定量的ソフトウェアテスト完了判断基準の一考察

○ 堀 明広	パナソニックモバイルコミュニケーションズ
仲 孝浩	シャープビジネスコンピュータソフトウェア
向井 清	住商情報システム
金子 喬	日立システムアンドサービス

ソフトウェアのテストに明快な終わりはない

- 良く言われること
  - テストでバグがあることは，証明できる
  - テストでバグがないことは，証明できない
- テストを続けていれば，バグは出てくるもの
- 経済的な理由から，どこかで何かをきっかけに，テストの継続をあきらめなければならない

テストの「完了」を適切に判断するには，どう考えるべきか？
定量的尺度を使ってテスト完了判断を行うには何が必要か？

検討するテストフェーズ

何れの開発モデルも，テストは最下流に位置する

完了判断
（顧客にリリースする直前）

上流 → 下流

＜検討対象＞

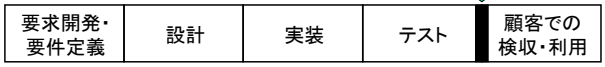
「Verification & Validation」の"Verification"をメインに据えて，テストの完了判断を定量的に行うための考え方，前提条件を探る

「テストを完了した」とはどういうことか？

＜本稿の考え方＞

そのテストで目的としていたものを達成していると言える状態

完了判断
（顧客にリリースする直前）



ここのテストの目的は何？

テストの4つの目的

従来から言われている目的

- ソフトウェアに潜むバグを検出し，修正
- 求められる要件を満足していることを確認

- バグの発生状況から，未だ潜むバグの存在を推測し品質の成熟度を判断
- バグの作込み要因と，より上流の検証フェーズでの流出要因を分析して，開発プロセスの問題を特定・対策

情報を収集し，アクションにつなげる活動とした捉え方

完了判断基準とは

そのテストの目的を達成しているか，判定する材料

- 本稿のアプローチ
  - テストの目的をゴールに据えて，メトリクスを設計・検討するためのGQMパラダイムに当てはめる
  - 定量的ソフトウェアテスト完了判断基準を構築

GQMパラダイムとは

■Basili教授らによって提案されたソフトウェア測定 及び 評価のための総合的な枠組み
■GQMを階層構造で表すことにより、測定 及び 評価目的を明確化

- 測定の目標(Goal)を明確化
- その目標の達成方法 又は 評価方法を質問(Question)の形式で記述
- その質問に定量的に答えるための測定量(Metric)を定義

ソフトウェアテストのGoalとQuestion

- <G1>ソフトウェアに潜むバグを検出し，修正
  - G1Q1:妥当な質と量のテスト項目を設定
  - G1Q2:設定したテスト項目を実行
  - G1Q3:検出したバグを修正
  - G1Q4:検出したバグを正しく修正していることを確認
- <G2>要件を満足していることを確認
  - G2Q1:要件を網羅するテスト項目を設定
  - G2Q2:プログラムの全機能を動作させるテスト項目を実行
- <G3>バグ発生状況から品質成熟度を判断
  - G3Q1:予め想定したバグの数だけ，バグを検出できている
  - G3Q2:信頼度成長曲線が収束している
- <G4>開発プロセスの問題を特定・対策
  - G4Q1:バグの発生状況を，開発プロセスにフィードバックしている

テスト目的 ブレイクダウン ⇨ 必要なメトリクスを設計 ⇨ 完了判断基準 構築

G1Q1:妥当な質と量のテスト項目を設定

- テストの質をどう測るか？
  - 自分たちは何をテストしようとしているのか？
  - 網羅すべきテストのカテゴリを計画し，それに沿ってテスト設計できているか？

カテゴリ別テスト件数

- テストの量は十分か？
  - テスト対象ソフトウェアの規模に見合う量か？

テスト件数と
プログラム規模との比率

G1Q1:妥当な質と量のテスト項目を設定(続き)

- テストの取捨選択
  - 意図したとおりにテストを設計し，実行することが困難な状況もある(テストにかけられる要員，時間)
  - 合理的にテスト項目を間引く

■カテゴリ別テスト件数
■テスト件数とプログラム規模との比率
■実施しないテストカテゴリの明確化，リスク対応策

G1Q2:設定したテスト項目を実行

＜着目点＞
実際に実行したテスト項目数

- 予め計画・設計したテスト項目が，テスト環境や時間等の制約で全て実施できないこともある
- 実質的にテスト設計の段階で間引くことと同義
- テストを実施しなかったことに対するリスクを検討

■設定したテスト項目数
■実際に実施したテスト項目数の比率
■実施したテスト項目数と，プログラム規模との比率
■実施しなかったテストの明確化，そのリスク・対応策

G1Q3:検出したバグを修正

＜着目点＞
テストの終盤で発生したバグの重要度とバグ修正のトレードオフ

- 発見したバグは全て修正することが原則
  - バグの修正にはデグレードの危険が常につきまとう
  - テスト終盤のバグ修正には格段の注意が必要
  - バグの重要度によっては，修正しない選択肢もある
- バグの重要度
  - 重要度の定義は，一種の価値観の表れ
  - 重要度の定義の合意，判定の仕組み
- 考え方の根底
  - 検出したバグを全て修正するという，いわば100点満点を基準にするのではなく，”合格点”を合理的に設定

G1Q3:検出したバグを修正（続き）

- 修正しないとしても，バグの原因解明は必須
  - 不具合の現象が軽微であっても，原因はクリティカルなものかもしれない
- 未修整バグと，信頼性の評価は分けて考察する必要あり

■バグ発生件数（修正済みの件数と未修整の件数）
■未修整バグの各々の重要度
（使用者や他システム等への影響）
■重要度を判定する基準と，それに則って判定する仕組み

G1Q4:バグを正しく修正していることを確認

＜着目点＞
デグレードの傾向分析（発生頻度や部位等）

- 内在する問題を明らかにし対処することも可能
- 回帰テストの範囲を判断する材料
- 新規バグかデグレードか，切り分けする仕組み

■デグレード発生件数
■デグレード発生傾向の分析結果

ソフトウェアテストのGoalとQuestion

- <G1>ソフトウェアに潜むバグを検出し，修正
  - G1Q1:妥当な質と量のテスト項目を設定
  - G1Q2:設定したテスト項目を実行
  - G1Q3:検出したバグを修正
  - G1Q4:検出したバグを正しく修正していることを確認
- <G2>要件を満足していることを確認
  - G2Q1:要件を網羅するテスト項目を設定
  - G2Q2:プログラムの全機能を動作させるテスト項目を実行
- <G3>バグ発生状況から品質成熟度を判断
  - G3Q1:予め想定したバグの数だけ，バグを検出できている
  - G3Q2:信頼度成長曲線が収束している
- <G4>開発プロセスの問題を特定・対策
  - G4Q1:バグの発生状況を，開発プロセスにフィードバックしている

G2Q1:要件を網羅するテスト項目を設定

＜着目点＞
再利用したソフトウェアに対するテスト

- 再利用ソフトのテストを省略するのは危険
  - 再利用ソフトを的確に組み込めているかどうかは，再利用ソフトそのものの品質とは別の問題
  - 新規開発部が再利用部に悪影響を及ぼす可能性
  - 再利用部に潜む論理的ミスが顕在化する可能性
  - 機能の実装漏れ可能性

■要件の数，粒度
■テスト項目数

G2Q2:全機能を動作させるテスト項目を実行

＜着目点＞
カバレッジ（網羅度を判定・向上）

- カバレッジの効果
  - 網羅度を高める指針になるため，テストの抜けを抑制
  - 論理的に到達できないパスが分かり，カバレッジの結果を分析することで，バグを摘出できる可能性
- 注意すべき点
  - 論理の抜け（パスの抜け）はカバレッジでは検出できない
  - テスト項目の評価ができる環境（データの積み重ね）

■カバレッジとテスト項目数等との対比
（データの積み重ねによるテストの質と量の検証）

ソフトウェアテストのGoalとQuestion

- <G1>ソフトウェアに潜むバグを検出し，修正
  - G1Q1:妥当な質と量のテスト項目を設定
  - G1Q2:設定したテスト項目を実行
  - G1Q3:検出したバグを修正
  - G1Q4:検出したバグを正しく修正していることを確認
- <G2>要件を満足していることを確認
  - G2Q1:要件を網羅するテスト項目を設定
  - G2Q2:プログラムの全機能を動作させるテスト項目を実行
- <G3>バグ発生状況から品質成熟度を判断
  - G3Q1:予め想定したバグの数だけ，バグを検出できている
  - G3Q2:信頼度成長曲線が収束している
- <G4>開発プロセスの問題を特定・対策
  - G4Q1:バグの発生状況を，開発プロセスにフィードバックしている

G3Q1:予め想定した数のバグを検出

＜着目点＞

摘出バグ数を目標管理

- ソフトウェアに仕込まれたバグの総量を見積もり，レビューを含めて各フェーズで除去する数を想定
- 如何に合理的にバグの総量を見積もるか？
  - 過去のプロジェクトデータからパターンを見出して予測
- バグの数が予測値より少なければ，テストの内容を見直す
- 予想した値にまで達したら，テストは十分行ったと判断

G3Q1:予め想定した数のバグを検出（続き）

- この考え方の問題点
  - バグの数が目標にまで達したらそこで安心感が芽生える
    - それ以上のバグを検出しようとする意欲が薄らぐ（過去の実績を盲信する危険）
  - 想定以上の数のバグが検出された場合には，この基準はあまり意味を持たなくなる
    - あまりに多くバグが検出された際には，一度テストをストップし，設計工程に立ち返る

■バグの数の見積り値と実績値の差異

G3Q2:信頼度成長曲線が収束している

- テストフェーズのある時点でバグを検出するペースはピークに達して，その後なだらかになる（これは直感的にも理解できる）
- 収束度合いの定量的判断方法
  - 信頼度成長曲線の全体の傾きと終盤の傾きの比
  - MTBF（平均故障間隔）等

＜着目点＞

テスト終盤になってもバグが発生して収束判断をするに至らないケースはどうするか？

G3Q2:信頼度成長曲線が収束している（続き）

- そもそも，なぜテストフェーズ終盤になってもバグが検出されるのか？
  - 単純に，バグを検出するテストが完了していない
    - 信頼度成長曲線が未だ成長過程
  - デグレードにより新たにバグが作りこまれ，それが終盤になって検出
  - テストを進めるにつれて，そのプログラムの理解がより深まり，テストを追加することによって，新たにバグを検出

G3Q2:信頼度成長曲線が収束している（続き）

- バグをテスト終盤になって発生させないようにするには？
  - バグをより早期に検出するように努める
    - バグを検出したら他にも同種のバグが残存することを疑い，芋づる的にバグを検出する
    - バグの発生状況を分析することが有効
      - バグ発生のパターン（バグ発生部位，バグ作り込み要因）
      - 類似するバグを検出するテストを追加して，潜在するバグを早期に検出

本稿では「欠陥分析アクティビティ」と呼称

G3Q2:信頼度成長曲線が収束している（続き）

- テスト終盤になって発生したバグも同様に分析
  - 未だ残存するバグの可能性をリスクとして捉え，テスト完了判断の材料にする
  - テスト終盤になって検出したバグを，未検出のバグのシグナルとして捉える考え方
  - 分析の元となるバグの重要度とは分けて考慮する必要あり
  - 軽微なバグが残っていて，そのプログラムが90点の出来と捉えられても，同種の未検出バグを考慮すると，引き算した10点が本当は10点ではないのかもしれない

■信頼度成長曲線の収束判断
（曲線収束度の評価方法，MTBF）
■欠陥分析アクティビティの分析結果と，リスクの評価

ソフトウェアテストのGoalとQuestion

- <G1>ソフトウェアに潜むバグを検出し，修正
  - G1Q1:妥当な質と量のテスト項目を設定
  - G1Q2:設定したテスト項目を実行
  - G1Q3:検出したバグを修正
  - G1Q4:検出したバグを正しく修正していることを確認
- <G2>要件を満足していることを確認
  - G2Q1:要件を網羅するテスト項目を設定
  - G2Q2:プログラムの全機能を動作させるテスト項目を実行
- <G3>バグ発生状況から品質成熟度を判断
  - G3Q1:予め想定したバグの数だけ，バグを検出できている
  - G3Q2:信頼度成長曲線が収束している
- <G4>開発プロセスの問題を特定・対策
  - G4Q1:バグの発生状況を，開発プロセスにフィードバックしている

G4Q1:バグ発生状況を開発プロセスにフィードバック

- 前述の欠陥分析アクティビティは，分析結果を主にテストプロセスにフィードバックしていた
- これを開発プロセスにフィードバックするように拡張
  - 欠陥分析アクティビティを掘り下げると，そのバグを作り込んだプロセスと，上位の検証プロセスに存在する問題点が分かってくる
  - その作り込み要因と流出要因を特定できれば，プロセスに潜む欠陥から生成された同種のバグを机上で検出することができるようになる
  - 開発とテストを別の組織で行う場合，テストチームが開発チームにプログラムを差し戻すことがある。(端的な例)

G4Q1:バグ発生状況を開発プロセスにフィードバック(続き)

- 信頼関係がなければこの活動は成り立ちにくい
  - 開発チームとテストチームの二人三脚
  - この協調関係を築き上げるには:
    - テストチームには献身の心構え
    - 開発チームには最終的に品質を作り込むのは自身にあるとの自覚が必要
  - こう言う意味で，テストフェーズは単に検証工程でなく，生産工程なのだ

■差し戻し回数
■工程別アクション実施件数
■指摘項目改善率

テスト完了判断基準の大項目

- <0>体系的なテスト設計(前提条件)
- <1>テストの実施状況
- <2>バグ発生状況
- <3>バグの修正状況
- <4>バグの修正品質
- <5>信頼度成長曲線の収束度合い
- <6>検出バグの発生傾向分析
- <7>開発プロセスへのフィードバック状況

<0>体系的なテスト設計(前提条件)

- テスト項目は，的確・効率よくバグ検出できるように，論理立てて設計していなければならない。
- これを測るため，着目したカテゴリ別に件数が分かるようにすることが必要。
- 同時に，要件を満たしていることをくまなく網羅していることを，要件とテスト項目数を対比できるようにしなければならない。
- これらはテスト対象の規模に見合った量でなければならない。
- 上記の条件を満たしつつ，有限なリソース内で収まるように，計画する必要がある。

<0>体系的なテスト設計(前提条件)(続き)

- バグを終盤で発生させると，その対処にはコストと危険が伴うため，バグを可能な限り早期に検出する必要がある。
  ムダにテストを行うと時間のロスにつながるため，テスト項目数はむやみに増やすことはできない。
- リソースと効率的なテストとのトレードオフで，テストを実施しないと意志決定した領域がもしあれば，それはリスクとして管理下に置き，モニターする必要がある。

テスト完了判断基準

- <1>テストの実施状況
  - 設定したテストは，全て実施済みでいること。
  - 必要なテストが，環境面や他の事情で実施できない項目があれば，その理由とリスクの度合い・対応策を明確にする
  - カバレッジとテスト項目数，過去の実績と照らし合わせ，一定の基準を満たしていること。
- <2>バグ発生状況
  - 過去の実績と照らし合わせて目標に定めた件数に見合った数のバグを検出できていること。
  - バグとテスト項目数の比を検討し，妥当な基準にあること。
  - 少数のテストであまりに多くのバグが検出されていたり，バグの検出数が目立って少ないなどが見受けられた場合には，その理由を分析し，対策すること。

テスト完了判断基準（続き）

- <3>バグの修正状況
  - 検出したバグは原則として全て修正していなければならない。ただし，重要度と他への影響に応じ，残存するバグ件数を許容する基準を設けても良い。
  - その場合には予め下記を実施していること。
    - 重要度の定義，個別のバグのランクを意志決定する仕組みを予め計画に盛り込み，関係者間で合意している
    - 顧客に残存するバグの存在を通知し，了承されている

テスト完了判断基準（続き）

- <4>バグの修正品質
  - デグレードが発生した件数と，発生状況の分析結果により，未だ存在するデグレードバグを類推し，リスクが許容できる範囲であること。
- <5>信頼度成長曲線の収束度合い
  - 信頼度成長曲線が収束傾向にあること。
- <6>検出バグの発生傾向分析
  - 欠陥分析アクティビティの分析結果によりリスクを評価し，リスクが許容範囲であること。
- <7>開発プロセスへのフィードバック状況
  - 欠陥分析アクティビティの分析結果を開発プロセスにフィードバックしたものに，積み残しがないこと。
  - 実施すべき見直しが全て完了していること。

テスト完了の意志決定

- 導出した基準を全て満足するものが，完了判断基準ではない
  - メトリクスは，ある観点での側面を切り出して表したもの
  - それぞれのメトリクス単独では全体像を正しく見渡せるものではない
  - 様々な角度からの情報を集約し，総合して全体像を捉えることが，合理的なテスト完了判断につながる

単に数値と基準の差異を見るのではなく，その意味するところを，
開発チーム，テストチーム，品質保証チーム等の関係者で議論を尽くす

出来ていることと出来ていないことを明確化し，
「テストを継続しない」ことに対するリスクを評価

ソフトウェアのテストに明快な終わりはない

- テスト完了判断基準は，それ単独で機能するものではない。
- ソフトウェア開発のライフサイクル全体を通して品質保証計画を立て，その中の一要素として基準を設計することが必要
- 基準を満たしていてもなお未知のバグが存在する可能性がある
- 万能な完了判断基準は存在しない。これを踏まえて完了判断基準を設計する
- 導出した7つの基準は，テストフェーズのタイムリミット間近になってからデータをかき集め，後追いで基準に照らし合わせて判断に用いるものではない。

自らが置かれた状況で，日々きめ細かい軌道修正をし，
マネージメントすることこそが，何よりも重要

これからの指針

- 今回検討したのは，ほんの一例
  - テストの目的，実現手段や評価方法はもっと別の観点があるはずだ
- 現時点で認識している課題
  - 「テストの質の評価」と「プログラムの規模の表し方」に，まだ曖昧な点が多い
  - 欠陥分析アクティビティでバグを分析し，そのリスクの評価を完了判断に加えることを提案したが，それを定量的に表す方法が確立できていない

本検討は，ソフトウェア技術者ネットワーク
（S-open）のメトリクスSIGによるものです。
メンバー皆様方に，心より，感謝いたします。