# テスト設計
# 初心者向けチュートリアル

テスト設計コンテスト実行委員

# 目次

- はじめに
- テストの困っていることを共有しよう
- テストエンジニアに求められていること
- 状態遷移を活用しよう
- 演習1(状態遷移図・表を書いてみよう)
- ～休憩～
- 演習2
- 演習3
- まとめ
- テスト設計コンテストのご紹介

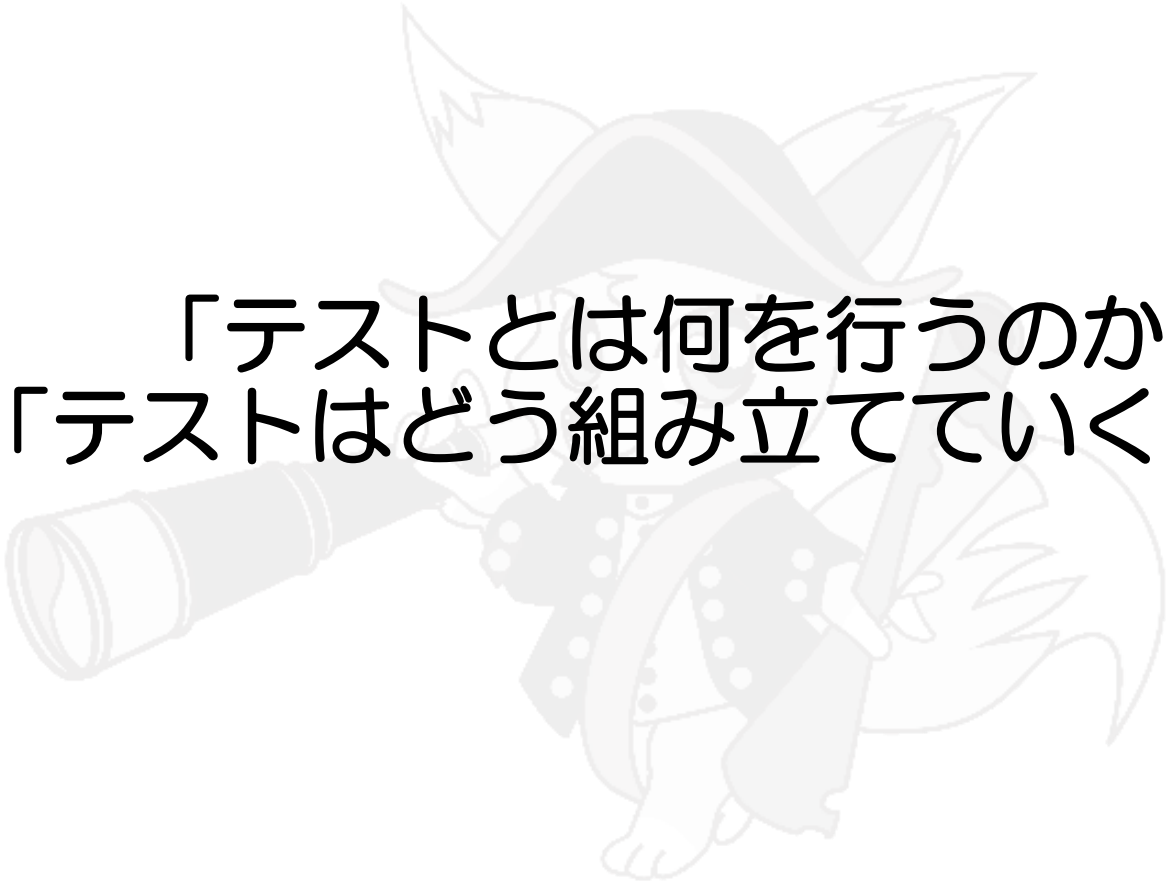
# はじめに

# はじめに

このセッションでは、下記の方を対象に行います。

「経験1年目～5年目相当」のテスト初心者

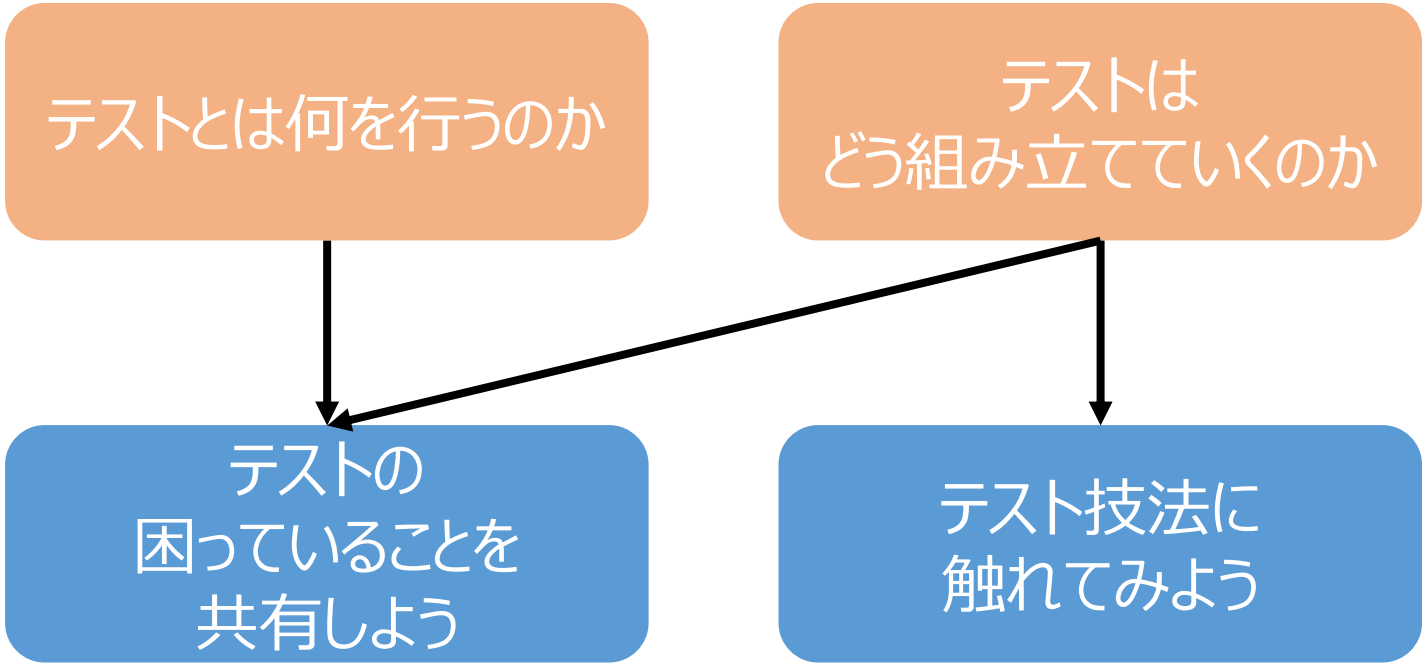
- テスト設計をやり始めたばかりの方
- テスト設計のメリットをいまいち感じていない方
- テスト技法を利用したことがない方



「テストとは何を行うのか」
「テストはどう組み立てていくのか」

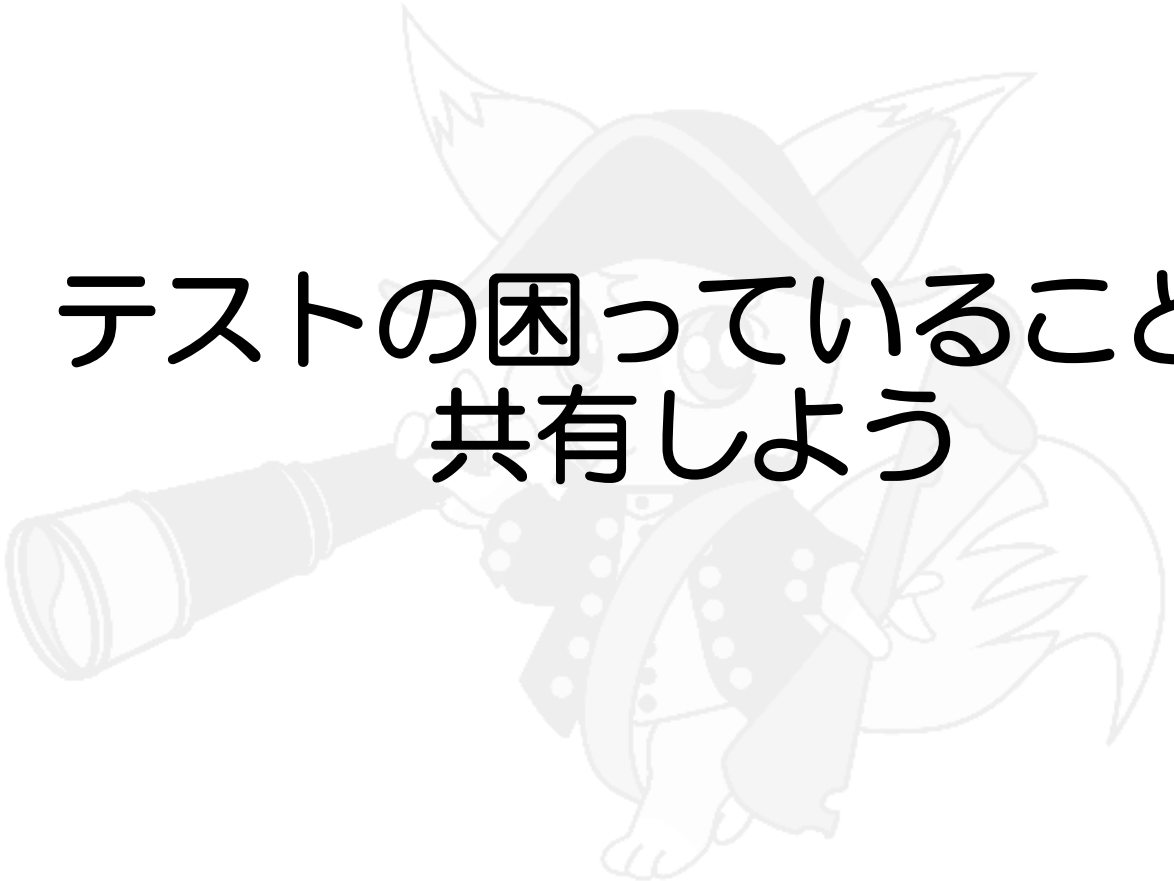
# はじめに

それぞれ下記の章で解説していきます。



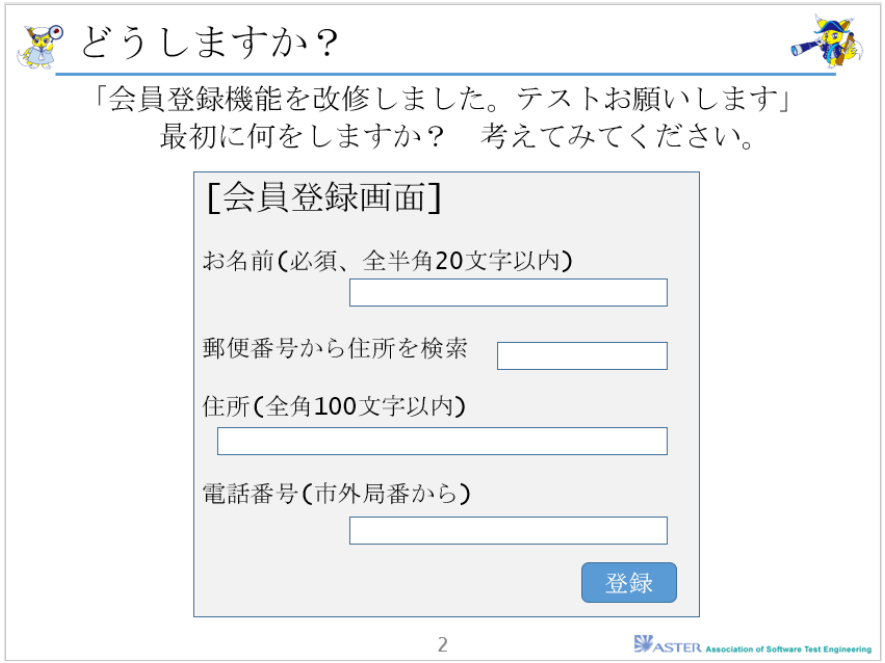

※演習あり

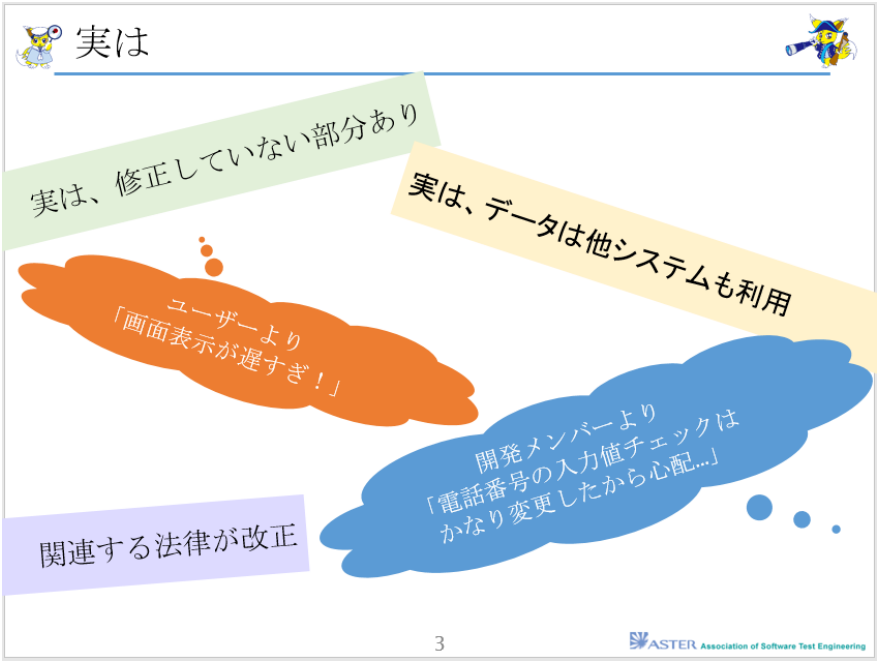
# テストの困っていることを共有しよう



# テストの困っていることを共有しよう

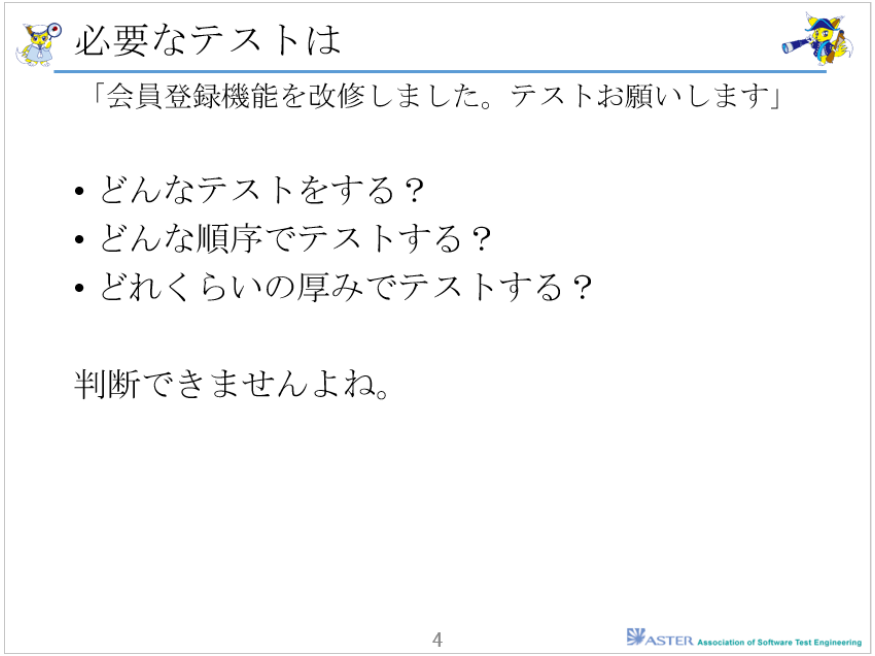
- テストあるある！
  - 「これちょっとテストしておいて」




テスト要求分析チュートリアルより

# テストの困っていることを共有しよう




- その他
  - 組織によってはテストの価値が低く見られている
    - テストは技術力がない人がやればいいよ
  - 前と同じテストすればいいよ
    - 手順がわからない
    - やる意味がわからない
      - 削減することができない
  - バグが出て怒られる

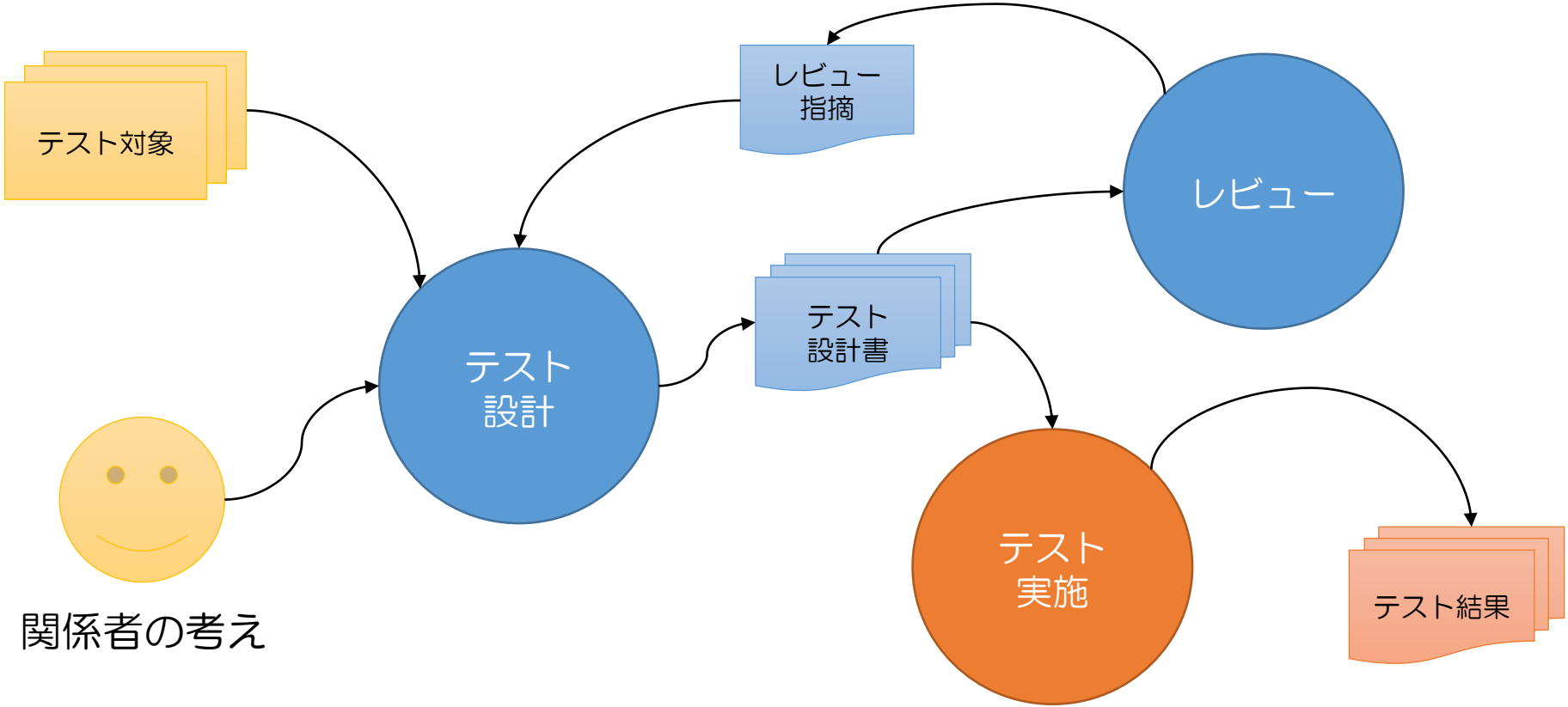
# なんで設計すると嬉しいの？

- テスト設計をするとテストの困ったが解消できる！
  - 効率
    - 同じようなテストケースがある
  - 範囲(網羅)
    - ちゃんとテストできているかわからない
    - どこまでテストすればいいのかわからない
  - ストレス
    - 根拠
      - やっているテストに意味があるかわからない
      - テストケースが減らせない
    - いつまでたってもテストが終わらない/終わりがみえない
- バグが多く見つかる（かもしれない）

# 「テストを設計する」とはどういうこと？

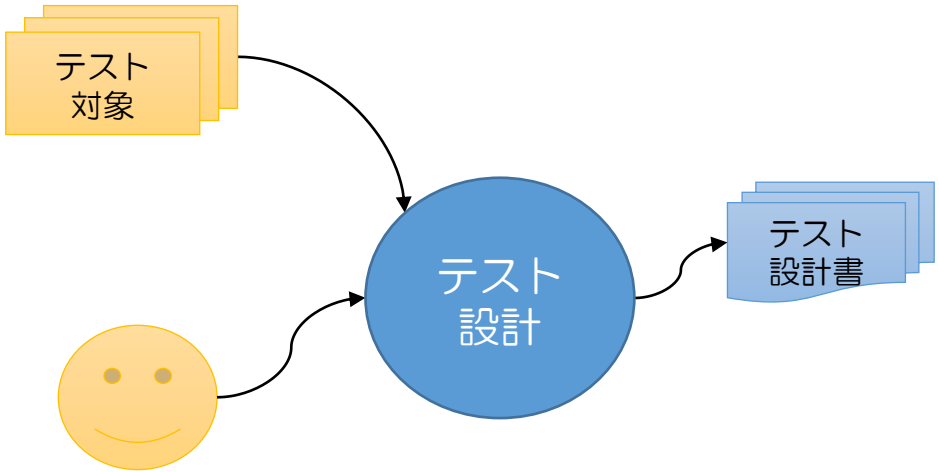
- テストを設計すると良くなること
  - 管理がしやすくなる！
  - テストがよくなる！
    - 無駄がなくなる
    - 気づきが増える

# 「テストを設計する」とはどういうこと？



# 「テストを設計する」とはどういうこと？

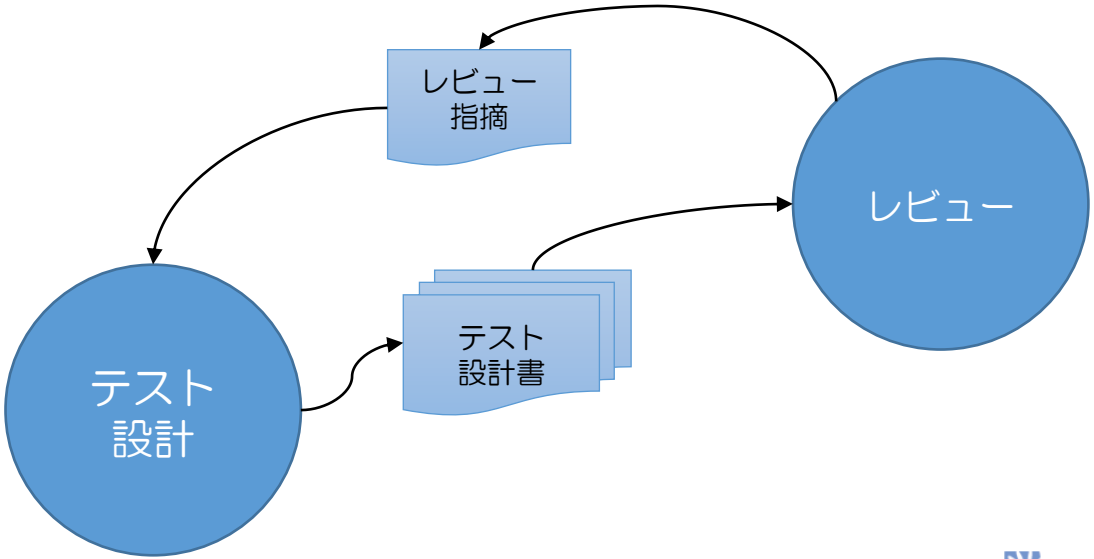
- 効率的にかつ網羅的に欠陥が検出できる準備をすること
  - 別の言い方にすると
    - テスト実施する前にちゃんとテスト対象と向き合って考えましょう！ということ




関係者の考え

# 「テストを設計する」とはどういうこと？

- 階層的な設計書を作ることでレビューができる
  - チームでの共有ができる
    - 考え方
    - 範囲
    - 厚み（狙い所）



# 「テストを設計する」とはどういうこと？

- テストを設計するには多くの視点、多くの考え方が必要となる
  - でも人間ってそんなにたくさんのことを一度に考えられない

- だから工程分割をする！


テスト要求の
獲得と整理/
テスト要求
モデリング

テスト
アーキテクチャ
モデリング

テスト技法の
適用による
テストケースの
列挙

手動/自動化
テストスクリプト
(テスト手順)の
記述

# 工程分割（要求分析、アーキテクチャ設計 等 ）の利点

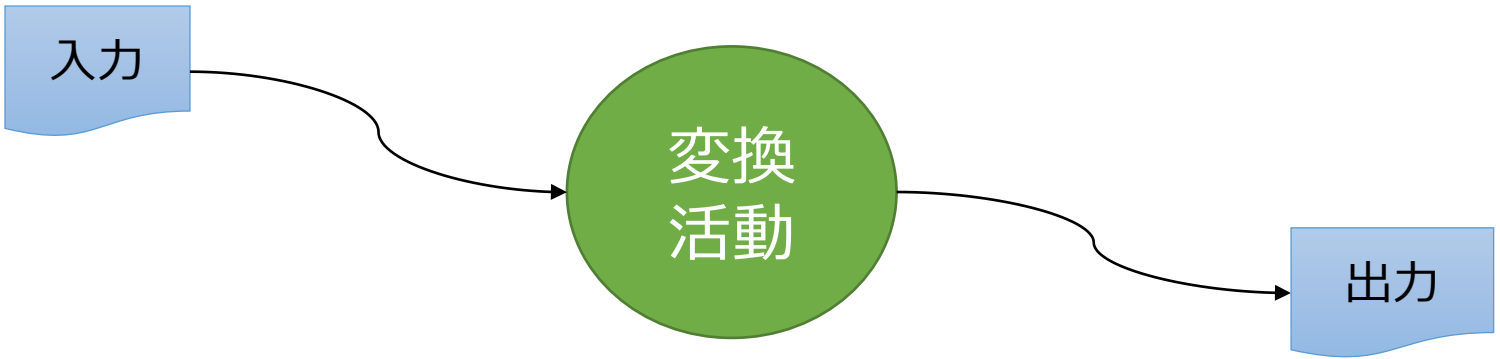
1. 焦点をあわせることができる
   - 頭のリフレッシュ！
2. 終わりが見える
   - 範囲とゴールを決めることでストレス軽減
3. 都度レビューができる(レビューのタイミングが作りやすい)
   - 戻り作業が減らせる

- ワンポイント
  - 別にウォーターフォールを表しているわけではないので繰り返して良い！

# おまけ１『プロセス』

- 工程分割の分割するところを決めることを広い意味で『プロセス』定義という

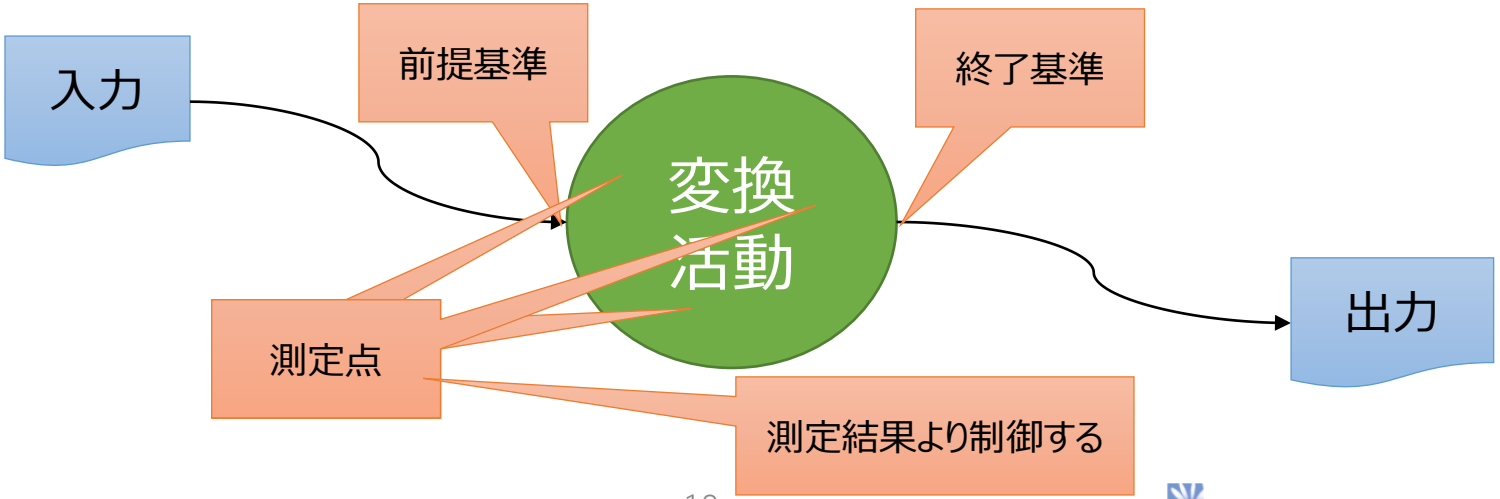
- プロセスってなんぞ？
  - プロセスは　入力を　出力に　変換する活動



# おまけ２『プロセス』

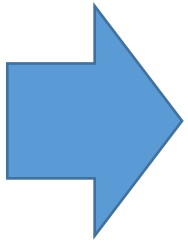
- プロセスがあると嬉しいこと
  - 前提基準および終了基準
    - 何を揃える必要があるか
    - どうなると終わりなのか
  - 測定ができる
  - 制御できる
    - 活動の状況をコントロールできる

- 管理者も嬉しい！



# 要求分析をすると嬉しいこと

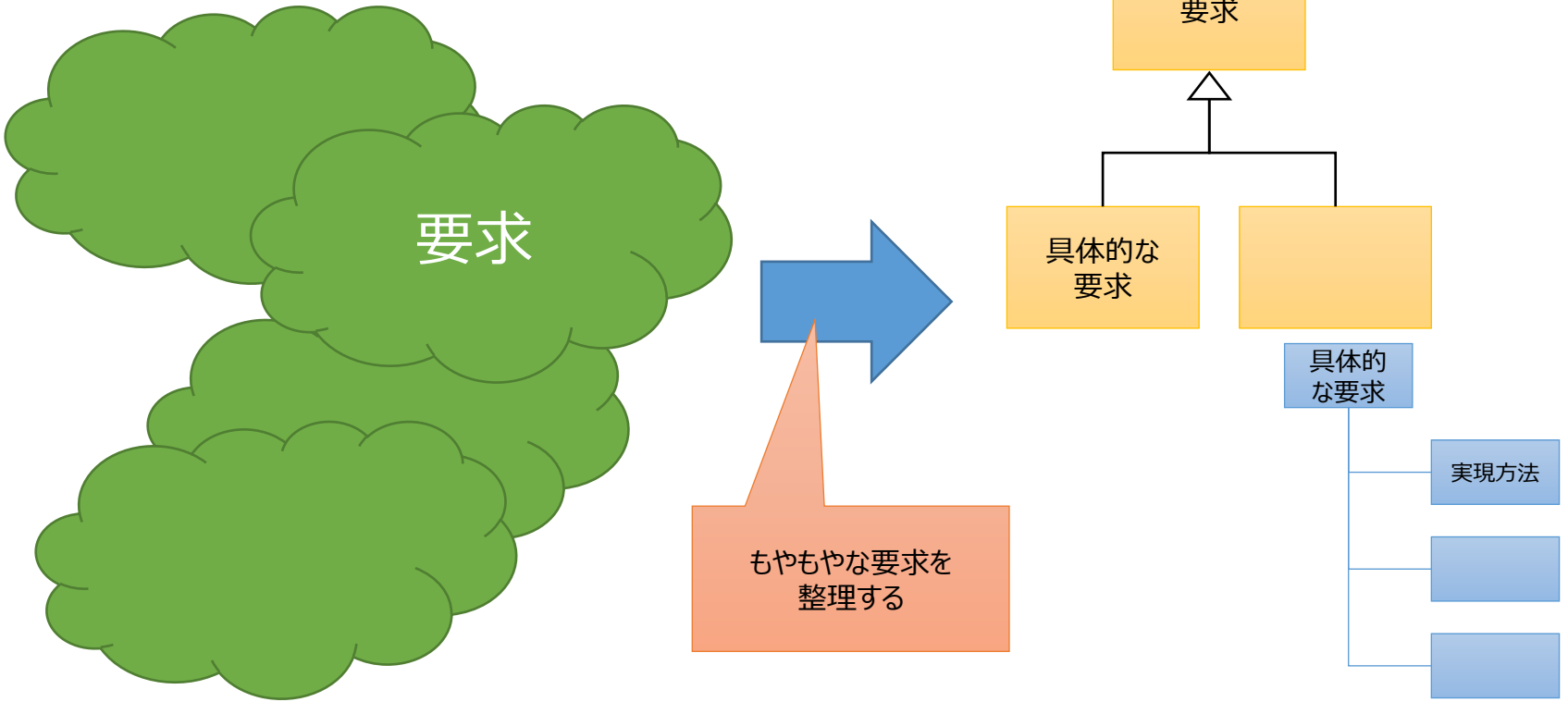
- 要求分析ですること
  1. 製品を理解する
     - 製品の構成
     - 製品の取り巻く環境
  2. 周りの人たちの気持ちを理解する
     - 使う人の気持ち
     - 提供する人の気持ち
     - 管理する人、保守する人の気持ち
  3. テスト(開発)チームを理解する
     - 自分たちの目指している方向
     - 組織の目指している方向
     - 会社の目指している方向



- 何をどこまでなんでテストする必要があるのか整理できる
- テストの範囲と厚みの方向性を決める！

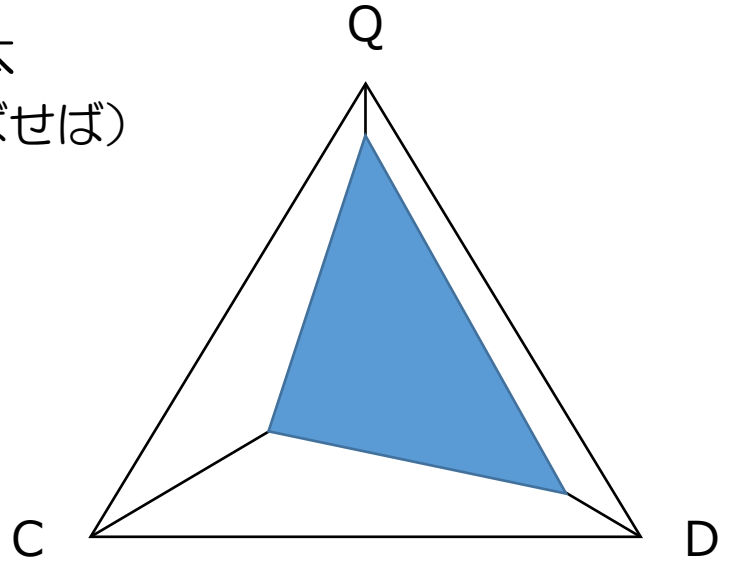
※要求分析チュートリアルがあり、もっと詳しく紹介しています。
【テスト要求分析チュートリアル】
<https://aster.or.jp/business/contest/doc/2020_TRA_1-3_V1.0.0.pdf>

# 要求分析をすると嬉しいこと



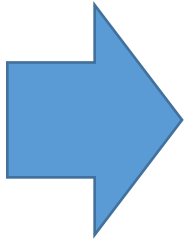
# おまけ２『QCD』

- テストの範囲、厚みの方向性を考えるとき『QCD』という指標を利用することがある
  - Q（品質：Quality）
  - C（価格：Cost）
  - D（納期：Delivery）
- それぞれトレードオフの関係が基本
  - いっぱい時間をかければ（納期を伸ばせば）品質は良くなる



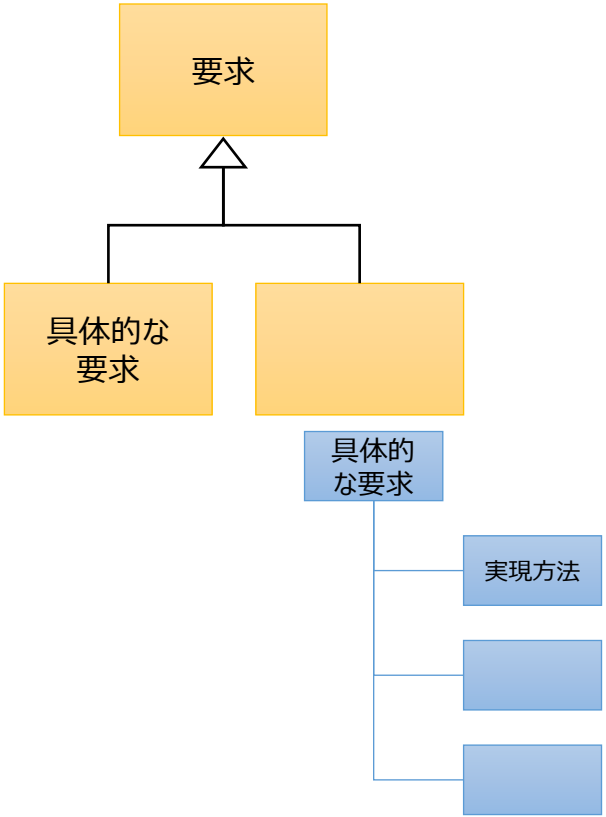
# アーキテクチャ(構造)を作ると嬉しいこと

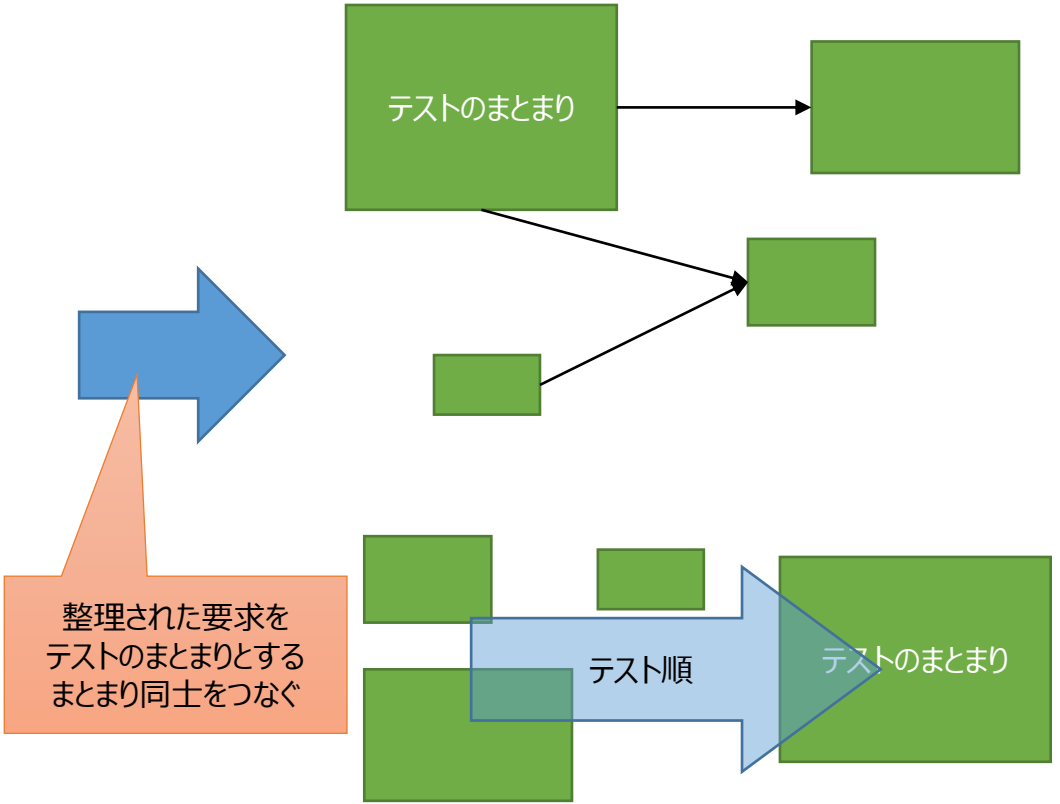
- アーキテクチャ設計ですること
  1. テストのまとまりをつくる
     - 階層
     - 観点
  2. テストの上限を決める
     - テストの範囲
       - どこまでテストするのか
     - テストの厚み
       - どこをたくさんテストするのか
  3. テストのまとまりどおしのつながりを見えるようにする
     - 影響箇所
     - 順序



- 抽象化されているので全体像がわかりやすい
- まとまりごとでの再利用が可能
- テストの実際の範囲と厚みを決める！

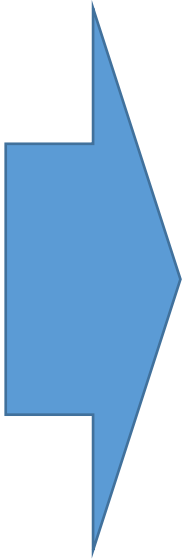
# アーキテクチャ(構造)を作ると嬉しいこと



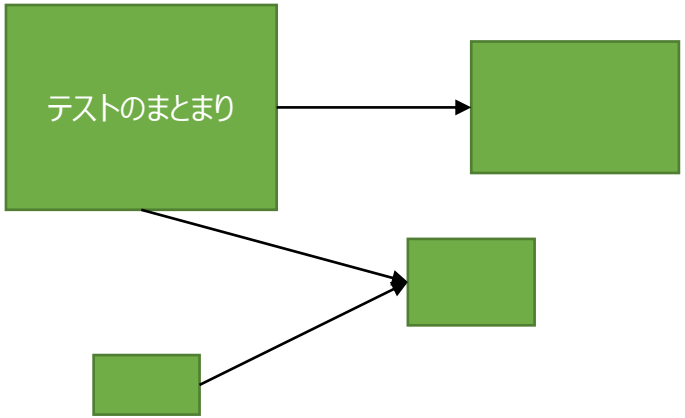
# 詳細設計すると嬉しいこと

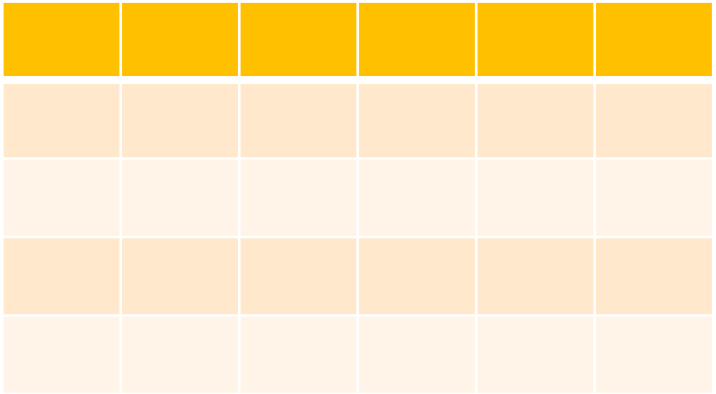
- 詳細設計ですること
  1. 効率を考える
     - 同じテストがないか
     - 無駄なテストはないか
  2. 網羅を考える
     - 全部テストできているか
       - でも効率も考える
  3. 欠陥が見つけられるテストを考える
     - 作るとき間違いやすそうなところ
     - 考慮が不足してそうなところ
  4. 保証するテストを考える
     - ちゃんと動くこと
     - 多く使う機能

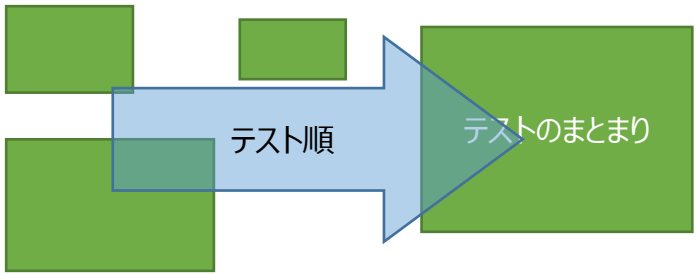


- テスト実施の効率化、ムダ削減
- テストケース抽出の根拠が見える

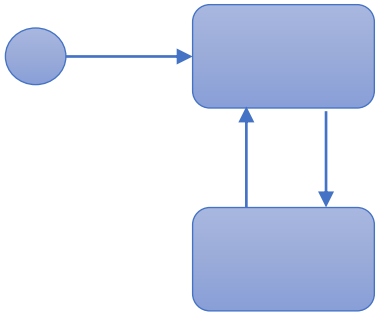
# 詳細設計すると嬉しいこと






テストのまとまり内外で
分析を行い実際の
テストをする値を抽出する



# 状態遷移を活用しよう

# テストエンジニアに求められていること

- V&V（Verification & Validation：検証と妥当性確認）
  - Verification　検証
    - 設計どおりに作られているか
  - Validation　妥当性確認
    - 要求を満たせているか

# 「テストを設計する」とはどういうこと？

- テストを設計すると良くなること
  - 管理がしやすくなる！
  - テストがよくなる！
    - 無駄がなくなる
    - 気づきが増える

# 具体的な困ったを共有

- 要求は曖昧です
  - 仕様書にかかれていないところを背景を考えると当たり前になっている部分
    - エラーが起きたらブザーが鳴る
    - OKを押したら確認してくれる
    - 2回目の際は動かない

- 上記のような点をテストで発見する技

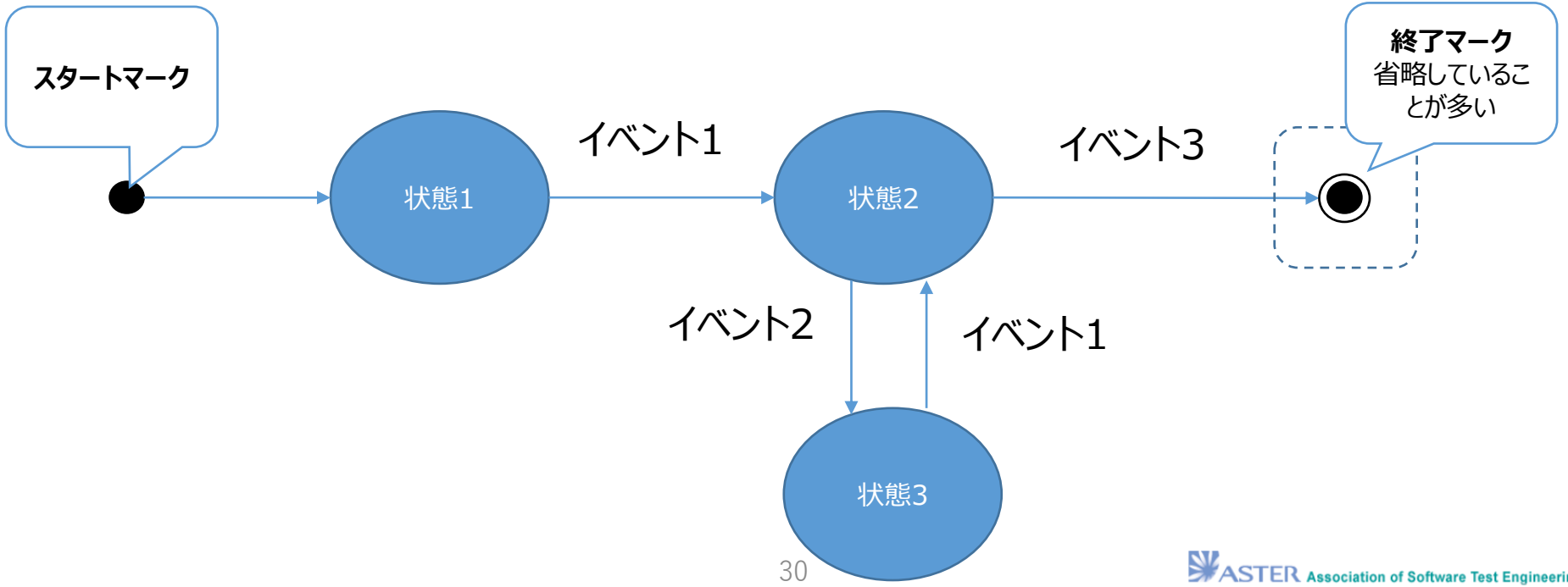
## 状態遷移テスト

# 状態遷移とは

- ある物のライフサイクルを表したもの
- 基本的に「状態」と「イベント」で表される
  - 状態：期間
    - ○○から□□まで
    - 例
      - 小学生：小学校入学から卒業するまで
      - 開店中
        - 開店時間から閉店時間まで
        - 店員が開場し、入れるようになってから
  - イベント：瞬間
    - □□を　過ぎたら・超えたら・されたら
    - 例
      - 3分過ぎたら(経過したら)
      - 80℃を超えたら
      - クリックされたら
- 図と表で表せる

# 状態遷移図とは

- 図はシナリオのシミュレーションやぱっと見の確認に効果的
  - 行き止まりがあったら変なのでは？
  - スタート状態そこでいい？
  - ライフサイクルちゃんと回ってる？



# 状態遷移表とは

- 表は状態とイベントの網羅的確認に効果的
  - 状態ごとにイベントが来たときの動きを確認する

| イベント<br>状態 | イベント1 | イベント2 | イベント3 |
|---|---|---|---|
| 状態1 | 状態2へ | ー | ー |
| 状態2 | ー | 状態3へ | 終了へ |
| 状態3 | 状態2へ | ー | ー |

# おまけ：状態遷移表の流派

流派1：

| イベント ＼ 状態 | 状態1 | 状態2 | 状態3 |
|---|---|---|---|
| イベント1 | 状態2へ | ー | 状態2へ |
| イベント2 | ー | 状態3へ | ー |
| イベント3 | ー | 終了へ | ー |

流派2：

| 現在 ＼ 次 | 状態1 | 状態2 | 状態3 | 終了 |
|---|---|---|---|---|
| 状態1 | ー | イベント1 | ー | ー |
| 状態2 | ー | ー | イベント2 | イベント3 |
| 状態3 | ー | イベント1 | ー | ー |

# おまけ：状態遷移表の表記

- 遷移の表記は“→”で表されることもあります。

| イベント / 状態 | イベント1 | イベント2 | イベント3 |
|---|---|---|---|
| 状態1 | →状態2 | ー | ー |
| 状態2 | ー | →状態3 | →終了 |
| 状態3 | →状態2 | ー | ー |

# 仕様書がすべてではない

要求をすべて文章に表すには限界がある

例えば

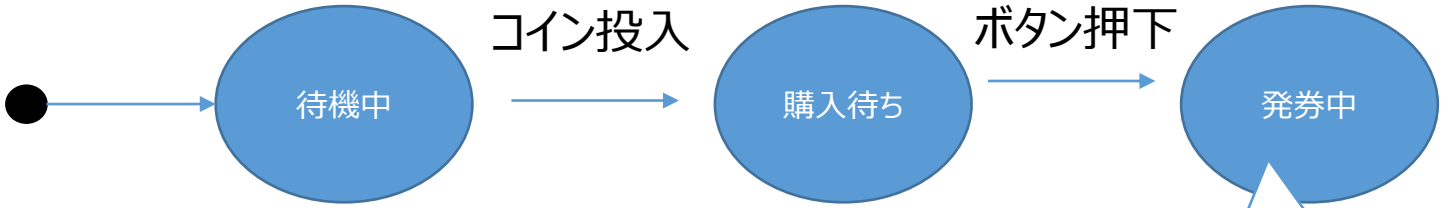
- あるコインを1枚入れると購入できるようになる
- 購入ボタンを押すと1枚発券される
- コインがすでに入っている場合追加投入できない

要求抜けてますか？？？どんなのが抜けているでしょうか？

※機器のエラーはないものとする

# 状態で考えよう

- 図



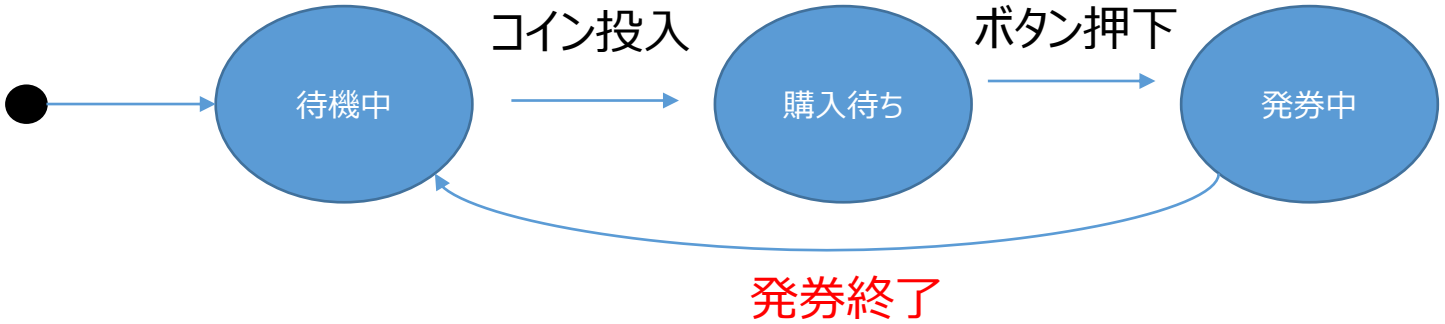

行き止まりを発見
このあとどうなるんだろうか？

- 表

| | コイン投入 | ボタン押下 |
|---|---|---|
| 待機中 | 受け付けて購入待ちへ | どうなるんだろう？ |
| 購入待ち | どうなるんだろう？ | 発券を開始し、発券中へ |
| 発券中 | どうなるんだろう？ | どうなるんだろう？ |

# 状態で考えよう

- 図




- 表

| | コイン投入 | ボタン押下 | 発券終了 |
|---|---|---|---|
| 待機中 | 受け付けて購入待ちへ | 何もしない | N/A(ありえない) |
| 購入待ち | 受け付けない | 発券を開始し、発券中へ | N/A(ありえない) |
| 発券中 | 受け付けない | 何もしない | 待機中へ |

# 状態遷移図・表を利用すると

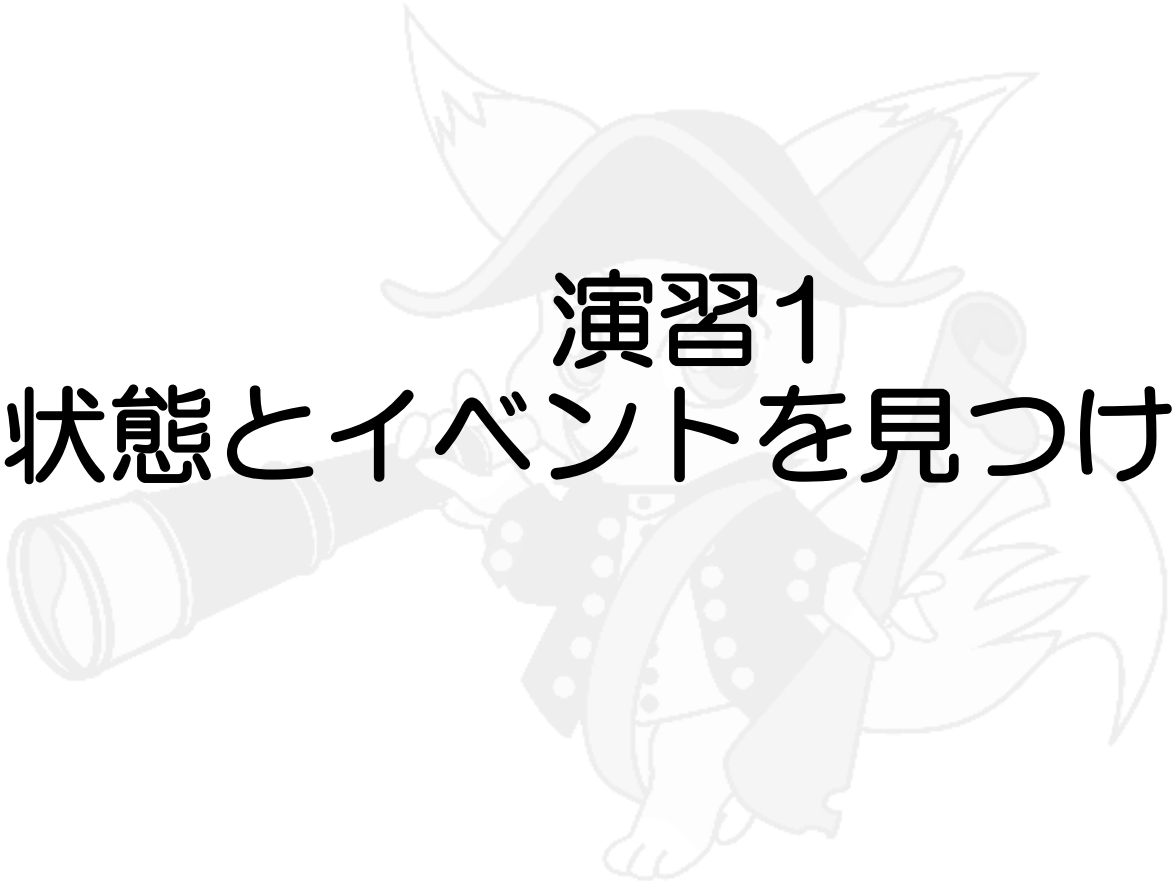
- レビューがし易い
  - 図：システムの動きをシミュレートできる
  - 表：網羅的に確認しやすい

- 仕様の抜け・行間が発見できる
  - 図：そもそもの状態抜けやイベント抜けが発見しやすい
  - 表：状態とイベントを組み合わせた動作の抜けが発見しやすい

# 状態遷移で隠れた要求を見つけ出そう

- 要求は曖昧
  - 背景や経験で提示されないことがある
- 妥当性確認では表現されていない・暗黙の仕様を見つけ出し、要求を満たしているか確認することが必要

- その一つの手段に状態遷移テストを活用しよう！！！



# 演習1
# 状態とイベントを見つけよう

# 例題：ストップウォッチの仕様


- ハードウェア仕様
  - 押しボタン
    - スタートボタン
    - リセットボタン
  - カウンタ
    - 6桁
    - 単位は　AA’AA’’BB　AAAAは秒　BBは10ミリ秒単位
- ソフトウェア仕様
  - 電池を入れたらカウンタが00’00’’00表示する
  - スタートボタンを押すと計測を開始する
    - 時間通りにカウンタの数字が増えていく
  - 計測中にスタートボタンを押すと計測が一時停止し、カウンタの数字が増えなくなる
  - 一時停止中にスタートボタンを押すと一時停止したカウンタの数字から計測が再開する
    - 10ミリ秒以下は内部的に1ミリ秒までもち、1ミリ秒以下は0から再開
  - 一時停止中にリセットボタンを押すとカウンタは停止し、00’00’’00となる
  - 99’99’’99のあと計測を続けると00’00’’00に戻りそのまま数字が増えていく

# 演習1：状態遷移図・表を書いてみよう

## ※この演習は演習1シートを使用します

- STEP1：状態を見つけよう
  - ストップウォッチにはどんな状態があるでしょうか？

- 状態：期間
  - ○○から□□まで
  - 例
    - 小学生：小学校入学から卒業するまで
    - 開店中
      - 開店時間から閉店時間まで
      - 店員が開場し、入れるようになってから

# 演習1：状態遷移図・表を書いてみよう

## ※この演習は演習1シートを使用します

- STEP2：イベントを見つけよう
  - ストップウォッチにはどんなイベントがあるでしょうか？

- イベント：瞬間
  - □□を　過ぎたら・超えたら・されたら
  - 例
    - 3分過ぎたら(経過したら)
    - 80℃を超えたら
    - クリックされたら

# 演習1：状態遷移図・表を書いてみよう

※この演習は演習1シートを使用します

- STEP3：状態遷移表をつくろう
  1. 先程あげた状態とイベントを状態遷移表に配置してみよう
  2. 状態とイベントが交差するところに「後状態」を記入しよう

- STEP4：状態遷移図をつくろう

# 回答例　表

STEP3：状態遷移表を作ってみよう

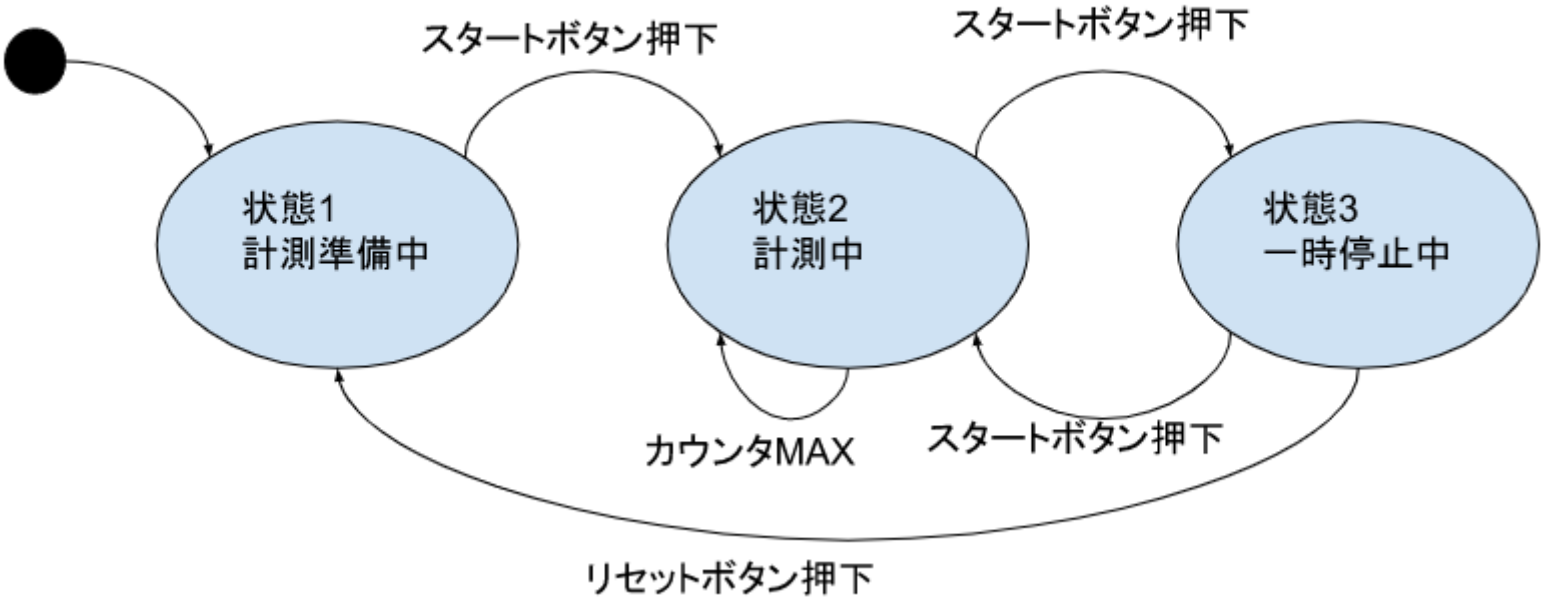
| 状態 ＼ イベント | | イベント1<br>スタートボタン押下 | イベント2<br>リセットボタン押下 | イベント3<br>カウンタMAX |
|---|---|---|---|---|
| 状態1 | 計測準備中 | →計測中<br>カウンタ開始 | - | N/A |
| 状態2 | 計測中 | カウンタストップ<br>→一時停止中 | - | カウンタリセット<br>(遷移しない) |
| 状態3 | 一時停止中 | →計測中<br>カウント開始 | カウンタリセット<br>→計測準備中 | N/A |

凡例）N/A：ありえない、-：受け付けない

# 回答例　図

STEP4：状態遷移図を作ってみよう

※以下を変更しても良いですし、好きに書いてください。困った場合は参考ツールを参照して作成してください。



# 状態遷移からテストケースを作成しよう

- 以下でテストケースができます
  - 前状態(今の状態)
  - イベント(操作)
  - 後状態(期待値)

✓ 状態の典型的な順序をカバーする。
  ✓ ハッピーパス
✓ すべての状態をテストする。
✓ すべての遷移をテストする。
✓ 遷移を特定の順序でテストする。

✓ 無効な遷移をテストする

# 状態遷移からテストケースを作成しよう

- 先程のストップウォッチでテストケースを作ると以下のようになる

| No | 前状態 | イベント | 後状態 | 何を見るか | |
|---|---|---|---|---|---|
| 1 | 計測準備中 | スタートボタン押下 | 計測中 | カウンタが進み始めること | ハッピー |
| 2 | 計測準備中 | リセットボタン押下 | 計測準備中 | カウンタに変化がないこと | 無効な遷移 |
| 3 | 計測準備中 | カウントMAX | N/A | | |
| 4 | 計測中 | スタートボタン押下 | 一時停止中 | カウントが止まり<br>その時のカウントがカウンタに表示されること | ハッピー |
| 5 | 計測中 | リセットボタン押下 | 計測中 | カウントを続けること | 無効な遷移 |
| 6 | 計測中 | カウントMAX | 計測中 | カウンタが00'00"00になり、カウントが止まらず、進むこと | ハッピー |
| 7 | 一時停止中 | スタートボタン押下 | 計測中 | カウントを再開すること | ハッピー |
| 8 | 一時停止中 | リセットボタン押下 | 計測準備中 | カウンタが00'00"00になり、カウントが止まること | ハッピー |
| 9 | 一時停止中 | カウントMAX | N/A | | |

# おまけ：順序を考えたものの例

| No | 前状態 | イベント | 後状態 | 何を見るか |
|---|---|---|---|---|
| 1 | 計測準備中 | リセットボタン押下 | 計測準備中 | カウンタに変化がないこと |
| 2 | 計測準備中 | スタートボタン押下 | 計測中 | カウンタが進み始めること |
| 3 | 計測中 | リセットボタン押下 | 計測中 | カウントを続けること |
| 4 | 計測中 | カウントMAX | 計測中 | カウンタが00’00”00になり、カウントが止まらず、進むこと |
| 5 | 計測中 | スタートボタン押下 | 一時停止中 | カウントが止まり<br>その時のカウントがカウンタに表示されること |
| 6 | 一時停止中 | スタートボタン押下 | 計測中 | カウントを再開すること |
| 7 | 計測中 | スタートボタン押下 | 一時停止中 | カウントが止まり<br>その時のカウントがカウンタに表示されること |
| 8 | 一時停止中 | リセットボタン押下 | 計測準備中 | カウンタが00’00”00になり、カウントが止まること |
| 9 | 計測準備中 | カウントMAX | N/A | |
| 10 | 一時停止中 | カウントMAX | N/A | |

スムーズにするため追加したもの
必ずしも追加が良いとは限らない

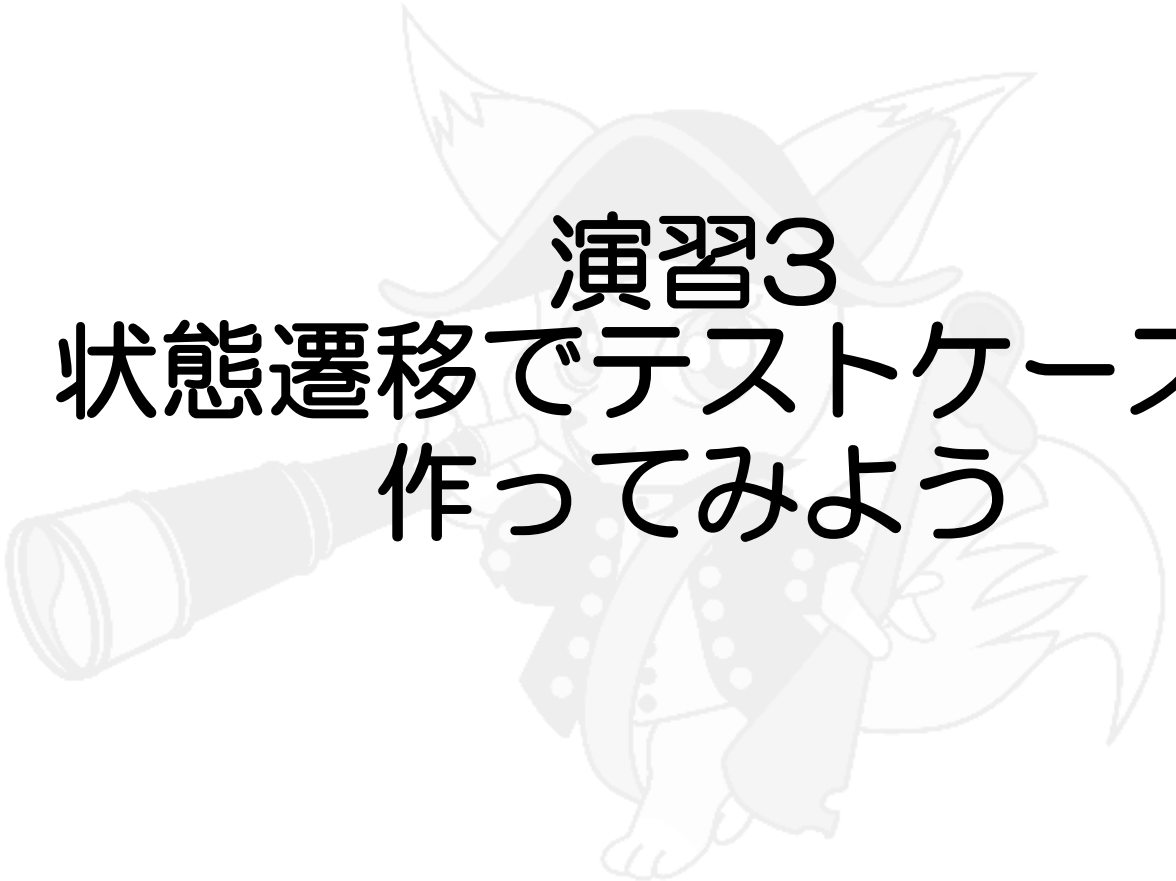
# おまけ：順序を考えたものの例2

| No | 前状態 | イベント | 後状態 | 何を見るか |
|---|---|---|---|---|
| 1 | 計測準備中 | リセットボタン押下 | 計測準備中 | カウンタに変化がないこと |
| 2 | 計測準備中 | スタートボタン押下 | 計測中 | カウンタが進み始めること |
| 3 | 計測中 | リセットボタン押下 | 計測中 | カウントを続けること |
| 4 | 計測中 | スタートボタン押下 | 一時停止中 | カウントが止まり<br>その時のカウントがカウンタに表示されること |
| 5 | 一時停止中 | スタートボタン押下 | 計測中 | カウントを再開すること |
| 6 | 計測中 | スタートボタン押下 | 一時停止中 | カウントが止まり<br>その時のカウントがカウンタに表示されること |
| 7 | 一時停止中 | リセットボタン押下 | 計測準備中 | カウンタが00’00”00になり、カウントが止まること |
| 8 | 計測準備中 | スタートボタン押下 | 計測中 | カウンタが進み始めること |
| 9 | 計測中 | カウントMAX | 計測中 | カウンタが00’00”00になり、カウントが止まらず、進むこと |
| 10 | 計測準備中 | カウントMAX | N/A | |
| 11 | 一時停止中 | カウントMAX | N/A | |

# ～休憩～

# 演習2
# 状態遷移でヌケモレを見つけよう

# 演習3
# 状態遷移でテストケースを作ってみよう



# まとめ

# まとめ

- 状態遷移は隠れた要求を見つけたり、抜けを見つけるのに適している
- 図と表の2つのビューにより、説明しやすい・理解しやすい

- 隠れた要求や抜けを早く見つけることで開発を楽に！
  - テストをするのは開発の最後だけでない、早い段階から
    - W字モデル
    - シフトレフト

# やる気になった方々へ

自分の作ったテスト設計が良いものか知りたくないですか？
他の人がどんなふうにまとめているのか知りたくないですか？

そんな方のためにASTERでは、

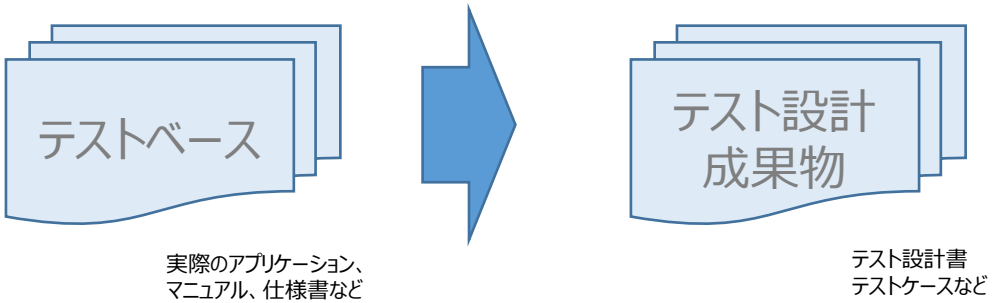
**テスト設計コンテスト**を開催しています！！

# テスト設計コンテストについて

- テスト設計コンテストの目的
- テスト設計コンテストの形式
- テスト設計コンテスト参加のメリット！
- テスト設計コンテストのこれまでの歩み
- テスト設計コンテスト OPENクラス審査基準
- テスト設計コンテスト OPENクラス成果物

# テスト設計コンテストの目的

- 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




テスト設計コンテストはこちら

https://aster.or.jp/testcontest/index.html

# テスト設計コンテストの形式

参加者を問わないOPENクラスと、
全チームメンバが30歳以下限定のU-30クラスがあります。

テスト設計成果物の良さを競うコンテスト。
バグ出しコンテストではありません。

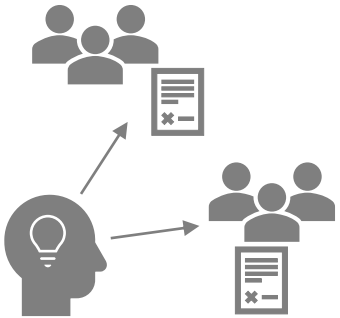
各チームは共通のテストベースに対するテスト設計をおこなって、
成果物を提出し、予選会や決勝戦でプレゼンテーションする。

# テスト設計コンテスト参加のメリット！

## テスト設計経験豊富な審査委員からのフィードバック

予選・決勝後に、審査を担当した全審査委員からのフィードバックコメントシートが届きます。

※U-30クラスはさらに予選後、
チームごとに時間を取ってフィードバック



## 他チームのテスト設計成果物から学ぶ

どのチームも同じテストベースを扱う

自分たちが一度テスト設計をしたテストベースなので、他チームのテスト設計成果物を見たときの学びが深い

# テスト設計コンテストのこれまでの歩み

| 開催年 | 結果 | 開催年 | 結果 |
|---|---|---|---|
| 2011 | 【大賞】めいしゅ館（東海）<br>【湯本賞】奥村 健二（東海）、【にし賞】堀米 賢（東京） | 2012 | 【最優秀賞】TETTAN（東京）<br>【審査員特別賞】あまがさきてすとくらぶ（関西） |
| 2013 | 【優勝】TETTAN（東京） 2連覇<br>【準優勝】Yuki Da RMA（北海道） | 2014 | 【優勝】ＴＦＣ　ＫＡ･ＲＩ･ＹＡ （東海）<br>【準優勝】MKE98（審査委員推薦枠：東海） |
| 2015 | 【優勝】しなてす（東京）<br>【準優勝】TEVASAKIplus （東京） | 2016 | 【優勝】SASADAN Go （書類選考）<br>【準優勝】しなてす （東京） |

| 開催年 | 結果 | |
|---|---|---|
| 2017 | OPENクラス<br>【優勝】STUDIO IBURI（北海道）<br>【準優勝】わんだーズ （東京） | U-30クラス<br>【優勝】でこパン462<br>【準優勝】SHINNOSUKE |
| 2018 | OPENクラス<br>【優勝】てすにゃんV3（審査委員特別枠）<br>【準優勝】フワパン（東京） | U-30クラス<br>【優勝】TBD<br>【準優勝】新米 |
| 2020 | OPENクラス<br>【優勝】セクシーゴリラ<br>【準優勝】出席番号となり同士 | U-30クラス<br>【優勝】一等米<br>【準優勝】王バーフロー |
| 2021 | OPENクラス<br>【優勝】テス豆<br>【準優勝】シン･田町補充計画 | U-30クラス<br>【優勝】はじめての共同作業<br>【準優勝】まちがいさがし。 |
| 2022 | OPENクラス<br>【優勝】テス豆<br>【準優勝】ジョゼ | U-30クラス<br>【優勝】勇往米進<br>【準優勝】つよつよなりたいはんぺん |

# テスト設計コンテスト OPENクラス審査基準

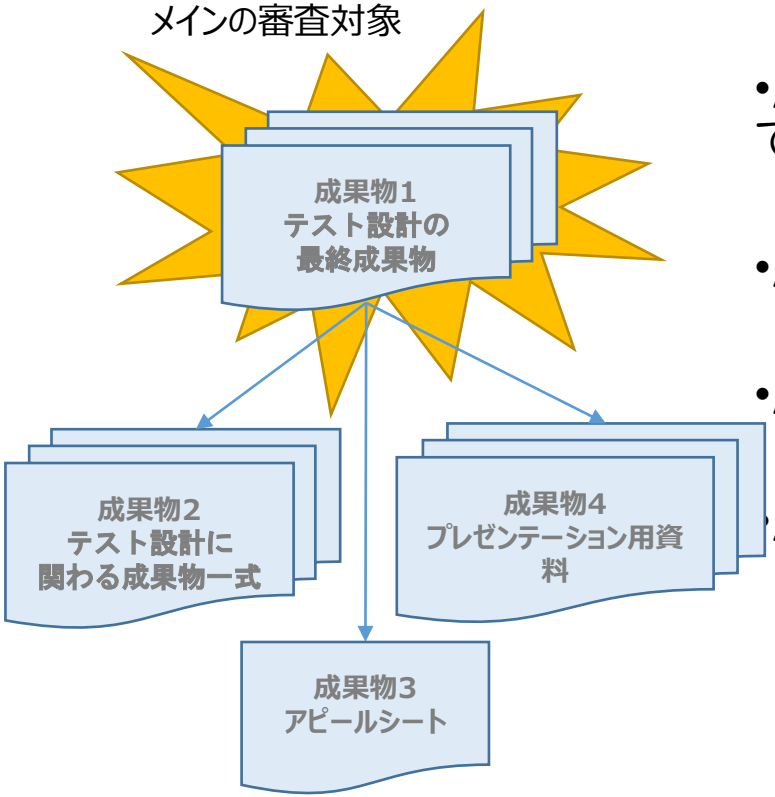
*成果物*審査点　合計120点満点
- テスト分析・テスト設計点(40点)※
- テスト詳細設計・実装点(30点)
- 工程間一貫性点(10点)
- 文書点(20点)
- 総合点(20点)

※U-30クラスは以下に分かれています
- テスト妥当性点(20点)
- テスト分析・テスト設計点(20点)

- プレゼンテーション技術点　15点

審査基準はテスト設計コンテスト 公式ページにて公開
https://aster.or.jp/testcontest/open.html#sinsa
https://aster.or.jp/testcontest/u30.html#sinsa2

# テスト設計コンテスト OPENクラス成果物




- 成果物１ ： テスト設計の最終成果物：40ページまで
  - メインの審査対象
- 成果物２ ： テスト設計に関わる成果物一式
  - テスト設計を行う過程
- 成果物３ ： アピールシート
  - 各チーム紹介、狙いなどを表現
- 成果物4：プレゼンテーション用資料
  - プレゼンテーション：20分(発表15分、質疑応答5分)で自チームのテスト設計を説明

# Q&A

以上で終わりになります。
ご清聴ありがとうございました。

# APPENDIX:参考文献


資料

[Kazman] Humberto Cervantes, Rick Kazman, ADD 3.0: Rethinking Drivers and Decisions in the Design Process, SATURN2015.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=436536

JaSST'16 Tohoku
http://www.jasst.jp/symposium/jasst16tohoku/report.html

ソフトウェアテストシンポジウム 2021 東海 【初心者向け】テストを導くためのテストアーキテクチャの組み立て方
https://speakerdeck.com/goyoki/cetam

チュートリアル資料

テスト設計チュートリアル ちびこん編
https://aster.or.jp/testcontest/tutorial.html#tutorial_chibicon

テスト設計チュートリアル テスコン編
https://aster.or.jp/testcontest/tutorial.html#tutorial_tescon

ソフトウェアテストシンポジウム 2020/2021 北陸 テスト設計初心者向けチュートリアル
http://www.jasst.jp/symposium/jasst21hokuriku/pdf/B1.pdf

Youtube

テスト要求分析チュートリアル(2:00:15)
https://www.youtube.com/watch?v=KrMwXNtotbU

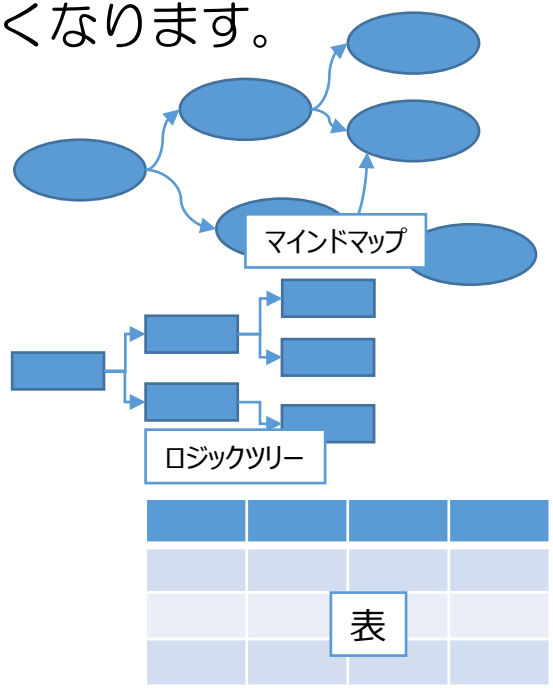
# APPENDIX:図や表の活用

図や表を活用すると考えがまとまりやすくなります。

- 俯瞰
- ルール
  - 配置
  - 線の意味

よくあるツール

- 図
  - マインドマップ
  - ロジックツリー
- 表
  - デシジョンテーブル




使いこなすためのポイント

- 条件が複雑なときは分解する
  - 考える範囲を減らす
- バラバラな粒度をあわせる
  - 同じレベル、隣の行、列で浮いていないかがチェックできます。