

# Desenvolvimento Ágil e Enxuto de Software

*Prof. Breno de França*

breno@ic.unicamp.br

# *Por que um método mais Ágil?*

## Mercado: Competitividade

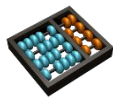


Fonte:  
<http://www.yourchennai.com/2016/12/28/goa-institute-of-management-sees-strong-growth-in-placements-this-year/>

## *Time-to-market*



Fonte:  
<https://www.a-star.edu.sg/Collaborate/Programmes-for-SMEs/Overview.aspx>



# *Por que um método mais Ágil?*

Quando você descobre se um produto (código ou especificação) está correto?

- ☐ Quando termina
- ☐ Quando você revisa/testa
- ☐ Quando um par revisa/testa
- ☐ Quando o cliente/usuário vê
- ☐ Quando entra em produção



# Por que um método mais Ágil?

## Frequência de *Feedback*

Meses?

Semanas?

Dias?

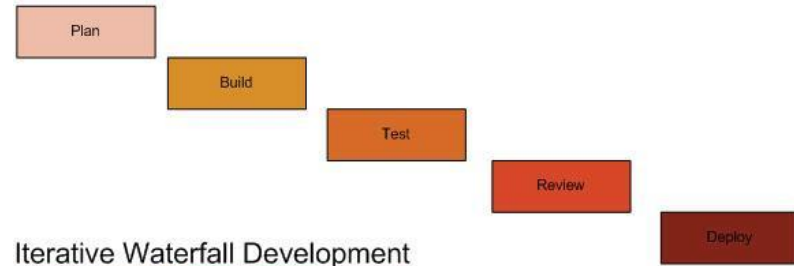
Horas?

Minutos?

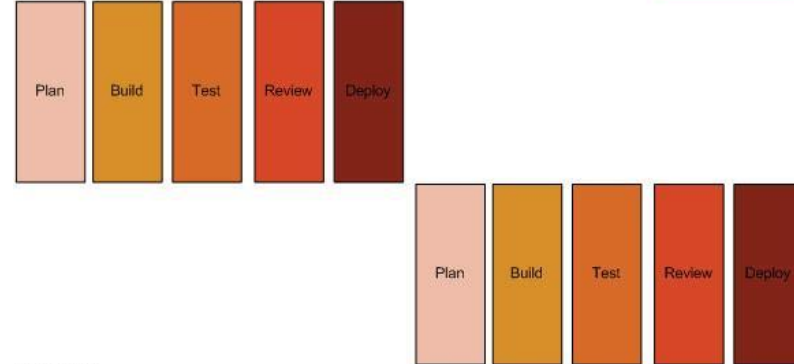
## Flexibilidade

Quanto e quando eu posso mudar?

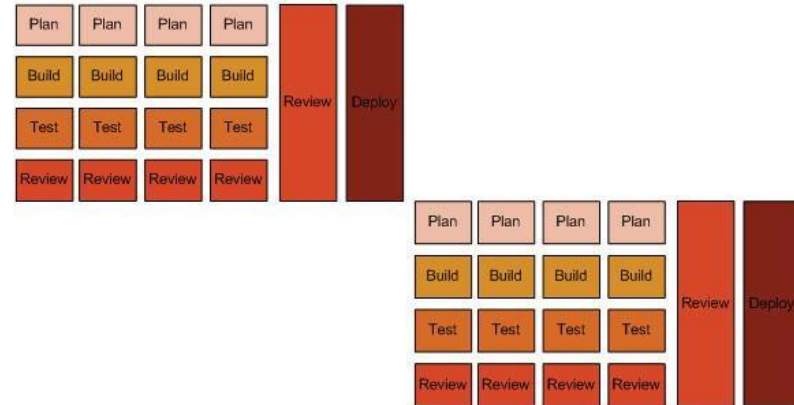
Waterfall



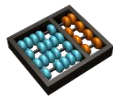
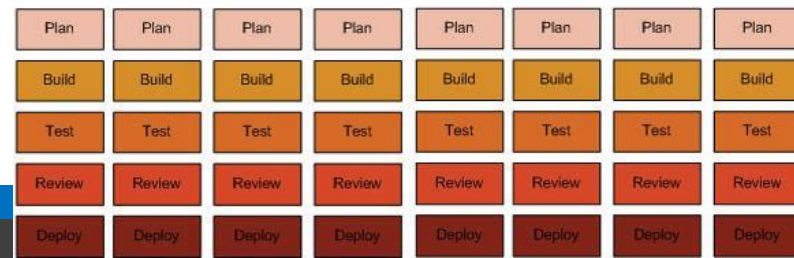
Iterative Waterfall Development



Scrum



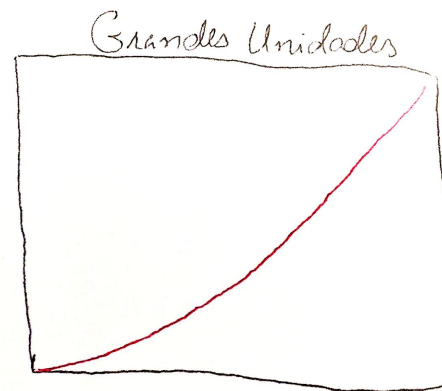
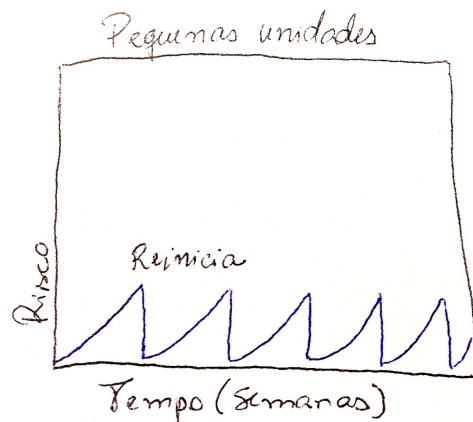
Lean



# Objetivos

Entrega contínua de software funcionando  
(*working software*)

- Demonstrar ao cliente para ilustrar o progresso do desenvolvimento
- Verificar se atende às necessidades
- *Early feedback*: confiança de que o software atende às necessidades
- Em ciclos curtos e regulares



# Valores



Fonte: <https://tinyurl.com/ycnv2xv4>

**Indivíduos e interações** 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

*[agilemanifesto.org](http://agilemanifesto.org)*



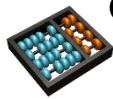
# Valores e Princípios

**Trabalhar em conjunto:** desenvolvedores e pessoas de negócio devem trabalhar em conjunto diariamente em todo o projeto.

~~**Indivíduos motivados:** construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.~~ *Princípio? Quem apoiaria o desenvolvimento com pessoas desmotivadas?*

**Conversas cara-a-cara:** o método mais eficiente e eficaz de transmitir informações para, e dentro de um time de desenvolvimento, é através de uma conversa cara-a-cara.

**Equipes auto-organizáveis:** as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.



*[agilemanifesto.org](http://agilemanifesto.org)*

# Valores e Princípios

**Software funcional:** é a medida primordial de progresso do projeto.

**Simplicidade:** a arte de maximizar a quantidade de trabalho que não precisou ser feito.

~~**Excelência técnica:** contínua atenção à excelência técnica e bom design. Alta qualidade de bom design facilitam manutenção e mudanças, tornando o projeto mais ágil. Princípio? Quem gostaria de trabalhar com uma equipe desqualificada?~~



*agilemanifesto.org*



# Valores e Princípios

**Satisfação do cliente:** maior prioridade é satisfazer o cliente por meio da entrega antecipada e contínua de software de valor.

**Abertura a mudanças:** abertura a mudança de requisitos, sejam estas cedo ou tarde no projeto. A habilidade de reagir à mudanças tardias é vista como uma vantagem competitiva.

**Entrega frequente:** entregar frequentemente software funcionando, em poucas semanas ou meses, preferencialmente em ciclos curtos.



*[agilemanifesto.org](http://agilemanifesto.org)*

# Valores e Princípios

**Andamento sustentável:** Processos ágeis promovem um andamento sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.

**Reflexão contínua:** em intervalos regulares, a equipe deve refletir sobre como ficar mais eficiente, então, ajustam-se e otimizam seu comportamento de acordo.



*[agilemanifesto.org](http://agilemanifesto.org)*

# Valores e Princípios

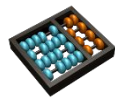
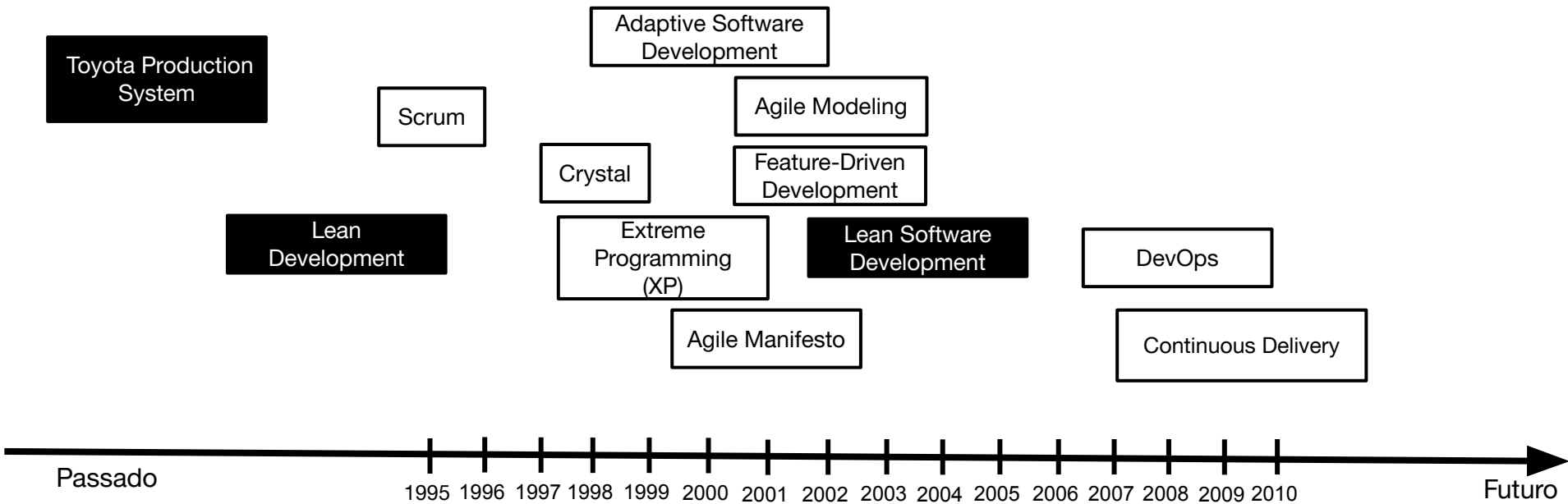
## Em resumo:

*Desenvolvedores de software **motivados** e com **poder de decisão**, considerando sua **excelência técnica** e o **design simples**, criam **valor de negócio** por meio da entrega do software **funcionando** aos usuários em intervalos **curtos e regulares**.*



Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6), 1213-1221.

# Histórico - Métodos

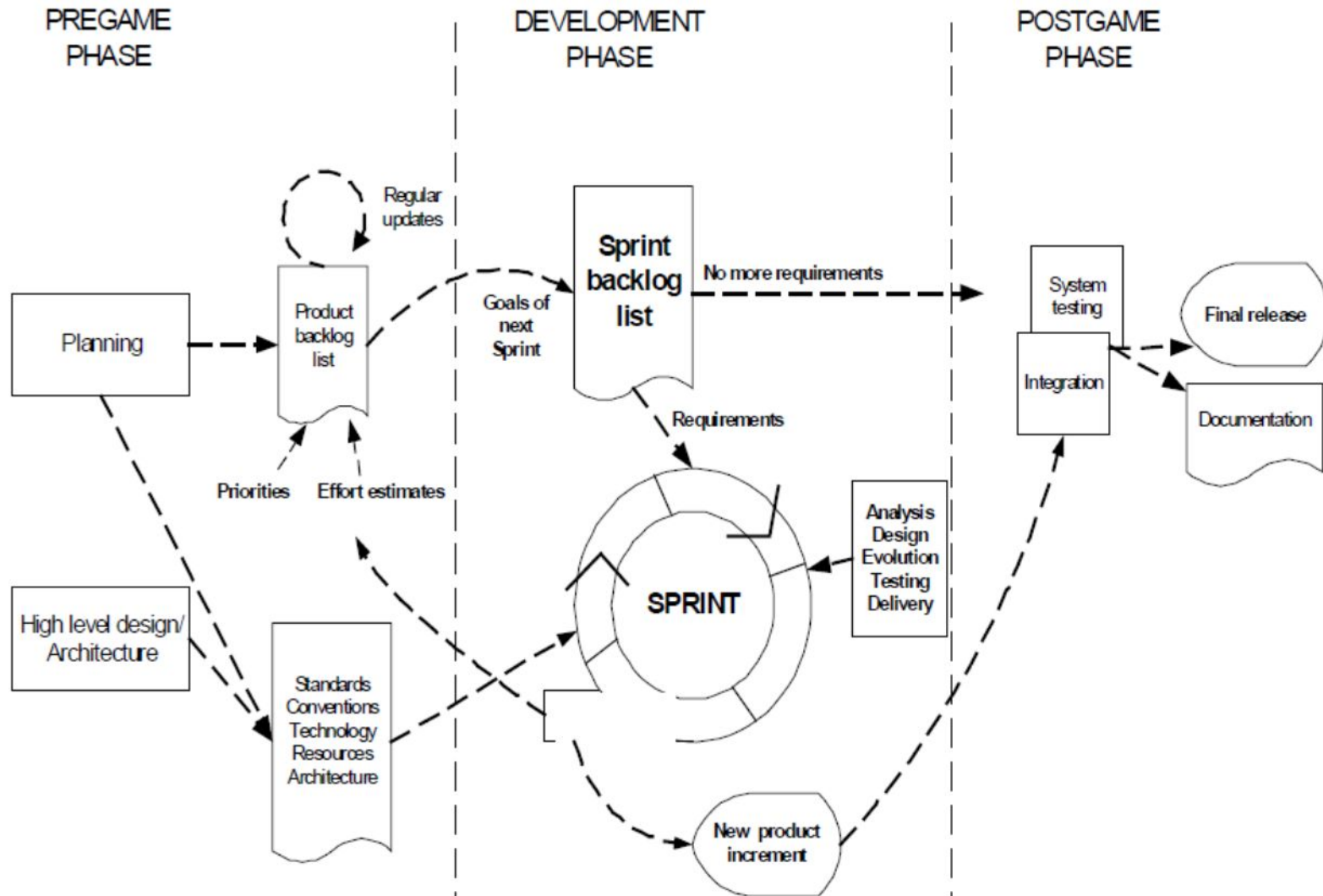


Lean

Agile

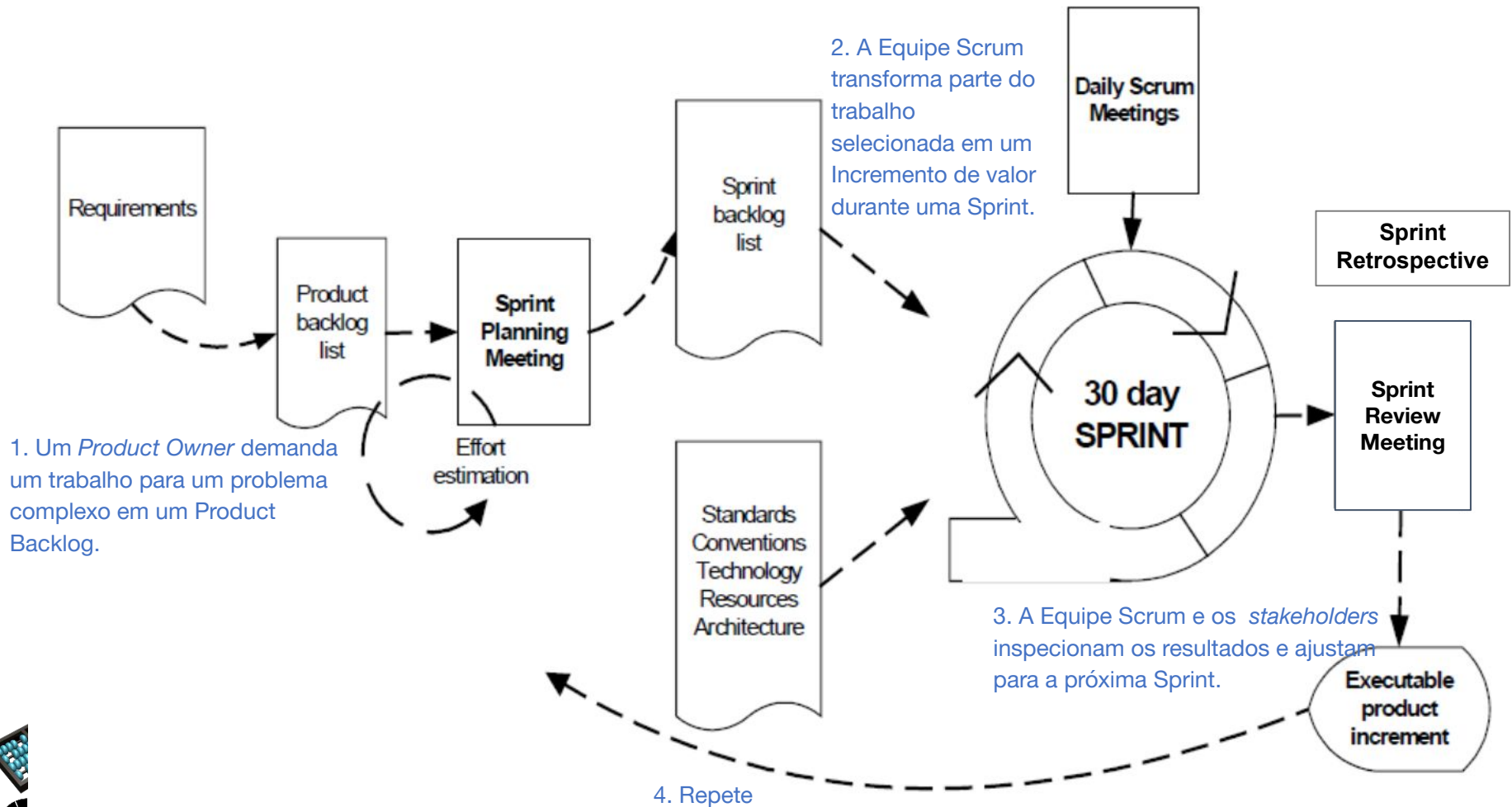


# Scrum



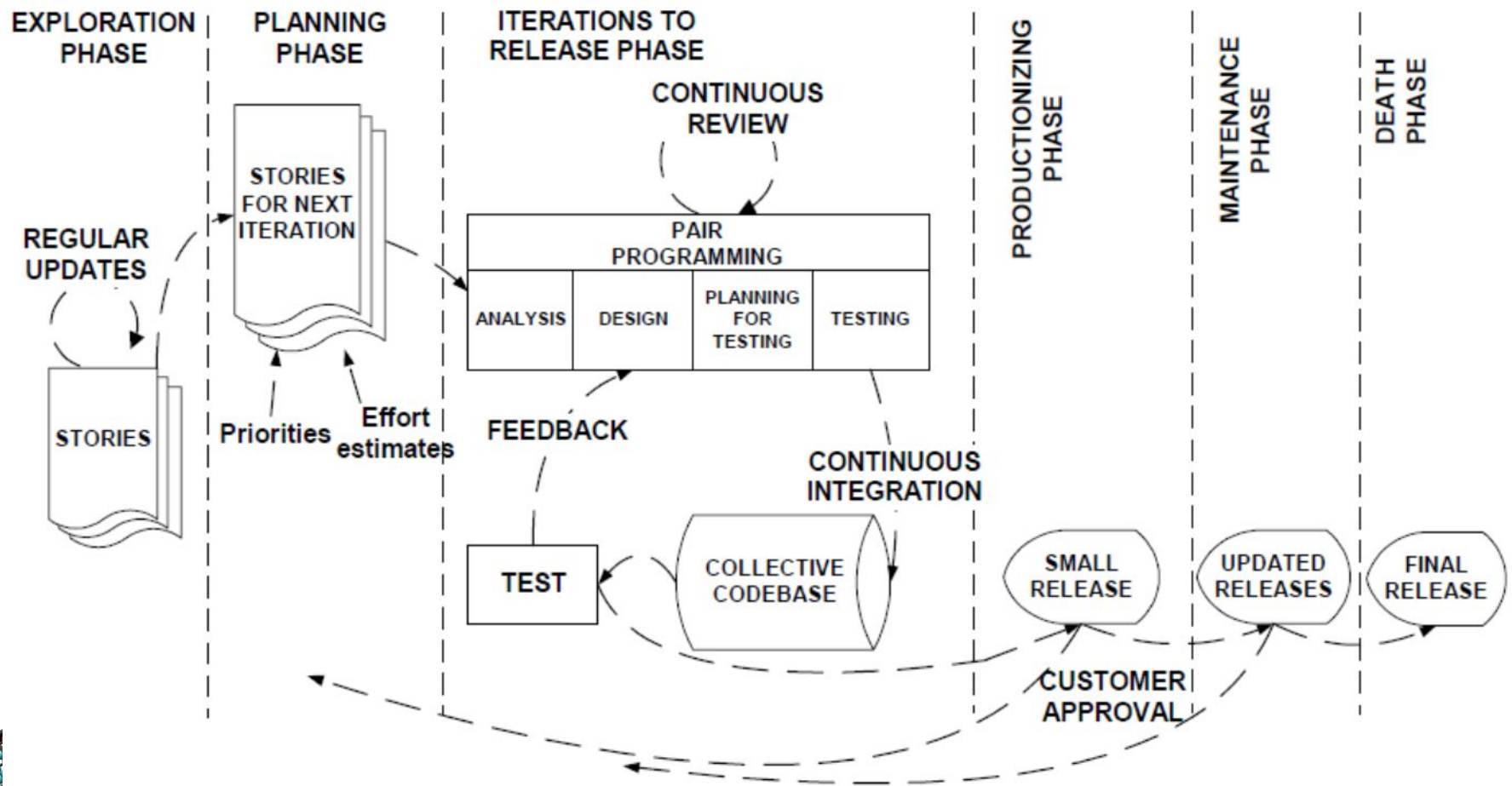
Fonte: Abrahamsson, Pekka, et al. "Agile software development methods: Review and analysis." (2002).

# Scrum



Fonte: Abrahamsson, Pekka, et al. "Agile software development methods: Review and analysis." (2002).

# eXtreme Programming (XP)



Fonte: Abrahamsson, Pekka, et al. "Agile software development methods: Review and analysis." (2002).

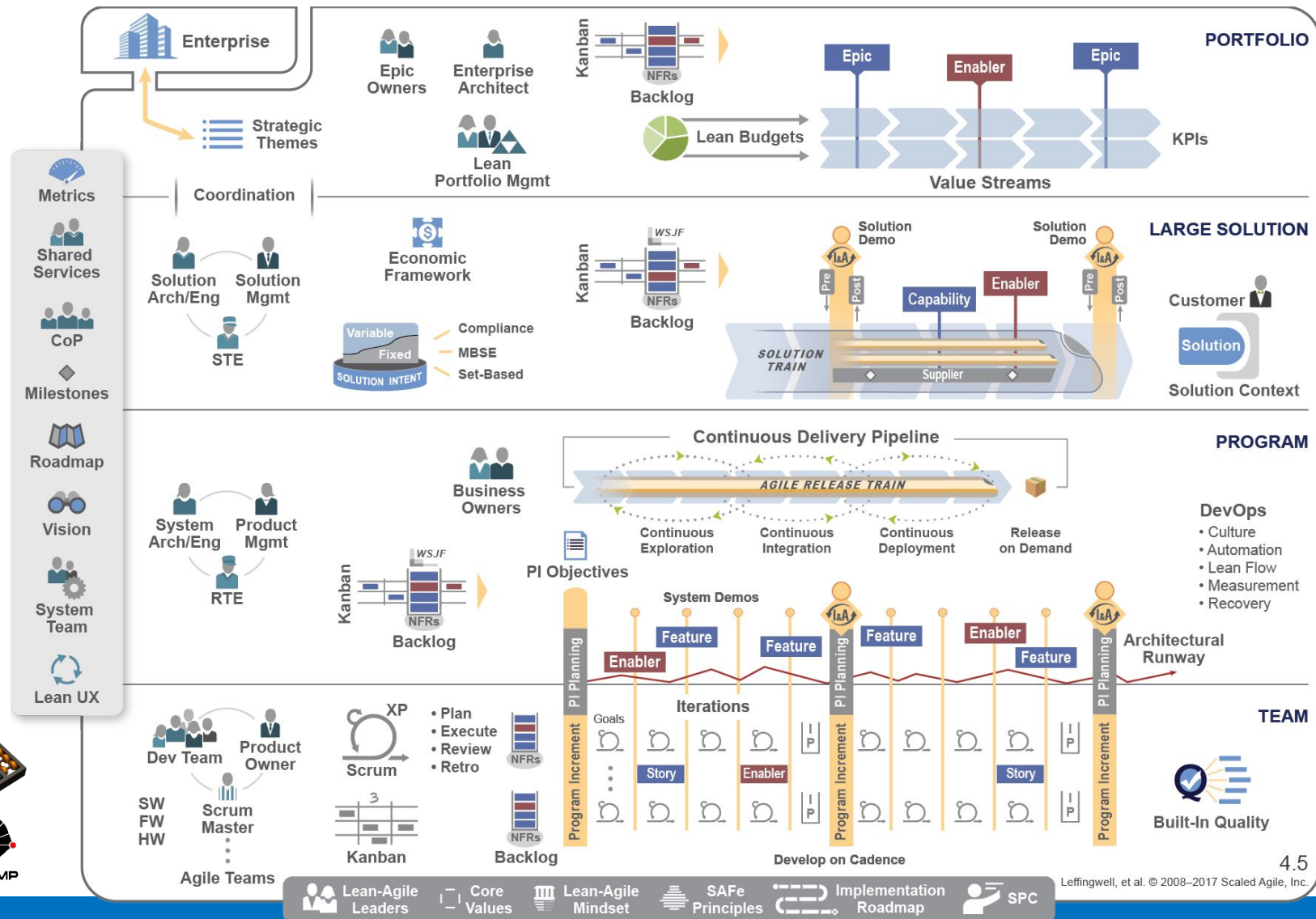
# eXtreme Programming (XP)



Fonte: Beck, Kent. Extreme programming explained: embrace change. Addison-Wesley Professional, 2000.



# SAFe (Scaled Agile Framework)



# Práticas Ágeis

# Práticas Ágeis

## Engenharia de Requisitos



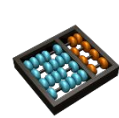
- Cliente Presente (*on-site customer*)
  - *Feedback* imediato
- Histórias de usuários
  - Histórias ou features: Requisitos mais específicos. Úteis para determinar o progresso do projeto

# Práticas Ágeis

## Projeto e Implementação



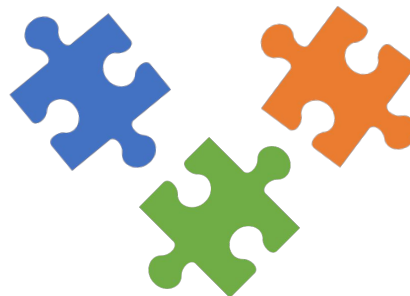
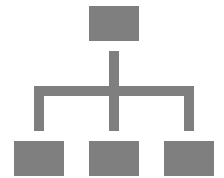
- *Refactoring*
  - Melhoria contínua do código (manutenibilidade, legibilidade, simplificação)
- Padrão de Codificação
  - Desenvolvedores estruturam e escrevem código de forma uniforme
- Posse Coletiva do Código
  - Todos são donos do código e podem alterá-lo



# Práticas Ágeis

## Projeto e Implementação

- Arquitetura de baixa dependência
  - Independência entre as funcionalidades
  - Entrega contínua de software funcionando
  - Flexibilidade pela facilidade de mudança



# Práticas Ágeis

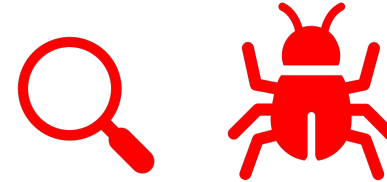
## Garantia da Qualidade

- Desenvolvimento Orientado a Teste (TDD)
  - Implementar os casos de teste antes das funcionalidades
  - Verificar a implementação (capturar defeitos) o mais cedo possível
  - Promove automação
- Automação de testes



# Práticas Ágeis

## Garantia da Qualidade



- Programação em pares
  - Dois desenvolvedores compartilham uma estação de trabalho
  - Um atua como revisor (impacto e qualidade)
- Integração Contínua
  - Demandas são desenvolvidas como incrementos
  - Integradas ao produto e entregues o quanto antes
  - Ciclo: *integrate-build-test*
- Revisões e Inspeções
  - Não são formais no contexto ágil (*peer review*)



# Práticas Ágeis

## Garantia da Qualidade

- Gerência de Configuração
  - Consistência entre versões de artefatos → Integridade do sistema
  - Mecanismos de controle de versão e mudança





# Práticas Ágeis/Lean

## Liberações (*Releases*) de Software



- Entregas incrementais ao cliente
  - Entrega contínua como incremento da versão anterior
  - Cliente recebe “valor” de forma contínua
- Separação em liberações internas e externas
  - Interna: Software com qualidade para ser liberado, mas não são por questões estratégicas
  - Externa: Software liberado

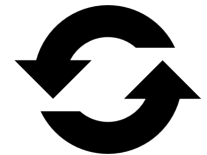


# Práticas Ágeis

## Planejamento do Projeto



- Iterações curtas
  - Iterações definidas com base no feedback do cliente
  - Trabalho finalizado e disponibilizado ao cliente de forma rápida
  - *Sprints* (Scrum)



# Práticas Ágeis

## Planejamento do Projeto



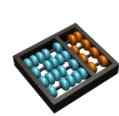
- Planejamento adaptativo com requisitos/estórias de usuário de maior prioridade
  - Lista de prioridades (*backlog*)
  - Features são selecionadas para a próxima iteração
  - Lista de prioridades pode sofrer alterações
  - Evita entrega de features não necessárias

# Práticas Ágeis

## Planejamento do Projeto



- Time-boxing
  - Data de início e fim são fixadas para iterações e projeto
  - Escopo baseado no prazo definido
  - Acelera a criação de valor
- *Planning game*
  - Planejamento de cada iteração organizado em Workshop
  - Envolve cliente, desenvolvedores e gerentes
  - Resolução de conflitos e Implementação da feature correta na iteração correta



# Práticas Ágeis

## Gerenciamento da Equipe



- Desenvolvimento co-locado
  - Facilitar a comunicação de pessoas de diferentes papéis
  - Comunicação direta substituindo documentação adicional
  - Diminui a necessidade de encontros formais (reuniões)
- Equipes inter e/ou multifuncionais
  - Entendimento comum do processo
  - Pelo menos um membro de cada disciplina
  - Diminui espera por documentação

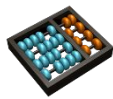


# Práticas Ágeis

## Gerenciamento da Equipe



- 40 horas semanais
  - Evitar horas extras → Pessoas descansadas produzem mais e cometem menos erros
- Reuniões em pé (*Stand-up meetings*)
  - Reuniões diárias
  - Comunicação e reflexão sobre o trabalho realizado e em andamento
  - Curta duração (15 minutos)
  - *O que foi realizado desde a reunião anterior?*
  - *O que deve ser feito até a próxima reunião?*
  - *Quais as dificuldades de atingir os objetivos?*



# Práticas Ágeis

## Gerenciamento da Equipe



- Autonomia: Equipe decide sobre as próprias tarefas
  - Aumenta o comprometimento com a tarefa
  - Alinhado com as competências
  - Atribui responsabilidades



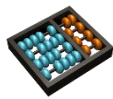
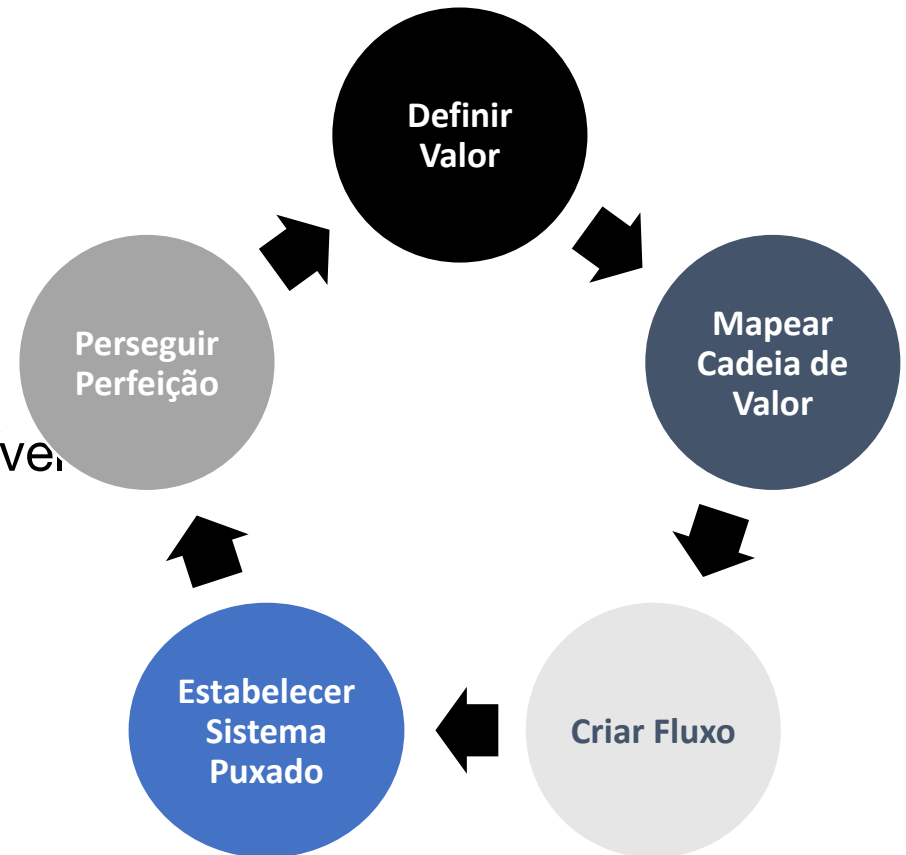
# Desenvolvimento Enxuto (*Lean*)



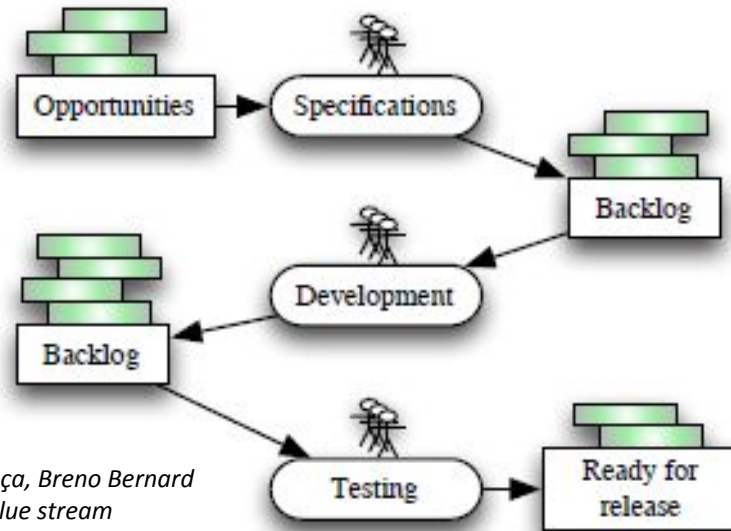
# Desenvolvimento de Software Lean

## Princípios:

1. Eliminar desperdício
2. Amplificar o aprendizado
3. Decidir o mais tarde possível
4. Entregar o mais rápido possível
5. Empoderar a equipe
6. Construir com qualidade
7. Ver o todo (fim-a-fim)

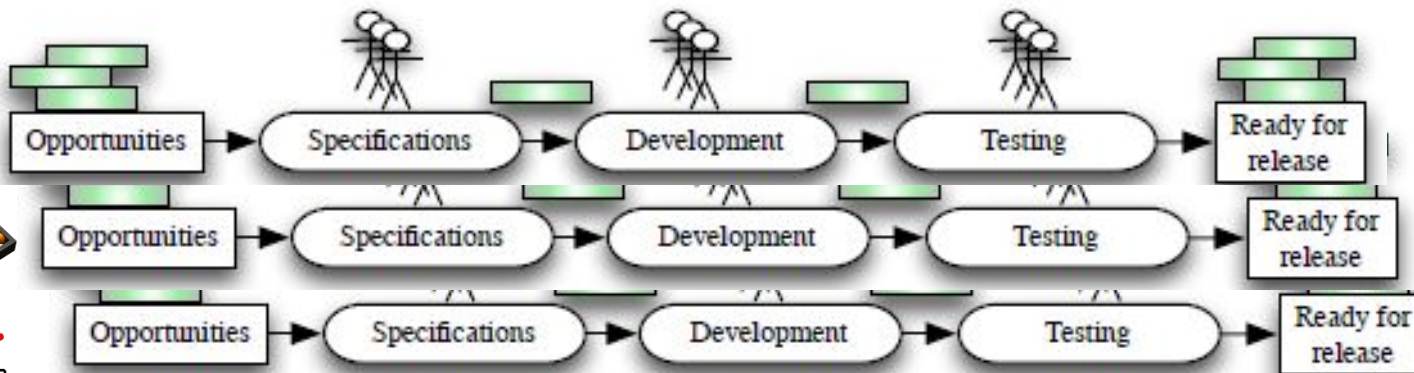


# Sistema Puxado (*Pull System*)



Useful/Needed?

Fonte: Ali, Nauman Bin; Petersen, Kai; de França, Breno Bernard Nicolau. "Evaluation of simulation-assisted value stream mapping for software product development: Two industrial cases." *Information and Software Technology* 68 (2015): 45-61.



Explicit Demand



# Desperdício em Software

Waste Type	Description
Inventory	The cost of storing materials until they are needed. The material might never be used.
Extra Processing	The cost of processing that is unneeded by a downstream step in the manufacturing process. (Sometimes an inefficiency from not seeing the entire process.)
Overproduction	The cost of producing more quantity of components than necessary for the present.
Transportation (of goods)	The cost of unnecessarily moving materials from one place to another place.
Waiting	The cost of waiting for a previous upstream step to finish.
Motion (of people)	The cost of unnecessary picking up and putting things down.
Defects	The cost of rework from quality defects.
Value (added by [1])	The cost of producing goods and services that do not meet the needs of the customer.
Non-utilized Talent (added by [7])	The cost of unused employee creativity and talent.

## Desenvolvimento de Software Lean

← Trabalho parcialmente pronto

← Reaprendizado

← Features extra

← Handoffs

← Atrasos

← Mudança de tarefas/contexto

← Defeitos

# Desperdício em Software

Waste	Description	Observed Causes
Building the wrong feature or product	The cost of building a feature or product that does not address user or business needs.	User desiderata (not doing user research, validation, or testing; ignoring user feedback; working on low user value features) Business desiderata (not involving a business stakeholder; slow stakeholder feedback; unclear product priorities)
Mismanaging the backlog	The cost of duplicating work, expediting lower value user features, or delaying necessary bug fixes.	Backlog inversion Working on too many features simultaneously Duplicated work Not enough ready stories Imbalance of feature work and bug fixing Delaying testing or critical bug fixing Capricious thrashing
Rework	The cost of altering delivered work that should have been done correctly but was not.	Technical debt Rejected stories (e.g. product manager rejects story implementation) No clear definition of done (ambiguous stories; second guessing design mocks) Defects (poor testing strategy; no root-cause analysis on bugs)
Unnecessarily complex solutions	The cost of creating a more complicated solution than necessary, a missed opportunity to simplify features, user interface, or code.	Unnecessary feature complexity from the user's perspective Unnecessary technical complexity (duplicating code, lack of interaction design reuse, overly complex technical design created up-front)
Extraneous cognitive load	The costs of unneeded expenditure of mental energy.	Suffering from technical debt Complex or large stories Inefficient tools and problematic APIs, libraries, and frameworks Unnecessary context switching Inefficient development flow Poorly organized code
Psychological distress	The costs of burdening the team with unhelpful stress.	Low team morale Rush mode Interpersonal or team conflict
Waiting/multitasking	The cost of idle time, often hidden by multi-tasking.	Slow tests or unreliable tests Unreliable acceptance environment Missing information, people, or equipment Context switching from delayed feedback
Knowledge loss	The cost of re-acquiring information that the team once knew.	Team churn Knowledge silos
Ineffective communication	The cost of incomplete, incorrect, misleading, inefficient, or absent communication.	Team size is too large Asynchronous communication (distributed teams; distributed stakeholders; dependency on another team; opaque processes outside team) Imbalance (dominating the conversation; not listening) Inefficient meetings (lack of focus; skipping retros; not discussing blockers each day; meetings running over (e.g. long stand-ups))

**Fonte:** Sedano, Todd, Paul Ralph, and Cécile Péraire. "Software development waste." In *Proceedings of the 39th International Conference on Software Engineering*, pp. 130-140. IEEE Press, 2017.



# Práticas Lean

## Fluxo fim-a-fim (E2E)

- *Value-stream mapping*
  - Criar o mapa de valor agregado atual
  - Analisar o mapa atual
  - Identificar motivos para tempos de espera e propor melhorias
  - Criar o mapa futuro
- Engenheiro Chefe (Líder Técnico)
  - Responsável pelo sucesso e falha da equipe de desenvolvimento
  - Não tem autoridade formal sobre a equipe
  - Habilidades gerenciais e técnicas



# Práticas Lean

## Fluxo fim-a-fim (E2E)

- Gerenciamento de Inventário (*Backlog*)
  - Teoria de filas
  - Teoria de restrições
    1. Identificar a restrição do sistema
    2. Elencar soluções candidatas para remover a restrição
    3. Selecionar a solução com base na restrição
    4. Remover a restrição com a solução selecionada
    5. Voltar ao primeiro passo e verificar a existência de restrições adicionais



# Práticas Lean

## Fluxo fim-a-fim (E2E)

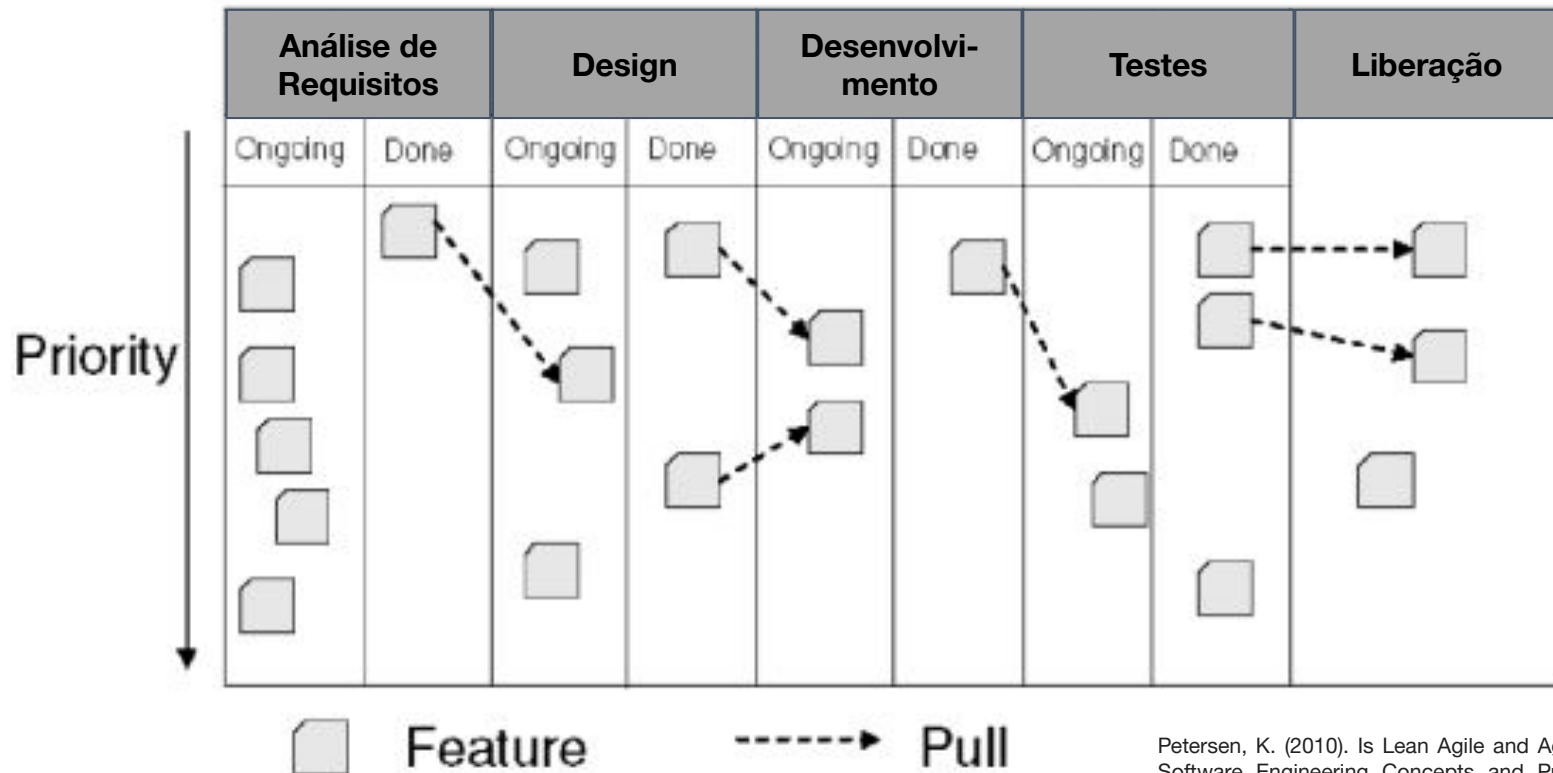
### *Kanban*

- *pull system*: os requisitos são retirados de uma **lista** com base nas **prioridades** definidas pelo cliente conforme a **carga de trabalho** da equipe
- Evita sobrecarga de trabalho
- Evita planejamento a longo prazo
- Dependência complexas entre demandas (*features*)
- **Quadro Kanban**: progresso do desenvolvimento



# Práticas Lean

## Fluxo fim-a-fim (E2E) *Kanban*



Petersen, K. (2010). Is Lean Agile and Agile Lean?. Modern Software Engineering Concepts and Practices: Advanced Approaches, 19.



# Ágil vs Lean: Objetivos


Aspecto	Lean	Ágil
Cliente	Criar valor para o cliente e assim focar somente em atividades que acrescentam valor.	Ter um produto funcionando que atenda às necessidades do cliente.
Velocidade de desenvolvimento	Criação rápida de valor e ciclos de curto tempo.	Entrega contínua de software funcionando.



Petersen, K. (2010). Is Lean Agile and Agile Lean?. Modern Software Engineering Concepts and Practices: Advanced Approaches, 19.



# Ágil vs Lean: Princípios

Agile Principles	Lean Principles
<b>People Management and Leadership</b>	
AP04: Work together	LP01: Eliminate Waste (W5)
AP05: Motivated Individuals	LP05: Respect people
AP08: Sustainable pace	
AP11: Self-organizing teams	
AP06: Face-to-face conversation	LP02: Amplify Learning
<b>Quality of the Product (Technical)</b>	
AP07: Working Software	LP01: Eliminate Waste (W7)
AP09: Technical Excellence	LP06: Build Quality In
<b>Release of the Product</b>	
AP03: Frequent Deliveries	LP01: Eliminate Waste (W1)
	LP02: Amplify Learning
	LP04: Deliver as Fast as Possible
<b>Flexibility</b>	
AP02: Welcome Change	LP03: Defer commitment
<b>Priority of Customer Needs/Value</b>	
AP01: Customer Satisfaction	LP01: Eliminate Waste (W1 to W7)
	LP02: Amplify Learning
	LP03: Defer commitment
	LP04: Deliver as Fast as Possible
	LP05: Respect people
	LP06: Build Quality In
	LP07: See the Whole
AP10: Simplicity	LP01: Eliminate Waste (W1 to W7)
<b>Learning</b>	
AP12: Continuous reflection	LP02: Amplify learning
<b>E2E Flow</b>	
	LP07: See the Whole



Petersen, K. (2010). Is Lean Agile and Agile Lean?. Modern Software Engineering Concepts and Practices: Advanced Approaches, 19.

# Ágil vs Lean: Práticas

- **Similares:** Práticas de [garantia da qualidade](#) e [liberação](#) de software. Um total de 15 práticas compartilhadas
- **Apenas para Ágil:** Cliente Presente, Padrões de Codificação, Posse Coletiva do Código, Planning Game, 40hs semanais, Reuniões em Pé.
  - Apoiam princípios de lean e podem contribuir para esta abordagem
- **Apenas para Lean:** Princípio de [Arquitetura de Baixa Dependência](#) e todos os princípios associados à perspectiva E2E. Potencialmente fortes para apoiar desenvolvimento ágil.



# Ágil vs Lean

- Os princípios e práticas do desenvolvimento ágil não foram **individualmente** inovadores na Engenharia de Software
- Porém, a forma como estes foram combinados em um *framework* teórico e prático, **sim**.



Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6), 1213-1221.

# BizDev e DevOps

# Agilidade → Continuidade → +Valor



*O que acontece quando agilizamos o desenvolvimento?*

*Como aumentar entrega de valor?*

# BizDev

- Interface entre **Negócios** e **Desenvolvimento**
- **Papéis** e **Responsabilidades** mudam de acordo com a metodologia adotada
  - Práticas também variam



# BizDev

## Papéis e Responsabilidades

- **eXtreme Programming:**
  - **Cliente:**
    - Escrever histórias
    - Escrever testes funcionais
    - Definir quando um requisito está satisfeito
    - Definir prioridade para requisitos





# BizDev

## Papéis e Responsabilidades

- **Scrum:**
  - **Product Owner:**
    - Tornar estórias (*product backlog*) visíveis
    - Definir estimativas para itens do *backlog*
    - Traduzir itens do *backlog* em *features* a serem desenvolvidas
    - Responsável pelo product backlog
  - **Time:**
    - Participar da criação e manutenção de *product backlog*
  - **Cliente**
    - Participa na criação e manutenção de *product backlog*

# BizDev

## Papéis e Responsabilidades

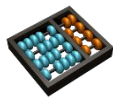
### Analista Funcional - ProductOwner - Sênior

A Combinar | Campinas - SP (1)

Empresa Confidencial

Análise e desenvolvimento de planos e roteiros de testes Análise e Levantamento de Melhorias e Manutenções: Entendimento de Demandas de Melhorias, Especificação de Testes, Execução de planos de testes. Chamados técnicos: Atendimento aos chamados diários, Entendimento da demanda e validação das entregas e report ao especialista responsável pelo produto Suporte aos usuários: Dar suporte a dúvidas referentes aos Produtos de sua responsabilidade. Ensino Superior completo na área de Tecnologia da Informação. Conhecimentos e praticas em Agile (Scrum), TDD (Diferencial), VSTS online. Linguagens de marcação: Html 5, Css 3. Bibliotecas JS: jQuery, Ajax, AngularJS, Bootstrap, NodeJS será um diferencial. Frameworks / CMS: Frameworks .Net 3.5, .Net 4.0, ter experiencia em gerenciadores de conteúdo é um diferencial (ex: Drupal). SGBD: PL SQL - Desenvolvimento de queries e execução para validação direta em banco de dados. Ferramentas de testes unitários, Design responsivo e cross- browser, POO, Versionamento GIT.

*Tem algo de estranho aqui?*



# BizDev

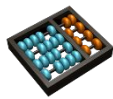
## Papéis e Responsabilidades

### Analista Funcional - ProductOwner - Sênior

A Combinar | Campinas - SP (1)

Empresa Confidencial

Análise e desenvolvimento de planos e roteiros de testes Análise e Levantamento de Melhorias e Manutenções: Entendimento de Demandas de Melhorias, Especificação de Testes, Execução de planos de testes. Chamados técnicos: Atendimento aos chamados diários, Entendimento da demanda e validação das entregas e report ao especialista responsável pelo produto Suporte aos usuários: Dar suporte a dúvidas referentes aos Produtos de sua responsabilidade. Ensino Superior completo na área de Tecnologia da Informação. Conhecimentos e praticas em Agile (Scrum), TDD (Diferencial), VSTS online. Linguagens de marcação: Html 5, Css 3. Bibliotecas JS: jQuery, Ajax, AngularJS, Bootstrap, NodeJS será um diferencial. Frameworks / CMS: Frameworks .Net 3.5, .Net 4.0, ter experiência em gerenciadores de conteúdo é um diferencial (ex: Drupal). SGBD: PL SQL - Desenvolvimento de queries e execução para validação direta em banco de dados. Ferramentas de testes unitários, Design responsivo e cross- browser, POO, Versionamento GIT.



# BizDev

## Papéis e Responsabilidades

- **Crystal:**
  - **Business expert:**
    - Possuir conhecimento sobre o contexto do negócio
    - Cuidar do plano de negócios
    - Perceber quais requisitos estão mudando e quais estão estáveis
  - **Business analyst-designer:**
    - Comunicar e negociar com usuários
    - Especificar requerimentos e interfaces a partir de negociações com usuários
    - Rever design



# BizDev

## Papéis e Responsabilidades

- **Feature Driven Development:**
  - **Domain expert (usuário, cliente, patrocinador, business analyst):**
    - Possuir conhecimento sobre quais objetivos os requisitos devem alcançar
    - Passar esse conhecimento para os desenvolvedores, para garantir que eles entreguem um sistema competente



# BizDev

## Papéis e Responsabilidades

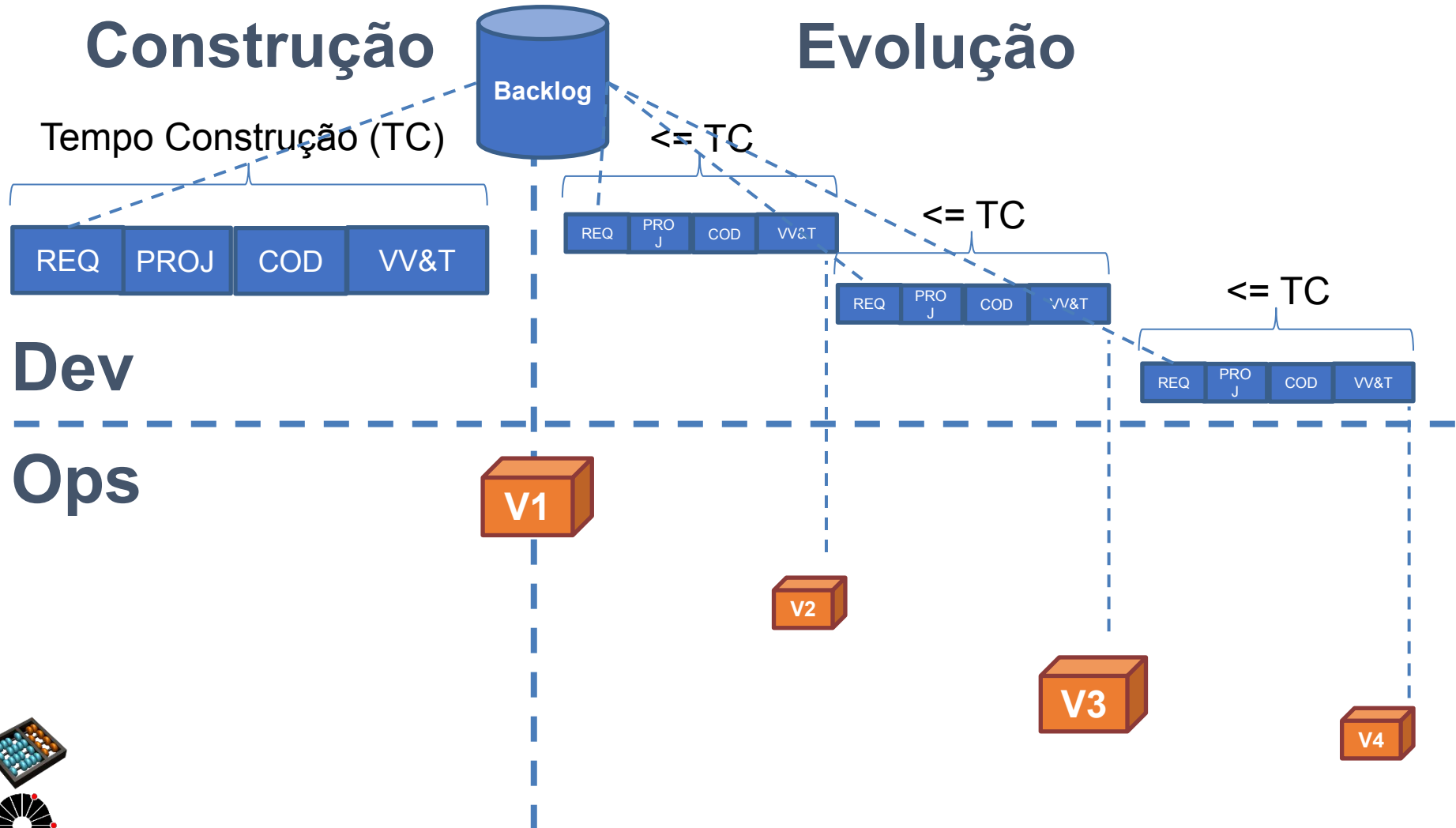
- **Dynamic System Development Method**
  - **Ambassador User:**
    - Trazer conhecimento da comunidade de usuários para o projeto
    - Disseminar informação sobre o projeto para usuários
    - Garantir que quantidade necessária de feedback é obtida
  - **Visionary:**
    - Garantir que requisitos essenciais são descobertos na fase inicial do projeto
    - Garantir que projeto sempre rume em direção ao cumprimento desses requerimentos essenciais



# DevOps

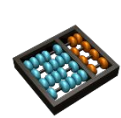
## Construção

## Evolução



# Motivações para adotar DevOps

- Pressão Externa
  - Ex.: Necessidade de entregar mudanças frequentes; modelo SaaS.
- Comunicação Ineficaz
  - Ex.: Problemas de comunicação entre desenvolvimento e operações; Falta de feedback na utilização dos sistemas.
- Estrutura e Política Organizacional
  - Ex.: Ambientes de produção burocráticos; Desenvolvimento lidando com tarefas operacionais; falta de alinhamento entre as equipes.





# Motivações para adotar DevOps

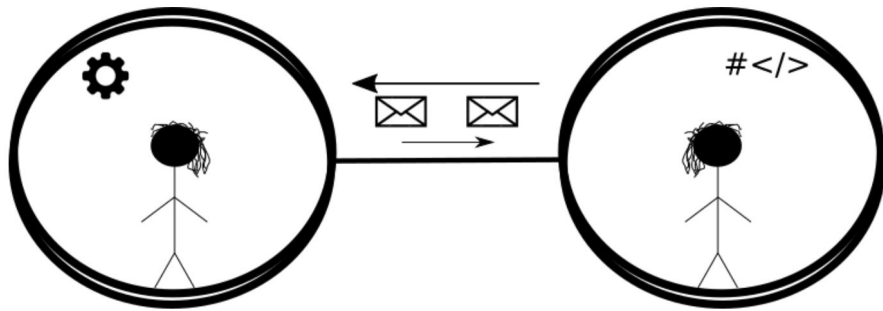
- Demandas por qualidade
  - Ex.: Sistemas complexos demandando cada vez mais características não funcionais; Falta de escalabilidade; Dificuldade em evoluir sistemas em larga escala e fortemente acoplados.
- Processo de liberação
  - Ex.: Agilidade no desenvolvimento expondo gargalos na operação; Efeitos imprevisíveis na implantação; Medo de mudar o sistema após um release estável.
- Questões sociotécnicas
  - Ex.: Diferenças culturais entre desenvolvimento e operações causando conflitos; Amplificação de problemas de engenharia de software em cenários distribuídos.

# Afinal, o que é DevOps?

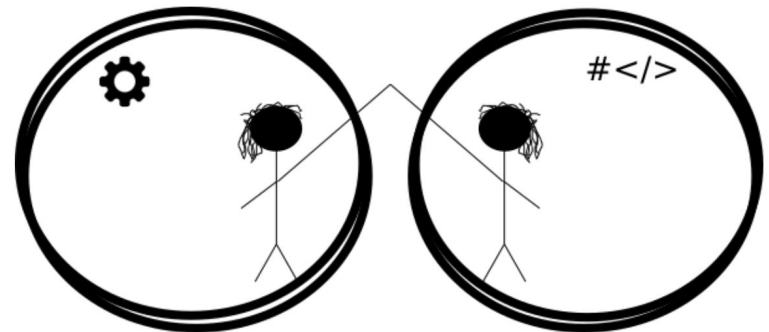
*“DevOps é um neologismo que representa um **movimento** de profissionais de TIC abordando uma **atitude diferente** em relação à **entrega** do software por meio da **colaboração** entre as funções de **desenvolvimento e operação** de sistemas de software, com base em um conjunto de **princípios e práticas**, tais como cultura, automação, medição e compartilhamento.”*



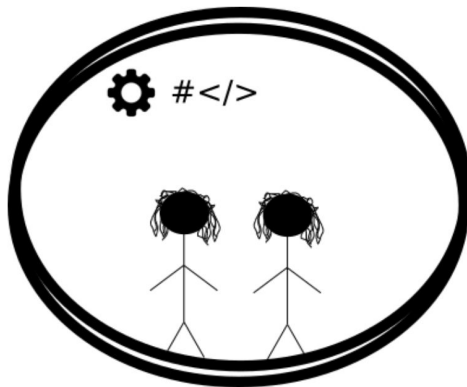
# Organizações DevOps



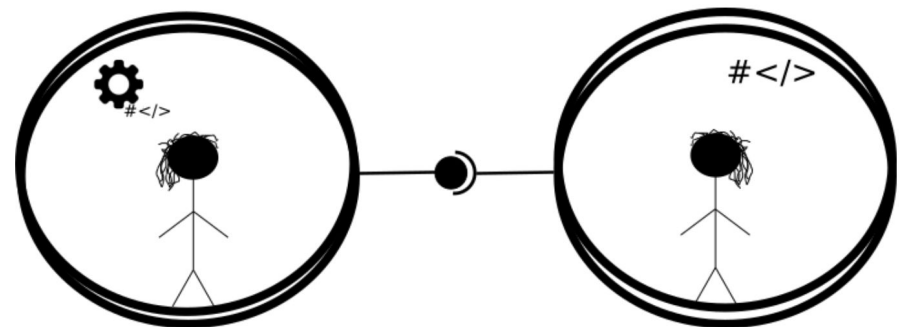
Departamentos Isolados



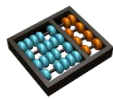
DevOps Clássico



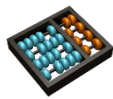
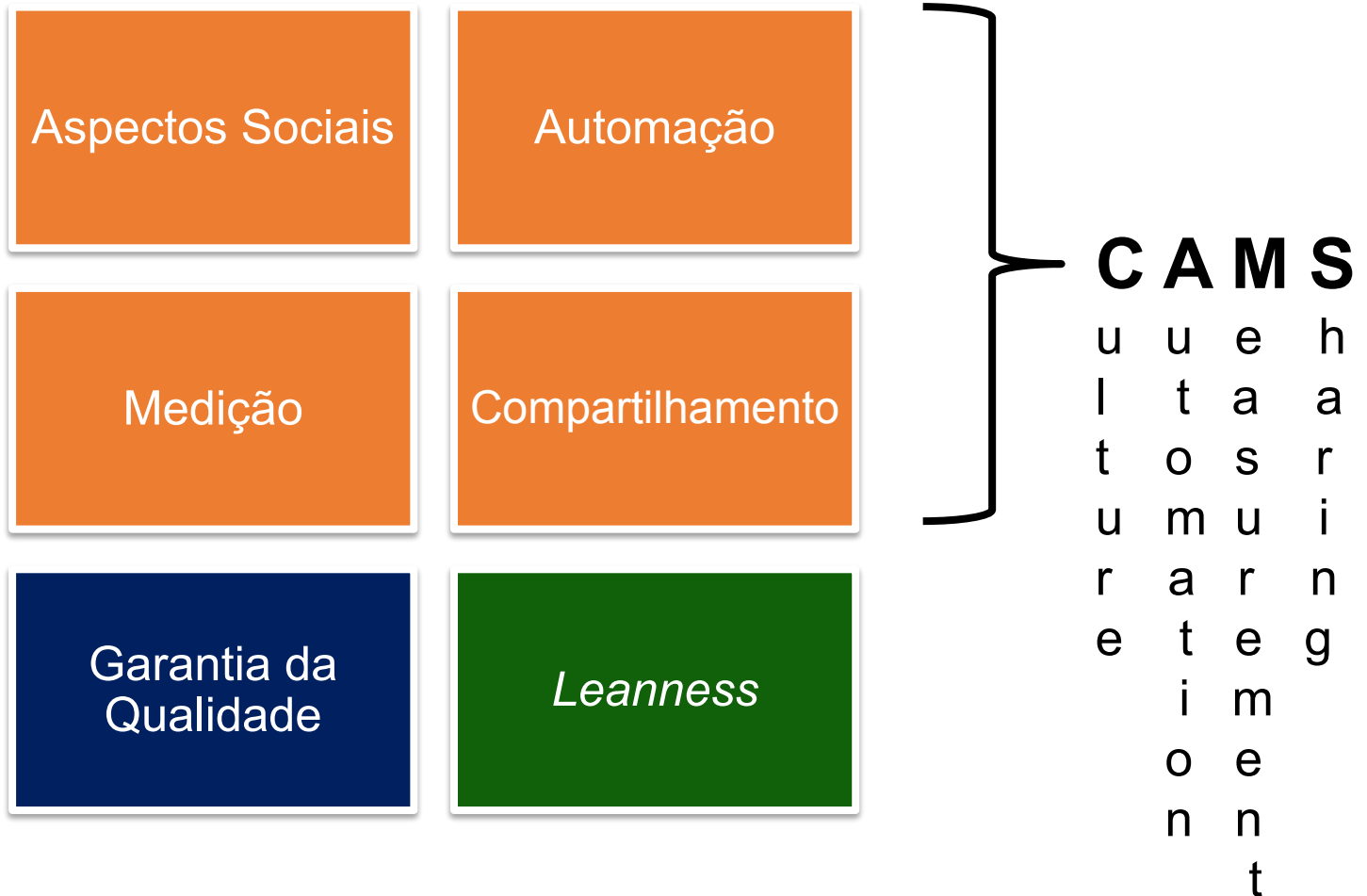
Equipes Multifuncionais



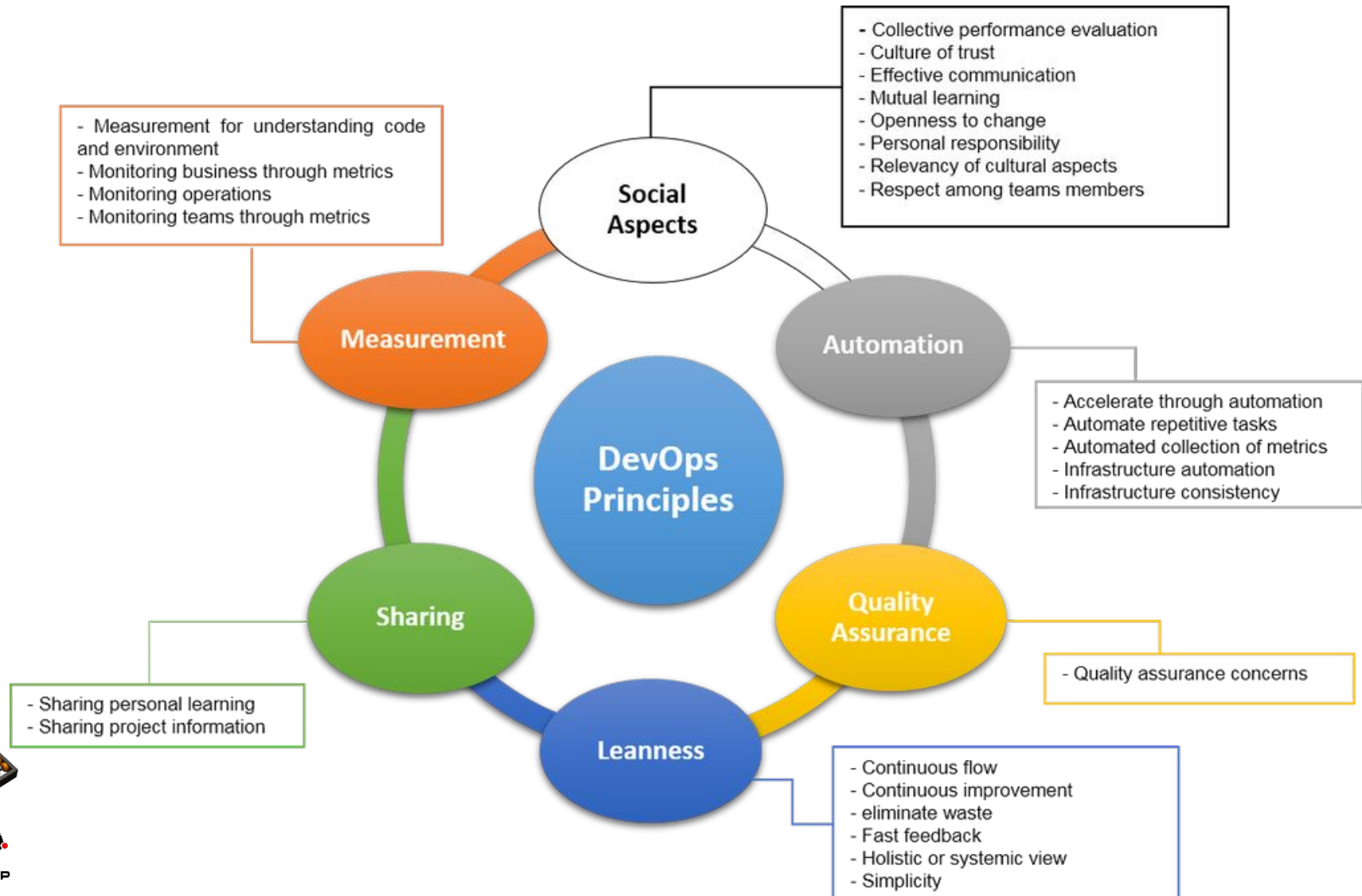
Equipes de Plataforma



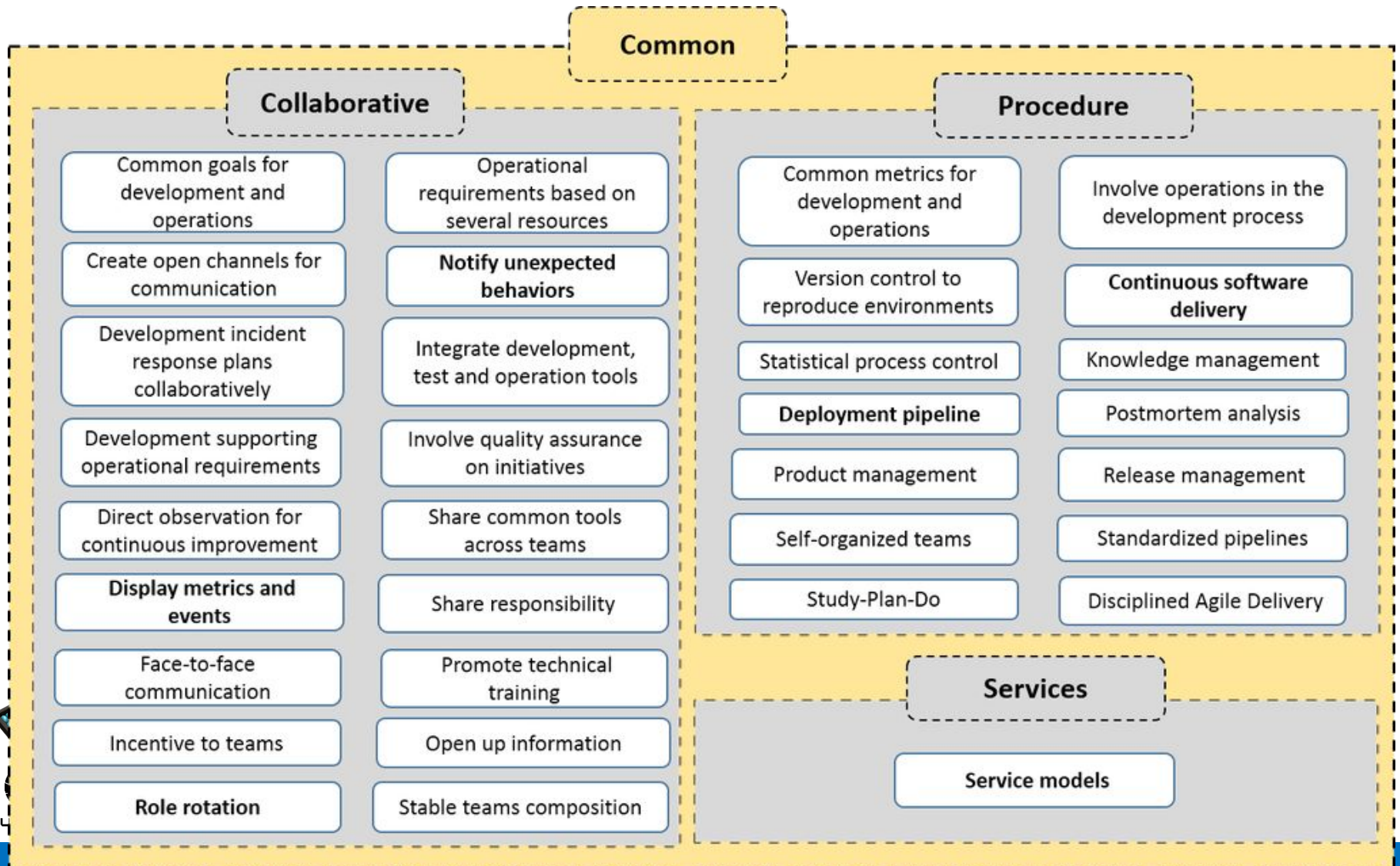
# Princípios DevOps



# Princípios de DevOps



# Práticas de DevOps





# Práticas de DevOps

## Development

**Automated software testing**

Source code peer review

Automatic code security testing

Prioritize defects corrections

**Continuous integration**

Shared code responsibility

**Software configuration management**

Nonfunctional testing

Standard quality profiles

## Operations

Application severity matrix

**Infrastructure configuration management**

Continuous production monitoring

Incremental operation features

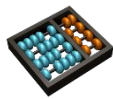
**Virtualization of development and testing environments**

Production data analysis

Behavior-driven operations

**Automated infrastructure provisioning**

**Real-time user monitoring**



# DevOps

## Benefícios

### – Organização

- Ex: Melhoria da eficácia e eficiência operacional; Melhoria da estabilidade de TI.

### – Pessoas

- Ex: Melhoria da comunicação e colaboração entre membros das equipes, e com clientes.

### – Processo

- Ex: Economia financeira e permitir o uso mais eficiente de recursos a partir de práticas de integração e testes contínuos.

### – Produto

- Ex: Problemas expostos mais cedo no ciclo de vida, permitindo antecipação ou prevenção de defeitos.

## Desafios

### – Cultural

- Ex: Organizações resistem a implementar mudanças na forma como as coisas funcionam.

### – Infraestrutura

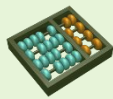
- Ex: Falta de provisionamento automatizado de infraestrutura.

### – Gerenciamento

- Ex: Dificuldade no alinhamento de estratégias e processos entre as equipes de desenvolvimento e operações de forma a atingirem objetivos comuns.

### – Técnicos

- Ex: Dificuldade em introduzir princípios quando os processos existentes precisam de conformidade com padrões e regulações estritas.



UNICAMP