FIAP

MBA+

MBA EM ARQUITETURA E DESENVOLVIMENTO NA PLATAFORMA .NET

FIAP

MBA+

METODOLOGIA DE DESENVOLVIMENTO ÁGIL (SCRUM)

Prof. Frederico Oliveira
proffrederico.oliveira@fiap.com.br
2018

RELEMBRANDO: Entrega do projeto final

- **09/06/2018 (sábado) até as 23:59hs, na plataforma FIAP, um documento contendo:**
  - Nome dos integrantes do PO (cliente) e do time (Aula 1);
  - Descrição do problema/necessidade e *Business Model Canvas* (Aula 1);
  - Planilha contendo *user story* conforme solicitado na Aula 2, com suas prioridades e estimativas iniciais;
  - Backlog selecionado e Definição de Pronto (Aula 3);
  - Sprint Backlog e descrição de como o time se auto-organizou (Aula 3);
  - Produto realizado na Sprint (Aula 4);
  - Novos itens priorizados pelo PO durante a realização da Sprint (Aula 4);
  - Importância das reuniões diárias, itens mais relevantes discutidos na Reunião de Revisão e discussões realizadas na Retrospectiva (Aula 4/5);
  - Parecer final do produto pelo Product Owner.

REVISÃO DA ÚLTIMA AULA

Sprint

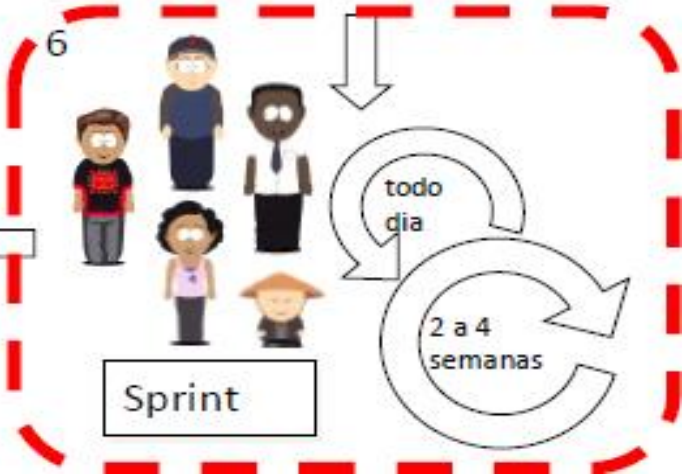
FIAP

Sprint Planning 1

Backlog Selecionado

Sprint Planning 2

Sprint Backlog

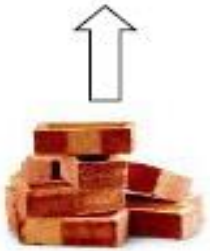


Sprint

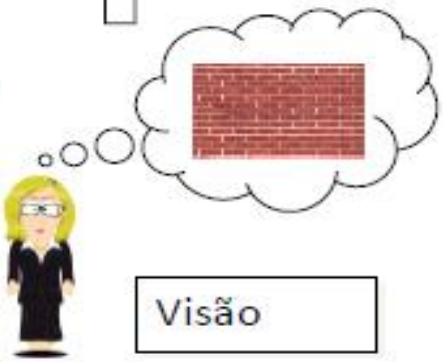
Entrega

Retrospectiva

outra iteração

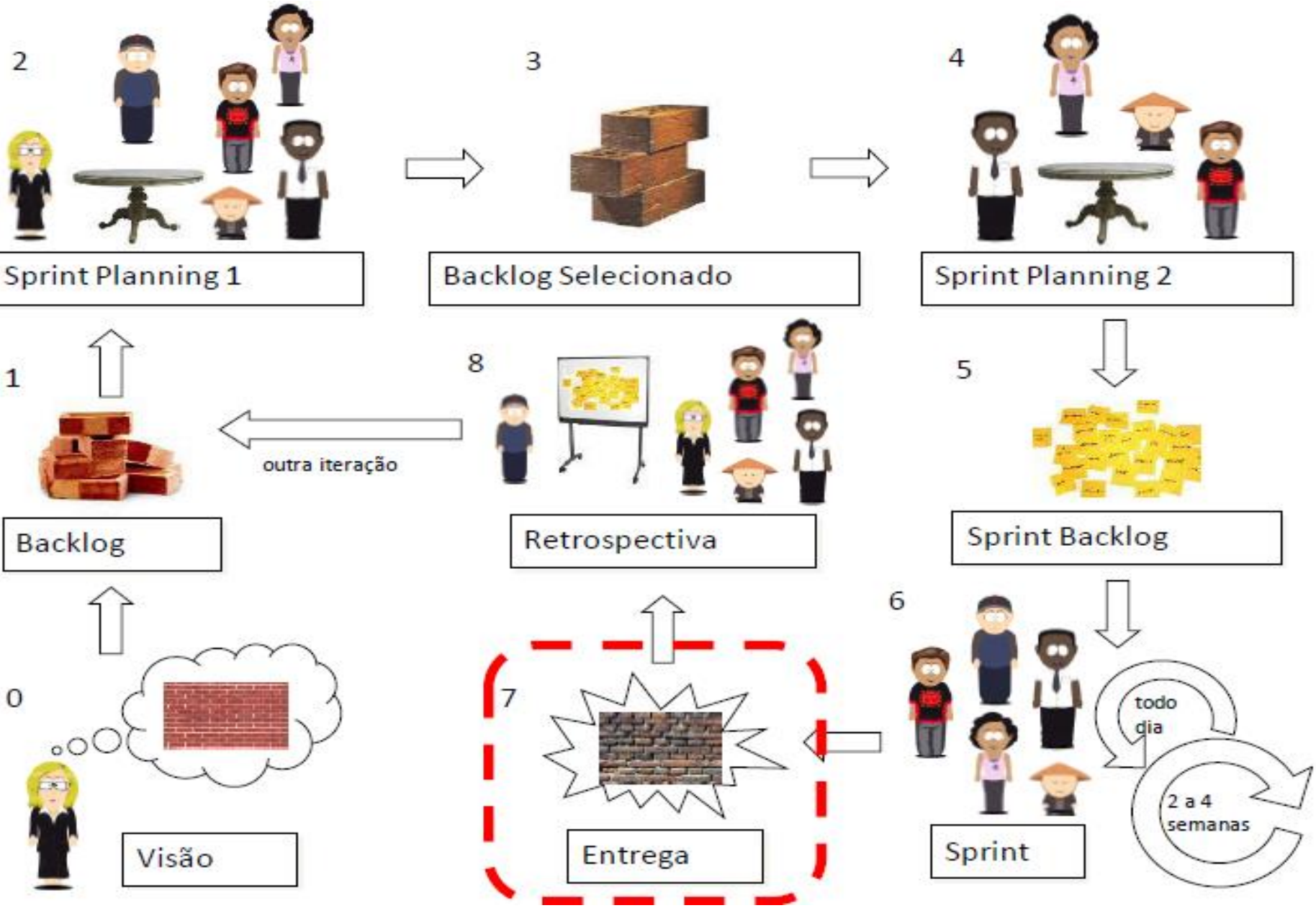


Backlog



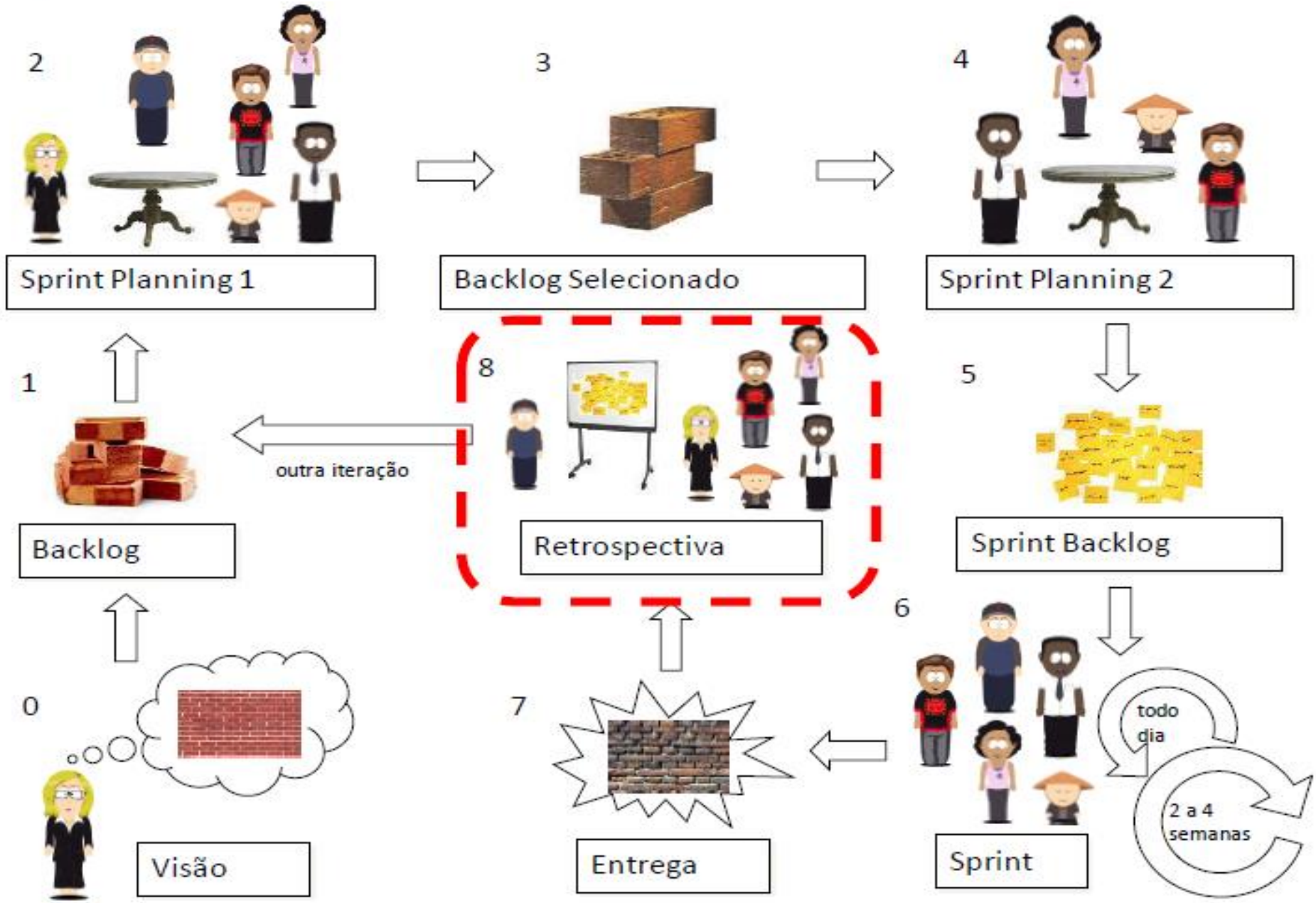
Visão

Entrega


2
Sprint Planning 1
3
Backlog Selecionado
4
Sprint Planning 2
5
Sprint Backlog
6
todo dia
2 a 4 semanas
Sprint
7
Entrega
8
Retrospectiva
outra iteração
1
Backlog
0
Visão

RETROSPECTIVA

Retrospectiva



Retrospectiva

- Reunião que repassa por tudo o que aconteceu durante a *Sprint*, com foco em **melhoria contínua**.
- Inspecionar como a última *Sprint* foi em relação às **pessoas**, aos **relacionamentos**, aos **processos** e às **ferramentas**;
- Identificar e ordenar os **principais itens que foram bem e as potenciais melhorias**;
- Criar um **plano para implementar melhorias** no modo que o Time *Scrum* faz seu trabalho;
- Participam dela todos os envolvidos no projeto;
- *Scrum Master* é o facilitador da reunião;
- Duração: **3 horas para uma *Sprint* de 4 semanas**;

Retrospectiva

O que não deve acontecer na reunião

- Acusações entre membros da Equipe *Scrum*;
- Falta de respeito ou de educação;
- Tentativas de isenção sobre um possível fracasso da *Sprint*. Tanto o sucesso quanto o fracasso são de responsabilidade de toda a Equipe *Scrum*;
- Falar somente dos pontos positivos;
- Ter um membro da Equipe “dominando” a reunião. Todos devem contribuir;
- Ter pressa para terminar;
- Ser uma reunião política, evitando entrar em assuntos mais delicados que necessitam discussão;
- Finalizar a reunião sem um plano de ação de melhorias identificado.

Retrospectiva

Como organizamos retrospectivas

- Nós alocamos 1 a 3 horas dependendo da situação da *Sprint*;
- **Participantes: *Product Owner*, Equipe toda e *Scrum Master*;**
- Deve ocorrer em uma sala fechada onde não a reunião não possa ser interrompida por terceiros;
- ***Scrum Master* mostra o *Spring Backlog* e, com a ajuda do Time, resume a *Sprint*. Eventos e decisões importantes, etc;**
- Cada pessoas tem a chance de dizer, sem ser interrompido, o que eles acharam que foi bom, o que eles acharam que poderia ter sido melhor, e o que eles gostariam de fazer de forma diferente no próximo *Sprint*;
- **Devemos comparar a velocidade estimada com a velocidade realizada. Se existir uma grande diferença temos analisar o motivo;**
- Quando o tempo já está quase acabando, o *Scrum Master* tenta resumir as sugestões concretas sobre o que poderemos fazer melhor no próximo *Sprint.*

Retrospectiva

Como organizamos retrospectivas

OU

Sprint	Nome	Papel	Pontos positivos	Pontos negativos	Pontos de melhoria	Ações

Retrospectiva

Exemplos de coisas que podem aparecer na Retrospectiva

- **Interferências externas demais**
  - **Ações**:
    - Peça ao Time para recordar melhor as interferências no próximo *Sprint*. **Quem interrompeu, quanto tempo tomou**. Será mais fácil resolver o problemas depois.
    - Peça ao Time para **concentrar as interferências no *Scrum Master* ou *Product Owner***;
    - Peça ao Time para designar uma pessoa para ser o "**goleiro**". Todas as interrupções serão repassadas à ele, assim o resto do Time pode ser concentrar. Pode ser o *Scrum Master* ou ser revezado entre o resto.

Retrospectiva

Exemplos de coisas que podem aparecer na Retrospectiva

- **“Nós nos comprometemos com coisas demais e só metade está pronta”**
  - **Ações**:
    - O Time provavelmente não vai se comprometer com coisas demais no próximo *Sprint*. Ou no mínimo não errar por tanto.

- **“Nosso ambiente é barulhento e bagunçado demais”**
  - **Ações**:
    - Tente criar um ambiente melhor, ou tire o Time do local.
    - Se não for possível, considere a velocidade real deste Sprint para os próximos e deixar claro que isto se deve ao ambiente barulhento e bagunçado.

Retrospectiva

Encerrar a reunião

- **Em vez de: "vocês precisam melhorar na próxima *Sprint*, hein?"**
- **Diga: "eu tenho certeza que podemos e vamos fazer melhor na próxima *Sprint*"**

- **Em vez de: "*pessoal, este projeto não está indo bem! Espero que depois desta reunião vocês consigam identificar pontos de melhoria*"**
- **Diga: "*pessoal, estamos com dificuldades, mas vamos conseguir superá-las. Tenho certeza de que, depois da reunião de hoje, cada um de nós sai daqui com vontade de fazer diferente, com vontade de fazer melhor, e nós vamos conseguir fazer melhor, porque aprendemos muito hoje nesse papo que tivemos.*"**

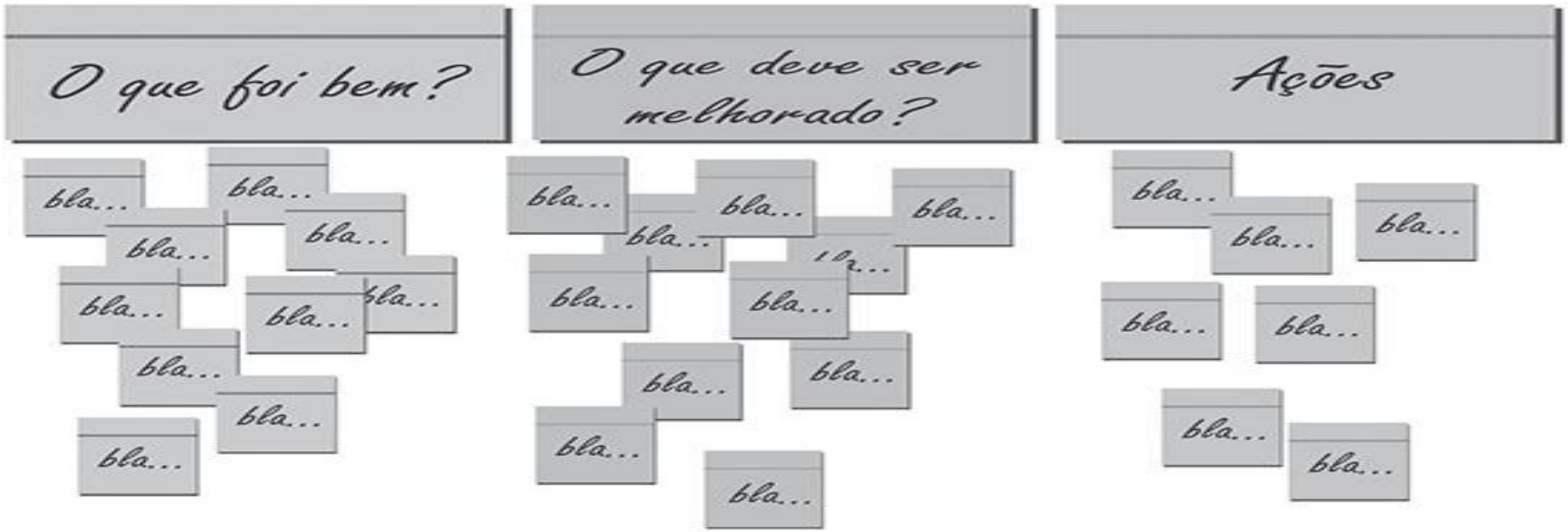
O *Scrum Master* deve ter uma visão crítica referente ao processo, mas uma atitude motivadora perante as pessoas.

SESSÃO “MÃO NA MASSA” – RETROSPECTIVA PROJETO FINAL USANDO MÉTODOS ÁGEIS/SCRUM

Sessão “mão na massa”
Retrospectiva

A. Tempo para a Reunião de Retrospectiva: **10 minutos**.

Entregáveis para o projeto final:



OU

Sprint	Nome	Papel	Pontos positivos	Pontos negativos	Pontos de melhoria	Ações

SESSÃO “MÃO NA MASSA” – APRESENTAÇÃO FINAL
PROJETO FINAL USANDO MÉTODOS ÁGEIS/SCRUM

Projeto Final

- **Apresentação Final do projeto**:
  - **Necessidades** do PO e **objetivos** do projeto.
  - ***Backlog* selecionado** para a 1ª Sprint.
  - Produto realizado na *Sprint* (protótipos).

- Cada equipe terá **8 minutos** para a apresentação.
- Perguntas/Respostas podem ser feitas durante **2 minutos**.

eXtreme Programming (XP)

Origem

Kent Beck criou o **XP** após um projeto crítico na **Chrysler**. Ele foi convidado para conduzir o desenvolvimento de um sistema problemático para **folha de pagamento**, cujos custos e prazos já haviam estourados algumas vezes sem alcançar resultados significativos.

Beck conduziu o projeto usando uma **proposta radical** que tornou a equipe capaz de entregar um sistema com ótima qualidade e que transformou um projeto fadado ao fracasso em um case da indústria de software. **As práticas de engenharia de software usadas nesse projeto formaram a base para a criação da programação extrema**.

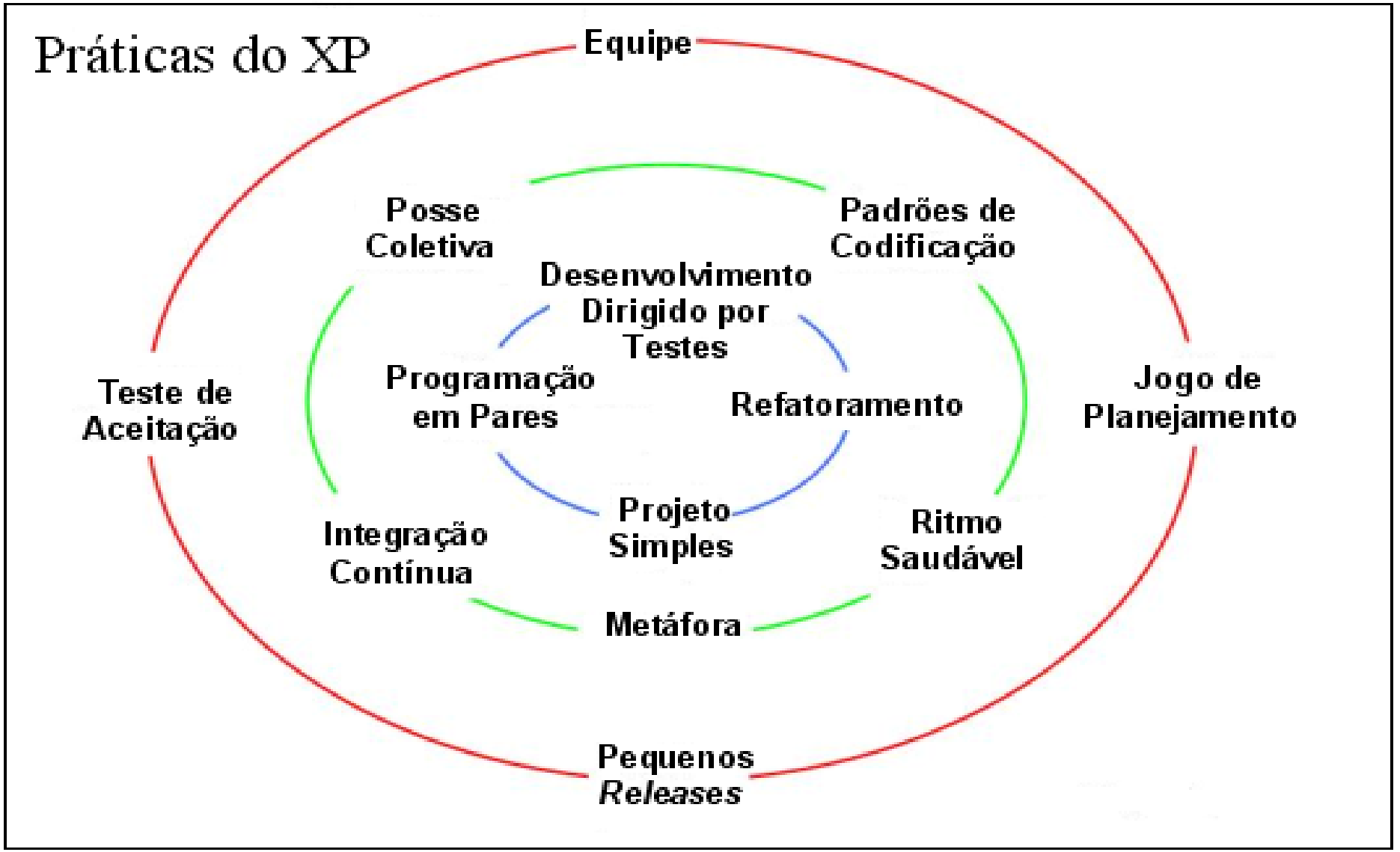
Valores

- **Comunicação**
  - O time deve manter informação fluindo entre todos;
  - Permite ao time aprender com os problemas.
- **Simplicidade**
  - A melhor solução é a mais simples!
- **Feedback**
  - Rapidez nas respostas do sistema (teste) e do usuário (requisitos). O que achamos certo hoje pode não ser amanhã!
- **Coragem**
  - Fazer mudanças no código, refazer o código, mudar sem medo
- **+ Respeito**
  - Os membros devem ser importar uns com os outros. Todos são importantes para o projeto!

Princípios

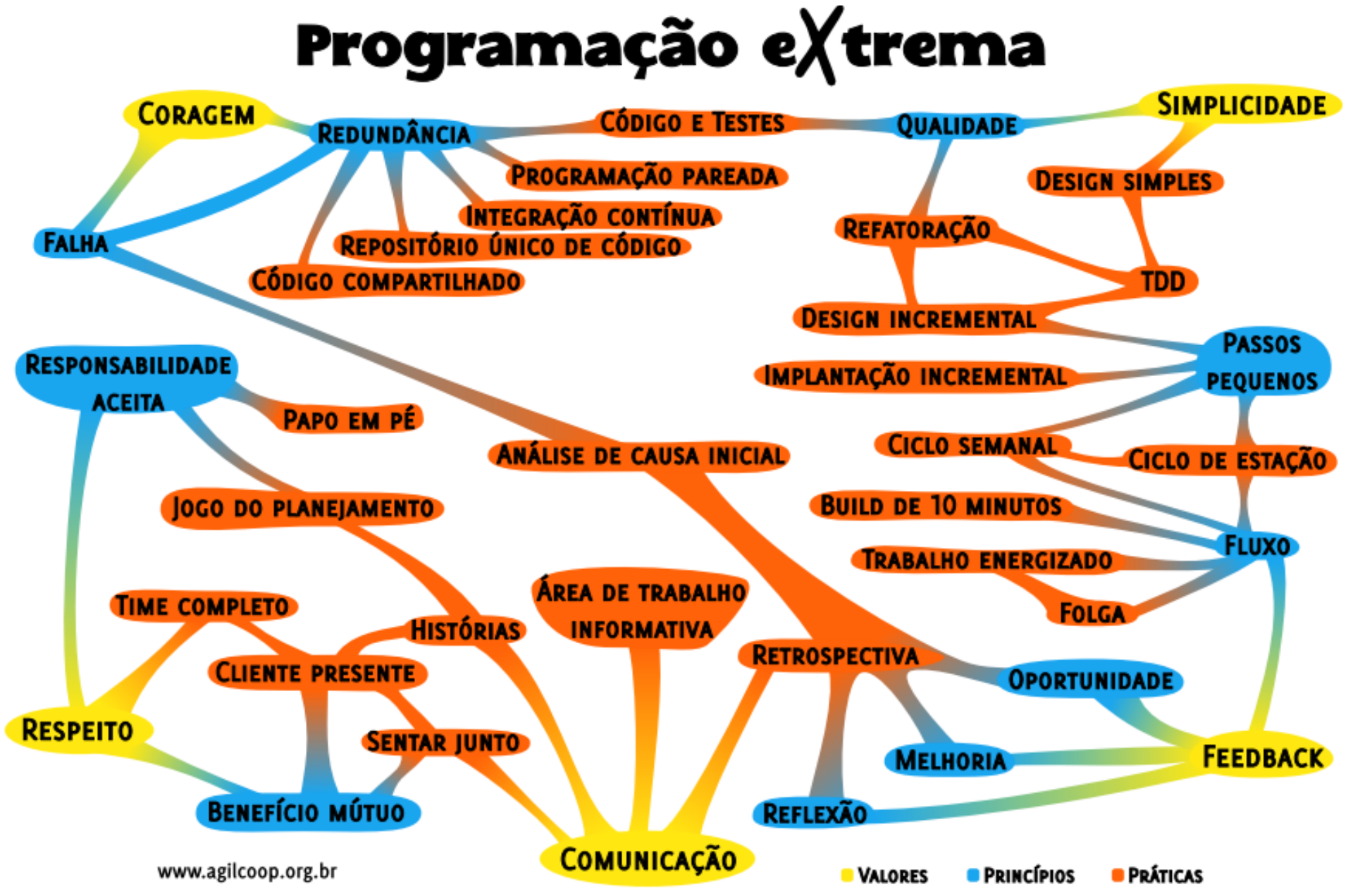
- **Humanidade –** a produção do software depende das **pessoas**.
- **Economia –** a equipe deve conhecer as necessidades de negócio e definir **prioridades que agreguem o máximo de valor no menor intervalo de tempo**.
- **Melhoria – melhorias** devem ser implementadas **constantemente**.
- **Semelhança – boas soluções devem poder ser aplicadas novamente**, inclusive em outros contextos. ***Padrão?***
- **Diversidade –** a **equipe** deve reunir muitas **habilidades**.
- **Passos pequenos –** a **integração do código**, o **desenvolvimento dirigido por testes** e a **entrega de novas versões** devem manter um tamanho **pequeno** o suficiente para que a qualidade seja mantida.
- **Reflexão –** periodicamente a equipe deve **refletir** sobre seu próprio **trabalho**.
- **Fluxo –** o **ritmo de trabalho** deve ser **sustentável** ao longo do tempo.
- **Oportunidade –** a equipe deve estar sempre disposta a melhorar.
- **Redundância – testar frequentemente** reduz as possibilidades de entregas erros ao cliente.
- **Qualidade –** a **qualidade não é um fator negociável**.
- **Aceitação da responsabilidade –** as **responsabilidades são aceitas** e não impostas.

Práticas


Práticas do XP
Equipe
Posse Coletiva
Padrões de Codificação
Desenvolvimento Dirigido por Testes
Programação em Pares
Refatoramento
Teste de Aceitação
Jogo de Planejamento
Projeto Simples
Integração Contínua
Ritmo Saudável
Metáfora
Pequenos Releases

Fusão dos valores, princípios e práticas

FIAP



Por que esse nome (XP)?

- **A razão é o uso EXTREMO do senso comum:**
  - Se entregas frequentes é uma boa prática, vamos entregar software o tempo todo >> **iterações curtas**.
  - Se testar é bom, vamos testar o tempo todo e deixar o cliente testar >> **testes unitários** e de **aceitação**.
  - Se integrar o sistema é bom, vamos integrar o sistema com maior frequência possível >> **integração contínua**.
  - Se revisar código é bom, vamos revisar código o tempo todo >> programação pareada (**Programação em Pares**).
  - Se desenho é bom, vamos torná-lo parte do dia a dia do desenvolvedor >> **refatoração**.

Por que devo adotar XP (novamente)?

FIAP

Foco forte em Técnicas de Programação

E se juntar Scrum e XP?

Técnicas de gerenciamento + Técnicas de Programação

Programação em Pares

FIAP

- Inicie por fazer em uma *Sprint* como uma ‘prática piloto’.
- **Benefícios**:
  - Aumenta a **qualidade do código**;
  - Aumenta o **foco do desenvolvedor** (por exemplo: quando o par lhe diz: “*Ei, estas coisas são necessárias para este Sprint?*”);
  - **Nivela o conhecimento** da equipe de forma rápida;
  - *Revisão e Teste em Par:* use o conceito também pós-desenv.!
- **Cuidados**:
  - É cansativa e não deve ser feito todos dias (no início);
    - **Trocar os pares frequentemente é bom**.
  - Não desconsidere um excelente programador simplesmente porque ele não se sente confortável com programação em par.

Desenvolvimento Orientado a Testes

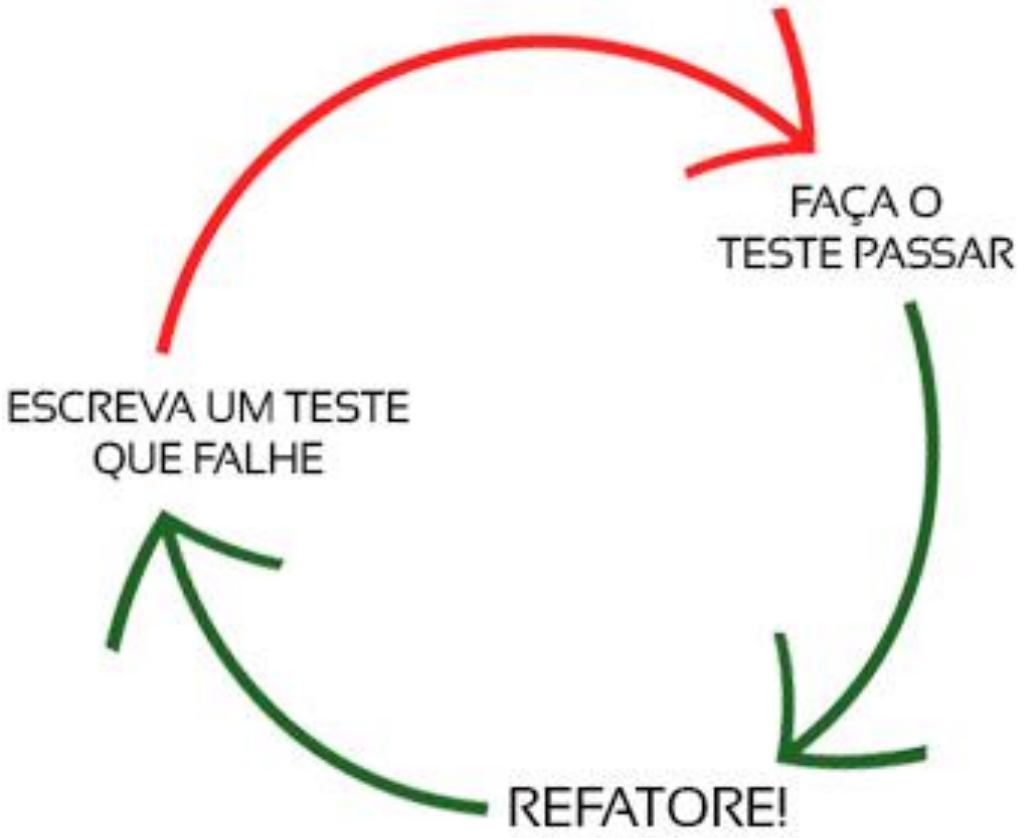
FIAP

- Todos já sofremos de uma nova versão do produto que traz novas funcionalidades, mas faz as anteriores pararem de funcionar. Certo?

- Caso algo pare de funcionar, o desenvolvedor é rapidamente notificado, e consegue corrigir o problema antes de mandar a versão para o cliente.

- **TDD** se baseia em pequenos ciclos, **onde para cada funcionalidade do sistema um teste é criado antes**. Este novo teste criado inicialmente falha, já que não temos a implementação da funcionalidade em questão e, em seguida, implementamos a funcionalidade para fazer o teste passar!

Desenvolvimento Orientado a Testes



Desenvolvimento Orientado a Testes

FIAP

- **Reflexões**:
  - **TDD é difícil**. Leva um tempo para o desenvolvedor pegar o jeito. Em muitos casos a única maneira é fazendo ele programar em par com alguém que é bom em TDD;
  - TDD **tem um efeito profundamente positivo no *design*** do sistema;
  - Tenha certeza que você investirá o **tempo necessário para tornar mais fácil escrever os testes**. Isso significa obter as ferramentas certas, educar as pessoas, etc.

Integração Contínua

FIAP

“É uma prática de desenvolvimento de software onde os membros de um time integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia. Cada integração é verificada por um *build* automatizado (incluindo testes) para detectar erros de integração o mais rápido possível.”

Martin Fowler

Propriedade coletiva do código

FIAP

- **Programação em Pares** com a mudança frequente dos pares leva **automaticamente a um maior nível de propriedade coletiva de código**;

- **Não morre uma *Sprint*** apenas porque alguém importante ficou **doente ou saiu do projeto**;

- O código possui um único dono: a **Equipe**;

- Todos compartilham a responsabilidade das alterações;

Time / Ambiente de trabalho informativo

- **Sentar Junto**
  - Faça com que o Time trabalhe em um espaço aberto grande o suficiente para que todos fiquem juntos, ou bem próximos;
  - A integração entre as pessoas melhora a comunicação, o entendimento do sistema e a qualidade final.
- **Time completo**
  - Garanta que o time conte com pessoas com todas as habilidades necessárias para o projeto ter sucesso.
- **Ambiente informativo**
  - Deve ser possível saber como o projeto está somente passando pelo local de trabalho do time.

Time / Ambiente de trabalho informativo

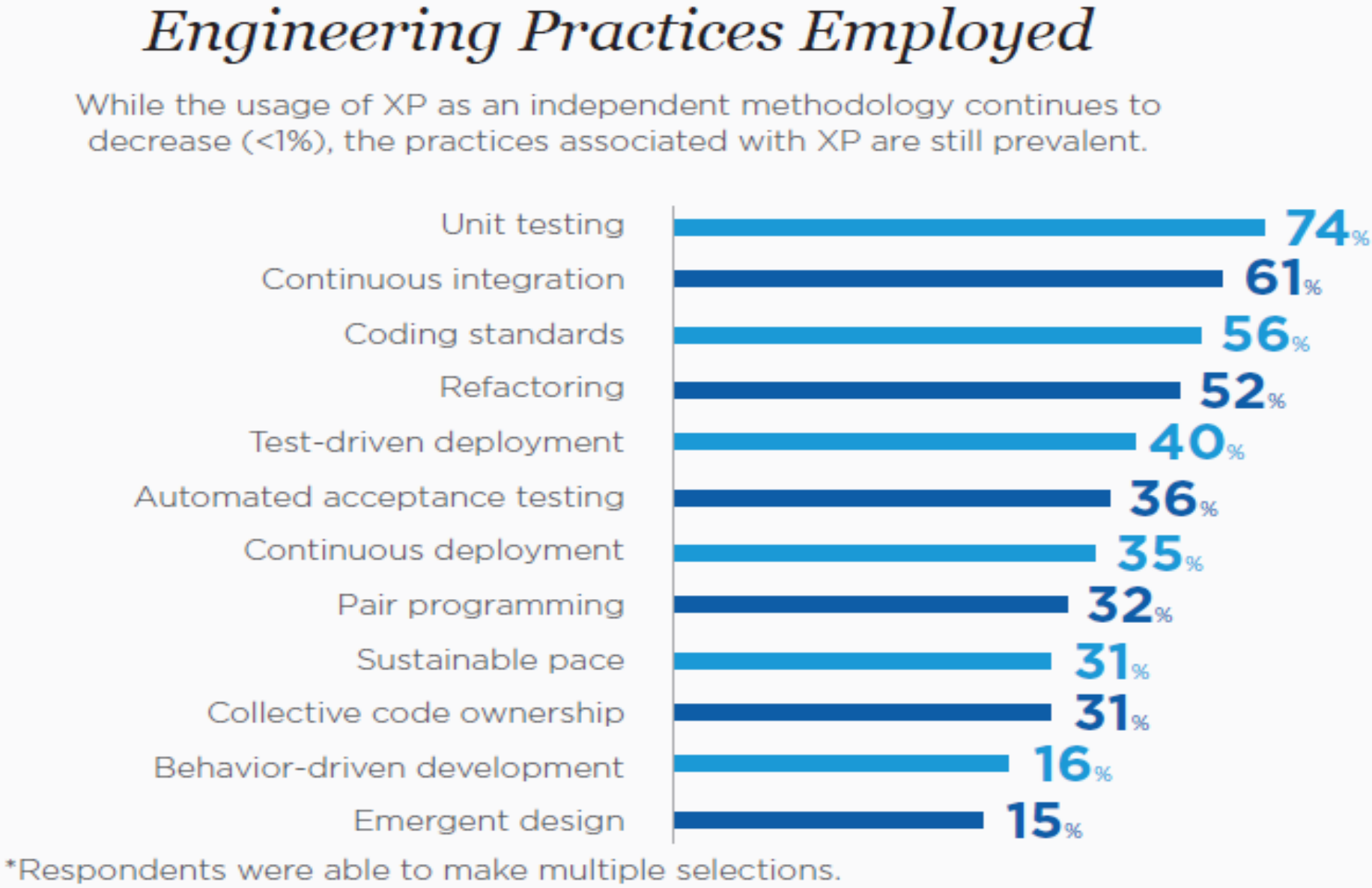
Padrão de Codificação

- Comece com ele simples e vá crescendo conforme a necessidade;
- Escreva apenas regras que não sejam óbvias para todo mundo e correlacione com algum material já existente, sempre que possível;
- Exemplos:
  - Use a convenção de codificação da Sun/Oracle como padrão (http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html);
  - Nunca, jamais, nem pense em capturar exceções sem logar o *stacktrace* (*log.debug()* é uma boa);
  - Evite abreviações. Abreviações bem conhecidas como *DAO* são bem-vindas;
  - Os métodos que retornam *collections* não devem retornar nulo.

Ritmo Saudável

- Prioriza a qualidade de vida da equipe e evitar trabalhar em excesso de horas-extras;
- Se os analistas trabalham em excesso, ficam cansados, consequentemente queda de produtividade;
- Assim, o tempo de trabalho ganho com as horas extras é perdido devido ao cansaço.

Principais práticas (Annual State)



Sessão “mão na massa”

O que podemos levar para as nossas empresas em relação às 5 aulas que tivemos?

Mensagem Final

Sempre o foco deve ser
a satisfação do nosso cliente!

OBRIGADO!

Resumo – 5ª Aula

- Sessão “mão na massa”
- XP