ソフトウェア工学　78－7
(1991. 3. 8)

テストコストの減少率に注目したレビュー評価尺度の提案と実験的評価

楠本　真二　　松本　健一　　菊野　亨　　鳥居　宏次

大阪大学　基礎工学部　情報工学科

あらまし　ソフトウェア開発におけるテストコストを小さく抑える有効な手法の一つに技術レビューの実行がある．本報告では，技術レビューによるテストコストの減少率に注目して新しいレビュー評価尺度M_kを提案する．次に，実際のソフトウェア開発過程から収集したデータを使用して従来の評価尺度との比較，評価を行った．その結果，従来の評価尺度に対するM_kの優位性，M_kの有効性が実験的に確認された．更に，レビュー工程で収集されるデータとテスト工程で収集されるデータの関係を調べ，レビュー工程で収集可能なデータのみを利用してM_kの値を推定する方法についても述べる．

A New Metric for Evaluating Cost Effectiveness of Technical Reviews

Shinji KUSUMOTO, Ken-ichi MATSUMOTO, Tohru KIKUNO, and Koji TORII

Department of Information and Computer Sciences
Faculty of Engineering Science
Osaka University
1-1 Machikaneyama, Toyonaka, Osaka 560, Japan

Abstract　In this paper, we present a new metric M_k for evaluating the cost effectiveness of technical reviews during software development. The cost effectiveness is defined to be the degree to which testing costs are reduced by technical reviews. M_k can be interpreted as combined two conventional metrics, M_f by Fagan and M_c by Collofello for technical reviews. Next, by an experimental evaluation of the metrics using data collected during the review and test phases in an industrial environment, we show the validity and usefulness of the proposed metric M_k. Finally, we present a method to estimate the value of the metric M_k using only the data collected during review phase.

1. まえがき

今日，コンピュータシステムにおけるソフトウェアの役割はますます重要になり，ソフトウェア中のフォールト(欠陥)が重大なシステム故障を引き起こすことが少なくない．それに伴い，ソフトウェアのテストに要するコスト(テストコスト)は増大し，その割合がソフトウェア開発コスト全体の50〜80%に達する場合も多くなってきている[6]．従って，テストコストを小さく抑える技術の開発は，ソフトウェア開発における生産性の向上を目指す上で非常に重要である．

ソフトウェアレビューはテストコストを小さく抑える代表的技術の一つである．特に，ソフトウェア開発の各工程で作成されるプロダクトを評価する技術レビュー(Technical review)は実際によく行われている．設計工程で行われる設計レビューとコーディング工程で行われるコードレビューが最もよく知られている．

技術レビューの実行によってプロダクト中のフォールトの多くをテスト工程以前に取り除くことができる．従って，技術レビューの実施によってテストコストは小さくなる．また，一般に技術レビューによるフォールト除去コストがテストによるフォールト除去コストに比べて小さいため[1]，ソフトウェア開発コスト全体も小さくなる．

しかし，技術レビューは知識集約型の作業である．同じ情報を用いて，同じ手順に従って作業を行ったとしても，レビュー作業者の習熟度や対象となるプロダクト(ソフトウェア構成要素)に対する理解度などに差があれば，技術レビューで除去されるフォールト数や技術レビューに要するコストは大きく異なる[3]．従って，テストコストを小さくし，ソフトウェア開発の生産性を向上させるためには，技術レビューの実施技術と共に，技術レビューの評価技術を確立することが重要である．

これまでに，技術レビューの有効性を評価するための尺度(レビュー評価尺度)が幾つか提案されている[2][5]．例えば，Myersは発見されたフォールト数を用いて技術レビューの有効性を評価している[5]．一方，Faganは*Error detection efficiency*というレビュー評価尺度を提案している[2]．この尺度では，技術レビューで発見されたフォールト数とレビュー実施前にレビュー対象となるプロダクト中に含まれていたフォールト数に基づく評価を行う．

最近では，フォールト数だけでなくソフトウェア開発コストに着目したレビュー評価尺度も提案されている[1]．例えば，Collofelloは技術レビューに要したコストと技術レビューによって節約されたコストに基づいて，*Cost Effectiveness*という評価尺度を提案している[1]．しかし，開発コストが大きく異なる2つのプロジェクト間でこの*Cost Effectiveness*を用いた比較を行うことは適切でない．

本研究では，先ず，技術レビューによるテストコストの減少率に基づいて定義される新しいレビュー評価尺度M_kを提案する．次に，従来の尺度との比較を実際のソフトウェア開発過程から収集したデータを使用して行う．定義よりM_kはFaganの評価尺度とCollofelloの評価尺度を組み合わせたものと見ることが可能である．なお，FaganやCollofelloの評価尺度が適用できないような場合にもM_kは適用できる．

以下，2.では本研究で対象とするソフトウェア開発過程とレビューについて述べる．3.では従来の3つの評価尺度について紹介する．4.では技術レビューの有効性を評価する新しい尺度M_kを提案する．5.では，あるコンピュータメーカーの新人研修において収集したデータを使用して，従来の尺度と提案する尺度の比較を行う．最後に6.ではまとめと今後の課題について述べる．

2. 準備

2.1 ソフトウェアレビュー

IEEE標準によると，ソフトウェアレビューは，「開発計画から想定される状況からのズレ(矛盾)を突きとめ，その解消のための改善を促すことを目的として，ソフトウェアの構成要素やプロジェクトの現状を開発者全員が参加して評価する活動[1]」と定義されている[8]．ここで，ソフトウェアの構成要素とはプロジェクト計画書，要求仕様書，設計仕様書，ソースコード等，開発途中で作成される全てのプロダク

[1]文献(8)におけるソフトウェアレビューの定義(原文)は次の通り．An evaluation of software elements or project status to ascertain discrepancies from planned results and to recommend improvements.

トが含まれる．

通常，ソフトウェアレビューは2つのタイプに分類される[8]．一つは，管理レビュー (Management review) であり，もう一つは技術レビュー (Technical review) である．管理レビューではプロジェクトレベルでの開発計画の評価とプロジェクトの現状のその計画に対する評価を行う[8]．一方，技術レビューではソフトウェアの構成要素の評価を行う[8]．本研究では，技術レビュー (特に，設計レビューとコードレビュー) に着目する．以下，技術レビューを単にレビューと呼ぶ．

レビューにおいては次の (1)〜(3) の項目に対する判定結果に基づいて，改善の具体的方法を検討する[8]．

(1) ソフトウェア構成要素が要求仕様書を満足しているか．

(2) プロジェクトで用いている計画，標準，ガイドラインに従って，ソフトウェア構成要素の開発が行われたか．

(3) ソフトウェア構成要素に対する変更は正しく実施され，その影響は変更仕様書 (change specification) によって明確にされた部分にのみ現れているか．

本研究で議論するレビューでは，主に (1) に着目した評価が行われるものとする．

2.2 ソフトウェア開発過程

本研究では，図1に示すソフトウェア開発過程を考える．開発過程は設計工程，コーディング工程，テスト工程の3つの工程に分れる．また，開発はチームによって行われ，最初に与えられた要求仕様書にはフォールトが含まれず，また，開発途中で要求仕様書の変更は行われないと仮定する．

2.3 設計工程

設計工程では要求仕様書に基づいて開発すべきシステムのアーキテクチャ，ソフトウェア構成要素，インタフェース等が決められる．次にそれらが文書化され，要求仕様書を満足することが検証される[7]．通常，設計工程の最後に設計レビューが行われる．設計レビューではチームの全メンバーにより設計ドキュメントが要求仕様書を満たしているかどうかの確認がなされる．

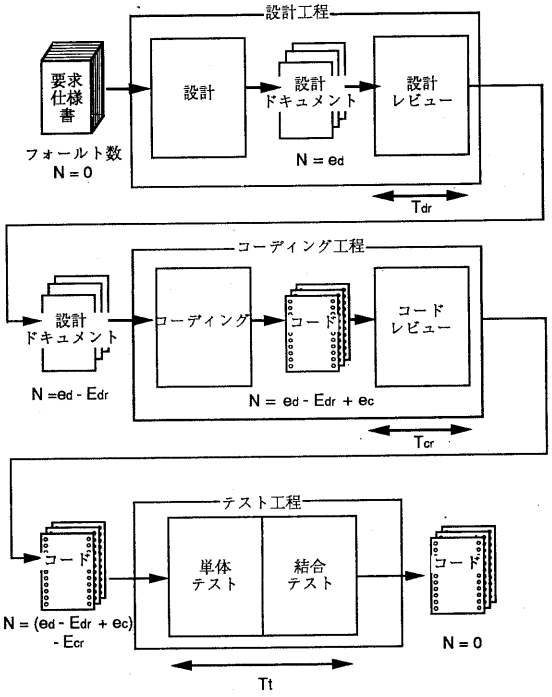

図1 ソフトウェア開発過程

設計作業における設計者の誤りが原因で，設計ドキュメントには e_d 個のフォールトが作り込まれるとする．設計レビューによって設計ドキュメント中の E_{dr} 個のフォールトが除去されるとする．従って，設計工程終了時に設計ドキュメントに含まれるフォールトの数は $(e_d - E_{dr})$ 個となる．なお，設計レビューに要する時間は T_{dr} とする．

2.4 コーディング工程

コーディング工程では，チームの各メンバーが自分に割り当てられた設計ドキュメントに基づいてコードを作成し，そのデバッグを行う．

2.3 で述べたように，設計ドキュメントには $(e_d - E_{dr})$ 個のフォールトが含まれている．この時，設計ドキュメントに含まれていたフォールトが原因で，それと同数のフォールトがコードに作り込まれるものとする．また，コーディング作業におけるコード作成者の誤りが原因で，更に e_c 個のフォールトがコードに作り込まれるものとする．

コーディング工程の最後に，作成されたコードに対してチームの全メンバーによるコードレビューが行われる．コードレビューでは作成されたコードが設計ドキュメントと要求仕様書を満たしているかどうか確認される．コードレビューによってコード中の E_{cr} 個のフォールトが除去されるとする．従って，コーディング工程終了時にコードに含まれるフォールトの数は $\{(e_d - E_{dr}) + e_c - E_{cr}\}$ 個である．なお，コードレビューに要する時間は T_{cr} とする．

2.5 テスト工程

テスト工程では，作成された全てのコードが要求仕様書を満足しているかどうかが評価される．特に最終段階では結合されたコードの評価が行われる[7]．

通常，テスト工程は単体テストと結合テストに分れる．単体テストでは，作成されたコード毎にその動作確認をコード作成者自身によって行い，フォールトを除去する．結合テストでは，作成された全てのコードを結合し，その動作確認を開発者全員によって行い，フォールトを除去する．

単体テストと結合テストによりコード中の $\{(e_d - E_{dr}) + e_c - E_{cr}\}$ 個のフォールトが全て除去されるものとする．ただし，各フォールトの除去が単体テストと結合テストのどちらで行われたかは区別しないものとする．

なお，テスト時間 T_t は各コード作成者による単体テスト時間と開発者全員による結合テスト時間の和とする．

3. 従来の評価尺度

既存のレビュー評価尺度のうち，代表的な 3 つの尺度について簡単に述べる．

3.1 Myers の尺度 M_m

Myers はブラックボックステスト，ホワイトボックステスト，コードレビュー，及び，これらを組み合せた方法の有効性の評価を行った[5]．評価は発見されたフォールト数によって行う．

Myers の考え方を図 1 に適用すると，設計レビューとコードレビューの有効性の評価尺度は次式で表される．

$$M_m = E_{dr} + E_{cr} \tag{1}$$

レビュー対象のソフトウェア構成要素に同数のフォールトが含まれ，かつ，レビューやテストに要するコストが重要でない場合にのみ，M_m をレビュー評価尺度として用いることができる．

3.2 Fagan の尺度 M_f

Fagan は詳細設計レビュー，コードレビュー，単体テストレビューの有効性を評価した[2]．評価において，*Error Detection Efficiency* と呼ばれる評価尺度 M_f を導入した．M_f は総フォールト数に対するレビューでの除去フォールト数の割合で定義される．

Fagan の考え方を図 1 に適用すると，設計レビューとコードレビューの有効性の評価尺度は次式で表される．

$$M_f = \frac{E_{dr} + E_{cr}}{e_d + e_c} \tag{2}$$

この評価尺度 M_f はレビュー対象のソフトウェア構成要素に同数のフォールトが含まれていない場合にも適用できる点で M_m よりも優れている．しかし，M_f もまた，M_m と同様に，レビューに要するコストは評価に用いていない．

3.3 Collofello の尺度 M_c

Collofello はレビューやテスト等のフォールト除去技術をコスト面から評価した[1]．評価において，*Cost Effectiveness* と呼ばれる評価尺度 M_c を提案した．M_c はフォールト除去技術の実施によって節約されたコストとレビューに費されたコストの比で定義される．

Collofello の考え方を図 1 に適用すると，設計レビューとコードレビューに対する有効性の評価尺度は次式で表される．

$$M_c = \frac{\Delta C_t}{C_r} \tag{3}$$

ただし，ΔC_t はレビューによって減少したテストコストの絶対値を，C_r はレビューコストをそれぞれ表す．

レビューによるフォールト 1 個当りの平均除去コストを c_r とすると，レビューコスト C_r は次式で表現される．

$$C_r = (E_{dr} + E_{cr}) \cdot c_r \quad (4)$$

レビューによるテストコストの減少量ΔC_tについて説明する．レビューによって除去された$(E_{dr}+E_{cr})$個のフォールトがレビューではなくテストによって除去されたと仮定した時，それらの除去に必要なテストコストがΔC_tであると考えることができる．今，レビューによってフォールトが除去されなかった場合でも，テストによるフォールト1個当りの平均除去コストは変化しないと仮定する．この時ΔC_tは次式で表現される．

$$\Delta C_t = (E_{dr} + E_{cr}) \cdot c_t \quad (5)$$

ここで，c_tはテストによるフォールト1個当りの平均除去コストである．式(3)(4)(5)より，M_cは次式のように表すことができる．

$$\begin{aligned} M_c &= \frac{(E_{dr} + E_{cr}) \cdot c_t}{(E_{dr} + E_{cr}) \cdot c_r} \\ &= \frac{c_t}{c_r} \end{aligned} \quad (6)$$

すなわち，M_cはレビューによるフォールト1個当りの平均除去コストとテストによるフォールト1個当りの平均除去コストの比となる．

今，レビューコストC_rはレビュー時間$(T_{dr}+T_{cr})$によって，一方，テストコストC_tはテスト時間T_tによって，それぞれ表すことができると仮定する．この時c_r，c_tはそれぞれ次式で表現される．

$$c_r = \frac{T_{dr} + T_{cr}}{E_{dr} + E_{cr}} \quad (7)$$

$$c_t = \frac{T_t}{(e_d + e_c) - (E_{dr} + E_{cr})} \quad (8)$$

4. 評価尺度の提案

4.1 新しい尺度M_k

従来の評価尺度M_m，M_f，M_cの中ではレビューによるテストコストの減少量を考慮しているM_cが最も実用的であると考えられる．

しかし，次のような2種類のソフトウェア開発プロジェクト間でレビューの有効性を比較することを考えた場合，M_cはレビュー評価尺度として適当でないことが分かる．

(1) プロジェクトA—レビューによって節約されたテストコストを100，レビューに要したコストを10とする．

(2) プロジェクトB—レビューによって節約されたテストコストを1000，レビューに要したコストを100とする．

コストの単位はman-months等である．また，プロジェクトA，B共にレビューによってフォールトが除去されなかった場合に必要なテストコストを1000とする．

このとき，プロジェクトA，B共にM_cによる評価値は10となり，レビューの有効性は等しいと結論される．しかし，プロジェクトBにおけるレビューの方がプロジェクトAにおけるレビューよりも有効であることは明らかである．なぜならば，プロジェクトAではレビューを行っても更にテストコストとして900が必要がある．これはレビューによって節約されたコストの9倍の値である．一方，プロジェクトBでは，全テストコストがレビューによって節約されており，テストによってフォールトを除去する必要はない．

こうした事実に基づき，我々は次のような評価尺度M_kを新しく提案する．レビューによって節約されたコストと，レビューによってフォールトが除去されなかった場合に必要となるコストの比としてM_kを定義する．この考え方を図1に適用すると次式が得られる．

$$\begin{aligned} M_k &= \frac{\Delta C_t - C_r}{C_t + \Delta C_t} \\ &= \frac{(E_{dr} + E_{cr}) \cdot c_t - (E_{dr} + E_{cr}) \cdot c_r}{(e_d + e_c) \cdot c_t} \\ &= \frac{E_{dr} + E_{cr}}{e_d + e_c} \cdot \frac{c_t - c_r}{c_t} \end{aligned} \quad (9)$$

4.2 他の尺度との比較

ここでは，提案する尺度M_kと従来の尺度M_m，M_f，M_cの間の関係を議論する．

定義式(2)(6)(9)よりM_k，M_f，M_cの間には次の関係が成り立つ．

$$M_k = \frac{E_{dr} + E_{cr}}{e_d + e_c} \cdot \frac{c_t - c_r}{c_t}$$

$$= \frac{E_{dr}+E_{cr}}{e_d+e_c}\left(1-\frac{c_r}{c_t}\right)$$
$$= M_f\left(1-\frac{1}{M_c}\right) \quad (10)$$

このように，提案する尺度 M_kは従来の尺度 M_f と M_cを組み合わせたものとなっている．

ここで，M_fの値が定数であると仮定すると，上式より M_kは M_cの逆数に比例する．

$$M_k = a_1\left(1-\frac{1}{M_c}\right) \quad (11)$$

但し，a_1は定数である．

次に，c_rと c_tの比が一定であると仮定する．このとき，M_k は M_fに比例する．特に，c_tが c_r に比べて非常に大きい(すなわち，M_cの値が非常に大きい)場合，次式のように M_kと M_fは近似的に等しくなる．

$$M_k = M_f\left(1-\frac{1}{M_c}\right)$$
$$\simeq M_f(1-0)$$
$$= M_f \quad (12)$$

また，$c_r \ll c_t$で，かつ，レビュー前にソフトウェア構成要素に含まれるフォールト数が一定とみなせる場合，次式のように M_kは M_mの定数倍になる．

$$M_k = \frac{E_{dr}+E_{cr}}{e_d+e_c}$$
$$= a_2(E_{dr}+E_{cr})$$
$$= a_2 M_m \quad (13)$$

但し，a_2は定数である．

5. 評価実験

5.1 実験の概要

従来のレビュー評価尺度と新たに提案した評価尺度を実際のソフトウェア開発過程から収集したデータを使用して比較する．データはあるコンピュータメーカの新人研修において収集されたものである．研修の主な特徴は次の通りである．

(1) 7つのチームが，それぞれ，同じシステム(事務処理用のファイル処理システム)をJSP法に従って，プログラミング言語COBOLを用いて作成する．

(2) 各チームは4人～5人で構成されている．チームの構成は研修中の成績等に基づいてチーム間での能力差がないように構成されている．

(3) 開発は各チームが作成したテストデータがすべて正しく実行された時に終了する．なお，テストデータ作成方針は全チームに同一のものが与えられている．

5.2 実験データ

実験データを表1にまとめる．このうち，総フォールト数 (e_d+e_c)，レビューによって除去されたフォールト数 $(E_{dr}+E_{cr})$，レビューコスト C_rは各開発者の申告データに基づいて算出した．一方，テストコスト C_tは自動収集可能な端末使用時間に基づいて算出した[4]．その他の値 c_r，c_t，ΔC_t，及び，$C_t+\Delta C_t$ は定義に基づいて他の実験データより算出した．

表1 実験データ

Team	総フォールト数 ed+ec	レビューで除去されたフォールト数 Edr+Ecr(=Mm)	レビューコスト Cr (= Tdr+Tcr) (min.)	レビューによるフォールトの平均除去コスト cr(=Tt/Mm) (min.)	テストコスト Ct (=Tt) (min.)	テストによるフォールトの平均除去コスト ct (min.)	レビューによるテストコストの減少量 △Ct (min.)	レビューを行わなかった場合のテストコスト Ct+△Ct (min.)
T1	21	8	500	62.5	5947	457.5	3632	9579
T2	30	15	620	41.3	5032	335.5	5000	10032
T3	22	7	510	72.9	7186	479.1	3333	10519
T4	18	6	570	95.0	4367	363.9	2222	6589
T5	28	13	570	43.8	5274	351.6	4642	9916
T6	38	18	560	31.1	4866	243.3	4390	9256
T7	47	15	420	28.0	7521	235.0	3488	11009

6. 分析

6.1 評価値の比較

7つのチーム (T1〜T7) のレビューに対する4つの尺度による評価値を表2にまとめる．表2より，チームT2に対する評価値はどれも高く，逆に，チームT4に対する評価値はどれも低いことが分かる．そこで，4つの尺度による評価値の間の相関係数を調べてみた．その結果を表3にまとめる．

表3より尺度間にはある程度高い相関のあることが分かる．特に，M_kとM_fの間の相関係数の値は0.99と非常に高い．このことは，M_cの値が今回の実験で得られた程度の大きさ (3.90〜8.30) であっても，4.2の式 (12) で表したような関係がM_kとM_fの間に成り立つことを示している．

チームT2とチームT4に対する評価値はどの尺度もほぼ同じであるが，チームT7に対する評価値は尺度によって大きく異なっている．M_f，及び，M_kによる評価値は非常に低い．特に，M_fによる評価値は7チーム中で最も低い．しかし，M_cによる評価値は逆に高く，7チーム中で最も高い．

チームT7の場合，レビューによって除去されたフォールト数が15個と比較的多い．しかし，レビュー時間(レビューコストC_r) は420分で他のチームと比べて約16%〜33%も短い．結果として，レビューによるフォールトの平均除去コストc_rは28.0となり，7チーム中で最小であり，M_cの値を大きくする原因の1つとなっている．一方，チームT7の総フォールト数は47個と非常に多く，テストコストC_tも7521で7チーム中で最大である．従って，チームT7は比較的除去が容易なフォールトだけを短時間のレビューの間に除去したものと考えられる．

こうした現象は，比較的容易に除去できる多くのフォールトと，除去が非常に困難な少数のフォールトがソフトウェア構成要素中に混在するような場合に一般に見られる．この場合のレビューでは除去の容易なフォールトが多数除去されるためc_rは小さくなり，レビューコストC_rも小さくなる．一方，テストでは除去の困難な少数のフォールトのためにc_tは大きくなる．レビューによって除去されたフォールトは除去が容易であるので，それらを仮にテストによって除去することになったとしても，その平均除去コストはc_tよりもかなり小さいと考えられる．従って，c_tに基づいて式 (5) によって求めたテストコストの減少量ΔC_tは実際の値よりも大きくなる．C_rが小さくなると同時に，ΔC_tが大きく評価されるために，M_cの値が非常に大きくなる．

M_kも，M_cと同様に，ΔC_tに基づいて算出される．しかし，定義式 (9) から分るように，たとえΔC_tが実際の値よりも大きく評価されていてもM_kの値が大きくなることはない．むしろ，そうした場合には，テストコストC_tが大きくなるので，M_kの値は小さくなる．

このように，M_cによる評価が妥当ではないと判断されるような場合でも，M_kによる評価は可能である．

6.2 評価値 M_kの予測

提案する尺度M_kでは，総フォールト数$(e_d + e_c)$とテストによるフォールトの平均除去コストc_tを評価に用いている．これらの値はテスト工程が終了するまで知ることができない．従って，M_kによるレビューの評価はテスト終了後に行うことになる．このことはM_kの尺度としての有用性を著しく低くする恐れがある．

表2　レビュー評価値

Team	Mm	Mf	Mc	Mk
T1	8	0.38	7.27	0.327
T2	15	0.50	8.06	0.437
T3	7	0.32	6.54	0.268
T4	6	0.33	3.90	0.251
T5	13	0.46	8.14	0.411
T6	18	0.47	7.84	0.414
T7	15	0.32	8.30	0.279

表3　評価値間の相関

Mf	0.65	——	——
Mc	0.76	0.52	——
Mk	0.72	0.99	0.66
	Mm	Mf	Mc

そこで，レビュー工程での評価を可能とするため，レビュー工程で収集可能な幾つかのデータと(e_d+e_c)，及び，c_tとの関係を調べてみた．その結果を表4にまとめる．表4より，(e_d+e_c)，及び，c_tの逆数は共に，レビューによるフォールトの平均除去コストc_rの逆数と非常に相関の高いことが分る．(e_d+e_c)とc_rの関係を図2に，c_tとc_rの関係を図3に，それぞれ示す．また，(e_d+e_c)とc_r，c_tとc_rの間には次の関係が成り立つ．

$$(e_d+e_c)=4.88+1090\left(\frac{1}{c_r}\right) \qquad (14)$$

$$\frac{1}{c_t}=0.00114+0.0848\left(\frac{1}{c_r}\right) \qquad (15)$$

式(14)(15)を用いて算出した7チームに対するM_kの予測値(レビュー工程でのデータのみを利用して求めた推定値)$\hat{M}_k$とそれぞれの推定誤差を表5にまとめる．表5より推定誤差の平均は約7%であり，レビュー工程のデータのみを利用した精度の高い予測が可能であることが分かる．

7. むすび

本研究では，レビューによるテストコストの減少率に基づく新しいレビュー評価尺度を提案し，実際のソフトウェア開発過程から収集したデータを使用してその評価を行った．更に，レビュー工程で収集

表4 データ間の相関係数

	Edr + Ecr	Tdr + Tcr	Cr	1/Cr
ed+ec	0.84	-0.43	0.88	0.98
1/Ct	0.82	-0.26	0.73	0.92

表5 Mkの推定値と推定誤差

チーム	$\widehat{Mk}$	$\left\|\frac{\widehat{Mk}-Mk}{Mk}\right\|\times100$ (%)
T1	0.303	7
T2	0.420	4
T3	0.289	8
T4	0.286	14
T5	0.376	9
T6	0.400	4
T7	0.300	8

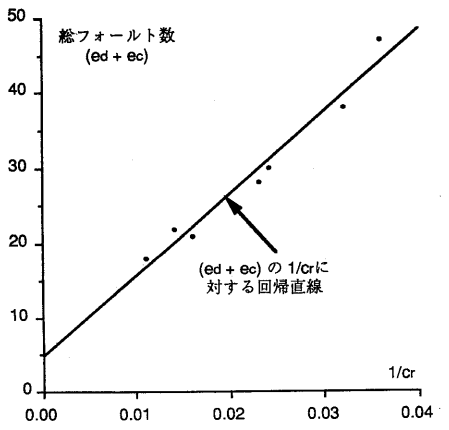

図2 総フォールト数と1/Crの関係

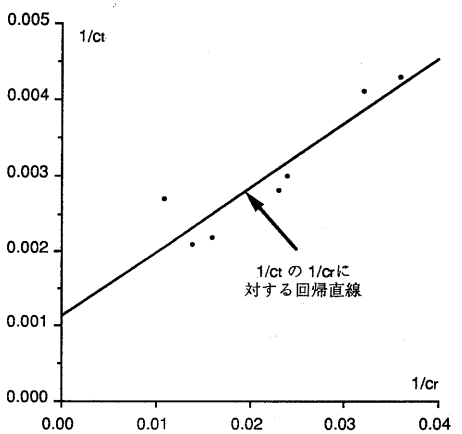

図3 1/Crと1/Ctの関係

されるデータとテスト工程で収集されるデータとの間の関係を調べ，レビュー工程で収集可能なデータのみを利用して，M_kの値を推定する方法について述べた．

尺度によるレビューの評価結果をソフトウェアの生産性向上に役立てるためには，レビューの評価に必要なデータの収集，及び，開発者に対する評価結果のフィードバックを自動的に行うシステムの開発が今後は重要であると考えている．

謝辞 本研究は，一部，平成2年度文部省科研(重点領域(1))「高度ソフトウェアC01」(02249109)の援助を受けている．

文　献

(1) J. S. Collofello and S. N. Woodfield: "Evaluating the effectiveness of reliability-assurance techniques", J. Syst. & Software, Vol.9, No.3, pp.191-195 (1989).

(2) M. E. Fagan: "Design and code inspections to reduce errors in program development", IBM Syst. J., Vol.15, No.3, pp.182-211 (1976).

(3) S. Kusumoto, K. Matsumoto, T. Kikuno and K. Torii: "Experimental evaluation of metrics for review activities", Proceedings of 10th Software Symposium, pp.236-241(1990).

(4) S. Kusumoto, K. Matsumoto, T. Kikuno and K. Torii: "GINGER: data collection and analysis system", IEICE Technical Report, SS90-5 (1990).

(5) G. J. Myers : "A controlled experiment in program testing and code walkthroughs/inspections", Commun. ACM, Vol.21, No.9, pp.760-768 (1978).

(6) G. J. Myers: "The Art of Software Testing", John Willey & Sons, Inc. (1979).

(7) "IEEE Standard Glossary of Software Engineering Terminology," IEEE, ANSI/IEEE Std 729-1983 (1983).

(8) "IEEE Standard for Software Reviews and Audits," IEEE, ANSI/IEEE Std 1028-1988(1988).

盛光印刷所