テスト設計コンテスト2017 本選

テスト設計成果物の紹介

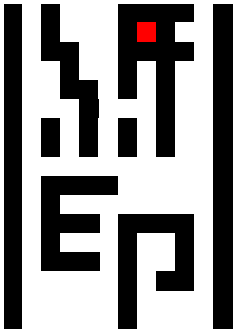
チーム

「紙印テスト倶楽部」

メンバー

小田部

チーム紹介



チーム名 **「紙印テスト倶楽部」**

コミケにて**テスト設計本**を**70部完売！**

新たなネタを披露しに来ました

メンバー紹介

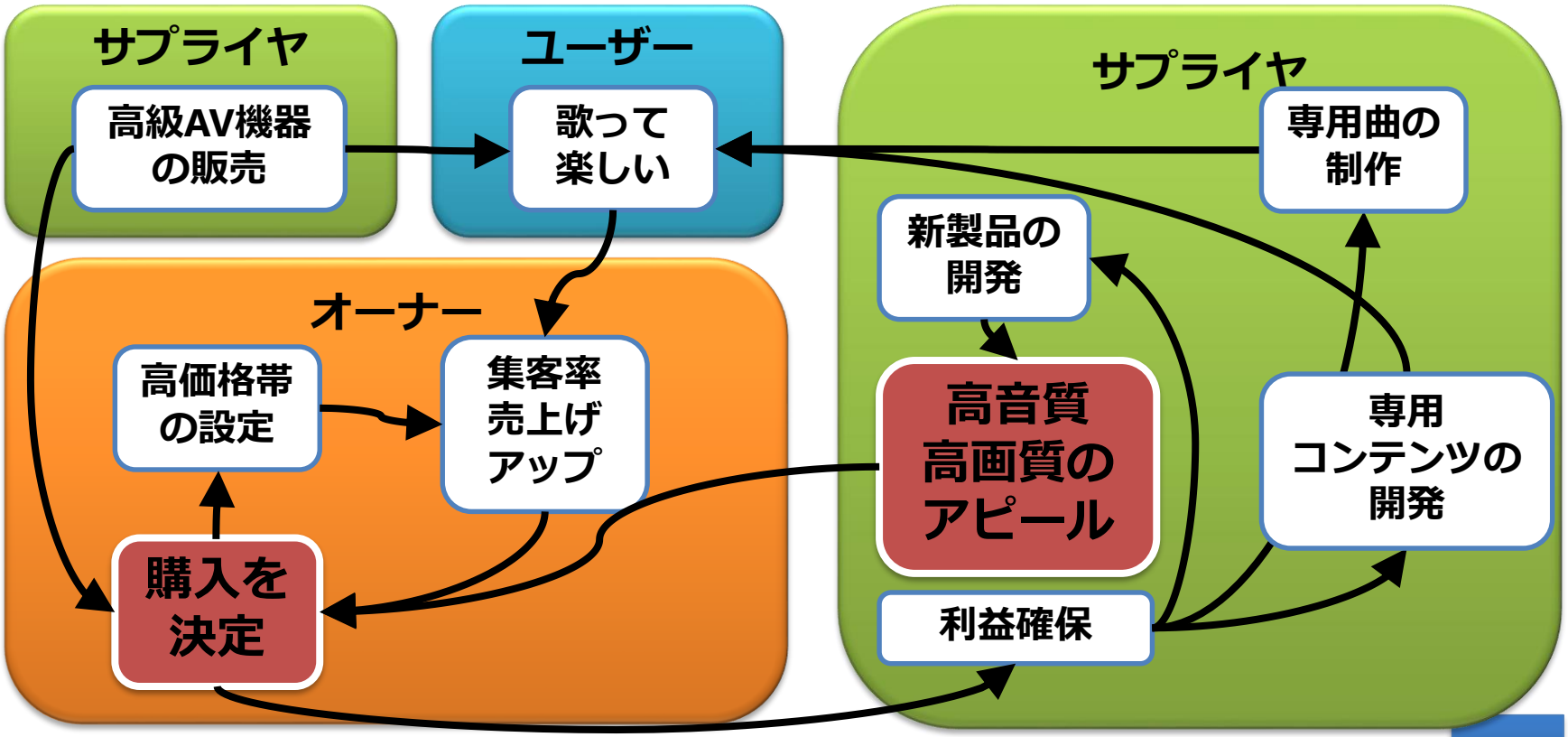
小田部

テスコン5年目にして

安定のボッチ

テスト設計の目的

- 各ステークホルダにとっての『**価値**』の最大化
  - その上で、最も重視すべきステークホルダはオーナーである
    - 購入されなければ他のステークホルダの要求も満たされない



開発背景と原理原則

◆開発スタイル

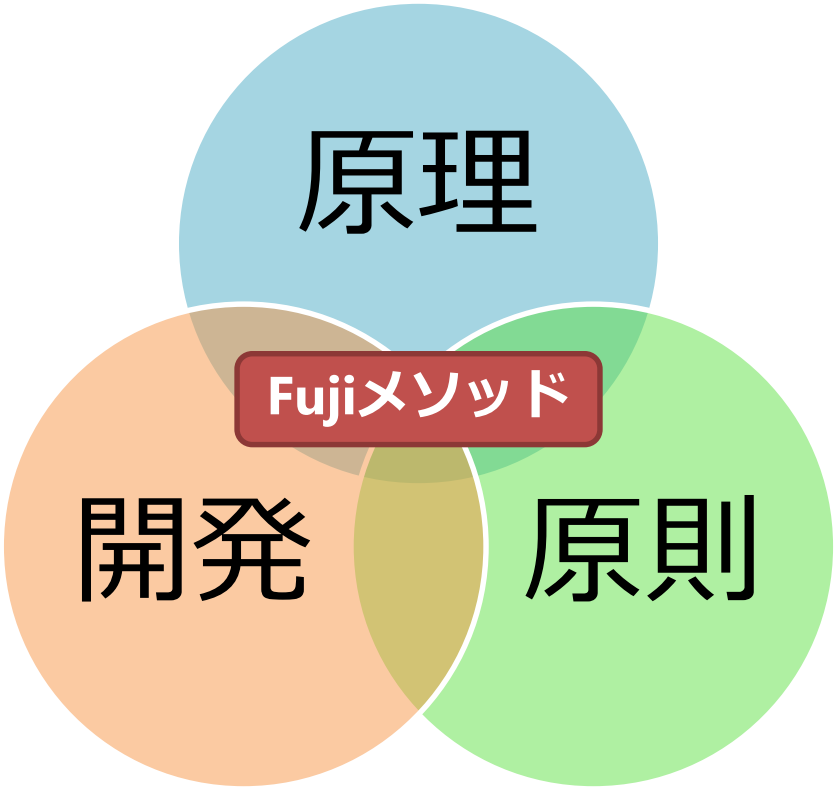
◆短納期、繰返し、修正や変更が前提

◆原則

◆自分、組織、世の中を『楽にする』こと

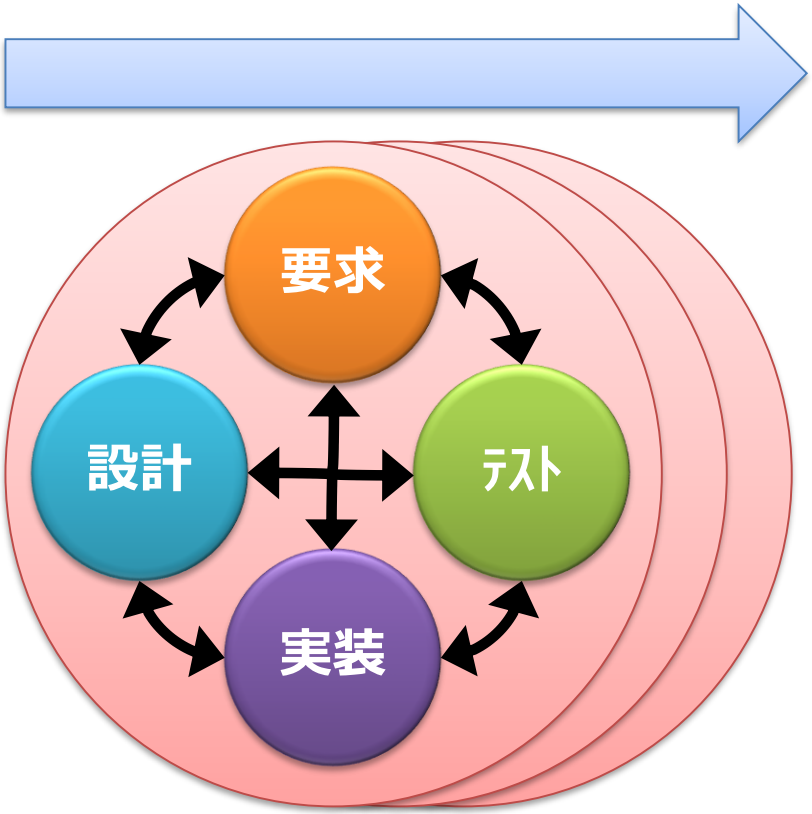
◆原理

◆理解、分解、再構築に加えて連携と統合



Fujiメソッド概要１：開発

- 以下の開発現場を想定
  - 自社開発
  - **短納期・繰返し**
  - **修正及び変更**が前提
- テスト設計方針
  - **軽量・高速**
  - **極小化・柔軟化**



Fujiメソッド概要２：原則

- **工程を厳選する**
  - 「掛けたリソース ＜ 得られる効果」であること
  - **「必要なとき、必要なだけ」**成果物を作成する
    - 極小単位に「分解」したテスト対象に対して作業する

- **成果物の再利用性を高める**
  - 既存の成果物を流用する事で、**短時間での成果物作成**を可能にする
  - 必要な情報のみを成果物から抽出する事で、**開発現場への柔軟な対応**が可能となる

- **常に次の工程を考慮して作業する**
  - 次工程での作業が効率よく進むように、現工程の成果物を作成する
    - 工程同士を**「連携」**させる

Fujiメソッド概要３：原理

事前調査と読込み

「カラオケ上達100の裏ワザ」

- カラオケのルーツはスナックの流し演奏者
- グループでワイワイ楽しむ
- 1人でストイックに歌の練習
- スナックで他の客の注目を浴びながら歌う
- カラオケコンテストで腕試し
- ネットで全国のカラオケ愛好者と交流

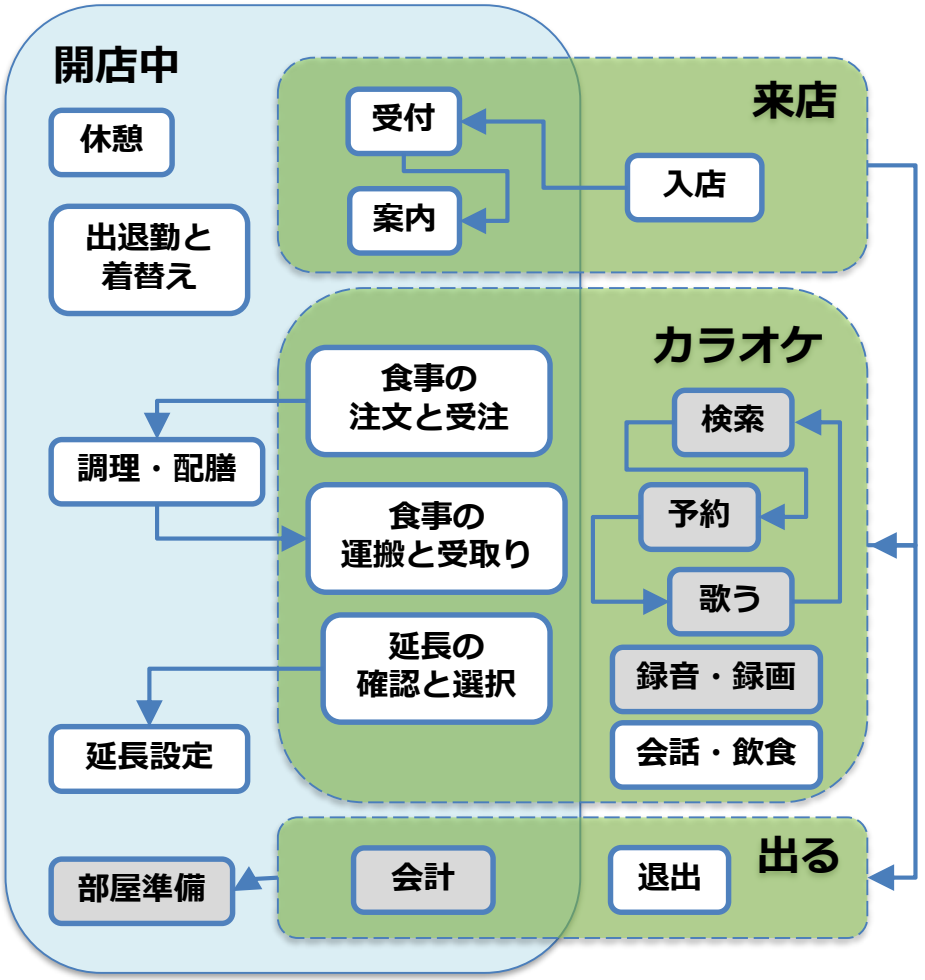
人物像と利用内容を具体化

ステークホルダ分析

		当り前	魅力	反対
ボックス店	個人	曲の調整 練習環境	歌唱評価 全国比較	誤評価 曲が少ない
	団体	簡単操作 見やすい画面	曲が充実 演出盛上がり	使いにくい 曲が少ない
	ｵｰﾅｰ	一括設定 手間いらず	人気有料コンテンツ 専用の曲と映像	飽きられる
ナイト店	個人	課金明確 曲がある	周囲の注目 名音声ﾌｨﾙﾀ	課金トラブル 他の客が占有
	ｵｰﾅｰ	一括設定 手間いらず	人気有料コンテンツ ｵﾌﾗｲﾝで快適操作	課金トラブル 客同士ﾄﾗﾌﾞﾙ
サプライヤ		機器の設定 しやすさ	自動接続 遠隔監視	不明な設定トラブル 解決しにくい

ユーザーアクションの分析

- テスト対象の利用を含む、**各ステークホルダの行動を想定**する
  - 開店中のカラオケ店を想定
- ステークホルダの行動にテスト対象の**システムが対応できるか**分析する
  - 「部屋準備」を早く終わらせるには、**一括標準設定機能**が欲しい
  - **録画中に店員の割込み**が入ったら嫌だろうな
  - 使い方を間違えたときに復帰出来るか？

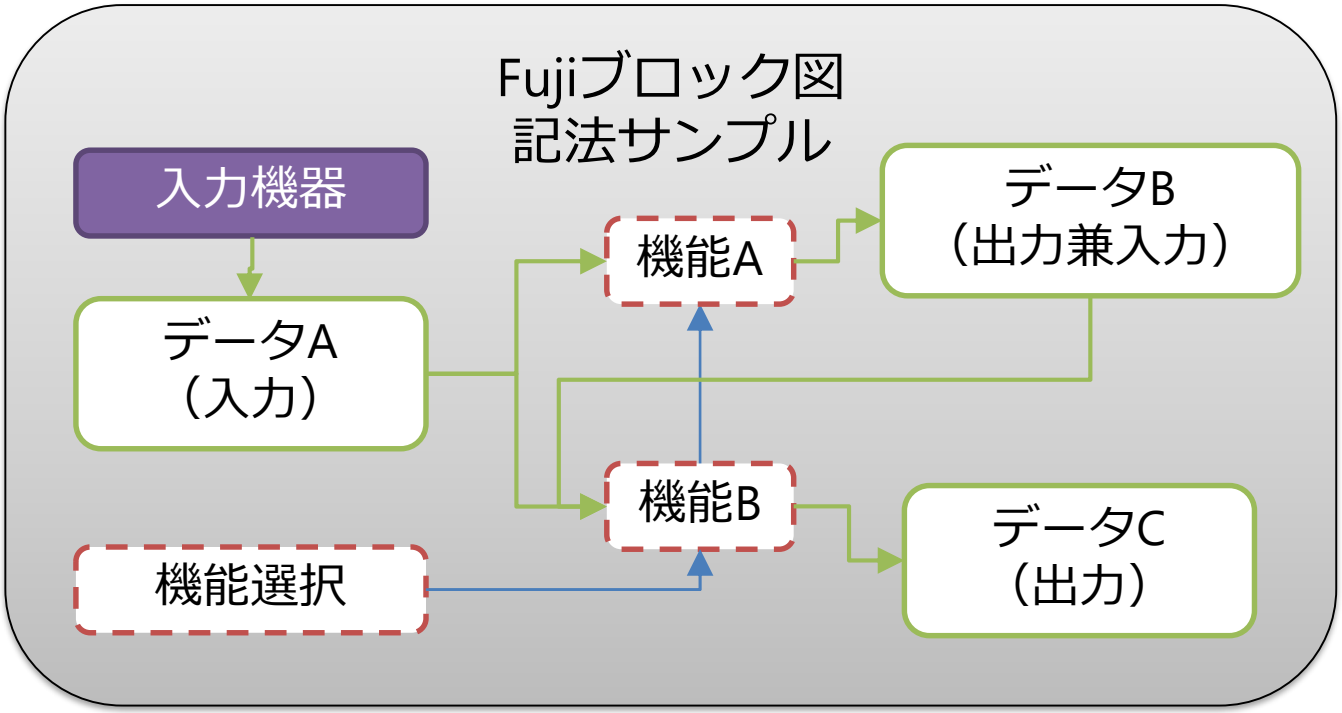


レビューとリスク分析

- テスト設計の**全プロセス中**で疑問点が出てきたときに随時追加
- 各プロセスは**様々な視点**で作業していたのでレビューを代用できた
- **重点リスク**の判断基準
  - 致命的な欠陥に直結
  - ステークホルダの要求に直結
  - 複雑な仕様

ﾓﾃﾞﾙ化（Fujiﾌﾞﾛｯｸ図）

- テストベースを**個々の機能とその入出力でモデル化**
  - 個々の機能単位での**軽量・高速・柔軟なテスト設計**を可能にした

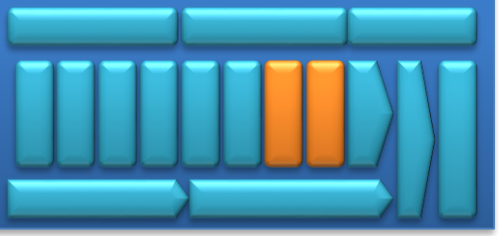


発散分析

分類整理を前提としたマインドマップ

- ラルフチャートと論理的機能構造を参考に、メインブランチを設定

詳細分析と分類整理



ツリー構造で分類整理

- 抜け漏れのチェックと再分析および「操作」「テストタイプ」「テストレベル」のタグ情報を追加し、テスト実装の内容を詳細化

Func　Type　Lv

テスト対象

- 入力
  - 内容
  - 入力元
- 出力
  - 内容
  - 出力先
- 内部
  - 処理
  - 状態
- ノイズ
  - ノーマル
  - アクティブ

◆ 抜け漏れ確認と再分析

◆テスト実装に使用する項目を取捨選択し分類整理

◆重複OK！

フィルタリング と集約

全分析結果をフィルタリングで集約

- 全分析結果を「操作・作業」「テストタイプ」「テストレベル」でフィルタリングし、集約された情報を用いてテスト実装

Func | Type | Lv

- 整理結果A
- 整理結果B
- 整理結果C
- 整理結果D
- 整理結果E

フィルタリング・集約

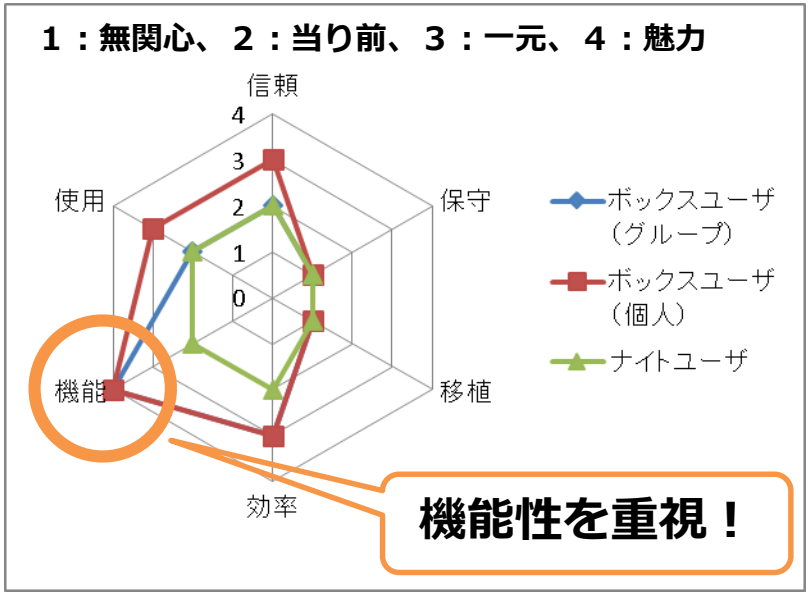
◆テスト実装に
必要な情報のみを抽出

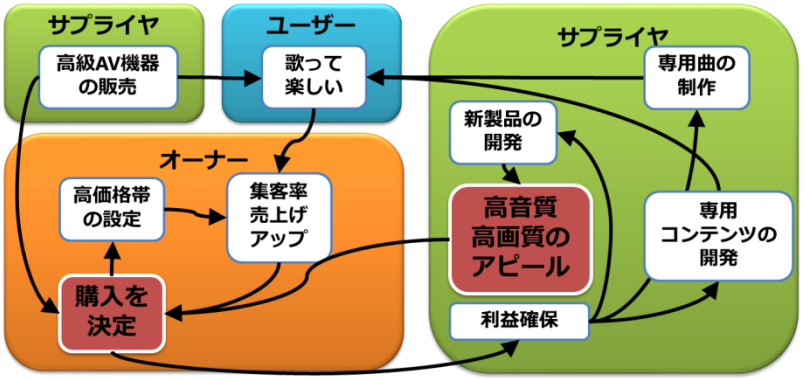
◆ **最も知見が蓄積**された
時点の情報を利用できる

◆ **重複した情報**も
まとめてテスト実装

価値の最大化分析

- 品質特性と狩野モデルを用いた**価値の定量化**
  - 比較可能にすることで最も重視すべき価値を検討しやすくする
- ビジネスモデルの構築
  - カラオケシステムの**価値が最大になる条件**を模索する
  - 価値の最大化に必要な**テスト観点**を決定する





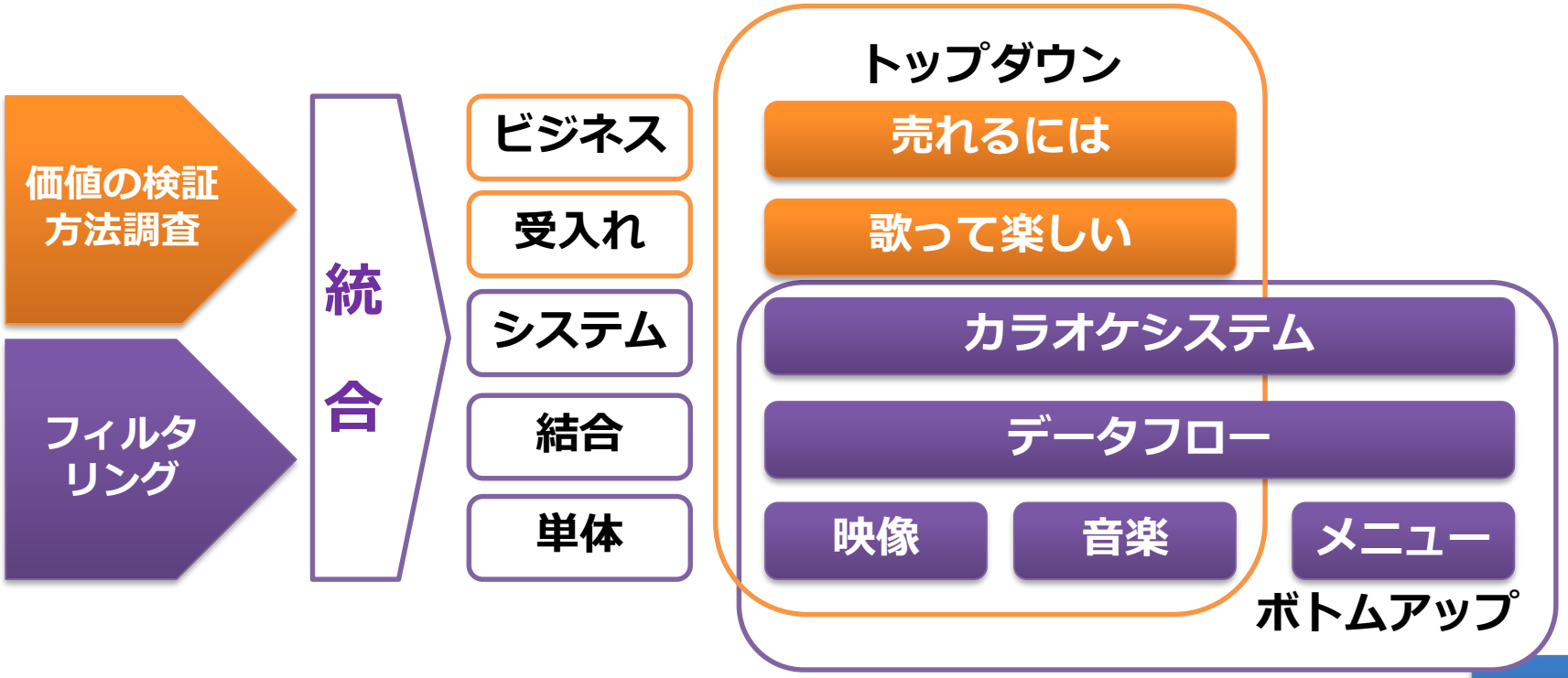
価値の検証方法調査

- **「ビジネステスト」**をテストレベルに追加
  - テスト対象が**売れる**（ステークホルダにとって価値がある）ことを確認する
  - ビジネスモデルに対するテスト
  - **テスト設計自体の価値**も高める

- **「売れる」**のテスト観点
  - 他社および既存製品と比べて**最も高音質・高画質**なカラオケシステムであること
  - **このカラオケシステムでのみ**、高音質・高画質を実現できること

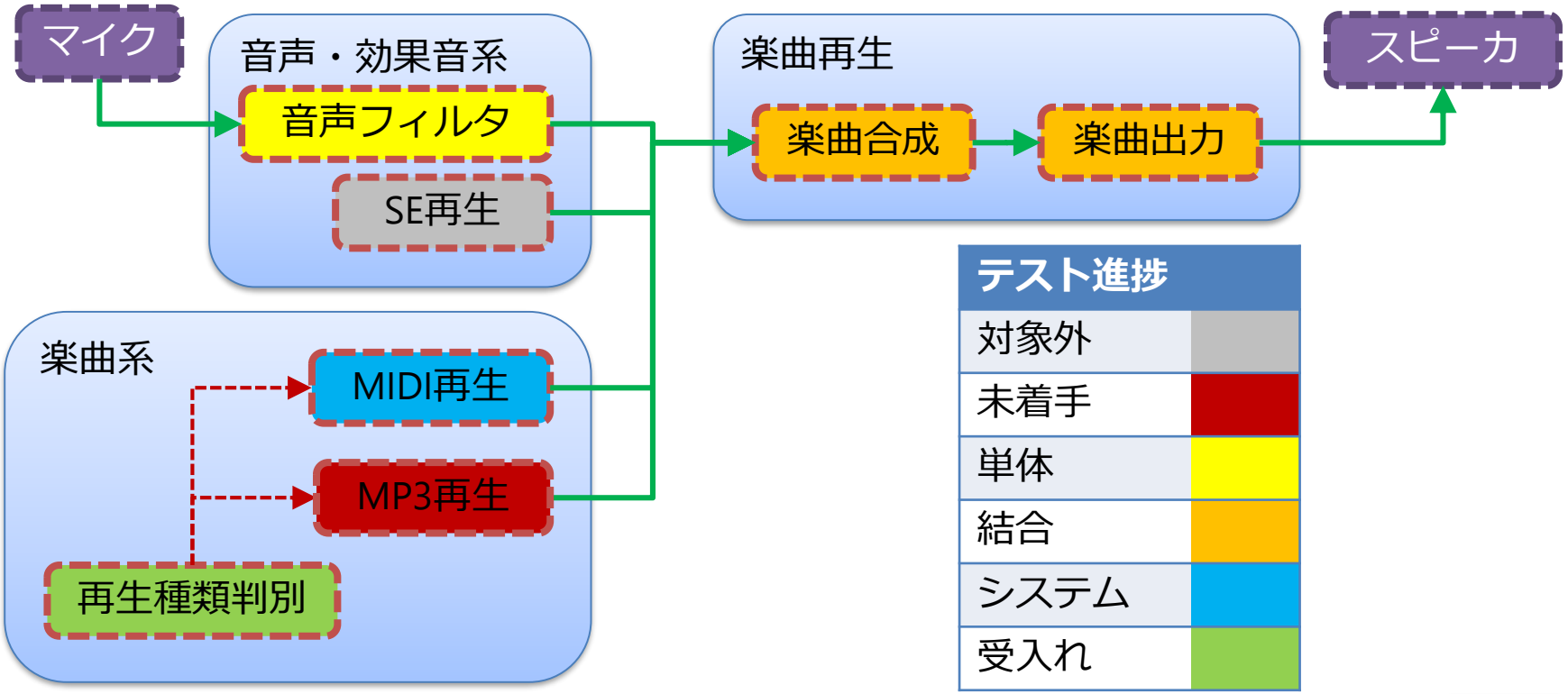
統合（相互補完）

- トップダウンとボトムアップの分析結果を**摺り合わせ**、各テストレベルのテスト観点を決定する
  - トップダウンは主に**最終的な品質レベル**を決定する
  - ボトムアップは主に**同時に確認すべき機能**を決定する



テストアーキテクチャ設計

- 開発に合わせたテストアーキテクチャ
  - Fujiブロック図より作成
  - 開発とテストの**進捗から次に実施するテストを決定**



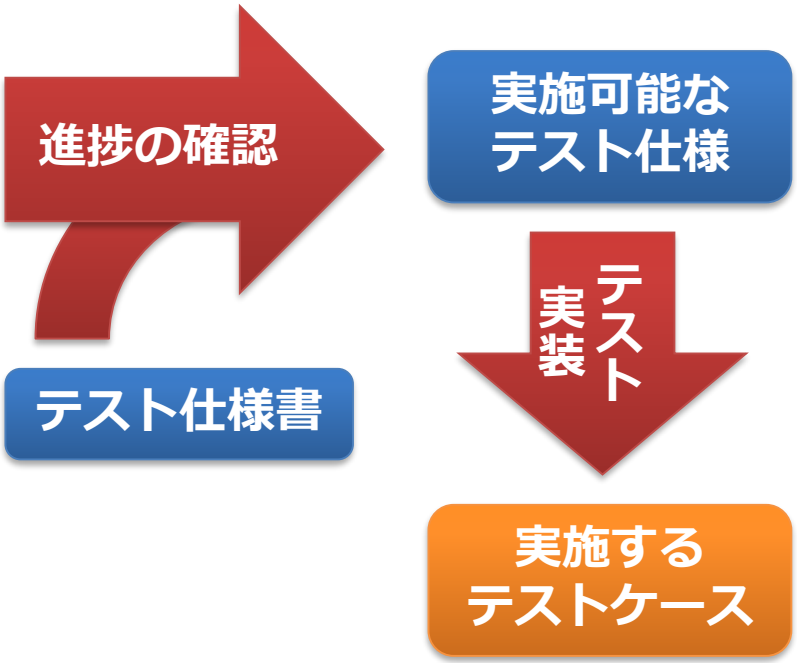
連携型テストケース

- **開発の進捗**に合わせて実施可能なテストケースを実装する
  - 開発済みの機能と実施済みのテストレベルから次に**実施可能なテストケースを判定**し、必要な情報を**テスト仕様書から抽出**してテスト実装する
  - テストアーキテクチャと連携する事で、開発状況に合わせたテストケースの実装と実施が可能になる

テストアーキテクチャ

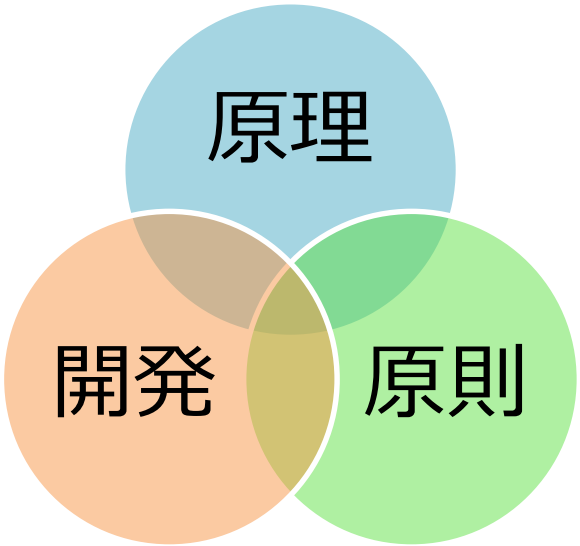
開発進捗
楽曲合成
音声ﾌｨﾙﾀ
MIDI再生
MP３再生

テスト進捗
対象外
未着手
単体
結合
システム
受入れ



Fujiメソッドまとめ

- **開発スタイル**
  - 開発範囲やテスト状況を俯瞰したテストアーキテクチャと、それに連携したテスト実装方法を用いる事で、**短納期、繰返し、修正や変更が前提の開発スタイルに対応可能**となった
- **原則**
  - 極小まで分解したテスト対象に対して各工程を連携させたテスト設計を実施する事で、**軽量高速かつ柔軟なテスト設計が可能**となった
- **原理**
  - **理解、分解、再構築**とその連携を意識する事で、目標とする開発スタイルや原則に対応したテスト設計をデザインできた
  - **ビジネステスト**までテストスコープを広げることで、売るためのテスト設計が可能になった
  - トップダウンとボトムアップの分析結果を**摺り合わせ**ることで、完成度の高いテスト設計を実現した



参考文献

- 唯野 奈津実著：「カラオケ上達100の裏ワザ」
- 湯本 剛氏　発表資料：「テストアーキテクチャの具体例」
  （http://jasst.jp/symposium/jasst12tokyo/pdf/A2-6.pdf）
- 小田部 健著：「テスト設計の実例と解説 2016年版」