

## Metodologias e práticas ágeis para programadores



# ELTON D. DE OLIVEIRA

---

**Profissão:** Programador

**Formação:** Engenharia de Computação

**Empresa:** TMax Tecnologia

**GitHub:** eltonoliveira

**E-mail:** elton.douglas.oliv@gmail.com

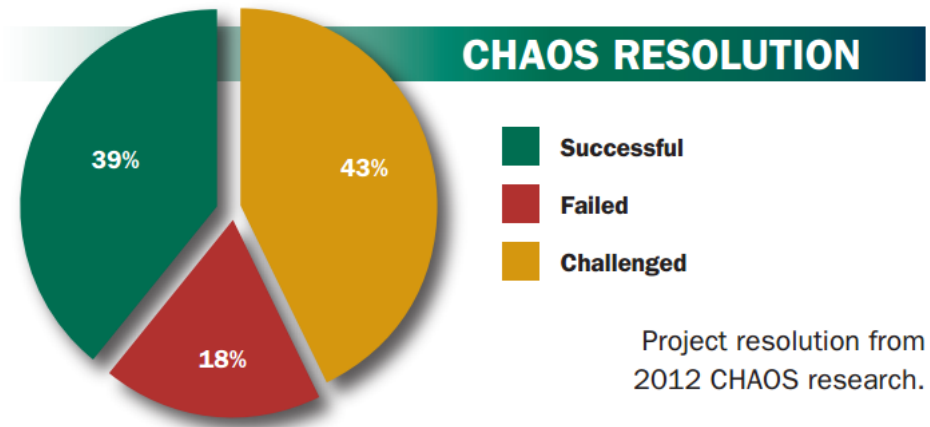
# FATORES RESPONSÁVEIS PELO SUCESSO EM PROJETOS DE SOFTWARE

<b>Factors of Success</b>	<b>Points</b>
Executive management support	20
User involvement	15
Optimization	15
Skilled resources	13
Project management expertise	12
Agile process	10
Clear business objectives	6
Emotional maturity	5
Execution	3
Tools and infrastructure	1

Fonte: Standish Group – Chaos Manifesto 2013

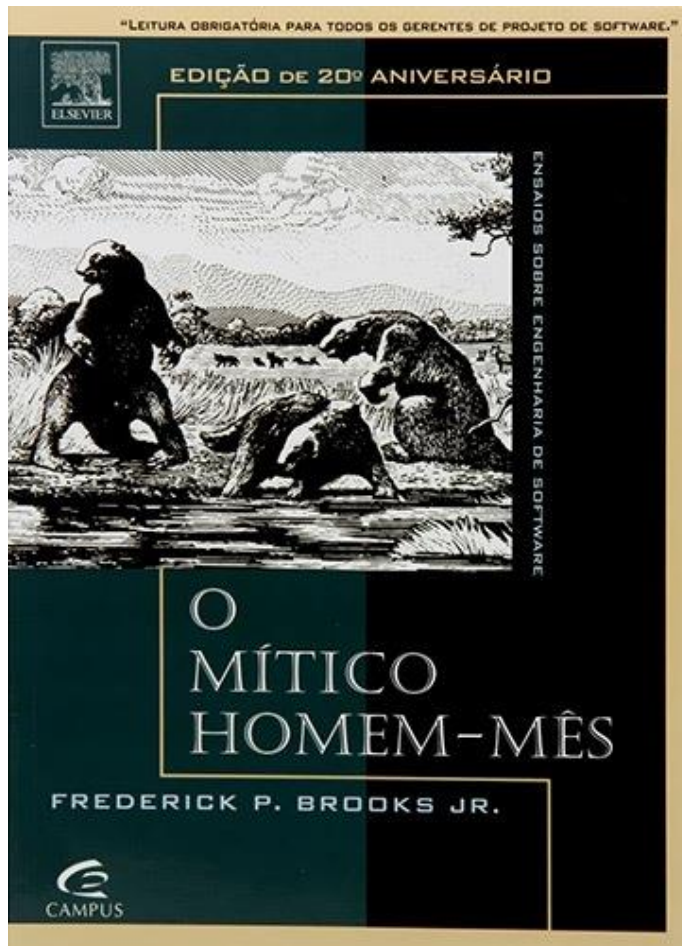
# O STATUS ATUAL SEGUNDO O *STANDISH GROUP*

The 2012 CHAOS results show another increase in project success rates, with 39% of all projects succeeding (delivered on time, on budget, with required features and functions); 43% were challenged (late, over budget, and/or with less than the required features and functions); and 18% failed (cancelled prior to completion or delivered and never used). These numbers represent an uptick in the success rates from the previous study, as well as a decrease in the number of failures. The low point in the last five study periods was 2004, in which only 29% of the projects were successful. This year's results represent a high watermark for success rates in the history of CHAOS research.



*Fonte: Standish Group – Chaos Manifesto 2013*

# MITOS



Enquanto isso, em 1975...

Acrescentar mais programadores reduzirá o atraso na entrega do sistema

Fazer horas extras reduzirá o atraso na entrega do sistema

Essa tecnologia (linguagem, ferramenta, *framework*, etc) irá resolver todos os nossos problemas

# O MITO DA BALA DE PRATA

*“Não só não existem balas de prata à vista como a própria natureza do software torna improvável que venha a existir alguma.”*

**O Mítico Homem-Mês – Frederick Brooks**



# ESTAMOS MELHORANDO?

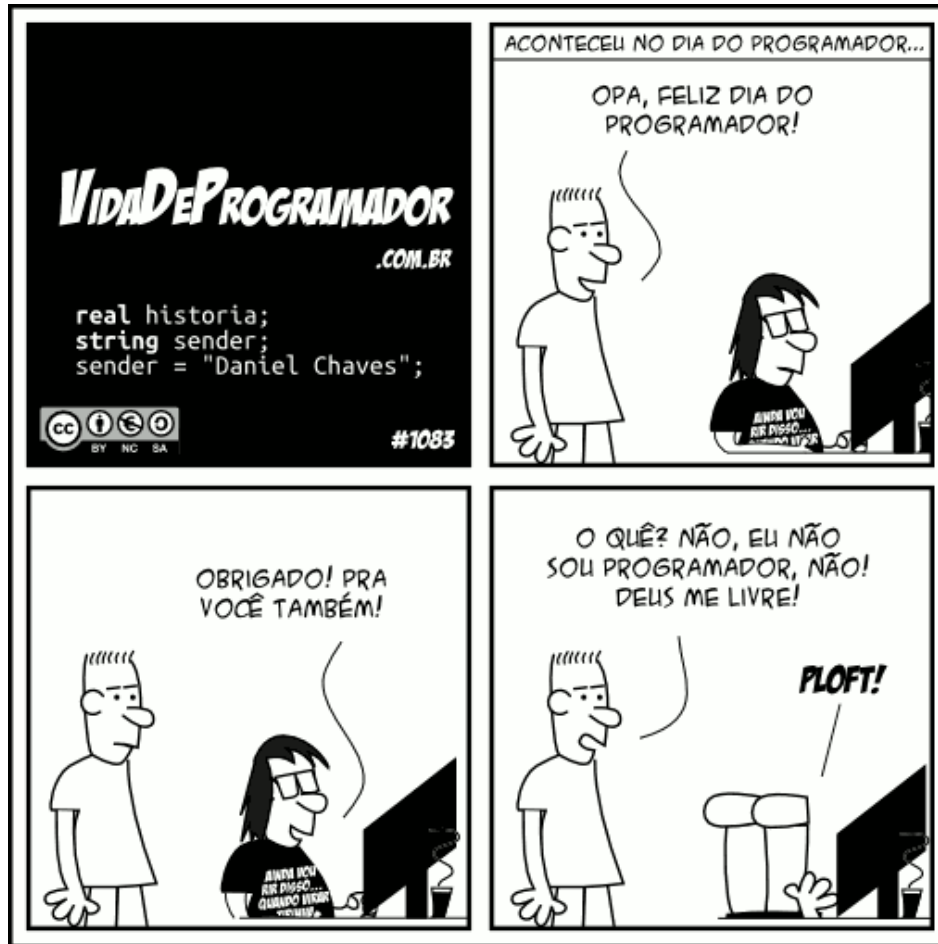
## RESOLUTION

	2004	2006	2008	2010	2012
<b>Successful</b>	29%	35%	32%	37%	39%
<b>Failed</b>	18%	19%	24%	21%	18%
<b>Challenged</b>	53%	46%	44%	42%	43%

Project resolution results from CHAOS research for years 2004 to 2012.

*Fonte: Standish Group – Chaos Manifesto 2013*

# COMO NÓS, PROGRAMADORES, PODEMOS AJUDAR?



Fonte: Vida de Programador

Mais *Know-why* (o porquê) e nem tanto *Know-how* (a habilidade)

Seja profissional

Seja comprometido com a sua carreira

Crie códigos para humanos e não para máquinas



# CÓDIGO LIMPO



Dê nomes auto-explicativos para as suas variáveis, classes e funções.

Busque criar códigos com alta coesão e baixo acoplamento

Evite comentários desnecessários e use-os somente quando fizer algum sentido

Evite duplicação de código

# ALGUNS EXEMPLOS

```
1 <?php
2 class Geral
3 {
4     private $m;//Memória
5
6     /** Método para dividir */
7     public function div($var1, $var2)
8     {
9         if((is_int($var1) || is_float($var1)) &&
10            ((is_int($var2) || is_float($var2)) && ($var2 != 0)))
11         {
12             return $this->m = $var1 / $var2;
13         }
14         else
15         {
16             return NULL;
17         }
18     }
19
20     public function cadastrarUsuario(array $data)
21     {
22         $dbTable = new DbTableUsuarios();
23         return $dbTable->insert($data);
24     }
25
26 }
```

Exemplo 1: ruim

```
1 <?php
2 /**
3  * Classe para representar um calculadora.
4  */
5  * @abstract Classe para a realização das quatro operações básicas:
6  * dividir, Multiplicar, Somar e subtrair.
7  */
8  * @name Calculadora
9  * @category matematica
10  * @author Elton D. de Oliveira <elton.douglas.oliv@gmail.com>
11  * @license GPL http://www.gnu.org/licenses/gpl.html
12  */
13 class Calculadora
14 {
15     /**
16      * @var int|float|object
17      */
18     $memoria;
19     /**
20      * @var int|float|object
21      */
22     const DIVISOR_NAO_SUPORTADO = 0;
23
24     /**
25      * @param int|float $dividendo valor que será dividido
26      * @param int|float $divisor valor pelo qual o dividendo será dividido
27      */
28     * @name dividir
29     * @access public
30     * @return int|float|Exception
31     */
32     public function dividir($dividendo, $divisor)
33     {
34         if(self::_eumDividendoValido($dividendo) && self::_eumDivisorValido($divisor))
35         {
36             return $this->memoria = $dividendo / $divisor;
37         }
38         else
39         {
40             throw new Exception(
41                 "O divisor e o dividendo devem ser valores do tipo int ou float.",
42                 "O divisor deve ser diferente de " . self::DIVISOR_NAO_SUPORTADO .
43                 "MyException::DIVISOR_E_OU_DIVIDENDO_INVALIDOS
44             );
45         }
46     }
47
48     /**
49      * @param int|float $dividendo Valor que será dividido
50      */
51     * @name _eumDividendoValido
52     * @access protected
53     * @return boolean
54     */
55     protected function _eumDividendoValido($dividendo)
56     {
57         return self::_eumIntOuUmFloat($dividendo);
58     }
59
60     /**
61      * @param int|float $divisor valor pelo qual o dividendo será dividido
62      */
63     * @name _eumDivisorValido
64     * @access protected
65     * @return boolean
66     */
67     protected function _eumDivisorValido($divisor)
68     {
69         return self::_eumIntOuUmFloat($divisor) && $divisor != self::DIVISOR_NAO_SUPORTADO;
70     }
71
72 }
```

Exemplo 2: um pouco melhor

# O CODIFICADOR LIMPO



Seja sincero sobre suas estimativas

Deixe a todos os interessados pelo projeto cientes do andamento e de possíveis atrasos

Aprenda a dizer não

Você será contratado pelo seu conhecimento e demitido pelo seu comportamento

# TDD – DESENVOLVIMENTO GUIADO POR TESTES



# MANIFESTO ÁGIL

## Manifesto para Desenvolvimento Ágil de Software

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

**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

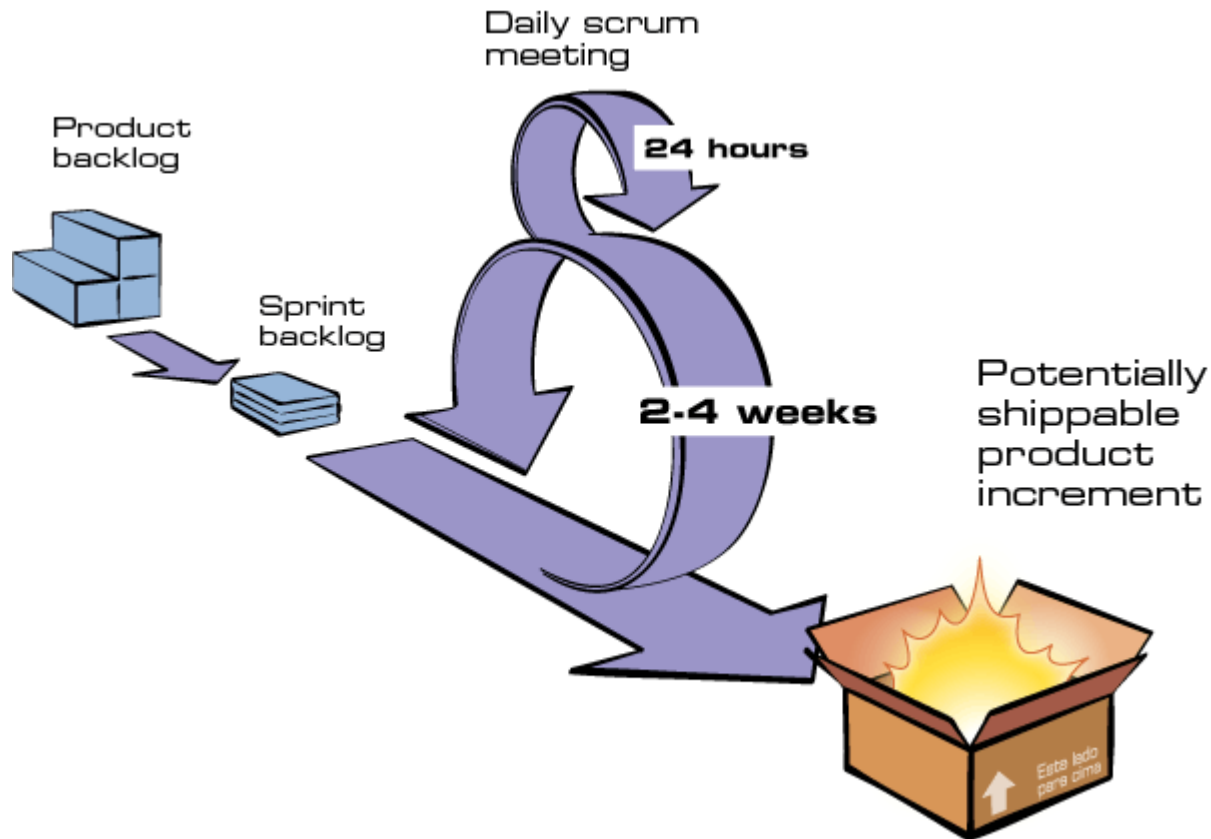
Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

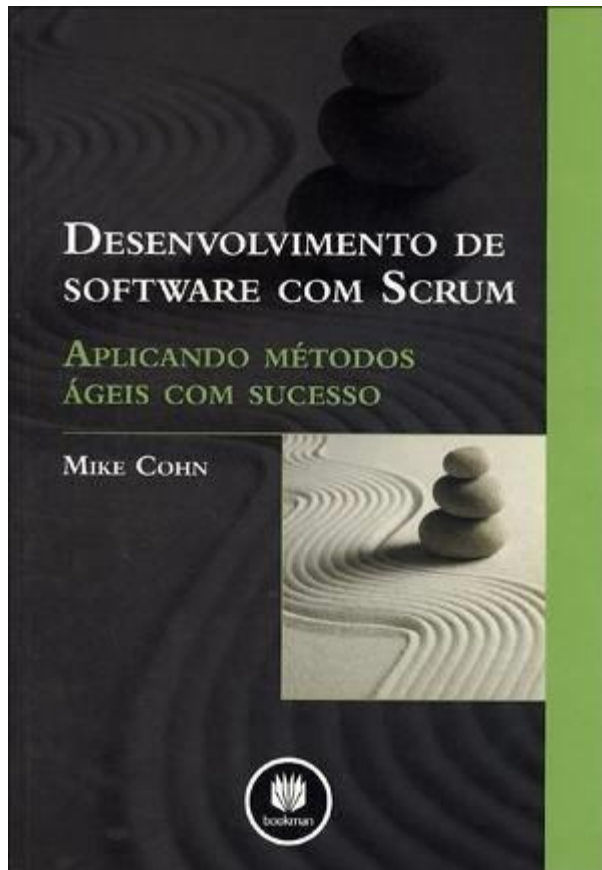
© 2001, os autores acima  
esta declaração pode ser copiada livremente em qualquer formato,  
mas somente integralmente através desta declaração.

Fonte: Manifesto Ágil (<http://www.agilemanifesto.org>)

# O FLUXO DE DESENVOLVIMENTO ÁGIL



# SCRUM



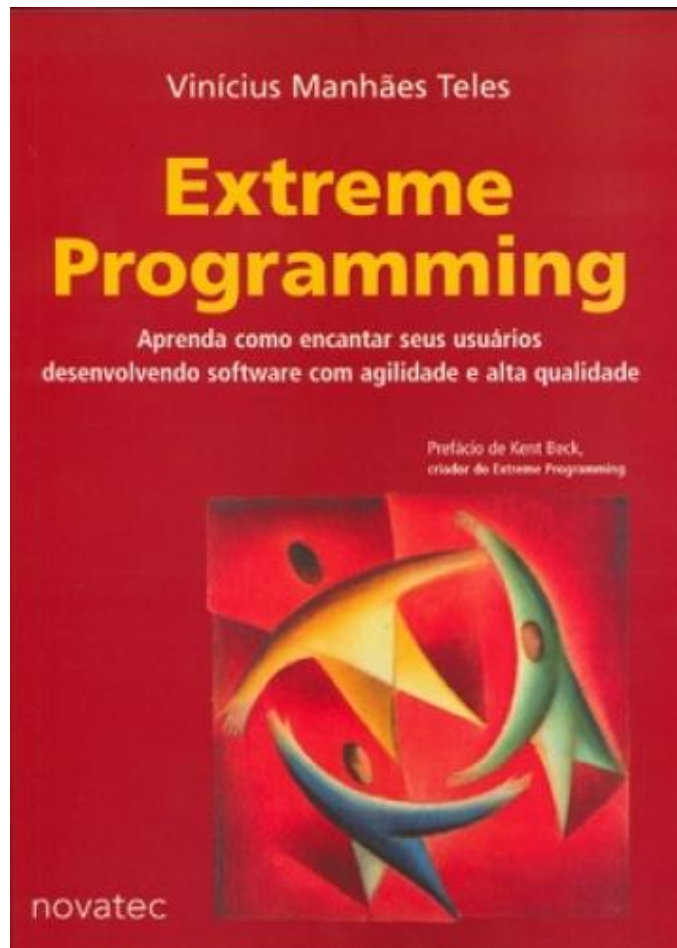
Esteja comprometido e não somente envolvido

Crie e priorize a sua lista de requisitos (*product backlog*)

Planeje e estime pequenas entregas (*sprints*)

Faça entregas contínuas

# EXTREME PROGRAMMING - XP



Programação em par

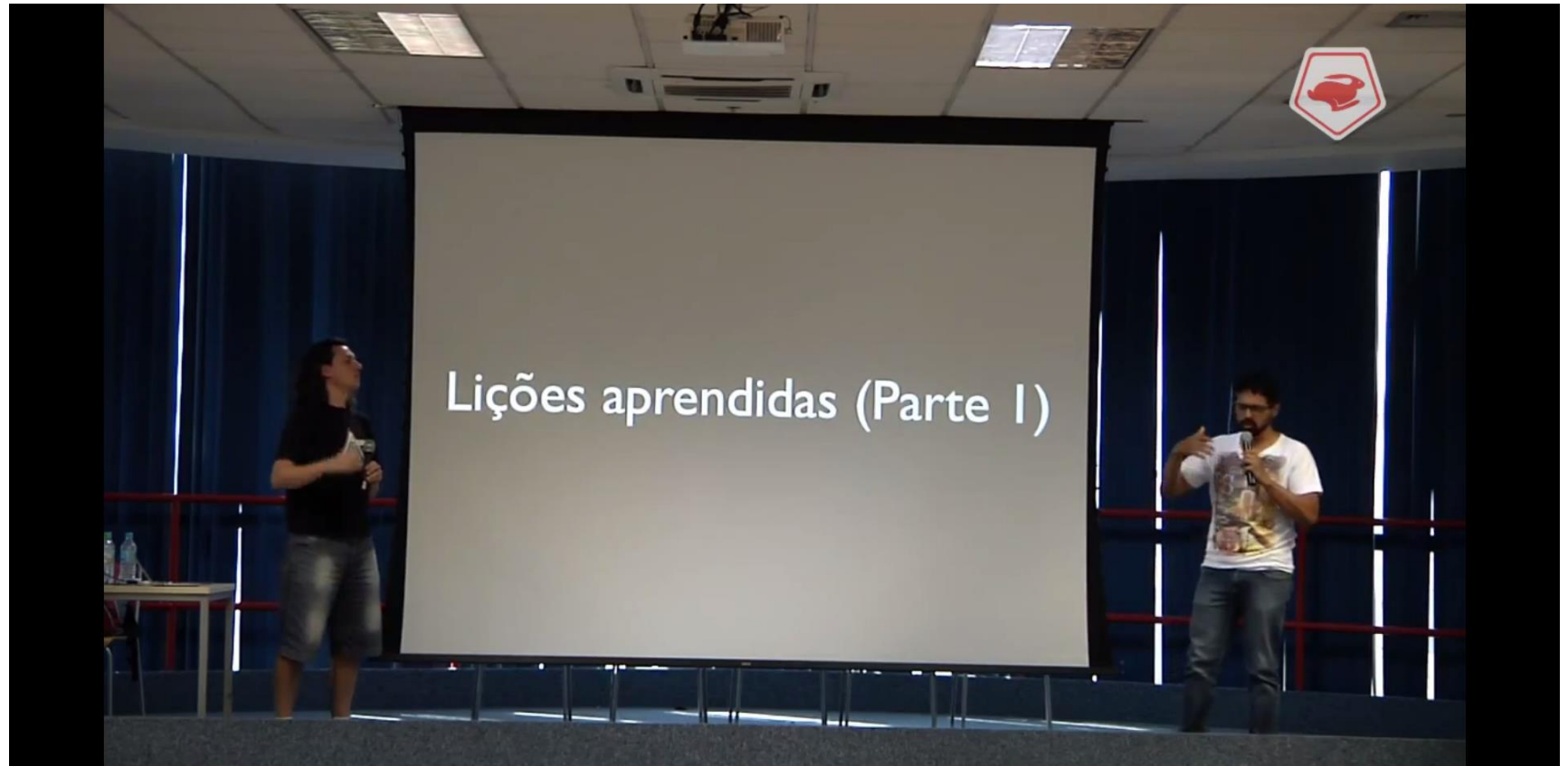
Desenvolvimento guiado por testes

Padronização de código

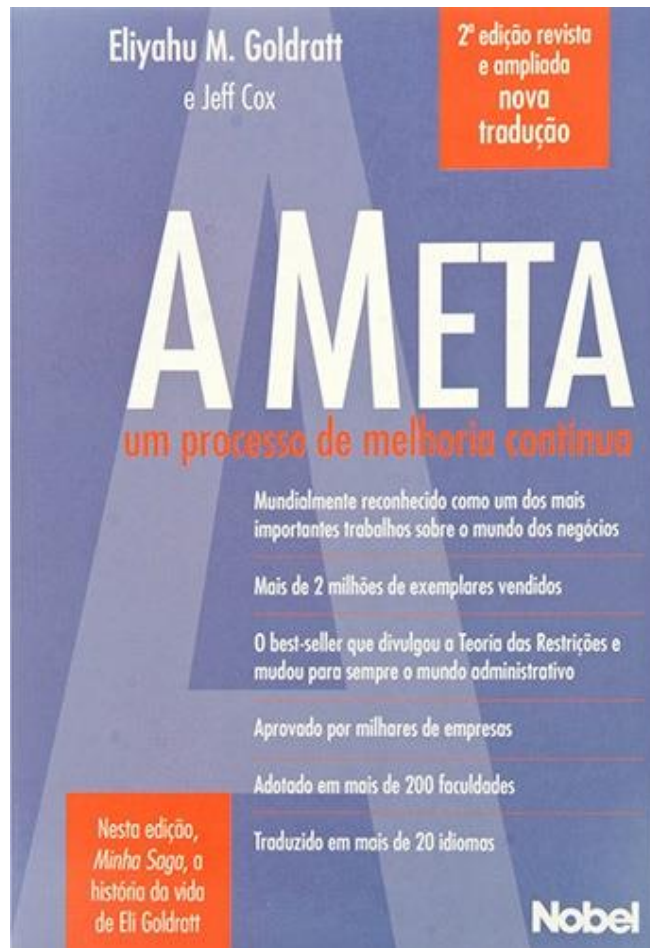
Código coletivo



# EXEMPLO DE UM CASO REAL NA LAMBDA 3



# ALÉM DO SOFTWARE



Identifique qual é a meta da sua empresa, do seu cliente e a sua própria meta

Identifique as restrições do seu processo

Aplique o conceito de melhoria contínua para todas as fases do seu processo

Invista em relações ganha-ganha com seus clientes

# OUTRAS FONTES

- Podcasts e Videocasts
  - Metodologias, palestras, artigos
    - <http://www.infoq.com/br/>
  - Banco de dados:
    - <http://imasters.com.br/perfil/databasecast/>
  - Ruby
    - <http://www.grokpodcast.com/>
  - .Net
    - <http://podcast.dotnetarchitects.net/>
  - Python
    - <http://henriquebastos.net/>
  - PHP
    - <http://www.youtube.com/user/eltonminetto>



# OBRIGADO!

**Currículos:** [elton@tmax.com.br](mailto:elton@tmax.com.br)