

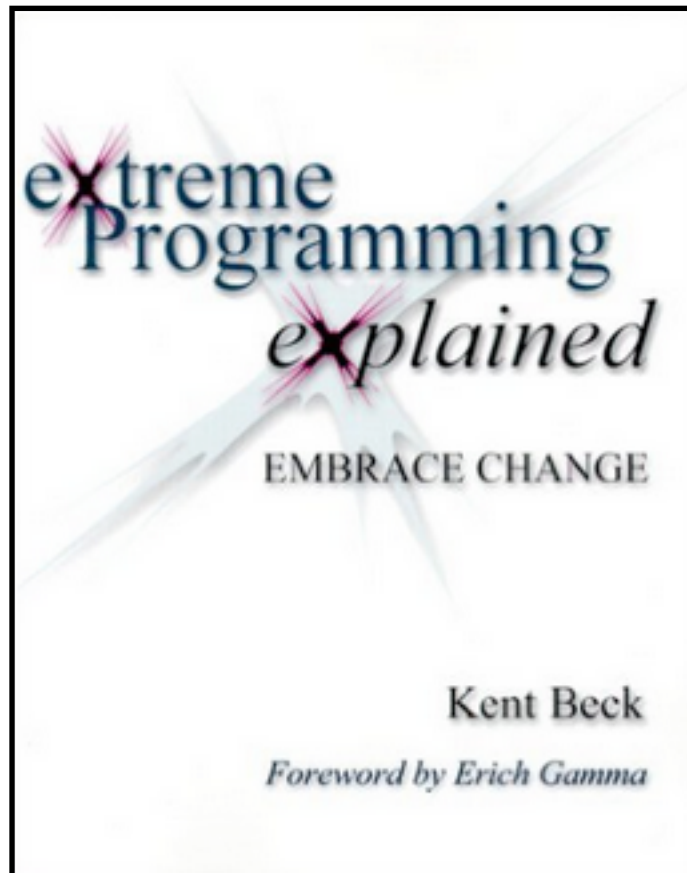
A dynamic splash of blue water with many droplets and bubbles, creating a sense of movement and freshness. The water is splashing upwards and outwards from the center.

**XP lives, XP dies,
XP lives again !!**

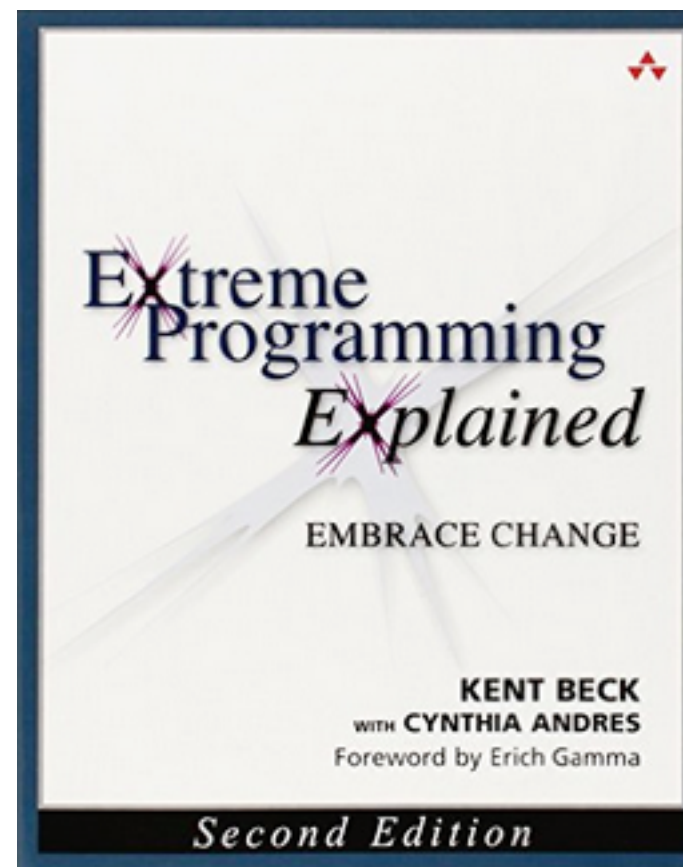
2015年9月
XP祭り2015

ワイクル株式会社
角征典 (かど まさのり)
kado.masanori@waicrew.com

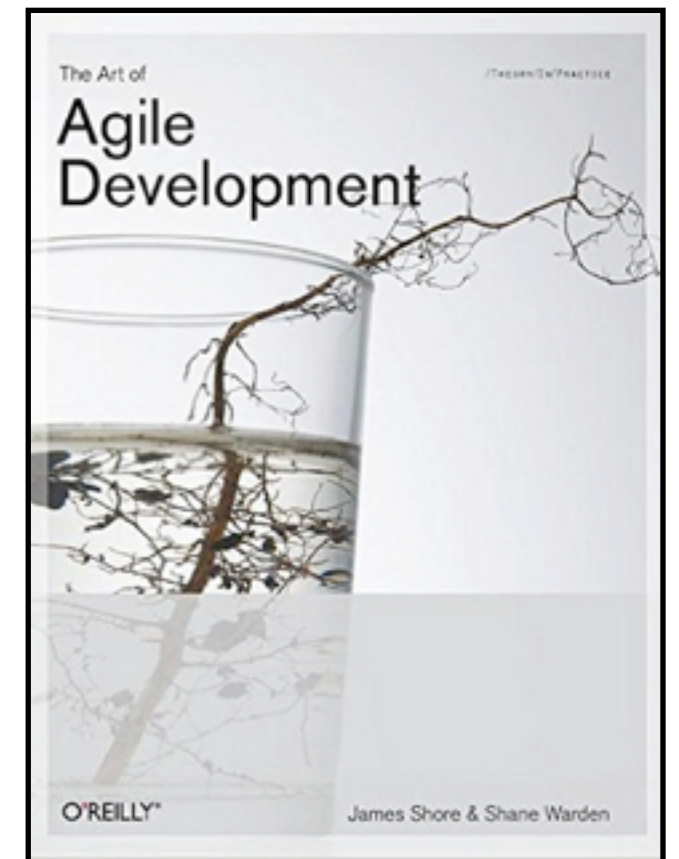
XP lives (past)



1999/10



2004/11



2007/11

翻訳もあるよ

当時のツールたち

(おじさんたちに聞いてみよう)

JUnit

1997-



2004-

cruisecontrol

2001-



2000-



trac

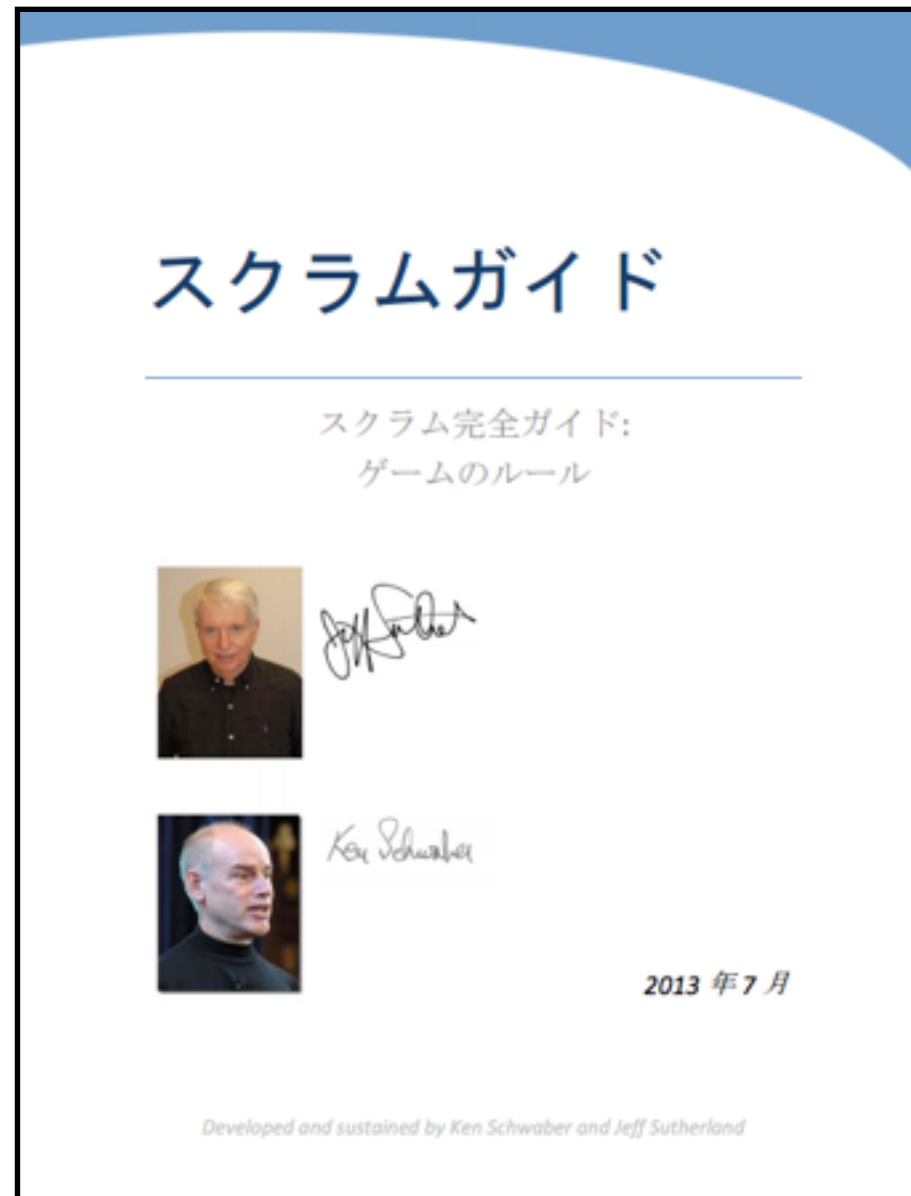
Integrated SCM & Project Management

2005?-



2004?-

XP dies ... and Scrum lives



Scrum lives ...



By 2006,
we had **30,000** CSMs.

By early 2009, there
were more than
60,000 CSMs.

<https://www.scrum.org/About/Origins>

... and gets dying

(自分で作った組織に追い出される)



(2009年8月に辞任勧告)

in August 2009 when the
Scrum Alliance board of
directors unanimously asked
for my **resignation**

<https://www.scrum.org/About/Origins>

ダメなスクラム (へ口へ口Scrum)

FlaccidScrum



Martin Fowler

29 January 2009

There's a mess I've heard about with quite a few projects recently. It works out like this:

- They want to use an agile process, and pick Scrum
- They adopt the Scrum practices, and maybe even the principles
- After a while progress is slow because the code base is a mess

<http://martinfowler.com/bliki/FlaccidScrum.html>

Agile is Dead

PragDave

Programmer. Author. Publisher. Speaker. Bad Gardener.

Home Archives CodeKata

Search

MAR 4TH, 2014 | [COMMENTS](#)

Agile Is Dead (Long Live Agility)

Thirteen years ago, I was among seventeen middle-aged white guys who gathered at Snowbird, Utah. We were there because we shared common beliefs about developing software, and we wondered if there was a way to describe what we believed.

It took less than a day to come up with a short list of values. We published those values, along with a list of practices, as the [Manifesto for Agile Software Development](#):

- Individuals and Interactions* over Processes and Tools
- Working Software* over Comprehensive Documentation
- Customer Collaboration* over Contract Negotiation, and
- Responding to Change* over Following a Plan



Recent Posts

- [Personal Area Stereo](#)
- [How Do You Use a Catalog?](#)
- [Proud to Be an American](#)
- [Peroxide Us an Opportunity](#)
- [A Simple Elixir Macro](#)
- [Thinking in Transforms—Handling Options](#)
- [Elixir: State Machines, Metaprogramming, and Generating Tests](#)
- [Tony Benn's Five Questions for the Powerful](#)

<http://pragdave.me/blog/2014/03/04/time-to-kill-agile/>

達人デイヴの激怒事件

2人の"アジャイルコンサル"の会話

▶  <10年プログラム書いてないよw

▶  <俺なんか20年書いてないよwww

▶  <なめとんのか💢

基調講演の予定を変更

「Agile is Dead」

<http://www.infoq.com/jp/news/2014/11/pragmatic-dave-agility>

アジャイルの意味が失われた

Andy

Andy Hunt, Pragmatic Programmer, Author, Speaker, Publisher

« L-mode/R-mode and the Default Mode Network (DMN) | Main | It's an Experiment »

May 06, 2015

The Failure of Agile

I am proud to be one of the 17 founders/authors of the [The Agile Manifesto](#) back in 2001. I think it provided a jolt of energy, hope of a better way of doing things, of creating software and making the world work better. It was a pivotal turning point.

But in the 14 years since then, we've lost our way. The word "agile" has become sloganized; meaningless at best, jingoist at worst. We have large swaths of people doing "flaccid agile," a half-hearted attempt at following a few select software development practices, poorly. We have scads of vocal agile *zealots*—as per the definition that a zealot is one who redoubles their effort after they've forgotten their aim.

And worst of all, agile methods themselves have not been agile. Now there's an irony for you.

How did we get into this mess?



<http://blog.toolshed.com/2015/05/the-failure-of-agile.html>



NDC
15-19 June-Oslo, Norway

THAT'S WHAT 'AGILE' IS SUPPOSED TO BE.

- Not backlog grooming
- Not standup meetings
- Not burndown charts
- Not velocity
- Not TDD



<https://vimeo.com/131410262>

GROWS

A Modern Software Development Suite. Because one size does not fit all.

<http://growsmethod.com/>

名前だけアジャイルが多すぎる

Unfortunately, I find too many people and organizations that are agile in name only and don't really embrace the values or practices. There will continue to be failed adoptions, great successes and sustained improvement efforts.



<http://sdtimes.com/james-grenning-on-agile-work-still-needs-to-be-done/2/>

おかしいアジャイルが増えている



<http://www.akitaonrails.com/2010/06/16/railsconf-2010-video-interview-robert-martin-english>

デイヴ様からのご提案



<http://www.thoughtworks.com/talks/the-death-of-agile>

1. 「アジャイル」を広めよう

まずは言葉に興味を持ってもらおう

2. みんなに届けられる商品

書籍、研修、コンサル、認定資格、カンファレンス

3. 難しく見えるような工夫

カタカナ用語、新しい役割、新しいメトリクス

4. 開発者に売りつけよう👍

こんなに魅力的なアジャイルを使っていない開発者はバカだ！

5. 企業に売りつけよう👍👍

これからは「エンタープライズアジャイル」の時代です！

6. IT業界以外にも！👍👍👍

アジャイルで組織変革！ ハードウェアもアジャイル！ リーン！

デイヴ様からのご提案



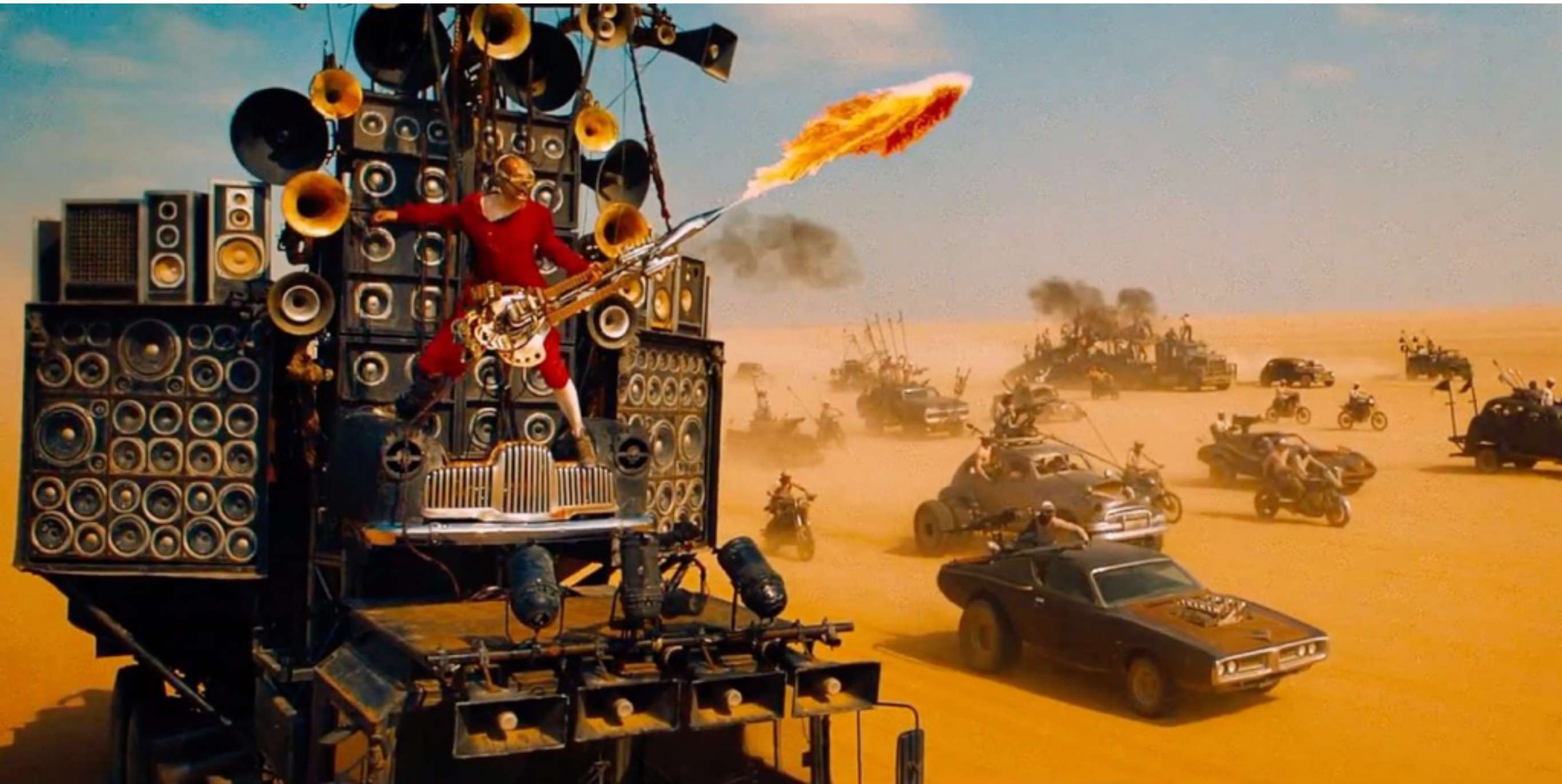
<http://www.thoughtworks.com/talks/the-death-of-agile>

一方的に「与える」モデル💀



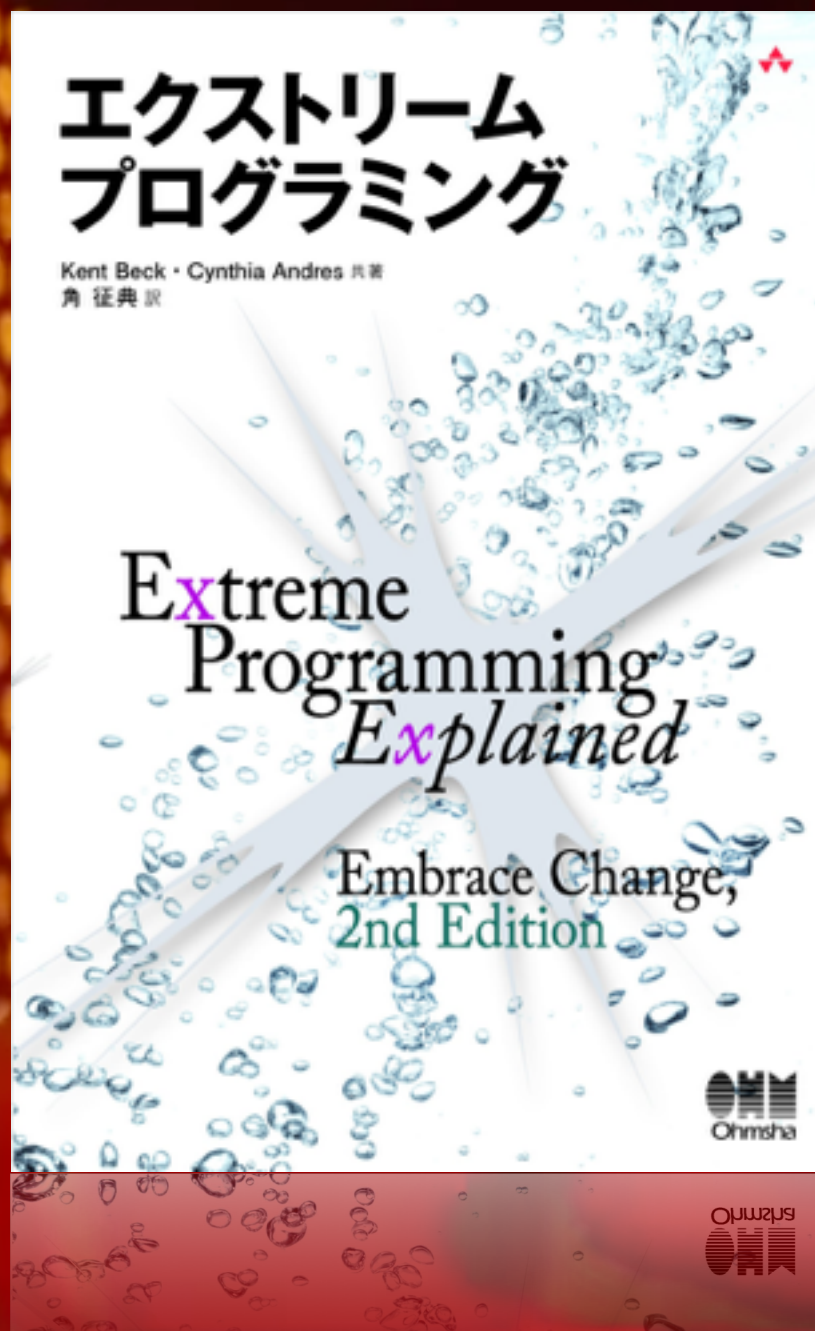
Mad Max: Fury Road

Welcome to the Wasteland (again)



Mad Max: Fury Road

XP lives again !!



Mad Max: Fury Road

XPE2ndのいいところ

やり方が書いてない！

(自分たちで考えるしかない)

教え方がわからない！

(自分たちで考えるしかない)

私からのご提案

これから 「"俺も"XPを導入したい」 という皆様へ

▶ 0. プリクエル 

▶ 1. 人間性を重視したときめき 

▶ 2. 適応型計画づくりとその技術 

▶ 3. もっとエキストリーム 

お前は誰だ



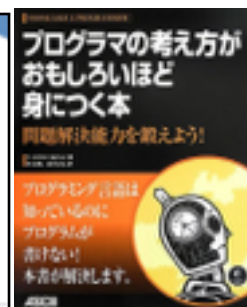
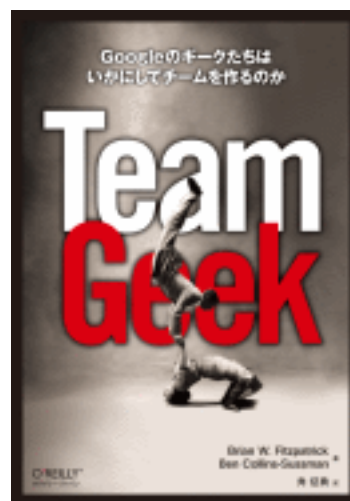
▶ 角 征典 (@kdmsnr)

▶ ワイクル株式会社 (取締役、プログラマ)

・アジャイル開発／リーンスタートアップの導入 **研修** & **コンサル**

▶ 東京工業大学大学院理工学研究科 (特任講師)

▶ **書籍** 翻訳



プログラミングチョットデキル (本当の意味で)

▶ 公開できるものは少ない……。

▶ Re:VIEW

- <https://github.com/kmuto/review>
- 組版ツール。XPE2ndもぜんぶコレ。

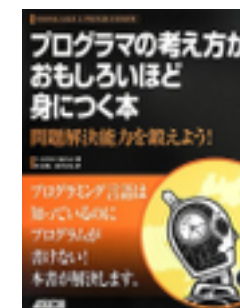
▶ 卒業制作っぽいやつ

- <https://github.com/jointry/jointry>
- Scratchモドキのペアプロツール。

▶ プログラミングの本



<http://www.slideshare.net/kenshimuto/review-36769754>



20分経過

0. プリクエル



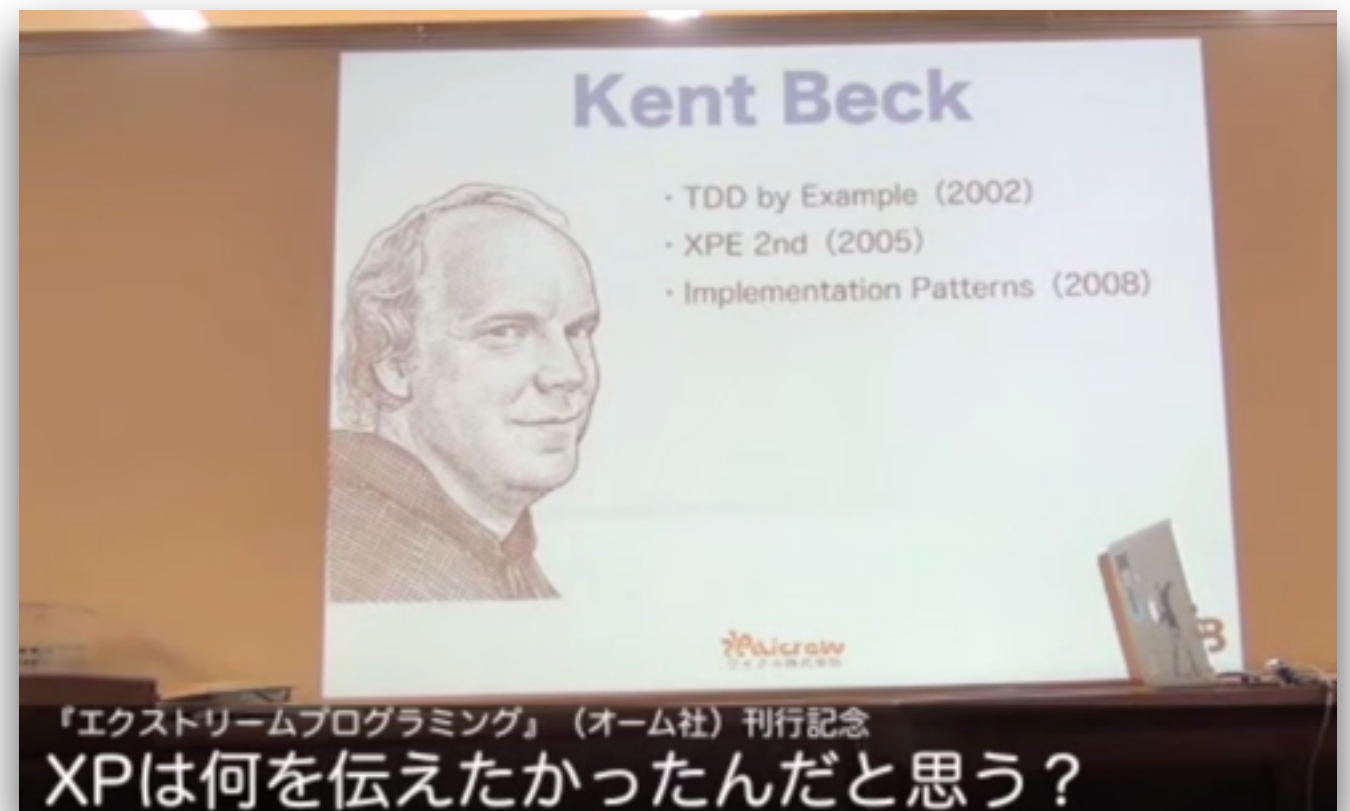
すでに通った道（あとで見てね）

今日はこれとは違ったことを話します

XPは 何を伝えたかったんだと思う？

2015年6月26日
ジュンク堂書店池袋本店

ワイクル株式会社
角征典（かど まさのり）
kado.masanori@waicrew.com



<http://www.slideshare.net/kdmsnr/xpjunkudo-20150626>

https://www.youtube.com/watch?v=YRFWVVS_2Epo

アジャイルマニフェストの歴史

History: The Agile Manifesto

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground and of course, to eat. What emerged was the Agile Software Development Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

Now, a bigger gathering of organizational anarchists would be hard to find, so what emerged from this meeting was symbolic a *Manifesto for Agile Software Development* signed by all participants. The only concern with the term *agile* came from Martin Fowler (a Brit for those who don't know him) who allowed that most Americans didn't know how to pronounce the word agile.

Alistair Cockburn's initial concerns reflected the early thoughts of many participants. "I personally didn't expect that this particular group of agilites to ever agree on anything substantive." But his post-meeting feelings were also shared, "Speaking for myself, I am delighted by the final phrasing [of the Manifesto]. I was surprised that the others appeared equally delighted by the final phrasing. So we did agree on something substantive."

Naming ourselves "The Agile Alliance," this group of independent thinkers about software development, and sometimes competitors to each other, agreed on the *Manifesto for Agile Software Development* displayed on the title page of this web site.

But while the Manifesto provides some specific ideas, there is a deeper theme that drives many, but not all, to be sure, members of the alliance. At the close of the two-day meeting, Bob Martin joked that he was about to make a "mushy" statement. But while tinged with humor, few disagreed with Bob's sentiments that we all felt privileged to work with a group of people who held a set of compatible values, a set of values based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work. At the core, I believe Agile Methodologists are really about "mushy" stuff about delivering good products to customers by operating in an environment that does more than talk about "people as our most important asset" but actually "acts" as if people were the most important, and lose the word "asset". So in the final analysis, the meteoric rise of interest in and sometimes tremendous criticism of Agile Methodologies is about the mushy stuff of values and culture.

<http://www.agilemanifesto.org/history.html>

**Q. 「アジャイル」を
提案したのは誰？**

A. マーティン・ファウラー



<https://www.flickr.com/photos/pragdave/173640462>

なぜなら発音が違うから

▶  アジャイル

<http://www.macmillandictionary.com/dictionary/american/agile>

▶  アジル

<http://www.macmillandictionary.com/dictionary/american/agile>

若い人は知らないかも……。

FrontPage

ここは、[Martin Fowler's Bliki](#)の翻訳Wikiです。

Martin Fowler氏本人の許可を得て公開しています。Wikiですので、どなたでも参加可能です。ご自由にページの追加、修正、変更を行ってください。

- まずは [お読みください](#) をどうぞ。
- ご意見は [ご意見箱](#) までどうぞ。

まだ翻訳していないページは、[InHandOrNot](#)で確認できます。是非「新規作成」してください :-)。

13" 1) 1) 1) 1) 1)



—Martin

Martin Fowler's Books

- See Also [シグニチャシリーズの判断基準](#)



更新日時:2005/04/01 20:17:22

キーワード:

参照:

Generated by [Hiki](#) 0.9dev (2009-08-17).
 Powered by [Ruby](#) 1.8.7-p371 (2012-10-12).
 Founded by [kdmsnr](#).

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

最近、移動しました。

<http://bliki-ja.github.io/>



見てね！



Conversational (対話)



"Not Just Code Monkeys"



Martin Fowler
ThoughtWorks



<https://www.youtube.com/watch?v=Z8aECe4lp44>

フラストレーション



<https://www.youtube.com/watch?v=Z8aECe4lp44>





チーム志向越境型 アントレプレナー育成プログラム

English

Google™ カスタム検索



ホーム



概要



ビジョン



組織



イベント



共同研究



アクセス



本プログラムでは、様々なステークホルダーとの間の自律的な協力関係を保ちながら、専門の違い、文化の違い、性別の違いなどの境界を乗り越え、多様な価値観を許容し、互いに協力しながらチームとして活動することにより、イノベーションを起こすことのできる人材を育成します。

> • ||

<http://www.eng.titech.ac.jp/~cbe/>

第3回
EDGEプログラム シンポジウム

イノベーションを



2015年9月18日(金)

10:00~17:50 (交流会18:00~20:00)

早稲田大学 早稲田キャンパス 井深大記念ホール

申込方法: 早稲田大学EDGEプログラム
ホームページより事前登録

参加費: 無料 (ただし、交流会は参加費
一般2,000円、学生1,000円)

主催: 早稲田大学 WASEDA-EDGE
人材育成プログラム(文部科学省補助事業)

協賛: 早稲田大学 リーディング理工学博士プログラム/
実体情報学博士プログラム/革新的イノベーション
創出プログラム(COI STREAM)/ナノ・ライフ創
新研究機構/ナノテクノロジーフォーラム

何度も協力して知識を増やす

第 15 章

XP のスケーリング

XP がどのようにスケールするかをよく質問される。100 人が週一の会議で仕事を詳細に計画することはできない。だが、100 人でもコミュニケーション、フィードバック、シンプリシティ、勇気、リスペクトの精神で一緒に働くことはできる。12 人のコミュニティよりも、100 人のコミュニティを作ったり維持したりするほうがはるかに難しいが、それでも常に行われていることだ。

ソフトウェア開発においては、プロジェクトの人数だけがスケールを決める尺度ではない。ソフトウェア開発は以下のさまざまな要素でスケールする。

問題の複雑さ

XP は、専門家同士の密な連携が必要とされるプロジェクトに最適だ。そのようなプロジェクトで最初に課題となるのは、お互いの専門分野を少しずつ学びながら、全員を協力させることである。たとえば、私は生命保険のプロジェクトに参加したことがある。保険数学の学習を始めたときには、アクチュアリー（保険数理士）が辛抱強くペアになってくれた。1 か月後、私はわずかな間違いを発見できるようになった。数か月後、役に立てることが増えてきた。その後、私はアクチュアリーになったわけではない。それでも、私とアクチュアリーが別々に、それぞれユーザーインターフェイスとシステムの特定の機能に取り組むよりも、はるかに強力なシステム（とチーム）ができたと思う。

新しいソフトウェア開発手法

(original: [The New Methodology](#))

[マーチン・フォウラー](#)

チーフサイエンティスト, [ThoughtWorks](#)

過去数年にわたり、「ライトな」ソフトウェア開発手法が急速に関心を集めつつある。それらは、官僚制に対する解毒剤とも、ハッキングのライセンスとも見なされているが、ソフトウェア関係者全ての興味をかきたてている。このエッセイで、私は「ライトな」開発手法の単に「軽い」側面だけでなく適応的な性質や人間中心主義に着目しながら、それらが流行る理由について掘り下げてみたい。また、この系統のプロセスに対してサマリーとリファレンスを提供し、この踏み出されてまもない道に行くべきかどうかを選択するために、考慮すべき要因について考えてみたい。

http://www007.upp.so-net.ne.jp/kengai/fowler/newMethodology_j.html

まとめ職人すごい

goto;
conference



Agile

Adaptive Planning

Plan-Driven

Predictive Planning

<http://martinfowler.com/articles/newMethodology.html>

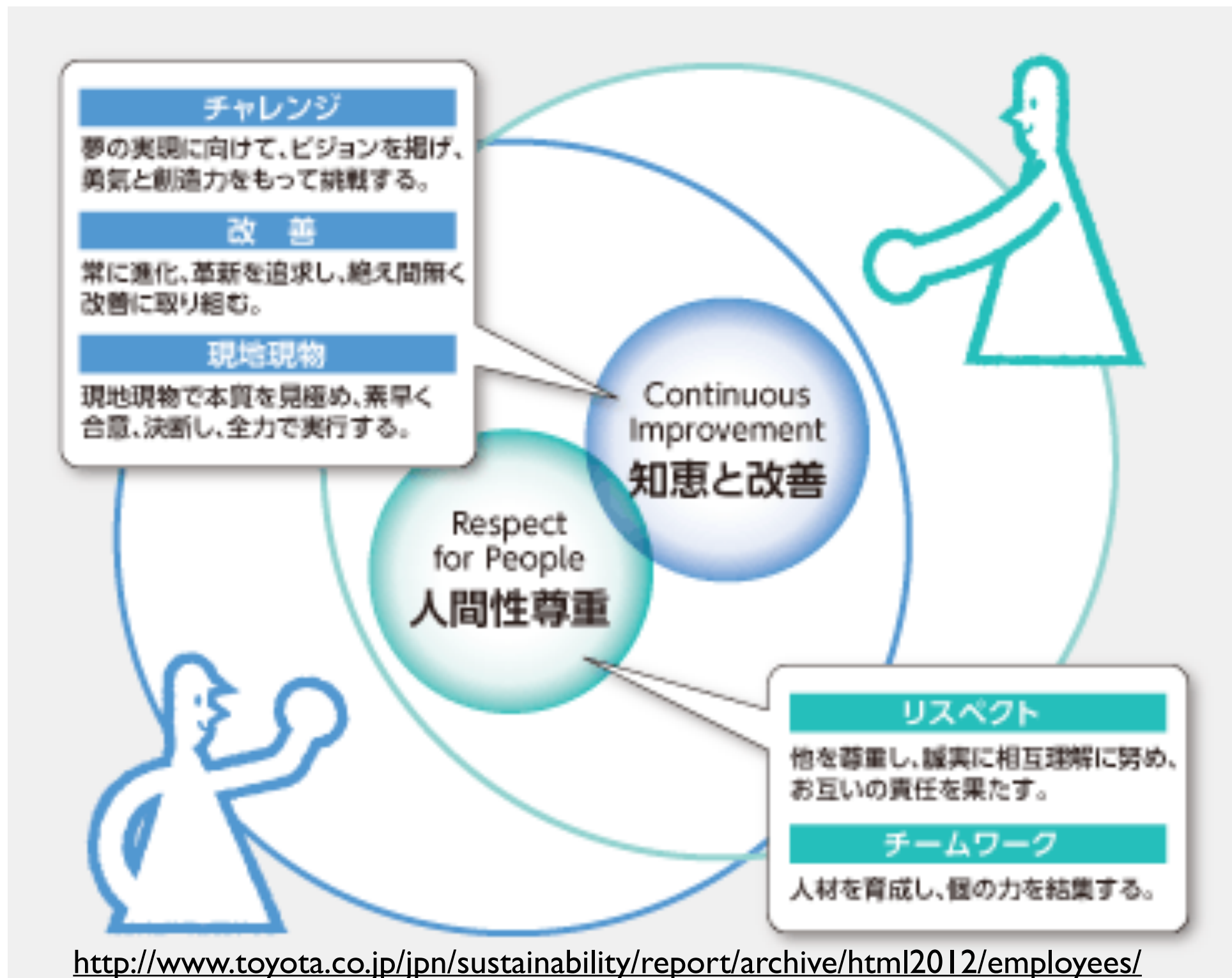
People-first

Process-first

goto;

<https://www.youtube.com/watch?v=TgdFA72crHM>

トヨタウェイと（だいたい）同じ



30分経過

1. 人間性を重視したときめき💖



https://en.wikipedia.org/wiki/Frederick_Winslow_Taylor


ワイクル株式会社

第18章

テイラー主義とソフトウェア

フレドリック・テイラーは、世界初の生産技術者（*industrial engineer*）である。それ以前から工場の効率性を研究する人たちはいたが、テイラーは情熱とカリスマ性を携えてこの分野に乗り込み、インダストリアルエンジニアリングという新しい分野を生み出した。彼と弟子たち（特にフランク・ギルブレス、リリアン・ギルブレス、ヘンリー・ガント）の成果により、工場の生産性を体系的に向上させる、厳密で説得力のある論拠が示されたのである。彼は、自分の学説に「科学的管理法」という強力なメタファーを選んだ。

科学的管理法



https://en.wikipedia.org/wiki/The_Thinker



[https://en.wikipedia.org/wiki/Modern_Times_\(film\)](https://en.wikipedia.org/wiki/Modern_Times_(film))



<http://www.sherlockology.com/props/sherlocks-magnifier>

我々はモノではない



Mad Max: Fury Road

モノの証明（認定）



giflike.com

モノの証明（認定）

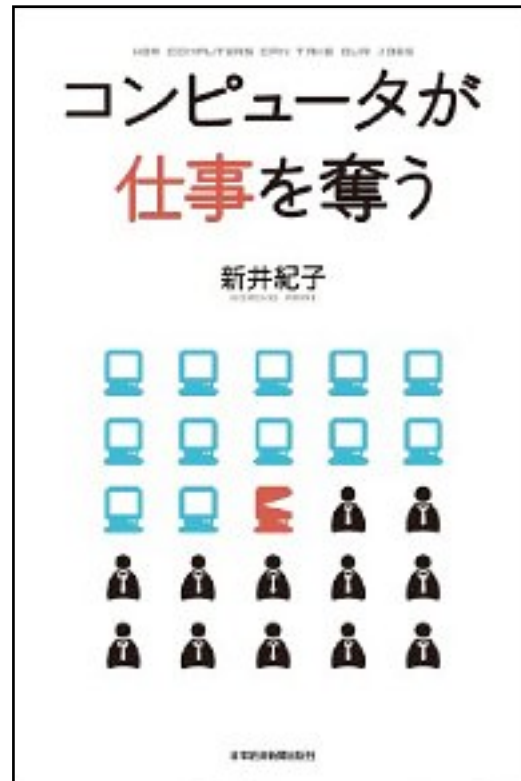
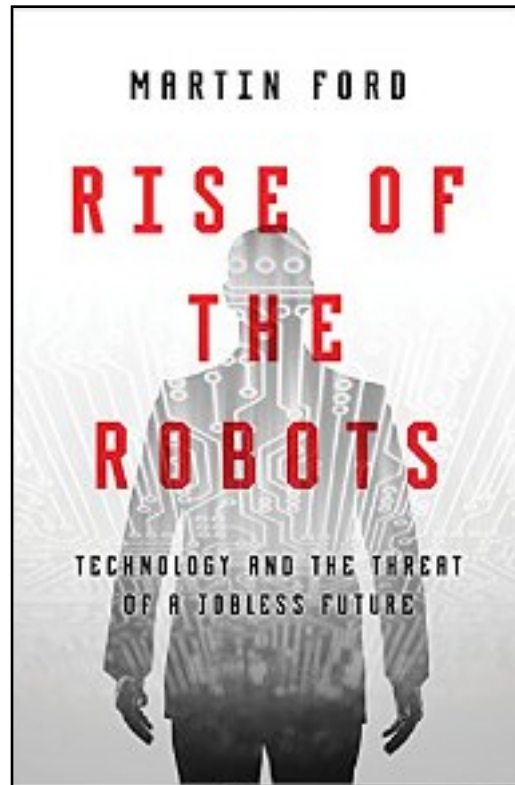
第21章

認証と認定

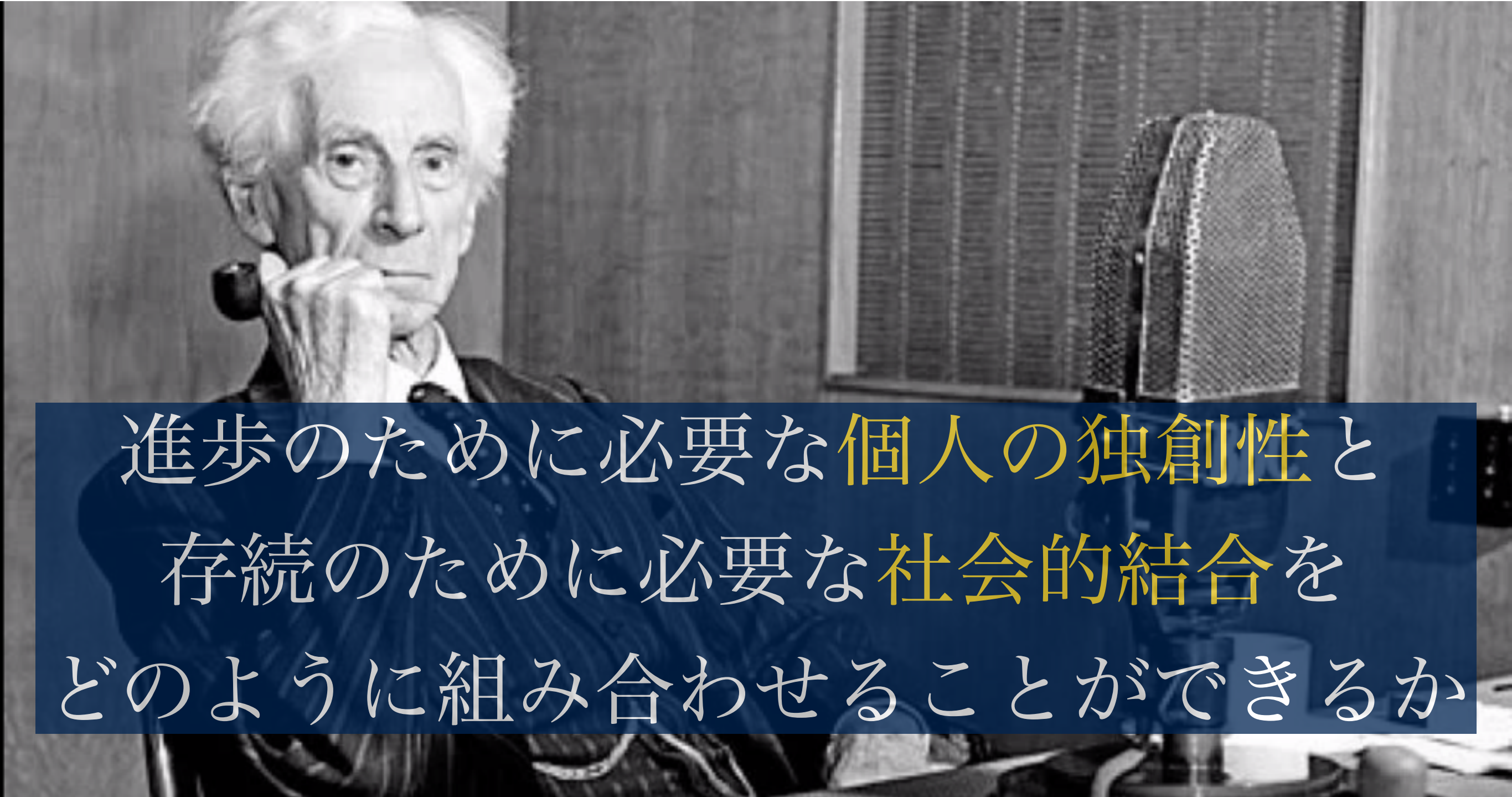
同じく一般的なテーマの話題として、個人またはチームの認証（*certification*）の話がある。認証プロセスにおける認証機関は、自らの信用をかけて認証した人の適正を保証し、その人に対する一定の責任を受け入れている。たとえば、専門医資格を持った医者が不適正だとわかれば、その認証機関の信頼性は失われてしまう。

コンピューティングの分野には、そこまで踏み込んだ認証は存在しない。他人の開発の選択について、法的な責任を負いたいと思う人がいるだろうか？ 認証機関が自分たちの認証に責任を持たないのであれば、それは単なる証明書の発行と金儲けをしているだけだ。

モノではない何か



バートランド・ラッセル



進歩のために必要な**個人の独創性**と
存続のために必要な**社会的結合**を
どのように組み合わせることができるか

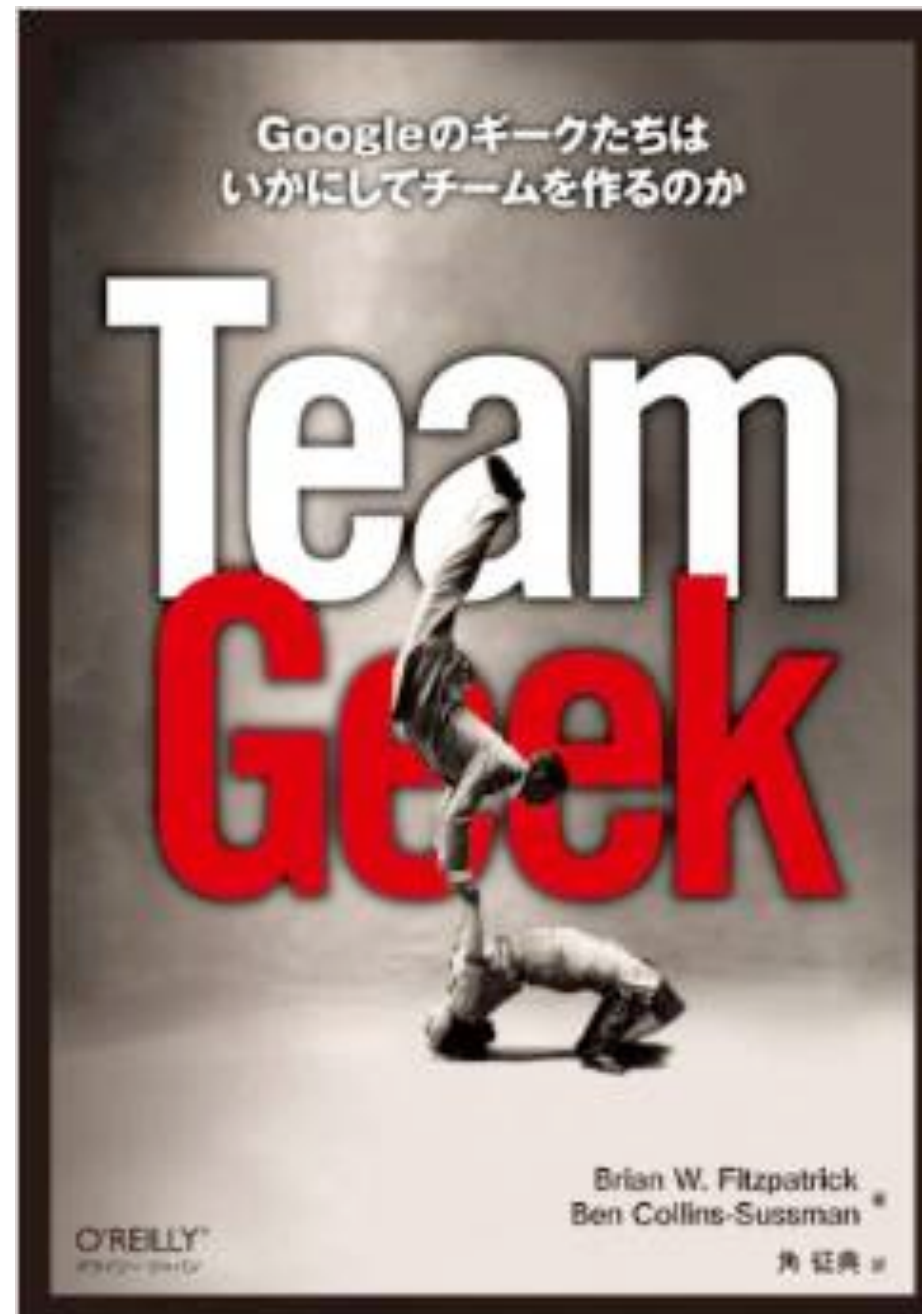
<https://www.youtube.com/watch?v=9EF4I7HM0zI>

XP とは何か

エクストリームプログラミング (XP) はソーシャルチェンジである。XP とは、~~以前はうまくいっていたかもしれないが、今では最高の仕事の邪魔になっている習慣やパターンを手放すことだ。~~XP とは、これまで自分たちを守ってきくれたが、今では生産性の妨げになっているものを捨て去ることだ。何だか自分がさらけ出されたような気持ちになるかもしれない。

とはいえ、誰とでも仲良くなれるはずがない👾

新しいソーシャルチェンジ



三本柱 (HRT)

▶ 謙虚 (Humility)

- 自分が間違ってるかもしれないと考えよう

▶ 尊敬 (Respect)

- 一緒に働く人のことを大切に扱おう

▶ 信頼 (Trust)

- 自分以外に仕事を任せよう

QiitaやKobitoを作る開発チームの文化

こんにちは,  yaottiです.

今日はQiitaやQiita:Team, Kobitoを開発するチームでぼくたちがどういう文化, 価値観を大切にしているかをお話したいと思います.



「驕り高ぶらないようにしよう」「相手を尊敬しよう」「人を信頼してまかせよう」といったHRTの概念は以前よりチーム内で無意識に尊重されていましたが, 「HRTを大事にしよう」と明文化してわかったのは「当たり前だと感じる!=十分実践できている」ということでした.

<http://blog.qiita.com/post/74997115585/increments-dev-team-culture>

KAIZEN エンジンア行動指針

行動哲学

- HRT
- 自律的に動く
- 暗黙知を作らない
- 許可を求めるな謝罪せよ
- 官僚的にならない
 - × 「〇〇さんの許可を得ないとできません」
- 技術的な間違いは躊躇せず指摘
- 3度同じ事を繰り返す時は自動化



<https://speakerdeck.com/mdoi/kai-fa-suruyouniyun-yong-suruinhura-jaws-days-2015>

まず「HRTの文化」を作る

ここは好き嫌い関わらず、最低限守ってもらう

三本柱 (HRT)

▶ 謙虚 (Humility)

- 自分が間違ってるかもしれないと考えよう

▶ 尊敬 (Respect)

XPの価値の1つ

- 一緒に働く人のことを大切に扱おう

▶ 信頼 (Trust)

- 自分以外に仕事を任せよう

ペアプログラミング



http://en.wikipedia.org/wiki/Pair_programming

ペアプログラミング (の演習)

- ▶ チームの最小単位は「ペア」
- ▶ 「楽しい」を最初に経験する
 - 普段から実際にやれというわけではない
- ▶ フィードバック、コミュニケーション

▶ 実施例：

XPの価値の2つ

- クックパッド社新卒研修 (2015) 他数社
- 公開研修もやってます：

<http://waicrew.doorkeeper.jp>

40分経過

シンプルシティ

XPの価値の1つ



<http://time.com/3822899/marie-kondo-2015-time-100/>

こんまり流片付け術

- ▶ 1. モノを捨てること
- ▶ 2. 収納場所を決めること

こんまり流片付け術

▶ 1. モノを捨てること

- 「捨てるモノ」ではなく「残すモノ」を選ぶ

▶ 2. 収納場所を決めること

こんまり先生いわく

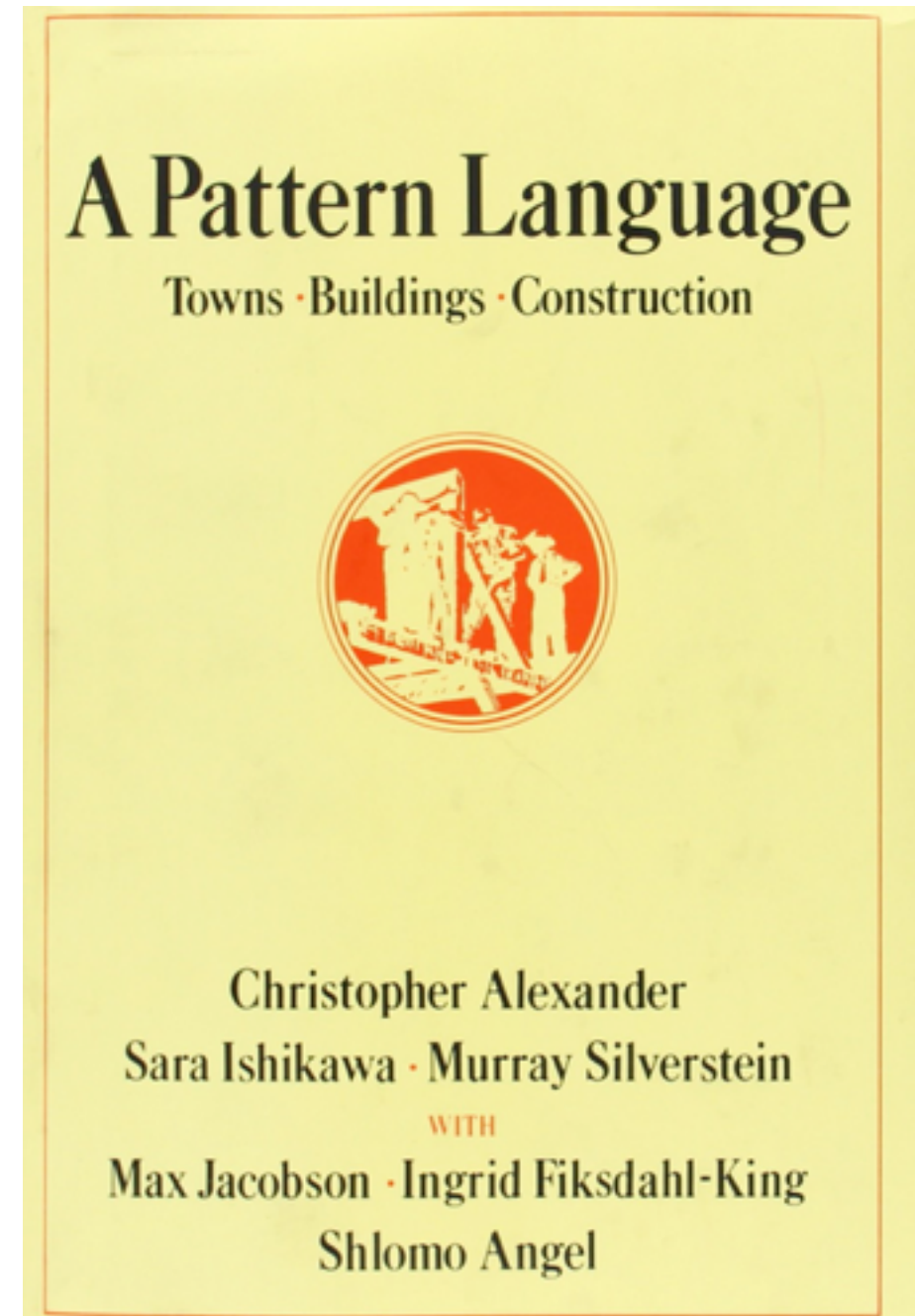
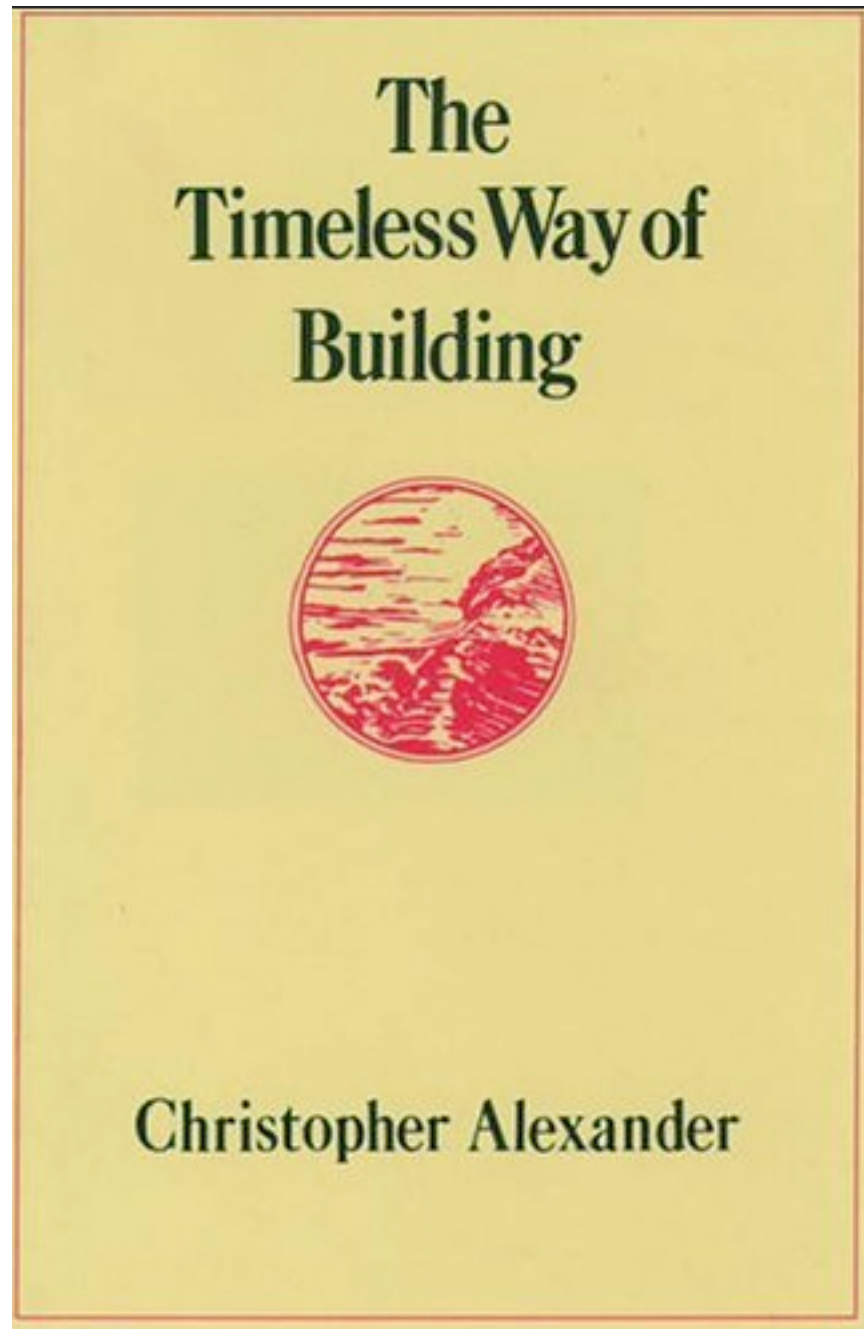
“

Does it spark joy?

(それはときめく💖か?)

”

なんか聞いたことがある！



第23章

時を超えたプログラミングの道 ⇒ ときめき💖の道

そう遠くない昔、人々は自分たちのニーズ、気候、文化に適した空間の設計と構築の方法を知っていた。建築家のクリストファー・アレグザンダーは、そのように著している。子どもの頃、大工だった曾祖父の話を聞いたことがある。家族で新しい町に引っ越すと、すぐに臨時仕事を始めて、お金を節約したそう。そして、お金がたまったら、材木を購入して、家を造った。建築家としての訓練を受けたわけでもないのに、家族に適した家の設計方法を知っていたのである。

プラクティス == パターン

エクストリームプログラミングは、ケント・ベックがクリストファー・アレグザンダーのパターンランゲージをソフトウェア開発に応用しようとした取り組みである。アジャイルソフトウェア開発の出発点として重要であるだけでなく、逆にここでの取り組みを建築や空間設計に活かすことができるだろう。この本には心地良い空間を設計するためのヒントが溢れている。ぜひソフトウェア開発に限らず、空間を設計する人にも届いてほしい。

▶ 江渡浩一郎

メディアアーティスト、国立研究開発法人産業技術総合研究所主任研究員、ニコニコ学会β実行委員長

プラクティス == パターン

初版は世田谷の坂道を歩きながら読んだ覚えがある。見上げるとカラスが急降下してきた。この本はアジャイルの基本書というだけでなく、単独のパターンではなく体系としてのパターンランゲージの記述を建築以外の分野で初めて試み成功した貴重な実践例である。そして第2版でパターンランゲージは実践していく中で経験を取り込み書き換えられ進化していくということが学べる。変化を受け容れるということの自己適用が実践されている貴重な書。これを次にはもう一度、建築・まちづくりに持ち込み直すのが楽しみ。

▶ 羽生田栄一

株式会社豆蔵、UBrainTV

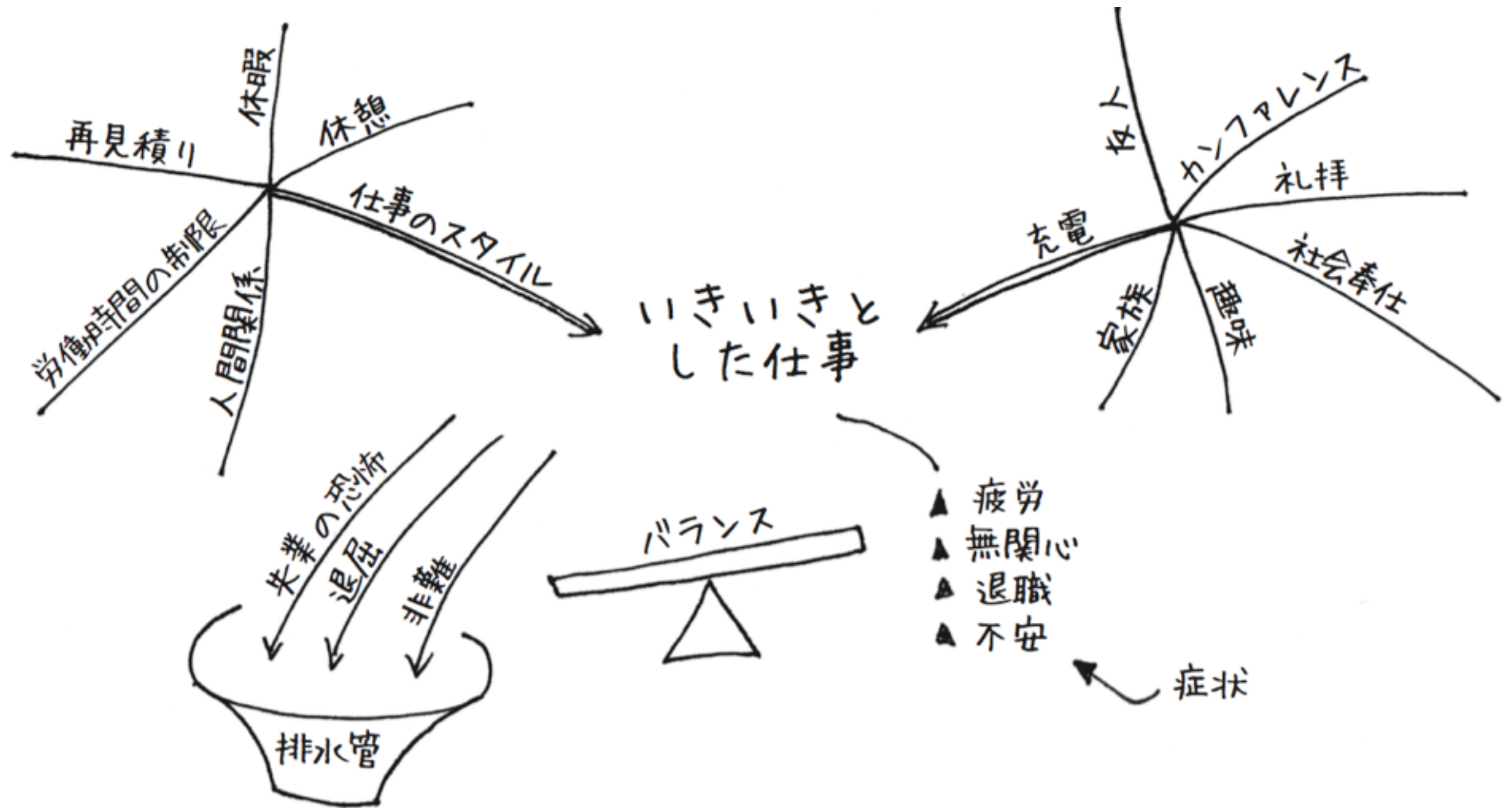
プラクティス == パターン

著者はXPの5年間の実践を経て、顧客やビジネスと寄り添ったソフトウェア開発の価値・原則・プラクティスを、より洗練された「言葉」として整理し、それを組み合わせることで成功確率の高い仕事の進め方を記している。本書は、その言葉たちを大切に扱った新訳であり、実に読みやすい。しばしばアジャイル開発が誇大に評価されるような今日において、読者は本書の言葉に触れることでその本質と限界を理解し、言葉を用いて自らのよりよい仕事の進め方を考え、語り、形作ることができる。

▶ 鷺崎弘宜

早稲田大学グローバルソフトウェアエンジニアリング研究所所長

ときめく💖プラクティスを選ぶ



▶ 図 8 「いっきいきとした仕事」のマップ

第21章

エクストリームの純度

もはやどうでもいい！

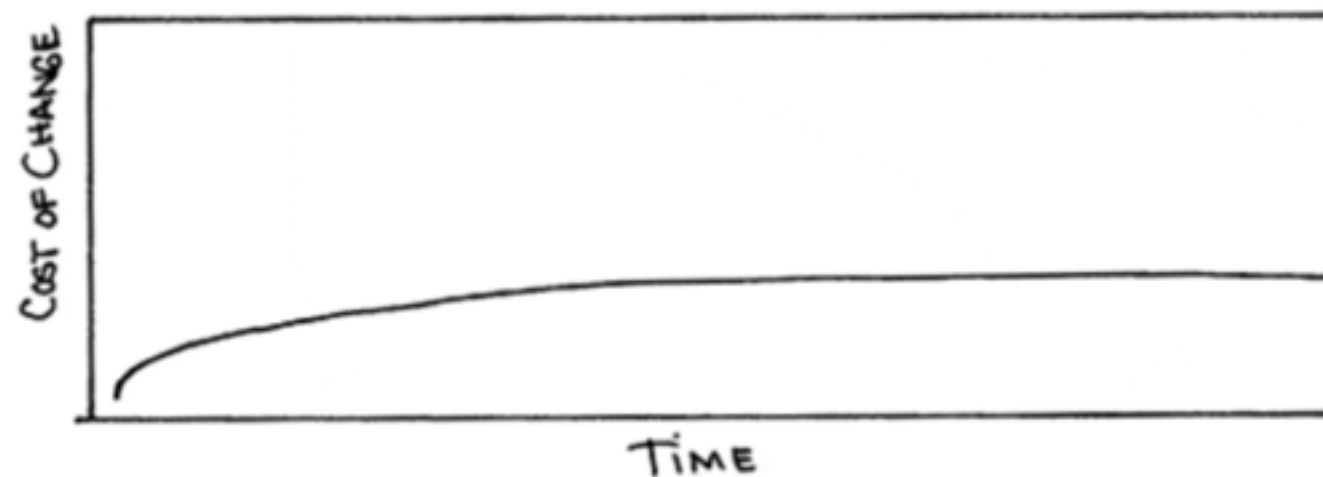
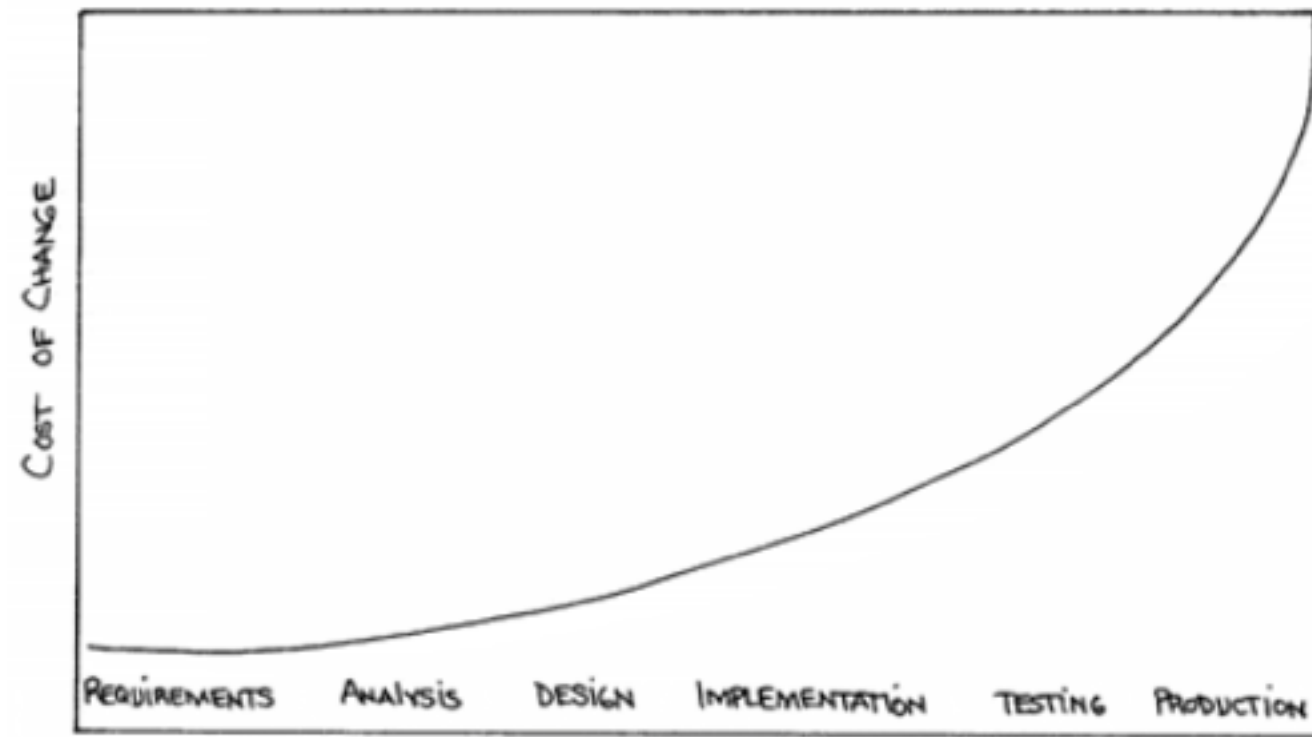
「私のチームはエクストリームですか？」と何度も聞かれる。グラフや指標を組み合わせて「エクストリーム」を計測しようとする人たちがいる。ふりかえりのツールとして使うのであれば納得できる。だが、10点が5点の2倍優れているというような計測方法はバカげている。「私のチームはエクストリームですか？」の質問に対して、バイナリな（客観的な二者択一の）答えや数値的な答えを期待するのはおかしい話だ。

ここまでのまとめ

- ▶ 人間性が大切
- ▶ 人をモノ扱いする認定とかいらない
- ▶ まずはHRT（with 『Team Geek』）
- ▶ ペアプログラミングをやってみよう
- ▶ ときめく💖プラクティスを選択しよう

2. 適応型計画づくりとその技術🌿

XPが目指している（いた？）もの



Kent Beck, "Extreme Programming Explained"

できることを増やすプラクティス (enabling practices)

設計の終焉?

(原題: 🌐Is Design Dead?)

🌐マーチン・ファウラー

チーフサイエンティスト, 🌐ThoughtWorks

エクストリーム・プログラミング(XP)をかじってみて、多くの人がこう感じただろう。XPは「ソフトウェア設計など消え失せる」と言っているのではないかと。というのもXPでは、多くの設計作業が「料金前払いのデカイ設計("Big Up Front Design")」などとからかわれているばかりか、UML、柔軟なフレームワーク、そしてパターンまでもを含む設計技法が、そんな扱いを受けているか、完全に無視されているからだ。実際には、XPにもたくさんの設計作業が含まれている。しかしそれを、既存の設計プロセスとは違うやり方で行っているのだ。「進化的設計」という考え方がある。XPはこの考え方を、「進化」を実行可能な設計戦略へと変換するプラクティス(実践)を用いることで、復興させたのだ。さらにXPは、新たな挑戦課題と新たな技能とを示している。設計者は、「シンプルな設計を行うにはどうしたらよいか」「リファクタリングを行ってきれいな設計を維持するにはどうしたらよいか」そして「進化的な方法でパターンを用いるにはどうしたらよいか」ということを学ばなくてはならないからだ。

http://www.objectclub.jp/community/XP-jp/xp_relate/isdesigndead

できることを増やすプラクティス

▶ ソフトウェアの設計：

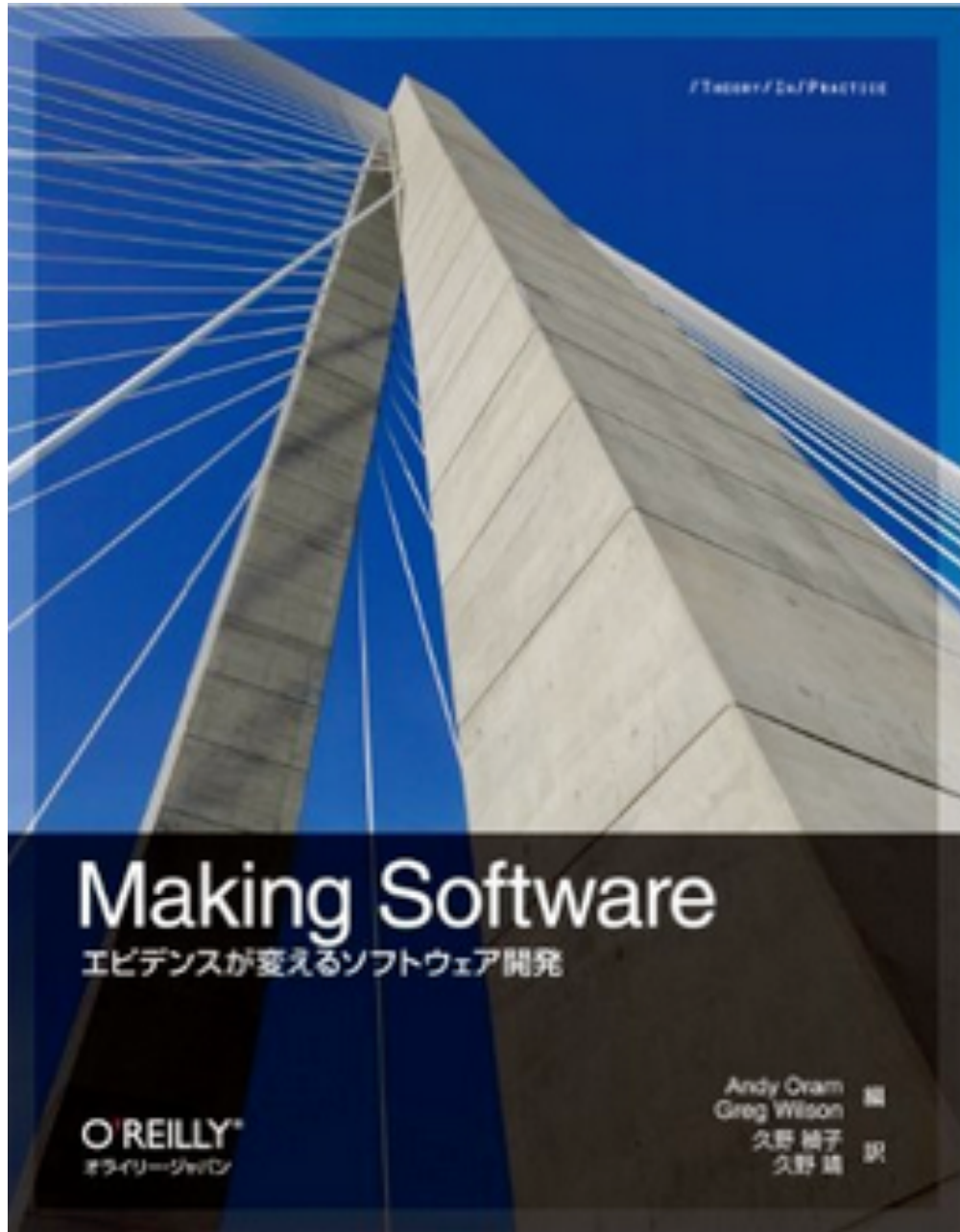
- 1. テスト（自動テスト）
- 2. 継続的インテグレーション
- 3. リファクタリング

▶ 他の分野にもあるはず！

- けど、何かはわからない……。

<http://bliki-ja.github.io/SpreadingIncrementalism/>

バリー・ベームからの反論



10章
アーキテクティング：いつ、どれだけ？
バリー・ベーム (Barry Boehm)

いや、それでもダメ

アーキテクティングは前もって行う必要があるのか？

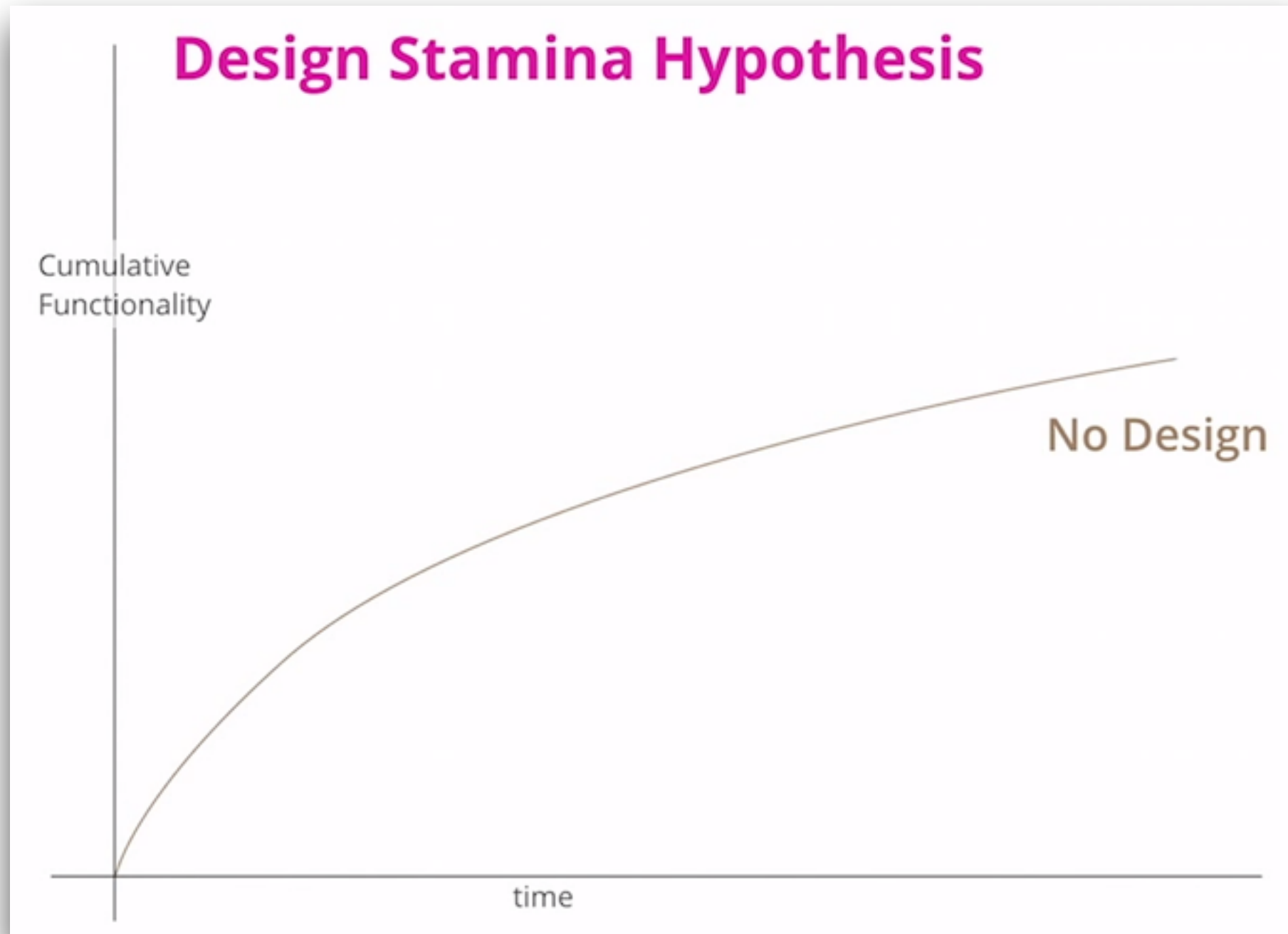
多くのアジャイルプロジェクトは、十分なアーキテクティングを行うことで成功を納めますが、ただしアーキテクティングの活動はリファクタリングという形でライフサイクル全体に渡って広く浅く行います。この方法は、プロジェクトの規模が小さく重大でないものである限りにおいては有効ですが、そうでない場合は「最も簡単なものからやる (easiest-first)」ことが原因で困難に陥る可能性があります。

引用元：『Making Software』（オライリージャパン）

10章「アーキテクティング：いつ、どれだけ？」 by バリー・ベーム

設計=スタミナ仮説

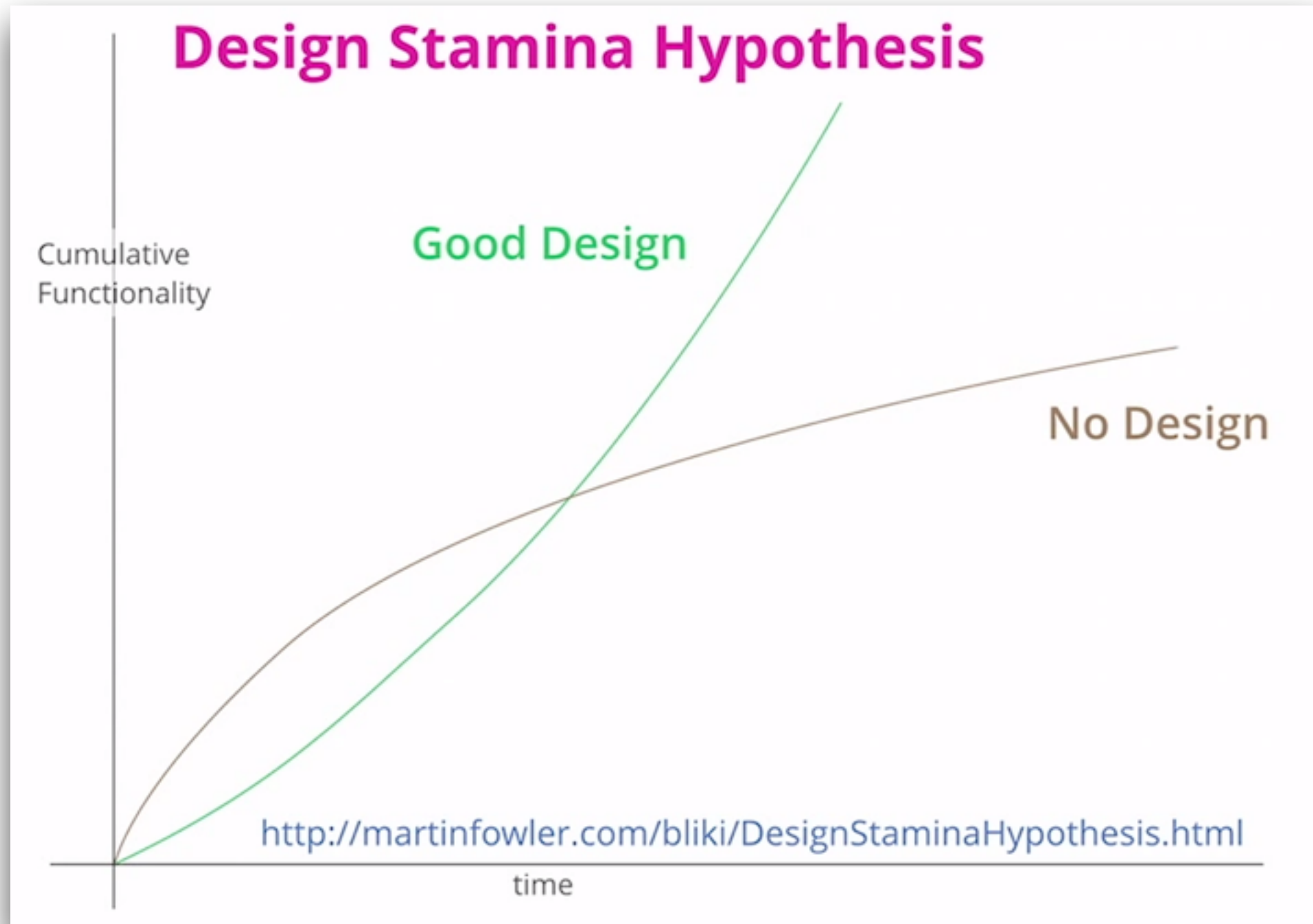
<http://bliki-ja.github.io/DesignStaminaHypothesis/>



<https://www.youtube.com/watch?v=DngAZyWMGR0>

設計=スタミナ仮説

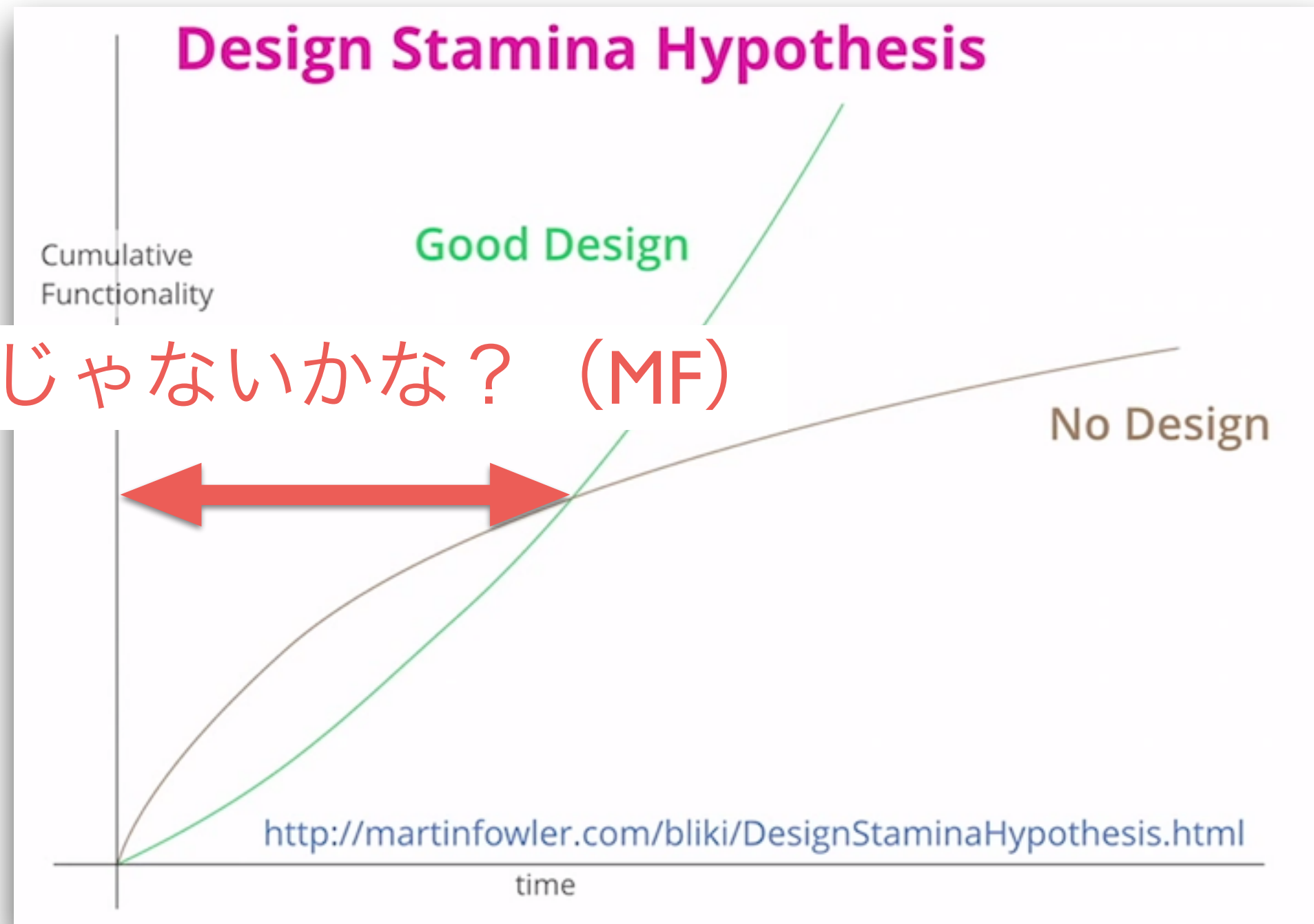
<http://bliki-ja.github.io/DesignStaminaHypothesis/>



<https://www.youtube.com/watch?v=DngAZyWMGR0>

設計=スタミナ仮説

<http://bliki-ja.github.io/DesignStaminaHypothesis/>



<https://www.youtube.com/watch?v=DngAZyWMGR0>

技術的負債の四象限

意図的



島本和彦『燃えよペン』



井上雄彦『SLAM DUNK』

無鉄砲

慎重



高橋陽一『キャプテン翼』



横山光輝『三国志』

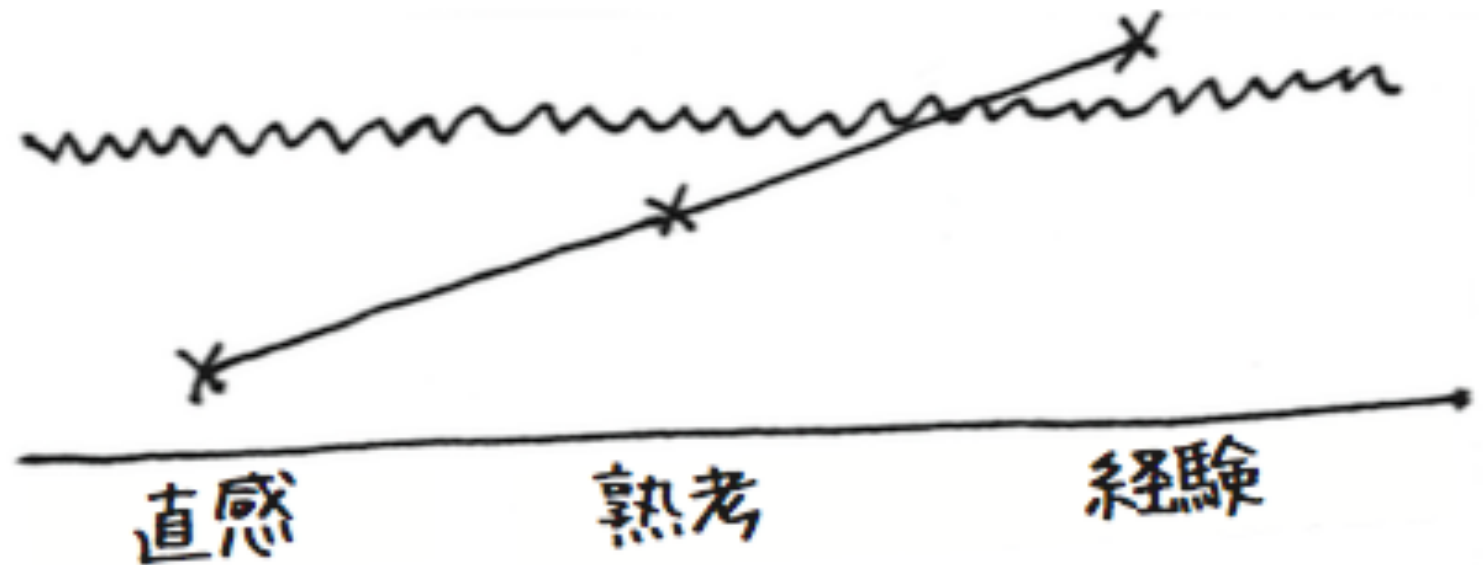
不注意

KB : 「いつやるか？」 「あとで」 (ん？ ということ？)

第 14 章

設計：時間の重要性

インクリメンタルな設計は、機能を早期に届ける方法であり、プロジェクト期間にわたって毎週継続して機能を届ける方法である。XP は設計のインクリメンタル主義を極限まで高めている。設計が日常業務の一環になれば、プロジェクトはもっとスムーズに進むというものだ。本章では、インクリメンタルな設計の技術的、経済的、人間的な理由について深く掘り下げる。



▶ 図 20 熟考するだけでは不十分

「経験」してから設計する

▶最終責任時点

- 意思決定は遅いほうがチャンスは大きい (by ポッペンディーク)
- とはいえ、いつが「最終」なのか不明 🥵

▶セットベース開発

- 複数の案を実装してみる
- → そこから選ぶ 😊
- 「熟考」だけよりいいものができる (可能性が高い)

▶可逆性を確保する

- できるだけあとでやり直せるようにする (→次ページ)

可逆性を確保せよ



Taming Complexity with Reversibility

 KENT BECK · TUESDAY, JULY 28, 2015

<https://www.facebook.com/notes/kent-beck/taming-complexity-with-reversibility/1000330413333156>

できることを増やすプラクティス

(あとで読んでね)

- ▶ Development servers
- ▶ Code review
- ▶ Internal usage
- ▶ Staged rollout
- ▶ Dynamic Configuration
- ▶ Correlation
- ▶ IRC
- ▶ Right hand side units
- ▶ Shadow production
- ▶ Frequent pushes
- ▶ Data-informed decision
- ▶ Advance countries
- ▶ Soft launches
- ▶ Double write/bulk migrate/double read



<https://twitter.com/martincronje/status/636170140166561793>

Facebookの事例

Development and Deployment at Facebook

Dror G. Feitelson
Hebrew University

Eitan Frachtenberg
Facebook

Kent L. Beck
Facebook

たぶんKB

Abstract

More than one billion users log in to Facebook at least once a month to connect and share content with each other. Among other activities, these users upload over 2.5 billion content items every day. In this article we describe the development and deployment of the software that supports all this activity, focusing on the site's primary codebase for the Web front-end. Information on Facebook's architecture and other software components is available elsewhere.

D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and deployment at facebook,"
IEEE Internet Comput., vol. 17, no. 4, pp. 8–17, 2013.

Facebookの事例

▶ 永久に開発（継続的デプロイ）

- 実際のデプロイは週1回。内部リリースは1日2回。

▶ 「A/Bテスト」で実験

- 「経験」してから選択する
- 「まだ要求開発や仕様書で消耗してるの？」

▶ 6週間の新人研修

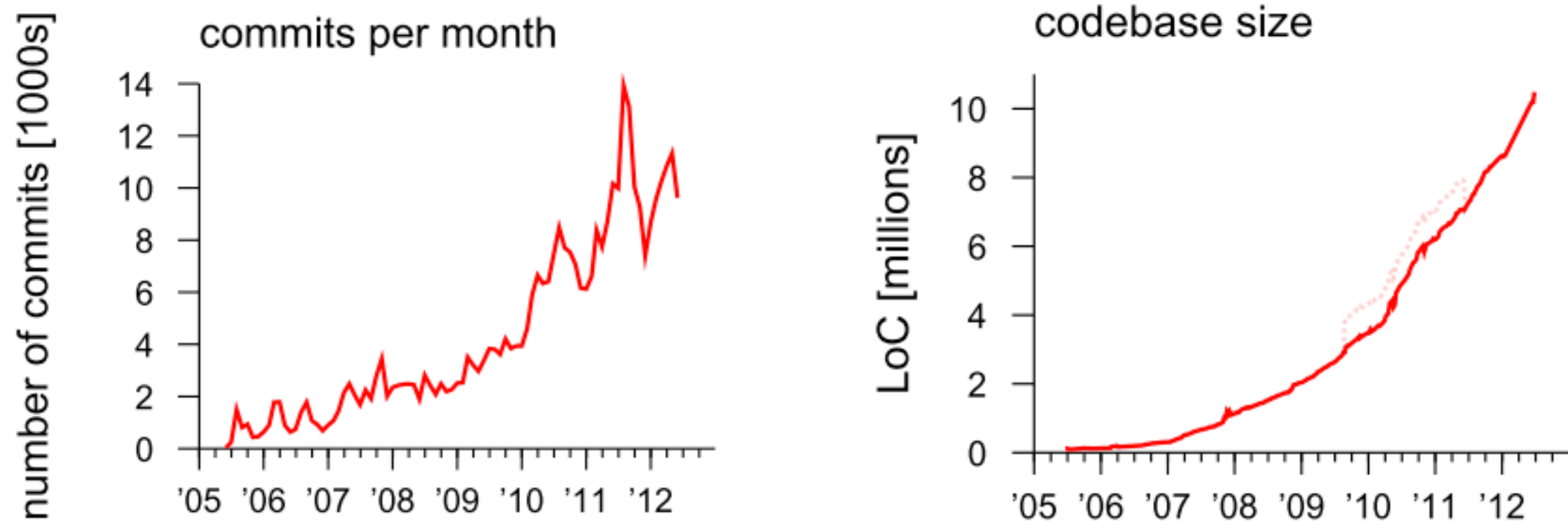
- 書いたコードをすぐにリリースする訓練

▶ エンジニアには責任と覚悟が欠かせない

D. G. Feitelson, E. Frachtenberg, and K. L. Beck, “Development and deployment at facebook,”
IEEE Internet Comput., vol. 17, no. 4, pp. 8–17, 2013.

Facebookの事例

ますます活性化



D. G. Feitelson, E. Frachtenberg, and K. L. Beck, “Development and deployment at facebook,”
IEEE Internet Comput., vol. 17, no. 4, pp. 8–17, 2013.

その他の事例（2つ）



Railsの事例

12章

アトリビュートメソッドの進化

ここまで読んできたなかで、メタプログラミングのコードや例にたくさん触れてきた。だが、現実の大規模なシステムでメタプログラミングを使ったときに何が起きるのかと、今でも疑問に思っていることだろう。すぐにコードが複雑になり、予期しない方向に進んでいくこの乱雑な世界で、このような洗練された技法をどのように使えばいいのだろうか？

Paolo Perrotta 『メタプログラミングRuby 第2版』 (オライリー・ジャパン)

Railsの事例

Rails のやり方は「最初から正しくやる」よりも「進化的設計」に傾いている。その理由は大きく2つある。まずは、Ruby が柔軟性の高い言語だからだ。メタプログラミングを使うときには特にそうだ。したがって、状況に合わせてコードを進化させていくことが簡単にできるのである。もうひとつの理由は、最初から完璧なメタプログラミングのコードを書くことは難しいからだ。すべてのコーナーケースを明らかにすることは困難である。

以上を簡単にまとめると、以下のようになる。「コードはできるだけシンプルに保とう。必要になったら複雑にしていこう」。最初は一般的なケースを正しく扱うコードを書こう。あとから特殊ケースを追加できるように、シンプルにしておくのである。これはほとんどのコードに当てはまるルールだが、メタプログラミングを使うコードには特に当てはまる。

Paolo Perrotta 『メタプログラミングRuby 第2版』 (オライリー・ジャパン)

[English] [日本語]

Re:VIEW

easy-to-use digital publishing system for
books and ebooks



View project on
GitHub

// Publish Your (e)Books with Re:VIEW

Re:VIEW is a powerful and easy to use digital publishing system for books and ebooks.

// Usage

```
$ gem install review
$ review-init my-awesome-book
$ cd my-awesome-book
$ (... edit files ...)
$ rake epub    ## generating EPUB
$ rake pdf     ## generating PDF with LaTeX
```



Download
.zip file



Download
.tar.gz file

is maintained by @kmuto.

This page was generated by GitHub
Pages using the Architect theme by
Jason Long.

<http://reviewml.org/>

まあまあ長い開発期間



<http://www.slideshare.net/kenshimuto/review-36769754>

Re:VIEWの事例

- ▶ 普通のFLOSS開発（自動テスト + TravisCI）
- ▶ 課題駆動開発（リアルに困ったものを実装する） ※熟考 + 経験
 - ・自分の仕事（組版や出版事業）に使っているので切実
 - ・「あったら誰かが使いそう」みたいな機能はあんまり入らない
- ▶ 書籍データが別にあるので後方互換性が超大切
 - ・安易に新機能入れたら対応が大変😓（増刷のつらみ……）
- ▶ コミケ駆動開発（2、6、10月リリース）が特徴的
- ▶ 開発者会議を不定期に実施
 - ・チケットの落ち穂拾いと利用者（≡開発者）のヒアリング
- ▶ ムトゥ神（@kmuto）のスライドおすすめ <http://www.slideshare.net/kenshimuto/>

Re:VIEWの開発

「自分のため」
に使いやすいもの
であり続けること

<http://www.slideshare.net/kenshimuto/write-once-publish-anywhere>

できるだけ理解しやすいコードを書く

祝：5万部突破😊



ここまでのまとめ

- ▶ 適応型プランニング（変化を受け入れるには？）
- ▶ できることを増やすプラクティス
 - テスト、CI、リファクタリング
- ▶ とはいえ、設計重要（⇒ 技術的負債）
 - 「設計」の発想を転換 → 「経験」してから設計する
 - 自分たちの選択したプログラミング言語が設計を教えてくれる
- ▶ 熱心に使う/作る人がいれば続けられる
 - リーダブルコードも読んでね

55分経過

たぶん超える……（予言）

3. もっとエキストリーム🚀

はじめに

作業のなかで、私は自分の想像力の限界を思い知らされた。第1版が出版された5年前に「エクストリーム」だと思っていたプラクティスが、今ではありふれたものになっていたからだ。本書に登場するプラクティスも、おそらく5年後には控えめなものになっているだろう。

自分たちで
プラクティス（パターン）
を作るしかない!!

プロジェクト

⇒プロダクト

▶ もはや（実質的に）終わりはない

- プロジェクト：有期性の業務（PMBOK）
- 短期決戦の戦略思考（戸部、野中他『失敗の本質』）

▶ すくなくとも両者を区別しよう

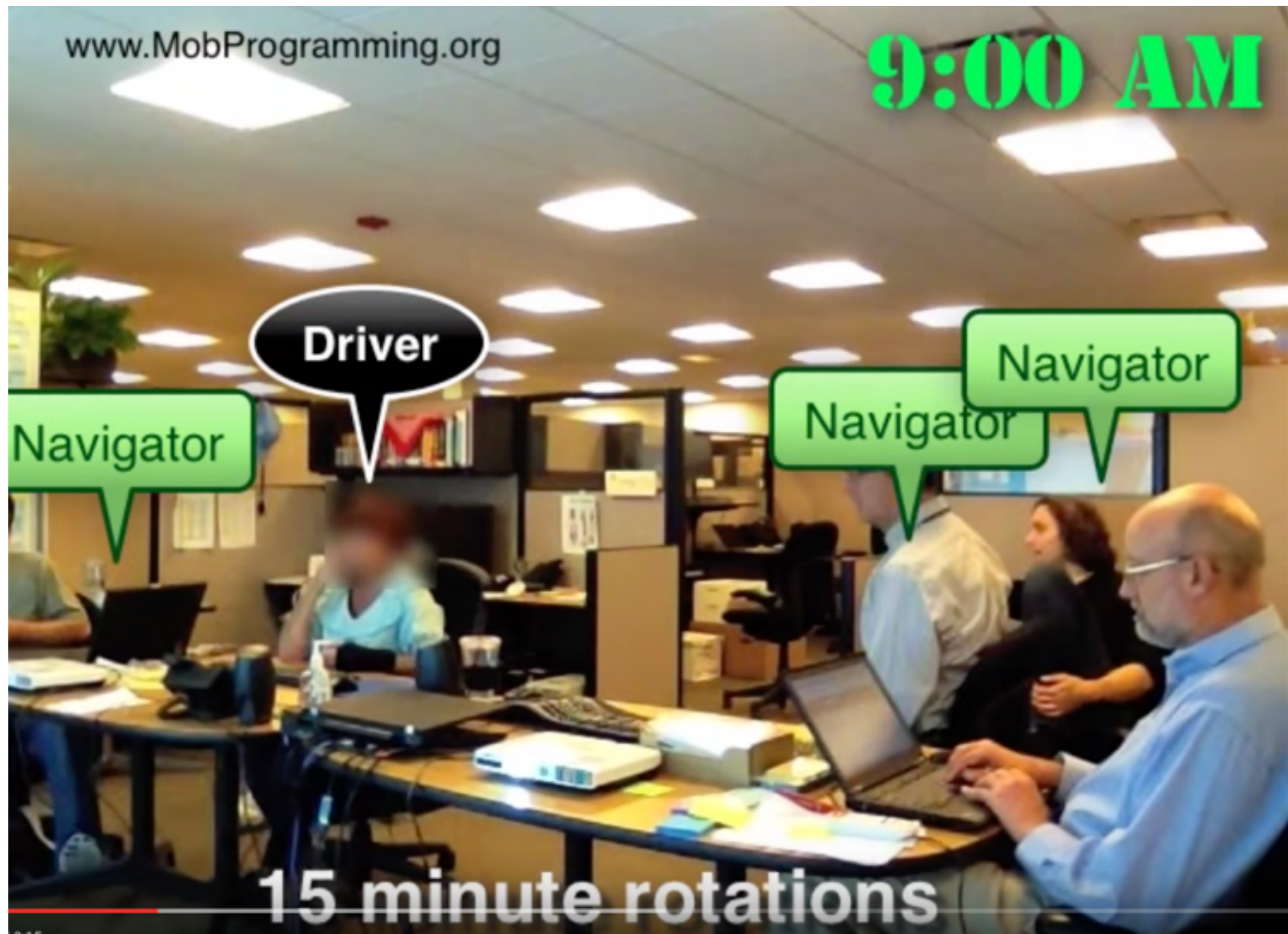
- Project Manager / Product Manager

▶ チームにプロダクトをつけよう

- プラクティス「チームの継続」「責任の引き受け」
- Uncle Bob Martin『Clean Coder』も参照（宣伝……）

ペアプログラミング

⇒ モブプログラミング



https://www.youtube.com/watch?v=p_pvslS4gEI

Mob Programming and the Transition to XP

Moses M. Hohman, Ph.D.

ThoughtWorks, Inc.

651 W Washington Blvd

Suite 500

Chicago, IL 60661 USA

+1 312 373 1000

mmhohman@thoughtworks.com

Andrew C. Slocum, M.S.

ThoughtWorks, Inc.

651 W Washington Blvd

Suite 500

Chicago, IL 60661 USA

+1 312 373 1000

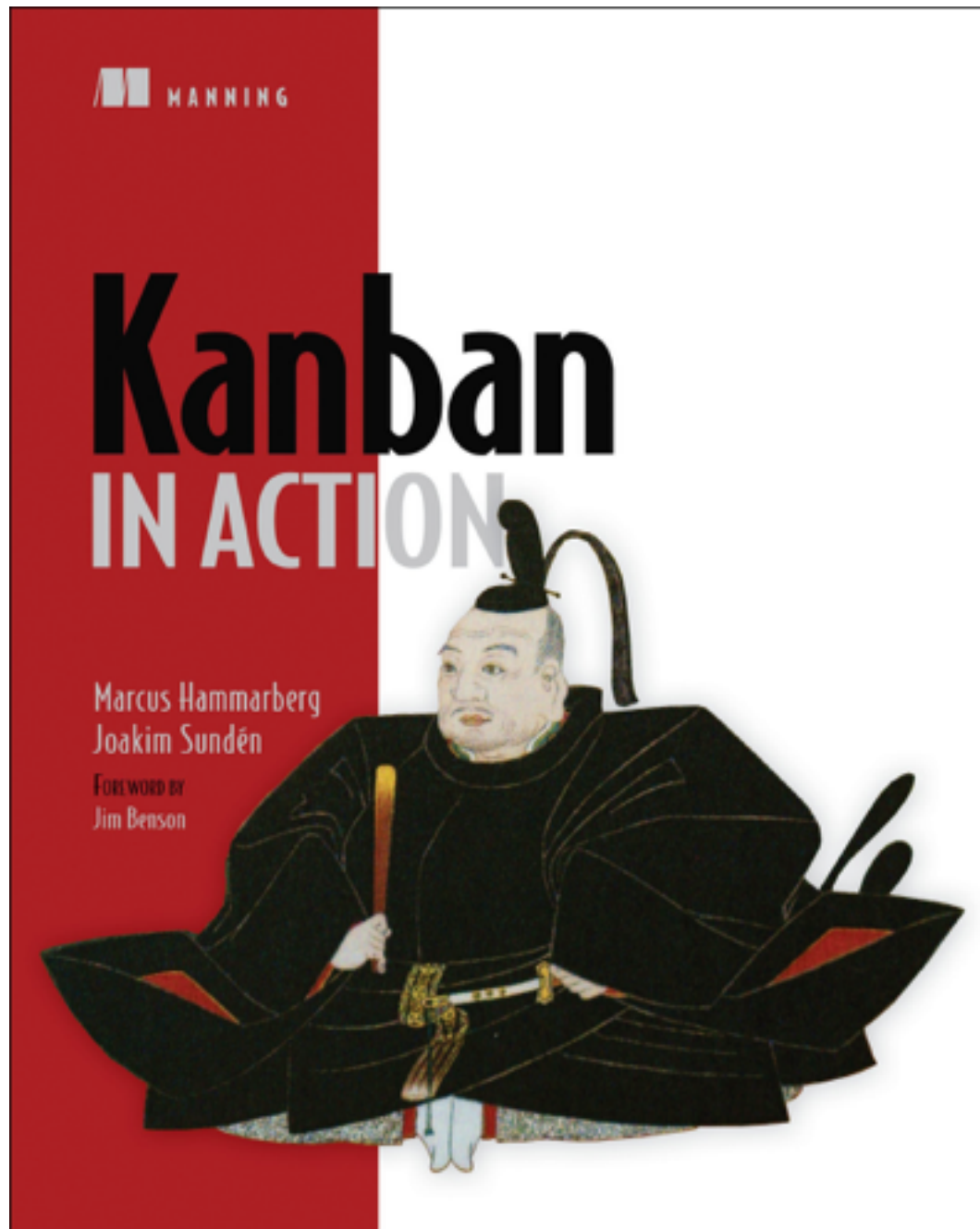
acslocum@thoughtworks.com

Hohman, Moses M., and Andrew C. Slocum. "Mob Programming and the Transition to XP."
proc. First XP Universe Conference. 2001.

こんなときに使おう

- ▶ プロジェクトの初期
- ▶ 難しい or 新しい技術を学ぶとき
- ▶ 新しいメンバーが入ったとき
- ▶ トラブル発生時
- ▶ などなど

懸念点

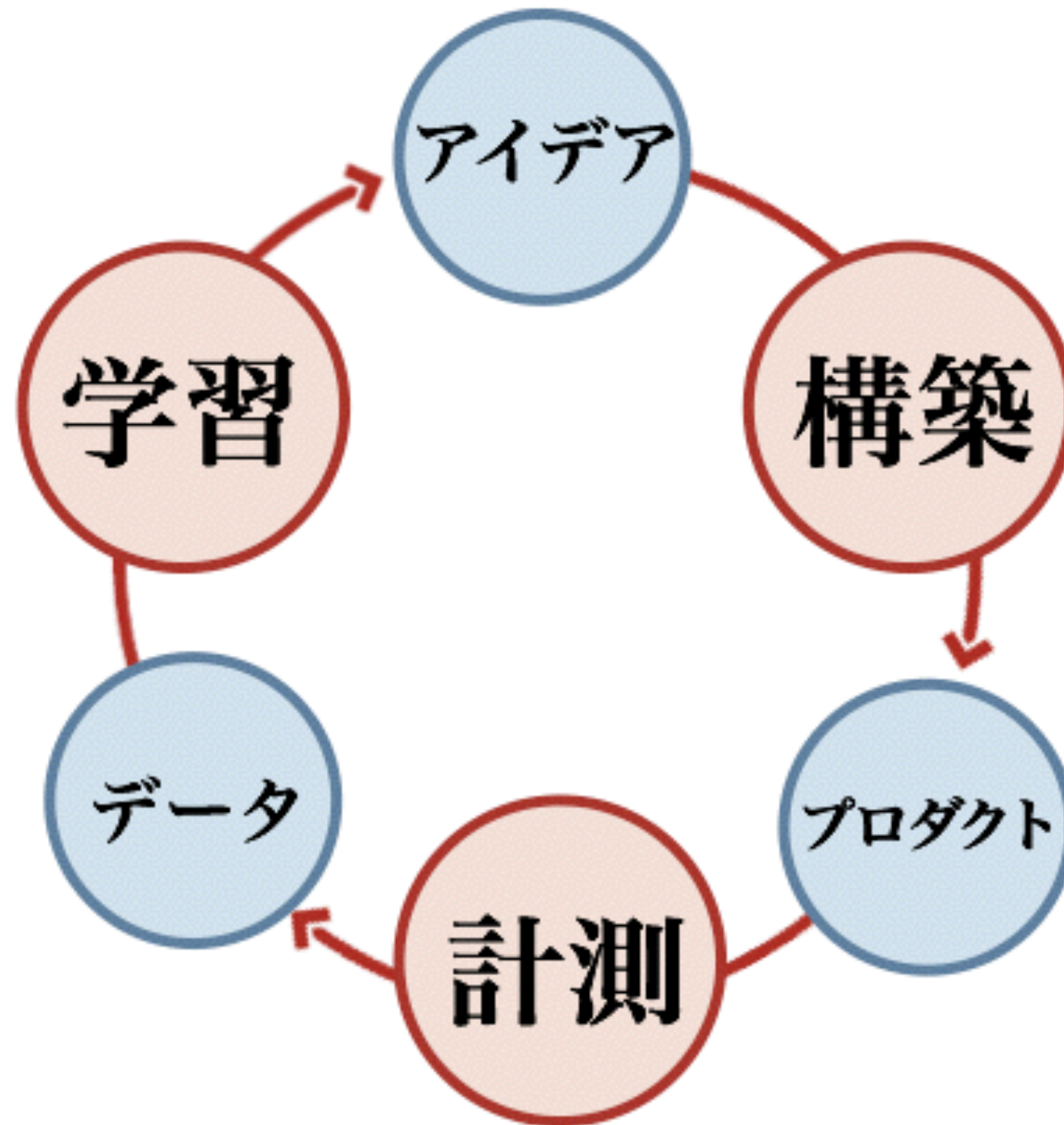


For many organizations, mob programming is **too extreme**; a WIP limit of one is probably not suitable for everyone.

翻訳はしばし待て!!

ストーリー

⇒ 仮説（指標も一緒に）



作ったら終わりではない

▶ 仮説は検証されるまで続く

- 「何を検証するか」を明確に

▶ コードを書かなくても検証可能なこともある（アイデア次第！）

- 『Lean Analytics』にいろいろ載ってる
（宣伝ばかりでスミマセン……）

その他いろいろ

インクリメンタルな設計 ⇒

▶ ⇒ リバーシブルな設計

- Facebookの話みたいな感じ

▶ ⇒ ランタイムな設計

- 動かしながら設計を変えていく
- （それSmalltalkでできるよ）

テストファースト ⇒

▶ ⇒ 仮説テストファースト

- 仮説を検証するために何ができるか
- 結果が出るまでに時間がかかるのが難点

▶ ⇒ 探索的テストファースト

- テスターと協業
- フィードバックループの深化

本物の顧客参加 ⇒

▶ ⇒ 本物の顧客開発

- 建物の外に出て、顧客を見つけ出す

▶ ⇒ 本物のユーザー参加

- 早い段階から実際に使ってもらおう

チームの縮小 ⇒

▶ ⇒ マイクロチーム (cf. マイクロサービス)

- Bounded Contextごとにチーム文化を生成
- それぞれが自律している (ex. Spotifyのチーム)

▶ ⇒ ひとりのチーム

- フルスタックエンジニア (死語?)
- スタンドプレーから生じる～みたいな

▶ ⇒ マシンのチーム

- チャットで話しかけたらボットだった!!
- データ駆動で意思決定を支援 (cf. 「アイアンマン」のジャービス)

[NEW] プロボノ (ボランティア + 公共性)

- ▶ 弁護士や医師などの活動
- ▶ 社会に役立つソフトウェア開発
 - ex. Code for Japan
- ▶ 社会的な意味での
「ソーシャルチェンジ」

他にも考えてみてください！

全体のまとめ

- ▶ おじさんたちの青春 (XP lives)
- ▶ もはやアジャイルは死んだ (Agile/XP dies)
- ▶ XPなら自分たちでやれる (XP lives again)
- ▶ 人間性の回復
 - HRT、ペアプロ、ときめき、ゆとり
- ▶ 適応性の維持
 - できることを増やす、経験重要、続けることが正義
- ▶ もっとエクストリームにする **たゆまぬK.U.F.U.**

おわりに

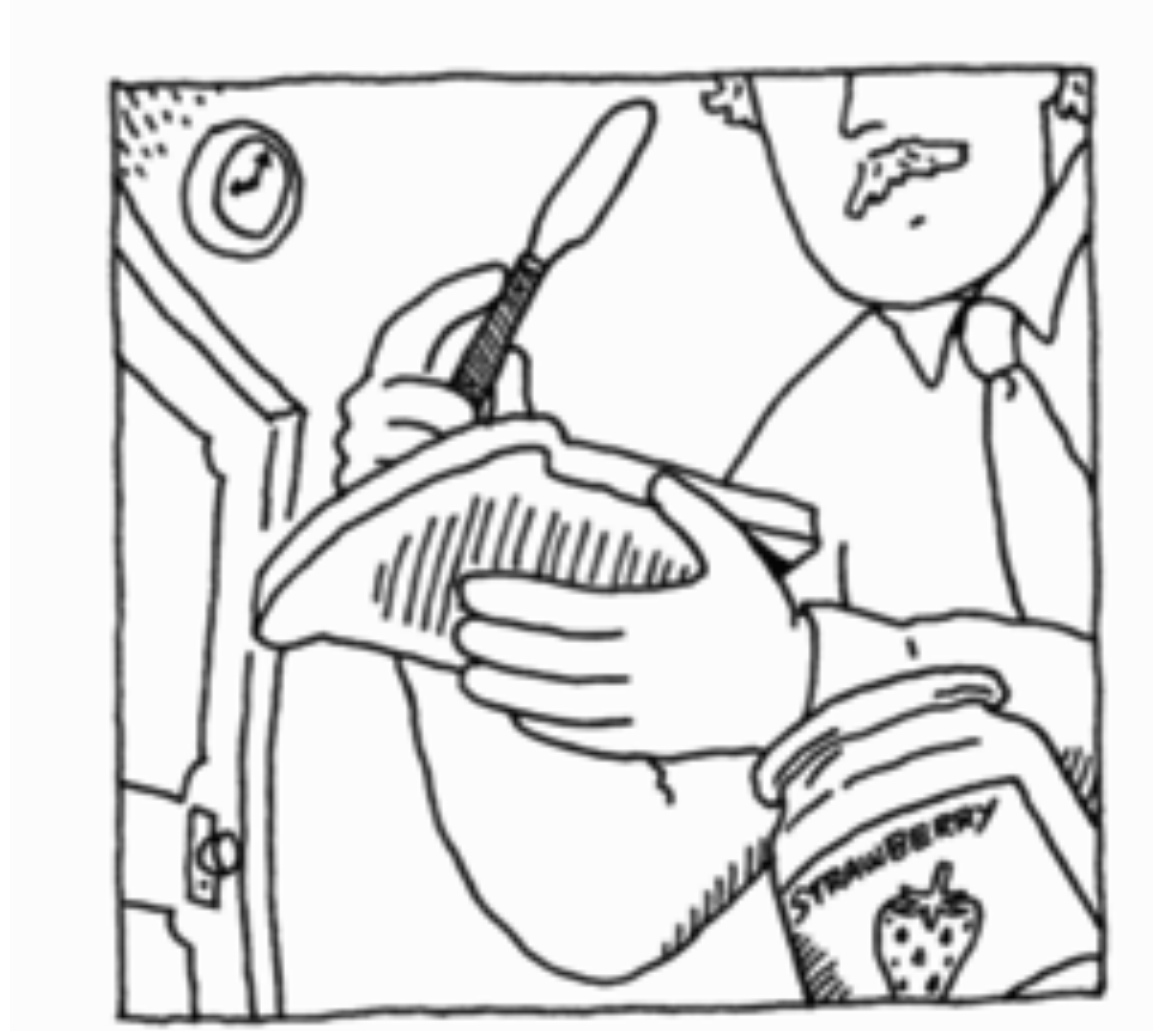
ラズベリージャムの法則



広げれば広げるほど薄くなるさ。

G.M.ワインバーグ『コンサルタントの秘密』

イチゴジャムの法則



粒があれば、どこまでも薄くはならない。

G.M.ワインバーグ『コンサルタントの道具箱』

自分の道を歩き出す「勇気」を

XPの価値の最後の1つ



荒木飛呂彦『ジョジョの奇妙な冒険』

エクストリーム プログラミング

Kent Beck・Cynthia Andres 共著
角 征典 訳

Extreme Programming *Explained*

Embrace Change,
2nd Edition

OHM
Ohmsha

おしまい & 質疑応答

@kdmsnr 

kado.masanori@waicrew.com

XPに限らず、
お話ししましょう :-)