ソフトウェアレビューの質を参加者へのテストにより評価する試み*

5N－7

的場 政明† ††† 高田 義広† 鳥居 宏次† ††
†奈良先端科学技術大学院大学 情報科学研究科
††大阪大学 基礎工学部　†††日本ユニシス(株)

1 はじめに

高品質のソフトウェアを効率良く開発するためには，テスト工程でバグを検出・修正するだけでなく，それ以前の工程であるレビューによってバグを検出・修正することが有効であることが知られている[1]．従って，ソフトウェア開発を管理する上では，テストだけでなく，レビューやインスペクションが十分に行なわれたかどうか(プロセスの質と呼ぶことにする)を確認することも重要である．ところが，レビューやインスペクションの方法については，多くの提案があるものの，それらのプロセスの質を確認する方法については，ほとんど提案がない[2]．また，テストについては，その質を評価するため尺度(例えば，テスト網羅率や信頼度成長モデルに基づく残存バグ数の推定値)がいくつも提案され実用されているが，レビューやインスペクションについてはほとんどない．レビューに費やした時間や調べた項目の数を尺度とすることも考えられるが，それらの有効性について疑問が残る．レビューやインスペクションの質を評価するための実用的な尺度が望まれる．

そこで，我々は，レビュー直後(または，途中)のレビュー参加者に対して試験を行なうことにより，レビューの質を間接的に定量化することが可能であるかどうかについて検討している．本稿では，コードレビューの直後に試験する方法を提案し，レビュー参加者の試験の成績とレビューで見逃したバグの数との関係について実験結果を報告する．

2 試験方法

提案する試験方法は以下の通りである．

1. レビュー参加者は，ソースコード，プログラム仕様書，チェックリストの3種類の文書を参照しながらコードレビューを行なうものとみなす．
2. ソースプログラムが作成された後からコードレビューが終るまでの間に，試験官(レビュー参加者以外の第3者)が試験問題を作る．試験官は，レビューに使用される3種類の文書を元に問題を作る(3.参照)．
3. レビュー直後(または，途中)のレビュー参加者に試験問題を与える．試験時間は特に限定しない．
4. 回答を採点し，コードレビューの質を間接的に評価する．

なお，ここでは，ソースコード中のバグの有無だけをレビューの対象とするものとみなし，可読性やプログラムの実行効率などを考えない．

3 試験問題

試験問題は次の3個の問題からなる．

問1．コード把握度問題

ソースコードに対する理解の度合を試す問題であり，yes/noで答えられる2種類の小問題からなる．各種類の問題はそれぞれ次のように作られる．

- ソースコード中の分岐文，ループ文，関数呼び出し文，代入文の中から，無作為にいくつかの文を選び出す．そして，各文について，その文の動作を説明する記述を作る．半数の文には，記述ごとに1個の誤りを意図的に混ぜる．これらの記述について，実際のソースコードと一致するかどうかを問う．
- いくつかの関数を無作為に選び出す．そして，その関数の中の関数呼び出しの回数や順番，または，条件分岐の回数や順番を説明する記述を作る．半数の文には，やはり誤りを混ぜる．これらの記述について，実際のソースコードと一致するかどうかを問う．

どちらの問題も，レビューにおいてコードの細部を確認した参加者ほど，正答率が高いと期待される．

問2．仕様把握度問題

プログラム仕様に対する理解の度合を試す問題であり，yes/noで答えられる多数の小問題からなる．各小問題は，プログラムへの入力と，その入力を与えた時のプログラムの出力についての記述からなる．そして，その記

*An Experiment to Evaluate The Quality of Software Review Process by Examining Reviewers
Masaaki Matoba, Yoshihiro Takada, and Koji Torii
Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma-shi, Nara 630-01, Japan

述が仕様書と合致するかどうかを問う．レビューにおいて仕様の細部を確認した参加者ほど正答率が高いと期待される．プログラム仕様書のみを元に作られる．

問 3．コード変更部発見問題

この問題は，問 1 と同様に，ソースコードに対する理解の度合を試す問題である．但し，問 1 とは形式が異なる．問題は，ソースコードのいくつかの箇所を試験官が変更することによって作られる．このソースコードを提示し，どこが変更されたかを問う．やはり，レビューにおいてコードの細部を確認した参加者ほど正答率が高いと期待される．本実験では，ソースコード中の任意の数行が削除された．なお，削除された行数は，予め参加者に知らされた．

4 実験

7 回のコードレビューを独立に行ない，7 回の試験を行なった．各レビューには，説明者と被験者の 2 名が参加した．説明者は，ソースコードの内容をレビュー前から読んでいたが，被験者は読んでいなかった．説明者は，ソースコードの概略を説明するだけの役割を負い，バグを探さなかったし試験も受けなかった．説明者は，7 回とも，筆者の 1 人が勤めた．被験者は，7 回とも異なる．被験者は，説明者の説明を聞きながら実際にバグを探し，レビュー終了後に試験を受けた．7 回のレビューとも，同じチェックリストが使用された．

7 回のレビューとも同じプログラムを対象とした．大阪大学情報工学科の演習で学生が作成した Pascal コンパイラである．C 言語で記述されており，863 行の大きさであった．事前のテストにより，15 個のバグが潜んでいることがわかっていたが，被験者には知らされなかった．被験者は，奈良先端大の 7 名の大学院生が勤めた．各レビューに要した時間は 1.5〜2.5 時間であった．

問 1 として 100 個の小問題，問 2 として 90 個の小問題を与えた．問 3 では 5 箇所を削除したソースコードを提示した．

5 実験結果

7 名の試験の成績と見逃したバグの数を表 1 にまとめる．問 1，問 2 の正答率とは，小問題の総数に対する，正しく答えられた小問題の数の比である．問 3 の正答率は，次式で計算した．

$$\text{問 3 の正答率} = \frac{\text{正しい指摘の個数} - \text{誤った指摘の個数}}{\text{実際に削除した行の数}}$$

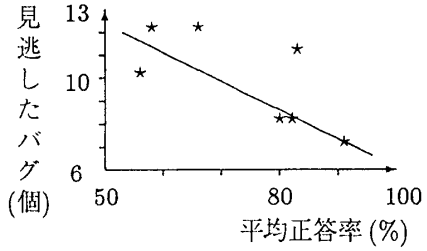
本実験のように，同じプログラムを同じ条件の下でレビューしても，見逃したバグの数が被験者によってかなり異なることがわかる (7〜12 個)．また，問 1〜3 の各問題の正答率は，見逃したバグの数と関連があることがわかる．図 1に試験の平均正答率と見逃したバグの数の相関を示す．縦軸はレビュー時に見逃したバグの数で、横軸は試験の平均正答率を示す．相関係数は 0.701 であった．

表 1: 試験の結果

	A	B	C	D	E	F	G
見逃したバグ (個)	7	8	8	10	11	12	12
問 1 の正答率 (%)	87	88	83	64	82	67	76
問 2 の正答率 (%)	84	69	81	62	83	64	59
問 3 の正答率 (%)	100	/	80	40	80	40	60
平均正答率 (%)	90	79	81	55	82	57	65

(レビュー参加者 B は問 3 に回答しなかった.)

図 1: 平均正答率と見逃したバグの数



6 おわりに

実験の回数は十分とは言えないが，提案した試験の成績と見逃したバグの数とは，関連していることがわかった．今後は，更に実験を繰り返して，次の点を調べる予定である．

- レビューの方法が異なっても，試験の成績と見逃したバグの数とに同じような関係がみられるか?
- 対象とするプログラムの種類が違っても，同じような関係がみられるか?
- 試験の中のどのような種類の問題が，レビューの質の評価に最も役立っているか?

参考文献

[1] Fagan, M. E.: Design and code inspection to reduce errors in program development, IBM Systems J., Vol.15, No.3, pp. 182–211 (1976).

[2] Hollocker, C. P.: Software Reviews and Audits Handbook, John Wiley & Sons (1990).