# テスト設計のとっかかり

株式会社日立製作所
加藤大受
daiju.kato.fc@hitachi.com



## テストをどう考えてますか？




## テストの目的

1. バグを未然防止するための情報をプログラマに提供する。
2. ソフトウェアの稼働中に発生するリスクを予測するための管理用情報を提供する。
3. 特定の環境下ではバグがないことを保証する。
4. 容易に**テスト可能な設計法**を提供する。すなわち、検証が簡単で、不良の摘出、保守が容易な設計方式を提供する。
5. 要求仕様書に記述された仕様、あるいは、要求仕様書に記述していない仕様をもとに、対象ソフトウェアが正しく動作しないことを証明する。これは「ソフトウェアの破壊(breaking the software)」ともよぶ。
6. 対象ソフトウェアを検証する。すなわち、ソフトウェアが正しく動作することを確認する。

『実践的プログラムテスト入門』
～ソフトウェアのブラックボックステスト～
ボーリス・バイザー著／小野間彰、山浦恒央、石原成夫訳
「1.3.2 ソフトウェアテストの理由づけ」より　抜粋




## テストの最終目標

**品質保証を支援することにある。すなわち、品質に関連した情報を収集し、プログラマにその情報をフィードバックすることで、過去に発生した不良を防止し、ソフトウェアの品質をよい方向へと導くことである。**

『実践的プログラムテスト入門』
～ソフトウェアのブラックボックステスト～
ボーリス・バイザー著／小野間彰、山浦恒央、石原成夫訳
「1.3.2 ソフトウェアテストの理由づけ」より　抜粋




## テスト設計を考える前に




## 一概にテストといっても

- ◆機能テスト
- ◆単体テスト
- ◆結合テスト
- ◆システムテスト
- ◆性能テスト
- ◆負荷テスト
- ◆ストレステスト
- ◆ユーザビリティテスト
- ◆回帰テスト
- ◆環境テスト

テストアイテムは山ほどあり、どのようなテストを、どの程度の期間やるのかを考える作業が必要

**テストは単純作業ではなく、想像力と知識が必要な作業です**

## テスト設計とは

テスト設計とは、
「**どうやってテストをするか**」
を考える作業のこと。

その前に

テスト設計をするには、
「テスト戦略とテストアプローチ」
が必要




## テスト戦略とは

テスト戦略とは、
「一つ以上のプロジェクトのために、
組織またはプログラムが実行されるべき
テストレベルとそのレベルでテストを
定義するハイレベルのドキュメント。」

JSTQB『ソフトウェアテスト標準用語集 日本語版Ver 1.1』




## テストアプローチとは

テストアプローチとは、
「あるプロジェクトのためのテスト戦略を実現化したもの。
この中には、(テスト実施) プロジェクトの目的と評価済みリスクを基に決めた決定事項、テストプロセスの開始場所、適用するテスト設計技法、テスト終了基準、実施するテストの種別を含む。」

『ソフトウェアテスト標準用語集 日本語版Ver 1.1』より




## 効率的なテストを取り組む

効率的なテストを取り組むには、テスト戦略とテストアプローチの基、「テスト計画」が必要。
そのためには、まずは正しい「テストプロセス」を理解することが必要。

テストプロセス全体のアクティビティ

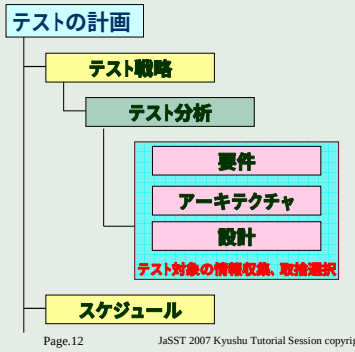
- 計画とコントロール
- 分析と設計
- 作成と実行
- 終了基準の検証とレポート
- 終了作業




## テスト設計を考えよう




## テスト設計までの全体像



# テスト計画

**IEEE std 829-1998のテスト計画書のテンプレート**

1.テスト計画書識別番号
2.目次
3.参考資料
4.用語集
5.概要
6.テスト項目
7.ソフトウェアのリスクに関する問題
8.テスト対象機能
9.テスト非対象機能
10.アプローチ
11.テスト項目の合否基準
12.一時定期基準と再開要件
13.テスト資料
14.テストタスク
15.環境条件
16.責任
17.スタッフの配置とトレーニング
18.スケジュール
19.プランニングリスクと対応策
20.承認




# テスト分析

**■テスト分析をなぜ行うのか**

◆完璧なテストはできない

◆スケジュール、リソース、環境などの制約を踏まえ、もっとも適したテストを遂行するため

何を対象とするのか

◆RFP、要求仕様書、アーキテクチャ定義書、設計仕様書
◆標準文書（RFC、ISO/IEC標準）、業界標準、社内標準
◆会議の議事録、設計メモ、メール．．．．．




# テスト要件を洗い出す

テスト計画時にどんなテストが必要なのかをすべて洗い出す。洗い出し方は何でもOK。必要な情報をすべて書き出すことが重要。

- テストマトリクス
- KJ法
- マインドマップ

機能要求と非機能要求と分けて考えてみよう



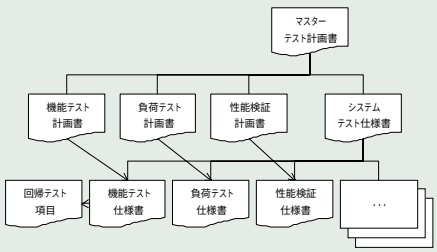

# テスト分析

計画したテストをISO9126の品質特性にマッピングして、品質が保証できるかを分析した例

| | | 実施 | 担当 | ISO/IEC 9126 品質特性 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 機能性 | 信頼性 | 使用性 | 効率性 | 保守性 | 移植性 |
| 機能テスト | 設計 | | QAエンジニア、テスター | | | | | | |
| | 実装 | ○ | | ○ | △ | | △ | | |
| | β版 | ○ | | ○ | △ | | △ | | |
| | テスト | | | | | | | | |
| 回帰テスト | 設計 | | QAエンジニア、テスター | | | | | | |
| | 実装 | ○ | | ○ | △ | | △ | | |
| | β版 | ○ | | ○ | △ | | △ | | |
| | テスト | ○ | | | | | | | |
| システムテスト | 設計 | | 要求管理者、品質管理者 | | | | | | |
| | 実装 | ○ | | ○ | ○ | ○ | ○ | | |
| | β版 | ○ | | ○ | ○ | ○ | ○ | | |
| | テスト | ○ | | ○ | ○ | ○ | ○ | ○ | ○ |



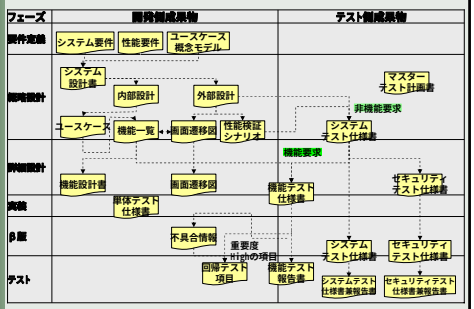

# テスト関係のドキュメント



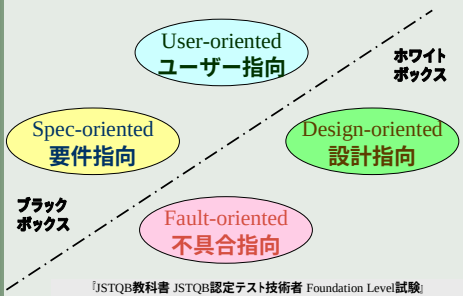




# 開発側成果物とテスト側成果物



## テスト設計の視点




『JSTQB教科書 JSTQB認定テスト技術者 Foundation Level試験』
P14. 図1.1 テスト設計の視点




## テスト設計のポイント

**少ないテストで漏れなく網羅**

そのためには

**テスト対象を網羅するための３つの視点**

**要件ベース　設計ベース　コード･ベース**




## テスト対象を網羅するための3つの視点

| 視点 | 概要 |
|---|---|
| 要件ベース | 業務要件及びそれを機能仕様に落としたレベルでソフトウェアやシステムを網羅する。内部状態を知らない状態のテストを行う。プログラムの内部やシステム設計を知ったうえで、機能仕様から見た動作をテストすることもある。 |
| 設計ベース | 機能要求として現れない、設計として考慮していることを網羅する。要件ベースを補う形で技術要素特有のリスクに関して考慮する。基本的にグレーボックス・テストを行う。 |
| コード・ベース | 一般的にホワイトボックス・テストと呼ばれる、関数やメソッド単位ごとにプログラム構造を網羅しているかを確認するテスト。主に単体テストで実施する。 |




## 要件ベースのテスト設計

**要件からテストを設計する方法のこと。**
**要件からテストケースを考える手法としてデシジョンテーブルテストなどがある。**

デシジョンテーブル（判断表）

| 見出し部 | |
|---|---|
| 条件記述部<br>（行動の全ての条件を記述する部分） | 条件指定部<br>（条件の成否、条件の範囲などを指定する部分） |
| 動作記述部<br>（行動内容の全てを列記する部分） | 動作指定部<br>（上の条件が満たされた場合の行動を指定する部分） |

## デシジョンテーブルの記入法

| | 指定形式 | 意味または内容 |
|---|---|---|
| 条件の指定 | Y | 規則を満足するには、この「Y」に対応する条件が満たされなければならない。（条件成立） |
| | N | 規則を満足するには、この「N」に対応する条件が満たされなければならない。（条件不成立） |
| | 語句、値またはコード | 条件記述部を補足する形で、この語句、値またはコードによって完成させる。この完成した条件が満たされなければならない。コードを用いる場合、その意味を注記する。（条件成立） |
| | －または＃ | 規則を満足するには、「－」か「＃」に対応する条件は無関係または論理的に成立しない。（条件無関係） |
| 動作の指定 | X | 規則が満足されると、記述された動作を実行する。（動作） |
| | 語句、値またはコード | 動作記述部を細くする形で、語句、値またはコードによって完成させる。この完成した動作が実行され、コードを用いる場合、その意味を注記する。（動作） |
| | －または＃ | 規則が満足されると、記述された動作は実行されない。（無実行） |




## デシジョンテーブルの例

会員登録処理で、名前と住所の記載があれば「会員」、名前と電話番号ならば「準会員」の場合とすると、

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 条件 | 名前の記載あり | Y | Y | Y | Y | N | N | N | N |
| | 住所の記載あり | Y | Y | N | N | Y | Y | N | N |
| | 電話番号の記載あり | Y | N | Y | N | Y | N | Y | N |
| 行動 | 会員登録 | X | X | | | | | | |
| | 準会員登録 | | | X | | | | | |
| | 登録しない | | | | X | X | X | X | X |

## コード・ベース視点のテスト設計

コード・ベース視点のテスト設計はホワイトボックステストで、代表的なテスト設計技法として、「制御パステスト」がある。

制御パステストとは、モジュール内のロジックをすべて通すようにテストデータを与えるテスト

```
pubic void foo(int x, int y){
   if ( x != 0) {
        y = y / x ;
        if ( y > 0 ) {
             y = y -1 ;
        }
   }
}
```

(x=1, y=1), (x=0,y=0), (x=1, y=0) という3種類の入力値を利用すると、すべてのパスが通る




## コード・ベースのテスト設計

コード・ベースのテストの場合、つまり、ホワイトボックステストでは、ツールを利用すると効率よくテストが可能

- ◆カバレッジ計測ツール
- ◆メトリクス計測ツール
- ◆単体テストツール
- ◆コード解析ツール
- ◆プロファイルツール
- ◆・・・・・・・




## 効率よくテスト設計するために




## テスト設計の第一歩

- ◆機能要求
  - ●機能テスト
  - ●テストケースを使ったテスト
- ◆非機能要求
  - ●性能テスト、負荷テスト
  - ●シナリオを使ったテスト

それぞれテスト設計のやり方が異なる




## テストケース設計

**テストケース**とは、テストを行う際に、プログラムにどのような入力を与え、その結果としてどのような出力が得られるべきかを記述したもの。

テストケースの書き方

- ◆目的
  - ●テストによって得たい情報
- ◆入力条件
  - ●入力するデータの値、入力方法や入力するときのソフトウェアの状態
- ◆実施手順
  - ●テストを実施する手順全般。正式にはテスト手順書に記載
- ◆期待結果
  - ●期待される結果。期待される結果は、仕様書を基にした値、他のテスト済みの製品との比較、過去のテスト結果、既存システムとの比較から導かれる。




## テストケース設計

効率よいテストとは最低限のテストケースで、漏れなくテストすること

効率を上げるには、テスト設計技法を知ることが重要

- ◆同値分割
  - ●入力値を有効値の集合と無効値の集合に分け，各集合から代表値を選び，その代表値を与えてプログラムの動作を確認するテストのこと。有効値も仕様や内部構造などによってそれ以上に分割することもできる。
- ◆境界値分析
  - ●入力データとして処理結果が変わる境界の値を与えて，プログラムの動作を確認するテストのこと。
- ◆グレーボックステスト
  - ●ソフトウェアやシステムの内部構造を知った上で、ブラックボックステストのように、機能仕様から見た動作をテストする方法。

## テストケース設計

3歳未満、3歳以上6歳未満、6歳以上でプログラム上の処理が違う場合、の同値クラスと境界値は？

同値クラス：3歳未満、3歳以上6歳未満、6歳以上という3つに分類される。
もし、年齢として適当でない数値や数値以外のものも入力される可能性を考えるならば、「同値クラス」は4つに分かれる。
境界値：2歳、3歳、5歳、6歳




## テストケース設計

作成時のポイント１

各テストケースに優先順位をつける
- ◆テスト実施ができないケースを考慮
  - ●テストケースを作成する時間があってもテストが実施できるかどうかは別。
  - ●テスト設計カバレッジ≠テスト実施カバレッジ
- ◆回帰テストでの利用を考慮
  - ●優先順位が高いテストケースは回帰テストでの再利用が効果的である




## テストケース設計

作成時のポイント２

異常系テストケースを作成する
- ◆正常系テスト
  - ●正しい結果が返るテストケース
  - ●正しいエラー結果が返るテストケース
- ◆異常系テスト
  - ●本来起こらない動作やユーザーが起こしやすいミスによって引き起こされる動作を基にしたテストケース

正常系テストケースと異常系テストケースの比率を考慮して設計




## テストケース設計

作成時のポイント３

トレーサビリティを確保
- ◆トレーサビリティの確保によって意味のあるテストであることを証明
  - ●機能要求と詳細機能設計のトレーサビリティの確保
  - ●詳細機能設計とテストケースのトレーサビリティの確保

機能IDをつけて管理することが望ましい




## シナリオテストの設計

シナリオテストの設計は、業務フローやユースケースを基に行う。
1. 業務フローから基本フローを導きだし、シナリオを洗い出す。
2. 例外フローを業務フローの中で不自然な流れから洗い出す。

洗い出したフローを仕様書として加工

各シナリオパターンで表示されるが画面、処理とシナリオが流れるための必須条件、事後条件を記載してテスト仕様書とする。




## シナリオテストの実施

本番を意識して実施
- •データ
- •システム環境

テスト結果の分析が必須

シナリオテストを利用し、性能テスト、負荷テストを設計する
- ◆ シナリオテストはフローが正しいことを確認するテスト
- ◆ 性能テストは性能要件を満たすかどうかを確認するテスト
- ◆ 負荷テストは予想される負荷がかかっても正常に動作するかを確認するテスト

## 性能テストの設計

性能要件を満たすかを確認するテスト
- ◆簡易テスト：システム構成の決定後に実施
- ◆本テスト：システムテスト時に実行

早め早めにテストを実施

レイヤーを分けて実施できる場合は、レイヤーごとに測定を実施
Webアプリケーションの場合では、
- ●DB層での測定
- ●ミドルウェア層での測定
- ●アプリケーション層での測定




## 性能テストの設計

テストケースの作成
- ◆性能要求を基に、測定すべき内容を記載
  - ●メニューを選択後のレスポンス時間
  - ●問い合わせの処理時間
  - ●画面の表示時間
  - ●etc.
- ◆性能測定を行う環境
- ◆測定手順
- ◆期待される結果（時間）

できる限り具体的に記述する




## 負荷テストの設計

テストケースの作成
- ◆要求される負荷を基に、測定すべき内容を記載
  - ●CPU負荷
  - ●I/O負荷
  - ●ストレージ負荷
  - ●ネットワーク負荷
  - ●同時アクセス負荷
  - ●etc.
- ◆負荷測定を行う環境
- ◆測定手順
- ◆期待される結果(動作、対応負荷)

できる限り具体的に記述する




## 回帰テストの設計

回帰テストとは
「変更により、ソフトウェアの未変更部分に欠陥が新たに入り込んだり、発現しないことを確認するため、変更実施後、既にテスト済みのプログラムに対して実行するテスト。ソフトウェアや、実行環境が変わるたびに実施する。」

JSTQB『ソフトウェアテスト標準用語集 日本語版Ver 1.1』




## 回帰テストの設計

回帰テストの設計
- ◆発見された不具合の修正確認
- ◆不具合の修正による影響確認
- ◆機能テストの優先順位の高いテストケース

回帰テストのポイント

回帰テストの実施条件をテスト計画で明確に
- ◆タイミング
- ◆実施回数
- ◆合格条件




## 閑話休題
## ～よくある話ですが～

## 仕様書がないけどテストする

◆テスト対象物の仕様書がない、あるいは仕様書がまったく更新されていない。あるものは、
- ●稼働システム
- ●ソースコード
- ●ビルド環境

◆顧客の要望により修正しなければならなくなった。どうやってテストするか。

こういう状況でもきちんとテストするのがテストエンジニア

**まずは分析**




## テスト対象を分析しよう

◆ソフトウェアの動作環境
- ●システム構成（ハードウェア、ネットワーク）
- ●ソフトウェア構成（OS,ミドルウェア）
- ●ソフトウェア設定

◆ソースコードの分析
- ●アーキテクチャの分析
  - ✓構造図の作成
  - ✓言語によっては仕様書作成ツールに通してみる
- ●DB利用時：スキーマ構造

**◆ユースケースの分析**
- ●業務フローの作成




## 機能テスト設計

◆機能一覧作成
- ●機能を項目（メニュー階層）ごとに洗い出す
- ●機能の優先順位を作成

◆権限マトリックスの作成
- ●権限で表示・非表示される機能を洗い出す

ここまでできれば後はテストケース作成

◆機能一覧をベースにテストケースを作成
- ●優先順位によって作成するテストケース数を決定




## 非機能要求のテスト

◆性能テスト
- ●既存システムの顧客からのヒアリングで性能要求を決定
- ●性能劣化がおきていないことを確認

◆負荷テスト
- ● 既存アクセスログやソフトウェア構成から必要な要件を洗い出す
- ● 顧客からのヒアリングで対応負荷要件を決定
- ● 業務フローからシナリオを作成し、テストケースを作成

**◆システムセキュリティテスト**
- ● 既存システムのセキュリティ状況を洗い出す
- ● 現状からセキュリティ要件を洗い出し、顧客にヒアリング
- ● セキュリティ要件を基にセキュリティテストのテストケースを作成




## 作成した資料を仕様書にする

◆作成した資料
- ● システム設計書
- ● ソフトウェア設計書
- ● ソフトウェア構造書
- ● ユースケース定義書
- ● 性能要件
- ● セキュリティ要件
- ● 各テスト計画書
- ● 各テスト設計書
- ● 各テスト仕様書（テストケース）

◆報告書の作成
- ● テスト結果のテスト報告書を作成

**こういう開発・テストは正道ではありません**




## テスト設計に磨きをかけるために

## バグの分析

発見されたバグを分析することによって
- ◆エラーの推測
- ◆品質の分析
- ◆回帰テストの効果測定
- ◆今後の工数･テストケース分析のデータ

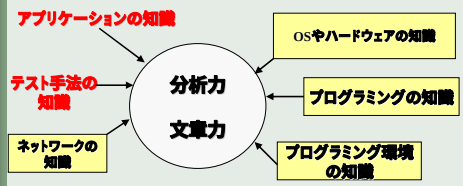
バグ管理システムを利用しよう

バグをデータとして生かしていくためには
- ◆バグ管理システムの構築が必要不可欠
- ◆バグの内容、手順を分かりやすく記載
- ◆バグの深刻度、レベルを分類し、優先順位付けを行う。回帰テスト設計時のデータとして利用




## 知識の習得

テストエンジニアに必要なスキル




いろんな角度からテスト対象を分析できる知識を習得しよう
テスト技法の習得も忘れずに




## テストに悩んだら

- ◆機能テストのテストケースに悩んだら開発者に相談しよう
- ◆入力値の組み合わせが多すぎてテストケースが爆発するときは組み合わせを減らす技法(All-Pair法、直行表)を利用しよう
- ◆シナリオテストに悩んだらユーザーに相談しよう
- ◆テスト技法を知りたいときはテスト関連書籍を参照しよう
- ◆テストを行う時間があまりにないときは、テスト不足による品質リスクを洗い出して、PMと交渉しよう
- ◆どんなにテストしても品質が上がらないときは、リスクを洗い出して、PMに相談しよう
- ◆他の人の意見を聞きたいときはTEFのMLを利用しよう
- ◆もっといろいろと勉強したいときは、TEFの勉強会やJaSSTに参加しよう

いろんな形で悩みを解決していくことができるでしょう




## ご静聴ありがとうございました