現場の失敗から学ぶ
自動テストの設計プロセス

○　林　尚平

E-mail：shouheihayashi1979@gmail.com

■概要

テストの工数削減/効率化を目的に自動テストの導入を行う現場は多くなっていますが導入・運用に成功している現場は多くありません。

成功させるためには自動テストの正しい知識、
ノウハウを知ることが必要です。
これまでに経験した現場の失敗事例を挙げ、
自動テスト導入を成功するための設計プロセスを紹介します。

■自動テストの3つの失敗要因

よくある現場の自動テストの失敗要因を3つに分類しました。

①「実装スキル」の失敗→必要な試験が自動化できない
→自動化ツール選定スキル、プログラム技術など実装に必要な技術がない

②「進め方(プロセス)」の失敗→手戻りが多く進まない
→自動テストの技術/知識の不足のため進め方がわからない

③「人・組織」の失敗→方向性/目的が定まらない
→チーム内で自動テストの理解の差で考え方が纏まらず、方向が定まらない

今回は**②進め方(プロセス)**を説明します

■自動テストの進め方（設計プロセス）について

自動テストを導入する際には、どの自動化ツールを使うか、
どうすれば自動化できるかに焦点が当たりがちです。

この点も重要ですが自動テストの導入に
重要なのは**進め方（設計プロセス）**の考え方になります。

自動テストの設計プロセスを正しく理解することで
リスクを把握し、設計事項の漏れをなくし、
自動テストを成功に導くことができます。

■自動テストの導入を成功させる4つのプロセス

自動テストの設計プロセスを紹介します

①計画　　：自動化の方針/計画を決める
②設計　　：自動化の内容、処理構成など詳細を決める
③実施　　：シナリオ作成、実行確認
④振り返り　：実績検証と改善

4つのプロセスを実際にあった現場の失敗事例を踏まえ紹介します。
※全ては説明できないのでポイントを絞って説明します。

【①計画】

計画プロセスでは以下の自動化の方針を決定し計画を立てます。

(1)目的と役割を決める
(2)成功基準を決める
(3)自動化内容を決める
(4)テストケースの内容を分析する
(5)評価対象のシステムを分析する
(6)自動化ツールを選定する
(7)テスト計画を作成する

【現場の失敗例】
最初に自動化ツールを決める

【現場の失敗例】
実装に着眼しすぎて運用フェーズを考えない

【①計画】 (1)目的と役割を決める

自動化の方針を決めるには役割と目的を決める必要があります。
開発手法やテスト内容で異なるのでそれぞれで検討が必要です。

- ◆ 目的:**テスト工数を削減し効率化**
- ◆ 役割:**デグレ確認**(品質向上ではない)

【現場の失敗例】
自動化することが
目的になってしまう

【現場の失敗例】
不具合検出を
役割とする自動テストをする

この目的と役割を達成することが終わりではありません。
この後に何をするかが重要です。

「削減工数で追加試験を行い、総合的に品質を上げる」、
「安定した品質のソフトを短期間にリリースする」などを行い、
品質とスピードの両立を行うことが自動テストの本当の目的です。

【①計画】 (2)成功基準を決める

自動テストの方針を決めるには成功基準を定める必要があります。
※個人の見解になります。

No	成功基準
①	1回の実行で手動に比べ最低30%以上の効率化/工数削減ができる
②	必要な自動化するテストが検討できており自動化できている
③	シナリオが連続で正しく実行出来ること
④	結果の誤判定などなく全て正しく実行出来ること
⑤	実行から結果確認まで人間の手を入れずに最後まで自動で実行出来る事
⑥	メンテナンスや条件初期化などリスクの対策済み

必要なテストが自動化でき、長期間運用出来る仕組みが
揃っている事が成功の条件だと考えています。

【現場の失敗例】
成功基準を考えず
成り行きで基準を考える

【①計画】 (3)自動化の内容を決める

ここでは何の試験を自動化するかを検討します。
何度も実施する価値があり、
不具合出ると問題になる試験を自動化するべきです。

【現場の失敗例】
自動化しやすい
試験を自動化する

試験内容	試験詳細
基本動作	基本機能で不具合が出ては問題のため自動化することで確認する。
市場不具合	一度市場で発生した不具合は2回出すと問題のため確認する。
顧客/ユーザの要求の高い機能	安心/安全や顧客の要求の高い機能の確認を自動化し確認する。
ユーザがよく使う機能	よく使われる機能で不具合が出ると問題なので自動化し確認する。
新機能など注目の高い機能	新機能や法律に関する機能で不具合を出すと問題のため自動化する
デグレが多い機能	開発期間でデグレが多い機能は自動化し確認する。

【①計画】 (4)テストケースの内容を分析する

自動化できないテストケースを自動化しようとすると
シナリオ作成時に大きな無駄工数が発生します。
事前に自動化可能なテストケースか確認が必要する

・試験条件、手順、期待結果が正しいことを確認
・曖昧な表現がなく、必要な情報が明確に記載されていること

【現場の失敗例】
期待結果が曖昧で
シナリオが作成できない

【現場の失敗例】
試験実施したことが
無いテストケースを自動化する

【①計画】 (5)評価対象のシステムを分析する

シナリオ作成時には評価対象を理解しておくと、
実装の成功率や作業効率も上がります。

・評価対象の仕様の深い理解
・仕様書にない仕様・動作の理解
・評価対象の品質状況の調査

【現場の失敗例】
深い仕様を知らず手順を変えて
シナリオの実装ができない

【現場の失敗例】
不具合が多く、
シナリオの実装ができない

【①計画】 (6)自動化ツールを選定する

検討した自動化の方針を実現できる自動化ツールを選定する。
自動化ツールの特徴・仕様の理解しておく必要があります。

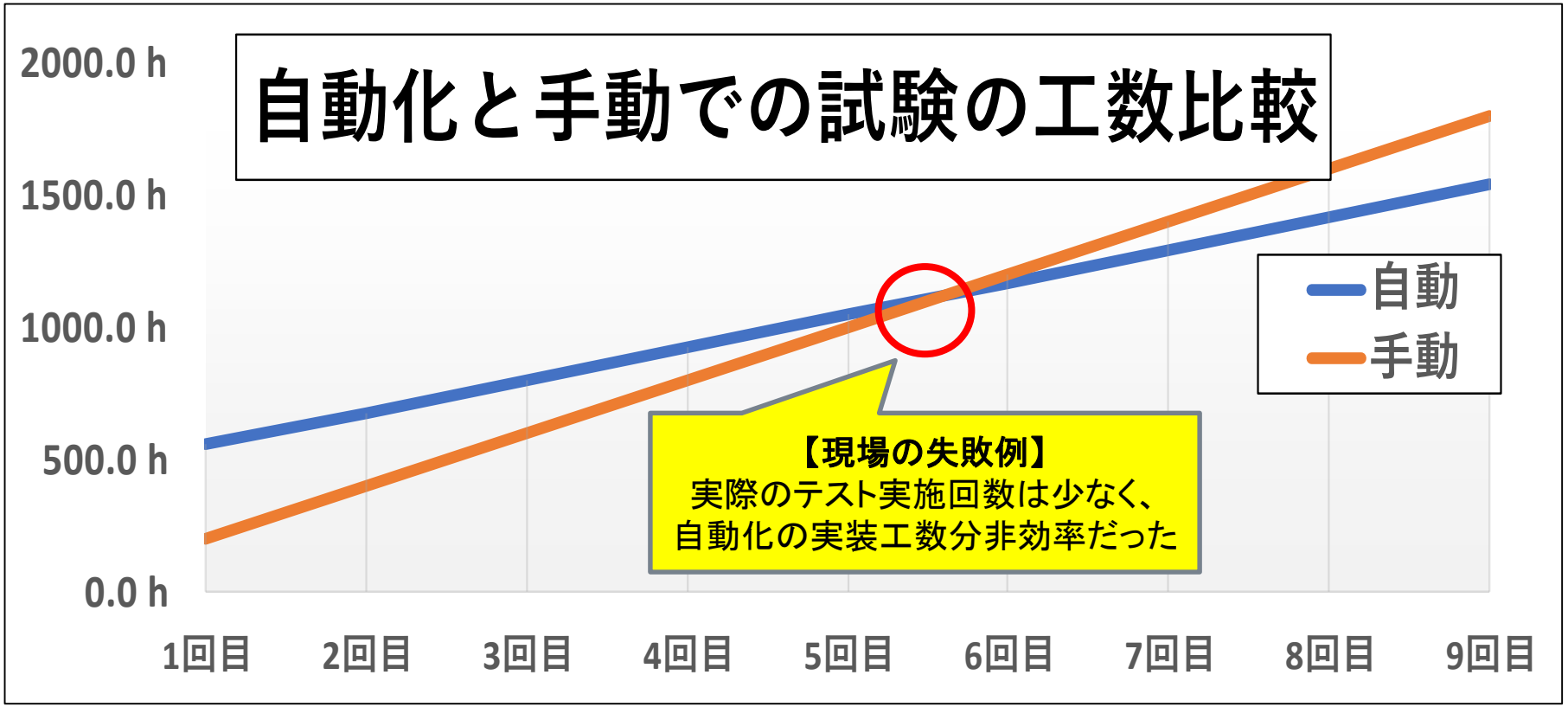
- **テスト内容は多機能な自動化ツールが必要なのか**
- **フリーツールでも実現できるテスト内容か**
- **自動化したいテストが実現できる自動化ツールか**
- **共通関数が使える自動化ツールか**
- **長時間の連続操作が可能か**

【現場の失敗例】
試験手順だけ自動化できるか
調査する

【現場の失敗例】
簡単に自動化できるツールを選ぶ

【①計画】 (7)テスト計画を作成する

自動と手動との試験実施の工数比較を行い、損益分岐点を出し
自動化が有効であるテストの実施回数を算出する



【②設計】 (1)自動化対象を抜き出す

自動化方針に合致しているテストケースを基準を作り抜粋する。
自動化する項目は厳選する必要があります。
切り分けの基準はプロジェクトの状況に応じて見直しが必要です。

画面名	テスト内容	自動化範囲の切り分け基準						総合得点	自動化対象
		仕様変更	自動化有用性	重要度	使用頻度	シナリオ作成工数	テスト複雑度		
メール	表示確認	—	—	—	—	—	—	—	済
	初期値確認	2点	3点	1点	1点	3点	2点	12点	△
	設定値反映	1点	2点	1点	2点	2点	2点	10点	×
	エラーチェック	2点	5点	3点	1点	5点	5点	21点	○
	機能動作確認	—	—	—	—	—	—	—	済
	設定値保持	1点	2点	2点	2点	2点	2点	11点	×
ショッピング	表示確認	—	—	—					済
	初期値確認	2点	3点	1点					△
	設定値反映	1点	2点	1点					×
	エラーチェック	5点	5点	1点					○
	機能動作確認	—	—	—					済
	設定値保持	1点	2点	1点					×

【現場の失敗例】
テストの優先度にかかわらず、
全ての試験を自動化する

【②設計】 (2)シナリオの共通化する処理の設計

シナリオ数が多くなると、修正/作成工数を削減するため、
共通関数を作成する必要があります。

・前提条件を設定する処理
・画面遷移などの同じ手順
・結果判定やエラー処理
・実施結果をExcelファイルへ書き込む処理
　など

【現場の失敗例】
共通化処理を行っておらず
仕様変更発生時に
メンテナンス工数が膨大になる

何度か実施するのであれば共通化を行うべきです。

【③実施】シナリオ作成、実行確認

実施のプロセスではこれまでのプロセスで
決定した自動化方針や詳細内容を元にシナリオを作成する。

・シナリオを作成する
・シナリオを実行して確認する

【現場の失敗例】
共通関数化、自動化できるテストケースなどの
分析していない場合、このプロセスで問題が噴出

1本1本のシナリオでは正常に動いても、
纏めて実行するとNGとなる場合がたびたびあります。

【④振り返り】 実績検証と改善

自動テストで重要なプロセスです。
改善を行うことで更なる効率化を目指します。

- 事前に設定した成功基準と比べ有効だったか検証する
- 計画していた削減工数と実績の工数との比較
- シナリオに追加が必要な共通機能の検討と導入
- さらに工数削減/効率化をおこなうための対策
- シナリオ実行時に発生した問題と対策

【現場の失敗例】
振り返りを行わず、
失敗したらそれで終わる

【現場の失敗例】
発生した問題がスキル不足で
解決できない

■自動テストの設計プロセスのまとめ

自動テストをプロセスに従って導入を進めることで、
以下のメリットがあります。

・リスクを事前に潰しこみできる
・プロセス毎の作業の漏れがなくなる
・自動テストの全体像が見える
・プロセス定義することで作業が俗人化しない

プロセス定義を行うことで手戻りによる無駄工数をなくし、
リスクを軽減した自動テストが可能になります。

ご静聴ありがとうございました。