『Javaから Rubyへ』より も大切なこと

What's more important than "From Java To Ruby"

角谷 信太郎

s-kakutani@esm.co.jp

(株)永和システムマネジメント

KAKUTANI Shintaro; Eiwa System Management,Inc.; a strong Ruby proponent

島根大学 松江キャンパス; 2007-11-16(金)

本日のお品書き

- ✓ 自己紹介
- ✓ 『JavaからRubyへ』
- ✓ 『JavaからRubyへ』よりも大切なこと: TDD
- ✓ RSpecの紹介とTDD実演

本日のお話の要点

✓ テスト駆動開発は、善きプログラマとしての身だしなみです

✓ 今日のお話をきっかけに、興味をもってらえると嬉しいです

✓ 『JavaからRubyへ』も買ってください :-)

自己紹介

角谷 信太郎

- ✓ (株)永和システムマネジメント
- ✓ テスト駆動開発者
- ✓ 日本Rubyの会理事
- ✓ http://kakutani.com

株式会社永和システムマネジメント

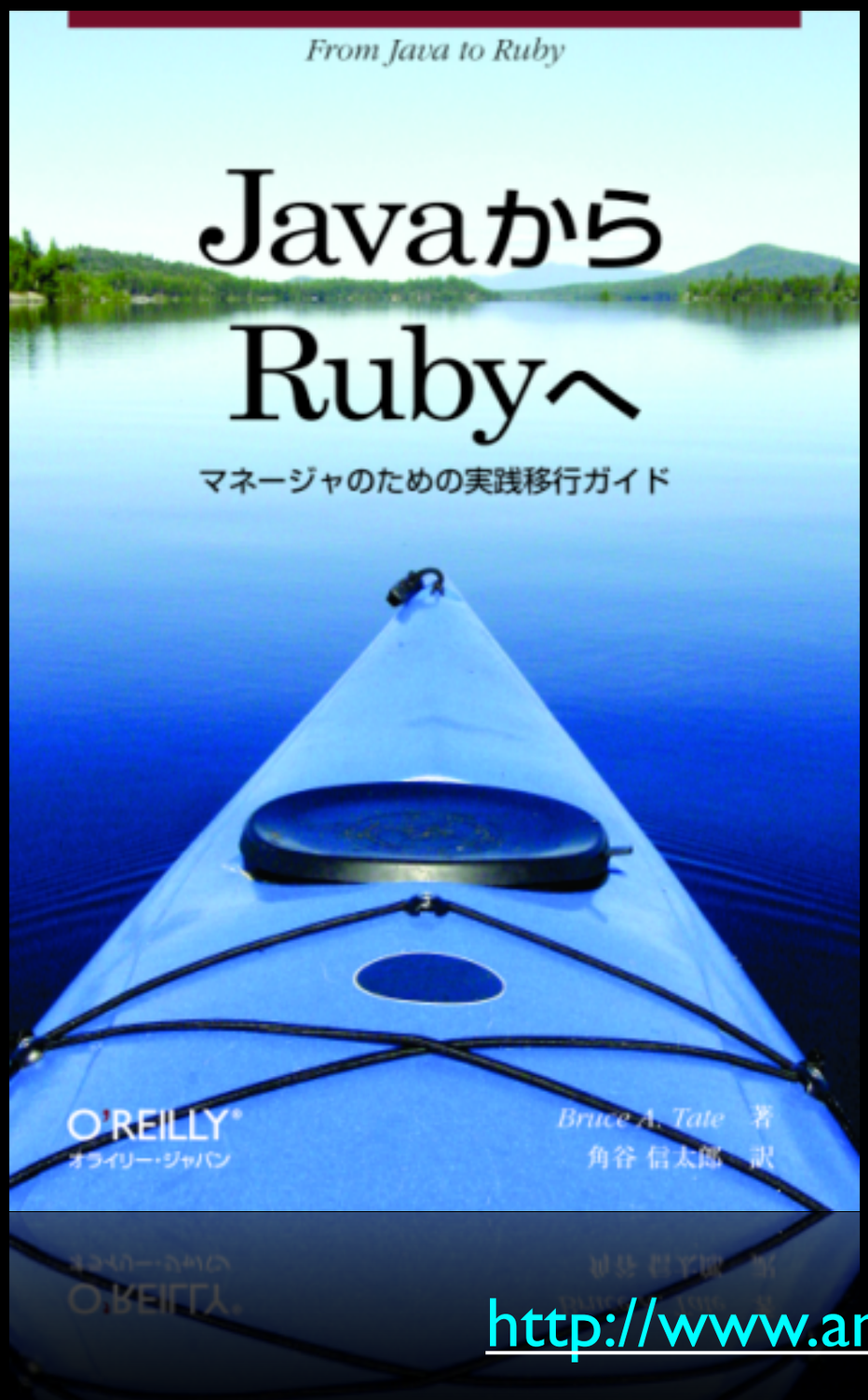
(株)永和システムマネジメント

- 福井本社 (1980年～:28期)
  - 金融,、医療、パッケージ、”オープン系”
- 東京支社 (2002年～)
  - OO、UML、アジャイル開発、Java、Ruby
- 資本金: 6,168万円 (2006年7月)
- 従業員: 230名 (2007年7月)
- 年商: 22.5億円 (2006年7月)
- 関連会社: 

  - (株)チェンジビジョン / (株)アフレル
- コミュニティ:
  - オブジェクト倶楽部(http://www.ObjectClub.jp/)

『JavaからRubyへ』

マネージャのための実践移行ガイド



Bruce A. Tate 著

角谷信太郎 訳

オライリー・ジャパン 発行

第3刷(fixed 78 bugs)

http://www.amazon.co.jp/o/ASIN/4873113202/kakutani-22

『WEB+DB PRESS』 Vol.41

連載 第3回載ってます:

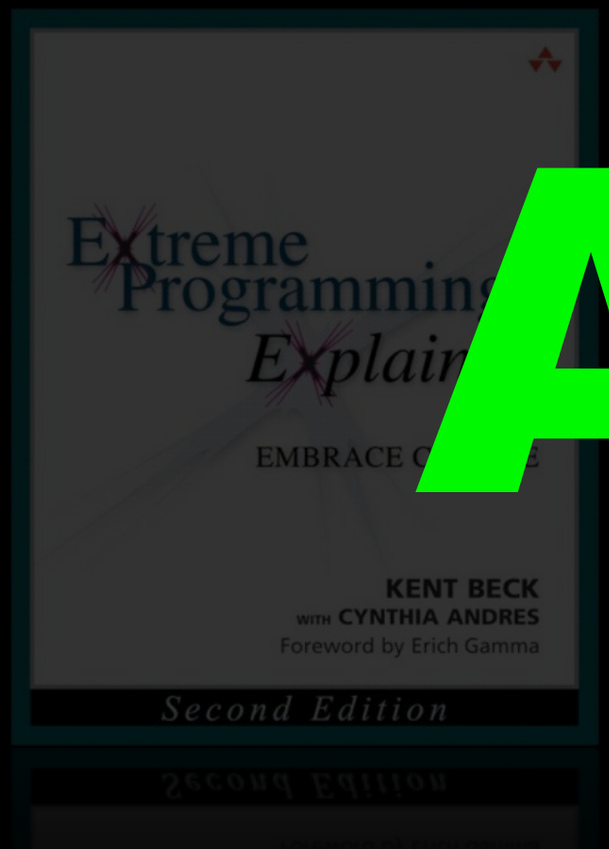
“アジャイル開発者の習慣 ~ acts_as_agile”

角谷信太郎

http://www.amazon.co.jp/o/ASIN/477413256X/kakutani-22

Ruby

×



Agile

eXtreme
Programming

私のミッション・ステートメント

And there's business value in fun - after all motivation is a major factor in programmer productivity.

- Martin Fowler

そして、楽しさにはビジネス価値があります -- 結局、モチベーションこそがプログラマの生産性を左右するのです。

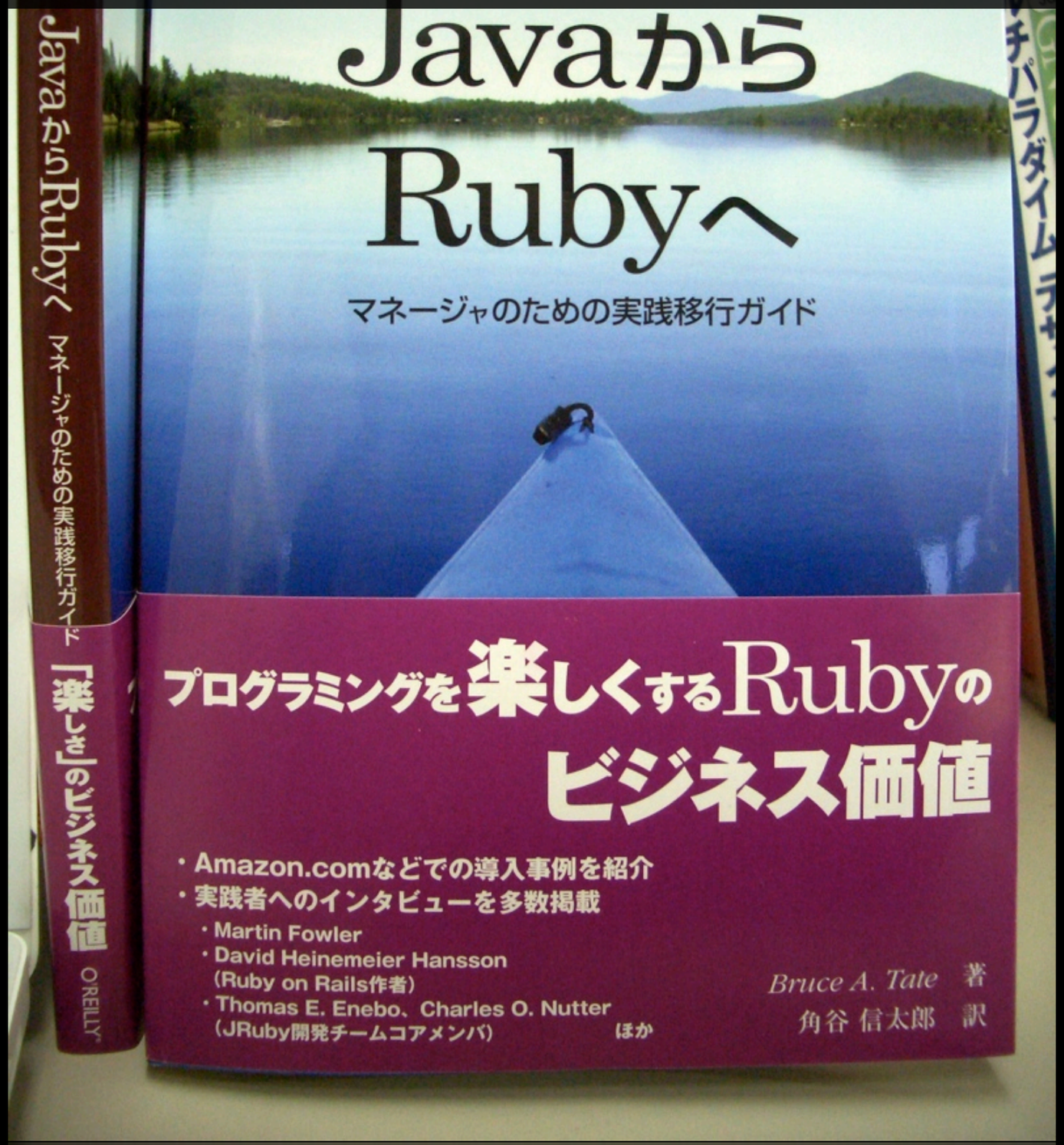
-- マーチン・ファウラー

✓ 達成感

✓ 充実感

✓ 成長感

“楽しさ”のビジネス価値

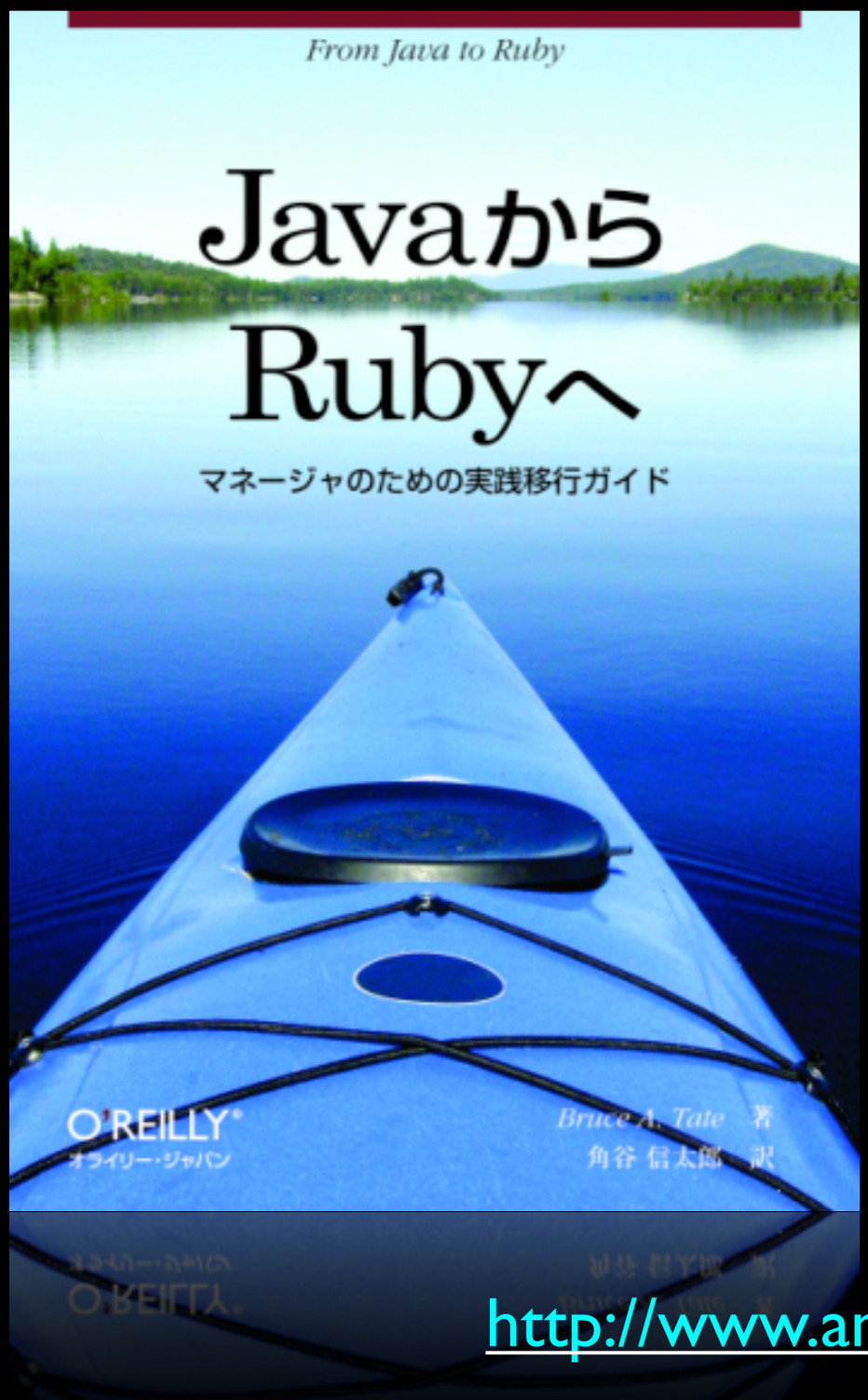

Javaから
Rubyへ
マネージャのための実践移行ガイド
JavaからRubyへ マネージャのための実践移行ガイド
「楽しさ」のビジネス価値
O'REILLY
プログラミングを楽しくするRubyの
ビジネス価値
・Amazon.comなどでの導入事例を紹介
・実践者へのインタビューを多数掲載
・Martin Fowler
・David Heinemeier Hansson
(Ruby on Rails作者)
・Thomas E. Enebo、Charles O. Nutter
(JRuby開発チームコアメンバ) ほか
Bruce A. Tate 著
角谷 信太郎 訳

よろしく

お願いします

『JavaからRubyへ』

マネージャのための実践移行ガイド



Bruce A. Tate 著

角谷信太郎 訳

オライリー・ジャパン 発行

第3刷(fixed 78 bugs)

http://www.amazon.co.jp/o/ASIN/4873113202/kakutani-22

翻訳の意図

- ✓ 北米のソフトウェア開発現場でのRuby受容の紹介
- ✓ 思考フレームワークの提供
  - ✓ 新技術の組織への適用
  - ✓ JavaとRubyはひとつの具体例
- ✓ 変化と“信頼貯金”について

情報収集
Gather Information

“苦痛”の確認
Validate Pain

→

評価の確立
Establish Rewards

Stop!

Stop!

限定的な展開
Limited Deploy

パイロット実施
Build Pilot

→

限定的な展開
Limited Deploy

Stop!

Stop!

広範な展開
Broad Deploy

統合
Integrate

→

普及
Ramp Up

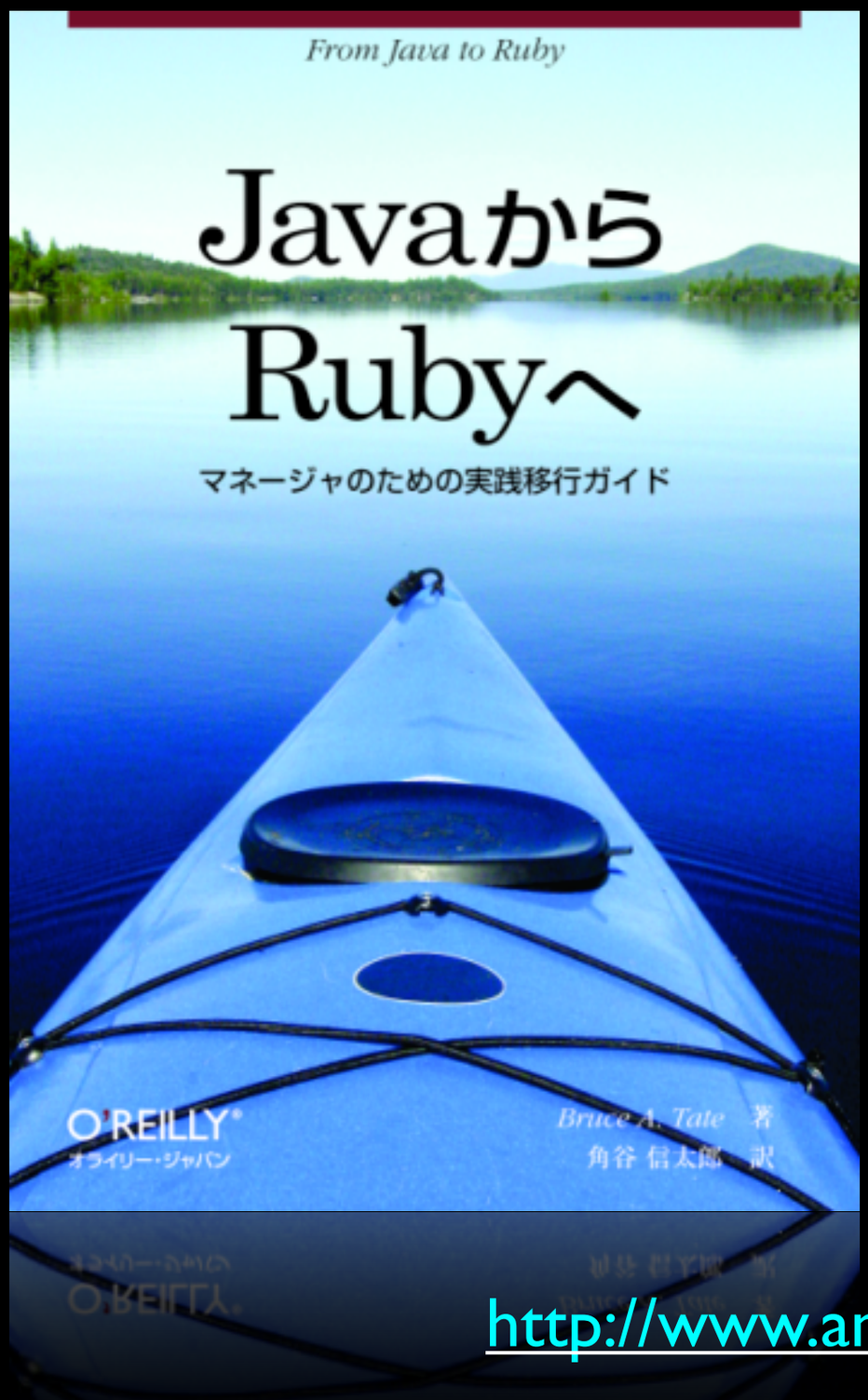
Stop!

Stop!

続きは

『JavaからRubyへ』

マネージャのための実践移行ガイド



Bruce A. Tate 著

角谷信太郎 訳

オライリー・ジャパン 発行

第3刷(fixed 78 bugs)

http://www.amazon.co.jp/o/ASIN/4873113202/kakutani-22

『JavaからRubyへ』

よりも大切なこと

テスト駆動開発

Test Driven Development

テストの分類

- ✓ Developer Testing
  - ✓ 開発者が行う、開発促進のためのテスト
- ✓ Customer Testing
  - ✓ お客様と機能の確認のために用いる、進捗管理のためのテスト
- ✓ QA Testing
  - ✓ 品質保証のためのテスト

Developer Testing

✓ プログラマの、

✓ プログラマによる、

✓ プログラマのための、

✓ プログラムとして書く、

✓ プログラミングを進めていくための、

✓ テスト

テスト:善きプログラマとしての身だしなみ

- ✓ 現代のFOSSでは常識
- ✓ パッチと共に求められることも
- ✓ 躾のできたハッカー
- ✓ RubyConf2007
- ✓ seattle.rb

プログラミングで積極的に活用すること

テスト駆動開発(まとめ)

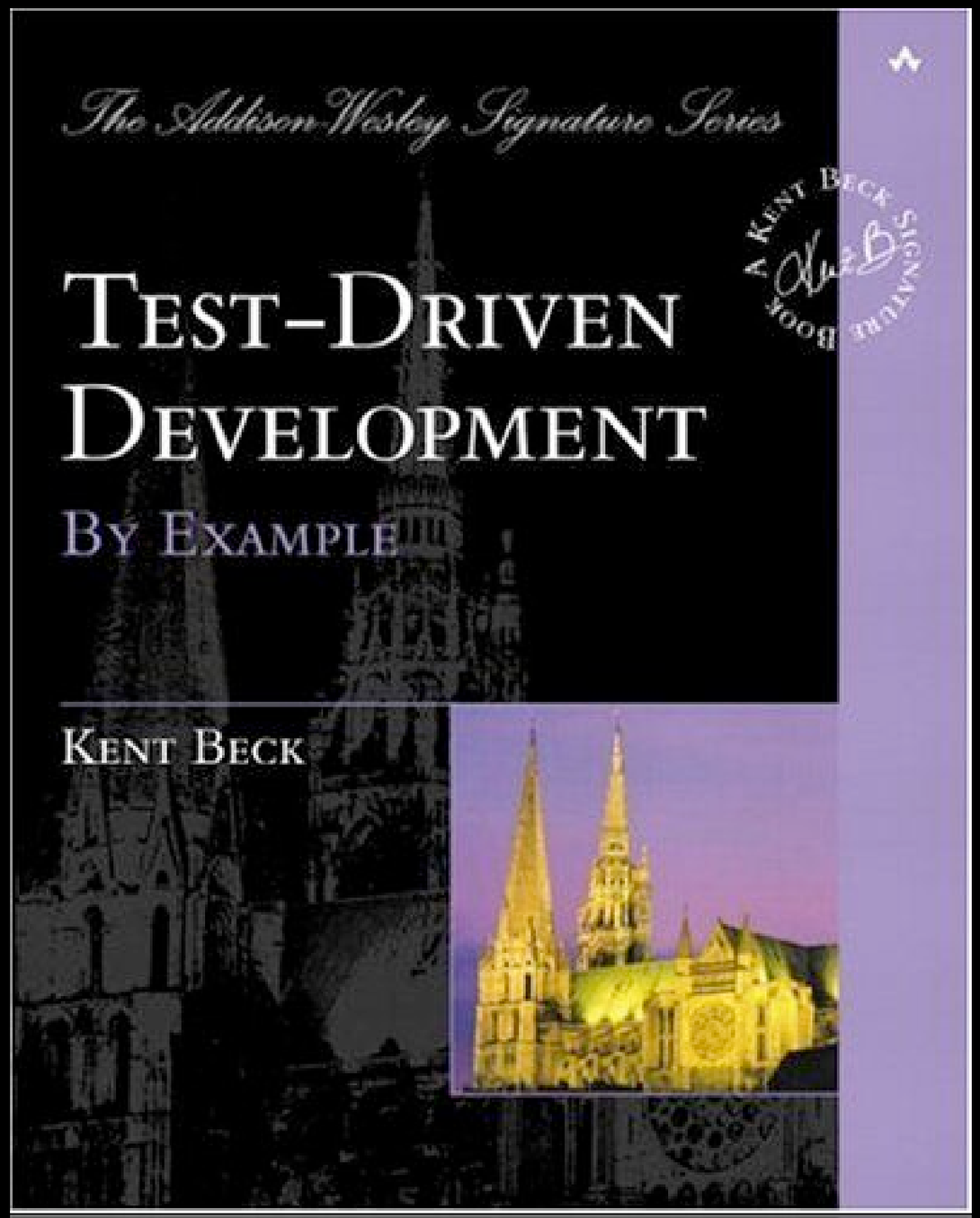
✓ プログラマのための設計技法

✓ プログラミングにリズムと
フィードバックをもたらす

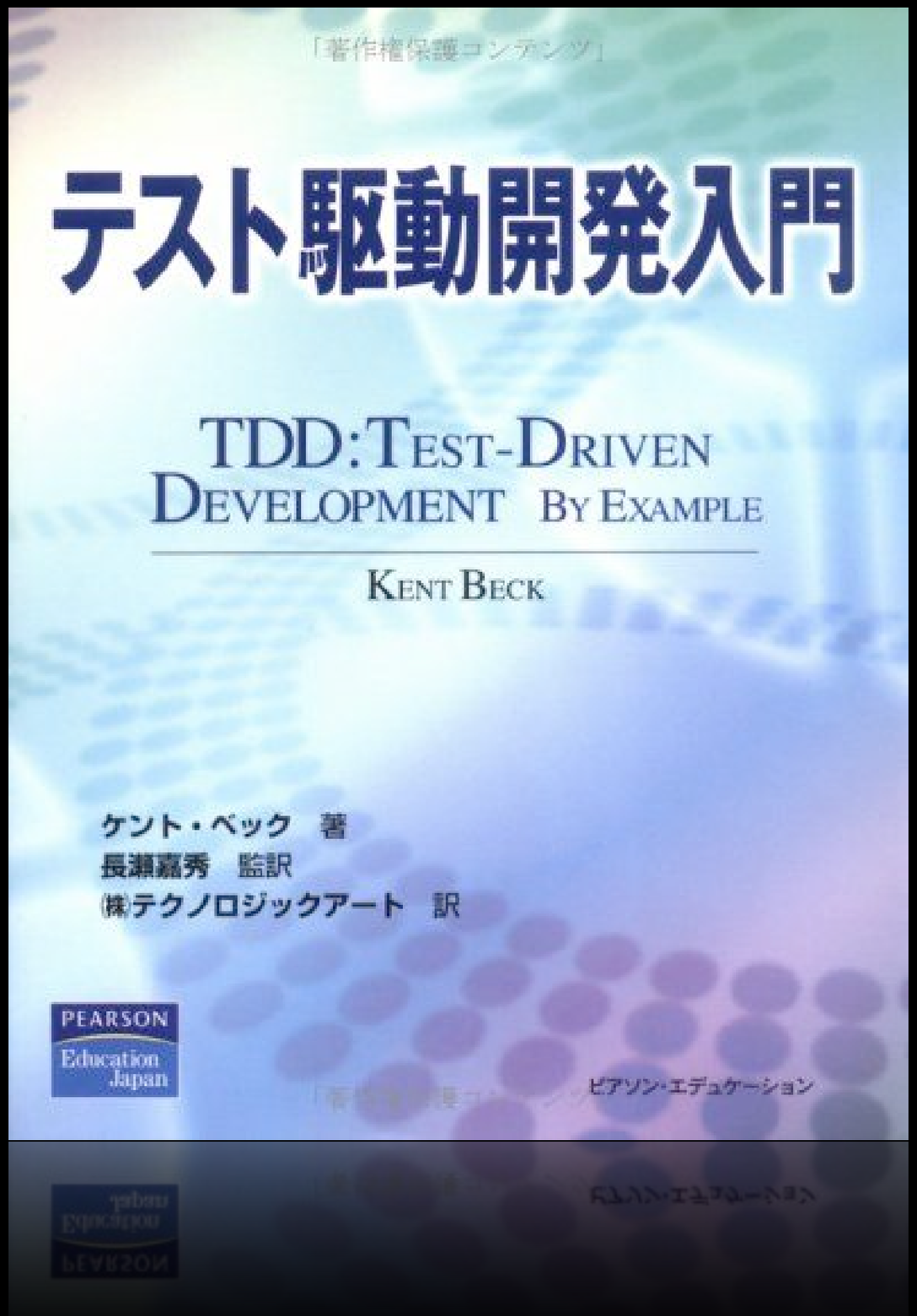
✓ プログラマとコードの健康を
保つ

2002

TDD by Example



テスト駆動開発入門


「著作権保護コンテンツ」
テスト駆動開発入門
TDD: Test-Driven Development By Example
Kent Beck
ケント・ベック 著
長瀬嘉秀 監訳
㈱テクノロジックアート 訳
PEARSON Education Japan
ピアソン・エデュケーション

偉大な書籍は

偉大な1行から

はじまる

Clean code that works, in Ron Jeffries' pithy phrase, is the goal of Test-Driven Development(TDD).

「動作するきれいなコード」、
このRon Jefferiesの簡潔な言葉こそが
TDDのゴールである。

動作する
きれいなコード

“Clean code that works”

動作するきれいなコード

- ✓ 予測可能な開発
- ✓ コードから学習する
- ✓ 生活の質の向上
- ✓ 相手を信頼する/に信頼される
- ✓ 心地良さ

TDD: 2つのルール

✓ テストに失敗したときだけ
コードを書く

✓ 重複を取り除く

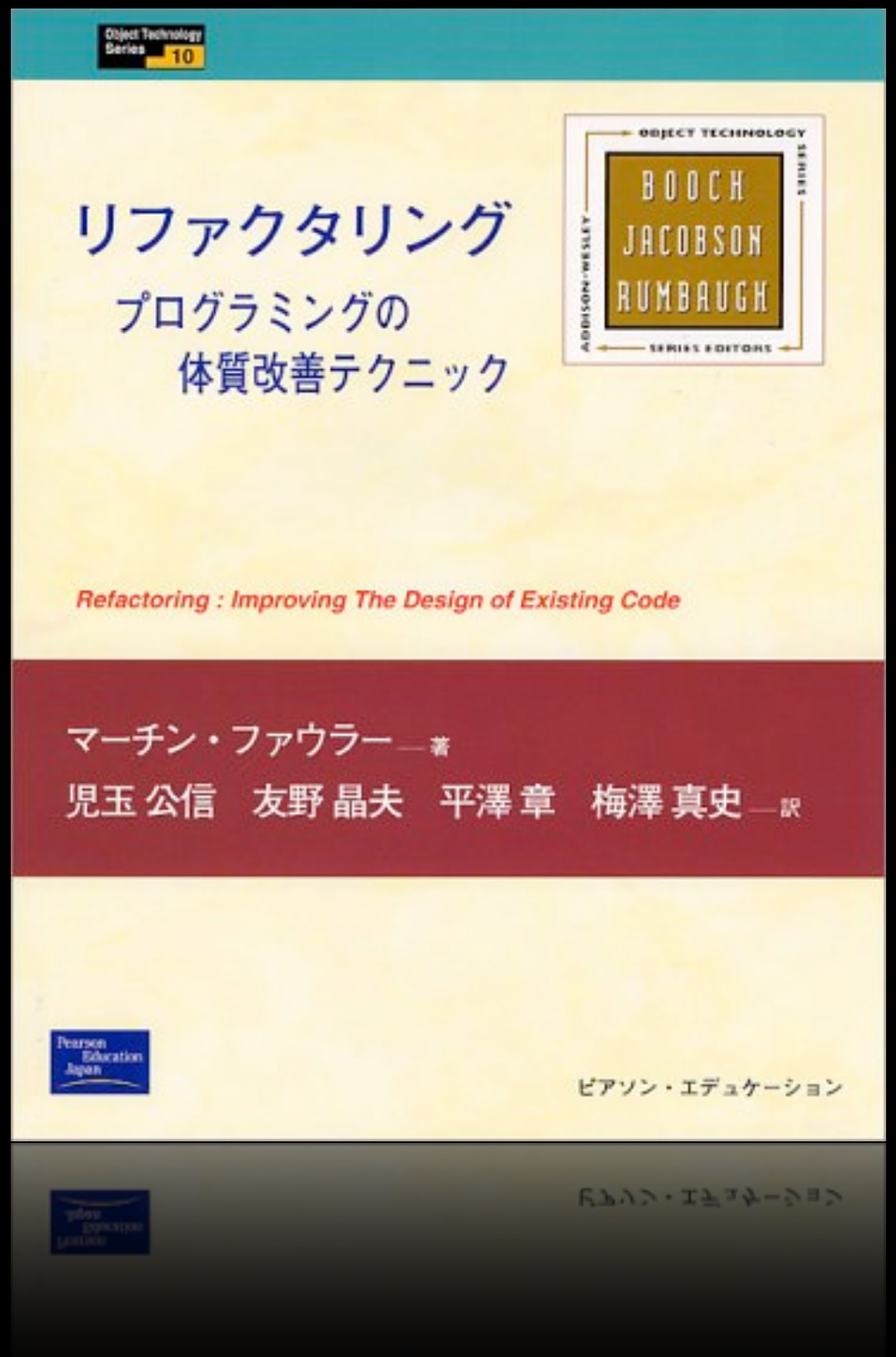
2つのルールが求めるもの

- ✓ 有機的な設計
- ✓ 自分でテストを書くこと
- ✓ 素早いフィードバック
- ✓ 高凝集・疎結合

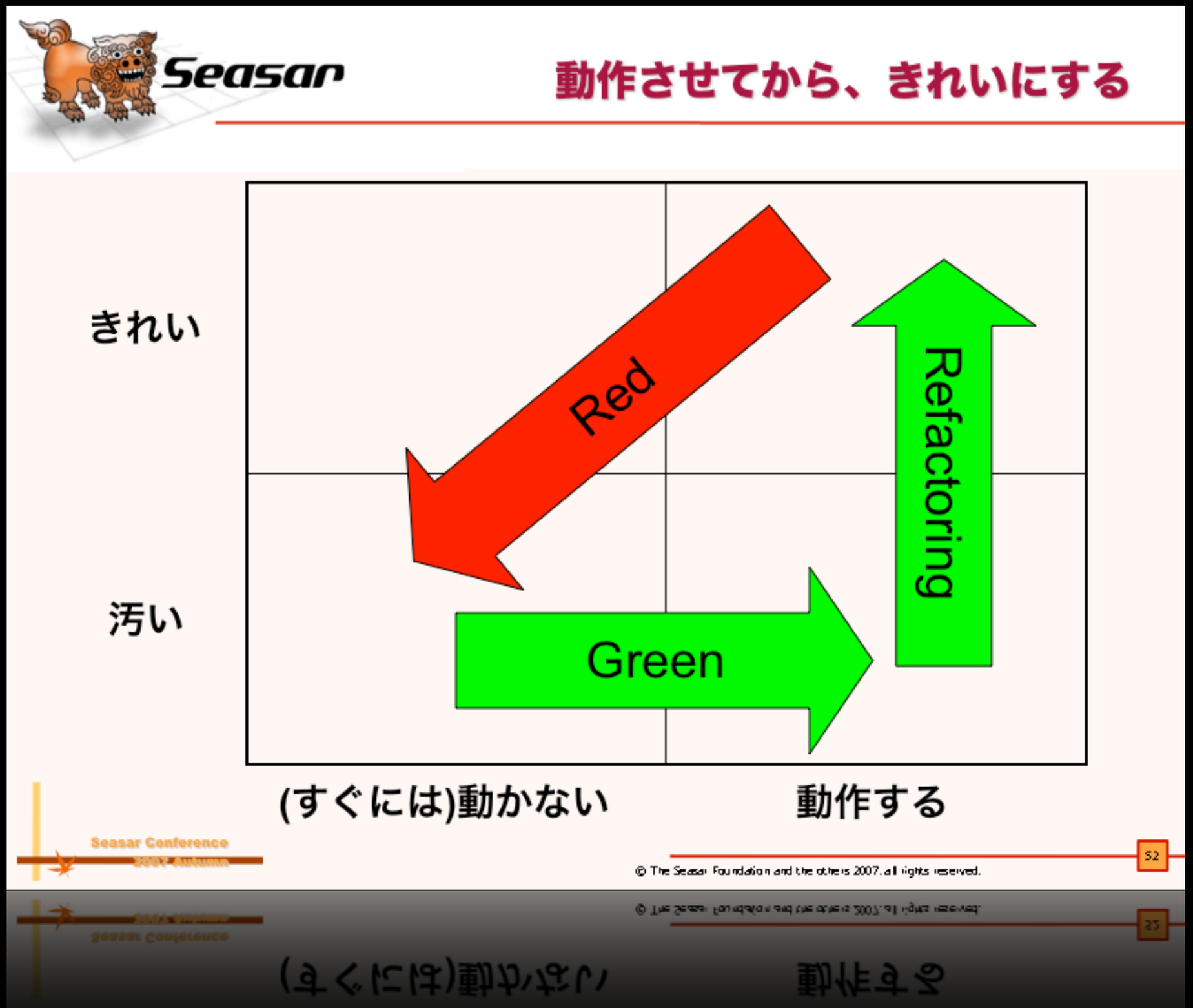
2つのルールが導くもの

- ✓ Red
  - ✓ テストに失敗している状態
- ✓ Green
  - ✓ コードが動作している状態
- ✓ Refactoring
  - ✓ コードの意味を変えずにきれいにする

リファクタリング


Object Technology Series 10
リファクタリング
プログラミングの
体質改善テクニック
OBJECT TECHNOLOGY SERIES
ADDISON-WESLEY
SERIES EDITORS
BOOCH
JACOBSON
RUMBAUGH
Refactoring : Improving The Design of Existing Code
マーチン・ファウラー 著
児玉 公信　友野 晶夫　平澤 章　梅澤 真史 訳
Pearson Education Japan
ピアソン・エデュケーション

動作するきれいなコード



TDDのマントラ

レッドグリーンリファクタレッドグリーンリファクタレッドグリーンリファクタレッドグリーンリファクタレッドグリーンリファクタレッドグリーンリ

ファクタレッドグリーンリファクタレッドグリーンリファクタレッドグリーンリファクタ……

動作するきれいなコード

- ✓ 不安
  - ✓ きれいなコードから遠ざかってしまう
- ✓ 勇気
  - ✓ 自分の不安と向き合う
- ✓ フィードバック
  - ✓ どこ居るのか? どこへ向かうべきか?

TDDの社会的意味

- ✓ 能動的な品質保証(QA)
- ✓ 正確な見積り
- ✓ 協調作業
- ✓ 新たなビジネス関係

テストが開発を駆動する

✓ 具体的なフィードバック

✓ 決定とのギャップ認識，制御

✓ 出荷可能なソフトウェア

✓ 不安に立ち向かう勇気

✓ コードに自信を持つこと

“The translation of a feeling into a test is a common theme of TDD.”

感情をテストにすることが、
TDDに共通するテーマである。

感情をテストにする

感情をテストにする

- ✓ 不安だ。
- ✓ 何かがおかしい。
- ✓ これでいい。
- ✓ 退屈だ。

TDDとは

✓プログラミングの決定と
フィードバックとのギャッ
プの認識、

✓ギャップを制御する技術

TDDは、

✓ テスト技法ではない

✓ 分析手法/設計技法であり、

✓ 開発のあらゆるアクティビティを構造化するための技法

テスト駆動開発(まとめ)

✓ プログラマのための設計技法

✓ プログラミングにリズムと
フィードバックをもたらす

✓ プログラマとコードの健康を
保つ

RSpec

RSpecとは

✓「テスト」が設計であることを強調するテスティングフレームワーク

✓Rubyを使って、プログラムの振舞の本質に集中できる書き方を可能に

rspec.rubyforge.org

RSpec-1.0.8: Home

RSpec 1.0.8

Home

Documentation | Community | License | Changes | Download | Upgrade | Examples

Overview

RSpec is a framework which provides programmers with a Domain Specific Language to describe the behaviour of Ruby code with readable, executable examples that guide you in the design process and serve well as both documentation and tests.

Here is how you do it

Start with a very simple example that expresses some basic desired behaviour.

```
# bowling_spec.rb
require 'bowling'

describe Bowling do
  before(:each) do
    @bowling = Bowling.new
  end

  it "should score 0 for gutter game" do
    20.times { @bowling.hit(0) }
    @bowling.score.should == 0
  end
end
```

Run the example and watch it fail.

```
$ spec bowling_spec.rb
./bowling_spec.rb:4:
  uninitialized constant Bowling
```

Now write just enough code to make it pass.

```
# bowling.rb
class Bowling
  def hit(pins)
  end

  def score
    0
  end
end
```

Run the example and bask in the joy that is green.

```
$ spec bowling_spec.rb --format specdoc

Bowling
- should score 0 for gutter game

Finished in 0.007534 seconds

1 example, 0 failures
```

Take very small steps

Don't rush ahead with more code. Instead, add another example and let it guide you to what you have to do next. And

Rubyist Magazine 0021号
“スはスペックのス”(第1回)

http://jp.rubyist.net/magazine/?0021-Rspec

RSpecのインストール

```
$ gem install -y rspec
```

インストールの確認

```
$ spec -v
RSpec-1.0.8 (r2338) - BDD for Ruby
http://rspec.rubyforge.org/
```

スペックファイルの構造

```
describe Class, "コンテキスト" do
  before(:each) do
    # コンテキストのお膳立て
  end

  it "期待する振舞いの名前" do
    # ここに期待する振舞いを書く
  end
end
```

簡単な例

```
describe Array, "with some entries" do
  before(:each) do
    @array = %w(A B C)
  end

  it "should not be nil" do
    @array.should_not be_nil
  end

  it "should last element is 'C'" do
    @array.last.should == 'C'
  end
end
```

実行(-cオプション)

```
$ spec -c array_spec.rb
..

Finished in 0.00812 seconds

2 examples, 0 failures
```

実行(-fsオプション)

```
$ spec -c -fs array_spec.rb
Array with some entries
- should not be nil
- should last element is 'C'

Finished in 0.008545 seconds

2 examples, 0 failures
```

簡単な例

```
describe Array, "with some entries" do
  before(:each) do
    @array = %w(A B C)
  end

  it "should not be nil" do
    @array.should_not be_nil
  end

  it "should last element is 'C'" do
    @array.last.should == 'C'
  end
end
```

期待する振舞の書き方

✓ Object#should(matcher)

```
@array.last.should == 'C'
# @array.last が 'C' と等しいことを期待
```

✓ Object#should_not(matcher)

```
@array.should_not be_nil
# @array.nil? がfalseであることを期待
```

さまざまなマッチャ

- ✓ 演算子マッチャ
  - ✓ ==, ===, <, <=, =~, >, >=
- ✓ ビルトインのマッチャ
  - ✓ change,raise_error, satisfy, be_close...
- ✓ be_xxxマッチャ
- ✓ be_true / be_false
- ✓ have_xxx マッチャ
- ✓ ユーザ定義のマッチャ

続きは

Rubyist Magazine 0021号
“スはスペックのス”(第1回)

http://jp.rubyist.net/magazine/?0021-Rspec

実演

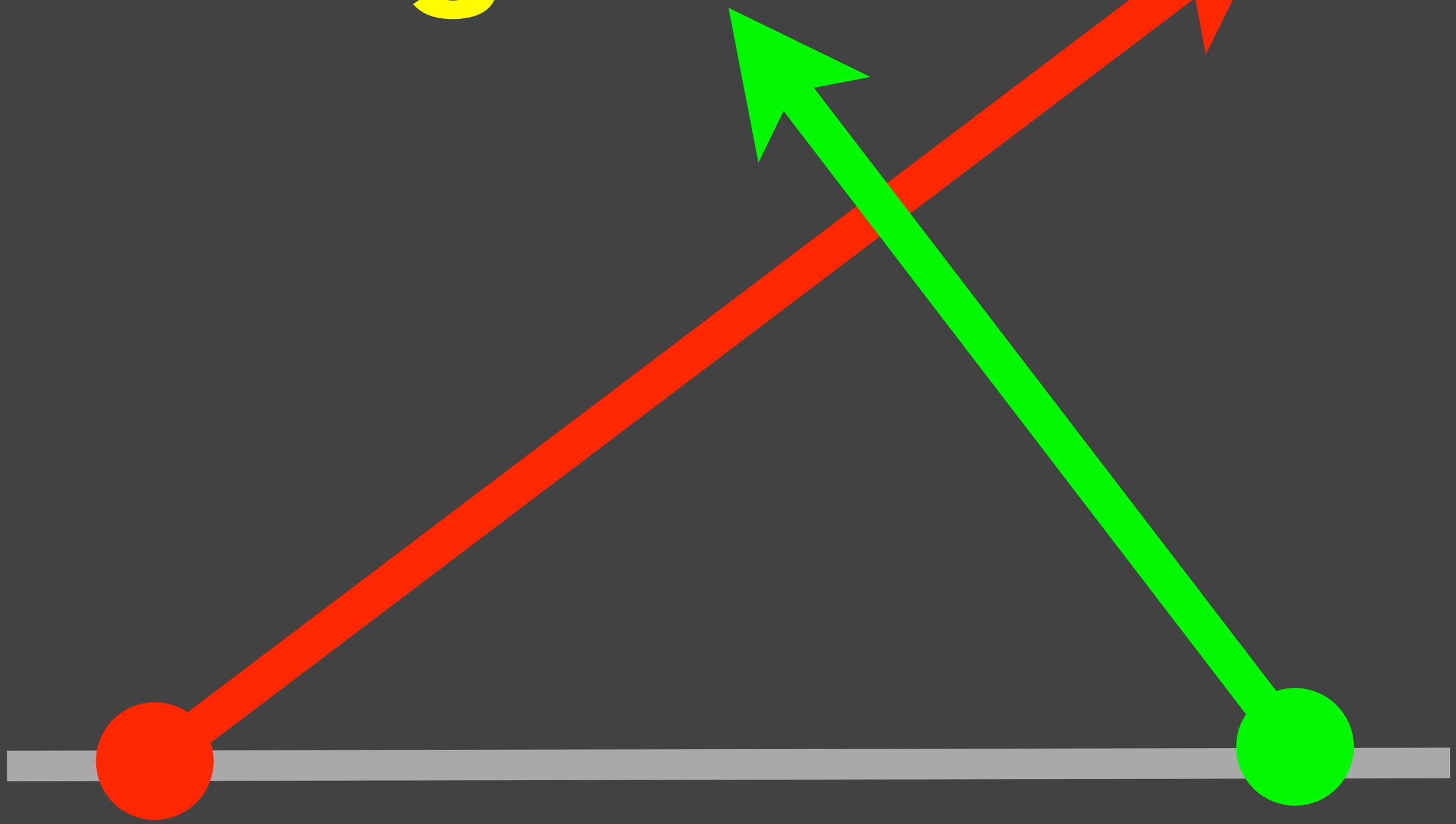
スタックの仕様

- ✓ FILO(LIFO)のデータ構造
  - ✓ First In Last Out
- ✓ push: 一番上に積む
  - ✓ nilはpushできない
- ✓ pop: 一番上を取り出す(取り除く)
  - ✓ スタックが空だとpopできない
- ✓ peek: 一番上を取得(取り除かない)
  - ✓ スタックが空だとpeekできない

紹介するTDDの技法

- ✓ Fake It
  - ✓ いんちき
- ✓ Triangulate
  - ✓ 三角測量。二方向から挟み撃ちに
- ✓ Obvious Implementation
  - ✓ ふつうに実装する

Triangulate



空のスタック

- ✓ Stack#empty?
  - ✓ true を返すこと
- ✓ Stack#pop
  - ✓ 例外を発生させること
- ✓ Stack#peek
  - ✓ 例外を発生させること

空のスタック

- ✓ Stack#empty?
  - ✓ true を返すこと
- ✓ Stack#pop
  - ✓ 例外を発生させること
- ✓ Stack#peek
  - ✓ 例外を発生させること

スタックに1件だけpush

- ✓ Stack#push(item)でオブジェクトを積めること
- ✓ Stack#empty?はfalse
- ✓ Stack#peekで積んだオブジェクトを覗けること
- ✓ Stack#popで積んだオブジェクトを取り出す(取り除く)こと

スタックに複数件push

- ✓ Stack#push(item)で積んだ逆順で取りだせること
- ✓ あとは1件だけpushした場合と同じ

nilをpushした場合

- ✓ 例外が発生すること
  - ✓ ArgumentError

空のスタック(再び)

- ✓ Stack#empty?
  - ✓ true を返すこと
- ✓ Stack#pop
  - ✓ 例外を発生させること
- ✓ Stack#peek
  - ✓ 例外を発生させること

テスト駆動開発(まとめ)

✓ プログラマのための設計技法

✓ プログラミングにリズムと
フィードバックをもたらす

✓ プログラマとコードの健康を
保つ

本日のお話の要点

✓ テスト駆動開発は、善きプログラマとしての身だしなみです

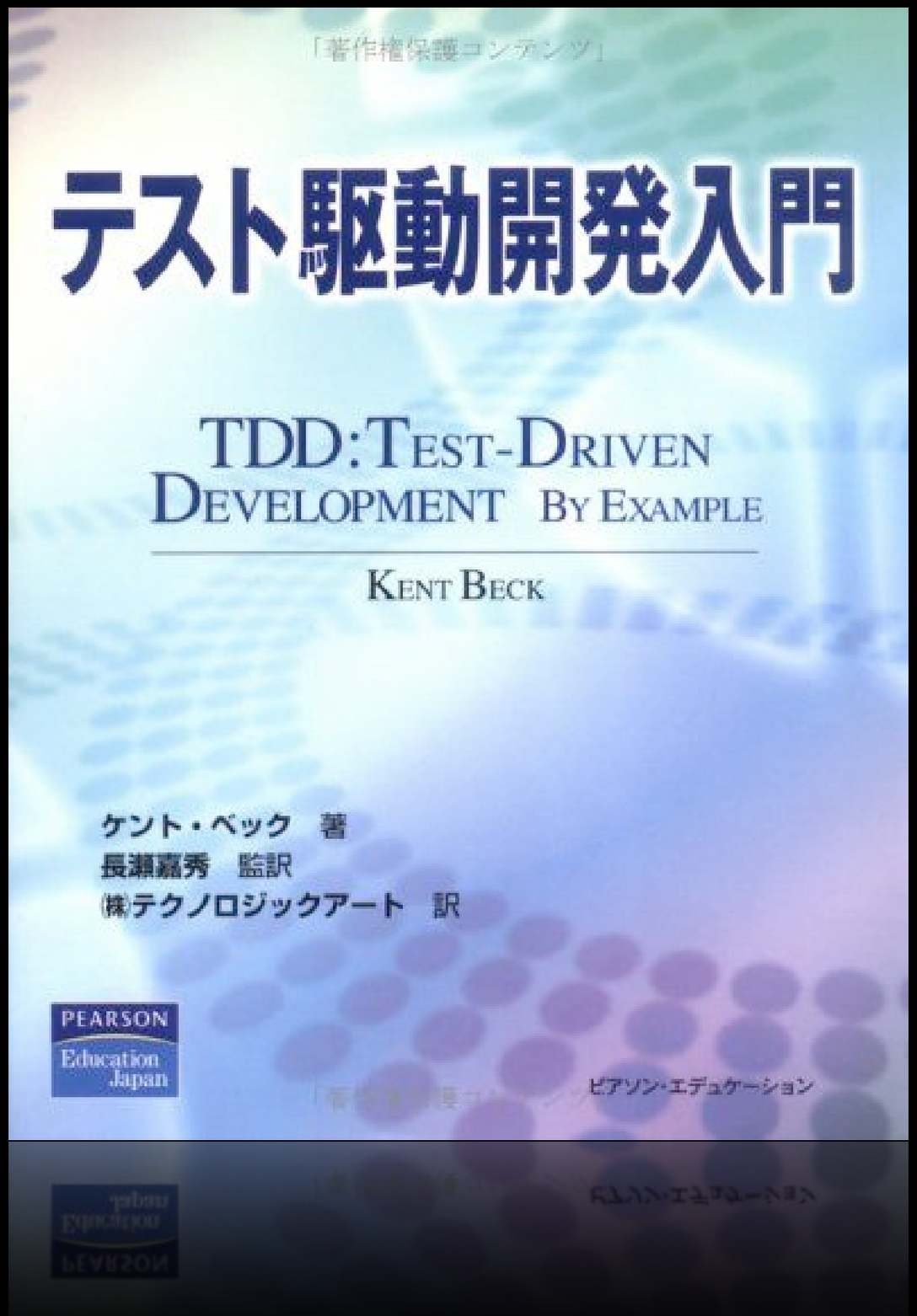
✓ 今日のお話をきっかけに、興味をもってらえると嬉しいです

✓ 『JavaからRubyへ』も買ってください :-)

紹介したTDDの技法

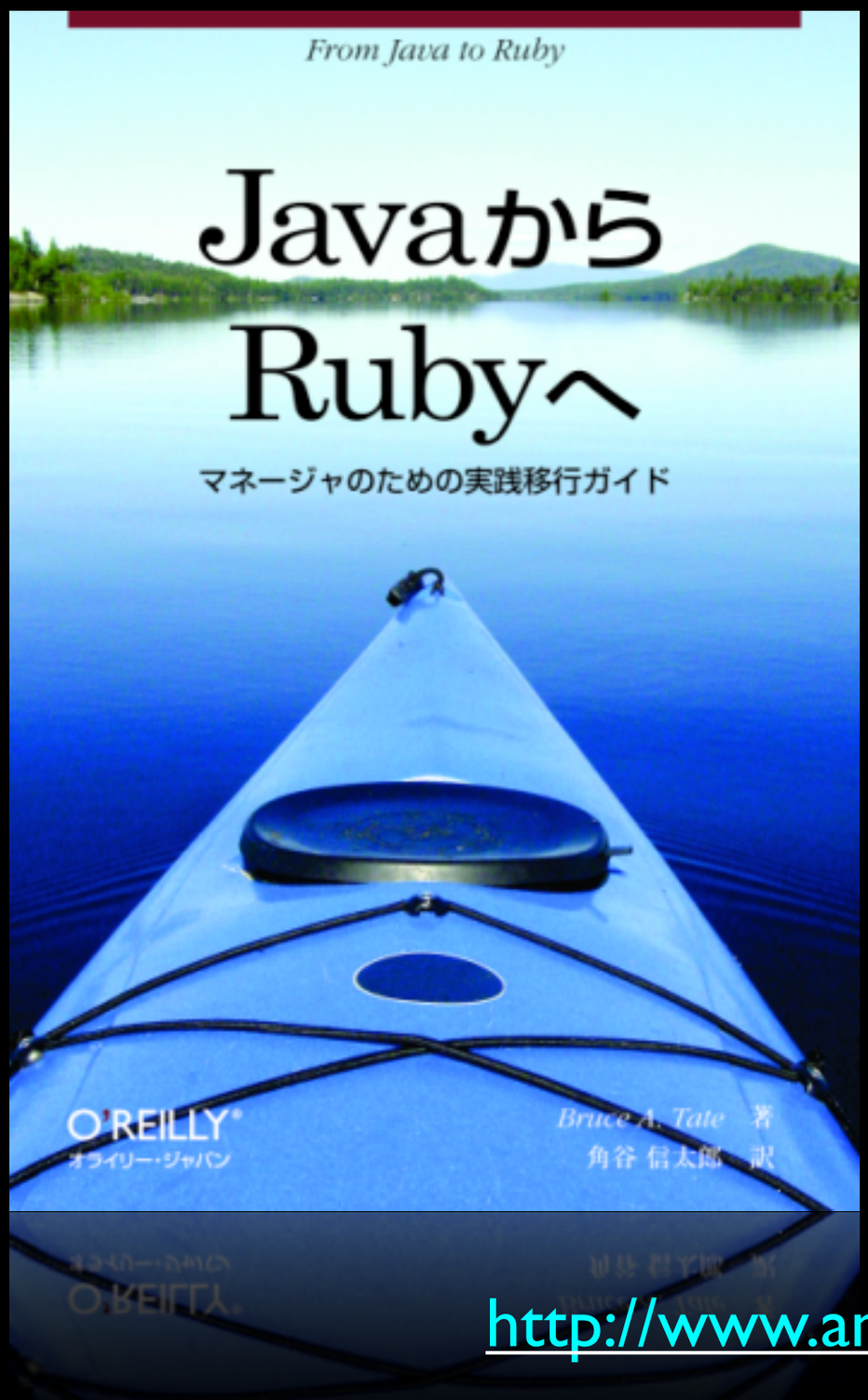
- ✓ Fake It
  - ✓ いんちき
- ✓ Triangulate
  - ✓ 三角測量。二方向から挟み撃ちに
- ✓ Obvious Implementation
  - ✓ ふつうに実装する

テスト駆動開発入門


「著作権保護コンテンツ」
テスト駆動開発入門
TDD: Test-Driven Development By Example
Kent Beck
ケント・ベック 著
長瀬嘉秀 監訳
㈱テクノロジックアート 訳
PEARSON Education Japan
ピアソン・エデュケーション

『JavaからRubyへ』

マネージャのための実践移行ガイド



Bruce A. Tate 著

角谷信太郎 訳

オライリー・ジャパン 発行

第3刷(fixed 78 bugs)

http://www.amazon.co.jp/o/ASIN/4873113202/kakutani-22

ご清聴

ありがとう

ございました

何かご質問は?

Do you have any questions?