Metodologias Ágeis para o Desenvolvimento de Software

Adolfo Neto (DAINF – UTFPR)
http://www.dainf.ct.utfpr.edu.br/~adolfo

Quem sou eu?

- Professor do DAINF-UTFPR desde 07/2008
- Formação:
  - Bacharel em Ciência da Computação – UFAL (1990-1994)
    - Fiz algumas implementações significativas em Pascal, Lisp e C++.

Quem sou eu?

- Bacharel em Ciência da Computação – UFAL – 1990-1994
  - Nesta época os principais livros de Engenharia de Software (Sommerville e Pressman) não falavam de métodos ágeis.
  - Hoje falam!
  - Orientação a Objetos estava começando. Aprendi a programar em C++

Quem sou eu?

- Mestre em CC – UFPE – 1994-1996
  - Comprei meu primeiro livro de Java: Java 0.9!
  - Não implementei nada significativo neste período
  - Fiz a disciplina Engenharia de Software. O trabalho final do curso foi um documento de requisitos!!
- Professor Substituto da UFAL – 1995-7
  - Lecionei Engenharia de Software
  - Não implementei nada significativo neste período (e os alunos TAMBÉM NÃO!!!!)

Quem sou eu?

- Professor do CEFET-AL – 1997-2003
  - Lecionei C++, Pascal, Java
  - Não implementei nada significativo neste período (e os alunos também não!)

Quem sou eu?

- Doutorado em CC – USP – 2003-2007
  - 2003: Duas disciplinas de POO: Smalltalk, AspectJ e outros tópicos avançados em POO
    - Primeiro contato com metodologias ágeis. As idéias me pareceram MUITO atraentes!! Meio termo entre o caos e o excesso de burocracia!
    - CRC, Padrões de Projeto, Arquitetura de SW, etc.
    - Na segunda disciplina, professor foi Fabio Kon, que depois veio a ser um dos fundadores da Agilcoop e do CCSL

Quem sou eu?

- Doutorado em CC – USP – 2003-2007
  - 2005: fui monitor da disciplina **Tópicos (Avançados) de Programação Orie**
  - Voltei a implementar (solitariamente) sistemas interessantes (provadores de teoremas)
    - KEMS: http://kems.iv.fapesp.br

Quem sou eu?

- Inmetrics 2005-2007: Desenvolvedor Java
  - Comecei desenvolvendo um sistema fruto de uma terceirização
    - Péssima experiência com métodos tradicionais:
      - Casos de Uso
      - Documento de Requisitos
      - Diagramas UML – Rational Rose
      - SEM NUNCA VER A CARA DO CLIENTE FINAL!!!
    - Mas, tive contato com alguma técnicas ágeis:
      - Testes unitários automáticos
      - Cobertura de testes (Clover)

Quem sou eu?

- Inmetrics 2005-2007: Desenvolvedor Java
  - Experiência com Scrum em um projeto
  - Nunca programação pareada
    - Horário flexível demais
- Rápidas passagens pela USP-Ribeirão Preto, CEFET-SP-Sertãozinho e UDESC-Joinville
- Longo tempo sem programar... (desde o final de 2006)
- Mas sempre curioso com relação a métodos ágeis (e acompanhado os passos da Agilcoop)

Roteiro

1. O que são e como surgiram as metodologias ágeis?
2. Dois exemplos de metodologias ágeis:
   - XP e Scrum
3. Quais são as principais "técnicas" usadas nas metodologias ágeis?
   - Programação Pareada
   - Desenvolvimento com testes a priori
   - Refatoração

Roteiro

1. Metodologias Ágeis e o Mercado de Trabalho
2. Quem são os principais divulgadores/especialistas/praticantes?
3. Eventos sobre metodologias ágeis no Brasil e no mundo
4. Bibliografia básica sobre metodologias ágeis

O que são e como surgiram as metodologias ágeis?

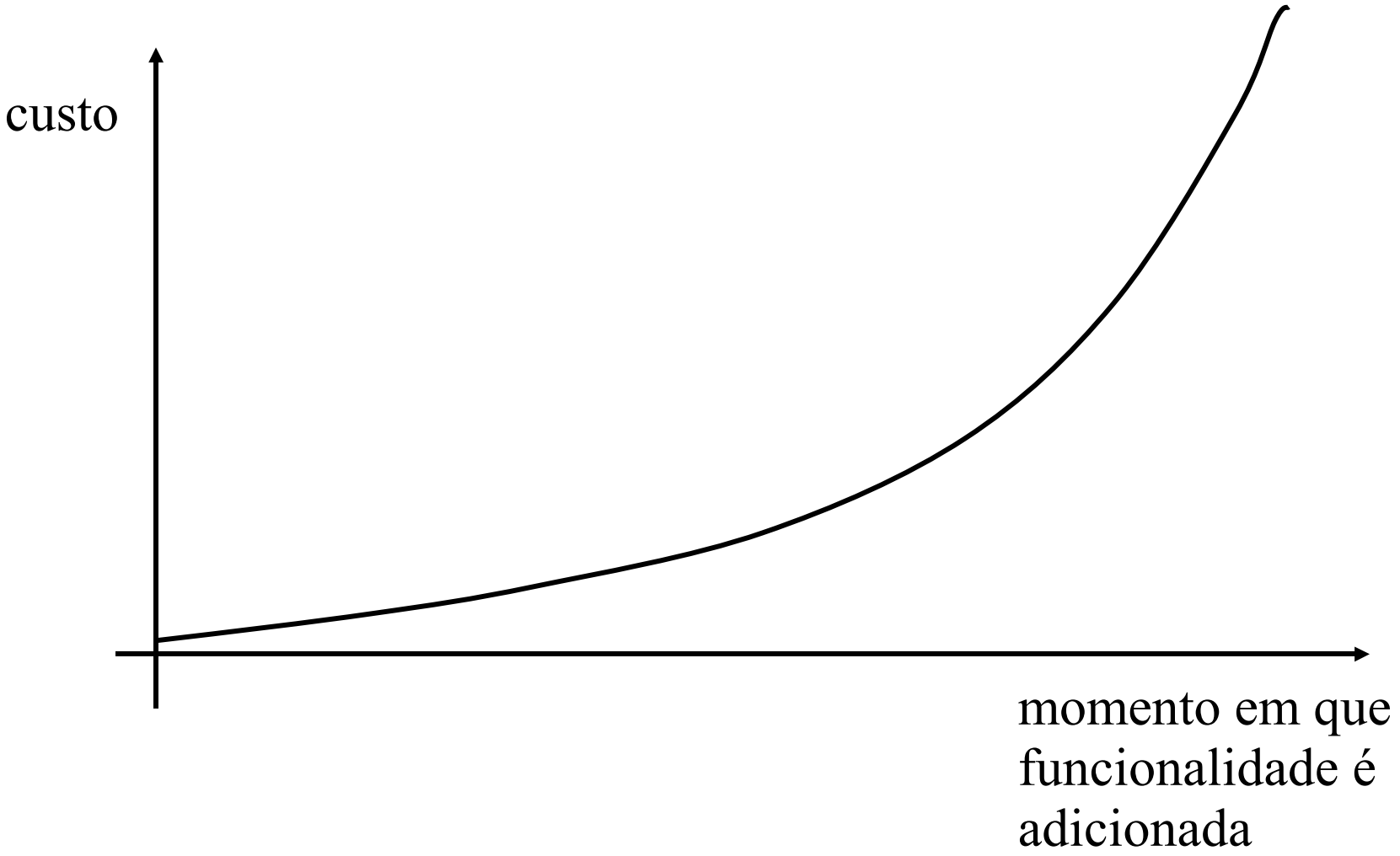
Engenharia de Software

- Não é parecida com Engenharia Civil!
  - Após construir uma casa não é fácil mudar uma parede de lugar!!!
  - Mas em software, “mudar uma parede de lugar” é sim relativamente fácil...
- Tampouco é muito parecida com outras engenharias!!!
- Software é flexível!!!

Engenharia de Software Tradicional

- Desenvolvimento ad-hoc de software em geral produz resultados muito ruins
  - Especialmente em sistemas grandes
- Desejo de criar uma **engenharia** para que se tenha controle sobre desenvolvimento de software
- Engenharias tradicionais colocam grande ênfase em **projetar antes de construir**

Visão Tradicional da Evolução do Software



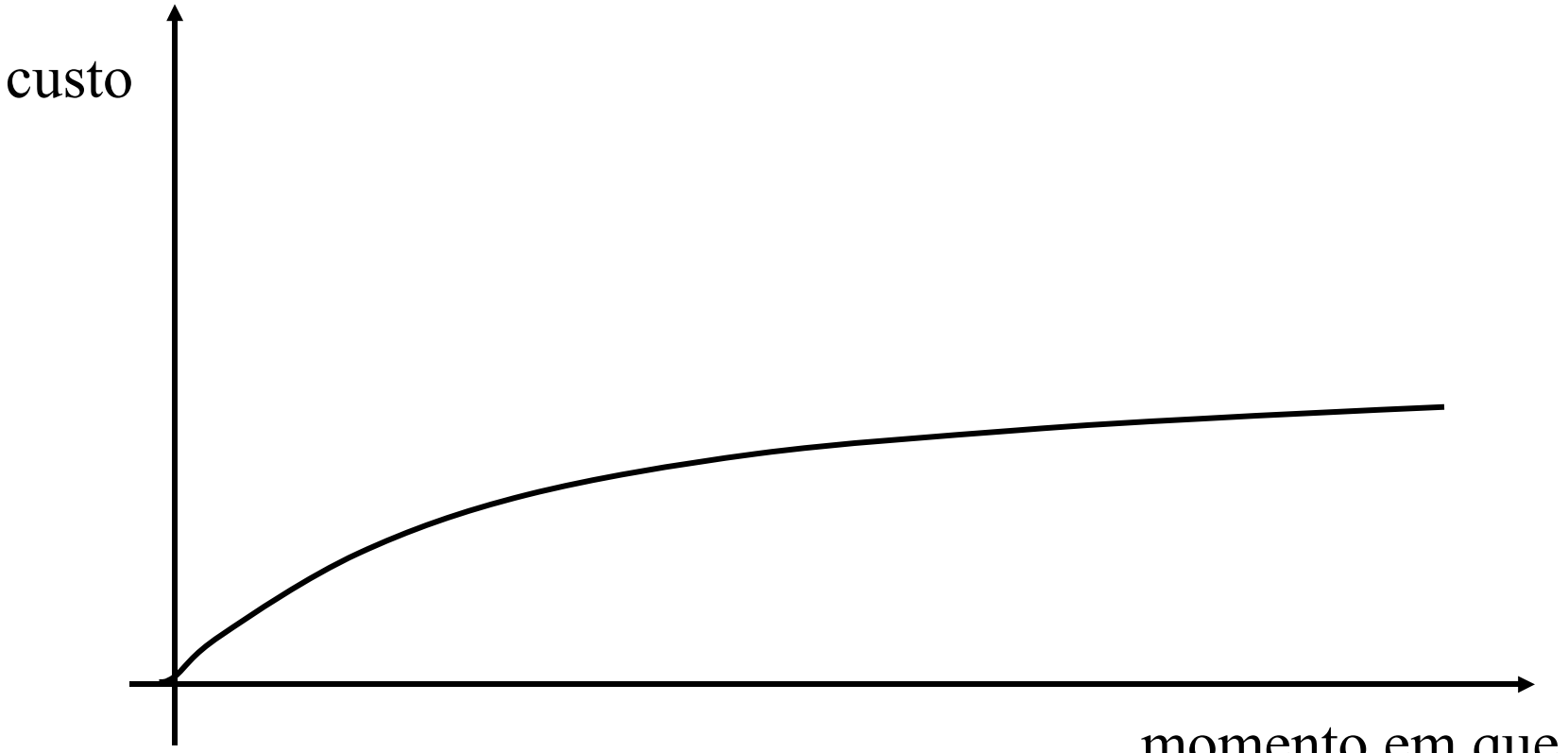
Queremos Poder Alterar Software

- No inicio do projeto, normalmente não se sabe precisamente o que se quer
- Software evolui para atender ao business
  - Software nunca fica “pronto”
- Obviamente isso só é possível porque software é uma entidade abstrata

Portanto…

- Precisamos parar de tentar evitar mudanças
  - Mudanças são um aspecto intrínseco da vida do software
- Precisamos de uma metodologia de desenvolvimento que nos permita alterar **constantemente** o código sem comprometer sua qualidade

O que queremos é…



Novos ventos no mundo do Desenvolvimento de Software

- Sociedade demanda
  - grande quantidade de sistemas/aplicações
  - software complexo, sistemas distribuídos, heterogêneos
  - requisitos mutantes (todo ano, todo mês, todo dia)
- Mas, infelizmente,
  - não há gente suficiente para desenvolver tanto software com qualidade.

Problemas

- Com metodologias de desenvolvimento
  - Supõem que é possível prever o futuro
  - Pouca interação com os clientes
  - Ênfase em burocracias (documentos, formulários, processos, controles rígidos, etc.)
  - Avaliação do progresso baseado na evolução da burocracia e não do código
- Com software
  - Grande quantidade de erros
  - Falta de flexibilidade

Como resolver esse impasse?

- Melhores Tecnologias
  - Padrões de Projeto (reutilização de idéias)
  - Componentes (reutilização de código)
  - Middleware (aumenta a abstração)
- Melhores Metodologias
  - Métodos Ágeis
  - outras...

Metodologias de Desenvolvimento de Software OO

- “Tradicionais”
  - Comunidade de Engenharia de Software
  - IEEE/ACM ICSE
  - p.ex. Carnegie-Mellon SEI
  - RUP, CMM, etc.

- Ágeis
  - Comunidade de POO
  - ACM OOPSLA
  - p.ex. Johnson @ Illinois, Beck, Cockburn, Jeffries, Cunningham…
  - XP, Crystal, Scrum, etc.

Métodos Ágeis de Desenvolvimento de Software

- Movimento iniciado por programadores experientes e consultores em desenvolvimento de software.
- Questionam e se opõem a uma série de mitos/práticas adotadas em abordagens tradicionais de Engenharia de Software e Gerência de Projetos.
- Manifesto Ágil:
  - Assinado por 17 desenvolvedores em Utah em fevereiro/2001.

O Manifesto do *Desenvolvimento Ágil de Software*

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

Princípios do Manifesto Ágil

- Objetivo: satisfazer o cliente entregando, rapidamente e com freqüência, sistemas com algum valor.
  - Entregar versões funcionais em prazos curtos.
  - Estar preparado para requisitos mutantes.
  - Pessoal de negócios e desenvolvedores juntos.
  - Troca de informações através de conversas diretas.

Principais Métodos Ágeis

- Concentrar-nos-emos em :
  - Programação eXtrema (XP)
- Outros métodos ágeis interessantes:
  - Crystal (uma família de métodos)
  - Scrum
  - Adaptive Software Development
  - Feature Driven Development
  - etc.

Agilidade

- Existe em outras áreas?
  - Produção televisiva (comercial)
  - Produção cultural
    - Livros
      - Os escritores não começam a escrever pelo primeiro capítulo e vão até o último...

Programação Extrema (XP)

O Novo XP

- XP = e**X**treme **P**rogramming = Programação Extrema
- BECK, Kent. Extreme Programming Explained – Embrace Change. 1999.
- BECK, Kent e ANDRES, Cynthia.Extreme Programming Explained – Embrace Change. Second Edition. Addison Wesley, 2005.

KENT BECK

Kent Beck e os padrões

- I first discovered patterns as an undergraduate at the University of Oregon. Many of the students in my freshman dorm (accurate nickname "Gonads") were in the School of Architecture. Since I had been drawing goofy house plans since I was six or seven, they pointed me in the direction of Christopher Alexander. I read all of The Timeless Way of Building standing up in the university bookstore over the course of several months. I had been working at Tektronix for a year and a half when I came across Alexander again. I found a battered old copy of Notes on the Synthesis of Form in Powell's. Alexander's excoriation of methodologists in the introduction to the second edition resonated with my biases, leading me to Timeless Way again. It seemed everything he didn't like about architects, I didn't like about software engineers. I convinced Ward Cunningham that we were onto something big.

- Ward and I were consulting with a group that was having trouble designing a user interface. We decided, in Ward's VW Vanagon on the way over, to try out the pattern stuff we'd been studying. Alexander said the occupiers of a building should design it, so we had the users of the system design the interface. Ward came up with a five pattern language that helped them take advantage of Smalltalk's strengths and avoid its weaknesses:

  - Window per Task
  - Few Panes
  - Standard Panes
  - Nouns and Verbs
  - Short Menus
- Fonte: http://c2.com/ppr/about/author/kent.html

Valores de XP

- Comunicação
- Simplicidade
- Retroalimentação (Feedback)
- Coragem
- Respeito

Valores de XP

- Comunicação
  - A maioria dos problemas e erros são causados por falta de comunicação
  - Maximizar a comunicação entre a equipe e o cliente
  - Comunicação entre pessoas: direta
  - Comunicação entre pessoas e artefatos: estes devem ser de leitura fácil e estar atualizados

Valores de XP

- Comunicação
- Simplicidade
- Retroalimentação (Feedback)
- Coragem
- Respeito

Valores de XP

- Comunicação
- Simplicidade
- Retroalimentação (Feedback)
- Coragem
- Respeito

Valores de XP

- Comunicação
- Simplicidade
- Retroalimentação (Feedback)
- Coragem
- Respeito

Valores contrários XP

- Comunicação
  - preferir papel a olho-no-olho
- Simplicidade
  - fazer mais do que o necessário para resolver o problema atual pensando em antecipar a resolução de problemas futuros
- Retroalimentação (Feedback)
  - Retroalimentação só bem perto do final. Contrato assinado.
- Coragem

- Respeito
  - Longas jornadas de trabalho para terminar o projeto

Princípios de XP - I

- Humanidade
- Economia
- Benefício mútuo
- Auto-similaridade
- Melhora contínua
- Diversidade
- Reflexão

Princípios de XP - II

- Fluxo (Flow)
- Oportunidade
- Redundância
- Falha
- Qualidade
- Passos de bebê
- Responsabilidade aceita

Práticas de XP

- Categorias
  - Análise de Requisitos e Planejamento
  - Fatores relacionados a equipes e fatores humanos
  - Projeto (Design)
  - Codificação e lançamento (liberação) de código
- Muitas práticas poderiam se encaixar em diferentes categorias...

Práticas Principais

- Análise de Requisitos e Planejamento
  - Estórias
  - Ciclo semanal
  - Ciclo quinzenal
  - Folga

Práticas Principais

- Fatores relacionados a equipes e fatores humanos
  - Sentar junto
  - Time inteiro
  - Ambiente de trabalho informativo
  - Trabalho energizado
  - Programação Pareada

Práticas Principais

- Projeto (Design)
  - Design incremental
  - Programação com testes primeiro

Práticas Principais

- Codificação e lançamento (liberação) de código
  - *Build* de dez minutos
  - Integração contínua

Práticas Secundárias/Decorrentes

- Análise de Requisitos e Planejamento
  - Envolvimento real do cliente
  - Entrega (Deployment) incremental
  - Contrato de escopo negociado
  - Pague-pelo-uso

Práticas Secundárias/Decorrentes

- Fatores de equipe e fatores humanos
  - Continuidade da equipe
  - Times que encolhem

Práticas Secundárias/Decorrentes

- Projeto (Design)
  - Análise Causa-Raiz

Práticas Secundárias/Decorrentes

- Codificação e lançamento (liberação) de código
  - Código e testes
  - Código compartilhado
  - Base de código única
  - Entrega (deployment) diária

Referências

- BECK, Kent e ANDRES, Cynthia.Extreme Programming Explained – Embrace Change. Second Edition. Addison Wesley, 2005.
- MARCHESI, Michele. The New XP: An Analysis of Extreme Programming Explained - Second Edition.
- http://en.wikipedia.org/wiki/Extreme_progra

Referências

- http://en.wikipedia.org/wiki/Kent_Beck

Gerenciamento de Equipes com Scrum

O que é Scrum?

- Processo de controle e gerenciamento
- Processo iterativo de inspeção e adaptação
- Usado para gerenciar projetos complexos
- Entrega valor de negócio periodicamente
- Compatível com CMMi até o nível 3, ISO e outras metodologias
- Extremamente simples, mas muito difícil

Princípios

- Os envolvidos trabalham em equipe com:
  - Responsabilidade
  - Transparência
  - Honestidade
  - Auto-organização

- Fornecer software funcionando
  - De forma incremental
  - Potencialmente entregável

Tipos de Processo

- Processo prescritivo
  - Funciona em ambientes controlados
- Processo empírico
  - Funciona para processos complexos e imprevisíveis

Origens de Scrum

- Jeff Suttherland - www.jeffsutherland.com
- Ken Schwaber - www.controlchaos.com
- Mike Beedle - www.mikebeedle.com

- Inspiração
  - Desenvolvimento Iterativo e Incremental em empresas (DuPont) nos anos 80
  - Lean – Sistema de Produção da Toyota
  - Produtividade de Equipes

Papéis

- Product Owner
- Equipe
- ScrumMaster

Product Owner

- Define a visão do produto
- É o representante dos clientes
- Entende do negócio
- Define o objetivo do Sprint
- Elege prioridades de negócio
- Gerencia o Backlog

Equipe

- Responsável pela entrega
- Multi-funcional
- Auto-organizada e auto-gerenciada
- Todos os membros igualmente comprometidos com um objetivo comum
- Geralmente equipes pequenas (até 10)
  - Equipes grandes geralmente se comportam como várias equipes pequenas

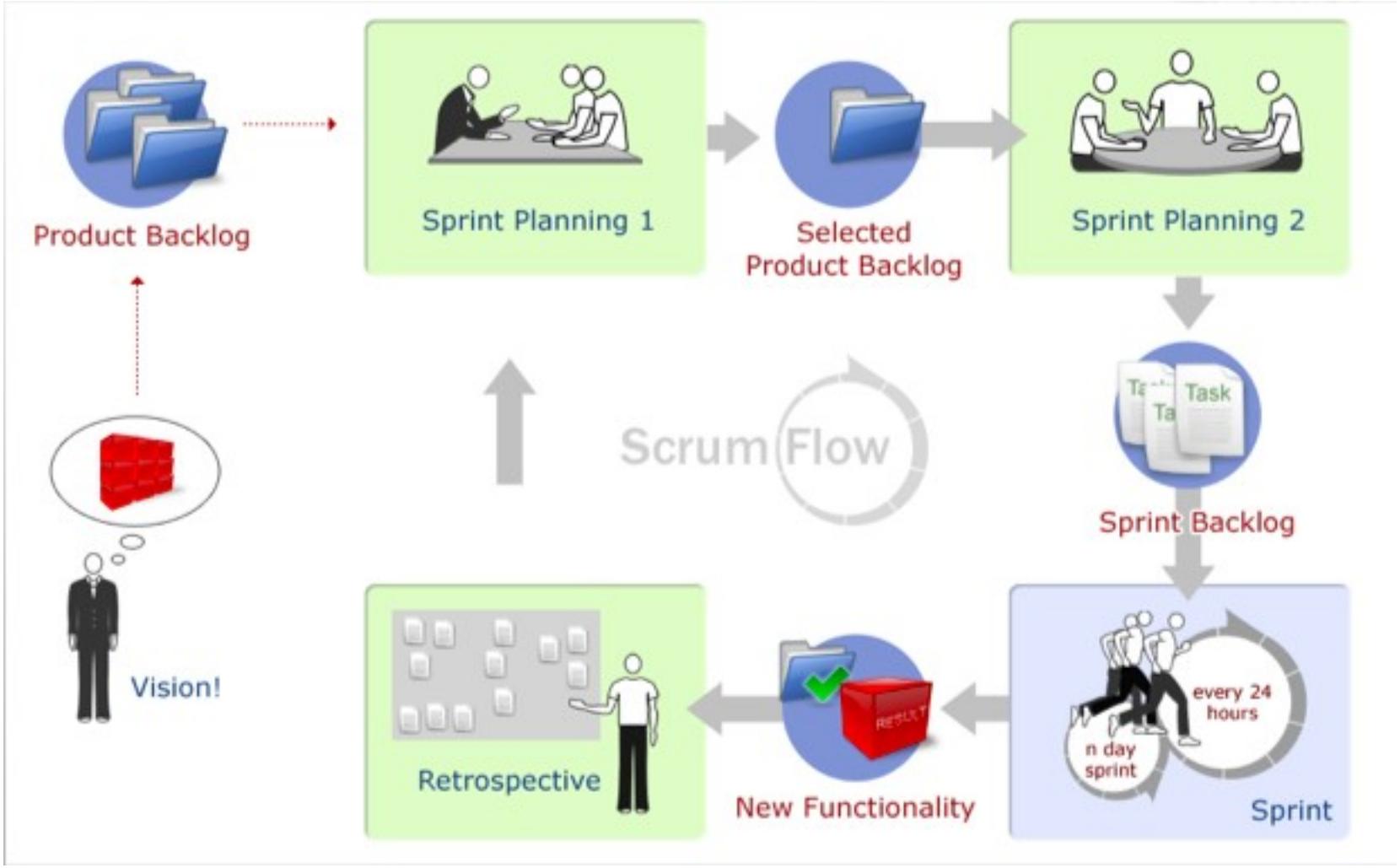
ScrumMaster

- Conhecimento do processo
- Remove impedimentos
- Protege a equipe
  - Riscos e interferências externos
  - Excesso de otimismo
- Auxilia o Product Owner a maximizar o retorno do investimento

Artefatos

- Backlog do Produto
- Backlog Selecionado
  - Não muda durante o Sprint
- Backlog do Sprint
  - Tarefas priorizadas
- Backlog de Impedimentos

Visão do Processo



Planejamento

- Reunião de Estimativa
  - Entrada: Backlog do Produto priorizado
  - Saída: Itens relevantes do Backlog do Produto estimados
  - Participantes: Equipe e ScrumMaster
- Sprint Planning I
  - Entrada: Backlog priorizado e estimado
  - Saída:
    - Objetivo do Sprint
    - Backlog Selecionado
  - Participantes: Todos

Sprint

- Sprint = Iteração
- Tamanho fixo
  - Recomendação:
    - Antigamente: 30 dias
    - Atualmente: 2 semanas
- Durante o Sprint:
  - Daily Scrum (Stand-Up Meeting)

Daily Scrum 1/2

- Pequenos encontros diários da equipe
  - geralmente pela manhã
  - galinhas e porcos (só os porcos falam)
  - todos os porcos devem participar
- Questões que aparecem devem ser resolvidas após a reunião
- Tempo fixo: 15 minutos

Daily Scrum 2/2

- Questões que devem ser respondidas por cada porco:
  1) O quê você fez ontem?
  2) O quê você vai fazer hoje?
  3) Quais os problemas encontrados?

- Evita: Como um projeto atrasa um ano?
  - Um dia por vez ...
  - Qualquer deslize pode ser corrigido de imediato

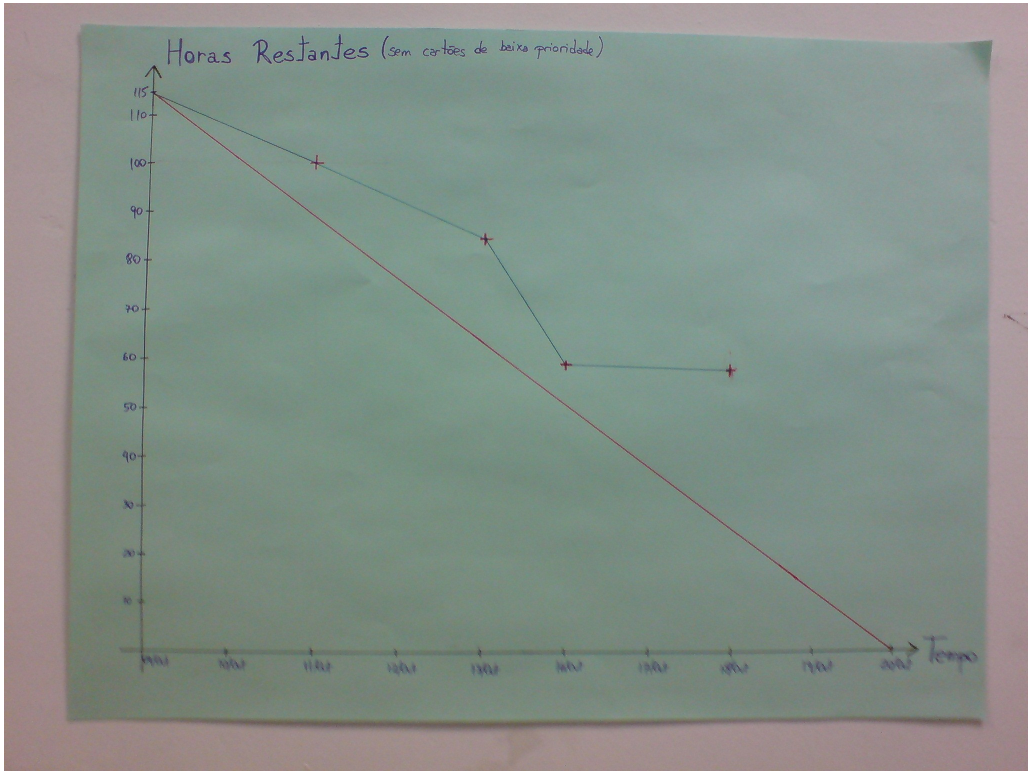
- ScrumMaster cuida dos impedimentos identificados

Local do Encontro

- Sempre o mesmo local e hora
- Pode ser o local de desenvolvimento
- Sala bem equipada, quadro branco, etc.
- A sala já deve estar arrumada antes
- Todos devem participar
- Galinhas ficam na periferia
- Ficar em pé ajuda a terminar rápido
- Punições (atrasos/faltas)

Acompanhamento

- ScrumMaster atualiza o Burn-Down Chart
  - um gráfico de “quanto falta”



Sprint Review

- Reunião onde o Product Owner:
  - Valida os itens entregues
  - Verifica se o objetivo do Sprint foi atingido
- Na forma de demonstração ou apresentação
- Momento para celebrar o sucesso

Retrospectiva

- Momento para reflexão e aprendizado
- Gera discussões para alimentar o próximo Sprint
- Quem está no controle?
  - Equipe: Backlog
  - Instituição: Backlog de Impedimentos

Sprint de Entrega

- Não faz parte do Scrum padrão, mas é bem usado na prática
- Um último *Sprint* para “fechar” o produto
- O objetivo é:
  - Preparar a versão de produção
  - O foco é a eliminação de erros

É só isso?

- Scrum é simples, mas não é fácil!
- Comece usando Scrum para a equipe identificar problemas
- Pode ser usado com metodologias focadas em aspectos técnicos

Precisamos de Certificação?

- Certified Scrum Master
- Certified Scrum Product Owner
- Certified Scrum Practioner
- Certified Scrum Trainer
- Certified Scrum Coach

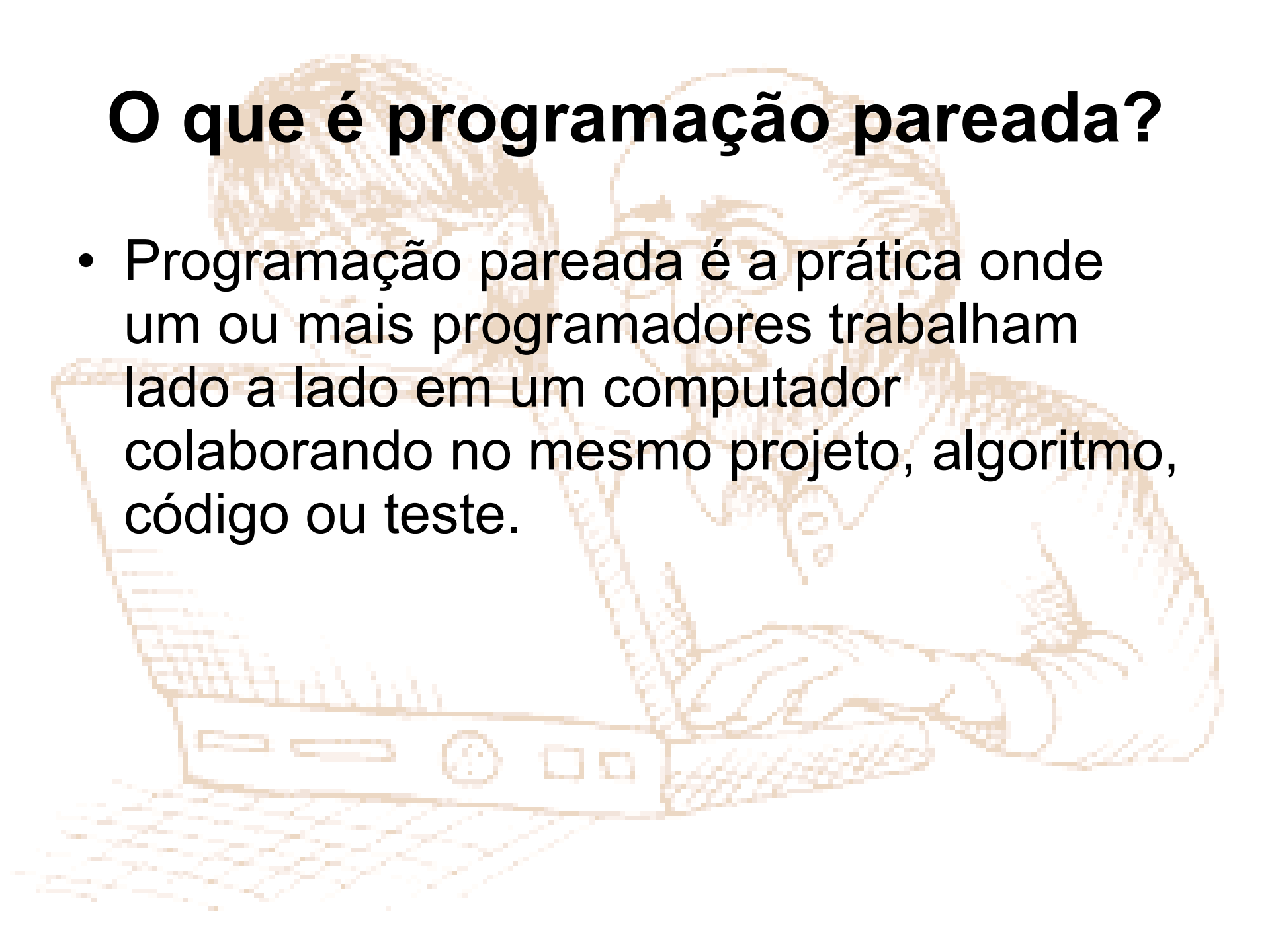
Mais Informações

- Agille Alliance - www.agilealliance.org
  - Ótima fonte sobre métodos ágeis
- Scrum Alliance - www.scrumalliance.org/
- Mountain Goat Software
  - www.mountaingoatsoftware.com
  - Site de um treinador de *Scrum Masters*
- Site do Ken Schwaber - www.controlchaos.com

Programação Pareada

O que é programação pareada?

- Programação pareada é a prática onde um ou mais programadores trabalham lado a lado em um computador colaborando no mesmo projeto, algoritmo, código ou teste.



A principal vantagem...

Quando se está trabalhando em par se trabalha o dia todo. Pois ao trabalhar sozinho, você vê o seu e-mail, lê blogs e etc. E essas coisas não acontecem com programação em par. Ao fim de um dia de programação em par você está cansado, pois você realmente pensou e trabalhou o dia todo. Mas você fica contente, pois sabe que teve o trabalho realizado.

Obie Fernandez em sua palestra no Rails Summit Latin America 2008

O que é programação pareada?

- O par é composto de:
  - um motorista: que digita no computador ou registra o projeto
  - um navegador: que observa o trabalho do motorista e identifica problemas, clarifica questões e faz sugestões.
- Os parceiros devem trocar de papéis de tempos em tempos para compartilhar o trabalho igualmente e obter o máximo da sua experiência com a programação pareada.

Importante!

- Um controla o teclado e o mouse
- O outro observa *criticamente*
- Os dois estão continuamente dialogando
- De tempos em tempos, o controle muda de mãos
- Ao longo de um projeto, os companheiros são trocados

Vale a pena?

- Aparentemente não...
- Usa dois “recursos” em vez de um
  - Pagar o salário de dois desenvolvedores para trabalharem num mesmo problema???
- Os dois usam o mesmo computador a maior parte do tempo (o outro fica ocioso...)

Por que fazer?

- Pesquisas mostraram que:
  - Aumenta a qualidade do software
  - As tarefas são completadas em menos tempo (mas não necessariamente em menos da metade do tempo que um programador sozinho gasta)
- Tão importante quanto a programação pareada são a análise pareada e o projeto (design) pareado

Por que fazer?

- À medida que os pares se acostumam a trabalhar em pares, a produtividade aumenta
- Retroalimentação, debate e troca de idéias entre os parceiros diminuem significativamente a probabilidade de se continuar com um mau *design*.

Por que fazer?

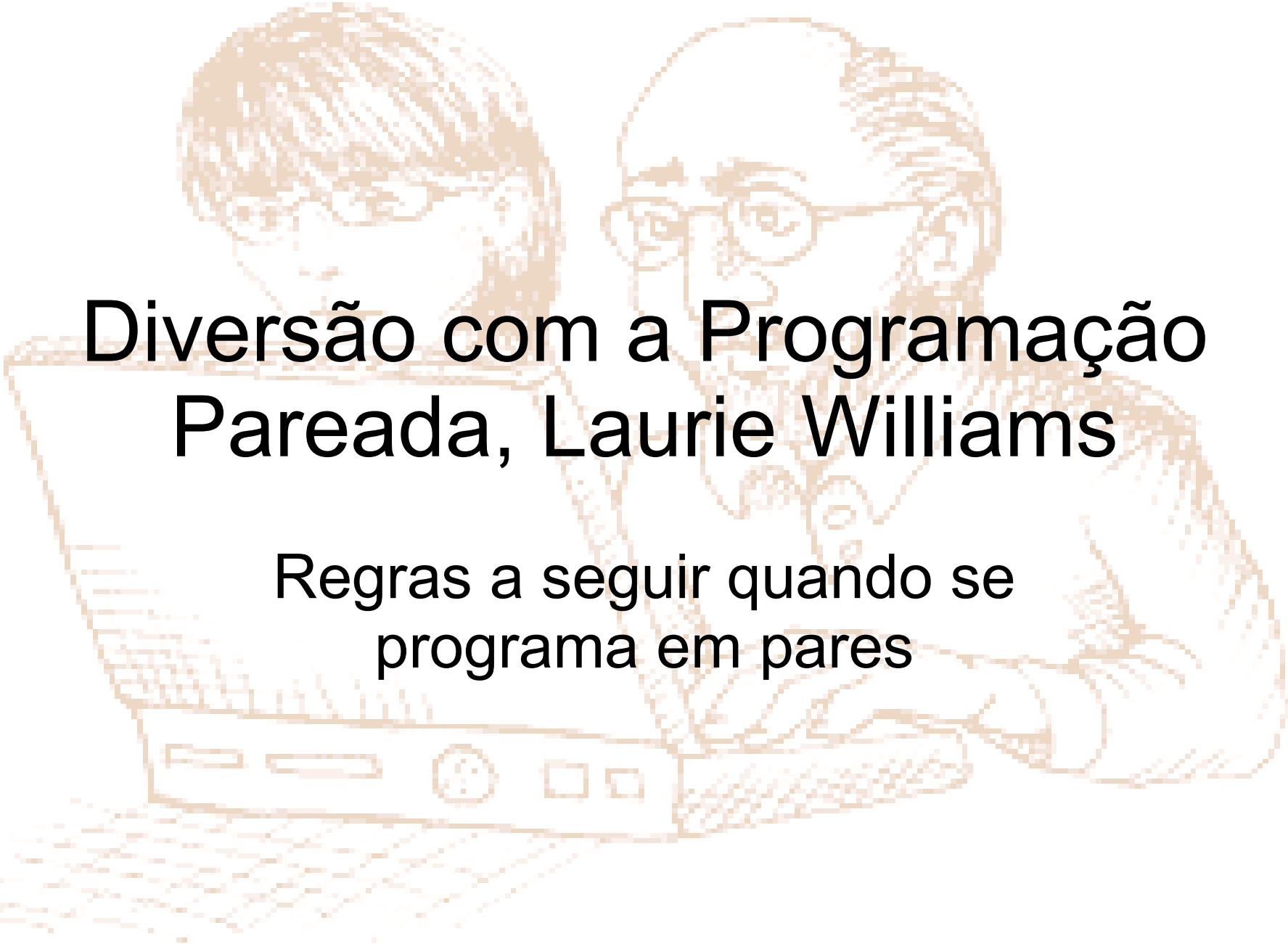
- A presença do outro força cada programador a concentrar-se totalmente no problema em questão
- Cada parceiro ajuda o outro a manter a disciplina de escrever testes primeiro
- Treinamento cruzado: um programador ensina o outro

Quando (talvez) não fazer?

- Desenvolvimento distribuído
  - Mas existe plugin para Eclipse que permite programação pareada distribuída
  - E existe VNC e MSN/Skype...
- Problemas culturais
  - Dificilmente isto se aplica aos brasileiros...

Mais pesquisas...

- Pesquisas adicionais (em empresas e em diferentes países) são necessárias para certificar a efetividade da programação em pares
- Sempre usando um grupo de controle que não utiliza programação pareada



Diversão com a Programação Pareada, Laurie Williams

Regras a seguir quando se programa em pares

Faça!

- **Fale**
  - Nem um minuto deve se passar sem que você esteja se comunicando de alguma forma.
- **Escute**
  - Cada parceiro deve estar totalmente envolvido no projeto
- **Troque os papéis**
  - Troque de lugar para obter novas perspectivas e compartilhar o trabalho.
- **Seja paciente**
  - É uma virtude! Explicando-se você ajuda você mesmo a aprender.

Faça!

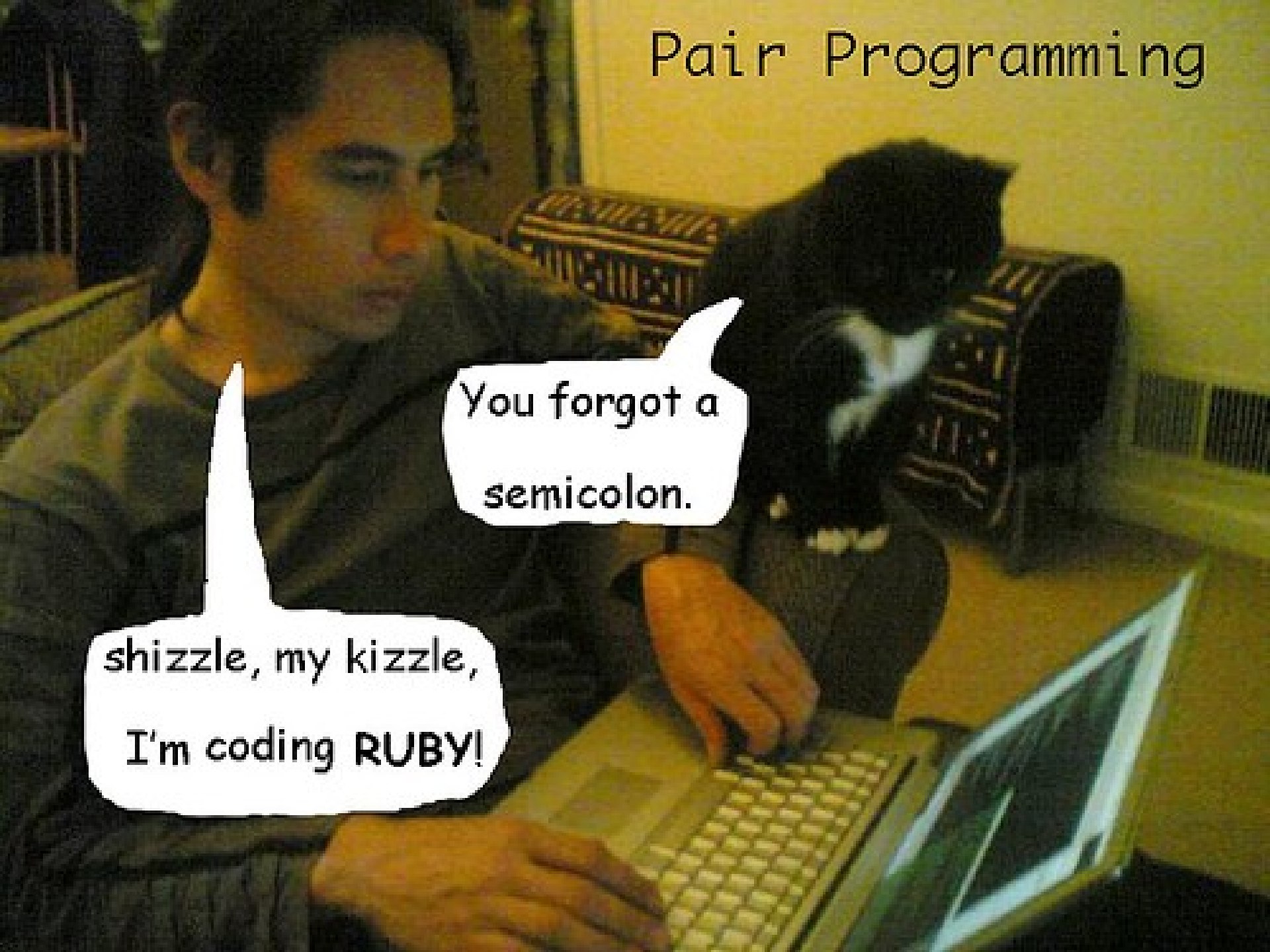
- **Respeite**
  - Todo mundo tem algo a oferecer.
- **Faça paradas**
  - Quando você voltar, você poderá perceber algo que não estava percebendo antes.
- **Prepare-se**
  - Faça qualquer trabalho preliminar antecipadamente.
- **Limpe-se**
  - Como você se sentiria se a pessoa com quem você fosse passar horas cheirasse mal?
- **Divirta-se**
  - Trabalhar com outra pessoa pode ser muito mais divertido

Não Faça!

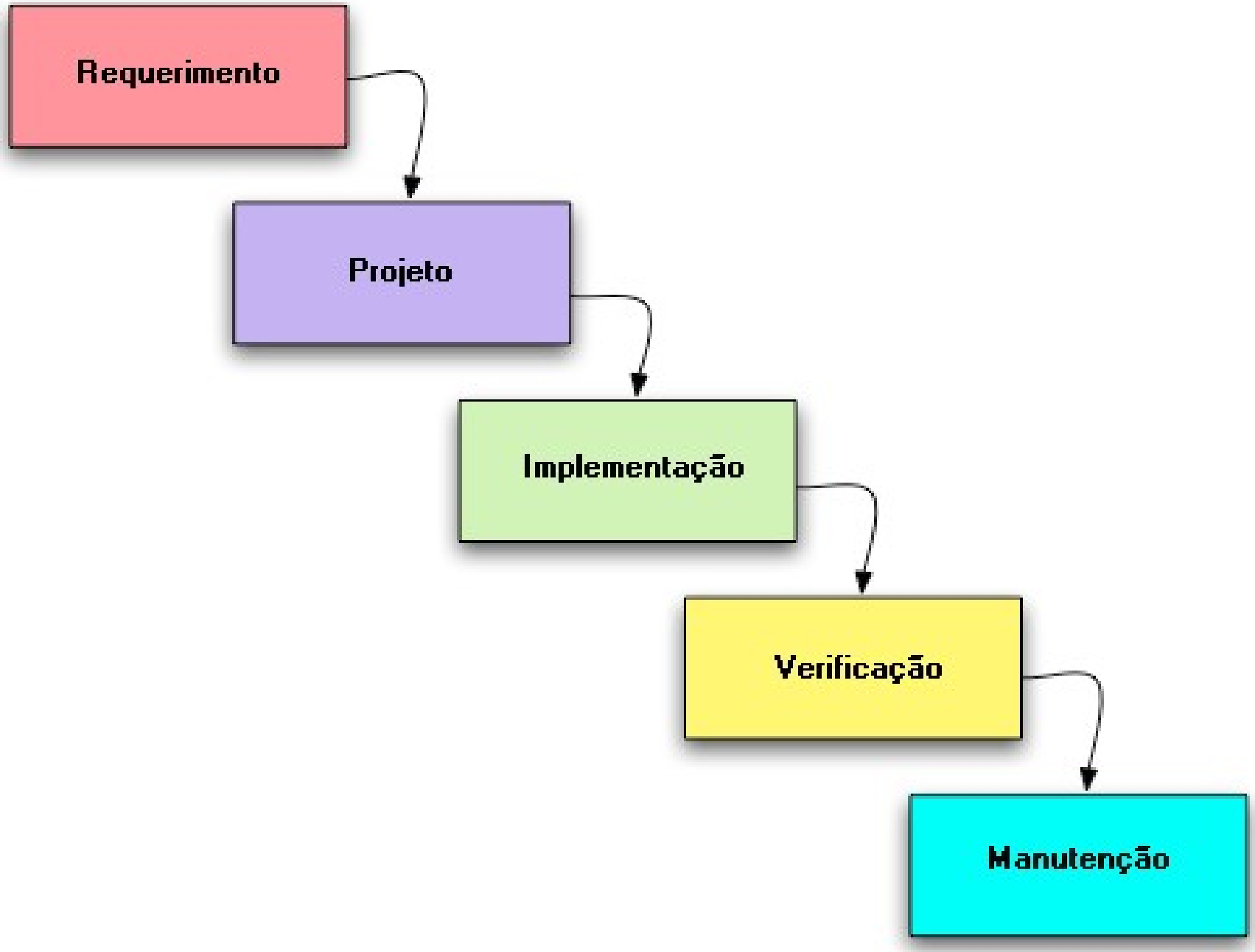
- **Seja mandão**
  - Você não quer ser a única pessoa na turma com a qual ninguém quer trabalhar!
- **Fique intimidado**
  - Você sabe muito mais do que você acha que sabe.
- **Seja quieto**
  - Não tenha medo de falar se você não concorda com seu parceiro.
- **Sofra em silêncio**
  - Se você tem um problema com seu desempenho na programação, informe ao seu professor.

Referências

- WILLIAMS, Laurie. Fun with Pair Programming. Disponível em < http://agile.csc.ncsu.edu/pairlearning/worksheet.pdf>. Acesso em: 30 out. 2008.
- WILLIAMS, Laurie. Vídeo “Divirta-se com a Programação Pareada: uma introdução à Programação Pareada para estudantes”. Disponível em < http://agile.csc.ncsu.edu/pairlearning/educators.php#ppvideo>. Acesso em: 30 out. 2008.
- Williams, Laurie, Kessler, Robert R., Cunningham, Ward, and Jeffries, Ron, *Strengthening the Case for Pair-Programming*, IEEE Software, Vol. 17, No., 4, pp. 19-25, July/Aug 2000
- Williams, Laurie and Kessler, Robert R., *All I Really Need to Know about Pair Programming I Learned In Kindergarten,* Communications of the ACM, Vol. 43, No., 5, pp. 108-114, May 2000.
- Williams, L., *The Collaborative Software Process*. PhD Dissertation, 2000.


Pair Programming
You forgot a semicolon.
shizzle, my kizzle, I'm coding RUBY!

Desenvolvimento com testes a priori


Requerimento
Projeto
Implementação
Verificação
Manutenção

Testes

- No modelo em cascata:
  - Elicitação de requisitos
  - Projeto
  - Construção (implementação ou codificação)
  - Integração
  - Teste e depuração
  - Instalação
  - Manutenção de software

Fonte: Wikipedia

Tipos de Testes

- Unidade
- Integração
- Interface de Usuário
- Aceitação
- Mutação
- Desempenho
- Estresse
- Segurança

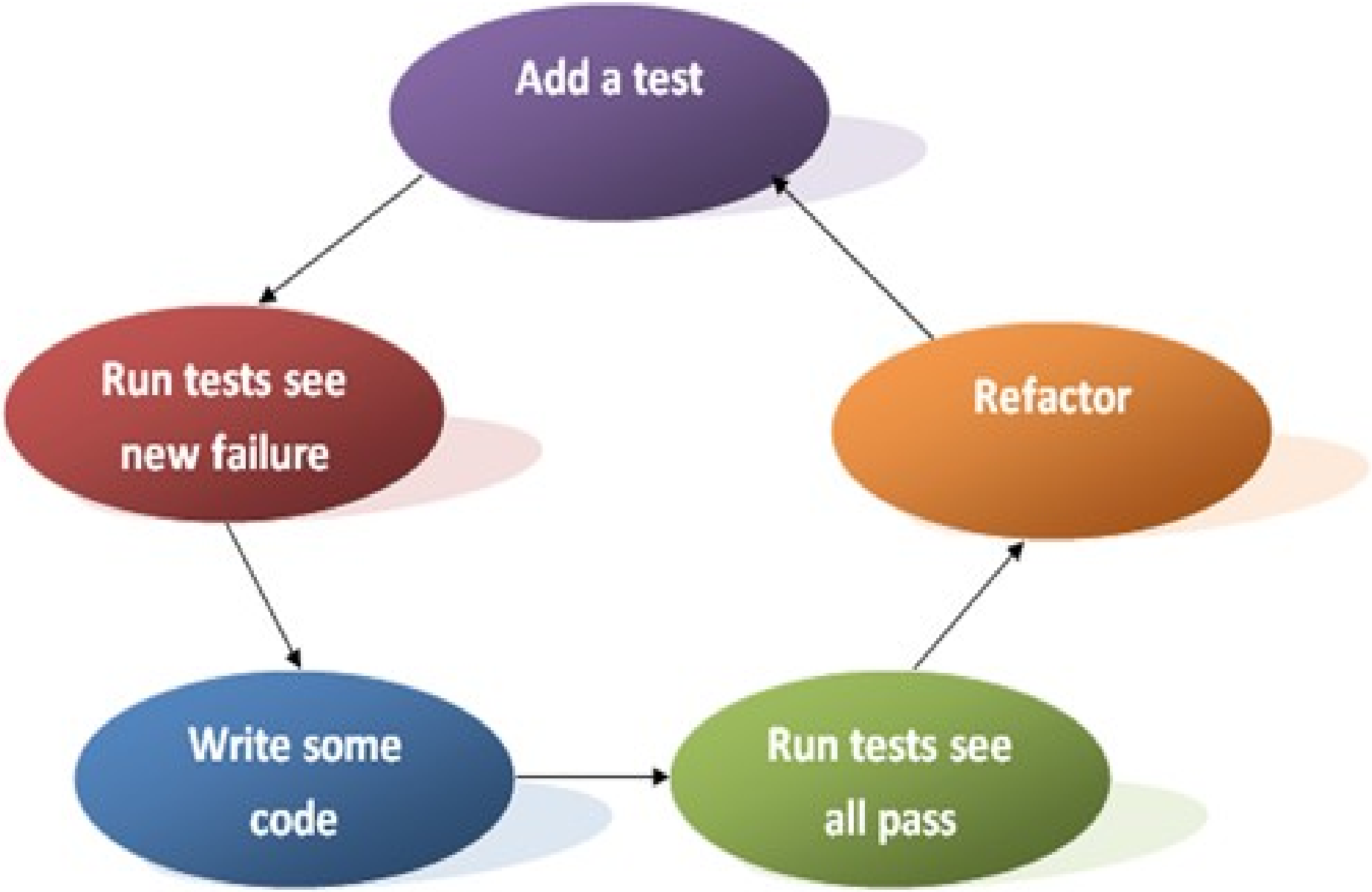
Testes Automatizados

- Teste de Software:
  - Executar o programa a ser testado com alguma entrada e conferir visualmente os resultados obtidos
- Teste Automatizado:
  - Programa ou script executável que roda o programa a ser testado e faz verificações automáticas em cima dos efeitos colaterais obtidos.
- Testar **NÃO** é depurar
  - Testar é verificar a presença de erros
  - Depurar é seguir o fluxo para identificar um erro conhecido

Testes de Unidade

- Unidade: módulo/classe ou função/método
- Teste básico e muito importante
  - Encontra muitos erros
- Não deve se preocupar com as responsabilidades de outras unidades
- Caixa-branca
  - Focando na funcionalidade

The TDD Process



- Livros:
  - Kent Beck, “Test-Driven Development: By Example”, Addison-Wesley Professional, 2002
  - Martin Fowler et al, “Refactoring: Improving the Design of Existing Code”, Addison-Wesley Professional, 1999
  - David Astels, “Test Driven Development: A Practical Guide”, Prentice Hall PTR, 2003
- Online:
  - www.testdriven.com
  - www.xprogramming.com

Exemplo

- Testes de Unidade
  - JUnit 4.0

JUnit 4.0

Sobre o JUnit

- Arcabouço para testes de unidades – pequenos testes para as várias partes (módulos, métodos, classes, etc) de um sistema
- Usado fortemente em XP, TDD e outros métodos ágeis
- Código-fonte livre
- Testes de unidades mostram a presença de erros, não a ausência deles

Fonte: [3]

Um exemplo

- Aplicação de brinquedo
- Contas bancárias
  - Conta Corrente
  - Conta Especial

O que queremos fazer

- Criar uma conta com um identificador e um depósito inicial
- Toda conta corrente possui um saldo
- Operações
  - Saques
    - Impedir saques de valores acima do saldo
      - Lançar exceção neste caso
  - Depósitos

Testes primeiro!!!

- Criação de conta corrente
  - ContaCorrente cc = new ContaCorrente(“20315-8”,1000.0);
- Teste para verificar se não é nulo:

```
@Test
  public void contaNaoNula(){
  assertNotNull(cc);
}
```

Como passar no primeiro teste

- Exemplo usando o ECLIPSE

Resumo do que foi utilizado

- @Test
  - Anotação colocada antes de cada método de teste
- @Before
  - Inicialização que ocorre antes de cada teste
- @After
  - Finalização após cada teste

Resumo do que foi utilizado

- @Test
- @Before
- @After
- @Test(expected=ClasseDeExcecao.class)
  - Captura de exceção
- assertXXX
  - Verificação de valores

Antes e Depois de Cada Teste

```
import org.junit.*;
import static org.junit.Assert.assertEquals;

public class NovelaNovoTest {
	private ArquivoDeNovelas novelas;

	@Before
	public void rodaAntesDeCadaTeste() {
		novelas = new ArquivoDeNovelas();
	}

	@After
	public void rodaDepoisDeCadaTeste() {
		novelas = null;
	}

	@Test
	public void testNovelaExiste() {
		assertEquals("Arquivo deve possuir Sassaricando", true,
				novelas.possuiNovela("Sassaricando"));
	}
}
```

Fonte: [3]

Antes e Depois de Cada Teste

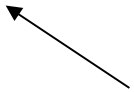
- É possível ter quantos métodos @Before e @After quiser
- Métodos @Before e @After são herdados das superclasses. Métodos @Before das superclasses são executados antes dos métodos @Before da subclasse. Métodos @After das superclasses são executados depois.
- Objetivo: agrupar código comum a vários testes

Fonte: [3]

Antes e Depois da Classe de Testes

```
@BeforeClass
public void rodaAntesDeTodosOsTestes() {
        novelas = new ArquivoDeNovelas();
        novelas.carregaDadosDaRede();
}

@AfterClass
public void rodaDepoisDeTodosOsTestes() {
        novelas = null;
}
```



Processamento pesado!

- Somente um método por classe
- Bom para inicializações demoradas

Fonte: [3]

Exceções Esperadas

```
@Test(expected=ArithmeticException.class)
  public void testDivisaoPorZero () {
    int n = 2 / 0;
}
```

- Se exceção não for gerada, teste falha

Fonte: [3]

Teste de tempo

```
@Test(timeout=500)
public void buscaTodasAsNovelas(){
     novelas.buscaTodas();
}
```

- Teste falha se demorar mais do que *timeout* milissegundos
- Útil para testes de redes e conexões com bancos de dados
- Detalhe: @After não é executado caso falhe!

Fonte: [3]

Teste de Tempo

- Aplicações modernas existem tempo de resposta máximo.
- O tempo de resposta pode mudar na medida em que o sistema evolui
- Testes de tempo garantem que o tempo de resposta máximo seja satisfeito, mesmo que o sistema se torne mais complexo
- O sistema cresce em funcionalidades sem perder performance!

Fonte: [3]

Outras Funcionalidades

- @Ignore – Para desabilitar temporariamente um teste
- assertEquals(Object[] expected, Object[] actual) – compara vetor elemento a elemento

Fonte: [3]

Resumo

- Precisa do JDK 5 para rodar
- @Test para anotar um teste
- @Before e @After para rodar antes e depois de cada teste.
- @Test recebe parâmetro de timeout
- @Test recebe parâmentro de excessão esperada

Outros Arcabouços Unit

- http://en.wikipedia.org/wiki/List_of_unit_testing_framewor
- SQLUnit – testes para Stored Procedures (http://sqlunit.sourceforge.net/)
- JPDFUnit – testa documentos PDF gerados dinamicamente (http://jpdfunit.sourceforge.net/)
- XMLUnit – teste para arquivos XML (http://xmlunit.sourceforge.net/)
- Watij – teste de aplicações Web em java (http://watij.xwiki.com/xwiki/bin/view/Main/WebHome)

Fonte: [3]

Próximos Passos

- Começar a usar o JUnit
- Ler mais sobre desenvolvimento dirigido por testes
- Aprender categorias e padrões de testes
- Convencer demais membros da empresa a fazerem testes primeiro

Referências Principais

[1] JUnit.org. Disponível em <http://junit.org>.Acesso em: 30 out. 2008.

[2] BECK, Kent e SAVOIA, Alberto. JUnit 4 and Java SE 5: Better Testing by Design. Disponível em: http://developers.sun.com/learning/javaoneonline/j1sessn . Acesso em: 20 out. 2008.

[3] CUKIER, Daniel. JUnit 4.0. Disponível em < http://gsd.ime.usp.br/seminars/2006/junit4.ppt>. Acesso em: 30 out. 2008.

Referências Adicionais

- BECK, Kent. **Simple Smalltalk Testing: With Patterns. http://www.xprogramming.com/testfram.htm**
- http://www-128.ibm.com/developerworks/java/libr
- http://www.instrumentalservices.com/media/articl
- http://junit.sourceforge.net/doc/cookbook/cookbook.htn
- http://j2meunit.sourceforge.net/doc.html
- http://en.wikipedia.org/wiki/Unit_test

Fonte: [3]

Refatoração

Refatoração

- Refatorar é melhorar o código sem alterar sua funcionalidade
- Antes de fazer uma mudança, você refatora o código para que a mudança seja simples de fazer
- Refatoração contínua possibilita manter um design legal, mesmo com mudanças freqüentes

O Livro de Refatoração

- “Refactoring: Improving the Design of Existing Code”, escrito por Martin Fowler, contém dezenas de “receitas de refatoração”
- Se você ainda não sabe, este livro vai te ensinar Orientação a Objetos
  - Por exemplo, teu código tem switches em atributos de tipo?

Código Não-OO

```
int JobSpec::requests(){
   if( js_type == jst_fixed ){
        return 1;
   } else if( js_type == jst_downey ){
        return js_options;
   } else if( js_type == jst_nas ){
        fatal( “not implemented yet" );
   } else {
        fatal1( "unknown js_type: %s", js_type );
   }
   return 0;          // this avoids a warning
```

Código OO

```
…
class DowneyJobSpec: public JobSpec {
   …
   int requests(){
       return js_options;
   };
   …
}
```

Ferramentas

- Java
  - Eclipse 3.2: Alt+Shift+T
    - Rename, Move, Change method signature, ...
    - 10 ou mais opções...
  - Netbeans 6.1: Menu Refatorar
    - Mais de 10 opções...

Metodologias Ágeis e o Mercado de Trabalho

Empresas Ágeis no Brasil

- **Região Centro-Oeste**
  - **Em Brasília-DF**
    - SEA
- **Região Nordeste**
  - **Em Fortaleza-CE**
    - Fortes Informática (com diversas filiais)
- **Região Sudeste**
  - **Em São Paulo-SP**
    - Objective Solutions
    - Paggo
    - Caelum (com filiais no Rio de Janeiro-RJ e em Porto Alegre-RS)
    - LocaWeb
    - Teamware (com filiais em Brasília-DF e na Argentina)
  - **No Rio de Janeiro-RJ**
    - ImproveIt

Casos de Sucesso no Brasil

- Assembléia Legislativa do Estado de São Paulo
- Ikwa
- Laboratório de Programação eXtrema
- ITM
- Paggo
- Parceria com Scopus e LARC
- Sistema Janus
- LocaWeb

Ikwa

- Canal de conteúdo, serviços e networking para auxiliar estudantes e profissionais em suas decisões e em seu crescimento acadêmico e profissional.
- **Primeira start-up Web 2.0 brasileira a receber investimento de venture capital.**
- Utiliza metodologia ágil de desenvolvimento de software.
- Em poucos meses, a equipe atingiu níveis elevados de flexibilidade e eficiência, conseguindo tornar o tempo de resposta a novas solicitações extremamente baixo.

Ikwa

- Novas funcionalidades passaram a entrar em produção semanalmente, sem perda de qualidade e com um layout que recebe inúmeros elogios de seus usuários.
- Site: www.ikwa.com.br
- Fonte: [1]

LocaWeb

- A LocaWeb é a **maior empresa de hospedagens de sites da América Latina.** No final de 2006, Daniel Cukier começou a usar **XP** na equipe de desenvolvimento de serviços de voz (LocaWeb Telecom). Usando a metodologia, a equipe conseguiu entregar funcionalidades de forma rápida e com baixo índice de bugs.

  A diretoria da empresa percebeu as vantagens dos métodos ágeis e decidiu realizar um treinamento envolvendo toda área de tecnologia. Em agosto de 2007, Daniel e Maurício Dediana realizaram um curso para 85 pessoas das áreas de desenvolvimento e produtos da empresa.

  A partir daí, as equipes começaram a adotar **Scrum** e algumas práticas de XP. Os diretores ficou muito satisfeitos com o resultado. Hoje a empresa é muito mais eficiente em entregar software de qualidade para seus clientes.

  Vale ressaltar foram utilizados padrões para introduzir novas idéias para disseminar a inovação dentro da empresa. Sem esses padrões, talvez não tivesse ido tão longe.
- Fonte: [1]

Paggo

- Em 2005 Alexandre Freire foi convidado para implantar a metodologia ágil de Programação eXtrema em uma start-up.
- Após 6 meses, o grupo de tecnologia da empresa conseguiu transitar para sua adaptação de XP com sucesso e conseguiu investimentos que garantiram a criação da Paggo.
- Hoje em dia a Paggo aplica práticas ágeis não só na tecnologia, mas em todos os departamentos da empresa.
- Fonte: [1]

Casos de Fracasso

- Relato de fracasso na adoção de XP em duas empresas: http://www.exoftware.com/i/white_paper/file
- Quem relata é o consultor da empresa irlandesa que tentou implantar XP
- Detalhe: as duas empresas faliram...

Empresas Globais

- http://objectmentor.com
- http://www.thoughtworks.com/
- http://www.rallydev.com/

Exemplos de Vagas “Ágeis”

- **Desenvolvedor Java** – Porto Alegre-RS
- Requisitos : Experiência mínima de 1 ano para Júnior e 2 anos para Pleno. Conhecimento em Frameworks (Struts, Axis), APIs (Webservices, JDBC, Portlets, EJB, JMS, XML, SAX), Padrões de Projetos e boas práticas de Java (MVC, Conceito de Camadas), J2EE, J2EE Patterns, Conhecimento de Redes, OO, RMI, metodologias ágeis, IDEs Eclipse e RAD, banco de dados Oracle, SQL ou PL/SQL, Sistemas de versionamento Subversion e CVS. Inglês Intermediário. Ensino Superior Completo ou em Andamento. Remuneração a combinar (CLT ou PJ)
- Fonte: http://www.empregonaweb.com/empresa/vaga.action?id=5272
- Data: 30/10/208

Empresas (com) simpatizantes na região de Curitiba

- Siemens ??
- Cinq Technologies ??

Sugestões

- Introduzir métodos ágeis na sua empresa (usando os padrões para introduzir novas idéias)
- Criar sua própria empresa de desenvolvimento de software usando como um dos diferenciais a utilização de métodos ágeis

Referências

[1] Casos de Sucesso. Agilcoop. http://www.agilcoop.org.br/portal/sucessos

[2] Empresas Ágeis no Brasil. Lista Agilcoop. http://www.agilcoop.org.br/portal/empresas

[3] Empresas Ágeis no Brasil. Lista completada por mim. http://adolfoneto.wikidot.com/empresas-ageis-no-

Divulgadores, Evangelistas, Praticantes de Metodologias Ágeis

Signatários do Manifesto Ágil

- Kent Beck (XP)
- Martin Fowler (Refactoring book, ThoughtWorks)
- Ron Jeffries
- Alistair Cockburn (Crystal)
- Ward Cunningham
- Jim Highsmith
- Ken Schwaber (Scrum)
- Jeff Sutherland
- Dave Thomas

Signatários do Manifesto Ágil

- Robert C. Martin
- Mike Beedle
- Arie van Bennekum
- James Grenning
- Andrew Hunt
- Jon Kern
- Brian Marick
- Steve Mellor

Outros

- Laurie Williams (Pair Programming)
- Scott Ambler (Agile Modeling)

No Brasil

- Fábio Kon
  , Alfredo Goldman e todo o pessoal da Agilcoop (IME-USP)
- Vinícius Manhães Teles (ImproveIt)

Organizações

- http://www.agilealliance.org/
- http://www.agilcoop.org.br/

Eventos sobre/relacionados a Metodologias Ágeis

XP Conference

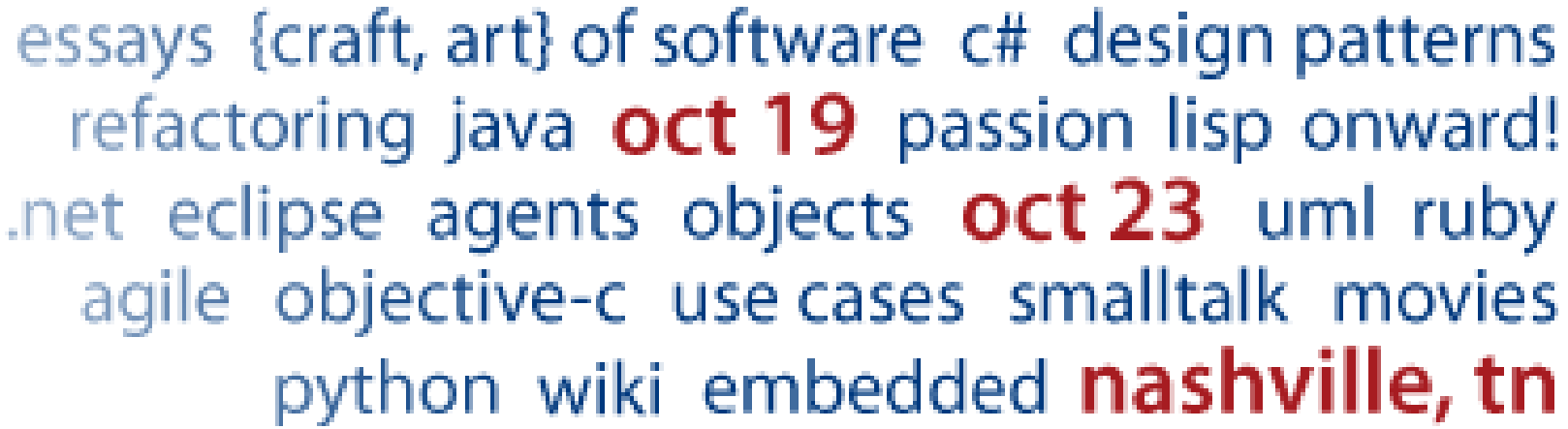
- *International Conference on Agile Processes in Software Engineering and eXtreme Programming*
- XP 2008: http://www.xp2008.org
- XP 2007: http://www.xp2007.org/
- Artigos:
  - XP 2008: http://www.informatik.uni-trier.de/~ley/db/conf/xpu/xp2(
  - Todas as edições: http://www.informatik.uni-trier.de/~ley/db/conf/xpu/inde

Agile Conference

- http://www.**agile2008.org**

OOPSLA

- http://www.oopsla.org/oopsla2008/



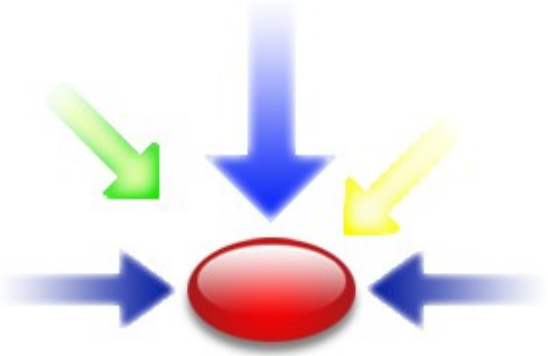
QCon

- http://qconsf.com/

EVENTOS NO BRASIL

Encontro Ágil

- http://www.encontroagil.com.br/
- São Paulo-SP, 11/10/2008



Falando em Agile

- http://www.caelum.com.br/falando-em-agile
- 23 e 24/10/2008 – São Paulo-SP
- Patrocínio: globo.com, Borland, Yahoo! Brasil

Agile Day

- http://professoradolfo.blogspot.com/2008/1
- Porto Alegre-RS, 15/10/2008

Eventos Acadêmicos

- ESELAW 2008
  - http://web.unifacs.br/eselaw/
  - 5 a 7 de novembro de 2008 – Salvador-BA
  - This edition of ESELAW also emphasizes three perspectives: Open Source, **Agile Methods** and Experimental Software Engineering.

Bibliografia Básica sobre Metodologias Ágeis

Adolfo Neto (DAINF – UTFPR)
http://www.dainf.ct.utfpr.edu.br/~adolfo

Extreme Programming

- TELES, Vinícius M. Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. Novatec, 2004.
- BECK, Kent. Programação Extrema Explicada – A colha as Mudanças. Addison-Wesley, 1999
- BECK, Kent & ANDRES, Cynthia. Extreme Programming Explained 2nd Edition, Addison-Wesley, 2004

Métodos Ágeis em Geral

- Craig Larman, Agile and Iterative Development: A Manager's Guide, Addison-Wesley, 2003
- Alistair Cockburn, Agile Software Development, Addison-Wesley, 2001
- Alistair Cockburn, Agile Software Development: The Cooperative Game (2nd Edition), Addison-Wesley, 2006
- Jim Highsmith, Agile Software Development Ecosystems, Addison-Wesley, 2002
- Robert C. Martin, Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002

Testes

- BECK, Kent. ***Test Driven Development: By Example.***

Refatoração

- Martin Fowler. Refactoring: improving the design of existing code.

Scrum

- **Scrum and XP from the Trenches, Henrik Kniberg**. http://www.infoq.com/minibooks/scrum-xp-f
- http://www.controlchaos.com/

Banco de Dados Ágeis

- Scott Ambler e Pramod Sadalage, "Refactoring Databases: Evolutionary Database Design", Addison-Wesley Professional, 2006
- Scott Ambler, "Agile Database Techniques", Wiley Publishing, 2003

Lean

- Mary e Tom Poppendieck, "Lean Software Development: An Agile Toolkit", Addison-Wesley, 2003
- Mary e Tom Poppendieck, "Implementing Lean Software Development: From Concept to Cash",Addison-Wesley, 2006

Livros não-específicos

- PILONE, DAN e MILES, RUSSELL. USE A CABEÇA DESENVOLVIMENTO DE SOFTWARE.

Gestão Participativa (Ricardo Semler)

- SEMLER, Ricardo. *Virando a própria mesa*. São Paulo : Best Seller, 1988.
- SEMLER, Ricardo. *The Seven Day Weekend: Changing the Way Work Works*. 2003
- SEMLER, Ricardo. *Você está louco!: uma Vida Administrada de outra forma*. 2006

Ler mais em http://pt.wikipedia.org/wiki/Ricardo_Semler

Sítios e Blogs

AgilCoop

- http://agilcoop.org.br/
- Agilvídeo
- Agilcast (podcast)
- Slides
- Artigos, dissertações, etc.
- Encontro Ágil
- E muito mais....

O Manifesto Ágil

- http://agilemanifesto.org/

Sites

- Martin Fowler. http://www.martinfowler.com/
- Ron Jeffries: http://www.xprogramming.com/index.htm

Blogs

- Agilblog da Locaweb
- Blog da Fratech
- Revista Visão Ágil: http://visaoagil.wordpress.com
- Blog de James Shore
- Gerenciamento Ágil de Projetos

Páginas

- Links sobre Métodos Ágeis na página de R

- Slides de Roger Pressman sobre Metodologias Ágeis (2005)
- Eu uso metodologia ágil, e você?
- Como a Pixar promove a criatividade coletiva

Coding Dojos

- São Paulo: http://groups.google.com/group/dojo_sp/?p
- Floripa: http://dojofloripa.wordpress.com/

Evidence-Based Software Engineering

- http://www.dur.ac.uk/ebse/
  - Primeira resposta ao termo no Cuil e no Google

Conclusões

- Métodos Ágeis chegaram para ficar
- São um meio termo entre o caos total e o "excesso de burocracia"
- Várias práticas podem ser usadas inclusive com outros métodos (refatoração, desenv. com testes a priori, etc.)

Proposta de Rob Mee e Kent Beck para a Indústria de Software no Brasil

- NÃO seguir o modelo da Índia (baseado em CMMi nível 5 – graças a muita mão-de-obra barata)
- Apostar em XP e
  - Aumentar a produtividade de software para o mercado interno
  - Com a folga, satisfazer o mercado externo

Próximos passos

- Ler A Nova Metodologia, de Martin Fowler
- Conversar com seus superiores/colaboradores hierarquicamente sobre a possibilidade de adoção de metodologias ágeis
- Assistir o vídeo-palestra sobre extreme programming Manhães Teles
- Ler “Good Agile, Bad Agile”, de Steve Yegge

Adicional

- Desenvolvimento de Software Orientado a Aspectos
  - AspectJ: http://eclipse.org/aspectj