テストの“質”評価と欠陥分析による ソフトウェア信頼性評価

－振り返り型品質保証 から 予測型品質管理へ－

Sopen（ソフトウェア技術者ネットワーク）
ソフトウェアメトリクスSIG
堀　明広

はじめに

- ＜悪魔の証明＞:「ない」ことを証明することの難しさ
  - 「あることの証明」は、特定の「あること」を一例でも提示すればすむが、「ないことの証明」は、厳密には全称命題の証明であり、全ての存在・可能性について「ないこと」を示さねばならないためである。すなわち、「ないことの証明」は「あることの証明」に比べ、一般に困難である場合が多い。
  - 例:「第二次世界大戦では鎖鎌を武器にした兵士がいた」
  - これを「ないこと」として否定する場合は第二次世界大戦に参加した人間全てを調べなければいけない。しかしそのような調査は実行不可能である。一方、一人でも鎖鎌を使った人間がいることを証拠により裏付けられれば、「あること」の証明は可能である。

Wikipediaから一部引用

テストでそのプログラムにバグがあることは証明できるが，バグが存在しないことは証明できない。

ソフトウェアテストとバグについても，同じ命題を抱えている。

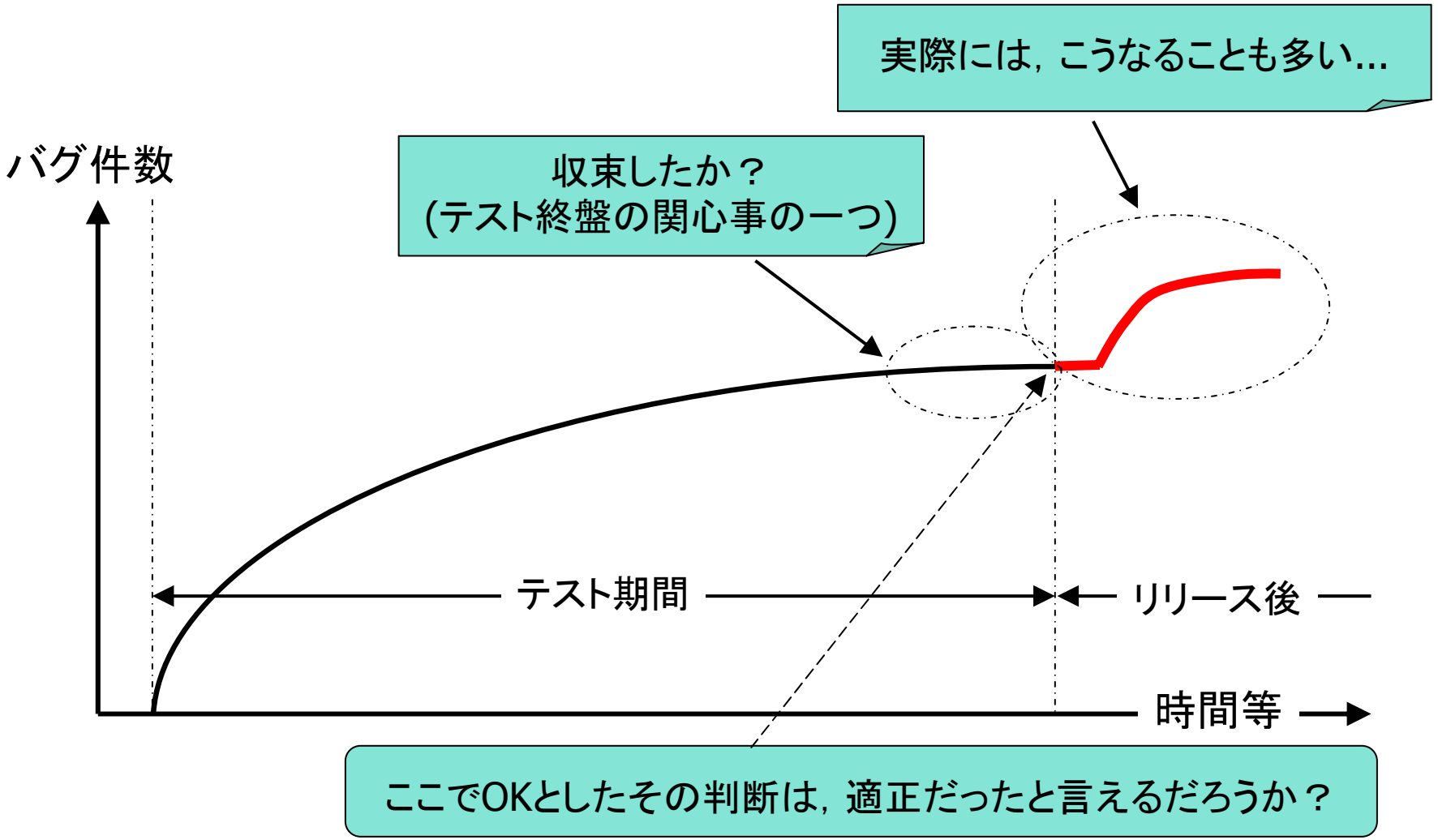
信頼度成長曲線の前提条件

- ゴエル･オクモトモデルの場合：
  - エラーは互いに独立である。
  - 発見されるエラー数は残存エラー数に比例する。
  - 各エラーは同じ発見のされやすさ（確率）を持つ。
  - 発見されたエラーは完全に除去される。

バグを検出するペースはテストフェーズのある時点でピークに達し，
その後なだらかになることは，直感的にも理解できる

成熟した組織では，信頼度成長曲線とバグ成長曲線が
マッチすることが多いとも聞く。　だが，しかし。。。

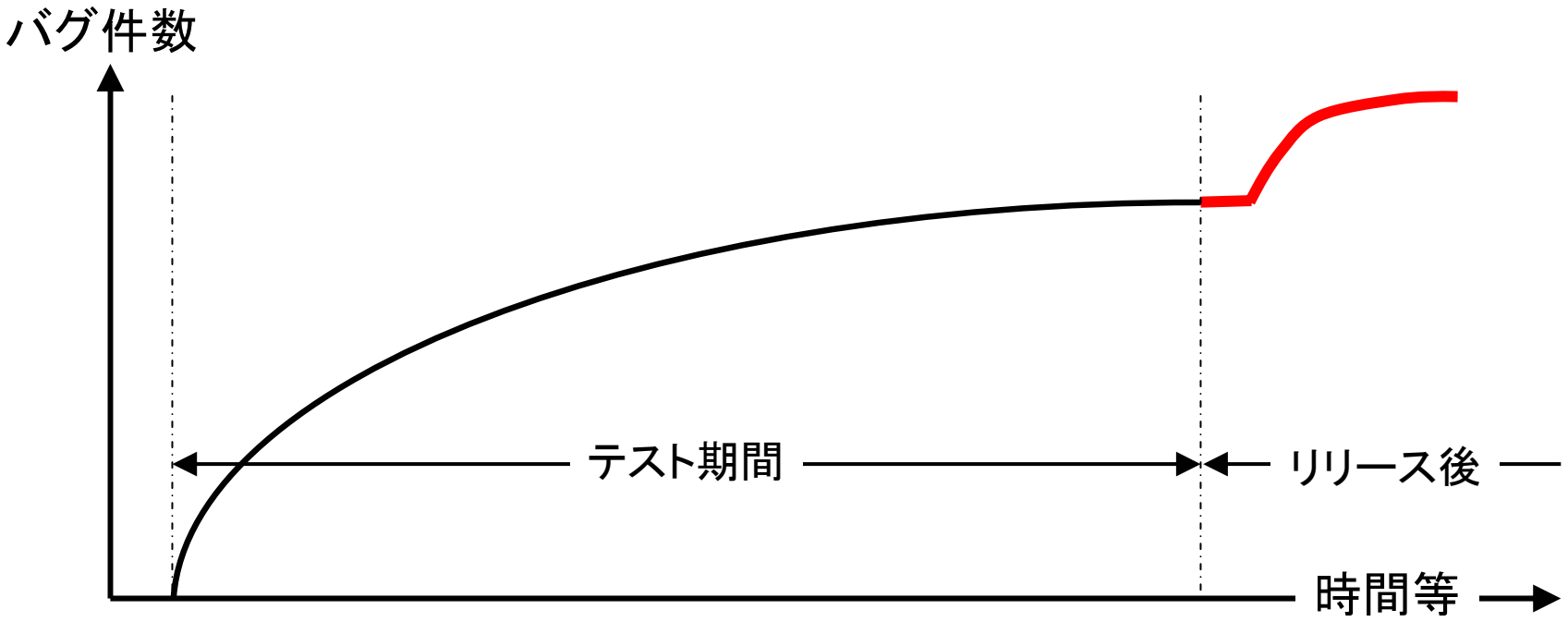
信頼度成長曲線の悩み



「収束した／しない」の前に，考えなければならないことがあるのではないだろうか？

信頼度成長曲線は，何を表しているのか？

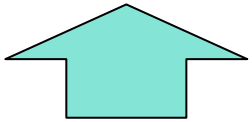
- 信頼度成長曲線は，作り込んだバグを取り除けたペースを表したものに過ぎず，ソフトウェアの信頼性を直接表したものではない。
- 信頼度成長曲線は，そのソフトウェアをテストした**観点・範囲内**でのものに過ぎない。



「テストしてもバグが発見されない」のは？

信頼度成長曲線が収束している

そのソフトウェアの品質があるレベルにまで到達しているが故なのか？
それとも，テストが甘いからなのか？



この両者を判別するには，どうすれば？

信頼度成長曲線と「テストの観点」

≪思考の出発点≫

- テストを実施する前に，そのテストの”バグ検出力”はどの程度のものなのか。何時，どんなバグを検出しようとしているのか，明らかにしたい。
- 信頼度成長曲線が，どういった意図と経緯で描き出されたものなのか，その根拠を明らかにしたい

「どんなテストを実施して，
その結果，バグはどのようなペースで抽出できているのか」

信頼度成長曲線に，「テストの観点」というファクターを付け加える

信頼度成長曲線に「テストの観点」をリンクさせる

≪本稿で提案する手順≫

①ソフトウェアテストの観点の洗い出しと，
テスト設計技法の決定

②テスト観点毎のテスト件数の見積もり

③設計したテスト項目の実施順序の決定と，それぞれのテスト観点で検出されるバグ数の見積もり

④テスト消化ペースとバグ抽出ペースの予測を時系列に並べてグラフ化

⑤テストの"質"の評価

⑥テスト消化とバグ発生状況の予実績を分析・対策

①テストの観点の洗い出しと，テスト設計技法の決定

- 目的
  - 基本的なところでテストの観点ヌケを生じさせないため

- テストの観点の例（SQubok（Guide to the Software Quality Body of Knowledge）より）
  - 境界値テスト，デシジョンテーブルによるテスト，原因結果グラフによるテスト，状態遷移テスト，ランダムテスト，モデルベースドテスト，制御フローテスト，データフローテスト，トランザクションフローテスト，コールフローテスト，エラー推定テスト，ミューテーションテスト，運用プロファイルによるテスト，ローカライゼーションテスト，ユーザー環境シミュレーションテスト，整合性確認テスト，オブジェクト指向テスト，GUIテスト，データベーステスト，並行プログラムのテスト，プロトコル適合性テスト，セーフティ・クリティカルシステムのテスト，直交表を用いたテスト，リスクベースドテスト，，，

"〇〇テスト"と名付けられているテストは，そのテストに持たせている観点や実施フェーズ，テスト設計技法を表意したもの

これから行うべきテストには，**どんな観点が必要であるか**，そのテストはどんな技法で設計すべきか，「テスト計画・設計のアーキテクチャ」を描き出す。

テスト計画・設計のアーキテクチャ

- ドメインやそのプロダクト特性，組織の形態に依って異なる。
  - 例えば:
  - 組織共通のソフトウェアテスト計画策定のガイドラインを策定
  - このガイドラインに則ってテスト計画書を策定
- テストの観点を洗い出すと伴に，テスト設計の方法・技法も検討・決定
- ≪留意点≫
  - ガイドラインを盲信して「考えること」を忘れてはならない！
  - 検討したテストの観点は，文書化することが重要！
    - 曖昧さの排除，ヌケの防止
    - 関係者との共有，ノウハウの伝承と継承
    - プロセス改善の媒体
    - そのテスト項目が，何を狙っているものか，どうやって設計したのか（これはノウハウの固まり。大切にしなければもったいない）

②テスト観点毎のテスト件数の見積もり

- 「実施すべきテストの観点」を抽出したら，それらを整理し，各々の観点で，テストの件数を見積もる。
  - テスト観点とテスト対象（機能，コンポーネント等）について，テスト件数見積もる。
- 実際のテスト項目は，複数の観点を併せ持っているものが多い。
- 言うまでもなく，テストにかけるリソース（人，モノ，金，時間）は有限。最小のコストで，最大の観点を網羅し，より多くのバグを抽出できるように，最適化する。

③テスト項目の実施順序とテスト観点毎のバグ数の見積もり

- テストの実施順序
  - テストの優先順位をデザインするため
  - 重要なバグを早期に抽出することを念頭に置いて計画
  - リグレッションテストや負荷テスト等を，テストフェーズのどのタイミングで発動するかも計画
  - ここはRisk Based Testingの思想を使う
- バグ数の見積もり
  - そのテスト観点の意図するところが，バグを検出するためのものであるのか，正常動作することを確認するためのものであるか，識別するため。
  - テスト対象のソフトウェアの弱点は何処と想定しているか，明確化するため

テスト観点毎にバグ数を見積もるには(その1)

- 過去プロジェクトの実績値を用いる
  - 過去のプロジェクトで，どんな観点のテストで，どれだけのバグを抽出できているか，標準値を設定
  - 現行プロジェクトの特性と変動要因を加味して見積もる。
- 過去プロジェクトのデータが整備されていない場合には
  - テスト観点毎にバグ件数を抽出する期待度を設定することで代替えする。(開発技術者，テスト技術者の内観で良し)

≪イメージ≫

テスト観点	テスト件数	バグ抽出する期待度
機能テスト	○○件	0.5
競合テスト	○○件	0.3
構成テスト	○○件	0.2

テスト観点毎にバグ数を見積もるには(その2)

- 過去プロジェクトでソフトウェア規模とバグ総件数のデータがある場合は，以下の手順でテスト観点毎のバグ数を見積もる
  - 過去プロジェクトでソフトウェア規模とバグ総件数から，バグ密度(件/SLOC等)を算出。
  - 変動要因を加味して現行プロジェクトのバグ密度を設定し，総バグ件数を算出
  - 上記で算出した総バグ件数を，「テスト観点毎のバグ抽出する期待度」に則して配分

テスト観点毎にバグ数を見積もるには(その3)

≪イメージ≫

過去プロジェクトデータ:
規模　　:100kstep
総バグ数　:200件

⇒ バグ密度:2.0件/kstep

現行プロジェクトデータ:
規模　　:150kstep
バグ密度　:2.2件/Kstep

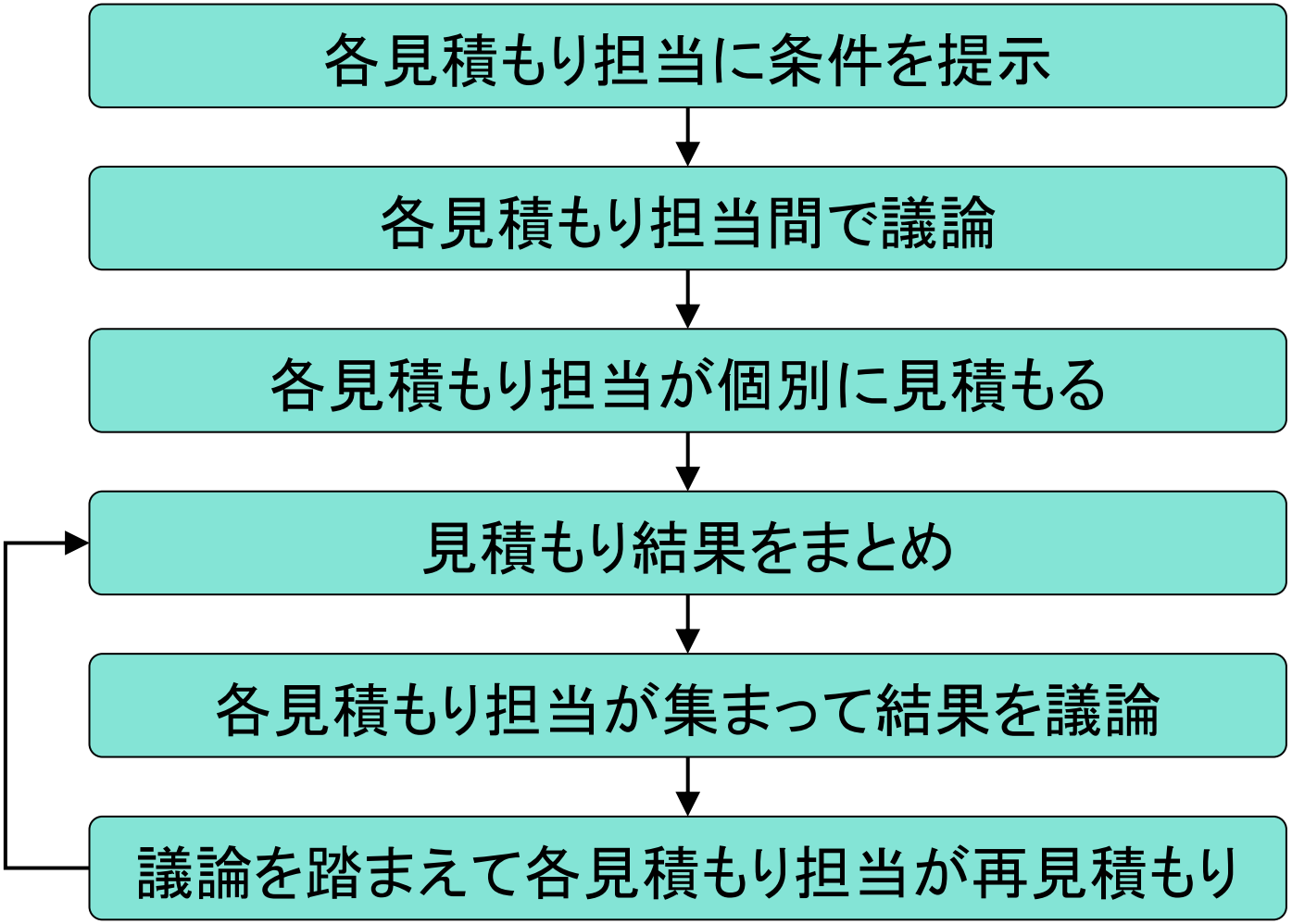
⇒ 総バグ数見積値:330件

デルファイ法を用いると効果的

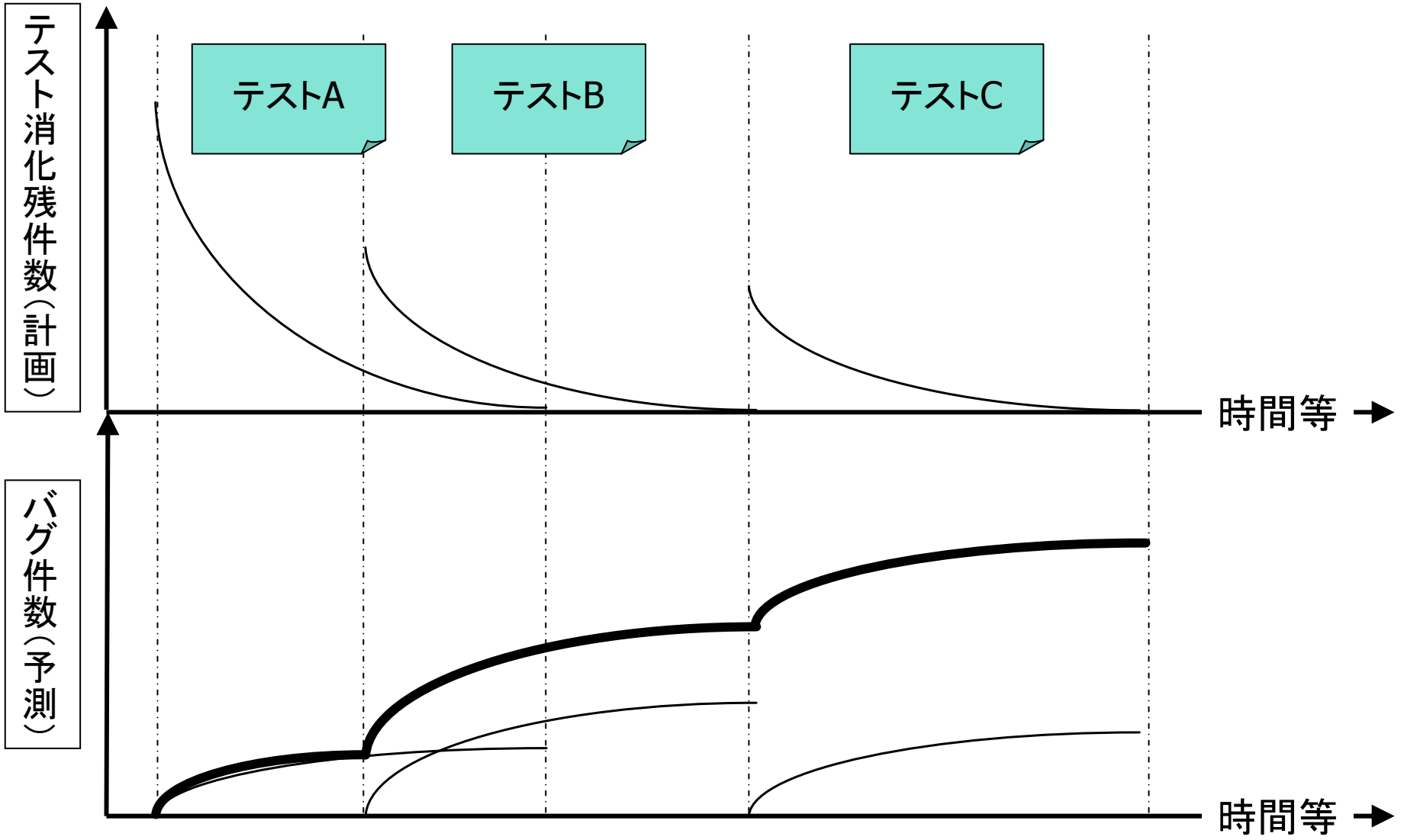
テスト観点	テスト件数	バグ抽出する期待度	バグ数見積値
機能テスト	○○件	0.5	330×0.5=165件
競合テスト	○○件	0.3	330×0.3= 99件
構成テスト	○○件	0.2	330×0.2= 66件

＜ご参考＞デルファイ法：見積もり手法の一種

複数人数で見積もりし，それを付き合わせて見積もりを煮詰めていく方法



④テスト消化ペースとバグ抽出ペースをグラフ化



⑤テストの"質"の評価

プロジェクトリーダ，開発リーダ，QA，PMO等

- 以下項目について，関係者によるレビューを実施
  - どんな観点のテストを，何処に，どれだけ実施するか
  - どんなテストに，どんなテスト設計技法を用いるか/用いたか
  - テストの組み立て（実施順序，発動時期）は妥当か
  - テスト観点を粒度としたバグ抽出件数とペースは妥当か

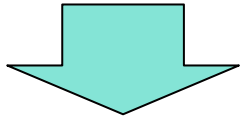
規模が大きいプロジェクトは，
進捗に合わせて分割して実施する

どの時期に，どんな観点のテストをどれだけ実施するか，
その結果，バグはどんなペースで抽出するか。

テストを実施する前に，テストの“質”（観点・項目数，組み立て）を評価し，
バグ抽出するペースをテスト観点毎に積み上げて予測する。

⑤テストの"質"の評価（続き）

- ゴエル・オクモトモデル：
  - トップダウン方式によるマクロモデルの位置づけ



- テスト観点ベースのバグ見積もり（本稿）：
  - テスト観点毎に詳細化した，ボトムアップ式のWBSモデル的な方法で予測を行うもの
  - 又は，テスト観点の粒度でバグ抽出の期待度を設定し，それに応じてバグ総数を配分するもの

⑤テストの"質"の評価（続き）

- テストの“質”の評価（テスト計画レビュー）の効能
  - テスト観点のヌケを防止できる。
  - プロジェクトの環境・状況を共有することにより：
    - テストの観点，テスト設計の内容が充実する
    - テストの組み立て（実施手順）の最適化が図れる。
  - どの時期に，どれだけバグを抽出するか見込むことにより，リソース配置の最適化を図ることができる。
  - そのソフトウェアの弱点がどこにあるかを共有でき，それぞれの立場で対策を講じることができる

以上により，テスト消化とバグ抽出が想定どおりであるか否か，
進捗管理する下地を作ることができる。

⑥テスト消化とバグ発生状況の予実績を分析・対策

テスト実施フェーズに入ったら，狙ったとおりにバグを検出できているか，
モニターする。

- テスト観点毎に，予測したバグ数と実績値を比較して，多いか／少ないか
  - 多すぎる場合：
    - 設計品質・実装品質がそのテストに耐えられていない
      - 問題のある箇所に立ち返って対策し，類似バグは机上で潰す。
      - テスト中にソースコードが変更されることにより，新規にバグが作り込まれたりデグレードしたりするのを防止する。
  - 少なすぎる場合：
    - テストが甘いことを疑う
      - そのテスト観点は妥当であったか（その観点を設定した根拠は？）
      - テスト観点は妥当であっても，テスト項目の設計方法に問題は無いか？（テスト設計プロセスに着目）
      - オペレーションに何か問題はないか？ 見過ごしていないか？

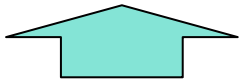
⑥テスト消化とバグ発生状況の予実績を分析・対策（続き）

バグ数だけに気を取られず，バグの内容も重要する

5W1H	欠陥除去編	欠陥作り込み編
What	どんな欠陥を検出したのか	どんな欠陥を作り込んだのか
When	何時検出したのか	何時作り込んだか
Where	どこのテストで検出したのか	どこに作り込んだか
Why	なぜ事前に検出できなかったのか	なぜ作り込んだのか
How	どのようにして検出したのか	どのようにして作り込んだのか
Who	誰が検出したのか	誰が作り込んだのか

⑥テスト消化とバグ発生状況の予実績を分析・対策（続き）

- そのバグは：
  - そのテストで発見したことで，妥当なのか？
    - もっと上流の（例えば単体テスト）のバグではないのか？
    - その観点に相応しいバグか？
      - 偶然発見したものではないか？
      - 別の観点で検出されるものが，先に検出されたものなのか？
      - 別の観点でも網羅できてなく，幸運に恵まれただけなのか？
  - 他者から発見された場合，それは妥当なのか？
    - 自分で発見すべきものなのか？
    - 自分たちで見過ごしていた場合，テスト観点・テスト設計・オペレーションの何処に問題があったか？



テストの観点をベースに，分析を細かく実施し，
開発とテストにフィードバックをかけ続ける

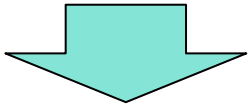
欠陥分析のフィードバック

- そのテスト観点で抽出すると見込んだバグは，実際に検出できているか？
- そのバグ件数は，当初見込んでいた件数に対して見合っているか？

テスト実施中に，テスト観点の粒度で分析し，
テストの観点・組み立て，開発のウィークポイントにきめ細かくフィードバックし，
出荷までに品質の安定化を図る

本稿で提案した手法の主旨

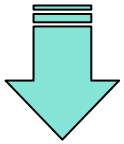
- 本稿で提案した手法の主旨は，信頼度成長曲線を，計画どおりにプロットすることが目的ではない。
  - プロジェクトが置かれた環境・状況・特質を踏まえ，過去の経験に基づいて以下を明確化
    - どんな取り組みが必要か（テストの観点，組み立て）
    - どんなバグを，どうやって，どれだけ抽出するか
  - 特異点を抽出する足がかりを作り，有効な対策を講じてマネジメント



品質確保の大ゴールを達成する可能性を高めることが目的

本稿で提案した手法の主旨（続き）

「テストを実施して，バグが何件抽出された」



「何の目的で何をテストし，どれだけのバグを，何時，どれだけ抽出するか」
を計画して，その進捗をマネジメント

振り返り型の品質保証から，予測型の品質管理へ！

テストの"質"の評価

「何の目的で何をテストした結果，
どれだけの，どんな内容のバグを，何時，どうやって，誰が抽出した」

↓

データを積み重ねて蓄積し，分析

↓

どんなテストが必要か，それによって，どんなバグを抽出できるのか，
ノウハウを蓄積

↓

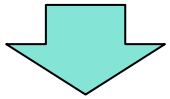
テスト計画・テスト設計を底上げ

↓

テストを実施する前に，そのテストで抽出できるバグの予想(テストの質)を評価

テストによる「テストの改善，開発の改善」

- テストは，バグを検出することだけが目的ではない
  - テスト計画とテスト設計を，データを蓄積して進化させる



- このノウハウを，開発プロセスにフィードバック
  - こんなバグを作り込まないためには，どんなアーキテクチャが必要か
  - こんなバグをレビューで早期に抽出するには，どんな方法があるか

テスト計画・テスト設計のノウハウを，
設計技法・レビュー・インスペクションの熟成に展開

最初から正しく作り，バグを早期に抽出し，テストで効率的に絞りに絞る

開発技術者，テスト技術者，SQA，SEPGが一体となった，
地道なプロセス改善の継続

今後の課題

- テストにはどんなものがある？
  - 様々な文献で，様々なテスト手法が提案されている
  - これらをまとめ，テストはどんな構造を持っているのかをまとめ，テスト計画・設計の土台を検討したい。
- バグはどんな構造を持っている？
  - バグにはどんなものがある？
  - 例えば，"設計モレ"と"設計不良"はどう違うのか？
  - バグは人間の“失敗”の産物か？ 失敗学の構造は応用できないか？
- テストの観点とバグの構造をリンクさせる
  - 考察を進め，テストのアーキテクチャ構築・テスト計画・テスト設計・欠陥分析が，より緻密に行えるようにする。
  - プロセスの改善ポイントを具体的に抽出する土台を作る。

＜終わりに＞品質に，悩み続ける

- ソフトウェア品質を向上させるのには，特効薬は存在しない。
  日常的に改善を一歩ずつ積み重ねるより他に，道は無い。
  （今のところ）

- 品質に悩みは尽きない。
  しかし，悩み続けることが，大切なのではないだろうか。
  悩み続けるからこそ，次に，一歩前に進めるのではないだろうか。

S+open http://www.s-open.net/