ソフトウェアテストを勉強して設計やマネジメントが上手になった話

IIJ Labセミナー 2018/06/12

kyon_mm

Speaker

- kyon_mm 株式会社オンザロード
  MaaS事業部 テストアーキテクト
- ソフトウェアテスト、スクラム、ソフトウェア構成管理、TDD
- Groovy/Scheme/Scala/F#

Agenda

- ソフトウェアテストを勉強する前の話
- ソフトウェアテストを勉強している
- ソフトウェアテストが役に立つ
- エントリーポイント
- まとめ

ソフトウェアテストを勉強する前の話

- 最初はJavaアプリケーションの開発
- そこそこ難しい書籍とかも読んでいた
  - Effective xxx系、〜パターン、など
- でも、開発が詰まってきたときにうまく引き継ぐことなんてできなかった。

開発に詰まったときの話1

- kyon_mm「〜がまだ出来ていません」
- 同僚「どれどれ、一緒に見てみようか」
- 同僚「んー。わかったけど、これだとうまく動かしながら直すのは難しいな。書き直そう」
- PM「また時間がかかってしまう。。。」

開発に詰まったときの話2

- kyon_mm「バグの原因がまだわかりません」
- 同僚「どれどれ、一緒に見てみようか」
- 同僚「んー。これはどのテストからやり直せばいいんだ。何が動いているのか検討がつかない。」
- PM「また時間がかかってしまう。。。」

そこからほんの少し調べたり

- ちょっとだけTDDを勉強したりもしたけど実践できなかった。
    - 密結合していてテストを入れられない
    - 古いソフトウェア環境で世の中のチュートリアルと乖離がある
    - 良いテストがなにかわからない。
- 結局テストに対して時間かけても意味なさそう。って思った。

ソフトウェアテストを勉強する前の状況

- そのソフトウェアで
    - 何が出来ていて、何が出来ていないのか
    - なんのためにそれがあるのか
- を正しく説明できる状態にないと、
  「うまくいかなかったとき」にたくさん時間がかかる。
- テストによって上手に開発する可能性を捨てている。
- 上のことを、kyon_mmは昔は鈍感で気づいていなかった。

Agenda

- ソフトウェアテストを勉強する前の話
- ソフトウェアテストを勉強している
- ソフトウェアテストが役に立つ
- エントリーポイント
- まとめ

勉強するキッカケ

- キッカケは友人がソフトウェアテストのイベントへの参加を誘ってくれたこと。
- ソフトウェアテストの基本を勉強してみようといろいろやってみた。
    - ソフトウェアテストの技法を勉強
    - カンファレンスなどの発表資料を読み込み

勉強しはじめたとき

- テストケースは何となくかけるようになった。
  - ドメイン分析テスト => 異常系含めたテストケースの効果的な作り方
  - 原因結果グラフ => 仕様を別表現にしてテストケースを効果的に作る

ソフトウェアテストを勉強して悩んだこと

- 前よりはただしくソフトウェアの状況を表現できるようになった
- ただし、
  - バグはまだそこそこ残っていて、終盤に見つかって困ることになる。
  - バグ発見まで時間がかかる。

あれ？テストって使えない？

テストって使えない？

- 説明できない => 説明できるテスト

- バグが早くに見つかる => コストの割に見つからない

- ということは

  「テストは誰がやっても一緒なのではないだろうか？」

  「面倒で地道なことをたくさんやるだけ？」

いや

テストが使えないとかの前に

開発が上手になっていない

なぜ上手にならないのか？

今思えば
全体のために仕事をしていなかった。

- 「ソフトウェア開発ビジネス全体」に対して「現場のチーム」として

  「最高のROI」を出すような取り組みをできていない。

- そうなってしまった原因

  「自分が出来ることしか想像できない」

  「ビジネスのWhyを理解せずに仕事している」

いろんなことを知らないと
開発は上手にならない。

テストは幅広いので
勉強のキッカケにはよさそう。

ソフトウェアテストを出発点にしていろんなことを勉強する

- 分析/要件定義/設計/実装/評価 のどれを勉強しても、それぞれのアウトプットを「評価する」という考え方はソフトウェアテストが使えそう。
    - 要件定義が出来たことのテストはどうする？とか。
- 何事においても、「それが出来たということはどういう基準なのか」を考えることができるようになる。
    - そのためにテストの技法が使えそう。どんなテストを考えて、どんな優先順位付けをしていくのか。

ソフトウェアテストで学べること

- 未知のものを「如何に効率よく学習するか」の知見がつまっている。
- ソフトウェアについての知見ではあるが短い時間でどうやって対象を把握する方法の塊。
- これはそもそもいろんなことに応用ができる。

ソフトウェアテスト

- テストは

  なんらかの差分がどこにありそうで、

  どの程度の差分なのかを、

  効率的に見つける方法

ソフトウェア開発への取り組み方が変わった

- ソフトウェアテストの勉強は続けるとして
- ソフトウェア開発における様々な知見を吸収するようにした。
- 全体を見ながらソフトウェアテストをできるようになってくると、早く、効率的にバグを見つけることができるようになってきた。
- そして、そもそもソフトウェア開発が上手になってきた。

Agenda

- ソフトウェアテストを勉強する前の話
- ソフトウェアテストを勉強している
- ソフトウェアテストが役に立つ
- エントリーポイント
- まとめ

テストの勉強が役に立った例

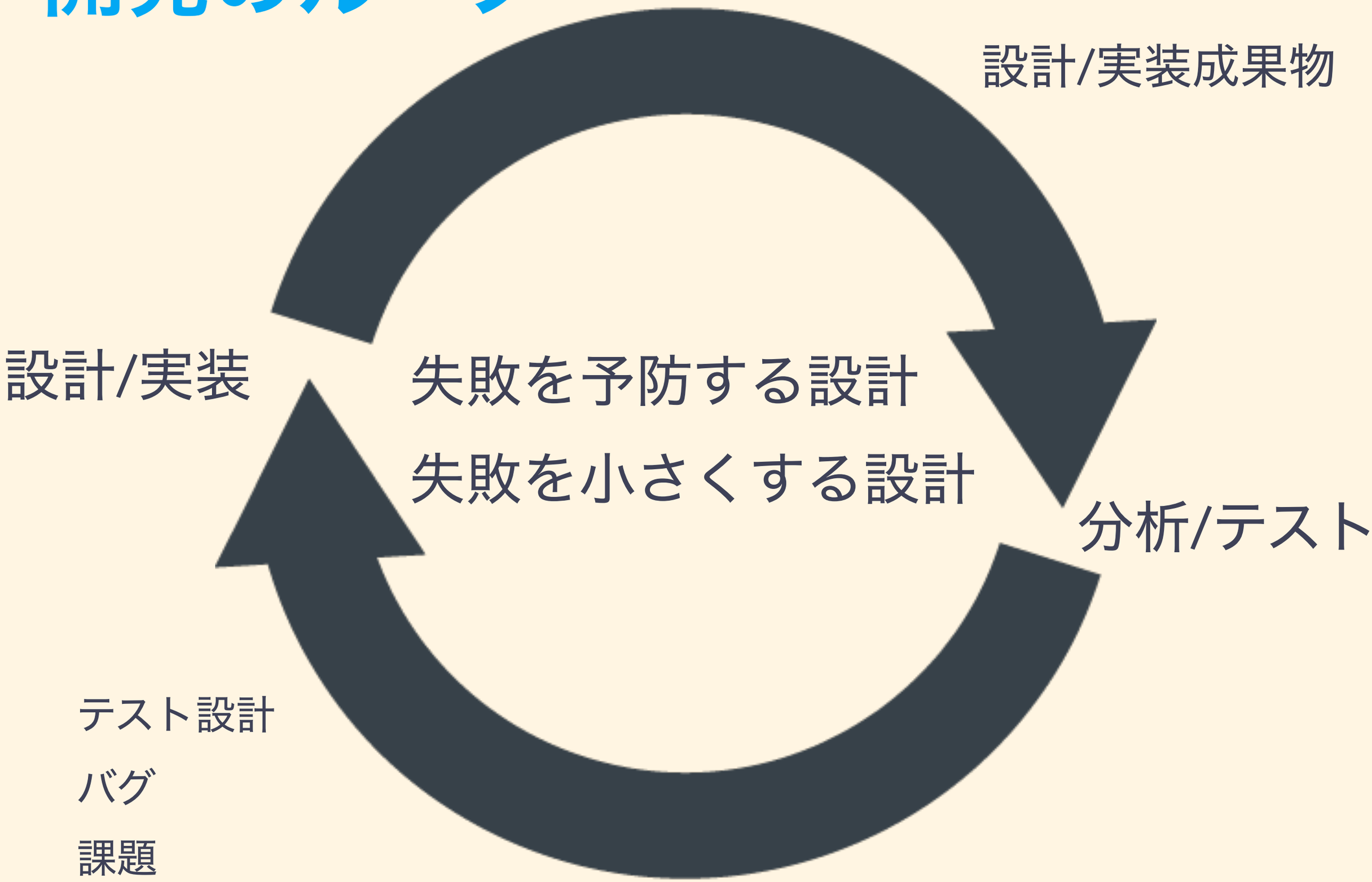
- 網羅対象と網羅基準をどうするのか？
    - ～に関する技術調査で何を調べるべきか？
- デシジョンテーブル/ドメイン分析テスト
    - ユニットテストなどで仕様のぬけもれをへらす
- 組み合わせテスト
    - サブシステム同士や機能同士の独立性を高める設計

テストプロセスで見る関連

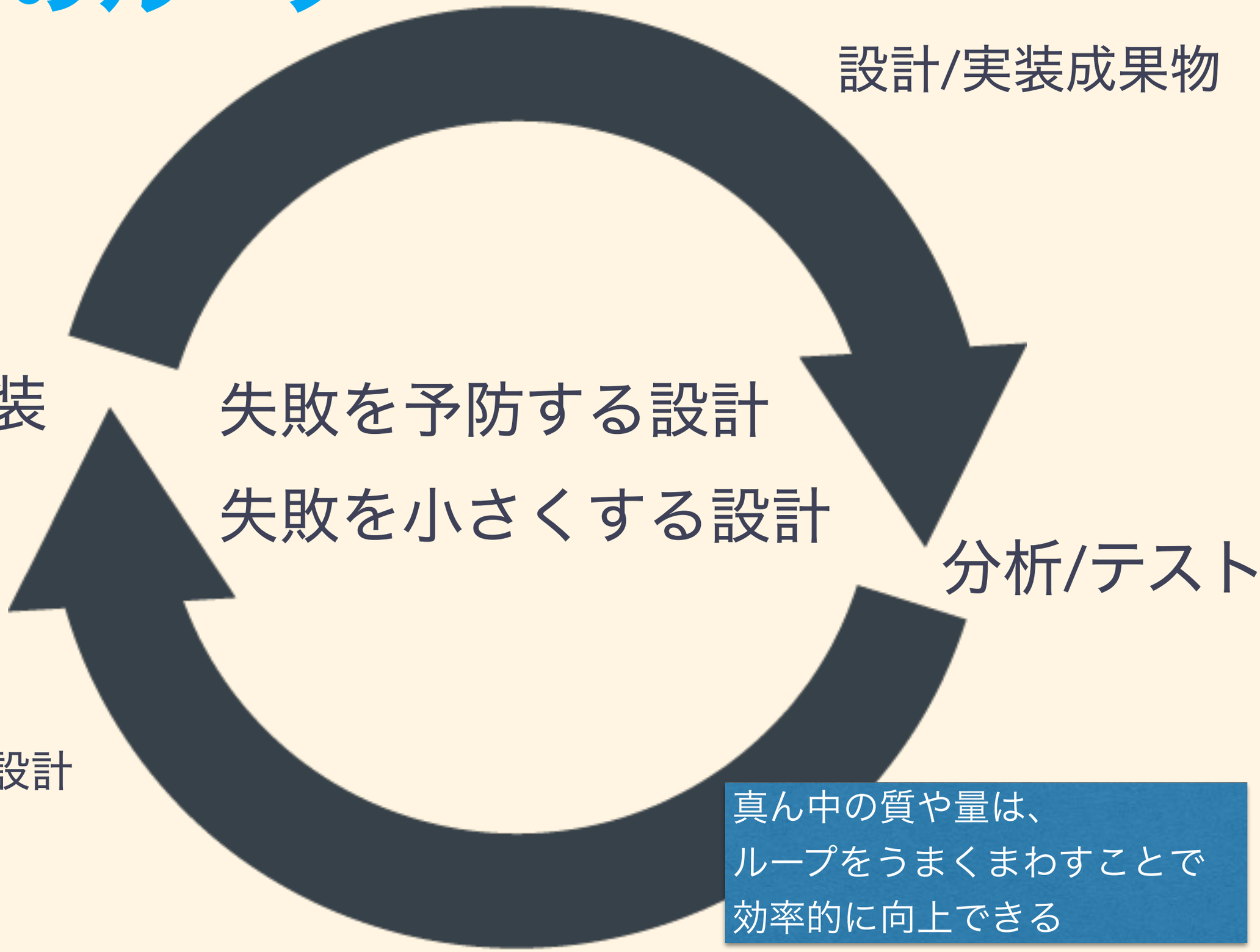
- テスト戦略策定 => プロジェクトマネジメント、プロダクトマネジメント
- テスト分析 => UX、技術文書、課題管理
- テスト設計 => システム設計、技術文書
- テスト実装 => 技術文書、自動テスト
- テスト実施 => タスク管理、自動テスト
- テスト報告 => 課題管理、技術文書、プロジェクトマネジメント

設計がうまくなった話

開発のループ

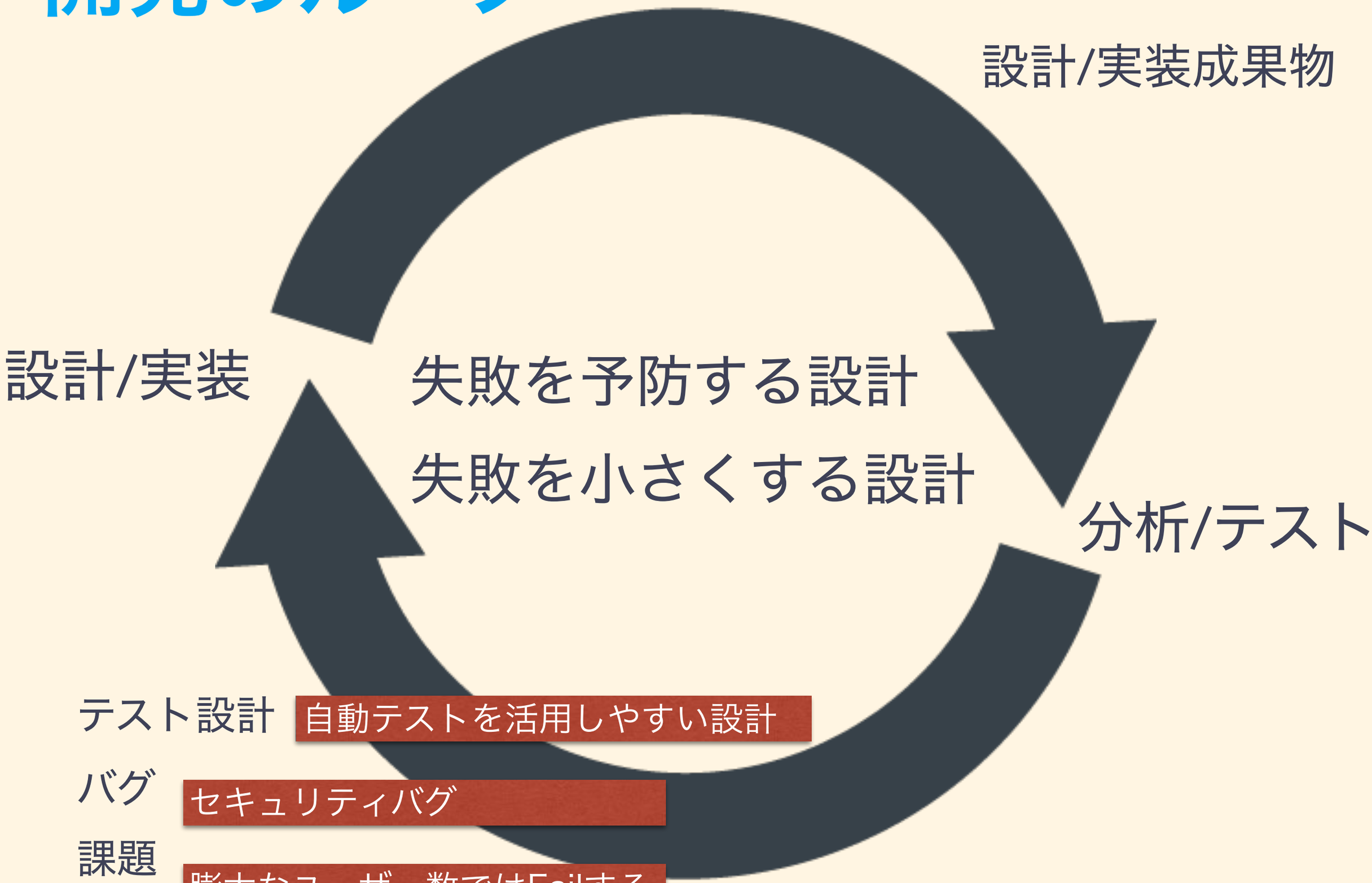


開発のループ



真ん中の質や量は、
ループをうまくまわすことで
効率的に向上できる

開発のループ

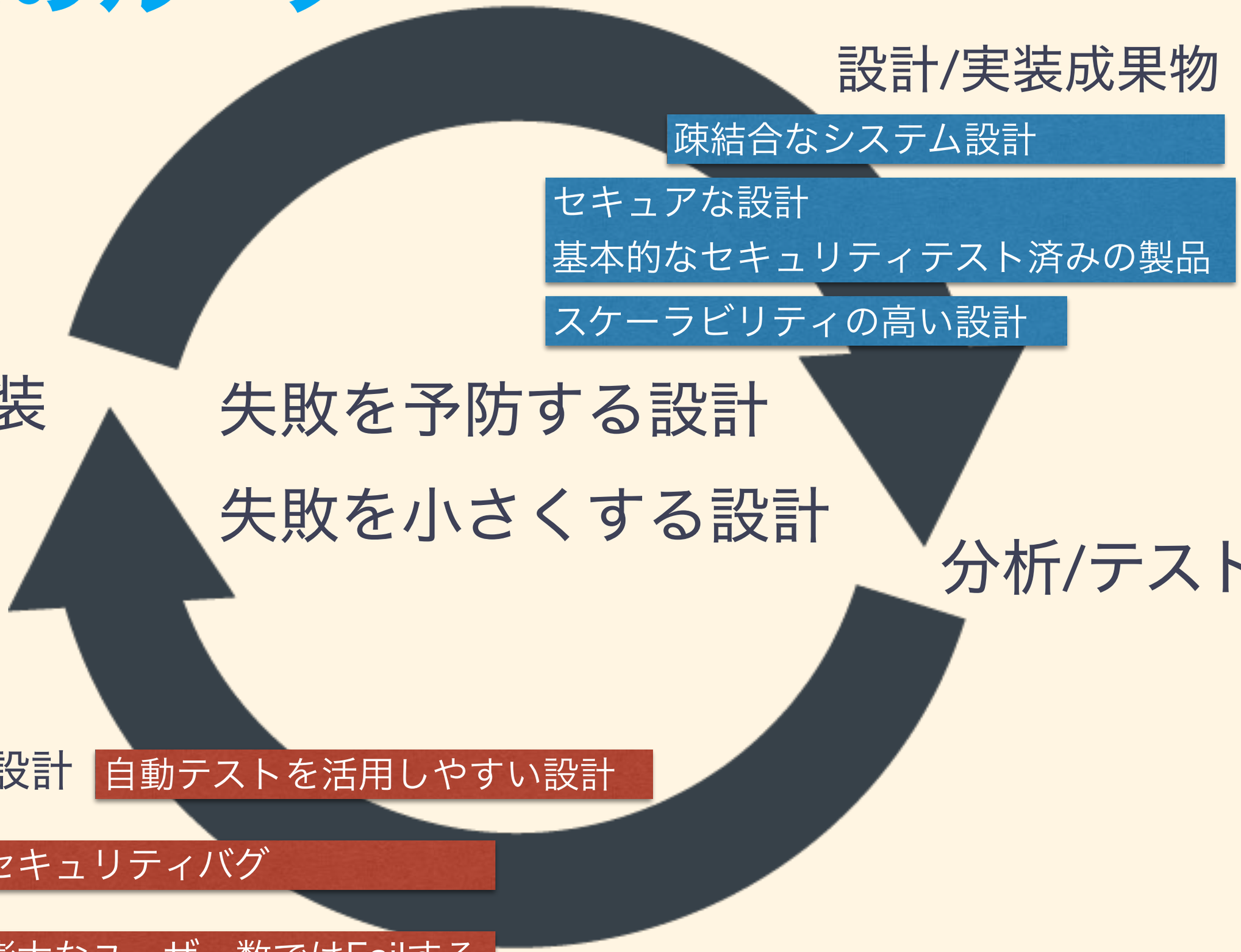


テスト設計 自動テストを活用しやすい設計

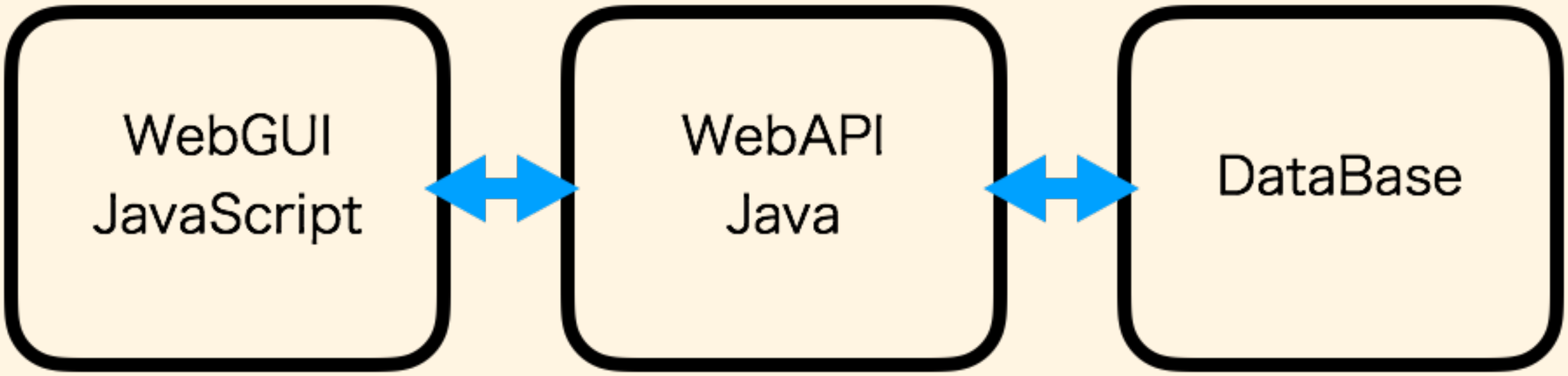
バグ セキュリティバグ

課題 膨大なユーザー数ではFailする

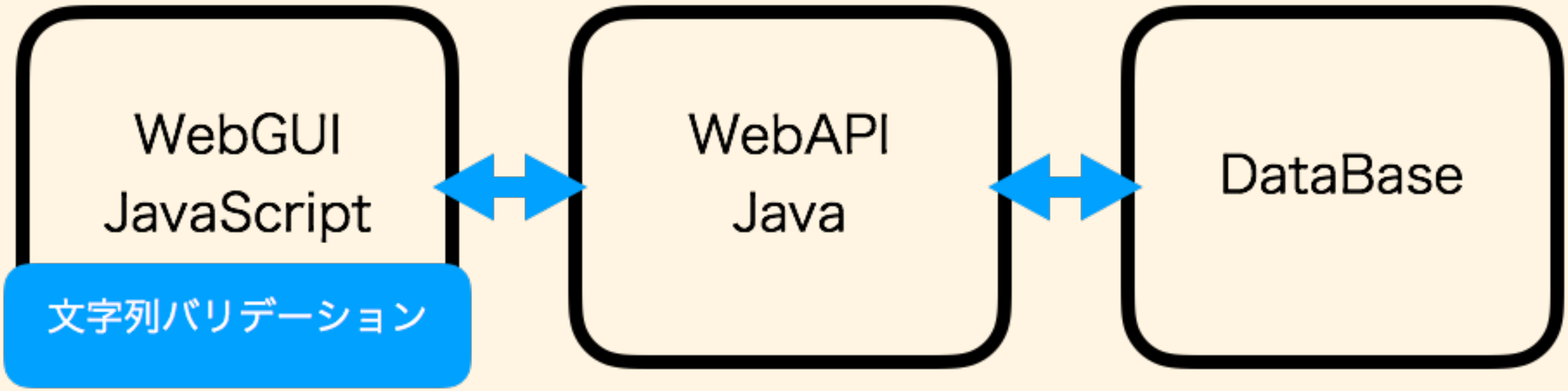
開発のループ


設計/実装成果物
疎結合なシステム設計
セキュアな設計
基本的なセキュリティテスト済みの製品
スケーラビリティの高い設計
設計/実装
失敗を予防する設計
失敗を小さくする設計
分析/テスト
テスト設計
自動テストを活用しやすい設計
バグ
セキュリティバグ
課題
膨大なユーザー数ではFailする

Webアプリケーションの例



最初のWebアプリケーション



セキュリティテスト/レビューを実施すると、SQLインジェクションなり、不正な操作を行える文字列を入力できてしまう構造になっていることが判明
このようにテストによってシステムの構造が浮き彫りになります。

修正後のWebアプリケーション

脆弱な構造を修正する。
そしてこれはTDDなどで設計として記述することができる。

テストの捉え方

- 設計や仕様や知見を成長させていくという目的

  「なんのためにいつ作られたテストか」によって2つに大別できる。

- 構造を浮き彫りにするテスト
  - セキュリティテスト、パフォーマンステスト、A/Bテスト、静的解析
- 構造を決定するテスト
  - TDD、セキュリティテスト、パフォーマンステスト

テストの捉え方

- 設計や仕様や知見を成長させていくという目的

  「なんのためにいつ作られたテストか」によって2つに大別できる。

- 構造を浮き彫りにするテスト
  - セキュリティテスト、パフォーマンステスト、A/Bテスト、静的解析
- 構造を決定するテスト
  - TDD、セキュリティテスト、パフォーマンステスト

あるテストがどちらかのみに存在するというような排他的な分類ではありません。

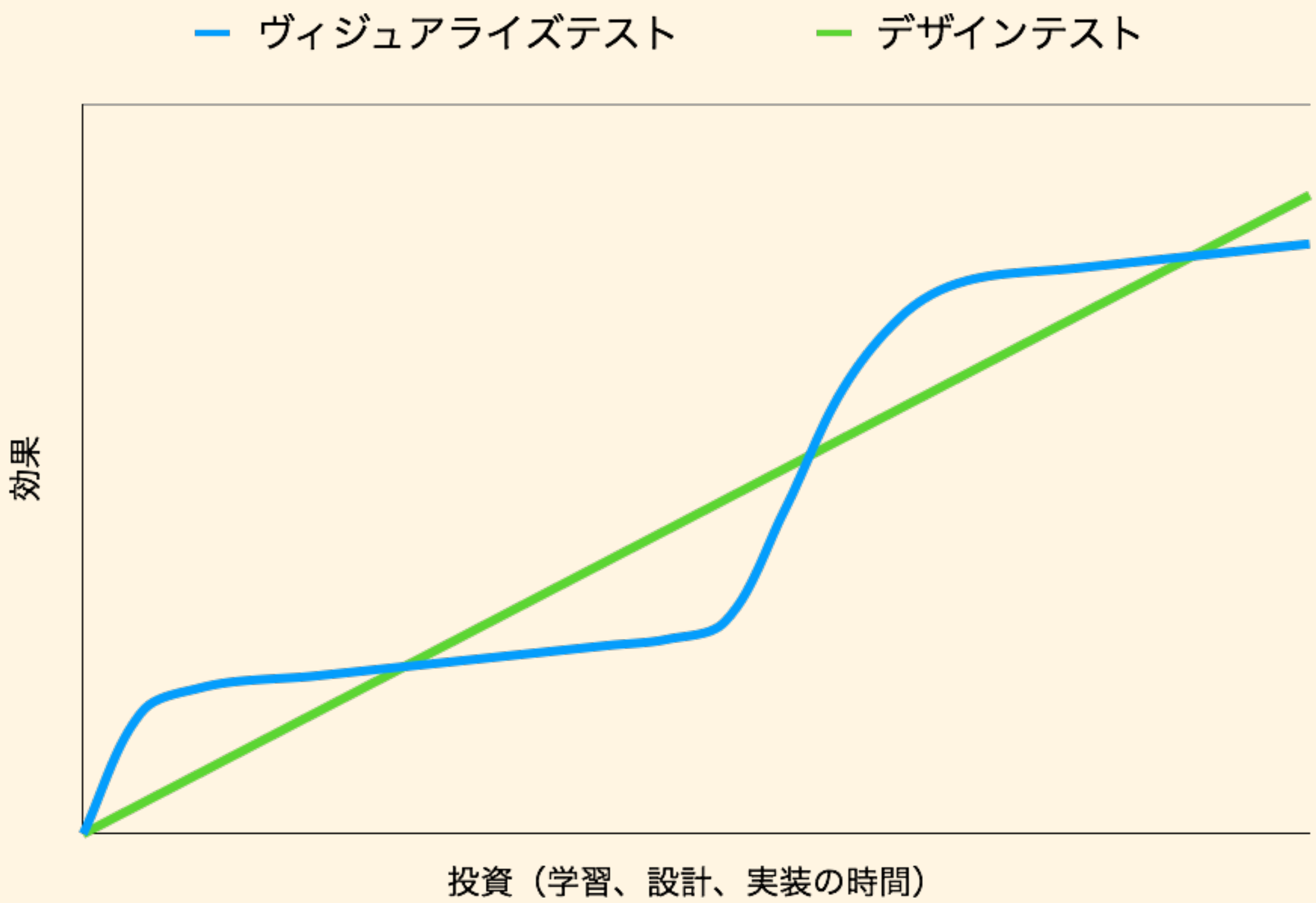
なぜ設計や構造に着目するのか

- システムで構造を変更することはとてもコストがかかる。逆に、構造を変えずに済む修正は簡単。

- クリティカルな構造的欠陥は早期に発見したり、頑強に設計した構造で実装できていることを実証したい。

- ソフトウェアテストはそのためにも使える。

テストの学習効果



構造を浮き彫りにするテスト における学習効果の特徴

- 構造を浮き彫りにするテストはテストツールが豊富なこともあり、初期投資ですぐに効果が出やすい。
- ツールで底上げされたところからさらに踏み込んで構造を浮き彫りにするためには多くの学習や経験が必要。
- 還元主義的にモノゴトを分解して見れるようなスキルをつけていくことと、俯瞰して要件と課題をつなげて見ていくスキルが必要。

構造を決定するテスト
における学習効果の特徴

- 構造を決定するテストはテストツールはあまり豊富ではないぶん、なかなか効果が出にくい。

- 積み上げて学習していく内容が多いため着実に効果が出ていきます。

- 常に検証範囲に入っているようなテストにするためのドキュメンテーションやコミュニケーションスキルが必要。

- 自動化、ドキュメント、ペアテスティングのように、構造を決定するテストは設計情報として扱うためのスキルが重要。

どちらかのスキルが不足？

- ソフトウェアテストを勉強してもなかなか効果がでない！

っていうときはもしかしたら、どちらかのテストに関しておろそかになっていないかを疑ってみましょう。

マネジメントがうまくなった話

マネジメントにも役に立つ

- チームに合わせてテスト戦略を策定できると、
    - 計画的に品質を向上させることができる
        - 構造欠陥のようなバグを早期から検査するような体制をつくり、大幅な実装修正の可能性を減らせる
- 結果として見積もりしやすくなる。

プロダクトマネジメント

- どういったことが起きると困るのか？リスクマネジメントもソフトウェアテストの範疇
- 特にソフトウェアの品質やバグについて詳しい人がリスクマネジメントに関わることで、中長期的にROIの高い計画を立てやすくなる。

プロジェクトマネジメント

- チームメンバーにとって「構造を決定するテスト」と「構造を浮き彫りにするテスト」のどういった部分が不足しているのかを把握できると、

- 計画的にどのようにメンバーのスキルアップを図るべきか？どのようなエキスパートに頼るべきかを議論し、計画できる。

ソフトウェアテストは
リスクベースが基本

- (kyon_mmは)ソフトウェアテストはリスクベースに進めるのが基本(と考えている)。

- マネジメントの大切なことに、「できるだけ交換可能なもので構成していくこと」がある。

- ソフトウェアテストでも複数の手段を構築してリスクに対処していくことが多く、それらを普通に出来るようになってくると、マネジメントがだんだん上手になる。

Agenda

- ソフトウェアテストを勉強する前の話
- ソフトウェアテストを勉強している
- ソフトウェアテストが役に立つ
- エントリーポイント
- まとめ

エントリーポイント

- ソフトウェアテスト全体
- 自動テスト
- ソフトウェアテストコミュニティ

ソフトウェアテスト全体

- 知識ゼロから学ぶソフトウェアテスト 【改訂版】

  高橋 寿一

- ソフトウェア・テストの技法 第2版

  ＧＪマイヤーズ

- マインドマップから始めるソフトウェアテスト

  池田暁, 鈴木三紀夫

- 基本から学ぶソフトウェアテスト

  Cem Kaner

自動テスト

- 初めての自動テスト —Webシステムのための自動テスト基礎

  Jonathan Rasmusson

- テスト駆動開発

  Kent Beck

- システムテスト自動化 標準ガイド

  Mark Fewster, Dorothy Graham

- 継続的デリバリー 信頼できるソフトウェアリリースのためのビルド・テスト・デプロイメントの自動化

  Jez Humble, David Farley

ソフトウェアテストコミュニティ

- Slack https://testingcommunityjp.slack.com
- WACATE http://wacate.jp/
- とちぎテストの会議 http://d.hatena.ne.jp/tochigitestnokaigi/
- JaSST http://jasst.jp/

Agenda

- ソフトウェアテストを勉強する前の話
- ソフトウェアテストを勉強している
- ソフトウェアテストが役に立つ
- エントリーポイント
- まとめ

まとめ

- ソフトウェアテストはいろんな人の
  効率的に発見する方法がまとまっている。

- 様々なことに応用がきき、仕事のいろんなことに対して、必要十分か？完成しているのか？などを定義したり、計測することまで考えられるようになる。

- 私達がつくるソフトウェアの品質を「表現する」スキルを直接的に学べる。