JaSST'10 東海 発表資料
2010/10/22

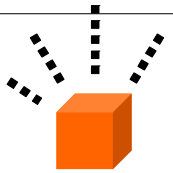
アジャイルテストとその活用方法

日本アイ・ビー・エム株式会社 グローバル・ビジネス・サービス AMS-SI ビジネス開発　増田聡
日本アイ・ビー・エム株式会社 金融事業　金融クライアントIT推進　第一クライアントIT部　瀧口健太郎

Agenda

1. はじめに
2. アジャイルとは
3. アジャイルテストとは
    1. アジャイルテストの4象限
4. アジャイルテストのプラクティス
5. アジャイルテストにおける自動化
    1. 自動化テストのカテゴリー
    2. 第1象限おけるツール活用－xUnit
    3. 第2象限おけるツール活用－Fit / FitNesse
    4. 第3象限おけるツール活用－Data Generator
    5. 第4象限おけるツール活用－自作負荷テストツール
    6. 継続的インテグレーション－Rational Team Concert
    7. 自動化ツール活用のポイント
    8. アジャイルチーム全体のアプローチ
6. おわりに

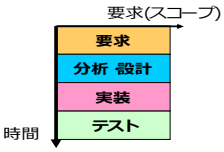
1. はじめに

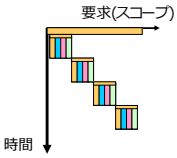


『アジャイルテストとその活用方法』

- アジャイルテストとはアジャイルソフトウェア開発におけるテストの考え方、およびプラクティスのことです。
- 私達は「実践アジャイルテスト」を翻訳する機会を得て、テストの視点からアジャイルソフトウェア開発を俯瞰する経験をしました。また、翻訳後、このアジャイルテストの考え方やプラクティスを実践するため、アジャイルテストをサポートするツール等を試行してみました。
- 本講演では、アジャイルテストを実践的に活用することを目的に、アジャイルテストの考え方やプラクティスの概要、試行したツールなどをご紹介します。

2. アジャイルとは





- アジャイル開発プロセスの特徴

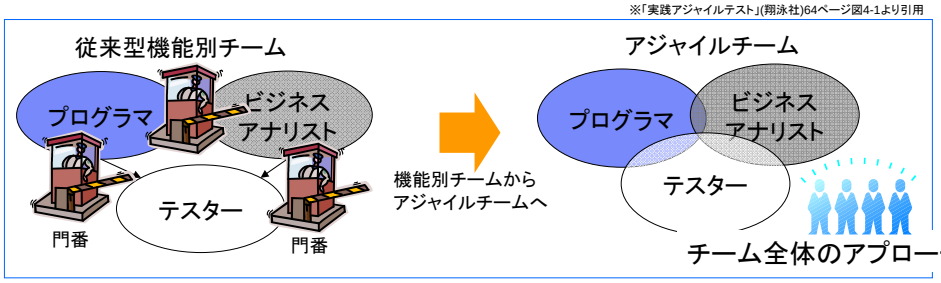
観点＼プロセス	ウォーターフォール開発プロセス		アジャイル開発プロセス
成功の尺度	計画への適合することが成功。	⇔	変化へ対応すること。動くコードを開発すること。
マネジメントの方法	命令と制御を主とする。	⇔	リーダーシップ、協業を主とする。
要求と設計の関係	あらかじめ全てを定義する。	⇔	継続的、ジャストインタイムで要求と設計を実施する。
コーディングと実装	全機能を同時にコーディングし、後でテストする。	⇔	コーディングとユニットテストを繰り返す。できたものから、順番に納品する。
テストと品質保証	計画に基づき、後でテストをする。	⇔	継続的・同時発生的に早期にテストをする。
計画とスケジューリング	PERT・詳細・範囲固定である。時間と工数を見積もる。	⇔	期日固定である。範囲を見積もる。

出展:「アジャイル開発の本質とスケールアップ」

3. アジャイルテストとは

- アジャイルテストとはアジャイルソフトウェア開発におけるテストの考え方、およびプラクティスのことである。
- アジャイルテストにはチーム全体アプローチ、自動化などのプラクティスがある。

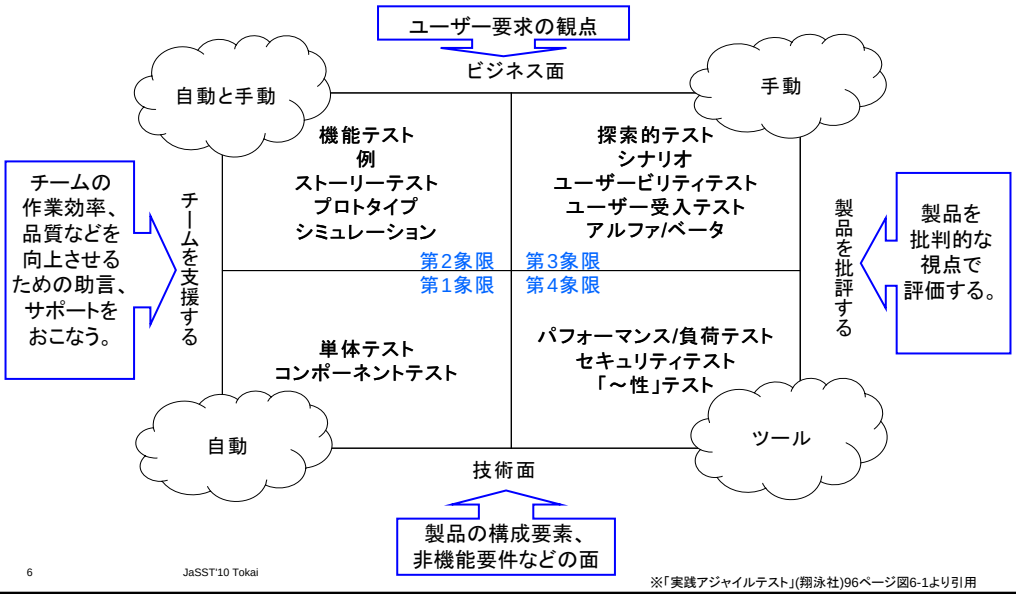
※「実践アジャイルテスト」(翔泳社)64ページ図4-1より引用



チーム全体アプローチは日本で既に取り組まれているアプローチであり、日本はアジャイルテストのプラクティスを適用しやすい環境にあると考えられる。

3. アジャイルテストとは
3－1.アジャイルテストの4象限

アジャイルテストは、ビジネス面または技術面の軸とチームを支援する目的か製品を批評する目的かの軸による4象限の分類で説明される。



※「実践アジャイルテスト」(翔泳社)96ページ図6-1より引用

4. アジャイルテストのプラクティス (1/2)

1. チーム全体のアプローチを取る

- プログラマと一緒に座り、自ら会議に参加する。
- 問題をチーム全体の問題としてとらえる。

2. アジャイルテストの考えを採用する

- 継続的により良い方法を探す。
- 良い本やブログ、記事を読み、新しいアイデアやスキルを身につける。

3. 自動リグレッションテストを適用する

- 自動リグレッションテストはチームの仕事である。テスターだけの仕事ではない。
- シンプルに始める。自動スモークテストや自動単体テストだけでも効率化される。

4. フィードバックを与え、受ける

- フィードバックはアジャイルの中心的な価値である。
- 反復計画会議と振り返りに十分な時間をかけて改善する方法を探る。

4. アジャイルテストのプラクティス (2/2)

5. コアプラクティスの基礎を築く

1. 継続的インテグレーションをする。
2. テスト環境を管理する。
3. 技術的な負債(*1)を管理する。
4. 段階的に作る。
5. コーディングとテストはひとつのプロセスとする。
6. 各プラクティスの相乗効果を図る。

6. 顧客と共同作業する

- テスターはまとめ役になる。

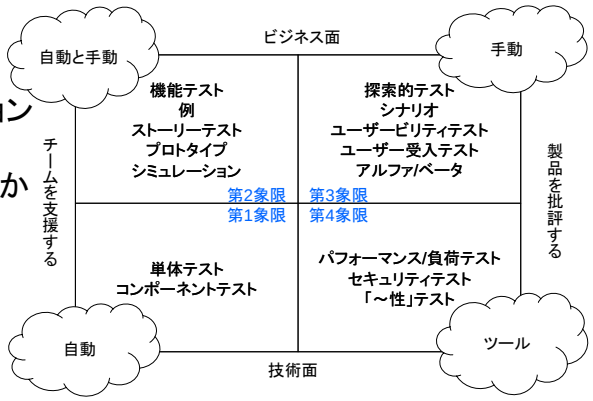
7. 広い視野を持つ

- 現在のストーリーがビジネスの重要なスキームに合うか評価できるようにする

*1:技術的な負債(Technical Dept) ≒ 未解決の技術的な課題の累積

5. アジャイルテストにおける自動化
5-1. 自動化テストのカテゴリー

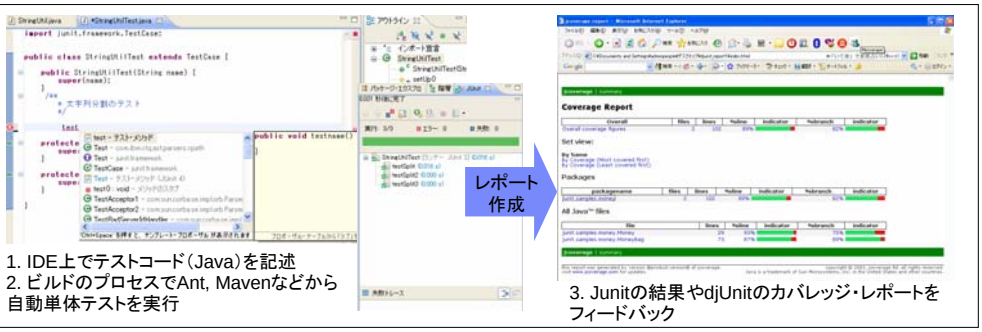
- 何を自動化できるのか
  - 単体テスト
  - 機能テスト
  - テストデータ生成
  - 負荷テスト
  - 継続的インテグレーション
- 何を自動化すべきでないのか
  - ユーザビリティテスト
  - 探索的テスト



5. アジャイルテストにおける自動化
5-2.第1象限おけるツール活用－xUnit

- アジャイルテストの自動化は単体テストが基本
- 一般的にxUnit系のツールを用いて、テスト対象のシステムと同じ言語で記述される
- 自動ビルドのプロセスで自動単体テストを実行させ、素早いフィードバックを実施することが必要

Eclipse + JUnit/djUnitの例

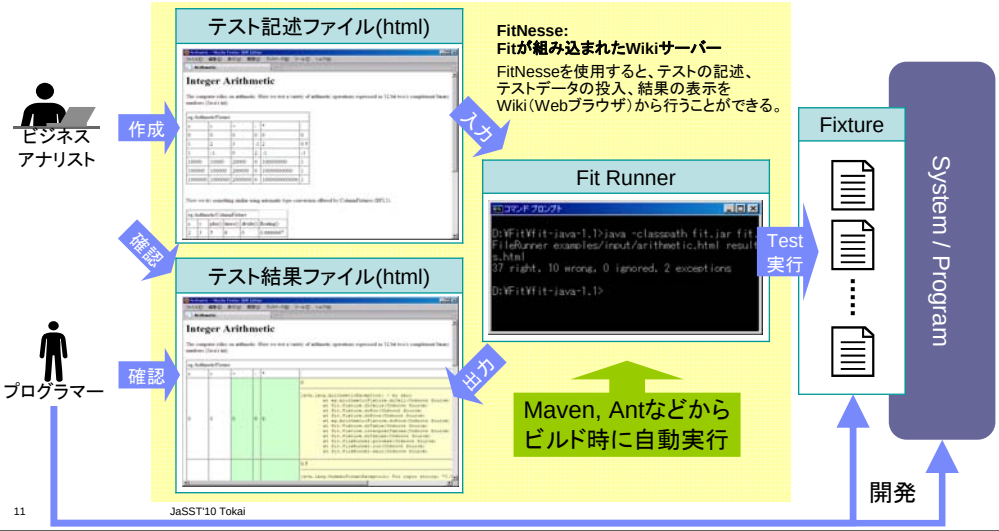


1. IDE上でテストコード（Java）を記述
2. ビルドのプロセスでAnt, Mavenなどから自動単体テストを実行

3. Junitの結果やdjUnitのカバレッジ・レポートをフィードバック

5. アジャイルテストにおける自動化
5-3.第2象限におけるツール活用－Fit/FitNesse

- チームを支援するビジネス面のテストにも、自動化ツール適用の効果がある



5. アジャイルテストにおける自動化
5-4.第3象限におけるツール活用－Data Generator

- 第3象限のテストにおいても、テストデータの作成など自動化が可能

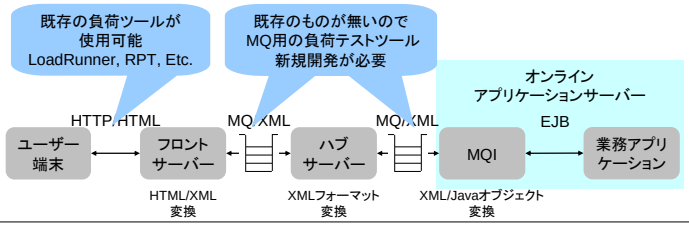
5. アジャイルテストにおける自動化
5-5.第4象限におけるツール活用－自作負荷テストツール

- 現実には、申請・承認などを伴うワークフローアプリケーションが多い。
- パフォーマンステストは、ワークフローを始めから終わりまで全部実施しようとすると、大部分のアプリケーション開発が終了するのを待たなければならない。
- プロジェクトの早期にテストを実施する方法を見つける必要がある。

自作負荷テストツールの例

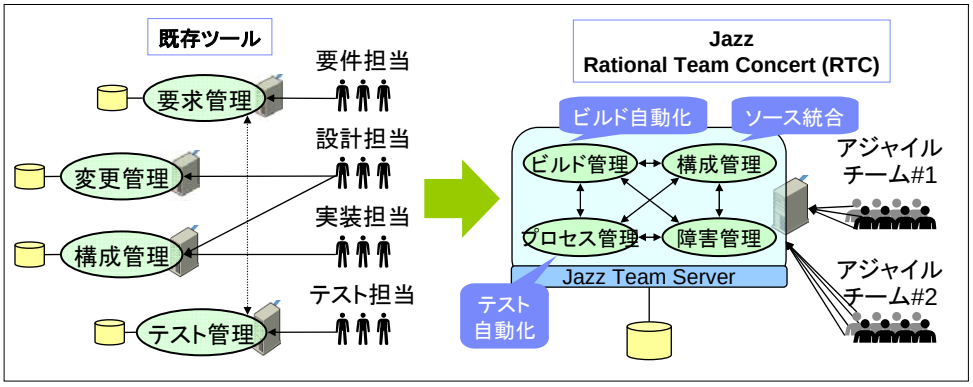
開発の背景
- HTTPサーバーのコードが完成する前に業務アプリケーションのパフォーマンス・テストを行いたい
- 業務アプリケーション単体のパフォーマンス・テストを行いたい
- ハブサーバー経由のパフォーマンス・テストを行いたい

⇒ MQ のプロトコルでXMLメッセージを送受信する負荷ツールが必要



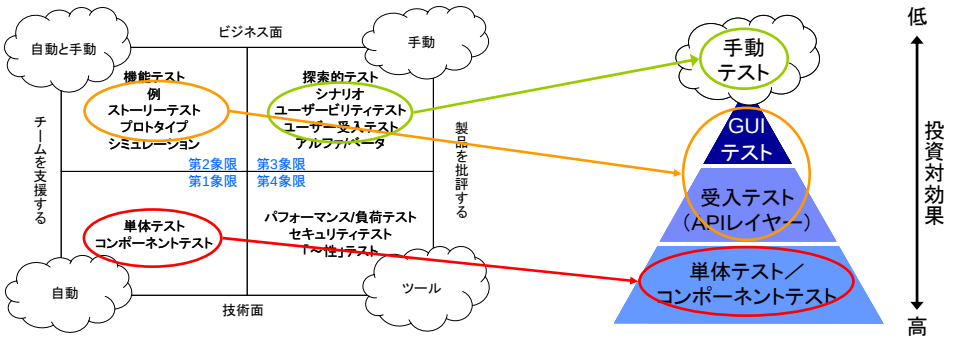
5. アジャイルテストにおける自動化
5-6. 継続的インテグレーション－Rational Team Concert (RTC)

- 正常に動くビルドを少なくとも1 日1 回作り出す。
- 毎日統合ビルドを実施することで、プログラム間の不整合を早期に発見可能である。
- 実現にはソース統合、ビルド自動化、テスト自動化が必要である。



5. アジャイルテストにおける自動化
5-7. 自動化ツール活用のポイント

- アジャイルテストの4象限およびテスト自動化のピラミッドを用いて
  どこでテスト自動化が必要かを判断しツールを活用する

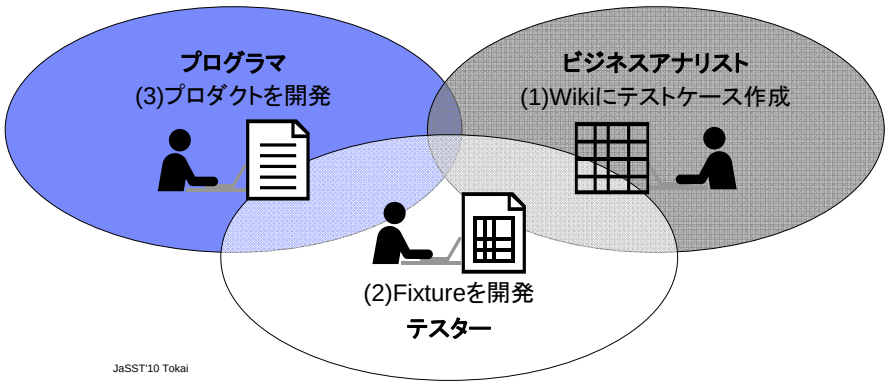


アジャイルテストにおいて自動化の効果が最もあるのは単体テスト。
ピラミッドの頂点から自動化を始めるのではなく、
基礎がきちんとしていることを確かめながら底辺から始めるべき。

※「実践アジャイルテスト」(翔泳社)274ページ図14-2より引用

5. アジャイルテストにおける自動化
5-8. アジャイルチーム全体のアプローチ

- テスターや品質保証の人たちだけが、品質に責任をもっているわけではない。
  アジャイルチームはすべてのテスト作業に責任をもつ。
  - FitNesseなどのツールを使用することにより、ビジネスアナリスト、プログラマ、テスターが
    ひとつのフレームワーク上でテストケース作成からプロダクト開発まで作業ができる。



6. おわりに
本日のまとめ

1. アジャイルテストとはアジャイルソフトウェア開発におけるテストの考え方、およびプラクティスのことである。
2. チーム全体アプローチは日本で既に取り組まれているアプローチであり、日本はアジャイルテストのプラクティスを適用しやすい環境にあると考えられる。
3. アジャイルテストの自動化においてはアジャイルテストの4象限およびテスト自動化のピラミッドを用いてどこでテスト自動化が必要かを判断しツールを活用する。
4. テスターや品質保証の人たちだけが、品質に責任をもっているわけではない。アジャイルチームはすべてのテスト作業に責任をもつ。

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したものではなく、またそのような結果を生むものでもありません。本プレゼンテーションに含まれている情報については、完全性と正確性を帰するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本プレゼンテーションまたはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。 本プレゼンテーションに含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引きだすことを意図したものでも、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本プレゼンテーションでIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本プレゼンテーションで言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したものでも、またそのような結果を生むものでもありません。
パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。