ATDDで素早く安定したデリバリを実現しよう！

導入の効果・躓きポイントから改善の工夫

KDDI Agile Development Center

自己紹介

KDDIアジャイル開発センター株式会社

高崎　和成

ソフトウェアエンジニア
グループ会社向け契約管理基盤開発
内製開発支援

KDDIアジャイル開発センター株式会社

政野　博紀

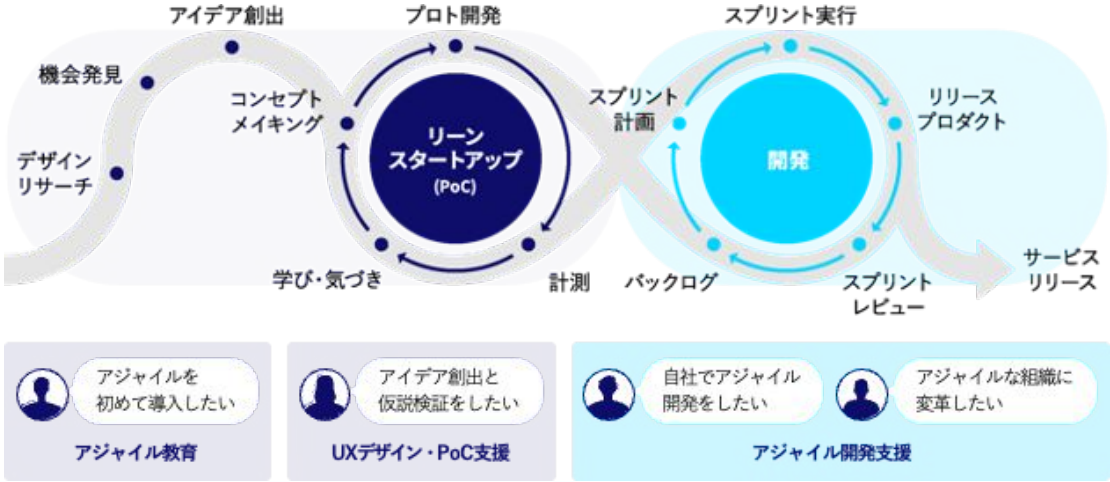
ソフトウェアエンジニア
ATDD導入期のチームに後発参画、その拡大に従事

KDDIアジャイル開発センターについて

KDDI Agile Development Center

約10年間“アジャイル開発“に徹底的にこだわり続けてきた DX専業のエンジニア集団

re-INNOVATE YOUR BUSINESS “Be Agile, Update Culture”



本日おはなしすること

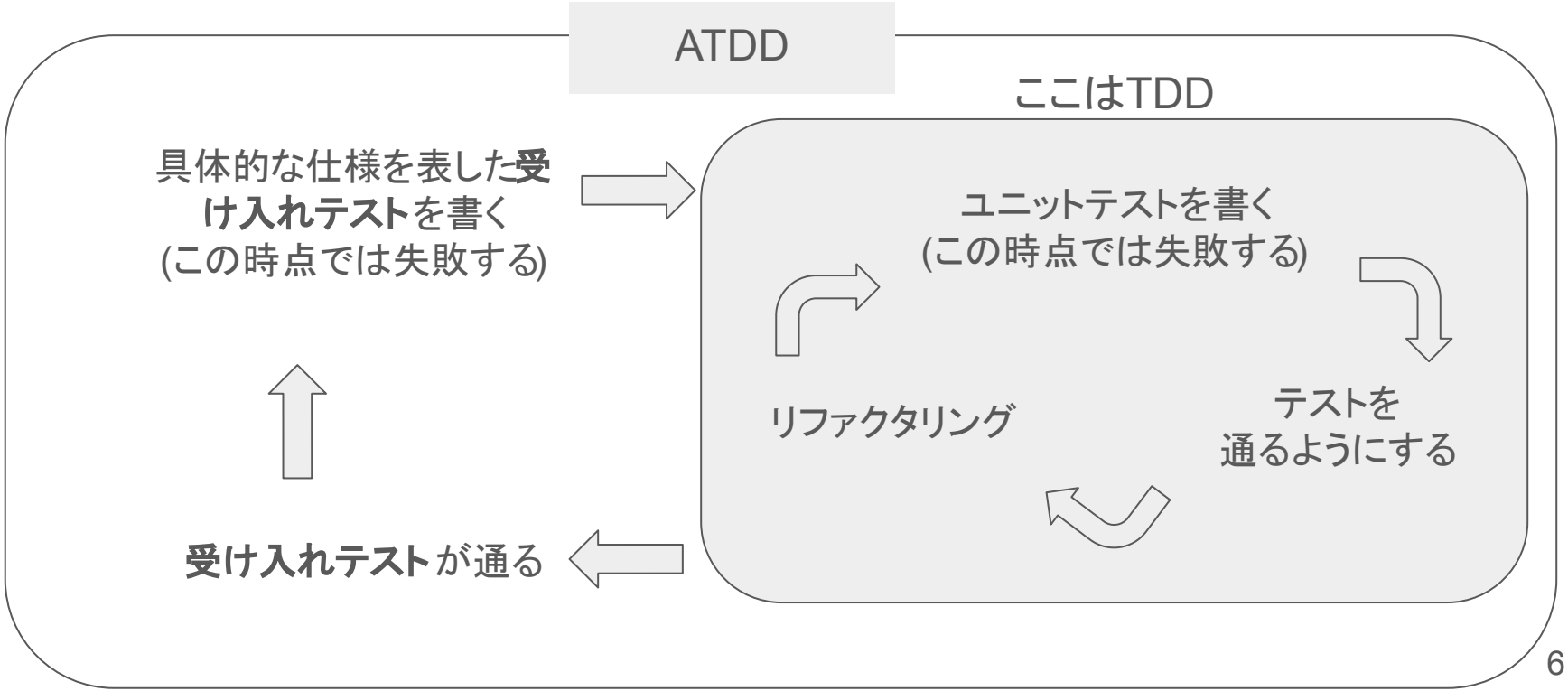
- ATDDの概要
- 導入してみた際の課題
- 感じた良し悪し
- 導入効果
- 今から導入する人に向けたアドバイス

私たちのチームの経験談でお話しします。
よりよい開発のヒントになれば幸いです。

受け入れテスト駆動開発 (ATDD)とは

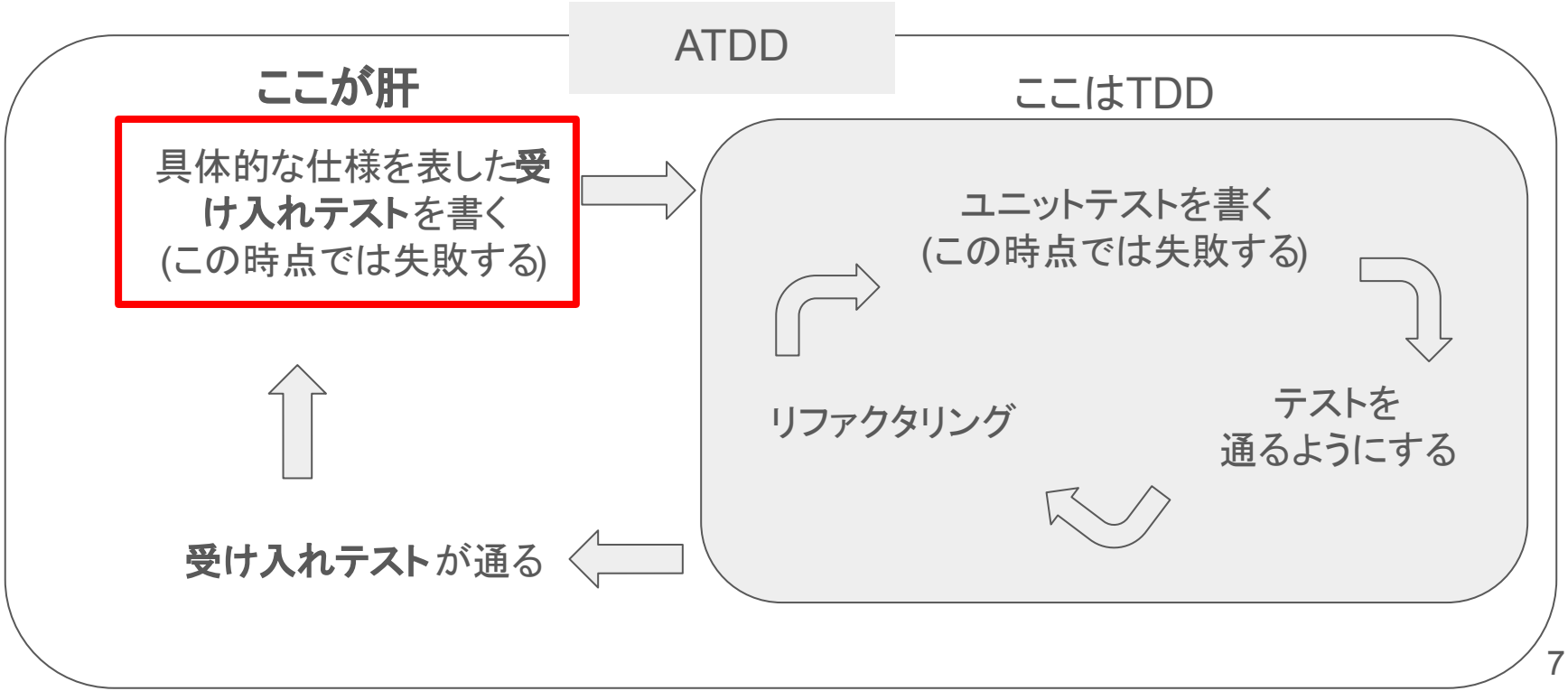
受け入れテスト駆動開発（ATDD)とは

開発着手前に受け入れテスト(具体的な仕様)を書き出す手法



受け入れテスト駆動開発（ATDD)とは

開発着手前に受け入れテスト(具体的な仕様)を書き出す手法



ATDD の 3 steps

```
Scenario: Eメールで検索できる
    Given 以下の契約が存在する
        | 会社名          | Eメール             |
        | テスト1株式会社 | sample1@example.com |
        | テスト2株式会社 | sample2@example.com |
    And 契約情報検索画面が表示されている
    When 検索画面に "sample1@example.com" を入力する
    And 検索ボタンを押下する
    Then 検索結果に以下の内容が表示されていること
        | 会社名          | Eメール             |
        | テスト1株式会社 | sample1@example.com |
```

```
When("検索画面に {string} を入力する", async (key: string) => {
  const property = getProperty("Eメール または 会社名");
  await page.locator(property.id).fill(key);
});
```

Step 1. 発見

具体的な仕様を話し合う

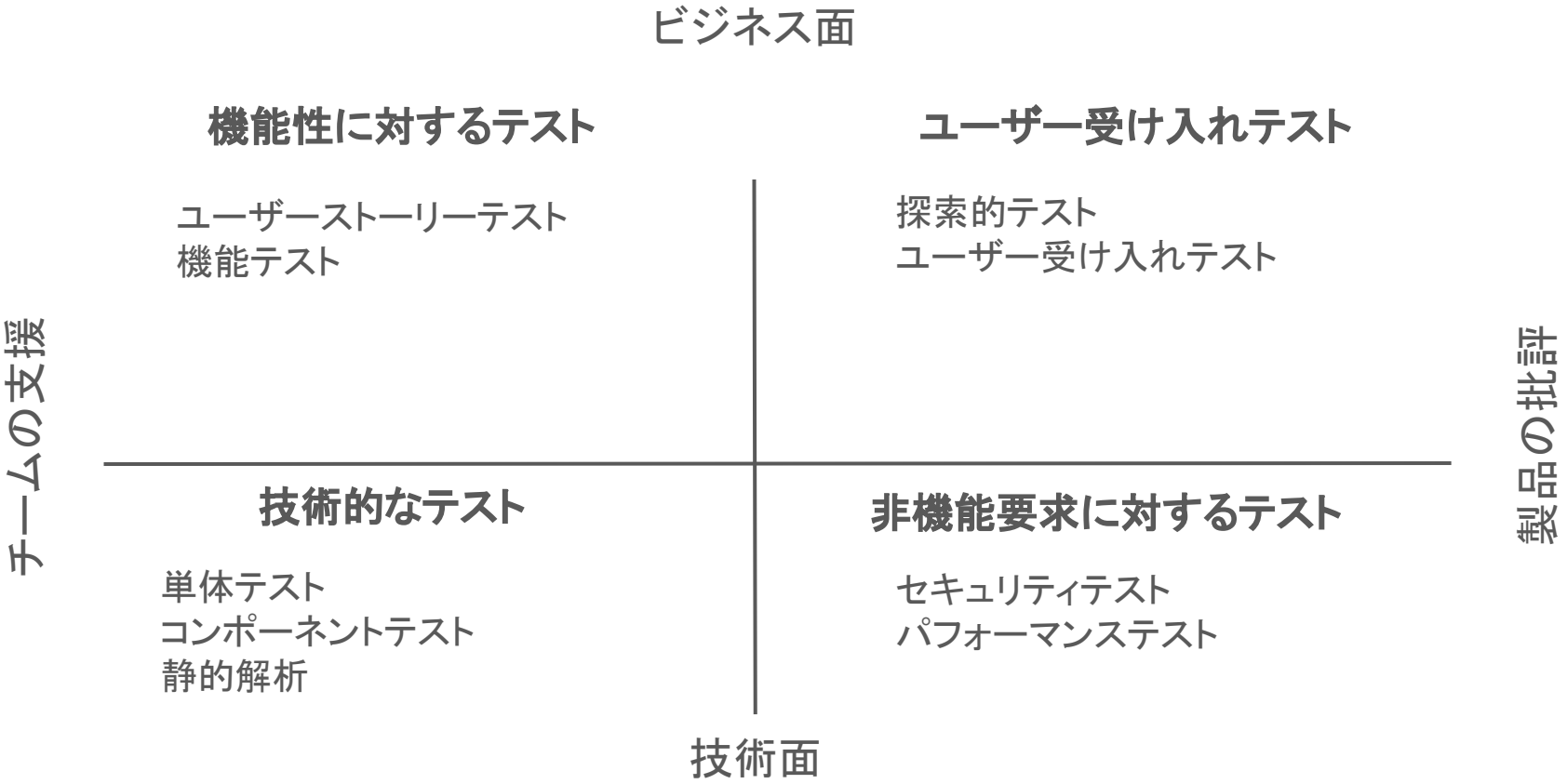
Step 2. 定式化

自然言語のテストとして書き出す

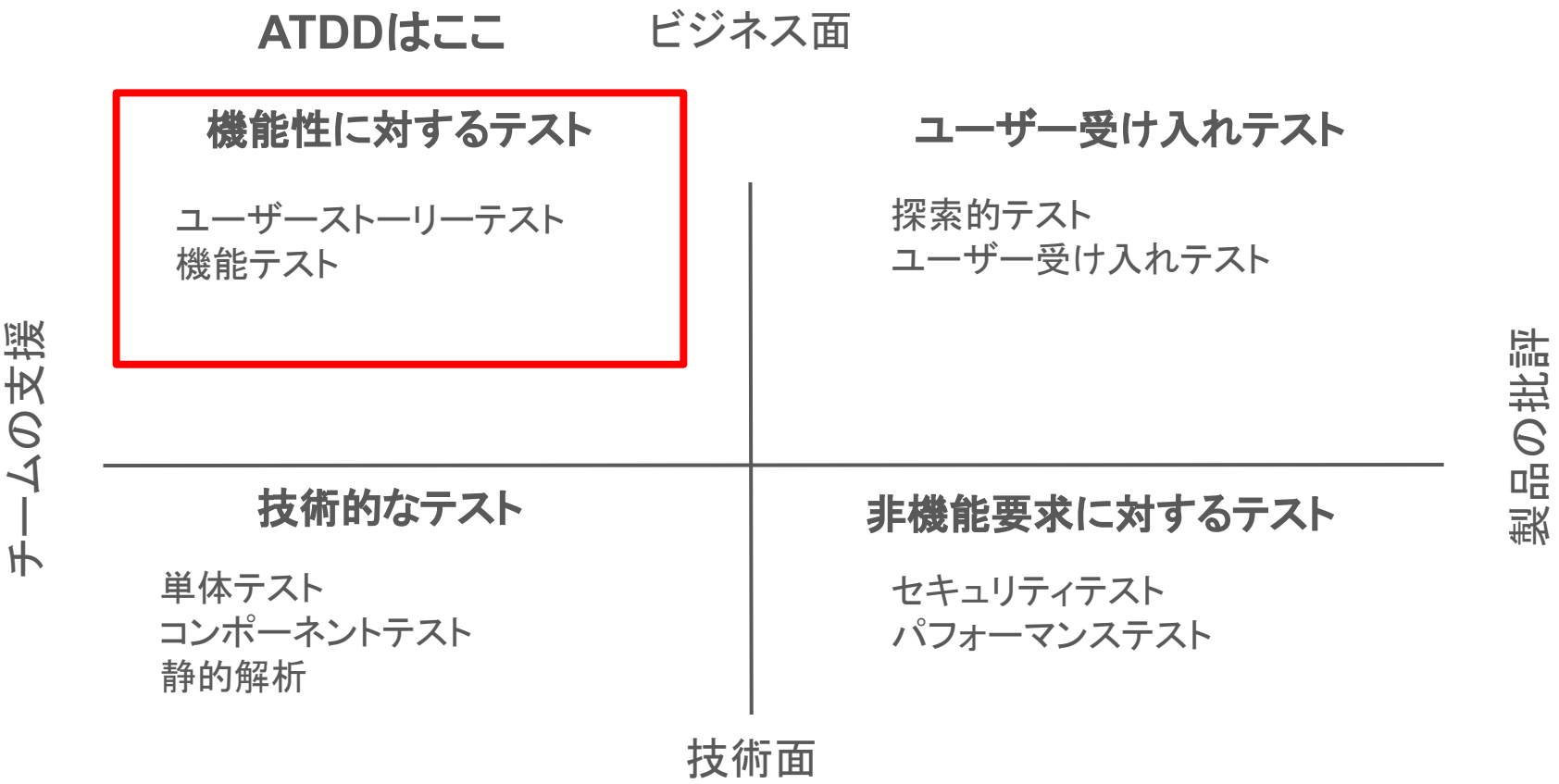
Step 3. 自動化

テストを自動化する

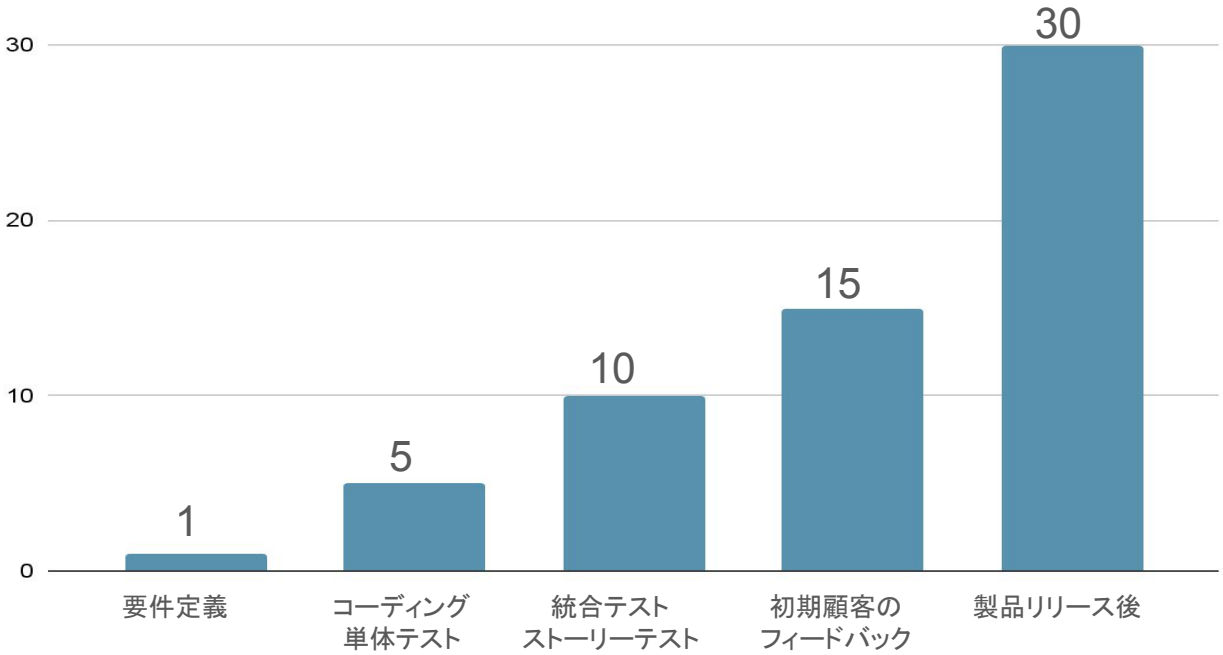
アジャイルテストの四象限



アジャイルテストの四象限



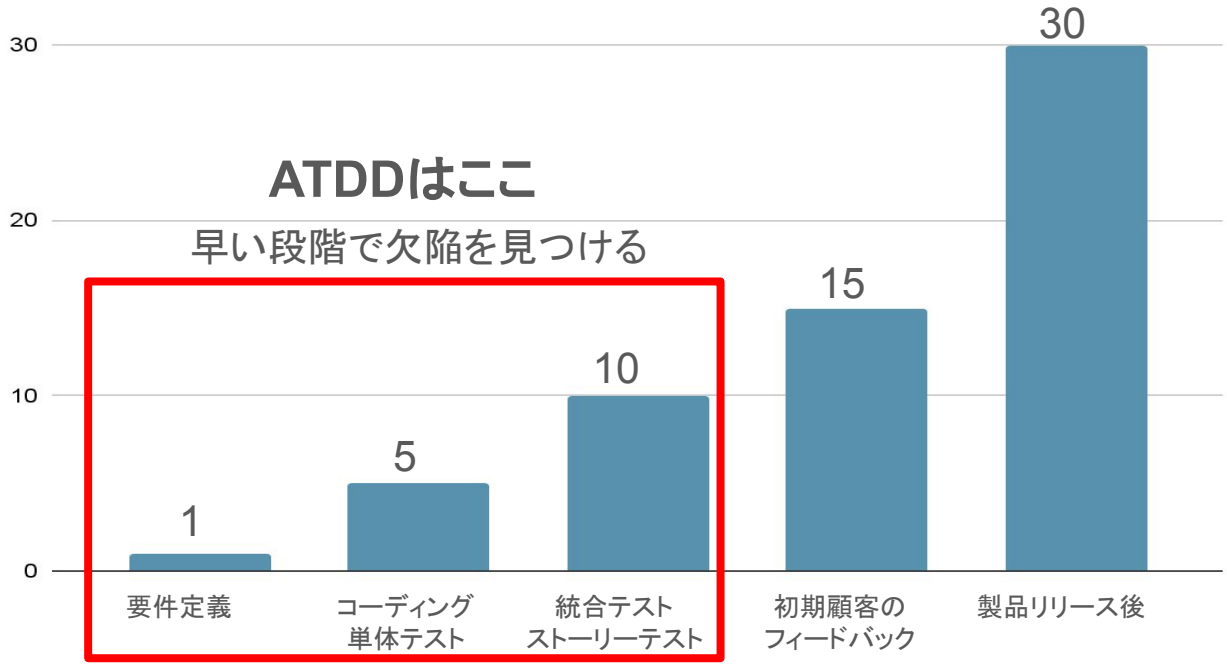
欠陥の発見が遅れるほど、修復コストは膨れる



ソフトウェア開発の各段階で発見された欠陥の修復にかかるコストの相対比

https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

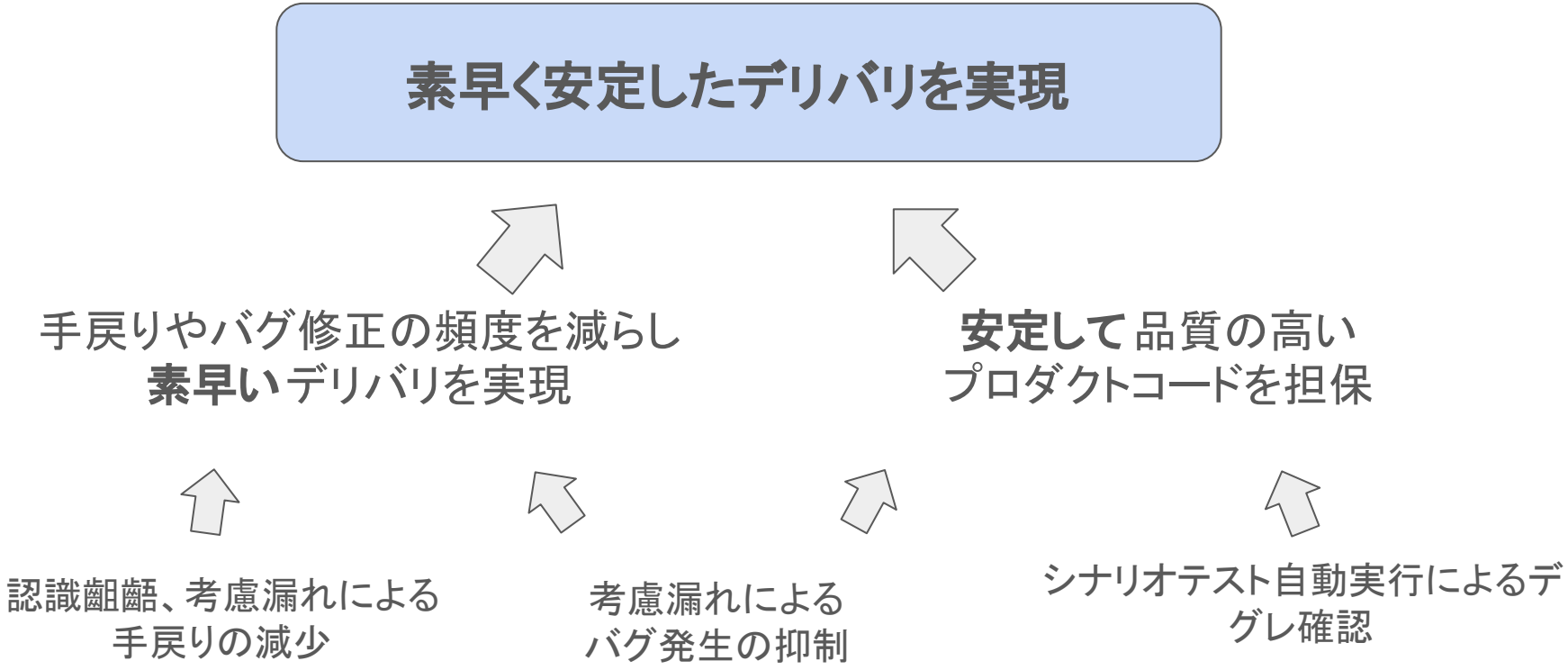
欠陥の発見が遅れるほど、修復コストは膨れる



ソフトウェア開発の各段階で発見された欠陥の修復にかかるコストの相対比

https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf

ATDD の恩恵


素早く安定したデリバリを実現
手戻りやバグ修正の頻度を減らし
素早いデリバリを実現
安定して品質の高い
プロダクトコードを担保
認識齟齬、考慮漏れによる
手戻りの減少
考慮漏れによる
バグ発生の抑制
シナリオテスト自動実行によるデグレ確認

ATDD導入の背景

社内で受けたアジャイルテスティング研修で　ATDDを知った

アジャイルテストとは

アジャイル開発における品質保証の取り組み

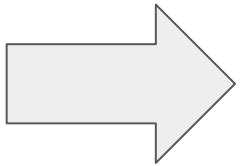
- アジャイルテストの四象限
- テストマニフェスト
- 探索的テスト
- **受け入れテスト駆動開発 (ATDD)**

https://scruminc.jp/training/agile-testing/

まずは試してみることにした

私達のチームでも手戻りは発生していた

ATDDに興味があった

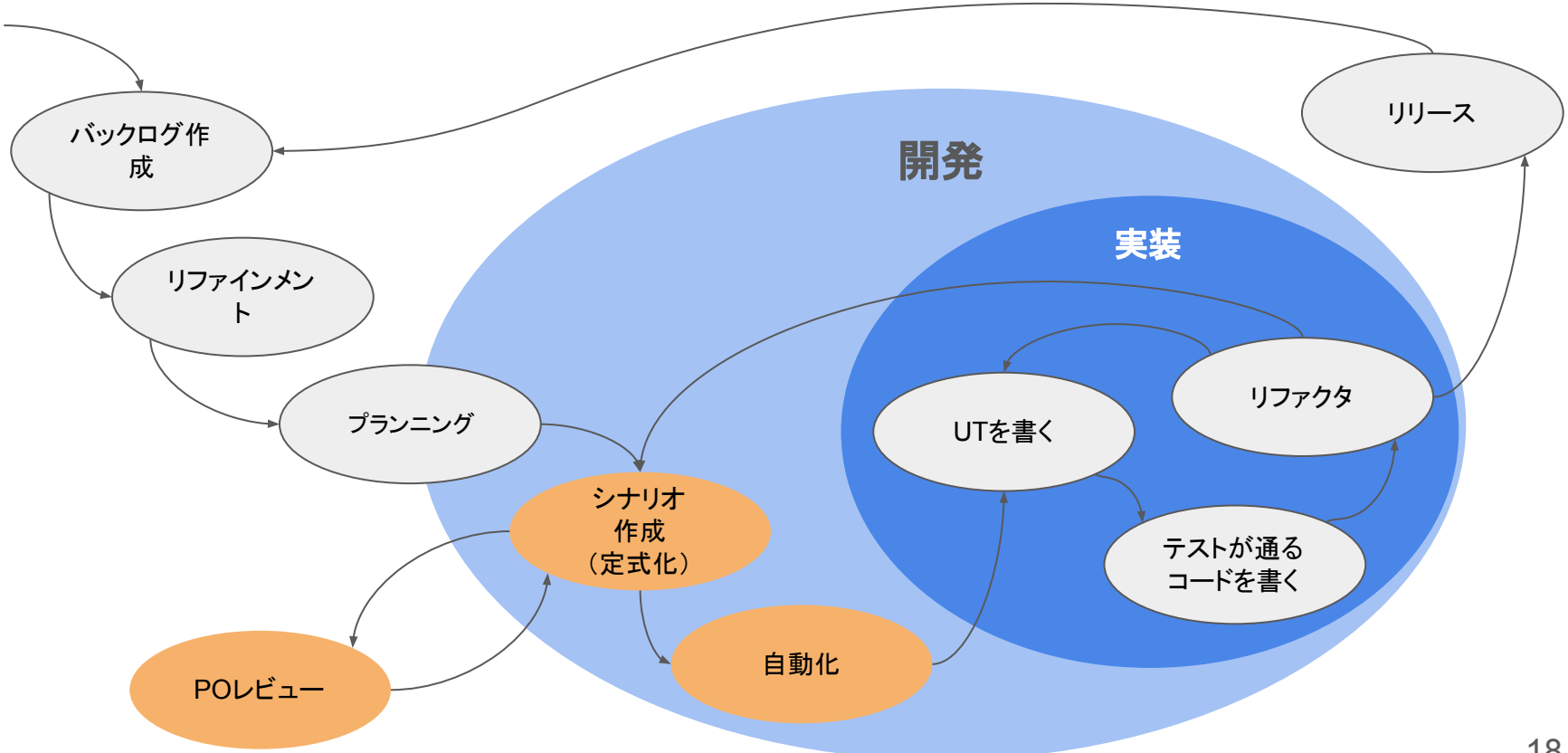


まずはやってみよう

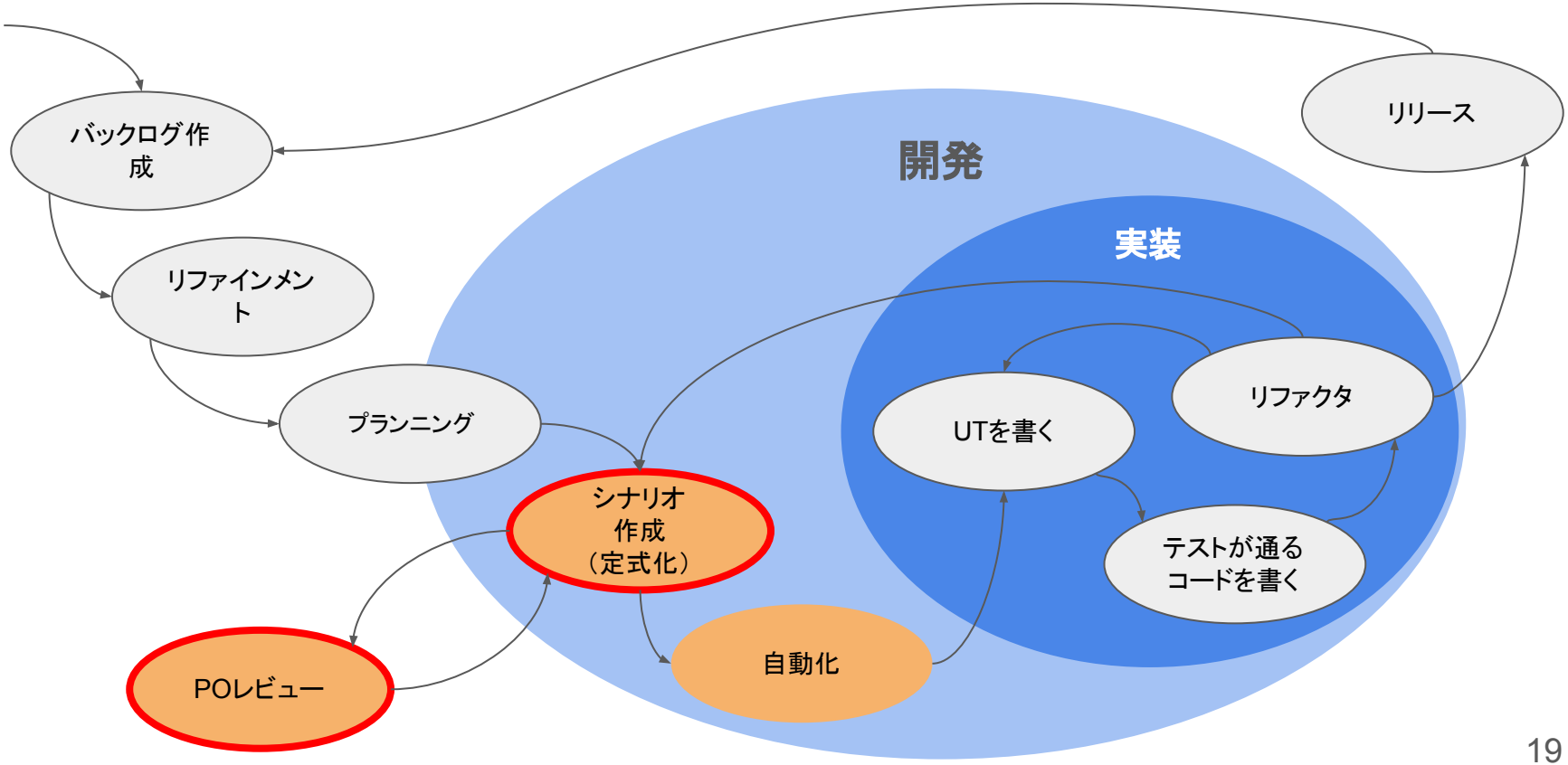
ATDDはじめてみました

KDDI Agile Development Center Corporation

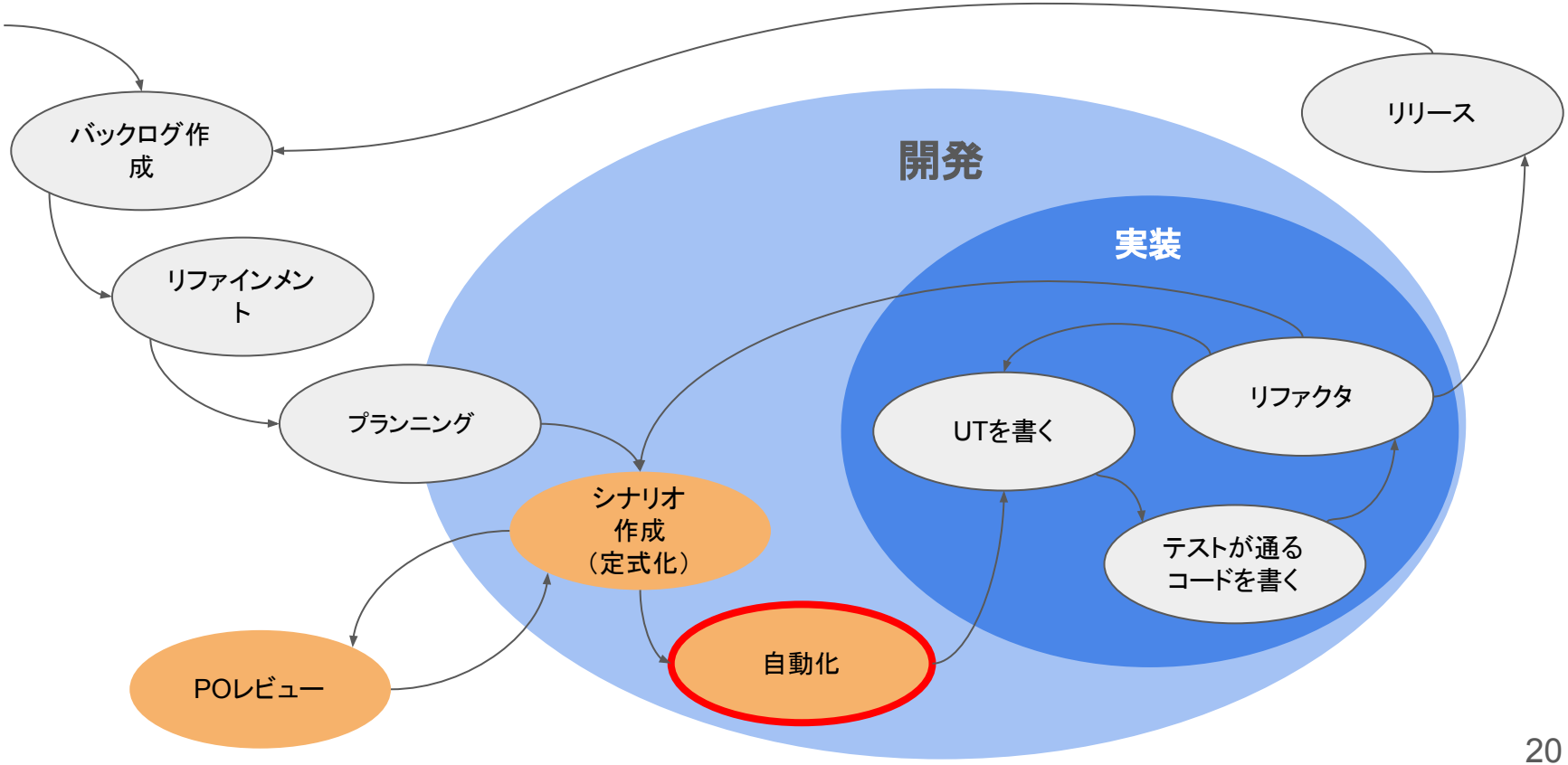
既存の開発サイクルをベースに　ATDDのプロセスを追加



既存の開発サイクルをベースに　ATDDのプロセスを追加



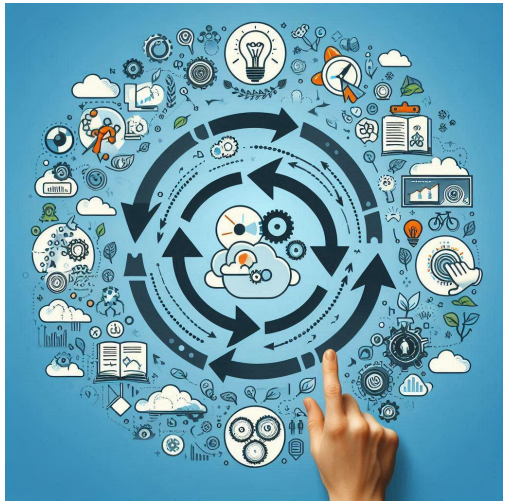
既存の開発サイクルをベースに　ATDDのプロセスを追加



実際に始めて見えてきた課題

課題1

導入と
メンテナンスが大変



課題2

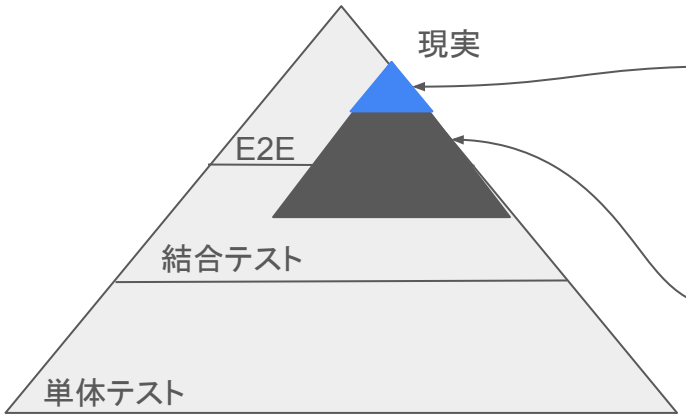
テスト⇔実装の
細かいサイクルを
回すのが大変

課題3

開発目線が先行し，ビ
ジネスフレンドリでな
かった

課題① ― 導入とメンテナンスが大変　～ATDD導入前の理想と現実～

ATDD導入前



理想的なテストピラミッド

スナップショットテスト
（自動）

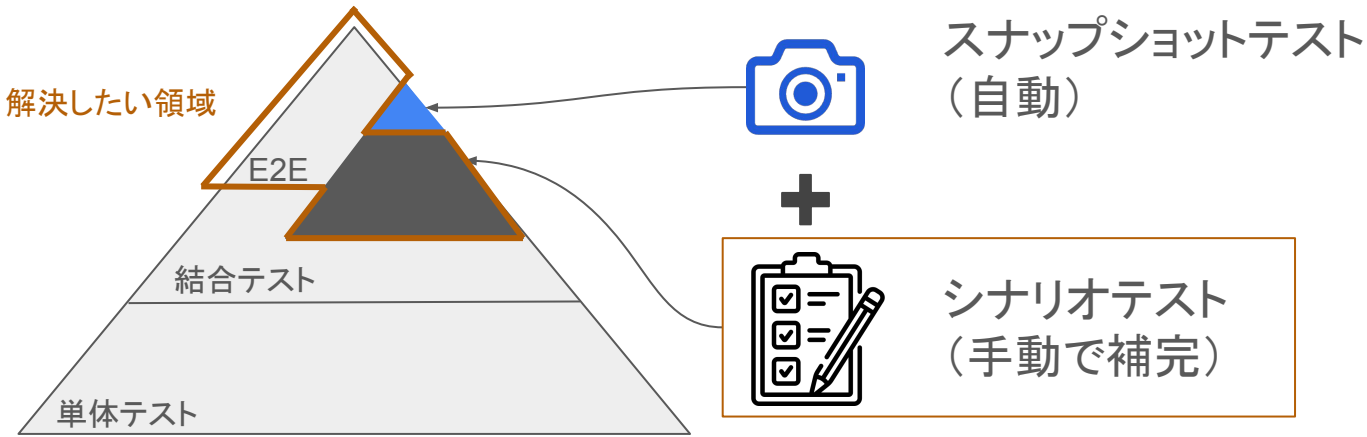
＋

シナリオテスト
（手動で補完）

毎週のリリースにあたり、手動テストの手間を要する

課題① ― 導入とメンテナンスが大変　～ATDD導入前の理想と現実～

ATDD導入前

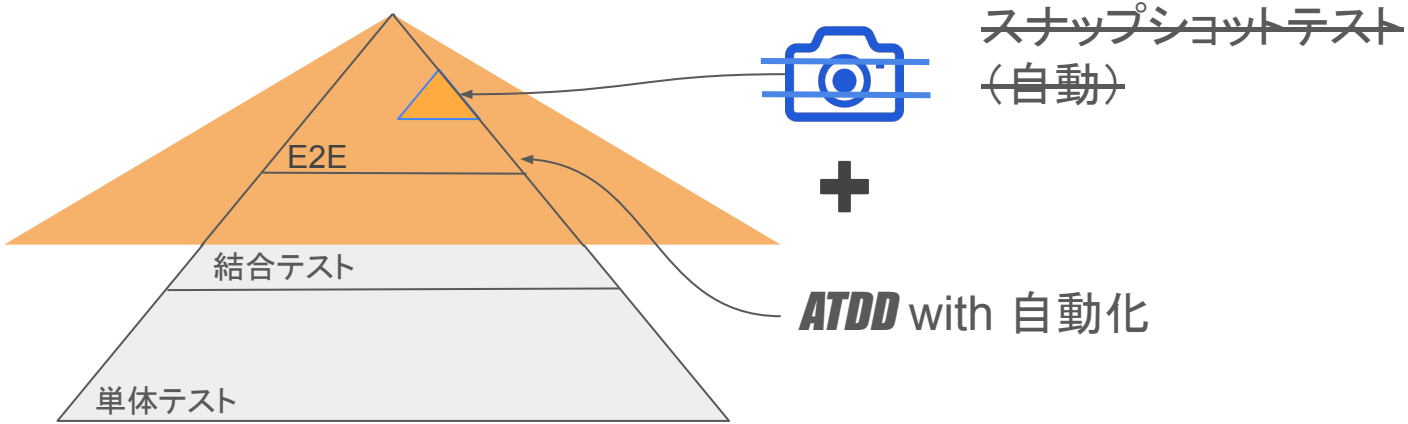


毎週のリリースにあたり、手動テストの手間を要する

→ ATDD導入するなら，完全自動化しつつ不足領域を補いたい！

課題① ― 導入とメンテナンスが大変　～ATDDによる自動化の光と影～

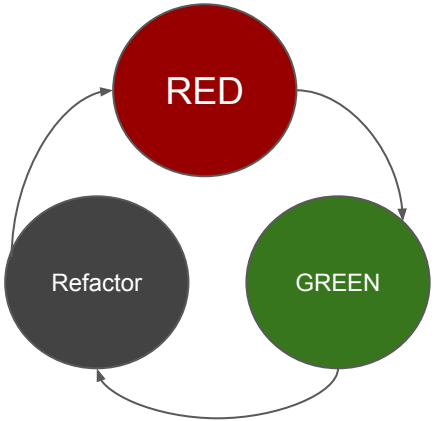
ATDD導入後



理想形とはギャップがある

- 実際やってみると、テストケースはたくさん作る傾向
- 開発者の安心感は増したが、割高感はあり
- 重複テストの見直しや、慣れによって回るように

課題② ― テスト⇔実装の細かいサイクルを回すのが大変



ATDDも小さくサイクルを回すのが大事

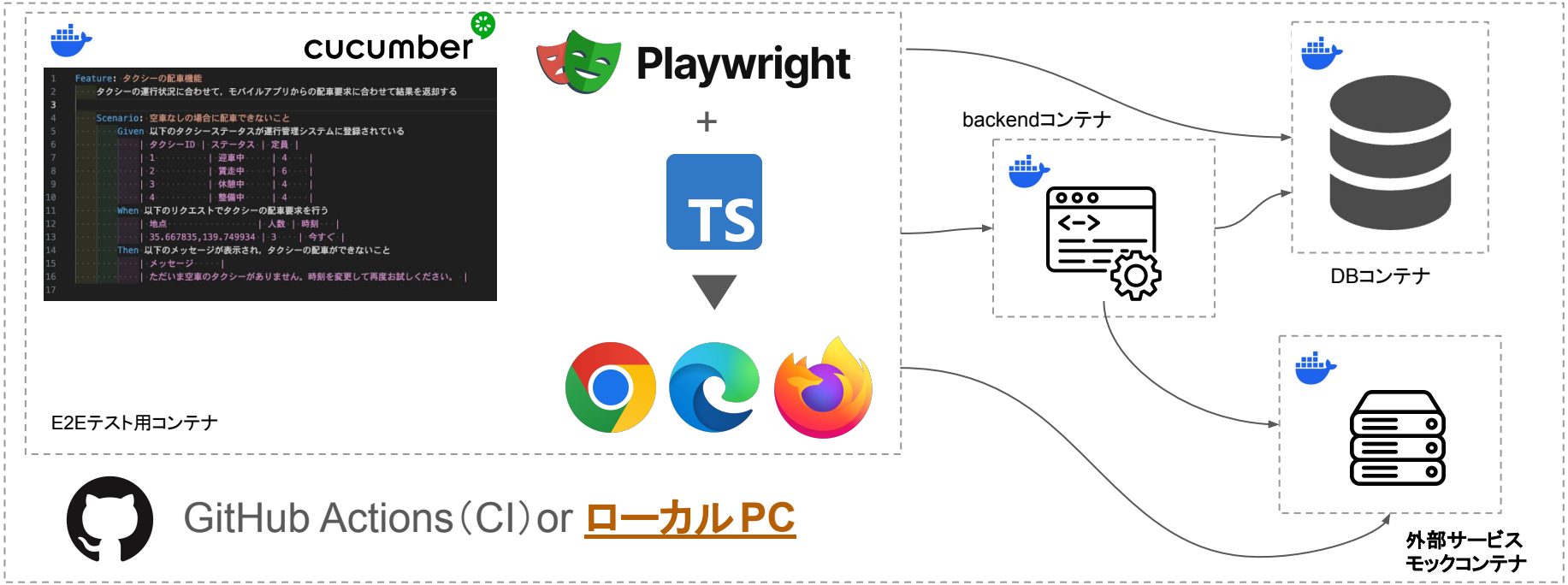
阻害要因

- 実行準備で色々立ち上げる必要（サーバ, API, DB, etc…）
- 外部システム連携があって複雑
- CI上で実行時間が長い

→ せめてローカル環境で気軽に試せるようにしよう！

課題② ― テスト⇔実装の細かいサイクルを回すのが大変　～楽なテストの工夫～

コンテナベースでのE2Eテスト環境の構築



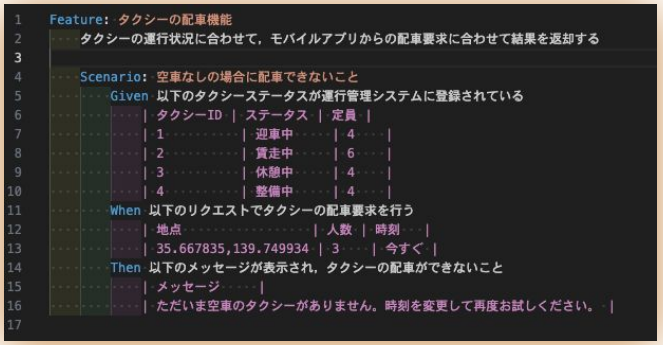
いつでも誰でも同時でも1コマンドで実行可能な状態がオススメ

課題③ — 開発目線が先行し，ビジネスフレンドリでなかった

表面的なシナリオファイル

テストデータファイル
for開発者

統一されたシナリオファイル



気づき

- シナリオにアクセスしやすい，見やすいことは大事
- 開発者のみで進めると，視点が偏りがち
- ビジネス側でF/Bを受けやすい状態にして、開発前に気づきを得られる

今だから感じる
ATDDの良い点・大変な点

ATDDの良い点

事前に認識を
合わせる癖がつく

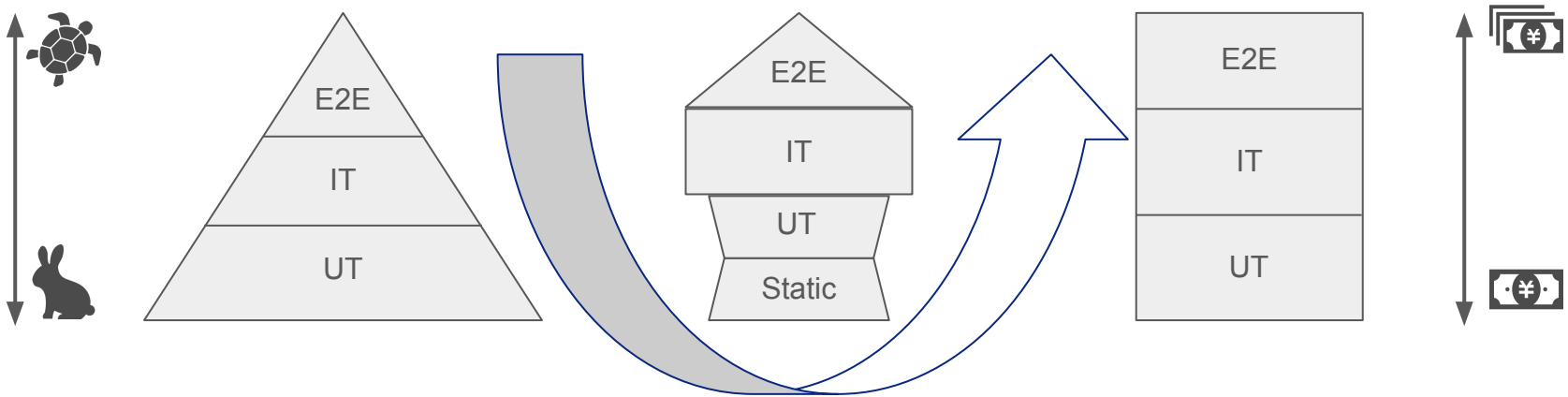
生きているドキュメントが出
来上がる

チームが共通言語で
話すことができる

一方で・・・

ATDDの大変な点 ―テストピラミッドと「さじ加減」

テストの理想形や落としどころ → 見極めが難しい



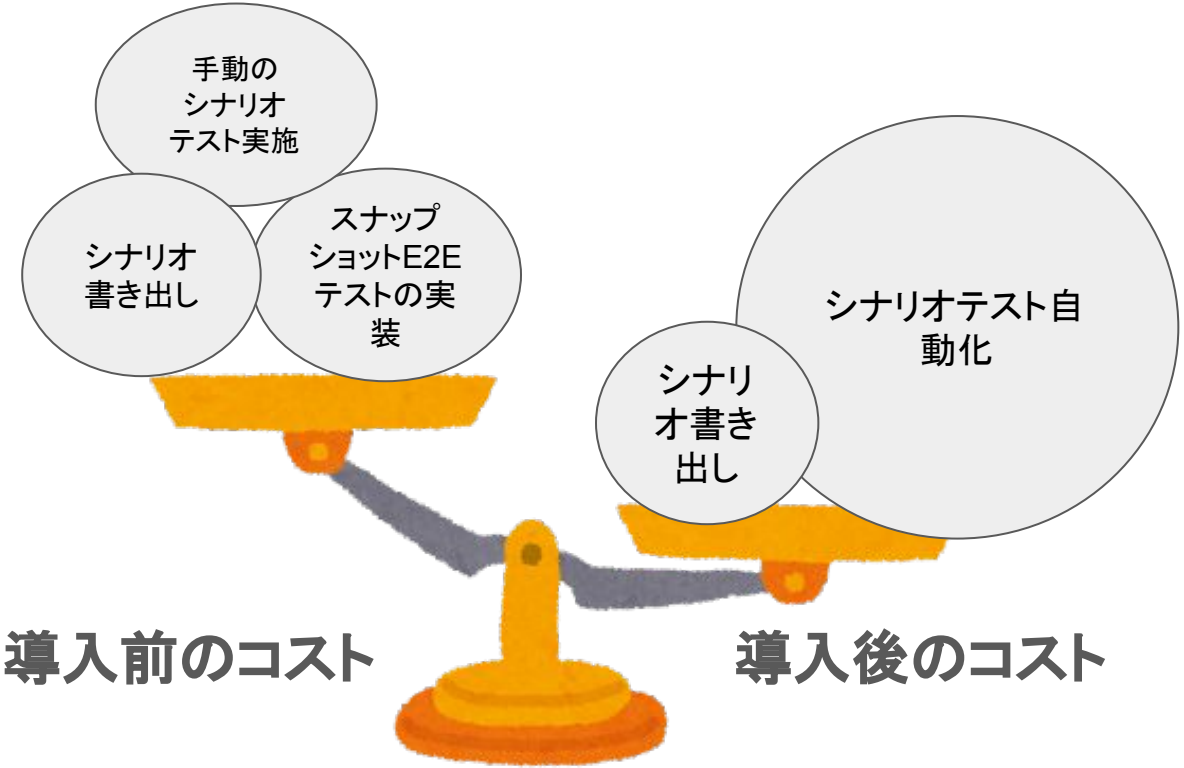
- 高コストで時間がかかることを最初やる
- 開発者主導でホワイトボックス視点が入りがち
- つい上位レイヤで品質を担保したくなる

→「さじ加減」を覚え，一旦サイクルを回し，適宜棚卸しもやる

自分たちのチームにとって
導入効果はあったか？

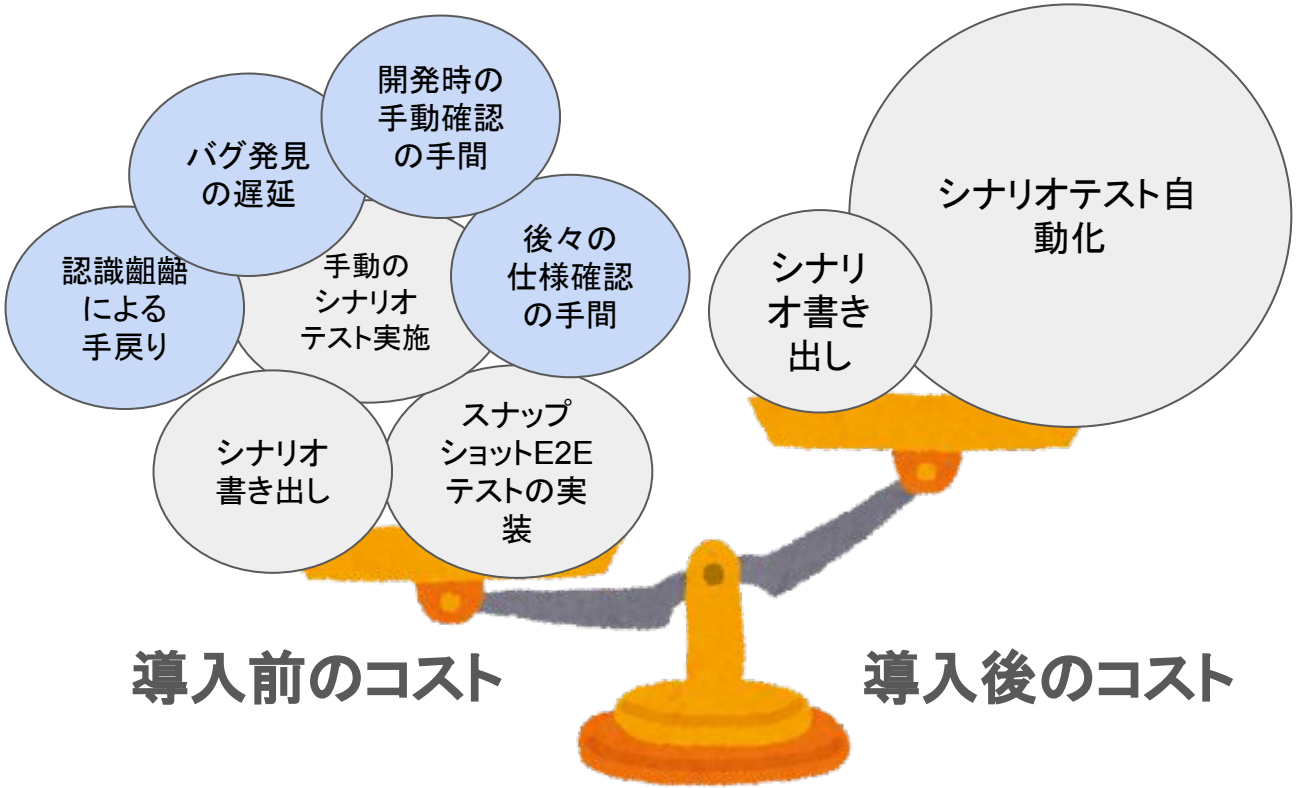
ATDD導入前後のコスト比較 ：一見すると …

単純な実装時のコストはATDD導入後の方が大きい



ATDD導入前後のコスト比較 ：実際は

ATDD導入により未導入時の隠れたコストが明らかになった



ATDD導入前後のコスト比較

ATDD導入により未導入時の隠れたコストが明らかになった

ATDD導入により
トータルのコストが減少

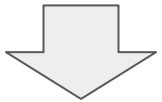
導入した効果：経験から言えること

- シナリオテスト自動化の実装やメンテナンスはそれなりに大変
- メリットは手戻り解消だけではない
  - 早期のバグ発見
  - リグレッションテストの充実
  - 開発中にE2Eテストをこまめに実行できる
  - 後からの仕様確認に役立つ
- **ATDD導入は、コストはかかるが後々救われる**

結論

結論

ATDD導入により**実装の品質が向上する**



素早く安定したデリバリを実現できる

これから始めるなら

導入時期

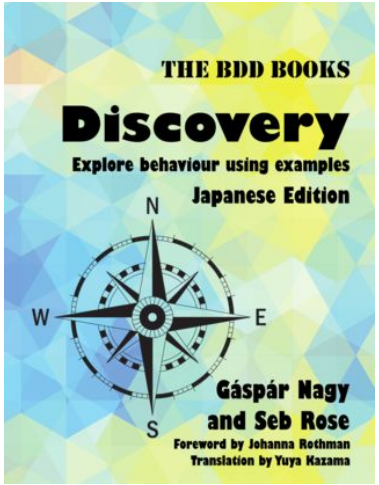
- プロジェクトの早めの段階，身軽なうち
- 人の入れ替わりや引継ぎタイミング

導入方法

- 具体例を書き起こして，シナリオだけからでもOK
- 自動化は具体例やシナリオに慣れてきたら，早めに

その他

- 「とりあえず」ではなく，少し学んでからがお勧め



KDDI Agile Development Center

Be a Change Leader.

アジャイルに力を与え
共に成長し続ける社会を創る