テストの設計意図を届けよう
～テストケース・テストスクリプトだけ渡していませんか？

JaSST'22 Tokyo

テスト設計コンテスト　U-30クラス　セッション

井芹 久美子（テスト設計コンテスト審査委員会）
坂 静香（テスト設計コンテスト審査委員会）

このセッションについて

想定している聴講者

- テストケース・テストスクリプトを作成したことがある
- 自分が作成したテストケース・テストスクリプト・その他テスト設計成果物に対して満足していない、なにかモヤモヤしている
- 自分が作成したテスト設計成果物をリーダーや開発者に確認してもらうと、無反応や厳しい反応など、嬉しくないことが起こる
- 自分が作成したテストケース・テストスクリプトを実行してもらうと、自分が思ったように実行されずに確認漏れが出てしまう
- 上記のような悩みを持つメンバーに対して、どうアプローチしようか悩んでいる
- 特に悩みはないけれども、なんか参考になるかも？と思っている
- テスコンU-30に挑戦しようかなー・・・と思っている

このセッションについて

- このセッションでお話しようとしていること
  - テストの設計意図を届けよう
  - テスト設計の過程を見せよう
  - Whyを明確にしよう
  - 構造を見せよう
  - 次にできる小さな工夫

テスト設計コンテストU-30クラスについて

http://www.aster.or.jp/testcontest/

TEST DESIGN COMPETITION

- テスコンはJaSSTの企画セッションとして始まった
  その名の通り、テスト設計の腕を磨き競う場
- ‘13からASTERの事業として開催
- ‘17から、OPENクラスとU-30クラスに分けて開催
  - U-30は30歳以下の実務経験の浅い技術者が経験を積む場
  - 提示したテストベースに対して、「まずはやってみよう」「チュートリアルの内容を踏まえて一通りのプロセスをやりきろう」を目指してテスト設計に挑戦していただく

テスト設計コンテスト’22 開催します！

- 参加受付中！　テストベースがスマホアプリに変わりました！！
- 予選後に審査委員とじっくり話せるフィードバック会を開催！
- 決勝戦は9/17(土)AM！

自己紹介

- 坂 静香
  - JaSST Tokyo実行委員・TOCfE Bootcampスタッフ
  - テスト設計コンテストU-30クラス審査委員
  - ↑なのに、テストのスタイルはフリーダム探索派（野生動物）

- 井芹 久美子
  - テスト設計コンテストU-30クラス審査委員、JSTQB技術委員
  - WACATE2020冬の招待講演やJaSSTなどで登壇経験を持つ
  - 書籍「実践ソフトウェアエンジニアリング(第9版)」の翻訳に参加

講演のすすめかた

Vimeo (セッション後はDiscord音声チャンネル)

Discord テキストチャンネル

セッション時間内

前半
・背景・困りごと
・テスト設計の過程を見せる
・Whyを明確にする
講師

後半
・構造を見せる
・次にできる小さな工夫
講師

残り時間で質問をピックアップして回答をしていきます
講師

Ask the speaker (10分程度)

セッション後、Discord TrackC の音声チャンネルで対話形式で質疑応答を受け付けます
講師 参加者

参加者

質問は随時Discordのセッションチャンネルに記載してください

その他感想やフィードバックなど、ご自由にコメントを記載してください

講師

Discordに記載された質問に対して、回答を返信していきます

(セッション後も口頭回答したものも含め、残りの質問に回答を入れていきます)

前半パートでお話すること

- 背景
  - **テストの成果物を見る人たち**
  - **テスト設計成果物を見せるときの困りごと**
- テストの設計意図を届けるために始めよう
  - **テスト設計の過程を見せよう**
  - **Whyを明確にしよう**
  - 構造を見せよう
- 次にできる小さな工夫

テスト設計の成果物を見る人たち

テスト設計に込められた意図を汲みたい人達の関心事

自分の実装に問題がないか確認したい
実装時に気をつけることを把握したい
心配な箇所をテストしてくれるか確認したい

開発者

なるべく手数は減らしたい
ポイントを押さえてテスト実行したい

テスト実行者

テスト実行のコントロールをしたい
優先度を決める情報を得たい
テスト環境構築の作戦をたてたい

テストリーダー

受け入れ基準を満たすことを証明する
テストができているか把握したい

プロダクトオーナー

様々な立場の関心事に応えられる成果物を作り、見せる必要がある

テスト設計の成果物ができた後にどうしていますか？

以下のようなことが起きていて、困っていたりしませんか？

- 誰かに**確認してもらわず**、そのままテスト実行している
- テストリーダーや開発者にレビューをお願いしているが、たいてい**無反応**でそのままテスト実行することになる
- 成果物を見せた相手から**「なにを確認したくてこのテストをするの？？」**と質問される（仕様書どおりに動くことを確認したいに決まっているのに？？・・・と思う）
- レビュー会で説明したら、テストケースに対する指摘ではなく、「それで、**網羅できているんですか？**」と聞かれる
- レビュー会で、テストケースを読み続けて**時間切れ**になる

テスト設計成果物で、なにか改善できることはあるだろうか？

いきなりこんな成果物を見せていませんか？

仕様書章	要求ID	仕様ID	ケースID	前提条件	手順	期待結果	優先度	備考
3.1 沸騰行為	pot-310	pot-310-11	310-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	沸騰ランプを点灯し、保温ランプを消灯する	A	
	pot-310	pot-310-12	310-12-1	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	操作パネルの温度/モード表示窓にポットの中の水の温度を表示する	C	
	pot-310	pot-310-12	310-12-2	沸騰行為中	1.沸騰中に温度表示が上がった瞬間にポットの蓋を開けて温度計で温度を測る	測定温度の小数点以下を四捨五入した温度が操作パネルの温度/モード表示窓に表示されている	C	
	pot-310	pot-310-21	310-21	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.沸騰するまで様子を見る	操作パネルの温度/モード表示窓に100と表示される	A	テスト実施機は開発機ではないため、温度は操作パネルの温度/モード表示窓で確認する
	pot-310	pot-310-31	310-31-1	ポットの蓋が開いている	1.ポットに110℃以上の液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る	C	基本は開発側でテストしてもらう 余裕があれば実機でも確認する
	pot-310	pot-310-31	310-31-2	保温モードが高温モードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る	C	基本は開発側でテストしてもらう 余裕があれば実機でも確認する
	pot-310	pot-310-31	310-31-3	保温モードが節約モードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	沸騰行為を止める (30秒間ブザーが鳴る)	C	基本は開発側でテストしてもらう 余裕があれば実機でも確認する
	pot-310	pot-310-31	310-31-4	保温モードがミルクモードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	沸騰行為を止める (30秒間ブザーが鳴る)	C	基本は開発側でテストしてもらう 余裕があれば実機でも確認する
	pot-310	pot-310-31	310-31-5	ヒーターが動作しない状態の機体を用意する	1.ポットに98℃のお湯を入れる 2.ポットの蓋を閉める 3.ポットの中のお湯が92℃になるまで待つ	沸騰行為を止める (30秒間ブザーが鳴る)	C	基本は開発側でテストしてもらう 余裕があれば実機でも確認する
	pot-310	pot-310-31	310-31-6	沸騰行為中	1.蓋を開ける 2.しばらく放置して様子を見る	沸騰行為が終わる (沸騰しない)	A	
	pot-310	pot-310-31	310-31-7	給湯して水位を第1水位センサーギリギリにする	1.沸騰ボタンを(100msec以上)押して沸騰行為を開始させる 2.蒸発して水位が第1水位センサーより下になる	沸騰行為が終わる (それ以上温度が上がらない)	B	
	pot-311	pot-311-11	311-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.操作パネルの温度/モード表示窓に100と表示されてから、沸騰行為が終わるまでストップウォッチで時間を測定する	100℃になってから3分間沸騰行為が続くこと (ヒーターをONにし続けること)	A	
	pot-312	pot-312-11	312-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.操作パネルの温度/モード表示窓に100と表示されてから3分待つ	沸騰行為が終わり、保温行為をする	A	3分はカルキ抜きの加熱
3.2 保温行為	pot-320	pot-320-11	320-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.操作パネルの温度/モード表示窓に100と表示されてから3分待つ	保温ランプを点灯し、沸騰ランプを消灯する	C	pot-312-11の確認と兼ねる
	pot-320	pot-320-12	320-12	カルキ抜きの加熱終了後 給湯していない	1.操作パネルの温度/モード表示窓の温度表示を観察する 2.保温中に温度表示が下がった瞬間にポットの蓋を開けて温度計で温度を測る	測定温度の小数点以下を四捨五入した温度が操作パネルの温度/モード表示窓に表示されている	C	
	pot-320	pot-320-21	320-21-1	保温モードが高温モードになっている	1.沸騰ボタンを(100msec以上)押して沸騰させる 2.保温になって98℃になるまで待つ 3.そのまま30分おきに温度表示を確認する 4.30分おきに給湯して、お湯の温度を測る	水温が98℃に保たれる	A	観察時間はできれば1日を通して行う 難しければ1日のうち3回計測する
	pot-320	pot-320-21	320-21-2	保温モードが節約モードになっている	1.沸騰ボタンを(100msec以上)押して沸騰させる 2.保温になって90℃になるまで待つ 3.そのまま30分おきに温度表示を確認する 4.30分おきに給湯して、お湯の温度を測る	水温が90℃に保たれる	A	観察時間はできれば1日を通して行う 難しければ最低3回は計測する
	pot-320	pot-320-21	320-21-3	保温モードがミルクモードになっている	1.沸 2.保 3.そ 4.30			

話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版　9ページ を参照して作成

テスト設計の意図を知るために読みたい、だけど・・・

もしも、大量のテストケース・テストスクリプトだけを渡したら、
どのようなことが起こるのでしょうか？

- ・・・・・（思わずファイルを閉じて現実逃避する）
- 読み解くために時間をかける必要がある
- 読み解いたとして、抜け漏れがあるかどうかを判別するのが困難
- テストの手順や期待結果はわかるが、何を確認したくてこのテストをやるのか？テストの必要性がよくわからない
- 何を確認したいのかは書かれていても、期待結果がなんか違う（合致しないため、作成者に時間をかけて確認する必要がある）

テスト実行の時点で・・・

もしも、困っていることが解決されないままテスト実行に入ったら、テスト実行者はどうするのでしょうか？

- 書かれたとおりに実行して、書かれた期待結果を確認する（抜け漏れがあってもフォローはされない）
- 書かれたとおりに実行するが、気になる箇所も確認する（テスト実行時に実行結果が残らない、あるいはテストケースが追加されるが、テスト実行者のスキルに依存する）
- レビューア同等の反応を示すため、いろいろ気になってテスト実行が進まない
- テストスクリプトを無視する（チェックだけつける）

テストの意図が届かないままテストが終了してしまう

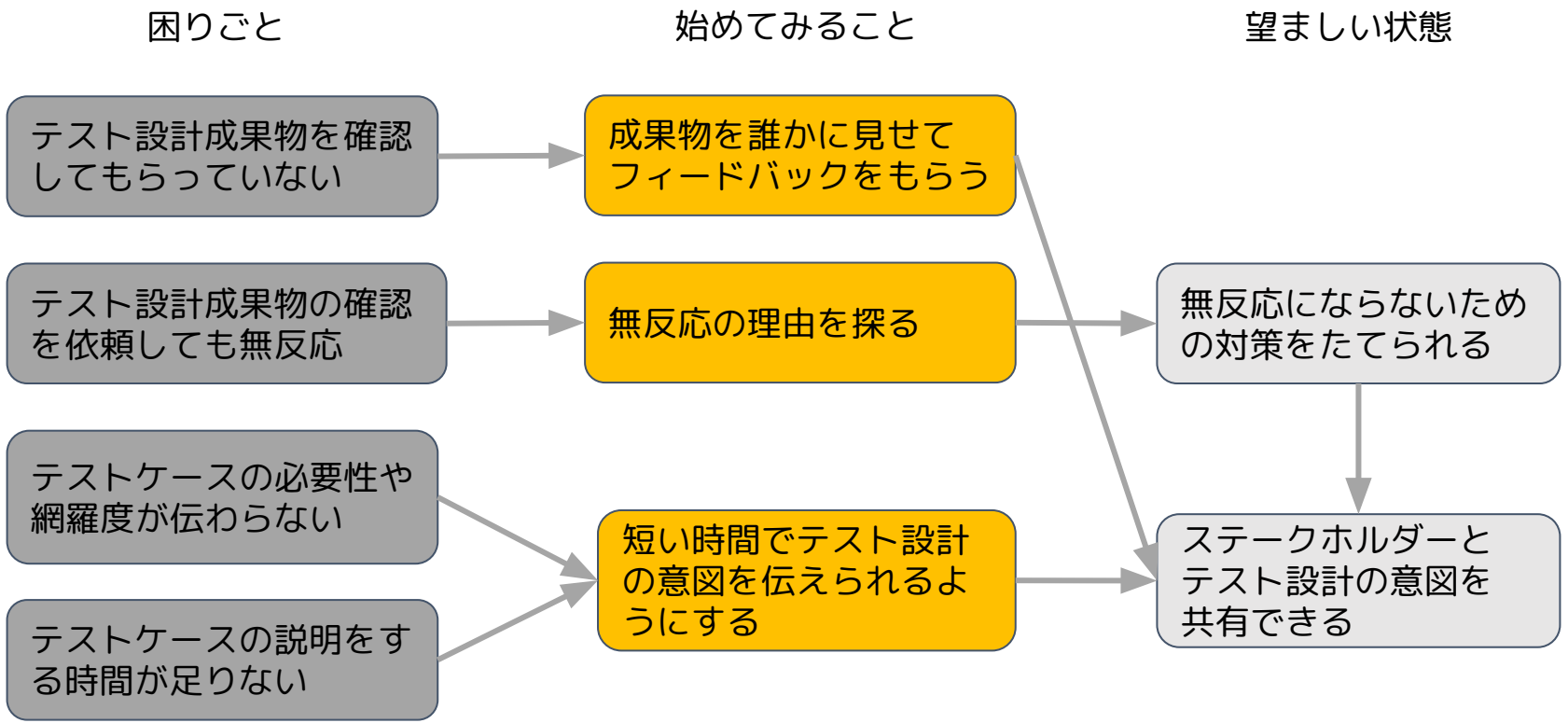
「この成果物でテスト実行したら、
狙った通りにテストができる」ようにしたい

だけど・・・

- 周りから「大丈夫」と言われないまま実行に進む
- 指摘により、大丈夫ではないことはわかったが、
  その場しのぎの回答をするだけで、実行に進む
- その場で何も指摘がなかったのに、
  テスト実行やその後で問題が発覚する

テスト設計の意図について、ステークホルダーと認識合わせが
できていないことで、誤解に気づかない・抜け漏れに気づかない
ステークホルダーへの説明責任を果たせないままテストが終わる

テストの意図を届けるために始めよう



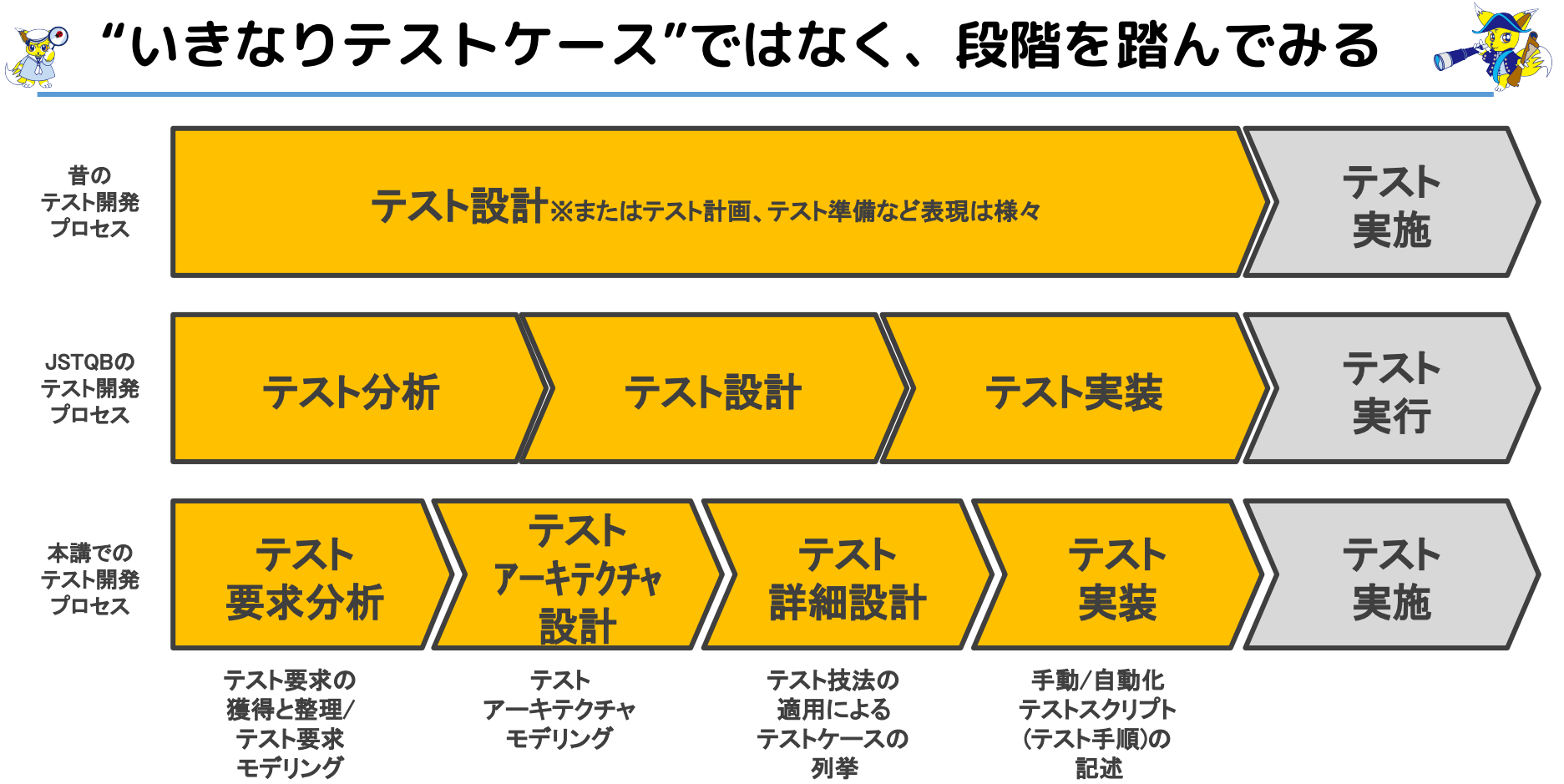
テストの意図を届けるために始めよう

1. 成果物を誰かに見せてフィードバックをもらおう
   a. 成果物をベースに対話することで互いの理解を促進する
   b. フィードバックによりテスト設計成果物を磨く
   c. フィードバックに隠れたプロダクト愛を探す
2. 無反応の理由を探ろう
   a. 無反応≠問題が無い　ということに気づく
   b. 成果物を渡す相手を知り、相手が求めることを知る
3. **短い時間で設計の意図を伝えられるようにしよう**
   a. テストケースになるまでの過程を見せる
   b. Whyを明確にする
   c. 構造を見せる

テスト設計成果物で改善してみよう

テストケースになるまでの過程を見せる

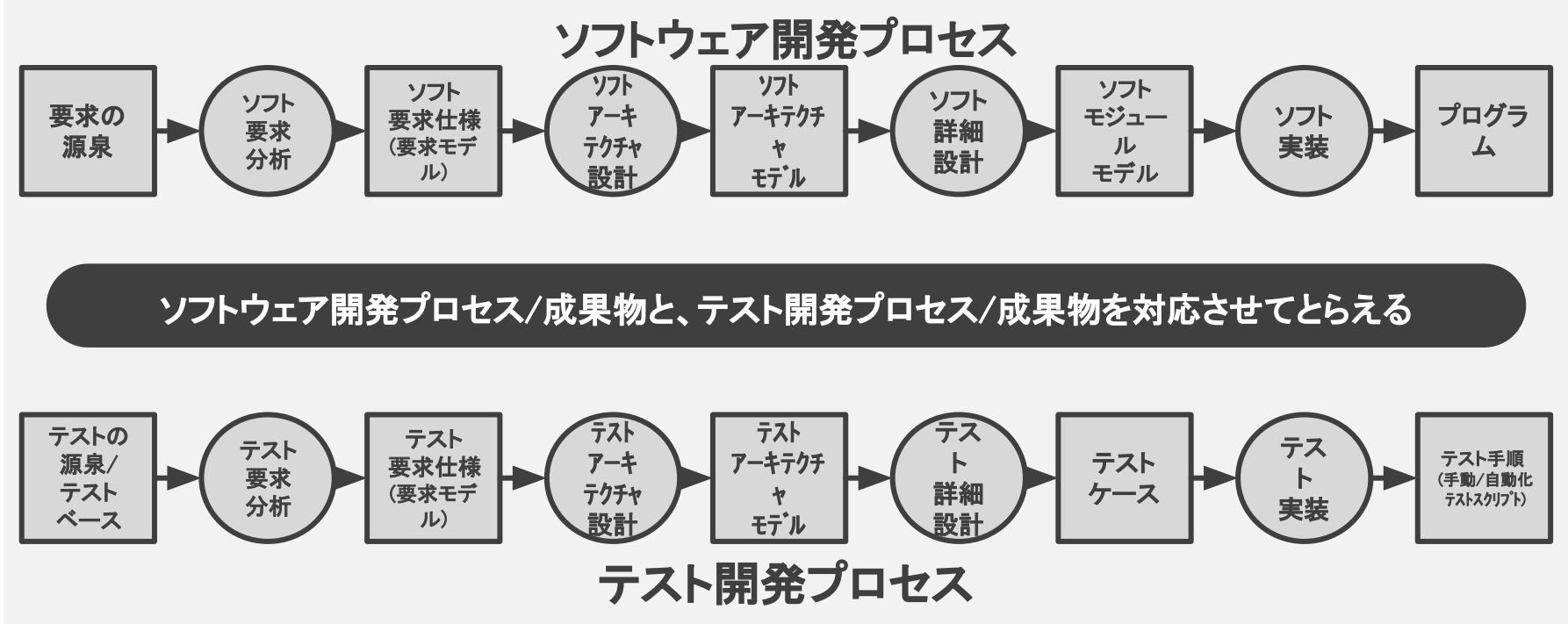
“いきなりテストケース”ではなく、段階を踏んでみる



引用元：テスト設計チュートリアル ちびこん編 ‘21 資料　67ページ

プロセスごとに成果物をつくることを目指す

テストケースを開発成果物と捉えて段階的に詳細化していく必要がある



引用元：テスト設計チュートリアル ちびこん編 ‘21 資料　69ページ

テストケースになるまでの過程を見せる

とはいえ・・・具体的にどんな成果物を作ればよいのかわからない・・・

成果物として整っていなくても大丈夫！

テストケースを作る過程で、もとになっている情報はあるはず！それを見せよう！

テスト設計の途中過程で、自分が考えたことを出力しよう

- テスト対象を理解するために描いてみた雑な絵
- 仕様書を読んだときのつぶやき
- 雑に書き散らかしたメモをちょっとまとめたもの

普段行なっている「○○テスト」について、
テスト目的やテストケースのつくりかたを見直してみよう

テストケースになるまでの間に参照したものを辿れるようにしよう

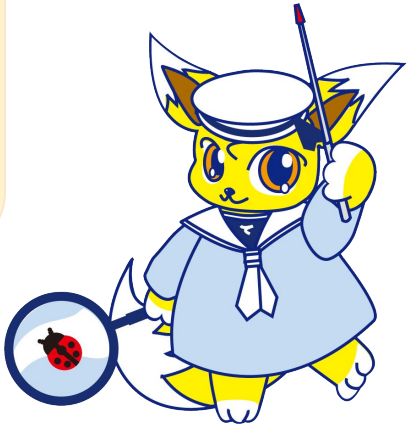
- 仕様書
- メール文面・チャット投稿
- 議事録

短い時間で設計の意図を伝える二つのポイント

さらに伝えやすくすることに挑戦してみよう！
すべてを事細かに伝えるほどには時間がないから・・・

- **Whyを明確にする**
- 構造を見せる

具体例を挙げながら
解説します！



Whyを明確にする

どんな「Why」があるのだろうか？

テスト設計コンテストのフィードバックから拾ってみた
（審査員がテスト設計成果物をみて、確認したくなったこと）

- テストの目的
  - なぜ　そのテストを行う必要があるのか？
- 妥当性の根拠
  - なぜ　このテスト設計で妥当と判断したのか？
  - なぜ　そのテストは設計・実装されなかったのか？
  - なぜ　このカバレッジ基準で妥当と判断したのか？
  - なぜ　その優先度・重要度をつけたのか？
- 採用理由
  - なぜ　この技法・手法を適用したのか？
  - なぜ　その品質特性を選んだのか？
  - なぜ　そのリスクを対策しようとしたのか？
  - なぜ　その分け方にしたのか？
- 存在理由
  - その成果物は　なんのために必要なのか？

Whyに答えられる？

列挙したWhyに答えてみよう

- 答えられない・・・
  - これまで、どうしてか？を考えたことがなかった
  - どうやったら考えられるのかがわからない
- そんなこと答えなきゃいけないの？？
  - これまでとずっと同じなのに？
  - 常識でしょう？
  - 打ち合わせで認識合わせしているし、わかっているはず！
- 答えているけれど伝わらない・・・
  - 書いているのに見つけてもらえない
  - 相手に納得してもらえない

テスト設計成果物で、なにか改善できることはあるだろうか？

答えられないのは、Whyを考えられないのだろうか？

書かれたとおりに実行するが、気になる箇所も確認する
（テスト実行時に実行結果が残らない、あるいは**テストケースが追加される**が、テスト実行者のスキルに依存する）

12ページ(テスト実行の時点で・・・)の記載を再掲

テスト設計時にも同じことが起こっている
テストケースをよーくみると、仕様に書かれていない、気になる箇所を盛り込んだテストケースが時折登場している

優先順位も、なにか方針を持ってつけていることだけは伝わる

・・・きっと、テスト設計の段階で思いついている

Whyが隠れているだけで実は考えている！

まずは「考えている」ことを自覚しよう！認識しよう！出力しよう！

Whyに答えられる？

列挙したWhyに答えてみよう

- 答えられない・・・
  - これまで、どうしてか？を考えたことがなかった
  - どうやったら考えられるのかがわからない
- そんなこと答えなきゃいけないの？？
  - これまでとずっと同じなのに？
  - 常識でしょう？
  - 打ち合わせで認識合わせしているし、わかっているはず！
- 答えているけれど伝わらない・・・
  - 書いているのに見つけてもらえない
  - 相手に納得してもらえない

テスト設計成果物で、なにか改善できることはあるだろうか？

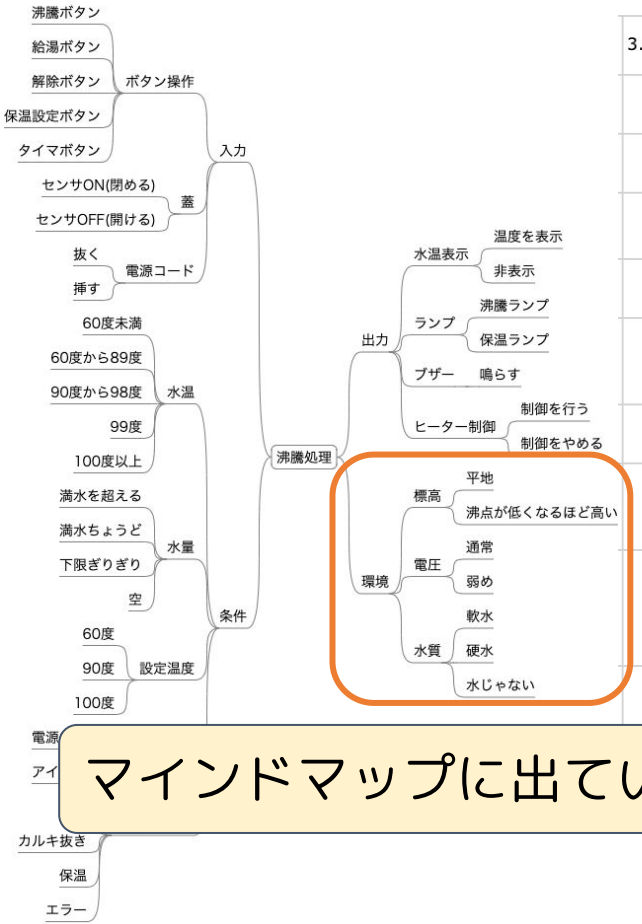
Whyの答えが伝わらないのはなぜだろう？

- **Whyの答えが隠れる**
  - 根拠となる情報が出力できていない
    根拠が複数あるのに一部しか示していない
  - 根拠となる情報を繋げていない
  - **暗黙の了解は実は得られていない、忘れられている**
- **示したWhyの答えのとおりに成果物ができていない**
  - 間に挟まるはずの成果物が出せていない
  - 繋げたくても繋がらない
    （繋げようとしてもうまくいかなくて諦めている）
- **Whyに対する答えは唯一絶対解ではない**
  - ヒトの考えは十人十色
    （自分の伝えたい解は他の人が考える解と同じとは限らない）
  - ヒトの解釈も十人十色
    （自分が期待する解釈を他の人がしてくれるとは限らない）

Whyの答えが伝わらないのはなぜだろう？

- Whyの答えが隠れる
  - 根拠となる情報が出力できていない
    根拠が複数あるのに一部しか示していない
  - **根拠となる情報を繋げていない**
  - 暗黙の了解は実は得られていない、忘れられている
- 示したWhyの答えのとおりに成果物ができていない
  - **間に挟まるはずの成果物が出せていない**
  - 繋げたくても繋がらない
    （繋げようとしてもうまくいかなくて諦めている）
- Whyに対する答えは唯一絶対解ではない
  - ヒトの考えは十人十色
    （自分の伝えたい解は他の人が考える解と同じとは限らない）
  - ヒトの解釈も十人十色
    （自分が期待する解釈を他の人がしてくれるとは限らない）

段階的に出力しているのに繋がらない・・・



3.1 沸騰行為	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	沸騰ランプを点灯し、保温ランプを消灯する
	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	操作パネルの温度/モード表示窓にポットの中の水の温度を表示する
	沸騰行為中	1.沸騰中に温度表示が上がった瞬間にポットの蓋を開けて温度計で温度を測る	測定温度の小数点以下を四捨五入した温度が操作パネルの温度/モード表示窓に表示されている
	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.沸騰するまで様子を見る	操作パネルの温度/モード表示窓に100と表示される
	ポットの蓋が開いている	1.ポットに110℃以上の液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る
	保温モードが高温モードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る
	沸騰行為中	1.蓋を開ける 2.しばらく放置して様子を見る	沸騰行為が終わる (沸騰しない)
	給湯して水位を第1水位センサーギリギリにする	1.沸騰ボタンを(100msec以上)押して沸騰行為を開始させる 2.蒸発して水位が第1水位センサーより下になる	沸騰行為が終わる (それ以上温度が上がらない)
	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.操作パネルの温度/モード表示窓に100と表示されてから、沸騰行為が終わるまでストップウォッチで時間を測定する	100℃になってから3分間沸騰行為が続くこと (ヒーターをONにし続けること)
	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.操作パネルの温度/モード表示窓に100と表示	沸騰行為が終わり、保温行為をする

マインドマップに出ている観点をテストケース上に見つけられない

話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版　9ページ を参照して作成

段階的に繋がりを見せているつもりでも・・・

章	要求ID	確認したいこと
3.1 沸騰行為	pot-310	水を沸騰させること
	pot-311	カルキ抜きをすること
	pot-312	カルキ抜きが終わったら保温すること
	pot-320	設定モードの温度でポット内の水温を保持すること

テストで確認したいことと、
実際にできたテストケースで
乖離がある

3.1 沸騰行為	pot-310	pot-310-11	310-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	沸騰ランプを点灯し、保温ランプを消灯する
	pot-310	pot-310-12	310-12-1	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	操作パネルの温度/モード表示窓にポットの中の水の温度を表示する
	pot-310	pot-310-12	310-12-2	沸騰行為中	1.沸騰中に温度表示が上がった瞬間にポットの蓋を開けて温度計で温度を測る	測定温度の小数点以下を四捨五入した温度が操作パネルの温度/モード表示窓に表示されている
	pot-310	pot-310-21	310-21	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.沸騰するまで様子を見る	操作パネルの温度/モード表示窓に100と表示される
	pot-310	pot-310-31	310-31-1	ポットの蓋が開いている	1.ポットに110℃以上の液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る
	pot-310	pot-310-31	310-31-2	保温モードが高温モードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る

話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版　9ページ を参照して作成

Whyを明確に伝えるために

テスト設計の途中過程で、自分が考えたことを出力しよう
テストケースになるまでの間に参照したものを伝えよう

19ページ(テストケースになるまでの過程を見せる)の記載を再掲

途中過程を出力しただけでは、個々の情報は繋がらない
もう少し、ステップアップしてみる

- **根拠を示すもの**を意識して出力しよう
  - テストケース上で根拠を示してみよう
    そのテストで確認したいことを書いてみよう
  - **次の成果物に繋がるようなまとめ方**をしてみよう
- 出力したものや参照したものを繋げてみて、
  抜けている箇所があるかどうかを確認しよう

テストケース上に、隠れたWhyを見せてみる

仕様書章	要求ID	仕様ID	ケースID	確認したいこと	前提条件	手順	期待結果
3.1 沸騰行為	pot-310	pot-310-11	310-11	沸騰行為による、温度制御行為の表示	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	沸騰ランプを点灯し、保温ランプを消灯する
	pot-310	pot-310-12	310-12-1	沸騰行為による、温度制御行為の表示	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	操作パネルの温度/モード表示窓にポットの中の水の温度を表示する
	pot-310	pot-310-21	310-21	沸騰行為の温度制御方式	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.沸騰するまで様子を見る	操作パネルの温度/モード表示窓に100と表示される
	pot-310	pot-310-31	310-31-1	沸騰行為の停止	ポットの蓋が開いている	1.ポットに110℃以上の液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る
	pot-310	pot-310-31	310-31-2	沸騰行為の停止	保温モードが高温モードになっている	1.ポットに110℃未満だが3分経過しても98℃を下回らない液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る

あれ・・・？？
確認したいことと、期待結果にずれが生じている？
テストケースも確認したいことも足さないとだめみたい

話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版　9ページ を参照して作成

繋げるための成果物をつくってみる

章	キーワード	確認したいこと	パラメーター	該当するテストケース	該当する要求ID	要求仕様（概要）	気がかり
3.1 沸騰行為	入力	操作を行うことで沸騰を開始する	沸騰ボタン 蓋 電源コード	6.3状態遷移テストとして作成	(2章に該当する)	沸騰行為に遷移する	
	条件	条件が整っていれば沸騰行為をする (沸騰行為の温度制御方式)	水温 水量 設定温度 ポットの状態	310-21-xxx デシジョンテーブル作成	pot-310-21	"目標温度ON/OFF方式でヒーターを制御して、沸騰させる (目標温度100℃≦温度になるまで、ヒーターをONにし続ける)"	設定温度と水温の組み合わせはみておきたい（仕様としてよいのか？も含み確認したい）
	条件	100℃になったら3分間沸騰行為を続ける (カルキ抜きの実行)	水温 水量 設定温度 ポットの状態	6.3状態遷移テストとして作成	pot-311-11	サーミスタが100℃になったらさらに3分間ヒータをONにし続ける	
	条件	カルキ抜きが3分続いたら沸騰行為をとめて保温行為に遷移する (カルキ抜き終了後に保温行為に遷移する)	水温 水量 設定温度 ポットの状態	6.3状態遷移テストとして作成	pot-312-11	カルキ抜きが3分続いたら沸騰行為をとめて保温行為に遷移する	
	条件	条件が合致したときに沸騰行為を止める (沸騰行為の停止)	水温 水量 ポットの状態	6.3状態遷移テストとして作成	pot-310-31	以下のいずれかの状態になったら沸騰行為を止める ・エラー検知 ・蓋センサoff ・全ての水位センサがoff	
	出力	沸騰行為の際に適切な出力をする (沸騰行為による、温度制御行為の表示)	(入力・条件・環境いずれも正常な値を任意で設定)	310-11-xxx 310-12-xxx	pot-310-11 pot-310-12	沸騰ランプを点灯して保温ランプを消灯する 温度/モード表示窓にサーミスタの温度を整数で表示する	入力トリガーでランプが点灯しないことがあるかもしれない →入力のパターンを確認しておく
	環境	環境に応じた振る舞いをする (沸騰行為の温度制御方式)	標高 電圧 水質	310-21-xxx	pot-310-21？	要求仕様を確認する必要あり	沸点が100℃に満たないときにポットとして使えないのではないか？

それぞれの過程で小さく分割して成果物を出していくと、繋がりや過不足が見えやすくなるよ

繋がりやすくつくることも大事なんだね

話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版　9ページ を参照して作成

Whyに答えられる？

列挙したWhyに答えてみよう

- 答えられない・・・
  - これまで、どうしてか？を考えたことがなかった
  - どうやったら考えられるのかがわからない
- そんなこと答えなきゃいけないの？？
  - これまでとずっと同じなのに？
  - 常識でしょう？
  - 打ち合わせで認識合わせしているし、わかっているはず！
- 答えているけれど伝わらない・・・
  - 書いているのに見つけてもらえない
  - 相手に納得してもらえない

テスト設計成果物で、なにか改善できることはあるだろうか？

Whyに対する答えに納得してもらえないのはなぜだろう？

よく耳にする回答例

- 従来通りにしました！
- 仕様に書かれていることを一通りテストケースにしたので、網羅できています！
- テスト実行工数が少ないから、優先順位の高いものから進めて時間の許す限り実行する方針です！
- テストを効率よく実施するために組み合わせ技法を使いました！

検証してみる

仕様に書かれたことを一通りテストケースにする → 網羅したテストケースができている → テスト設計が妥当である

結果として・・・

結果として・・・

これでいいのかな？・・・
なにか足りない気がする・・・

検証してみる

仕様に書かれたことを一通りテストケースにする

テストで網羅できている

テスト設計が妥当である

仕様にすべての情報が書かれている

仕様に間違いや検討不足がない

隠れた前提を満たしていないと、納得してもらえなさそう

テストケースを作成しながらこの前提を満たすこともあるかも

検証してみる

仕様に書かれたことを一通りテストケースにする

仕様にすべての情報が書かれている

仕様に間違いや検討不足がない

仕様の検証をするためのテストについては、抜け漏れがない

仕様の検証以外に必要なテストも揃っている

テスト設計が妥当である

仕様に書かれたことをテストケースにするだけでは足りないかもしれないね

Whyの回答を検証するために、問いを投げかける

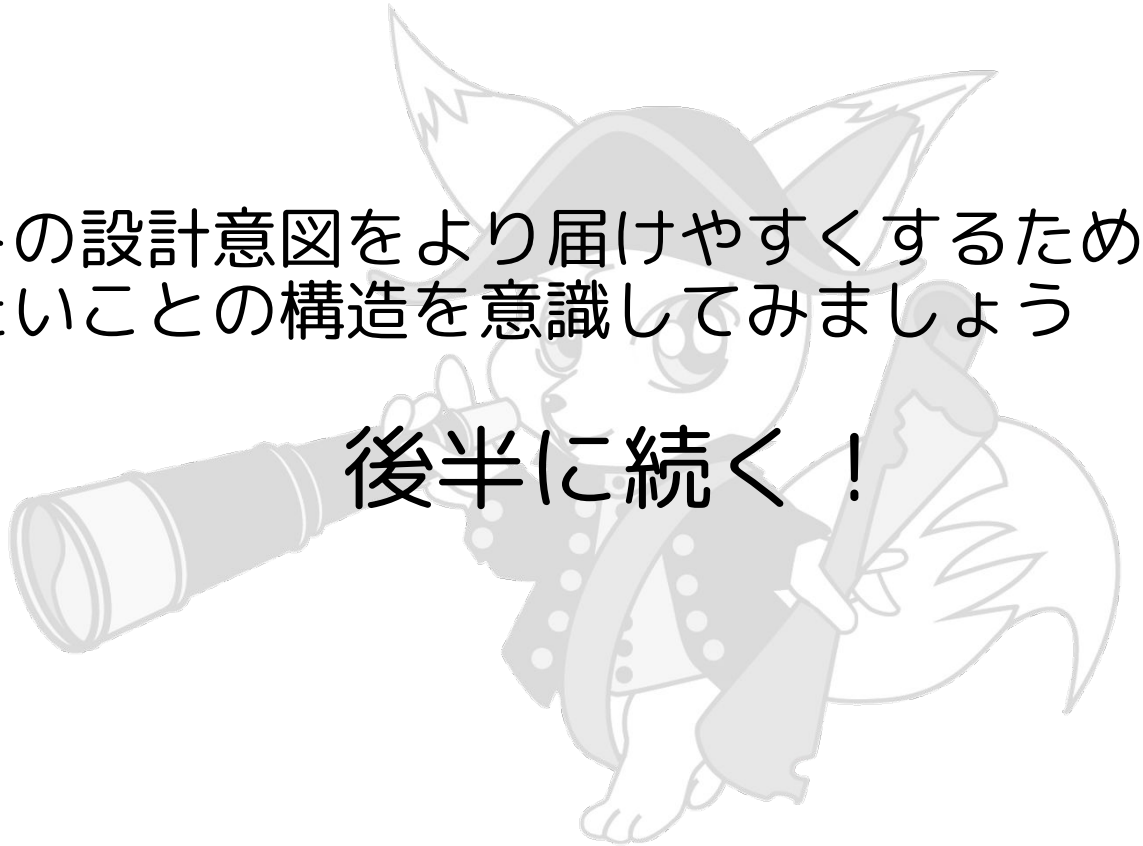
検証のための”問い”を投げかけて、より納得がいく回答にする

- どうして”従来通り”であれば問題がないのだろうか？
- 仕様に書かれていることの確認で十分なのだろうか？
  仕様に書かれたことを確認すると、どんな嬉しいことがあるのだろうか？
- 優先順位に実行して時間切れになったら残りは実行しなくてよいと判断した根拠はなんだろう？
  実行しないとその先どんなことが起こるだろうか？それは共有できているだろうか？
- その技法を使うことで嬉しくなるのだろうか？
  （使わなくても困らないと思ったりはしないか？）
- 指定したパラメーターが適切だと言える根拠は？

ここまでのまとめ：明日から始めてみよう

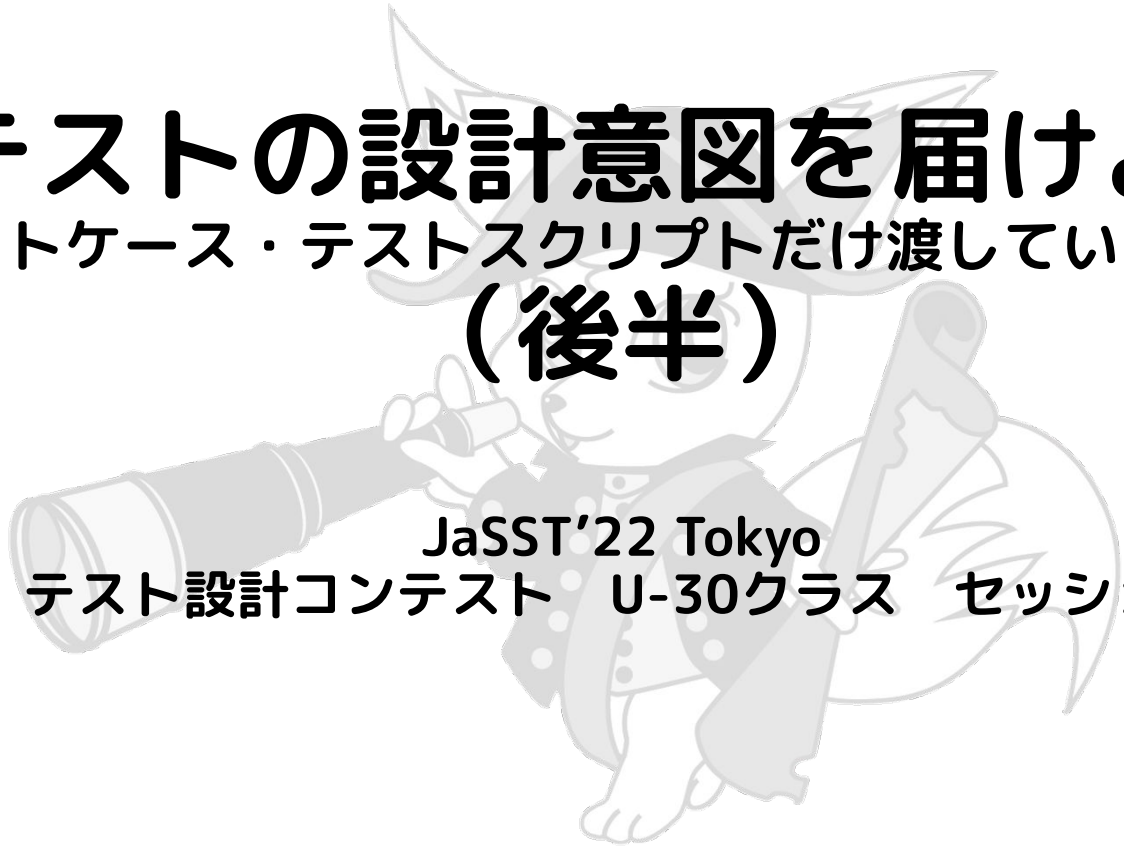
- テストの設計意図を届けるための第一歩
  - まずは**「ちゃんと考えている」**ことに気づこう
  - 成果物を**関係者が理解を深めるたたき台**と考えよう
    （指摘が出てこそ認識共有できる、理解が深まる）
- 過程を見せて情報量の調節をする
  - テストケースやテストスクリプトなどの最終成果物だけでは情報量が多すぎて伝わりにくくなることに気づこう
  - **過程の考えを段階ごとに出力しよう**
- Whyを明確にして納得度を上げる
  - 「根拠はなんだろう？」と問いかけをしていこう
  - **根拠がわかるように情報を繋げてみよう**
  - あえて提示しないと根拠が伝わらないことを意識しよう

テストの設計意図をより届けやすくするために・・・
伝えたいことの構造を意識してみましょう



後半に続く！

テストの設計意図を届けよう
～テストケース・テストスクリプトだけ渡していませんか？
（後半）



JaSST’22 Tokyo
テスト設計コンテスト　U-30クラス　セッション

後半パートでお話すること

短い時間でテストの設計意図を伝えるためのポイント3つ目をお話しする

- テストケースになるまでの過程を見せる
- Whyを明確にする
- **構造を見せる**

その後、

- **テストの設計意図を伝える準備ができたら次にできる小さな工夫**
- **おわりに**

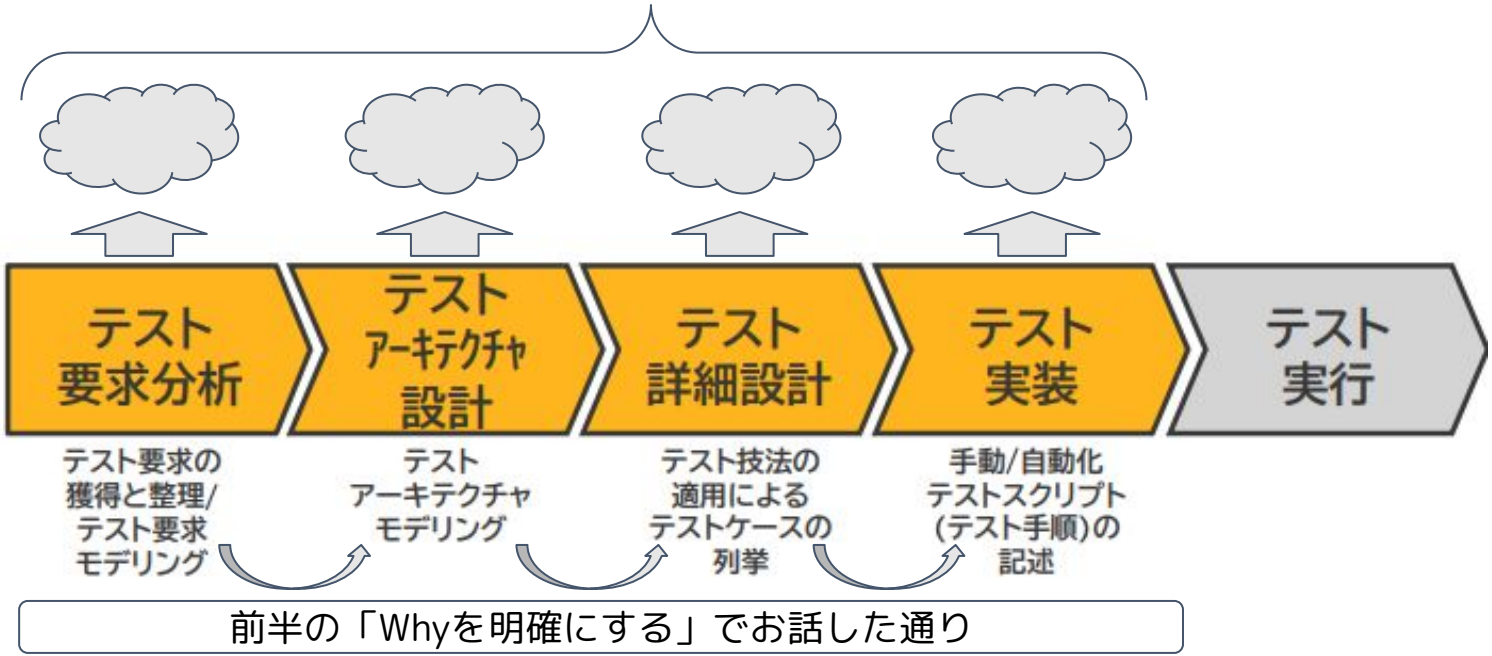
をお話しして、このセッションはおしまい

構造を見せる

本題の前に：何の構造？

個々のプロセスで理解したことや考えたことの構造についての話

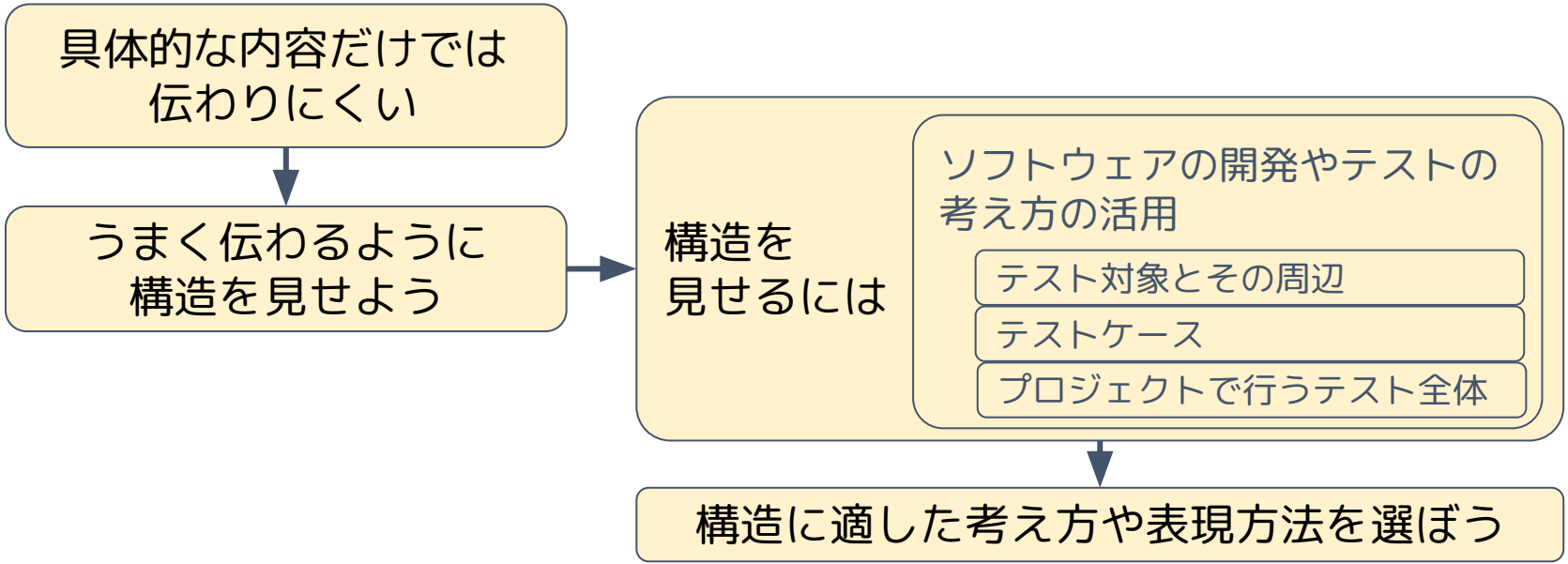
ここからは
こちらに
注目！



前半の「Whyを明確にする」でお話した通り

プロセスの図の引用元：テスト設計チュートリアル ちびこん編 '21 資料　67ページ

「構造を見せる」の構造



TEST DESIGN COMPETITION

テスト設計コンテスト(テスコン)の参加チームの成果物を一部引用している
ひとまとまりの仕様についてテスト設計を検討した結果の一部であり、
現場での実践についてイメージを膨らませる際の参考にしていただきたい

具体的な内容だけでは伝わりにくい：ある日の夕食の場合

具体的なレシピは以下の通り
一目でどんな夕食かイメージするのは難しい

だいこんを切ってアク抜きをする
鶏肉を切る
しょうがをすりおろす
絹さやのヘタとスジをとる
鍋に水を入れて火にかける
ゆでた絹さやを皿に取り出す
鍋に油と鶏肉を入れて炒める
鍋に流れ出た余分な油を捨てる
鍋にだいこんを入れて炒める
水を入れて火が通るまで煮込む
しょうが、醤油、みりん、蜂蜜を入れ煮込む
鶏肉とだいこんを皿に盛り絹さやを散らす
出汁をつくる

鍋に水を入れて火にかける
鍋にほうれん草を入れてゆでる
ゆでたほうれん草を水で冷やす
ゆでたほうれん草の水気をしぼって切る
ボールに出汁と醤油を入れほうれん草を浸す
別の皿にほうれん草を盛る
出汁をつくる
長ねぎを切る
鍋に長ねぎと出汁を入れてゆでる
とうふを切る
鍋にとうふを入れる
具材に火が通ったらみそを入れる
お椀に盛り付ける

具体的な内容だけでは伝わりにくい：ある日の夕食の場合

工夫すると分かりやすい

夕食

主菜

だいこんを切ってアク抜きをする
鶏肉を切る
しょうがをすりおろす
絹さやのヘタとスジをとる
鍋に水を入れて火にかける
ゆでた絹さやを皿に取り出す
鍋に油と鶏肉を入れて炒める
鍋に流れ出た余分な油を捨てる
鍋にだいこんを入れて炒める
水を入れて火が通るまで煮込む
しょうが、醤油、みりん、蜂蜜を入れ煮込む
鶏肉とだいこんを皿に盛り絹さやを散らす

副菜

出汁をつくる
鍋に水を入れて火にかける
鍋にほうれん草を入れてゆでる
ゆでたほうれん草を水で冷やす
ゆでたほうれん草の水気をしぼって切る
ボールに出汁と醤油を入れほうれん草を浸す
別の皿にほうれん草を盛る

汁物

出汁をつくる
長ねぎを切る
鍋に長ねぎと出汁を入れてゆでる
とうふを切る
鍋にとうふを入れる
具材に火が通ったらみそを入れる
お椀に盛り付ける

具体的な内容だけでは伝わりにくい：ある日の夕食の場合

工夫すると分かりやすい

夕食

主菜

だいこんを切ってアク抜きをする
鶏肉を切る
しょうがをすりおろす
絹さやのヘタとスジをとる
鍋に水を入れて火にかける
ゆでた絹さやを皿に取り出す
鍋に油と鶏肉を入れて炒める
鍋に流れ出た余分な油を捨てる
鍋にだいこんを入れて炒める
水を入れて火が通るまで煮込む
しょうが、醤油、みりん、蜂蜜を入れ煮込む
鶏肉とだいこんを皿に盛り絹さやを散らす

副菜

出汁をつくる
鍋に水を入れて火にかける
鍋にほうれん草を入れてゆでる
ゆでたほうれん草を水で冷やす
ゆでたほうれん草の水気をしぼって切る
ボールに出汁と醤油を入れほうれん草を浸す
別の皿にほうれん草を盛る

汁物

出汁をつくる
長ねぎを切る
鍋に長ねぎと出汁を入れてゆでる
とうふを切る
鍋にとうふを入れる
具材に火が通ったらみそを入れる
お椀に盛り付ける

主食(白米等)がない？！

具体的な内容だけでは伝わりにくい

- 理解に時間がかかる
- 過不足がないか分かりにくい

だいこんを切ってアク抜きをする
鶏肉を切る
しょうがをすりおろす
絹さやのヘタとスジをとる
鍋に水を入れて火にかける
ゆでた絹さやを皿に取り出す
鍋に油と鶏肉を入れて炒める
鍋に流れ出た余分な油を捨てる
鍋にだいこんを入れて炒める
水を入れて火が通るまで煮込む
しょうが、醤油、みりん、蜂蜜を入れ煮込む
鶏肉とだいこんを皿に盛り絹さやを散らす
出汁をつくる

鍋に水を入れて火にかける
鍋にほうれん草を入れてゆでる
ゆでたほうれん草を水で冷やす
ゆでたほうれん草の水気をしぼって切る
ボールに出汁と醤油を入れほうれん草を浸す
別の皿にほうれん草を盛る
出汁をつくる
長ねぎを切る
鍋に長ねぎと出汁を入れてゆでる
とうふを切る
鍋にとうふを入れる
具材に火が通ったらみそを入れる
お椀に盛り付ける

具体的な内容だけでは伝わりにくい

テスト設計成果物でも同じ

- 理解に時間がかかる
  - 必要なところをピンポイントで必要なだけ読むのが難しい
- 過不足がないか分かりにくい
  - 要素間の関連を見落とす、過不足を見落としていないか不安になって読み返す

具体的な内容だけの成果物の例

要求ID	仕様ID	ケースID	前提条件	手順	期待結果
pot-310	pot-310-11	310-11	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	沸騰ランプを点灯し、保温ランプを消灯する
pot-310	pot-310-12	310-12-1	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す	操作パネルの温度/モード表示窓にポットの中の水の温度を表示する
pot-310	pot-310-12	310-12-2	沸騰行為中	1.沸騰中に温度表示が上がった瞬間にポットの蓋を開けて温度計で温度を測る	測定温度の小数点以下を四捨五入した温度が操作パネルの温度/モード表示窓に表示されている
pot-310	pot-310-21	310-21	保温行為中 給湯していない	1.沸騰ボタンを(100msec以上)押す 2.沸騰するまで様子を見る	操作パネルの温度/モード表示窓に100と表示される
pot-310	pot-310-31	310-31-1	ポットの蓋が開いている	1.ポットに110℃以上の液体を入れる 2.ポットの蓋を閉める	30秒間ブザーが鳴る

なぜ？　どうすれば良い？

成果物の例(表)は本セッション前半の資料に掲載したものの一部を再掲

うまく伝わるように構造を見せよう

困りごと	なぜ？
理解に時間がかかる	何がどこまでどこに書かれているか分かりにくい
	具体的すぎる情報しかなく取捨選択や変換が必要である
過不足がないか分かりにくい	関係性が表現されていない、分かりにくい

具体的な内容だけの場合に起こりがちな困りごと

うまく伝わるように構造を見せよう

困りごと	なぜ？	どうすれば良い？
理解に時間がかかる	何がどこまでどこに書かれているか分かりにくい	全体像を見せる
	具体的すぎる情報しかなく取捨選択や変換が必要である	構成要素の要点を適切な抽象度で見せる
過不足がないか分かりにくい	関係性が表現されていない、分かりにくい	構成要素の関係性を見せる

うまく伝わるように構造を見せよう

困りごと	なぜ？	どうすれば良い？
理解に時間がかかる	何がどこまでどこに書かれているか分かりにくい	全体像を見せる
	具体的すぎる情報しかなく取捨選択や変換が必要である	構成要素の要点を適切な抽象度で見せる
過不足がないか分かりにくい	関係性が表現されていない、分かりにくい	構成要素の関係性を見せる

3点を同時に実現することを
この講演では
伝えたいことの**構造を見せる**
と表現する

うまく伝わるように構造を見せよう

対象の質がレビューに与える影響

間違い探しレビュー実験結果

入力情報は等価（同一間違い10件混入）

【A:複雑】VS【B:構造化】

【複雑・ランダムな情報羅列】

A 情報

間違い探し レビュー

平均検出率（%）	平均時間（分）
73	8.0

<レビュールール>
各自すべての情報の確認を終了したら経過時間と検出した間違い数を記録する

【構造化・意味ある分類】

B 情報

間違い探し レビュー

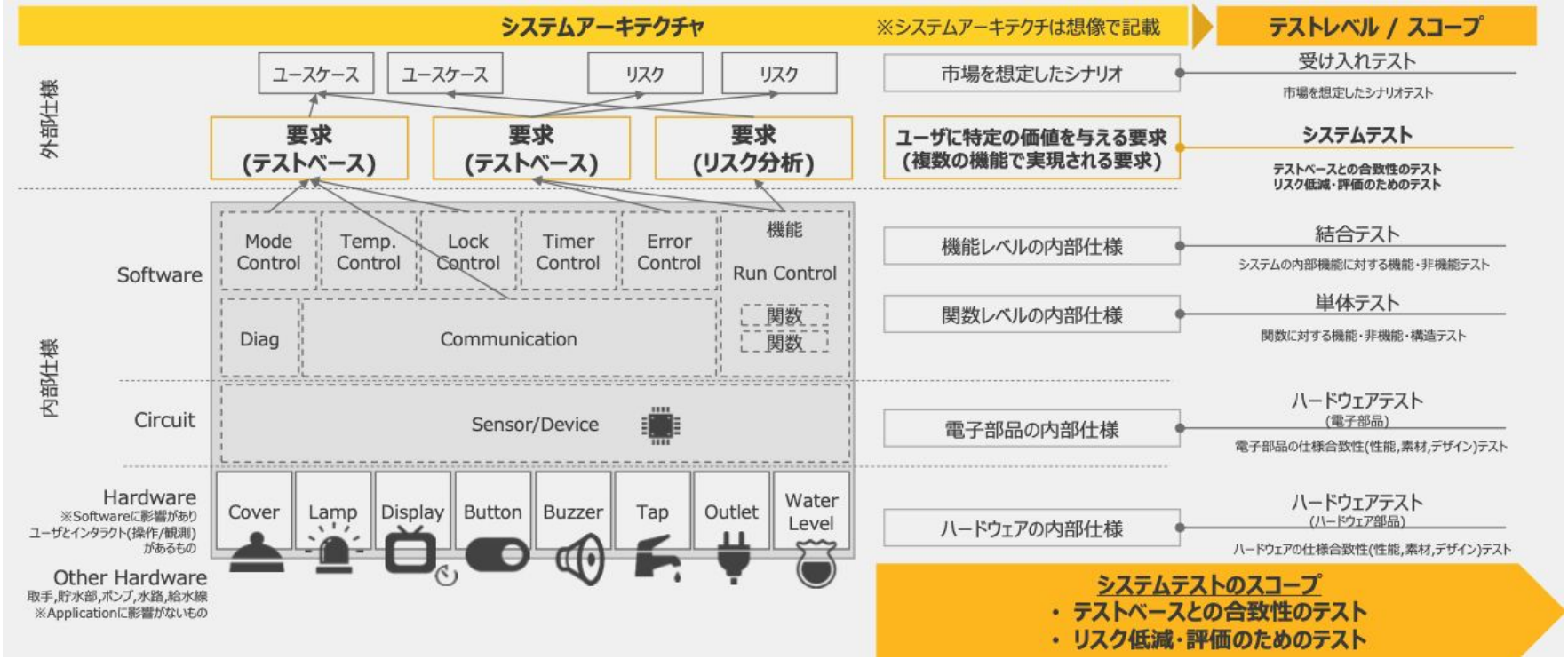
平均検出率（%）	平均時間（分）
91	5.6

【B:構造化】の方が
・「71%の時間」で
・「検出率が18point 高い」

引用元：レビューが教えてくれたこと～for WACATE2021Winter 8ページ

テスコン成果物より：構造を見せている例

「はじめての共同作業」チームの成果物より
成果物作成者がどのようにシステムやテストを捉えているか全体像が理解しやすい

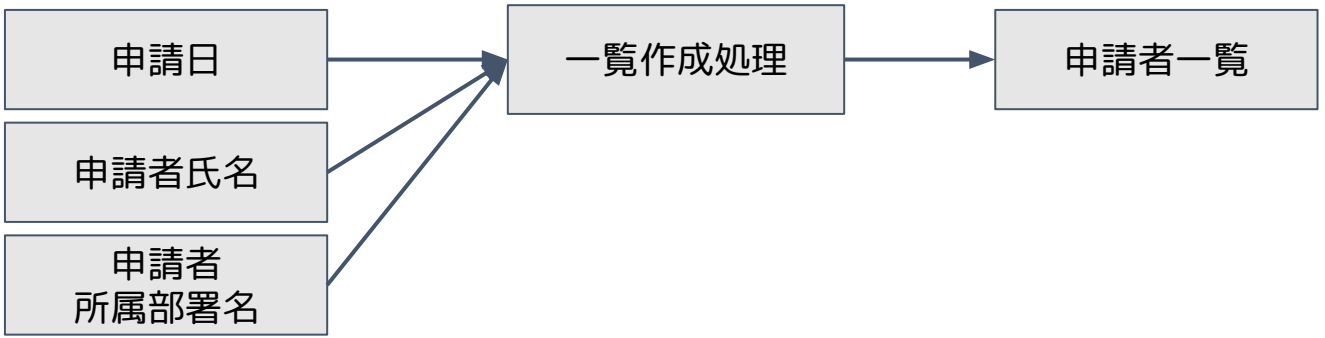


引用元：テスト設計コンテスト'21 はじめての共同作業_成果物0_プロセス全体像.pdf 2ページ

構造を見せるには

仕様について理解したことの構造を見せる場合を考えてみる

簡単な図形による処理と入出力の関係性の表現も、構造を見せている一例である



いきなり構造をうまく見せられず、検討や準備が必要なことがある

- 構造を見せるための着眼点が、複数ありそう
- 全体像や構成要素間の関係性を、分かりやすく表現するのが難しい
- 文書などから抜き出しただけでは、要素の要点が分かりにくい

構造を見せるには

仕様について理解したことの構造を見せるための、検討や準備を含む手順は…

1. どのような着眼点から見た構造を伝えるか、検討して決める
2. インプットになる情報を収集し、必要な要素を抜き出す
   a. インプットは文書やチャット、ヒアリング結果、記憶など、さまざま
   b. 要素は構造を見せたい対象により異なる
3. 要素の要点を抽出したり抽象度を調整したりする
   a. 不要な情報の除外、不足情報の補足、グルーピングや分解で要点を明確にする
   b. 複数の要素に共通する性質をもとに抽象化する、個々の例を考えて具体化する
4. 要素同士の関係性や全体の中での位置付けが分かるように、まとめる
   a. どう表現するかの検討も行う。俯瞰しやすいよう、図表を活用すると良い

着眼点や全体像をどう表現するかなど、ゼロから検討するのは大変
手がかりになるものはないだろうか？

ソフトウェアの開発やテストで使われる考え方を活用しよう

ソフトウェアの開発やテストの考え方の活用

何の構造を見せるかにより使いやすい考え方が異なる
例えば…

- テスト対象とその周辺
  - 状態遷移図と状態遷移表
- テストケース
  - テストフレーム
- プロジェクトで行うテスト全体
  - テストコンテナ

特に、**テスト対象とその周辺**
を詳説する

ソフトウェアの開発やテストの考え方の活用

何の構造を見せるかにより使いやすい考え方が異なる
例えば…

- テスト対象とその周辺
  - 状態遷移図と状態遷移表
- テストケース
  - テストフレーム
- プロジェクトで行うテスト全体
  - テストコンテナ

テスト対象とその周辺の構造を見せる

枠内は、架空の経費申請システムについての文章である
どのような着眼点で構造を見せると良いだろうか？

申請作成者が申請作成を開始すると、作成中という状態になる。申請作成者は作成を終えたら、申請ボタンを押す。申請ボタンの代わりに一時保存ボタンを押すかEscキーを押すと一時保存処理が実行されるが、状態は変わらず作成中のままである。

申請ボタンを押した時点での予算残高に比べて申請額が高額の場合は所属部門部長承認待ちとなる。予算残高に比べて申請額が同額か少額の場合は所属部門の課長承認待ちとなる。所属部門の課長または所属部門の部長が所属部門承認ボタンを押すと、経理部課長であるAさんまたはBさんによる承認待ちとなる。

取引実績がある取引先についての申請では、AさんまたはBさんが経理部承認ボタンを押すと申請が完了状態となる。取引実績がない取引先についての申請でAさんかBさんが経理部承認ボタンを押すと、経理部部長承認待ちとなる。この場合、経理部部長Cさんが経理部部長承認ボタンを押すことで申請が完了状態となる。

各承認者は、申請に不備を見つけた場合、差し戻しボタンを押す。差し戻した場合、通常は1つ前の状態に戻る。ただし、Aさんが差し戻したとき、または、Bさんが差し戻したときは常に申請者宛となるため、作成中となる。

テスト対象とその周辺の構造を見せる

1段落目に状態が複数登場。経費申請システムなので承認待ちなどの状態がありそう
また、ソフトウェア開発では状態遷移図や状態遷移表を使うことがある
状態遷移図と状態遷移表を使って、状態と遷移に着眼して構造を見せることにする

申請作成者が申請作成を開始すると、**作成中という状態**になる。申請作成者は作成を終えたら、申請ボタンを押す。申請ボタンの代わりに一時保存ボタンを押すかEscキーを押すと一時保存処理が実行されるが、**状態は変わらず作成中のまま**である。

申請ボタンを押した時点での予算残高に比べて申請額が高額の場合は所属部門部長承認待ちとなる。予算残高に比べて申請額が同額か少額の場合は所属部門の課長承認待ちとなる。所属部門の課長または所属部門の部長が所属部門承認ボタンを押すと、経理部課長であるAさんまたはBさんによる承認待ちとなる。

取引実績がある取引先についての申請では、AさんまたはBさんが経理部承認ボタンを押すと申請が完了状態となる。取引実績がない取引先についての申請でAさんかBさんが経理部承認ボタンを押すと、経理部部長承認待ちとなる。この場合、経理部部長Cさんが経理部部長承認ボタンを押すことで申請が完了状態となる。

各承認者は、申請に不備を見つけた場合、差し戻しボタンを押す。差し戻した場合、通常は1つ前の状態に戻る。ただし、Aさんが差し戻したとき、または、Bさんが差し戻したときは常に申請者宛となるため、作成中となる。

テスト対象とその周辺の構造を見せる

必要な要素を抜き出す際、状態遷移図と状態遷移表を手がかりにできる
ここでは、状態／イベント／遷移の条件、を必要な要素と考えた
それぞれに関係する部分を、オレンジ／緑／青で色分けした

申請作成者が申請作成を開始すると、作成中という状態になる。申請作成者は作成を終えたら、申請ボタンを押す。申請ボタンの代わりに一時保存ボタンを押すかEscキーを押すと一時保存処理が実行されるが、状態は変わらず作成中のままである。

申請ボタンを押した時点での予算残高に比べて申請額が高額の場合は所属部門部長承認待ちとなる。予算残高に比べて申請額が同額か少額の場合は所属部門の課長承認待ちとなる。所属部門の課長または所属部門の部長が所属部門承認ボタンを押すと、経理部課長であるAさんまたはBさんによる承認待ちとなる。

取引実績がある取引先についての申請では、AさんまたはBさんが経理部承認ボタンを押すと申請が完了状態となる。取引実績がない取引先についての申請でAさんかBさんが経理部承認ボタンを押すと、経理部部長承認待ちとなる。この場合、経理部部長Cさんが経理部部長承認ボタンを押すことで申請が完了状態となる。

各承認者は、申請に不備を見つけた場合、差し戻しボタンを押す。差し戻した場合、通常は1つ前の状態に戻る。ただし、Aさんが差し戻したとき、または、Bさんが差し戻したときは常に申請者宛となるため、作成中となる。

テスト対象とその周辺の構造を見せる

見つけた要素の要点を抽出したり抽象度を調整したりする
大事なことは、**何に着眼して伝えたいか意識して考える**こと

今回の例では、伝えたいのは状態とその遷移

↓

具体的な個人名の情報などは不要なので、除外する

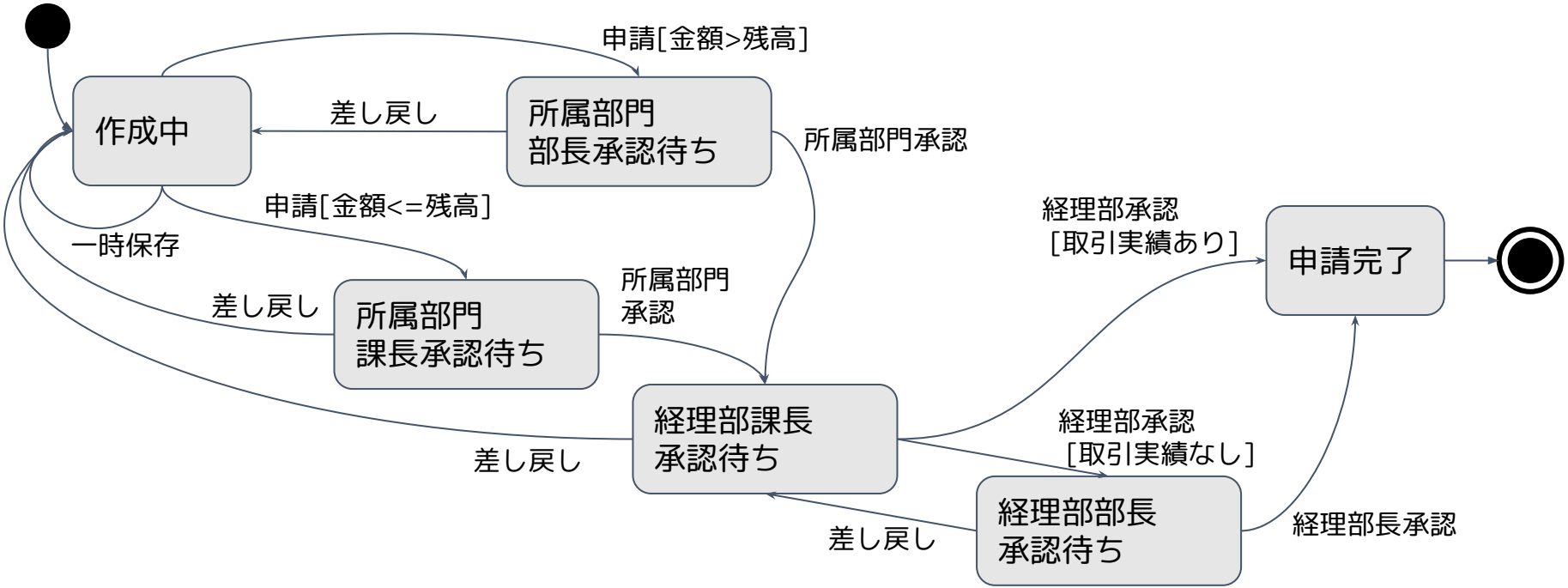
「経理部課長であるAさんまたはBさんによる承認待ち」

Aさんによる承認待ち
Bさんによる承認待ち
→ 経理部課長による承認待ち

テスト対象とその周辺の構造を見せる

状態遷移図で、整理した要素の関係性や全体の中での位置付けを表現した

状態の数や複雑さが分かりやすくなった



テスト対象とその周辺の構造を見せる

更に状態遷移表で表現すると、状態とイベントの関係を俯瞰しやすくなる

状態 / イベント	作成中	所属先部門 課長承認待ち	所属部門 部長承認待ち	経理部課長承 認待ち	経理部部長 承認待ち	申請完了
申請 [金額<=残高]	所属部門 課長承認待ち					
申請 [金額>残高]	所属部門 部長承認待ち					
一時保存	作成中					
所属部門承認		経理部 課長承認待ち	経理部 課長承認待ち			

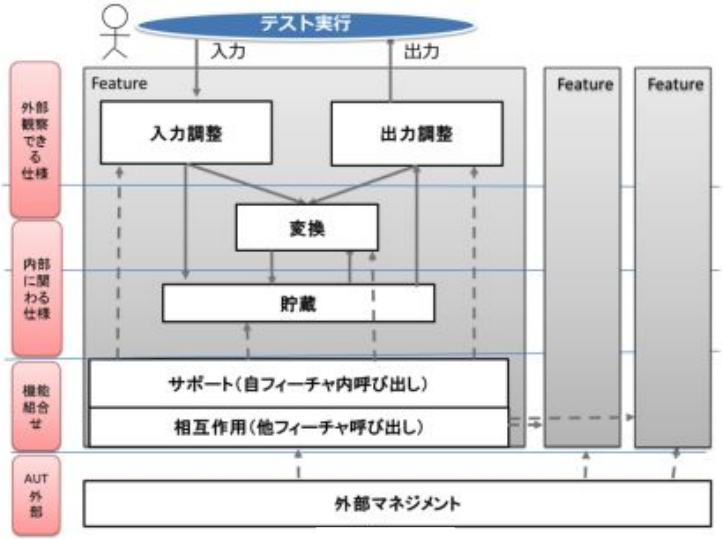
仕様について理解したことの構成要素とその関係性、全体像が
分かりやすくなった

テスト対象とその周辺の構造を見せる

ソフトウェアやその使われ方などの構造を見せる。例えば…

- IPO（入力、処理、出力）を整理する
- テストで使われる考え方を活用する
  - ゆもつよメソッドの論理的機能構造

    『大村朔平さんの書籍「一般システムの現象学」に書かれていたものを、湯本さんがテスト分析のモデルとして改変したものである。』
    『論理的機能構造はテスト対象システムを入力調整、出力調整、変換、貯蔵といった要素で分解したものである。』
- ソフトウェアの要件定義、設計で使われる考え方を活用する
  - ユースケース図とユースケース記述



仕様の理解やテスト内容についての考えを見せたいときに活用できる
複数の着眼点で構造を表現できないか、検討すると良い

『』内の引用元：　JaSST'21 Tohoku　ゆもつよメソッドワークショップ　仕様書読みとモデル図作成　8ページ
図の引用元：JaSST'15 Tokyo「解決！テストアーキテクチャ設計」講演資料：「湯本からのコメント」4ページ

テスコン成果物より：デシジョンテーブル、CFDの例

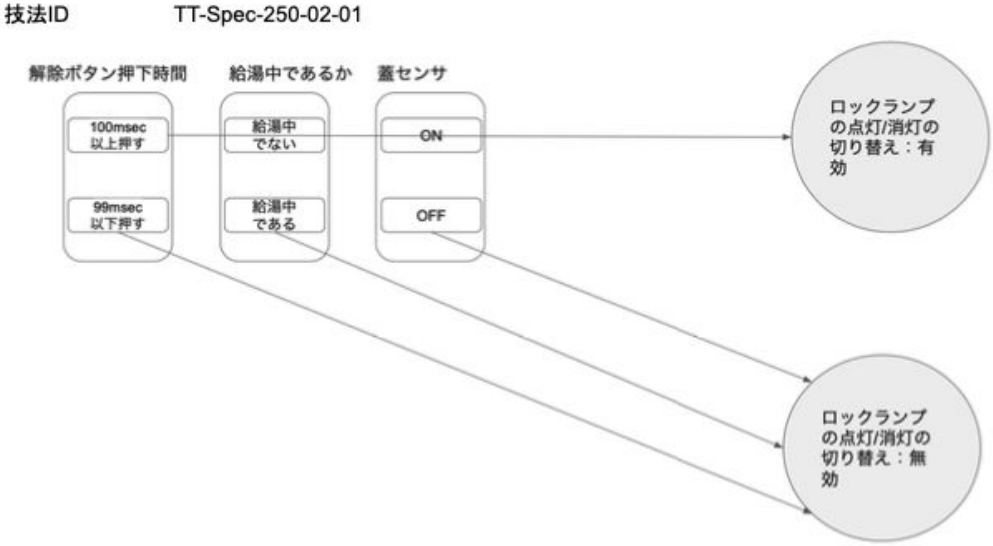
「まちがいさがし。」チームの成果物より

テスト対象のどこに着目されたのか、分かりやすい

技法ID　TT-Spec-250-02-02

			1	2	3	4
条件	解除ボタン押下時間	100msec 以上押す	Y	N	Y	Y
	給湯中であるか	給湯中でない	Y	Y	N	Y
	蓋センサ	ON	Y	Y	Y	N
動作	ロックランプ切り替え有効		X			
	ロックランプ切り替え無効			X	X	X

技法ID　TT-Spec-250-02-01



引用元：テスト設計コンテスト'21 まちがいさがし。_成果物2_004_テスト技法適用書.pdf 1ページ

ソフトウェアの開発やテストの考え方の活用

何の構造を見せるかにより使いやすい考え方が異なる
例えば…

- テスト対象とその周辺
  - 状態遷移図と状態遷移表
- テストケース
  - テストフレーム
- プロジェクトで行うテスト全体
  - テストコンテナ

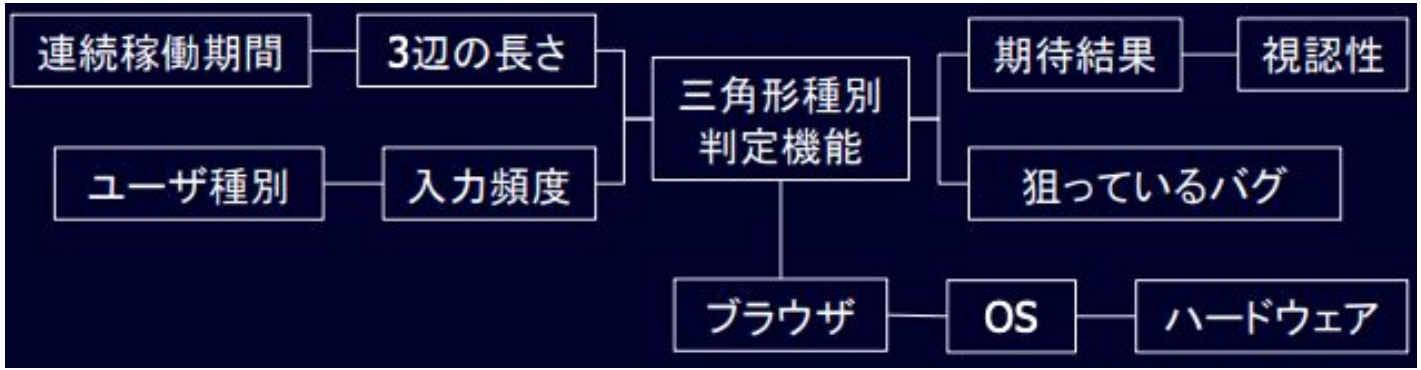
この講演でのテストフレームとテストコンテナの説明における「テスト観点」は、
『何をテストするのか（テストケースの意図）のみを端的に記述したもの』をさす
例えば、OS、ユーザー種別

『』内の引用元：テスト設計チュートリアル テスコン編 '21 資料 7ページ

テストケースの構造を見せる

テストフレームを考える方法がある

『テストケースの構造をテスト観点で表したものを「テストフレーム」と呼ぶ』



テスト観点の組み合わせが視覚化される

↓

テストケースに必要なテスト観点を議論しやすくなる

必要なテスト観点の組み合わせはある？　組み合わせる必要がないテスト観点はない？

図と『』内の引用元：テスト設計チュートリアル テスコン編 '21 資料　48ページ

プロジェクトで行うテスト全体の構造を見せる

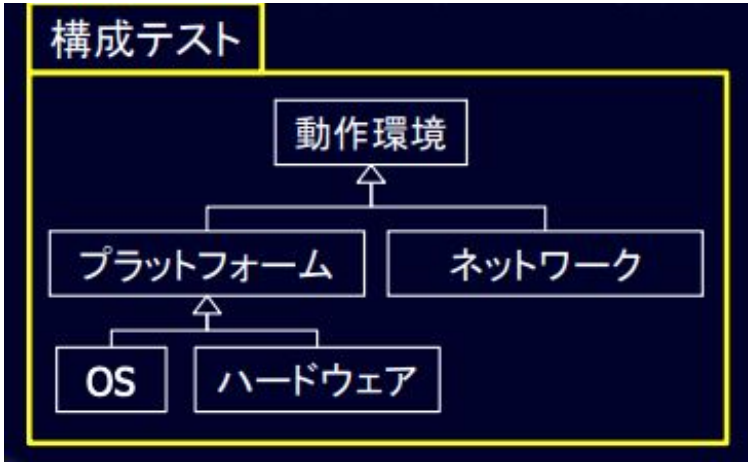
テストコンテナを考える方法がある

- 複数のテスト観点をグルーピングしたものをテストコンテナと呼ぶ
  - テストタイプやテストレベル、テストサイクルなどを表現する

上記箇条書きそれぞれの引用元：テスト設計チュートリアル テスコン編 '21 資料　23ページ

各テスト観点をどのテストタイプ、テストレベルなどで扱うか分かりやすくなる

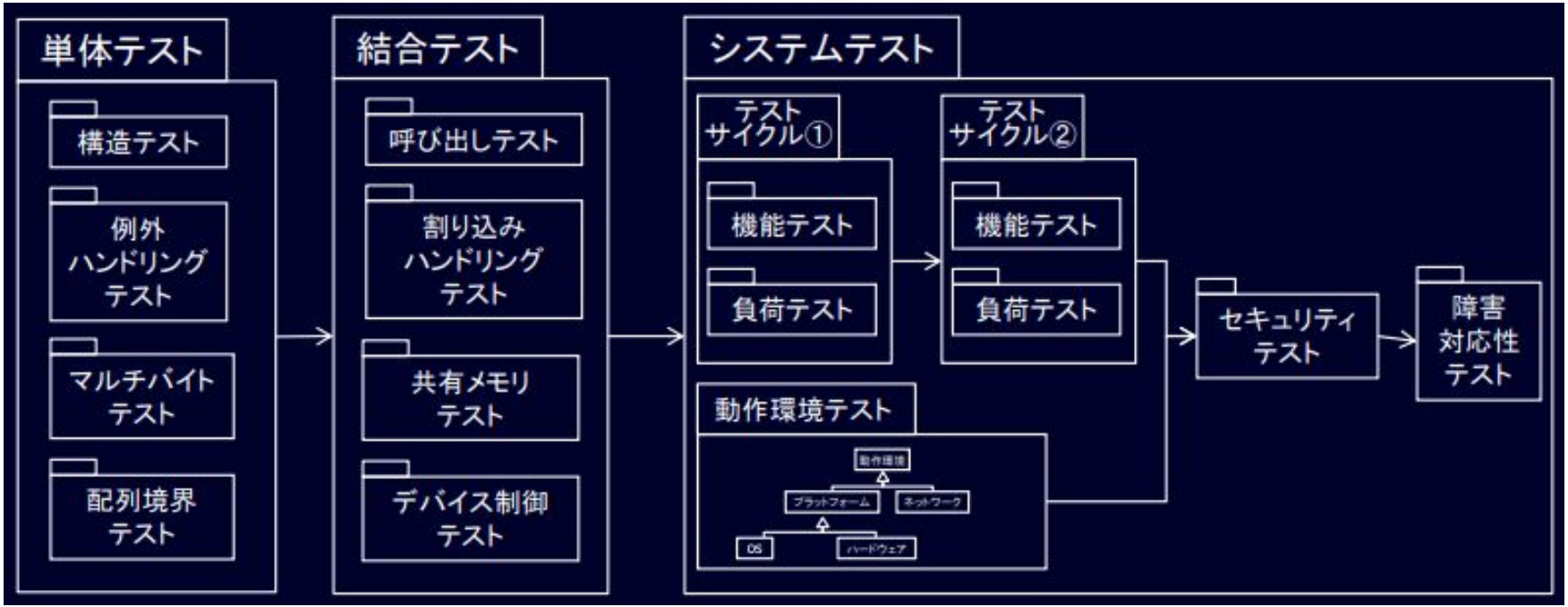
テストタイプ、テストレベルなどの、それぞれの役割が分かりやすくなる



図の引用元：
テスト設計チュートリアル
テスコン編 '21 資料　23ページ

プロジェクトで行うテスト全体の構造を見せる

テストコンテナのモデルを活用すると、全体像を分かりやすく見せられる



図の引用元：テスト設計チュートリアル テスコン編 '21 資料　24ページ

ソフトウェアの開発やテストの考え方の活用

何の構造を見せるかにより使いやすい考え方が異なる
例えば…

- テスト対象とその周辺
  - 状態遷移図と状態遷移表
- テストケース
  - テストフレーム
- プロジェクトで行うテスト全体
  - テストコンテナ

ご紹介した以外にも考え方はあり、独自の考え方が使われることもある

ソフトウェア開発以外でも使われる一般的な表現方法が役立つこともある

構造をうまく伝えるには、適切な考え方や表現方法を選ぶ必要がある

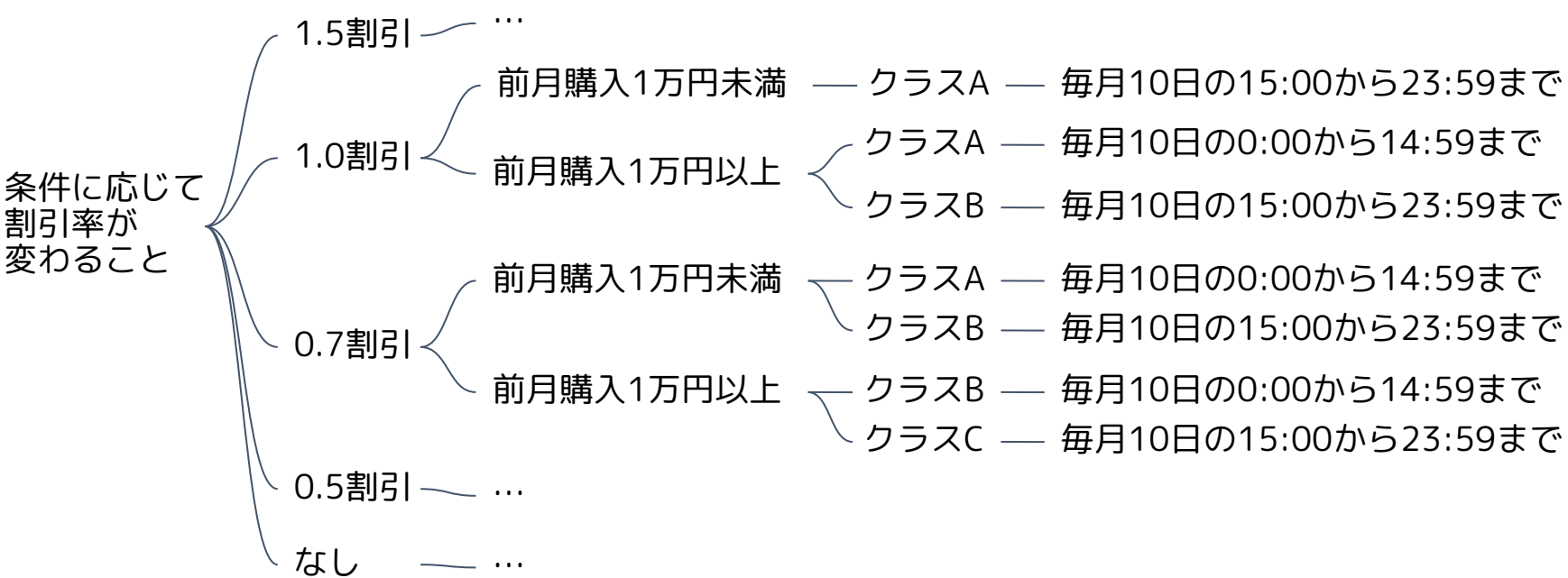
構造に適した考え方や表現方法を選ぼう

枠内は架空のECサイトの割引についての文章だが、分かりにくい

前月購入1万円以上のクラスAの会員が毎月10日の0:00から14:59までに購入した場合は1.0割引
前月購入1万円未満のクラスAの会員が毎月10日の0:00から14:59までに購入した場合は0.7割引
前月購入1万円以上のクラスBの会員が毎月10日の0:00から14:59までに購入した場合は0.7割引
前月購入1万円未満のクラスBの会員が毎月10日の0:00から14:59までに購入した場合は0.5割引
前月購入1万円以上のクラスCの会員が毎月10日の0:00から14:59までに購入した場合は0.5割引
前月購入1万円未満のクラスCの会員が毎月10日の0:00から14:59までに購入した場合は割引なし
前月購入1万円以上のクラスAの会員が毎月10日の15:00から23:59までに購入した場合は1.5割引
前月購入1万円未満のクラスAの会員が毎月10日の15:00から23:59までに購入した場合は1.0割引
前月購入1万円以上のクラスBの会員が毎月10日の15:00から23:59までに購入した場合は1.0割引
前月購入1万円未満のクラスBの会員が毎月10日の15:00から23:59までに購入した場合は0.7割引
前月購入1万円以上のクラスCの会員が毎月10日の15:00から23:59までに購入した場合は0.7割引
前月購入1万円未満のクラスCの会員が毎月10日の15:00から23:59までに購入した場合は0.5割引
前月購入1万円以上の会員が毎月20日に購入した場合は0.5割引
前月購入1万円未満の会員が毎月20日に購入した場合は割引なし
毎月15日と最終日以外の日の購入の場合は割引なし

構造に適した考え方や表現方法を選ぼう

仕様の約半分を表現してみた
条件の全体像や条件同士の組み合わせの規模は分かりやすいだろうか？



構造に適した考え方や表現方法を選ぼう

こちらの方が、条件の全体像や条件同士の組み合わせの規模を確認しやすい

				1	2	3	4	5	6	7	8	9
条件	購入日時	毎月10日	0:00-14:59	Y	Y	Y	Y	Y	Y	N	N	N
			15:00-23:59	N	N	N	N	N	N	Y	Y	Y
		毎月20日	終日	N	N	N	N	N	N	N	N	N
		上記以外の日	終日	N	N	N	N	N	N	N	N	N
	会員クラス			A	A	B	B	C	C	A	A	B
	前月購入金額	前月購入1万円以上である		Y	N	Y	N	Y	N	Y	N	Y
動作	割引	1.5割		-	-	-	-	-	-	X	-	-
		1.0割		X	-	-	-	-	-	-	X	X
		0.7割		-	X	X	-	-	-	-	-	-
		0.5割		-	-	-	X	X	-	-	-	-
		なし		-	-	-	-	-	X	-	-	-

構造に適した考え方や表現方法を選ぼう

適さない場面では、伝えたいことをうまく表現できないことがある

- 意図や全体像が分かりにくい
  - 具体的な内容だけの場合と同様に、何が書かれているか全部読むまで分からない
- 要素間の関係性が表現しきれない
  - 包含関係、順序関係がある場合など関係性が読み取りにくくなることがある

うまく伝えるため、**構造に適した考え方や表現方法を選ぶ**
そのためには…

考え方や表現方法を
幅広く学び選択肢を増やす

テストだけでなく
ソフトウェアの要件定義や設計の考え方、
各種図表などの表現方法

特に伝えたい要素や
関係性を明らかにする

一言で言うと何か、自問自答してみる
➢ 一番、伝えたいことは？
➢ 成果物の目的から見て伝えるべきは？

ここまでのまとめ

具体的な内容だけでは、理解に時間がかかる、過不足がないか分かりにくい
改善するには…

- 理解したことや考えたことの構造を見せる
  - ソフトウェアの開発やテストで使われる考え方を活用すると良い
    - 適した考え方や表現方法を選ぶことが大事

3つのポイントをおさえて、短い時間でテストの設計意図を伝えよう

- **テストケースになるまでの過程を見せる**
- **Whyを明確にする**
- **構造を見せる**

テスコンU-30クラスの審査でもここまでの内容と同じ話が…

過去の審査のフィードバックより

- ただし、**この成果物の意図（Why）を示さないとその可否も判断しがたい**ので、ぜひWhyやHowについても記述するようにしてください。
- テストアーキテクチャとして”テストコンテナ評価モデル”にて**テスト全体の構造を示していますが、（略）ざっくりとしたものとなっており「どんなテストをされるのか？」が読み取れません。**

審査基準より

- テストケースの構造が適切に表現されているか
- テスト要求分析・テストアーキテクチャ設計とテスト詳細設計・実装の間のトレーサビリティが取れているか

TEST DESIGN COMPETITION

審査基準はテスト設計コンテストの公式サイト(U-30クラス)で公開中
http://www.aster.or.jp/testcontest/u30.html

なお、テスト設計コンテスト以外でも参考にできる審査基準もある。例えば…

- テストスコープが明示されているか、明示されたスコープは妥当性があるか
- リスクを考慮した優先順位付けが行われているか

審査コメント(先頭が四角の箇条書き)の引用元：テスト設計コンテスト’21 審査員コメント　但し太字は本資料作成者による
審査基準(先頭が黒丸の箇条書き)の引用元：テスト設計コンテスト　U-30クラス

テストの設計意図を伝える準備ができたら
次にできる小さな工夫

テストの設計意図を伝える準備ができたら、次は？

意図を伝える先に目指すのは、より良いソフトウェアやサービスづくり
そのために、狙った通りにテストができるテスト設計成果物をつくりたい
フィードバックを得て、理解を促進し成果物を磨き実現に近づける

前半パートでも似たような話があった

1. 成果物を誰かに見せてフィードバックをもらおう
2. 無反応の理由を探ろう
3. 短い時間で設計の意図を伝えられるようにしよう

ここからは
フィードバックを
よりうまく得る
ための話

これはお話し済

フィードバックを、よりうまく得るための小さな工夫を２つ紹介する

- セルフレビューを始めよう、工夫しよう
- フィードバックを得る目的を明らかにしよう

箇条書き1～3は本セッション前半の資料に掲載したものの一部を再掲

セルフレビューを始めよう、工夫しよう

読んだ人がフィードバックしやすい状態にしておく
自身で確認して、問題が見つかったら改善しておこう

- テストの設計意図など、伝えたいことが伝わりやすく書かれているか？
- 伝えたいことに集中して読めるか？
  - 誤字脱字はないか？　成果物の中や成果物同士に不整合はないか？

作成者自身が問題を見つけるのは難しい。やり方を工夫することも大事

- 校正ツールを使う
- 声に出して読む
  - 読んでいて書いていないことを補足したら、成果物への記載を検討する
- 時間をあけて読む
  - 「ドキュメントの字面通りに解釈して問題がないかどうか、用語やコード体系などの一貫性が保たれているかどうか、誤りがないかどうか――などは、書いた直後にセルフチェックをしても見つけるのは難しいものです。ドキュメントを書いたときの記憶が鮮明に残っていて、自分の頭のなかで勝手に内容を補足してしまうからです。最低でも1日は空けてチェックしてください。」
    - 引用元：森崎修司. なぜ重大な問題を見逃すのか？間違いだらけの設計レビュー改訂版（日経BP Next ICT選書）(Japanese Edition) (Kindle の位置No.1167-1172). Kindle 版.

フィードバックを得る目的を明らかにしよう

目的が変わると、誰が適任も変わる
適任者に読んでもらうと、フィードバックを得られる可能性が高まる

例えば…

ソフトウェア要求から見て十分なテストになっているか知りたい → ソフトウェア要求の内容や経緯を詳しく知っている人

ソフトウェアの設計上の不安に対応したテストになっているか確認したい → ソフトウェア設計者

目的を具体的に伝えられると、より適切に読んでもらえる可能性が高まる

目的をできるだけ具体的にするために、自問自答してみる

- どのようなフィードバックを得られれば、成功か？
- フィードバックを得た後で、成果物や自分がどのようになると良いか？

フィードバックをうまく得るための小さな工夫のまとめ

- セルフレビューを始めよう、工夫しよう
  - **フィードバックを得やすくなるように、**
    誰かに読んでもらう前に自身で確認して事前に改善しておこう
- フィードバックを得る目的を明らかにしよう
  - **目的を明らかにして、**
    適切な相手から、より確実にフィードバックを得よう

おわりに

自分の「分かった」と他人の「分かった」は意外と異なる

「分かる」ということについて、派生開発などで有名な清水吉男さんの言葉をご紹介したい

『実は、「分かる」という言葉の裏には何人も越えることの出来ない限界が隠れています。それは、
　　「分かる」とはその人が分かったと思う範囲でのみ分かったのである
という問題です。』
『簡単な例としては、“プログラミングの前に設計書を書いてください”と依頼したとき、受け手にとっては、この依頼の中に解釈不能な言葉は一つも含まれていないことから、おそらく「分かりました」と即答するでしょう。
でもこのとき、依頼者のイメージしている「設計書」と「受け手」のイメージした「設計書」は、表紙の「○○設計書」は一致しているでしょうが、その内容や構成までも一致しているとは限りません。それどころか、殆どの場合ずれているのです。』

成果物を作った人と読む人が同じように分かり、知恵を出しあって、
良いものづくりをする足掛かりとして、テストの設計意図を届けよう

- テストケースになるまでの過程を見せる
- Whyを明確にする
- 構造を見せる

『』内の引用元：Software Manager の為の講座　「分かる」の問題について
但し本資料作成者により文字修飾は割愛した

完璧でなくても実践を

少しでも、自分だけでできるところからでも、ぜひ実践を！
理解が深まったり疑問が出たり、更に学びやすくなります

実践や学びの場としてテスト設計コンテストの参加や予選・決勝戦の
見学を検討していただけたら、U-30審査員一同とても嬉しいです
情報はテスト設計コンテストの公式サイトからどうぞ
http://www.aster.or.jp/testcontest/index.html

テストの設計意図を届けて、
チーム全体やステークホルダー同士で力を合わせて、
より良いソフトウェアやサービスをつくりましょう！

参考文献と引用文献

参考文献と引用文献：前半後半共通、後半

いずれも、本書作成時点の情報である

前半後半共通

- 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
- 話題沸騰ポット要求仕様書 (GOMA-1015型) 第７版
  - 以下からダウンロード可能である
    https://www.sessame.jp/workinggroup/WorkingGroup2/POT_Specification.htm

後半

- テスト設計コンテスト'21 各チームの成果物
- レビューが教えてくれたこと～for WACATE2021Winter
  - https://speakerdeck.com/kitanosirokuma/rebiyugajiao-etekuretakoto-for-wacate2021winter
- @IT 「問題解決力」を高める思考スキル（4） 構造化：全体像を見極め、構成要素を整理する
  - https://atmarkit.itmedia.co.jp/ait/articles/0311/29/news004.html
- 英語版Wikipedia IPO model
  - https://en.wikipedia.org/wiki/IPO_model
- @IT 明日から使えるシステム開発プロジェクトの進め方 再入門（4） 基本設計をスムーズに進めるための5つのポイント――要件定義書の読み合わせ、データ構造、IPO、外部接続 「設計のゴールは各機能でのIPO（INPUT→PROCESS→OUTPUT）の決定」
  - https://atmarkit.itmedia.co.jp/ait/articles/1510/27/news012_3.html
- 猫の抽象化
  - https://qiita.com/akiyama924/items/d26545a30b94734d6ddc

参考文献と引用文献：後半

- 秋山 浩一 (著), ソフトウェアテスト技法ドリル―テスト設計の考え方と実際, 日科技連出版社,2010年
- 梅津 正洋 (著), 竹内 亜未 (著), 伊藤 由貴 (著), 浦山 さつき (著), 佐々木 千絵美 (著), 高橋 理 (著), 武田 春恵 (著), 根本 紀之 (著), 藤沢 耕助 (著), 真鍋 俊之 (著), 山岡 悠 (著), 吉田 直史 (著), ソフトウェアテスト技法練習帳 ~知識を経験に変える40問~, 技術評論社, 2020年
- (2021年版) ソフトウェアテストの上流設計-4 テスト分析と論理的機能構造
  - https://note.com/yumotsuyo/n/nccd9f9600564
- JaSST'21 Tohoku　ゆもつよメソッドワークショップ　仕様書読みとモデル図作成
  - http://www.jasst.jp/symposium/jasst21tohoku/pdf/S3-1-3.pdf
- JaSST'15 Tokyo「解決！テストアーキテクチャ設計」 講演資料：「湯本からのコメント」
  - http://jasst.jp/symposium/jasst15tokyo/pdf/B2-3.pdf
- テスト設計チュートリアル テスコン編 '21 資料
  - http://www.aster.or.jp/business/contest/doc/2021_tescon_V1.0.0.pdf
- テスト設計コンテスト　U-30クラス
  - http://www.aster.or.jp/testcontest/u30.html
- テスト設計コンテスト'21 審査員コメント
- 森崎修司. なぜ重大な問題を見逃すのか？間違いだらけの設計レビュー改訂版（日経BP Next ICT選書）(Japanese Edition) . Kindle 版.
- Software Manager の為の講座　「分かる」の問題について
  - https://affordd.jp/koha_hp/LectureManager/MG.wakaru.html
  - 以下より『Software Manager の為の講座』をクリックし、
    更に『「分かる」の問題について』をクリックでも参照できる
    https://affordd.jp/koha_hp/