コストを考慮した自動テストの導入と維持
～フリーウェアを利用した低コストのテスト自動化の実現～

株式会社フォーラムエイト TestGroup

○古江　智和

目次

背景（1）

・2002年　リリース前の自社開発製品をターゲットに、自動テストをメインに実施する部署としてTestGroupが発足。
- **短期間で数種類のテストツールを検討**した結果、Mercury Interactive社のWinRunnerを選択　⟶　本格運用開始。
- **様々な試行錯誤**を繰り返しつつ自動テスト自体はほぼ現在の運用形態に近い内容に収束したが**テストツールの特性上対処できない不満や要望**も残った。

・2007年　フリーウェアであるUWSCでテストを作成する機会が発生し、性能及びコストパフォーマンスの高さを実感。新規テスト作成及び既存テストのリプレースをUWSCで実施することに。

⟶ 2009年現在、ほぼすべてのテストをUWSCで実施中。

背景（2）

テスト自動化を通して発生する様々なコストとリスク

短期間で数種類のテストツールを検討
試行錯誤を繰り返して現在の業務に収束
テストツールの特性上対処できない不満や要望
etc…

我々の経験知を基に、上記各内容を説明。
自動テスト導入検討時の御参考に。

テスト自動化のコストとは？

テスト自動化のコストとリスクを把握する

導入前
ツールの選定／購入コスト

導入前後
ツールの習熟に関わるコスト

導入後
テストスクリプトの作成／メンテナンスコスト

テスト自動化のコストとは？（1）

ツールの選定／購入コスト

- 短期間での選定を要求される
  - ・ツールに対する一定のスキルがないと最適な選択は困難
  - ・高価なツールは導入後、後戻りできない。
- 商用テストツールは相当高価なものが多い（2002年当時）

実は極めてリスクの高い作業

テスト自動化のコストとは？(2)

テストスクリプトの作成／メンテナンスコスト

- テストスクリプトは複数回使えてなんぼの世界
  - 同じスクリプトを何回使うかで自動テストの成否が決まる
  - スクリプトの変更を最小限に抑えて使い続けるには？

・テストオブジェクトへの適用時期を考慮
・スクリプトを作り込み過ぎないテクニック

メンテナンス（改修）コストの抑え込みがポイントであるが
テストツール以外の要因（テストの種類・適用時期）も大きい

テスト自動化のコストを下げるには？(1)

ツールの条件

- 価格
  - 購入コストは導入後の運用、維持における足枷
    - 必要ライセンス数の購入が困難
    - 他ツールへの乗り換えが困難
- シンプル
  - 多機能な程、作成／メンテナンスコストが増大
  - ツールによるリソースの消費はテスト実行に悪影響
  - スクリプト関数は多いほどスキル習熟が要求される
  - レポート作成、BTS連携等TestSuite的機能は必要？

テスト自動化のコストを下げるには？(2)

テストスクリプトの作成／メンテナンスコストを下げる条件

- テストの種類
  - 回帰テストが基本
    - 最少のGUI操作で出力結果を比較して取得
    - 出力比較には別ツールを使用した方が効率的
- 適用時期の考慮
  - 安定バージョン（Ver1.0.0以降）のみに適用する
    - ファーストリリース前、仕様は常に揺れ動く
  - 大規模改修が予想されるアプリに適用しない

メンテナンスコストの発生が少ないと想定される時期を考慮
すれば、自動回帰テストは十分に有効なテストとなる

テスト自動化のコストを下げるには？(3)

テストスクリプトの作成／メンテナンスコストを下げる条件

- シンプルコーディング

自動テストはインタフェースの変更を予測できない
アプリケーションが相手

・GUI操作は必要最小限の記述で
・過度に作り込まない（戻値のチェック等必要最小限）

スクリプトに求められるものは完璧な動作ではなく
メンテナンスの早さ、容易さ

テストツール(1)

最初に導入したツール(WinRunner)の問題点

- 高価
  - ライセンスの追加が困難
  - 他部署への導入が困難
- 多機能
  - 使いこなすには一定のスキルが必要
  - 環境構築からしてある程度敷居が高い
- 重い
  - 手動では出現しないエラーの発生
    （リソースの消費がテストオブジェクトの動作に影響？）

テストツール(2)

リプレースしたツール(UWSC)による問題点の解決

- 安価
  - ライセンスの追加が容易
  - 他部署への導入が容易

→ Test部署で作成した スクリプトを 他でも簡単に利用可能に

- シンプル
  - 必要最低限な関数の数
  - プログラマであれば習熟は容易

→ スクリプト作成／メンテナンス コストの低減 ↓ 他のテストへ時間を回せる

- 軽快
  - 自動テスト運転時間の短縮

→ テストデータを増やせる or テスト報告書提出時間短縮

テストツール(3)

WinRunnerとUWSCの比較

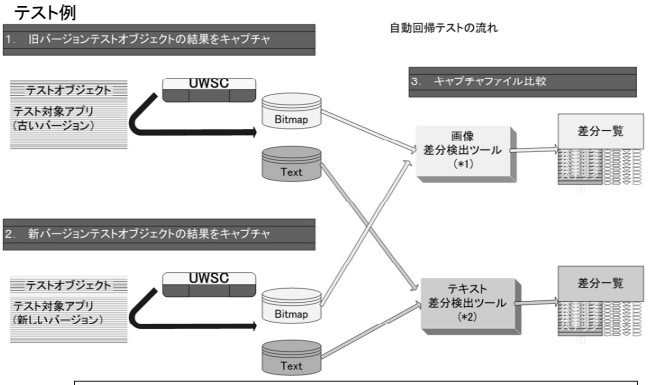
	UWSC	WinRunner
価格	フリーウェア（シェア版有）	100万円前後
IDE	無し	有り
性格	Windows自動化ソフト	企業向機能テストツール
関数の数	80個前後	670個前後
スクリプト例 (*1)	*CLKITEM(id, "開く", CLK_MENU);*	(*2)参照 GUIMAPを使用しない記述

(*1)ファイルサブメニュー「開く(O)…」を選択する動作

(*2)*menu_select_item("{class: ¥"menu_item¥", label:¥"開く(O)…Ctrl+O¥, parent: ¥"{class: ¥¥¥" menu_item¥¥¥", label: ¥¥¥"ファイル(F)¥¥¥"}¥"}");*

テストツール（4）

テスト例



(*1) **独自開発ツール使用(ExcelVBA+Delphi)** http://www.vector.co.jp/soft/winnt/prog/se347531.html
(*2) **WinMerge等のフリーウェア** http://www.geocities.co.jp/SiliconValley-SanJose/8165/winmerge.html

「JaSST'08 Tokyoフリーウェアで作る自動テスト －UWSCの利用例－」P.9 から抜粋

課題(1)

回帰テスト以外への自動テストの適用を模索

取り組み例

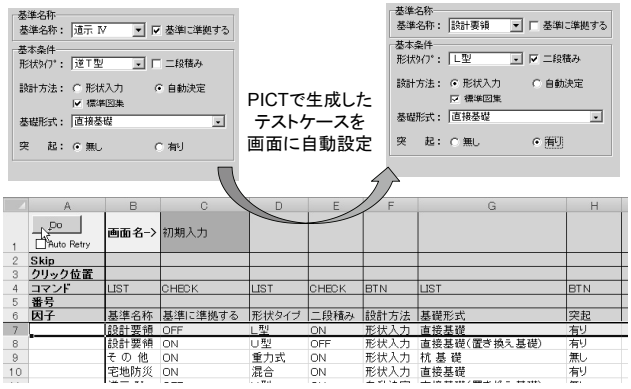
- Pairwiseテストケース生成ツールと連携して組合せテストを半自動化する

使用するツール（すべてフリーウェア）
・PICT（Pairwiseテストケース生成ツール）
・AssistAllpair（PICTのExcelフロントエンド）
・UWSC（テストケース内容をテストオブジェクトに設定）

以前は作成したテストケースを見ながら手動で設定していたが、画面への設定を自動化することにより効率が大きく向上した

課題(2)

組合せテストの半自動化



	A	B	C	D	E	F	G	H
1	Do / Auto Retry	画面名->	初期入力					
2	Skip							
3	クリック位置							
4	コマンド	LIST	CHECK	LIST	CHECK	BTN	LIST	BTN
5	番号							
6	因子	基準名称	基準に準拠する	形状タイプ	二段積み	設計方法	基礎形式	突起
7		設計要領	OFF	L型	ON	形状入力	直接基礎	有り
8		設計要領	ON	U型	OFF	形状入力	直接基礎(置き換え基礎)	有り
9		その他	ON	重力式	ON	形状入力	杭基礎	無し
10		宅地防災	ON	混合	ON	形状入力	直接基礎	有り