2025
11 / 13

13:00 - 13:20

2002年のBeck × Cooper論争をヒントにする

巨匠たちのすれ違いから学ぶ、AI時代のアジャイルで重要なこと

株式会社SHIFT
アジャイル推進部 部長

秋葉 啓充　Hiromitsu Akiba

その常識、変えてみせる。SHIFT

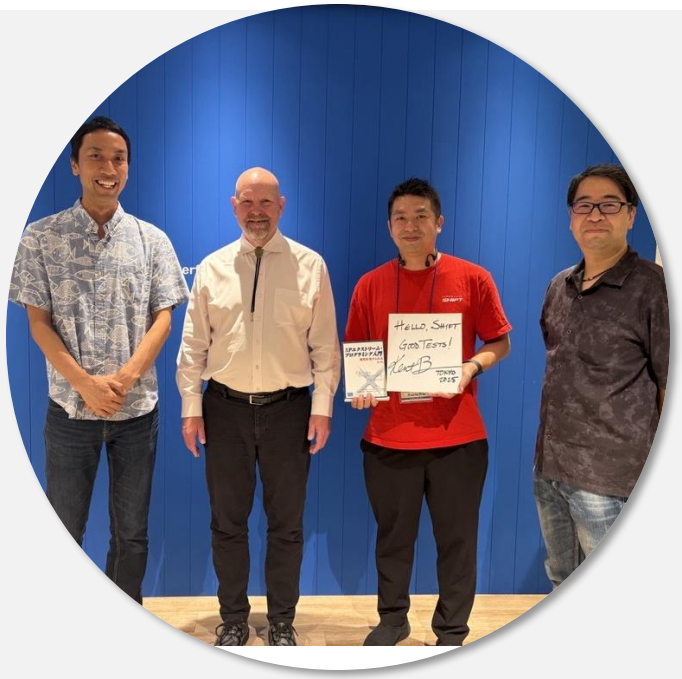
巨匠たちのすれ違いから学ぶ、AI時代のアジャイルで重要なこと

～2002年アラン・クーパー／ケント・ベック論争を参考に～

みなさんは、この先が読めない時、どうしていますか？ 私は、過去の出来事を学ぶことで、トレンドや過度の期待に流されず、冷静に判断できるようになると考えています。 2002年、ケント・ベックとアラン・クーパーが交わした論争は、「設計か、適応か」という問いでした。AI時代を生きる私たちに、その対話は“どう動くか”を考えるためのヒントになります。 アテンション・エコノミーといわれる派手さやスピードが重視される今こそ、熱狂や扇動の嵐に踊らされず、一人ひとりが自分の軸で考えることが求められています。 よりよい未来をつくるために、「考え抜く力」を取り戻す。そのきっかけになる時間にしたいと思います。

自己紹介

- 秋葉 啓充｜Akiba Hiromitsu



株式会社SHIFT
アジャイル推進部
部長

SHIFT歴6年目
2020年SHIFT入社後、自動化・開発・AIの技術系マネジメントを経験

大阪出身
趣味は📚読書、⚽フットサル
👨‍💻サンデープログラマー

心理的安全性

GitHub Trends # クラウド海峡横断部

RPA # ローコード # AI # 自動化 # CI/CD # 哲学

ポストモダン # 2男児の父 # 人間は9タイプ

フットサル # IoT

Kent Beck # Alan Cooper

今日はそんなケント・ベックについての話をしようと思います！

ケント・ベック

テストコードの産みの親

ケント・ベック

（Kent Beck）

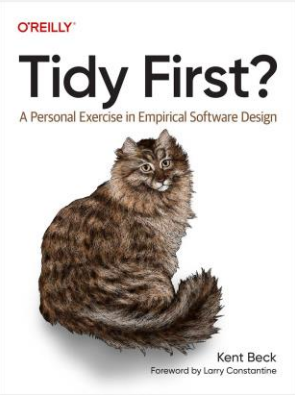
ケント・ベック氏は、オレゴン大学でコンピュータサイエンスの修士号を取得し、ソフトウェア開発の分野で多大な貢献をしてきました。彼は、**エクストリームプログラミング（XP）の考案者**であり、**アジャイルマニフェストの起草者**の一人としても知られています。また、ウォード・カニンガムとともにCRCカードを普及させ、Smalltalkのユニットテストフレームワークである**SUnitを開発**しました。さらに、エーリヒ・ガンマと共同でJavaのユニットテストフレームワークである**JUnitを開発**し、テスト駆動開発（TDD）の普及にも尽力しています。

『Smalltalk Best Practice Patterns』

『エクストリームプログラミング』

『テスト駆動開発』

『Tidy first?』



ケント・ベックは私の神様

- 私（秋葉）のキャリアスタートはJavaプログラマー
- ユーザーは仕様追加・仕様変更が多かった
- 時間がかかった部分のほとんどがテスト
- JUnitのおかげで再テストの時間が大幅に省略された
- だから、私にとってケント・ベックは神様

そんな神様がどうやら悩んでいる

～The value of 90% of my skills had **dropped to $0**.
At first, this realization was disheartening — I'd built my career on skills that were **being made obsolete by AI**. But upon reflection, I began to see this shift in value as an opportunity to recalibrate my skills and leverage the remaining 10% in a new way.～

私のスキルの90%の**価値は0ドルにまで下がって**しまったのです。
最初は、この認識に落胆しました。**AIによって時代遅れ**になりつつあるスキルを基盤にキャリアを築いてきたのです。しかし、よくよく考えてみると、この価値観の変化は、自分のスキルを再調整し、残りの10%を新たな方法で活用する機会だと捉えるようになりました。

出典：Web) https://tidyfirst.substack.com/p/90-of-my-skills-are-now-worth-0

彼が時代を先取りしていた頃を

あらためて確認してみよう

アジャイルの起源

ドット・コムバブル崩壊 (2000)

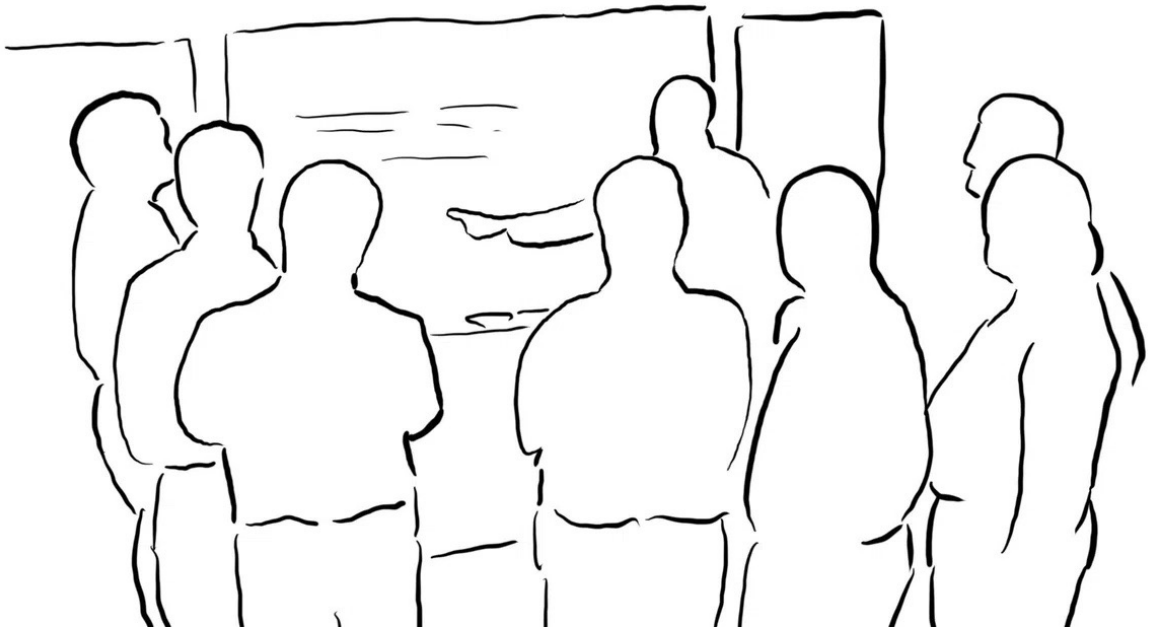
1990年代後半からインターネット関連企業の株価が急騰した後、2000年に暴落

- **株価の急騰**：ナスダック指数は1995年から2000年の間に約5倍に急騰
- **投機的な熱狂**：多くの投資家は、インターネットが持つ将来性への過度な期待から、実際の収益や経営状態を無視して投機的な投資を行った
- **.com企業**：投資の対象となったのは、社名に「.com」がつく多くのインターネット関連ベンチャー企業
- **ベンチャー投資**：ベンチャーキャピタルからの大量の資金が流入

アジャイルマニフェスト誕生！(2001)

ドットコムバブル崩壊後に
スノーバードに17人の有志があつまり、
アジャイルマニフェストを発表

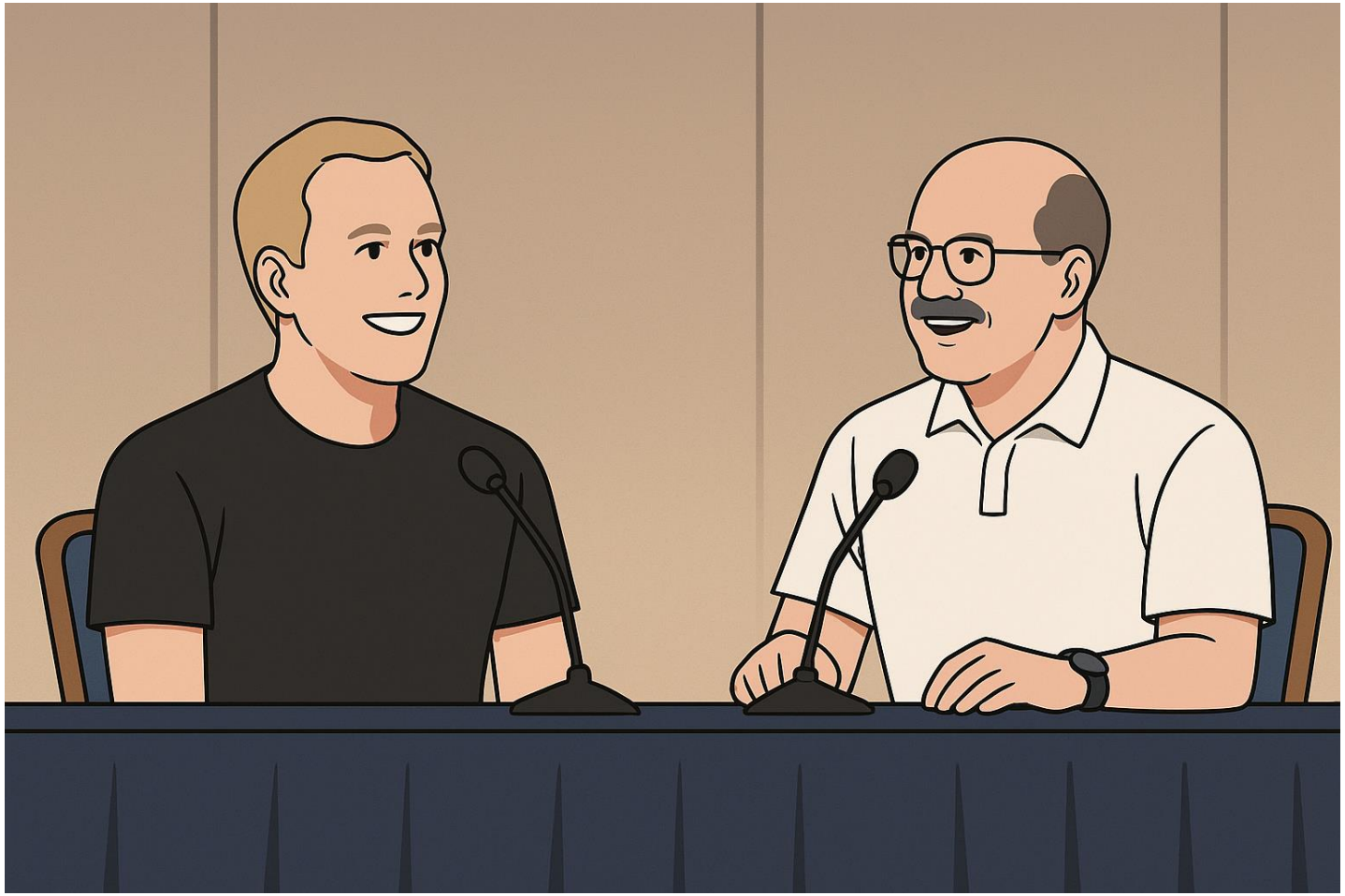
- 従来の「計画主導型」開発の限界露呈
- 「動くソフトウェア」への渇望
- リスクの低減と早期の価値創出



巨匠たちのすれ違い

開発カンファレンスに登壇 (2002)

Extreme Programming vs. Interaction Design



参考：https://web.archive.org/web/20061127000003/http://www.fawcette.com/interviews/beck_cooper/default.asp

ペルソナ概念の産みの親

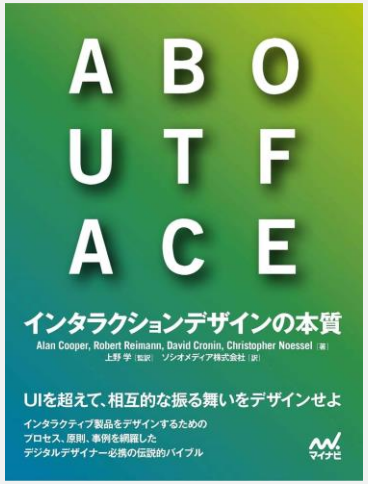
アラン・クーパー

（Alan Cooper）

1952年カリフォルニア州生まれ。1976年に自ら創業したStructured Systems Groupで、初期のビジネス向けマイクロコンピュータ用ソフトを開発し、その後1988年にはVisual Basicの原型を開発し、「**Visual Basicの父**」と称される。1992年に妻スーとともにCooperを設立し、Goal-Directed Designという手法を確立、ペルソナやシナリオ、ペアデザインなどのUX手法を普及させる。ソフトウェア開発とアジャイルとの統合にも注力し、業界に大きな影響を与えている。

『コンピューターは難しすぎる』

『About Face: The Essentials of Interaction Design』



コードはコンクリート

Alan Cooper

The instant you start coding, you set a trajectory that is substantially unchangeable.

If you can think this stuff through before you start pouring the concrete of code, you get significantly better results.

コードを書き始めた瞬間に、軌道はほとんど変えられなくなる。コードというコンクリートを流し込む前に考え抜くことで、はるかに良い結果が得られるのだ

If you’re building a skyscraper, you can’t move a wall.But in software, you can — if you build it right.

ビルを建てたら壁は動かせない。だがソフトウェアでは、正しく作れば壁を動かすことができる

Kent Beck

XPとは犬小屋しか作れない

Alan Cooper

Building a doghouse and building a skyscraper are different acts.XP works when you’re building doghouses.

犬小屋を建てるのと、ビルを建てるのは違う行為だ。XPは犬小屋を作るときにはうまくいく

Good design is not what you think before coding,
but what you learn while coding.

良い設計とは、コーディングの前に考えるものではなく、
コーディングの中で学ぶものだ

Kent Beck

ゴールは設定できるのか

Alan Cooper

Interaction designers define the behavior of the product.
Engineers build it.
The collaboration must start from a clear goal.

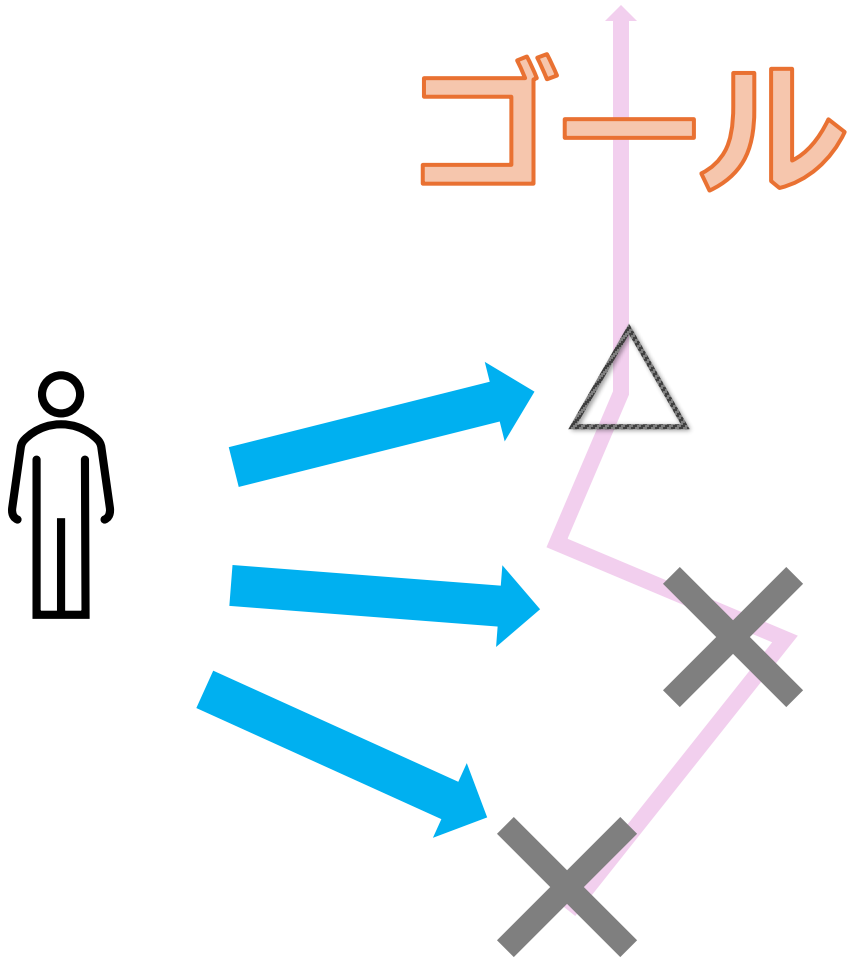
インタラクションデザイナーはプロダクトのふるまいを定義し、
エンジニアがそれを構築する。
協働は明確なゴールから始まらなければならない

You can’t plan the path of change;
you can only respond to it in small, reversible steps

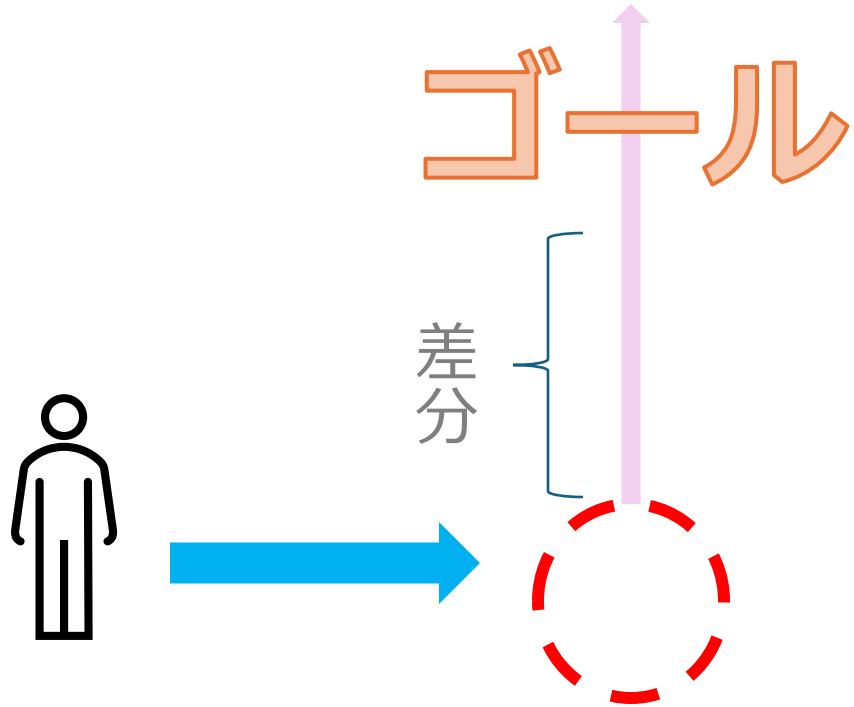
変化の経路を計画することはできない。
できるのは、小さく、可逆的なステップで応答することだけだ。

Kent Beck

ケント・ベックの主張
- 実装が先
- ゴールは予測できない
- 実践と対話を繰り返すことが重要



アラン・クーパーの主張
- 設計が先
- ゴールから考えないといけない
- 間違っていたとしても差分が明らかになる



すれ違いの理由

すれ違いの理由

一見、単なる「設計が先か実装が先か」にもかかわらず議論が深まらなかった理由は、同じ言葉を使いながらレイヤーが異なっていたため

用語	Alan Cooper	Kent Beck
User **ユーザー**	・想定上の他者 ・観察・構想の対象	・対話の相手 ・現場で共に学ぶ存在
Designer **デザイナー**	・創造者 ・観察者	・UI設計者
Engineer **エンジニア**	・実装者	・観察者 ・実装者

巨匠たちの考える根源的な問い

- 両者に共通するのは、バブル崩壊前のような**すべてを実装に先立って設計し、ユーザー行動を限定していくことへの懸念**であった
- **ユーザーは自らが欲しいものを自ら語ることができない**
- **ユーザー行動を創造していくため**のアプローチの違いがそれぞれの視点の違いであった

2002年の論争のあと、世界はどちらを選んだか？

アラン・クーパーは「正しい設計がすべての出発点」と語り、
ケント・ベックは「対話と変更こそが設計そのもの」と応じた。

→ そして、世界はアラン・クーパーではなく、
ケント・ベック的なアジャイルが主流となっていった

そして現代へ

AI時代の現代

- 2020年代に入り、生成AIはプログラミングの内部構造を大きく変えた。
- GitHub Copilot、ChatGPT、Claude、Cursor――等のAIエージェントで高速に仮説検証ができる土台が整いつつある
- ケント・ベックの対話的生成のスピードは、AIによって加速度的に成長していった

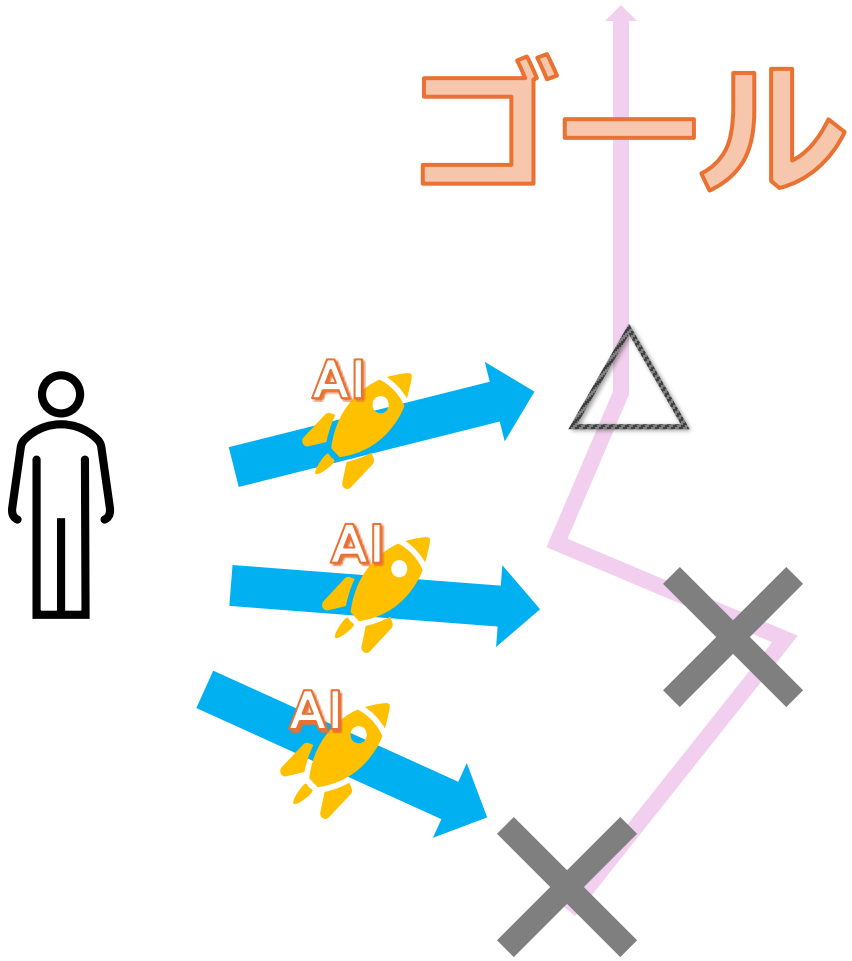
生成AI利用のレポートを見てみよう

DORAレポート

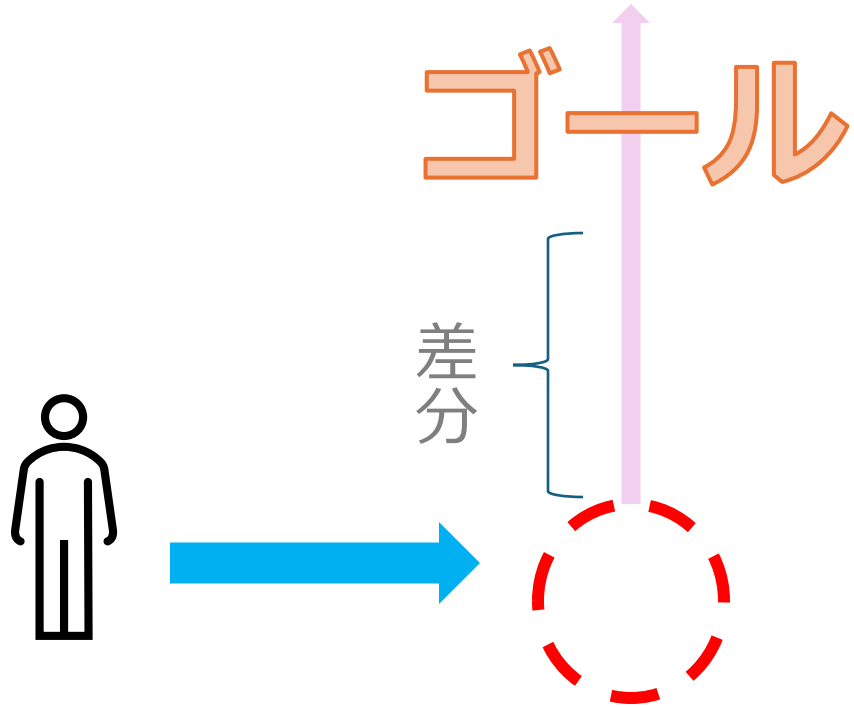
- Googleの調査「Impact of Generative AI in Software Development」(2024)によると
  - AI導入後、ドキュメント品質＋7%、レビュー速度＋3%、コード品質＋3%
  - だが、組織スループットは-1.5%、安定性は-7%
- つまり、AIは「開発の生産性」は高めたが、全体の生産性には直接的にはまだ寄与できていない

参考：https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report

ケント・ベックの主張
- 実装が先
- ゴールは予測できない
- 実践と対話を繰り返すことが重要



アラン・クーパーの主張
- 設計が先
- ゴールから考えないといけない
- 間違っていたとしても差分が明らかになる



まとめにかえて

まとめにかえて

- 現代は世界中がAIに夢中になっている
- AI時代の今も巨匠たちの「設計か実装か」の20年前の論争は終わっていない
- はたして、われわれはAI利用によって対話的生成のスピードがあがることに一喜一憂していて良いのでしょうか

ソフトウェアは誰のためのものか

When engineers design for themselves, they produce software that works for engineers, not for users.

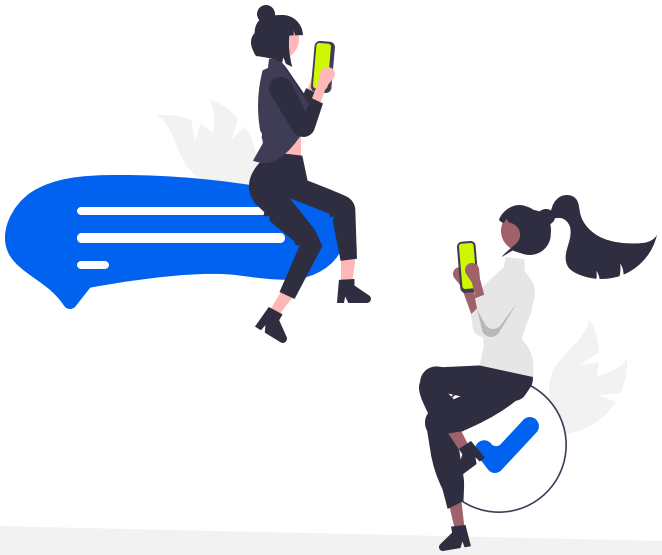
エンジニアが自分たちのために設計すると、それはユーザーではなく
エンジニアのためのソフトウェアになる

出典： Cooper, The Inmates Are Running the Asylum (SAMS Publishing, 1999), p. 16. （邦訳題：『コンピューターは難しすぎる』）

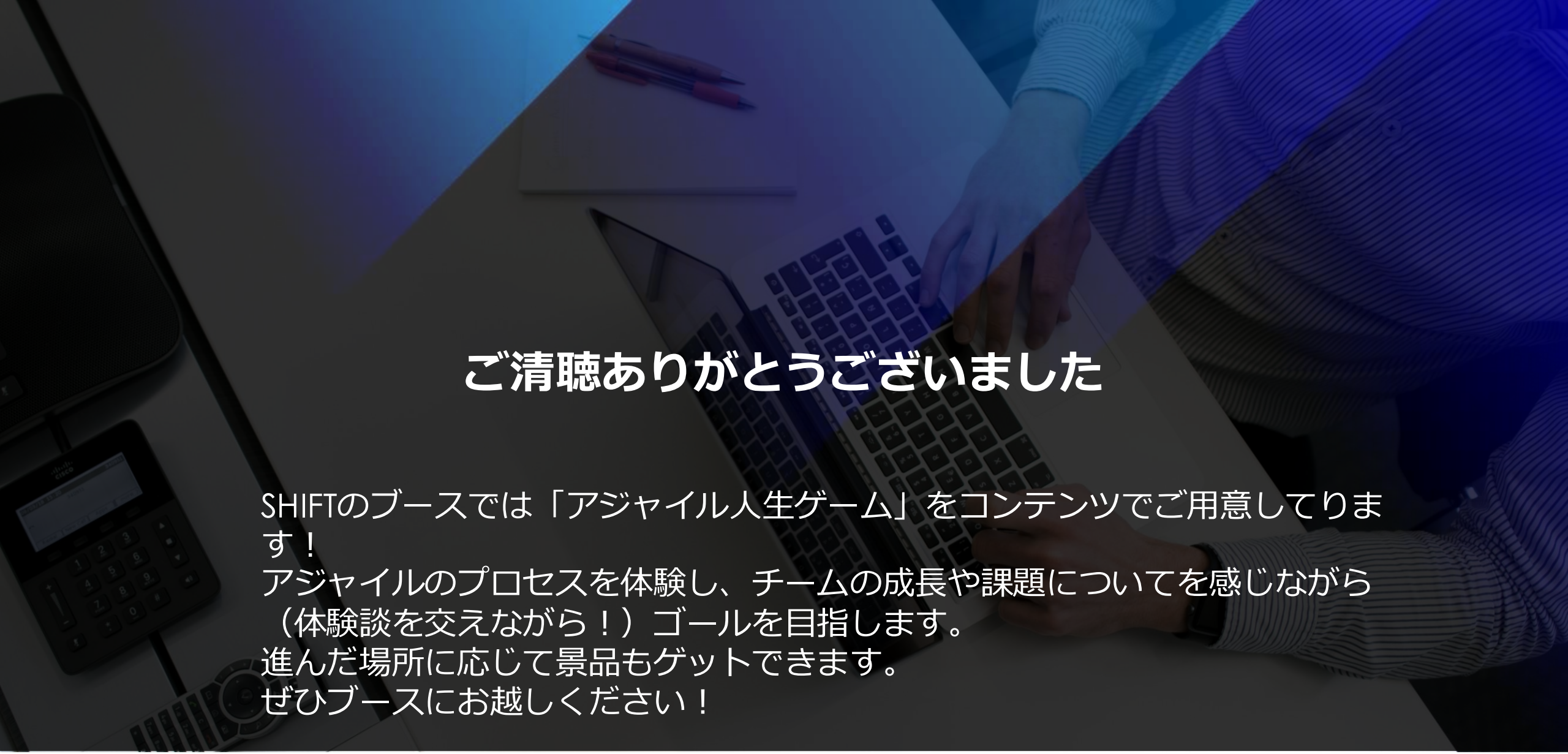
アンケートにご協力お願いします。

その常識、変えてみせる。
SHIFT

本セッションの感想をお聞かせください。



回答完了画面のご提示で、SHIFTブースにてノベルティをプレゼント！



ご清聴ありがとうございました

SHIFTのブースでは「アジャイル人生ゲーム」をコンテンツでご用意してります！
アジャイルのプロセスを体験し、チームの成長や課題についてを感じながら（体験談を交えながら！）ゴールを目指します。
進んだ場所に応じて景品もゲットできます。
ぜひブースにお越しください！

その常識、変えてみせる。 SHIFT