テスト設計チュートリアル

ちびこん編 '23

2023/5/24

テスト設計コンテストU-30クラス審査委員
井芹　洋輝

自己紹介

- テスト設計コンテストU-30クラス
  ファウンダー/初代審査委員長（'17〜'22）
- テストエンジニア、QAエンジニア、開発者、コンサルタントとして、テストに関わる様々な活動に従事。現在は車載ソフトウェア開発にて、テストのテックリードやQAリードを担当

このチュートリアルについて

- テスト設計初学者を対象に、テスト設計の一通りの流れを、本質的な要点に焦点を当てて解説します
  - 一通りの流れの具体的なイメージを持っていただくのがゴールです
  - テスコンに合わせシステムテストのテスト設計をテーマにします

- 【テスト設計コンテストに参加予定の方へ】
  解説の具体例に2023年のテスト設計コンテストU-30テストベースを使用していますが、そこでの解説は絶対解ではありません。
  解説の要点・方向性から使えるものを各自判断でピックアップいただき、テスト設計に反映ください

このチュートリアルで使用する用語について

- チュートリアル、テスト設計コンテストU-30クラスで、以下の用語を使用します
  - テスト観点
    - テストすべきこと（VSTeP準拠）
    - 例：テスト条件、テストパラメータや、それを抽象化したもの
  - テストアーキテクチャ
    - テストの全体像を、テストの構成要素とその関係性、連携の段取りで表現したもの
- その他はすべてJSTQB/ISTQBの用語定義に従います
  - https://glossary.istqb.org/ja_JP/search

テスト設計コンテストについて

- 課題のテストベースに対してテスト設計を行い、その成果物を指定の審査基準で点数化して競い合うコンテスト
  - https://www.aster.or.jp/testcontest/
  - OPENクラス、30歳以下対象のU-30クラスの2クラス制
- チュートリアル、改善フィードバック、相談会（U-30クラス）、他チームとの比較・競い合いなど、テスト設計の技能を磨く機会が豊富です
- 参加募集中です。気軽にエントリーいただけると幸いです

アウトライン

- 【本編】テスト設計の流れ
  - 必要なテストを分析する
  - テストアプローチを考える
  - テストアーキテクチャを設計する
  - テスト条件を分析する
  - テストケースを設計する
  - テスト手順を実装する
- 【Appendix】
  - 探索的テストの設計
  - テスト設計技法の選択

JSTQBのテスト活動との対応

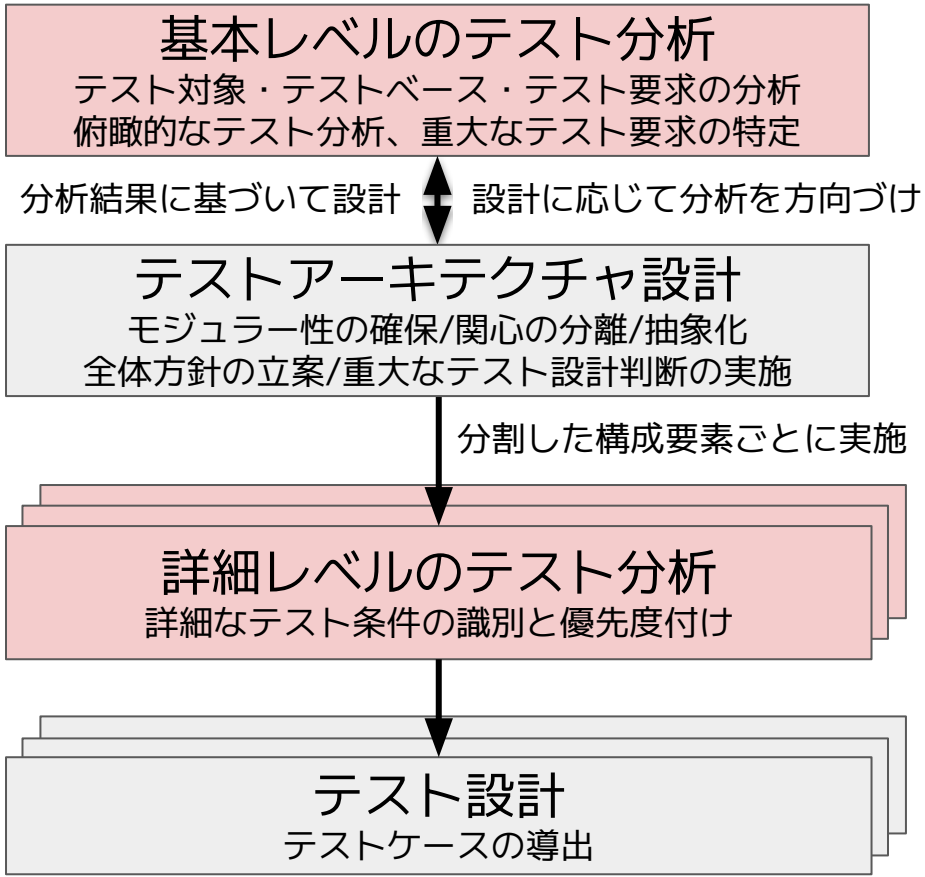
必要なテストを分析する

テストの顕在的・潜在的なニーズ・制約を収集・分析・整理して、自分たちのテストの責務、目的、課題・リスクを具体化するのが、テスト設計活動の立脚点になります

なお現場のテスト・品質保証は一般的に複雑で難易度が高いため、プロジェクト内外の連携で分担しながら実現します。
そのため、テストの責務・目的の具体化は、プロジェクト内外の仲間とどうテスト活動を分担・協力していくかすり合わせしながら進めていきます

【必要なテストを分析する】
前提：テストアーキテクチャ設計で複雑なテスト分析を支える

- 現場のテストは一般的に複雑で大規模
- 一定以上の複雑さ・規模を持つテスト設計では、テストアーキテクチャ設計を通して、テスト条件を分析できる粒度までテスト活動をモジュライズするアプローチが有効



【必要なテストを分析する】
テストの要求を集め整理する

- テストへの要求・制約を収集し、要求分析を行う
- テストの要求の識別・開拓の留意点：
  - プロダクトユーザ以外にも、重要なステークホルダがいる
    - プロジェクトの仲間や関係者、法規・標準関係、外部ベンダ
  - テスト対象の仕様書だけでなく、それ以外の補強情報も活用する
    - 類似製品情報、業界知見、設計・コード、上位の分析情報
  - テスト設計に影響するマネジメント情報も収集する
    - スケジュール、プロセス、リソース（ヒト・モノ・カネ・環境）、体制
  - 受け身だけではなく、主体的に要求獲得手段を開拓する
    - プロトタイピング、ユーザビリティ評価のような機会
  - テスト要求分析の改善フィードバックサイクルをまわす
    - 反復開発でテストを実施して、その結果分析から要求分析を見直す

【必要なテストを分析する】
テストの責務を確立する

- テスト対象、テストの十分性基準（いつ、どれぐらいまで）、テストのスコープ、テストで対応すべき状況・制約を明確化する
- 責務検討のアプローチ：全体の責務を明確化し、その中での分担・連携を検討しながら自分達の責務を明らかにする
  - プロダクトの全体構成を設計し、構成要素ごとのテスト責務を検討
  - 全体の開発プロセス・テストレベルを組み立て、工程ごとのテスト責務を検討
  - 全体体制を計画し、構成チームごとのテスト責務を検討
  - プロジェクトレベルの問題解決や課題・リスクへの対応でどう連携するか検討しながら、テスト責務を検討

【必要なテストを分析する】
テストの責務を確立する（プロダクト構成観点）

- 「U-30クラス テストプロジェクト要求補足書 2023」抜粋：
  - システムテストを担当するチームは、テスト対象のネイティブアプリケーション部分 のみをテストする。以下はシステムテストの対象外である。
    - Warikan アプリケーションを除く NomiKui 会サービス（アカウント認証）のテスト
    - ジャス Pay 機能（QR コードの生成、送金処理）のテスト
    - サービスサーバ（割り勘結果の記録）のテスト

【責務具体化アプローチ】テスト対象の全体構成が具体化され、担当が指定されている。そのため、以下に注力して、責務を具体化する
●ネイティブアプリの責務の詳細の具体化
●ネイティブアプリとそれ以外の責務境界の具体化
●担当と担当外の結合についてのリスクやテスト要求の分析

【必要なテストを分析する】
テストの責務を確立する（体制・プロセス観点）

- 「U-30クラス テストプロジェクト要求補足書 2023」記述：
  - コンポーネントテスト
    - Class、Widgetごとに、開発者判断でC0カバレッジ100%、仕様網羅のテストを作成・実行。CIでシステムテスト開始時は全合格状態を確保する
  - 統合テスト
    - 結合したアプリケーションに対し、起動できること、一通りの画面遷移ができることを確認。CIでシステムテスト開始時は全合格状態を確保する

【責務具体化アプローチ】テストレベルの担当と方針を指定。以下で深堀する
●担当のシステムテストの責務を具体化する
●他のテストレベルが簡易的にしかテストしない。その改善を行いつうつ、テスト全体の観点で、コンポーネントテスト・結合テストが漏れているところの補完も、テストチームの責務として検討する
※また、テストができる事・得意な事を提案して要求外の責務も開拓する

【必要なテストを分析する】
テストの目的を設定する

- テストの計画づくり、テスト分析、責務設計を通して、本質的なテストの目的を具体的に定義し、理解を得る
- 「U-30クラス テストプロジェクト要求補足書 2023」記述：
  - アプリケーションが用途を満たしていることを確認する
  - アプリケーションがリリースできる品質レベルであることを確認する
  - テストエンジニアの観点を活かし、テストの活動を通して、テストベースへ改善のフィードバック（曖昧さ、記述不足、矛盾に対する、指摘や改善提案）を行う

目的の具体化のため、要望されている内容の中での「用途」「品質レベル」を具体化する
さらに責務設定で見つかった目的を補強する（他のテストレベルの補完、連携システムとの連携の確認）

【必要なテストを分析する】
必要なテストの全体像を分析する

- 基本レベルでテスト要求を分析・整理し、テスト仕様の全体像を具体化。テストアーキテクチャ設計を着手可能にする
  - テスト対象
    - 対象項目（機能やコンポーネント）の整理・分解
    - 品質リスクの分析
    - テストベースの是正内容の特定
  - テストスコープ
    - テストの責務分担。分担の境界についての詳細情報の特定
  - テストの十分性
    - 品質リスクをどこまでコントロールするかなど、テスト十分性を示すものの設定
    - プロセス標準といった十分性についての要求特定
    - 必要な品質タイプやテストタイプといった、抽象レベルのテスト観点の整理
  - テストで対応すべき状況・制約
    - テストリソースの具体化、スケジュール等のマネジメント要求の特定
    - 課題やテストマネジメントリスクの特定

【必要なテストを分析する】
テスト分析の付随作業

- テストベースやテスト要求の欠陥解消・改善
  - テスト分析ではテストベースの欠陥（欠落、不整合、誤りや、テスタビリティ不足）が多く見つかる。それらを早期から修正することで、テスタビリティ含む品質向上・欠陥予防、開発やテストの手戻り・ブロック軽減を実現できる
- リソース・テスト環境の確保
  - テストのリソースや環境の不足・構築遅れは定番のテストブロック要因のため、初期のテスト分析から必要なものを分析し、段取りを立て、確保に着手する

テストアプローチを策定する

テスト設計において、困難な課題への対応や、一貫性を保った活動が求められる場合では、テスト活動の根幹となるテストアプローチを策定し、それをチームで推進するのが有効です

なお必要なテストアプローチは、プロジェクトの状況によって変化します。チームが活動する間、継続的にテストアプローチを見直して、チームの方向性を正していくことが重要です

【テストアプローチを策定する】

- テストアプローチ
  - テスト活動の進め方、考え方全般
    特定のプロジェクト向けにテスト戦略をテーラリングしたもの
    （JSTQB準拠）
- テストアプローチの策定
  - 有効なテストアプローチを支える基礎を日頃から蓄積する
    - 技術、プロセスや方法論、知恵や能力、環境、頼り先
  - テストに関する重大なリスク・課題を分析して、必要なテストアプローチを立案・推進する
    - プロジェクトリスク、品質リスクの分析とマネジメント
    - チームでの振り返り・課題だし
    - テスト成果物に対する有効性・効率性・その他品質の評価

【テストアプローチを策定する】

- 「U-30クラス テストプロジェクト要求補足書 2023」抜粋：
  - 高頻度の仕様変更、リリースに対応する
  - チーム内でのテスト担当のローテーションや引継ぎのため、テスト実行者が変わってもテストの再現性をある程度確保
  - アジリティ重視の開発のため、テスト活動の工数はなるべく小さく実施できるようにする。テストの責務分析に基づく最適化や探索的テストのアプローチ導入といった、アジリティ確保の施策の活用を推奨

【テストアプローチの例】
テストのアジリティ・変更性の確保のため、探索的テストの割合を最大化する。
探索的テストでは、テスト実行者の変更に対応できる詳細度のテストチャータを作成する

テストアーキテクチャを設計する

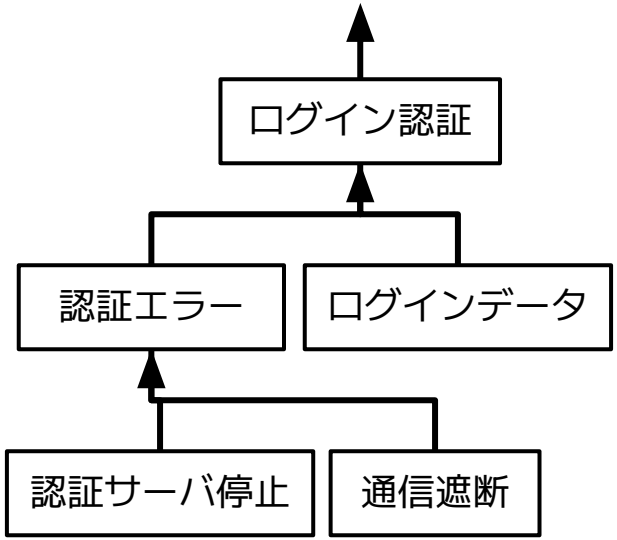
詳細なテスト観点やテスト条件をすべてまとめて分析すると、しばしば分析結果が発散・爆発し、テスト分析・設計の漏れや冗長性、誤りを誘発します。また、テスト全体を横断するような、大規模な方針や工夫が組み立てにくくなります

テスト構造全体を俯瞰・抽象レベルで整理・設計（＝テストアーキテクチャ設計）すると、関心の分離や、横断的なテストアプローチ運用を容易にし、それら課題を改善できます

【テストアーキテクチャを設計する】
テストアーキテクチャについて

- テストアーキテクチャ
  - テストの全体像の構成要素や関係性を、抽象・俯瞰レベルで表現したもの
- テストアーキテクチャの主な構成要素
  - テストの構成要素
    - 【JSTQB】テストレベル、テストタイプ、テストスイート
    - 【VSTeP】上記＋テストコンテナ、テストフレーム、テスト観点
  - テストの構成要素の関係性
    - 構成要素間でどう役割分担するか、どう連携するか
      （例：デプロイメントパイプライン）
  - テスト設計・実装の基本的な方針やルール
    - 一律適用するテスト設計・実装のルールやアスペクト
      （例：特定のテスト管理ツールにテストウェアの形式を統一）

【テストアーキテクチャを設計する】
テスト観点レベルでテストの構成要素を導出する



VSTePのテスト観点モデル

テストの責務の細分化・具体化アプローチ：

- テスト観点を、より詳細・具体的なテスト観点に分解する
- 具体的なテスト観点をグルーピング・抽象化してテスト観点の抽象・具体構造を得る

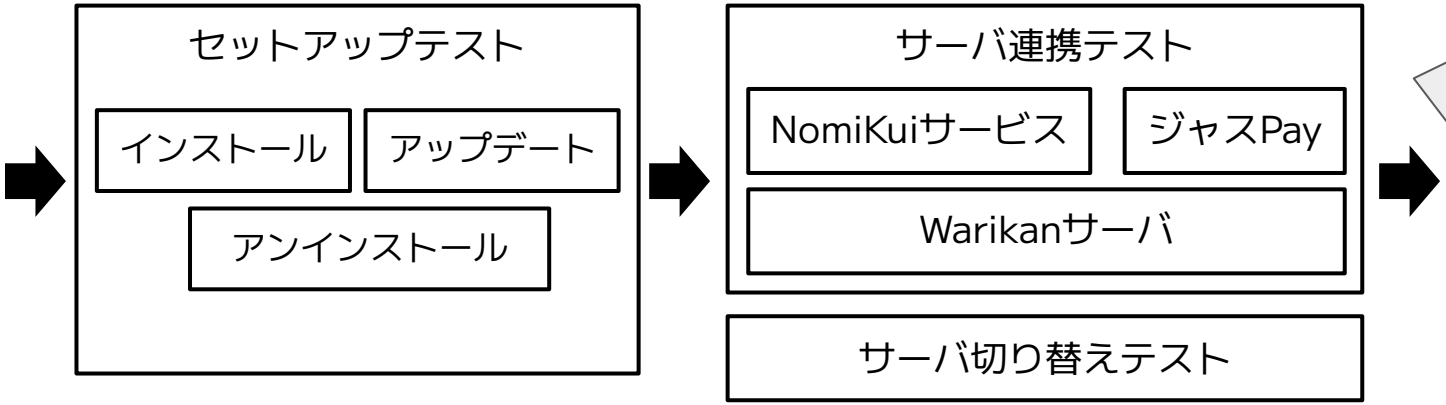
細分化・具体化の考え方：

- テスト観点の包含(has-a)関係を分析して分解する
  - 例）クライアントとサービスサーバでテスト分割
- テスト観点の抽象・具体（is-a）関係を分析して分解する
- 既存のパターンやモデルに基づいて分解する
  - 例）標準の品質特性モデルを観点に分析する
- テスト活動がやりやすくなるように分割する
  - 凝集度を高くして責務を明確にする
  - 結合度を低くしてばらばらにテストしやすくする
  - テスト設計アプローチの違いに基づいて分割する
  - テストリソースに合わせて分割する
    - 例）テストチームと専門チームでテスト分担

【テストアーキテクチャを設計する】
テストの責務間の連携設計で構成要素や関係性を導出する

課題	テスト設計方針
個人間支払いのセキュリティ保証	【機能テスト】ログイン認証、個人間送金の基本機能をそれぞれのユーザストーリーテストで実施 【セキュリティテスト】セキュリティリスクに対し、リリースできるセキュリティレベルであることを確認するテストを実施

品質リスクの高い所へのテストをピンポイントで厚くして、全体の機能テストの網羅性を緩和



工数やリソースの大きいテストの前に、セットアップや基本機能の品質を確保する段取りをテストに反映

【テストアーキテクチャを設計する】
テストの責務ごとにテストの厚み・網羅基準を調整する

- テストアーキテクチャレベルでテストの厚み・網羅基準を工夫することで、全体の有効性・効率性を改善するほか、のちのテスト設計を容易にする
- テストアーキテクチャレベルでの工夫：
  - ピンポイント型と一律網羅型を組み合わせる
    - 例）詳細なテストが必要なテストをピンポイントで補強し、全体の網羅型のテストの網羅基準を緩和する
  - テストの品質・特性に応じてテストの網羅基準を調整する
    - 例）詳細な網羅はなるべく効率的に実施できるテストで実施する
  - フィードバックに合わせて柔軟に調整する仕組みを入れる
    - 例）事前評価と、そこで得たリスクに基づくピンポイントテストを組み合わせる

テスト条件を分析する

漏れがなく効率的なテスト設計を行うためには、テスト観点・テスト条件の事前の分析が有効です

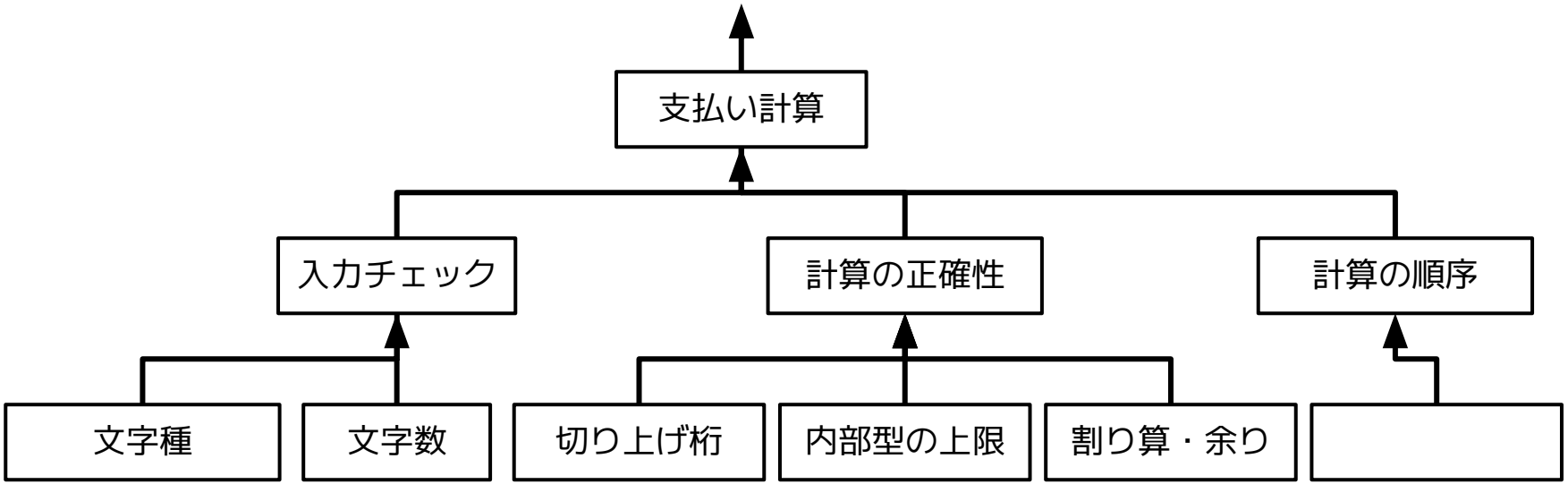
テスト条件の分析では、テストベースに書いてあることをなぞるだけでは必要なテストが欠落する可能性があります。
外部の知見・テスト設計者の知恵で補完したり、本質的なモデルを見出しそのモデルに基づいてテスト条件を分析したりするアプローチで、非明示的だがテストすべきテスト条件を見つけ出す必要性があります

【テスト条件を分析する】

1. テスト観点を詳細に洗い出す
    - テストベースだけでなく、テスト設計者の知見や補足情報を使って潜在的なテスト観点も洗い出す
    - 構造化と抽象化の工夫で、漏れなく有効で効率的なテスト観点の分析を行う
    - テストアーキテクチャ設計と相互フィードバックし、テストアーキテクチャとテスト観点を洗練させる
2. テスト観点のモデル・形式を見いだし、テストパラメータを特定するためのテスト条件を識別する

【テスト条件を分析する】
テスト観点を洗い出す

- 「U-30クラス テストプロジェクト要求補足書 2023」記述からテスト観点モデルを作成：



【テスト条件を分析する】
テスト条件を識別する

- 「U-30クラス テストプロジェクト要求補足書 2023」内容
  - テスト観点「支払いの割合表示」 の構成仕様
    - 支払いの割合は、初期状態でスライダーの 50 を指している。スライダーを左右へ移動することで、10 ずつ変化させる。スライダーを変化させることで、スライダーの下の文字も連動し更新する

【モデル】状態
【テスト条件の識別アプローチ】
・状態、遷移、遷移トリガを分析し、テスト条件を見つける
・遷移条件、初期状態、隠れた遷移などを深堀し、テスト条件を見つける
【分析作業】状態遷移モデルを作成する

テストケースを設計する

テスト分析で「何をテストするか」を具体化したら、要求・目的に基づいてテスト網羅基準を設定し、それに沿って必要なテストケースを設計します

テストケースの設計でも、テスト観点に対し本質的なモデルや形式を見出すアプローチが有効です。モデル・形式を見出すことで、それに沿った網羅基準設定や技法導入が容易になります

【テストケースを設計する】

- テスト条件に対するテスト網羅基準を定める
  - テスト分析やテストアーキテクチャ設計で具体化したテストの十分性基準に従って、テスト条件の網羅基準を定める
- 網羅基準に従ってテストケースを設計する

【テストケースを設計する】
網羅基準を設定し、テストケースを設計する

- テスト観点・テスト条件に、網羅基準を設定可能なモデルや形式を見出すアプローチが有効

ロジックに関するモデルとカバレッジ基準

モデルの例	適用できるテスト網羅基準の例
集合、値、選択肢	同値パーティションカバレッジ
パラメータの組	n-wiseカバレッジ デシジョンテーブルカバレッジ
境界を持つ値	境界値カバレッジ
状態	nスイッチカバレッジ
フロー、順序	制御フローカバレッジ

【テストケースを設計する】
網羅基準を設定し、テストケースを設計する

- 「U-30クラス テストプロジェクト要求補足書 2023」抜粋
  - テスト観点「支払いの割合表示」 の構成仕様
    - 支払いの割合は、初期状態でスライダーの 50 を指している。スライダーを左右へ移動することで、10 ずつ変化させる。スライダーを変化させることで、スライダーの下の文字も連動し更新する

【モデル】状態　※複数のモデル要素を持つ事が多い。状態はその一つ
【網羅基準】品質リスクが低いため、0スイッチカバレッジ100%（遷移パスを一通り網羅）を基準とする
【テストケース】網羅基準を達成する実行パスをテストケースに展開

テストを実装する

テスト実行の準備では、テストケースを実行可能な手順に具体化し優先度を割り当てます。そして実行に必要なテストウェアを揃えて、テストを実行可能にします

テスト実装では、テストを取り巻く状況や制約に合わせて、構造化や詳細度の調整を実施して、保守性を中心としたテストの品質を確保します

【テストを実装する】

- テストの要求・制約に合わせてテスト実行のためのテストウェアを構築する・手順書を揃える
  - トレーサビリティ確保重視：トレーサビリティ管理ツール対応
  - テスト実行コストにばらつき：手戻り時コストが少なくなるように順序付け
  - テスト実行者に制約：実行者に合わせて情報を詳細記述
- テスト実行のためのテストウェアの品質を作りこむ
  - テストスクリプトの構造化（例：データ駆動テスト）
  - テスト手順の利用者に合わせた可読性の確保
  - トレーサビリティ維持、変更管理等の各種作業の自動化容易性確保

【テストを実装する】
テスト実行の条件に合わせて適切な実装を行う

- 「U-30クラス テストプロジェクト要求補足書 2023」抜粋：
  - 高頻度の仕様変更、リリースに対応する
  - 将来的なチーム内でのテスト担当のローテーションや引継ぎのため、テスト実行者が変わってもテストの再現性をある程度確保できるようにテスト実装を行うこと
  - 探索的テストのアプローチ導入といった、アジリティ確保の施策の活用を推奨する

●同じチーム内メンバーが分かる詳細度での、ハイレベルテストケースあるいはテストチャータを記述した探索的テストとして、テストウェアを実装する
●変更性が高くなるようにドキュメンテーションを工夫する（重複記述は共通化して管理する、テスト要求とのトレーサビリティを明確化する）

まとめ：テスト設計の流れ

- 必要なテストを分析する
- テストアプローチを考える
- テストアーキテクチャを設計する
- テスト条件を分析する
- テストケースを設計する
- テスト手順を実装する

Appendix 1：探索的テストの設計

探索的テストの設計

- 「U-30クラス テストプロジェクト要求補足書 2023」抜粋：
  - 探索的テストのアプローチ導入といった、アジリティ確保 の施策の活用を推奨する
- 探索的テストは、スクリプトテスト（テスト手順をドキュメント化して実行する）と異なる強み・弱みをもっています。
  探索的テスト、スクリプトテストの弱みを補いあい、強みを活かしあうアプローチで、近年重視されるテストのアジリティや保守性の向上といった様々な恩恵を開拓できます

探索的テストについて

- 知識や動かして得たフィードバックを使って、テストの作成と実行を動的に並行して実施するアプローチ
- 探索的テストは、様々なアプローチがある
  - フリースタイル
  - テストチャータを用いるもの
    - テストチャータ：どのような目的で、どのようなアプローチで探索するか、探索のガイドを明文化したもの
  - スクリプトテストを補強するもの
  - セッションで管理しながら実行するもの
- 有効で効率的な探索的テストを実施するためには、要求や制約に応じた適切なテスト設計アプローチが求められる

探索的テストの有効性・効率性を高めるテスト設計

- テスト分析、テストアーキテクチャ設計：普遍的に重要
  - 全体の中でどこで探索的テストを活用するか位置付ける
  - どのような責務・目的で探索的テストを行うか明確化する
  - 探索的テストが必要なリソースを分析する
- テスト条件の分析、テスト設計、テスト実装：柔軟に調整
  - 探索的テストの責務に応じて工夫する
    - 仕様書の合致性検証、実現性検証をしたい
      - 仕様項目を整理してテストチャータとする
        - ユースケース一覧、ユーザストーリー覧、フィーチャー覧
    - 品質リスクが一定以下であることを確認したい
      - 品質リスクや、重要な障害を整理してテストチャータとする
        - 一定レベル以上の品質リスク一覧

探索的テストを支えるテストアプローチ

- 探索的テスト特有の課題に対応するテストアプローチを推進する
  - 効果はテスト実行者の属人的な知識・能力依存
    - 優れたテスト実行者を確保（人材要件を明確化し、人材確保）
      - 例）プロダクトオーナーや開発者を確保
    - テストエンジニアの知識・能力を高める機会を確保
      - 例）反復的テストなどで学習フィードバックサイクルを確保
  - テスト設計の品質が分かりにくい
    - テストチャータ等、品質を評価できる成果物を作成
  - 記録が残らない・再現しにくい
    - テストログレポートの設定、録画等ツールによるカバー

Appendix 2：テスト設計技法の選択

テスト設計技法の活用

- 業界で蓄積されているテスト設計技法は、テスト分析・テスト設計の進め方を導く
- テスト設計技法の適切な活用の恩恵：
  - テスト要求に対して効率的で有効なテストケースを作成できる
  - ミスを避けながら効率的にテストケースを作成できる
  - 曖昧・複雑な対象を整理しながらテストケースを作成できる
  - テストベースの理解、テストベースの欠陥検出を助ける
  - 議論や教育といったテストのコミュニケーションを支える

テスト設計技法の選択

- テスト設計技法の特徴
  - 対応できるテストベースのモデル・形式
  - 対応できるテスト要求
  - 強み・弱み

対応するモデル・形式に適合するか

対応するテスト要求に基づくか

強みを伸ばし、弱みを軽減できる状況か

- 上記の特徴に合致するかが選択基準となる

テスト設計技法の選択の例：
クラシフィケーションツリー法の場合

テスト対象の特徴・状況

- 環境構成のテストをしたい
- それぞれの環境構成の単機能・組み合わせをテストしたい
- 組み合わせはリスクに応じて柔軟に設定したい
- テスト環境情報は複雑。テストベースの記述は曖昧で不十分

クラシフィケーションツリー法の特徴
【対応するモデル・形式】同値パーティションのセット、組み合わせ
【対応するテスト要求】組み合わせバグ、単機能バグを網羅検出
【強み】複雑・曖昧なテスト条件を整理しながら分析できる。ツリー構造で全体を俯瞰しながら分析できる。
様々な網羅基準を選択可能

モデル、テスト要求が合致し、強みを活かせる
→技法として採用を判断できる

テスト設計技法の活用の留意点

- 不適切なテスト設計技法の導入は、テスト設計の効率性・有効性を損なう
  - 例）ユーザストーリテストを導入するために、USDM形式のテストベースをテストのためだけに全てユーザストーリ形式に書き換える
    - テスト用のテストベース二重定義は保守性が悪く、費用対効果がマイナスになる（本当にユーザストーリが良いならば、元のテストベースから修正するのが妥当である）
- テスト設計技法への盲従は、誤ったテスト設計を見逃す
  - 「方法論・手法・技法を正しく実践することで効果が出る。効果が出ないのは正しく実践できていないからだ」という教条への信仰で思考停止しない
  - 妥当な仕事ができているか、プロとしての自らの主観判断・感覚を忘れないこと

まとめ

- 【本編】テスト設計の流れ
  - 必要なテストを分析する
  - テストアプローチを考える
  - テストアーキテクチャを設計する
  - テスト条件を分析する
  - テストケースを設計する
  - テスト手順を実装する
- 【Appendix】
  - 探索的テストの設計
  - テスト設計技法の選択