ITジャーナリストや現役書店員、編集者が選ぶ デジタル人材のための Book Review ブックレビュー

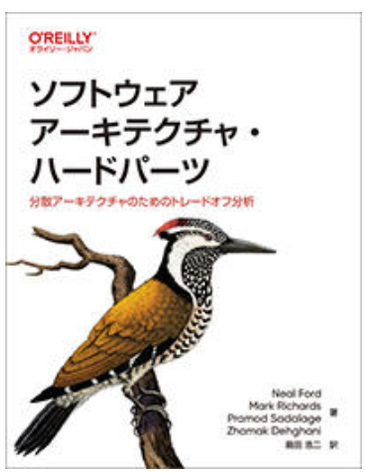
今月の書籍

レビュワー:高橋 征義

・『ソフトウェアアーキテクチャ・ハードパーツ』

・『テスト駆動開発』

分散システムのバリエーションとトレードオフを掘り下げる



本の詳細はこちら
（外部サイト）

本書は『**ソフトウェアアーキテクチャの基礎 —— エンジニアリングに基づく体系的アプローチ**』の**続編**にあたる書籍である。著者は前著の著者陣と新たに加わった2名の計4名。前著は「～の基礎」というだけあって、概論からソフトスキルまで含む入門編のような位置づけだったが、そちらと比較すると**本書は応用編**と言える。

タイトルに含まれる「ハード」という言葉は、「難しい」という意味に加え、「**堅い＝変更するのが大変**」という意味が込められている。つまり前著より難しいところに踏み込むわけで、いささかの手応えは感じるかもしれない。もっとも、読み手としてはとても**気になるところを掘り下げ**てくれているため、**前著は下準備で本書が本番**、という趣きもある。ある程度**ソフトウェアアーキテクチャ**に知識がある人であれば、前著は飛ばして本書から読み始めてもよいかもしれない。ただし本書で比較されるアーキテクチャのバリエーションは**前著第II部で詳説**されているものなので、適宜参照した方が理解は進みそうである。

また、今回は「**Sysops Squadサーガ**」という**架空のサービスの事例**が随所に登場する。ここでは**サービスに生じた課題とそれを解決するアーキテクチャ**の検討が、**物語の対話形式**で具体的に語られており、取り扱う題材がハードなものであっても、**親しみやすくなるよう工夫**されている。

本書のサブタイトルである「**分散アーキテクチャのためのトレードオフ分析**」にある通り、本書は**分散アーキテクチャ**が論点の中心となる。とはいえ、最近流行のマイクロサービスやイベント駆動アーキテクチャを称賛したり、その導入方法を解説したりする、**といった類の書籍ではない**。本書のポイントはサブタイトルの後半にもなっている「**トレードオフ分析**」である。

本書第1章の章タイトルは「**『ベストプラクティス』がないとどうなる？**」である。この章タイトルの通り、本書のスタンスは、「**分散アーキテクチャにはベストなアーキテクチャというものはなく、全てはトレードオフである**」というものだ。つまり**どんなアーキテクチャにもメリットとデメリット**があり、それと現在の状況や環境を元に**最悪（ワースト）ではない選択肢を選ぶこと**がアーキテクトの役割であるとされる。

本書は2部構成になる。前半である**第I部「分解する」**では、**アーキテクチャの分解**、**データの分解**、**サービスの分解**について語られる。明確な構造が崩れてしまって混沌としたソフトウェアは「**巨大な泥団子**」とも言われるが、このようなソフトウェアには分散アーキテクチャ以前に、まず**適切な粒度に分解し構造を整理**しなければならない。それを怠って分散アーキテクチャを導入すると「**分散した巨大な泥団子**」になる、という厳しい現実が語られる。

これを回避するため、コードやデータを分解した上で、最終的には**サービスとして分解**するのだが、このサービスをどう分解するかも難しい。分散アーキテクチャのさまざまなバリエーションに加えて、サービスの粒度をどうするかという問題については、**粒度分解要因と粒度統合要因**が紹介される。つまり、**サービスを分解したくなる要因**があると同時に、**サービスを統合したくなる要因**もあるのだ。まさに**トレードオフ**である。

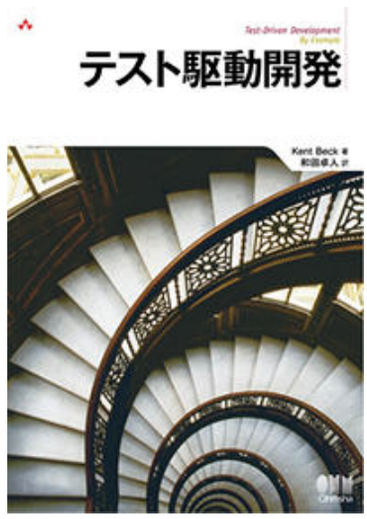
もちろん、分解された各サービスは何かしらの方法で**通信し連携しあう必要**がある。そちらは後半である**第II部「つなぎあわせる」**でさらに細かく解説されるのだが、各章で語られる**分散処理**、とりわけエラーが起きた際の**リカバリ方法**（の難しさ）についての解説を読むと、分散アーキテクチャの**しんどさ**がしみじみと伝わってくる。さすがの**ハードパーツ**とも言えるが、それでもさまざまな分類やそれぞれのメリット・デメリットが分かるだけでも、**アーキテクチャを決定する際の心強い援軍**として役立つだろう。

分散アーキテクチャに興味のある方、悩んでいる方なら、前著に引き続き、読んでおきたい一冊である。

『ソフトウェアアーキテクチャ・ハードパーツ』

著者：ニール・フォード、マーク・リチャーズ、ピラモド・サドラージ、ゼマック・ディガニ
翻訳：島田 浩二
出版社：オライリー・ジャパン
https://www.oreilly.co.jp/books/9784814400065/

自動テストの時代に、ソフトウェア開発を駆動するための設計技術



本の詳細はこちら
（外部サイト）

本書は**ケント・ベック**による**テスト駆動開発（TDD）の古典**とも言える『**Test Driven Development: By Example**』を、2017年に新たに翻訳し直し、**新装版として復刊した本**である。

再刊にあたっては、JavaのコードをJUnit 5、Pythonのコードを**Python3で書き直し**たり、本文中の**サンプルコードを増やして読みやすく**する等の**独自の工夫**が加えられている（本書「訳者あとがき」より）。2020年代に読むには**原著よりもおススメ**かもしれない。

本書は**3部構成**になっており、**複数国の通貨を扱うコードを題材にテスト駆動開発を実践**していく第1部、**テスティングフレームワークを自作しながら（！）そのしくみを解説**する第2部、これまでの復習も兼ねつつ、**テストで使われるパターンや方法を挙げていく**第3部、そして訳者により書き下ろされた「付録C　訳者解説：**テスト駆動開発の現在**」を含めた3つの付録からなる。

原著の刊行から20年ほど経った現在、テストの**自動化を取り巻く環境は大きく変わった**。CI/CDはめずらしいものではなくなり、ソフトウェアのフレームワークにはテストを実行するしくみを**あらかじめ持っている**ものも少なくない。日常的に開発をしながらテストを書き、自動で実行させるための**土台は大幅に改良**されて久しい。その意味では、とりわけテスティングフレームワークを自作する第2部は、最初は軽く読み流して、既存のxUnitツールを使えばよいかもしれない。

しかしながら、TDDが一般的になっているとは言えない。「**テストを自動で実行できる・していること**」と「**開発をテストで駆動すること**」には大きな違いがある。いわゆる**リグレッションテスト（回帰テスト）**、つまりコードの挙動が変わっていないかを知るためのテストコードを、**プロダクションのコードを書くよりちょっとだけ早く書く**だけでは、本書が目指した「**テスト駆動開発**」とは**言えない**のではないか？……このあたりの**テストの受容や意味の希薄化・形骸化**について、もっとも**網羅的で入手しやすい日本語の文献は、本書の「付録C」**だと言っても過言ではない。この付録Cだけでも本書を読む価値はあるのではないか。

では、今あらためて「**テスト駆動開発**」とはどのようなものなのかを知るにはどうすればよいか。そのためにおすすめしたいのが**本書の第1章**である。ケントがコードを前に、あれやこれやと考えながらコードを書いたり消したりしていく、その試行錯誤からは、**まさしくテストによって開発が駆動されていく様子**が力強く伝わってくるはずだ。

本書の翻訳を行った**t-wadaさんこと和田卓人氏**は**TDDの第一人者**として知られている。そのきっかけとなったのも本書であり、大変な開発現場に放り込まれて帰る途中の深夜バスの中で原著を写経しながらTDDをマスターしたという逸話もよく知られている。あのt-wadaさんにもそんな時代があったのか……という不思議な感慨も覚えるが、それからTDDの布教に励み、一時期は翻訳書が品切れで入手困難になっていたことを嘆いていた氏が、ついに本書の翻訳を手掛け、新装版として復活させたことは慶賀に堪えない。

デジタル人材のための ブックレビュー

テストコードをこれから書き始めたい人はもちろん、テストを書いてはいるけれど今ひとつしっくりこない、という人にも、改めて本書を勧めたい。

『テスト駆動開発』

著者：ケント・ベック
訳者：和田 卓人
出版社：オーム社
https://www.ohmsha.co.jp/book/9784274217883/

今月のレビュワー

高橋征義（たかはし・まさよし）

札幌出身。Web制作会社にてプログラマとして勤務する傍ら、2004年にRubyの開発者と利用者を支援する団体、日本Rubyの会を設立、現在まで代表を務める。2010年にITエンジニア向けの技術系電子書籍の制作と販売を行う株式会社達人出版会を設立、現在まで代表取締役。著書に『たのしいRuby』（共著）など。好きな作家は新井素子。