テスト設計コンテスト
チュートリアル
ちびこん編　2025

2025/05/24
井芹 久美子（テスト設計コンテスト審査委員会）

このチュートリアルについて

テスト設計コンテスト（テスコン）運営組織にて開催している
2つのチュートリアルのうち、こちらは比較的経験の浅い方向けです。
ソフトウェアテストについて今一度振り返りたい方も、お気軽にどうぞ。

想定聴講者は、

- テストケースの作成またはレビューを実施したことがある方
- テストケースやテスト手順をつくる前のアクティビティについて話を聞いてみたい方

テスト設計コンテスト参加の経験や予定の有無は
問いません

講演者とテスト設計コンテストの紹介

- テスト設計コンテスト（テスコン）について
  - テスト設計の出来を競うコンテスト
  - 目的
    - ソフトウェアテストを分析設計から行うことを周知し、**ソフトウェアテストエンジニアに対する教育の機会を提供**する
    - コンテストという形式をとることにより、**ソフトウェアテストが創造的な作業であり、楽しいということを経験してもらい**、若年層及び初級テストエンジニアからベテランテストエンジニアまでテストへの興味を高める
    - ソフトウェアテスト業界における**技術開発を**競技を通じ、**促進する**
  - 若手向けのU-30クラス、どなたでも参加できるOPENクラスがある
- 講演者について
  - テスト設計コンテスト U-30クラス審査委員。

テスト設計コンテストの目的は https://www.aster.or.jp/testcontest/index.html より引用。ただし太字は本資料作成者による

本資料におけるテスト開発プロセスと用語

本資料で採用しているプロセスとJSTQBのプロセスとの対応は以下の通り。

用語は、説明している場合を除き、基本的に本資料作成時点のJSTQB/ISTQBの用語定義に従います。https://glossary.istqb.org/ja_JP/search

図は　テスト設計チュートリアル ちびこん編 '21 資料　67ページより引用。ただし青点線枠は本資料作成者による

ゴール

次の2点を達成することを目指しています。

実装までのテスト開発プロセスのイメージを掴んでいただく

テストケースやテスト手順を
どのようにつくるのか、そのために
どのような情報収集や準備が必要か、
見てみましょう。

実装までのテスト開発プロセスの中で特に気を付けたいことを知っていただく

テスト活動では、テストの内容は
当然のこと、説明のしやすさや
ドキュメントの保守性など、幅広い
面で注意が必要です。
その中で、特に気を付けたいことを
ご紹介できればと思います。

コンテンツ

2つあるゴールにあわせた2テーマについてお話しします。

[テーマ1]
テストケース・テスト
手順をつくるまでの活動例

テスト要求分析を中心にお話しします。ただし、実際には反復的になることがありますが、シーケンシャルに説明します。
各テスト技法の詳細や、成果物の具体的な作成手順は説明しません。

[テーマ2]
審査コメントを参考に
気を付けたいポイント

テスコンU-30クラスの審査員が参加チームの皆さんにフィードバックしたコメントを分析し、見えてきたポイントを解説します。

テストの活動の助けになると考えた情報をお話ししますが

絶対的な解ではありません、考えるヒントにしていただければと思います

[テーマ1]

テストケース・テスト手順を
つくるまでの活動例

このアプリをテストする場合、どのように考えますか？

「三角形判定アプリ」は，ソフトウェアテスト学習用の、 AndroidOSとiOS向けのスマホアプリである．ユーザーは，三角形の3 辺の長さの値を入力する．判定ボタンを押下すると，入力された値を3辺の長さとする三角形が不等辺三角形，二等辺三角形，正三角形のいずれなのかが表示される．設定画面で、ライトモード/ダークモード、文字の大きさ（大/中/小）を選択し設定することができる。

補足
- 辺の長さとして有効な値は、1～999の整数。
- デフォルトはライトモード、文字の大きさは中。
- 利用状況調査のためアプリからサーバにログを送信する。ログは別のツールで解析する予定。

▲開発中の画面

30秒、各自で考えてみてください(あてませんのでお気軽に)

仕様は、本「ソフトウェア・テストの技法 第2版」2ページを参考に作成
画像は、テスト設計チュートリアル ちびこん編 '24 資料　11ページから引用

短い仕様でも気になることは沢山見つかる

テストした方がよいことは沢山ありそう。

不等辺三角形になるような3つの値を入力してみる

二等辺三角形になるような3つの値を入力してみる

正三角形になるような3つの値を入力してみる

辺の長さに数値以外を入力してみる

2つの辺だけ入力して1辺は未入力のままとする

1つの辺だけ入力して2辺は未入力のままとする

ダークモードで動作させてみる

最新バージョンのAndroidOSやiOSで動作させてみる

繰り返しインストール、アンインストールしてみる

判定にかかる時間を計測してみる

…

すぐにテスト実行を開始できそう？

思いついた内容をもとにテストすれば十分だろうか？

No.	アクション	期待結果（判定結果）
1	辺の長さ(3,3,3)を入力して判定ボタンを押す	判定結果として「正三角形」と表示される
2	辺の長さ(4,3,3)を入力して判定ボタンを押す	判定結果として「二等辺三角形」と表示される
3	辺の長さ(6,3,4)を入力して判定ボタンを押す	判定結果として「不等辺三角形」と表示される
4	辺の長さ(a,3,4)を入力して判定ボタンを押す	文字種エラーになる
5	辺の長さのうち1辺のみ(3)を入力して判定ボタンを押す	未入力エラーになる
6	辺の長さのうち2辺のみ(3,2)を入力して判定ボタンを押す	未入力エラーになる
…	…	…

一方で、分からないこと、曖昧なこともある

辺の長さの入力値について、文字の種類はどのくらいテストする？
古いOSもテストする？
インターネットに接続できない状態でもテストする？
ログについてもテストが必要？
どんなテストを重点的にする？　全機能、同じくらいでいいのだろうか？
使える工数はどのくらい？

テストがつくれそうな情報でも

納得できるテストがつくれるくらい十分な情報とは限らない

以降の説明の背景にある考え方

本資料で大事にしている考え方は次の3つ。

段階的に進める	多面的に考える	テスト全体を俯瞰しバランスをとる
仕様などから一気に具体的なテストケースやテスト手順を作成するのではなく段階的に考えていく	できるだけ多くの情報を集めて状況にあったテストを考える	テストケース1件ずつや「判定機能のテスト」といった細かい粒度で考えるだけでなくテスト全体の構造を意識する

この3つを実現するため、まずはテスト要求分析に取り組みましょう。

テスト要求分析

分からないこと、曖昧なことが沢山ある

辺の長さの入力値について、文字の種類はどのくらいテストする？
古いOSもテストする？
インターネットに接続できない状態でもテストする？
ログについてもテストが必要？
どんなテストを重点的にする？　全機能、同じくらいでいいのだろうか？
使える工数はどのくらい？

分からないこと、曖昧なことがあるので、まず情報を集める

テストの目的や責務は？	テスト観点(テストすべきこと)は他にないか？	テストの優先順や厚みは？	技術面や組織に関する制約は？

どんな情報を集めるとよさそうだろうか？

集めたい情報

一般的に、以下の情報は必要そう。

どのようなプロダクトか？

仕様は？　仕様書のほか、開発中のメモ等

利用シナリオは？

強み、弱みは？

過去バージョンの評判や故障はどうか？

ステークホルダーは誰でそれぞれの関心事は？

ユーザーは誰？　どんな関わり方か？

開発メンバーが不安に思っているところは？

営業やカスタマーサポートチームの関心は？

組織や経営層の期待は？

プロダクトの周辺は？

関連する法令は？　慣習は？

連携システムは？　同時に使うシステムは？

開発の状況や現在の品質は？

3H(初めて、変更、久しぶり)の部分は？

前プロセスのレビュー結果やテスト結果は？

3Hは　https://note.com/akiyama924/n/n79b902f69ddf　を参考にした

テストケースのもとにならない情報も集める

テストへの要求は、
テストケースを導くもの（エンジニアリング的テスト要求）のほか、
技術的・組織的な制限に関する情報（マネジメント的テスト要求）
があり、両方を集める。

エンジニアリング的テスト要求	マネジメント的テスト要求
想定する使い方	工数
ユーザや開発者がミスをしそうなところ	テストチームのスキル
自信があるところ、ないところ	利用できるツール
複雑な機能	今後の開発やテストの再利用の予定
…	…

テスト設計チュートリアル テスコン編 ‘22 資料　14ページ、
JaSST’22 Hokuriku　目指せテスト設計リーダー!! テスト設計チュートリアル　資料　26ページを参考に作成

集められない情報がある場合はどうするか？

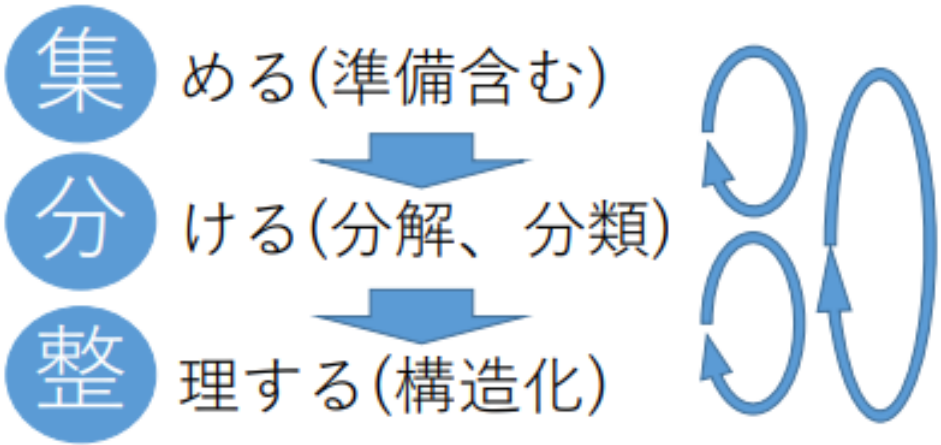
現実には、契約や組織の都合で集められないことがある。
集められない情報は推測して補うが、以下を注意する。

できるだけ関連情報を集める	ステークホルダーを巻き込む	情報が増えたときに備えて見直しやすいようにしておく
前提を明らかにする、確度を上げる	情報や思考の幅を広げる、ステークホルダーの納得感を醸成する	変わるかもしれない部分だと分かるようにする

集めた情報から多くを引き出す

集めた情報の中には、そのままではテストのインプットに
しづらいものや、異なる情報同士を組み合わせることで新たな
情報に繋がるものもある。

集めた情報は、理解を深めるために分解や構造化をする。



図は　テスト設計コンテスト'21　-　テスト要求分析チュートリアル　資料　57ページより引用

情報を分解・分類、構造化する例　その1

画面遷移について明らかにするために、分解・分類、構造化を試みる。

「三角形判定アプリ」にはTOP画面、入力画面、判定結果画面、設定画面がある。TOP画面で開始ボタンを押すと入力画面に遷移する。入力画面では、辺の長さとして値を入力できる。また、判定ボタンを押すと、判定結果画面に遷移する。判定結果画面では、判定結果を表示する。3辺の長さが入力済であり、すべて有効な値の場合は、その値に応じて三角形の種類を表示する。判定結果画面には、もう一度ボタンがあり、押すと入力画面に遷移する。それぞれの値は有効な値だが三角形が成立しない場合、判定結果画面には、三角形不成立エラーのメッセージを表示する。入力された値のうち1つ以上が有効な値でない場合は無効値エラーのメッセージを、未入力がある場合は未入力エラーのメッセージを表示する。TOP画面以外の画面には戻るボタンがあり、押すと一つ前の画面に戻る。ただし、判定結果画面の戻るボタンを2秒以上押した場合は、一つ前の画面ではなく、TOP画面に戻る。

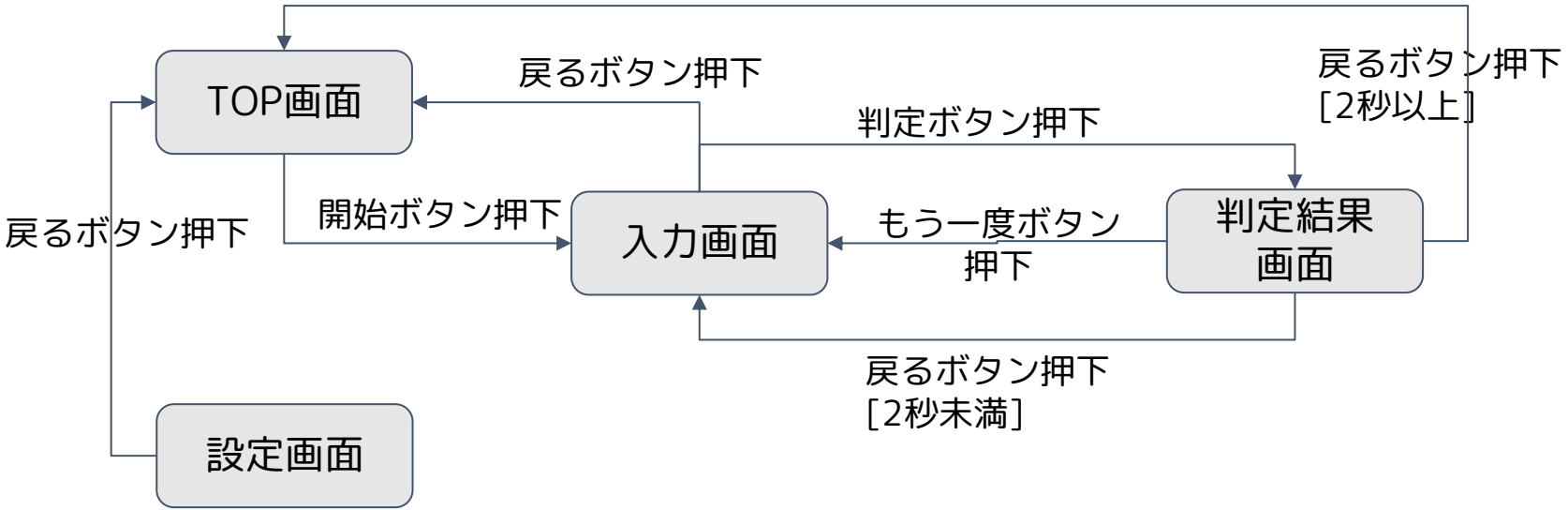
情報を分解・分類、構造化する例　その1

遷移元や遷移先の画面・遷移を引き起こすイベント・条件に分類する。

「三角形判定アプリ」にはTOP画面、入力画面、判定結果画面、設定画面がある。TOP画面で開始ボタンを押すと入力画面に遷移する。入力画面では、辺の長さとして値を入力できる。また、判定ボタンを押すと、判定結果画面に遷移する。判定結果画面では、判定結果を表示する。3辺の長さが入力済であり、すべて有効な値の場合は、その値に応じて三角形の種類を表示する。判定結果画面には、もう一度ボタンがあり、押すと入力画面に遷移する。それぞれの値は有効な値だが三角形が成立しない場合、判定結果画面には、三角形不成立エラーのメッセージを表示する。入力された値のうち1つ以上が有効な値でない場合は無効値エラーのメッセージを、未入力がある場合は未入力エラーのメッセージを表示する。TOP画面以外の画面には戻るボタンがあり、押すと一つ前の画面に戻る。ただし、判定結果画面の戻るボタンを2秒以上押した場合は、一つ前の画面ではなく、TOP画面に戻る。

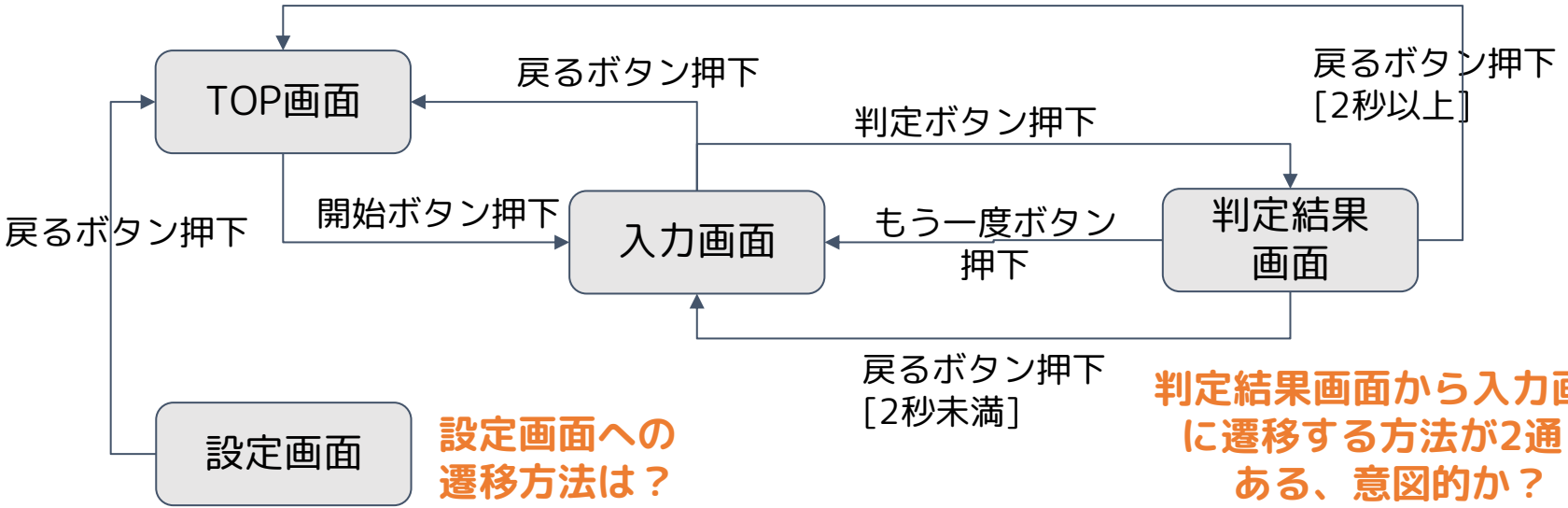
情報を分解・分類、構造化する例　その1

結果を図や表で表現し、全体を眺めてみる。
重複や抜けなど、気になるところはないだろうか？



情報を分解・分類、構造化する例　その1

結果を図や表で表現すると、曖昧な点や疑問が見えやすくなる。
必要に応じて更に情報収集し、テストへの要求を理解する。



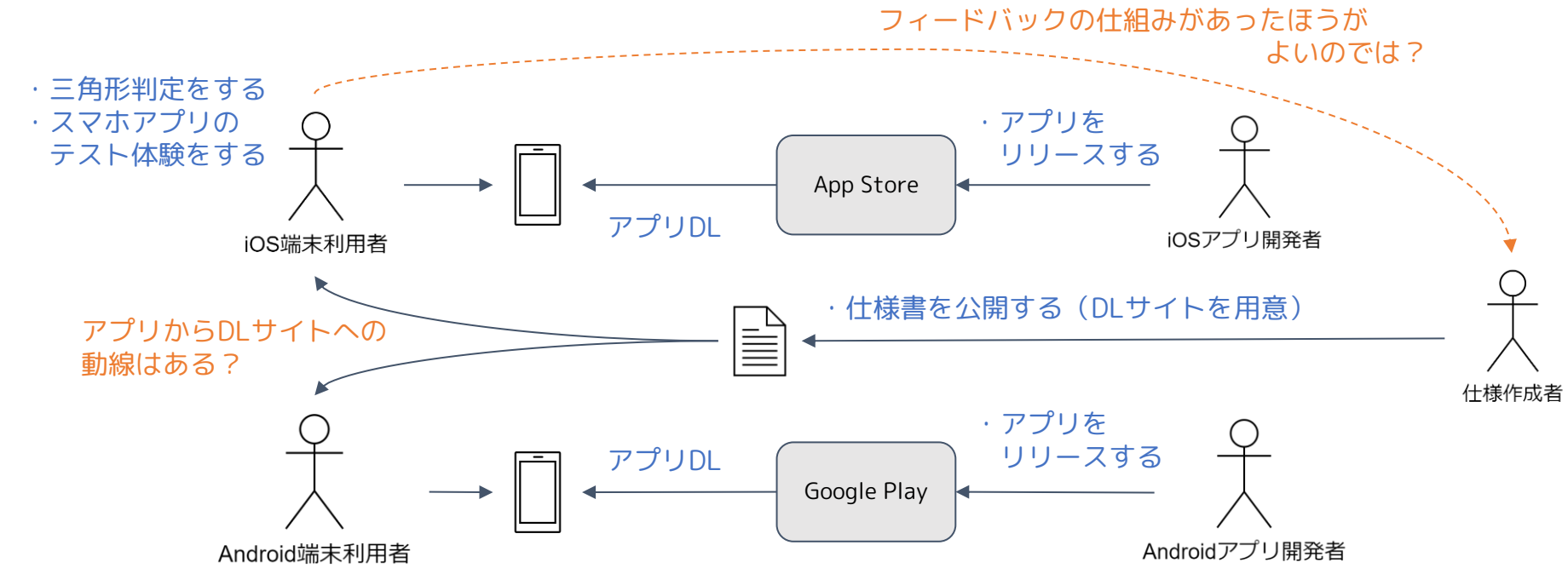
なお、この例では、更に状態遷移表のように表で表現しても効果的である

情報を分解・分類、構造化する例　その2

テスト対象である「三角形判定アプリ」を取り巻く環境の図示を試みる。
まず、どのような人やシステムが、どのように関係するかを洗い出す。

- 人
  - ユーザー　…公開されるアプリや仕様書をダウンロード (DL) する
    - iOS端末利用者
    - Android端末利用者
  - 開発チーム
    - 仕様作成者　…仕様書を作成し、公開する
    - iOSアプリ開発者　…アプリをリリースする
    - Androidアプリ開発者　…アプリをリリースする
- システム　…アプリをDL、インストールできるようにするために必要
  - App Store
  - Google Play

情報を分解・分類、構造化する例　その2



対象OSバージョンのほかに、
端末も複数機種・複数メーカー必要そう

気になることを遠慮せず書き込む

テスト設計チュートリアル ちびこん編 '24 資料　22ページより引用

テストの目的や責務を決定する

理解したテストへの要求をもとにするが、担当範囲外の品質に関する活動の情報も集めておき開発全体から見て適切な目的や責務にすることが重要。

責務	プロダクトのどの部分が対象か？ どのテストレベル、どのテストタイプが対象か？ 例　ネイティブアプリケーション部分のみをテスト対象とする。
目的	テストで得たい情報は何か？ テスト結果を受けて行われる活動のために何をすべきか？ 例　本番リリースできる品質であることを確認する。

抽象的過ぎると解釈がぶれ、後続の活動がしづらい。
一方、具体化には情報収集等の手間がかかるので必要な程度を見極める。

例は、ASTER U-30テスト設計コンテストストプロジェクト要求補足書2024 Ver. 1.0　を参考に作成

集めた情報とそこからテストの目的や責務を検討した例

統合テストの後、人事部による受入テストの前に行う
非機能テストはまだ実施していない
ログ出力機能は直近のリリース対象外で、プログラム実装もしていない
社内にアクセシビリティガイドラインがあり適合が期待されている
このアプリは、今後も継続的なバージョンアップを行う予定がある
仕様書の作成は開発ドキュメントの作成経験が浅いメンバーが担当している

- 責務
  - アプリの機能と非機能について、システムテストを行う
    - ただし対象は判定機能と設定機能とし、ログ出力機能は対象外とする
- 目的
  - 受入テスト開始可能な品質であることを確認する
  - アクセシビリティガイドラインへの適合を確認する
  - 仕様書に対するフィードバックを行い仕様書の改善に貢献する

テスト観点を考える

集めた情報やテストの目的・責務を踏まえて、テスト観点を検討する。

本講演の「テスト観点」は、テストすべきことを指す。

- 三角形判定アプリのテスト観点の例
  - 正三角形など、三角形の形
  - 負荷
  - OSの種類やバージョン
  - 入力のしやすさ
  - テスト初心者など、ユーザーの種類
  - …

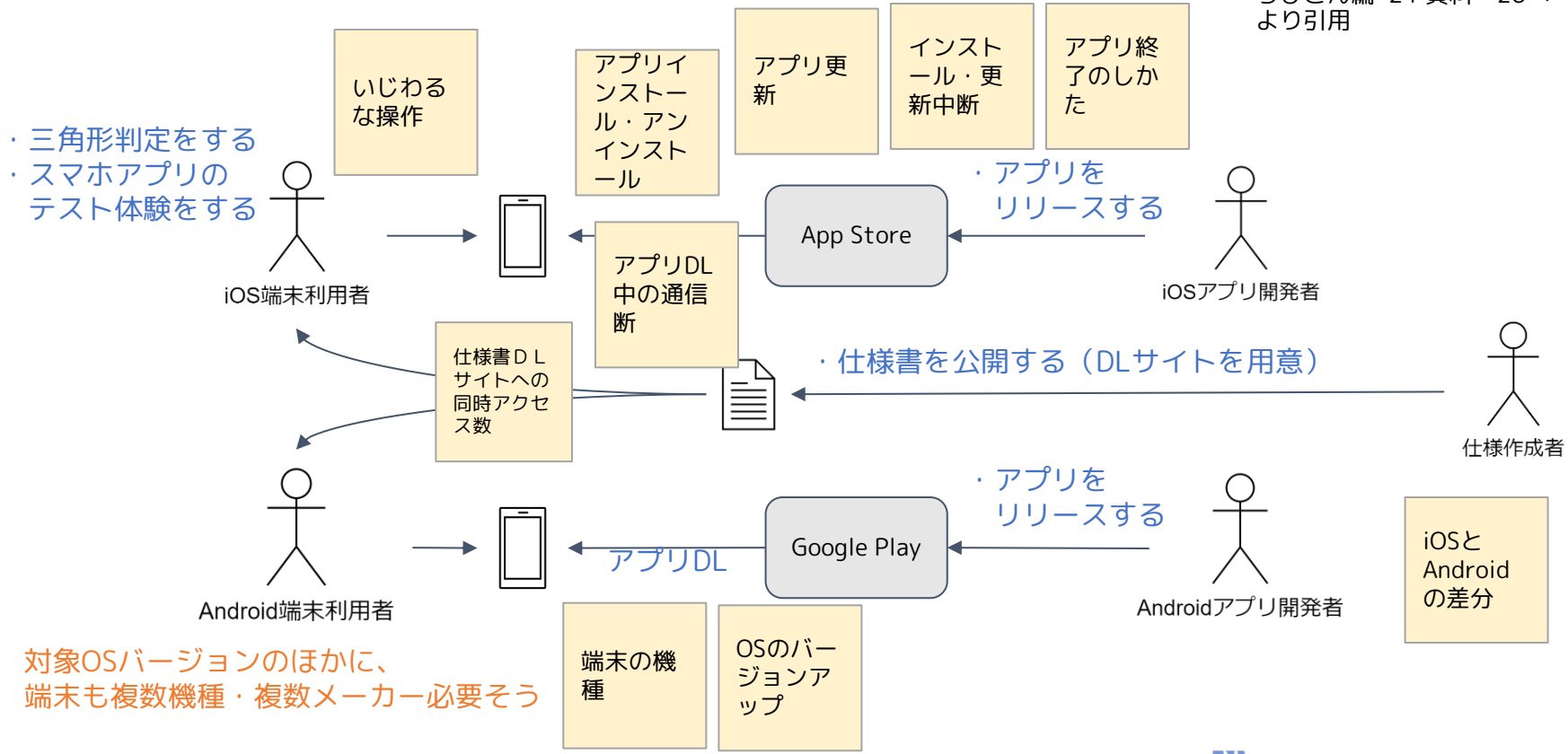
抽象度は様々で、そのままテストケースに登場するものも、テストケース作成時には具体化が必要なものも含む。

テスト設計チュートリアル ちびこん編 ‘21 資料　12ページを参考にした

集めた情報をもとにテスト観点を洗い出す

例えば、テスト観点だと思うものを書き込んでみる。

図は　テスト設計チュートリアル ちびこん編 '24 資料　26ページ より引用



テスト観点を考える

様々な方法を使って考えるとよい。

テスト対象自体の情報を基に考える	組織標準等まとまった情報を参考にする	起こりうる欠陥や故障から考える	抽象度を変えてみる	チームで知恵を出し合う
仕様書の情報を図や表に変えてみる、別々に書かれている情報を組み合わせて検討してみる	ISO/IEC25010等(SQuaREシリーズ)や6W2Hを参考にする。 とらわれすぎないように注意	例えば、三角形判定アプリで不正な判定結果が表示されるとしたらどんな場合？	例えば、文字種入力のエラーはエラーの一種、他にエラーはあるだろうか？	モブ作業や、経験がある人と意見交換する等。

生成AIを使える環境では、例えば、生成AIを利用して作成した叩き台をベースに検討を進める、ブラッシュアップしていく、といった方法も考えられる

グルーピングを通してブラッシュアップする

過不足や不揃いが見えたら、より良くするチャンス。

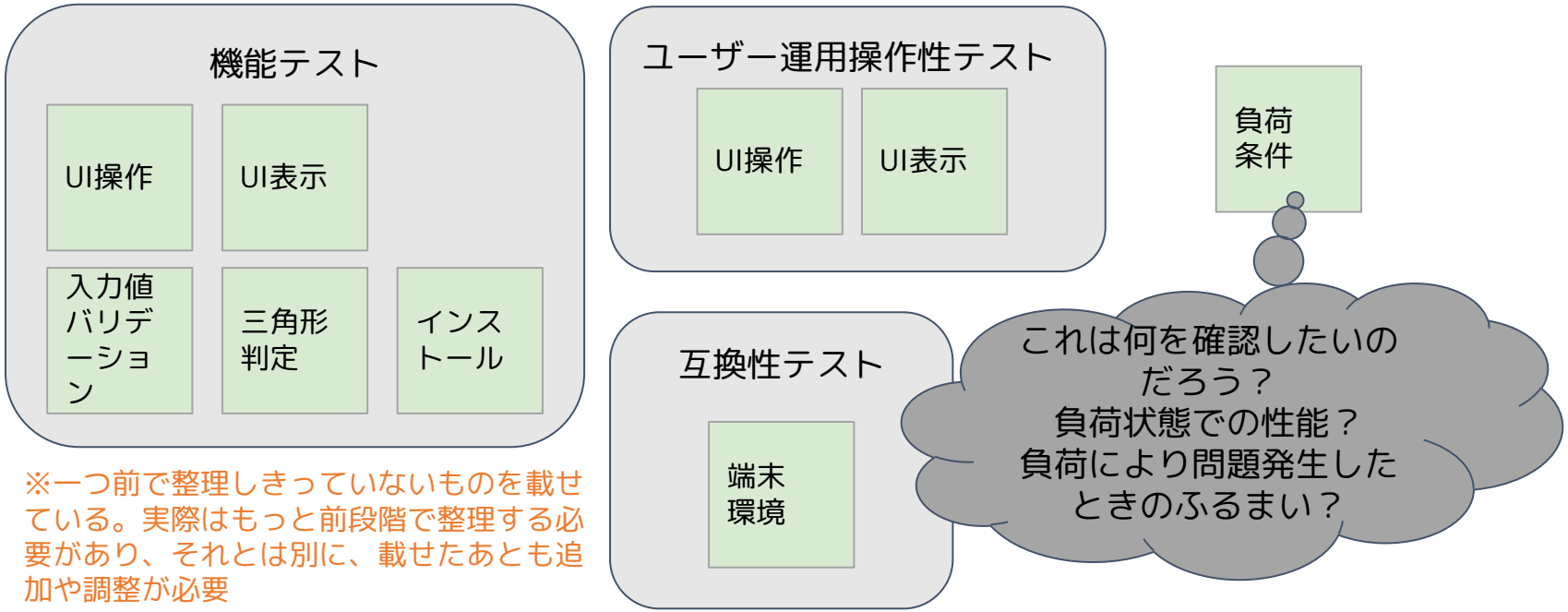
例は　テスト設計チュートリアル ちびこん編 '24 資料　28ページより引用

出した観点をグループ分けしてラベルをつけた例

※ここから、さらに整理が必要

グルーピングを通してテスト観点をブラッシュアップする

テストタイプでグルーピングした例。
曖昧なテスト観点が見つかったら、意図を再確認したりグループを見直す。



※一つ前で整理しきっていないものを載せている。実際はもっと前段階で整理する必要があり、それとは別に、載せたあとも追加や調整が必要

例は　テスト設計チュートリアル ちびこん編 '24 資料　29ページより引用

テストの優先順位や厚みを考える

どのようなテスト観点を優先するか、手厚くするか、検討する。

より必要性が高いテストにリソースを割けるよう、複数の角度から考える。

使われ方から考える	• 多くのユーザーが頻繁に使う部分のテストを厚くする • メインターゲットとなるユーザーが気にしそうな部分のテストを優先的に行う
プロダクトの特徴から考える	• 売り文句やプレスリリースに関するテストを優先する • 関係するシステムや人が多く問題が起こった場合に対応が難しい部分のテストを厚くする
リスクから考える	• データ破損など重篤な問題が一定以上の確率で発生する恐れがある機能のテストを厚くする • 詳細は次ページ以降へ

テストの優先順位や厚みを考える

リスクから
考える

- 通常、一つ一つのリスクについて、悪影響を及ぼす事象や状況の顕在化可能性と、顕在化した場合の影響度を検討する
  - 例えば、「三角形判定アプリ」では判定ボタンの連打により故障の恐れがあるとする。その顕在化可能性は、どのくらいだろうか？

10秒、各自で考えてみてください(あてませんのでお気軽に)

テストの優先順位や厚みを考える

リスクから
考える

- うまく使えば有効な方法だが、難しい
  - 顕在化可能性と影響度を決めるのは、難しい
  - 例えば、判定ボタンの連打により故障の恐れがあるとして、その顕在化可能性はどのくらいだろうか？　小さいこどもの利用を想定するかといった前提の置き方や、経験が異なる人同士で、判断は分かれそうである
  - 判断が難しいから念のためテストする、という選択肢もある
    しかし、高額な環境や煩雑な準備が必要なテストならば、どうだろうか？
- 少なくとも、ステークホルダーが納得できるよう工夫する
  - 顕在化可能性と影響度の検討にステークホルダーにも参加してもらう、検討の前提や判断基準をできるだけ明確にする等の工夫をして、テストのステークホルダーが納得できるように決める

制約への対応を考える

一般的に、テストには工数面や環境面等で制約がある。

テストの実現性を高めるために、前述したマネジメント的テスト要求をもとにして制約を明らかにし対処方針を決める。

あるテスト環境は
10日間しか使えない

- 利用可能日に合わせてテストの実行順を決定する
- 一部のテスト実行を自動化し夜間の実行をしやすくする

今後予定されている
仕様の改善や
追加開発に備えたい

- 仕様とテストのトレーサビリティを確保し、仕様変更によりテストのどこに影響があるか分かりやすくする
- リグレッションテストを用意してプログラムの修正を行いやすくする

テストアーキテクチャ設計

全体像を設計する

本資料での「テストアーキテクチャ」は、テスト観点の全体像を、それらの関係性等の特定の関心事に基づいてモデル化したもの。
テスト観点やその組み合わせ、テストタイプ等が含まれうる。

テストアーキテクチャ設計により、全体像を分かりやすく整理し、バランスの検討やテストケース作成をやりやすくする。

そのため、例えば以下を行う。

- テストケースの構造を可視化する。まとめて扱いたいテスト観点を考える。
- テスト全体の構造を可視化する。
  例えば、「システムテスト」にどのようなテストを含めるか、
  「機能テスト」や「性能テスト」等に前後関係はあるか、等を考える。

テストアーキテクチャの定義は　https://www.aster.or.jp/testcontest/u30.html#Submission3　を参考にした

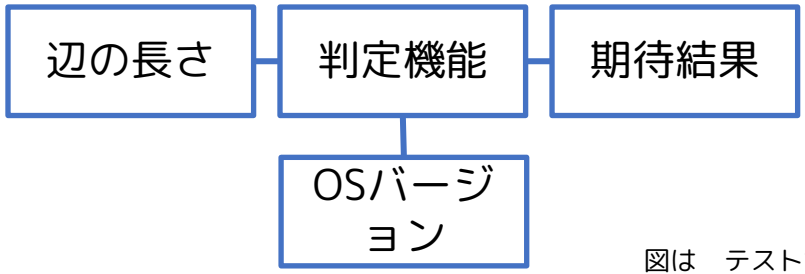
テストケースの構造を可視化する

あるテストケースで、複数のテスト観点をまとめて扱いたいことがある。

「三角形判定アプリ」の例

辺の長さ	OS バージョン	期待結果 (三角形の種類)
(3,3,3)	iOS18	正三角形

まとめて扱いたいテスト観点を明らかにしテストケースの構造を検討する。



様々なテストをする場合、通常、
テストケースの構造が複数できる。

図は　テスト設計チュートリアル ちびこん編 '22 資料　49ページの図を参考に作成

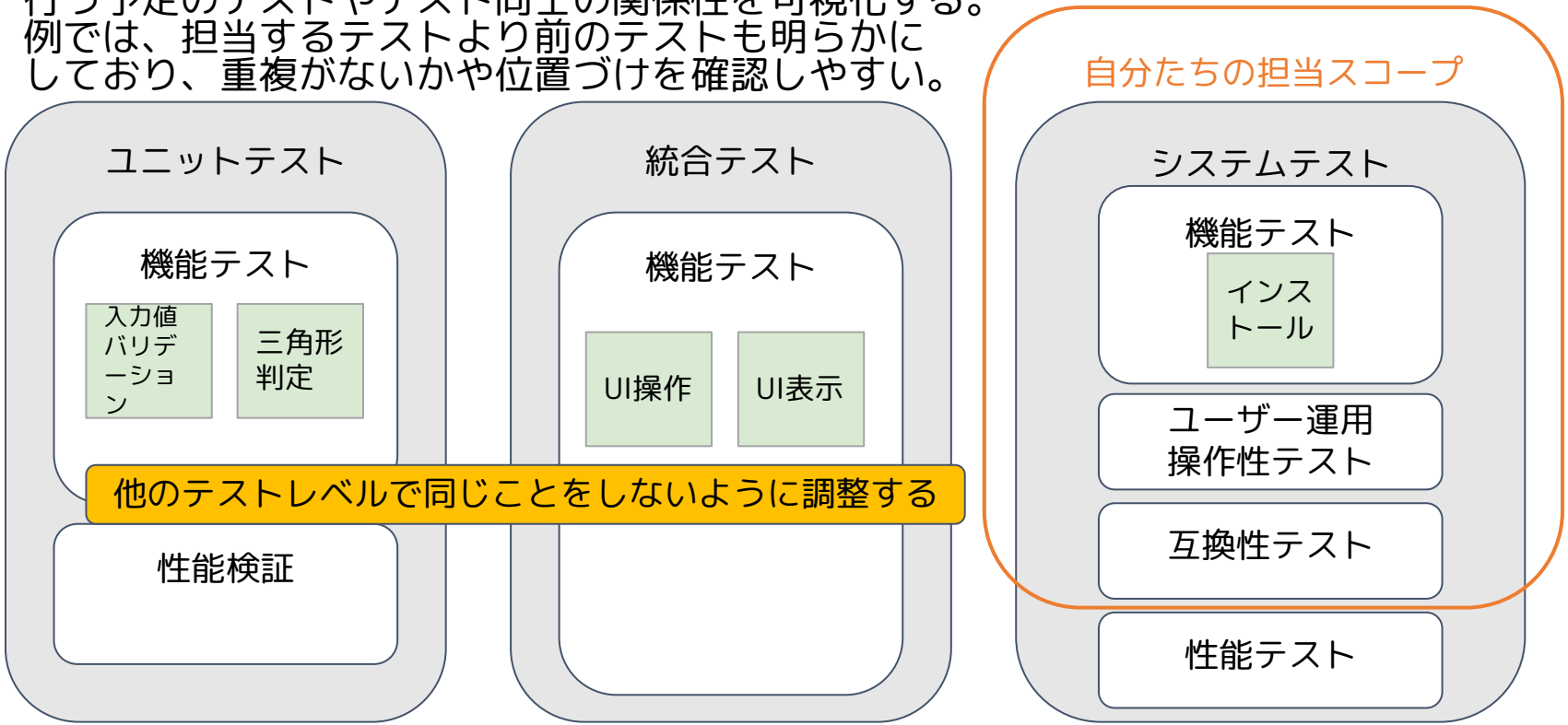
テストケースの構造を可視化すると何が嬉しいか？

テストケースについて、扱うテスト観点をもとに構造を考えることは、意図に沿ったテストケースをつくる助けとなる。

- テストケースに紐づくテスト観点の十分性について早期から議論やレビューがしやすくなる
  - テストケースの構造自体はテストケースで使う値の検討前から考え、議論やレビューをすることができる。
    - まとめて扱いたいテスト観点は不足なく含まれているか？余分なテスト観点が含まれており必要以上に複雑なテストケースになっていないか？
- テストケースの意図を分かりやすく示せる
  - 「大項目」「小項目」といった見出しを使えばテストケースを表現することはできるものの、各テストケースの意図が理解しづらくなる。

テスト全体の構造を可視化する

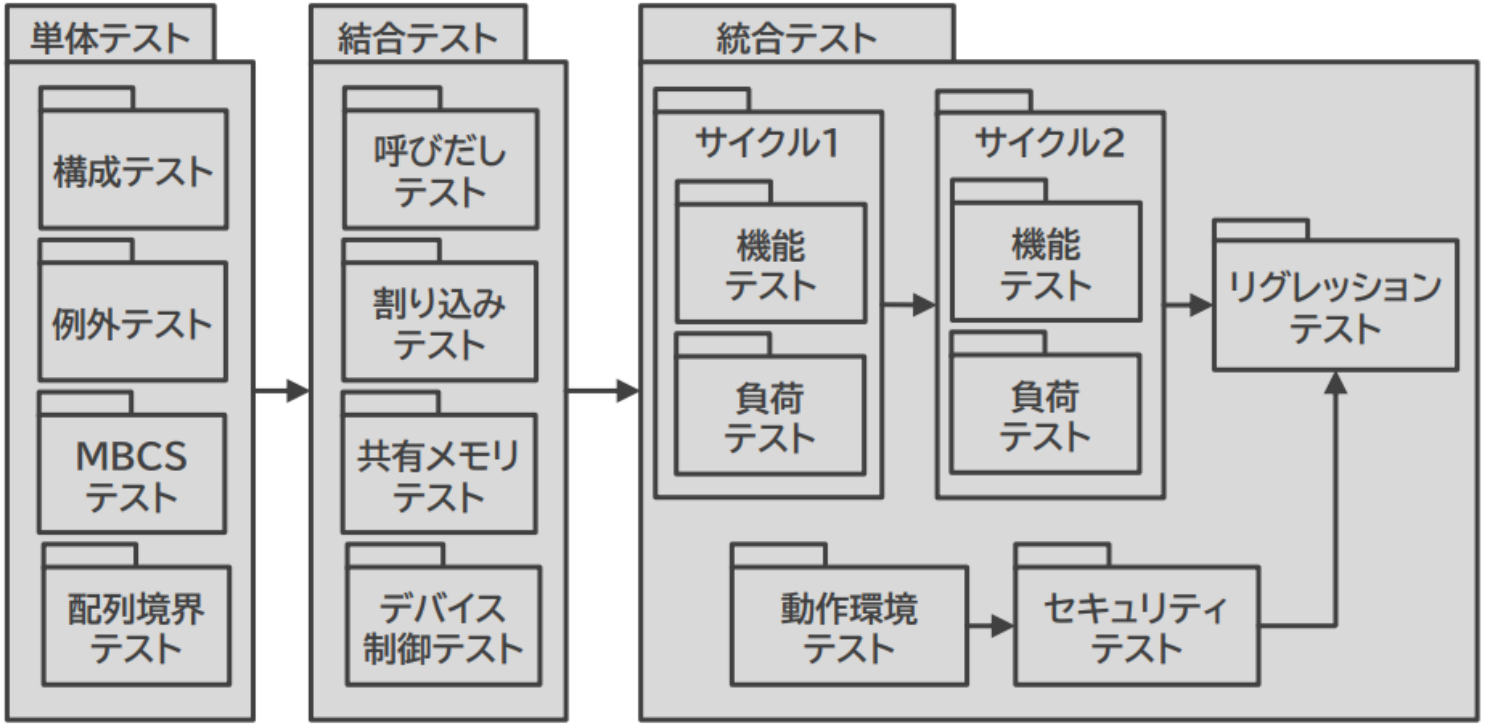
行う予定のテストやテスト同士の関係性を可視化する。
例では、担当するテストより前のテストも明らかにしており、重複がないかや位置づけを確認しやすい。



例は　テスト設計チュートリアル ちびこん編 '24 資料　31ページから引用

テスト全体の順序を示した例

サイクルや前後関係を表現すると、意図の共有や作業分担のときに役立つ。



図は　テスト設計チュートリアル ちびこん編 '21 資料　44ページから引用

テスト全体の構造を可視化すると何が嬉しいか？

コミュニケーションや学習の場面で良いことがある。

- テスト観点同士やテストタイプ同士等の関係性を俯瞰でき、見通しが良くなり、バランスや後続の活動を検討しやすくなる
  - 大きな抜け漏れや重複、偏りに気づきやすくなる
  - 前後関係やサイクルの有無を示すと、テストの詳細設計、実装や実行のタイミングを考える上でのインプットにできる
- 情報量がテストケースに比べて少なく、短時間で全体像を掴む際に便利である
  - 自分たちの責務範囲やその周辺のテスト、相互の関係性を短時間で共有するのに役立つ
  - レビュアーに説明する際や、前提知識が少ない新メンバーが学習する際等、最初から細かなテストケース1件1件を見せるよりも、どのようなテストが行われるか伝えやすくなる

テスト詳細設計

テストで使う具体的な値を考えテストケースをつくる

テスト観点それぞれについて、とりうる値を特定しテストケースをつくる。

テスト技法を使うと体系的に考えやすく、網羅度もあわせて検討しやすい。

「三角形判定アプリ」では、
辺の長さの入力項目は有効な値は
1～999の整数、「境界値分析」を
適用してテストしたい。

→0,1,999,1000を使おう。

→更にピンポイントで確認したい値を
検討、使う値を特定する。

	値	補足
入力値	-1	負の値、エラーになる
	0	
	1	
	999	
	1000	
	数字以外	エラーになる

値を特定したら、テストアーキテクチャ設計で検討したテストケースの
構造を使ってテストケースを作成する。

値の決定方法

値は、テストやテストケースの目的と整合するように決定する。

	値
入力値	-1
	0
	1
	999
	1000
	A(半角英字)
	B(全角英字)
	2(全角数字)
	!(半角記号)
	…

過去に文字種が関係する重篤な故障があり、再発しないことを確認したいならば、文字種を細分化してテストする必要があるかもしれない

	値
入力値	10(有効な値)
	2000(無効な値)
	(未入力)

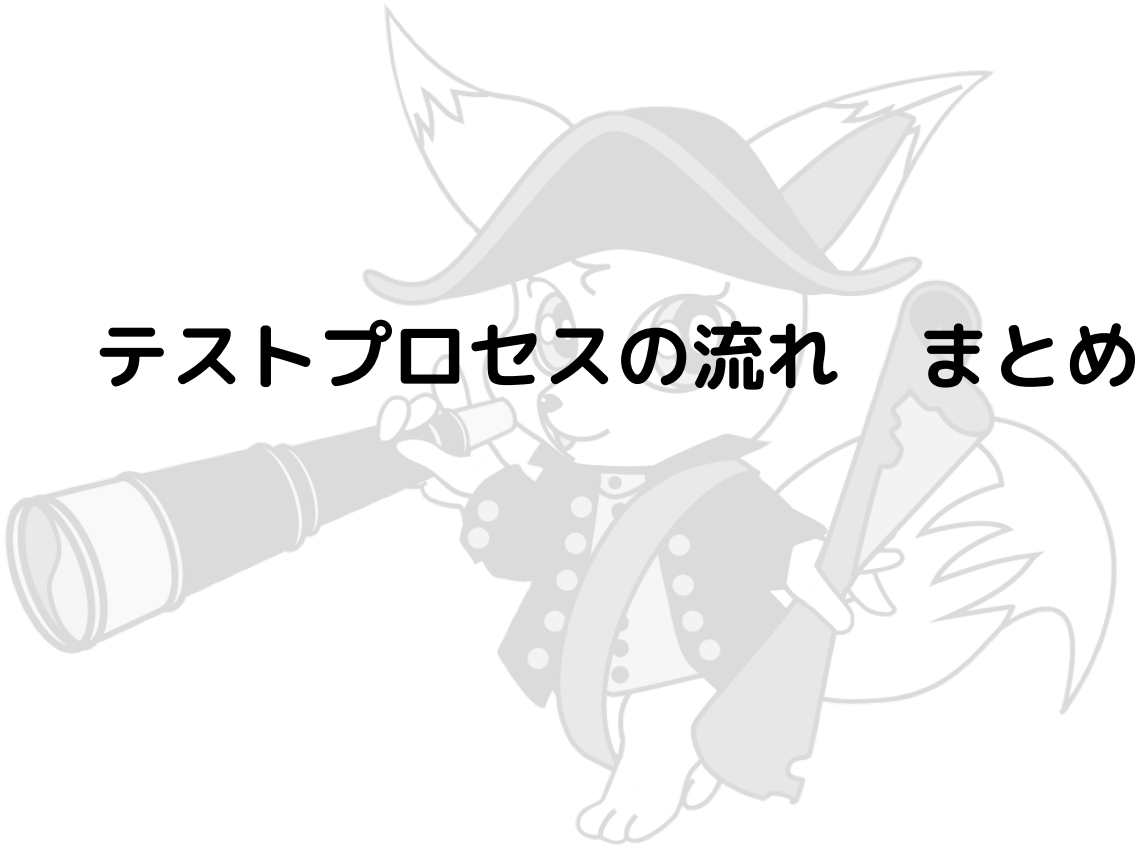
テスト開始判定を行うためのスモークテスト用の値なら、割り切って実施することを狙っても良いかもしれない

テスト実装

テストが実行できるように手順をつくる

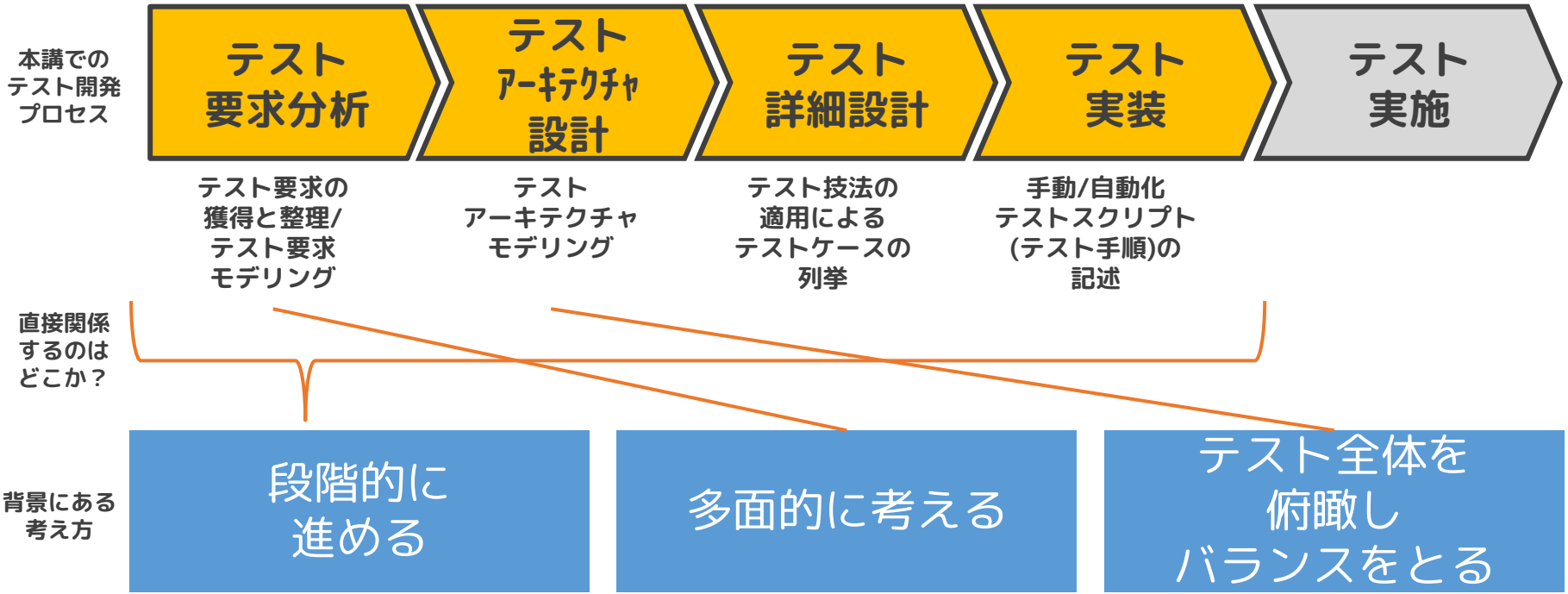
テスト実行や、ドキュメントの保守の効率化を考えられるとよい。
手動でのテスト実装を想定した場合、次のようなことを考える。

準備やテスト実行を同時にできないか？	• 準備に時間がかかる複数のテストケースについて、準備や実行をまとめて行うことで手間を減らせるのでは • 1回の作業で複数のテストケースを確認できないか
続けて実行した方が効率的では？	• 入力値に応じて正しく三角形が判定されるかテストする前にアプリのインストールのテストをすれば、準備の手間が省けそう
テスト実行者から見て分かりやすい？	• テスト対象のプロダクトの知識が浅いメンバーが実行するならば、複雑な手順は具体的に書いた方がよさそう
仕様変更時の更新の手間を減らすには？	• テスト手順は記述量が比較的多い。複数のテスト手順に共通する内容をまとめて書く等、今後の保守活動に配慮する



テストプロセスの流れ　まとめ

ここまでの説明と背景にある3つの考え方の関係性



テスト開発プロセスの図は　テスト設計チュートリアル ちびこん編 '21 資料　67ページより引用

3つの考え方を大事にしたい理由

段階的に考える

特に複雑なテストでは
大きな抜けを防ぐために
徐々に具体化や詳細化を
行いたい。
また、それぞれ活動内容
が異なるので、分けた方
が難易度が下がる。

多面的に考える

状況に適したテストを
検討するためには不可欠。
また、テストを通じ情報
を整理することで欠陥の
早期発見のチャンスが
増えるという利点もある。

テスト全体を俯瞰し バランスをとる

大きな抜けや優先順の
間違いを防ぐため。
バランスをとるためには
全体やテスト間の関係性
の可視化が必要で、この
可視化により細部を理解
しやすくなる利点もある。

テスト開発プロセスの概観が分かったところで
次のページから気を付けたいポイントを見てみましょう

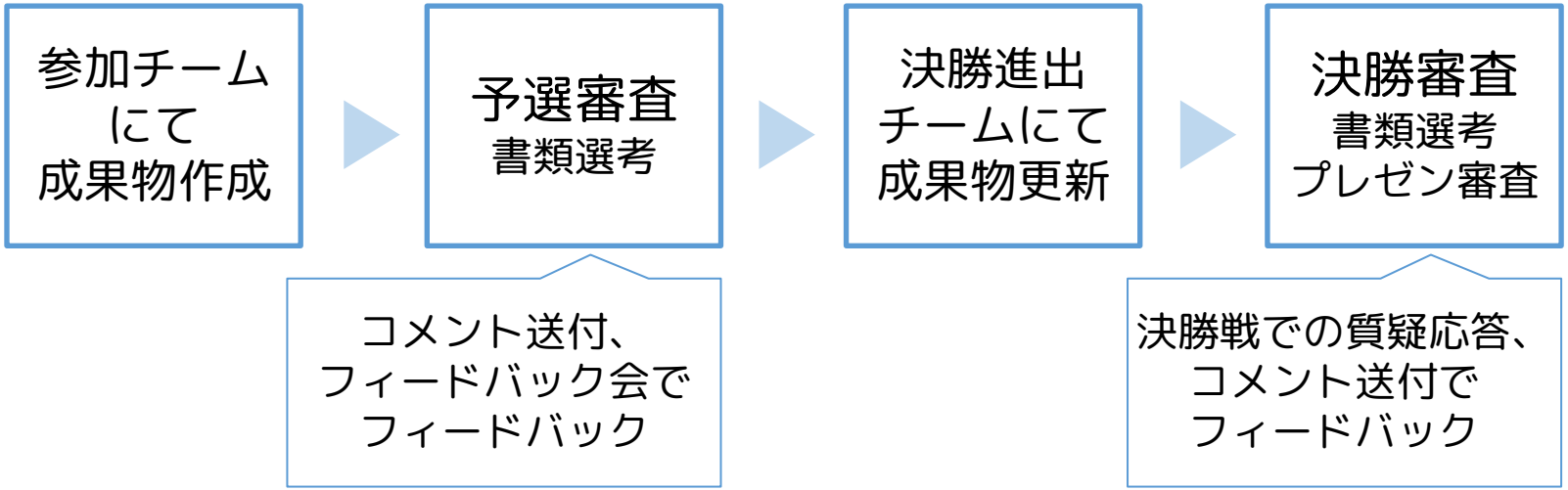
[テーマ2]

審査コメントから見えてきた
気を付けたいポイントと対処方法

ご紹介するポイントの出所の説明

テスコンU-30クラスでは、審査員が参加チームの成果物
（テスト要求定義～テスト実装の成果物）を審査した後で、
フィードバックを行っている。

テスト設計コンテスト'24 U-30クラスの審査の流れ



ご紹介するポイントの出所の説明

2024年度のテスコンU-30クラスの予選で審査員からお送りした
各チームへのフィードバックコメントの分析をもとに、
気を付けたいポイントをご紹介します。

テスト設計コンテスト'24 U-30クラスの審査の流れ

普段のテスト活動でレビューを受けたり行ったりする際、あるいは、
テストケースを作るときのヒントとしてお聞きいただければ嬉しいです。

気を付けたいポイント

取り上げるのはこの4つ。

- 根拠や意図、結果や実現方法の分かりやすさ
- 多面性
- 状況に合ったテスト活動
- ビジネス文書としての可読性

- 説明
- コメント例
- 対策

補足

- 審査コメントをそのまま引用せず、要約や改変をして載せている場合があります。
  太字等、文字の装飾は本資料作成者によるものです。
- 特定のチームや審査員について取り上げる意図はありません。
  コメントを受けたチームや書いた審査員が推測できても、胸の中にしまって
  おいていただけますようお願いいたします。

根拠や意図、結果や実現方法の分かりやすさ

説明	成果物の背景にある根拠や意図が読み取れない。 または、ある成果物をもとに活動した結果や実現方法の詳細が分からない。

何が困るか？

- 根拠や意図が分からないと、テストアーキテクチャなどの成果物を読む側は妥当性が判断できない。
- 結果や実現方法が分からないと、意図通りに活動が行われたか判断できない。結果として、やはり妥当性が判断できない。
- 意図や根拠が不明だったり、結果や実現方法が曖昧ということは、前提や目的と食い違った活動になっている恐れがある。

（次ページに続く）

根拠や意図、結果や実現方法の分かりやすさ	
コメント例	1. 根拠や意図が読み取れない • **何を狙いとして(Why)、どのようにしてテストアーキテクチャの図を作ったのか(How)**が示されていません。そのため、テストアーキテクチャの意図が分かりませんでした。 • 非機能テストで、特定の特性について責務範囲外としていますが、**責務範囲外とした理由**が示せると、納得感が高まると思います。 • テスト要求分析でユーザーの行動などを分析されていますが、それらのアウトプットが唐突に提示されている印象を持ちました。**何を考えたのか、どんな点を意識して出力したのか**を分かりやすくしていただけると、理解しやすくなると思います。 （次ページに続く）

根拠や意図、結果や実現方法の分かりやすさ	
コメント例	2. 結果や実現方法の詳細が分からない • 使用したテスト技法について、以下を例としてテスト技法を適用した具体的な内容を成果物に掲載することを検討してください。 ● 同値分割法であれば、対象とした要素に対して識別した有効と無効の同値パーティションを示す ● 境界値分析であれば、対象とした要素に対してモデル化した数直線と、そこから得られた境界値を示す ● デシジョンテーブルテストであれば、対象とした条件の組み合わせと動作に対するデシジョンテーブルを示す ● 状態遷移テストであれば、状態遷移図および/または状態遷移表と、どのような基準でテストケースを網羅するのかを示す （次ページに続く）

根拠や意図、結果や実現方法の分かりやすさ	
コメント例	3. 成果物間の繋がりが分からない • PFD(Process Flow Diagram)で成果物の全体像を示されています。**この PFDの情報と実態の成果物とに違いがある**と、読み手が迷子になってしまい、自分達の伝えたいことが伝わりにくくなるためもったいないです。
対策	✓ 段階的にテストを考え、インプットの情報や、メモやホワイトボードのコピーなど思考過程がわかる情報をとっておく。 ✓ 途中で全体を見直すタイミングを設け、成果物間の整合性をとる。 ✓ 経験や現状を成果物の読者とどの程度共有できているかに依って、説明の細かさを調整する必要があることに注意。

気を付けたいポイント

多面性	
説明	インプットが仕様書のみ等、テストを考えるための情報収集が十分行われたように見えない。 あるいは、様々な情報が収集されているが、情報を活用する余地が残っている。
コメント例	1. 情報の収集、分析 • テスト観点の内容としては、**仕様書あるいは一般的な機能の範囲内の記述にとどまっています。** • **起こりそうなバグやユースケースを想定**し検討された点は良かったです。 • リグレッションテストについても検討しており、**今後のリリースに向けた対応が考慮できています。** （次ページに続く）

気を付けたいポイント

多面性	
コメント例	2. 集めた情報の活用 • ユーザビリティテストは他のテストと変わらないように見えます。**ユーザビリティテストだからこそ検出できる不具合やテスト対象の改善点がある**かと思います。そのためのテスト内容と、テストのやり方が考慮されると更に良くなると思います。 • ペルソナを設定するとしたら、**テスト対象である割り勘支援アプリの価値をより多く実感するペルソナがよい**と思います。
対策	✓ 多様な情報を集める。集めるだけではなく、抽象化や具体化をするなどして発想を広げる。 ✓ しっかり情報収集や検討をしたと思っても、目的と整合しない結果になったら何かがずれているか目的の見直しが必要。目的を踏まえて十分多面的に検討できているか、集めた情報を活かしきれているか、確認しながら進める。

状況に合ったテスト活動	
説明	テスト活動の制約が考慮されていない、あるいは、費用対効果の面で再考の余地がある。そのため、活動の実現性や効率が十分高くない恐れがある。
コメント例	1. 体制、工数 • 記載されている内容だけで探索的テストを行うとしたら、**エキスパートレベルのエンジニアが必要かもしれません**。また、視認性や操作性についても、着目点を決めておかないと、個人任せになりすぎてしまい、テスト設計時に狙った意図と異なるテストをしてしまうかもしれません。 • **テスト実行工数が大きく**、実行完了までの時間が掛かってしまうのではないかという懸念があります。 2. 今後の開発やテストを想定した保守性の考慮 • 文書全般において、**保守性を考慮すると各文書には更新履歴があると良い**でしょう。 （次ページに続く）

状況に合ったテスト活動	
対策	✓ マネジメント的テスト要求も意識的に収集する。 ✓ チームメンバーが持つスキルや利用可能なリソースを把握し、事前の教育や機材の調達等、必要なスキルやリソースとの差分を埋める方法を検討する ✓ 費用対効果を考える。例えば、1回だけ実施する予定の社内研修の教材用アプリのテスト実行に数カ月かけるのは妥当だろうか？ ✓ テスト対象の過去や未来の情報にも着目する ✓ 状況が変化しやすい場合は、反復的に制約の確認と対処方法の検討を行えるようにスケジューリングする

ビジネス文書としての可読性	
説明	読み方や文書の構造が分かりづらい、文字の大きさ等が極端で見辛い。 結果、誤解を生んだり、読者に時間をかけさせたりする恐れがある。 補足として、テスコンU-30クラスでは提出物のページ数に制限が設けられていることも影響している可能性はある。制限がなくても、様々な事情から読者に不親切な文書ができる場合が考えられるため、ここで取り上げる。
コメント例	1. 読み方の明記 • **IDの振り方のルール**を明示している点はよいと思います。 • **凡例**がなかったり、説明が少なかったりすることがあり、わかりにくさや保守性の低下に繋がっています。 2. 文字の大きさや図表の大きさ、見せ方に関する配慮 • **ひとつの図が複数ページに分割されている、読むためにかなり拡大する必要がある、**など、読者に負荷をかけてしまう状態になっています。 • 一部の図は、**全体を俯瞰して確認しづらい**です。 （次ページに続く）

気を付けたいポイント

ビジネス文書としての可読性	
コメント例	3. 文書構造の複雑さ • テストアーキテクチャの図は理解できるのですが、共通要素が多く、もっと**シンプルに表現できるのでは**と思います。 • 文書量が多いこともあり、**文書間の関連をより明確にするか、より単純化して作成されると**、文書を読む側の印象は更に良くなると思います。
対策	✓ 書かないと伝わらないことは書く必要があるが、文章量が多すぎても理解しづらい。理解しやすさや、可読性を高めるよう文書構成の工夫を検討する。 ✓ 図表が大きすぎる場合は、抽象度が粗い図表と細かい図表を作成したり、スコープを分けたりして、複数に分割する。保守性とのトレードオフになることがあるので注意は必要。 ✓ 作成者が作成直後に見ても、分かりづらさに気づきにくい。他の人にレビューを依頼するか、自分で確認する場合は時間を置いて見直す。

気を付けたいポイント

テストケース作成やテスト成果物のレビューにて、ご参考になる情報を何かしら掴んでいただけていたら、幸いです。

- 根拠や意図、結果や実現方法の分かりやすさ
- 多面性
- 状況に合ったテスト活動
- ビジネス文書としての可読性

さいごに

皆さんが、以下のゴールを達成できていると嬉しいです。

実装までのテスト開発プロセスのイメージを掴んでいただく

実装までのテスト開発プロセスの中で特に気を付けたいことを知っていただく

本日の話は絶対的な解ではありません。

考えるヒントとしていただき、みなさんが置かれている状況により合ったテストに繋げていただければ幸いです。

既にうまくできているという方、実践や試行の場を探している方は、
もしよければテスコンへのご応募を検討していただければ嬉しいです。

おまけ　もう少し学びたいなと思ってくださった方へ

2025/06/04にテスコン編も開催予定です。

- テスト設計チュートリアル テスコン編 '25　https://aster.connpass.com/event/355108/

過去のちびこん編の資料や動画は、公開されています。
チュートリアルは、担当の審査員ごとに伝え方に特色があります。
過去分も確認してみていただけると、また違った発見があるかもしれません。

- 2024年度
  - https://www.aster.or.jp/testcontest/doc/2024_chibicon_v1.0.0.pdf
- 2023年度
  - https://speakerdeck.com/goyoki/test-design-tutorial
- 2021年度
  - https://www.aster.or.jp/business/contest/doc/2021_chibicon_V1.0.0.pdf
- テスト設計コンテストチャンネル
  - https://www.youtube.com/@aster-test-design-competition

参考文献と引用文献

参考文献と引用文献

いずれも、本書作成時点の情報である

- テスト設計チュートリアル ちびこん編 ‘24 資料
  - https://www.aster.or.jp/testcontest/doc/2024_chibicon_v1.0.0.pdf
- テスト設計チュートリアル ちびこん編 ‘23 資料
  - https://speakerdeck.com/goyoki/test-design-tutorial
- テスト設計チュートリアル ちびこん編 '21 資料
  - http://www.aster.or.jp/business/contest/doc/2021_chibicon_V1.0.0.pdf
- テスト設計チュートリアル テスコン編 ‘22 資料
  - https://www.aster.or.jp/testcontest/doc/2022/2022_tescon_V1.0.0.pdf
- JaSST’22 Hokuriku　目指せテスト設計リーダー!! テスト設計チュートリアル　資料
  - https://www.jasst.jp/symposium/jasst22hokuriku/pdf/B1.pdf
- JaSST’13 Kyushu　ちょっと明日のテストの話をしよう　資料
  - https://www.jasst.jp/symposium/jasst13kyushu/pdf/S5.pdf
- テスト設計コンテスト'21　-　テスト要求分析チュートリアル　資料
  - https://www.aster.or.jp/business/contest/doc/2020_TRA_1-3_V1.0.0.pdf

参考文献と引用文献

いずれも、本書作成時点の情報である

- テスト設計コンテスト
  - https://www.aster.or.jp/testcontest/
- テスト設計コンテスト U-30クラス
  - https://www.aster.or.jp/testcontest/u30.html
- ASTER U-30テスト設計コンテストストプロジェクト要求補足書2024 Ver. 1.0
  - https://www.aster.or.jp/testcontest/doc/aster_u30_tdc_supplement_2024.pdf
- テスト設計コンテスト'24 U-30クラス 予選 審査コメント
- J.マイヤーズ/T.バジェット/M.トーマス/C.サンドラー(原著), 長尾真(監訳),松尾正信 (翻訳). ソフトウェア・テストの技法 第2版. 近代科学社, 2006年.
- ISTQB テスト技術者資格制度 Foundation Level シラバス Version 2023V4.0.J02
  - https://jstqb.jp/dl/JSTQB-SyllabusFoundation_VersionV40.J02.pdf
- ISTQB テスト技術者資格制度 用語集
  - https://glossary.istqb.org/ja_JP/search
- 27号：欠陥の偏在
  - https://note.com/akiyama924/n/n79b902f69ddf