

サンキュー、XP

XP祭り 2026 基調講演

2026-09-05

ワイクル株式会社 代表取締役

角 征典（カド マサノリ）@kdmsnr

kado.masanori@waicrew.com

eXtreme Programming

30 GREAT YEARS!

Thanks XP!



平凡な男が辿った数奇な人生
原作 スティーヴン・キング × 脚本・監督 マイク・フラナガンが
仕掛ける感動のミステリー

トム・ヒドルストン キウエテル・イジョフォー カレン・ギラン WITH ジェイコブ・トレンブレイ AND マーク・ハミル



サンキュー、 チャック

5.1 fri

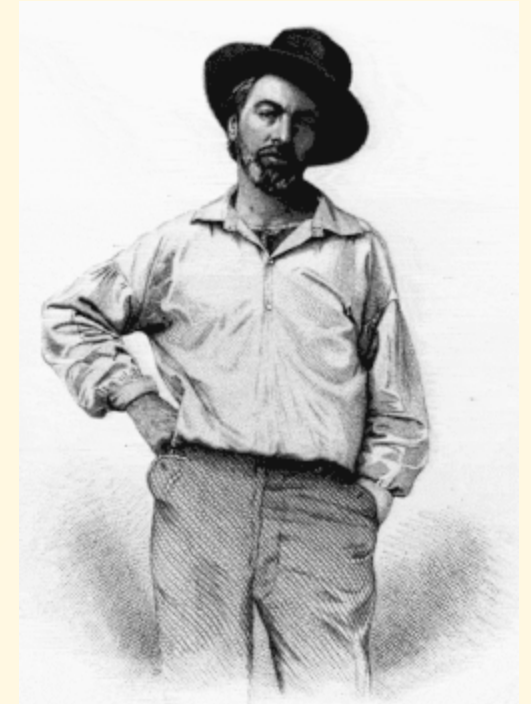


世界の終わりに明かされる、
愛すべき贈り物とは？
サブライム

SHOCHUKU KINOKUNIYA TOHO GAGA

"I contain multitudes" by ウォルト・ホイットマン

- わたしのなかにはたくさんのものがある [1]
- おれは多様性をかかえている [2]
- わたしの中にはいろんな面がいっぱいあるんだ [3]
- 人生は山あり谷あり (Life contains ...) [4]



-
1. 高山真由美（訳）『チャックの数奇な人生』文藝春秋
 2. 飯野友幸（訳）『おれにはアメリカの歌声が聴こえる: 草の葉(抄)』光文社
 3. ボブ・ディラン「I contain multitudes」中川五郎（訳）
 4. テイラー・スウィフト「The End of an Era」エピソード1

画像：https://en.wikiquote.org/wiki/Walt_Whitman

✧ XP Contains Multitudes

XPにはたくさんものがある

「サンキュー、チャック」 的アジェンダ

第3幕（現在） サンキュー、XP

XPの名前は忘れられようとしている

第2幕（全盛期） XPプログラマーたち

XPは「ソフトウェアで大切なこと」にたどり着いた

第1幕（過去） XPにはたくさんのものがある

2人の思想とエクストリームなプログラミング

3 サンキュー、XP

eXtreme Programming

30 GREAT YEARS!

Thanks XP!

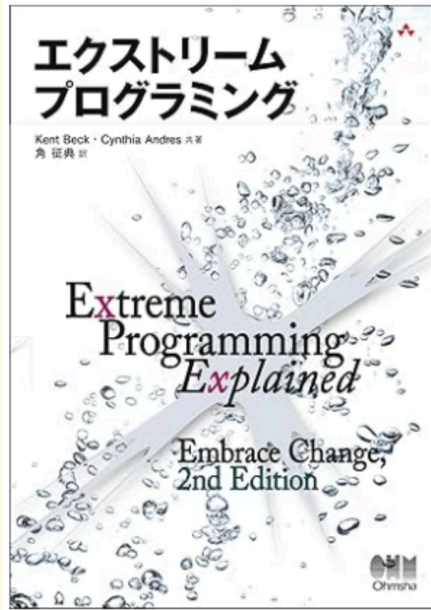
生命力が失われた荒野（2020） by ケント・ベック^[1]



1. [Extreme Programming Creator Kent Beck: Tech Has a Compassion Deficit](#)

画像：映画『マッドマックス 怒りのデス・ロード』（2015）

新版の発刊とXP祭り基調講演（2015）



**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

**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

https://kdmsnr.com/slides/20150626_what-did-xp-tell-you/

https://kdmsnr.com/slides/20150912_xp-lives-again/

2015年頃の雰囲気「アジャイルは死んだ」



アジャイルは死んだ by デイヴ・トーマス

1. アジャイルが**形容詞**から**名詞**に成り下がった
 - 形容詞「アジャイル」は目指すべき状態
 - 名詞「アジャイル」は商品として販売できる
2. 商品になった**認定資格**と**思考停止**（スクラム批判）
 - 自分で考えることよりも与えられたプロセスを
3. **大きな組織向けのフレームワーク**の登場（LeSSとSAFe批判）
 - アジャイル開発は小さなチーム向けのもののなのに？
4. 特定の**プラクティスの宗教化**
 - 「それ@t_wadaの前でも同じこと言えんの？」ゴメン

犬が指先をじっと見る問題（2015）

プラクティスそのものに注目してしまう問題



月を差している指に集中するな by ブルース・リー

有名な「Don't Think, FEEL.」の直後のシーン



アジャイルは本当に死んだのか？

- マネジメント・ファッション (MF) 理論
 - ある経営手法を集团的に信じること (Abrahamson, 1996)
 - 普及する手法には**解釈可能性**がある (Benders & Van Veen, 2001)
 - 普及すると概念がさらに**曖昧**になる (Giroux, Hélène, 2006)
 - MF理論から見たアジャイル (Madsen, 2020)
 - 元々「アジャイル」には**解釈可能性**と概念の**曖昧**さがあった
 - コンサルタント、コーチ、研修、認定資格が普及を後押しした
 - 結果、他の業種まで（**中身はさておき**）名前は広まった
- ▶ "アジャイル"は曖昧なので、マネジメント理論としては成功した

でも、XP流行ってないですよね？

©蓮見翔（ダウ90000）

いわゆるストレートな質問キタコレですねwww^[1]

いやいや、継続的インテグレーション、リファクタリング、テスト駆動開発、ユースストーリー、ペアプログラミングとか^[2]、だいたいXPから生まれたか、XPがプラクティスとして整理して世に広めたものなんですよ。なので、我々がいま当然のように使っている開発プラクティスのかなりの部分はXPの遺産なわけで、XPって「流行ってない開発手法」ではなくて、むしろ成功しすぎた結果、個別のプラクティスが現代のソフトウェア開発に完全に吸収されて、いちいちXP由来だと意識されなくなっただけっていうか、空気みたいな存在なんですよ。あるいは、電気や水道とかと同じで、なくなったときに困る、そういうインフラみたいな存在になってるんですよ。ここはめちゃくちゃ重要ですね。試験に出ますね、出ないほうがおかしいので、出ますね😊

1. <https://dic.pixiv.net/a/フォカヌポウ>

2. <https://bliko-jp.github.io/ExtremeProgramming>

ドットコムバブルの崩壊で約10年間が失われた

9.11で仕事がなくなり、精神的に辛い時期だったと語っている



2002年からの失われた10年間のようなもので、

XPはもう一般的な技術ですよね？

わざわざ語らなくてもよくないですか？

そう、一般的な技術になってしまったのだ

- 「これはどういうことですか...？ 一般的な技術です。」
- 「あれが **XP（エクストリームプログラミング）** だよ。」

いわゆる「ゾルトラーク現象」というやつですね🧙‍♂️

eXtreme Programming

30 GREAT YEARS!

Thanks XP!

アジャイルの死後にシンプリシティを叫ぶ (1/2)

- もう一度、世界を変えよう
 - 冒頭からカッコイイですね
- (XPの5つの価値は) 人生にとって核となる価値
- 変化曲線が指数関数的になったように見える世界で (中略) 「変化を抱擁する」必要がある
 - プロセスをシンプルにし、**仕組み化する**
 - **実用的**なものと**趣味的**なものを混ぜ合わせる
 - 趣味として遊んだ結果を次の仕事に取り入れる



アジャイルの死後にシンプリシティを叫ぶ (2/2)

島田さんの「**訳者あとがき**」が素晴らしいので読もう（以下、引用）

- Daveが辿り着いたひとつの核心が「**agency（行為主体性）**」
 - 自分で考え、自分で選び、自分で変えていく
 - 方法論や組織論そのものよりも、**開発者一人ひとりが自分の仕事に対する主導権をどう取り戻すか**
- （そのために）Daveが辿り着いたのが「**simplicity**」
 - 抱え込んでいる複雑さを減らしていく
 - **開発者としてのagencyを回復していく**

方法論や組織論よりも 開発者一人ひとりの主導権

誰かを変えようとするのではなく自分が変わる

ソフトウェアクラフトマンシップ宣言（2009）^[1]

自らソフトウェア開発の水準を高め、他者にもその技術を伝達する

- 動くソフトウェアだけでなく、**精巧に作られたソフトウェアも**
- 変化への対応だけでなく、**着実な価値の付加も**
- 個人との対話だけでなく、**専門家のコミュニティも**
- 顧客との協調だけでなく、**生産的なパートナーシップも**

⚠ **Martin Fowler**は、ビジネスと開発の分断が広がると批判^[2]

1. <https://manifesto.softwarecraftsmanship.org/>

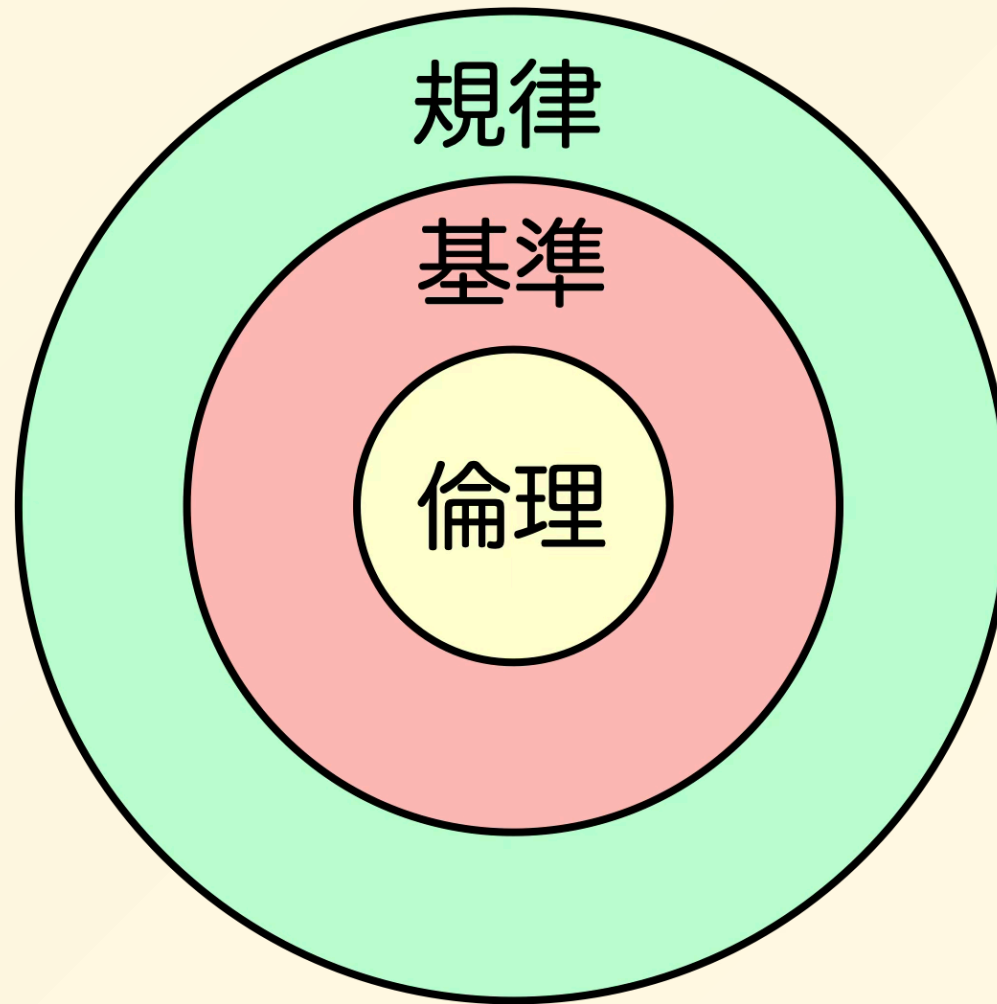
2. <https://bliki-jp.github.io/CraftmanshipAndTheCrevasse>（2011）

クラフトマンシップ by アンクルボブ (2021)

- クラフトマンシップとは、何かをうまくやる方法を知っている状態である。それは、優れた指導と豊富な経験の結果である。
- プログラマーが自分で正しさを決めるだけでは不十分である。これからは、**規律、基準、倫理**が必要になるはずだ。

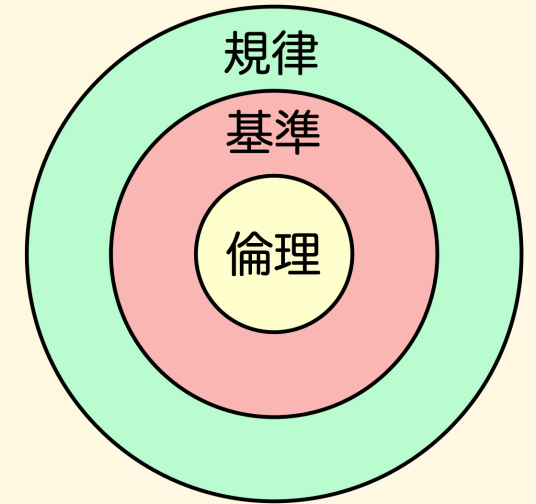


クラフトマンシップの規律、基準、倫理



クラフトマンシップの規律、基準、倫理

- **倫理**（守るべきこと） a.k.a. プログラマーの誓い
 - 無害、誠実、チームワーク
- **基準**（期待されていること）
 - 品質、生産性、勇気
 - 倫理に対応させようとしているようだが、いまいち違いがわからない.....
- **規律**（XPのプラクティス）
 - テスト駆動開発、リファクタリング、シンプルな設計、協力的プログラミング、受け入れテスト



プログラマーの誓い (1/3)

無害／品質に関わる約束

- 約束1：有害なコードを作らない
- 約束2：欠陥のあるコードを故意に残さない
- 約束3：コードが正常に動作する証拠を用意する

プログラマーの誓い (2/3)

誠実／生産性に関わる約束

- 約束4：小さく何度もリリースする
- 約束5：執拗に作品を改善する
- 約束6：生産性をできる限り高める

プログラマーの誓い (3/3)

チームワーク／勇気に関わる約束

- 約束7：相互にカバーする
- 約束8：正直に見積もる
- 約束9：仲間のプログラマーを尊重する
- 約束10：学習と向上を怠らない

特に言うことはないですよね😊

特に言うことはないのに
「無害／品質に関わる約束」は
あんまり守られていない!!

個人の倫理や基準だけでは足りない

ミスター・ベイツvsポストオフィス (2024)

英国史上最大規模の冤罪スキャンダルを描いた衝撃作



イギリスの郵便局冤罪事件

- 2000年に運用開始した会計システム「ホライゾン」に不具合
 - 郵便局長は赤字分を自腹で補填しなければならなかった
- 不正経理などの罪で郵便局長約1000人が起訴される
 - 収監：少なくとも236人
- 事件との関連が疑われる自死：少なくとも13人
- 被害者への賠償支給を開始（2021年春）
 - 2026年7月現在も賠償金の支払いは終わっていない
- **不具合を知らながら、それを組織として認めなかったことが問題**

<https://www.mystery.co.jp/column/k78ys6nr6j0f4hpj.html>

<https://jp.reuters.com/economy/QOIKQXZ36FNJDBFDEYX5JTU5MM-2025-07-09/>

<https://www.bloomberg.co.jp/news/articles/2025-07-09/SZ3UQ0T0G1KW00>

他の国でも見られる事件

▶ カナダの給与システム問題（システムの不具合）^[1]

政府職員の給与計算にミス。財政的損失は51億カナダドル（約36億米ドル）。これからクラウドに置き換え予定（2027年夏～2031年）。

▶ 日本の「消えた年金」問題（データ管理ミス）^[2]

1997年、制度ごとの年金番号を「基礎年金番号」に統合。2026年3月時点でも、約1,652万件の不明記録が残る。

▶ オランダの育児手当事件（AIの判定ミスと差別的制度）^[3]

税務当局が数万人の親を「不正受給者」として認定。制度上の人種差別。2021年1月に内閣総辞職。最終的な補償完了目標は2027年末。

1. <https://spectrum.ieee.org/it-management-software-failures>

2. <https://www.nenkin.go.jp/service/nenkinkiroku/torikumi/sonota/kini-cam/20150601-05.html>

3. <https://globe.asahi.com/article/15088788>

じゃあ、どうすればいいのか？ 🤔

eXtreme Programming

30 GREAT YEARS!

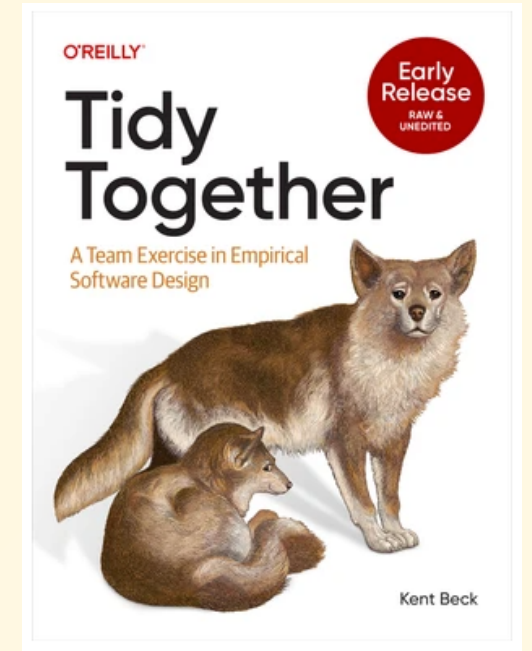
Thanks XP!

XPはソーシャルチェンジである

この言葉は「人間関係の変化」だと言われてきた

人間関係を整理 (tidy) したいケント・ベック

- "人間関係は私にとって非常に難しい概念です。私は生まれつき人の心理や社会的な状況、チームの力学を理解するのが得意ではありません。"
- "しかし幸いなことに、特別な才能がなくても建設的な人間関係を築くための普遍的な原則がいくつか存在します。"



② えっ、人間関係の原則？.....大丈夫そ？（不安）

「信頼工場」としてのXP^[1]

- LLMの登場でコードが爆速で進んでいる
 - コードと信頼は両輪だが、**信頼が追いついていない**
- 思い返してみると、XPは**コードと信頼**の両方を生み出すものだった
 - XPの開発者の信頼できる行動から、コードの信頼性が生まれる
 - つまり、XPは **信頼工場 (Trust Factory)** である ← **いい表現！**

③ だが、「信頼」とは何だろう？ 信頼が目的なのか？

最終的には「ギークが安全に感じる世界」が理想

“ 私の使命は「ギークが世界で安全に感じることを助ける」です。 [1] ”

つまり、これまでの話をまとめると、

「信頼の獲得 → 人間関係の変化 → 世界で安全に感じる」

ということになる。

これはこれで大切なのだが.....。

② 安全になったギークは、その才能を何に使うのか？

その才能を何に使うのか？

- "（シリコンバレーは）優秀な技術者を大量に呼び込むことに成功している。集めた貴重なリソースが何に使われるのかと言えば、**写真共有やチャットなどちょっとしたアプリを開発するためだ。**"
- "シリコンバレーはいまなお警察・医療・教育・国家安全保障といった**公的な領域へ足を踏み入れることに乗り気ではない。**"



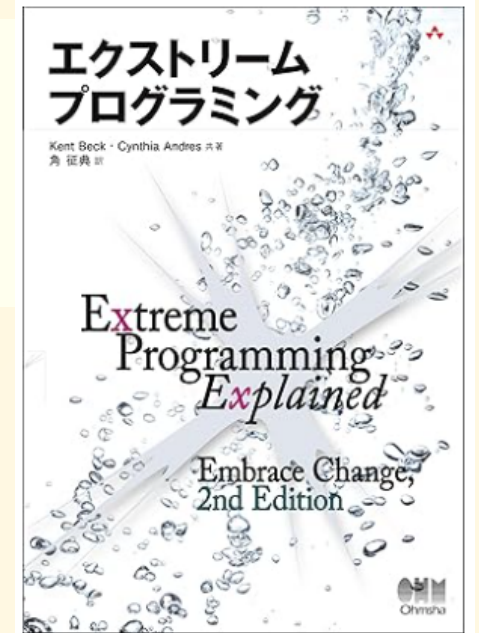
💡 軍事利用は賛同しないが、この問題意識は納得できる。

身近な「ソーシャルチェンジ」でいいのか？

信頼や人間関係や安全を実現するだけでいいのか？

“
XPとは、あなたが**自分の理想**について考え、
その理想にもとづいて**行動する**ための方法だ [1]

もっと言えば、
顧客や組織の利益に還元できない「**理想**」を語り、
その理想にもとづいて「**行動**」すべきではないのか？



思考の補助線としてのハルトムート・ローザ



**私たちには時間がない、
あふれんばかりに勝ち取っているはずなのに**

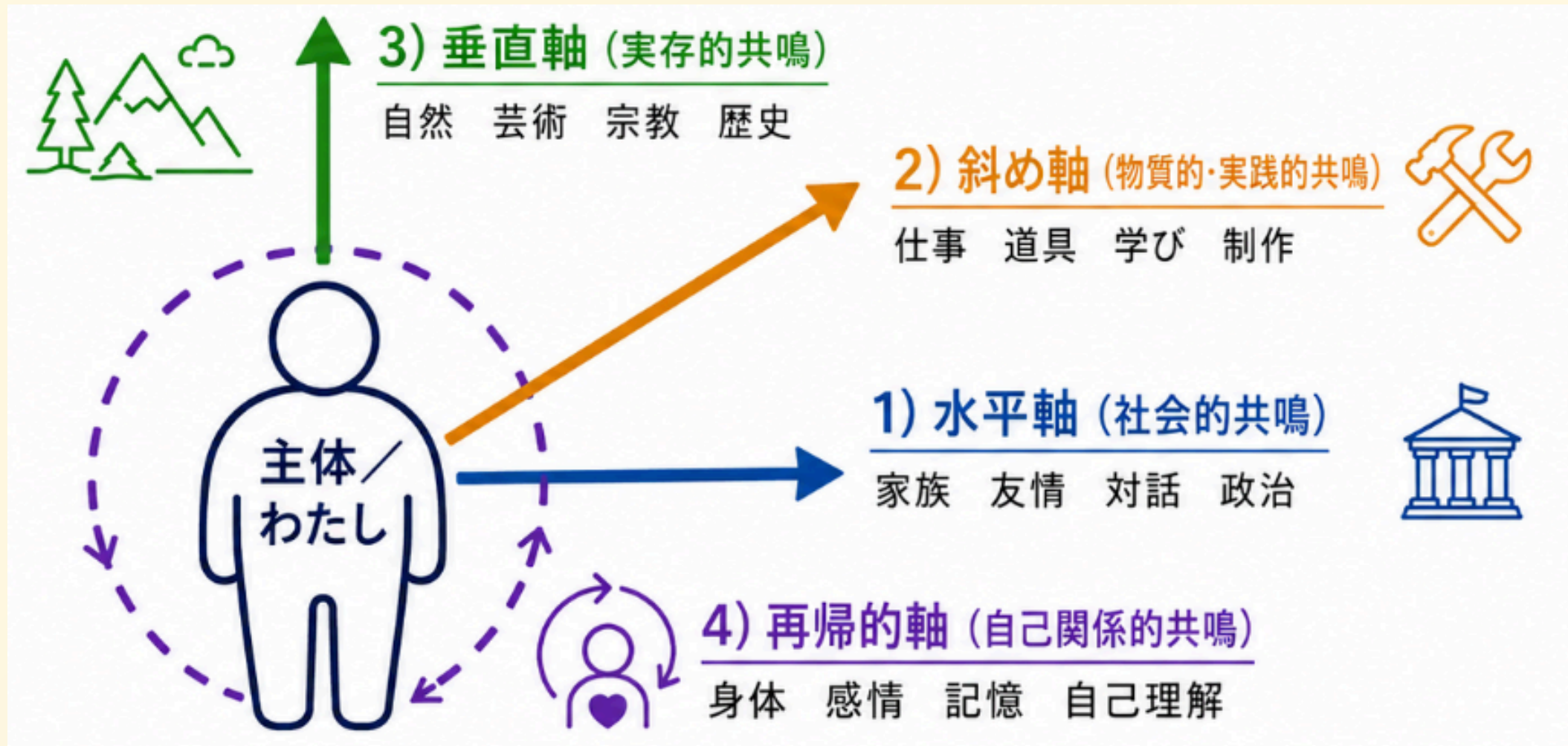
近代の技術革新はなぜ人々を時間欠乏から解放しないのか？
近代のパラドクスに隠された加速の論理の解明、その文化的・構造的起源の探求、
個人・集団のライフスタイルへの影響や歴史経験の変化に関する分析まで――

社会・経済のあり方が根本から問われるいまこそ読みたい、よき生のための社会学

ドイツで最も注目されている社会学者の著書、初邦訳

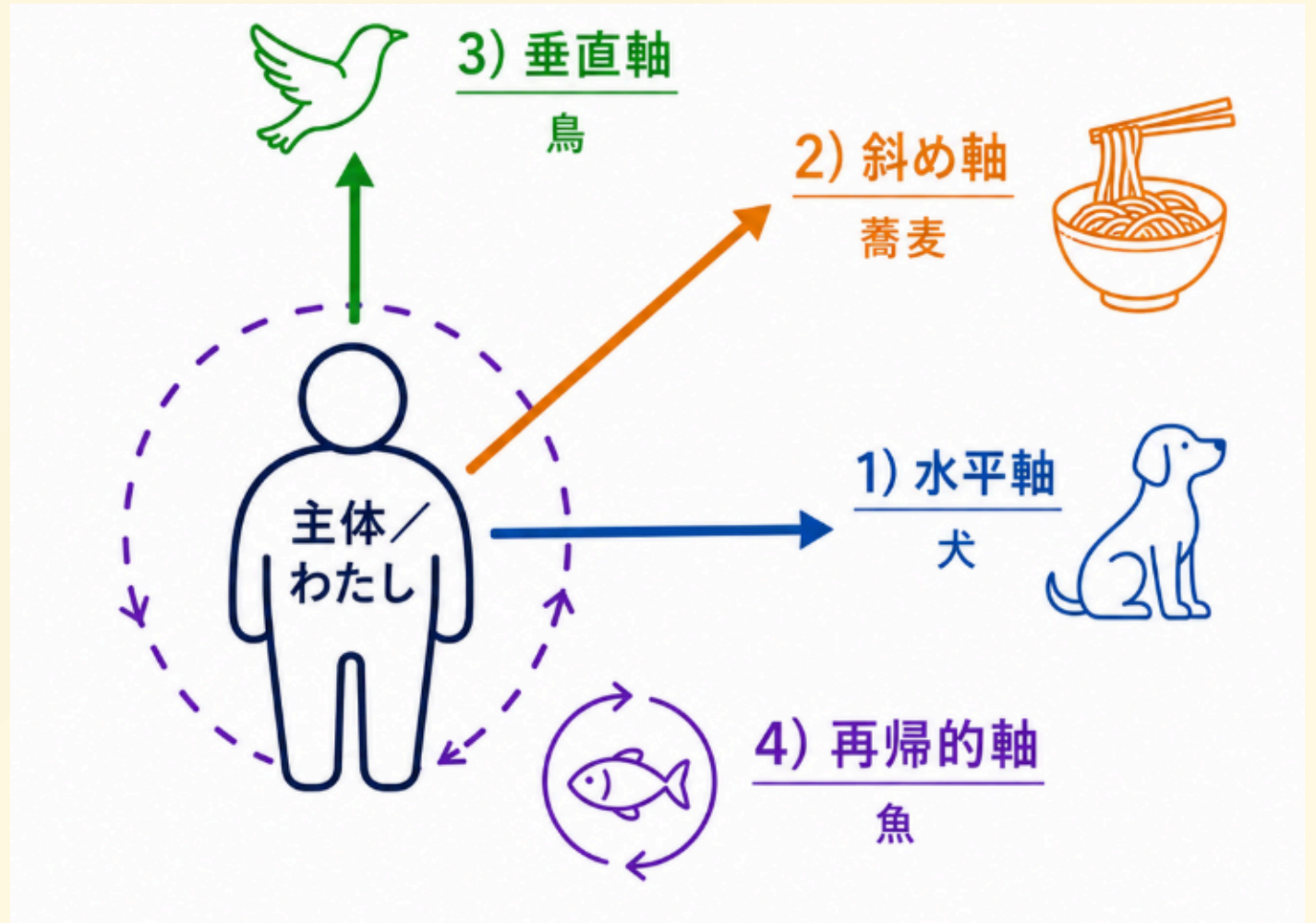
福村出版

加速社会に対抗する「共鳴」(Rosa, 2019)



(オタク向け) 押井守のモチーフとの対比

- **犬**：仲間、家族、政治
- **蕎麦**：仕事、早いほど偉い（立ち食い蕎麦）
- **魚**：無意識、自分の頭で整理できない部分
- **鳥**：神、自然、宗教



XP（信賴工場）のプラクティスを共鳴の軸で読む

- 🐶 **水平軸**（同僚や顧客との関係性）：
 - ペアリング、週次計画、顧客をチームに含める
- 🍜 **対角軸**（構築するプロダクト）：
 - リファクタリング、継続的デプロイ、オブザーバビリティ
- 🐟 **再帰軸**（自分が書くコードに対する自信）：
 - プログラマーテスト、継続的インテグレーション
- 🦉 **垂直軸**（自分たちの存在を超えた何か）：
 - 🦴 ここが足りないのかもしれない

なぜ僕はコードを書くんだろう？



以前は「形式」も存在理由になり得たが.....



LLM時代にプログラミングの「形式」の意義は薄れてしまったようだ

XPの「形式」に集中するな！理想（垂直軸）を見る！

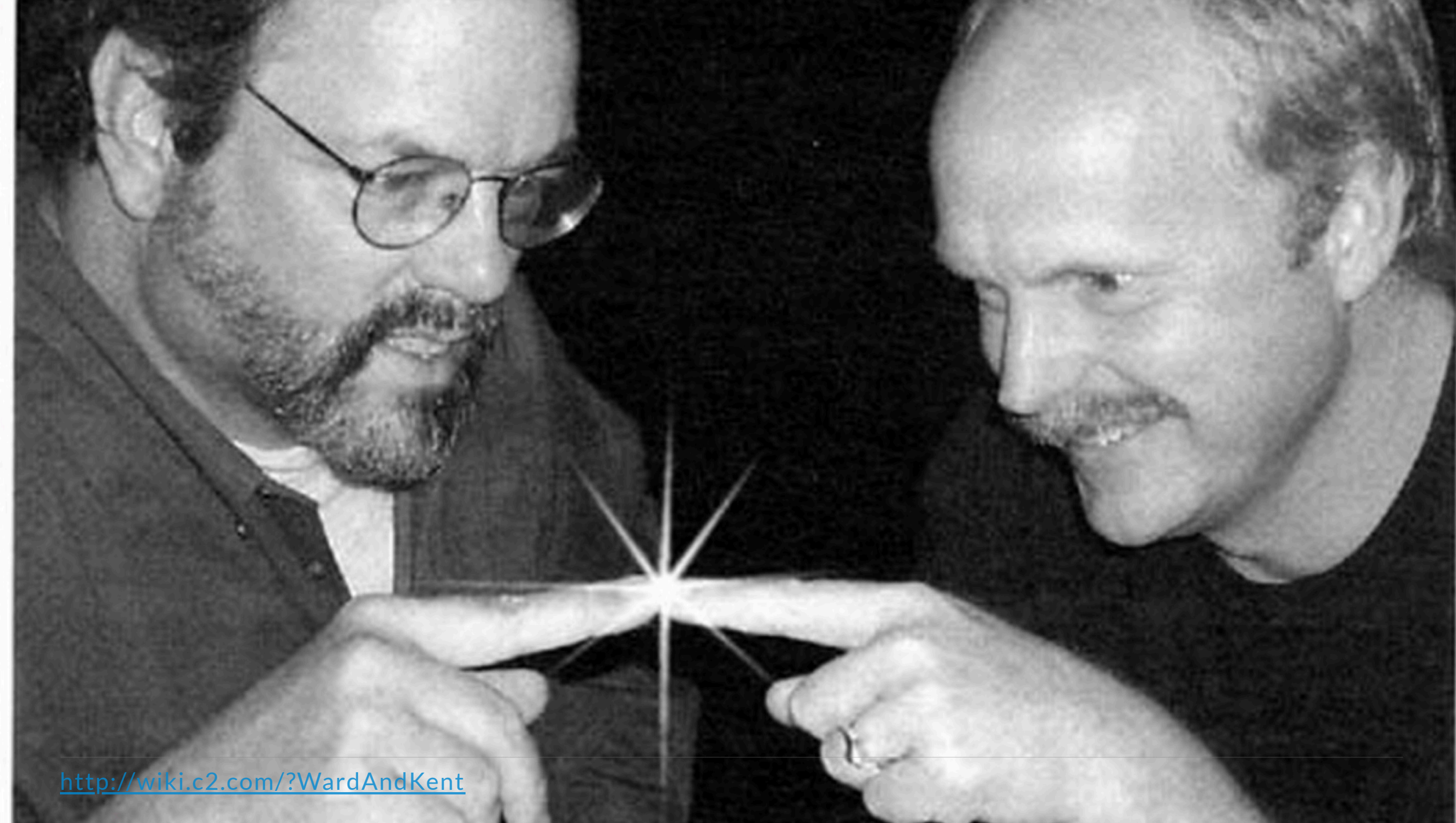


しかし、理想とは何かは誰も教えてくれない.....。

あなたの「理想」は何ですか？

XPは「理想は何か」と問いかけて
星になってしまったのである ✨

2 XPプログラマーたち



星のように登場したWardAndKent ✨

ケント・ベック says:

“
ウォードが私の情報源。私は彼が自然にやっていたことを説明しているだけ。
”

ウォード・カニンガム says:

“
ケントの最大の功績は我々が発見したものを極限（エクストリーム）まで高めてくれたこと。
”



音量調節の番号が11まである

だが、XPは（弱き）人間中心の規律 (Beck, 1998)

Extreme Programming: A Humanistic Discipline of Software Development

Kent Beck

CSLife and First Class Software

P.O. Box 226

Boulder Creek, CA

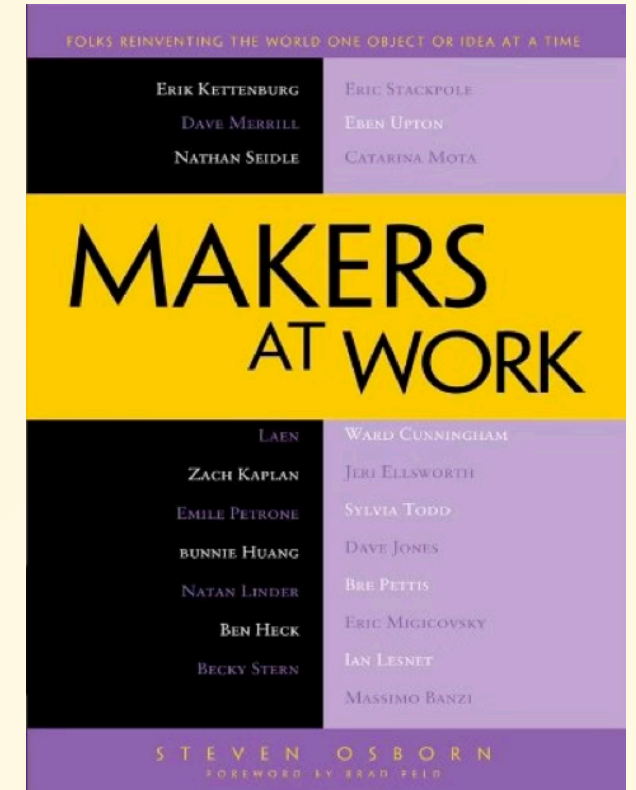
email: 70761.1216@compuserve.com

- 人間は**恐怖**を感じるが、恐怖は**経験**で払拭される
- 経験するには**積極性**が必要 → XPの価値「**勇気**」につながる

恐怖を「対話」で払拭する

ワードは「対話」を重視する人物だった

- 私は本を読むのが遅く、学校ではそれほど優秀ではありませんでした。でも、先生の話予測しながら聞くのは好きでした。
- 今でも人の話を聞いたり、コンピューターに耳を傾けたりすることが好きです。
- プログラミングしているときの自分の気持ちを話すことにも抵抗はありません。
- Kentも同じでした。だから、2人でただ一緒に座って、問題を解いていました。



☕ 月曜のコーヒー談義から始まる仕事のリズム

- 月曜： **Wardとコーヒーを飲みながら話し合い、その週のゴールを決める。**
- 火曜・水曜：実装する。
- 木曜：成果物をデモして、フィードバックをもらい、さらに洗練する。
- 金曜：その週の技術レポートを書いて仕事を終わる。たとえ失敗しても、翌週にうまくやればいい。



毎週月曜日に決まったリズムで
朝はコーヒーを飲みながら

対話は必ずしも「共鳴」ではない (Rosa, 2023)

- 共鳴の4つの条件：魅力的、応答性、変容、制御不可能
- 共鳴のない偽りの対話 🤗：
 - ときめかない、伝わらない、影響を受けない、意外性がない
- 共鳴しているとき、人間は**中動態**にある
 - ダンス 🩰🕺、即興演奏 🥁、白熱した議論 🗣️においては、主体が能動的であると同時に完全に受動的でもある
 - 例：私はダンスを踊っているのか？ 踊らされているのか？
- 人間が共鳴するとき、**「場」が行為主体性（agency）**を持つ
 - 達人デイヴと同じ言葉だが、所有者が違う

第49回 トロント国際映画祭 観客賞受賞

サンキュー、チャック

5.1 fri

映画『サンキュー、チャック』 <https://www.youtube.com/watch?v=ABYLtRT62v4>



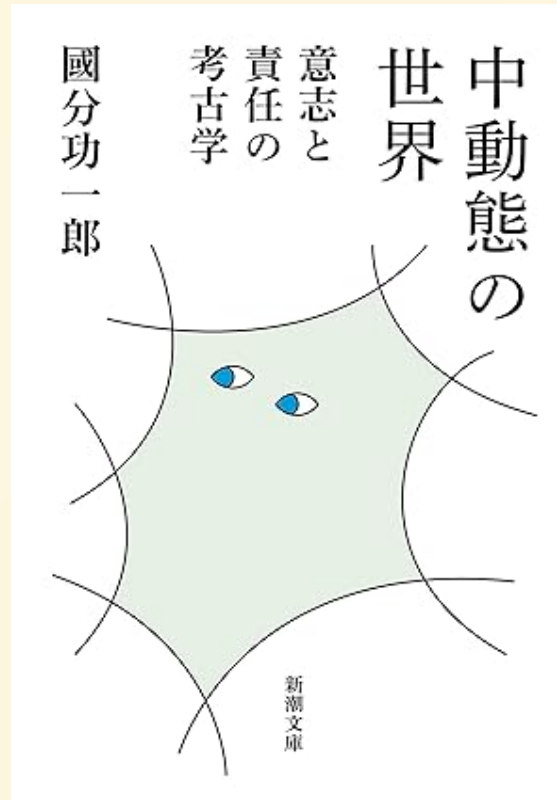


“

踊って踊られ相互関係
贈って贈られ相思相愛
笑いあえる日常 幸せじゃん

”

中動態といえば國分先生



- このためにギリシャ語教室に通っていてヤバい（リスペクト）
- 最近、TMNETWORKのファンだと知って、がぜん好きになった笑

中動態の世界

- **中動態**は能動態と受動態の**中間ではない**
 - 元々は**能動態**と**中動態**しかなかった
 - 受動態は中動態の派生形にすぎない
- 中動態は**主語が動作の内部にある**（場所になる）もの
 - 一方、能動態は主語の外で動作が完結するもの

中動態の例：XPの価値「リスペクト」（の動詞）

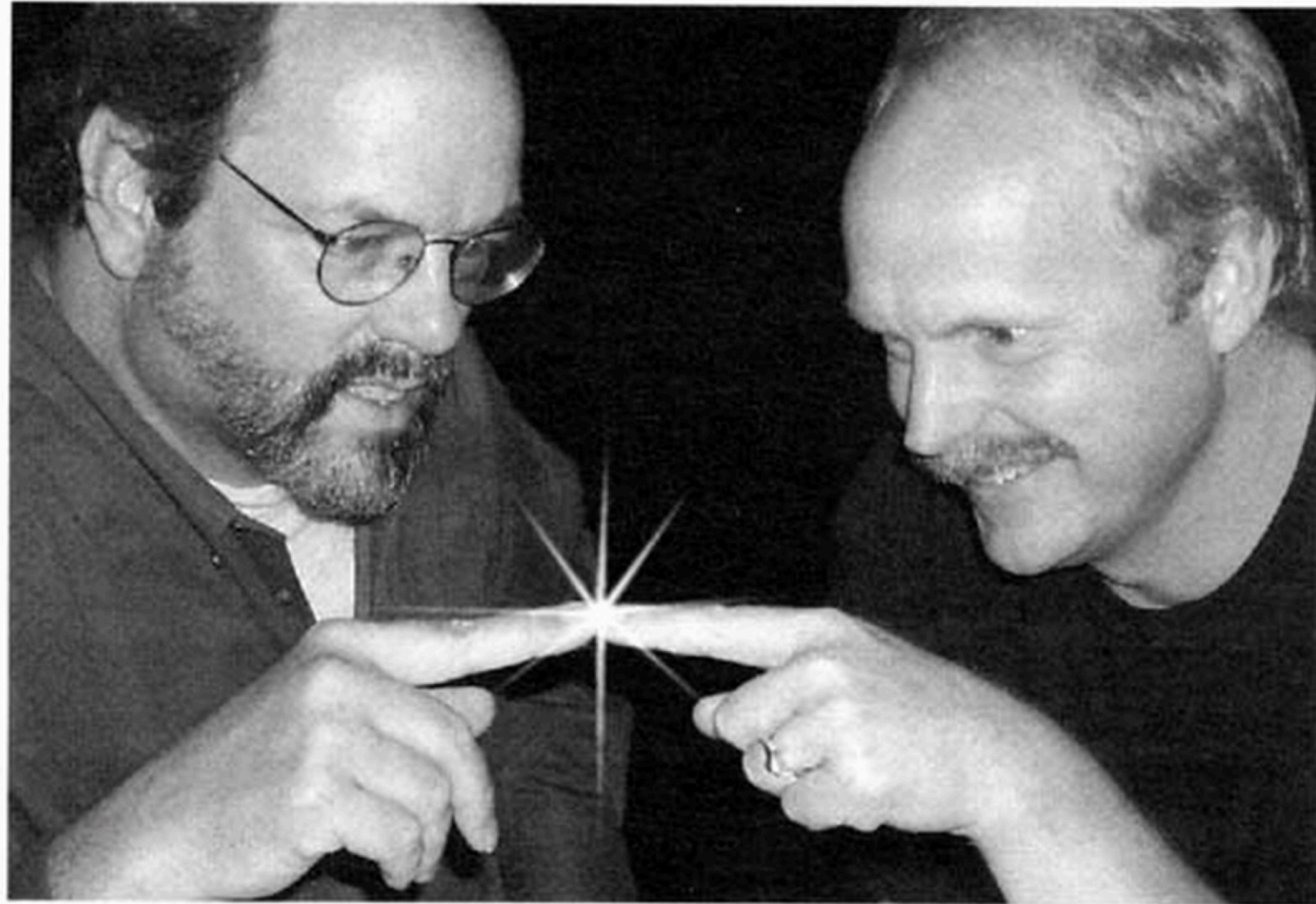
「リスペクトしています！」にも2種類ありそう

1. **ビジネスリスペクト**（能動態）：外からは見分けがつかないが、主語に何の変化もない、儀礼的な建前のリスペクト。
2. **ナチュラルリスペクト**（中動態）：心から思っているリスペクト。主語に何らかの変化を伴うような、共鳴のあるリスペクト。

② 中動態は主語が「動作の内部」にある

よく「組織を改善します」「アジャイルを導入します」とか言うけど、主語である「あなた」は、その変化の内部にいるのか？

WardAndKentの対話はお互いに「変化」していた

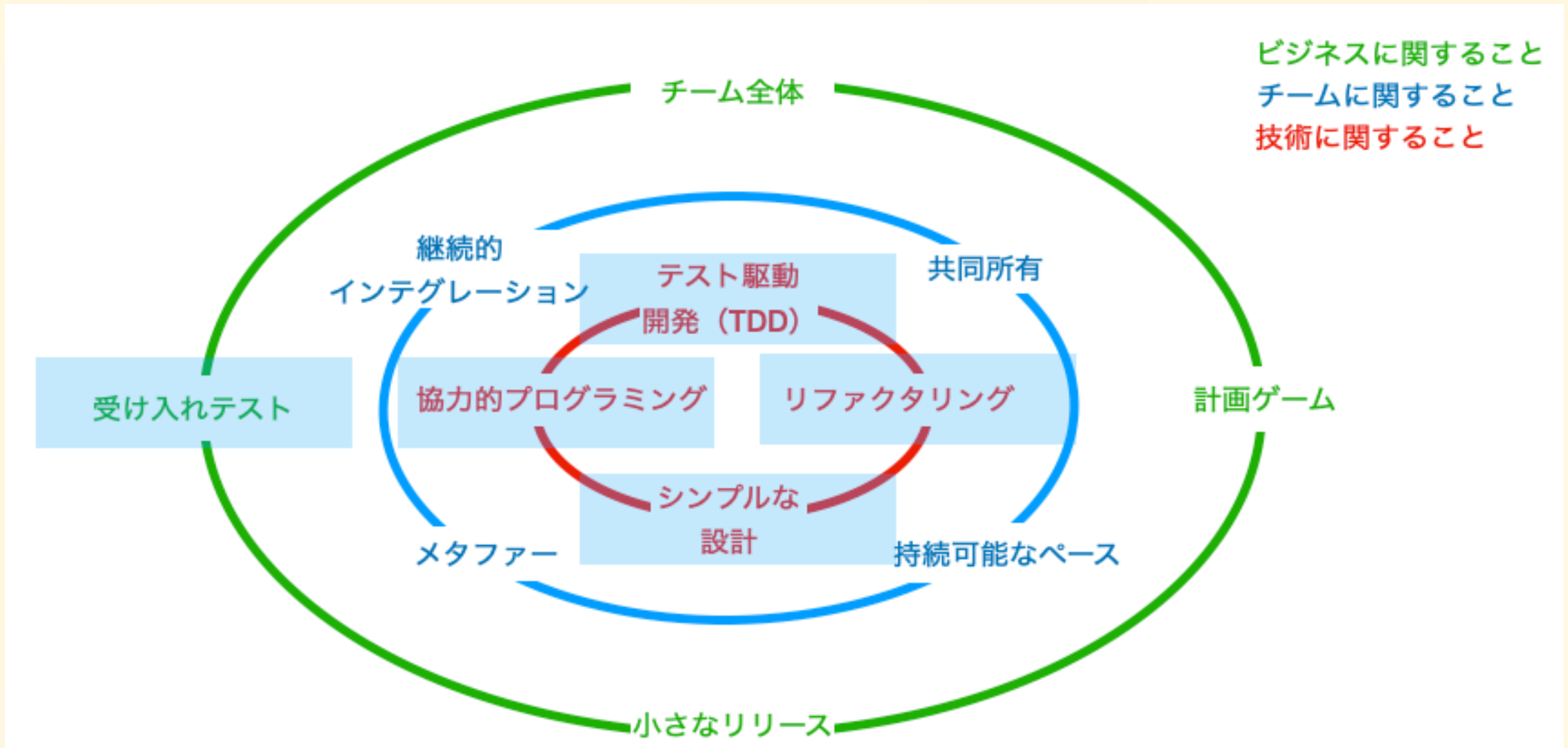


彼らが提唱する「ソフトウェアで大切なこと」

自ら「共鳴」しながらソフトウェアを開発する方法

1. コードを書くこと：当たり前（！）
2. テストすること：コードが完成したかどうかを教えてくれる
3. 設計すること：コードとテストだけではいずれ行き詰まる
4. 話を聞くこと：そもそも何のためのコードを書くのか？

XPの規律「サークル・オブ・ライフ」



ソフトウェアで大切なことをXPの規律にする

1. コードを書くこと：当たり前（！）
→ 協力的プログラミング
2. テストすること：コードが完成したかどうかを教えてくれる
→ TDD + リファクタリング
3. 設計すること：コードとテストだけではいずれ行き詰まる
→ シンプルな設計
4. 話を聞くこと：そもそも何のためのコードを書くのか？
→ 受け入れテスト

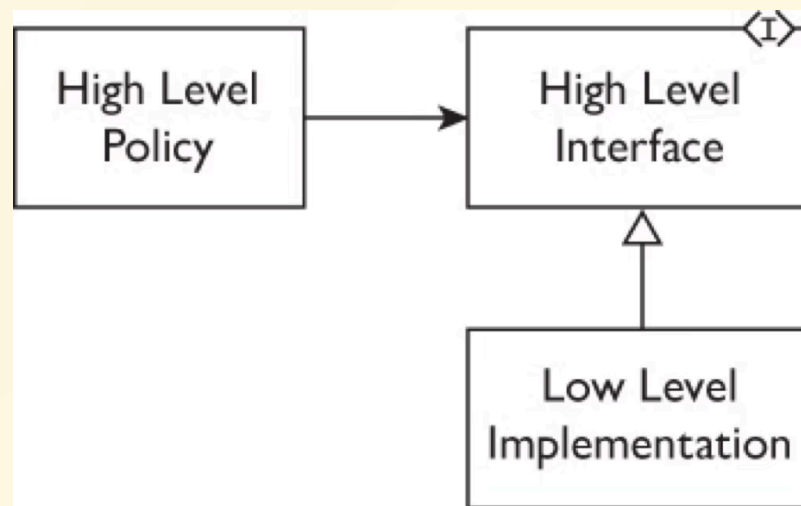
シンプルな設計

共鳴を続けるには「変化」しやすい設計でなければならない



シンプルとは

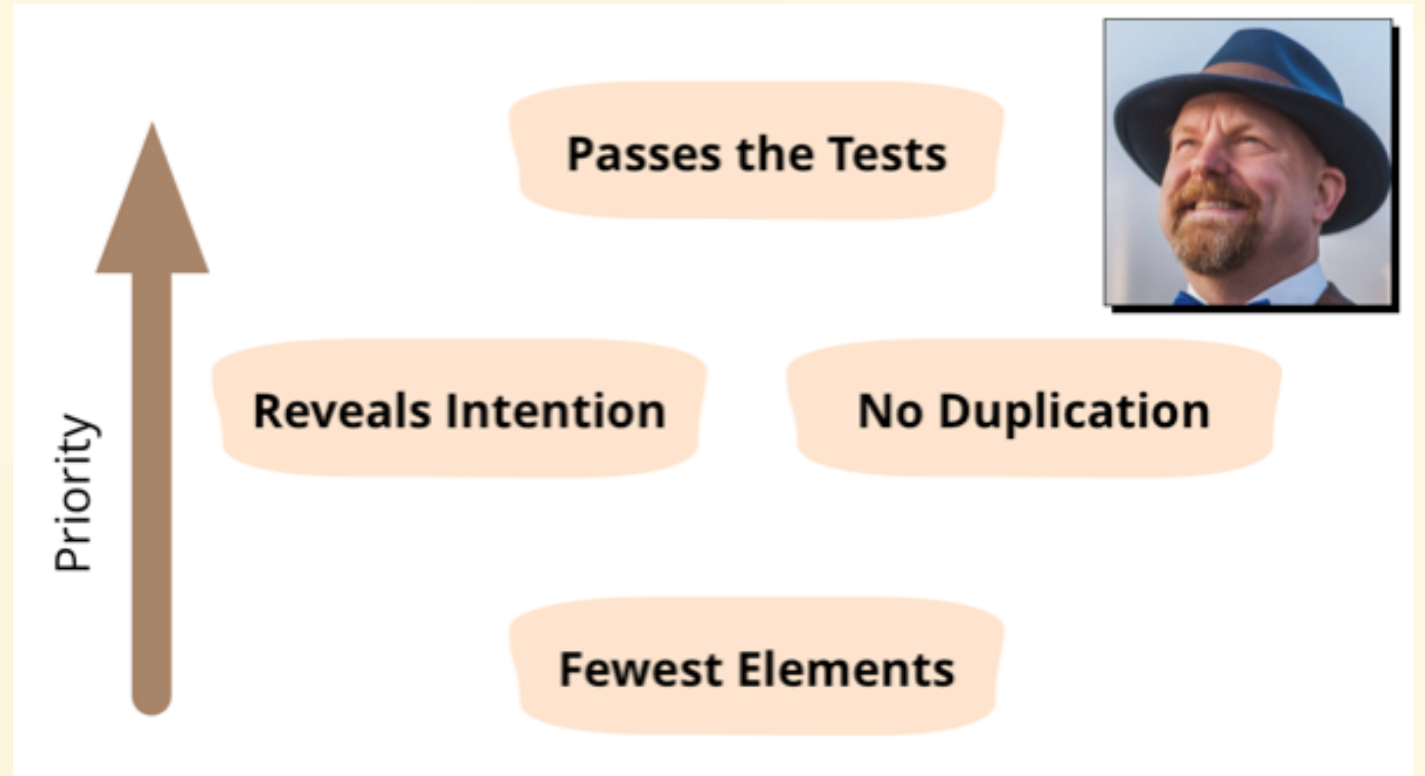
- シンプル：Simplex / Complex（複雑、複合）
 - いろんなものが複雑に混ざっていないこと [1]
 - （ソフトウェアでは）方針と詳細が混ざっていないこと
- たとえば「抽象化」で依存を管理する（詳細 → 方針） [2]



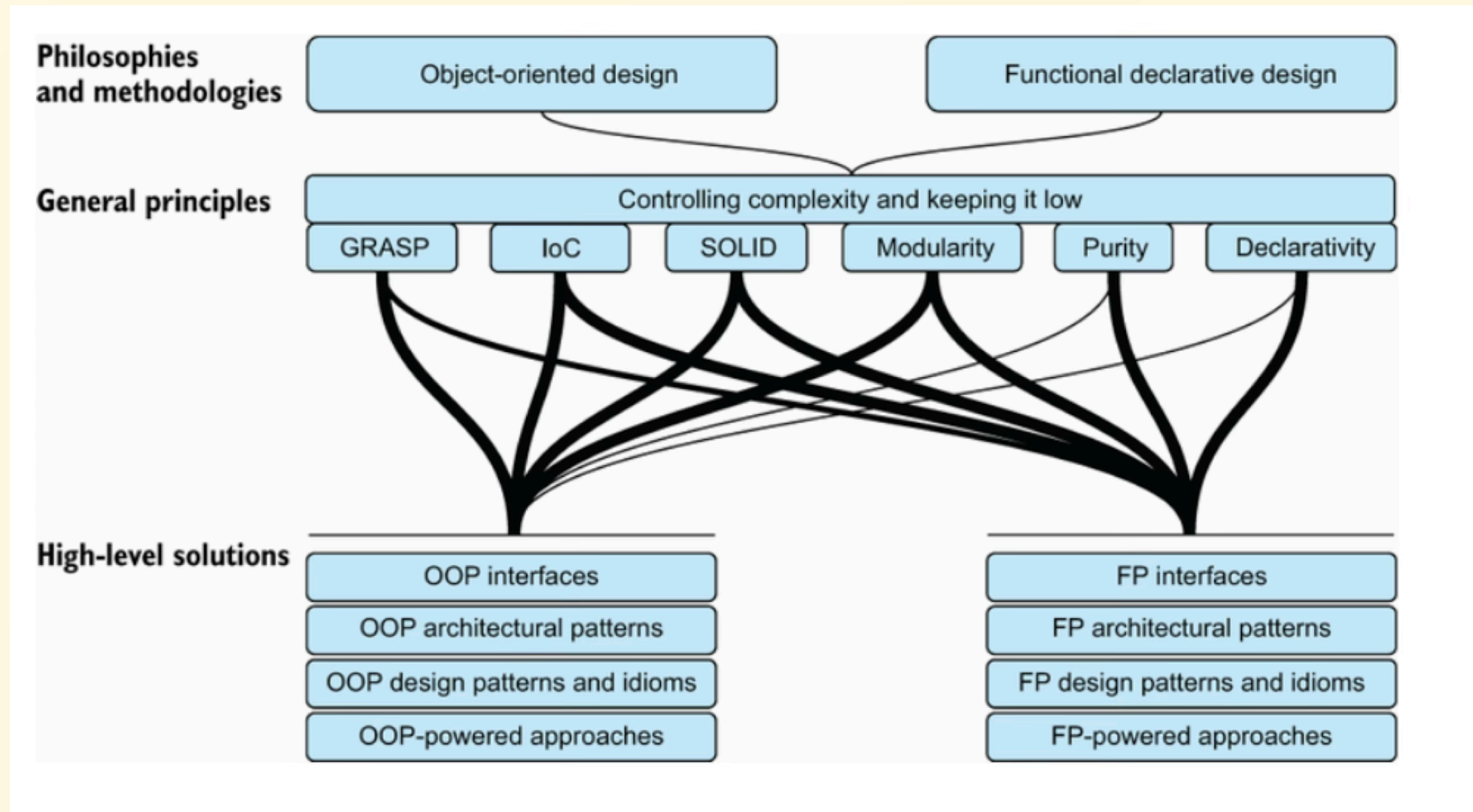
1. "Simple Made Easy" - Rich Hickey (2011)
2. Bob Martin 『Clean Craftsmanship』 KADOKAWA

ケント・ベックのシンプルな設計のルール

1. テストをパスさせる
2. 意図を明らかにする
3. 重複を排除する
4. 要素を最小限にする

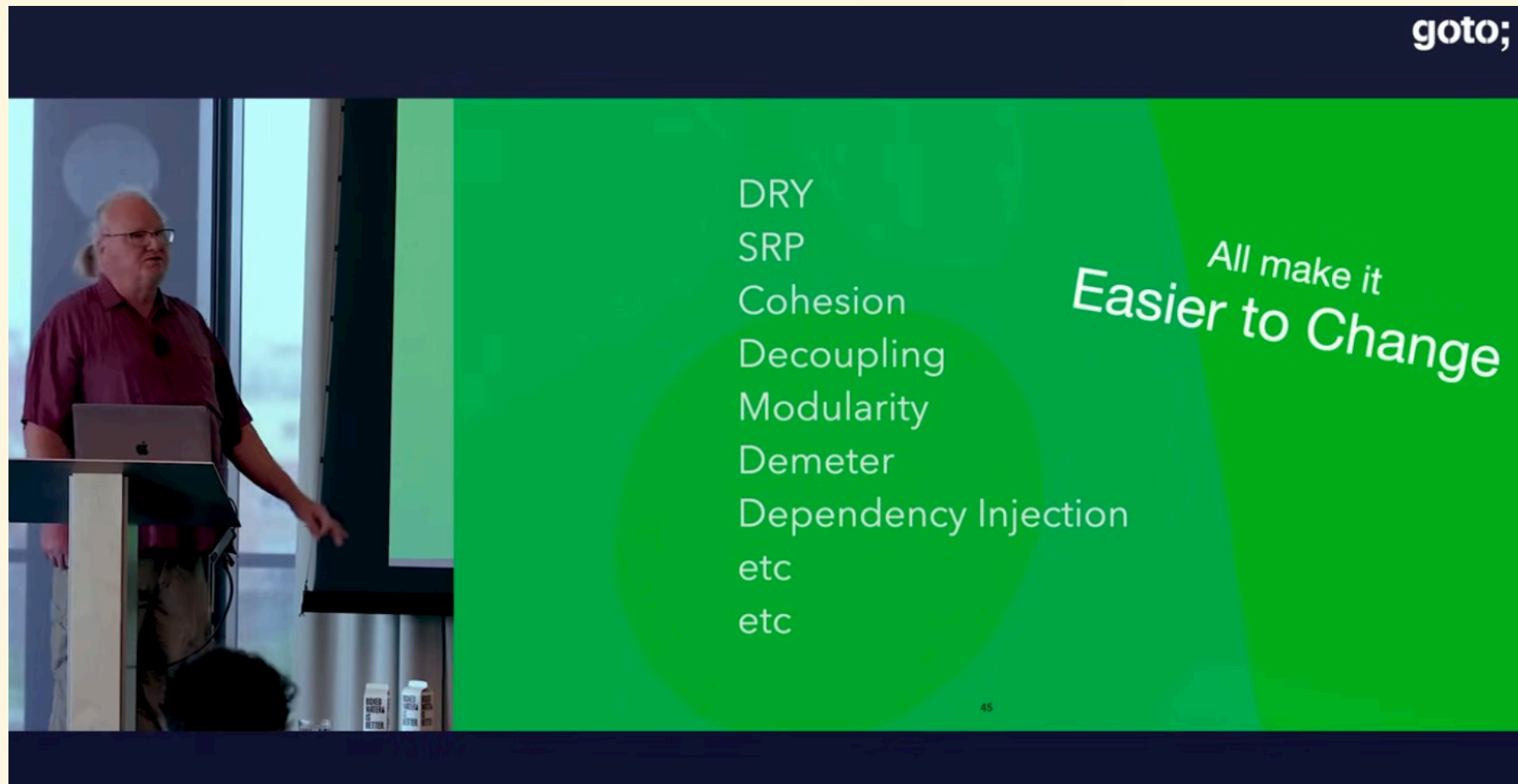


シンプル（複雑性の制御）を実現する方法はさまざま



デイヴ・トーマス 「覚えられないよ！」

すべては「変更しやすくする」ためだと思えばいい



「無意識」を活用して変更しやすさを感じ取れ！

そのためには「経験」を積み重ねることが重要



自分で未来を決めすぎない（制御不可能性）

- 老子の器："粘土をこねて器を作る。そこに空洞があるから、器としての役目を果たす"
 - **有之以為利 無之以為用**
 - 有が利となり、無が用となる（≡ 利用）
 - つまり "一見無駄に思える空間があつてこそ、有用なものが生まれる"

拡大解釈すれば「まだ決まっていない（変化する可能性もある）ものを受け入れる」ってコト！？



ソフトウェアで大切なこと（再び）

まだ決まっていない（変化する可能性がある）ものを受け入れる

1. コードを書くこと：当たり前（！）
2. テストすること：コードが完成したかどうかを教えてくれる
3. 設計すること：コードとテストだけではいずれ行き詰まる
4. 話を聞くこと：そもそも何のためのコードを書くのか？

ソフトウェアで大切なことをXPの価値に置き換える

まだ決まっていない（変化する可能性がある）ものを受け入れる

1. **【勇気（積極性）】**：当たり前（！）
2. **【フィードバック】**：コードが完成したかどうかを教えてくれる
3. **【シンプルシティ】**：コードとテストだけではいずれ行き詰まる
4. **【コミュニケーション】**：そもそも何のためのコードを書くのか？

ソフトウェアで大切なことをXPの価値に置き換える

まだ決まっていない（変化する可能性がある）ものを受け入れる

1. 【勇気（積極性）】：当たり前（！）
2. 【フィードバック】：コードが完成したかどうかを教えてくれる
3. 【シンプルシティ】：コードとテストだけではいずれ行き詰まる
4. 【コミュニケーション】：そもそも何のためのコードを書くのか？

“
これがソフトウェアのすべて。違うことを言う人は、何かを売りつけようとする人。
”

ソフトウェアで大切なことをXPの価値に置き換える

まだ決まっていない（変化する可能性がある）ものを受け入れる

1. **【勇気（積極性）】**：当たり前（！）
2. **【フィードバック】**：コードが完成したかどうかを教えてくれる
3. **【シンプルシティ】**：コードとテストだけではいずれ行き詰まる
4. **【コミュニケーション】**：そもそも何のためのコードを書くのか？

“

【リスペクト】

これがソフトウェアのすべて。違うことを言う人は、何かを売りつけようとする人。

”

第2版でリスペクトが追加された



リスペクト（中動態）

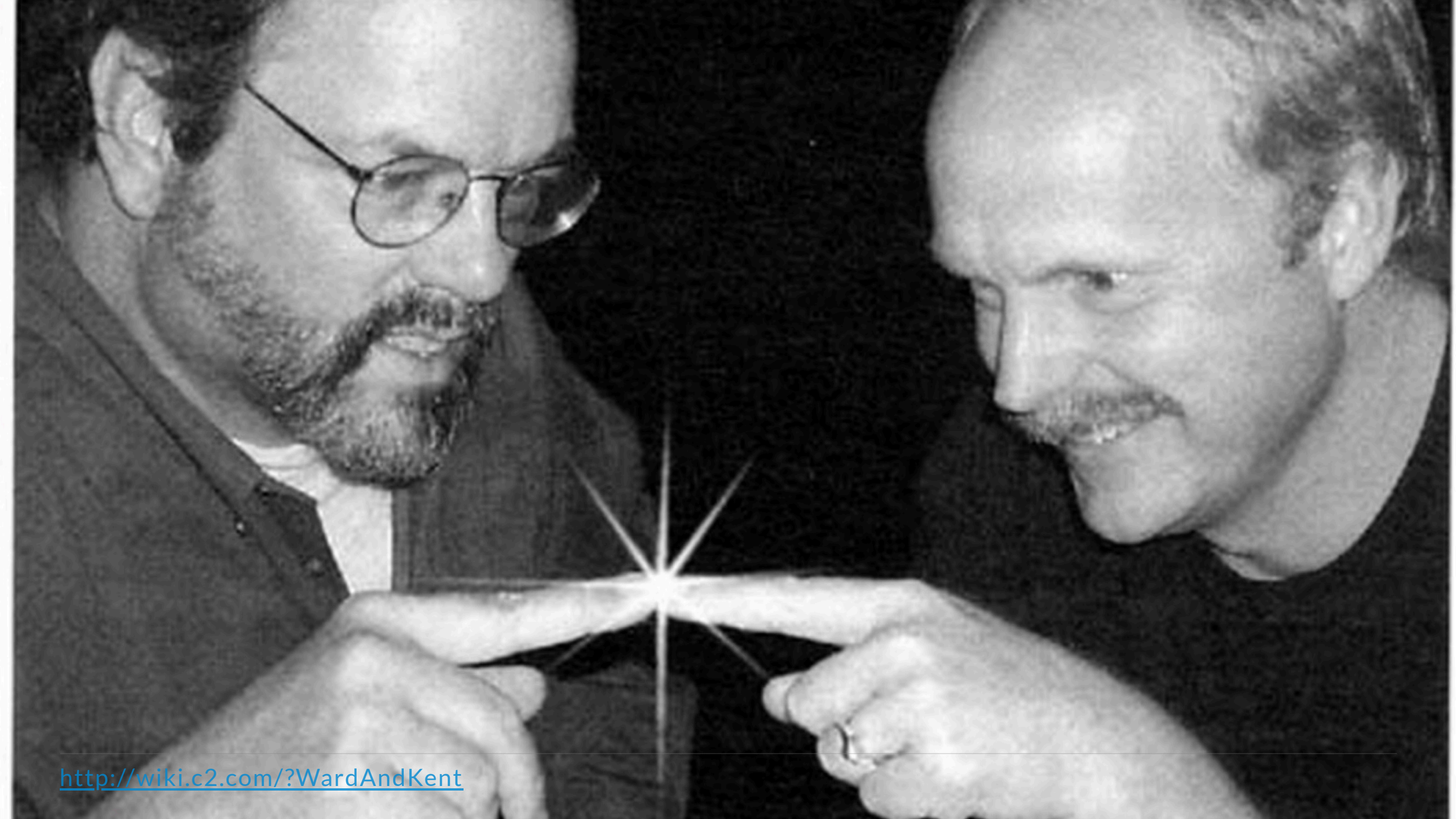
第49回 トロント国際映画祭 観客賞受賞

サンキュー、チャック

5.1 fri

映画『サンキュー、チャック』 <https://www.youtube.com/watch?v=ABYLtRT62v4>





高橋征義さん (@takahashim)



あなたの身近に リスペクトできるひとはいますか？

ビジネスリスペクトじゃなくて！

1 XPにはたくさんものがある

We, Programmers

我ら、 プログラマー

AdaからAIまで、コーダーたちの年代記

Robert C. Martin 著
角征典 訳

プログラミング
その始まりの
物語

 ZEN大学出版会



『我ら、プログラマー』第2部 目次

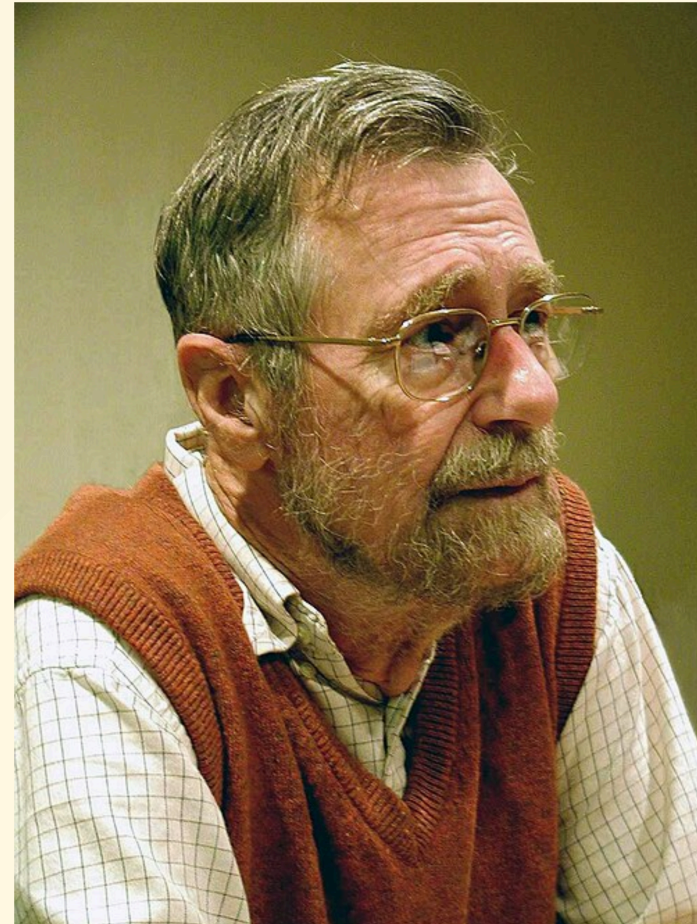
- 第2章 バベッジ：最初のコンピューターエンジニア
- 第3章 ヒルベルト、チューリング、フォン・ノイマン：最初のコンピューターアーキテクトたち
- 第4章 グレース・ホッパー：最初のソフトウェアエンジニア
- 第5章 ジョン・バックラス：最初の高水準言語
- 第6章 エドガー・ダイクストラ：最初の計算機科学者
- 第7章 ニゴールとダール：最初のOOPL
- 第8章 ジョン・ケメニー：最初の「みんなの」言語BASIC
- 第9章 ジュディス・アレン
- 第10章 トンプソン、リッチー、カーニハン

『我ら、プログラマー』第2部 目次

- 第2章 バベッジ：最初のコンピューターエンジニア
- 第3章 ヒルベルト、チューリング、フォン・ノイマン：最初のコンピューターアーキテクトたち
- **第4章 グレース・ホッパー：最初のソフトウェアエンジニア**
- 第5章 ジョン・バックラス：最初の高水準言語
- **第6章 エドガー・ダイクストラ：最初の計算機科学者**
- 第7章 ニゴールとダール：最初のOOPL
- 第8章 ジョン・ケメニー：最初の「みんなの」言語BASIC
- 第9章 ジュディス・アレン
- 第10章 トンプソン、リッチー、カーニハン



Grace Brewster Hopper
1906-1992



Edsger Wybe Dijkstra
1930-2002

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

https://en.wikipedia.org/wiki/Edsger_W._Dijkstra

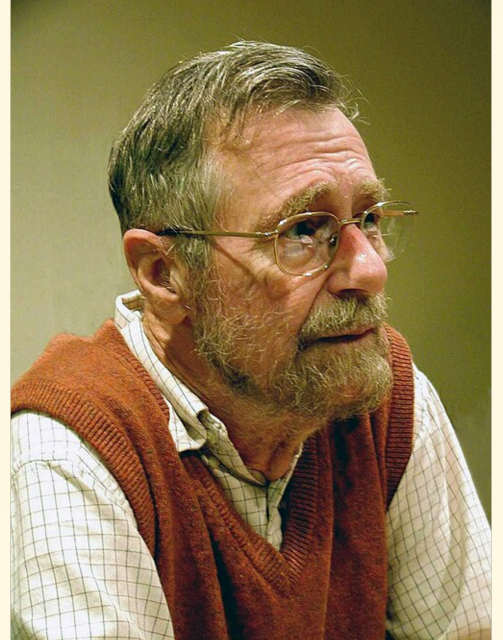
グレース・ホッパー (1906-1992)

- イェール大学で数学の博士号
- 36歳で大学教員を辞めて、海軍予備役に入る
- 海軍でHarvard Mark Iのプログラマーになる
- 最初期のコンパイラやCOBOLの開発に携わる
 - コンパイラで人間のミスを減らしたい
 - 「普通の人」が使える言語を目指す
- 人材教育や規律にも力を入れる
- コンピューター史上初の「物理的なバグ」の逸話も有名🦋



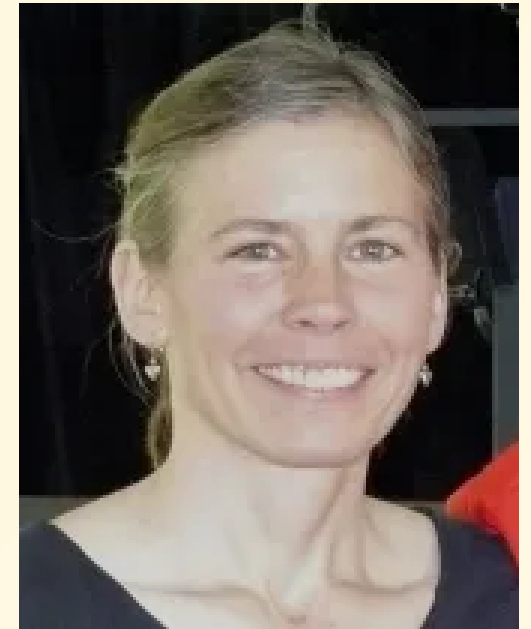
エドガー・ダイクストラ (1930-2002)

- 構造化プログラミングやダイクストラ法で有名
 - セマフォやALGOL 60のコンパイラなども
- 1952年にオランダ初のプログラマーになる
- 抽象化と数学的証明でプログラムの正しさを追求
 - 最初からバグを混入させるべきではない
 - 「普通の人」にプログラミングは難しい
 - COBOLは「冗長すぎる」「精神を蝕む」「誤った希望」
- ソフトウェアエンジニアに「コミュニケーション力」を求めるなど「文明に対する脅威」「最も浅はか」「滑稽でナンセンス」



2人の思想は「対立👊」している (Payette, 2014)

- 1 思想：実用主義 vs 形式主義
- 2 手法：デバッグ vs 証明
- 3 未来：規律 vs 学問



1 思想：実用主義 vs 形式主義

ホッパー

- 現場の仕事を進めることが重要
- 高度な人材の育成は難しい
- 普通の人が普通の方法で問題を解けるようにすべき

ダイクストラ

- プログラミングとは複雑性を排除する知的な挑戦
- COBOLのような人間に寄せた言語は精神をダメにする
- 最初から矛盾がないように論理的に構築すべき

2 手法：デバッグ vs 証明

ホッパー

- ソフトウェア開発には試行錯誤が欠かせない
- 使いやすいコンパイラやデバッガなどのサポートが必要
- もっとコンピューター「に」教育すべき

ダイクストラ

- ちゃんと教育を受けてないから試行錯誤することになる
- 数学的にプログラムの正しさを証明すれば、デバッグは必要ない
(実行して確かめる必要すらない)
- コンピューターを擬人化する人キモすぎ (職業的未熟さ)

3 未来：規律 vs 学問

ホッパー

- これからのプログラマー不足を解決したい
- プログラマーには厳格な規律が必要

ダイクストラ

- 新しい学問分野「コンピュータサイエンス」を目指す
 - 婚姻届の職業欄に「理論物理学者」と書いた苦々しい経験
- 形式論理と数学的証明に基づく開発手法が必要



2人は本当に「対立👊」しているのか？

- 2人とも「**計算機でうまく問題を解きたい**」のは同じ
- 2人とも「**複雑性を排除したい**」のは同じ
- 2人とも「**教育の必要性を認めているところ**」は同じ

一方、ケント・ベックは森と砂漠の比喻を使った

それは住んでいる環境によって考え方が違っているだけ



砂漠の住人を責めても森は生まれない



解決策：ギャングスター園芸家（ロン・フィンリー）



「小さなことから始める」だけでは不十分。セクシーに（格好良く）
やることが重要。「ギャングスター」の定義を置き換えよう。

XPにはセクシーさが足りない💧

- XPの名前の由来は「エクストリームスポーツ」。彼らのように危険なことをするには、**入念な準備**が必要だから。^[1]
 - ダンスの練習をしてないと、町中で踊ることはできない 🧑 🧚
 - あくまでも「弱き」人間のための**マジメな規律**だから
- さて、どうすればセクシーになるのか？
 - "「小さなことから始める」だけでは不十分"
 - XPerの定義を書き換えよう！

パラケルススの格言 by マーティン・ファウラー

“物事の多くは**単純な二元論**では語れない。

“「全てのものは毒であり、毒でないものなど存在しない。その**服用量こそが毒**であるか、**そうでないかを決めるのだ**」

“物事について「良い」か「悪い」かを考えるときには、「**どのような状況なのか**」と「**どれくらいの量なのか**」を考えるべきである。



XPerは「老子の器」になろう

一見無駄に思える空間があつてこそ、有用なものが生まれる^[1]

- 世の中には**相反する概念**があるが、**二元論で語るべきではない**

信頼 と 理想、加速 と 共鳴
挑戦 と 準備、個人 と 組織
規律 と 理論、社会 と 技術

- 片方だけに「エクストリーム」になるのは格好よくない
- 両方を含める器の大きさが、「エクストリーム」じゃないのか？

✧ XP Contains Multitudes

XPにはたくさんのものがある

理想を見失わず
相反するものを含み
共鳴しながら動き続けよう

XPの「形式」に集中するな！理想を見失うぞ！



eXtreme Programming

30 GREAT YEARS!

Thanks XP!

References

- Abrahamson, E. (1996). Management fashion. *Academy of Management Review*, 21(1), 254–285.
- Beck, K. (1998). Extreme programming: A humanistic discipline of software development. *International Conference on Fundamental Approaches to Software Engineering*, 1–6.
- Benders, J., & Van Veen, K. (2001). What's in a fashion? Interpretative viability and management fashions. *Organization*, 8(1), 33–53.
- Giroux, Hélène. (2006). “It was such a handy term”: Management fashions and pragmatic ambiguity. *Journal of Management Studies*, 43(6), 1227–1260.
- Madsen, D. Ø. (2020). The evolutionary trajectory of the agile concept viewed from a management fashion perspective. *Social Sciences*, 9(5), 69.
- Payette, S. (2014). Hopper and Dijkstra: Crisis, Revolution, and the Future of Programming. *IEEE Annals of the History of Computing*, 36(4), 64–73.
- Rosa, H. (2019). *Resonance: A sociology of our relationship to the world*. John Wiley & Sons.
- Rosa, H. (2023). Resonance as a medio-passive, emancipatory and transformative power: a reply to my critics. *The Journal of Chinese Sociology*, 10(1), 16.