第 5 分科会（マネージメントグループ）

客観的ソフトウェアテスト終了基準の提案

A Proposal of Objective Criterion for Software Testing Completion

主査　高橋　寿一　（ソニー株式会社）
副主査　増田　聡　（日本アイ・ビー・エム株式会社）
副主査　奥村　有紀子　（有限会社デバッグ工学研究所）
研究員　石橋　琢磨　（株式会社野村総合研究所）
　　加藤　篤典　（ブラザー工業株式会社）
　　共田　元樹　（株式会社日立システムアンドサービス）
　　戸森　貴弘　（株式会社アドバンテスト）
　　那須　俊博　（株式会社セゾン情報システムズ）

（敬称略　順不同）

1. 概要

ソフトウェアテストの終了に明確な基準は無い。ソフトウェアテストでそのプログラムにバグがあることは証明できても、バグがないことは証明できない。しかし、リソースの制約などの経済的な理由、顧客との約束などから、ソフトウェアテストを際限無く継続することは出来ずに、何かをトリガとしてソフトウェアテストを終了せざるを得ない。業種、ソフトウェアの規模などの違いを克服するようなソフトウェアテストの終了基準を定義したものが無いのが現状である。

その現状を解決すべく、本研究では、ソフトウェアテストの終了基準を研究し、その成果として、客観的にソフトウェアテストを終了する基準を提案する。具体的には、ソフトウェアテストのプロセスにおける各時点の品質にスコアを導入しモデル化し、スコアにより終了基準を検証するものである。

Abstract　In general, there is no accurate definition of software testing completion. It is possible that there are some bugs in the software program by software testing. But it is not possible that there are no bugs in the software program by software testing. Testers can't keep software testing endlessly because of the restriction of resources or the promise with customer. Actually testers stop software testing by some triggers. And they don't have the criterion for software testing completion, which is comprehensive under the difference of the kind of business or the software scale. So we studied and proposed about the objective criterion for software testing completion to break the current state. Concretely we defined the quality model of software testing process by using a concept of score.

2. 背景

多くの製品にソフトウェアが組み込まれ、多くの事業で情報システムが、事業の核をなしていることは、衆目の一致するところである。携帯電話、自動車など組み込み製品の分野では、ソフトウェアによる性能向上、機能数向上の結果、搭載されるソフトウェアは複雑化の一途を辿っている。電車の改札システム、銀行、証券系のオンライン処理なども大規模化、複雑化され、統廃合されることも多く、ソフトウェアの品質に社会の注目が近年非常に高まっている。この様な時勢で、ソフトウェアの不具合が社会問題になっており、ソフトウェアの不具合が GDP の 2 割～3 割に影響を与えると言われている。[1]

しかしながら、ソフトウェアを開発・テストする現場では、開発案件の増加からテストの時間は益々十分に取れなくなり、テストをどこで終えるべきかという判断基準から品質の観点が欠落する、或いは判断基準の見直しが短期間で繰り返されていると、色々な文献などで紹介されている。

当分科会での初期活動において、テスト終了条件に関して、参加メンバそれぞれの会社の事例を持ち寄って検討を行った。単体テストにおいては、カバレッジ計測等、ある程度一定の基準適用があると判ったが、結合テスト以降については、各社とも基準が異なっており、曖昧で客観性に欠けたものも多く、最終的には会社･部門毎の主観的な判断・ルールに頼っているのが実情であった。また、定量的な基準を定めている会社もあったが、その基準自体は過去の実績等から算出されている等、客観的なパラメタから生成されているものではなかった（例 バグ摘出密度、テスト実施密度)。過去の実績を元に見積もる手法自体を否定するものではないが、主観的に決定することが多く、根拠を示すことが難しいケースも多く見受けられた。

適切なテストの終了基準を開発側に提示することができず、結果としてテスト完了後のバグ発生による大幅な手戻り作業発生についての悩みを抱えているメンバも多かったことから、客観的な基準として提示可能な､品質に着目した､結合テストの終了基準を研究テーマとして採用した。

3. 検討内容

結合テスト以降におけるのテスト終了基準を検討するにあたり、以下の手順にて事例・文献を調査・検討することとした。

① テスト終了基準に関する各社事例、文献、論文を調査・分類し、既知の手法に対する理解を深める
② 既知の手法をメンバ各社にて適用する際の問題点・不足する点について検討する
③ 既知の手法を含む、客観的なテスト終了基準について検討する

以下、調査、検討した事項について説明する。

3.1. 事例と文献の調査

我々は、テストの完了基準を調査し、得られた内容を分類することを試みた。結果として、主に以下の４つのグループに分類した。

① 信頼度成長曲線による判定
- 信頼度成長曲線による、残存バグ予測、バグ摘出収束状況による判定

② 基準値による判定
- バグ密度（ここではプログラム規模に対するバグ摘出件数の割合とする）の、基準値に対する達成度合いによる判定
- テストケース密度（ここではプログラム規模に対するテストケース数の割合とする）の、基準値に対する達成度合いによる判定
- テスト期間（例 実施時間）の、基準値に対する達成度合いによる判定

③ バグ分析結果による判定
- 5WHY 手法等の、RCA(Root Cause Analysis)[2]による、バグ原因分析を元にした判定
- ODC(Orthogonal Defect Classification：直交欠陥分類)[3]等の手法により、摘出バグを分類し、その傾向・収束状況による判定
- 重要なバグの摘出・収束状況による判定

④ その他
- リスク・ベースド・テスト手法[4]による判定
- ①～③等の、各種メトリクスを組み合わせて判定する手法[5]
- 予算・期間による判定（品質に着目した終了基準なし）

3.2. 問題点の検討

我々メンバの、各社における既知の問題点を挙げ、調査した手法にて解決可能か探ったところ、興味深い手法は多かったものの、現場へ導入するには不十分な面があるとの結論に達した。以下に調査結果を元にメンバから出た意見を挙げる。

- 基準自体の客観性についての情報が不足している

  基準として採用するメトリクス自体は明確化されているが、達成度判断に使用する数値や点数配分が、「過去の実績等から判断」という例が多く、客観性に欠ける。
- 適用の前提条件が多い

  信頼度成長曲線等の適用方法については、「テストケース毎にバグが発生する確率が等しい」等の前提があることが多い。実際の現場では、これらの前提を満たすことができないことが多い。
- テスト作業自体の品質を評価する枠組みが不足している
- 複数の手法を組み合わせて評価する際に、組合せ方、評価の基準がない

  基準単体での評価は危険であり、これらの基準を組み合わせて評価すべきであるという例は見られるが、組合せ方、判断方法について明示されている情報が少ない。「総合的に判断」等、明確な判断ができない。

個々の基準・評価方法自体を否定するものでは無いが、これらの問題点を埋める、何らかの基準が作ることが出来ないか？と言う意見が上がり、基準を検討することとした。

3.3. 目標設定

基準を検討する上で、本研究で重視したのは、以下の事項である。

- 可能な限り明確に判断できること
  基準を使用する側の主観に大きく左右されないようにする。
- 可能な限り前提条件を少なくすること
  考え方を簡略化するための前提を除き、現実と乖離した前提を設けることを避ける。

テスト完了を明確に判断するには、その時点でテストを終了した場合に、どれだけの損益が発生するかを明示化することが適切であると考えた。損益を明確化することにより、エンジニアリング知識が不十分である可能性がある経営層へ、品質と損益の関係を説諭することができる。

テスト終了の目安を損益で表す際に、リスク分析の手法が採用できるのではないかと考えた。リスク分析手法において一般的な期待金額を求める方法として、EMV (Expected Monetary Value) がある。EMV においては、期待金額は以下の計算式で求められる。

リスクの発生確率　×　金額（結果額）　＝　期待金額

ここで、テスト完了時の品質リスクに着目した場合、上記式は、テスト完了後にバグが発生するリスク、およびそのバグ発生による損益によって以下のように表現できると考えた。

・リスクの発生確率　・・・　テスト完了後にバグが発生する確率（バグ残存可能性）
・金額（結果額）　・・・　残存バグが発生した場合の影響、または発生しうる金額
・期待金額　・・・　損益

最終的な目標としては、テストをそこで終了した際の損益を算出するモデルによって、テスト完了を判断しようというものであるが、その前段として、まずはバグが残存する可能性を表すことで、テスト完了のための基準として使用できるものと考えた。すなわち、バグ残存確率がゼロとなった段階でテストを完了させるという、簡易的な完了の基準である。そこで、本研究においては、段階的に、以下の目標を設定して活動することとした。

目標１：リスクの発生確率「＝バグ残存確率」をモデル化し、提案すること
目標２：提案モデルの効果を実際のプロジェクトで検証すること
目標３：リスクの影響度分析方法、金額(結果額)算出の検討と、損益算出方法のモデルを定めること

4. 検討したモデル

4.1. 基本コンセプト

「バグ残存確率」を単純に数値化する場合、その計算式は、以下のようになる。

バグ残存確率　＝（総バグ件数　－　摘出済みバグ件数）÷　総バグ件数

ここで、摘出済みバグ数は計測可能であるが、総バグ件数については、推測が必要である。バグ予測の手法として、過去の実績を元に見積もる方法や、テストケースの進捗状況と摘出バグ状況を元に信頼度成長曲線を使用して予測する方法等がある。しかしながら、これら従来の方法では、テスト完了時点までの作業品質、および成果物品質は、総バグ数予測結果に影響を与えない。

我々は、テストの完了時点での品質は、テスト自体の作業品質、および結合テストまでの作業・成果物に依存すると仮定し、モデリングすることを意図した。つまり、「今までの工程と成果物の品質が良ければ、残っているバグも少なく、逆に今までの工程と成果物の品質が悪ければ、残っているバグも多い」という考え方である。また、摘出したバグに対し、適切な対応を完了しない限り、バグが残存するリスクは低減できないと考えるが、その状態をモデルに取り入れることを意図した。

そこで、テストケースは、テスト開始前にレビューを実施する等により、一定の網羅性・妥当性を確保していることを前提とし、テストの進捗とバグ摘出状況から、それまでの工程の作業と成果物品質をスコアリングし、その結果を確率として表現するモデルを検討した。検討したモデルについて次項に示す。

4.2. プロジェクトの定義

モデルの簡易化のため、開発プロセスや対象フェーズといった、プロジェクトの条件、用語を定義した。定義内容は、別紙「付録１　プロジェクトの定義」を参照。

4.3. 詳細内容

バグ残存確率算出モデルの基本的な内容は以下の通り。

１：テスト開始時は 100%とする。

２：全てテストケースを消化すると 0%になる。

３：テスト実施中にバグが摘出されると増加する。なお、未修正のバグがある場合は、バグ残存確率は 100%になる（特異点１）。

４：バグ残存確率の増加量は一定量とするが、摘出したバグにより、増加したリスクを減少させるために必要な作業量が変化する。

５：バグ摘出後、適切な作業を行い、テストケース消化が完了すれば、３の増加分は減少し、さらに２の通り減少する。バグが修正されただけでは、増加したリスクは完全には回復せず、適切な見直し、回帰テスト、テストケース再消化が完了した段階で、完全に回復するものとする。

６：テストケース消化数をバグ摘出数が上回った場合、修正が追いついていないと、一時的に 100%を超えることがある（特異点２）。

4.3.1. バグ摘出時のバグ残存確率の増加について

摘出されたバグの内容に応じて、必要な作業が定まる。そこで、コーディング起因のバグは単体テストまでに摘出すべきであり、結合テスト以降は設計起因のバグを中心に摘出するという前提を重視し、バグの分類方法を決定した。決定した分類を「表 4.3-1」に示す。

表 4.3-1 摘出バグの分類

大分類	小分類	内容
設計	設計ミス	設計に起因したプログラムのバグ
単体テスト	項目漏れ	単体テストの項目が漏れていて、結合テスト時に見つかった場合のプログラムのバグ
	項目間違い	単体テストケースが間違っていて、結合テスト時に見つかった場合のプログラムのバグ
	結果間違い	単体テストケースは正しいが、単体テスト実施者が気づかず見逃していた場合のプログラムのバグ
結合テスト	項目漏れ	結合テストケースから漏れていたが、摘出されたプログラムのバグ
	項目間違い	結合テストケースが間違っていたが、摘出されたプログラムのバグ
	結果間違い	結合テストケースは正しいが、結合テスト実施時には気づかず見逃していたプログラムのバグ

4.3.2. バグ摘出後の作業によるバグ残存確率の減少について

摘出したバグを修正しただけでは、バグ残存リスクを完全に払拭することは出来ないと我々は考える。そこで、リスクを払拭するために必要な作業を以下に定義する。リスク払拭に必要な作業の定義を「表 4.3-2」に示す。

表 4.3-2 リスク払拭に必要な作業の定義

項目	内容
バグの修正	設計書起因のバグであれば設計書の修正と、それに伴うプログラムの修正。 プログラムのバグであれば、プログラムの修正。 テストケースの漏れや間違いであれば、プログラムの修正とテストケースの修正。 テスト結果の間違いであれば、プログラムの修正を行う。
類似見直し（水平展開）	設計書やテストケースにバグが見つかった場合、該当する設計書やテストケース以外に類似のバグがないか見直し、見つかれば設計書、テストケース、プログラムの修正を行う。※
回帰テスト（単体テスト）	バグを修正したプログラムに対する単体テストの（必要十分な）回帰テストの実施。
回帰テスト（結合テスト）	バグを修正したプログラムに対する結合テストの（必要十分な）回帰テストの実施。

※ 類似見直しの際に見つかったバグも同様にバグ残存確率が増加し、これらの作業が必要となる。

バグの分類と必要な作業の対応を、別紙「付録２　増減値マトリクス」に示す。
モデル簡略化のため、バグ摘出による増加量は一定とし、また、一つの作業によって回復する量は、バグ分類によって定まる必要な全作業で割った値とした。

4.4.　バグ残存確率の算出

表 4.3-1 に示した方法にて分類したバグ一覧と、別紙「付録２　増減値マトリクス」を元に、その時点でのバグ残存確率を算出する。計算式は以下の通り。

① テストの進捗による減少値（%）

消化済テストケース数　÷ 総テストケース数　× 100

② バグの摘出による増加値（%）

バグの摘出数 ÷ 総テストケース数　× 100

③ バグ摘出後の作業による減少値（%）

摘出バグ毎に「付録 1 増減値マトリクス」の〇の数から減少値を決定する。

確率の減少値は、以下の式の総和となる。

（1 ÷ 総テストケース数）×（実施済み作業の合計
÷　必要な作業の合計 ）× 100

本モデルの適用イメージを、別紙「付録３　適用イメージ」に示す。

5. 研究の成果と今後の課題

5.1.　研究目標の達成度

本研究における目標と達成度を以下に示す。

目標１：リスクの発生確率「＝バグ残存確率」をモデル化し、提案すること

成果１：リスクの発生確率をモデル化し、提案できた

摘出したバグは、修正しただけではバグが残存するリスクは完全に減らすことができないというコンセプトを元に、バグ残存確率をモデル化し、提案することができた。モデル簡略化のために前提条件を置かざるを得なかった等、一部制限される点もあるが、本目標は概ね達成することができた。

目標２：提案モデルの効果を実際のプロジェクトで検証すること

成果２：実際のプロジェクトへの適用は今後の課題

時間的制約から、実際のプロジェクトへの適用は実施できていない。

目標３：金額(結果額)算出の検討と、損益算出方法のモデルを定めること

成果３：換算方法を検討、実プロジェクトの中で可能性調査が必要

リスク発生の影響、金額(結果額)については、バグが残存した機能の重要度等をパラメタにして算出する方向性を検討しているが、モデルの実プロジェクトへの適用の中で、可能性を探って行く必要がある。

5.2. 今後の課題

本研究活動では、テストの客観的な終了判断のための基準となるモデルの仮説を立てるにとどまった。本来であれば、仮説を証明するための事例検証が必要であったが、時間的制約から、検証作業を行うことが出来なかった事が反省点である。今後の課題としては、事例によるモデルの検証を行い有効性について十分な論証を得ることと共に、最終目標へ向け、データ収集が必要となる。

課題１：実プロジェクトでの検証

① 確率増減要素の妥当性
② 金額換算モデルのためのデータ収集

また、仮説の検証だけではなく、モデルの精度向上のため、以下の点について議論を深める必要がある。

課題２：モデルの発展に必要な要素

① プロジェクト特性を考慮した確立増減要素の抽出
② 確率増減要素の細分化
③ バグの分類に応じた確率増加率の変動の検討
④ 作業毎の確率減少に寄与する比率の検討（現在は簡略化のため等分としている）
⑤ テストケース品質の評価
⑥ テスト終了判断のための閾値の検討

5.3. まとめ

本研究においてテスト終了の判断基準として使用するモデルを提案した。事例検証ができなかったため効果を定量的に提示することができなかったが、本研究で提案したモデルを、各社の実プロジェクトに適用し検証することで具体的な効果を示していきたい。

また、最終的な目標であるソフトウェア品質に潜むリスクを「金額」で評価することが可能になるよう進化させていきたい。

6. 参考文献

[1] 飯塚悦功、益田直美、西康晴,“日科技連ニュース　No.54　2007年5月号「シリーズ　ソフトウェアの「品質」を考える①今ここにある危機」”、日科技連、東京、2007

[2] Wikipedia、「Root Cause Analysis」 (http://en.wikipedia.org/wiki/Root_cause_analysis)

[3] R.Carig、S.P.Jaskiel、体系的ソフトウェアテスト入門、日経BP、東京、2005

[4] Ram Chillarege, Inderpal S. Bhandari, Jarir K. Chaar, Michael J. Halliday, Diane S. Moebus, Bonnie K. Ray, Man-Yuen Wong　IEEE Transactions on Software Engineering, Vol. 18, 「Orthogonal Defect Classification - A Concept for In-Process Measurements」No. 11, Nov. 1992.

[5] 堀 明広、仲 孝浩、向井 清、金子 喬、「定量的ソフトウェアテスト完了判断基準の一考察」、JaSST06 東京 経験論文、Jan. 2006

付録１　モデルの前提条件

表　付録１－１　前提とする条件

項目	内容
開発プロセス	ウォーターフォール
対象フェーズ	結合テスト
成果物の種類	設計書 プログラム 単体テストケース 単体テスト結果 結合テストケース 結合テスト結果

それぞれの用語を以下のように定義する。

(a)　成果物の定義について

プロジェクトの成果物を以下のように定義する。

- **設計書**
  プログラムの概要設計や詳細な設計が記述された文書
- **プログラム**
  プログラム本体
- **単体テストケース**
  プログラムを分割した単位をモジュールとした時の、モジュール単体で行うテストの項目とデータが記述された文書。
- **単体テスト結果**
  単体テストの実施結果や証左物。
- **結合テストケース**
  プログラムを分割せずに行うテストの項目とデータが記述された文書。
- **結合テスト結果**
  結合テストの実施結果や証左物。

(b)　工程の定義について

プロジェクトの成果物を作成する作業をアクティビティと呼び、以下のようなアクティビティを定義する。

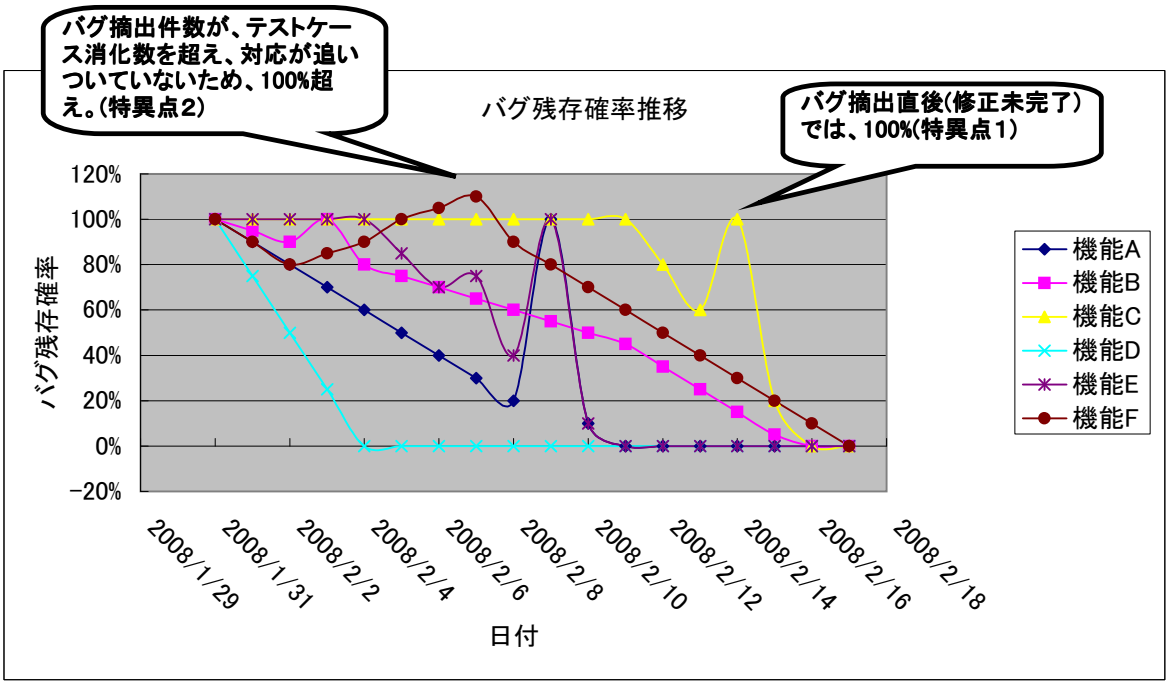
- **設計**
  設計書を作成する作業
- **プログラミング**
  プログラムを作成する作業
- **単体テスト項目作成**
  単体テスト項目を作成する作業
- **単体テスト実施**
  単体テストを実施する作業
- **結合テスト項目作成**
  結合テスト項目を作成する作業
- **結合テスト実施**
  結合テストを実施する作業

付録2　作業対象マトリクス

バグ分類	作業	作業対象					
		設計書	コード	単体テストケース	単体テスト結果	結合テストケース	結合テスト結果
設計ミス	バグの修正	○	○				
	類似見直し・水平展開	○					
	リグレッション単体テスト	○	○				
	リグレッション結合テスト	○					
	○の合計	4	2	0	0	0	0
単体テスト項目漏れ	バグの修正		○	○			
	類似見直し・水平展開		○	○			
	リグレッション単体テスト		○	○			
	リグレッション結合テスト						
	○の合計	0	3	3	0	0	0
単体テスト項目間違い	バグの修正		○	○			
	類似見直し・水平展開		○	○			
	リグレッション単体テスト		○	○			
	リグレッション結合テスト						
	○の合計	0	3	3	0	0	0
単体テスト結果間違い	バグの修正		○		○		
	類似見直し・水平展開		○		○		
	リグレッション単体テスト		○		○		
	リグレッション結合テスト						
	○の合計	0	3	0	3	0	0
結合テスト項目漏れ	バグの修正		○			○	
	類似見直し・水平展開		○			○	
	リグレッション単体テスト						
	リグレッション結合テスト		○			○	
	○の合計	0	3	0	0	3	0
結合テスト項目間違い	バグの修正		○			○	
	類似見直し・水平展開		○			○	
	リグレッション単体テスト						
	リグレッション結合テスト		○			○	
	○の合計	0	3	0	0	3	0
結合テスト結果間違い	バグの修正		○				○
	類似見直し・水平展開		○				○
	リグレッション単体テスト						
	リグレッション結合テスト		○				○
	○の合計	0	3	0	0	0	3

○は、バグを摘出した際に増加した確率を全て回復させるために必要な作業。

付録3　適用イメージ



評価日	機能A	機能B	機能C	機能D	機能E	機能F
2008/1/31	100%	100%	100%	100%	100%	100%
2008/2/1	90%	95%	100%	75%	100%	90%
2008/2/2	80%	90%	100%	50%	100%	80%
2008/2/3	70%	100%	100%	25%	100%	85%
2008/2/4	60%	80%	100%	0%	100%	90%
2008/2/5	50%	75%	100%	0%	85%	100%
2008/2/6	40%	70%	100%	0%	70%	105%
2008/2/7	30%	65%	100%	0%	75%	110%
2008/2/8	20%	60%	100%	0%	40%	90%
2008/2/9	100%	55%	100%	0%	100%	80%
2008/2/10	10%	50%	100%	0%	10%	70%
2008/2/11	0%	45%	100%	0%	0%	60%
2008/2/12	0%	35%	80%	0%	0%	50%
2008/2/13	0%	25%	60%	0%	0%	40%
2008/2/14	0%	15%	100%	0%	0%	30%
2008/2/15	0%	5%	20%	0%	0%	20%
2008/2/16	0%	0%	0%	0%	0%	10%
2008/2/17	0%	0%	0%	0%	0%	0%