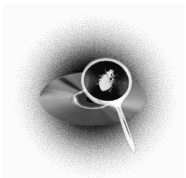
不具合の分析とフィードバックによるテストの設計



ソフトウェアテストシンポジウム2003
2003/3/6（木）
㈱エス・キュー・シー　西　康晴

Myersの呪縛

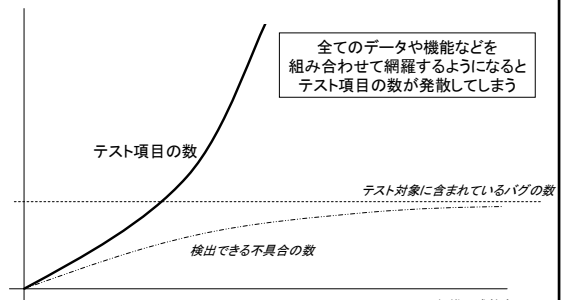
- 偉大なMyersは重要だが罪深い原則を提示した
  - 「一般に、プログラムのすべてのエラーを みつけることは、非現実的でもあり、 しばしば不可能でもある」
- これを聞いてテスト技術者は Myersの呪縛でがんじがらめになった
  - ならばテストの質を向上してもムダなのではないか
    - » 設計や分析の技術
    - » プロセス改善だけに取り組む
    - » あきらめる／悪循環に陥る
  - テストの質の向上が伴わない改善活動が増えて、 不具合の低減に結びつかない取り組みが増える
    - » ソフトウェア工学的活動全般に対する猜疑心を生むことになった
    - » テストの質を向上させようとする気概が失われた

Dr. Glenford J. Myers CEO, IP Fabrics. Leading roles in the 286, the 386 and the i960.

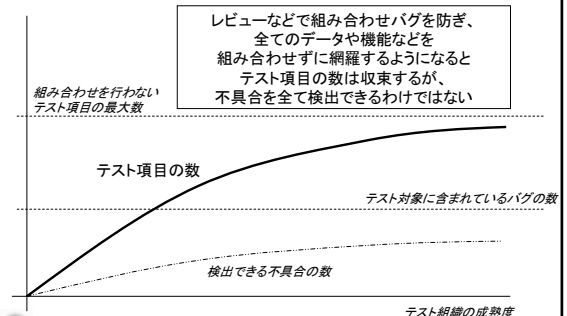
テスト項目数の発散の原因

- Myersの呪縛の本質的原因
  - ブラックボックステストで全ての不具合を検出しようとすると、 テスト項目数が発散してしまう
- ブラックボックステストでテスト項目数が発散する原因
  - 数値範囲のテスト項目は無限に存在する
    - » 1から10の範囲に存在する浮動小数点の数はとてもたくさんある
    - » 境界値分析を行えば、1つの数値範囲に対して4つのテスト項目でよい
    - » 範囲内が連続であることを保証するためには、レビューなどを行う
  - 組み合わせによるテスト項目は無限に存在する
    - » 機能やパラメータを組み合わせてテストしないといけない
    - » 組み合わせによってテスト項目数は指数関数的に増えていく
    - » 直交配列表はAll-pair法を用いる場合もある

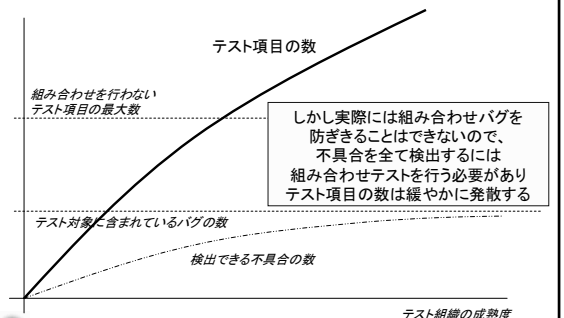
組み合わせによる急激な発散



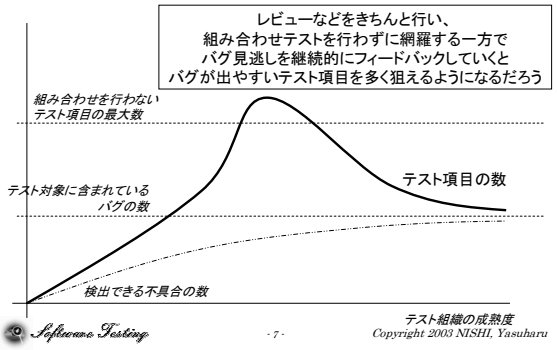
組み合わせの回避によるバグの見逃し



組み合わせによる穏やかな発散



不具合分析とフィードバックによる発散の抑制



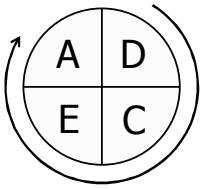
バグ見逃しのフィードバックによるテストの改善

- バグ見逃しをフィードバックしてテストを改善するためには、知識を再利用する必要がある
  - 組み合わせバグには傾向があるので、そこを狙えばよい
    - » バグは偏在する
    - » バグの多そうな部分をあらかじめ理解しておけば、効率的にテストできる
  - オブジェクト指向におけるクラス階層の概念を流用する
    - » クラス／インスタンス／継承構造
  - 組み合わせバグの見逃しのクラス階層ツリーを構築する
    - » 組み合わせバグのパターンを抽象化・構造化して蓄積する
    - » 蓄積したパターンに照らし合わせてテスト項目を設計する
  - テストを見逃しやすい兆候とバグを作り込みやすい状況を抽出する
    - » どのようにしてバグが発生したか／バグが発生する状況や条件
    - » 期待結果と実施結果／バグが発生するメカニズム

フィードバックによるテスト設計の改善

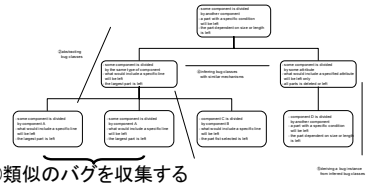
- バグ見逃しをフィードバックすることによって
  テストの設計を改善し、テスト項目数の発散を防ぐ

1. テストを設計する（Design）
   - 初回は組み合わせずにテストする
2. バグ見逃しを収集する（Collect）
3. バグ見逃しが起きやすい
   ポイントを抽出する（Extract）
4. 抽出したポイントを
   テスト設計に追加する（Add）



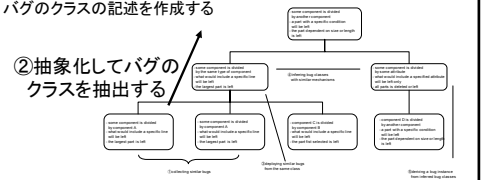
バグ見逃しのナレッジマネジメント

- バグ見逃しのクラス階層の構築手順 (図6/P.52)
  ①類似のバグを収集する
  - » バグの事例をインスタンスとして収集する
  - » バグの記述を書き直す
  - » 似たようなバグに分類する



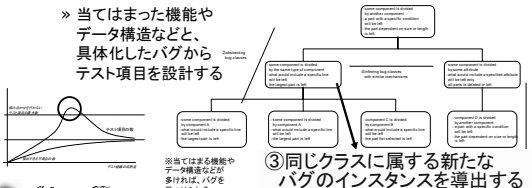
バグ見逃しのナレッジマネジメント

- バグ見逃しのクラス階層の構築手順 (図6/P.52)
  ②抽象化してバグのクラスを抽出する
  - » 似たようなバグとして分類されたバグを、本質的に共通な部分と異なる部分に区別する
  - » 共通な部分を残し異なる部分の記述を代名詞に置き換えることで、バグのクラスの記述を作成する



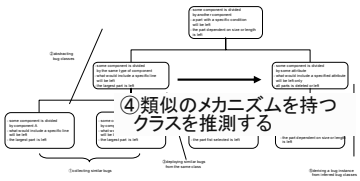
バグ見逃しのナレッジマネジメント

- バグ見逃しのクラス階層の構築手順 (図6/P.52)
  ③同じクラスに属する新たなバグのインスタンスを導出する
  - » クラスの記述に当てはまるテスト対象の機能やデータ構造を探す
  - » 当てはまったクラスの示すメカニズムや兆候を持つバグを具体化する
  - » 当てはまった機能やデータ構造などと、具体化したバグからテスト項目を設計する



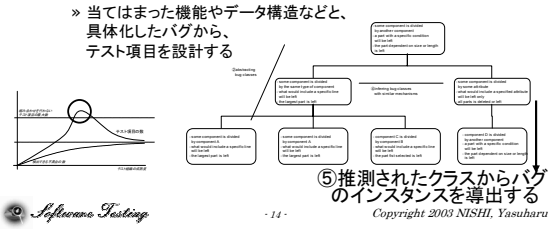
バグ見逃しのナレッジマネジメント

- バグ見逃しのクラス階層の構築手順 (図6/P.52)
  - ④類似のメカニズムを持つクラスを推測する
    - » 親クラスを抽出し、その親クラスから新たに可能な子クラスを論理的に推測する



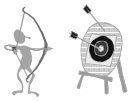
バグ見逃しのナレッジマネジメント

- バグ見逃しのクラス階層の構築手順 (図6/P.52)
  - ⑤推測されたクラスからバグのインスタンスを導出する
    - » クラスの記述に当てはまるテスト対象の機能やデータ構造などを探す
    - » 当てはまったクラスの示すメカニズムや兆候を持つバグを具体化する
    - » 当てはまった機能やデータ構造などと、具体化したバグから、テスト項目を設計する

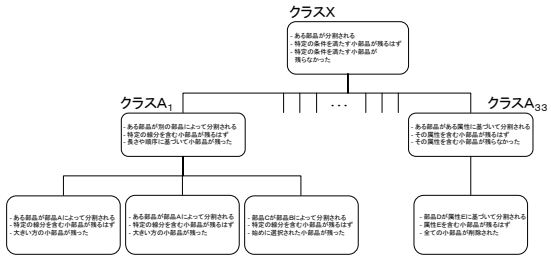


適用例：CADソフトウェアの仕様レビュー

- CADソフトウェアに対する仕様レビュー
  - データオブジェクトの組み合わせのバグが検出できなかった
    - » データオブジェクトの種類は20万件以上
  - テスト項目の設計もレビューでの指摘項目の設計も同じ
- 本手法を適用した
  - 以前に仕様レビューで見逃した3件の不具合から知識ツリーを構築した
    - » 過去の仕様レビューで見逃した他の不具合が検出できるかどうかで効果を測定することにした
    - » 知識ツリーを構築した結果、クラスを32個抽出した
  - 4件の不具合が該当した
    - » 7件の潜在的不具合を指摘可能な指摘項目が設計できた



適用例：CADソフトウェアの知識ツリー



知識の抽象化に関する注意点

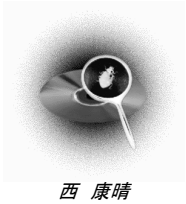
- 抽象化は慣れないと難しい作業である
  - 始めは時間がかかる
    - » 始めはバグ1件の抽象化に2時間かかる
    - » 慣れると1件あたり5分でできるようになる
  - 過度の抽象化は避ける必要がある：S/N比が落ちる
    - » 原因はスキル不足だ
    - » そもそも無理なプロジェクトだ
    - » 教育やトレーニングが充実していないのが原因だ
    - » ソフトウェア産業の構造的問題だ
    - » 不況がすべて悪いのだ
- 抽象化をする目的を常に忘れないようにする
  - テスト設計でバグを検出するのが目的
  - バグ見逃しの原因の再発を防ぐことが目的
    - » 原因が繰り返されるからバグ見逃しも繰り返される
    - » バグ見逃しは、人間の認知プロセスが持つ弱点の具現化である
    - » テスト項目をレビューし改善していき、組織全体での再発を防ぐ

まとめ

- Myersの呪縛から解き放たれる必要がある
  - 組み合わせによるテスト項目数の発散を回避するのが課題である
- バグ見逃しをフィードバックしてテスト設計を改善することは一つの方策である
  - バグ見逃しの知識を抽象化し構造化して再利用する
- 仕様レビューに適用したところ、効果がみられた
  - CADソフトウェアを対象に適用した

テスト項目をレビューし改善していくことで
バグ見逃しを防ぐことが重要である

何かご質問はありませんか？



西　康晴
nsh@mtd.biglobe.ne.jp