, o mercado queria um substituto para a Análise Essencial
  - UML é uma linguagem e não um processo. Ela fornece os elementos, mas não define QUANDO usar
  - O mercado rejeitou a UML por não compreendê-la
  - RUP, XP são processos que se utilizam da UML

Orientação a Objetos

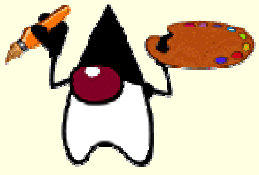
UML



Engenharia de Software

- Pressman (1995) destaca que, ainda que várias definições tenham sido dadas à ES, todas reforçam a exigência da disciplina de engenharia no desenvolvimento de software. Abrange um conjunto de três elementos fundamentais:
  - métodos, ferramentas e procedimentos.
- Desenvolvimento de Ciência ou Arte?!?

Software é



Engenharia de Software

- Os métodos detalham "como fazer" para se construir o software.
- As ferramentas proporcionam apoio automatizado ou semi-automatizado aos métodos.
- Os procedimentos constituem o elo de ligação que mantém juntos os métodos e suas ferramentas, e possibilita um processo de desenvolvimento claro, eficiente, visando garantir ao desenvolvedor e seus clientes a produção de um software de qualidade.

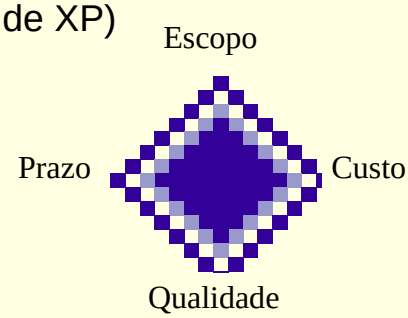
Engenharia de Software

- Qual é a nossa missão?
  - Desenvolver Software:
    - *Atendendo a todas as necessidades de todos os envolvidos*
    - *Com o nível de qualidade esperado por nossos clientes*
    - *Dentro do Prazo*
    - *Dentro do Orçamento*

As 4 Variáveis do Desenvolvimento de Software

- Tempo
- Custo
- Qualidade
- Escopo (foco principal de XP)

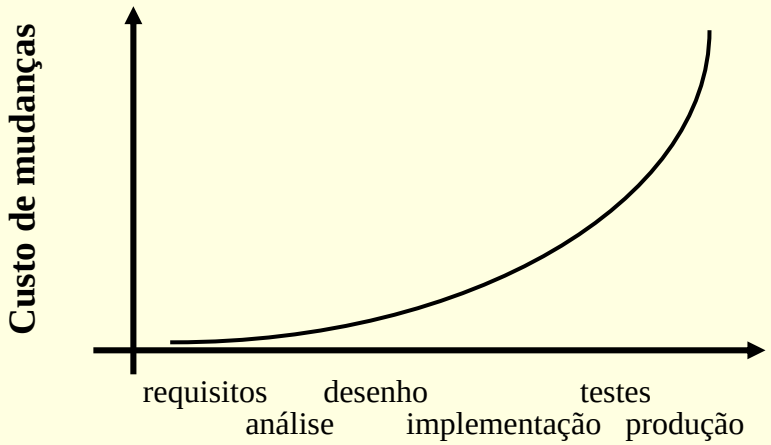
- Diamante Mágico



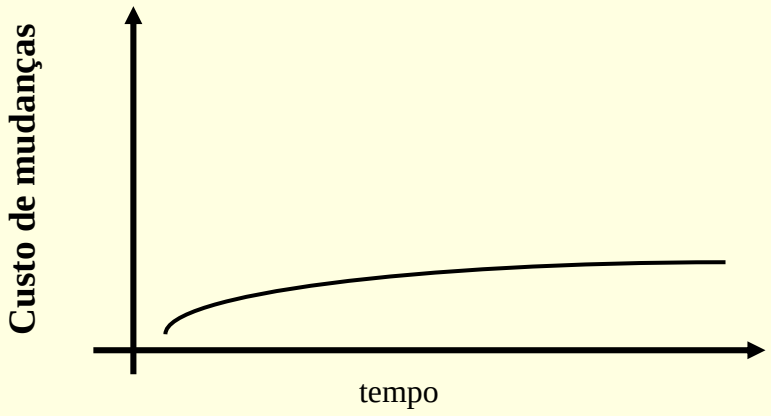
Premissas Básicas do Modelo Tradicional

- É necessário fazer uma análise de requisitos profunda e detalhada antes de projetar a arquitetura do sistema.
- É necessário fazer um estudo minucioso e elaborar uma descrição detalhada da arquitetura antes de começar a implementá-la.
- É necessário testar o sistema completamente antes de mandar a versão final para o cliente.

O que está por trás deste modelo?



E se a realidade hoje em dia fosse outra?



E se essa fosse a realidade?

- A atitude dos desenvolvedores de software seria completamente diferente:
  - Tomaríamos as grandes decisões o mais tarde possível.
  - Implementaríamos agora somente o que precisamos *agora.*
  - Não implementaríamos flexibilidade desnecessária (não anteciparíamos necessidades).

E essa é a nova realidade !
(pelo menos em muitos casos)

- Orientação a Objetos: facilita e cria oportunidades para mudanças.
- Técnicas de Refatoramento.
- Testes automatizados: nos dão segurança quando fazemos mudanças.
- Prática / cultura de mudanças: aprendemos técnicas e adquirimos experiência em lidar com código mutante.

Projeto X Construção

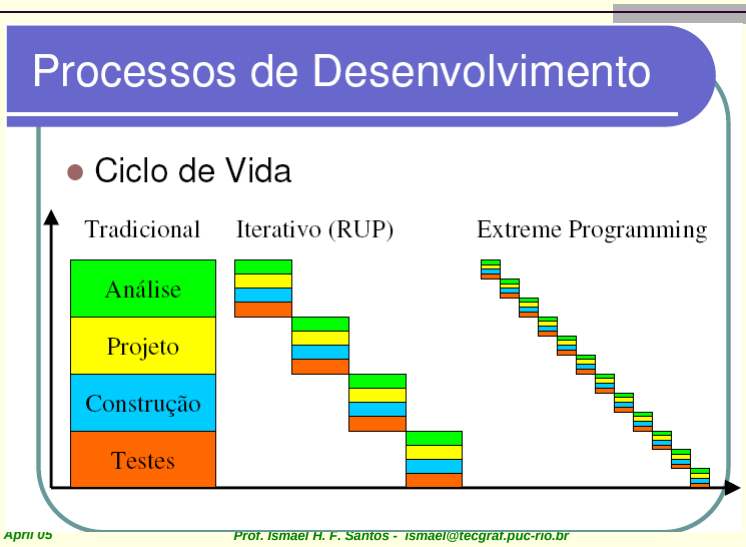
- Engenharia civil:
  - Projeto (10 % do esforço): difícil de estimar
  - Construção (90 %): planejamento detalhado
- Desenvolvimento de software
  - Projeto (85 %)
  - Codificação (15 %)
- Questões
  - Decisões de design são feitas na codificação.
  - "Construção" em software é automatizável ?
  - *Engenharia* de Software ?

Processo de Desenvolvimento

Processos de Desenvolvimento

- Os processos de desenvolvimento são compostos por diversas fases;
- Em cada fase é necessário executar diversas atividades.
- Esse esforço tem como alvo principal a construção de um sistema de qualidade.

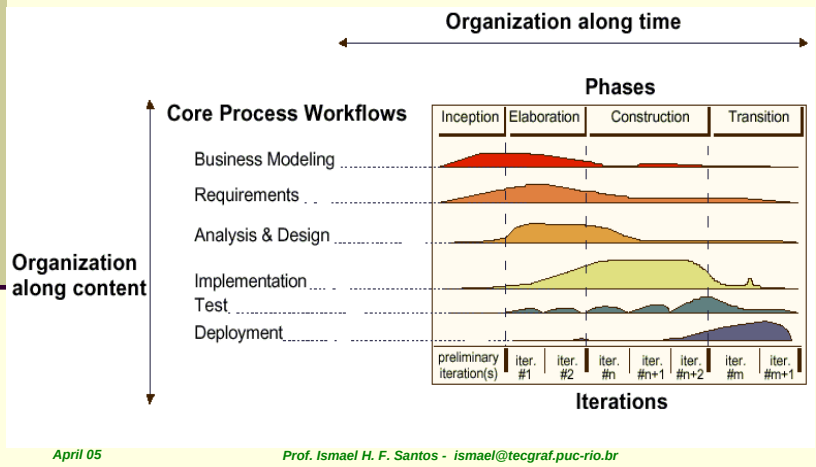
Processo de Desenvolvimento



MA-Overview

Engenharia de Software

- **RUP - Rational Unified Process**



O Processo Unificado da Rational

- Características:
  - É um processo de Engenharia Software
  - É um framework de processo
  - É um produto
  - **Compatibilidade total com a UML**

- Captura práticas consagradas no desenv de software:
  - Desenvolver software iterativamente
  - Gerenciar Requisitos
  - Usar arquiteturas baseadas em componentes
  - Modelar o software visualmente
  - Verificar a qualidade do software continuamente
  - Controlar mudanças no software

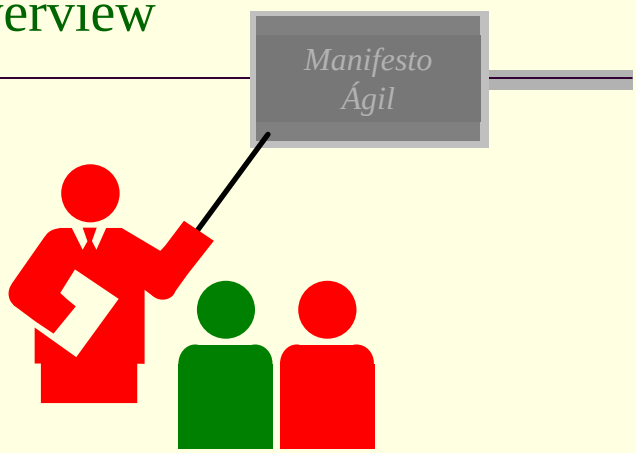
O Processo Unificado da Rational

- **Concepção**
  - Definição do Caso de Negócio do Projeto
  - Definição do Escopo
  - Verificação da Viabilidade do Projeto
- **Elaboração**
  - Análise do Domínio do Problema
  - Estabelecimento da Arquitetura do Sistema
- **Construção**
  - Desenvolvimento Iterativo e Incremental
  - Foco na Implementação e nos Testes
- **Transição**
  - Entrega do Software para os Usuários
  - Ajustes do Produto

O Processo Unificado da Rational

- Algumas questões
  - Como definir uma instância ideal do RUP para minha empresa?
  - E em pequenas e médias empresas?
  - Que pontos podem ser considerados a essência do RUP?
- Solução
  - Utilizar os valores e princípios dos Processos Ágeis como maneira para definir uma instância ideal do RUP.

MA-Overview



O Manifesto do Desenvolvimento Ágil

From www.agilealliance.org: We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions **over processes and tools**
- Working software **over comprehensive documentation**
- Customer collaboration **over contract negotiation**
- Responding to change **over following a plan**

That is, while there is value in the items on the right, we value the items on the left more.

O Manifesto Ágil (2001)

- "Descobrimos melhores maneiras de desenvolver software fazendo-o e ajudando os outros a fazê-lo. Através deste trabalho passamos a valorizar"
  - **Indivíduos e iteração** mais que processos e ferramentas
  - **Software que funciona** mais que documentação detalhada.
  - **Colaboração do cliente** mais que negociações contratuais.
  - **Responder às mudanças** mais que seguir um plano.
- "Isto é, enquanto há um certo valor nos ítens do lado direito, valorizamos *mais* os do lado esquerdo"
- Ref: http:// www.agilealliance.org

A Nova Metodologia

- Artigo de Martin Fowler (1999)
  - Em muitos casos, as metodologias rigorosas não funcionam direito.
- Manifesto Ágil (2001)
  - Pessoas mais que processos e ferramentas
  - Software funcionando mais que documentação
  - Colaboração mais que contratos
  - Lidar com as mudanças mais que seguir planos

Principais Metodologias Existentes

- *Crystal Family*
- *Adaptive Software Development* (ASD)
- SCRUM
- *Feature-Driven Development* (FDD)
- *Dynamic System Development Method* (DSDM)
- eXtreme Programming (XP)
- Agile Modeling (AM)

- Instância do RUP para XP
  - Object Mentor
  - Rational

Requisitos imprevisíveis e Mutantes

"O problema *deste* projeto
é que os requisitos mudam
o tempo todo"

- Rota tradicional:
  - Engenharia de Requisitos: fixar cuidadosa e detalhadamente o escopo antes de desenvolver.
  - Contrato de escopo fixo assinado com sangue (sign-off)
  - Limitar e desencorajar mudanças depois do sign-off

Requisitos imprevisíveis e Mutantes

- Problemas:
  - Planejamento/estimativas sobre atividades de design são muito arriscadas
    - (ficam lindas no Microsoft Project ☺)
  - Expectativa/prioridades do cliente
    - podem mudar
  - Mudanças nos negócios:
    - nem o cliente controla (concorrência, legislação, ambiente econômico)

E no mundo real ?

- Problemas
  - Depender de premissas difíceis de ocorrerem
  - Usar metodologias preditivas quando não dá (*neurose newtoniana*)
  - Achar que você trabalha na NASA (*cargo cult*)

Controlando o imprevisível

- Feedback
  - Implementações que funcionam (ou não) ligam o desconfiômetro.
  - Cliente experimenta com versão limitada (mas funcional) do software.
  - No documento ficou lindo ☺, mas na hora de implementar...☹
- Iterações curtas
  - Cada iteração se baseia na anterior
  - Iteração =/= release
  - Quanto dura uma iteração?
    - XP: 1-3 semanas
    - SCRUM: 4 semanas
    - DSDM, Crystal: até 6 semanas

O cliente adaptativo

- *Problema:*
  - contratos de preço e escopo fixos envolvem estimativas de alto risco.
- *Abordagem:*
  - < confronto → > colaboração, comunicação
  - Engajamento do cliente no desenvolvimento. Ex: cliente residente (XP)
  - Mudanças são feitas cedo, assim que os problemas aparecem.

Unidades intercambiáveis para programação

- *Rota tradicional:*
  - Administração científica (Taylor, 1911)
  - O processo é mais importante que as pessoas
  - Recursos humanos são intercambiáveis
  - Incentivos financeiros melhoram produtividade.
  - Só os papéis são importantes (analista, programador, testador)
  - Quanto mais especializado o trabalhador, melhor ele fará suas tarefas.

Unidades intercambiáveis para programação ?

- *Novas Idéias*
  - Mentalidade enxuta (*lean thinking*, anos 50)
  - A componente principal no desenvolvimento de software são as pessoas (Cockburn, 1999)
  - Recursos humanos não são intercambiáveis (DeMarco, 2002)
    - Ref: O mítico homem-mês (Brooks, anos 70)
  - Motivação intrínseca (fazer bem-feito) é mais importante que competição entre pessoas ou incentivo financeiro (Deming, anos 50)
  - "Generalizing Specialist": especialistas têm que ampliar o leque de conhecimentos fora de sua área, para não ficarem bitolados (Ambler, 2002).

Programadores são profissionais responsáveis !

- Fábrica taylorista
  - Quem faz o trabalho não decide como vai fazê-lo.
  - Estimativas são feitas pelo pessoal de planejamento
  - Operário não participa de projeto ou planejamento
  - Produção é a atividade-fim.
- Desenvolvimento de Software
  - Só quem faz o trabalho tem capacidade técnica para saber como fazê-lo.
  - Estimativas mais confiáveis são feitas pelo desenvolvedor.
  - Desenvolvimento *é* projeto, planejamento
  - "Produção" é automatizável (compilação, empacotamento)

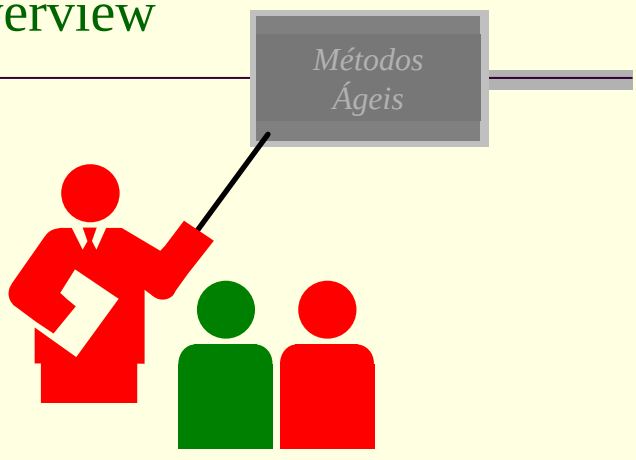
Gerenciando um processo orientado a pessoas

- Aceitação X Imposição.
- Comprometimento.
- Desenvolvedores tomam todas as decisões técnicas.
- Gerência atua facilitando a comunicação com o cliente.
- Transparência entre os participantes (incluindo o cliente)

Processo auto-adaptativo

- Aprendizado para melhoria do processo a cada iteração.
  - O quê fizemos melhor/pior?
  - O quê aprendemos ?
  - O quê nos intriga, ou incomoda, ou “cheira” ?
- Métodos voltados a adaptação:
  - ASD, Crystal
  - XP, não no início: faça “pelo manual” durante as iterações iniciais. Sinergia entre as práticas precisa ser compreendida pela equipe.

MA-Overview



Agilidade - Ref: XP (1997), Agile Alliance (2001)

- Objetivo:
  - Compromisso entre “nada de processo” e processos rigorosos → foco na eficiência.
- Meios:
  - Adaptabilidade,
  - Cada item de processo deve agregar valor,
  - Orientação a pessoas,
  - Comunicação
  - Aprendizado.
- Problemas:
  - Escalabilidade a equipes grandes/dispersas,
  - Cultura: mudança de paradigma

XP

- Prevê a Participação intensa do usuário como membro efetivo da equipe;
- Ciclos muito curtos – uma, duas semanas para dar retorno concreto;
  - Testes, Testes, Refactoring e Testes
    - Faça o essencial para resolver o seu problema
      - Documente Sim! O que realmente for feito
      - Muito Interessante para Projetos Pequenos e Médios

XP

- Simplicidade, Comunicação, Feedback e Coragem
- "Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:
  - Indivíduos e interação **MAIS QUE** processos e ferramentas;
  - Software em funcionamento **MAIS QUE** documentação abrangente;
  - Colaboração com o cliente **MAIS QUE** negociação de contratos;
  - Responder a mudanças **MAIS QUE** seguir um plano.
- Ou seja, mesmo tendo valor os itens à direita, valorizamos mais os itens à esquerda."

RUP x XP ?

- *Existem aspectos positivos e negativos em cada uma das abordagens;*
- *Nem todos os contratos podem ser feitos na base da "camaradagem"*
- *Não pense duas vezes: Teste Duas Vezes!!!*
- *Será que você realmente tem que pagar uma fortuna por uma ferramenta?*
- *A Documentação deve ser feita e faz parte do Produto final! Não vamos retroceder...*
- *Procure usar documentos padronizados*
- Cuidado com aspectos Religiosos

XP (eXtreme Programming)

- Projeto C3 (Chrysler) - Kent Beck (1996)
  - http://www.xprogramming.org
- Valores:
  - Comunicação
  - Simplicidade
  - Feedback
  - Coragem
- Práticas:
  - Pair Programming, Refactoring, Simple Design, Test-driven development
  - Collective Ownership, Coding Standard, Continuous Integration, Sustainable Pace
  - Customer tests, Whole Team, Planning Game, Small Releases, Metaphor

DSDM (Dynamic Systems Development Method)

- Proprietária do con$órcio DSDM (Reino Unido, 1994)
  - http://www.dsdm.org/
- Ciclo:
  - Estudo de viabilidade
  - Estudo do negócio (workshops)
  - 3 ciclos em paralelo, entrelaçados
    - Ciclo do modelo funcional -> análise e protótipos
    - Ciclo de design e build -> engenharia do produto
    - Ciclo de implementação -> implantação operacional
- Princípios:
  - Iterações fixas (2-6 semanas)
  - Releases frequentes
  - Qualidade total
  - Adaptabilidade a mudanças de requisitos

Família Crystal

- Alistair Cockburn (IBM – anos 90)
  - http://alistair.cockburn.us/
- Cada projeto uma metodologia.
  - 4 parâmetros determinam o método de desenvolvimento:
    - Tamanho da equipe
    - Localização geográfica
    - Criticalidade/Segurança
    - Recursos
  - A recomendação de quais os artefatos, papéis e ciclo de desenvolvimento de um projeto é parametrizada.
  - O processo é revisado no fim de cada iteração.

Open Source

- Richard Stallman (anos 80), Linus Torvalds (anos 90)
  - http://www.opensource.org/
  - Inicialmente, para software básico
- Maintainer:
  - Orienta o desenvolvimento
  - Decide o quê vai entrar no software “oficial”
- Catedral X Bazar
  - Catedral: releases pouco freqüentes, desenvolvimento centralizado (GNU, BSD)
  - Bazar: releases freqüentes, desenvolvimento mais espalhado (Linux kernel, apache.org)

Adaptive Software Development

- Jim Highsmith (1997)
  - http://www.adaptivesd.com/
- Sistemas complexos => Resultados imprevisíveis
- Ciclo:
  - → Colaboração → Especulação→Aprendizado
- Abordagem:
  - Do it *wrong* the first time: erre cedo, corrija cedo, não potencialize mal-entendidos.
  - *Good enough quality*: melhor compromisso entre dimensões de qualidade (extrínseca e intrínseca) para os recursos disponíveis.
  - Mecânica: RAD (rapid application development), sessões JAD (joint application development) com o cliente.

SCRUM

- Jeff Sutherland, Ken Schwaber (1993)
  - http://www.controlchaos.com/
- Sprints de 30 dias
  - Estabilizar requisitos em cada iteração
- Scrum (reunião de status) diária (15 min)
  - Guia o desenvolvimento daquele dia
- Foco em gerência e tracking
  - Pode ser combinado com métodos mais prescritivos (ex: XP@scrum)

SCRUM

- O termo **Scrum** é uma metáfora para uma situação em um jogo de Rugby. Esta situação envolve um grupo denso de pessoas, lutando pela posse da bola.
- um pouco de história...
- O **Scrum** não é um método completo... Não requer que seja utilizada nenhuma prática ou técnica para o desenvolvimen-to de software
- Utiliza pequenos times...
- É um método para gerenciamento de um projeto de software

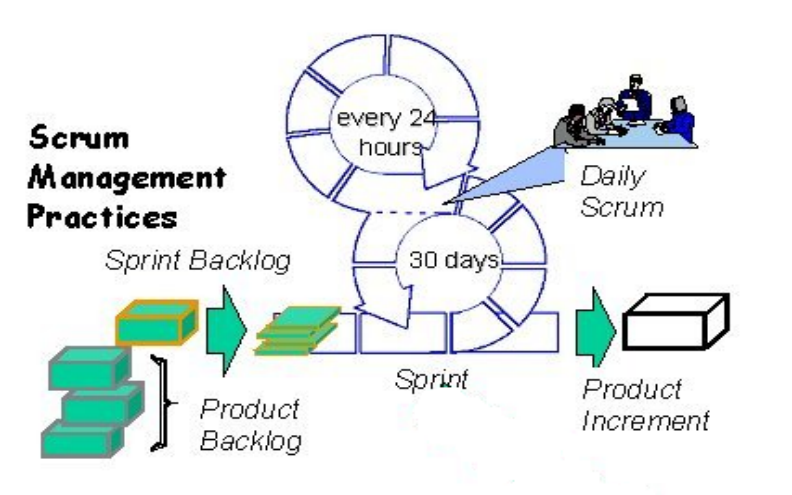
SCRUM

- Processos **definidos** X processos **empíricos** :
  - Um processo *definido* usa uma base de conhecimento sobre o processo: são descritos como reproduzíveis
  - Um processo *empírico* envolve atividades complicadas, não reproduzíveis e com resultados imprevisíveis
- Segundo Ken Schwaber, autor do *Agile Development Methods with **Scrum,*** as atividades envolvidas no desenvolvimento de software são complexas e poucas geram resultados repetidos
- O **Scrum** baseia-se nos métodos utilizados nas fábricas químicas, que utilizam muito *inspeções* e *ajustes*.

SCRUM

- Principais conceitos da metodologia **Scrum**:
  - Time: máximo 7 pessoas, multifuncionais, desenvolvedores e usuários
  - Backlog do Produto
  - Sprint: ciclo de desenvolvimento mensal
  - Sprint Backlog
  - Reunião de Planejamento do Sprint
  - Reunião do Scrum diário
  - Comunicação e retroalimentação
  - ScrumMaster: lider responsável
  - Incremento de produto potencialmente entregável: funcionalidades implementadas, testadas e com performance adequada...

SCRUM



Feature-driven development

- Jeff DeLuca, Peter Coad
  - http://thecoadletter.com/download/fddguide/
- 5 processos:
  - 1- Modelo geral (arquitetura)
  - 2 -Lista de features:
    - Levanta requisitos para todo o projeto
  - 3 - Plan by feature:
    - Define escopo de cada iteração (quais features)
    - Forma times para desenvolver cada feature.
  - (A cada iteração):
    - 4 - Design by feature
    - 5 - Build by feature

Sobre o Feature Driven Development (FDD)

- Baseado em modelos e guiado por características e implementado em ciclos curtos de iterações
- Os ciclos de implementação de uma característica são de no máximo 2 (duas) semanas
- O desenvolvedores gostam porque estão permanentemente recebendo novas tarefas...
- Os clientes gostam por que vêem os resultados rapidamente, gerando uma sensação de fechamento das atitividades...
- Busca-se focar os esforços nas funcionalidades que sejam úteis aos olhos dos clientes...
- Procura-se restringir a lista de funcionalidades (características) àquelas que os usuários podem entender (as features)

Sobre o Feature Driven Development (FDD)

- Uma *feature* ou *característica* é uma função com valor para o cliente e que pode ser implementada em duas semanas ou me-nos e é descrita da seguinte forma:
  - *<ação><artigo><resultado><preposição><artigo><objeto>*
    - Exemplos:
      - calcular o total de uma venda
      - calcular o total de compras de um cliente
- As features podem ser agrupadas. Neste caso são assim descritas:
  - *<ação - verbo no particípio><artigo><objeto>*
    - Exemplos:
      - Comprando um produto
      - Efetivando um pagamento

Sobre o Feature Driven Development (FDD)

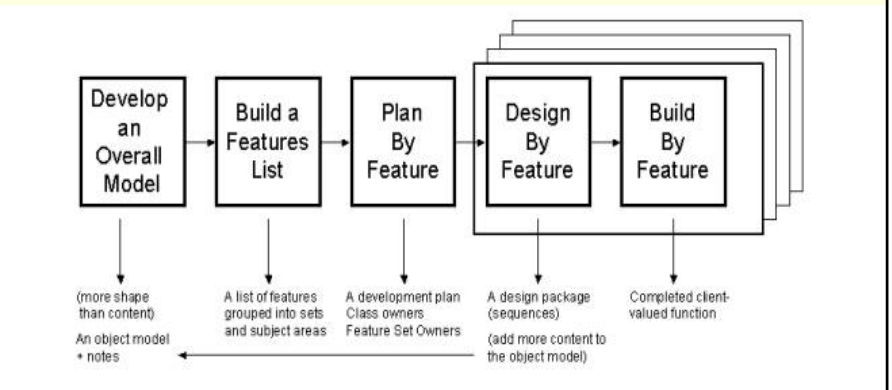
- Sobre os papéis:
  - Papéis chaves
    - Gerente de projeto, arquiteto-chefe, gerente de desenvolvimento, programador-chefe, dono-de-classe, especialista no negócio
  - Papéis de suporte
    - Gerente de liberações, gerente de configuração, administrador de rede, especialista na ferramenta, testador, documentador, etc...
  - Papéis adicionais
    - outros...

Sobre o Feature Driven Development (FDD)

- Sobre as práticas:
  - Modelagem dos objetos de negócio
  - Desenvolver por características
  - Posse de classes de código fonte
    - Cada classe tem um responsável e ele é responsável por sua construção e manutenção
  - Time de características
    - Cada feature tem um responsável
  - Builds regulares
  - Visible Progress Report
  - Inspeções

Sobre o Feature Driven Development (FDD)

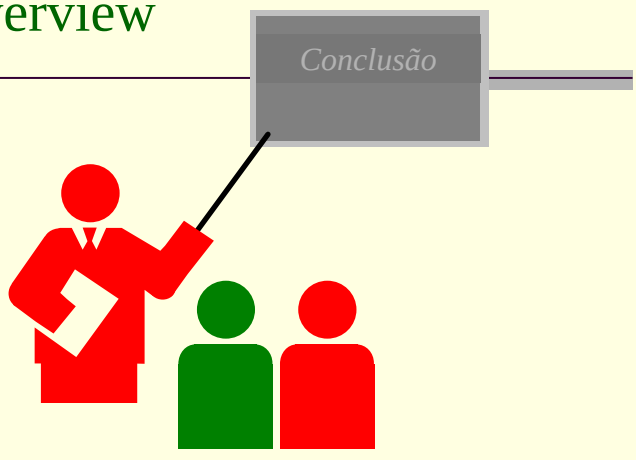
- sobre os processos...



Lean Development

- Mary Poppendieck (2000)
  - http://www.poppendieck.com/
- Focado na identificação de gargalos no processo de desenvolvimento de software
  - Metáfora (boa) de fábrica
  - Empresta idéias de
    - Qualidade Total, (Deming, anos 50)
    - Lean Production (Japão, anos 50)
    - Teoria de Sistemas Dinâmicos (MIT, anos 60)
    - Lean Construction (adaptabilidade na construção civil, anos 90)

MA-Overview



O futuro das metodologias ágeis (survey do Cutter Consortium)

- 200 organizações. Por faturamento:
  - >= US$ 1bi: 13%
  - >= US$ 100, < US$ 1 bi: 17%
  - >= US$ 5m, < US$ 100m : 33%
  - < US$ 5m: 37%
- Exposição a metodologias/normas tradicionais:
  - Rational Unified Process: 51%
  - CMM: 27%
  - ISO 9000: 26%

O futuro das metodologias ágeis (survey do Cutter Consortium)

- % de empresas com mais da metade dos projetos definidos como ágeis
  - 2001: 21%
  - 2002: 34%
  - 2003 (previsão): 50%
- Metodologias ágeis mais usadas (não caseiras)
  - XP: 38%
  - Feature-Driven Development: 23%
  - Adaptive Software Development: 22%
  - DSDM: 19%
- Complexidade dos projetos é similar (rigorosas X ágeis), ágeis trabalham com prazos similares, mas equipes *muito menores.*
- http://www.cutter.com/freestuff/apmupdate.pdf

Conclusão

- Questões em aberto:
  - Times grandes
  - Times dispersos geograficamente
  - Contratos com preço e escopo fixos
  - Resistências culturais
    - Cliente
    - Gerência
    - Desenvolvedores
    - Departamento Jurídico
    - Departamento de Qualidade

Conclusão

- O manifesto ágil:
  - Satisfação do cliente através de entregas mais cedo e contínuas, utilizando ciclos de iteração menores
  - Aceitação e acomodação de requistos em qualquer tempo do desenvolvimento
  - Desenvolvedores e usuários trabalhando juntos
  - Times motivados e em ambientes apropriados
  - Minimização de documentação e maximização de troca de informação *face2face*
  - Encorajamento de atitudes reflexivas e contínuo aprendizado
- O problema da avaliação métodos...

Conclusão

Comparação dos métodos : Tabela de notas

	Princípio	Scrum	FDD	XP
1	A maior prioridade é satisfazer o cliente através da entrega frequente e o mais cedo possível de software com valor agregado	3	1	3
2	Alterações sobre os requisitos são bem vindas, mesmo que tarde no desenvolvimento. Processos ágeis suportam a mudança, para a vantagem competitiva do cliente	3	2	3
3	Entrega de software com frequência, de algumas semanas a alguns poucos meses, com preferência para a escala de tempo mais curta	3	3	3
4	Os especialistas no negócio e os desenvolvedores devem trabalhar juntos diariamente durante o projeto	3	2	3
...	... (12)			
	Total de pontos	32	18	35

Conclusão

- Outros Agile Methods...
  - **ASD (Adaptive Software Development)**
    - Mission-driven, component-driven (results), time-limited, timeboxed; risk driven; change tolerant
  - **Crystal Clear**
    - Strong communications; frequent deliveries; reduce overhead; management by milestones and risk lists
  - **DSDM (Dynamic Systems Development Model)**
    - User involvement, stakeholder collaboration; empowered team; frequent delivery; backtracking to reverse changes; high-level requirements baselining; iterative and incremental development; integrated lifecycle testing;

Comparando as metodologias ...

METODOLOGIA	DIRIGIDA A	PRIORIZA
Engenharia da Informação	Construção do Banco de Dados	Informação
Análise Estruturada	Construção de sistemas	Procedimentos
Análise Orientada a Objeto	Construção de Componentes	Objetos
Métodos Ágeis	Implementação contínua de funções	Funções com valor para o cliente