社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

見逃し欠陥の回帰テスト件数を考慮したコードレビュー手法

田村　晃一† 　亀井　靖高† 　上野　秀剛† 　森崎　修司† 　松村　知子†
松本　健一†

† 奈良先端科学技術大学院大学情報科学研究科　〒 630–0192 奈良県生駒市高山町 8916–5
E-mail: †{koichi-t,yasuta-k,hideta-u,smrs,tomoko-m,matumoto}@is.naist.jp

あらまし　テスト工程での欠陥修正には膨大な再テストが必要になるケースがあり，コストの増大や納期の遅延原因になるばかりでなく，テスト漏れによる 2 次欠陥の危険も伴う．本稿では，コードレビューでの見逃し欠陥の修正により必要となる回帰テスト件数を考慮したレビュー手法を提案する．提案手法では回帰テストの件数が大きくなると推定される欠陥を優先的に検出することで，テスト工数の削減が期待される．提案手法の評価として，テスト計画書を用いた本手法でレビューするグループと，特に何も指定せずレビューするグループ間で削減された回帰テスト件数を比較した．その結果，回帰テスト件数を考慮したコードレビュー手法の方が回帰テスト件数の削減に寄与する欠陥をより多く発見できることがわかった．

キーワード　ソフトウェアレビュー，テストケース，回帰テスト

A Code Review Technique to Reduce Regression Test Cases for Omitted Fault

Koichi TAMURA†, Yasutaka KAMEI†, Hidetake UWANO†, Shuuji MORISAKI†, Tomoko MATSUMURA†, and Ken-ichi MATSUMOTO†

† Graduate School of Information Science,Nara Institute of Science and Technology
8916–5 Takayama, Ikoma, Nara, 630–0192 Japan
E-mail: †{koichi-t,yasuta-k,hideta-u,smrs,tomoko-m,matumoto}@is.naist.jp

Abstract　Some faults need many re-testing for fixing faults in test phase. These faults increase cost and delay of release, and may involve the risk of second fault for omiting the needed test. In this paper, using a test plan document, we conduct the review considering regression test cases for omitted fault in code review and evaluate its review technique effect. The reduction of test effort was expected by preferential detecting the fault that supposably derive many regression test cases. We compare its technique to Ad-hoc reading by the experiment that aim at code review. The result showed that the regression test-based reading could reduce more regression test cases than Ad-hoc reading.

Key words　Software Review, Test Case, Regression Test

1. はじめに

ソフトウェア開発における成果物もしくは中間成果物に対して作成者またはその他の開発者がその内容を確認することを目的として，レビューが実施されている．特に，欠陥を発見及び除去することを目的として実施されるレビューでは，テストと比較して早い段階での実施が可能であり開発効率の向上につながることが報告され [1] [11]，レビュー手法についての様々な研究が行われてきた [2] [4] [9].

レビューによる開発効率の向上は，欠陥を早期に発見することによる修正コストや修正確認コスト (修正が期待どおり行われたかを確認するコストと修正に伴い別の欠陥が混入されていないかを確認するコスト) の低減によりもたらされる．つまり，レビューで指摘される欠陥の修正と修正確認コストの合計の方が，テストにおける修正と修正確認コストの合計よりも小さい．例えば，エラーメッセージの誤字脱字のような修正や修正確認コストが小さい欠陥ばかりがレビューで指摘されると，レビューによる効率化の効果が小さいことが予測される．

従来提案されているレビュー手法は欠陥の広範囲な発見に寄与することを主目的としており，発見される欠陥数が大きくなることが期待される一方で，必ずしも見逃された場合の修正や修正確認コストの大小については言及されていない．例えば，主要なレビュー手法の 1 つである Checklist-Based Reading(CBR) [1] では，見つけるべき欠陥を過去の事例や経験に基づいて事前にチェックリスト化し，それを確認しながらレビューを行うが，チェックリスト作成に要するコスト，及びレビュー時の修正や修正確認コストに関する言及はない．

本稿では，修正と修正確認コストを意識したレビューを実施するための第一歩として，テストにおける修正確認コストの主要な要素である回帰テスト件数に着目し，見逃した場合に回帰テスト件数が大きくなると推定される欠陥を優先的に指摘するコードレビュー手法 (Regression Test-Based Reading, RTBR) を提案し，その効果を評価する．本稿では，回帰テストは欠陥修正自体を確認するテストを含めず，欠陥修正時に別の欠陥を混入していないかを確認するテストを指すものとする．実験による評価では，テスト計画書を用いることによりレビュー時に回帰テスト件数を考慮することができると想定し，テスト計画書を用いた提案手法により潜在的に軽減された回帰テスト件数を調べた．比較対象として，特定のレビュー手法を用いることなくレビューアの知識や経験に基づく Ad-hoc Reading(AHR) を用いた．

以降，2 章では，関連研究を説明する．3 章では，回帰テスト件数を考慮したコードレビューについて説明する．4 章では実験概要，レビュー対象，手順，結果について説明する．5 章では考察について議論する．最後に 6 章でまとめと今後の課題について述べる．

2. 関 連 研 究

従来，レビューにおける欠陥検出を効率的，効果的にするために様々な手法が提案されている．代表的なレビュー手法として Checklist-Based Reading (CBR) [1]，Scenario-Based Reading(SBR) [4] がある．SBR には，Perspective-Based Reading(PBR) [3] [8]，Defect-Based Reading(DBR) [6] [7]，Usage-Based Reading(UBR) [9] などの手法が含まれる．

CBR は過去の経験などに基づいて，見つけるべき欠陥をチェックリスト化し，それを用いてレビューを行う手法である．PBR は利用者，設計者，テスト担当者などの観点別に着目すべき事項を観点として明示し，多面的に欠陥を検出する手法である．それぞれのレビューアが異なる観点を持つことで，対象システムに対して綿密なレビューができる．

DBR は定義された欠陥の種類ごとに開発者がレビューを行うことで，効率的に欠陥を検出する手法である．異なる種類の欠陥を複数のメンバが個別にレビューすることで，重複を減らし高い網羅性を得ることができる．CBR，PBR，DBR はより多くの欠陥を検出することに焦点を当てているが，欠陥の優先度を考慮していないところが RTBR と異なる．

UBR はユースケースを用いることで，システムの利用者に大きな影響を与える欠陥を優先的に検出する手法である．RTBR では UBR と同様に欠陥に優先度を割り当てレビューを行うが，UBR ではユースケースに記述されていない動作についてはレビュー対象とならない場合がある．

野中はテストケースを用いてレビューを行う Test Case Based Reading(TCBR) を提案している [5]．テストの為のドキュメントを用いてレビューを行う点において，TCBR と RTBR は類似しているが，TCBR では見逃した際の修正，及び，修正確認コストを考慮しない点で RTBR と異なる．

3. 提 案 手 法

回帰テスト件数を考慮したコードレビュー手法 (RTBR) では，レビューアが回帰テスト件数を想定できるような情報 (例えばテスト計画書) を用いてレビューを行う．そして，欠陥修正に伴う回帰テスト件数が多くなると予想される部分からレビューを行うことで，見逃した場合に大きな回帰テスト工数が必要となる欠陥を中心に指摘でき，テスト時に発見された場合と比較して，修正確認コストを削減することができる．

RTBR では，テスト工数を考慮できるような情報がレビュー実施以前に存在している開発であることを前提としており，対象システムのソースコードに対して以下の手順を繰り返しながら，全ての部分に対して完了するまで，もしくは時間の許す限り，レビューを行う．

（1） テスト工数を考慮できるような情報から回帰テスト件数が最も多くなると予想される部分 (例えば，機能など) を選択する．

（2） 選択した部分が正しく実装されていることを確認し，欠陥を指摘する．例えば，必要な機能は実装されているか，表示は正しく行われるか等である．

（3） 次に回帰テスト件数が多くなると予想される部分を選択しレビューを行う．

図 1 はコーディング完了から回帰テスト終了までに，(a) レビューを行わない場合，(b) 回帰テスト件数が大きくない欠陥を指摘した場合，(c) 回帰テスト件数が大きな欠陥を指摘した場合，それぞれで費やされるであろう修正工数の違いを模式的に示している．図 1 に示す丸印の A，B，C はレビュー，もしくはテストで検出される欠陥であり，それぞれの回帰テスト件数は A ＞ B ＞ C の順で大きいとする．図 1(a) のレビューを行わない場合では，すべての欠陥がテスト工程で発見される．発見された欠陥は修正され，該当箇所に対しての修正確認テストと回帰テストが実施される．従って，図 1(a) では，すべての欠陥がテスト工程で発見されるため，図 1(b),(c) と比べて多くの回帰テストが必要となる．

図 1(b) の回帰テスト件数が大きくない欠陥を指摘した場合では，レビュー時に発見された欠陥を修正してからテストが実施される．そのため，すでに除去された欠陥 (B，C) に対する回帰テスト工数がレビューを行わない場合よりも削減される．

最後に，図 1(c) の回帰テスト件数が大きな欠陥を指摘した場合では，図 1(b) と同様に，レビューによって除去された欠陥に対する修正確認テスト実施の工数が削減される．加えて，図 1(c) では，見逃すことで欠陥 (C) よりも回帰テスト件数増大に

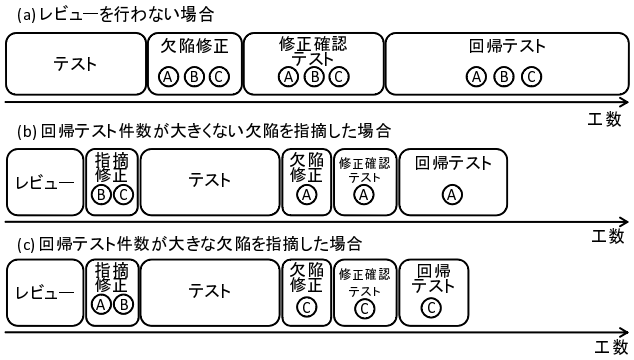

図 1　回帰テスト件数を考慮したレビューを行うときの修正工数

Fig. 1　The defect correction effort with regression test-based reading in the review.

つながるような欠陥 (A，B) が除去されるため，図 1(b) よりも，多くの回帰テスト工数が削減される．

4. 実　　験

4.1 実験概要

本実験では，テスト計画書を用いた RTBR が回帰テスト件数をどの程度減らせるのかを確認することを目的として，テスト計画書を用いた RTBR と AHR の比較を行った．本稿では，レビュー対象システムのテスト計画書をレビュー時に使用することで RTBR を実施する．テスト計画書を用いた RTBR では，テスト計画書に記述されている機能及び予定テスト件数からレビューアが回帰テスト件数を予測する．そして，レビューアが欠陥修正に伴う回帰テスト件数が多くなると予想される機能からレビューを行うことで，修正コストの大きい欠陥を指摘でき，テスト時に発見された場合の修正に伴う回帰テスト工数を削減することができる．

本実験におけるレビューアは，奈良先端科学技術大学院大学の情報科学研究科の博士前期課程及び博士後期課程の学生 8 名である．この 8 名を RTBR を実施するレビューア 4 名，AHR を実施するレビューア 4 名に分割し，比較実験を行った．レビューアの C 言語及びその他のプログラミング言語の経験年数を用いて，スキルの差が生じないようにグループの分割を行った．

実験結果について，RTBR と AHR のそれぞれで削減できた回帰テスト件数，及び回帰テスト削減効率を用いて評価を行う．また，従来，レビュー手法の効果の評価に使用されている発見効率と発見率も用いた評価も行う．

4.2 レビュー対象

レビュー対象は Web ページの入力フォームに情報を入力し，会議等のイベントの参加登録をするためのシステムである．対象は Linux 上で Apache Web サーバの CGI として動作する C 言語で書かれたソースコードである．規模は約 1,500LOC である．本システムは本実験用に，類似システムの業務での開発経験を持つ開発者により開発されたものであり，別の開発者により正しく開発されているか確認をした．図 2 はイベント受付システムの情報のやりとりを示したものであり，個々のイベントを主催するイベント管理者がシステム管理者にイベント作成の依頼をし，イベント作成画面からイベントを作成する．イベントを作成すると作成されたイベントの登録用の Web ページの URL が生成され，システム管理者からイベント管理者に伝えられる．イベント管理者はイベント登録用の URL をイベント参加者にアナウンスし，イベント参加者はイベント参加登録画面を通じて，イベント参加に必要な情報をイベント管理者に送信する．イベント管理者は，イベント管理者画面を通じて，登録者リストを見ることができる．

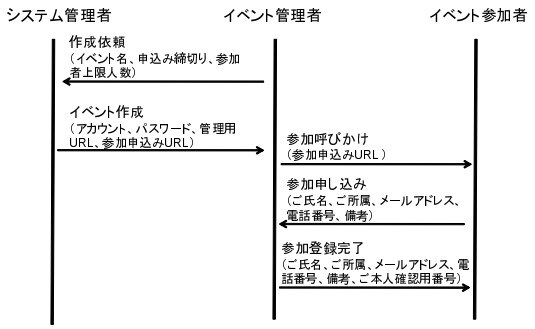

図 2　イベント受付システムの全体像

Fig. 2　The outline of event reception system.

レビューアには外部仕様書，内部設計書，ソースコードが渡される．提案手法でレビューをするグループにはテスト計画書が渡される．外部仕様書，内部設計書は自然言語で書かれている．また，ソースコードは C 言語で書かれている．テスト計画書は表 1 に示すような表形式のものであり，画面の URL(該当する画面が存在する場合)，機能名，予定テスト件数から成っている．

外部仕様書，内部設計書に欠陥は含まれていない．ソースコード中には 25 個の欠陥が含まれている．欠陥 1 件当たりの見逃し時に必要となる回帰テスト件数の最小値，中央値，平均値，及び最大値を表 2 に示す．ここで，単純合算値とは，欠陥の修正に伴い必要となる回帰テスト件数を表 1 のテスト計画書に基づいて，当該機能及び影響を与えられる機能の予定テスト件数を加算した値である．しかしながら，実際に回帰テストを実施する際には明らかに省略できるテストが存在する．明らかに省略できるテスト件数を減らしたものを推定値としている．たとえば，文字列の長さをチェックするライブラリ関数の欠陥を修正した場合，文字列の長さをチェックするテスト件数を全て合算したものが単純合算値であり,「参加者氏名」「参加者住所」のように類似の複数のテストを省略し，1 つのテストにした場合の件数を推定値としている．

なお，今回の実験で使用したレビュー対象は公開する予定である．

4.3 実験手順

評価実験は以下の手順で行った．

（1） レビュー対象としているシステムの説明を行う．

（2） RTBR についての説明を行う．AHR を用いるレビューアにはレビュー手法に関する説明は行わない．

表 1　実験に使用したテスト計画書の一部
Table 1　A part of the test plan document used in the experiment.

機能大分類	URL(該当時のみ)	機能小分類	予定テスト件数
参加登録機能	登録情報入力画面 (index.cgi)	イベント名ブラウザタイトル表示機能	4
		期限切れ確認機能	6
		参加人数超過確認機能	4
		ボタン押下によるページ遷移確認	1
	登録情報確認画面 (registration-confirm.cgi)	「ご氏名」条件判断とエラーメッセージ	7
		「ご所属」条件判断とエラーメッセージ	7
		「メールアドレス」条件判断とエラーメッセージ	8

表 2　欠陥見逃し時に必要となる回帰テスト件数の基本統計量
Table 2　The summary of the regression test cases for omitted faults.

	推定値 (件数)	単純合算値 (件数)
最小値	1	1
中央値	7	14
平均値	10.4	18.9
最大値	71	124

（3） レビュー作業を行う．欠陥を発見した際にはその内容と発見時刻を記録した．

所要時間としては，システムの説明が 30 分，レビュー手法の説明が 10 分，レビュー作業が 60 分で実施した．

4.4 評価尺度

テスト計画書に記述されている予定テスト件数に基づいて，各々のレビュー手法で削減できた回帰テスト件数 (削減件数) を評価する．また，削減回帰テスト件数が上位 50%に含まれる欠陥を見つけた個数 (削減効率) も評価する．削減件数及び削減効率は次の式 (1)，(2) の通り定義する．回帰テスト件数削減につながるような欠陥の指摘が多くできた場合，これらの式によって求められる値は大きくなる．

$$\text{削減件数 (件数/時)} = \frac{\text{全削減回帰テスト件数}}{\text{レビュー時間 (時)}} \quad (1)$$

$$\text{削減効率 (欠陥数/時)} = \frac{\text{削減回帰テスト件数の上位 50\%の発見欠陥数}}{\text{レビュー時間 (時)}} \quad (2)$$

また，文献 [4], [10] で使用されている評価尺度である発見効率と発見率も用いる．発見効率及び発見率はそれぞれ次の式 (3)，(4) で示される．

$$\text{発見効率} = \frac{\text{発見欠陥数}}{\text{レビュー時間 (時)}} \quad (3)$$

$$\text{発見率} = \frac{\text{発見欠陥数}}{\text{全欠陥数}} \quad (4)$$

4.5 結　　果

RTBR と AHR でレビューを行った結果を表 3 に示す．なお，表中の数値は推定値を示しており，カッコ内の値は単純合算値を示している．推定値における削減件数の平均値については RTBR が AHR の約 1.7 倍の値であり，単純合算値における削減件数の平均値については RTBR が AHR の約 1.3 倍の値であることから，RTBR の方が回帰テスト件数をより多く削減できた．推定値における削減効率の平均値については RTBR が AHR の約 2.3 倍の値であり，単純合算値における削減効率の平均値については RTBR が AHR の約 1.25 倍の値であることから，RTBR の方が多くの回帰テスト件数が必要となる欠陥をより多く発見することができた．発見効率及び発見率の平均値は RTBR が AHR の 3.75 倍の値であった．

図 3 は，推定値及び単純合算値において各レビューアが見つけた欠陥により削減される回帰テスト件数の累積を示している．図 3 の横軸はレビュー開始からの経過時間を示し，縦軸は削減された回帰テスト件数を示している．実線は RTBR のテスト件数を，点線は AHR のテスト件数を表しており，黒はそれぞれの推定値を，グレーは単純合算値を示している．AHR では，45 分後までは単純合算値と推定値が同じ値であった (黒とグレーの点線が重なっている)．なお，推定値と単純合算値の累積が同じ場合，推定値の値のみをグラフ上に示したため，単純合算値のグラフの一部が消えている．図 3 において，推定値は開始から 37 分後までは AHR の方が高い累積件数であったが，それ以降は RTBR の方が高い累積件数であった．また，単純合算値は開始から 35 分後までは AHR の方が高い累積件数であったが，それ以降は RTBR の方が高い累積件数であった．

図 4 は，推定値及び単純合算値において各レビューアが見つけた削減回帰テスト件数が上位 50%に含まれる欠陥の累積数を示している．図 4 の横軸はレビュー開始からの経過時間を示し，縦軸は累積欠陥数を示している．実線は RTBR のテスト件数を，点線は AHR のテスト件数を表しており，黒はそれぞれの推定値を，グレーは単純合算値を示している．RTBR では，23 分後までは単純合算値と推定値が同じ値であった (黒とグレーの実線が重なっている)．AHR では，実験開始から終了まで単純合算値と推定値が同じ値であった (黒とグレーの点線が完全に重なっている)．図 4 において，推定値は開始から 23 分後までは AHR の方が高い累積数であったが，それ以降は RTBR の方が高い累積数であった．また，単純合算値は開始から 40 分後までは AHR の方が高い累積数であったが，それ以降は RTBR の方が高い累積数であった．

表 3　RTBR と AHR の実験結果
Table 3　The result of RTBR and AHR.

	RTBR 平均値	RTBR 最小値	RTBR 最大値	AHR 平均値	AHR 最小値	AHR 最大値
削減件数	30(32.75)	0(0)	70(79)	17.75(25.75)	0(0)	64(64)
削減効率	2.25(1.25)	0(0)	6(3)	1(1)	0(0)	3(3)
発見効率	3.75	0	8	1	0	3
発見率	0.15	0	0.32	0.04	0	0.12

() 内の数値は単純合算値を表す

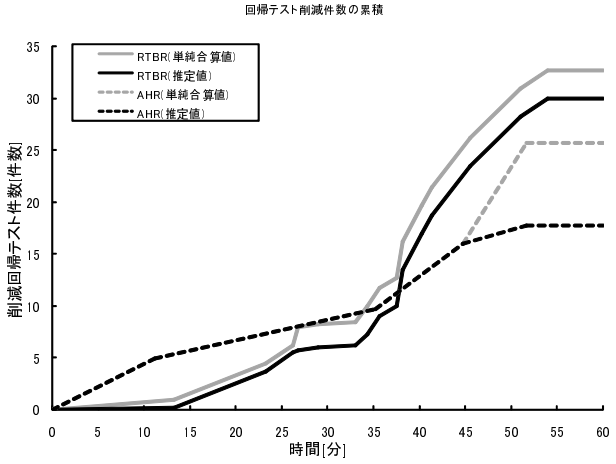

図 3　レビュー中に見つかった欠陥により削減される回帰テスト件数の累積
Fig. 3　The cumulative number of regression test cases by the detected faults in the review.

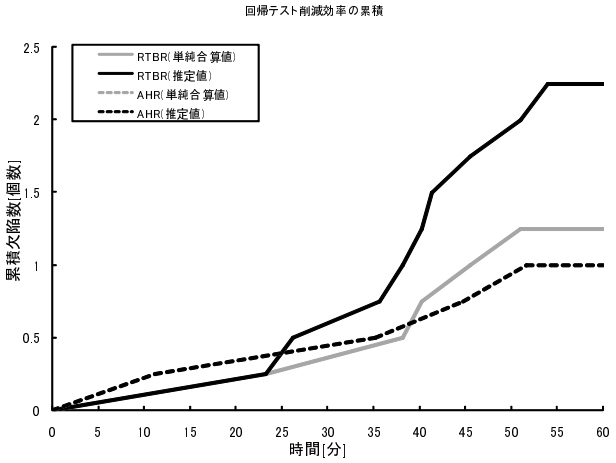

図 4　削減回帰テスト件数が上位 50%に含まれる欠陥の累積数
Fig. 4　The cumulative numbers of the top 50% faults that increases regression test cases.

5. 考　　察

5.1　提案手法の効果

実験の結果から，コードレビューにおいて RTBR を用いたレビューアのほうが，AHR と比べてより多くの欠陥を検出できることが示された．また，回帰テストの件数についても，RTBR によって検出された欠陥のほうが，AHR によって検出された欠陥よりも件数が大きかった．この結果は，RTBR を用いることで，AHR を用いた場合に比べ，回帰テストの件数を削減できる可能性があることを示している．

しかし，表 3 に示されているように，発見効率の平均値については AHR より RTBR が 3.75 倍大きいにも関わらず，削減効率については AHR よりも 2.25 倍の大きさに留まった．RTBR が多くの欠陥を検出しているにもかかわらず，必ずしも回帰テスト件数が大きい欠陥のみを検出しているわけではないことを示しているが，これは，AHR のグループの 1 人のレビューアが AHR の削減効率を大きくしていることが原因である．具体的には，AHR を実施した 4 人のレビューアにより合計 4 つの欠陥が発見され，そのうち 3 件は 1 人のレビューアによって検出されたものである．この 3 件の欠陥は回帰テストの件数がいずれも大きく，削減件数 (1 時間のレビューによって削減できた回帰テスト件数) が 64 と RTBR の最大値に近い値を示している (表 3)．AHR における削減効率については，この 1 人のレビューアによって平均値が押し上げられている．今後，被験者を増やして検討する必要がある．

また，図 3 が示しているように，レビューを開始してから 37 分までは AHR のほうが，回帰テスト件数をより多く削減している．この結果についても，AHR で最も多くの欠陥を検出した 1 人のレビューアがレビューの早期に重大な欠陥を検出した為である．今後，被験者を増やして検討する必要がある．

5.2　実験の妥当性

今回の実験では，提案手法のグループのみにテスト計画書を与えた．そのため，テスト計画書を見ることで，欠陥を容易に推測できる可能性がある．しかし，テスト計画書は類似システムを開発した実務経験者により一般的なテスト項目であることを確認している．従って，テスト項目が直接的に欠陥を示しているわけではないと考える．

設計終了段階でテスト設計が完了している開発であること，もしくは以前のバージョンの開発が存在していることが RTBR の前提になっており，製造段階でテスト計画書が存在しない場合，RTBR の適用はできない．しかし，V 字モデルをはじめとして多くの開発でテスト計画書を早期に作成する場合が増えてきていることや，派生開発や保守開発の割合が大きくなっていることから，それほど大きな制約とはならないと考えられる．

本実験で使用したシステムは規模が小さいため，回帰テスト件数が比較的容易に推測できた可能性があり，規模が大きくなった際にも今回の実験と同じように回帰テスト件数を推測できるとは限らない．スケーラビリティを確保するために，コー

ルフロー解析等の付加情報が必要となるかもしれない．また，本実験のレビューアは全員学生であり，レビューやプログラミングの経験がそれほど長くない．長年，開発に携わっているエンジニアによる比較実験も必要であると考える．

6. おわりに

本稿では，回帰テスト件数が多くなると推定される欠陥を優先的に検出し潜在的な回帰テスト件数を削減することを目的としたレビュー手法を提案した．提案手法 (RTBR) を評価するために，Ad-Hoc Reading(AHR) と比較実験を行い，以下の結果が得られた．

- RTBR の方が回帰テスト件数を AHR よりも約 1.7 倍多く削減できた．
- RTBR の方が AHR よりも約 3.75 倍多くの欠陥を検出することができた．
- RTBR の方が多くの回帰テスト件数が必要となる欠陥を AHR よりも約 2.3 倍多く検出できた．

テスト計画書を用いた RTBR を実施したからといって，必ずしも回帰テスト件数が大きい欠陥のみを検出しているわけではないことが確認された．また，RTBR を実施したからといって AHR を実施した場合よりも，早期に欠陥を発見できるわけではないことも確認された．

なお，RTBR は設計書レビューの実施以前にテスト工数を考慮できるような情報が存在しているような開発であれば，設計書のレビューにも適用が可能であると考えられる．

今後は，コードレビューに適用可能なその他の手法である PBR，CBR 等と比較実験を行う予定である．また，CBR や PBR と同様に，RTBR ではレビュー実施前の準備コストが必要となる．今後の研究では準備のコストも考慮してレビュー手法を評価していく予定である．

謝　辞

実験に参加して下さった奈良先端科学技術大学院大学情報科学研究科ソフトウェア工学講座の皆様に感謝いたします．本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．

文　献

[1] M. E. Fagan, “Design and Code Inspection to Reduce Errors in Program Development”, IBM Systems Journal, Vol.15, No.3, pp.182-211, 1976.

[2] O. Laitenberger, J. DeBaud and P. Runeson, “An Encompassing Life Cycle Centric Survey of Software Inspection”, The Journal of Systems and Software, Vol.50, pp.687-704, 2000.

[3] O. Laitenberger, J. DeBaud and P. Runeson, “Perspective-Based Reading of Code Documents at Robert Bosch GmbH”, Information and Software Technology, Vol.39, No.11, pp.781-791, 1997.

[4] F. Lanubile, T. Mallardo, F. Calefato, C. Denger and M. Ciolkowski, “Assessing the Impact of Active Guidance for Defect Detection: A Replicated Experiment”, Proc. International Software Metrics Symposium (METRICS’04), pp.269-279, 2004.

[5] 野中 誠, “設計・ソースコードを対象とした個人レビュー手法の比較実験”, 情報処理学会研究報告, Vol.2004, No.118, pp.25-31, 2004.

[6] A. A. Porter and L. G. Votta, “An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections”, Proc. International Conference on Software Engineering (ICSE’94), pp.103-112, 1994.

[7] A. A. Porter, L. G. Votta and V. R. Basili, “Comparing Detection Methods for Software Requirements Inspection - A Replicated Experiment”, IEEE Transaction on Software Engineering, Vol.21, No.6, pp.563-575, 1995.

[8] F. Shull, I. Rus, V. Basili, “How Perspective-Based Reading Can Improve Requirements Inspections”, IEEE Computer, Vol.33, No.7, pp.73-79, 2000.

[9] T. Thelin, C. Andersson, P. Runeson, and N. Dzamashvili-Fogelstrom, “A Replicated Experiment of Usage-Based and Checklist-Based Reading”, Proc. International Symposium on Software Metrics(METRICS’04), pp.687-704, 2004.

[10] T. Thelin, P. Runeson, and C. Wholin, “An Experimental Comparison of Usage-Based and Checklist-Based Reading”, IEEE Transaction on Software Engineering, Vol.29, No.8, pp.687-704, 2003.

[11] K. E. Wiegers, “Peer Reviews in Software - A Practical Guide”, Addison-Wesley, 2002.