単体テスト

1.単体テストとは

単体テスト（ユニットテストと呼ばれることもあります）は、プログラムを構成する比較的小さな単位（ユニット）が個々の機能を正しく果たしているかどうかを検証するテストです。

通常、関数やメソッドが**単体テスト**の単位（ユニット）となります。 プログラムが全体として正しく動作しているかを検証する結合テストは、開発の比較的後の段階でQAチームなどによって行なわれることが多いのとは対照的に、**単体テスト**は、コード作成時などの早い段階で開発者によって実施されることが多いのが特徴です。

2.単体テストの利点

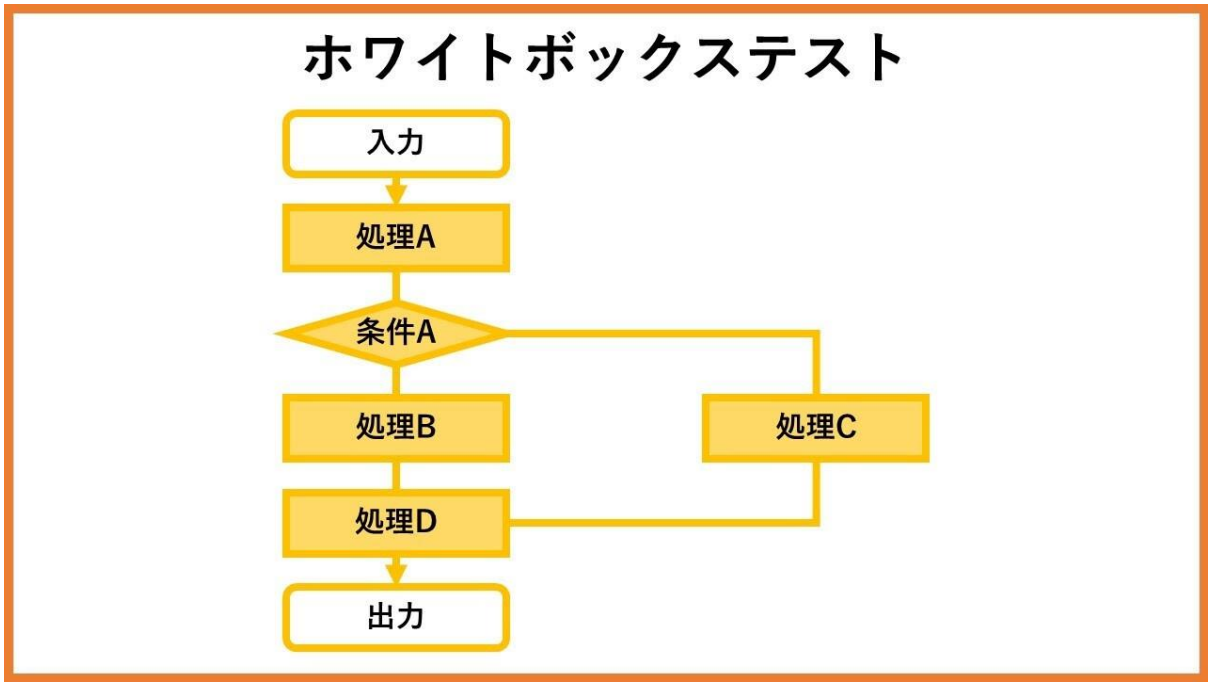
・モジュールが結合される前の段階でテストが実施されるため、問題の原因の特定や修正が容易。開発全体のバグ修正コストを下げる効果が高い。

・コードの内容をよく理解している開発者によって、コード作成と同時か直後に（または、『テスト駆動型開発』Test Driven Development:TDD と呼ばれる開発手法ではコードの作成よりも前に）テストケースが作成されるため、妥当性の高いテストケースを資産として残すことができ、後の拡張開発や改修時にも再利用できる。

3.単体テストの手法

3-1.ホワイトボックステスト

ホワイトボックステストは、プログラムの内部構造を理解していることを前提に、意図・想定した通りに動作するか確認を行うテスト手法です。プログラムの内部処理内容の網羅基準をどの程度に設定するかによって、準備にかかる時間・労力と得られる結果が変わってくるため、テストを実施する状況によって適切な基準を設定することが重要となります。



ホワイトボックステストの主な網羅基準は以下の通りです。

・命令網羅

全ての処理が最低1回は通るように設定。

・分岐網羅

全ての分岐を最低1回は通るように設定。

・条件網羅

条件分岐に対して真・偽を各1回は通すように設定。

・複数条件網羅

条件分岐に対してあらゆる組み合わせを通すように設定。

ホワイトボックステストは内部構造を隅々まで理解していることが重要となるため、一般的にテストの実施は開発者自身により行われます。

3-2.ブラックボックステスト

ブラックボックステストとは、プログラムの内部構造を考慮せず外部仕様を満たしているかどうかを確認するテスト手法です。主にデータのインプット・アウトプットにフォーカスしてシステム・ソフトウェアの全体を確認していきます。敢えて内部構造を考慮せずにテストを実施することで、開発時に予想や想定をしていなかった不具合等を発見できることが大きな特徴です。

ブラックボックステストのテスト項目・テストケースの作成は、一般的に外部仕様書を参考にして行われます。
こちらは結合テストのSTEPで詳しく触れていきます。

4.テストケースの考え方

テストケースの種類

また、ソフトウェアをテストする方法を示すテストケースには、正常系と異常系という2つの種類があります。

種類	**説明**
正常系	想定している入力に対して期待通りの出力を行うか
異常系	想定していない入力に対して問題なく対処できるか

いろんな種類のテストについて、正常系と異常系をもとにテストの手順を書いていくのがテストケース、ということになります。また、テストケースは手順書なので、読み手がいることになります。読み手が分かるような文章を心がけて書くことが重要になります。

4-2.テストケースの作り方

前で書いたテストケースの例は、必要なテストケースのひとつでしかありません。一桁の自然数同士のかけ算、つまり1〜9同士の掛け算になるので、全部で81通りの組み合わせがあります。
また、入力可能性という意味では無限にあり得ます。不正な入力として0や-1が入力される可能性も考慮しなければなりません。
この組み合わせについて、すべてのケースをテストするのは大変で、コストもかかります。
このようにテストケースが多いときに、品質をある程度保ちつつケースを減らす方法として、次の4つがあります。

1. 同値分割
2. 境界値分析それでは、実際にテストケースの作り方についてまとめていきます。テストにはユニットテストやシステムテストなどいろいろな種類があると書きましたが、基本的な作り方は次のような形になると思います。

順番	項目	内容
1	前提条件	テスト対象の前提となる値や状態はなにか
2	手順	どのようにテストするか
3	入力値	どのような値を入力するか
4	期待する結果	どのような結果を期待しているか

3.
4. 例えば、分かりやすい例として「一桁の自然数同士のかけ算をする計算機能」のテストケースについて考えてみます。一桁の自然数とはつまり1〜9の値を取りうるのですが、次のようなテストケースができます。

項目	内容
前提条件	5が入力されている
手順	かけるボタンを押す
	入力値を押す
	計算ボタンを押す
入力値	6
期待する結果	30が表示されている

5.
6.
7. ペアワイズ法
8. エラー推測

それぞれについて説明します。

4-3.同値分割

同値分割は入力をグループ化して、有効なものと無効なものに分けるやり方です。例えば「一桁の自然数」が入力だとすると、

- 1より小さい
- 1〜9
- 9より大きい

の3つのグループに分けることができます。つまり入力値として、例えば0、5、10の3つにすればよい、ということになります。

4-4. **境界値分析**

同値分割において、経験則的に「同値グループ間の境界にバグが発生しやすい」ということが分かっています。上の例でいうと、0と1、9と10などです。このような境界値では等号や不等号のミスなどでバグが起きやすくなるため、これを境界値分析で検出することができます。

4-5. **ペアワイズ法**

「ほとんどの不具合は1つまたは2つの要因によるものである」という経験則をもとにした方法で、たくさんある要因のうち「2つの要因の組み合わせだけは網羅する」、という観点で値を選ぶのがペアワイズ法です。

上の計算機能の例では要因は2つですが、テストの対象によってはもっと多くなるパターンも出てくると思います。ペアワイズ法を用いることで、組み合わせの数が多いときにテストケースを大きく削ることができます。

一方で、ペアワイズ法では本来検出できたはずのバグを取りこぼす可能性もあります。ペアワイズ法だけでテストせずに、組み合わせを選ぶときの参考にする程度でよいのかな、と思います。

4-6. **エラー推測**

これは上の3つの方法論とはちょっと変わっているのですが、「テストケースを作る人の経験に基づいてエラーが起きそうな値を決めるやり方」です。たとえば「一桁の自然数」という入力値に対して、負の数やヌル文字、空白、全角文字や小数などを用いてテストします。

4-7. **テストケースの観点**

ここまででテストケースの作り方と、入力値の選び方について書いてきました。最後に、テストケースを作るときの観点をざっとあげてみます。テストケースを作る対象のシステムによって違うと思いますが、参考になればと思います。

全般

- 入力に対する結果は正しいか
- 想定していない入力に対処できるか
- 入力の型や文字コードの種類は想定されているか
- 不適切な入力は想定されているか
- リクエストに対するレスポンスは正しいか

データ

- レコードは正しく更新されているか
- トランザクションは考慮されているか

セキュリティ

- 脆弱性は考えられないか
- 認証、認可のロジックは問題ないか
- 認証、認可失敗時の動作は問題ないか

エラー

- エラー処理はされているか
- 例外処理はされているか

画面

- データがないときの処理・表示は問題ないか
- 文言は正しいか
- 画面の遷移先は正しいか

アクセス

- 多重実行は考慮されているか

- 遅いネットワーク環境での利用は考慮されているか
- 処理がキャンセルされた場合は考慮されているか
- 同時アクセスは考慮されているか
- 同一ユーザの複数端末からの利用は想定されているか
- 利用環境の差異（OSやブラウザなど）は想定されているか
- 利用者の役割（ゲスト、管理者など）は考慮されているか

環境

- 実行環境の差異（開発環境、本番環境など）は想定されているか
- 時間やタイムゾーンの問題はないか

通信

- APIなどの通信先は正しいか
- APIなど通信先のステータスは考慮されているか
- 非同期処理のタイミングによるデータの有無は考慮されているか
- 非同期処理は必要なところでされているか
- メールやチャットなどへの通知は行われているか、送り先は正しいか

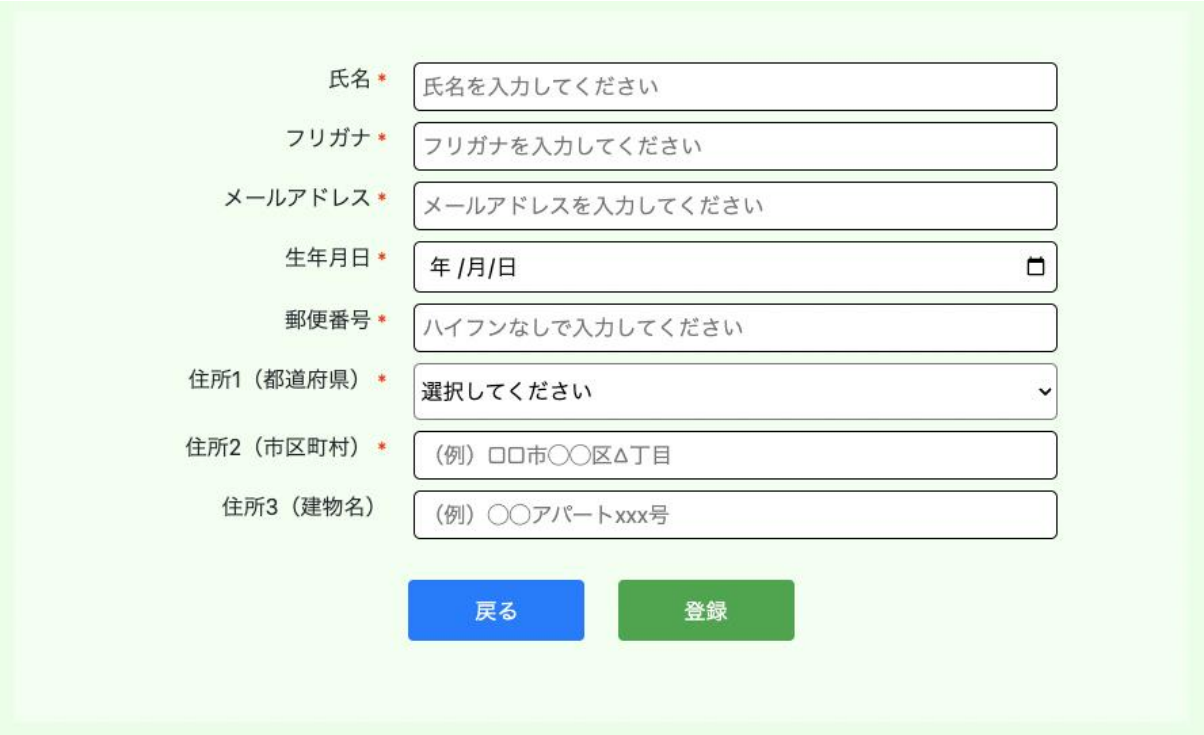
サンプル画像

5.課題
単体テスト仕様書の作成

5-1. テスト仕様書

STEP7で作成した管理システムの単体テスト仕様書を作成していきます。
とはいえ、いきなり「作成してください。」では難しいと思いますので、まずは例を用意しました。

例えば以下のような登録フォームがあります。
このフォームの単体テスト仕様書を作成していくとします。



5-2. フロントエンドのテスト項目

まず思いつくのは画面表示についてでしょうか。

- 入力項目は設計と合っているか
    - 必要な入力項目はあるか
    - 余計な入力項目はないか
- 項目名は合っているか
- プレースホルダーの文字列は正しいか
- 住所1の選択肢は正しいか（47都道府県、誤字脱字）
- 住所1の選択肢のvalueは都道府県コードに則しているか
- 必須項目に対して必須マーク(*)がついているか
- 必須ではない項目に必須マークがついていないか
- 戻るボタンが存在しているか
- 登録ボタンが存在しているか
- 全体的にレイアウトの崩れはないか
- 画面縦幅、横幅を変更した際のレイアウト崩れはないか

パッと思いつくのは上記くらいですが、これらをもとにテスト仕様書を作成し、レビューが行われて過不足について指摘されるので修正していきます。

また、同じCSSでもブラウザによって見た目が変わってくるので、見た目の部分については対応ブラウザ（要件定義の段階で決められています）全てでチェックする必要があります。
つまり、仮に見た目のチェック項目数が30個あって、対応ブラウザがGoogle Chrome, Microsoft Edge, Safariの3つだった場合90項目のチェックが必要になります。

例題:
上記のテスト項目以外に考えられるテスト項目をいくつか挙げてみましょう。

5-3. バックエンドのテスト項目

バックエンド的な部分の項目の方がテストする項目が多いです。
こちらもまたいくつかテスト項目例を挙げていきます。

- 氏名
    - 必須チェックがされているか
    - 255文字で工ラーとならないか
    - 256文字で工ラーとなるか
    - ひらがなの入力で工ラーとならないか
    - 全角カタカナの入力で工ラーとならないか
    - CJK互換漢字の入力で工ラーとならないか
    - 上記以外の文字を入力した場合工ラーとなるか
    - 工ラーメッセージは正しいか

- 郵便番号
    - 必須チェックがされているか
    - ハイフン込みで入力した場合工ラーとなるか
    - 7文字以外の入力で工ラーとなるか

    - 7文字の入力でエラーとならないか
    - 存在しない郵便番号の場合エラーとなるか
    - エラーメッセージは正しいか

- 住所1（都道府県）
    - 「選択してください」のまま登録しようとした場合エラーとなるか（必須チェック）
    - ブラウザの検証ツールを使ってvalueの値を存在しない都道府県コードにして登録しようとした場合にエラーとなるか
    - データベースに登録される値は選択した都道府県のコードになっているか
    - エラーメッセージは正しいか

上記は一部抜粋になりますので、このように各入力欄でそれぞれ考えられる項目を列挙していきます。
仮に一つの入力欄につき平均10項目としたとき、例に挙げたフォームは入力欄が8つあるので80項目、その他ボタンの遷移先やセキュリティ面の項目を考慮するとフロントの項目と合わせて1画面で軽く100項目は超えてきます。

例題:
上記のテスト項目以外に考えられるテスト項目をいくつか挙げてみましょう。

5-4. STEP7（8）の単体テスト仕様書

前項で学習した内容をもとに、STEP7（STEP8まで実装した方はSTEP8）で失踪した内容に対して単体テスト仕様書を作成しましょう。
テスト項目はSTEP7（もしくは8）の課題を参照して仕様を確認しながら作成していきましょう。

テンプレートを用意しましたので、こちらをコピーして使ってください。

テスト仕様書の作成ができたら実際にテストを行い、バグやエラーについては修正を行いましょう。