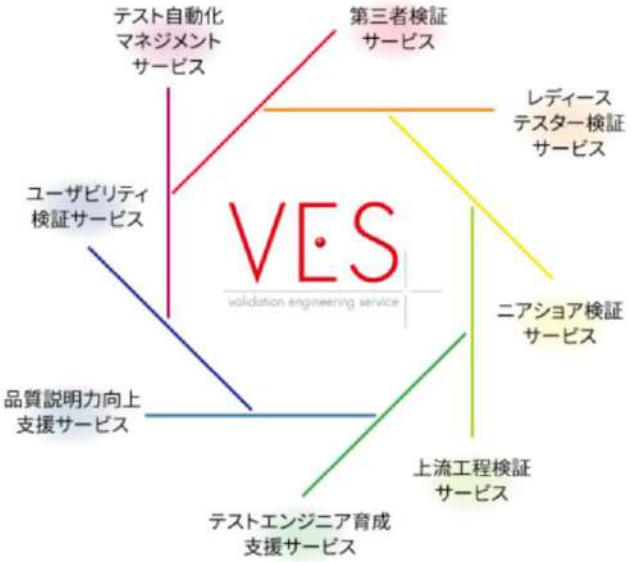
第三者検証専門会社　自動化最前線
– 現状とこれから –

2018年12月14日

株式会社ヴェス

会社紹介

社名	株式会社ヴェス
設立	2003年(１６期目)
従業員	138名
所在地	■東京本社（四ツ谷） ■滝沢ソフトウェア検証センター (岩手県滝沢市)



業務内容

- ＩＴ関連製品などの新商品開発に伴うソフトウェアの第三者検証サービス
- レディーステスター検証サービス
- ユーザビリティ検証サービス
- テストエンジニア育成支援サービス

講演者紹介

株式会社ヴェス
検証サービス本部
マネージャー

松田 浩志（Matsuda　Hiroshi）

1999年　富士ソフト株式会社　入社
2008年　V&V株式会社　出向
2011年　V&V株式会社解散に伴い　出向終了
2016年　株式会社ヴェス　入社

趣味：人狼ゲーム

大規模 第三者検証プロジェクトなどを担当。
スマートフォン専用放送局 NOTTV において、
新規立ち上げからサービス終了まで、品質面でかかわりました。

品質というキーワードで世の中を幸せにして、社会に貢献したいと考えています。

はじめに

第三者検証をとりまく現状

- 自動化要望が非常に強い（全業種）
- 自動化は必須スキル

テストを自動化していく中で、出てきた課題やその対策について、デモンストレーションを交えてお伝えしたいと思います。

目次

自動化　実績

自動化対応実績分野

金融

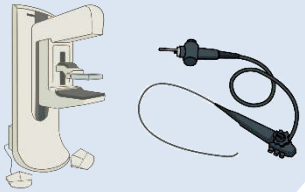
教育

動画配信サービス

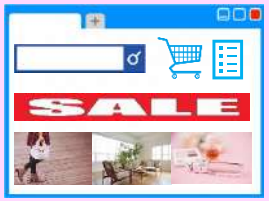
精密機器

ゲーム

医療



ECサイト



クラウドアプリ

自動化　求められていること

2．自動化　求められていること

■自動化の目的

主目的

・テスト実施　工数削減
・リグレッションテスト　工数削減
・人手がかからず実施できる
・属人化排除

副次効果

・継続的インテグレーションとしての連携
・開発サイクルの加速
・テスト網羅率の向上
・メトリクス収集
（部分的に：実施精度向上、ログ収集力向上）

自動化　現状

3．自動化　現状

業種と使われているツール群

金融決済POSレジ等　高機能ツール活用

EggPlant

医療系/精密機器等
windowsアプリ中心に中機能ツール活用

Ranorex/Silk Test/UWSC

クラウドサービス/webサイト/android app/iOSapp系

selenium/Appium/EarlGrey/XCUITest

使われているツール群の特徴

ツール群は２つにわかれる

- **レコード/オブジェクト取り込み型**
  ブラウザ操作を記録し編集する（オブジェクト取り込み）

- **スクリプト型**
  スクリプト記載を自力で記載する

レコード/オブジェクト取り込み型　有料が多い

メリット

・操作を記録（レコード）し、スクリプト作成
・UIキャプチャ機能とオブジェクト認識によりxPath等取得
・期待値画像との一致判定により、テスト結果自動判定
・マルチプラットフォーム対応（windows,web,android,iOS)
・有料のものは、サポート体制が強い（高レスポンス）

デメリット

・スクリプト記述や条件判断などの自由度が低い
・有料の場合、ライセンス数しか技術者配置ができない

スクリプト型

メリット

・操作を自分でスクリプト記載する為、自由度が高い
・selenium webdriver/Appium/XCUITestを中心に無料(OSS※1)
・技術発展が早く、今後の進化が期待できる
・他システム連携を自分で作成できる（AWS※２　上で動かすなど）

デメリット

・開発言語の理解が必須
・自動化エンジニアの育成が必要

※１ ソースコードの改変や再配布が自由に認められている無償のソフトウェア
※２ Amazonが提供しているクラウドコンピューティング（Webサービス）

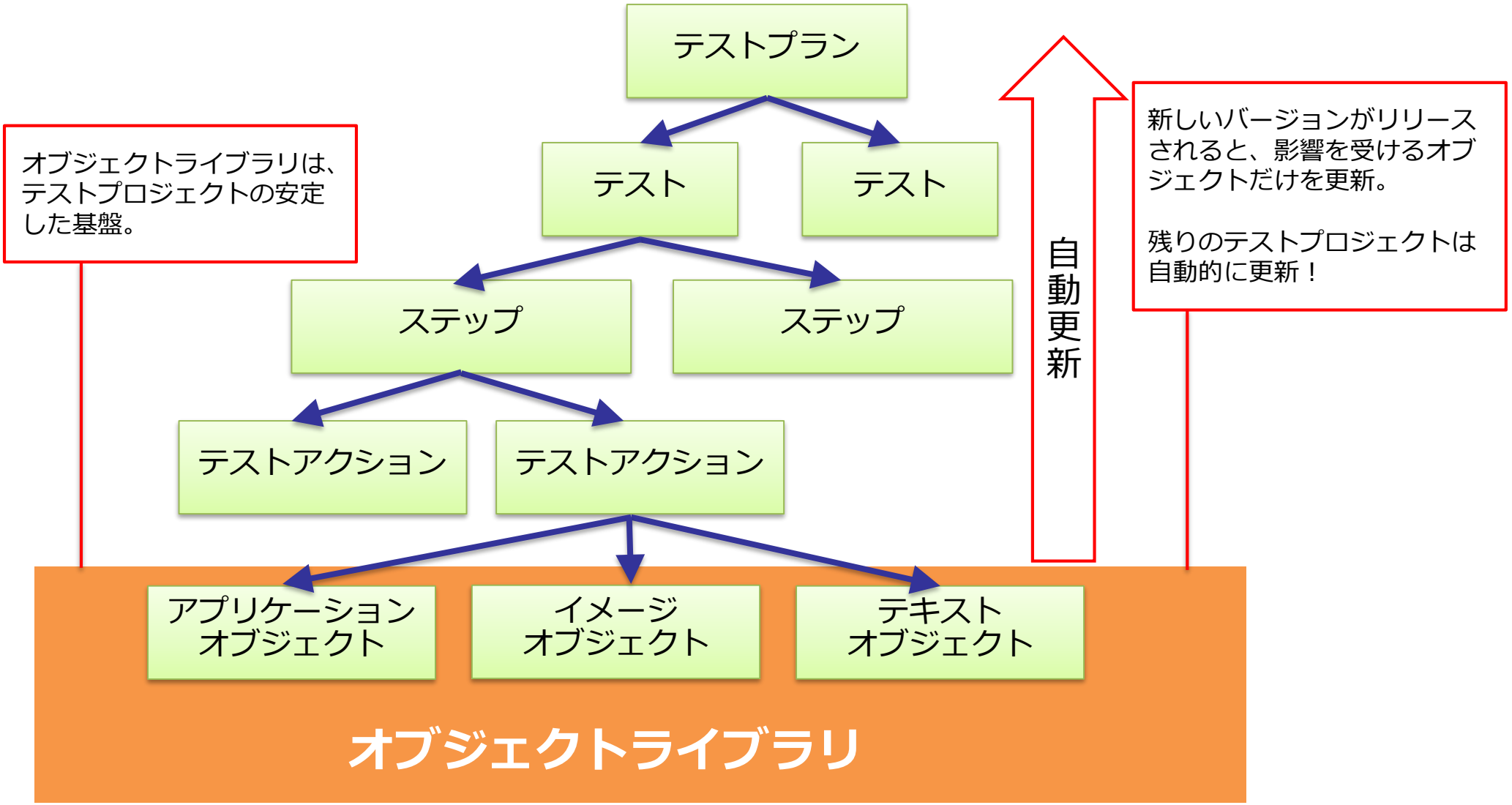
第３の分類？

オブジェクト取り込み特化型

日本初上陸

webのオブジェクトをすべて取り込み、その要素を
ドラッグ＆ドロップして捜査対象とする。
これらにアクションを付け加えることで、シナリオを
作成する。

オブジェクト構造の変化をアプデしてくれる



AscentialTestを使って、シナリオ作成します。

自動化　課題と対策

４．自動化　課題と対策

当初の自動化適用では、

- ▶ 計画の立案
- ▶ 自動化範囲の設定
- ▶ 自動化率の目標
- ▶ その後の運用コストを想定し、、、効果試算

など、戦略的な枠組みを考えていましたが

現実は厳しい・・・

４．自動化　課題と対策

■現実的な課題

・自動化スクリプト作成が難しい（対象要素がとれない）
・自動化スクリプトがうまく動かない（不安定）
・自動化スクリプト作成工数が想定より大きい
・自動化スクリプトのメンテ工数がばかにならない
・すべての試験を自動化できるわけではない
・スクリプト作成者が不足している
・次バージョンで、サイト構造が大きく変わり作り直し
・自動化率の目標値の高さが、現実乖離して、失敗認定
・はては　該当プロジェクト中止・・・

４．自動化　課題と対策

■課題への対策

・リグレッション試験から自動化

必ず繰り返すものを対象とする

作成が難しいところは除外するか、次回か次次回か

・リグレッション試験のテスト自動化領域を広げるアプローチ

自動化率の向上で自身が楽になるように

※高い自動化率を設定するのではなく、繰り返しの中で自動化率を上げていく

・継続的インテグレーションにて活用

（ビルド都度、必ずリグレッション試験3,000項目が回っているなど）

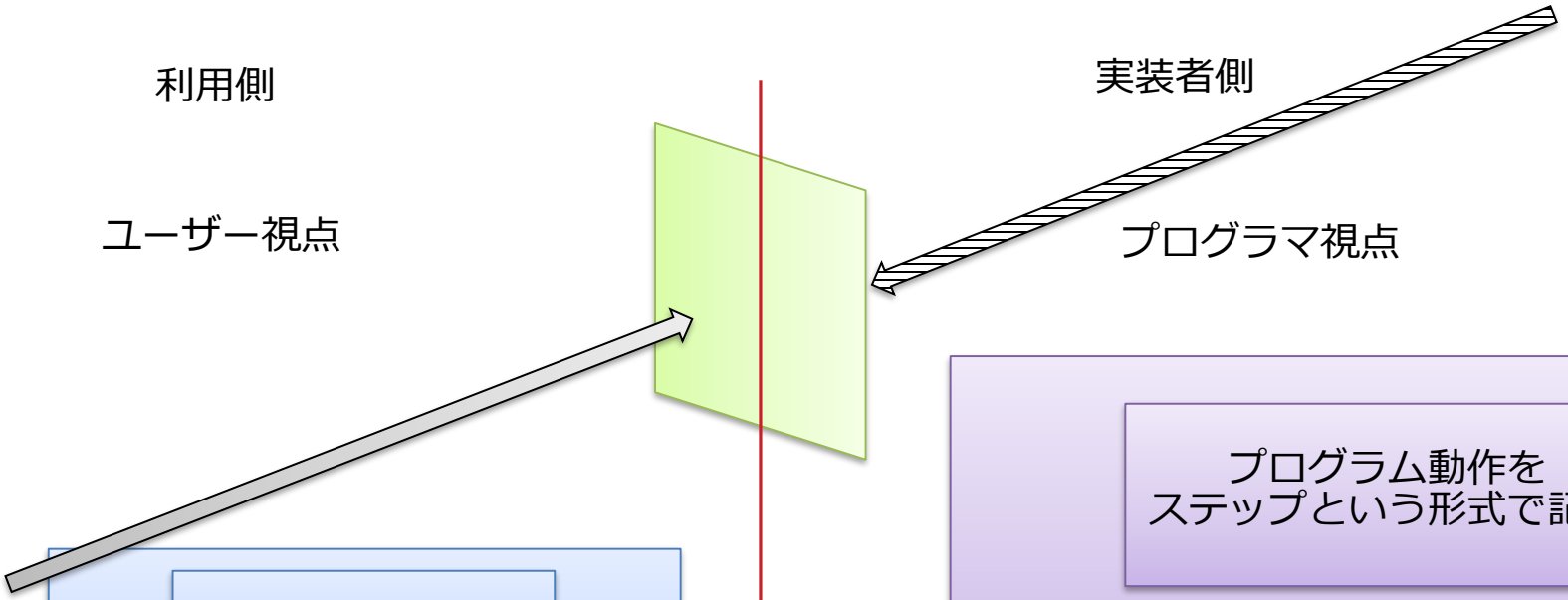
４．自動化　課題と対策

課題解決：
テスト自動化　テスト仕様作成とコード作成者の役割分担

「 Ruby + Rspec + Selenium + Capybara + turnip 」

- Ruby　　：言語(web系に人気がある)
- Rspec　　：rubyのテストフレームワーク
  （振舞駆動開発 (Behaviour Driven Development:BDD))
- Selenium：webの遠隔操作ドライバ
- Capybara：seleniumラッパーライブラリ(合否判定などを簡易化する)
- Turnip　　：rspec向けテスト自然言語対応ライブラリ
  （仕様記載ファイルと手順（プログラムstep）ファイルを分けて記載できる）

Turnipのメリット図式



Rspec + Turnip

featureファイル

Scenario: VESサイトにアクセスする
　　Given VESサイトにアクセスする
　　When　トップページを表示する
　　Then　titleに"第三者検証の株式会社ヴェス"と表示されていること

Scenario:　「アジャイル開発の資料はこちら」画面に遷移＋必要事項入力
　　When　「アジャイル開発の資料はこちら」をクリック
　　Then　titleに"アジャイル開発 資料ダウンロードフォーム"と
　　　　表示されていること

　　When　氏名に姓に"ヴェス"名に"太郎"を設定する
　　When　氏名カナに姓に"ヴェス"名に"タロウ"を設定する
　　When　貴社名に"株式会社ヴェス"を設定する
　　　・・・・・

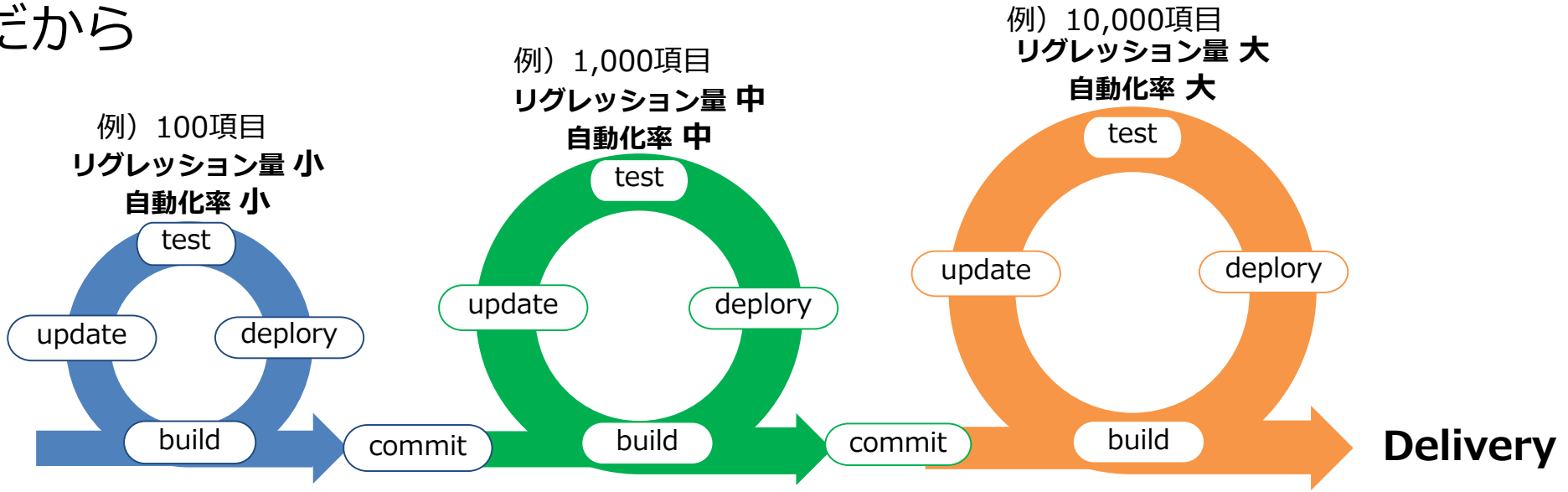
stepファイル

```
step 'VESサイトにアクセスする' do
  Capybara.app_host = "https://ves.co.jp/"
end

step 'トップページを表示する' do
  visit '/'
end

step %(titleに:text1と表示されていること) do |text1|
  expect(page).to have_title text1
end

step '「アジャイル開発の資料はこちら」をクリック' do
  click_link_or_button "・ 資料ダウンロード（アジャイル開発）"
end

step %(氏名に姓に:text1名に:text2を設定する) do |text1, text2|
  fill_in "col__132__9__0__0", with: text1
  fill_in "col__132__9__0__1", with: text2
end

step %(氏名カナに姓に:text1名に:text2を設定する) do |text1, text2|
  fill_in "col__133__9__0__0", with: text1
  fill_in "col__133__9__0__1", with: text2
end

step %(貴社名に:text1を設定する) do |text1|
  fill_in "col__134__1__0__0", with: text1
end

step %(部署名に:text1を設定する) do |text1|
  fill_in "col__135__1__0__0", with: text1
end

step %(電話番号に:text1を設定する) do |text1|
  fill_in "col__140__11__0__0", with: text1
end

step %(メールアドレスに:text1を設定する) do |text1|
  fill_in "col__142__10__0__0", with: text1
  fill_in "col__142__10__1__0", with: text1
end
```

4．自動化　課題と対策

■課題への対策

・テスト実施/リグレッション試験をカプセル化して、
　自分の仕事が楽になるように、自動化していく
・テスト設計実施以外の時間はすべて自動化対応工数へ
・プロジェクトの成熟度向上に伴い、自動化率向上していく
・リグレッションテスト量を大きく育てて、プロジェクトを包み込む
　イメージ。なぜなら、リグレッションテスト実施コストが超安価
　だから



自動化　これから

実端末操作をクラウドで（実機レスな時代）

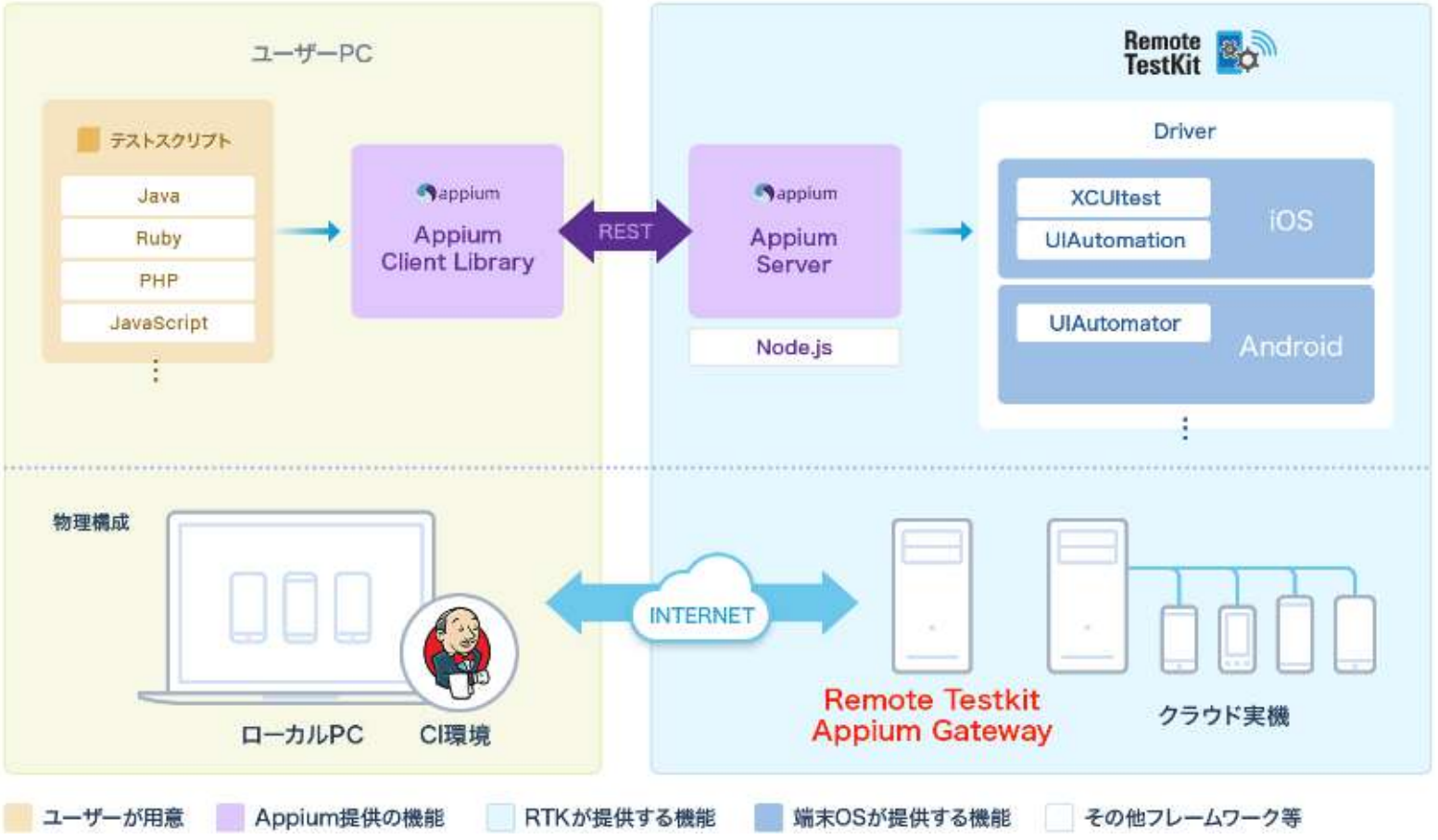
Remote TestKit:クラウド上で、実端末（500機種以上）を操作できるクラウドサービス（NTTレゾナント様）

Remote TestKit

https://appkitbox.com/testkit/

実端末操作をクラウドで（実機レスな時代）

Appiumからも呼び出し可能（＝自動実行可能）



5．自動化　これから　-実端末操作をクラウドで-

実端末操作をクラウドで（実機レスな時代）

継続的インテグレーション（CI/CD)の動作確認として、
実端末テスト実施を実現

Jenkins +GitHub + AndroidStudio + Selenium/appium + RemoteTestKit

▶ DEMO

5．自動化　これから

Selenium 要素取得をAIでメンテコスト小へ

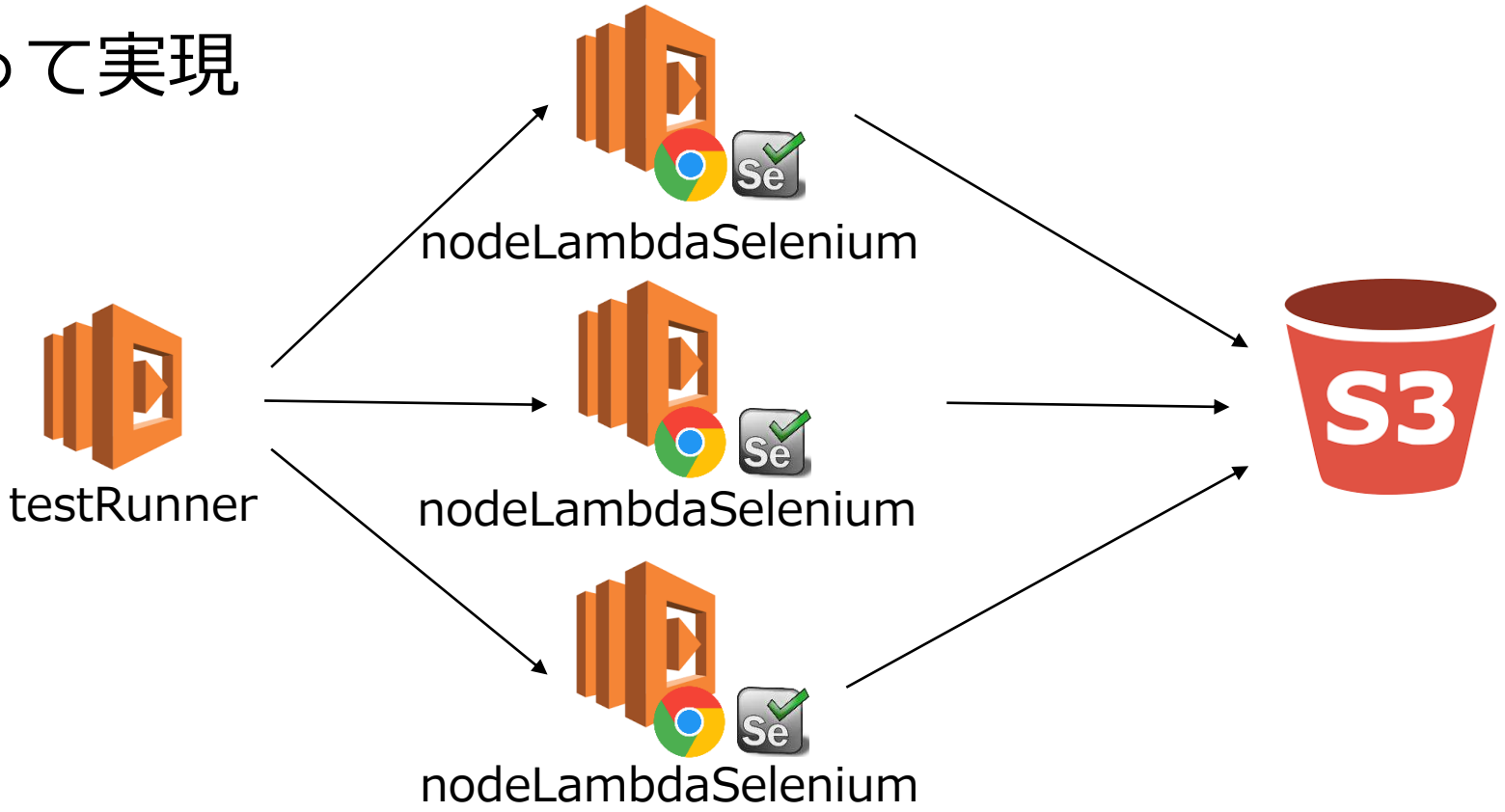
selenium/appium + test ai

With one API call, you can add the power of AI to your mobile test

```
driver.findElementByCustom("ai:search");
```

テスト実施の並列分散処理

Aws lambda + Chrome Headless + Selenium によって実現



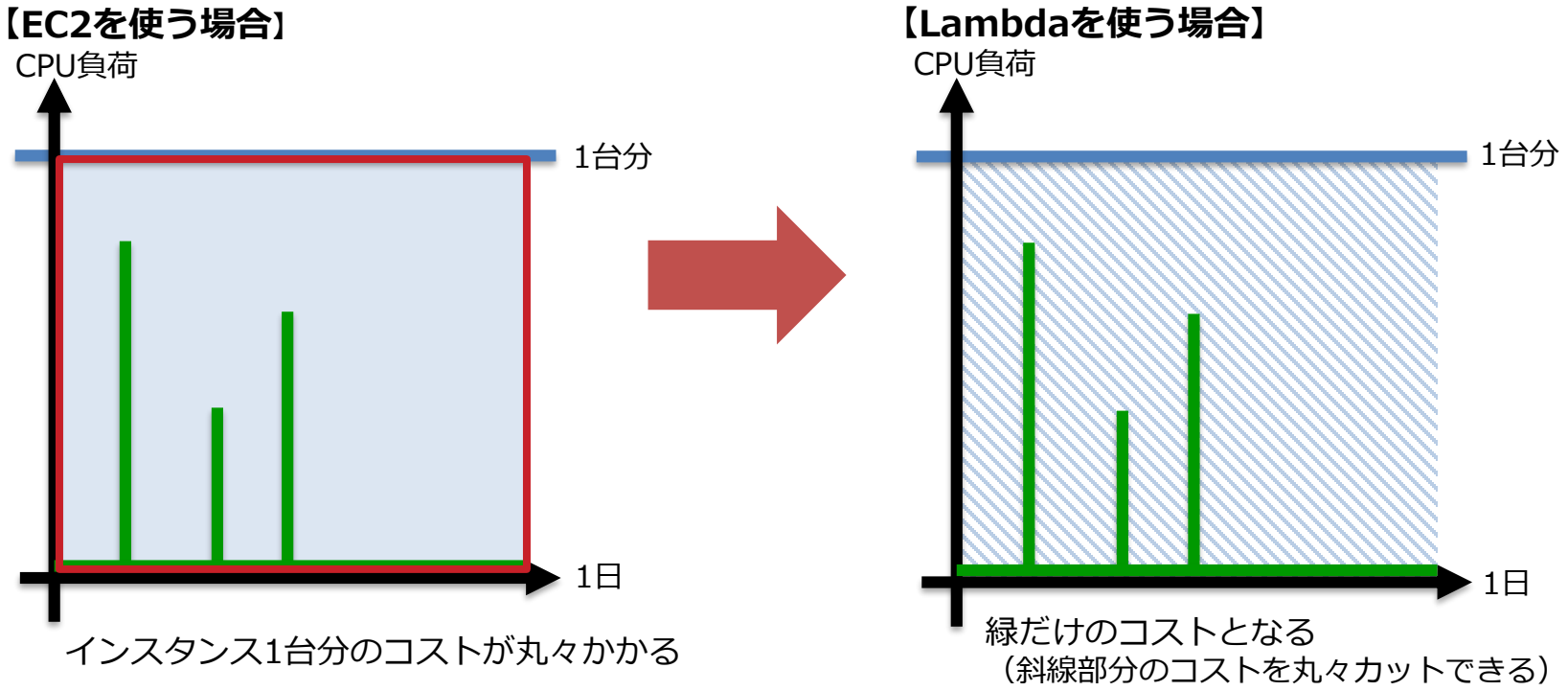
Aws lambda:サーバーレスによる実行サービス

Chrome Headless:表示部分がないchrome(高速で動作)

テスト実施の並列分散処理　サーバーレスとは

- クラウドサーバーではなく、クラウド実行関数として動作
- コール回数、専有メモリ＊時間による課金サービス

例）100万リクエスト$0.20



メリット	デメリット
・サーバー運用をする必要がない ・マイクロサービス（独立サービス）として活用できる	・ライブラリを自分で用意する ・メモリや容量の制限もある

テスト実施の並列分散処理　効果

Seleniumで
waitあり
(手動テスト再現)

20.8sec

Seleniumで
Waitなし

11.8sec

Lambda
+
headless

3.0sec

Lambda
100回
並列分散処理

? sec

テスト内容：ves ホームページ → アジャイル資料請求必要事項入力
→　必要事項の内容確認

テスト実施の並列分散処理　クイズ

クイズです！

DEMO

Lambda並列分散処理にて、テスト100回は、何秒になるでしょうか？

ヒント１：１回のテストは3.0sec
ヒント２：3.0*100回=300secです。
ヒント３：分散処理で多数アクセスすると、応答がおそくなります。
ヒント４：スレッド数　５０にて動かします

正解は、デモにて・・・・

テスト実施の並列分散処理　効果比較

Seleniumで
waitあり
(手動テスト再現)

20.8sec
×
100
=20,80sec
(34分)

Seleniumで
Waitなし

11.8sec
×
100
=1,180sec
(19分)

Lambda
+
headless

3.0sec
×
100
=300sec
(5分)

Lambda
100回
並列分散処理

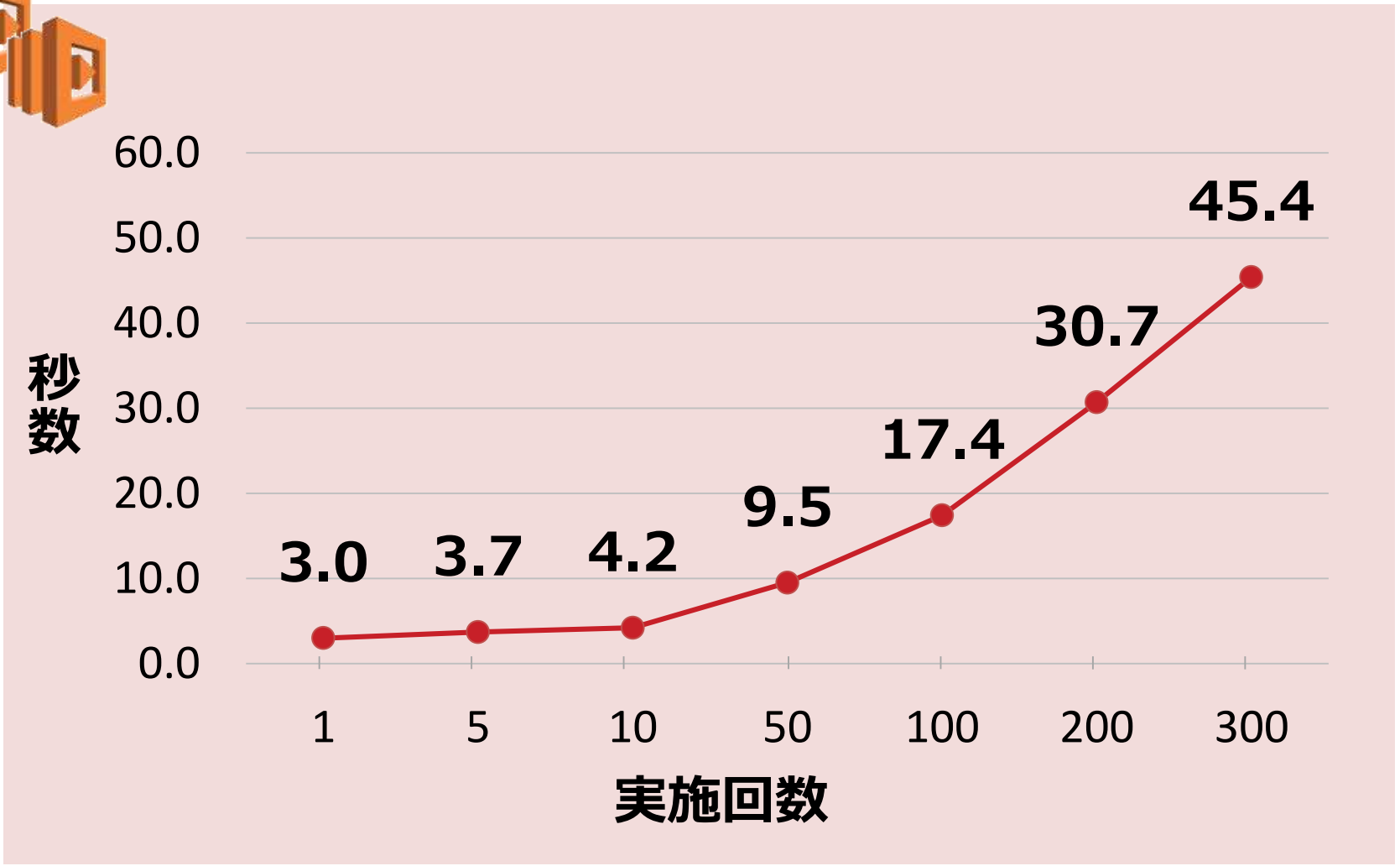
17.4sec

約1/120

約1/70

約1/20

並列分散処理の効果　<テスト回数別結果>



並列分散処理　価値

コストは同じだが、

価値は違う

120min

6min

まとめ

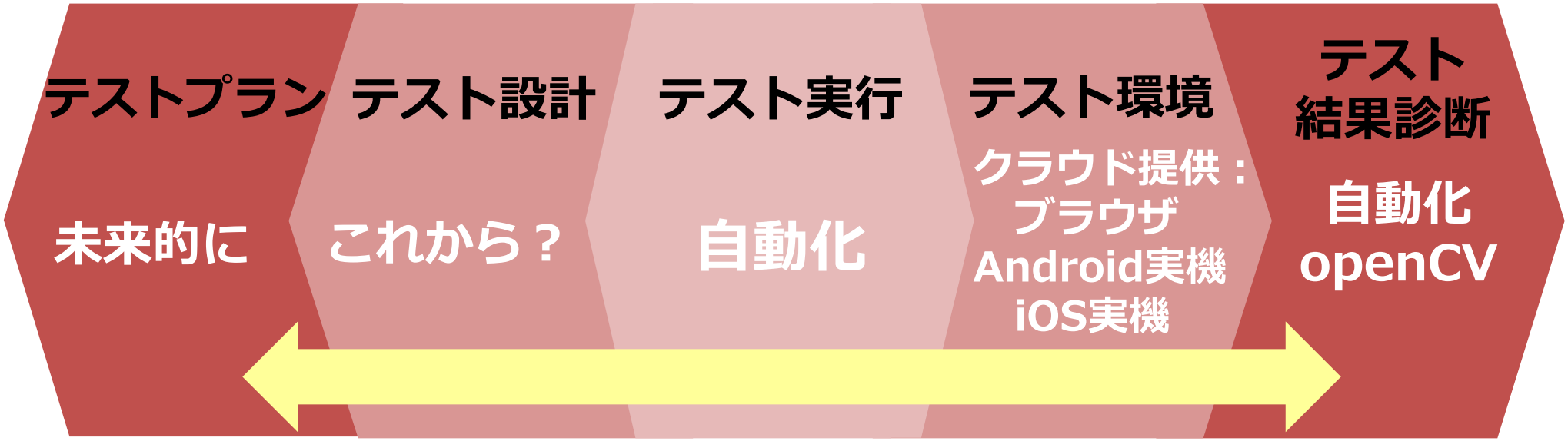
6．まとめ

- テスト自動化はリグレッション試験から
- 自動化率を継続的にあげていくプロジェクト成熟度向上を目指しましょう
- リグレッション試験を大きく成長させ、継続的インテグレーションでのテスト自動化と合わせて、継続的な品質確保に活用する

- 実端末操作をクラウドで　実機レスな時代
- Selenium 要素取得をAIで　メンテコスト小へ
- テスト実施の並列分散処理　実行時間縮小

6．まとめ

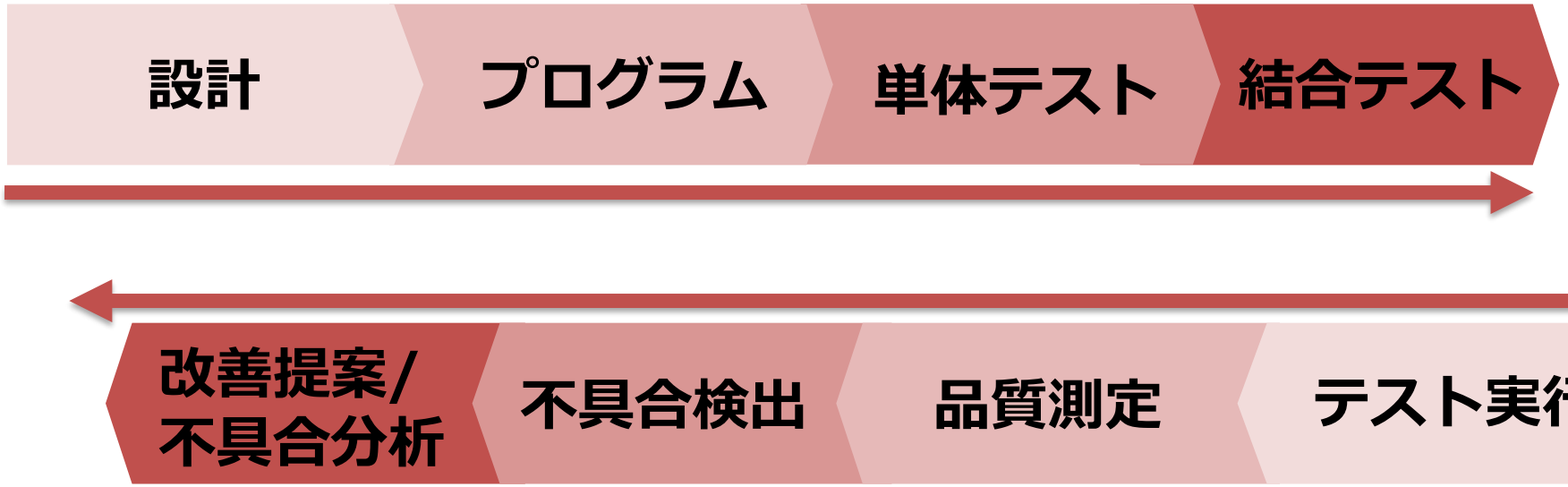
■自動化の未来



- 自動化の正体は、実施コストの**極小化**。
- 実施コストの極小化は、価値変動（シンギュラリティ）の可能性がある。
- テスト組み合わせパターン(ブラウザ＊ブラウザver＊OS*OSver）などを全通りできる可能性を秘めている。
- テスト設計が、要素とパターンを抽出して全実施の時代へ・・・

6．まとめ

開発の視点は、モノを作っていく行為（品質を作りたい）



テスト（検証）の視点は、モノの検査（動作）から品質測定して、
プログラムに影響を与える行為（品質を確認し、良くしたい）

自動化による実施コストの極小化は、テストにとって**価値が高い**
理由：モノを動かすことで品質測定できるため

6．まとめ

■自動化の目的　※更新版

主目的

・テスト実施　工数削減　➡ **実施コストの極小化**
・リグレッションテスト　工数削減　➡ **実施コストの極小化**
・人手がかからず実施できる　➡ **人手をかけない**
・属人化排除

副次効果

・継続的インテグレーションとしての連携　と　**実施即時性向上**
・開発サイクルの加速　➡ **テストの即時終了による高レスポンス**
・テスト網羅率の向上　➡ **極小化により網羅率が飛躍的に向上**
・メトリクス収集
（部分的に：実施精度向上、ログ収集力向上）

最後に

品質というキーワードで
世の中を幸せにし、
社会に貢献したいと考えています。

End