レビュー作業の質に着目した ソフトウェア最終品質の推定

金村 一弘†, 水野 修†, 菊野 亨†, 高木 徳生‡, 坂本 啓司‡

†大阪大学 基礎工学部 情報科学科
〒560-8531 大阪府豊中市待兼山町 1-3
e-mail: kanemura@ics.es.osaka-u.ac.jp
‡オムロン株式会社

あらまし:
本研究では, ある企業におけるソフトウェア開発におけるレビュー作業の質がソフトウェアの最終品質に与える影響を分析する. レビュー作業の質を単位レビュー工数当たりに発見, 除去された不具合の総数で測ることにし, メトリクスQLY_rで表す. 一方, ソフトウェアの最終品質を出荷後6ヶ月間に発見された不具合の総数で測ることにし, メトリクスSum_fで表す. まず, メトリクスSum_fの値が0とそうでないプロジェクトの間でメトリクスQLY_rの値に有意な差があることを順位和検定で示す. 次に, メトリクスQLY_rの値に基づいてメトリクスSum_fの値が0になるかどうかを推定するためのロジスティック回帰モデルを構成する. 更に, 実際のプロジェクトデータを利用した適用実験でこの回帰モデルの有効性を示した.

キーワード: ソフトウェアレビュー, ソフトウェア最終品質, 統計的分析

Studies on the Effects of Review Efficiency on Field Quality of Software Product

Kazuhiro Kanemura†, Osamu Mizuno†, Tohru Kikuno†,
Yasunari Takagi‡ and Keishi Sakamoto‡

†Department of Informatics and Mathematical Sciences,
Faculty of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
e-mail : kanemura@ics.es.osaka-u.ac.jp
‡OMRON Corporation

Abstract: In this paper we present the results of statistical analyses which try to clarify the effects of review efficiency to the field quality of software product. We define two software metrics QLY_r and Sum_f as follows : QLY_r represents the number of defects that are found and removed per the effort of review activity. Sum_f represents the number of defects that are reported during six months after delivery. First, we show that there exists a significant difference on the values of QLY_r between project with $Sum_f = 0$ and project with $Sum_f > 0$. Then we construct a logistic regression model $E(Y|x)$ which estimates the values of Sum_f based on the values of QLY_r. The experimental results using actual project data show the usefulness of the obtained model.

Keywords: software review, software quality, statistical analysis

1　まえがき

社会システムのコンピュータへの依存度が高まるにつれ, ソフトウェアの開発はより短い開発期間で, しかも少ない人的資源の下で行わねばならなくなっている. その反面, ソフトウェアに要求される品質は年々高いものとなり, 求められる品質を納期内に達成することが非常に難しくなりつつある.

従来のソフトウェア開発プロセスでは, 設計, コーディングに続くテストの工程で不具合の検出とデバッグを行っていた. しかし, 開発プロセスの後工程で発見された不具合を取り除くには多くの工数を必要とするため, テスト工程での作業は開発プロジェクトの致命的な遅れを招くこともあった [4]. 近年, 設計やコーディングといった開発の前工程の中で効果的に不具合を検出, 除去するための手法としてレビュー作業が注目されている [2, 5].

我々は, ある企業におけるレビュー作業の実施状況とその効果について, 次の (1), (2) を示すために統計的な分析を重ねてきた : (1) 適切な量のレビュー作業を実施することによって, 不具合の発見を前倒しできる, (2) レビュー作業を実施することが最終的な品質 (フィールド品質と呼ぶ) の向上につながる. これらの分析の中で, レビュー作業の工数が設計全体に占める割合に注目し, この割合をある値以上に保つことによって不具合の大半をレビュー作業で発見できることを示した.

近年この企業では, 全てのプロジェクトの平均としてみるとレビュー作業の量は確実に増加傾向にある. しかし, 最終的な品質の向上は期待する程には達成されないという状況が観測されている. このことはレビュー作業の量に注目するだけでは, プロジェクトの最終品質の向上に必ずしもつながらないことを示している.

レビュー作業は, 一般に作業者の能力に依存する部分が多い. 同じ時間のレビュー作業を行った場合でも, 作業者によって発見, 除去される不具合の数は大きく異なる. レビュー作業の量を測ったのでは作業に携わる人間の処理能力の差がほとんど反映されていない. そこで, 能力の差を考慮した新しい指標の開発が必要となる. レビュー作業において重要なのは作業の内容, つまりどれくらい不具合を発見, 除去できたかという作業の効果である. そうした指標によってレビュー作業の質を評価することが妥当だと思われる.

本研究では,「単位レビュー工数当たりに発見, 除去された不具合数」で「レビューの質」を計測することを提案する. この「レビューの質」に着目して, プロダクトの最終品質との関係を分析する. 具体的には, レビューの質を測るためのメトリクス QLY_r を導入する. また, プロダクトの最終品質については, 出荷後 6 ヶ月間に発見された不具合の総数を計測することを提案する. そのためのメトリクス Sum_f を導入する.

統計的な分析としては次の 2 つを示す.

(1) Sum_f の値が 0 のプロジェクトとそうでないプロジェクトの間で QLY_r の値に有意な差がある. 分析の手法として順位和検定を用いる.

(2) QLY_r の値を利用して, そのプロジェクトの Sum_f の値が 0 になるかどうかを推定するためのロジスティック回帰モデルが構成できる.

2　準備

2.1　ソフトウェア開発プロセス

本研究で対象としているソフトウェア開発プロセスは標準的なウォーターフォールモデルである. 図 1に示すように, 開発プロセスは上から順に, 計画, 設計, コーディング, テスト, デバッグ, 受け入れテストの工程から構成されている.

この開発プロセスでは, 不具合の発見, 除去はテスト, デバッグ工程で行われる. しかし, 開発の後工程で発見された不具合を除去するためには多大なコストがかかることが知られている [1].

この問題を解決するための有効なアプローチとして, (1) テスト作業の効率化, (2) 早期工程での不具合発見, 除去, などが考えられてる. この中でテスト作業の効率化に関しては, テストアイテムの絞り込み, 不要なテストの省略などの方法が提案されており, 実際に適用してかなりの効果を上げている. しかし, この方法では当初の予測を上回る量の不具合が発見された時などには, 結局多大なコストが必要となる. そこで注目されてきたのが (2) に示す, 開発初期段階での不具合の検出, 除去の手法である.

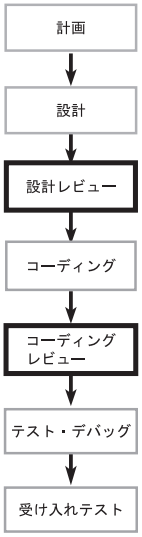

図 1: ウォーターフォールモデル

2.2 レビュー活動

開発の初期段階で不具合を検出, 除去する手法としてよく知られているのがレビュー作業である. レビュー作業は, 設計のある段階やコーディングの終了時に, 少人数 (例えば 2〜3 人) で行われる. 具体的には, 設計ドキュメントと要求仕様書との整合性を調べたり, 設計に基づいた適切な実装が行われているかを調べる.

ソフトウェアの設計工程において作りこまれた不具合の大半を, いわゆる設計レビュー作業の導入によって発見できるということがこれまでの研究によって示されている [2, 5].

我々は 1993 年以来, ある企業の協力の下でソフトウェアプロセスの改善に関する研究を行ってきた. この企業では 8 年ほど前に Software Engineering Process Group(SEPG) が設立され, ソフトウェアプロセス改善を図ってきている. これまでにも幾つかの改善方法を提案してきており, 実際の開発現場でも適用されている.

文献 [7] においては, レビュー作業の量の拡大がソフトウェア最終品質とチームの生産性に与える影響に関する分析が行われている. その中では, レビュー作業の実施率の増加につれてソフトウェアの最終品質 (ここでは, フィールド不具合の出現に関する品質) が向上するという結果が得られた. しかし, ほとんどレビュー作業をしなくても高い品質を示すプロジェクトが多数存在することもこの分析結果から確認されている. そのため, レビューの量だけでソフトウェアの品質を議論することは難しいと考えられた.

また, この企業では, ここ数年の傾向としてレビューの実施率が向上しているのに最終品質が思うように向上していないということも確認されている. このことからも, レビューの量に注目するだけでは, プロダクトの最終品質を説明することが難しいと思われるようになってきた.

2.3 ソフトウェアメトリクス

1. **開発規模** SLC

   SLC は, 再利用分も含めて, 設計, コーディングしたソースコードの行数を表す. 但し, コメント行は全て除いて集計する. なお, 再利用分についてはその修正の度合いに応じて行数を計算している. SLC の定義では次の記号を用いる.

SLC	開発規模 (単位は Kstep である.)
$newSLC$	新規に開発したソースコードの行数.
$slgSLC$	若干の修正を行って再利用したソースコードの行数.
$extSLC$	大幅な修正を行った上で再利用したソースコードの行数.
α, β	補正係数.

   開発規模 SLC は次式によって定義される.

   $$SLC = newSLC + \alpha \times slgSLC + \beta \times extSLC$$

2. **レビューの質** QLY_r

   QLY_r は, レビュー作業によって発見, 除去された不具合数をレビューにかかった工数で割ったものとして定義される. この QLY_r で行ったレビュー作業の品質を測る. QLY_r の定義では次の記号を用いる.

QLY_r	レビューの質 (単位は個 / 人月である.)
$Cost_r$	レビュー作業にかかった工数 (単位は人月である.)
Sum_r	レビュー作業によって発見, 除去された不具合数 (単位は個である.)

   レビューの質 QLY_r は次式によって定義される.

   $$QLY_r = \frac{Sum_r}{Cost_r}$$

3. **レビューの量** QTY_r

   QTY_r は, レビュー作業にかかった工数をそれ自体と設計工数の和で割ったものとして定義される (単位は%である). この QTY_r によって, レビュー作業を含めた総設計工数に対するレビュー工数の割合を表している. QTY_r の定義では次の記号を用いる.

$Cost_r$	レビュー作業にかかった工数 (単位は人月である.)
$Cost_d$	設計作業にかかった工数 (単位は人月である.)

   レビューの量 QTY_r は次式によって定義される.

   $$QTY_r = \frac{Cost_r}{Cost_r + Cost_d} \times 100(\%)$$

4. **不具合密度** ρ

   ρ は, プロジェクトの各段階 (レビュー, テスト・デバック, 受け入れテスト, 出荷後 6 ヶ月での調査) で発見, 除去された不具合の総数を開発規模で割ったものである. ρ の定義では次の記号を用いる.

ρ	作り込まれた不具合の密度 (単位は個 /Kstep である.)
Sum_{t0}	テスト・デバッグ作業の工程で発見, 除去された不具合の総数 (単位は個である.)
Sum_{t1}	受け入れテスト作業の工程で発見, 除去された不具合の総数 (単位は個である.)

Sum_f　　出荷後6ヶ月間に発見された不具合の総数.

作り込まれた不具合の密度 ρ は次式によって定義される.

$$\rho = \frac{Sum_r + Sum_{t0} + Sum_{t1} + Sum_f}{SLC}$$

3　対象とするプロジェクトデータ

ある企業で実施された, 1992 年から 1997 年までのソフトウェア開発プロジェクトの内, ある一定の条件の下に抽出した 51 件のプロジェクトを対象とする. どのプロジェクトもゼロから新規にソフトウェアを作成していく種類のものではなく, 既存のソフトウェアに修正や変更をし, 新規に作成した部分を追加して開発を行うものである.

3.1　プロジェクトのグループ分け

ここでは, メトリクス Sum_f の値を利用して 51 件のプロジェクトをフィールド不具合によって 2 つのグループ $Good$ と $Fair$ に分ける.

一般的にプロダクトの最終品質は, プロジェクトの特性に応じて許容される度合いが異なる. 今回分析の対象としたプロジェクトの特性としては, フィールド不具合の発生は, たとえ一件であっても望ましくないものがほとんどであった. そこで, メトリクス Sum_f の値を利用してプロジェクト群をフィールド不具合の発生しなかったプロジェクトのグループと発生したプロジェクトのグループに分類した. 前者をグループ $Good$, 後者をグループ $Fair$ と呼ぶ. (表 1, 表 2)

表 1: プロジェクトのグループ分け

グループ	Sum_f の値
Good	$Sum_f = 0$
Fair	$Sum_f > 0$

3.2　不具合密度 ρ の差

次に, プロジェクトで作り込んだ不具合の密度 ρ について考える. もし, フィールド不具合の発生したグループ $Fair$ での作り込み不具合密度 ρ の値が, グループ $Good$ に比べ大きい場合, フィールド不具合が多く発生した原因が, 作り込まれた不具合の数の大小によるのではないかと考えられる. そこで 2 つのグループ $Good$ と $Fair$ の標本に対して, それぞれの母集団の分布の中央値に差があるかどうか, つまり, 2 つの分布にずれがあるかどうかを検定する [3].

表 2: プロジェクト分類の内訳

PJ#	QLY_r	グループ
1	0.37	*Fair*
2	2.25	*Good*
3	5.56	*Good*
4	2.76	*Fair*
5	1.62	*Fair*
⋮	⋮	⋮
49	19.62	*Fair*
50	25.19	*Good*
51	27.44	*Good*

グループ $Good$ とグループ $Fair$ の作り込み不具合密度の平均をそれぞれ $\mu^{\rho}_{Good}, \mu^{\rho}_{Fair}$ と表す. それらの値の比較を表 3に示す. また, 順位和検定の結果を表 4 に示す.

表 3: 不具合密度 ρ の平均値

	平均値
μ^{ρ}_{Good}	20.91
μ^{ρ}_{Fair}	17.07

表 4: 順位和検定の結果

	順位平均
Good	30.353
Fair	23.824
p 値	0.1392

表 3に示すように, 作り込み不具合密度の平均値はグループ $Fair$ の方が少ない. また表 4に示す検定結果から, p 値が有意水準 (0.05) 以下ではないので, 2 つのグループ $Good$, $Fair$ の不具合密度の平均において「差がある」とはいえない. つまり, グループ $Good$ と $Fair$ の間では不具合密度の平均値に関してさほど差はないと考えることができる.

4　レビュー作業の質

4.1　仮説の定義

仮説:　　2 つのグループ $Good$ と $Fair$ の間でレビ

ューの質の値に差がある, つまり

$$\mu_{Good}^{QLY_r} > \mu_{Fair}^{QLY_r}$$

が成立する.

4.2 QLY_r の平均値の順位和検定

表1に示す2つのグループ $Good$, $Fair$ について, そのレビューの質 QLY_r の平均値をそれぞれ $\mu_{Good}^{QLY_r}$, $\mu_{Fair}^{QLY_r}$ と表す. それらの値を表5に示す.

表 5: レビューの質 QLY_r 平均値

	平均値
μ_{Good}^{QLYr}	8.41
μ_{Fair}^{QLYr}	4.56

この検定において用いた仮説 (h_0) と対立仮説 (h_1) を次に示す.

(h_0) : 2つのグループ $Good$ と $Fair$ の間でレビューの質の値に差はない, つまり

$$\mu_{Good}^{QLY_r} = \mu_{Fair}^{QLY_r}$$

が成立する.

(h_1) : 2つのグループ $Good$ と $Fair$ の間でレビューの質の値に差がある, つまり

$$\mu_{Good}^{QLY_r} > \mu_{Fair}^{QLY_r}$$

が成立する.

得られた検定結果を表6に示す.

表 6: 順位和検定の結果

	順位平均
Good	33.765
Fair	22.118
p 値	0.0084

表6の p 値 が有意水準 (0.05) 以下である. 従って仮説 h_0 は棄却され, 対立仮説 h_1 が採択される. これによって "2つのグループ $Good$ と $Fair$ の間でレビューの質 QLY_r の値に差がある" ということが統計的に示されたことになる.

4.3 QTY_r の平均値の順位和検定

まず, レビュー作業の量 QTY_r に関して, 先程と同様, グループ $Good$ とグループ $Fair$ の平均をそれぞれ $\mu_{Good}^{QTY_r}$, $\mu_{Fair}^{QTY_r}$ と表す. それらを比較した結果を表7に示す.

表 7: レビューの量 QTY_r の平均値

	平均値
μ_{Good}^{QTYr}	0.141
μ_{Fair}^{QTYr}	0.140

表7から, 平均値にほとんど差はないことが分かる. ここで, 再び順位和検定を用いて分析を行う. その結果を表8に示す.

表 8: 順位和検定の結果

	順位平均
Good	26.588
Fair	25.706
p 値	0.8416

表8より, p 値 が有意水準 (0.05) 以下ではない. 従って, 2つのグループ $Good$ と $Fair$ の間でレビューの量に関して,「差がある」とはいえない. つまり, レビューの量が最終品質に影響を与えているとは思われない.

4.4 考察

4.2より, レビュー作業の質 (つまり QLY_r の値) の大小がフィールド不具合が発生するか否か (つまり, $Sum_f = 0$ かそうでないか) に影響を与えていることが分かった. 一方, 3.2と 4.3の分析より, レビューの量や作り込み不具合数の大小がフィールド不具合の発生するのか否かに大きな影響を与えないことが分かった.

5 最終品質の推定

5.1 推定問題の定義

ここでは, レビューの質が高いプロジェクトで作成されるソフトウェアはその最終品質も高くなるかどうかについて考える. 具体的には, レビューの質 QLY_r の値に基づいてそのプロダクトの Sum_f の値が0になるかどうかを推定する.

2.3の定義より, レビューの質QLY_r は連続変数である. 一方, ソフトウェアの最終品質$FinalQly$はフィールド不具合が発生するかしないかという2値変数であり, $Sum_f = 0$かどうかで評価している.

ここでは, 目的変数が2値の場合に最もよく用いられる回帰モデルであるロジスティック回帰モデル [9] を適用する.

5.2 ロジスティック回帰モデル

ロジスティックモデルは次式

$$E(Y|x_1) = \frac{e^{b_0+b_1x_1}}{1+e^{b_0+b_1x_1}}$$

に基づいている. ここではレビューの質QLY_rを説明変数x_1に, フィールド不具合の有無($Sum_f = 0$かそうでないか)を表した$FinalQly$を目的変数Y に対応させたモデルを作成する. 係数b_0とb_1を表2の分析データを利用して推定した. その結果次に示すモデル式が得られた.

$$E(Y|x_1) = \frac{e^{-1.546+0.142x_1}}{1+e^{-1.546+0.142x_1}}$$

このモデル式に対する適合度や統計的な有意性について, 次節で検定を行う.

5.3 統計的検定

モデルの適合度と統計的有意性に関する検定結果をそれぞれ表9 に示す.

表 9: モデルの適合度

デビアンス	59.26
自由度	49
p値	0.150

- **モデルの適合度**
  モデルの適合度の判断はデビアンスと呼ばれる統計量を用いて行う. 一般にデビアンスが自由度とほぼ同じ値をとると適合度は良いとされる. 一方, p値 については0.05 以下になると適合度が悪いと判断される.

  表9より, デビアンスの値と自由度の値があまり離れておらず, p値 が0.05以下でないことより, 適合度はまずまず良好である.

- **モデルの有意性**
  モデルの有意性検定は, 仮説$H_0 : b_0 = b_1 = 0$ に対する尤度比検定によって行う.

  その結果, p値 $= 0.0173$となった. p値 が0.05 以下であることより, モデルは統計的に有意である.

5.4 モデルによる推定

今回作成したモデルに従い, QLY_rの値からフィールド不具合の発生を予測した. 但し, 新規のプロジェクトについて予測するのではなく, モデル作成に利用した51 件のデータに適用した. 予測の結果を表10に示す.

表中では次の記号を用いる.

- *Good* : フィールド不具合が発生しなかった($Sum_f = 0$となっていた) プロジェクトを表す.
- *Fair* : フィールド不具合が発生した ($Sum_f > 0$ となっていた) プロジェクトを表す.

表10の予測では, レビューの質の値QLY_r をモデル式に代入してフィールド不具合が発生する確率Eを計算した. 確率Eが0.5以上のものを$Fair$とし, 0.5 未満のものは$Good$と推定した. 表10中の数字の単位はプロジェクト数である.

表10中の実測の$Good$と$Fair$は, フィールド不具合が発生しなかったプロジェクトと発生したプロジェクトの実際の数を書いている.

表 10: 予測と実測の比較

実測	予測		一致割合
	Good	*Fair*	
Good	4	13	23.5%
Fair	1	33	97.1%

5.5 考察

表10より, 予測の精度はそれほど高くないことが分かる. 詳細に見ると, $Fair$と予測された46 件のプロジェクトのうち13 件は実際には$Good$であった. しかし, 開発中のプロジェクトに対してフィールド不具合発生の危険を警告するという目的を考えた場合, この予測誤りは大きな問題ではない.

最も注意すべき点は, $Good$と予測されたにもかかわらず, 実際には$Fair$となったプロジェクトである. この状況は極力避けなければならない. 今回の分析においては, 51 件のプロジェクトの中でただ1 件のみがこれに該当していた.

6 まとめ

今回の分析を通して, ソフトウェアの最終品質の向上にレビュー作業が重要な役割を果たすことが分かっ

た. しかもレビュー作業の量よりも質のほうが最終品質に与える影響は強い. 直観的には「質の高いレビューを実施したプロジェクトほど, フィールド不具合が発生しにくい」ということになる.

しかし, 今回の結果はあくまでもデータを提供して頂いた企業の, あるプロジェクト群についての分析結果である. 異なる企業や, 種類の全く異なるプロジェクトについては必ずしも今回の分析結果が当てはまるとは言えない.

更に, ソフトウェアの最終品質に影響を与え得る因子はレビュー以外にも無数に考えられる. レビューの質は, あくまでそれらの一つに過ぎない. このような点に関しては十分に留意しておかなければならない.

今後の課題としては, (1)どれくらいのレビューの質が確保されれば次の工程へ移行すべきか, (2) 質の高いレビュー作業が生産性やその他のコストに対してどのような影響を与えるのか, といった問題に対して統計的な手法で解を見つけることがある.

参考文献

[1] 藤野喜一, 花田収悦： ”ソフトウェア生産技術, ” 電子情報通信学会 (1985).

[2] M. E. Fagan: “Advances in software inspections, ” IEEE Trans. on Software Engineering, Vol.12, No.7, pp.744–751 (1986).

[3] 石村貞夫: ”統計解析のはなし, ”東京図書 (1989).

[4] W. S. Humphrey: *Managing the Software Process*, Addison Wesley, Reading, MA (1989).

[5] R. G. Ebenau and S. H. Strauss: *Software Inspection Process*, McGraw-Hill (1993).

[6] M. C. Paulk, C. V. Weber, S. M. Garcia, M. B. Chrissis and M. Bush: “Key practice of the capability maturity model, version 1.1, ” Technical Report CMU/SEI-93-TR-25, Software Engineering Institute (1993).

[7] Y. Takagi, T. Tanaka, N. Niihara, K. Sakamoto, S. Kusumoto and T. Kikuno: “Analysis of review’s effectiveness based on software metrics, ” Proc. of 6th Int’l Symposium on Software Reliability Engineering(ISSRE’95), pp.34–39 (1995).

[8] T. Tanaka, K. Sakamoto, S. Kusumoto and T. Kikuno: “Improvement of software process by process visualization and benefit estimation, ” Proc. of 17th Int’l Conference on Software Engineering(ICSE’95), pp.123–132 (1995).

[9] 丹後俊郎, 山岡和枝, 高木晴良： ”ロジスティック回帰分析 ”朝倉書店 (1996).

[10] O. Mizuno, T. Kikuno, K. Inagaki, Y. Takagi and K. Sakamoto: “Analyzing effects of cost estimation accuracy on quality and productivity, ” Proc. of 20th Int’l Conference on Software Engineering(ICSE’98), pp.410–419 (1998).

[11] O. Mizuno and T. Kikuno: “Empirical evaluation of review process improvement activities with respect to post-release failure, ” Empirical Studies on Software Development Engineering(ICSE’99 Workshop), pp.50-53 (1999).