2B-3

テスト管理システムを用いたテスト状況の可視化方法の提案

久連石　圭†　　河村　透†　　小笠原　秀人†　　佐々木　愛美†

†東芝　ソフトウェア技術センター

1. はじめに

近年、製品の開発期間の短期化に伴い、ソフトウェア開発におけるテスト工程の期間も短くなることが多い。そのため、ウォーターフォールモデルのように、全ての実装を終えてからのテストでは、製品のリリースまでにテストできる時間を十分に確保できず、リリース時に品質が十分向上しきれていないことがある[1]。

テスト期間が短い場合は、全ての機能の実装が終わっていない状態でも、実装済みの機能から順にテストしていくことで、テストできる時間を確保することができる。このように実装とテストを並行して実施する場合、実装された機能を早い段階からテストすることで、潜在する欠陥を早期に検出・修正し、いち早くソフトウェアの品質を向上させることがポイントになる。

2. 実装とテストを並行して実施する際の状況

ほとんどのソフトウェア開発では、ソースコードを一元管理する構成管理システムを用いる。構成管理システムでは、一連のソースコードの更新状態を“バージョン”（あるいは“リビジョン”）と定義し、名前（例えば v0.51 や r1832)をつけて管理している。

従来のテスト工程では、図１の(a)に示すとおり、実装が完了したバージョンを特定し、テストを実施することが多かった[2]。実装中にテストを並行して行う場合は、図１(b)のようにバージョンが頻繁に更新される中でのテストになるため、次のような状況がしばしば起こる。

2.1. 実装状況によって実施可能なテスト項目が変動する

実装とテストを並行して実施する場合、実装が全て完了するまでは実施できないテスト項目が存在する。そのため、テストの進捗が芳しくない状況が発生した場合、その原因がテストの実施の不備によるものか、機能未実装によるものか判別がつかない。

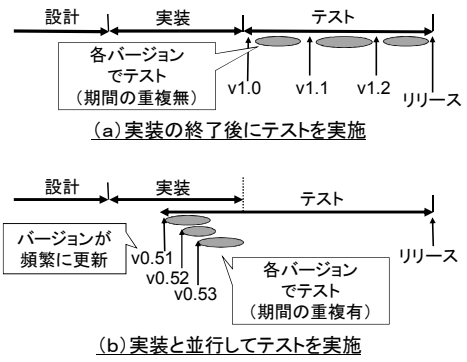

図１　実装後のテストと実装と並行したテスト

2.2. テスト担当者ごとにテストを実施するバージョンが異なる

実装と並行してテストを実施する場合は、テスト対象のバージョンが頻繁に更新され、バージョン毎に実施できるテスト項目も変動する。そのため、テスト担当者は特定のバージョンでテストを実施せず、目的に応じて各々が異なるバージョンに対してテストを行うことがある。各々のテスト実施バージョンが異なると、バージョンの更新順序とテスト結果の登録順序が一致しないことがある。その結果、テストの品質状況を確認する際には、従来の日付順での集計に加えて、バージョンの更新順序でテスト結果を集計しなければならず、集計や可視化にかかる負荷が増える。

3. テスト状況を可視化するためのテスト管理システムの提案

実装と並行してテストを実施する場合にテストの実施状況を確認できるように、実装済みの機能に対するテスト項目の数と合格しているテスト項目の数を、バージョンの更新順序に沿って自動的に集計し、可視化できるテスト管理システムを提案する。本システムの概要を図２に示す。

本テスト管理システムでは、テスト項目とテスト結果を蓄積している。開発者は機能を実装した際に実装した機能に対するテスト項目をシステムに通知する。また、テスト担当者は実装された機能から順にテストを行い、テスト結果を登録する。テスト状況を確認する場合は、システムに蓄積されたテスト項目やテスト結果を自動的に集計し、グラフを出力する。

Test Management System for Status Visualization
Kei KUREISHI†, Toru KAWAMURA†,
Hideto OGASAWARA†, Manami SASAKI†
†Corporate Software Engineering Center,
Toshiba Corporation

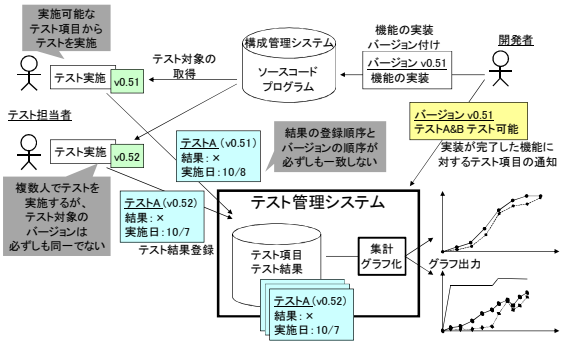

図２　テスト管理システムの概要

本システムでは、従来のテスト管理で利用していたテストの実施数や合格数を時系列に集計するグラフに加え、実装中にテストを行う場合にも状況を確認できるグラフを描画する。実装中には実施可能なテスト項目が変動するため、実装が完了した機能に対するテスト項目の数を集計して、合格したテスト項目の数と比較する。また、バージョンの更新順序とテスト結果の登録順序の不一致をなくすために、本システムでは図３に示すように、テスト結果をバージョン毎に分類した後、バージョン毎にテスト結果を集計し、集計値をグラフにプロットする。

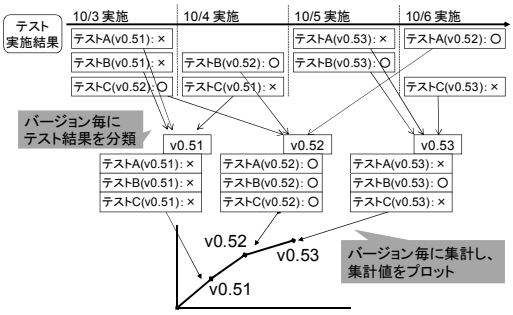

図３　バージョン毎のテスト結果集計

4.　本システムを用いたテスト状況の判断

提案したテスト管理システムは、登録したテスト結果を自動的に集計し、図４に示すグラフを描画する。グラフの①に示す範囲は未実装の機能に対するテスト項目の数であり、テストがまだできない状態を示している。また②に示す範囲は、実装は完了しているためテスト可能だが、まだ合否を確認していないテスト項目の数であり、テストの実施待ちの状態を示している。②の範囲に加えて線Ａと線Ｂの変化を確認することで、合格しているテスト項目の数が増えない原因を判別できる。例えば図４の(a)から(c)では、合格しているテスト項目の数（線Ａ）はほとんど増加していない。一方、実装済み機能に対するテスト項目の数（線Ｂ）は、(a)では線Ｂが殆ど増加していないため、機能が未実装でテストができない状態だと考えられる。また、(b)では線Ｂが日々増加しているため、機能は順に実装されているが、テストの進捗が芳しくない状態を示している。更に、(c)では線Ｂが減少しているため、テストは実施しているが、不合格が発生しており、一度実装した機能を再度修正していることを確認できる。

また、バージョンの更新順序に沿って、合格しているテスト項目の数を集計し、線Ａを描画することで、バージョンの更新に応じた品質の良否を確認できる。例えば、図４の(d)のように、バージョンの更新に伴って合格しているテスト項目の数が増えている状況からは品質が向上していることを確認できる。また、(e)のように合格しているテスト項目の数が減少することがある。ここでは、一度合格したテスト項目を再度テストしたときに不合格となる“機能デグレード”が発生している。この場合は、機能デグレードが発生したバージョンと変更内容を明らかにして、再度実装をやり直す必要がある。

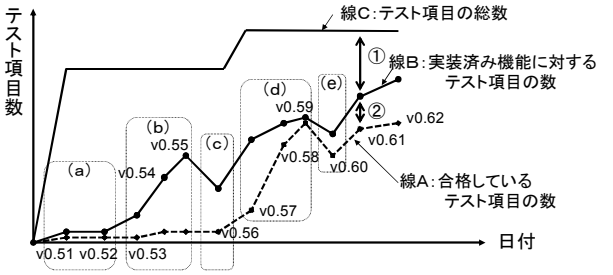

図４　提案するグラフ

5.　おわりに

本稿では、短納期化するソフトウェア開発において、実装が完了する前からテストを実施し、早期からソフトウェアの品質を確認するために、ソースコードの更新情報を活用することで機能の実装状況をふまえたテストの実施状況や品質の改善傾向を確認できるテスト管理システムを提案した。実装の完了前にテストを実施することで、早期に潜在する欠陥を検出でき、欠陥に対応する時間を十分に確保することが可能になる。その結果、予定されているリリース日までに、高い品質まで作り込むことができると期待される。

今後、提案したテスト管理ステムを用いたテスト管理を実践し、本システムを用いることで品質向上に寄与できていることを確認していく。

参考文献

[1] 岡崎毅久：ソフトウェアテストと品質保証の実際，日本テクノセンター（1999）
[2]Rex Black/テスト技術者交流会 監訳/トップスタジオ訳：基本から学ぶテストプロセス管理，日経 BP 社(2004)