@jun2014 (Jun Shikoda)

テスト仕様書

テスト自動化　QA　QAエンジニア　テストケース　テスト設計

最終更新日 2025年07月29日 投稿日 2015年11月11日

1.

テストとは

まず、**ウォーターフォール型開発・アジャイル型開発**と開発手法によってテストの柔軟さや変更コストが変わってきます。

ここでは、**アジャイル型開発**をメインとし、（開発）実装完了し（コードフリーズ）**プロダクトリリース**に向かって行うテストを考察します。

不具合の発生は主に、**要件定義**や**設計書段階**であり、（不具合）検出は**主に受け入れテスト**かリリース後の**本番環境**である割合が多い。

そのため、このフェーズを疎かにすると、**リリース後不具合が発生し、場合によってはユーザーへ謝罪や返金が必要**となる。

また、サブスクリプション型・SaaS型のビジネス企業としては**Charn数値**をどう抑えるのかが大事となる。

一企業として損失・信頼が失われる結果となり、一度 **「失った信頼を取り戻す」** のは並大抵では無いです。

予め、**テスト計画**や**テスト対策**を考え**テストを軽視**しないことです。

不具合のコストを考え、**リリース後の不具合コスト**は**単体テストの不具合コスト**の**数百倍**のコストとなります。

主なテスト種類

・単体テスト

・結合テスト

・システムテスト（機能テスト、負荷テスト、ボリュームテスト、セキュリティテスト、リグレッションテストetc）

・受け入れテスト（シナリオテスト）

・運用テスト

現場（案件）によっては、**「開発エンジニア」** が一連のテストしたり、また **「QAエンジニア」** がテストしたりと異なります。

私の意見としては、 **「開発エンジニア」** **「QAエンジニア」** 双方が確認すべきではないかと。

単体テストの実施は、**開発者**が担当します。
また、**結合テスト（Integration Test）以降のテスト**は**QAエンジニア、もしくはQAテスター**が担当する。

単体テストのルールや**網羅性**は開発とQAが一緒に考えることが品質を良くする上で大事です。
お互いテストの意義というものを見直すこともできます。

また、詳細仕様を把握している**PM、Pdmや開発者**。テスト設計やテストの種類を知っている**テスト担当**者が協力しあうことが品質を高めるのではないでしょうか。

特にスタートアップでは、要件機能仕様書も無いので（あっても完全なものではない）
コミュニケーションという点では。

「企画」・「開発」・「サポート」・「運用」 と連携し進めていく必要があります。

テスト仕様書を作成しテストケースを用意

「テスト計画」、「テスト設計」をし、「テスト仕様書」を作成します。
もちろん、「レビュー」も必要となりますので、見積には追加。

「テスト計画」

- どのようなスコープのテストをするのか（テストの種類）機能テスト、セキュリティテストや負荷テスト

- どこまで担保するのか（決めないと、最後でやったやっていないの問題となります）
- テストを中止する場合や再開する基準は（実施環境の設定不備）
- 実行環境の確認（例「ローカル環境（テスト環境）」→「Staging環境」→「本番環境」での確認なのか？）
- テスト結果（「OK」はどこまでの範囲か、「NG」はどこからなどを決める）
- テストツールは使用するのか（使用する場合は、対象案件での使用メリットも記載）
- 全体のスケジュール（定例MTGや、各テストの期間、実施担当者）
- 組織図（プロジェクトマネージャーやQAマネージャ、QAリーダー、QAテスターの役割の明記）
- どのようなテストデータを用意するか
- 仕様変更や、FIX決めの対応
- テストリスク（リスク発生頻度、重大度）と対策事項（リスクレベルの設定）はまとめているか

※機能テストのテストケースの中にセキュリティテストのテストケースなど、一緒に書かないように。
テスト工程は、テスト工程のケースのみチェックする。何を担保しているテストケースかわからなくなり
工数を増やす要因となる。

「テスト設計」

- 仕様書から読み取れるレベル（仕様書有が前提）

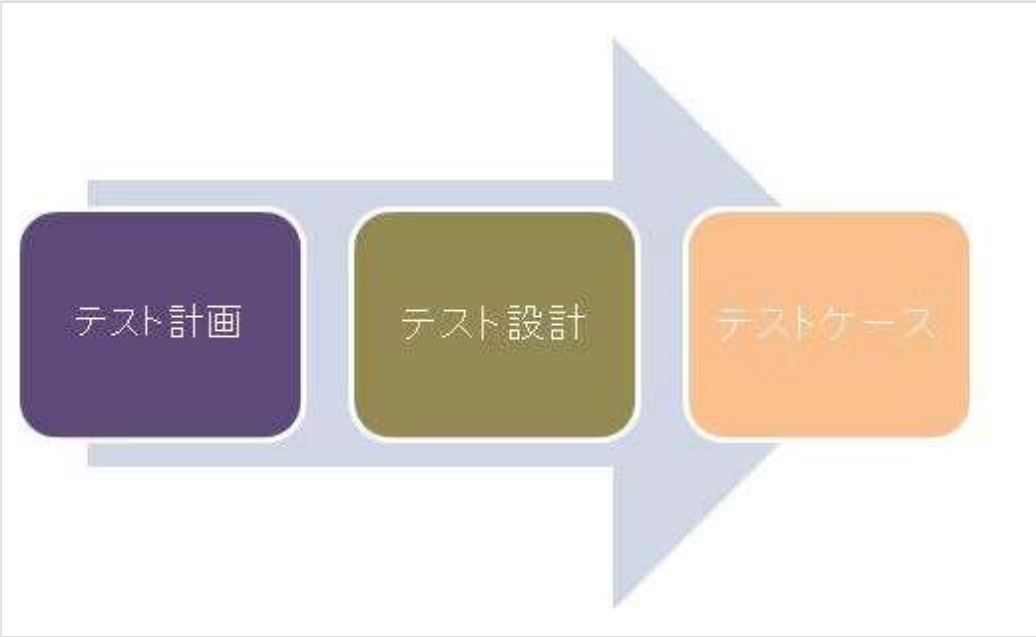
※仕様書が無い場合、どこまで担保するか線引きをしておく

何をもって確認「OK」とするか、また「NG」とするか。各部署との確認を明確化する。無いからわかりません。やりませんでしたと無いように、テスト準備段階で確認する。

- 内部構造を理解し設計するレベル（開発経験がある担当者）
- テスト経験者の勘や知識から実行する探索的テスト（※テスト担当者に依存する）

「テスト仕様書」

- 設計書をもとにテストケースにおこす。
- 無駄なケースが無いか確認する。**直交表**や**Allpair法**を使用し、必要なケースのみ。
- 前提条件を必ず記載していること。レビュー者の負担もある。
- 大項目、中項目、小項目、前提条件、操作方法は、わかりやすく。
  ※前もって、形式（テンプレート）については社内で共有しておくのがベストです。



「単体テスト」、「結合テスト」、「システムテスト」

テスト自動化だけでは難しいので箇所によっては、やはり手動になってしまう。開発用テストケースです。設計書の動作だけではなく、設計書に記載していない箇所もテストケースに入れます。あとは、Webブラウザ単位にテスト。バージョン単位にテスト。

※参考資料
スマホ最新機種
Android OS シェア
iOS OS シェア
ブラウザシェア
直交表とオールペア法の並行運用によるソフトウェアテスト

下記はテストケース例になります。想定は、WEB系のQAチームになります。※第三者検証のテスト会社になるともう少し、画面単位の粒度が変わります。ナレッジはありますが、開発者との距離が離れてしまう懸念もあります。しかし、開発者では気がつかない不具合を検出するには良いと思います。

1. 企画書(構成書)もしくは、設計書からテストケースを作成します

テスト準備シート

①テスト環境が用意されている(※テスト環境に不備がないかどうかも確認)
②Android検証用端末と実行用の「apkファイル」が用意されている
③iOS検証用端末と実行用の「ipaファイル」が用意されている(※リサインが必要であればこれも)
④不具合用親チケットが作成されている
⑤テスター用のアカウントが用意されている
⑥ステータス毎のテストデータが用意されている
⑦テストケースがレビュー済でレビュー修正されている
⑧使用WEBブラウザとバージョンが用意されている

⑨テストツール(Selenium、Jmeter、BurpSuite)が用意されている
※テストツール選定によって異なります。

テストデータ問題がある。
どれだけ用意したらいいのか。ここは難しいですね。どういう方法で作成すれば？？

1. オールペア法と直交表による組み合わせ(※禁則を除く)
2. テストデータツールを使用
   PICT（Pairwise Independent Combinatorial Testing tool）
   https://github.com/microsoft/pict
   Excelプラグインツールに「PictMaster」
   http://sourceforge.jp/projects/pictmaster/

テスト設計例

テスト仕様書に落とし込むまでの設計作成例です。
テストレベルは「どの種類のテストか」対象画面は「テスト対象」項目は「テストとなるオブジェクト」確認項目は「下記の項実施可否の中で○がある項目」これを仕様書に落とし込む

テストレベル	テストタイプ	対象画面	項目	確認項目
システムテスト	機能テスト	登録画面	氏名	正常
				エラー
			住所	正常
				エラー

「確認項目の通常入力リストー覧」　を作成し実施するものだけ〇をつけます。
設計書のレビューも〇の項目を確認することで設計時点で誰でもレビューできるようになります。

テスト仕様書を全てレビュー、例えば数千件のテストケースをレビューするのかというと現実的ではありません。

今回は例として「氏名」・「電話番号」・「住所」の設計となります。
確認項目は、下記の確認項目に紐づけテスト仕様書を作成する形となります。
テスト仕様書は項目に合わせ「〇」を付けたものだけ作成します。

確認項目	必須	実施可否
maxlengthが指定(※10文字である場合、9文字、10文字は入力可能。11文字は、入力できないこと)	○	可
maxlengthが指定されていない場合	○	可
空白文字	○	可
半角文字(ｱｲｳｴｵ、ｶｷｸｹｺ)	○	可
全角文字(あいうえお、かきくけこ)	○	可
機種依存文字(①②③④⑤⑥⑦⑧⑨⑩⑪⑫⑬⑭⑮⑯⑰⑱⑲⑳：㍉㍍㌔㌘㌧㌦㍑㌫㌢：㍻㍼㍽㍾：Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ)	○	可
小数点（第一位まで、第二位まで）	○	可
記号文字(〃 仝 ゝ ゞ 々 〆 ヾ ― ‐ ／ ○ ヽ ＿ ￣ ¨ ｀ ´ ° ゛ ＼ § ＾ ≫ ￢ ⇒ ⇔ ∀ ∃ ∠ ⊥ ⌒ ∂ ∇ ≡ ∨ ≪ † √ ∽ ∝ ∵ ∫ ∬ Å ‰ ♯ ♭ ♪ ‡ ～ ′ ≒ × ∥ ∧ ｜ … ± ÷ ≠ ≦ ≧ ∞ ∴ ♂ ♀ ∪ ‥ ° ⊃ ⊂ ⊇ ∩ ⊆ ∋ ∈ 〓 〒 ※ ″)	○	可
マイナス値	○	可
HTMLエスケープ(<>&"')	○	可
JavaScriptへの埋め込み文字(<>&"')	○	可
SQLエスケープ('"%_%__)	○	可
リンク表示、リンククリック遷移		可
ボタン表示、ボタンクリック遷移		可
プルダウン値、ラジオボタン、チェックボックス、テキストエリア		可
必須項目で未入力の場合の振る舞い		可
クリックを数回した場合の振る舞い		可
数回リロードした場合		可

数回クロージャした場合		可
存在しないパラメータを渡す		可
画面間のデータの受け渡し		可
ステータスによる画面表示処理の振る舞い		可

バックエンド確認（SQL、Linuxコマンドでの確認）

例となります。

確認項目	必須	実施可否
DataBaseに接続できる	○	可
新規、既存、テーブルがある。カラムの追加がある場合は確認。またテストデータ更新、追加した場合の値の確認。		可
「ユーザー」と「管理者」ロール権限の確認	○	可
メール自動設定時のcron設定のパス確認。crontab -l	○	可
「http」と「https」のプロトコル確認		可
config設定が正しい		可
エラーレベルは適切か。	○	可
エラー時には、Error.logに出力。	○	可
jQueryのバージョン確認(※使用ライブラリーの確認)		可
phpのバージョン確認、MySQLのバージョン確認（※バージョンによっては今まで使用できた関数が使えなかったりするので確認が必須）		可
メール送信時には、送信ログが出力。	○	可
ストアドプロシージャ(呼び出しの確認)	○	可
マスターDBのダンプ		可
MySQLスレーブサーバの確認		可

テスト仕様書に落とし込み

1. テスターにわかりやすいように、テスト詳細や、前提条件などを用意。

2. 重要度「高」「中」「低」やテスト区分「正常系」「異常系」も設定します。
3. テスターは、期待値が実測値とあっているかを確認し、テスト結果をプルダウンから選択 **「OK」** **「NG」** **「PN」** を作成。また、不具合管理票にも記載しましょう。
   「OK」 は、期待値と実測値が同じである
   「NG」 は、期待値と実測値が異なっている
   「PN」 は、テスト環境不備やテストケース自体実行できない場合
4. バグ検出率や、テストケース消化率を算出できるように。ここは関数を使用して集計を楽にしましょう。

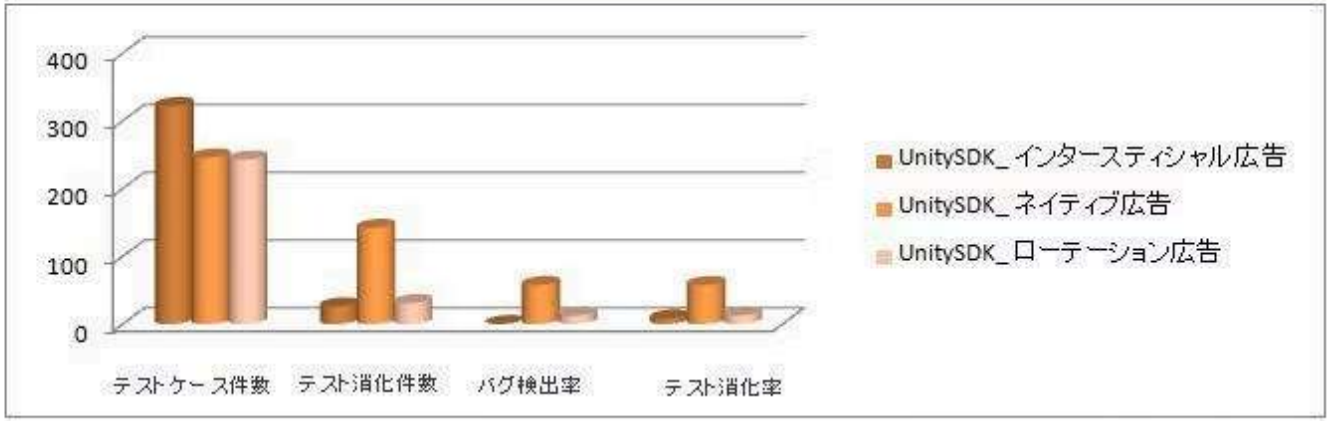
※テストを実行するための準備シートも用意

1. テストデータ
2. テスト環境の確認(DBに接続できる、対象のテーブルがある、phpのバージョンが正しい)

トップシート(ここで各シートの計算を表示しています)

1. テストケース件数
2. テスト消化件数
3. バグ検出率
4. テスト消化率

テスト仕様書



No	シート名称	テストケース件数	テスト消化件数	バグ検出率	テスト消化率(%)
1	UnitySDK_インタースティシャル広告	322	26	0	8.1
2	UnitySDK_ネイティブ広告	246	142	57.72	57.7
3	UnitySDK_ローテーション広告	243	31	12.76	12.8
	全シートテストケース総数	811	199	70.48	24.5

テストシート

テスト仕様書のテンプレート化

案件、その都度作成しては、作成工数やレビュー工数が膨れ上がってしまいます。
そのため、全体の機能のテンプレートを進めることにより作成者依存がなくなり、品質の偏りもなくなります。
また、テンプレートをバージョン管理することにより、どの機能がどのバージョンで管理されているかわかりやすくなります

※メンテナンスコストの問題もあるので、案件によります。

不具合の記載

テストケースには、テスト結果項目でNGを選択。

再現手順をバグ管理システムに登録する。
一般的には、JiraやRedmineが使われることが多い。Backlogも。

不具合の傾向分析利用

テストケースは、再利用できる材料ですし、不具合の傾向・分析をするには大事なアウトプットとなります。

また、JIRAなどテスト管理ツールに数値として残すことにより可視化でき、同様な機能・複雑な機能などのケース作成に再利用できます。