スはスペックのス ～RSpecによるテスト駆動開発の実演～

“S is for Spec”:Test Driven Development w/ RSpec

角谷 信太郎

日本Rubyの会
(株)永和システムマネジメント
s-kakutani@esm.co.jp

KAKUTANI Shintaro; Eiwa System Management,Inc.; Nihon Ruby-no-kai

JPUG北海道支部/Ruby札幌 合同セミナー;札幌エルプラザ; 2008-2-16(土)

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

寄稿，翻訳，監訳，連載


Rubyist Magazine
出張版
正しいRubyコードの書き方講座


From Java to Ruby
Javaから
Rubyへ
マネージャのための実践移行ガイド
O'REILLY


The Pragmatic Programmers
アジャイル
プラクティス
達人プログラマに学ぶ現場開発者の習慣


LifeHacks
PRESS
デジタル世代の「カイゼン」術
GTD
シンプル&ストレスフリーの仕事術


WEB+DB
PRESS
vol.39
バージョン管理→ビルド→リリース
構成管理実践入門
PHP最新活用術
"便利"関数&ライブラリ
CSS基礎講座


WEB+DB
PRESS
vol.41
Flash, Flex, AIR
Action Script 3.0 開発入門
Seasar2 実践カリキュラム
Comet


WEB+DB
PRESS
vol.40
Dave Thomas
パターンの基礎知識
Linux開発環境
ユーザビリティ
Vista文字化け対策
サーバの消費電力問題


WEB+DB
PRESS
vol.42
最新アルゴリズム&データ構造入門
ニコニコ動画
構築ノウハウ大公開
REST
Sennaで検索プログラミング

よろしく

お願いします

告知

RubyKaigi2008

CFP募集は

明日〆切

http://jp.rubyist.net/RubyKaigi2008/

よろしく

お願いします

提　供

Ruby札幌

Red, Green, Refactoring

TEST CLUB

テスト・クラブ

諸君　テスト・クラブへようこそ

テスト・クラブ　第1のルール

1. クラブのことを口外するな
2. クラブのことを口外するな
3. “ちょっといいかな”の声で
ドライバとナビゲータは交代
4. コーディングは1対1のペア
5. いちどに1つのテストケース
6. メッセとTwitterは切る
7. 書くテストに制限はない

８番目、最後のルール

“部員は入会し方初日にーー”

“必ずテストを書く”

Red, Green, Refactoring

TEST CLUB

テスト・クラブ

本日のお品書き

1. テスト駆動開発(TDD)のポイントを知る
2. Ruby用のツールであるRSpecを簡単に紹介
3. RSpecでTDDの実演

今日のまとめ

✓ TDDは設計技法であり、
開発の進め方である

✓ いちどにひとつずつ

✓ 不安をテストで表現する

✓ リズムとフィードバック

テスト駆動開発

Test Driven Development

✓ 1つのゴール

✓ 1つのテーマ

✓ 2つの主張

✓ 2つのルール

✓ 3つのモード

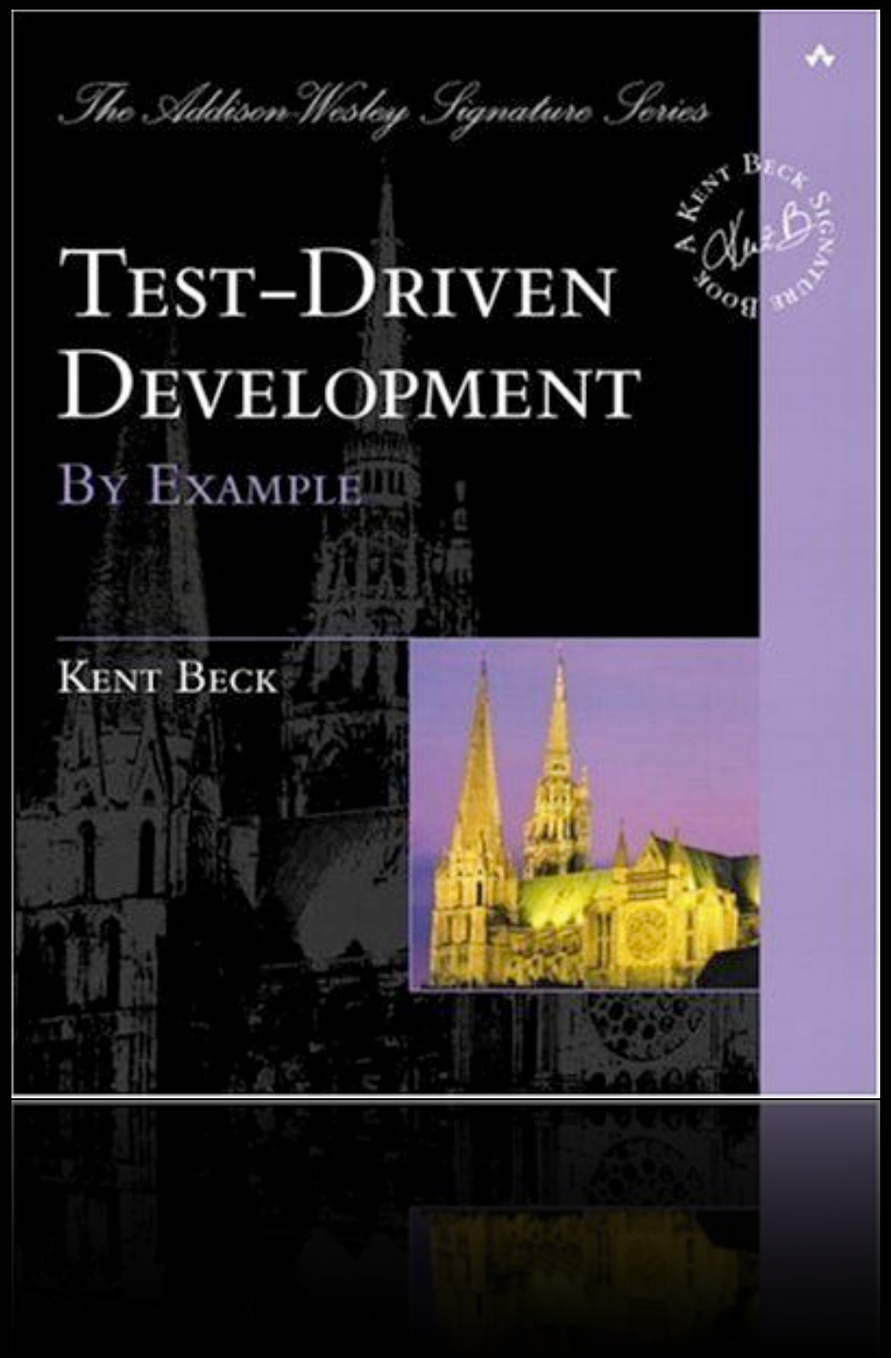
✓ 3つの技法

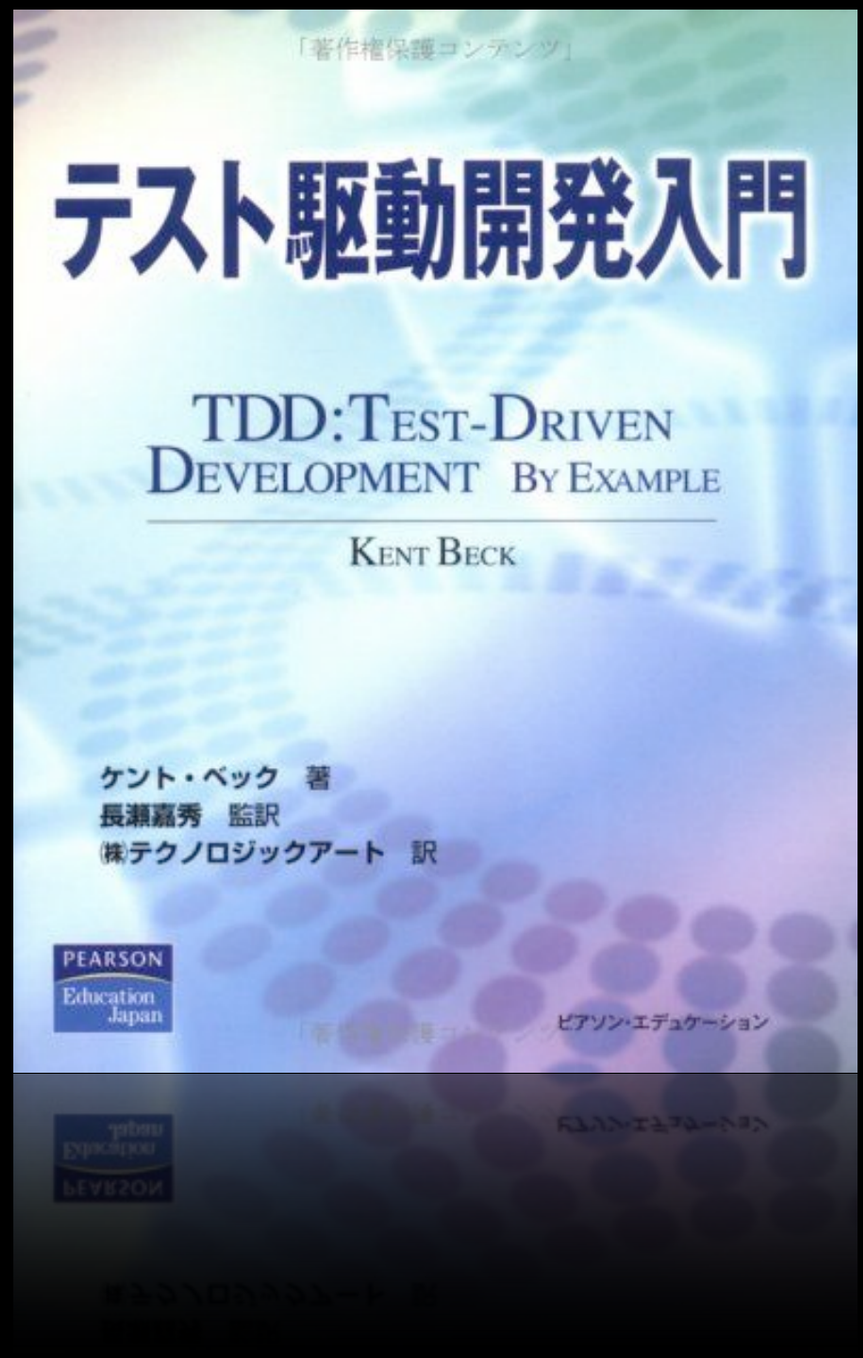
✓ 4つのモード

1つの

ゴール

TDD by Example





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”

1つの

テーマ

“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のテーマである。

感情を
テキストにする

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
  - ✓ テストを構造化する
  - ✓ 書くのを止める

2つの主張

TDDは、

1. 設計の技法

2. 開発の進め方

テスト駆動開発

Test Driven Development

設計の

技法

ソフトウェア設計とは何か?

What is Software Design?

By Jack W. Reeves

村上 雅章さんによる翻訳

http://www.biwa.ne.jp/~mmura/SoftwareDevelopment/WhatIsSoftwareDesignJ.html

ソフトウェア設計とは何か?

✓ 設計: ソースコード

✓ 製造: ビルド

コーディングは設計であり、テスティングとデバッギングも設計の一部であり、私たちが一般的にソフトウェア設計と呼んでいるものもやはり設計の一部なのです。

ソフトウェア設計はコーディングが完了し、「かつ」テストされるまでは完璧にならないのです。そして、テスティングは設計の検証と洗練を行うプロセスにおける礎となるものです。

設計の

技法

(2)

開発の

進め方

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

開発の

進め方

TDDは、

1. 設計の技法

2. 開発の進め方

テスト駆動開発

Test Driven Development

クは駆動のク

D is for Driven

2つのルール

1. テストに失敗したときだけコードを書く

2. 重複を取り除く

3つのモード

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

きれい

汚ない

RED

GREEN

Refactoring

動かない

動く

いま、どのモードなのかを意識するのが大事

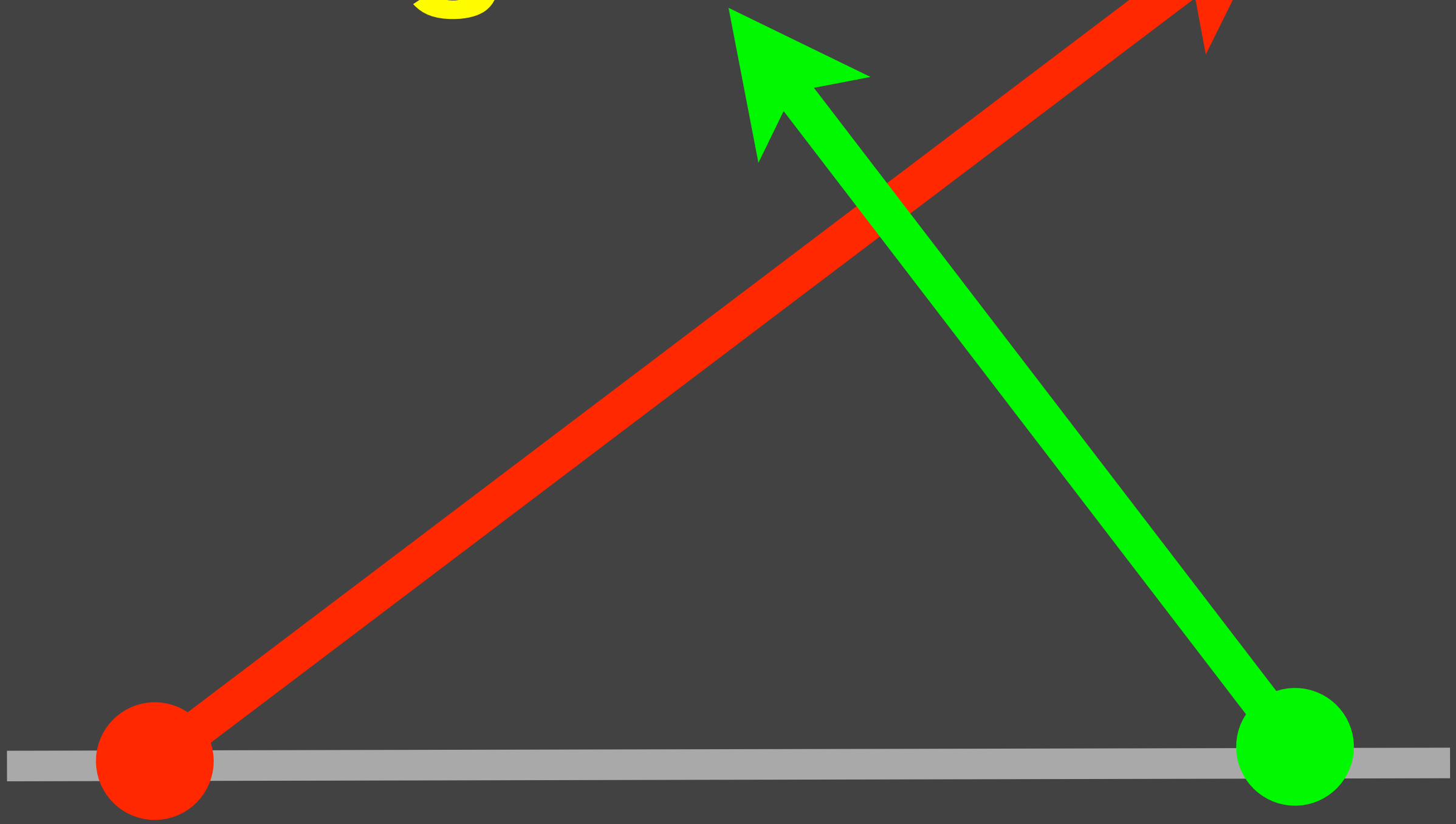
3つの

技法

3つの技法

- ✓ 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

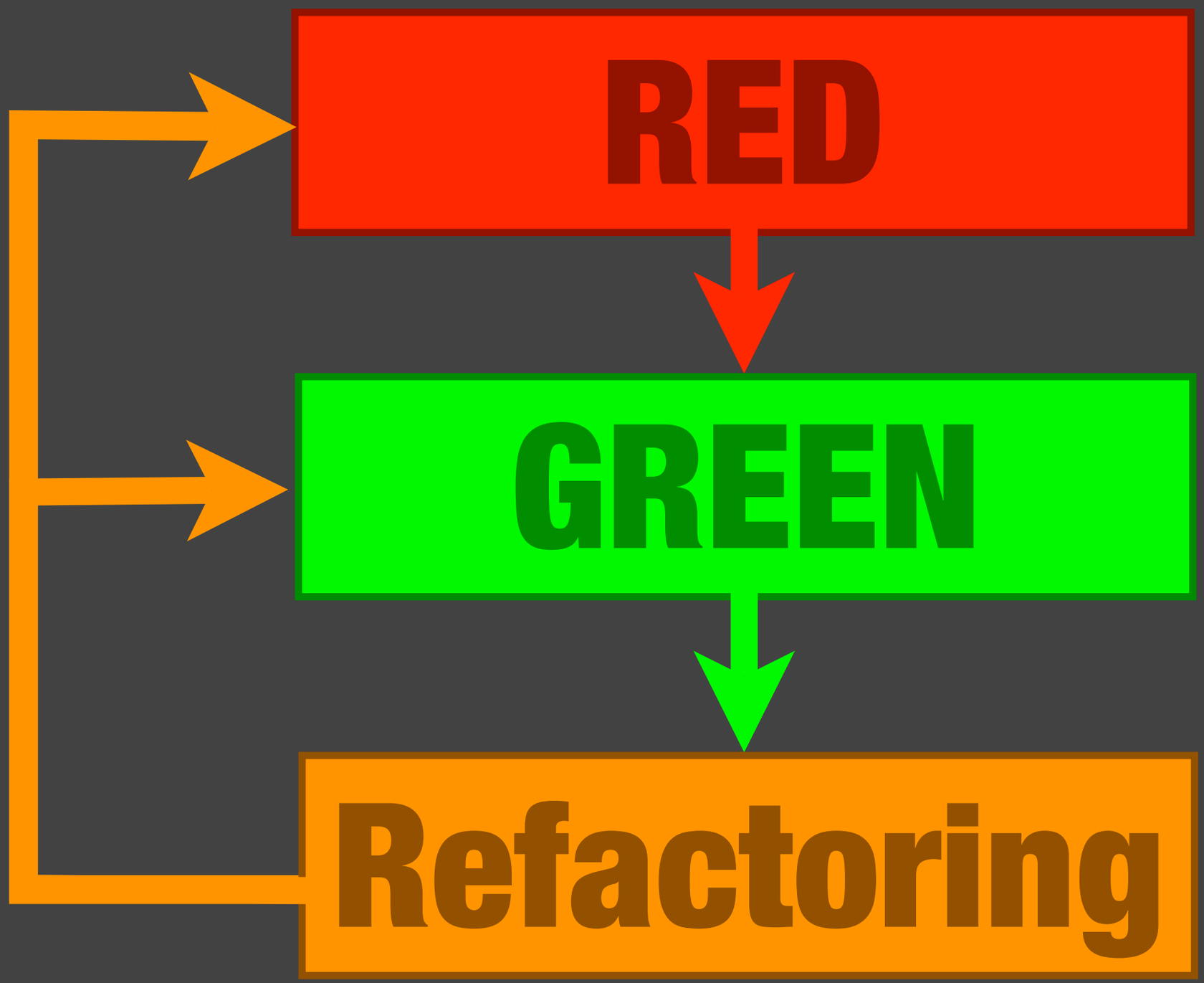
Triangulate



4コマ

モード

3つのモード

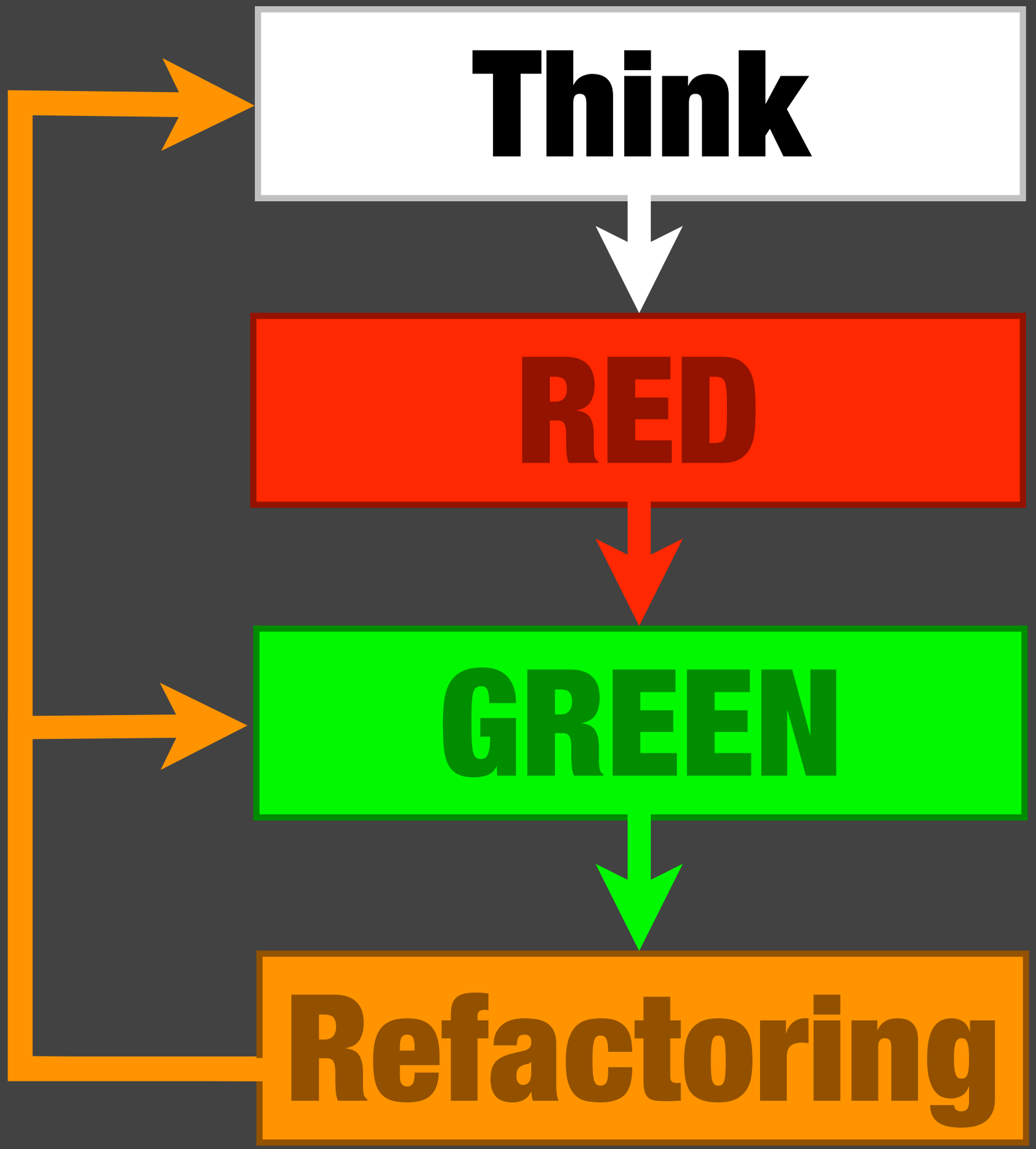

RED
GREEN
Refactoring

3つのモードで

明示されていな

いモードを追加

The Art of Agile Development


Think
RED
GREEN
Refactoring

TDDØ

マントラ

シンクレッドグリーンリファ
クタシンクレッドグリーンリ
ファクタシンクレッドグリー
ンリファクタシンクレッドグ
リーンリファクタシンクレッ
ドグリーンリファクタシンク
レッドグリーンリファクタシ

✓ 1つのゴール

✓ 1つのテーマ

✓ 2つの主張

✓ 2つのルール

✓ 3つのモード

✓ 3つの技法

✓ 4つのモード

今日のまとめ

✓ TDDは設計技法であり、
開発の進め方である

✓ いちどにひとつずつ

✓ 不安をテストで表現する

✓ リズムとフィードバック

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している

http://rspec.info

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

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 rspec
```

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
```

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

ボウリングの

スコア算出

The Bowling Game Kata

by Robert C.Martin(Uncle Bob)

Bowling Game Kata

Object Mentor, Inc.

www.objectmentor.com

blog.objectmentor.com

fitnesse.org

XProgramming.com

JU.org

www.junit.org

http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

コード=カタ

http://codekata.**pragprog.com**/

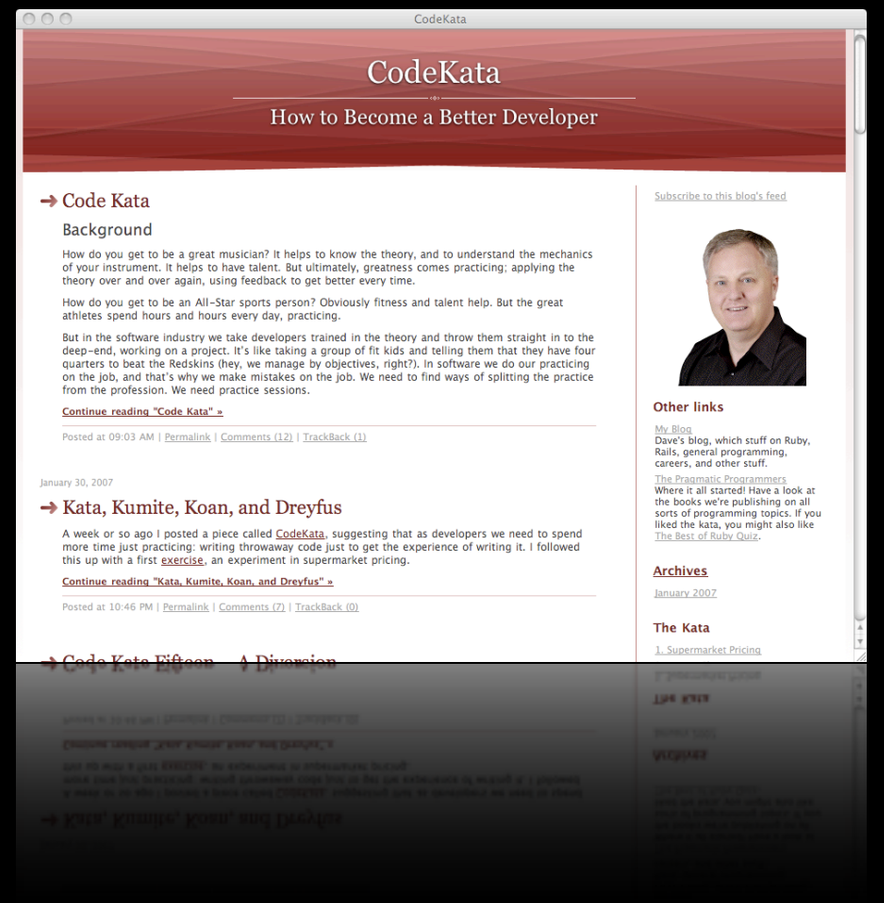
コード=カタ

✓ 空手の「型」

✓ 単純な定型を反復

✓ 素振り

✓ カラダでおぼえる



http://codekata.**pragprog.com**/

ボウリングのスコア

✓ 1フレームで2投するのが基本

✓ 1ゲームは10フレーム

✓ ストライクは、次の2投をボーナス加算

✓ スペアは、次の1投をボーナス加算

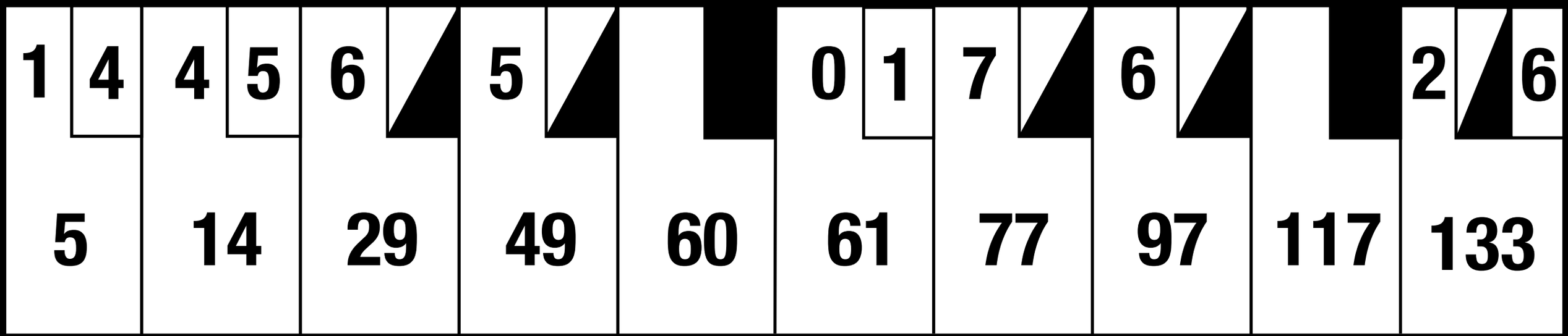
✓ 10フレーム目で10ピン倒せば
　合計で3投できる

今回の仕様

✓ 1ゲーム(10フレーム、最大21投)の
投球をすべて記録してから、

✓ 最後にまとめてスコアを算出する

受入テストケース




```
1,4,4,5,6,4,5,5, 10, 0,1,7,3,6,4,10,2,8,6
```

使用するツール

- ✓ Ruby 1.8.6 p111
- ✓ RSpec 1.1.3
- ✓ CarbonEmacs 22.1.50.1
- ✓ ZenTest 3.9.1
  - ✓ AutoTestを使いたい
  - ✓ autotest.el

作戦

1. スコアを集計する
2. ストライクの場合のボーナス
3. スペアの場合のボーナス
4. 受入テスト

今日のまとめ

✓ TDDは設計技法であり、
開発の進め方である

✓ いちどにひとつずつ

✓ 不安をテストで表現する

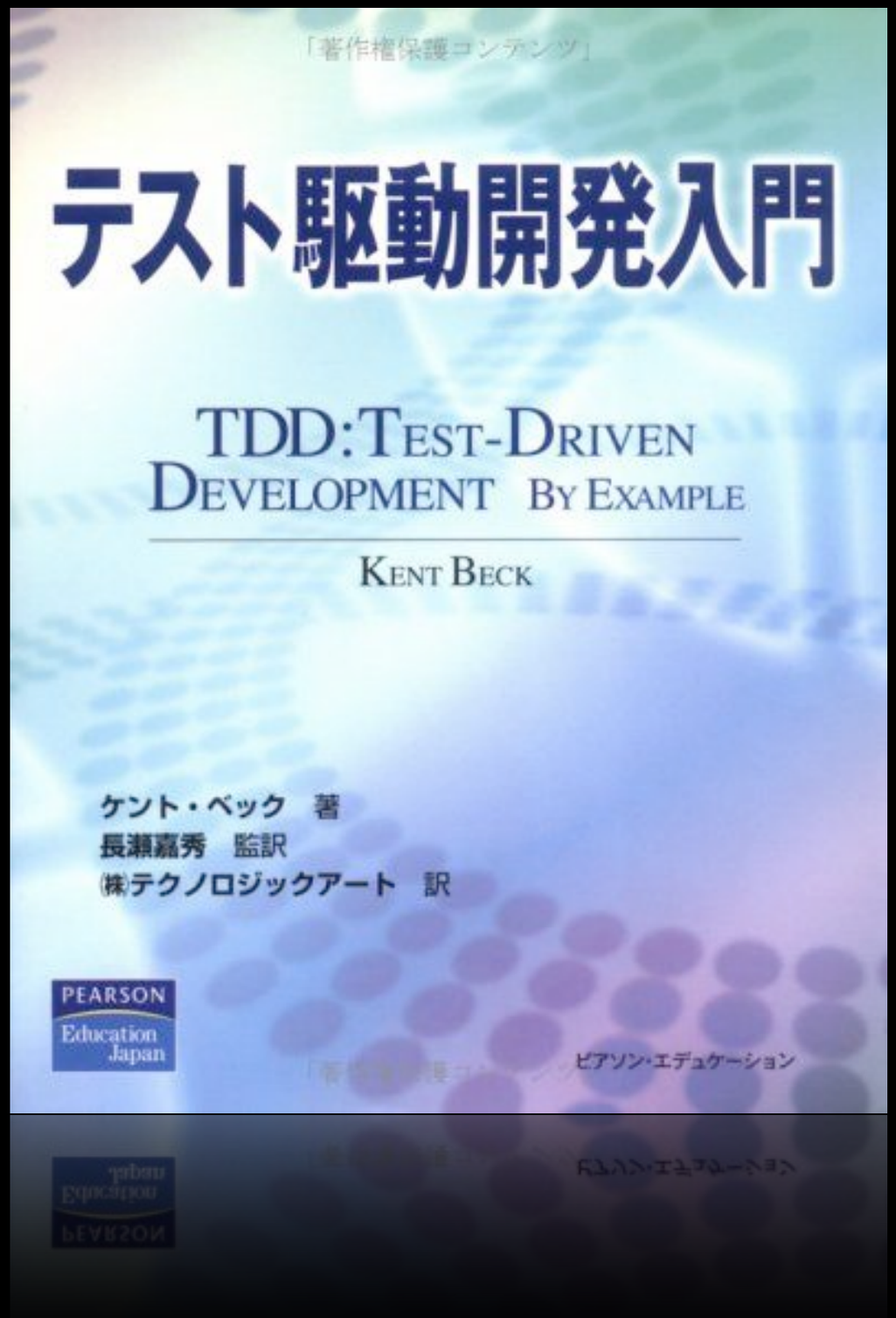
✓ リズムとフィードバック

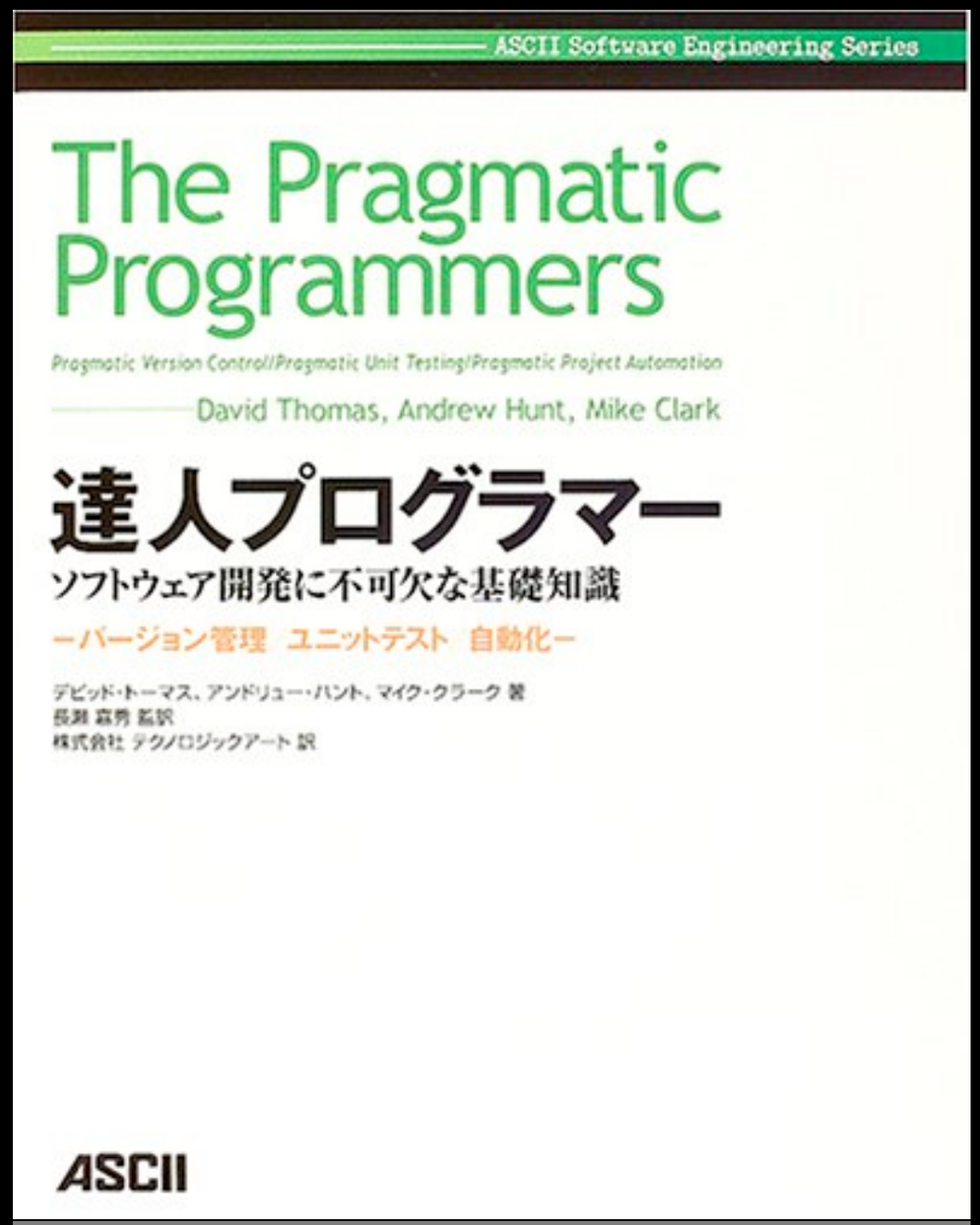
参考文献

参考文献

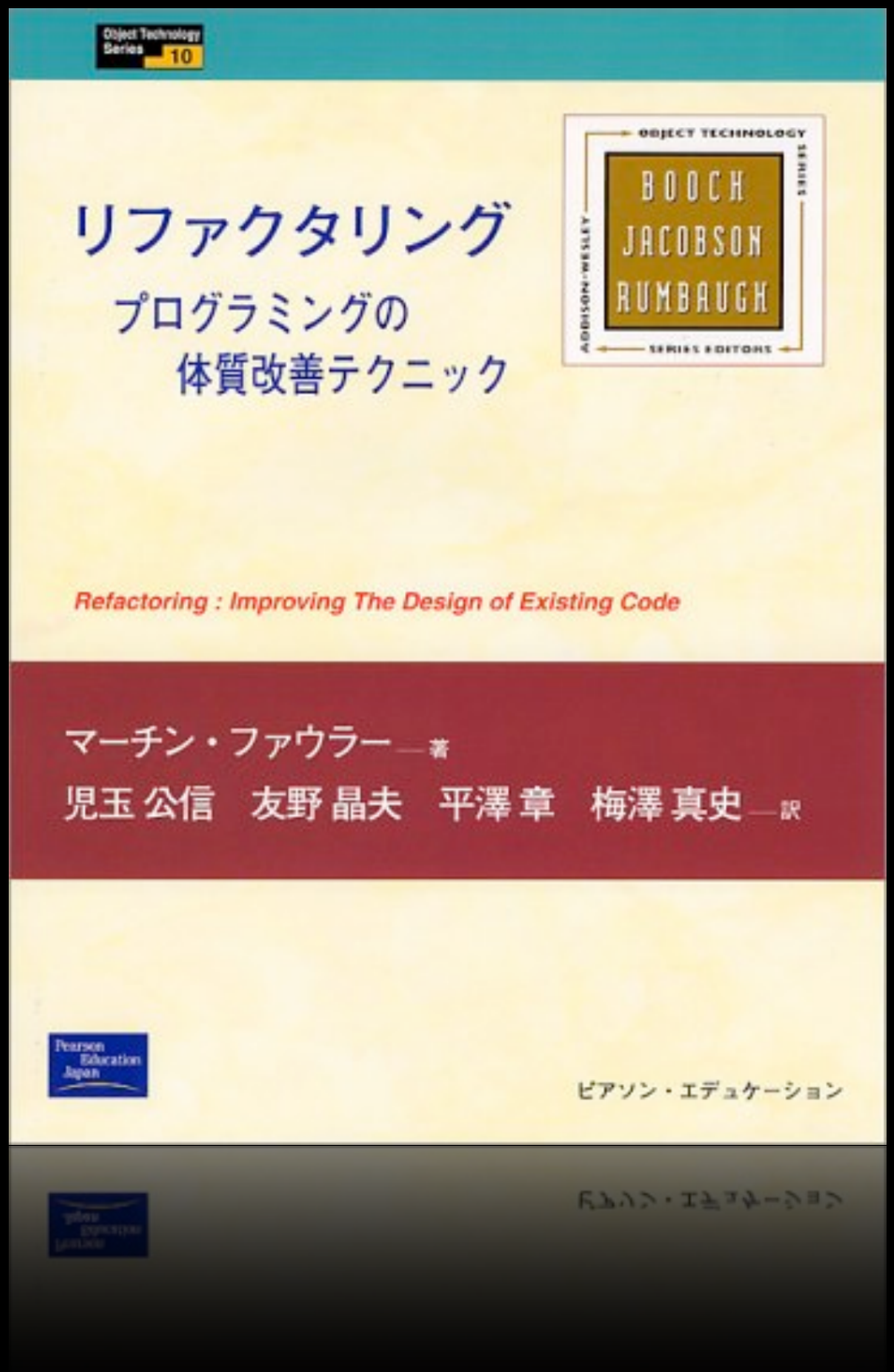
1. テスト駆動開発
2. リファクタリング
3. オブジェクト設計
4. デザインパターン

テスト駆動開発


「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

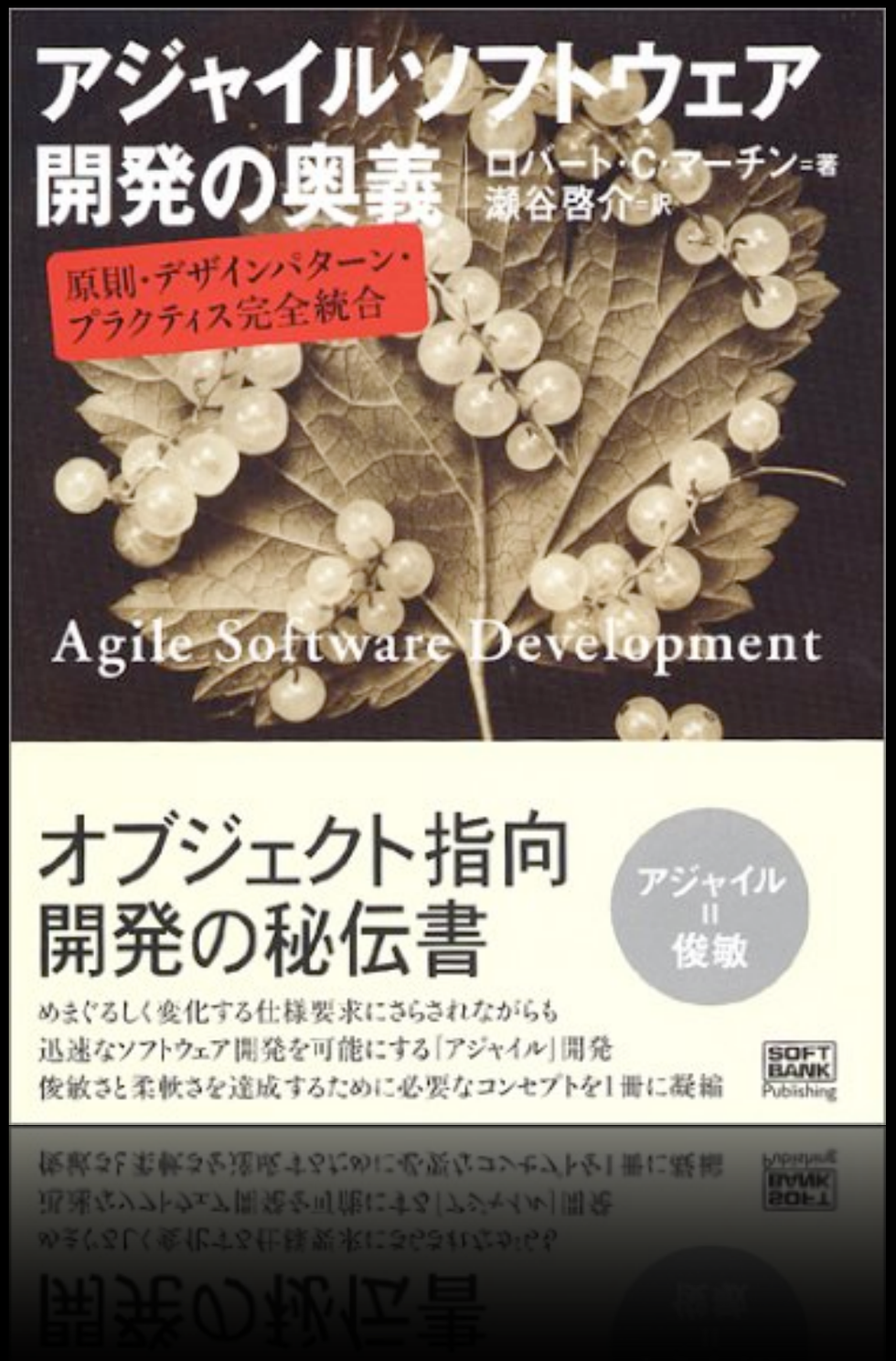

ASCII Software Engineering Series
The Pragmatic Programmers
Pragmatic Version Control/Pragmatic Unit Testing/Pragmatic Project Automation
David Thomas, Andrew Hunt, Mike Clark
達人プログラマー
ソフトウェア開発に不可欠な基礎知識
ーバージョン管理 ユニットテスト 自動化ー
デビッド・トーマス、アンドリュー・ハント、マイク・クラーク 著
長瀬 嘉秀 監訳
株式会社 テクノロジックアート 訳
ASCII

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
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS
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

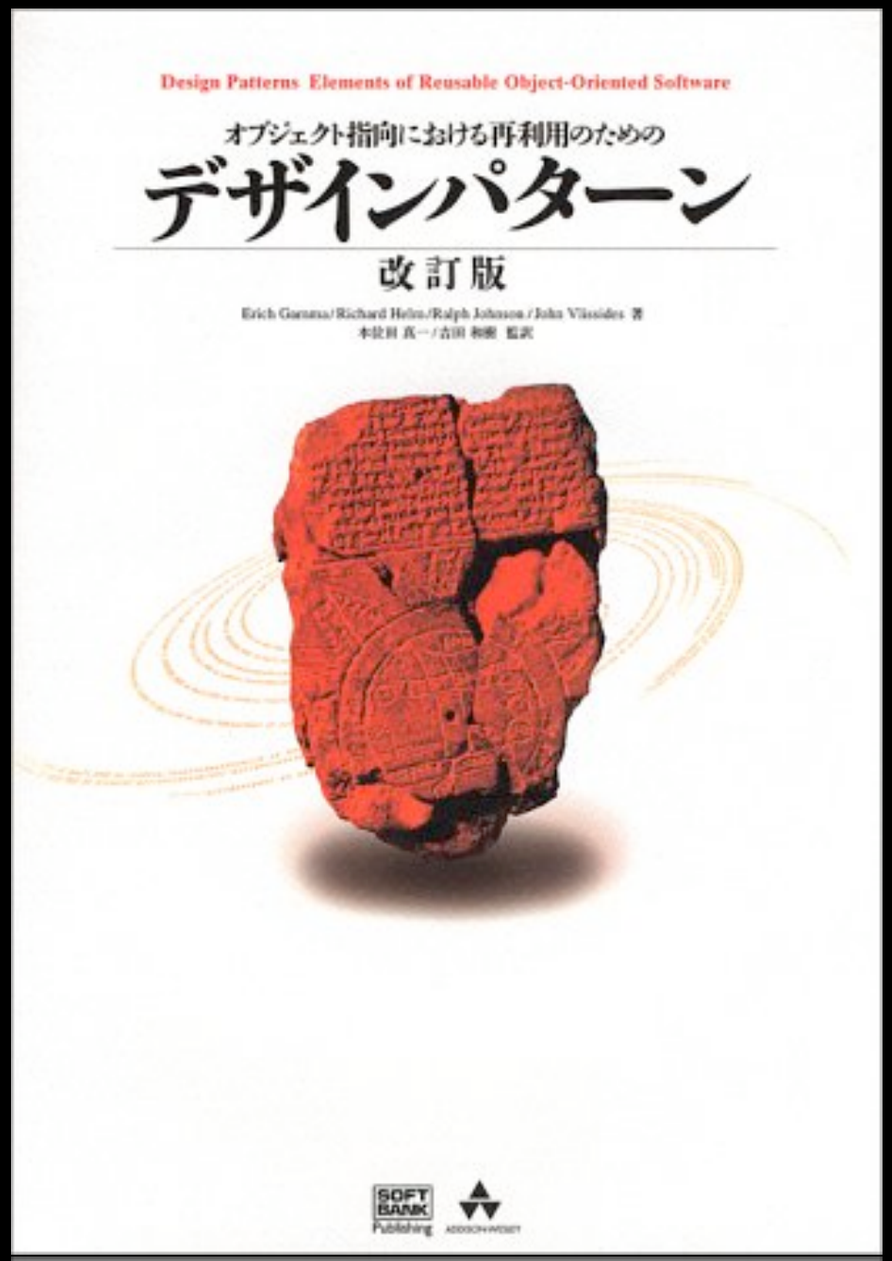

パターン指向
リファクタリング入門
Refactoring to Patterns
ソフトウエア設計を改善する27の作法
ジョシュア・ケリーエブスキー
Joshua Kerievsky
ウルシステムズ 小黒直樹 村上雅 高橋一成
越智典子
日経BP社
デザインパターンを
取り入れて
プログラムを改善しよう
ガンマ、ファウラー絶賛！
名著『デザインパターン』、『リファクタリング』に次ぐ、新たなバイブル登場

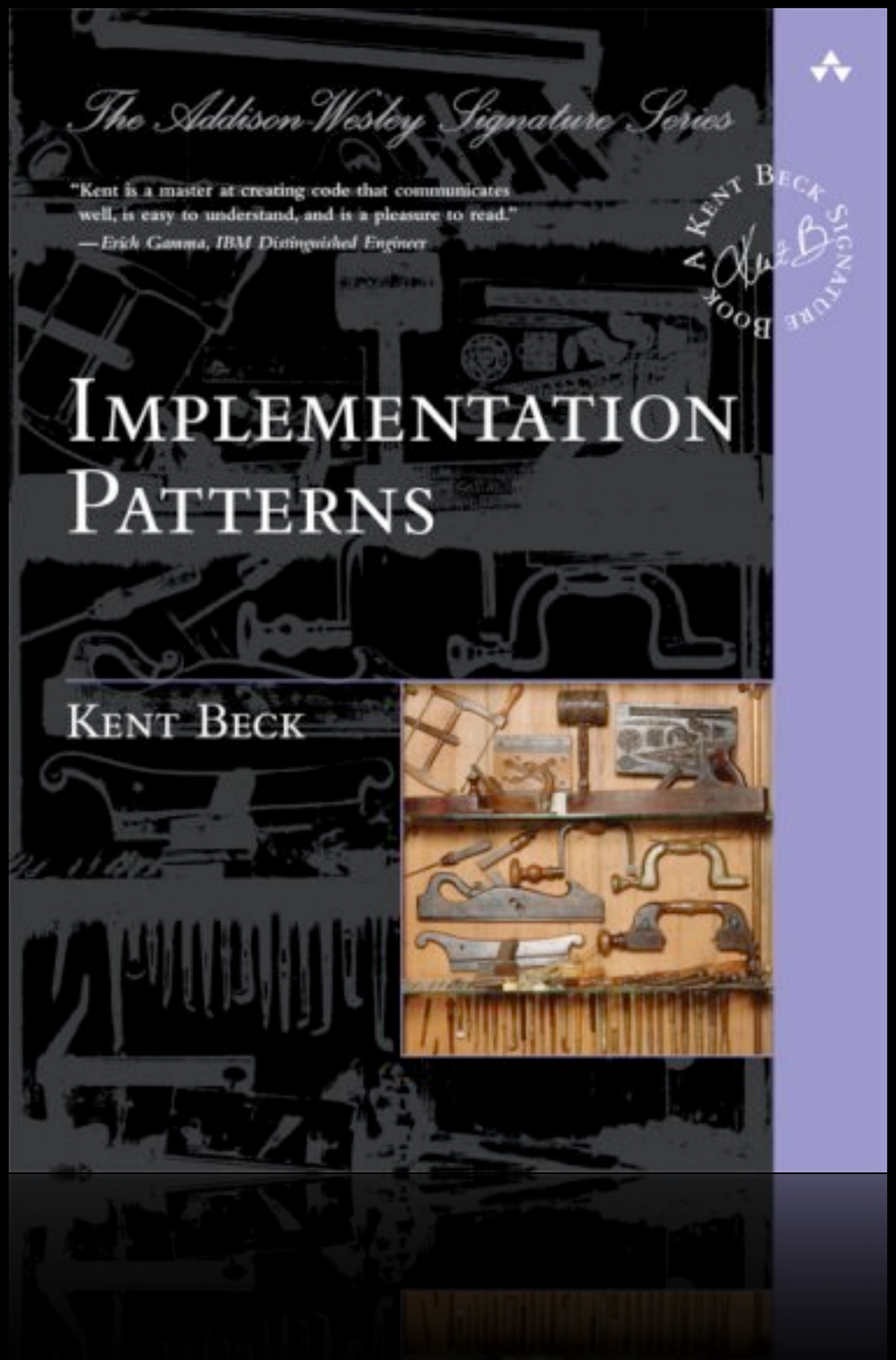
オブジェクト設計


アジャイルソフトウェア
開発の奥義
ロバート・C・マーチン=著
瀬谷啓介=訳
原則・デザインパターン・
プラクティス完全統合
Agile Software Development
オブジェクト指向
開発の秘伝書
アジャイル
＝
俊敏
めまぐるしく変化する仕様要求にさらされながらも
迅速なソフトウェア開発を可能にする「アジャイル」開発
俊敏さと柔軟さを達成するために必要なコンセプトを1冊に凝縮
SOFT
BANK
Publishing


Object Design:
Roles, Responsibilities, and Collaborations
オブジェクトデザイン
ロール、責務、コラボレーションによる設計技法
レベッカ・ワーフスブラック／アラン・マクキーン 著
株式会社オージス総研 藤井 拓 監訳
辻 博靖／井藤 晶子／山口 雅之／林 直樹 訳
SE

デザインパターン


Design Patterns Elements of Reusable Object-Oriented Software
オブジェクト指向における再利用のための
デザインパターン
改訂版
Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides 著
本位田 真一/吉田 和樹 監訳
SOFT BANK Publishing


The Addison Wesley Signature Series
"Kent is a master at creating code that communicates well, is easy to understand, and is a pleasure to read."
—Erich Gamma, IBM Distinguished Engineer
A KENT BECK SIGNATURE BOOK
IMPLEMENTATION PATTERNS
KENT BECK

ご清聴

ありがとう

ございました

Tyler Durden says...