JaSST'21 Kyushu

1番役に立った定番のソフトウェアテスト
（お題： 仕事観についての話）
腐ったオジサンの屍を乗り越えていけ（俺屍）

歳をとってからしてはいけないのは、
『説教と昔話と自慢話』 by 高田純次さん

2021.12.18(土)

NPO ASTER 秋山浩一

1. ゴールと道筋
2. 初級者(0～3年)の時に知っておきたかったこと
3. 中級者(4～9年)の時に知っておきたかったこと
4. 上級者(10年～)に知っておいてほしいこと
5. おすすめのテストの作り方

今日お伝えする内容は、すべて私が実際に使って役に立ったものです。誰でも無駄な回り道をせずに、最短で私を超えるテストエンジニアになることができます。

ゴールとゴールに至る道筋のイメージが大切です。

ゴールと道筋

ゴール：　状態
（とりあえず以下を達成する）

- ◆ 初級者：JaSST nanoで発表で卒業
- ◆ 中級者：海外で発表で卒業
- ◆ 上級者：無し
  - 六代目　尾上菊五郎（辞世の句）
    まだ足りぬ、踊りおどりて、あの世まで
  - 極めるとその先が見えてしまう

ゴールと道筋が見えると
何をすべきか分かる
（上級者にならなくてもいいと思う）

上級者
- 自分独自の道
- 指導

中級者
- 何かの既存技術を極める
- 何かのリーダーになる

初心者
- JSTQB FL合格
- 社内で発表する

結論: 1番役に立った定番のソフトウェアテスト

層別: 分ける

- いつでもすぐに使える道具にする
- 錆びていてはダメ

(とりあえず実現すること)

◆ テストの実務
- 同値分割法(同じものを集めることができる)
- 状態を分ける、環境を分ける、タイミングを分ける
- デシジョンテーブルも無限の条件組合せを分けている

◆ マネジメント
- 「それはそれ、これはこれ」
  - 好き嫌いと、評価は分ける
  - やっていないことと、問題は分ける
    - 不安になるのはわかるけど……
  - なくなったら明確に困るもの以外は、分けて捨てる

初級者(0～3年)の時に知っておきたかったこと(基本)

初級者は、この2つを守っていればOK

- 100点を目指さない
- 正しいことだけを蓄積する
  - ジャンルごとに信用する人を決める
  - 規格は割と信じてよい
  - 本とウェブは玉石混交

ゆっくりで大丈夫
間違ったことを覚えると
直すのは案外大変

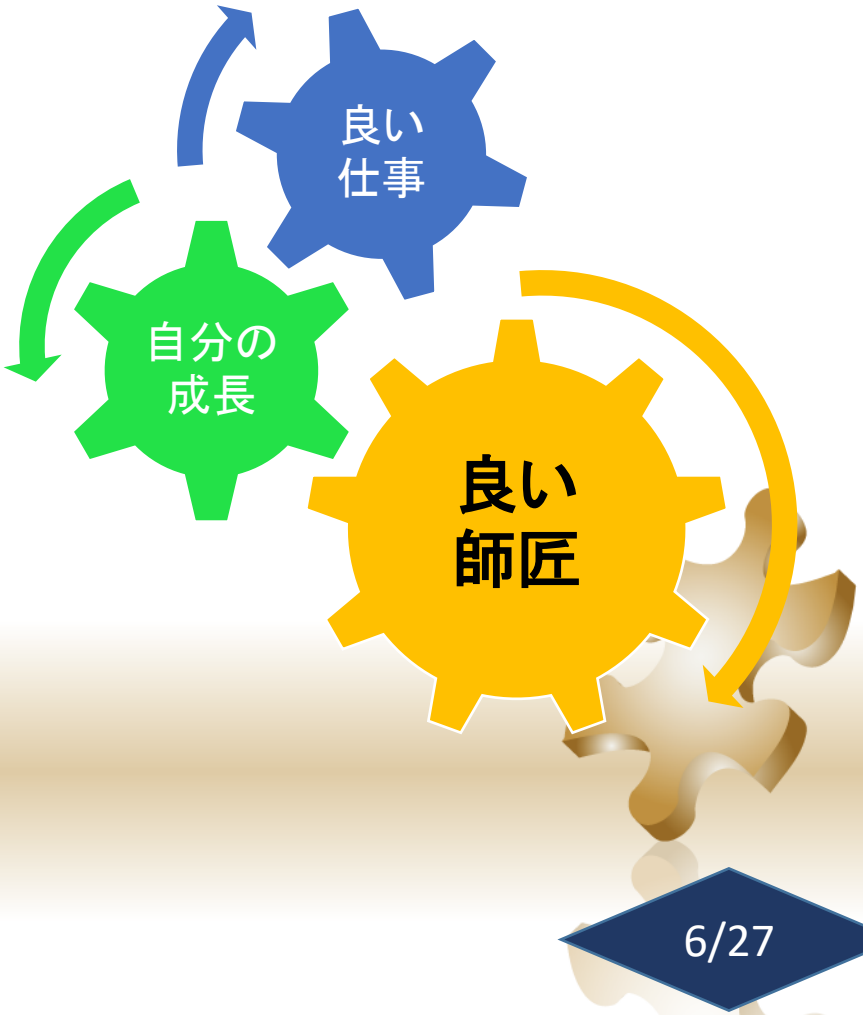
100点や
正解を
目指す

委縮する

初級者の時に知っておきたかったこと(＋α)

仕事は面白いのが当然(だって1/3日！)

- やる気はやると出てくる
- RPG(2つ無ければ転職)
  - ribbon：ほめられる
  - puzzle：挑戦し甲斐がある仕事
  - gold：報酬が高い



「面白い仕事に出会う準備をしている」というように自分をだまさない

初級者の時に知っておきたかったこと(気を付けろ！)

時間の浪費に気を付けろ！

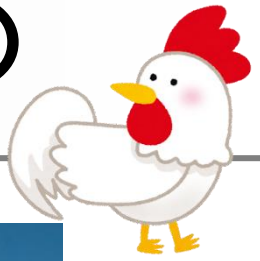
上司　自分

- ◆ トートロジー(犬が西向きゃ尾は東)
  - 残業多い→生産性を上げろ（同じことの言い換え）
- ◆ 品質を一人で上げる
  - テストで品質は上がるが、テストだけに頼らない

全員で品質に取り組む
特に「設計」品質
頭を使う箇所で差が出る

みんなで品質を上げるメカニズムを考える

中級者(4～9年)の時に知っておきたかったこと(基本)



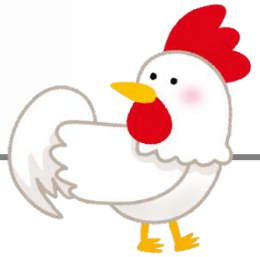
知識を増やす

- 合意形成(D-Case)
- サーバントリーダーシップ
- システム思考
- GQM+Stretegy
- 統計と論理的思考
- レジリエンス(安全・信頼)
- CCPMと見積もり
- その他(発想法、効率化)

安心： 何かを積み上げる

信頼： 監視，予見，対処，学習

中級者の時に知っておきたかったこと(＋α)



本業(テスト)以外の柱を1つ持つ (以下はおすすめ)

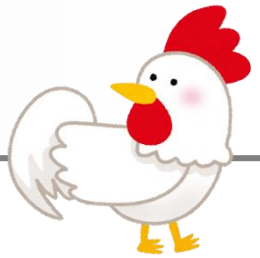
- ◆ CMMI: 習慣を実装する方法
  - CMMI V2 ($150)
  - https://tinyurl.com/yy8p7a33
- ◆ ソフトウェア工学
  - プレスマン
- ◆ TQM

プラクティスと価値のセット
- 【やること】 見積もり手法を作成する
- 【価値】 目標達成の見込みを高める

専門家Level

- L1: 初期
  - 課題に気づく
    実績を記録し、課題に対処
- L2: 管理
  - 目標管理
    目標を決めて、課題に対処
- L3: 定義
  - 標準化
    プロセスが安定している
- L4: 定量
  - 統計を活用
    ベースラインとモデルで管理と予測
- L5: 最適化
  - 事業の成功
    日々の活動が事業目標につながっていることの実感を持つ

中級者の時に知っておきたかったこと(取るべき行動)

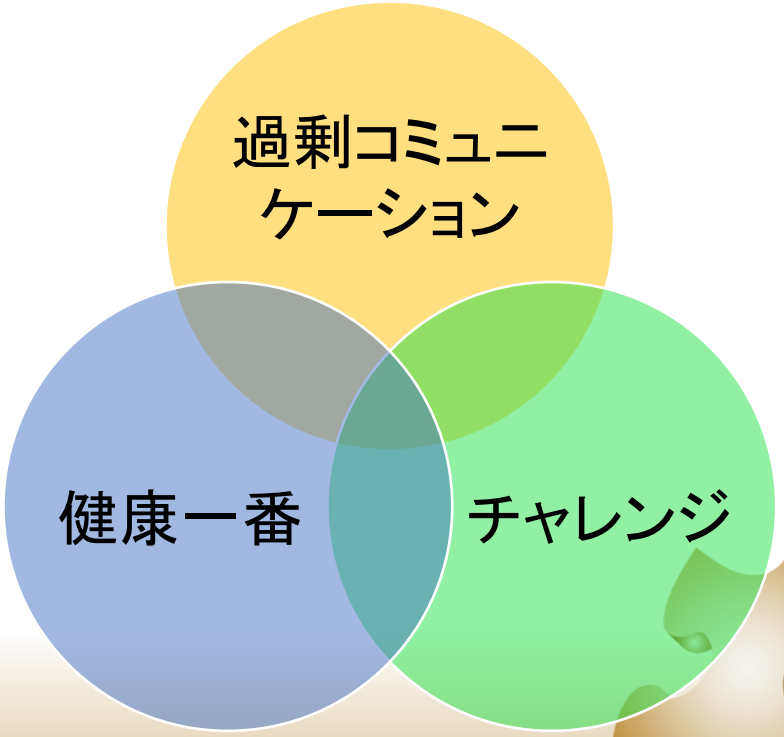


L1-3：プロセスを改善すれば、実績が改善される
L4-5：事業実績を向上させたいというニーズが、
プロセスを改善する主な原動力となる

ゼロをイチにする

(ゼロを100にしようとしない)

- ◆ 何度もPDCA(少しずつ前に進む)
- ◆ 英語読む(DeepLでも良い)
  - 海外カンファレンス参加
- ◆ TPI NEXTアセスメント
- ◆ 健康に良いこと



貪欲に
「話が来たら断らない」
(失敗したら話を持ってきた人の見る目が無い)

中級者の時に知っておきたかったこと(気を付けろ！)

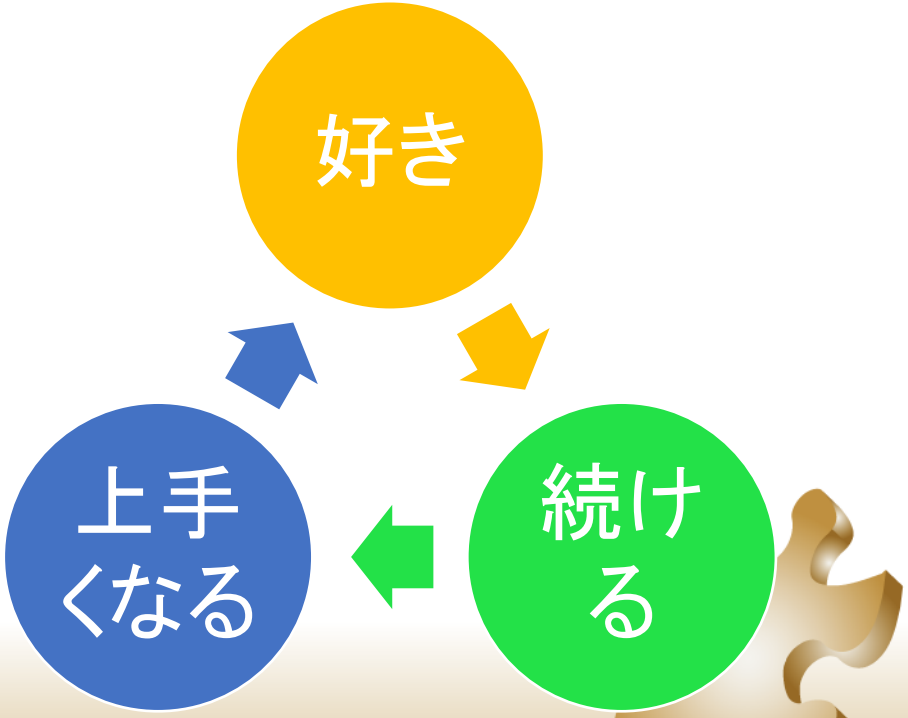
つよつよエンジニアと
言っても案外……。

自信を持つ

◆ **能力と成果**(関係は少ない)
- 自分に能力があると信じる
- 向き不向きは個性
- 謙遜していいのは、
  褒められた時だけ

※ **自分の方法を押し付けない**
- 「そういうこともあろうかな」

精神的ストレスを無くす



上級者(10年～)に知っておいてほしいこと(基本)

芯を外さない

- **なくなったら困るかで、仕事(＝していること)を見直す**
  - 思い切ってやめてみる(止められるのは上級者のみ)
- **システムのメカニズムを探す**
  - 悪循環を好循環に変える(案外、少しのことで変わる)
- **うれしいことが起こる機会を増やす**
  - 面白きこともなき世を面白く　住みなすものは心なりけり
    (高杉晋作　＋　野村望東尼)
- **活動能力の増大を招くものが良いもの**
  - 成長して当たり前のレベルが上がるか
- **上手くいっていなかったら間隔を短くする**

上級者に知っておいてほしいこと(してほしいこと)

他部署との調整

- **開発部門への依頼**
  - 開発者本人に【業務】のデモしてもらう
  - 3H(変化、はじめて、久しぶり)の確認
  - 良い設計になるように指導(テスタビリティ、設計レビュー)
  - 「テストで責任が持てるのはここまでです」宣言
  - 見つける問題の定義
    - いつから、どこで、誰が見つけられる
    - どうしたら簡単に見つけられる
    - そもそもその問題を作らなくできないか
- **全部門へ**
  - 品質意識の植え付けと自部門の価値のアピール

上級者に知っておいてほしいこと(気を付けろ！)

サイロになるな！

(自分のところだけ良くならない)

- **横展開しよう**
  - 統計的に効果を調べる → 効果ありなら社内に広める
  - 社外発表し反応を見る → 良い結果なら奨励し後続を作る
- **部下を育成しよう**
  - 信じて仕事を任す
  - 一人ひとりの育成とチームの育成を分ける
- **手法やプロセスは手段。運用段階にて目的化することで失敗する**
  - 新しいやり方で結果を出したいとの思いが、失敗したくないという気持を起こす
  - 「せめて新手法のやり方は間違えずに進めたい」と強く思う
  - 結果的に、「その新手法で何を実現するか」という目的を見失う

おすすめのテストの作り方（全体の流れ）

前工程の品質情報

テスト対象情報

① 準備

② 分析1

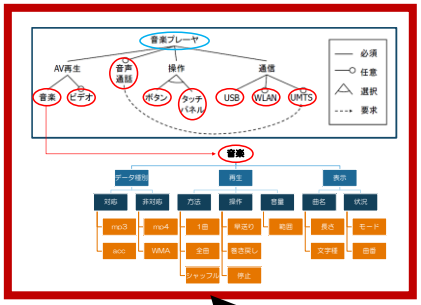
③ 分析2

④ 設計実装

テスト技法の適用

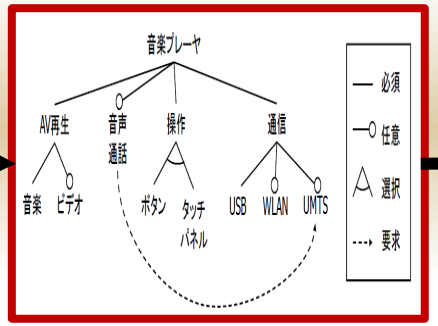
テストケース一覧表

テスト観点ツリー



テスト観点の洗い出し

フィーチャーツリー



機能分割

おすすめのテストの作り方（①　準備）

（1）　バージョン（仕様書・ソフト）の確認

- 目的：　テストするものを明らかにする
- 実施内容：　以下のバージョンを確認し、一か所にファイリングする
    1. テスト対象となるソフトウェア（バージョンが無いときには、リリース日）
    2. 仕様書などの「テストを作る源泉情報」（複数存在する全てについて）
    3. 動作環境と、対向ソフトウェア（＝　情報をやり取りする他システム）
    4. テスト計画書などのテスト関係ドキュメント

（2）　前工程の品質獲得状況の確認

- 目的：　前工程のプロダクト品質獲得状況を踏まえてテストを実施するため
- 実施内容：　前工程のテスト結果報告書を読み、発生した不具合を把握する
  レビュー会や移行審査会に参加することが望ましい

おすすめのテストの作り方（② 分析1）

（1） フィーチャーツリーの作成

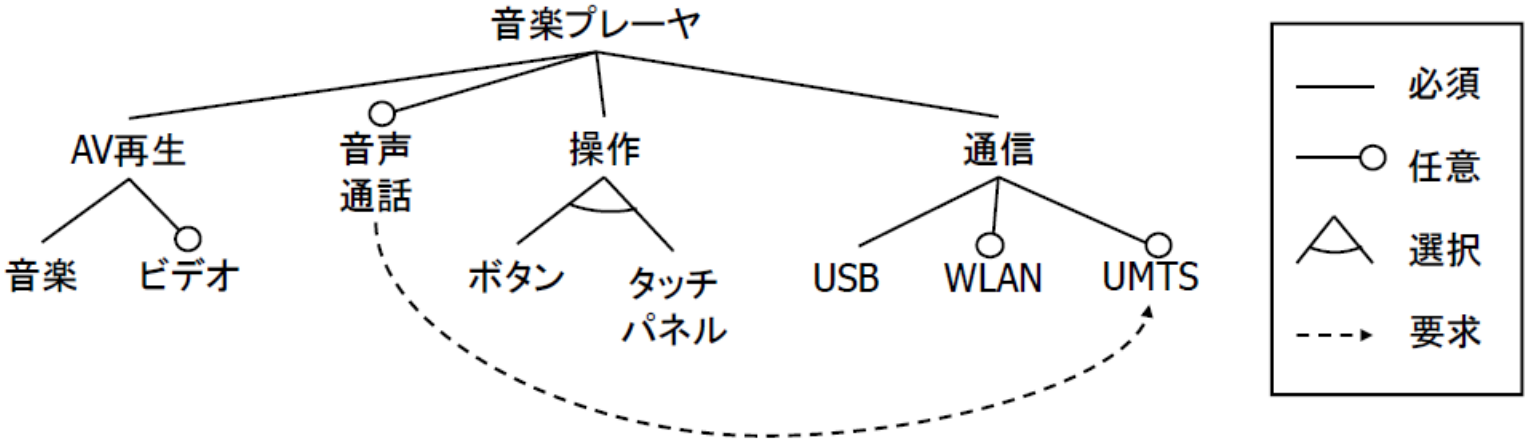
- 目的： テスト対象の機能をテスト設計可能な粒度まで細かく分解し、テストアイテムを明らかにする

  ※ 厳密には機能というよりフィーチャー（商品を特徴づけるもの）であるが、
  まずは機能を分解できることが重要

  ※ 機能で分割できる様になったらCIBFW（Condition, Item. Behavior, Fault, World）

  - 共通理解（ステークホルダーとテスト対象の共通理解が行える）
  - 部分理解でテストを作成できる（他との相互関係がないから）
  - テスト作成を分担できる（場所・時間・人）
  - **小さいものは理解しやすいので、作りやすいし、テストしやすい**

補足： フィーチャーツリー



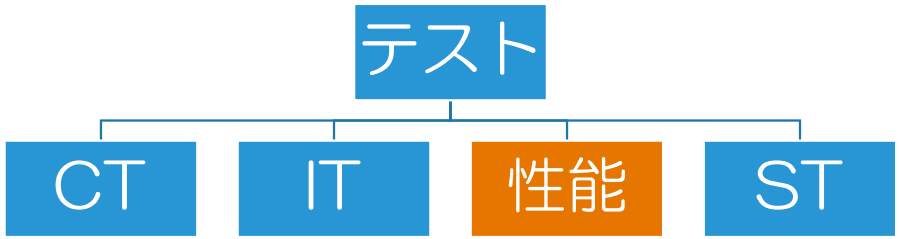
詳細化のレベル（仕様の理解に時間をかける！）

① それに関する情報を全て書き出すことが可能なレベルで理解している

② どんなテストをしたらよいか想像がつく

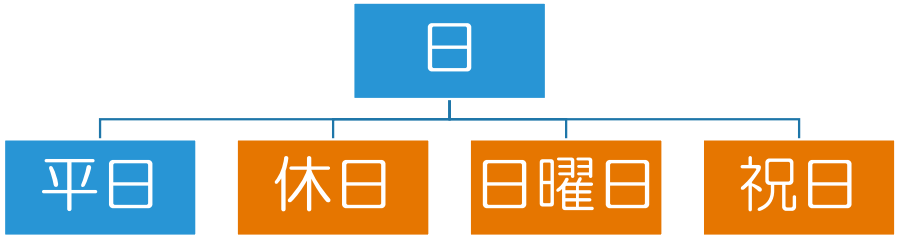
補足： 分解の4つの原則(MECEの実現)

①一貫性の原則



各段階における区分原理はただ1つでなければならない
※区分原理とは、分割するときの切り口のこと

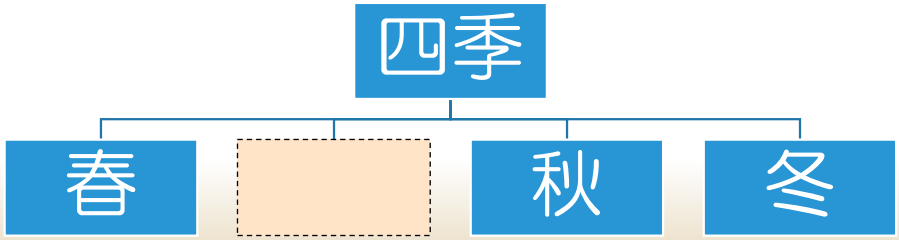
② 相互排除の原則



区分肢は互いに排他的でなければならない

※ 排他的とは、重なりが無いこと

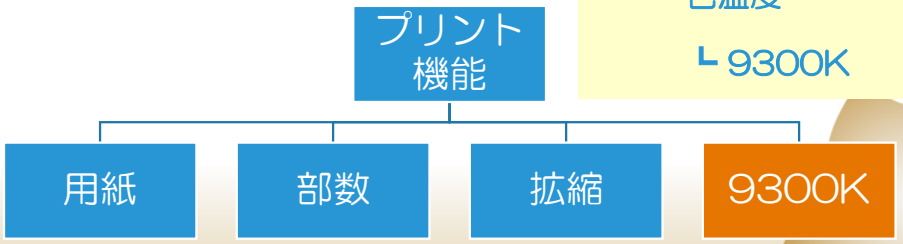
③ 一致の原則



区分肢は全ての場合を網羅する

※ 区分肢とは、分解された各部分のこと

④ 漸進の原則



グラフィックス
└プロファイル設定
└色温度
└ 9300K

区分は順序通りに上位概念から下位概念に進むべきで、飛躍してはならない
※ 区分とは、区切って分けること。分割

おすすめのテストの作り方（③　分析2）

（1）　テスト観点を木構造で追加

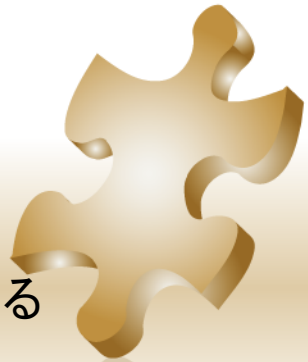
- 目的：　テストアイテムに対して何をテストするか、テスト条件を明らかにする
- 実施内容：　フィーチャーツリーの末端ノードであるテストアイテムに対して、テスト観点を木構造でつないでいく。このとき、末端ノードである「テスト条件」について、ハイレベルテストケース（※）が書ける粒度まで分解できたことを確認する

※　ハイレベルテストケースとは、抽象度が高く具体的な値は書かないテストケース

例）　設定温度の最大値に対して温度を上げる指示をしても、最大値のままであること

（2）　テスト条件（何をテストするか？）のカバレッジアイテムを決定

- 目的：　どこまで網羅するかを明らかにする
- 実施内容：　網羅的にテストするためのカバレッジアイテムと、目標値を決定する



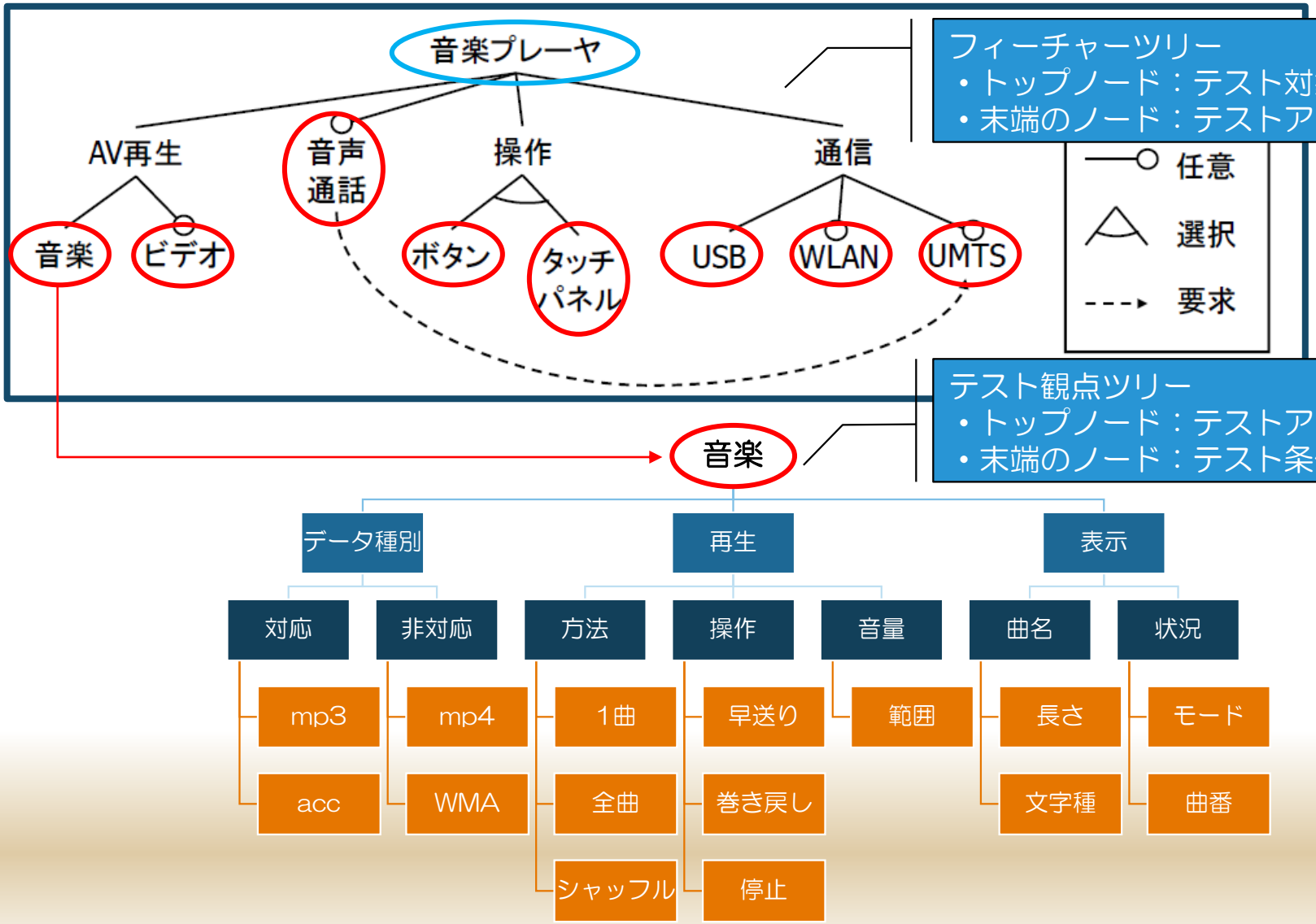
補足： テスト観点の例(対象ごとに違う)

- 環境
  - 1.外部
  - 2.リソース
  - 3.状態
    - データ
      - 設定
      - DB
    - トリガー
      - 外部
      - 内部
    - 時
      - 日付
      - 時間
    - 一般
      - 範囲
      - 順序
      - アナデジ
- 入力
  - 4.文字
  - 5.数字
- 出力
  - 6.画面
  - 7.帳票
  - 8.ファイル
    - ファイル名
      - 長さ
      - 禁止文字
    - コンテンツ
      - CSV
      - HTML
      - JSON
- 品質特性
  - 9.製品品質
    - 機能
    - 性能
    - 互換性
    - 使用性
    - 信頼性
    - セキュリティ
    - 保守性
    - 移植性
  - 10.データ品質
    - 正確性
    - 完全性
    - 一貫性
    - 信憑性
    - 最新性
    - 可用性
    - 移植性
    - 回復性

4つのテスト対象の切り口

10種類のテスト観点： フィーチャーツリーで見つけたテストアイテムに対して適用できるものを探す

例） フィーチャーツリーとテスト観点ツリー



おすすめのテストの作り方 （④ 設計実装）

（1） テスト技法の適用

- 目的： テストを網羅的に効率よく作成する
- 実施内容： テスト条件とカバレッジアイテムから、テスト技法を用いてテストを作成する

（2） テストケース表の作成

- 目的： テストケース（フォーマット例を次頁に示す）を作成する
- 実施内容： 入力値や期待結果について、具体的で誰がテストしても「同じ入力値」となり、「**同じYes/No**の答え」が返ってくるように記載する
  - **悪い入力値**の例）「10文字以上16文字以下のパスワード」、「任意の値」、「異常な値」、「すばやく」 → 「16文字（境界値）」等、具体的に書く
  - **悪い期待結果**の例）「UI崩れがないこと」、「UIが正しい」、「結果が正しいこと」、「適当な速度であること」、「問題が無いこと」、「違和感が無いこと」

これ使うかもリスト

項目	説明	URL
JaSST nano	最も発表のハードルの低いJaSST	https://www.swtest.jp/index.php?JaSSTnano
JSTQB FL	テスト担当者に要求される基本的な知識とスキル	http://jstqb.jp/syllabus.html
ASTERセミナー標準テキスト	ASTERセミナー標準テキストの解説	https://note.com/akiyama924/m/m6981810393cd
JIS規格	ユーザ登録すると読み放題（決定表、品質特性…）	https://www.jisc.go.jp/app/jis/general/GnrJISSearch.html
ソフトウェア開発分析データ集	IPAが収集したソフトウェア開発の定量データ	https://www.ipa.go.jp/ikc/reports/20200930.html
ソフトウェアメトリックス調査	JUASが収集したソフトウェアメトリックスデータ	https://juas.or.jp/library/research_rpt/swm/
高信頼化ソフトウェアのための開発手法ガイドブック	予防活動や検知活動に関わる手法や技法について解説および事例	https://www.ipa.go.jp/sec/publish/tn10-005.html

これ使うかもリスト

項目	説明	URL
統計の実務	仕事で「統計解析」を求められたときに参考にする	https://note.com/akiyama924/m/me17b12697128
CFD法	CFD法をマスターしたいときに読む	https://note.com/akiyama924/m/me17b12697128
原因結果グラフ	CEGTestツール	https://softest.jp/tools/CEGTest/
テスト設計ツール	GIHOZ	https://www.veriserve.co.jp/gihoz/
初心者向けおすすめ書籍	これからソフトウェアテストを学ぼうと考えている人に読んでほしい10冊	https://swquality.jp/swtest/1180/
SQiPライブラリ	SQiP活動の成果を公開	https://www.juse.jp/sqip/library/
はじめてのD-Case	合意形成手法	http://dimensions-japan.org/dcase/pdf/%E3%81%AF%E3%81%98%E3%82%81%E3%81%A6%E3%81%AED-Case.pdf

これ使うかもリスト

項目	説明	URL
派生開発協議会資料ライブラリ	XDDPやUSDMなどの資料	https://affordd.jp/libraries/
JaSST	JaSSTの発表資料	http://www.jasst.jp/
実践ソフトウェアエンジニアリング	ソフトウェア工学を網羅的に学ぶ書籍	https://www.amazon.co.jp/dp/4274227944
飯塚悦功プロジェクト	飯塚先生が品質について語っている動画	https://yusakunakao.com/perspective/iizuka_yoshinori_project/iizuka_yoshinori_toc/
DeepL	翻訳サイト。月に3ファイルまでなら無料	https://www.deepl.com/ja/translator
TPI NEXT	日本語版アセスメントシート	https://www.tmap.net/building-blocks/test-process-improvement-tpi
テスト設計コンテスト	「チュートリアル」と「過去のテスト設計コンテスト」	http://aster.or.jp/business/contest.html
ASTERソフトウェアテストチャンネル	ASTER提供のテスト動画	https://www.youtube.com/channel/UCKNO4SmQ7KRs4i2v_fLmCfA
質問箱	私の質問箱（お気軽に）	http://peing.net/ja/akiyama924

ご清聴ありがとうございました