テスト自動化の前に押えておきたい3つのポイント

2011年11月25日
日本ヒューレット・パッカード株式会社

HP SOFTWARE

ソフトウェアテストシンポジウム 2011 九州 JaSST'11 Kyushu
JaSST'11 Kyushu : Japan Symposium on Software Testing in Kyushu 2011
2011年11月25日[金] 会場 / 福岡システムLSI総合開発センター

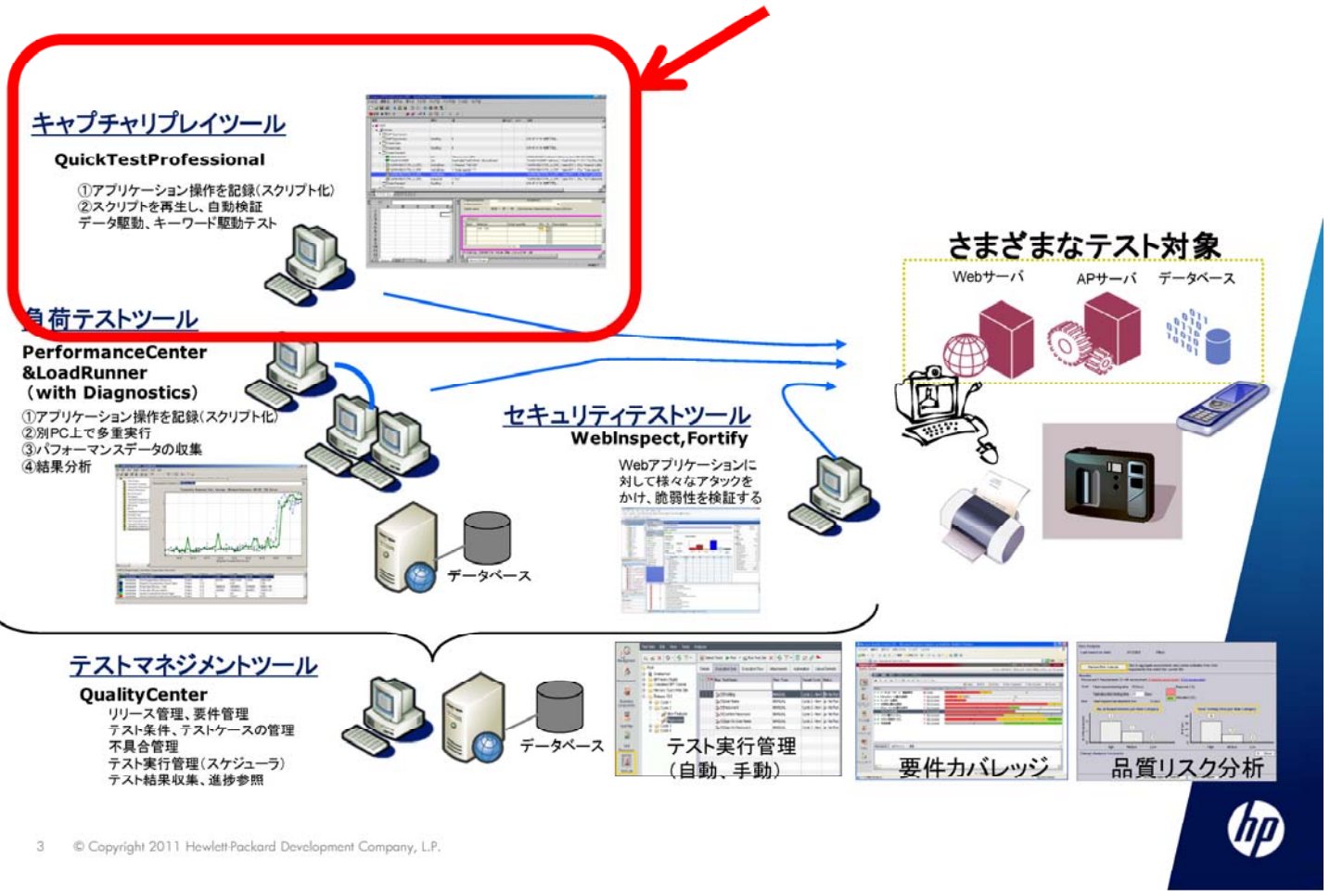
自己紹介

湯本 剛

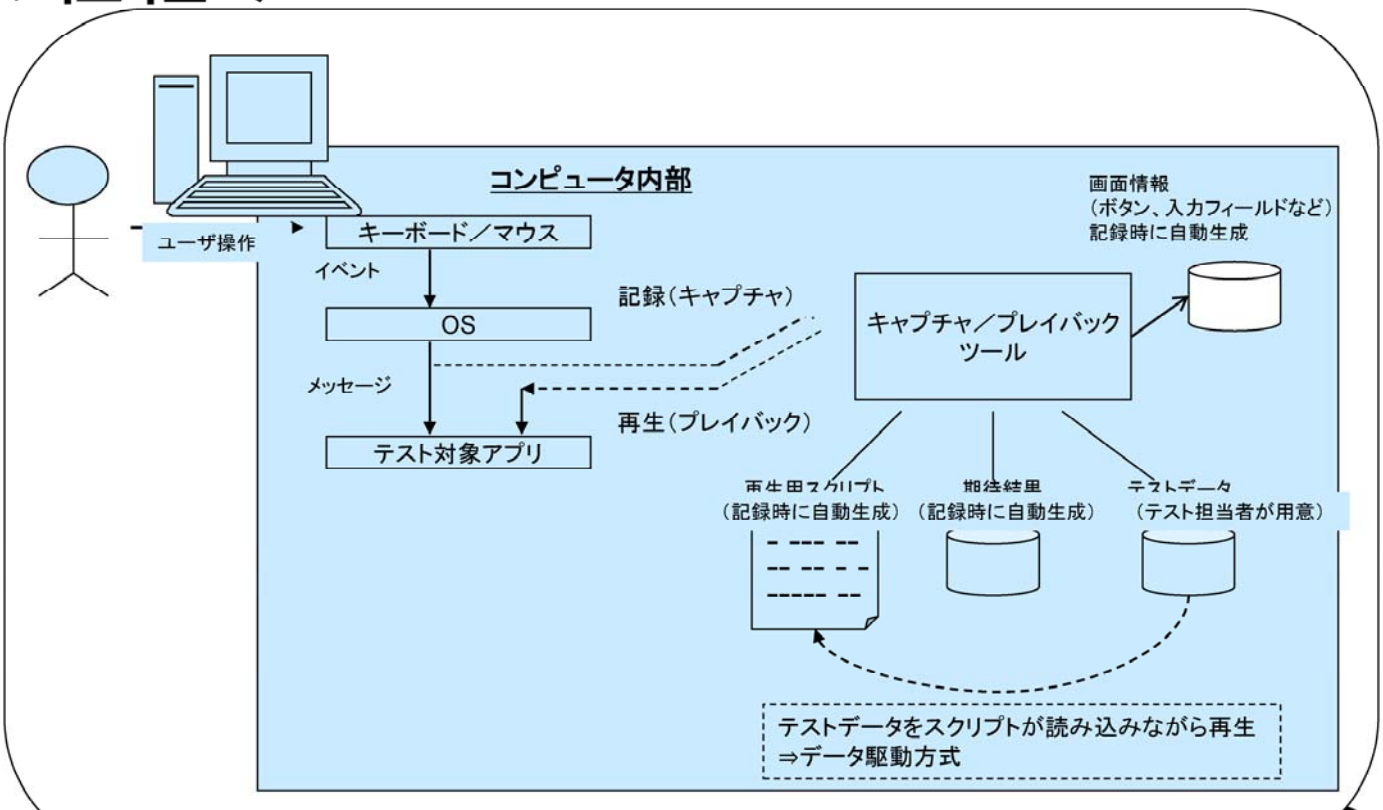
Consultant at HP Japan

- テスト現場で10年ほど経験を積んだ後、テストプロセス改善のコンサルティング、教育に約7年ほど従事
- 2010年8月よりHP Softwareのテストツール導入支援コンサルタントに転職し、主にテストケース管理ツール、キャプチャリプレイツールの導入支援に従事
- 外部活動
  - NPO法人ASTER　理事
    - JSTQB 技術委員
    - ISTQB CTAL WGメンバー
    - JaSST東京実行委員
  - ISO/IEC JTC1 SC7 WG26 エキスパート
  - 日科技連SQiPステアリング委員
- 書籍、Web記事や雑誌への執筆や翻訳
  - CIOオンライン http://www.ciojp.com/
  - BTOクラブ http://btoclub.jp/index.html
- Twitter http://twitter.com/yumotsuyo
- 得意分野
  - テストマネジメント、テスト分析

HPのテストツール



キャプチャ(記録)/リプレイ(再生)ツールの仕組み



アジェンダ

– テスト自動化で工数削減？

– いきなりスクリプティングしてませんか？

– アプリケーション開発の時にテストのこと考えていますか？

テスト自動化の前に押えておきたい3つのポイント

3つのポイント

– **テスト自動化で工数削減？**
  ・**テスト自動化の目的**

– **いきなりスクリプティングしてませんか？**
  ・**テスト開発プロセス**

– **アプリケーション開発の時にテストのこと考えていますか？**
  ・**自動テストとアプリケーション開発相乗効果**

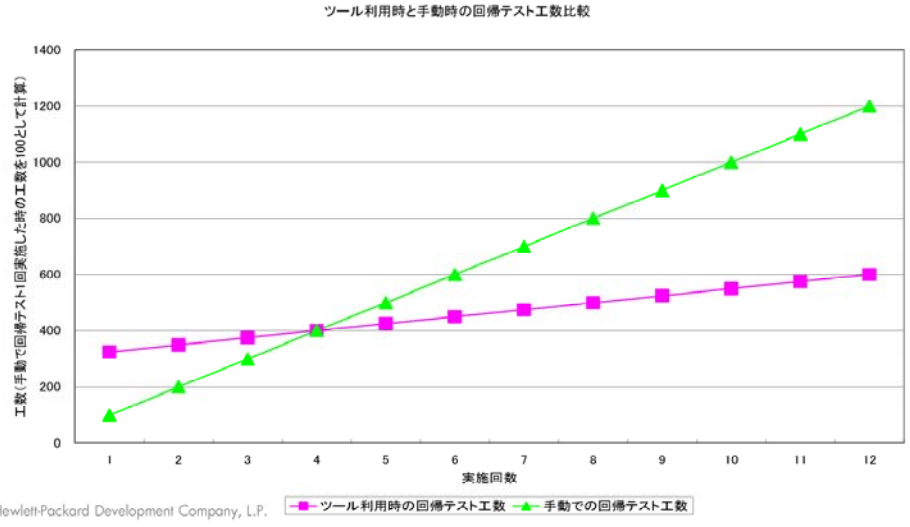
自動テストの効果事例…その1

某IT会社の試算では、手動テスト実施1回の工数を100とした場合、

・スクリプト作成工数：300

・自動テスト実施＋結果確認＋10%のスクリプト修正工数:25

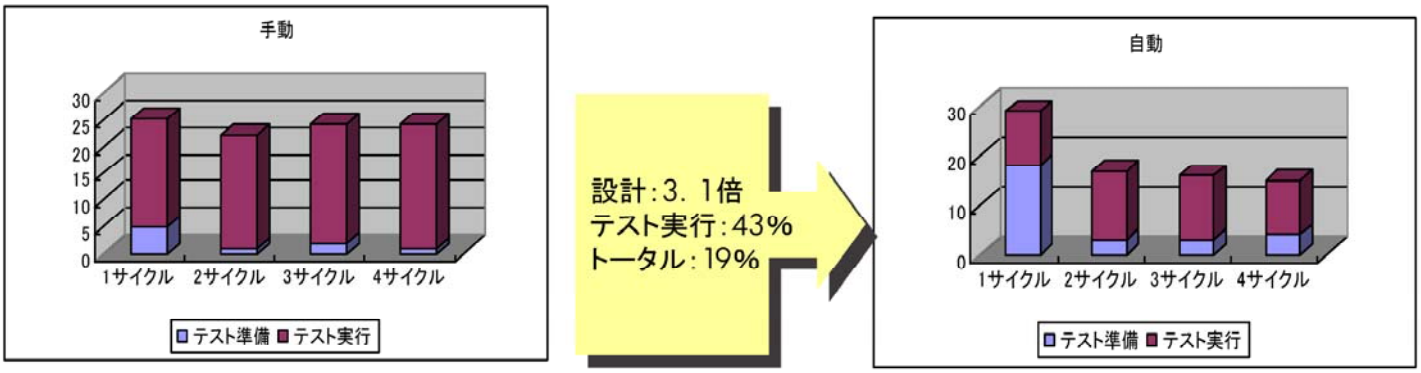
となり、自動テスト4回実施で手動テストと同等の工数となり、**4回以上実施するテストケースで自動化すると効果的**になるとの検証結果となった



自動テストの効果事例…その2

- テスト準備工数にて手動の2倍～5倍
- テスト実施工数にて1/10～1/20

事例：印刷処理マッチングテスト4サイクル実施時の比較データ



適切なテストマネジメントによる工数配分（準備工数割合を増やすこと）
スクリプトが複数回利用できるテスト戦略があれば自動化は成功する

何を自動化するべきか

- 自動化に**向いている**もの
  - 同一のシナリオに対して複数データでテストをする
  - ビルド変更やパッチ適用の度にテストを繰り返す
  - ビジネスに影響の大きい重要なアプリケーション
  - 単純な操作
  - アプリケーションの中で仕様が安定してきた機能

スクリプト利用回数**多**

スクリプト作成時間**短**

- 自動化を**避けた方がよい**もの
  - 複雑な検証やシナリオから始める(処理毎に複数のテストに分けた方がよい)
  - エラー処理や例外処理を1つのテストに含める(複雑な条件分岐が必要になる)
  - 安定していない機能の自動テスト（仕様変更が頻発する箇所）
  - すべてを自動化する
  - 1回しか行わないテストの自動化

一般的なシステム開発でのテスト機会

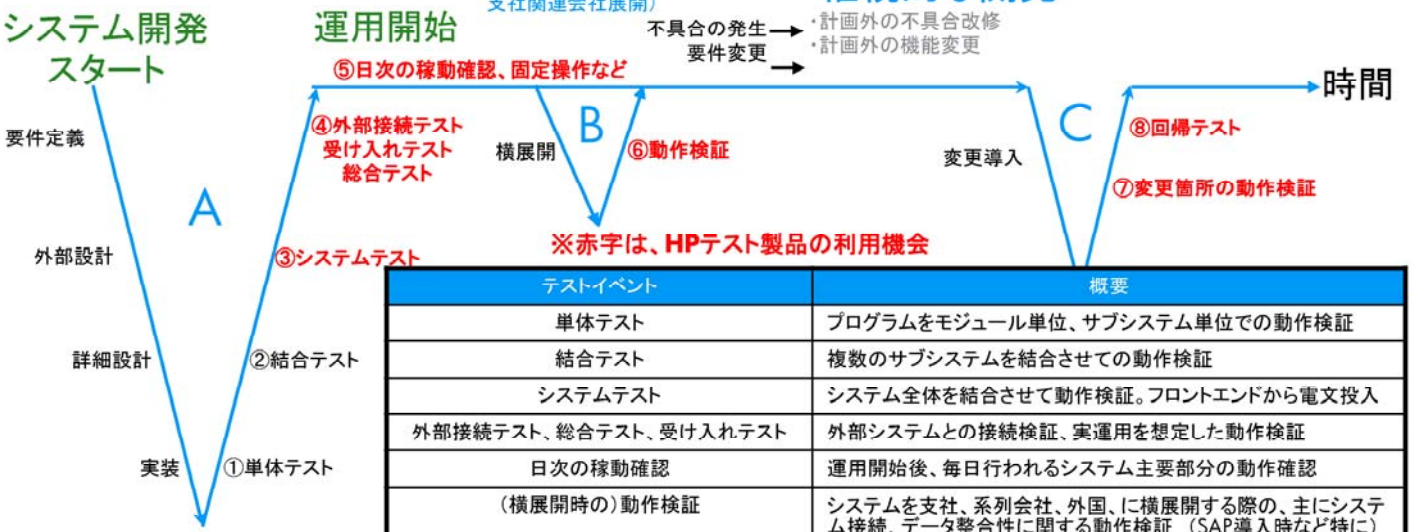
主に開発会社リード
納品して終了

エンドユーザーがリード。変更を管理する必要性
運用費中テスト工数の割合が大きい

A.新規開発

B.横展開
（グローバル展開、
支社関連会社展開）

C.変更導入
.継続的な開発
・計画外の不具合改修
・計画外の機能変更

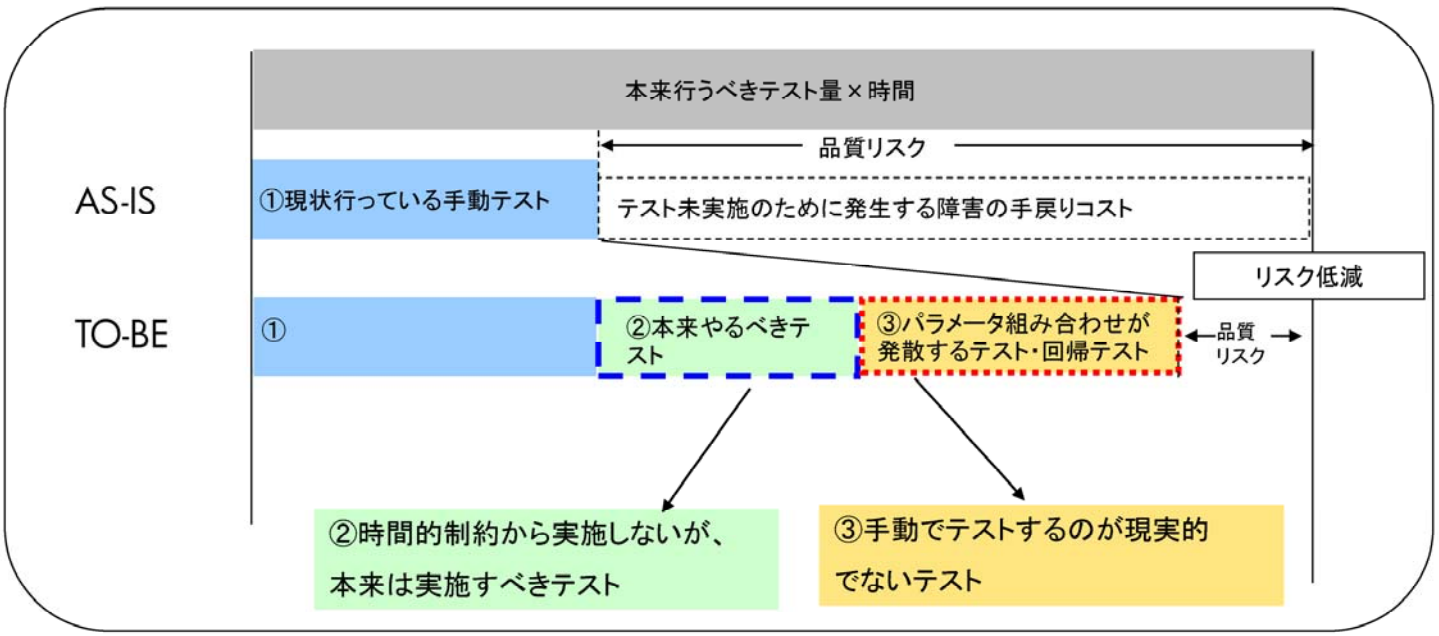


テストイベント	概要
単体テスト	プログラムをモジュール単位、サブシステム単位での動作検証
結合テスト	複数のサブシステムを結合させての動作検証
システムテスト	システム全体を結合させて動作検証。フロントエンドから電文投入
外部接続テスト、総合テスト、受け入れテスト	外部システムとの接続検証、実運用を想定した動作検証
日次の稼動確認	運用開始後、毎日行われるシステム主要部分の動作確認
（横展開時の）動作検証	システムを支社、系列会社、外国、に横展開する際の、主にシステム接続、データ整合性に関する動作検証 （SAP導入時など特に）
（変更導入後の）変更箇所の動作検証	変更が正しく導入されたか（不具合が改修されたか）、動作検証
回帰テスト	変更箇所以外の部分に変更の影響がないかどうか動作検証

本来、ライフサイクルの中で非常に多くの場面でテストが必要

一般的なテストの課題

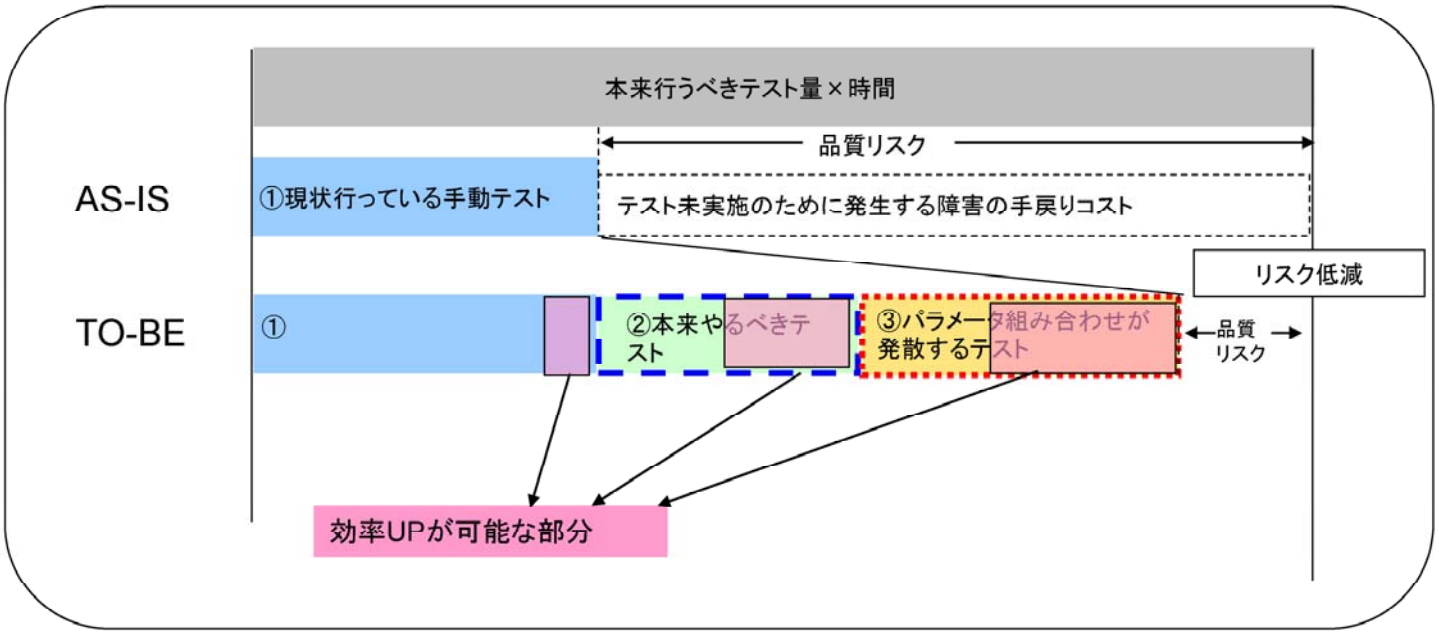
■ しかし、本来すべきテスト量が時間的、経済的制約から全て行えない



品質にリスクを抱えたまま市場へリリースしている

自動化のメリット(1)

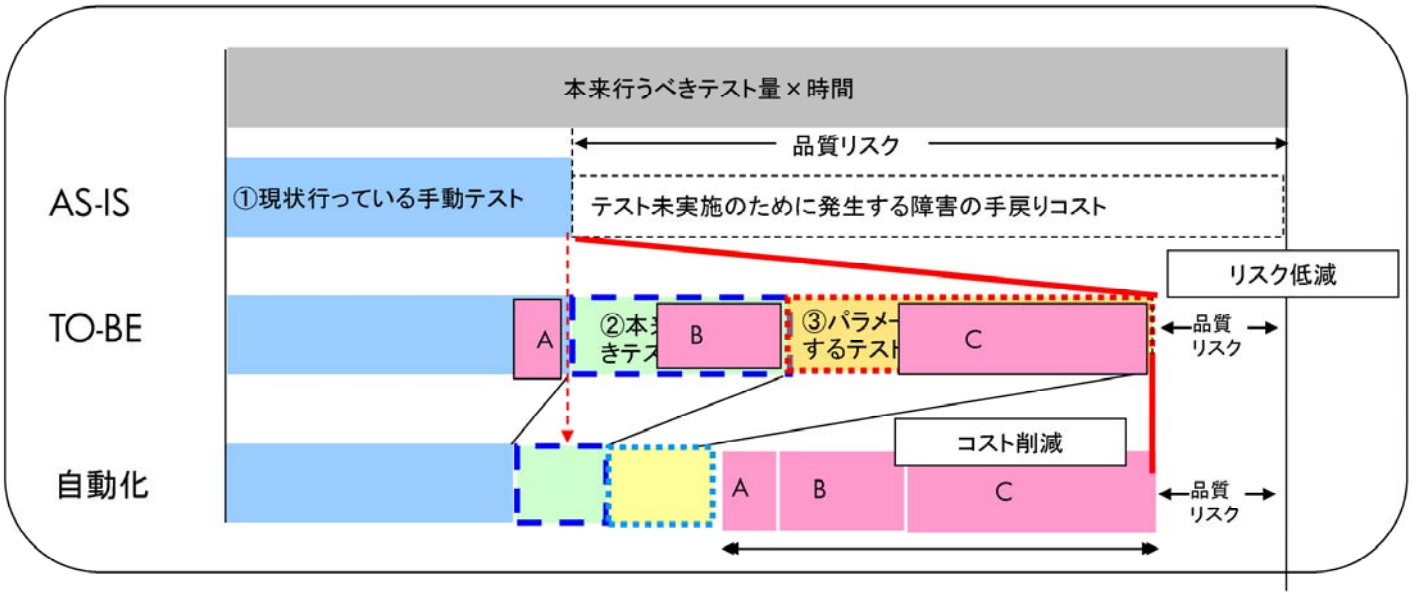
■ 機能テストのテスト自動化により5%～70%は効率化が可能



自動化でテスト量が増加し網羅率があがり、理想に近いテスト量を実施可能＝リスク低下

自動化のメリット(2)

■ 自動化でA～Cの工数を削減しつつ、品質に対するリスクを低減



今まで出来なかったテスト量でカバレッジを向上するのが真の効果

自動化の目的

- 工数削減
  - スモークテスト工数削減
  - リグレッションテスト工数削減
  - 繰り返し型テスト工数削減
  - テストデータ準備

- **テストの網羅率向上**
  （手動ではできないテスト量の実施）

- テスト結果進捗把握など管理工数削減

- 時間外テスト実行による期間短縮

- 確実な欠陥再現能力

品質に寄与することで「手戻りコスト」の削減を目指そう！

3つのポイント

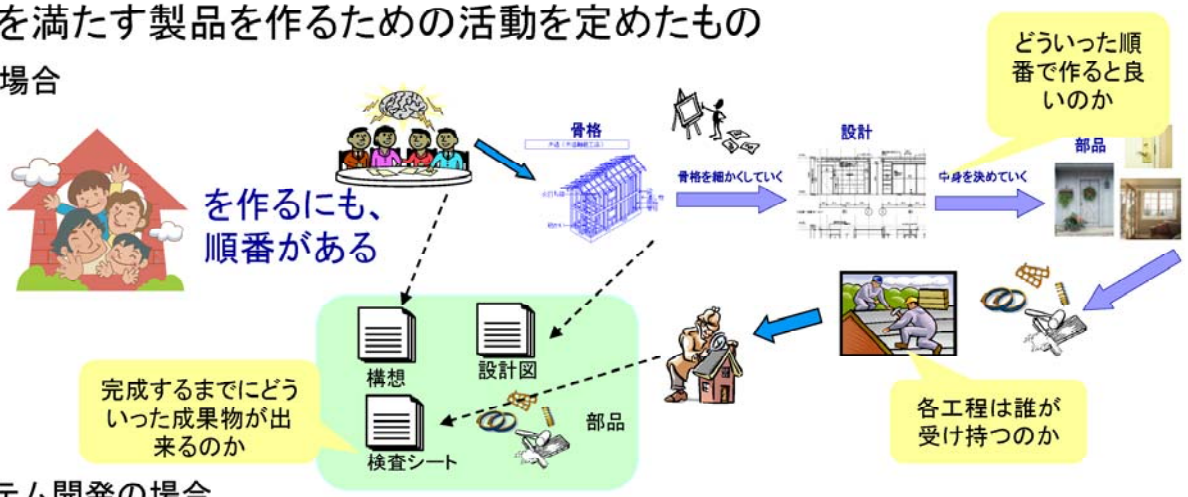
- **テスト自動化で工数削減？**
  - **テスト自動化の目的**

- **いきなりスクリプティングしてませんか？**
  - **テスト開発プロセス**

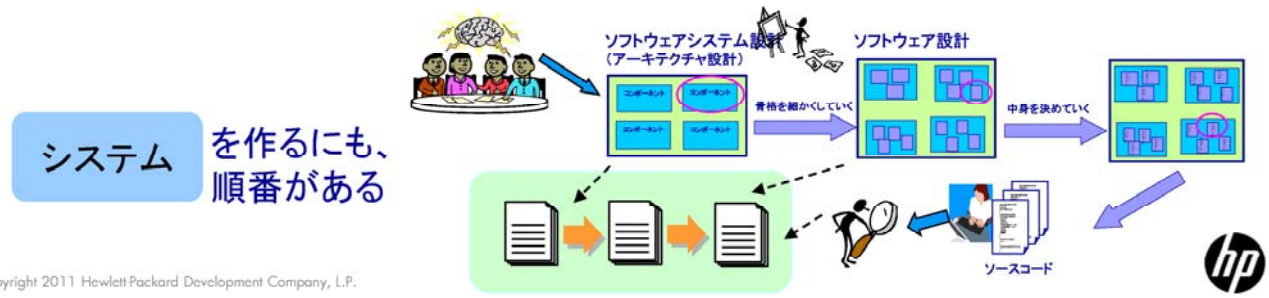
- **アプリケーション開発の時にテストのこと考えていますか？**
  - **自動テストとアプリケーション開発相乗効果**

ソフトウェア開発プロセス

– 要求を満たす製品を作るための活動を定めたもの
  ・家の場合



  ・システム開発の場合



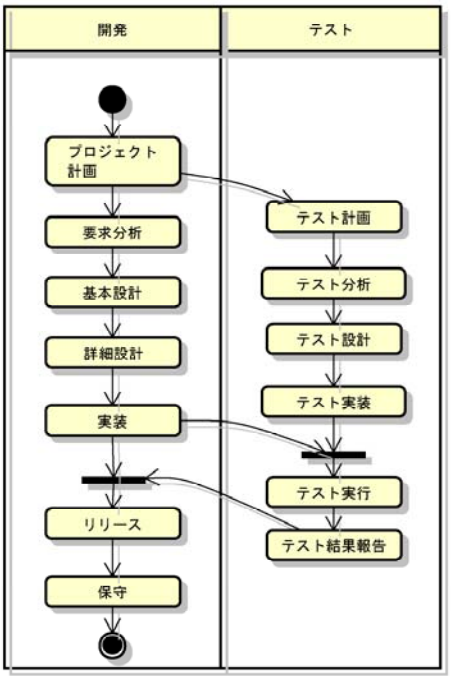
ソフトウェア開発ではよくいわれる事

– 顧客の要望を聞き、いきなりソースコードを書いてはいけない！何故なら…
  ・本来実現すべき事を間違える
  ・実装すべき仕様に抜け漏れが起きる
  ・ソースコードの何処を直して良いかわからない
  ・副作用に気づかない
– なので、開発プロセスが必要

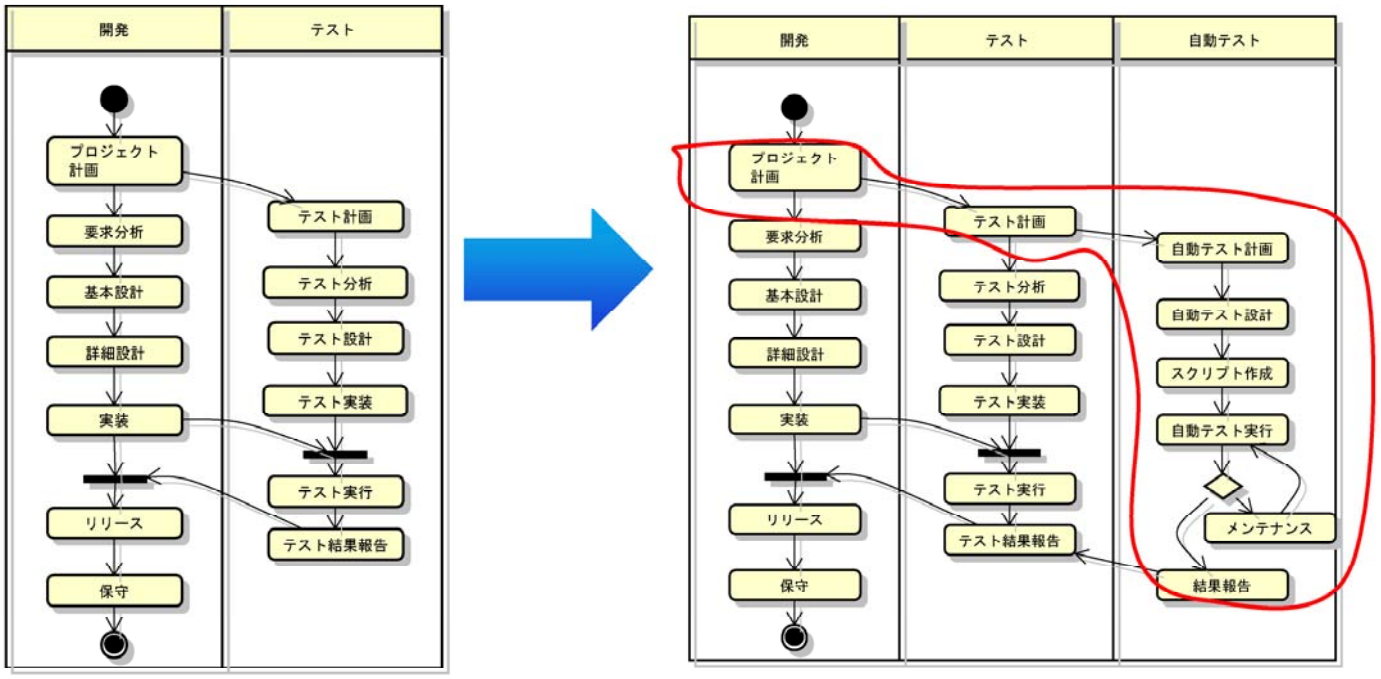
– テストはいきなりテストしてよいのか？
  ・テストの十分性、レビュー容易性、保守性、爆発防止
  ・テスト開発プロセスが必要になる

– テスト自動化に関しては？

テスト開発プロセス



自動テスト開発プロセス



自動テスト開発プロセス

- 自動テストの計画が必要
  - 自動化すべき箇所の特定が必要 （何でもかんでも自動化できない）
    - テスト結果の確認を含めて自動化できないか検討する
  - 自動化によって発生するリスクを明らかにする
    （操作性の確認は手動でやる、など）
    - 自動化により確認できなくなるテストケースをリストアップして対策をとる

- 自動テスト設計⇒次ページ
  - テスト対象をどう網羅するか？を考えてテストケースにする
  - 従来のテストケースが本来の考え方に沿っていると自動化しやすい
    - さもないと、自動化のために従来やるべきだがやっていない作業が負荷になる可能性もある
      （何をもってOKとすべきテストを自動化しているのか？を考えながらコードを書く破目に）

- スクリプティングのテクニックを駆使する
  - 共通モジュール化（スクリプト規約、共通モジュール定義）
  - データドリブン

- マネジメント
  - スクリプト開発体制整備（QTPスペシャリスト育成・開発期間にメンバをアサイン）
  - ソースバージョン管理
  - 進捗管理（スクリプト作成時・保守時）

[重要]無駄に作業を増やしてるわけではない

– 「プロセスが増えて作業が増える」と思うかもしれないが…

– プロセスは、**考慮してほしいこと**を明記しただけであり、むだにドキュメントを増やしてほしいわけではない

– **考慮しないでやる**と抜け漏れが出る→手戻り

– ドキュメントは、必要なものだけ作ること

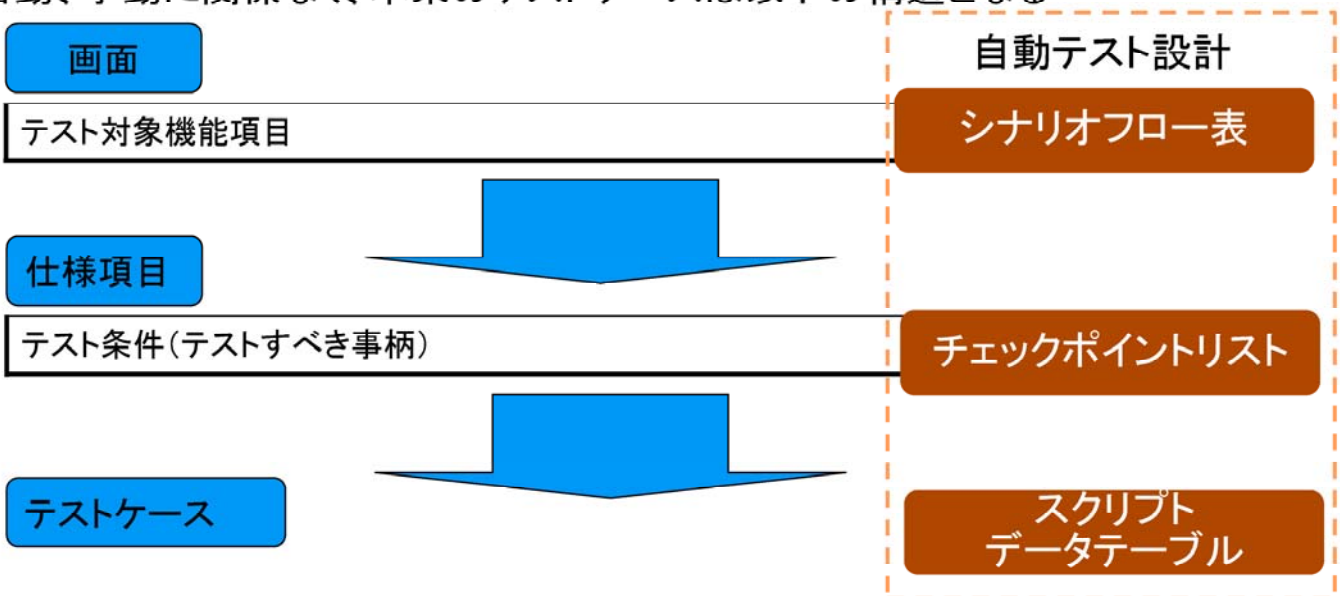
＜原則＞

・メンバー間の合意に必要なことを記す

・考慮すべきことの抜け漏れを見つけやすく記す

自動テスト設計（の前に知っておいてほしいこと）

自動、手動に関係なく、本来のテストケースは以下の構造となる



テストケース条件	値	操作	期待結果
- 入力データ - 事前状態 - 環境 - タイミング			- 出力結果 - 事後状態

自動テスト設計（シナリオフロー表）

テスト計画で特定した「自動テスト対象範囲」に基づき、自動テストでテスト対象の画面を記録・再生する順番（テストシナリオ）を決定する。決定したシナリオは自動テスト設計書の「シナリオフロー表」に記入する

シナリオフロー表

スクリプトフォルダ名を記載

有無を記載

スクリプトフォルダと一緒に管理する場合はこの欄への記載は必要なし

シナリオ名	**FlightFAX注文登録**				
シナリオ概要	新規のフライト予約を行い予約結果をFAXで通知する				
シナリオスクリプト スクリプトフォルダ名	フライト予約	**データドリブン**	有	**データドリブン用ファイル名**	flightRFax.xl

No.	画面名	スクリプトフォルダ名
1	ログイン	
2	フライト予約	
3	FAX注文	

シナリオ作成のコツ

自動テストをよりたくさん行うためには

不具合が出なくなった機能が、
・ディグレードせず
・品質を担保できていること
を自動でテストにする

- **シナリオを作るためのテスト対象選択の基本アプローチ**
  - **小さい単位（ex.画面）でスクリプトをモジュール化し、順々にシナリオの肉付けができるようにする**

①基本機能の基本操作	②リスクの高い機能・要求	③ビジネスフローに仕立てる

- **データを中心にシナリオ（実行順番）を構築**
  - **登録・更新は、事前準備データの条件や、タイミング・競合の問題、その後の検索への影響を考慮**
  - **テストデータを登録・削除(セットアップ／クリーンアップ)する手順をスクリプトに追記**
- **テスト対象選択方法**
  - **操作手順を単純にできるものをテストする　※テストスクリプトの作成・保守の工数を削減する上で重要**

＜その他テスト対象の選択ポイント＞

検索条件入力のバリエーション	項目数が多い
	組合せが多い
	操作の繰返しが多い
処理条件	内部処理が複雑である（単一操作で入力や事前条件のバリエーションが増えるため）
	結果を得るまでに時間がかかる（入力量、手順）
エラー処理	発生の条件が多い
	発生させるのに手がかかる
	エラー処理が複雑（チェックポイントが多くなる）

自動テスト設計（チェックポイントリスト）

仕様項目（テスト条件）とそれをどういう方法で確認するか(チェックポイントの内容）を決める

どのチェックポイントで確認するか（CP種別）を決める

- C:標準チェックポイント
- T:テキストチェックポイント
- B:ビットマップチェックポイント
- S:整合性チェック（EXCELとの比較など）
- O:チェックポイントを挿入せずスクリプト再生が出来ればOKとする

「ツールではどうテスト結果を確認するのか？」
を考えること。
設計をすると、「考えた事」を整理して見直す事が出来る

チェックポイントリスト

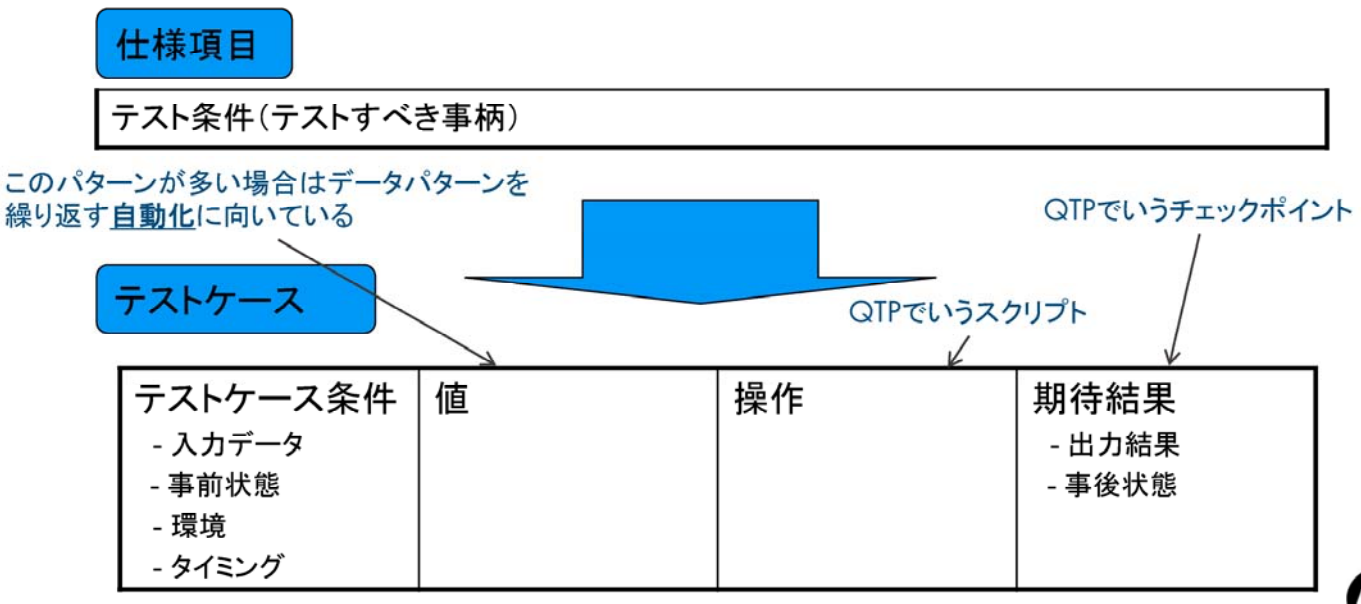
No.	シナリオ	画面	テスト条件	入力データ	チェックポイント内容	CP種別
1	FlightFAX注文登録	ログイン	ログイン可能なユーザでログインができること		次の画面に遷移できればOKとする	O
2		フライト予約	フライト金額が正しいこと		チケット枚数と単価の掛け算の値となることの確認	S
3			フライト予約登録をすると挿入ボタンがグレーアウトすること		オブジェクトのプロパティでディスエイブルになっていることを確認	C
4			フライト予約が登録できていること		フライト予約の番号で呼び出しができることを確認	O
5					呼び出した画面に表示された内容が登録時のデータテーブルの内容と同様であることを確認	S
6		FAX注文	FAXできていること		フライト予約画面にて「FAX送信に成功しました」というメッセージがでること	C

自動テスト設計（データテーブル）

入力データをスクリプトの中でパラメータ化し、データパターンだけ変更して多くのテストを実施すると自動化した際に効果が出る

どのテストが効果的かを把握するには、従来の（手動でも行う）テストケースを活用する

本来のテストケースとして必要なものが整理して記述されていると、テストスクリプトをどう作ればよいか、何が繰り返しの自動化に向いているかがわかるようになる



データパターンの作成ステップ①②

本来手動のテストでも行う部分

①テスト条件となる仕様項目を抽出

バージョンアップの更新頻度の設定値の入力チェック

②テストケース設計

ID	テストケース条件	値	操作	期待結果
テストケース1	変更前の設定条件	変更後と同じ、違う	設定完了ボタンを押下する	・期待結果 設定が反映完了の表示 ・事後状態 設定通りにバージョンアップのための確認処理 最新版のバージョンアップが済んでいない場合はバージョンアップを開始
	直近のバージョンアップ実施状況	変更前にバージョンアップなし、あり		
	前回の設定変更時にエラートなる設定をしている	している、していない		
	新しい設定条件	日毎、指定期間、時間、手動		
	設定後のバージョンアップ確認タイミング	設定実施時点直後 設定実施日より翌日以降		
	日付の有効範囲内	設定しない 設定可能な境界の上限、下限		
	時間が有効範囲内	設定しない 設定可能な境界の上限、下限		

データパターンの作成ステップ③④

③は自動テスト用に見直す作業（手動テストの流用も可能である）

④は本来の自動テストで行う設計（スクリプト作成時でもよい）

③テストケース条件の抽出

テストケース条件	変更前の設定条件	直近のバージョンアップ実施状況	前回の設定変更時にエラートなる設定をしている	新しい設定条件	設定後のバージョンアップ確認タイミング	日付の有効範囲内	時間が有効範囲内
値	変更後と同じ	変更前にバージョンアップなし、あり	していない	日毎	設定実施時点直後	設定しない	設定しない
	変更後と違う	変更前にバージョンアップあり	している	指定期間	設定実施日より翌日以降	設定可能な境界の上限	設定可能な境界の上限
				時間		設定可能な境界の下限	設定可能な境界の下限
				手動			

④データテーブル作成

QTPのデータパターンを記述する場所（データテーブル）
テスト可能な具体的な値をセット

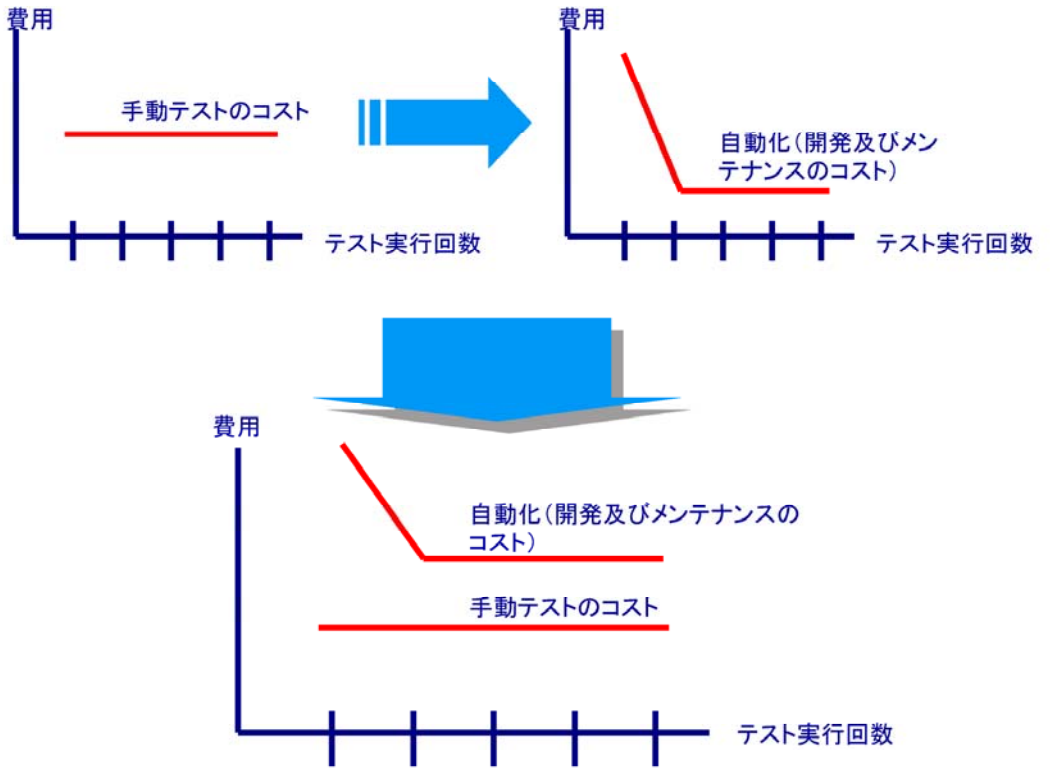
3つのポイント

– テスト自動化で工数削減？
  ・テスト自動化の目的

– いきなりスクリプティングしてませんか？
  ・テスト開発プロセス

– アプリケーション開発の時にテストのこと考えていますか？
  ・自動テストとアプリケーション開発相乗効果

回帰テストで効果が出ない

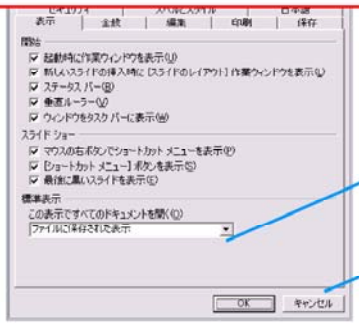


高橋寿一 「知識ゼロから学ぶソフトウェアテスト」から引用

バージョンアップすると動かない

– 例えばこんな原因
  - 開発者が仕様変更無しに作り変える
    – 見た目は変更がないのにウインドウやボタンなどの内部的な名称を変更する
    – リストー覧の並び順だけ変更する
    – オブジェクトの種類を変更する
  - GUIの仕様がFIXしない or 大きく変える
    – オブジェクトが増えたり減ったり、画面遷移が変わったりで作り直し

同じスクリプトを再利用するために作り込みが大事
だが、上記の例では必ずメンテナンスが発生してしまう



リストの並び順を変更
している

見た目は「OK」「キャン
セル」だが、内部的な
名前を変更している

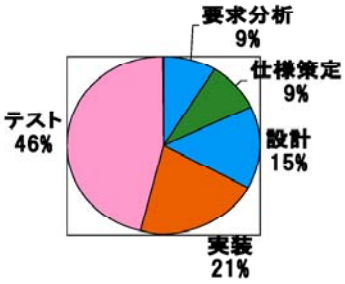
VerUPすると今まで動いていたスクリプトが動かない例

テストツールの限界を無視して開発

– スクリプトが上手く動かないケース
  - 独自に作ったオブジェクトを使う
    – オブジェクトを認識できない(ボタン・エディットボックスなど)
  - GUIにユニークな名前を付けていない
    – JSPで全て同じ名前のHTMLのタイトルを使うので一意に認識できない

– スクリプトの作りこみが必要になるケース
  - 環境依存部分
    – URL
    – アクセスした日時などの時間情報
    – アクセス回数　など
  - 画面には表示されないデータを取得して判定条件に使う
    – 見た目には同じデータなのにスクリプトが動かなくなる or
    チェックポイント(期待結果比較)でNGになる

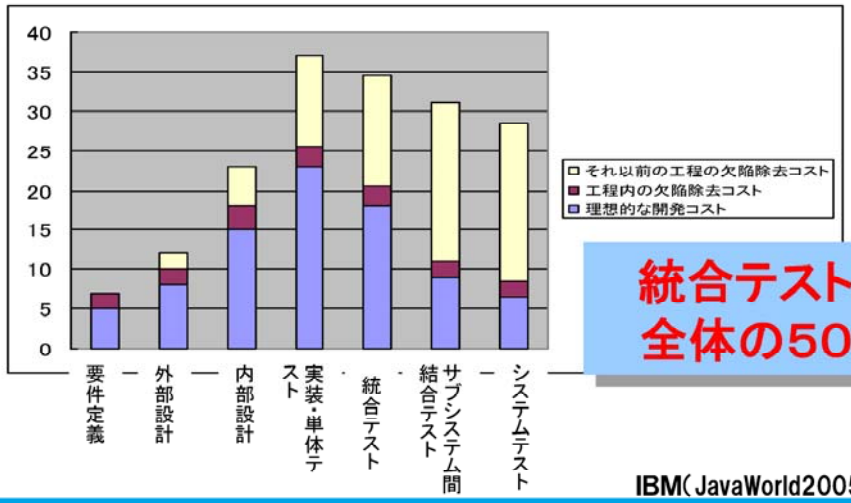
「テストを考慮して開発なんてしてられない」と言う人もいるが…



米国における一般的な各工程での費用の割合

「基本から学ぶソフトウェアテスト」(日経BP社)より引用

◎7000人月の開発プロジェクトにおける開発コスト内訳



開発にかかる全費用のうちテストが50%弱を占める

統合テスト以降が全体の50%以上

IBM(JavaWorld2005.9)より引用

テストがボトルネックに！テストを意識した開発は効果がある

たとえばこんな解決策

ここでオブジェクトの認識ができない事に気づくとそこから対処を始めないといけない

開発の流れ	画面設計	詳細設計	コーディング	ビルド	テスト
従来の自動テスト			ビルド終了までスクリプト作成に着手できない		スクリプト作成 → 自動実行
解決策		仕様書の不備を指摘	オブジェクト認識に関して開発者が単体テスト(自動)	ビルドと連携	自動実行
		画面設計書からテスト担当者が自動テスト設計、スクリプト作成			

①画面設計のオブジェクト情報を元にスクリプトを作ってしまう

・開発者が画面設計書にテスト自動化に必要な情報を全て記述

・ビルドがすんだら即自動テスト開始

・仕様変更時も画面仕様書から直すとスクリプトメンテナンスとの連携ができる

②早期レビューや自動連携

・テスト設計で見つかる画面設計書の不備をコーディング前に指摘

・テストの自動実行をビルドツールと連携させて自動化

プロジェクト全体でテストを意識した活動をすれば自動テストの問題を解決でき、開発の流れのパラダイムシフトを起こせる

まとめ

– テスト自動化で工数削減？
  ・テスト自動化の目的

– いきなりスクリプティングしてませんか？
  ・テスト開発プロセス

– アプリケーション開発の時にテストのこと考えていますか？
  ・自動テストとアプリケーション開発相乗効果