2014年9月　JaSST北海道

作る・流すだけじゃない！
テスト自動化を運用に乗せる為のもう一工夫

株式会社SHIFT 横田 真

SHIFT
Shift the inc.

AGENDA

1. 自己紹介
2. はじめに
3. 課題：作る・流すでは運用できないテスト自動化
4. 施策1：テストスクリプトのパーツ化
5. 施策2：テスト結果切り分け
6. 結び：成果・課題・今後の展望

AGENDA

1.自己紹介

2. はじめに

3. 課題：作る･流すでは運用できないテスト自動化

4. 施策1：テストスクリプトのパーツ化

5. 施策2：テスト結果切り分け

6. 結び：成果･課題･今後の展望

自己紹介

横田真
mail：makoto.yokota@shiftinc.jp

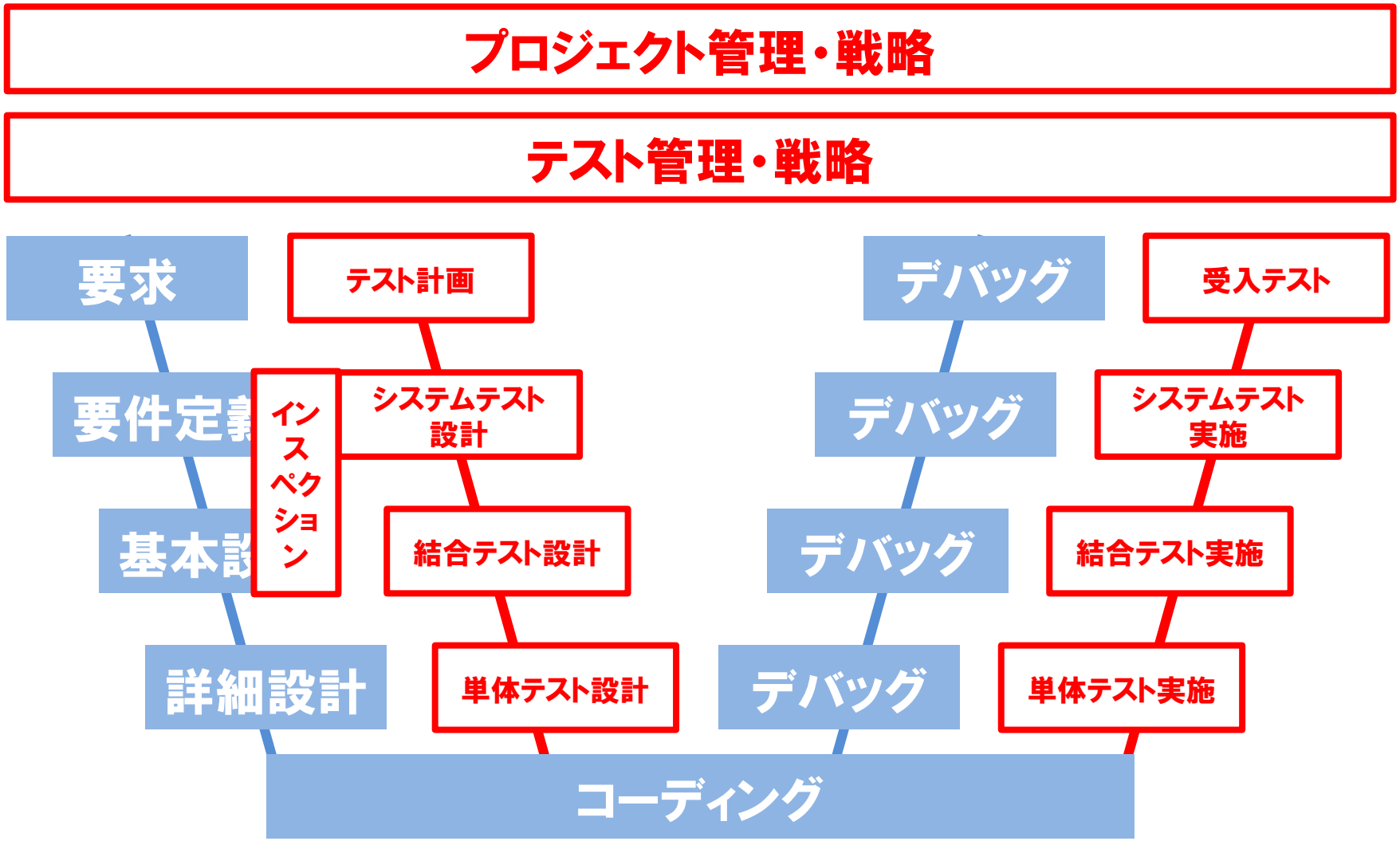
所属
株式会社SHIFT　ソフトウェアテスト事業部

職務内容
機能テスト設計～シナリオテスト設計
及びテスト実行管理を担当

会社概要

社名		株式会社SHIFT	
拠点	本社	〒 106-0041　東京都港区麻布台2-4-5 メソニック39MTビル12F	
	札幌	〒 060-0001北海道札幌市中央区北1条西3丁目2井門札幌ビル8F	
	福岡	〒 810-0001　福岡県福岡市中央区天神1-15-6綾杉ビル8F	
	インド・プネ	6F,Pride Icon, Mundhwa Bypass Road, Kharadi, Pune - 411014.	
	シンガポール	24 Peck Seah Street #04-03 Nehsons Building	
従業員数		400名 (2014年3月現在)　※契約社員・パート含む	
交通アクセス		神谷町駅 1番出口より　徒歩5分	
電話番号		03-6809-1128 (代表)	
FAX		03-6809-1197	
URL		http://www.shiftinc.jp/	
代表取締役		丹下　大	
相談役		加藤　隆哉	サイバード　元代表　ミドクラ　代表取締役
		海野　恵一	アクセンチュア　元代表　スウィングバイ2020　代表取締役

SHIFTのテスト領域



AGENDA

1. 自己紹介

2.はじめに

3. 課題：作る・流すでは運用できないテスト自動化

4. 施策1：テストスクリプトのパーツ化

5. 施策2：テスト結果切り分け

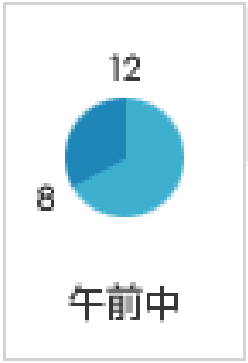
6. 結び：成果・課題・今後の展望

統合型ECサイト構築パッケージ評価案件に参画中

E-Commerce(EC)のパッケージソフト

ECサイトの構築・運営業務をサポート

※E-Commerce：電子商取引

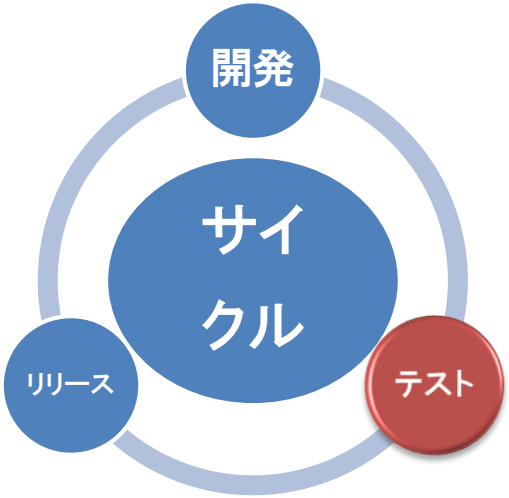


統合型ECサイト構築パッケージ評価案件に参画中

バックエンド・フロントエンドのテスト全般（設計・実行・自動化）を担当

⇒SeleniumIDEの自動テスト作成をご依頼頂き作業中

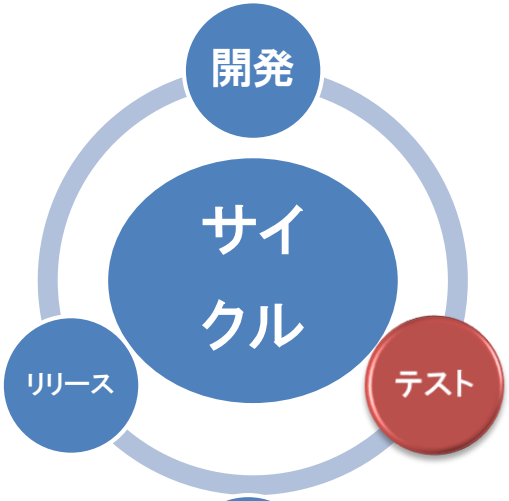
短期間開発サイクルに対応したテスト実施を迫られている



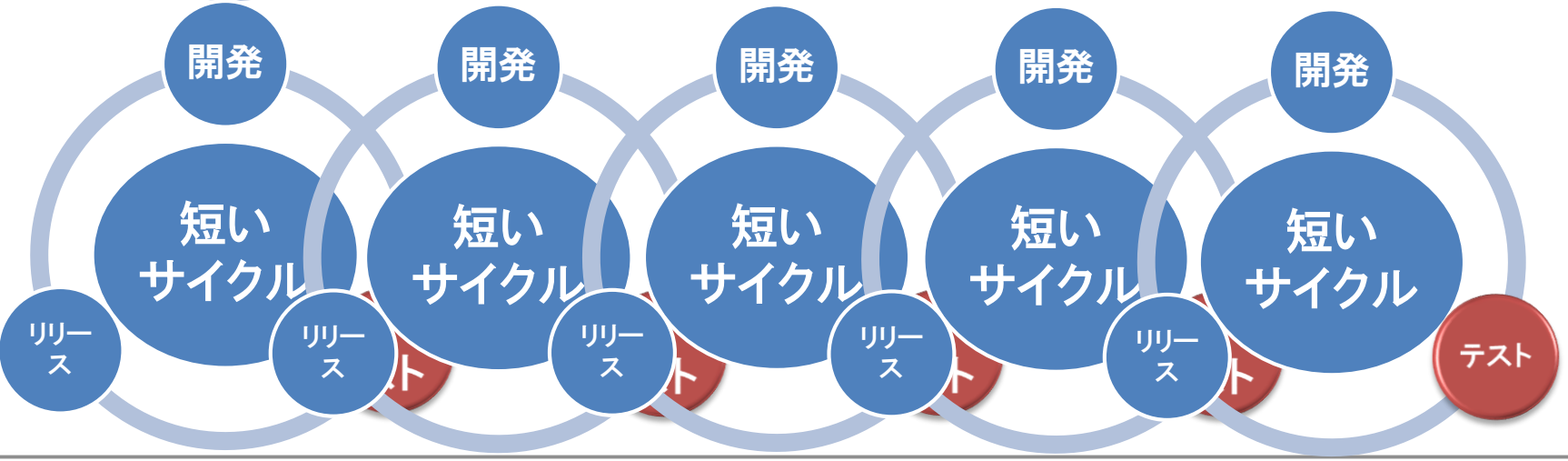
最近の開発体制の特徴
- 開発期間が短い
- 複数案件が同時に進行
- 機能数は増加

2.はじめに：ソフトウェア開発の現状

短期間開発サイクルに対応したテスト実施を迫られている

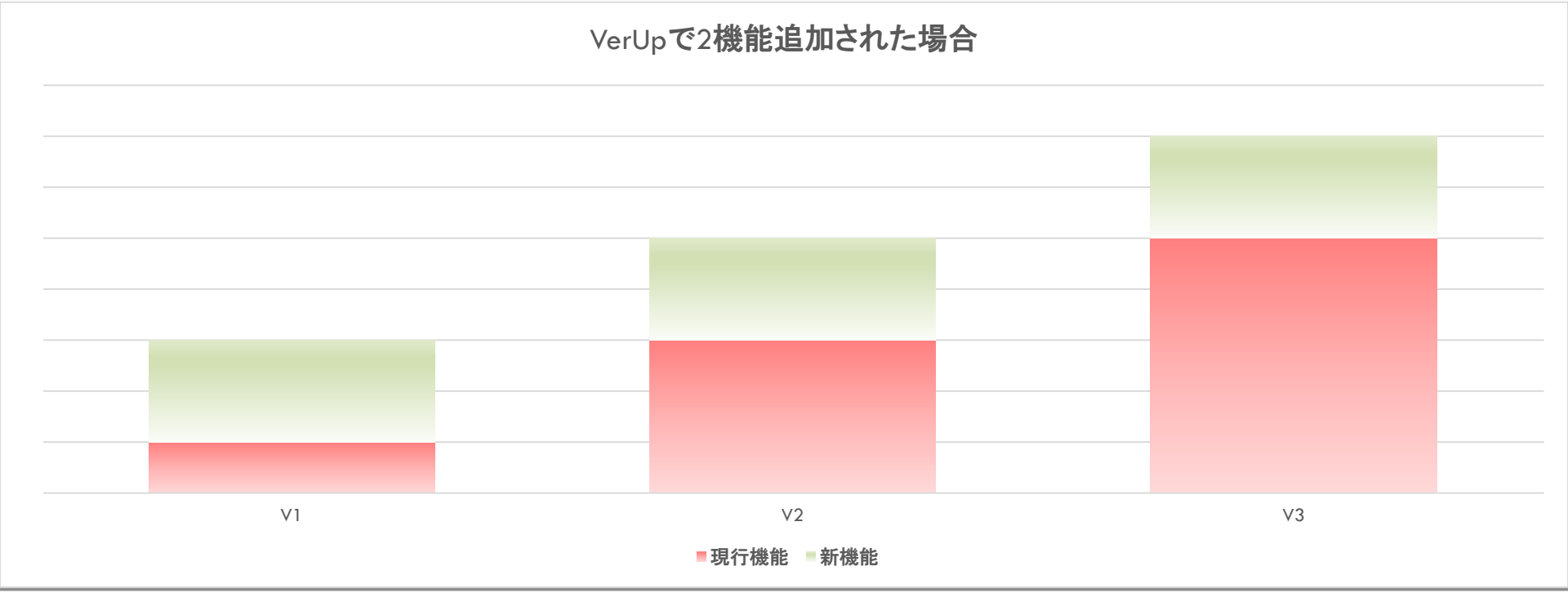


最近の開発体制の特徴
- 開発期間が短い
- 複数案件が同時に進行
- 機能数は増加



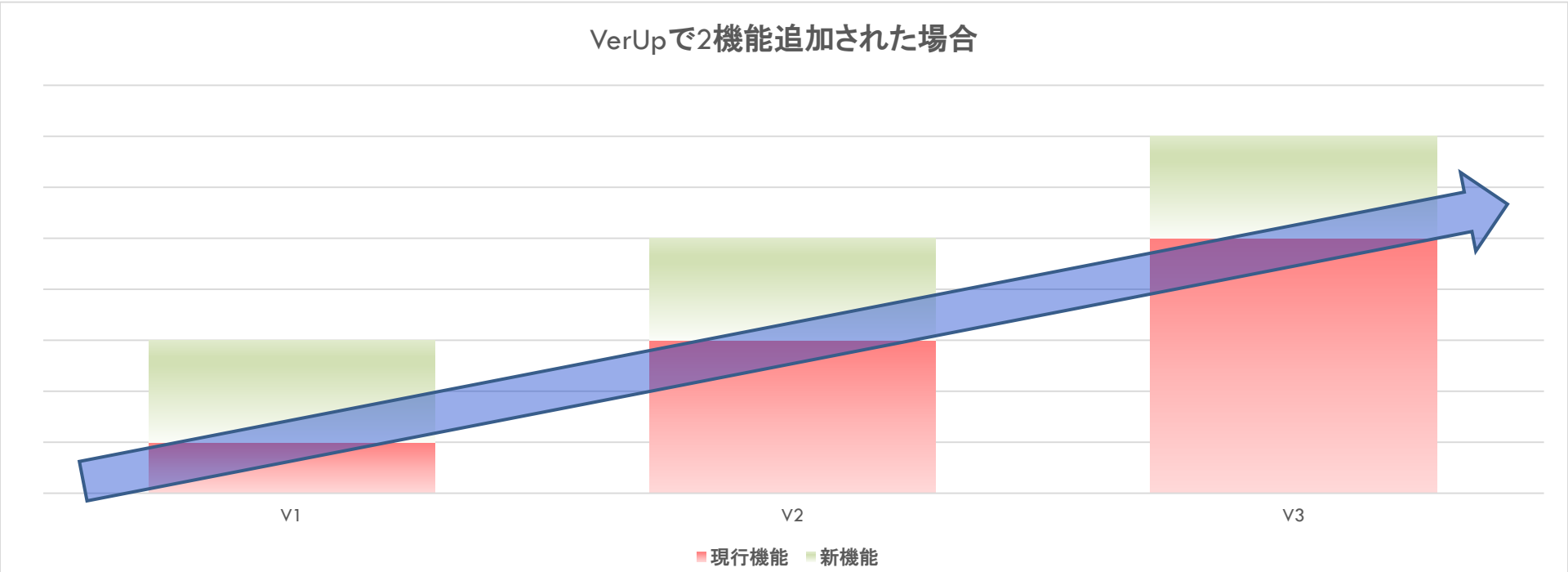
テスト対象機能は増え続ける

前バージョンの機能も保証
→ Version Up毎に対象機能は増える



テスト対象機能は増え続ける

前バージョンの機能も保証
→ Version Up毎に対象機能は増える



回帰テストを自動化してテスト工数を確保する

保守機能増加＝回帰テスト工数増加

⇒回帰テスト工数削減に向けて
品質管理チームは『テスト自動化』
を進めている

AGENDA

1. 自己紹介

2. はじめに

3.課題：作る･流すでは運用できない テスト自動化

4. 施策1：テストスクリプトのパーツ化

5. 施策2：テスト結果切り分け

6. 結び：成果･課題･今後の展望

3つのステップ：作成・運用・展開

・テストスクリプトを**作成**する
　→　実装作業

・テスト自動化を**運用**に乗せる
　→　実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
　→　カバレッジの向上

3つのステップ：作成・運用・展開

テストスクリプトを**作成**する
→　実装作業

・テスト自動化を**運用**に乗せる
→　実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
→　カバレッジの向上

ツール：Selenium IDEで作業

・FirefoxのAddOn
　→　環境構築が容易

・テストの記録、編集、デバッグを行う
　→　技術習得面のイニシャルコスト低

Selenium IDE：Suiteファイル

- Suiteファイル　Caseファイルのリンク集
  - Caseファイル
  - Caseファイル
  - Caseファイル

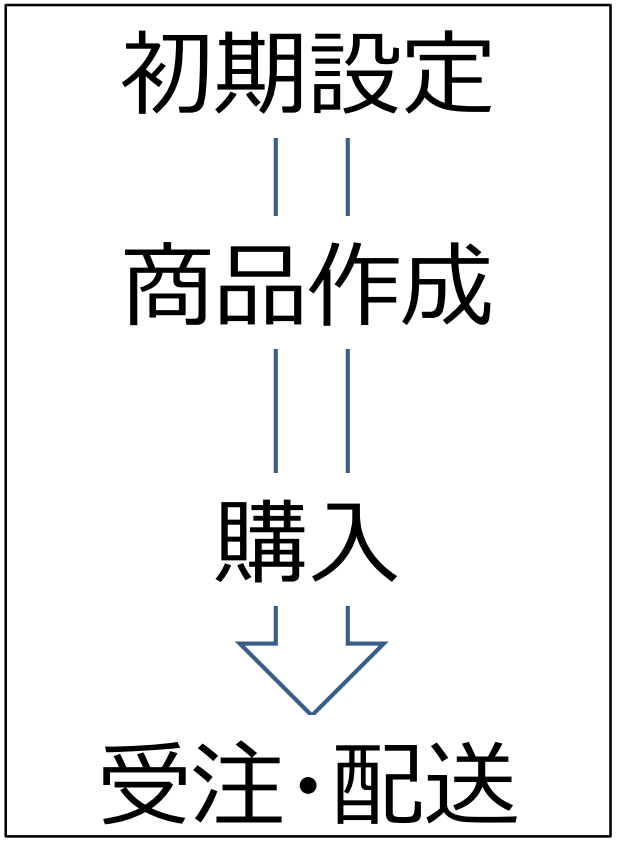
3.作る・流すでは運用できないテスト自動化：手段

Selenium IDE：Caseファイル

Caseファイル　テストスクリプトが記載

コマンド	対象	値
clickAndWait	link=加湿機用洗剤	
waitForPageToLoad	60000	
storeText	css=h2	Goods
getEval	storedVars.Goods_1_Nm = storedVars.Goods.substring(0,storedVars.Goods.ind...	
getEval	storedVars.Goods_1_Code = storedVars.Goods.substring(storedVars.Goods.ind...	
storeText	id=EC_sumPrice	Goods_1_Price
storeEval	storedVars.tax =${Goods_1_Price} - (Math.round((${Goods_1_Price} / 1.08)))	Goods_1_Tax
storeEval	${Goods_1_Price} - ${Goods_1_Tax}	Goods_1_NoTaxPrice
storeEval	insertComma(storedVars.Goods_1_Price)	Goods_1_PriceText
storeEval	storedVars.Goods_1_PriceText.replace(",","")	Goods_1_PriceData
storeEval	insertComma(storedVars.Goods_1_NoTaxPrice)	Goods_1_NoTaxPriceText
storeEval	insertComma(storedVars.Goods_1_Tax)	Goods_1_TaxText
storeEval	storedVars.Goods_1_TaxText.replace(",","")	Goods_1_TaxData
storeEval	Math.floor(storedVars.Goods_1_PriceData/100)	Goods_1_Point
verifyText	id=EC_sumPriceArea	${Goods_1_PriceText}円（税込）
verifyText	css=div.point	ショッピングポイント：${Goods
verifyElementPresent	css=select.CMP_SimCart_simCartCount	
select	css=select.CMP_SimCart_simCartCount	${Goods_1_Quantity}
verifyElementPresent	css=img[alt="カートにいれる"]	
click	css=img[alt="カートにいれる"]	
waitForPageToLoad	60000	

今回自動化したシナリオ概要



シナリオ例

他にも・・・

会員登録・変更
決済方法
ポイント利用購入
クーポン
ユーザランク割引

⇒400本のテストシナリオ

3つのステップ：作成・運用・展開

・テストスクリプトを**作成**する
　→　実装作業

・テスト自動化を**運用**に乗せる
　→　実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
　→　カバレッジの向上

3つのステップ：作成・運用・展開

・テストスクリプトを**作成**する
　→　実装作業

テスト自動化を**運用**に乗せる
→　実行・メンテナンス・結果のフィードバック

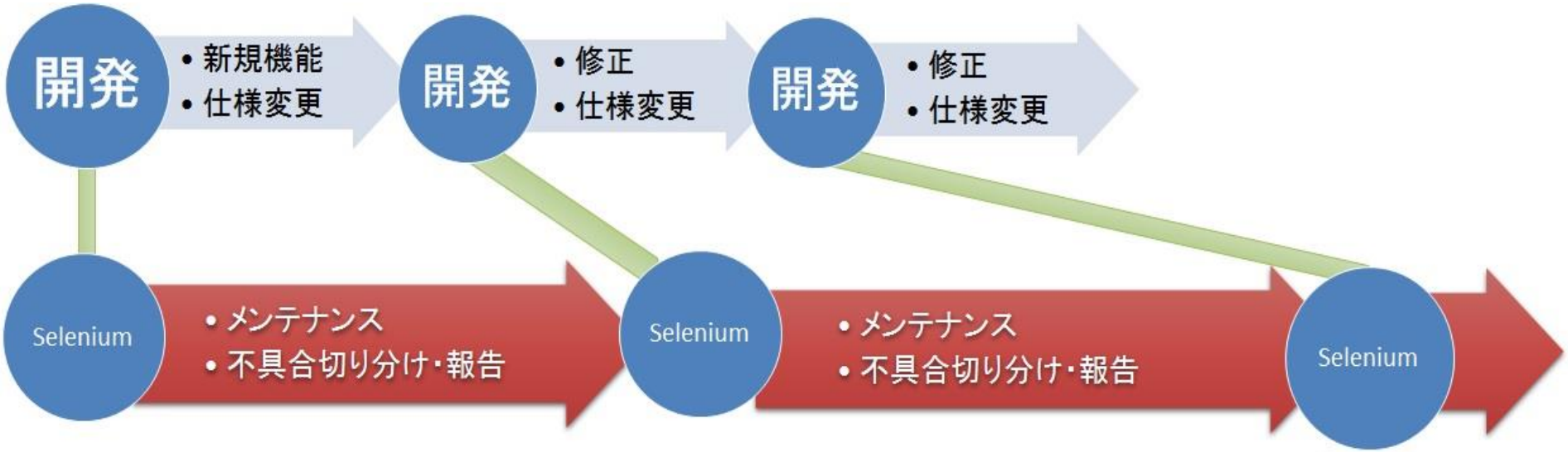
・テストシナリオを**展開**する
　→　カバレッジの向上

実導入

実際にテスト実行してみましたが・・・

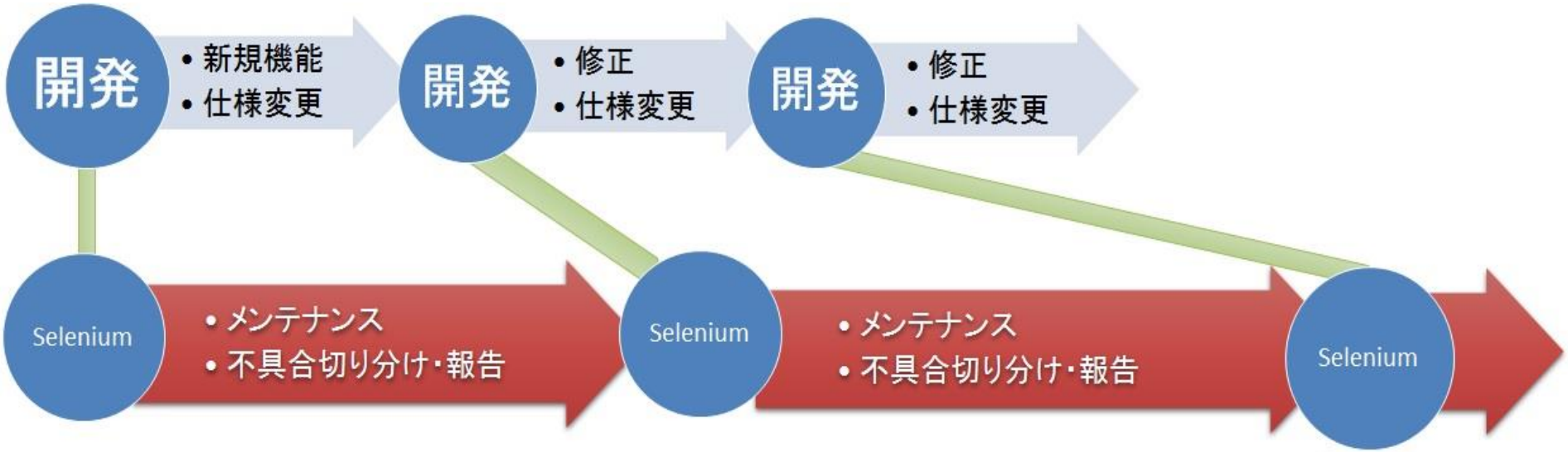
3.作る・流すでは運用できないテスト自動化：スピード不足

開発サイクルのスピード・テストスクリプトメンテナンスのスピード



3.作る・流すでは運用できないテスト自動化：スピード不足

開発サイクルのスピード ＞ メンテスピード・結果切り分け



開発サイクルに
テストサイクルが追い付かない

3.作る・流すでは運用できないテスト自動化：課題1

メンテナンス工数の不足

- テストスクリプトのメンテナンス

機能追加・仕様変更

→ スクリプト修正工数の不足

3.作る・流すでは運用できないテスト自動化：課題2

メンテナンス工数の不足

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → スクリプト修正工数の不足
- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → 切り分けに時間が掛かる

3.作る・流すでは運用できないテスト自動化：対応課題の決定

解決すべき課題

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → **スクリプト修正工数の不足**
- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → **切り分けに時間が掛かる**

⇒課題を解決し運用に乗せる

AGENDA

1. 自己紹介

2. はじめに

3. 課題：作る･流すでは運用できないテスト自動化

4.施策1：テストスクリプトの共通パーツ化

5. 施策2：テスト結果切り分け

6. 結び：成果･課題･今後の展望

解決すべき課題

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → **スクリプト修正工数の不足**
- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → **切り分けに時間が掛かる**

解決すべき課題

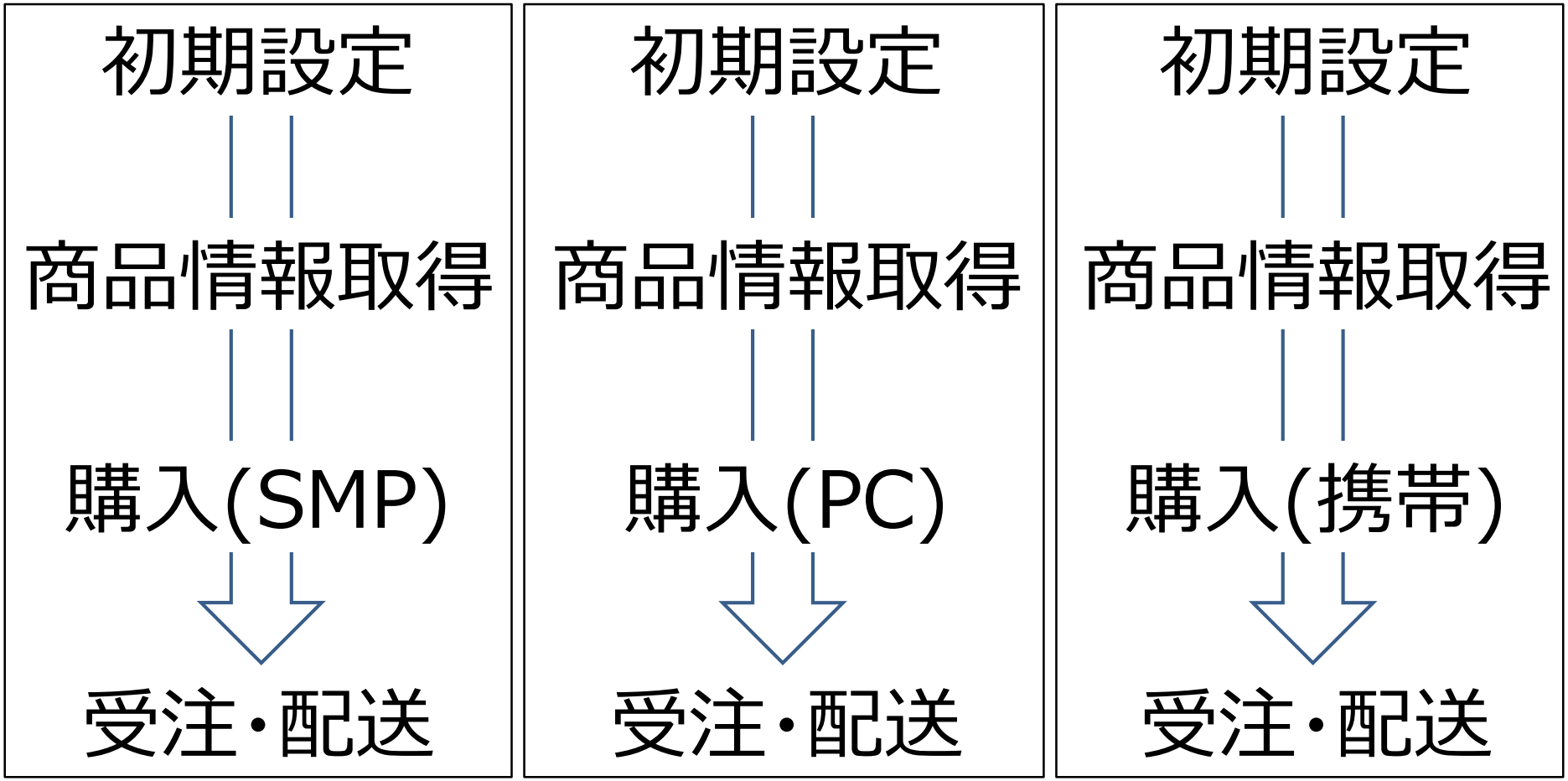
- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → **スクリプト修正工数の不足**
- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → **切り分けに時間が掛かる**

共通パーツ化

共通パーツ化？

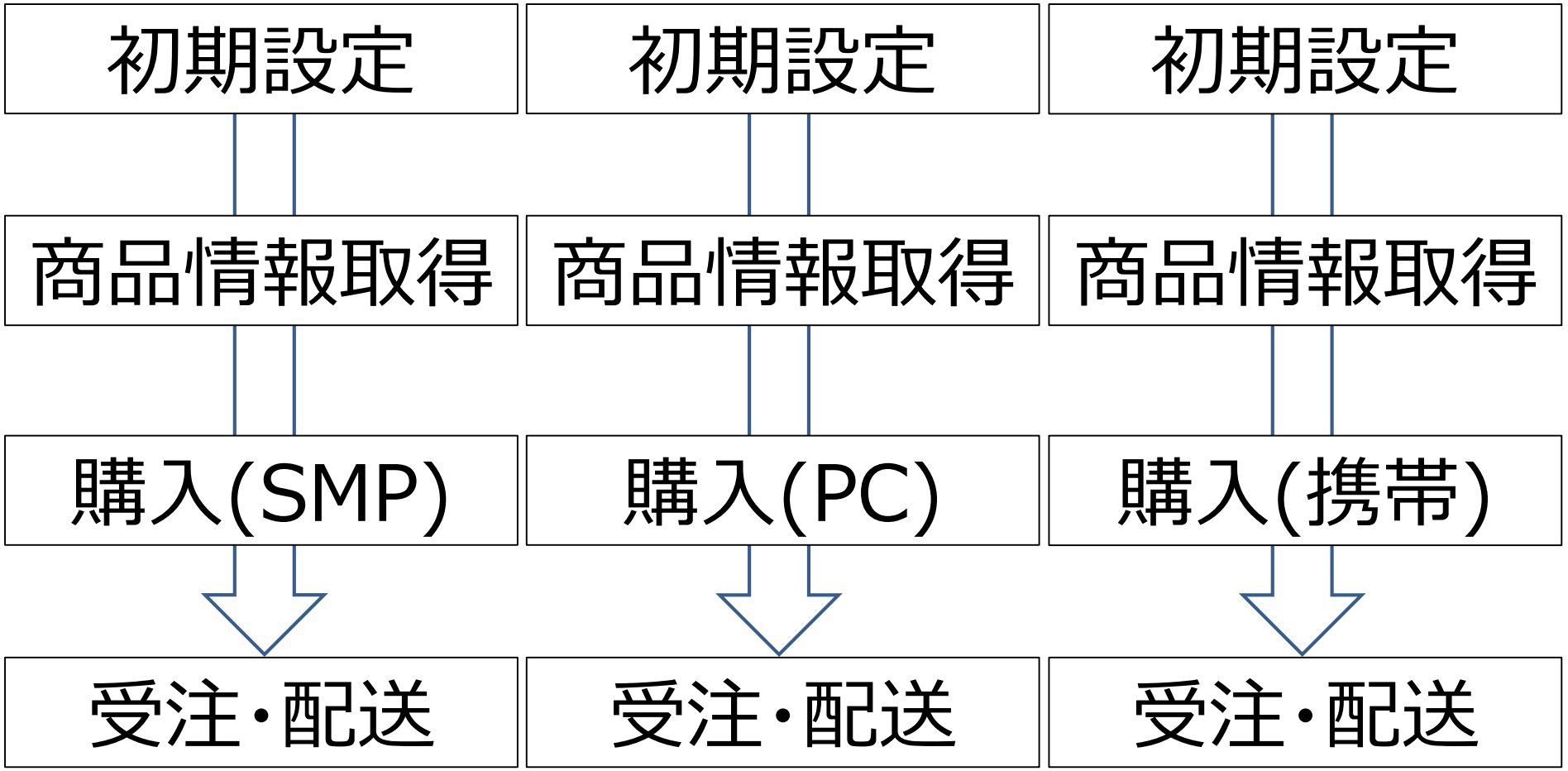
4.施策1：テストスクリプトの共通パーツ化

施策前のシナリオ状態



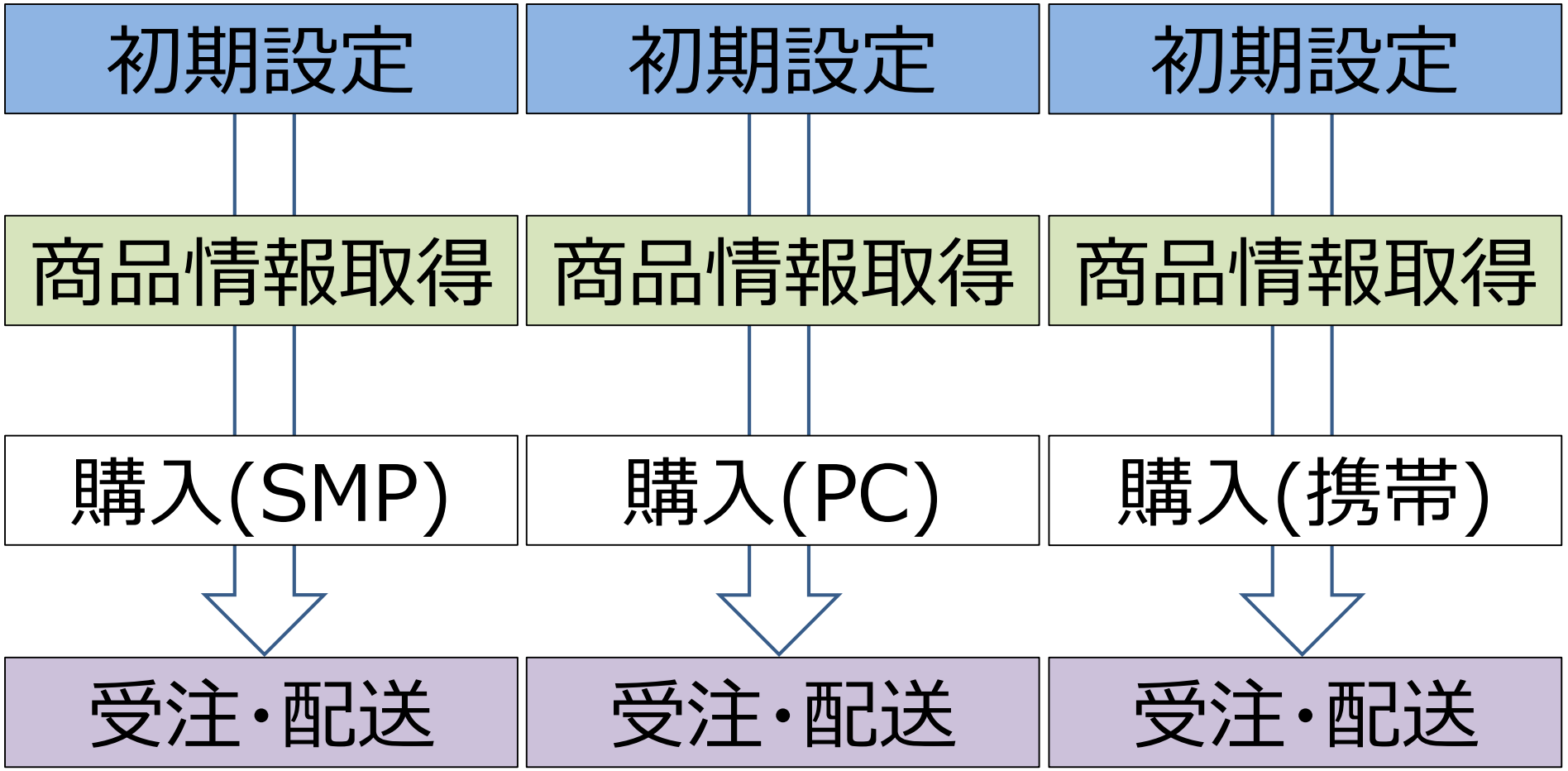
4.施策1：テストスクリプトの共通パーツ化

いくつかの処理に分けて考える



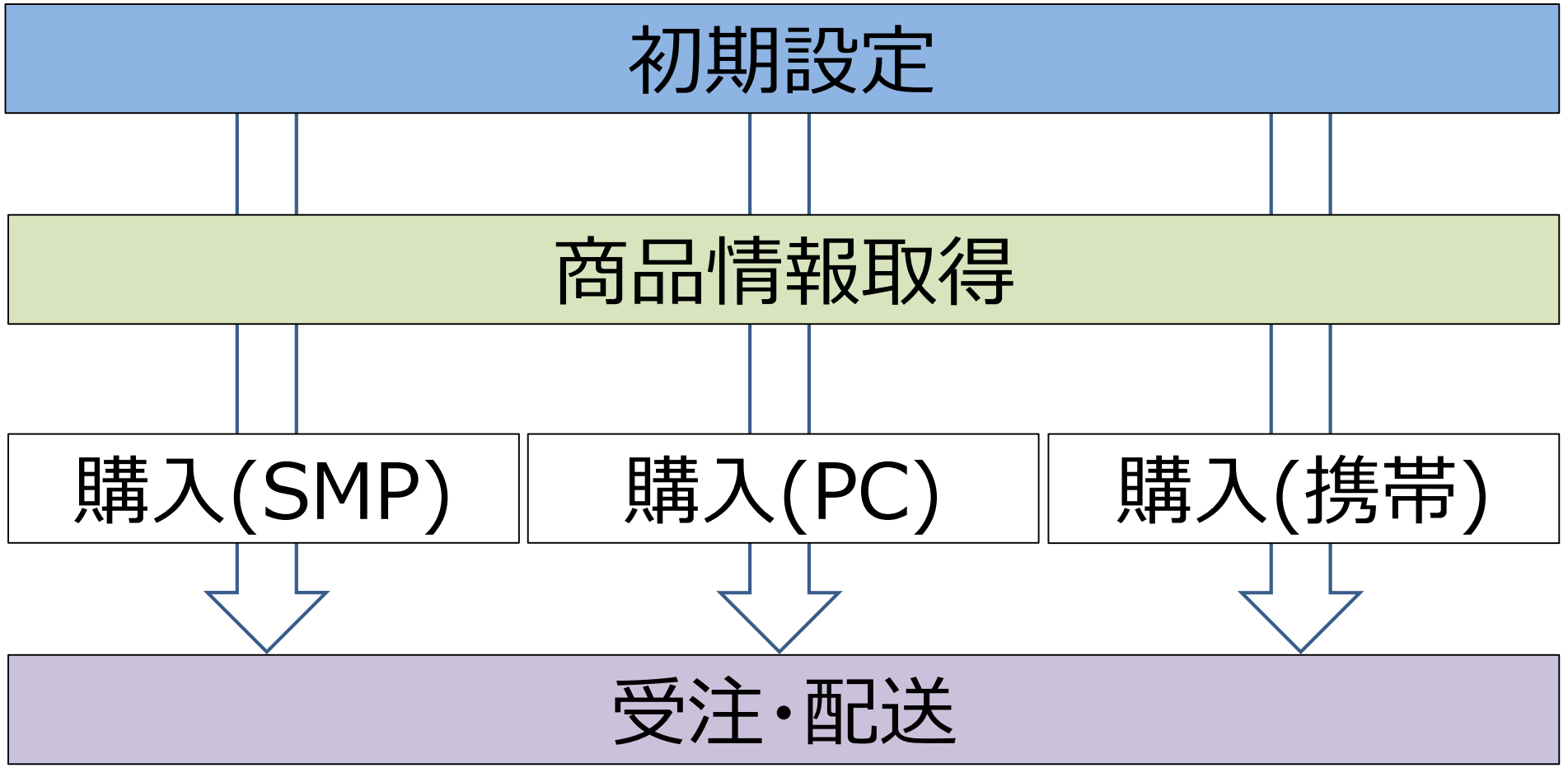
4.施策1：テストスクリプトの共通パーツ化

同じ処理をしている個所を調査



4.施策1：テストスクリプトの共通パーツ化

共通パーツ化：パーツを他シナリオと共有・流用



実作業開始

共通パーツ化作業開始

4.施策1：テストスクリプトの共通パーツ化

重要な機能確認を行うシナリオに絞って実施

自動化済みのシナリオは**400**本
→　**105**本をまずは施策対象に

4.施策1：テストスクリプトの共通パーツ化

重要な機能確認を行うシナリオに絞って実施

自動化済みのシナリオは**400**本
→　**105**本をまずは施策対象に
(選定基準)
ボトルネックになる処理を多く含む
重要なシナリオ

4.施策1：テストスクリプトの共通パーツ化

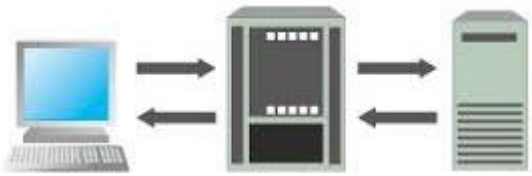
重要な機能確認を行うシナリオに絞って実施

自動化済みのシナリオは**400**本
→　**105**本をまずは施策対象に
(選定基準)
ボトルネックになる処理を多く含む
重要なシナリオ
→今後残りのシナリオでも再利用性の
　高い共通パーツが出来ると見込み

ECサイトで特に重要な機能

・金額計算機能
　小計・税金・割引計算

・フロントエンド～バックエンドの
　データ連携機能
　商品情報
　ユーザランク
　発注完了



4.施策1：テストスクリプトの共通パーツ化

機能を洗い出してから共通パーツ化作業を行う

- 多くのシナリオで使用すると思われる機能を洗い出し

- カテゴリーに分けたシナリオ内で共通する機能の洗い出し

- 環境(PC・スマートフォン)間の共通処理

共通パーツ化に向かない処理

- 使用回数の少ない機能

- 可変IDが使用されている処理

共通パーツ化に向かない処理

- 使用回数の少ない機能

- 可変IDが使用されている処理
  ※可変ID
  オブジェクトIDが画面内で
  動的に設定される構成

4.施策1：テストスクリプトの共通パーツ化

同じ処理をまとめ、パーツ化する機能を決定

1,672パーツ667パーツが同じ処理

パーツ名称	使用回数
00_00.業務シナリオ共通_store	89
00_00.業務シナリオ共通_会員削除BACK	28
00_00.業務シナリオ共通_会員登録PC	25
00_00.業務シナリオ共通_作成商品削除	6
00_00.業務シナリオ共通_受注 配送フロー	56
00_00.業務シナリオ共通_受注一覧確認(受注詳細･配送詳細)	31
00_00.業務シナリオ共通_出荷確認	24
00_00.業務シナリオ共通_商品購入_BACK	1
00_00.業務シナリオ共通_商品購入後確認_PC	40
00_00.業務シナリオ共通_商品購入後確認1(返品前)	1
00_00.業務シナリオ共通_商品購入後確認2(返品前)	3
00_00.業務シナリオ共通_商品情報ストア	52
00_00.業務シナリオ共通_商品情報ストア2(購入前)	2
00_00.業務シナリオ共通_通常商品カテゴリ設定	9
00_00.業務シナリオ共通_通常商品購入_BACK	9
00_00.業務シナリオ共通_通常商品購入_PC	16
00_00.業務シナリオ共通_通常商品在庫設定	10
00_00.業務シナリオ共通_通常商品登録	10

シナリオ共通パーツ使用数

4.施策1：テストスクリプトの共通パーツ化

実作業

共通パーツ化

Selenium IDE：共通パーツファイル作成

Caseファイルa

Caseファイルb

Caseファイルc

Selenium IDE：共通パーツファイル作成

Caseファイルa	→	共通パーツa
Caseファイルb	→	共通パーツb
Caseファイルc	→	共通パーツc

Selenium IDE：Suiteファイル変更

- Suiteファイル　Caseファイルのリンク集
  - Caseファイルa
  - Caseファイルb
  - Caseファイルc

4.施策1：テストスクリプトの共通パーツ化

Selenium IDE：Suiteファイル変更

Suiteファイル　Caseファイルのリンク集

- Caseファイルa → 共通パーツa
- Caseファイルb → 共通パーツb
- Caseファイルc → 共通パーツc

4.施策1：テストスクリプトの共通パーツ化

パーツ数を40%削減

1,672パーツ　→　1,005パーツに削減

修正対象大幅削減

→修正時間短縮

→修正漏れ防止

解決すべき課題

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → **スクリプト修正工数の不足**
- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → **切り分けに時間が掛かる**

AGENDA

1. 株式会社SHIFT
2. はじめに
3. 課題：作る･流すでは運用できないテスト自動化
4. 施策1：テストスクリプトの共通パーツ化
5. 施策2：テスト結果切り分け
6. 成果･課題･今後の展望

解決すべき課題

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → スクリプト修正工数の不足

解決

- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → **切り分けに時間が掛かる**

解決すべき課題

- テストスクリプトのメンテナンス
  機能追加・仕様変更
  → スクリプト修正工数の不足

解決

- テスト結果切り分け
  機能不具合・スクリプトミス・環境要因
  → 切り分けに時間が掛かる

5.施策2：テスト結果切り分け

テストスクリプトがエラーとなった個所をリスト化

テストスクリプトが止まる(エラーとなる)原因

・テストスクリプト起因のエラー
・ソフトウェア起因のエラー
→テスト実行者が判断できるもの
開発が調査しないと判断できないもの

⇒過去実績を分析し、リスト化し
開発チームと内容を協議

5.施策2：テスト結果切り分け

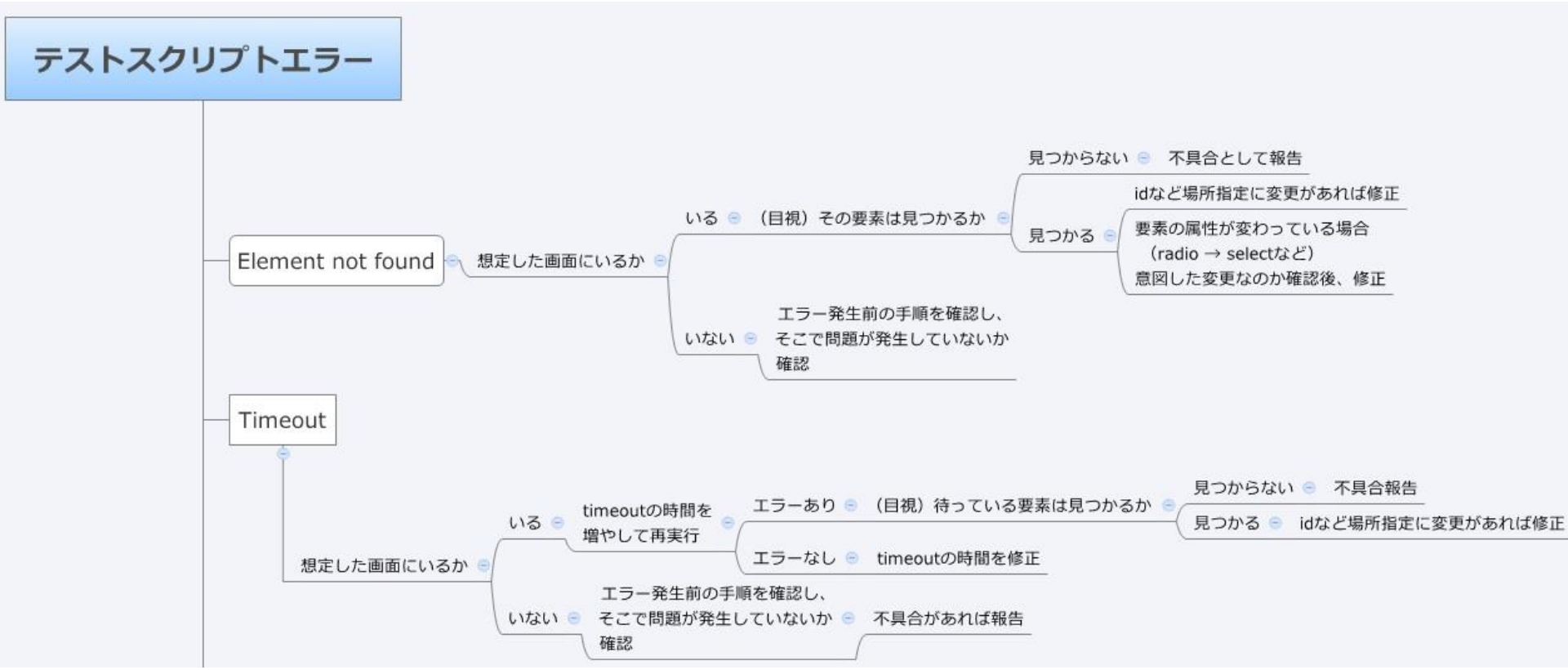
開発側とエラー発生時の対応切り分けの線引きをしておく

テスト実施者がどこまで判断するか定義

→テスト実施者の調査時間短縮
開発担当者の切り分け時間短縮

5.施策2：テスト結果切り分け

フローチャートを準備し不具合切り分けをわかりやすく

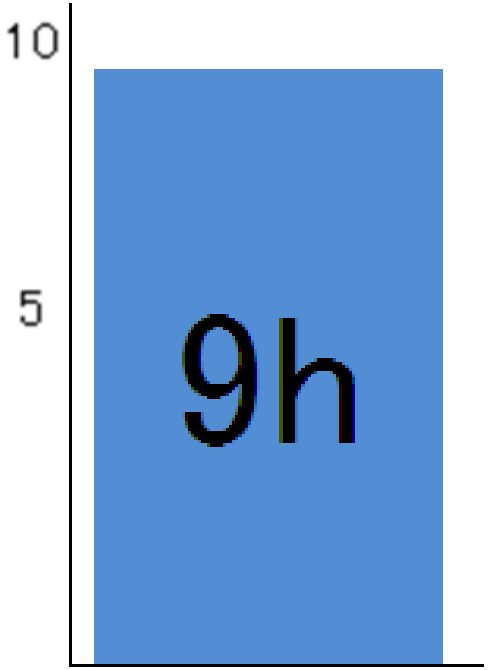


AGENDA

1. 株式会社SHIFT
2. はじめに
3. 背景
4. テスト自動化へのステップ
5. 施策1：テストスクリプトのパーツ化
6. 施策2：テスト結果切り分け
7. 成果・課題・今後の展望

成果：2つの施策で修正工数・切り分け工数圧縮

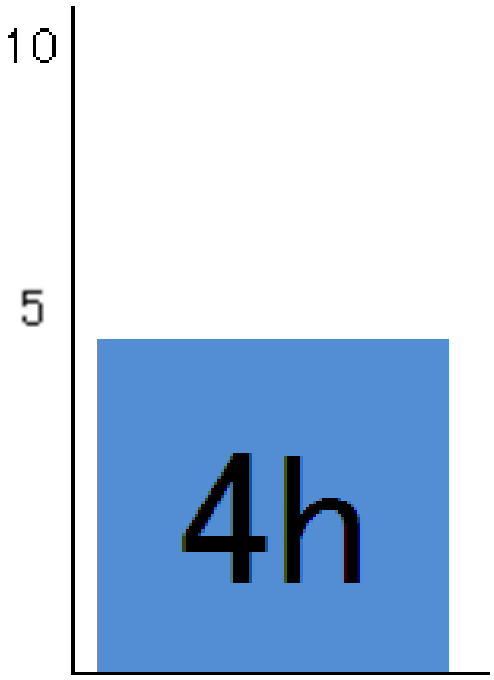
メンテナンス時間



成果：2つの施策で修正工数・切り分け工数圧縮

メンテナンス時間

55%圧縮／1シナリオ



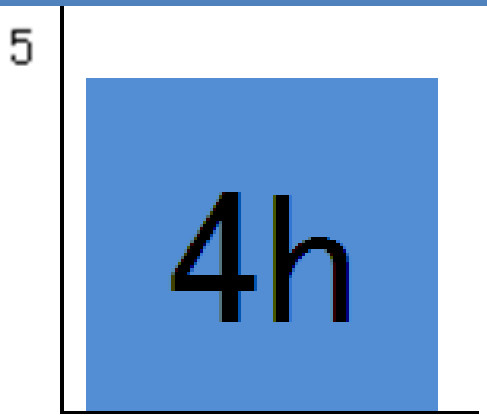
成果：2つの施策で修正工数・切り分け工数圧縮

メンテナンス時間

55%圧縮／1シナリオ

9h×105本＝945h(≒6人月)

→4h×105本＝420h(≒2.8人月)



展望：シナリオの拡張

・テストスクリプトを**作成**する
　→　実装作業

・テスト自動化を**運用**に乗せる
　→　実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
　→　カバレッジの向上

展望：シナリオの拡張

・テストスクリプトを**作成**する
　→　実装作業

・テスト自動化を**運用**に乗せる
　→　実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
　→　カバレッジの向上

展望1：シナリオの拡張

・テストスクリプトを**作成**する
→ 実装作業

・テスト自動化を**運用**に乗せる
→ 実行・メンテナンス・結果のフィードバック

・テストシナリオを**展開**する
→ カバレッジの向上

展望2：WebDriver化

Selenium IDE → WebDriver

・携帯電話の実機での自動実行
・OSの違い・OSのVer.の違い・ブラウザの違い
→シミュレータでは難しい確認ができる

ご清聴 誠にありがとうございました。