シンクイット
Think IT
THINK IT BOOKS

# これからはじめるTDD
# テスト駆動開発入門

吉谷 愛＝著

TDDがはじめてでも分かる!
新人エンジニアとベテランが会話しながら
テスト駆動開発を攻略

インプレス

- 本書は、インプレスが運営するWebメディア「Think IT」で、「初学者のためのTDD（テスト駆動開発）入門」として連載された技術解説記事を電子書籍およびオンデマンド書籍として再編集したものです。
- 本書の内容は、2015年3月までの情報を基に執筆されています。紹介したWebサイトやアプリケーション、サービスは変更される可能性があります。
- 本書の内容によって生じる、直接または間接被害について、著者ならびに弊社では、一切の責任を負いかねます。
- 本書中の会社名、製品名、サービス名などは、一般に各社の登録商標、または商標です。なお、本書では©、®、TMは明記していません。
- 本書（PDF版）のリスト内における「⇒」は1行での表示を意味します。

# はじめに

本書では、エクストリーム・プログラミング（extreme programming; 以下 XP）のプラクティスのひとつであるテスト駆動開発（てすとくどうかいはつ、test-driven development; 以下 TDD）について、

- 聞いたことはあるけれど内容は知らない方
- 概要は知っているけれど実際に使ったことがない方

といった、主に初学者を対象に、TDD の基本について全 7 章にわたってご説明していきます。

少しだけアジャイルをかじった程度の開発経験の浅いプログラマと、TDD 開発を実践しているプロジェクトリーダーによる会話形式でお届けしますので、RSpec 等のツールを使って TDD 開発をしてはいるものの、正直メリットが今一つ理解できていないという方も、もちろん歓迎です。

本書で使用するプログラムのサンプルコードは下記よりダウンロードできます。http://book.impress.co.jp/books/1114170210_4

# 目　次

# 第1章　エンジニアのスキルを伸ばす“テスト駆動開発”を学んでみよう

## 1.1　はじめに

本書では、エクストリーム・プログラミング（extreme programming; 以下 XP）のプラクティスのひとつであるテスト駆動開発 (てすとくどうかいはつ、test-driven development; 以下 TDD) について、

- 聞いたことはあるけれど内容は知らない方
- 概要は知っているけれど実際に使ったことがない方

等、主に初学者を対象に、TDD の基本について全 7 章にわたって説明していきます。RSpec 等のツールを使って TDD 開発をしてはいるものの、正直メリットが今一つ理解できていないという方も、もちろん歓迎です。

そしてこのシリーズは、**“少しだけアジャイルをかじった程度の開発経験の浅いプログラマ「古谷」が、TDD 開発を実践しているプロジェクトに参画するにあたって、プロジェクトリーダーの「高梨先輩」に、TDD について 1 から学ぶ”**という想定で記述していきます。なお、本書ではプログラミング言語 Ruby で TDD を実現しておりますが、必ずしも Ruby 経験者が対象ではありません。Ruby 未経験者でもプログラミング経験者であれば、ある程度理解できるように考慮して進めてまいります。

# 1.2　テスト駆動開発（TDD）について

**[古谷]**

古谷です。今回は、初心者の私に TDD について優しく教えてください。TDD への興味はあったのですが、スクラムと違って XP や TDD を嬉々として実践しているエンジニアの人達って、なんかレベル高すぎな感じで近寄りがたくて・・。

**[高梨]**

高梨です。今回はできるだけ分かりやすく初学者を意識して説明しますね。まず、TDD はアジャイルソフトウェア開発手法、特にエクストリーム・プログラミングにおいて強く推奨され有名になりました。古谷さんが言う通り、スクラムに比べると、XP 特に TDD は、なんだか敷居が高い感じがしますね。でも、僕は、初学者こそ TDD を学んでほしいし、学ぶべきだと思っています。なぜなら、TDD は、テストスキルだけでなく、設計スキルやリファクタリングのスキルを伸ばします。正しく TDD を導入すれば、それは確実にエンジニアを育ててくれます。

**[古谷]**

そうですか？ TDD でテストのスキルが伸びるのはイメージできますが、設計スキルやリファクタリングスキルが伸びるのはぴんときません。

**[高梨]**

後半になれば、意味が分かっていただけると思います。では、一番はじめは、テスト駆動開発の「価値」と「原則」について認識を合わせましょうか。これを見てください。何か質問や感想があればお願いします。

**[古谷]**

えーっと、「**インクリメンタルな設計**」ってなんですか？

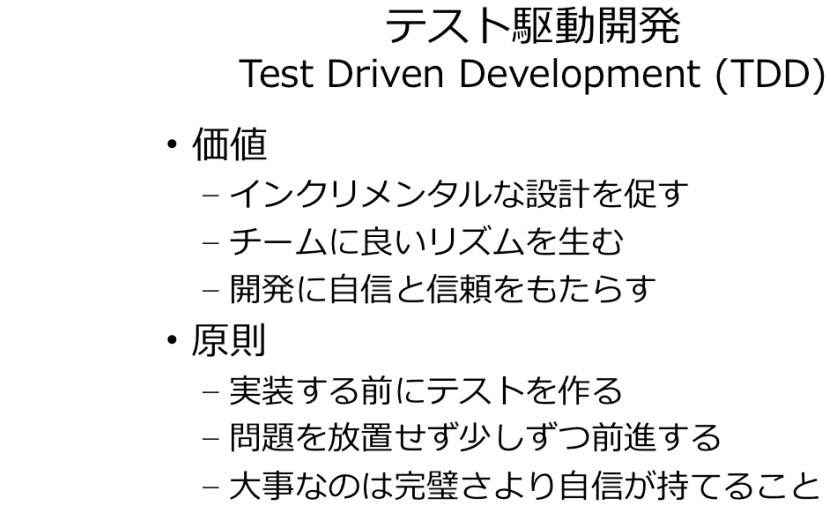


図 1.1　テスト駆動開発 Test Driven Development（TDD）

**[高梨]**

それについては、後で説明します。

**[古谷]**

あとは・・・「自信」という単語が 2 回出てきますね。

**[高梨]**

そうです。完璧さを求めてとにかくテストケースを増やすことが TDD の目的ではありません。書いたコードの不安感がなくなり動作に自信が持てることが大事になります。TDD は一言で言ってしまえば、**プログラマが、自分が書くコードに自信を持ちながら少しずつ進んでいく開発手法**ともいえます。そこでさっき古谷さんが質問した「インクリメンタル設計」が関係してきます。そのまえにまず「インクリメンタル設計の対極の立場にある「ビッグバン設計」についてお話ししますね。こちらを見てください。

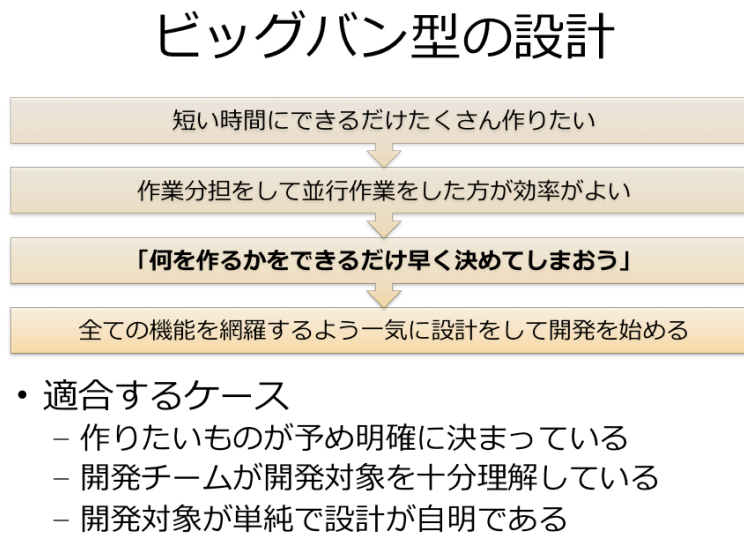


図 1.2　ビッグバン型の設計

**[古谷]**

あれ？これって、ウォーターフォールのことですか？

**[高梨]**

はい。そのため、「作りたいものが予め明確に決まっていない」場合、このやり方だと、よほど運が強く予見力のあるリーダーがいない限り、あらかた出来上がってしまったとおもっていたのに、中盤以降で仕様変更が頻発し、デスマーチまっしぐらになってしまう可能性があります。また、「作りたいものは予め決まっている」ようでも、フタを開けるとそうではないプロジェクトがとても多い。古谷さんもご存じのように、ウォーターフォールでも大半の開発は途中で仕様変更が入ります。もちろんそれが想定の範囲のブレならいいのですが、大きく設計を変更しなければならないようだと、手戻りが大きすぎて破綻します。結果、現場でコードのコピペなど力技でごまかして「想定の範囲内」に見せてしまうケースって、実はかなり多いと思うんですよね。本来保守のことを考えると、設計を見直すべきなのですが、その時間がなくて、ごまかしが横行しているのが現実ではないでしょうか。

**[古谷]**

確かにそんな感じですね…。そういえば、仕様変更に伴っての大幅な設計変更が発生するのを防ぐために、開発側がお客様に「仕様凍結」と言って、仕様変更を受け付けないようにしてしまうことをよく聞きます。自分が 1 プログラマの時は、そういう動きをしてくれるリーダーに感謝していましたが、結局出来上がったのはお客様が本来求めていないシステムになるのであれば、本末転倒ですよね。

**[高梨]**

そう、要は、本当はビックバン型の設計に向いていないものまで、ビックバン型でやってしまっているということが悲劇の温床なのです。例えば「既存システムを新システムに置き換える」ようなプロジェクトは仕様が決まっていて、一見ビックバン型に向いていそうですが、経験上色々な関係者の新システムへの期待が絡んで、実は非常に混沌としていることが多いんですよね…。

すみません、ちょっと話が長くなりました。ではいよいよ、ご質問にあったインクリメンタル設計についてご説明します。こちらをご覧ください。

**[古谷]**

これは・・・まさにアジャイルですね。

**[高梨]**

そうですね。インクリメンタルな設計の場合、まさにインクリメントしていく感じで、**最低限必要な機能から「少しずつ」開発を進め、その過程で設計自体も「少しずつ」進化させていきます。**

**[古谷]**

ありがとうございます。まだ TDD とインクリメンタルな設計が結びついてはいませんが、「インクリメンタルな設計」自体のイメージはなんとなくわきました。

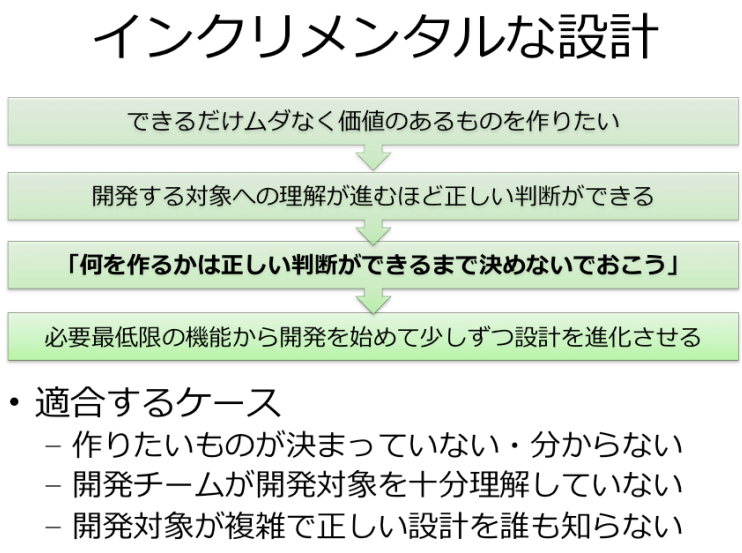


図 1.3　インクリメンタルな設計

**[高梨]**

では、具体的にどれくらい少しずつやるか、さっそく実践して感覚をつかんでいきましょうか？

**[古谷]**

えっ!? 私まだ TDD についてこれ以上のこと何にも知らないですけど!!

**[高梨]**

それでは、まずざっくり TDD の流れについて説明していきましょう。こちらを見てください。

**[古谷]**

三角形になっていますが、まずはどこからスタートですか？

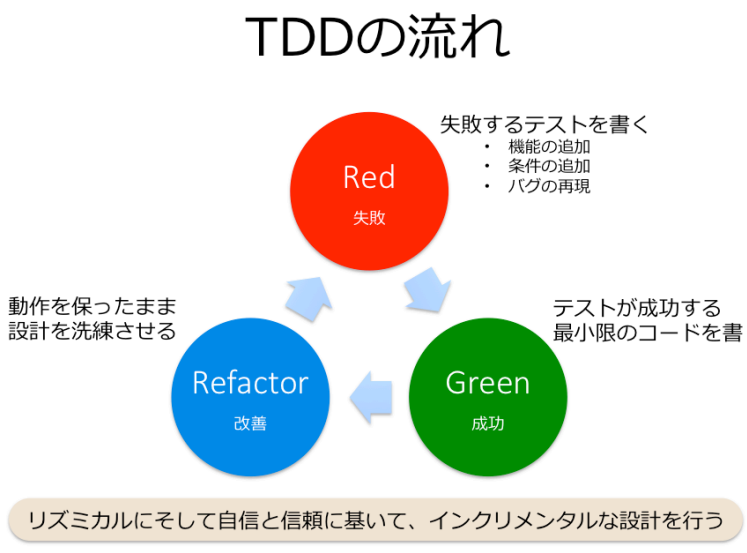


図 1.4　TDD の流れ

**[高梨]**

Red（失敗）の「失敗するテストケースを書く」からです。**TDD の場合、Red（失敗）と Green（成功）を繰り返し、Refactor（改造）の工程を経て再度 Red（失敗）に移ります。**まあでも百聞は一見に如かずですから、やってみましょう。今回は Ruby 言語での開発を想定して、簡単に TDD のツールのご紹介をします。

**[古谷]**

えー!? 展開早くないですかー!? あ、あと、私あまり Ruby 詳しくないですけど！

**[高梨]**

安心してください、最初はそんなに高い Ruby 言語の知識がなくても大丈夫です。TDD だけでなく Ruby の知識もついて一石二鳥ですよ。

**[古谷]**

は、はい・・・。

**[高梨]**

それでは、こちらが Ruby の場合に一般的に使われている TDD のツールです。

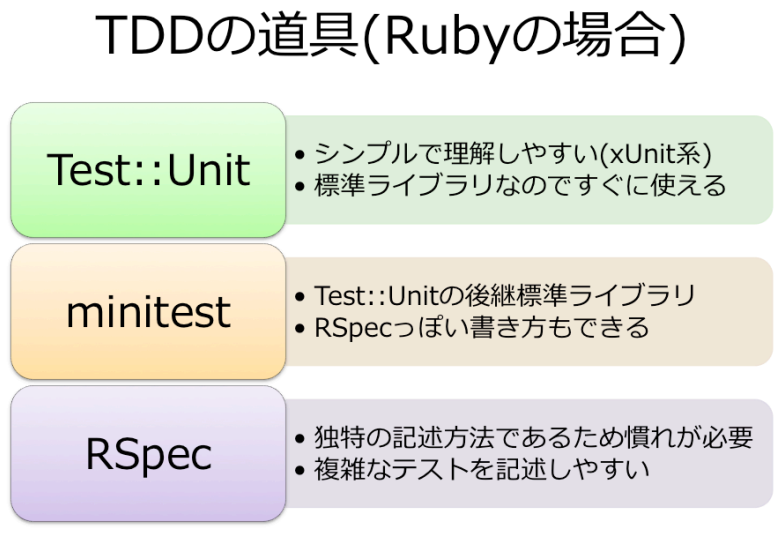


図 1.5　TDD の道具（Ruby の場合）

**[古谷]**

あ、「なんちゃら Unit」は、他の言語でもよく目にしますね。

**[高梨]**

そうです、あれと同じイメージで OK です。今回はこの中の minitest を使用します。Ruby1.9.3 から標準で提供されていますので、Ruby をインストールしていたら、何もしな