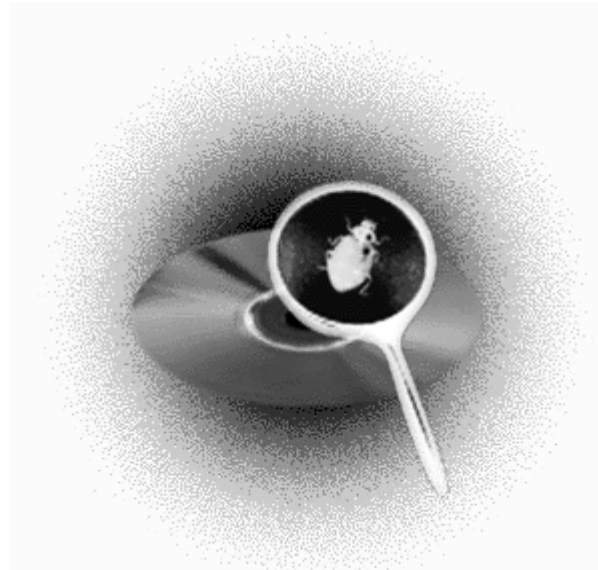


テスト観点に着目した ソフトウェアテスト設計プロセス



第29回 SEA関西プロセス分科会

2006/11/21(火)

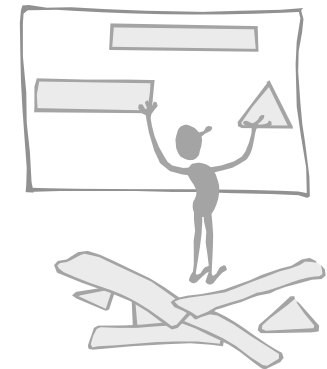
電気通信大学 電気通信学部 システム工学科

西 康晴

テスト設計の難しい点

- 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法が書いてある
 - » 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か？
- テストの「観点」は、テスト設計の本質である
 - 機能の網羅はちゃんと出来たけど、動作環境が色々あるのは気づかなかった
 - 境界値テストを設計したけど、そもそも同値クラスが浮かばなかった
 - 直交表を使ってテストを設計したけど、因子が抜けてしまった

テスト観点に着目した
ソフトウェアテスト設計プロセスが必要



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
- テストの「観点」とは何だろう？ — テスト対象のモデリング
 - テスト対象の持つ、テストすべき側面
 - テスト対象が達成すべき性質
 - テスト対象(及び含む世界)を、テストの立場からモデリングしたもの
 - » テストする必要が無い側面は、モデリングする必要が無い
 - » 達成する必要が無い性質は、モデリングする必要が無い
 - 抽象的で、階層構造を持つ
 - » 同値分割の包括的なもの



テスト観点のモデリングによるテスト設計プロセス

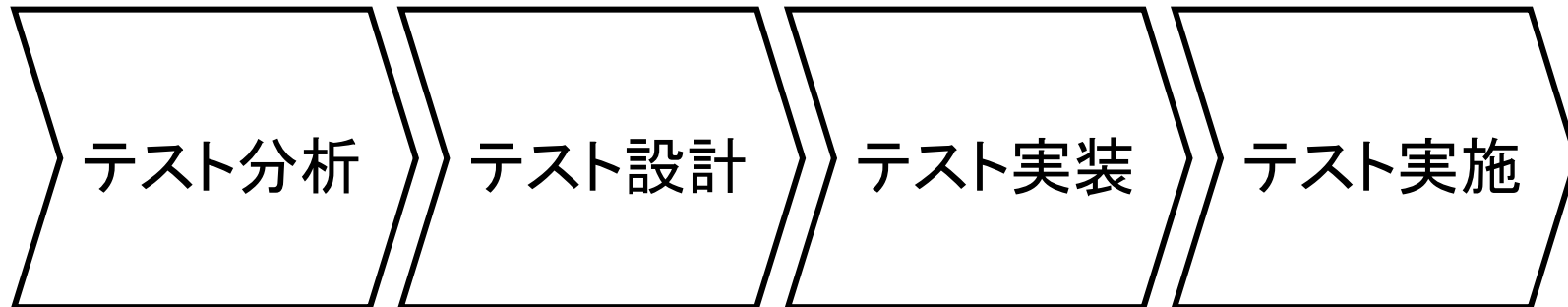
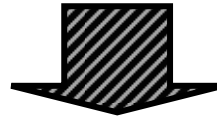
- テストの分析やアーキテクチャ設計は行われなことが多い
 - テスト要求モデリングやテスト設計モデリングは、ほとんど行われな
 - » いわゆるテスト設計技法は、テストの詳細設計以降で用いる技法ばかりである
 - » テストの詳細設計を行うために、状態遷移図などソフトの設計モデルを上流から流用したり、テストでリバーシして作成することはある
 - » そのため、テストの分析パターンや設計パターンは流通していない
 - » またテストの設計方法論もほとんど提案されていない
 - » テストの観点の抽出というのは、組織の暗黙的ノウハウである
- テスト観点のモデリングを行うテスト設計プロセスが必要
 - テストで考慮すべき観点を一覧でき、俯瞰的に整理できる
 - » Excelで「大項目・中項目・小項目」と詳細化していく組織が多い
 - » モデリングを行うと俯瞰できるだけでなく、ビジュアルに捉えられる
 - テスト観点のモデリングを「テスト分析プロセス」と呼ぶ
 - » テスト対象、ユーザの求める品質、使われる世界などをモデリングする
 - » これらはテスト設計から見ると、「そこに存在する」ものである
 - テスト分析モデルを基にテスト設計を行っていく
 - » まずテストアーキテクチャを構築する
 - » アーキテクチャを段階的に詳細化することで設計していく



テスト設計を、分析と設計(と実装)に分ける



テスト項目の抽出



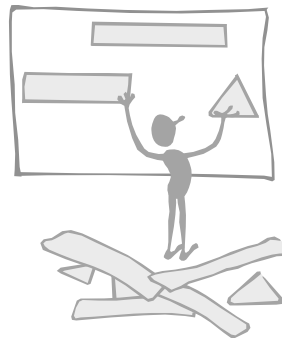
観点の列挙
・整理・検討

アーキテクチャの検討
・テスト項目の剪定
(トレードオフ)

テスト項目の
組み合わせと
詳細化

テスト観点に着目したテスト設計プロセスの概要

- **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**
 - 既存の技法でテスト設計する
 - » 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
 - » 自動実行ツール用のスクリプト作成を行う



NGT: Notation for Generic Testing

- テストのモデリングのための記法: 検討中

- 名称: NGT (Notation for Generic Testing)

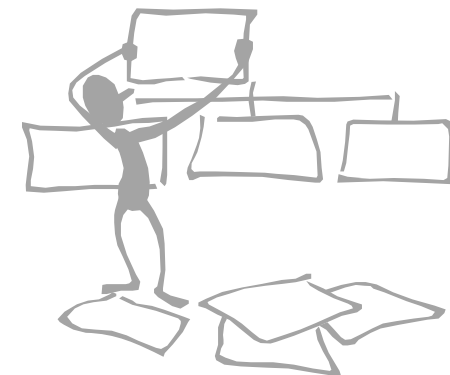
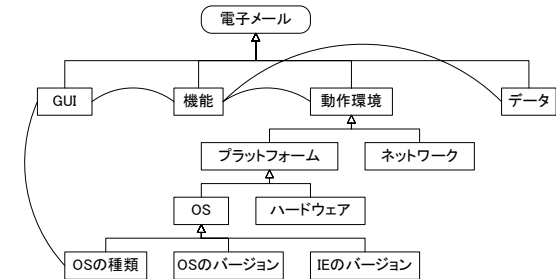
- » 一般的なテストのための記法
- » テスト分析モデルとテスト設計モデルを記述する
- » 個々のテスト技法で扱うものは記述しない

- テスト分析モデル

- » ビュー／フォーカス・クラス／関連によるテスト対象のモデリング

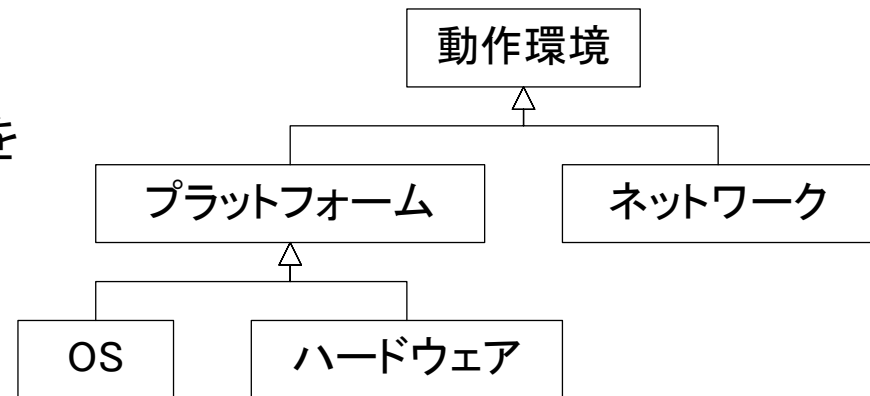
- テスト設計モデル

- » フォーカス・クラス、ズームイン/ズームアウト、
テスト項目数の概算によるトレードオフ
確定/暫定フォーカス・クラスによるリスクの明示

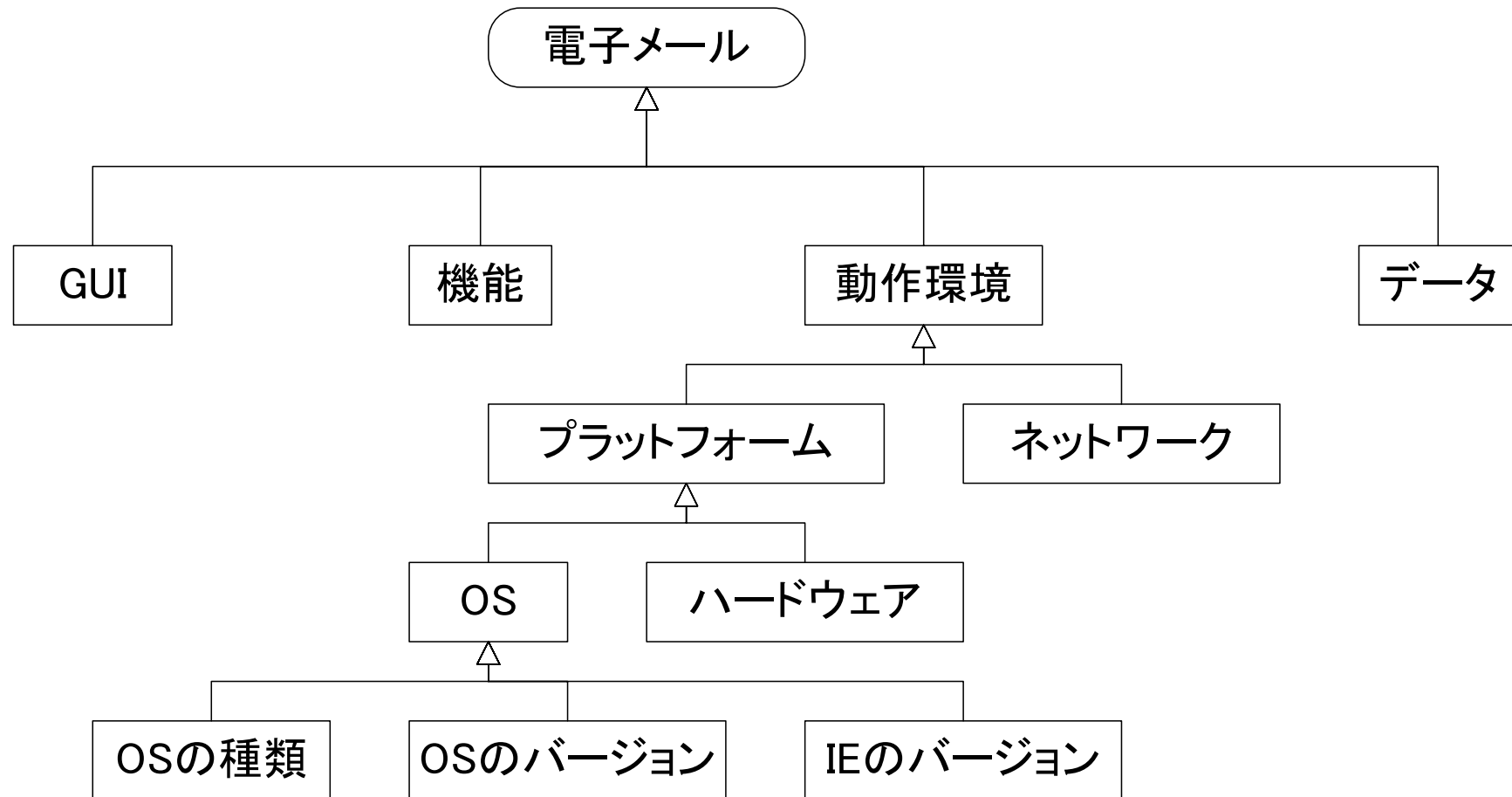


テスト分析:ビューの列挙・詳細化・体系化

- まず大まかなビュー(テスト観点)を列挙し、階層的に詳細化していく
 - 動作環境→プラットフォーム→OS→・・・
 - 各レイヤーの観点を「フォーカス・クラス」と呼ぶ
 - » 詳細化されたフォーカス・クラスは、テスト設計で同値クラスになる
 - » 詳細化の作業は、近くからモノを見る(ズームイン)ようなものである
 - フォーカス・クラスは階層構造を持つ
 - » フォーカス・クラスは「継承」する
 - » 白抜き矢頭の付き矢印で記す
 - 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

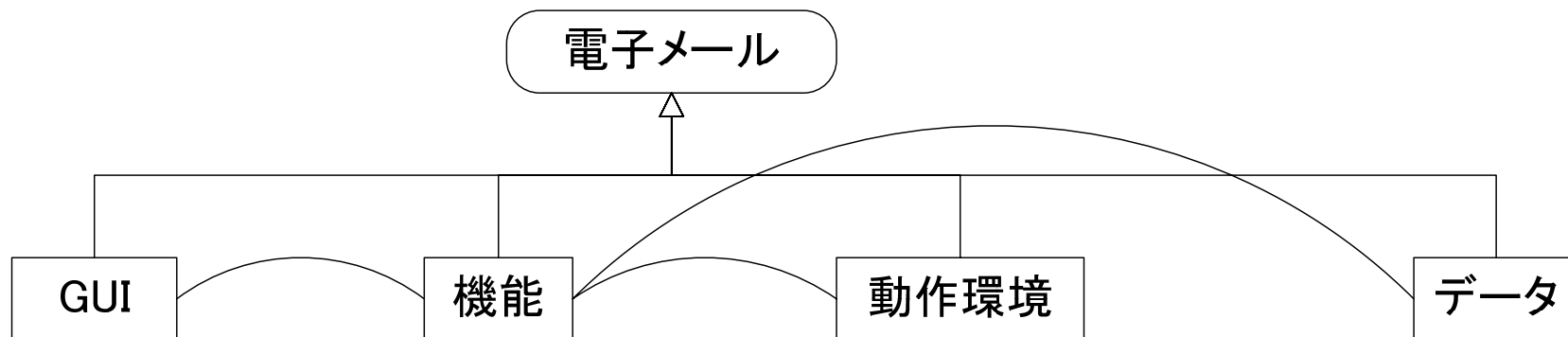


テスト分析:ビューとフォーカス・クラスの例

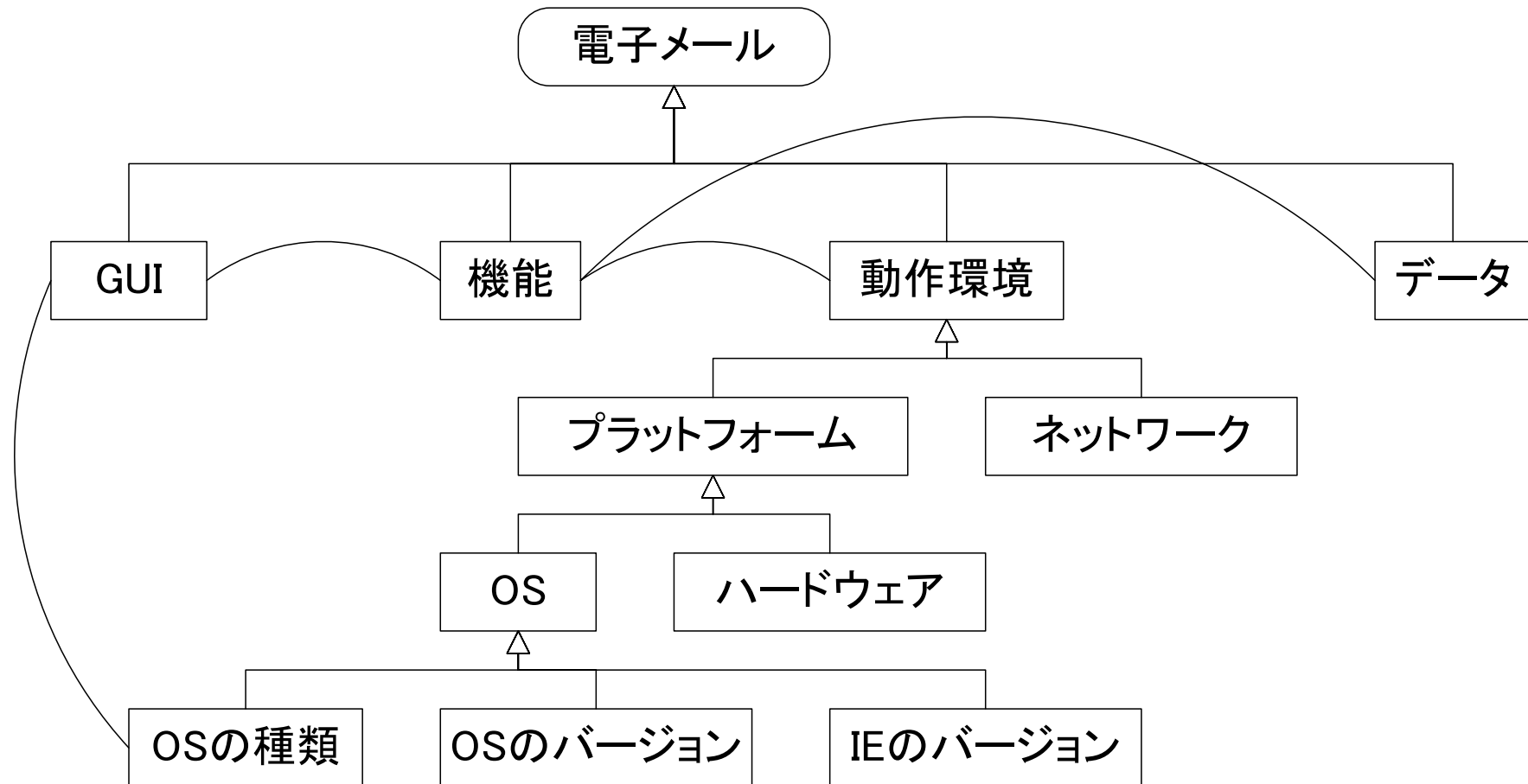


テスト分析：関連の検討

- 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



テスト分析モデルの例



テスト設計フェーズ・テスト実装フェーズ

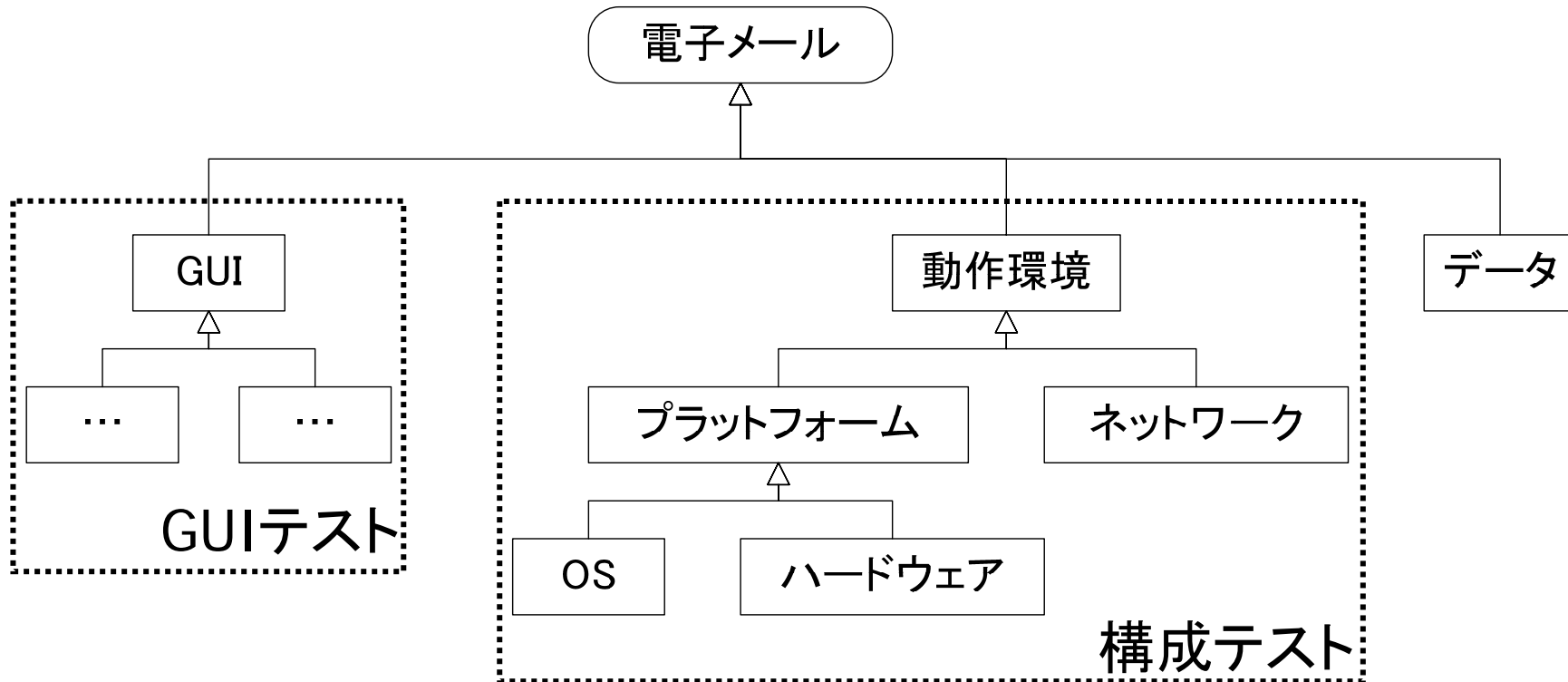
- テスト設計フェーズは3つに大きく分かれる
 - テストアーキテクチャ設計
 - » テストアーキテクチャ: テストを設計しやすいように、全体を整理し分割する
 - フォーカス・クラス設計
 - » 詳細化: フォーカス・クラスを段階的に詳細化し、具体的なテスト技法や同値クラスを導出し、網羅基準を定める
 - » 剪定: テスト項目数とリスクとのトレードオフを大まかに行う
 - » 確定: 間引いたテスト項目のリスクを明らかにし、可能であれば補填する
 - テスト詳細設計
 - » 既存の技法を用いてテスト設計を進めていく
- テスト実装フェーズ
 - テスト項目の集約・組み合わせ・詳細記述を行う

多段階の詳細化を行うことで
テスト項目の再利用を推進する



テスト設計: テストアーキテクチャ設計の例

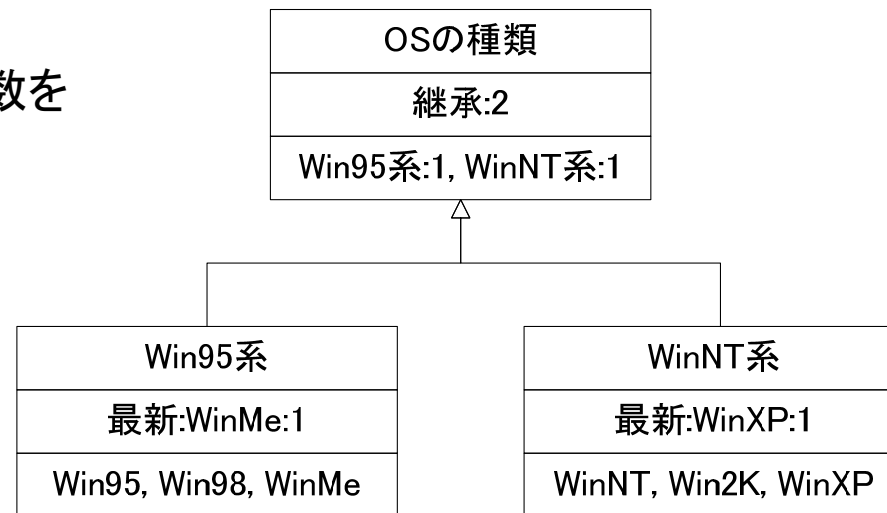
- カテゴリ型テストアーキテクチャ
 - 関連の少ないフォーカス・クラスのグループをテストカテゴリとしてまとめ分割していく



テスト設計：フォーカス・クラスの詳細化

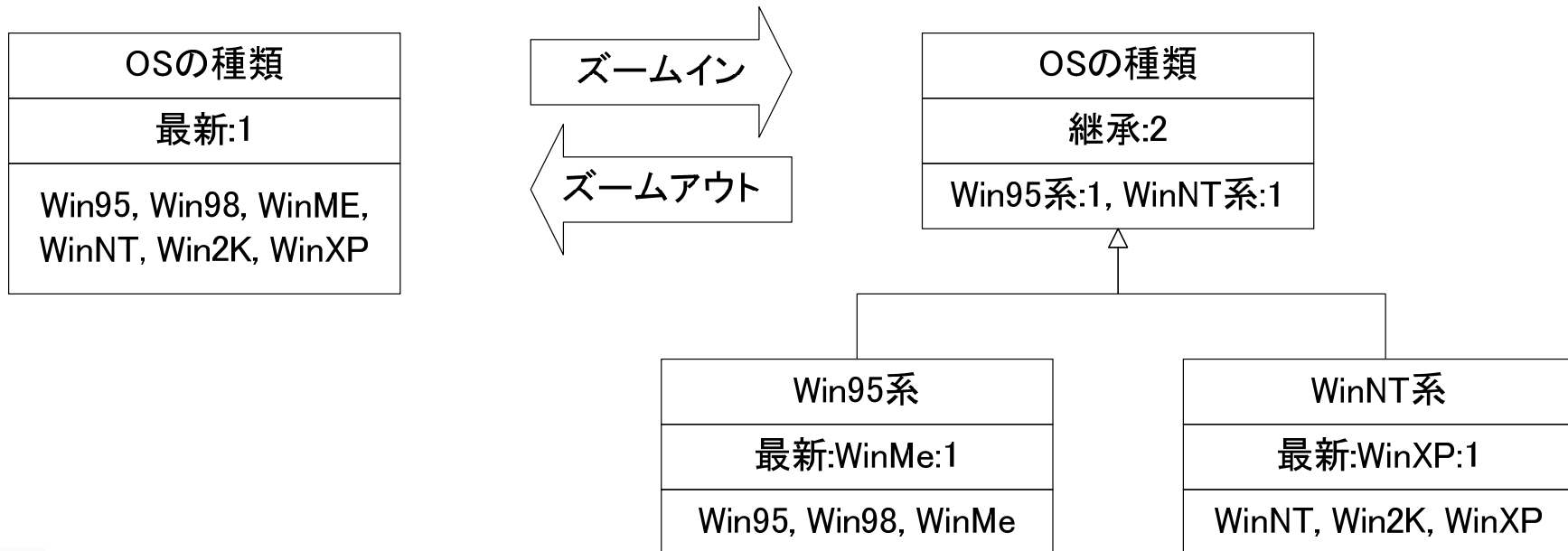
- フォーカス・クラスを詳細化しながら
網羅基準を定めテスト項目数を概算する
 - 網羅基準とテスト項目数を記述して概算する
 - テスト項目数を概算しやすいように
クラスメンバを記述する
 - 親クラスのテスト項目数は、
継承した子クラスのテスト項目数を
足し合わせる

クラス名
網羅基準と テスト項目数
クラスメンバ



テスト設計：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



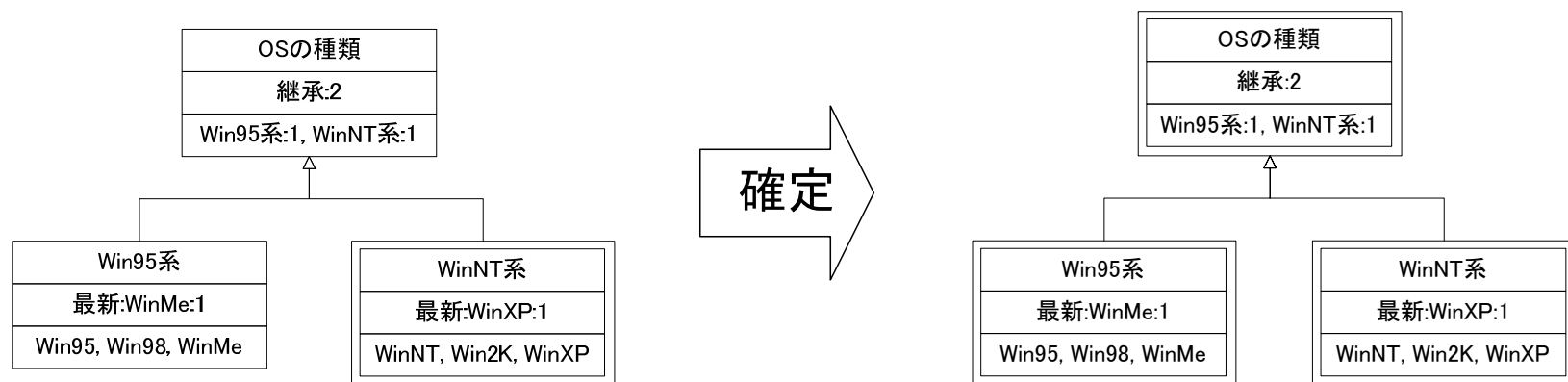
テスト設計：フォーカス・クラスと関連の剪定

- 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



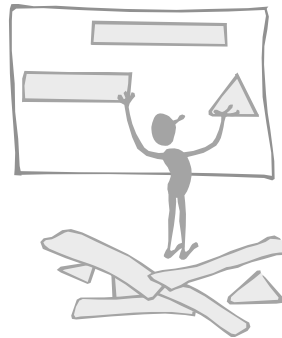
テスト設計:「確定」によるテスト漏れリスクの軽減

- 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



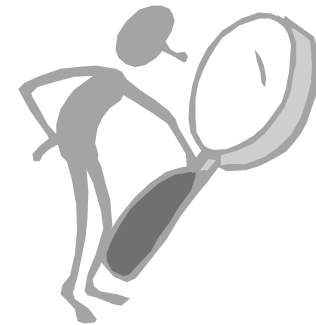
ビューに着目したテスト設計プロセスの概要

- **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**
 - 既存の技法でテスト設計する
 - » 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
 - » 自動実行ツール用のスクリプト作成を行う



テスト改善のためのテスト漏れの分析

- テスト漏れの原因を、ビューを軸にして分析し改善する
 - 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緩和のしすぎ
 - » ズームアウトのしすぎ
 - » 関連の組み合わせ基準の緩和・削除のしすぎ
 - 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し、バグの入り込みやすいハザード、分析、設計、実装のパターンリストを作る

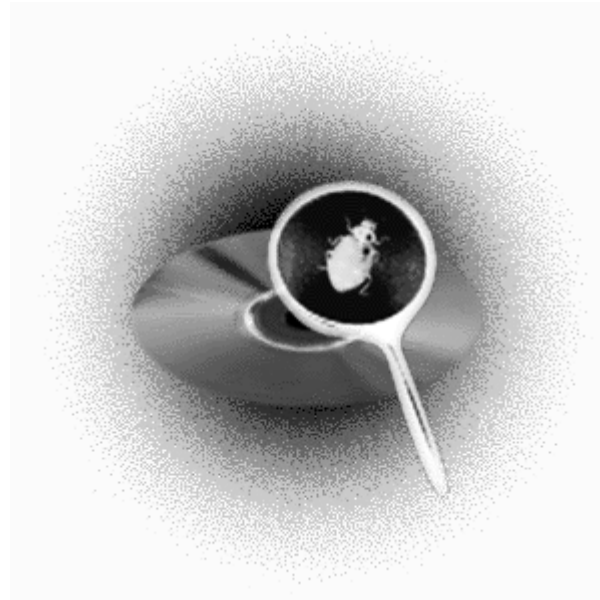


まとめ

- テストの「観点」は、テスト設計の本質である
 - テスト対象(及び含む世界)を、テストの立場からモデリングしたもの
- ビューに着目したテスト設計プロセスを紹介した
 - 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
 - » ビュー(フォーカス・クラス)間の関連を検討する
 - テストアーキテクチャ設計フェーズ: ビュー全体の整理と分割
 - 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／テスト実装フェーズ
 - テスト改善フェーズ: ビューを軸にしたテスト漏れの原因の分析と改善
- メリット
 - テストで考慮すべき観点を一覧でき、俯瞰的に整理できる
 - 全体のバランスを考えながら
テスト項目数とリスクとのトレードオフを検討できる
 - トレードオフに付随するテスト漏れのリスクを軽減できる



ご清聴ありがとうございました



電気通信大学 電気通信学部 システム工学科

西 康晴

nishi@se.uec.ac.jp

© NISHI, Yasuharu