

チェックリスト の12項目に該当 したらTDD脳

Testing Will Challenge Your Conventions

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

XP祭り2007;江戸川区文化センター; 2007-09-01(土)

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://kakatani.com>

IPA未踏ソフトウェア創造事業2006年度下期 千葉PM採択プロジェクト最終成果報告会

- ✓ **Ruby 1.9**～これからの**Ruby**～
- ✓ “エンタープライズ”の現場としての**Ruby**
- ✓ 2007.09.07 (Fri)
- ✓ <http://www.mitou-chiba.org/>

『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.40



連載 第2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/477413192X/kakutani-22>

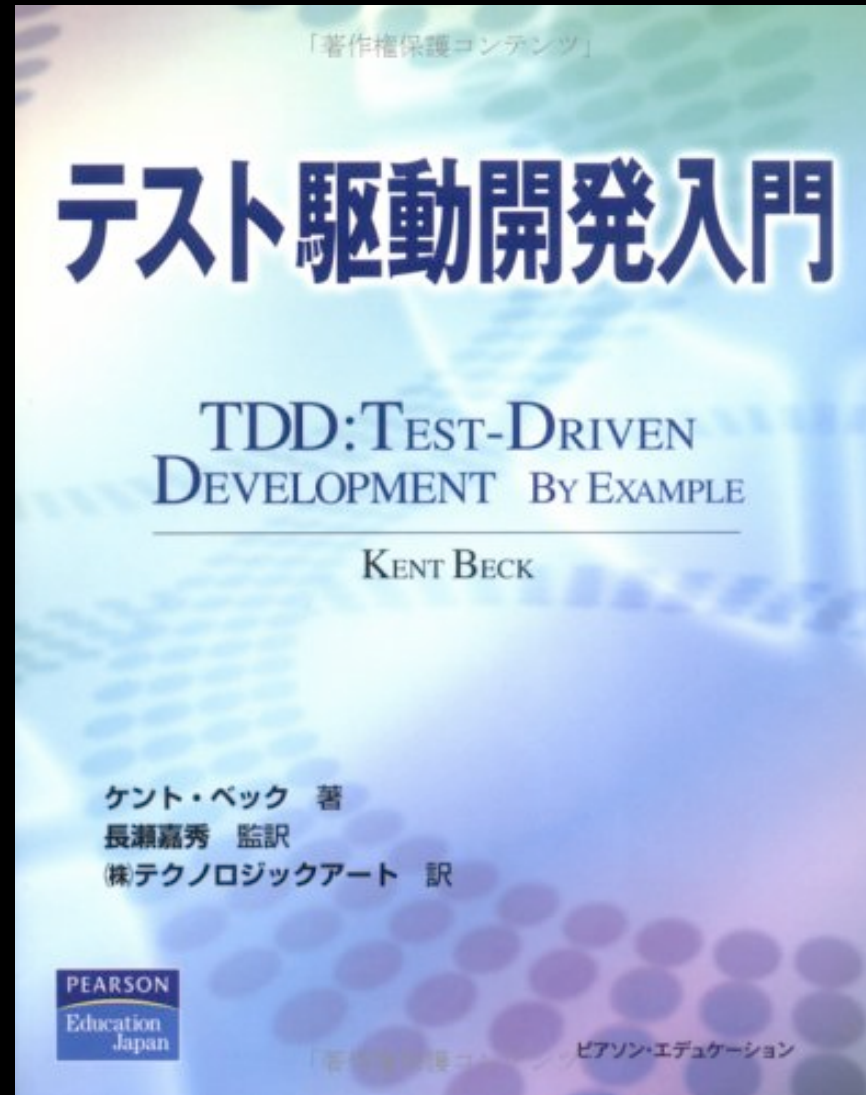
よろしく

お願いいたします

Test Driven Development

テスト駆動開発

テスト駆動開発入門



<http://www.amazon.co.jp/o/ASIN/4894717115/kakutani-22>

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

開発のあらゆる
アクティビティを
構造化する

TDDは

仕事と思考

のスタイル

まずは

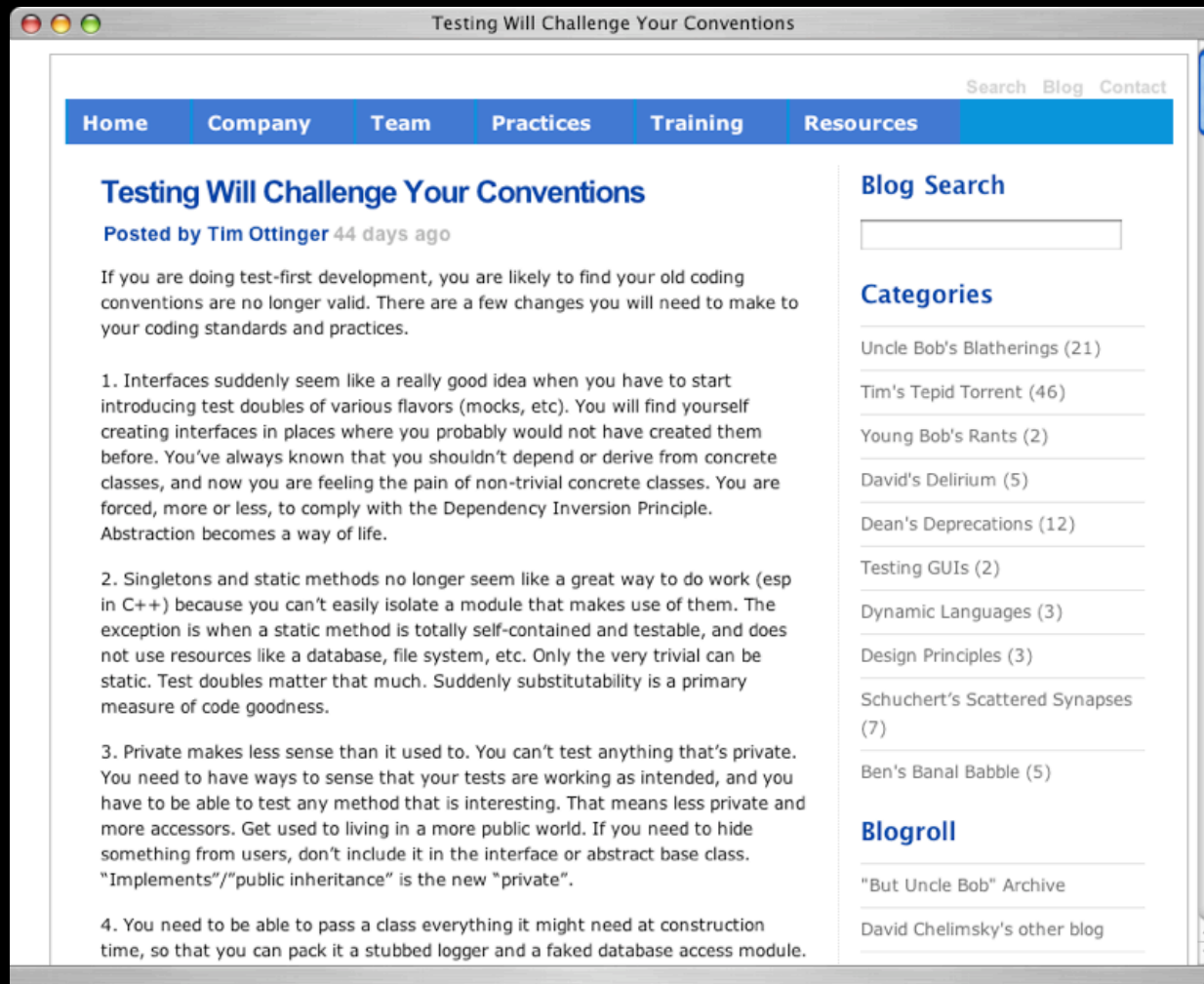
コードから

今日のの

元来夕

Testing Will Challenge Your Conventions

by Tim Ottinger



Testing Will Challenge Your Conventions

Posted by [Tim Ottinger](#) 44 days ago

If you are doing test-first development, you are likely to find your old coding conventions are no longer valid. There are a few changes you will need to make to your coding standards and practices.

1. Interfaces suddenly seem like a really good idea when you have to start introducing test doubles of various flavors (mocks, etc). You will find yourself creating interfaces in places where you probably would not have created them before. You've always known that you shouldn't depend or derive from concrete classes, and now you are feeling the pain of non-trivial concrete classes. You are forced, more or less, to comply with the Dependency Inversion Principle. Abstraction becomes a way of life.
2. Singletons and static methods no longer seem like a great way to do work (esp in C++) because you can't easily isolate a module that makes use of them. The exception is when a static method is totally self-contained and testable, and does not use resources like a database, file system, etc. Only the very trivial can be static. Test doubles matter that much. Suddenly substitutability is a primary measure of code goodness.
3. Private makes less sense than it used to. You can't test anything that's private. You need to have ways to sense that your tests are working as intended, and you have to be able to test any method that is interesting. That means less private and more accessors. Get used to living in a more public world. If you need to hide something from users, don't include it in the interface or abstract base class. "Implements"/"public inheritance" is the new "private".
4. You need to be able to pass a class everything it might need at construction time, so that you can pack it a stubbed logger and a faked database access module.

Search Blog Contact

Home Company Team Practices Training Resources

Blog Search

Categories

- Uncle Bob's Blatherings (21)
- Tim's Tepid Torrent (46)
- Young Bob's Rants (2)
- David's Delirium (5)
- Dean's Deprecations (12)
- Testing GUIs (2)
- Dynamic Languages (3)
- Design Principles (3)
- Schuchert's Scattered Synapses (7)
- Ben's Banal Babble (5)

Blogroll

- "But Uncle Bob" Archive
- David Chelimsky's other blog

<http://blog.objectmentor.com/articles/2007/07/17/testing-will-challenge-your-conventions>

チェックリスト
の**12**項目に該当
したら**TDD脳**

1. インターフェイスがかわいいよインターフェイス
2. Singletonやstaticメソッドは好みじゃない
3. privateへの思い入れがなくなった
4. コンストラクタ引数にはコラボレータを渡したい
5. 小さいメソッドが好き
6. 引き継ぎ？ テストコードはどこ？
7. 自分の書いたコードの最初のユーザは自分だ
8. 最適化を言い出すのはテストを揃えてから
9. テスト全体の実行速度に気を配る
10. 依存関係を少なくすることに気を配る
11. 「わかりやすい」ことがカッコイイ
12. テスト実行を支援するIDEが良いIDE

ひたすら

紹介してくよ

CTD

1. インターフェイスがわいいよ インターフェイス

✓ “インターフェイスに対して
プログラミングする”

✓ テストもプログラミングだよ

✓ テスト容易性ですよねー

✓ Mock, Stub, DIコンテナ

Rubyに無くね?

- ✓ **すべて**がインターフェイス
- ✓ **Duck Typing**
- ✓ インターフェイスは**空気**
- ✓ カッコいい**Mock/Stub**ライブラリたち

CS

2.Singletonやstaticメソッドは 好みじゃない

✓ だって**テスト**しづらいもん

✓ ほんとに**必要**かな？

✓ **テスト**できるなら使うけど

✓ **bliki**:“staticの置き換え”

✓ <http://capsctrl.que.jp/kdmsnr/wiki/bliki/?StaticSubstitution>

3

3. privateへの思い入れが なくなった

- ✓ だってテストしづらいもん
- ✓ インターフェイスが”public”
 - ✓ “published”
- ✓ 実装クラスが”private”
- ✓ テストの資産価値との関連

(4)

4.コンストラクタ引数には コラボレータを渡したい

- ✓ **協調**するオブジェクトを自分で**生成**しない
- ✓ だって**テスト**しづらいもん
- ✓ 単一**責務**の原則(**SRP**)
- ✓ **DI**コンテナ

55

5.小さいメソッドが好き

- ✓ だって**テスト**しづらいもん
- ✓ **10行**を越えると大きいよね
- ✓ メソッドが小さいと、それぞれの**テストも小さくなる**
- ✓ **数**は増えちゃうかもしれないけど
- ✓ 極端に**短い**とかはまた別の話



6.引き継ぎ?

テストコードはどこ?

✓ コメントよりもテストコード

✓ テスト容易性にはテストコードの読みやすさも含まれるよ

✓ ドキュメントのようなコード

を書きたい

575

7.自分の書いたコードの最初の ユーザは自分

- ✓ 自分の書いたコードのテストを書くのは自分だもん
- ✓ 自分がテストしづらいコードは他の人にも使いづらいはず
- ✓ 使いやすいAPIを設計したい



8.最適化を言い出すのは テストを揃えてから

- ✓ **速さ**とテスト容易性とは、
まず**テスト容易性**を選ぶ
- ✓ **最適化**の第**1**原則: **最適化するな**
- ✓ **最適化**の第**2**原則: **まだ最適化するな**
- ✓ プロファイリング



9. テスト全体の実行速度に気を配る

- ✓ 頻繁に実行したいじゃない？
- ✓ 実行速度の目安:
 - ✓ Red/Green/Refactorのサイクルで1秒
 - ✓ ユニットテスト全体で20~30秒
 - ✓ 「10分ビルド」
- ✓ 遅いテストは別実行(それCIで)

(10)

10. 依存関係を少なくすることに気を配る

- ✓ だって**テスト**しづらいもん
- ✓ **小さな**クラスを**たくさん**定義することを**躊躇**しない
- ✓ **協調**を担当するクラスと、**実装**を担当するクラス

(111)

11. “わかりやすい”ことが カッコイイ

- ✓ “わかりやすさ”は“巧妙さ”に
200倍勝る
- ✓ だってテストしづらいもん
- ✓ リファクタリングしやすい
- ✓ 意図が明瞭なテストコード

(12)

12. テスト実行を支援するIDE が良いIDE

- ✓ 開発とはテストの実行である
- ✓ モダンIDE
 - ✓ Eclipse, NetBeans, IDEA
 - ✓ Visual Studioも?
 - ✓ Emacs, Vim

まとめ

- ✓ TDDは**仕事**と**思考**のスタイル
- ✓ **TDD脳**チェックリスト**12**項目
- ✓ 短いので**原文**も読んでね
- ✓ **営業自重**

ご清聴

ありがとうございます

ございました

Happy Testing!