MÉTODOS ÁGEIS E GERÊNCIA DE PROJETOS

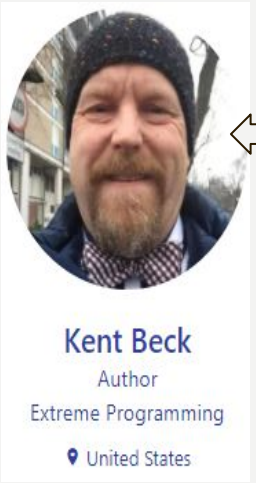
Prof. Radamés Pereira

2023/2

ENGENHARIA DE SOFTWARE II
MÉTODOS ÁGEIS E GERÊNCIA DE PROJETOS

In software development, perfect is a verb, not an adjective. There is no perfect process. There is no perfect design. There are no perfect stories. You can, however, perfect your process, your design, and your stories. – Kent Beck

Em desenvolvimento de software, perfeito é um verbo, não um adjetivo. Não existe um processo perfeito. Não existe um design perfeito. Não existem histórias perfeitas. Você pode, no entanto, aperfeiçoar seu processo, seu design e suas histórias.



Kent Beck é um engenheiro de software americano criador do Extreme Programming e Test Driven Development. Beck foi um dos 17 signatários originais do Agile Manifesto em 2001.

Importância dos Processos

A produção de um carro em uma fábrica de automóveis segue um processo bem definido. Primeiro, as chapas de aço são cortadas e prensadas, para ganhar a forma de portas, tetos e capôs. Depois, o carro é pintado e instalam-se painel, bancos, cintos de segurança e toda a fiação. Por fim, instala-se a parte mecânica, incluindo motor, suspensão e freios.

Importância dos Processos

Assim como carros, software também é produzido de acordo com um processo, embora certamente menos mecânico e mais dependente de esforço intelectual. Um processo de desenvolvimento de software define um conjunto de passos, tarefas, eventos e práticas que devem ser seguidos por desenvolvedores de software, na produção de um sistema.

Importância dos Processos

As equipes, para produzirem software com qualidade e produtividade, precisam de um ordenamento, mesmo que mínimo. Por isso, empresas dão tanto valor a processos de software. Eles são o instrumento de que as empresas dispõem para coordenar, motivar, organizar e avaliar o trabalho de seus desenvolvedores, de forma a garantir que eles trabalhem com produtividade e produzam sistemas alinhados com os objetivos da organização.

Importância dos Processos

Sem um processo — mesmo que simplificado e leve, existe o risco de que os times de desenvolvimento passem a trabalhar de forma descoordenada, gerando produtos sem valor para o negócio da empresa.

Processos são importantes não apenas para a empresa, mas também para os desenvolvedores, pois permitem que eles tomem consciência das tarefas e resultados que se esperam deles. Sem um processo, os desenvolvedores podem se sentir perdidos, trabalhando de forma errática e sem alinhamento com os demais membros do time de desenvolvimento.

Processo

Processo é o conjunto de passos, etapas e tarefas que se usa para construir um software. Toda organização usa um processo para desenvolver seus sistemas, o qual pode ser ágil ou waterfall, por exemplo. Ou, talvez, esse processo pode ser caótico. Porém, sempre existe um processo.

Método

Método define e especifica um determinado processo de desenvolvimento (a palavra método tem sua origem no grego, onde significa “caminho para se chegar a um objetivo”).

Exemplo: XP, Scrum e Kanban.

Métodos ágeis definem práticas, atividades, eventos e técnicas compatíveis com princípios ágeis de desenvolvimento de software. Também se usa o termo metodologia quando se fala de processos de software.

Exemplo: metodologias ágeis, metodologia orientada a objetos, etc.

A palavra metodologia, denota o “ramo da lógica que se ocupa dos métodos das diferentes ciências”, segundo o Dicionário Houaiss. A palavra também pode ser usada como sinônimo de método, segundo o mesmo dicionário.

Método

Todo método de desenvolvimento deve ser entendido como um conjunto de recomendações; cabe a uma organização analisar cada uma e decidir se ela faz sentido no seu contexto.

Como resultado, a organização pode ainda decidir por adaptar essas recomendações para atender às suas necessidades.

Logo, provavelmente, não existem duas organizações que seguem exatamente o mesmo processo de desenvolvimento, mesmo que elas digam que estão desenvolvendo usando Scrum, por exemplo.

Método

Em 2018, o Stack Overflow survey pesquisou sobre o método de desenvolvimento mais usado:

- Recebeu 57 mil respostas de desenvolvedores profissionais;
- Scrum (63% das respostas);
- Kanban (36%);
- Extreme Programming (16%);
- 15% marcaram Waterfall.

Metodologias ágeis

São conjuntos de práticas que proporcionam uma forma de gerenciar projetos mais adaptável às mudanças. Elas são estruturadas em ciclos curtos sendo que, a cada novo ciclo, é entregue um conjunto de funcionalidades pré-determinado.

Histórico dos Métodos Ágeis

A entrega de software em prazos e custos estabelecidos nem sempre é conseguida.

Formalidade nos modelos de processo propostos nos últimos 30 anos.

Métodos Ágeis:
• Propõem desenvolvimento de software de forma mais rápida, com um grande número de ciclos, mas com qualidade.

Histórico dos Métodos Ágeis

As definições modernas de desenvolvimento de software ágil evoluíram a partir da metade de 1990 como parte de uma reação contra métodos "pesados"

- Regulamentação
- Regimentação
- método cascata
- burocracia

Manifesto Ágil

O Manifesto Ágil, criado em 2001, descreve a essência de um conjunto de abordagens para desenvolvimento de software criadas ao longo da última década.

- Indivíduos e interação entre eles 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.

Manifesto Ágil

- **Satisfação do cliente** através de entregas mais cedo e contínuas, utilizando ciclos de iteração menores
  - Aceitação e acomodação de requisitos em qualquer tempo do desenvolvimento
  - Desenvolvedores e usuários trabalhando juntos
  - Times motivados e em ambientes apropriados

Manifesto Ágil

(Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.)

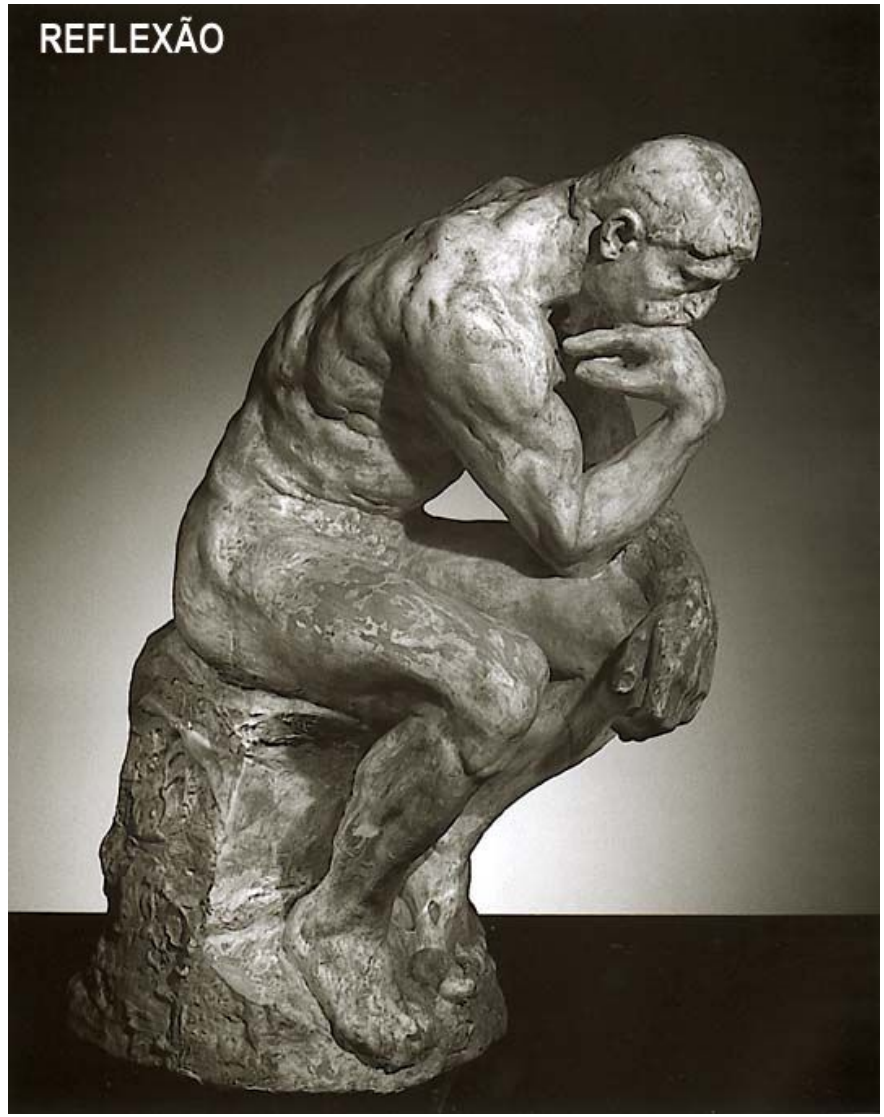
Manifesto Ágil

- **Minimização** de documentação e maximização de troca de informação face2face

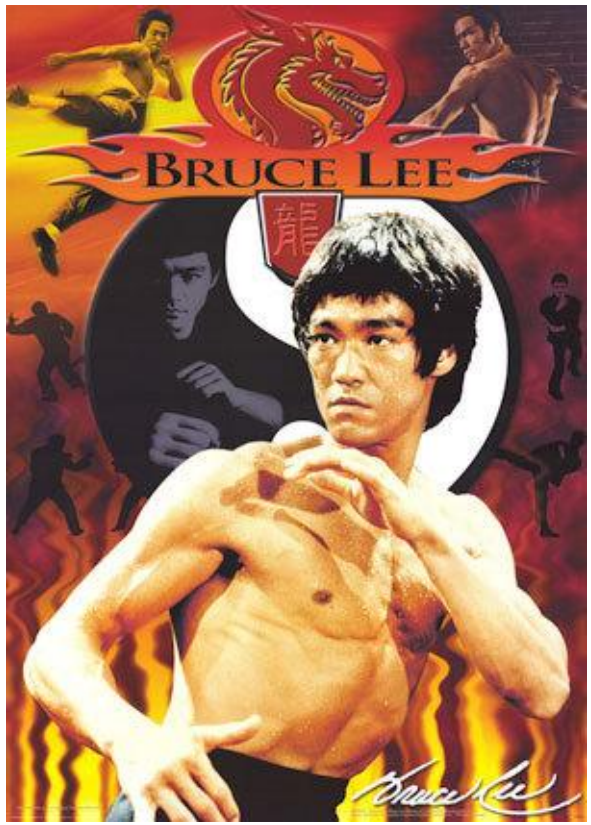
- **Encorajamento de atitudes reflexivas e contínuo aprendizado**

- **O principal recurso de uma empresa de tecnologia não são seus bens físicos: computadores, prédios, móveis ou conexões de Internet, por exemplo — mas sim seus colaboradores.**

Manifesto Ágil



Métodos Lightweight X Métodos Heavyweight



http://br.youtube.com/watch?v=UZq4sZz56qM&NR=1

Métodos Lightweight X Métodos Heavyweight

Sobre os métodos ***heavyweight***

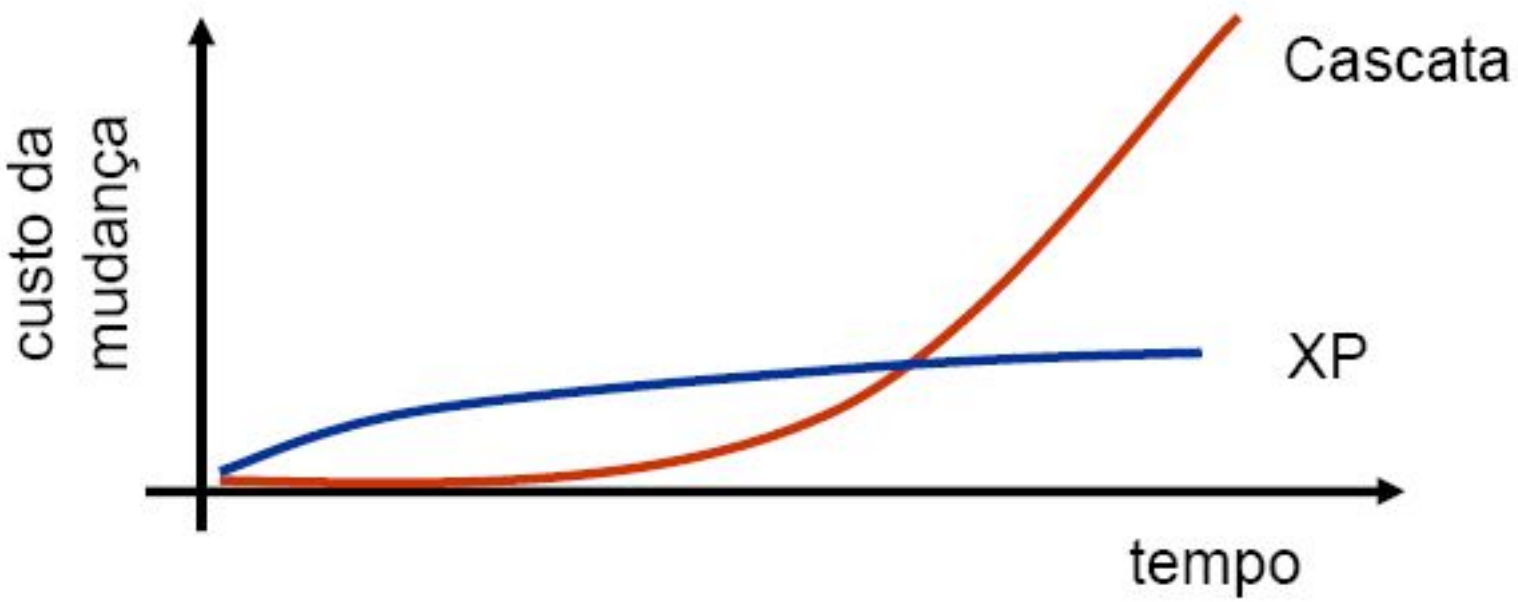
- são os métodos tradicionais
- Apregoam muito planejamento antecipado
- São baseados na produção de uma grande quantidade de documentação
- Resistentes a mudanças
- São considerados métodos pesados

Métodos Lightweight X Métodos Heavyweight

Sobre os métodos ***lightweight***

- no contraponto, são os métodos considerados métodos leves, ágeis
- nem todas as práticas relacionadas aos métodos ágeis são novas
- Devemos abraçar as mudanças
- Planejar o tempo todo

Métodos Lightweight X Métodos Heavyweight



Métodos Lightweight X Métodos Heavyweight

Características

Modelos Tradicionais

- Previsibilidade;
- Controlar mudanças;
- Burocráticos;
- Excesso de documentação;
- Enfatizam os aspectos de Engenharia do desenvolvimento.

Métodos Ágeis

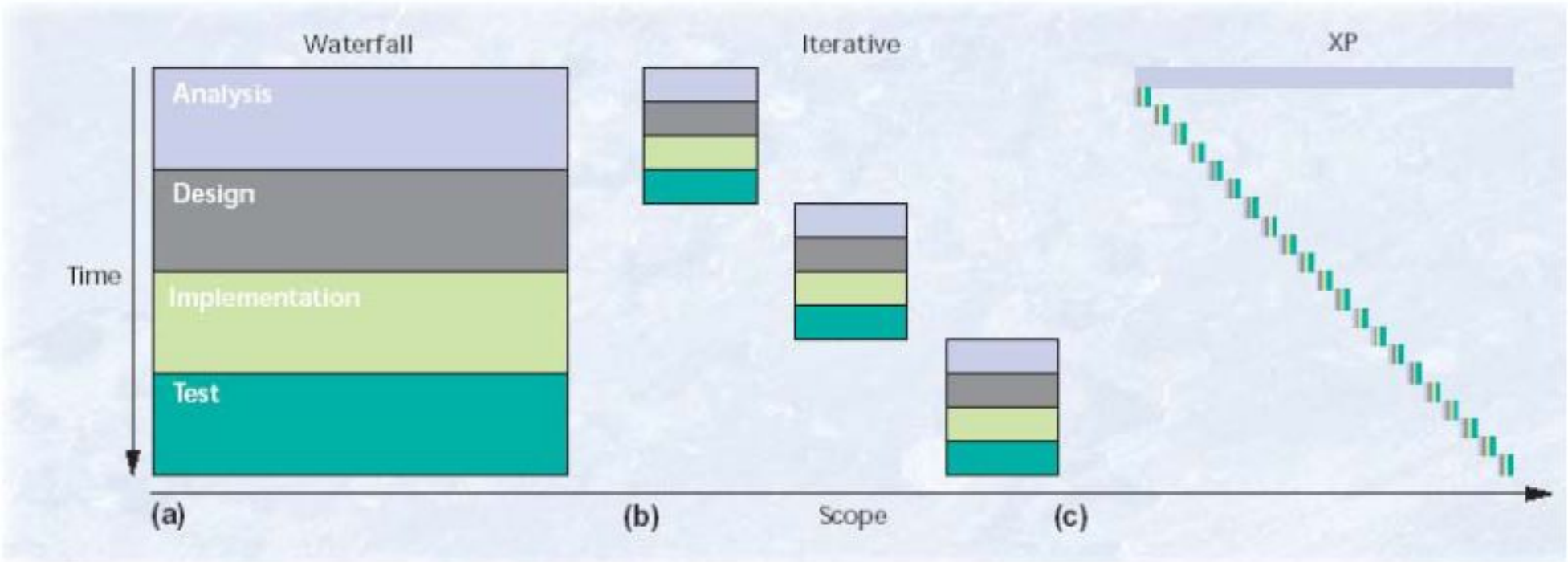
- Adaptabilidade;
- Planejamento é contínuo;
- Documentação essencial;
- Mudanças rápidas;
- Enfatizam os aspectos humanos do desenvolvimento.

Métodos Lightweight X Métodos Heavyweight

- (a) Cascata: Longo ciclo de desenvolvimento
- (b) Espiral: Iterações
- (c) XP: Iterações curtas

Métodos Lightweight X Métodos Heavyweight

Diferenças



Fonte: [Beck 1999b]

Métodos Lightweight X Métodos Heavyweight

Modelos Tradicionais:

- Modelo Cascata;
- Modelo de Prototipação;
- Modelo Espiral (iterativo e incremental);
- Modelo Concorrente.

Métodos Ágeis:

- Extreme Programming (XP);
- Scrum;
- Crystal;
- Agile Modeling (AM).

Extreme Programming

XP é um método leve recomendado para desenvolver software com requisitos vagos ou sujeitos a mudanças; isto é, basicamente sistemas comerciais, na classificação (Kent Back).

Iterações curtas:

• Duram de duas a quatro semanas.

Poucos artefatos exigidos:

• Artefatos devem ser simples e de valor.

• Código é o principal artefato.

• Existem outros: User Stories, Testes de Unidade, Testes de Aceitação, Estimativas (Stories e Tarefas), entre outros.

Extreme Programming, paradigmas

Menor quantidade de atividades no processo:

- Quatro atividades básicas: Codificação (Coding), Teste (Testing),Escuta (Listening) e Projeto (Designing).

- Atividades estruturadas de acordo com as práticas.

Poucos papéis

- Sete papéis: Programador (Programmer), Cliente (Customer), Testador (Tester), Investigador (Tracker), Orientador (Coach), Consultor (Consultant) e Gerente (Manager).

Extreme Programming, valores

- **Valores:** comunicação, simplicidade, feedback, coragem, respeito e qualidade de vida. São valores universais, para convívio humano. Servem para guiar projetos de desenvolvimento, e também a própria vida em sociedade.

- **Comunicação** é importante em qualquer projeto.

- A **simplicidade**, existem sistemas ou subsistemas mais simples, que às vezes não são considerados. Há riscos: os requisitos mudam, a tecnologia muda, a equipe de desenvolvimento muda, o mundo muda, etc. Controlar tais riscos é estar aberto ao **feedback** dos stakeholders. É difícil desenvolver o sistema de software "certo" em uma primeira e única tentativa. Frederick Brooks tem uma frase conhecida: "Planeje-se para jogar fora partes de seu sistema, pois você fará isso."

- **Feedback** é um valor essencial para garantir que as partes ou versões que serão descartadas sejam identificadas o quanto antes, de forma a diminuir prejuízos e retrabalho.

- XP também defende outros valores, como **coragem**, **respeito** e **qualidade de vida**, valores abstratos e universais.

Extreme Programming, princípios

Princípios: humanidade, economicidade, benefícios mútuos, melhorias contínuas, falhas acontecem, baby steps e responsabilidade pessoal.

Os princípios ligam os valores às práticas.

Extreme Programming, práticas

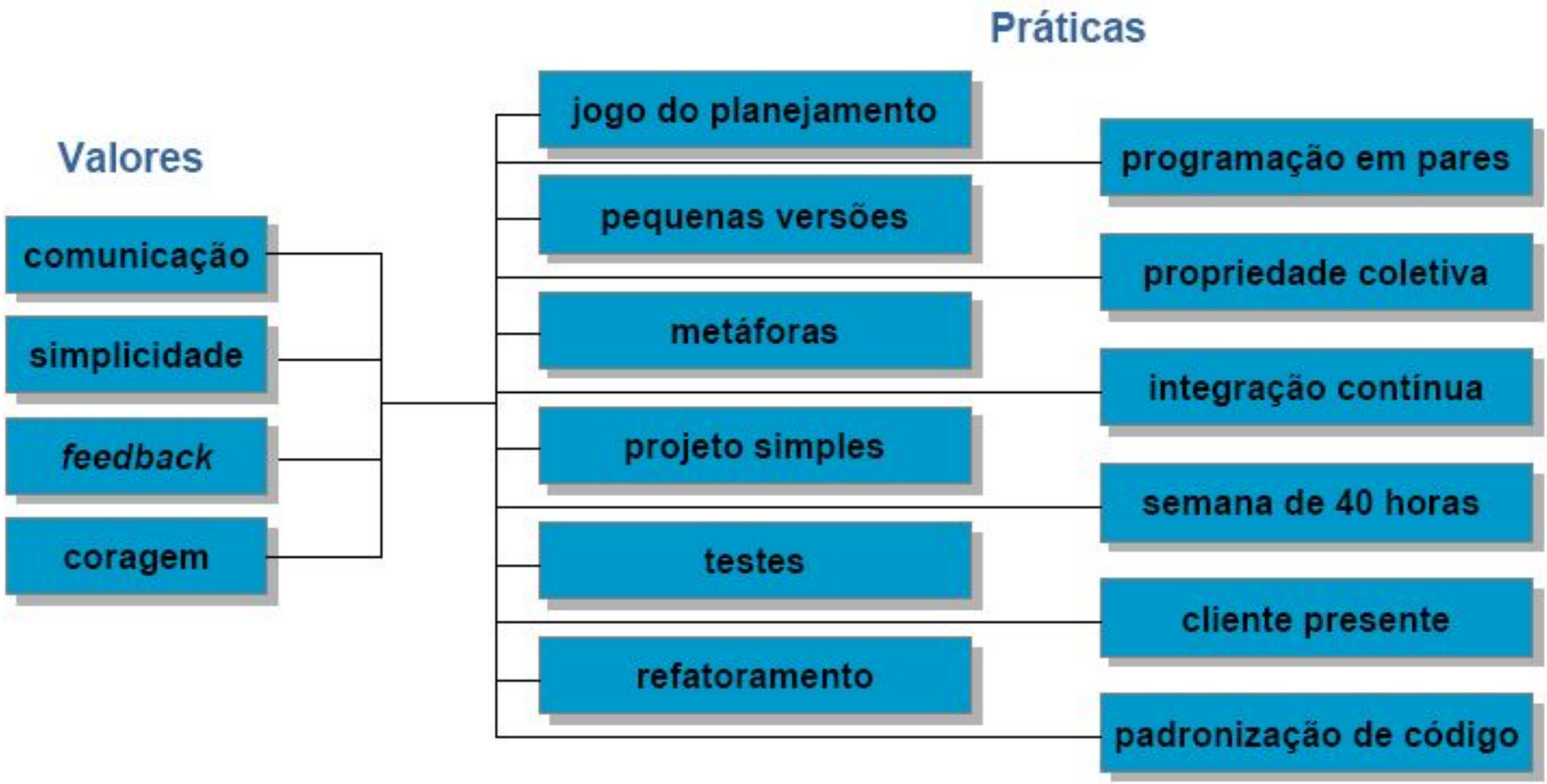
Práticas do Processo de Desenvolvimento: representante dos clientes, histórias dos usuários, iterações, releases, planejamento de releases, planejamento de iterações, planning poker, slack.

Práticas de Programação: design incremental, programação pareada, desenvolvimento dirigido por testes (TDD), build automatizado, integração contínua.

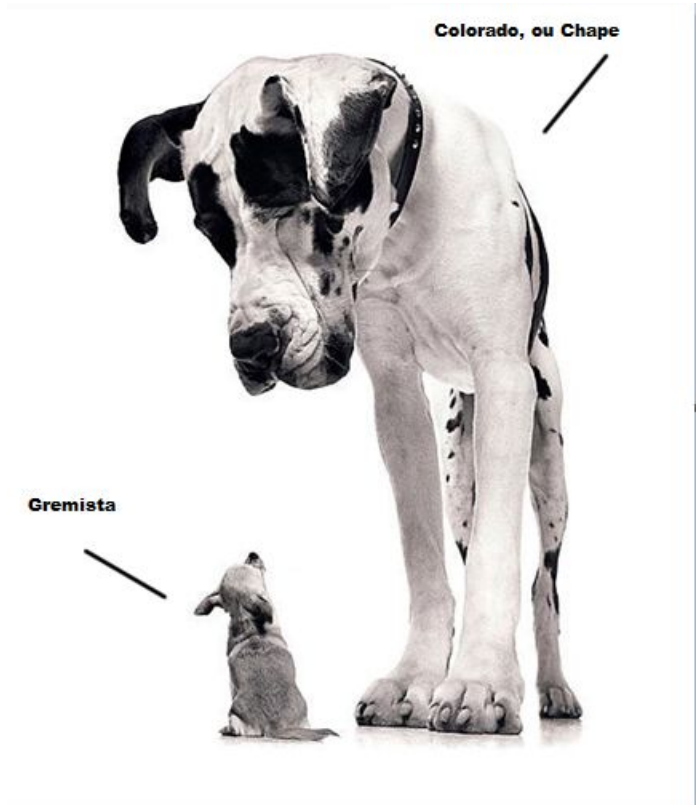
Práticas de Gerenciamento de Projetos: métricas, ambiente de trabalho, contratos com escopo aberto.

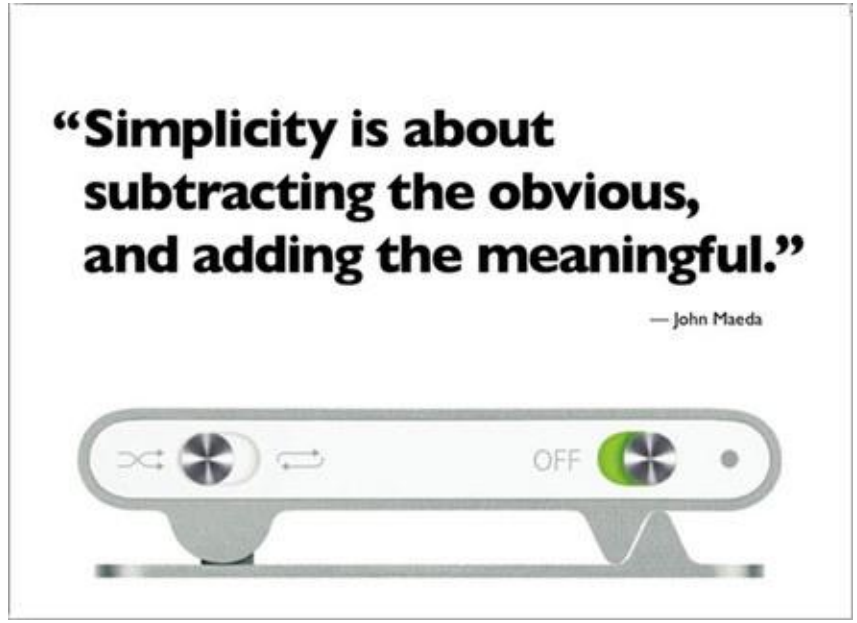
Extreme Programming, valores e práticas

- Equipes pequenas e médias (dois a dez membros);
- Reúne práticas de implementação em um conjunto coerente, acrescentando idéias de processo.



Extreme Programming





Propõe que o design de um sistema também seja definido de forma incremental e sugere que as equipes de desenvolvimento sejam pequenas.

Extreme Programming

Comunicação é fundamental!!!!!!

Extreme Programming, práticas

Jogo do Planejamento (The Planning Game)

- Determina rapidamente o escopo das próximas versões, combinando as prioridades de negócio e as estimativas técnicas.

Pequenas Versões (Small releases)

- A equipe deve colocar rapidamente um sistema simples em produção, uma versão pequena, e depois entregar novas versões em poucos dias ou semanas.

Poker Planning

- Como obter estimativas de forma consensual

Extreme Programming, práticas

Metáfora (Metaphor)

- Uma metáfora é uma descrição simples de como o sistema funciona. Ela fornece uma visão comum do sistema e guia o seu desenvolvimento.

Projeto simples (Simple design)

- O sistema deve ser projetado o mais simples possível. Complexidade extra é removida assim que descoberta.

Extreme Programming, práticas

Testes (Testing)

- Os programadores escrevem testes de unidade continuamente. Esses testes são criados antes do código e devem ser executados perfeitamente para que o desenvolvimento continue. Os clientes também escrevem testes para validar se as funções estão finalizadas.

Refatoração (Refactoring)

- Os programadores reestruturam o sistema durante todo o desenvolvimento, sem modificar seu comportamento externo. Isso é feito para simplificar o sistema, adicionar flexibilidade ou melhorar o código.

Extreme Programming, práticas

Programação em pares (Pair programming)

- Todo código produzido é feito em pares, duas pessoas trabalhando em conjunto na mesma máquina.

Propriedade coletiva (Collective ownership)

- Qualquer um pode alterar qualquer código em qualquer momento, o código é de propriedade coletiva.

Extreme Programming, práticas

Integração contínua (Continuous integration)

- Uma nova parte do código deve ser integrada assim que estiver pronta. Consequentemente, o sistema é integrado e construído várias vezes ao dia.

Semana de 40 horas (40-hour week)

- XP defende um ritmo de trabalho que possa ser mantido, sem prejudicar o bem estar da equipe. Trabalho além do horário normal pode ser necessário, mas fazer horas extras por períodos maiores que uma semana é sinal de que algo

está errado com o projeto.

Extreme Programming, práticas

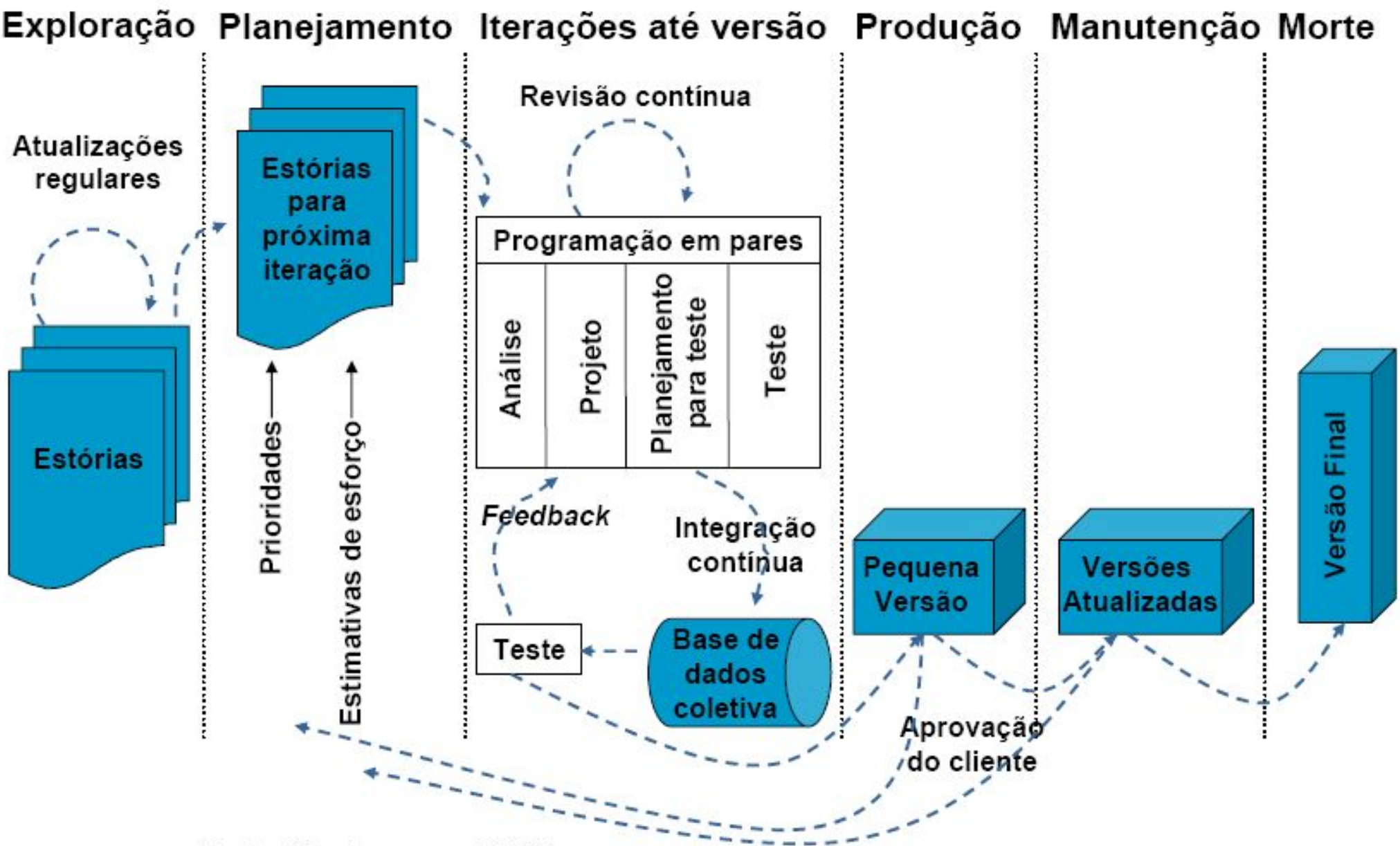
Cliente junto aos desenvolvedores (On-site customer)

- Os desenvolvedores devem ter o cliente disponível todo o tempo, para que ele possa responder às dúvidas que os desenvolvedores possam ter.

Padronização do Código (Coding standards)

- Os programadores escrevem o código seguindo regras comuns enfatizando a comunicação por meio do código.

Extreme Programming, gráfico de processo



Fonte: [Abrahamsson 2002]

Extreme Programming - Considerações

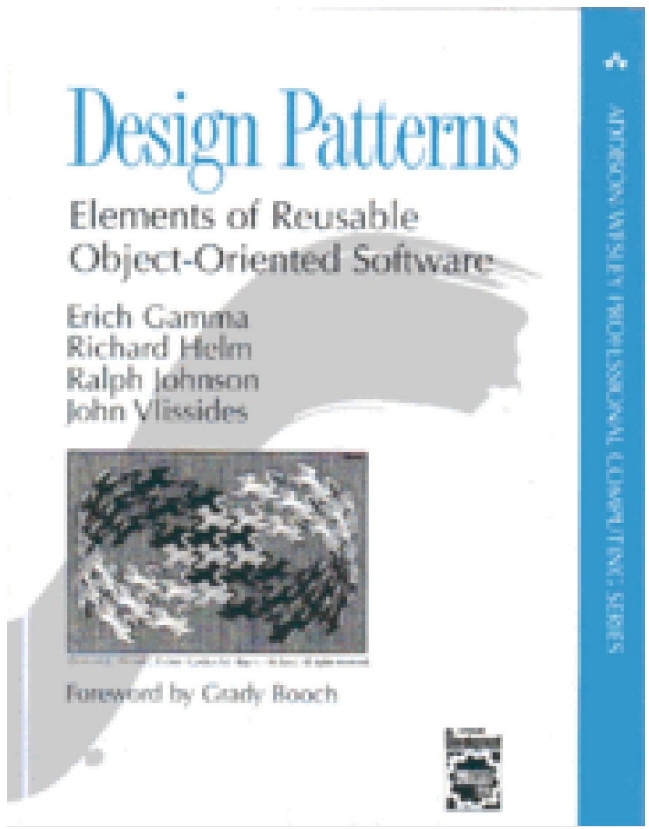
- XP não é um método detalhado que define um passo a passo para construção de software. Em vez disso, XP é definido por meio de um conjunto de valores, princípios e práticas de desenvolvimento.
- XP é inicialmente definido de forma abstrata, usando-se de valores e princípios que devem fazer parte da cultura e dos hábitos de times de desenvolvimento de software. Depois, esses valores e princípios são concretizados em uma lista de práticas de desenvolvimento.
- Frequentemente, quando decidem adotar XP, desenvolvedores e organizações concentram-se nas práticas. Porém, os valores e princípios são componentes chaves do método, pois são eles que dão sentido às práticas propostas em XP.
- Se uma organização não está preparada para trabalhar no modelo mental de XP — representado pelos seus valores e princípios — recomenda-se também não adotar suas práticas.

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.

Origens do Scrum

Desenvolvimento de software
a partir de padrões de projeto
(design patterns)



Mas, o que é isto ???

O que são padrões?

- No final dos anos 70, o arquiteto Christopher Alexander escreveu dois livros com a ideia.
- Cada padrão descreve um problema recorrente no nosso ambiente e, em seguida, o princípio de sua solução.
- A solução pode ser aplicada diversas vezes, nunca da mesma maneira.
- Um exemplo: escritório com janela.

Fundamentos do Scrum

Desenvolvimento de software depende muito de criatividade e de trabalho

- Logo, não é um bom candidato a processos pré-definidos
- Modelo de controle de processo empírico.
- O desenvolvimento nem sempre será repetitivo e bem definido
- Mas existem padrões que podem ser usados

História (processo)

Processo definido

Funciona em ambientes controlados

Processo empírico

- Processos de controle industrial

Funciona para processos

- Complexos e Imprevisíveis

Princípios

- Visibilidade, Inspeção e Adaptabilidade

Ênfases

Comunicação

Trabalho em equipe

Flexibilidade

Fornecer software funcionando

- incrementalmente

Princípios padrões

- **Backlog**
- **Equipes**
- **Sprints**
- **Encontros Scrum**
- **Revisões Scrum/Demos**

Desvantagens dos Métodos Ágeis

-Podem ser inadequados para projetos de grande complexidade;

-Dificuldade na aplicação em grandes equipes e empresas em que os funcionários estejam separados geograficamente;

-Ausência de estratégia de gestão de riscos;

-Falta de casos de sucesso de uso em projetos grandes e críticos;

Pesquisa sobre uso de métodos ágeis

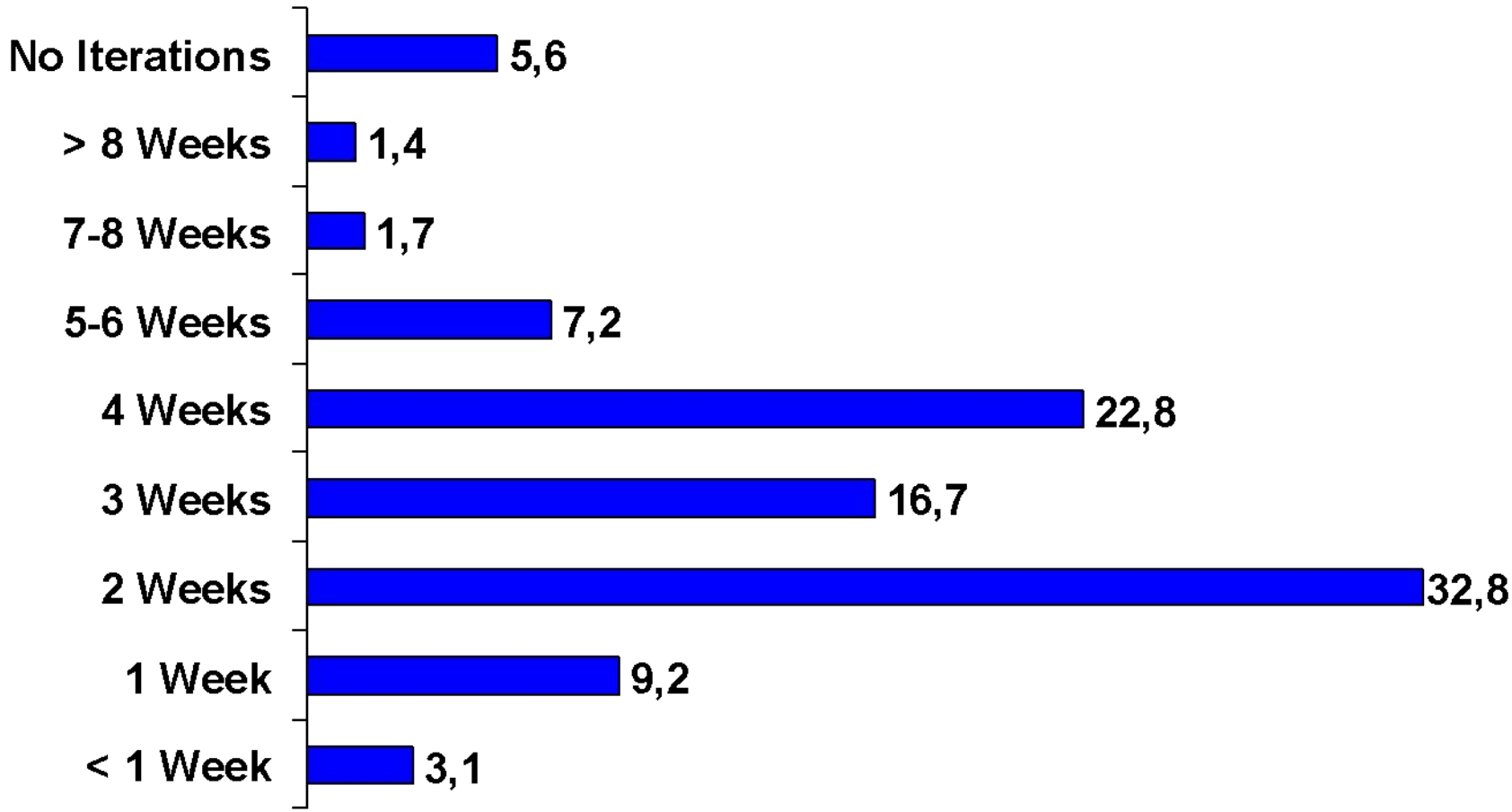
Dados sobre o uso dos métodos ágeis obtidos com:

- 65% desenvolvedores
- 37% tinham mais de 20 anos de experiência em TI
- 38% trabalharam em organizações com mais de 1000 funcionários
- 70% com origem nos EUA, 17% na Europa

Fonte da pesquisa

- http://agiledata.org/

Duração das iterações



Questões interessantes

1- Métodos ágeis servem apenas para grupos pequenos

2- Métodos ágeis não são apropriados para ambientes CMMI

3- Métodos ágeis funcionam apenas para equipes que trabalham no mesmo local

	Concordo plenamente	Concordo	Neutro	Discordo	Discordo completamente	Sem opinião
1	6.4%	30.6%	22.9%	24.8%	1.9%	13.4%
2	5.1%	10.8%	29.3%	14,00%	3.2%	37.6%
3	4.5%	19.1%	33.1%	21,00%	6.4%	15.9%

MÉTODOS ÁGEIS - MITOS

1. **É coisa de TI:**

   MITO! Embora as metodologias ágeis tenham sido propagadas a partir do desenvolvimento de software e produtos, elas não são exclusivas dessa área. Todo projeto, seja ele de uma área da empresa ou da sua vida pessoal, pode se ser realizado com mais qualidade e de maneira mais rápida quando feito a partir de metodologias ágeis. Ex. dividir as tarefas em partes.

2. **Scrum vai resolver todos os problemas do planeta:**

   MITO! Se ele fosse capaz de resolver todos os problemas, seria vendido por um preço altíssimo! Embora Scrum ajude muito, ele ainda não consegue resolver todos os problemas do mundo. Tenha o Scrum como um aliado para resolver seus projetos, descobrindo erros mais cedo e otimizando o tempo de desenvolvimento deles, porém saiba que eles ainda dependem de você e da sua equipe para serem finalizados.

3. **Os métodos ágeis têm pouco planejamento:**

   MITO! Muito pelo contrário. Planejamento reduz o tempo para um trabalho muito mais eficaz. O planejamento é feito em comum acordo com os membros da equipe, e com o cliente. O planejamento é capaz de provocar uma mudança no mindset, pois geralmente é a partir de um estudo abrangente que poderá ser gerado. Um mindset ágil é o conjunto de atitudes que sustentam um ambiente de trabalho ágil. Estas incluem respeito, colaboração, melhoria e ciclos de aprendizagem, orgulho na propriedade, foco na entrega de valor e capacidade de adaptação à mudança.

Métodos Ágeis e você!

Na análise dos métodos ágeis, ficou claro que algumas características colaboram nos hábitos das pessoas eficazes.

Proatividade

Proatividade é a capacidade (ou o hábito) de se antecipar aos acontecimentos. É a atitude daquele que não espera chover para consertar o telhado e não tem medo de mexer no time que está ganhando.

A pessoa proativa toma a iniciativa de realizar alguma tarefa antes que seja cobrada por isso ou que outro perceba a necessidade. Ela está sempre analisando o seu ambiente e vendo o que pode ser feito para melhorar. É um conceito intimamente relacionado à iniciativa e contínuo aperfeiçoamento.

Proatividade

Proatividade, linguagem

Linguagem Reativa	Linguagem Proativa
-Não há nada que eu possa fazer	-Vamos procurar alternativas
-Sou assim e pronto	-Posso tomar outra atitude
-Ela me deixa louco	-Posso controlar meus sentimentos
-Eles nunca vão aceitar isso	-Vou buscar uma apresentação eficaz
-Tenho de fazer isso	-Preciso achar uma resposta apropriada
-Não posso	-Eu escolho
-Eu preciso	-Eu prefiro
-Ah se eu pudesse...	-Eu vou fazer

Procure compreender

Ouça com Empatia!

"Procure primeiramente compreender", implica numa mudança profunda de paradigma. Tipicamente procuramos primeiro que nos compreendam.

A maioria das pessoas não conseguem escutar com a intenção de compreender, mas sim de expor seu ponto de vista.

Diagnóstico antes de prescrição

Compreenda!

Apesar de ser arriscado e difícil procurar primeiro compreender (ou diagnosticar) antes de prescrever algo, é um princípio correto em muitas áreas da vida.

Crie Sinergia

Crie Sinergia

Convergência das partes de um todo que concorrem para um mesmo resultado;

Efeito resultante da ação de vários agentes que atuam da mesma forma, cujo valor é superior ao valor do conjunto desses agentes, se atuassem individualmente.

Crie Sinergia

Quando você está se comunicando sinergicamente, está simplesmente abrindo o seu coração, sua mente e o modo de expressão para as novas possibilidades, novas alternativas e novas opções.

Referências

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

Referências

- [Beck 1999a] Beck, K. Extreme Programming Explained – Embrace Change. Addison-Wesley. 1999.
- [Beck 1999b] Beck, K. Embracing Change with Extreme Programming. IEEE Computer, October, 1999.
- [Beck 2001] Beck, K. et al. Manifesto for Agile Software development. 2001.
- [Abrahamsson 2002] Abrahamsson, P. et al. Agile software development
- methods: reviews and analysis. Espoo: VTT Publications, 2002.
- [Covey 1989] Os 7 hábitos das pessoas altamente eficazes