

Desenvolvimento de Software *Lean*



Curso de Verão 2009 - IME/USP

www.agilcoop.org.br

Eduardo Katayama e Hugo Corbucci

Introdução

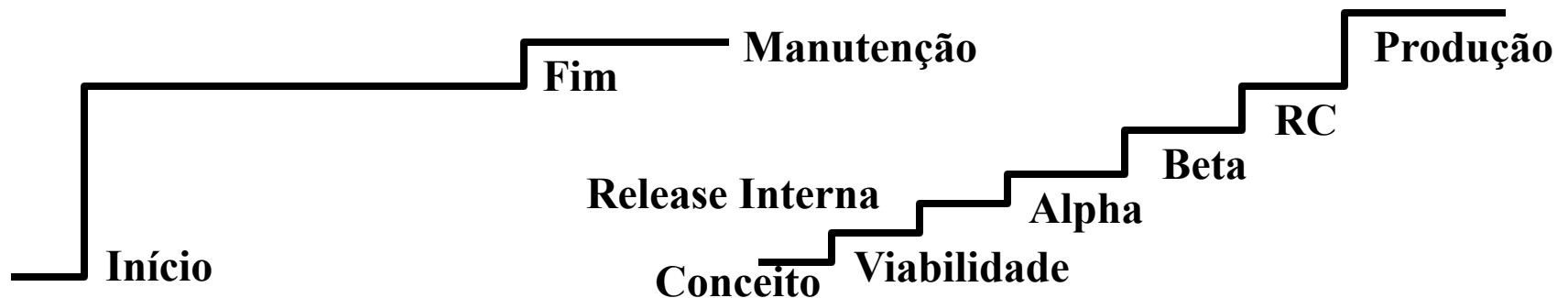
“Desenvolvimento de software é uma cadeia com diversos elos”

-- Kent Beck

- Software é um meio para um fim
- Como o software se encaixa no plano mais amplo (*big picture*)?

Introdução

- Como é o seu processo de desenvolvimento de software?
 - Quem é seu cliente?
 - Produção/manufatura ou novo produto?
 - Gerenciado como projeto ou produto?



Introdução

- Como clientes:
 - “Resolvam completamente meus problemas”
 - “Não desperdicem meu tempo”
 - “Ofereçam exatamente o que eu preciso”
 - “Entreguem valor exatamente onde e quando eu preciso”
 - “Reduzam o número de decisões que eu preciso tomar para resolver meus problemas”

Introdução

- Empresas que aplicam conceitos *Lean*:

The Dell logo, consisting of the word "DELL" in a bold, blue, sans-serif font. The letter "E" is stylized with a white diagonal line.

Produção / Manufatura

The Toyota logo, featuring the word "TOYOTA" in red, uppercase, sans-serif font, preceded by the Toyota symbol (three overlapping ellipses).

Produção / Manufatura +
Desenvolvimento de Produtos

The ZARA logo, consisting of the word "ZARA" in a blue, serif font.

Cadeia de Suprimento

The Google logo, consisting of the word "Google" in its multi-colored, sans-serif font.

Desenvolvimento de Software

Origens do Pensamento *Lean*

- O Sistema de Produção da Toyota:
 - Taiichi Ohno:
 - Fluxo “Just-In-Time”
 - Complexidade vs. Economia de Escala
 - *Autonomation*
 - “*Stop-the-Line*”
 - Shigeo Shingo:
 - Produção sem estoque
 - Trabalho organizado em tarefas pequenas
 - Zero Inspeções
 - “*Mistake-proof*”

Origens do Pensamento *Lean*

- Os valores foram expandidos para outras áreas:
 - Produção *Lean*
 - Manufatura / Operações *Lean*
 - Cadeia de Suprimentos *Lean*
 - Desenvolvimento de Produtos *Lean*
- Desenvolver software é criar um novo produto!
 - Sempre aparece algo novo

Problemas com Software

- Alguns dos motivos:
 - Requisitos que mudam rápido e constantemente
 - Tomada de decisões centralizada
 - Gerenciamento rígido do escopo
 - Práticas “tradicionais” de desenvolvimento (linear)
 - Pouco foco na qualidade do software produzido

Desenvolvimento de Software *Lean*

- Princípios *Lean* aplicados ao software:
 1. Elimine Desperdícios
 2. Inclua a Qualidade no Processo
 3. Crie Conhecimento
 4. Adie Comprometimentos
 5. Entregue Rápido
 6. Respeite as Pessoas
 7. Otimize o Todo

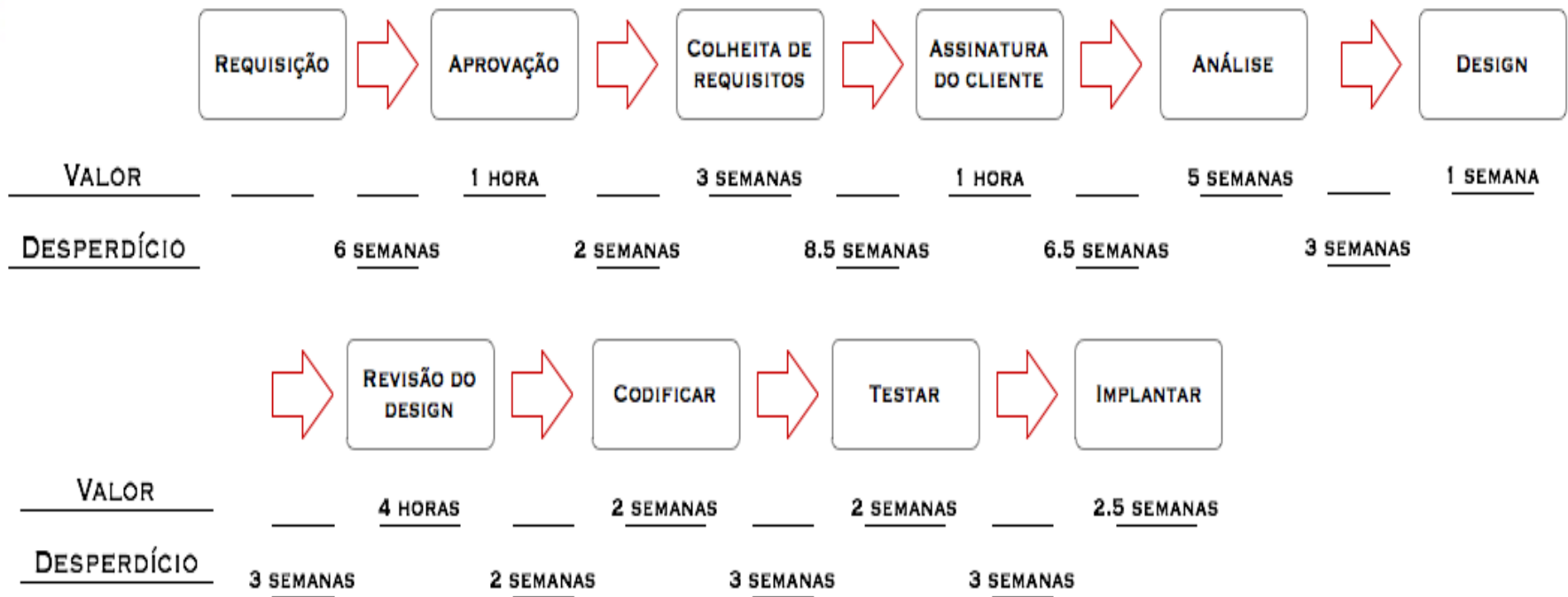
Elimine Desperdícios

“Desperdício é tudo aquilo que não agrega valor ao cliente”

-- Taiichi Ohno

- Este é o principal princípio *Lean*
- Software funcionando é o que vai trazer valor ao cliente
- É preciso aprender a identificar desperdícios

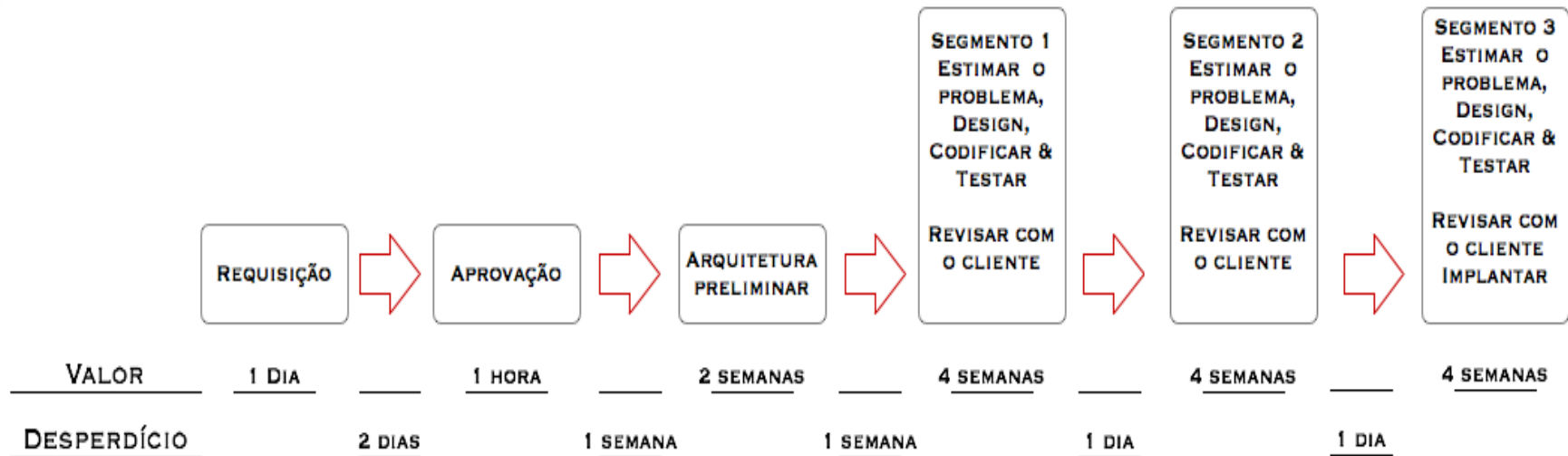
Traditional Value Stream



Tempo Total

Valor: ~17.5 semanas (33%)
Desperdício: ~37 semanas (66%)

Lean Value Stream

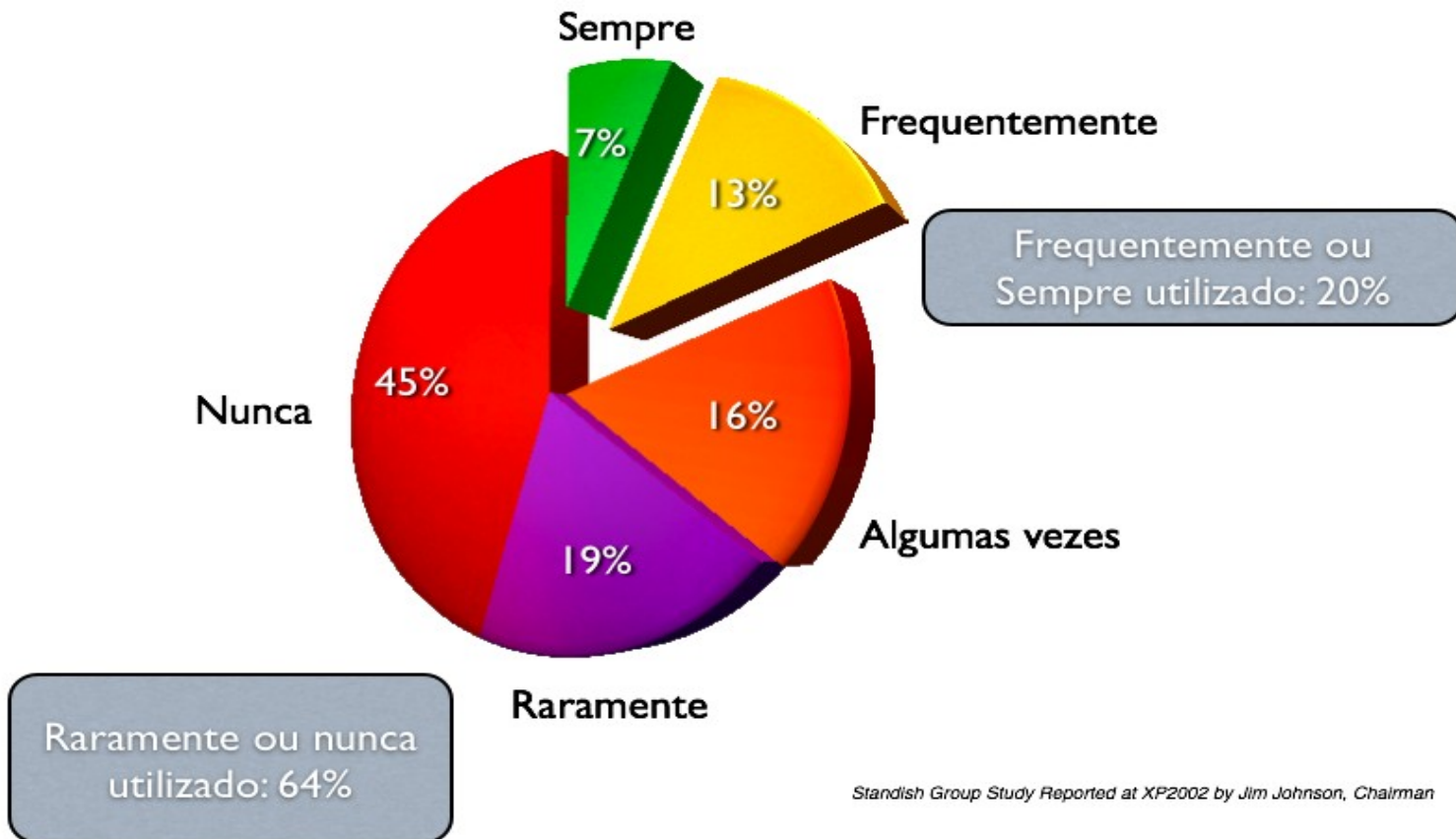


Tempo Total

Valor: ~14.2 semanas (84%)
Desperdício: ~2.8 semanas (16%)

Regra do 80-20

Funcionalidades e funções utilizadas em um sistema típico



Elimine Desperdícios

- Os sete desperdícios de software:
 - Trabalho incompleto (“em-progresso”)
 - Processos a mais
 - Funcionalidades a mais
 - Troca de tarefas
 - *Handoffs*
 - Atrasos
 - Defeitos

Inclua a Qualidade no Processo

*“Inspeccionar para prevenir defeitos é bom;
Inspeccionar para encontrar defeitos é
desperdício”*

-- Shigeo Shingo

- Não deixe os testes para o final
- Ciclos de teste muito longos geralmente gastam mais tempo corrigindo defeitos
- Ao invés de se esforçar para gerenciar defeitos, evite-os

Inclua a Qualidade no Processo

- Prevenindo defeitos com vários tipos de teste:

Testar Perspectiva do Negócio

Teste de requisitos	Testes de Aceitação Interesse de Negócio (Design do Produto)	Testes de Usabilidade Testes Exploratórios	Testes críticos
	Testes de Unidade Interesse do desenvolvedor (Design do código)	Testes de atributos Resposta, Segurança, Escalabilidade	

Testar Perspectiva da Tecnologia

Crie Conhecimento

“Não existe bala de prata”
-- Fred Brooks

- Metáfora: criar vs. preparar uma receita
- Incentive o compartilhamento de conhecimento tácito
- Buscar um processo “padrão” engessa
- O processo deve ser continuamente melhorado

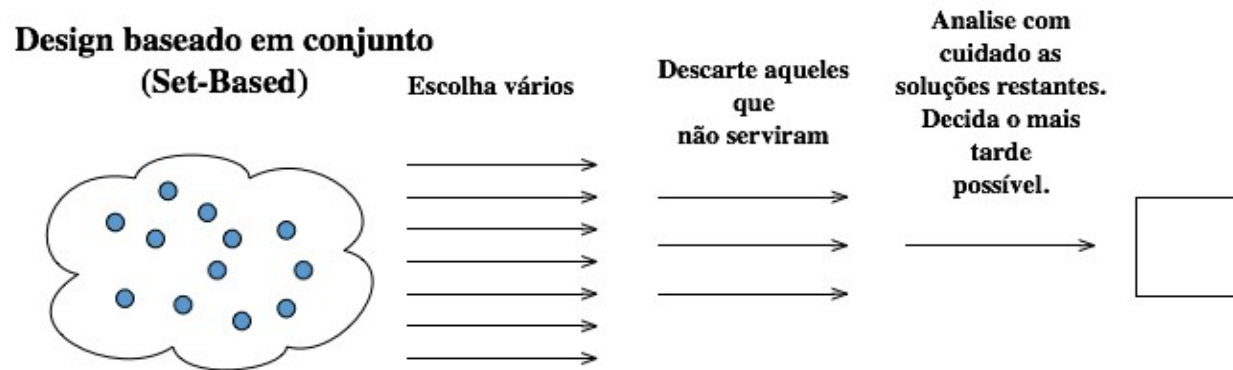
Crie Conhecimento

- Método científico (***Plan-Do-Check-Act***):
 - Enquadre o problema
 - Procure pela raiz do problema
 - Proponha uma solução
 - Implemente a solução
 - Verifique os resultados
 - Analise e adapte seus padrões
- Mito: Predições criam previsibilidade

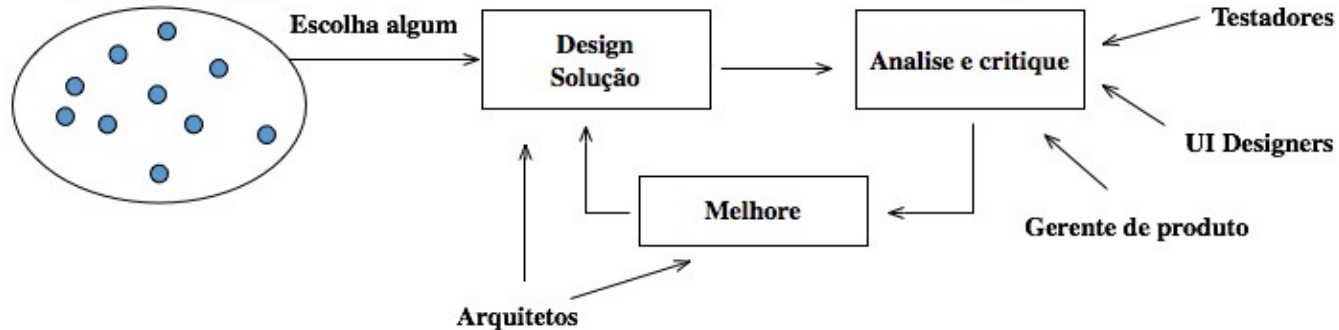
Adie Comprometimentos

- Decisões irreversíveis devem ser tomadas o mais tarde possível (*last responsible moment*)
- Real Options
- É preciso definir o momento da decisão
 - Quando houver mais informação
- Flexibilidade arbitrária também é ruim
- Um bom líder saberá alocar flexibilidade
- Mito: Um plano é um comprometimento

Adie Compromentimentos



Design baseado em pontos (Point-Based)



Adie Comprometimentos

- *Design* baseado em conjunto (*set-based*)
 - Na incerteza, experimente diversas soluções
 - Agende o momento da decisão
 - Sempre haverá uma solução que funciona
 - Paradoxo: Isso não é desperdício!
- Exemplo: Toyota Prius
 - 15 meses do conceito ao lançamento
 - 10 opções de motores híbridos desenvolvidos durante os 4 primeiro meses
 - Motores híbridos viraram item opcional

Entregue Rápido

“A moral da história é que devemos encontrar uma maneira de entregar software tão rápido, que nossos clientes não tenham tempo de mudar de idéia”

-- Mary Poppendieck

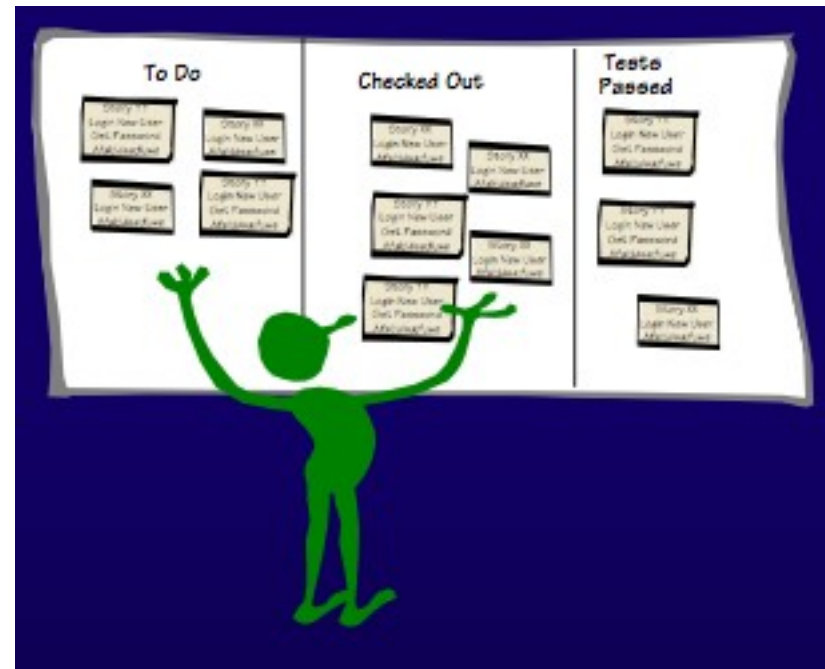
- Competir com base na velocidade traz grande vantagem competitiva
- Mito: Rápido e sujo vs. Lento e limpo

Entregue Rápido

- Utilize sistemas *Pull* em software
- Estabeleça uma cadência regular
- Utilize radiadores de informação
 - Kanban
 - Scrum Meeting
- Utilize pequenas unidades de trabalho
 - Limite o trabalho à capacidade

Entregue Rápido

- Kanban



Entregue Rápido

- Teoria das filas:
 - Tempo do ciclo =
$$\frac{\# \text{ Coisas em processo}}{\text{Taxa Média para completa}}$$
 - Lições:
 - Pequenas unidades de trabalho andam mais rápido
 - Possuir alguns recursos inativos diminui o tempo do ciclo
- Tempo gasto esperando na fila é Desperdício!
- Objetivo: Reduzir o tempo do ciclo!

Respeite as Pessoas

- 3 pilares estão relacionados às pessoas:
 - Liderança
 - Força de trabalho com conhecimento
 - Planejamento e controle baseado em responsabilidade
- Liderança:
 - Grande conhecimento técnico
 - Grande conhecimento do cliente
- Times completos

Respeite as Pessoas

- Pessoas são recursos?
- Papel da gerência é distribuir tarefas e monitorar?
- Exemplo: Planta da GM - NUMMI
- Motivação:
 - Propósito
 - Participação (*belonging*)
 - Segurança
 - Competência
 - Progresso

Respeite as Pessoas

“A verdadeira inovação da Toyota é sua habilidade em usufruir da inteligência dos trabalhadores ‘comuns’”

-- Gary Hamel

- Mova a responsabilidade e o poder de decisão para o nível mais baixo possível
- Teste de Litmus
 - Quando um trabalhador está chateado com algo do trabalho, eles reclama, ignora ou tenta arrumar?

Otimize o Todo

- Exemplo: Zara
- Círculo vicioso #1 no desenvolvimento de software:
 - Cliente pede nova funcionalidade, para ontem
 - Desenvolvedor ouve: Termine isso rápido!
 - Resultado: Mudanças feitas de qualquer jeito no código
 - Resultado: Complexidade do código aumenta
 - Resultado: Número de defeitos no código aumenta
 - Resultado: Tempo para adicionar funcionalidade cresce exponencialmente

Otimizar o Todo

- Círculo vicioso #2 no desenvolvimento de software:
 - Equipe de testes sobrecarregada
 - Resultado: Testes bem após codificação
 - Resultado: Desenvolvedores não recebem *feedback* imediato
 - Resultado: Desenvolvedores criam mais defeitos
 - Resultado: Equipe de teste tem mais trabalho
 - ...
- Mito: Micro-otimização leva à Macro-otimização

Otimize o Todo

- É preciso olhar para o processo todo
- Não adianta resolver os sintomas
- É preciso resolver a causa
- 5 Porquês

Otimize o todo

"Diga-me como serei medido, que eu te direi como me comportarei"

--- Eliyahu M. Goldratt

- Métricas:
 - Medir informação vs. Medir desempenho
 - Cuidado!
 - É fácil medir muitas coisas
 - É fácil medir as coisas erradas

Otimize o todo

- Diminua o número de métricas de desempenho
- Meça para cima:
 - Medidas no nível mais alto que direcionam para o comportamento correto
 - Medidas a nível de time, não de indivíduos
- Incentive a colaboração

Recapitulando...

1. Elimine Desperdícios
2. Inclua a Qualidade no Processo
3. Crie Conhecimento
4. Adie Comprometimentos
5. Entregue Rápido
6. Respeite as Pessoas
7. Otimize o Todo

O Início de um Caminho a Trilhar

- Comece onde está
- Encontre sua maior restrição
- Visualize sua maior ameaça
- Avalie sua cultura
- Treine
- Resolva seu maior problema
- Remova acomodações
- Meça
- Implemente
- Repita o ciclo

Referências

- Livros:
 - 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
 - Jim Johnson, “ROI, It’s Your Job”, Keynote Speech at Third International Conference on Extreme Programming (XP2002), 2002
- Online:
 - leandevlopment @ Yahoo Groups
 - www.poppendieck.com