テスト観点に着目した テスト開発プロセス(VSTeP)の概要

ソフトウェアテストシンポジウム2009東京
クロージングパネル
2009/1/29(木)
電気通信大学 電気通信学部 システム工学科
西　康晴

テストの開発方法論を生み出さなくてはならない

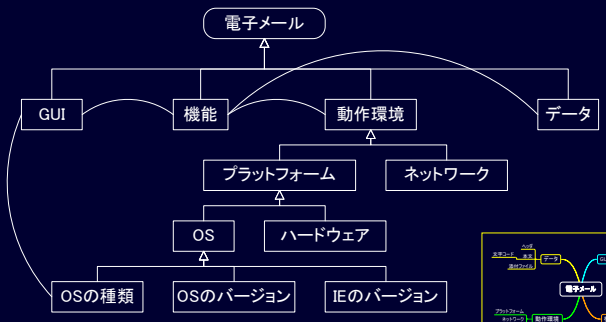
- テストの専門書を開いてみると
  - いろいろなテスト設計の技術が書いてある
    - » 境界値テスト、制御パステスト、状態遷移テスト. . . 。
  - 言っていることは分かるし、正しいと思うのだが、どう使ってよいのやら、よく分からない
    - » 果たして、どの技法をいつ使えばよいのだろう？
- 腕の良いテストエンジニアは、テスト設計の「前に」いろいろと考えているようだ
  - テスト分析&アーキテクチャ設計、もしくはテスト戦略立案を行っている
    - » お客様はどんな要求を持っているんだろう、このソフトは何を変えられるんだろう、仕様や設計のどの辺にバグがありそうなんだろう
    - » テストの全体像をどうしようか
    - » テスト戦略立案の重要性は強調されているが、具体的な方法論は無い
  - (テスト実施、)テスト設計からテスト開発へと テストのパラダイムを進化させ、テスト開発方法論を生み出さなくてはならない
    - » テスト実施 → (開発に例えると)いきなりプログラムを書く
    - » テスト設計 → フローチャートやモジュール構造図を書く
    - » テスト開発 → 分析・設計・実装といったライフサイクルに従って「開発」する

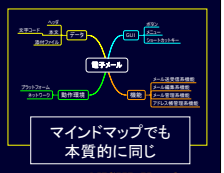
テストの「観点」

- テストには、様々な「観点」が必要だと言われている
  - Ostrandの4つのビュー
    - » ユーザビュー、仕様ビュー、設計・実装ビュー、バグビュー
  - Myersの14のシステムテスト・カテゴリ
    - » ボリューム、ストレス、効率、ストレージ、信頼性、構成、互換性、設置、回復、操作性、セキュリティ、サービス性、文書、手続き
  - ISO/IEC 9126の品質特性
    - » 機能性、信頼性、使用性、効率性、保守性、移植性
- テストにおいて重要なのは、全体像を把握しながら、テスト観点を漏れなくムラなく列挙し、きちんと詳細化することである
  - テスト観点には、お客様の求める品質、テスト対象の変化可能点、テスト対象の弱点などがある
  - テスト観点は階層構造を持ち、テスト観点同士に関連がある場合がある
  - 列挙されたテスト観点を用いると、テストの全体像を表すことができる

テスト観点を軸にして開発方法論を構築していく

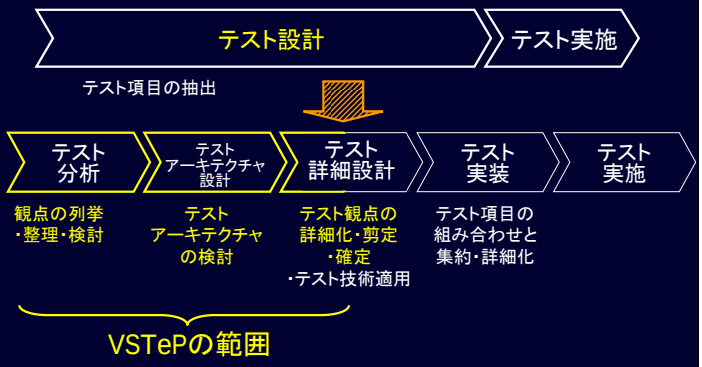
テスト観点図の例：ツリー風の記法





マインドマップでも本質的に同じ

テスト開発ライフサイクルと VSTeP の範囲



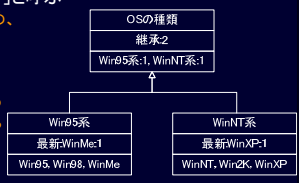
テスト観点に着目したテスト開発プロセス(VSTeP)の概要

VSTeP: Viewpoint-based Software Test Process

- テスト分析フェーズ
  - テスト観点(ビュー)を列挙・整理・検討・詳細化・体系化する
    - » 詳細化したビューを「フォーカス・クラス」と呼ぶ
  - ビュー(フォーカス・クラス)間の関連を検討する
    - » 組み合わせテストの基になる
- テストアーキテクチャ設計フェーズ
  - テストを設計しやすいように、ビュー全体を整理し分割する
- フォーカス・クラス設計フェーズ
  - 詳細化 / 剪定 / 確定
- テスト詳細設計フェーズ
  - 既存の技術でテスト設計を行う
    - » ユースケーステスト、状態遷移テストなど様々な技法がある
- テスト実装フェーズ
  - テスト項目の集約を行う
    - » 異なるテスト技法から得られた同じ価値のテストケースをまとめる
  - テスト項目の組み合わせを行う
    - » All-pair法や直交配列表などを用いる
  - テスト項目の詳細な記述を行う
    - » テストオペレータが実行可能な記述まで詳細化する
    - » 自動実行ツール用スクリプトを作成する

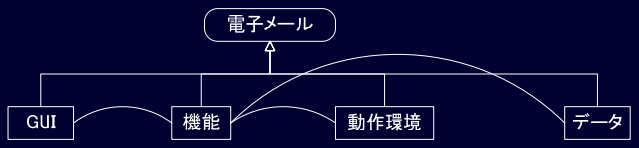
テスト観点（フォーカス・クラス）の列挙と詳細化

- まず大まかなテスト観点を列挙し、階層的に詳細化（ズームイン）していく
  - 各レイヤーの観点を「フォーカス・クラス」と呼ぶ
    - » 一番下のフォーカス・クラスは、同値クラスのようなものになる
  - フォーカス・クラスは階層構造を持つ
    - » 詳細化には、継承、合成集約、属性化、因果関係などがある
    - » MECE分析、詳細化タイプ分析により漏れを防ぐ
    - » 多段階の詳細化によって再利用しやすい設計となる
  - 階層の最上位のフォーカス・クラスを「ビュー」と呼ぶ
    - » 最上位のフォーカス・クラスに代表されるため、階層全体をビューと呼ぶこともできる
  - 詳細化しながら網羅基準を定めテスト項目数を概算する
    - » 網羅基準とテスト項目数を記述して概算する
    - » 下位のフォーカス・クラスのテスト項目数から上位のテスト項目数を算出する
    - » 比較的詳細な見積もりと考えることもできる



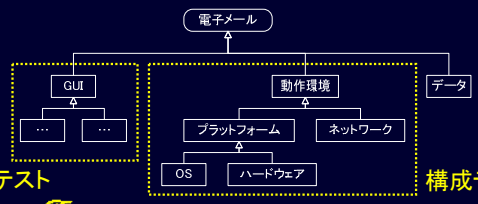
テスト観点間の関連の検討

- テストでは、複数の観点を組み合わせる必要がある
  - 例）負荷テストでは、搭載メモリ量という観点と、投入データ量という観点を組み合わせてテスト設計を行う
    - » 搭載メモリ量と投入データ量のバランスによって、テスト対象のふるまいが変化するからである
  - すなわち、観点同士は依存関係を持つ場合がある
    - » フォーカス・クラス間の「関連」として表現する
    - » 矢頭の無い線で記す



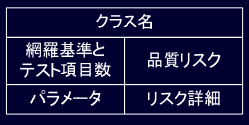
テストアーキテクチャの設計

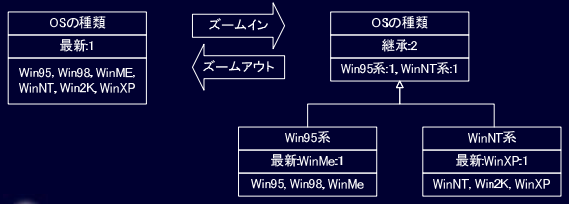
- 可能な条件を全て同時にテスト設計時に考慮するのは煩雑であり現実的でない
  - テスト設計しやすいまとまりや、重視するテスト観点のまとまりにテスト観点群を分割する必要がある
  - 例）カテゴリ型テストアーキテクチャ
    - » 関連の少ないフォーカス・クラスのグループをテストカテゴリとしてまとめ分割していく



フォーカス・クラスと関連の剪定

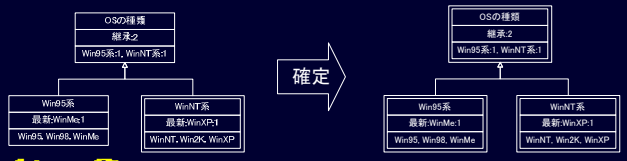
- 3つの方法でテスト項目数とリスクとのトレードオフ（剪定）を行いながら、計画されたテスト工数に合わせこんでいく
  - クラス内の網羅基準の緩和
  - ズームアウト
  - 関連の組み合わせ基準の緩和・削除





「確定」によるテスト漏れリスクの軽減

- 剪定には、潜在するテスト漏れのリスクが付随してしまう
  - 同値分割がきちんと行われていない（ズームアウトされた）フォーカス・クラス
  - 網羅基準が緩いフォーカス・クラス
  - 削除したり組み合わせ網羅基準を緩めた関連
- 潜在するテスト漏れのリスクを軽減する作業を「確定」と呼ぶ
  - 「確定」によって、各フォーカス・クラスやテスト設計全体のリスクを評価する
    - » ソフトウェア設計のレビューや発生頻度の予想など
  - リスクの小さい確定フォーカス・クラスと、リスクの大きい暫定フォーカス・クラスを区別して記す
    - » 子クラスが全て確定できたら、親クラスも確定できる



VSTePの効果

- 質の高いテスト開発
  - テストのモデリングの質の向上
  - 説明責任の確保
  - テストモデルのレビューの促進
  - テクニカルコミュニケーションの容易化
  - テストデザインパターンの抽出・再利用
- テスト開発の改善
  - テスト漏れの原因追及と改善
  - テストケースのリバースエンジニアリングによる十分性評価
- 上流の改善
  - テストとの一体化によるレビューの強化
  - バグの作り込まれやすい箇所の反映
  - Wモデルによる開発の見通しの向上と上流の分析・設計の質の向上
- テスト漏れの原因の例
  - テストのモデリングの失敗
    - » ビューのモレ／関連のモレ
    - » 不適切／曖昧なビューの解釈
  - 剪定の失敗
    - » クラス内の網羅基準の過度の緩和
    - » 過度のズームアウト
    - » 過度の組み合わせ基準の緩和・削除
  - 確定の失敗
    - » ソフトウェア設計の内部まで突っ込んで確定しなかった
    - » ソフトウェア設計を基に確定したのに、設計どおり実装されていなかった
    - » 利用頻度の見積もりを誤った
  - それ以外の失敗
    - » 不具合が作り込まれる原因にさかのぼって分析しパターン化する