情報処理学会論文誌 Vol. 50 No. 3 1144–1155 (Mar. 2009)

コードレビュー指摘密度を用いたソフトウェア欠陥密度予測

角田雅照[†1] 玉田春昭[†1,*1] 森崎修司[†1]
松村知子[†1] 黒崎章[†1,*2] 松本健一[†1]

本稿ではコードレビュー指摘密度がソフトウェア欠陥密度予測モデルの精度に与える効果を明らかにする．コードレビュー指摘密度はコードレビューでの指摘数をコード行数で除することにより定義する．ソフトウェア開発では，テストを実施する前に，ソースコードの品質を高めるためにコードレビューが行われている．一般的なソフトウェア開発プロジェクトではまずソースコードレビューが行われ，その後，単体テスト，結合テスト，システムテストが実施される．コードレビューにより欠陥が除去されることから，コードレビューでどの程度欠陥を除去できたかによって，コードレビュー後に実施されるテストにおける欠陥密度が変化する可能性がある．分析では，富士通株式会社において実施された 27 件のプロジェクトで収集されたデータを用いて，単体テスト，結合テスト，システムテスト別に欠陥密度予測モデルを重回帰モデルにより構築した．分析の結果，コードレビュー指摘密度を説明変数に用いない場合，システムテスト欠陥密度の予測はできない（決定係数 0.02）が，コードレビュー指摘密度を説明変数に用いることによりシステムテスト欠陥密度の予測が可能となる（決定係数 0.52）ことが示された．

Software Defect Density Prediction Using Code Review Defect Density

Masateru Tsunoda,[†1] Haruaki Tamada,[†1,*1]
Shuji Morisaki,[†1] Tomoko Matsumura,[†1]
Akira Kurosaki[†1,*2] and Ken-ichi Matsumoto[†1]

This paper clarifies the effect of code review indication density for accuracy of software defect density prediction model. Code review indication density is defined as code review indication number divided by lines of source code. In the software development, code review is conducted before software testing to enhance quality of source codes. Generally, in software development project, source code review was conducted at first, then, unit testing, integration testing, and system testing are done. Results of source code review may influence number of detected faults during software testing. In the analysis, we used 27 projects data recorded in Fujitsu Limited, and made prediction model of defect density of unit testing, integration testing, and system testing with linear regression model. As a result, when the code review defect density is not used for an explanatory variable, the system testing defect density cannot be predicted (the determination coefficient was 0.02), and when the code review defect density is used for an explanatory variable, the system testing defect density can be predicted (the determination coefficient was 0.52).

1. はじめに

ソフトウェア開発プロジェクトにおいて，テスト時における欠陥密度の予測を行うことは非常に重要である．欠陥密度は，欠陥数をソースコード行数で除することにより定義される．一般に，欠陥数とコード行数には強い相関があるため[4)]，欠陥数を行数で除した，行数あたりの欠陥数を示す欠陥密度が測定量として用いられる．予測された欠陥密度は，テストを完了する際の基準となり，予測された欠陥密度に達するまで欠陥を発見する作業が行われる．一般に，ソフトウェア欠陥密度の予測モデルの構築には，行数や複雑度などのソースコードに関する測定量を説明変数とした，重回帰モデルが用いられる．

ソフトウェア開発では，テストを実施する前に，ソースコードの品質を高めるためにコードレビューが行われている．一般的なソフトウェア開発プロジェクトではまずソースコードレビューが行われ，その後，単体テスト，結合テスト，システムテストが実施される．ソースコードレビューでは，複数のレビュアによって，あらかじめ準備されたチェックリストに従って，ソースコードに欠陥が含まれないか，コーディング規則にのっとっているかなどがチェックされる．単体テストでは作成されたモジュール単体のみを用いてテストを行う．結合テストでは，作成した複数のモジュールを組み合わせて動作の確認を行う．システムテストでは，すべてのモジュールを組み合わせ，実際の稼働状態と同様にしてテストを行う．一般に後のテストで発生する欠陥ほど，発生個所の特定が難しく，修正工数が大きくなる．

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
*1 現在，京都産業大学コンピュータ理工学部
Presently with Faculty of Computer Science and Engineering, Kyoto Sangyo University
*2 現在，大阪芸術大学
Presently with Osaka University of Arts

システムテストの後，ユーザによる受け入れテストが実施される．コードレビューにより欠陥が除去されることから，コードレビューでどの程度欠陥を除去できたかによって，コードレビュー後に実施されるテストにおける欠陥密度が変化する可能性がある．

そこで本稿ではコードレビューに着目し，コードレビュー指摘密度が，欠陥密度予測モデルの精度に与える効果を明らかにする．コードレビュー指摘密度はコードレビューでの指摘数をコード行数で除することにより定義する．分析では，単体テスト，結合テスト，システムテスト別に欠陥密度予測モデルを構築する．一般のソフトウェア開発では，単体テスト，結合テスト，システムテストそれぞれのテストごとに，テスト完了のための欠陥密度が設定されるため，実際のソフトウェア開発に予測モデルを適用するためには，各工程別の予測モデルが必要となる．これまで，単体テスト，結合テスト，システムテストそれぞれの欠陥密度とコードレビュー指摘密度との関連を分析した研究や，各工程別の欠陥密度予測モデルの説明変数にコードレビュー指摘密度を用いた場合の効果を明らかにした研究は存在しない．

分析では，富士通株式会社において実施された 27 件のプロジェクトで収集されたデータを用いた．まず，単体テスト欠陥密度，結合テスト欠陥密度，システムテスト欠陥密度とコードレビュー指摘密度を含む測定量との関連を明らかにするとともに，欠陥密度予測モデルの説明変数の候補を決定する．次に，コードレビュー指摘密度を説明変数に加えた予測モデルの精度と，コードレビュー指摘密度を説明変数に加えない予測モデルとの精度を比較し，コードレビュー指摘密度が，欠陥密度予測モデルの精度に与える効果を明らかにする．

以降，2 章では関連研究について述べる．3 章では分析の手順について説明し，4 章で分析に用いたデータについて説明する．5 章で分析を行った欠陥について述べ，6 章で分析結果の考察を行う．最後に 7 章でまとめと今後の課題について述べる．

2. 関 連 研 究

これまで，テスト工程全体や一部のテスト工程における欠陥密度，または出荷後の欠陥密度に着目し，欠陥密度予測モデルを構築した研究は数多く存在するが[6),8),15),21),23)]，本稿のように，単体テスト，結合テスト，システムテストを明確に区別して欠陥密度予測モデルを構築した研究は存在しない．また，単体テスト，結合テスト，システムテストにおける欠陥密度とコード行数，レビュー指摘密度などの測定量との関連を分析した研究も存在しない．多くのソフトウェア開発企業では，単体テスト，結合テスト，システムテストの各テストにおける欠陥密度は別個に管理している．したがって，ソフトウェア開発の現場で欠陥密度予測モデルを適用することを考慮すると，単体テスト，結合テスト，システムテストの各テストに分けて欠陥密度予測モデルを構築することが重要となる．

本稿では，各テストの欠陥密度予測モデルの精度に対するコードレビューの効果を分析しているが，同様にコードレビューに着目して欠陥密度予測モデルを構築した研究が，少数ではあるが存在する．Runeson ら[18)] は，コードレビュー（コードインスペクション）で指摘された件数に基づき，ソフトウェアの残存バグ数を推定するモデルを構築している．また，高田ら[21)] はコードレビュー指摘件数を説明変数の候補の 1 つとして，ソフトウェア欠陥数を予測している．ただし，これらの研究は，単体テスト，結合テスト，システムテストの各テストに分けて分析しておらず，また欠陥密度ではなく欠陥数を予測しており，コードレビュー指摘密度ではなくコードレビュー指摘数を用いている点が本稿と異なる．中野ら[16)] や小室ら[9)] はコードレビュー指摘密度と欠陥密度との関連を分析している．ただし，これらの研究も単体テスト，結合テスト，システムテストの各テストに分けて分析しておらず，また，コードレビュー指摘密度を用いた欠陥密度予測モデルも構築していない．

3. 分 析 方 法

本章では，分析の手順について説明する．分析では，まず各テストにおける欠陥密度と測定量との関連を分析し，次に各欠陥密度予測モデルを構築し，最後にレビュー指摘密度の予測精度に対する効果を分析する．以下に分析の詳細について述べる．

手順 1．各欠陥密度と測定量との関連分析

各テストにおける欠陥密度と（コード行数，レビュー指摘密度などの）測定量との関連を分析し，各欠陥密度と関連の強い測定量を明らかにする．測定量との関連の分析には，ピアソンの順位相関係数[25)] を用いる．ピアソンの順位相関係数では，測定量の値を大きさの順に順位に変換し，順位に基づいて相関係数を計算する．これにより，相関係数に対する外れ値の影響を小さくすることができる．以降，ピアソンの順位相関係数を単に相関係数と記す．また，あわせて相関係数の有意性検定（母相関係数が 0 であるかどうかの検定）を行う．

手順 2．欠陥密度予測モデルの構築

以下の手順により，各テストにおける欠陥密度予測モデルを構築する．

(a) 重回帰分析に基づく欠陥密度予測モデルの仮構築
(b) 外れプロジェクトの特定
(c) 外れプロジェクトを除外したデータによる，予測モデルの構築

本稿の目的は，高い精度で欠陥密度を予測するモデルを構築するために必要な測定量（説明変数）を明らかにすることである．ソフトウェア開発で得られる測定量は多くの場合，手作業

で収集しなければならず，測定量収集の手間を減らすためには，できるだけ収集対象の測定量を絞り込む必要がある．そこで，ソフトウェア欠陥密度予測モデルの構築にしばしば利用されている重回帰分析[4),13),14),22)]を用いて予測モデルを構築し，標準化偏回帰係数を示し，各説明変数が予測にどの程度役立つかを示すことにより，予測に有用な説明変数を明らかにする．

目的変数 y に対して k 個の説明変数 $x_1, x_2, \ldots, x_k$ が与えられる場合，重回帰分析によって得られる重回帰モデルは以下のように表される[12)]．

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon \quad (1)$$

ここで，β_0 は回帰定数，$\beta_1, \beta_2, \ldots, \beta_k$ は偏回帰係数，ε は誤差項である．重回帰分析では，誤差項は正規分布，分散均一性が仮定されている．分散均一性とは，n をデータ数とするとき，誤差項の各誤差 ε_i（$i = 1, 2, \ldots, n$）が同一の分散を持つという仮定である[12)]．作成した重回帰モデルの妥当性を確かめるために，ラグランジュ乗数検定（以降 LM 検定とする）により誤差項の分散均一性を確認し，コルモゴロフ–スミルノフ検定（以降 KS 検定とする）により誤差項の分布の正規性を確認する．

LM 検定は，誤差項の分散均一性を調べる検定法である[12)]．まず，以下の回帰モデルの決定係数を求める．

$$e^2 = \delta_0 + \delta \hat{y}^2 + u \quad (2)$$

ここで，e は残差，$\hat{y}$ は予測値，u はこの回帰モデルにおける誤差項，δ_0 は回帰定数，δ は偏回帰係数である．このモデルの決定係数にデータ数 n を乗じた値が，LM 検定の検定統計量となる．

手順 1 において各欠陥密度と関連の強かった測定量を説明変数の候補とし，重回帰分析の変数増減法により変数を決定する．重回帰分析の変数増減法は以下の手順で行う[24)]．

①どの変数も入っていない状態から開始する．

②もし，すべての変数が含まれていれば，取り込むべき変数はないという情報を持って ③ に進む．すべての変数が含まれていなければ，残りの変数を順番に 1 つずつ採用してみて，偏回帰係数の検定のための F 値を計算し，その値が最大となる変数を選ぶ．その F 値に対応する確率が指定された p_{in} より小さく，かつその変数を採用することになって多重共線性が生じない（VIF が 10 を超えない）ならば，その変数を取りこんで次のステップに進む．p_{in} より大きければ，取りこむべき変数はないという情報を持って ③ に進む．

③モデルに含まれている変数について，偏回帰係数の検定のための F 値を計算し，F 値が最小となる変数を選ぶ．その F 値に対応する確率が指定された p_{out} より小さいとき，取りこむべき変数がないという情報があれば終了する．そうでなければ，どの変数も落とさず ④ に進む．F 値に対応する確率が p_{out} より大きいとき，その変数を落とし（取りこむべき変数がないという情報があれば，それをキャンセルして），再び ③ に戻る．

④すべての変数が取りこまれていれば終了する．そうでなければ ② に戻る．

p_{in}，p_{out} は 0.05 から 0.5 の範囲で，重要な変数を落とさないことに重点をおくなら大きい値，むだな変数を取りこまないことに重点をおくなら小さい値を指定する[24)]．

また，モデル構築時にはリストワイズ除去法[11)]により，欠損値（変数に値が記録されていない）が含まれているプロジェクトを除外する．欠損値処理の方法には，欠損値が含まれているプロジェクトを除外するリストワイズ除去法以外にも，欠損値に各変数の平均値を挿入する平均値挿入法[11)]などがあるが，Strike ら[19)]は，ソフトウェア開発データを用いて予測モデルを構築する場合，リストワイズ除去法による欠損値処理が妥当であることを示している．そこで本稿では Strike らの実験結果に基づき，リストワイズ除去法を適用する．

適切な予測モデルを得るために，外れプロジェクトを除外してモデルを構築する．具体的には，手順 (a) で仮の予測モデルを構築し，手順 (b) で仮の予測モデルを用いて外れプロジェクトを特定し，手順 (c) で外れプロジェクトを除外してモデルを構築する．外れプロジェクトの特定には Cook の距離を用いる．Cook の距離は，あるケースをモデルの推定の計算から除外した場合に，すべてのケースの残差がどの程度変化するかを示す距離である．Cook の距離が大きいときは，回帰統計量の計算からケースを除外したことが係数を実質的に変化させたことを示す[17)]．一般に，Cook の距離が 1 より大きいケースはモデルの分析結果に大きな影響を与えていると見なされるため，それらのケース（プロジェクト）を手順 (c) で除外する．

予測モデルを構築後，モデルに対して分散分析を行い，統計的に有意であるかどうかを調べる．また，VIF と条件指標を用いて，モデルに多重共線性が発生しているかどうかを確認する（一般に，各変数の VIF が 10 を超える場合やモデルの条件指標が 30 を超える場合に，多重共線性が発生しているとされる[20),24)]）．多重共線性が発生している場合，偏回帰係数の分散が大きくなり，偏回帰係数の推定が不安定になる[17)]．また，各変数が欠陥密度に与えている影響を確認するために，偏回帰係数の有意確率と標準化偏回帰係数を調べる．

構築した予測モデルの精度を評価するために，決定係数と自由度調整済み決定係数を用いる．決定係数はデータに対する重回帰モデルの適合の良さ（予測の精度）を評価する指標であり，1 に近いほど重回帰モデルのデータに対する適合度が良いことを示す[12)]．一般に決定係数が 0.5 を超えると，有用なモデルが作成されているといえる[7)]．ただし，重回帰モデル

における決定係数は説明変数の数が増加するに従って高くなる傾向がある[12]．この問題を回避するために，データの数と説明変数の数によって決定係数を補正した，自由度調整済み決定係数を用いることとする．

手順 3．レビュー指摘密度の予測精度に対する効果の分析

レビュー指摘密度の予測精度に対する効果を確かめるために，レビュー指摘密度を説明変数に用いたモデルと，用いないモデルを構築し，それぞれの予測精度の比較実験を行う．以下の手順で実験を行う．

(a) 説明変数を決定するために，手順 2 の方法で予測モデルを作成する．
(b) リーブワンアウト法により学習データとテストデータを作成する．
(c) 学習データを用いてモデルを構築する．モデル構築時には変数選択を行わず，(a) で選ばれた説明変数を用いる．
(d) 構築されたモデルをテストデータに適用し，予測誤差を求める．

リーブワンアウト法は，データセットからケースを 1 件取り出してテストデータとし，残りのケースを学習データとすることを，すべてのケースに対して繰り返す方法である．なお，手順 2 のモデル構築時に Cook の距離が 1 を超えたプロジェクトについては，学習データに含めない（テストデータには含まれる）．

予測精度の評価指標として，絶対誤差を用いる．絶対誤差は予測値と実測値の差の絶対値によって定義され，絶対誤差が小さいほど予測精度が高いモデルであることを示す．絶対誤差の平均値，中央値，標準偏差を計算し，比較を行う．

4. 分析対象データ

4.1 概　　要

分析に用いたデータは，富士通株式会社のプロダクト部門において実施された 27 件のプロジェクトで収集されたものである．これらのプロジェクトはすべて派生開発である．派生開発とは，母体となるプログラムに対し，機能追加や修正を行うことにより，ソフトウェアを開発することを指す．本稿では，重回帰モデルの変数選択時における偏回帰係数の検定を除く，すべての検定（4.2，5.1 節の相関係数の検定，5.2，5.3 節の作成されたモデルにおける偏回帰係数の検定，重回帰モデルの検定，ラグランジュ乗数検定，コルモゴロフ–スミルノフ検定，5.3 節の絶対誤差の差の検定，等分散の検定）において，有意水準を 10%とした．

有意水準を高くしすぎると，第 1 種の過誤を起こす可能性が高くなる．ただし，有意水準を低くしすぎると，逆に第 2 種の過誤を起こす可能性が高くなる．第 1 種の過誤を起こす可能性は有意水準によって決まるのに対し，第 2 種の過誤を起こす可能性は標本数，有意水準，効果量によって決まる[3]．検出力分析[3] を行うと，たとえば本稿で用いた 27 件のデータで，有意水準 5%，効果量 0.3（Cohen[2] が中程度の効果量としている値である）として母相関係数が 0 であるかどうかの検定を行うと，第 2 種の過誤を起こす可能性は 65%と非常に高くなる．第 2 種の過誤を起こす可能性は 20%が望ましいとされている[2]．そこで，第 1 種の過誤と第 2 種の過誤のバランスを考慮し，ソフトウェア工学分野の他の論文[1),4),5),10),18),22)] と同様に有意水準を 10%に設定した．

表 1　分析に使用した測定量
Table 1　Detail of analyzed metrics.

測定量名	詳細
ソフトウェア全体のコード行数	
追加修正部のコード行数	
改造率	追加修正部のコード行数/ソフトウェア全体のコード行数
空行，注釈行率	ソフトウェア全体のコード行数に占める空行，注釈行の割合
関数定義率	ソフトウェア全体のコード 1 行あたりの関数定義数
ファイル定義率	ソフトウェア全体のコード 1 行あたりのファイル定義数
コードレビュー指摘数	コードレビューで指摘された項目数
単体テスト欠陥数	単体テストで発見された欠陥数
結合テスト欠陥数	結合テストで発見された欠陥数
システムテスト欠陥数	システムテストで発見された欠陥数

データに含まれる測定量を**表 1** に示す．データには欠陥数などの試験工程で収集される測定量と，コード行数などのソースコードに関する測定量が含まれている．ソースコードに関する測定量のうち，コード行数については，ソフトウェア全体での行数以外に，追加，修正部分の行数が記録されているが，関数定義数など，その他のソースコードに関する測定量は，ソフトウェア全体での値が記録されている．なお，各測定量には一部欠損値が含まれている（値が記録されていない）．

表 2　コード行数とコードレビュー指摘件数，欠陥数の関係
Table 2　The relationship between lines of code and defects on review, and defects on test.

		コードレビュー指摘件数	単体テスト欠陥数	結合テスト欠陥数	システムテスト欠陥数
ソフトウェア全体のコード行数	相関係数	0.03	0.07	0.18	0.20
	P 値	0.90	0.79	0.39	0.39
	N	24	16	24	21
追加修正部のコード行数	相関係数	0.88	0.94	0.87	0.53
	P 値	0.00	0.00	0.00	0.01
	N	27	19	25	24

4.2　欠陥密度の定義

欠陥密度の定義にあたり，欠陥数とコード行数との関連を分析した．通常，欠陥密度は欠陥数をコード行数で除したものとして定義されるが，分析対象のプロジェクトは派生開発であるため，ソフトウェアの一部のコードはまったく修正を行っていない．したがってソフトウェア全体でのコード行数に基づいて欠陥密度を計算することは適切でない可能性がある．そこで，欠陥密度の定義は，ソフトウェア全体でのコード行数に基づくのが適切か，もしくは追加修正部のコード行数に基づくのが適切かを明らかにするために，単体テスト，結合テスト，システムテストでの欠陥数と，ソフトウェア全体のコード行数との相関，および追加修正部のコード行数との相関を分析した．

分析の結果を**表 2** に示す．各テストでの欠陥数はソフトウェア全体でのコード行数と相関が低くなっていたのに対し，追加修正部のコード行数との相関は高く，統計的にも有意であった．特に単体テストの欠陥数は追加修正部のコード行数と非常に相関が高く，ソフトウェア全体でのコード行数とほとんど相関がなかった．これは，単体テストは追加，修正されたモジュールに対してのみ行われるためであると考えられる．システムテストでの欠陥数と追加修正部のコード行数との相関はあまり大きくなかったが，これはシステムテストでの欠陥数が 0 個であったプロジェクトが多かった（システムテスト欠陥数が記録されていた 24 件のプロジェクト中 11 件）ためであると考えられる．実際に，システムテストでの欠陥数が 0 個であったプロジェクトを除いて分析を行うと，相関係数は 0.82（$p = 0.01$）となった．

同様に，コードレビュー指摘密度は，コードレビューでの指摘件数を追加修正部のコード行数で除して定義する．コードレビューでの指摘件数とソフトウェア全体のコード行数との相関，および追加修正部のコード行数との相関を分析した結果を表 2 に示す．コードレビューでの指摘件数は追加修正部のコード行数との相関が大きく，かつ統計的に有意であり，ソフトウェア全体のコード行数との相関はほとんどなかった．

分析結果に基づき，本稿では追加修正部のコード行数に基づいて欠陥密度を定義する．予備分析を行ったところ，追加修正行数が 1,000 行未満のプロジェクトでは，欠陥数が少しでも変化すると欠陥密度が大きく変化し，欠陥密度予測モデルの構築が困難となった．よって，これらの規模の小さなプロジェクトでは欠陥密度の予測は困難と見なし，分析の対象外とした．以降の分析では，追加修正行数が 1,000 行未満のプロジェクトを除いた，19 件のプロジェクトを用いる．ただし，欠陥密度と測定量との関連分析（3 章の手順 1）においては，分析対象のデータ数が少なくなり，統計的な有意差が出なくなる場合があったため，追加修正行数が 1,000 行未満のプロジェクトを含んだ全プロジェクト 27 件のデータをあわせて分析した．

5. 分 析 結 果

5.1　各欠陥密度と測定量との関連分析

各テストにおける欠陥密度と測定量との関連を分析した結果について述べる．本稿では，全プロジェクトを用いた分析と，追加修正行数が 1,000 行以上のプロジェクトのみを用いた分析のどちらかにおいて，相関係数の絶対値が 0.3 以上の測定量を説明変数の候補とする．各欠陥密度と測定量との相関を**表 3** に示す．

(1)　ソフトウェア全体でのコード行数，追加修正部のコード行数，改造率と欠陥密度の関連分析

1,000 行以上のプロジェクトのみの場合，全プロジェクトの場合のどちらにおいても，ソフトウェア全体でのコード行数は単体テスト欠陥密度と関連があり（相関係数の絶対値が 0.3 以上であった），また 1,000 行以上のプロジェクトの場合，システムテスト欠陥密度とも関連があった．追加修正部のコード行数は，追加修正行数が 1,000 行以上のプロジェクトでは単体テスト欠陥密度と関連があり，全プロジェクトでは結合テスト欠陥密度と関連があった．改造率は 1,000 行以上のプロジェクトの場合，単体テスト欠陥密度，結合テスト欠陥密度と関連があった．

分析結果より，各欠陥密度予測モデルで説明変数の候補を表 3 のように決定した．ただし，1,000 行以上のプロジェクトの場合と全プロジェクトの場合の相関係数を比較すると，これらの測定量は係数が逆転するなど，分析結果に一定の傾向が見られないため，一般的に説明変数の候補とすべきであるかどうかは不明である．

表 3　各テスト欠陥密度と測定量の関連
Table 3　The relationship between defect density of each testing and metrics.

		単体テスト欠陥密度			結合テスト欠陥密度			システムテスト欠陥密度		
		LOC >=1000	全データ	説明変数候補	LOC >=1000	全データ	説明変数候補	LOC >=1000	全データ	説明変数候補
ソフトウェア全体のコード行数	相関係数	-0.38	-0.36		-0.27	-0.19		0.34	0.13	
	P 値	0.25	0.17	○	0.31	0.38		0.22	0.57	○
	N	11	16		16	24		15	21	
追加修正部のコード行数	相関係数	-0.45	0.27		-0.04	0.38		0.13	0.24	
	P 値	0.11	0.26	○	0.88	0.06	○	0.61	0.26	
	N	14	19		17	25		18	24	
改造率	相関係数	-0.15	0.54		0.26	0.46		-0.26	0.14	
	P 値	0.67	0.03	○	0.33	0.03	○	0.35	0.54	
	N	11	16		16	24		15	21	
空行，注釈行率	相関係数	-0.25	-0.20		-0.11	-0.23		-0.18	-0.21	
	P 値	0.45	0.45		0.68	0.27		0.51	0.35	
	N	11	16		16	24		15	21	
関数定義率	相関係数	0.02	0.54		-0.01	0.21		0.16	0.24	
	P 値	0.96	0.03	○	0.97	0.33		0.56	0.30	
	N	11	16		16	24		15	21	
ファイル定義率	相関係数	0.02	0.50		-0.23	0.12		0.11	0.29	
	P 値	0.96	0.05	○	0.40	0.59		0.71	0.20	
	N	11	16		16	24		15	21	
コードレビュー指摘密度	相関係数	0.48	0.37		0.08	0.02		-0.61	-0.48	
	P 値	0.08	0.12	○	0.77	0.92		0.01	0.02	○
	N	14	19		17	25		18	24	
単体テスト欠陥密度	相関係数	1.00	1.00		0.36	0.59		-0.11	-0.06	
	P 値	.	.	-	0.26	0.01	○	0.71	0.82	
	N	14	19		12	17		14	18	
結合テスト欠陥密度	相関係数	0.36	0.59		1.00	1.00		0.46	0.39	
	P 値	0.26	0.01	-	.	.	-	0.07	0.07	○
	N	12	17		17	25		16	22	
システムテスト欠陥密度	相関係数	-0.11	-0.06		0.46	0.39		1.00	1.00	
	P 値	0.71	0.82	-	0.07	0.07	-	.	.	-
	N	14	18		16	22		18	24	

(2)　空行・注釈行率，関数定義率，ファイル定義率と欠陥密度の関連分析

1,000 行以上のプロジェクトのみの場合，全プロジェクトの場合のどちらにおいても，空行・注釈行率と各欠陥密度の関連は弱かった．関数定義率は全プロジェクトの場合，単体テスト欠陥密度のみと有意な関連となり，また，ファイル定義率も全プロジェクトの場合，単体テスト欠陥密度のみと有意な関連となった．よって，ファイル定義率と関数定義率を，単体テスト欠陥密度予測モデルの説明変数の候補とする．

(3)　欠陥密度間の関連分析

1,000 行以上のプロジェクトのみの場合，全プロジェクトの場合のどちらにおいても，単体テスト欠陥密度と結合テスト欠陥密度との相関係数，結合テスト欠陥密度とシステムテスト欠陥密度との相関係数が 0.3 以上であった．よって，単体テスト欠陥密度を結合テスト欠陥密度予測モデルの説明変数の候補とし，結合テスト欠陥密度をシステムテスト欠陥密度予測モデルの候補とする．

(4)　コードレビュー指摘密度と欠陥密度の関連分析

1,000 行以上のプロジェクトのみの場合，全プロジェクトの場合のどちらにおいても，コードレビュー指摘密度は単体テスト欠陥密度とシステムテスト欠陥密度それぞれと有意な相関があった．結合テスト欠陥密度とはほとんど相関が見られなかった．よって，コードレビュー指摘密度を，単体テスト欠陥密度予測モデルとシステムテスト欠陥密度予測モデルの説明変数の候補とする．

5.2　欠陥密度予測モデルの構築

モデルの構築では，4.2 節で述べたように追加修正行数が 1,000 行未満のプロジェクトを除外した，19 件のプロジェクトを用いた．また，本稿では重要な変数を落とさないことに重点を置いたため，重回帰分析の変数増減法において p_{in} に 0.4，p_{out} に 0.5 を指定した．

(1)　単体テスト欠陥密度予測モデル

説明変数の候補は，表 3 のようにソフトウェア全体でのコード行数，追加修正部のコード行数，改造率，コードレビュー指摘密度，ファイル定義率，関数定義率である．モデルの構築を行ったが，Cook の距離が 1 以上となったプロジェクトが 2 件あり，それぞれ 2.3 と 6.2 であった．これらのプロジェクトを除外して再度モデルを構築すると，すべてのプロジェクトの Cook の距離は 1 以下となった．構築したモデルの概要を**表 4** に，係数を**表 9** に示す．このモデルは統計的に有意となり，LM 検定と KS 検定の結果（それぞれ $p = 0.64$, $p = 0.99$），誤差項の分散均一性と分布の正規性に問題は見られなかった．決定係数が 0.5 を超えたことから，予測に有用なモデルが構築されたといえる．また，3 章の手順 2 で述べたように，表 4 の条件指標が 30 以下，表 9 の各変数の VIF が 10 以下であることから，モデルに多重共線性は見られなかったといえる．説明変数にはレビュー指摘密度と改造率が採用され，うちレビュー指摘密度が統計的に有意な説明変数となった．

(2)　結合テスト欠陥密度予測モデル

構築したモデルの概要を**表 5** に，係数を**表 10** に示す．説明変数の候補は，表 3 のように追加修正部のコード行数，改造率，単体テスト欠陥密度である．モデルの構築を行うと，1 件のプロジェクトにおいて Cook の距離が 6.0 となった．このプロジェクトを除外して再度モデルの構築を行うと，すべての Cook の距離は 1 以下となった．構築したモデルは統計

表 4　単体テスト欠陥密度予測モデルの概要
Table 4　Defect density prediction model.

レビュー指摘密度	データ件数	決定係数	自由度調整済み決定係数	P 値	条件指標
使用	9	0.57	0.43	0.08	9.19
不使用	11	0.73	0.55	0.06	13.38

表 5　結合テスト欠陥密度予測モデルの概要
Table 5　Defect density prediction model.

レビュー指摘密度	データ件数	決定係数	自由度調整済み決定係数	P 値	条件指標
不使用	15	0.60	0.49	0.03	4.89

表 6　システムテスト欠陥密度予測モデルの概要
Table 6　System testing defect density prediction model.

レビュー指摘密度	データ件数	決定係数	自由度調整済み決定係数	P 値	条件指標
使用	15	0.52	0.44	0.01	5.82
不使用	15	0.02	-0.14	0.86	3.88

的に有意となり，LM 検定と KS 検定の結果（それぞれ $p = 0.32$，$p = 0.29$），誤差項の分散均一性と分布の正規性に問題は見られなかった．決定係数も 0.5 を超えたことから，予測に有用なモデルが構築されたといえる．表 5 の条件指標，表 10 の各変数の VIF から分かるように，多重共線性も見られなかった．説明変数には追加修正部のコード行数と改造率が採用され，どちらも統計的に有意な説明変数となった．

(3)　システムテスト欠陥密度予測モデル

説明変数の候補は，表 3 のようにソフトウェア全体でのコード行数，コードレビュー指摘密度，結合テスト欠陥密度である．モデルの構築を行ったが，Cook の距離が 1 より大きくなったプロジェクトはなかった．構築したモデルの概要を**表 6** に，係数を**表 11** に示す．モデルは統計的に有意となり，LM 検定と KS 検定の結果（それぞれ $p = 0.12$，$p = 0.12$），誤差項の分散均一性と分布の正規性に問題は見られなかった．決定係数も 0.5 を超えたこと

表 7　単体テスト欠陥密度予測モデルの予測誤差比較
Table 7　Comparison of error of unit testing defect density prediction model.

レビュー指摘密度	データ件数	絶対誤差平均	絶対誤差中央値	絶対誤差標準偏差
使用	11	2.73	2.06	2.57
不使用	11	2.50	2.15	2.10

表 8　システムテスト欠陥密度予測モデルの予測誤差比較
Table 8　Comparison of error of system testing defect density prediction model.

レビュー指摘密度	データ件数	絶対誤差平均	絶対誤差中央値	絶対誤差標準偏差
使用	15	0.75	0.47	0.68
不使用	15	1.06	0.61	1.09

から，予測に有用なモデルが構築されたといえる．表 6 の条件指標，表 11 の各変数の VIF から分かるように，多重共線性も発生していなかった．説明変数にはソフトウェア全体でのコード行数とコードレビュー指摘密度が採用され，コードレビュー指摘密度が統計的に有意な説明変数となった．

5.3　レビュー指摘密度の予測精度に対する効果の分析

レビュー指摘密度の予測精度に対する効果について分析した結果について述べる．絶対誤差は 3 章の手順 3 で述べたように，リーブワンアウト法に基づいてモデルを構築して求めた．なお，結合テスト欠陥密度予測モデルについては，レビュー指摘密度が説明変数の候補とならなかったため，分析対象に含まれていない．

(1)　単体テスト欠陥密度予測モデルの予測精度比較

まず，5.2 節で構築したレビュー指摘密度を説明変数に用いた予測モデルの絶対誤差を求めた．説明変数は 5.2 節で採用されたコードレビュー指摘密度と改造率とした．

次に，レビュー指摘密度を説明変数に用いない予測モデルの絶対誤差を求めた．まず，モデルの説明変数を決定するために，レビュー指摘密度を説明変数に含めずに予測モデルを構築した．説明変数の候補は，表 3 の候補からレビュー指摘密度を除いたものである．モデルの構築を行うと，すべてのプロジェクトの Cook の距離は 1 以下となった．構築したモデルの概要を表 4 に，係数を**表 12** に示す．モデルは統計的に有意となり，LM 検定と KS 検定の結果（それぞれ $p = 0.88$，$p = 0.85$），誤差項の分散均一性と分布の正規性に問題は見られなかった．決定係数も 0.5 を超えていた．また，表 4 の条件指標，表 12 の各変数の

表 9　コードレビュー指摘密度を用いた単体テスト欠陥密度予測モデルの係数
Table 9　Coefficients of unit testing defect density prediction model with code review defect density.

	標準化偏回帰係数	P 値	VIF
コードレビュー指摘密度	0.71	0.04	1.00
改造率	-0.27	0.35	1.00

表 10　結合テスト欠陥密度予測モデルの係数
Table 10　Coefficients of integration testing defect density prediction model.

	標準化偏回帰係数	P 値	VIF
改造率	1.16	0.01	2.28
追加修正部のコード行数	-0.82	0.04	2.28

表 11　コードレビュー指摘密度を用いたシステムテスト欠陥密度予測モデルの係数
Table 11　Coefficients of system testing defect density prediction model with code review defect density.

	標準化偏回帰係数	P 値	VIF
コードレビュー指摘密度	-0.85	0.00	1.46
ソフトウェア全体のコード行数	-0.34	0.19	1.46

VIF から分かるように，多重共線性も見られなかった．変数選択の結果，説明変数にはソフトウェア全体でのコード行数，改造率，ファイル定義率，関数定義率が採用され，うちソフトウェア全体でのコード行数のみが統計的に有意な説明変数となった．これらの測定量を説明変数とし，リーブワンアウト法により絶対誤差を求めた．

単体テスト欠陥密度の予測モデルにおいて，説明変数にレビュー指摘密度を用いても，予測精度が向上するとはいえなかった．レビュー指摘密度を説明変数に用いたモデルと用いなかったモデルそれぞれの欠陥密度の絶対誤差の統計量を**表 7** に，絶対誤差の箱ひげ図を**図 1** に示す．それぞれのモデルの決定係数と自由度調整済み決定係数を比較すると（表 4），どちらもレビュー指摘密度を用いないモデルの精度が高かった．また，絶対誤差の平均値と標準偏差を比較しても（表 7），レビュー指摘密度を用いないモデルのほうが小さくなっていた．絶対誤差の箱ひげ図を見ると，レビュー指摘密度を用いたモデルは誤差の大きなプロジェクトは比較的少ないが，誤差の小さなプロジェクトも少なく，精度が高いモデルであるとはいえない．

表 12　コードレビュー指摘密度を用いない単体テスト欠陥密度予測モデルの係数
Table 12　Coefficients of unit testing defect density prediction model without code review defect density.

	標準化偏回帰係数	P 値	VIF
ソフトウェア全体のコード行数	-1.31	0.05	6.75
関数定義率	0.56	0.40	8.56
改造率	-0.26	0.29	1.13
ファイル定義率	0.30	0.33	1.82

表 13　コードレビュー指摘密度を用いないシステムテスト欠陥密度予測モデルの係数
Table 13　Coefficients of system testing defect density prediction model without code review defect density.

	標準化偏回帰係数	P 値	VIF
ソフトウェア全体のコード行数	0.16	0.60	1.09
結合テスト欠陥密度	0.07	0.81	1.09

(2)　システムテスト欠陥密度予測モデルの予測精度比較

まず，5.2 節で構築したレビュー指摘密度を説明変数に用いた予測モデルの絶対誤差を求めた．説明変数は 5.2 節で採用されたソフトウェア全体でのコード行数とレビュー指摘密度とした．

次に，レビュー指摘密度を説明変数に用いない予測モデルの絶対誤差を求めた．最初に説明変数を決定するために，レビュー指摘密度を説明変数に含めずに予測モデルを構築した．説明変数の候補は，表 3 の候補からレビュー指摘密度を除いたものである．重回帰分析の変数増減法によってモデルの構築を試みると，説明変数の有意確率が低くなり，どの変数も説明変数として採用されなかった．そこで，変数選択を行わずにモデルを構築した．モデルを構築した結果を表 6 に，係数を**表 13** に示す．Cook の距離が 1.6 となったプロジェクトが 1 件あったが，そのプロジェクトを削除してモデルを構築しても，新たに別のプロジェクトの Cook の距離が 1 を超えることが繰り返し発生したため，プロジェクトを除去することは行わなかった．表 6 の条件指標，表 13 の各変数の VIF から分かるように，多重共線性は発生していなかったが，モデルは統計的に有意とならず，LM 検定と KS 検定の結果（そ

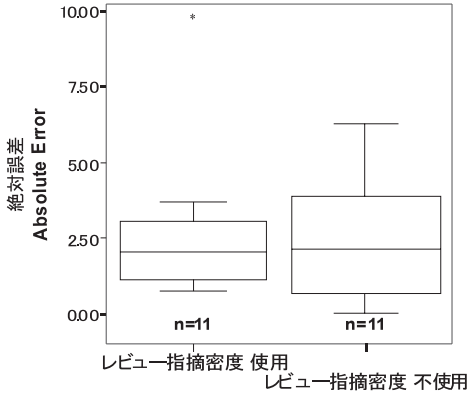


図 1　単体テスト欠陥密度予測モデルの誤差の箱ひげ図
Fig. 1　Boxplots of error of unit testing defect density prediction model.

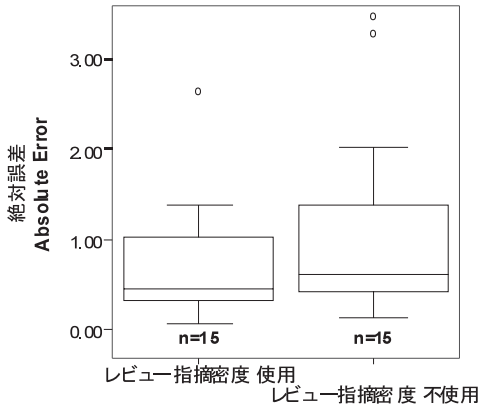


図 2　システムテスト欠陥密度予測モデルの誤差の箱ひげ図
Fig. 2　Boxplots of error of system testing defect density prediction model.

れぞれ $p = 0.75$，$p = 0.07$），誤差項の分散均一性に問題は見られなかったが，誤差項は正規分布しているとはいえなかった．決定係数も非常に小さな値となった．したがって，システムテスト欠陥密度の絶対誤差を求める際には，ソフトウェア全体でのコード行数と結合テスト欠陥密度を説明変数とした．

表 6 に示すように，説明変数にレビュー指摘密度を用いたモデルのほうが，レビュー指摘密度を用いないモデルよりも決定係数と自由度調整済み決定係数が大きくなっていた．表 6 は，コードレビュー指摘密度を説明変数に用いない場合，有意な予測はできない（決定係数 0.02）が，コードレビュー指摘密度を説明変数に用いると有意な予測ができる（決定係数 0.52）ことを示している．すなわち，説明変数にレビュー指摘密度を用いることにより，システムテスト欠陥密度の予測が可能となるといえる．また，表 11 に示すように，レビュー指摘密度の偏回帰係数が大きいことから，コードレビュー指摘密度はシステムテスト欠陥密度の予測に対する効果が高いといえる．

レビュー指摘密度を説明変数に用いたモデルと用いなかったモデルそれぞれの欠陥密度の絶対誤差の統計量を**表 8** に，誤差の箱ひげ図を**図 2** に示す．表 8 に示すように，絶対誤差の平均値，中央値，標準偏差も，レビュー指摘密度を用いたモデルのほうが小さくなっていた．図 2 を見ても，レビュー指摘密度を用いたモデルのほうが誤差が小さいことが分かる．等分散の検定を行った結果，標準偏差の差は統計的に有意であった（$p = 0.09$）が，Wilcoxon の符号付順位和検定を行うと（絶対誤差は正規分布していなかった），絶対誤差の差は統計的に有意ではなかった（$p = 0.36$）．したがって，コードレビュー指摘密度を説明変数に用いた場合，絶対誤差の分散は小さくなるといえるが，絶対誤差の差はあくまで参考結果にすぎないことに注意が必要である．

6. 考　　察

4.2 節の分析結果より，単体，結合，システムテストでの欠陥数は，ソフトウェア全体のコード行数とは関連が弱く，追加修正部のコード行数と関連が強いことが分かった．よって，コードレビューの時間やテスト項目数を決定する際には，追加修正部のコード行数に基づくことが適切であると考えられる．

欠陥密度間の関連を分析した結果，ある工程で欠陥密度が高い場合，次工程の欠陥密度も高くなる傾向が見られた．すなわち，単体テストの欠陥密度が高い場合は結合テスト欠陥密度が高く，結合テスト欠陥密度が高い場合はシステムテスト欠陥密度が高い傾向が見られた．ただし，ある工程での欠陥密度と次々工程の欠陥密度の関連，すなわち単体テストの欠陥密度とシステムテスト欠陥密度の関連は弱かった．分析データの提供元にインタビューを行ったところ，分析対象のプロジェクトのメンバのテストスキルが確立しており，ある工程で欠陥が取りきれなかった場合は，次の工程で欠陥を除去できており，次々工程まで欠陥が残されないことを示しているのではないかとのことであった．

表 12 の単体テスト欠陥密度予測モデルにおいて，ソフトウェア全体のコード行数の標準偏回帰係数は負の値となっており，これはソフトウェア全体のコード行数が増加すると，単体テスト欠陥密度が減少することを示している（なお表 3 に示すように，ソフトウェア全体のコード行数と単体テスト欠陥密度の相関は統計的に有意ではない）．分析データの提供元にインタビューを行ったところ，次のようなコメントが得られた．あくまで感覚的ではあるが，母体規模の大きなプロジェクトは，経験のある開発メンバによって開発が進められる傾向にあり，母体規模の小さなプロジェクトは，成長途上のプロジェクトであるため，経験の浅い開発メンバで構成されることが多くなっている可能性がある．経験の浅い開発メンバは，単体テストでバグ出しをしようとする傾向があるために単体テストの欠陥密度が高くなり，経験の豊富な開発メンバは，単体テストで発見できるようなバグは作りこまないため，単体テストの欠陥密度が低くなる傾向がある．そのためにソフトウェア全体のコード行数の標準偏回帰係数が負の値となっているのではないかとのことであった．

表 11 のシステムテスト欠陥密度予測モデルにおいて，コードレビュー指摘密度の標準偏回帰係数は負の値となっており，コードレビュー指摘密度とシステムテスト欠陥密度も負の相関となっていたことから，レビュー指摘密度が増加すると，システムテスト欠陥密度は減少するといえる．これは，コードレビュー時にシステムテストで発見されるような欠陥を発見し，削除できているためであると考えられる．分析データの提供元にインタビューを行ったところ，分析対象のプロジェクトは保守プロジェクトであり，開発メンバが比較的固定されているため，レビュアのスキルが高く，システムテストで発見されるような欠陥をコードレビューにより発見できているのではないかとのことであった．

コードレビューに着目した従来の研究においても，本稿の分析結果と同様の傾向が見られる．高田ら[21] は欠陥数予測モデルの説明変数の中で最も重要だったものはコードレビュー指摘件数であったと結論付けている．また，小室ら[9] は単体テスト以前の工程での欠陥密度を高めると，後の工程での欠陥密度が低下することを示している．ただし，中野ら[16] はコードレビュー指摘密度が極端に高い場合，後工程での欠陥が増加することを示している．単体テストなどに分けて分析されていないので詳細は不明であるが，本稿における単体テスト欠陥密度とコードレビュー指摘密度に正の相関があったことと同様の現象が見られた可能性がある．

システムテスト欠陥密度予測モデルの精度を向上させることにより，さまざまな効果が期待できる．一般に，後工程のテストで発生する欠陥ほど，発生個所の特定が難しく，工数が大きくなる．システムテスト欠陥密度をより高い精度で予測することにより，システムテストにおいて工数が掛かりそうかどうかを知ることができ，その結果，プロジェクトのスケジュール管理をより適切に行えると考えられる．また，システムテストにおける欠陥密度の予測値は，ソフトウェアを出荷するための品質基準とすることができると考えられる．システムテストでの欠陥を見逃すことは，最もクリティカルである出荷後の欠陥につながる．欠陥密度の予測値を参照することにより，システムテストにおいて欠陥を見逃していないかどうかをより正確に判断可能となることが期待される．

単体テスト欠陥密度とコードレビュー指摘密度に関連が見られたにもかかわらず，コードレビュー指摘密度を説明変数に加えた予測モデルの精度が向上しなかった理由について考察する．単体テストの欠陥密度はコードレビュー指摘密度に基づいて予測することもできるが，コードレビュー指摘密度を特に考慮しなくても，ソースコードに関する測定量のみによっても予測可能であると考えられる．そのため，コードレビュー指摘密度は予測精度のさらなる向上には役立たなかったと考えられる．

本稿で構築した欠陥密度予測モデルは，決定係数がそれほど大きくなく，精度向上の余地があると考えられる．説明変数に用いた空行・注釈行率，関数定義率，ファイル定義率はソフトウェア全体でのコードから計測されている．高橋ら[23] は追加修正行部分の複雑度に関する測定量を用いることにより，欠陥密度予測モデルの精度が向上することを示しており，それらを説明変数に加えることにより，予測モデルの精度向上が期待できる．

7. ま と め

本稿では，コードレビュー指摘密度を欠陥密度予測モデルの説明変数に取り入れることを提案した．ソフトウェア開発企業において収集された派生開発プロジェクトのデータを分析し，その分析に基づき，欠陥密度予測モデルを構築した．その結果，以下のことが明らかになった．

- 単体，結合，システムテストでの欠陥数は，ソフトウェア全体のコード行数とは関連が弱く，追加修正部のコード行数と関連が強い．
- 単体テスト欠陥密度と結合テスト欠陥密度，結合テスト欠陥密度とシステムテスト欠陥密度に関連がある．ただし，単体テスト欠陥密度とシステムテスト欠陥密度の関連は弱い．
- コードレビュー指摘密度は単体テスト欠陥密度と正の相関があり，システムテスト欠陥密度と負の相関がある．
- コードレビュー指摘密度はシステムテスト欠陥密度予測の予測に対する効果が高い．

今後の課題は欠陥密度予測モデルの精度を向上させることである．本稿で構築された欠陥密度予測モデルは十分に精度が高いとはいえないが，追加修正行部分のソースコードに関する測定量をモデルの説明変数に加えることにより，精度が向上することが期待される．また，重回帰分析以外を用いて予測モデルを構築し，予測精度を確かめることも今後の課題である．

謝辞　本稿執筆にあたり，ご協力いただきました富士通株式会社の上田直子様，伊藤雅子様に深謝いたします．本研究の一部は，文部科学省「e-Society 基盤ソフトウェアの総合開発」および「次世代 IT 基盤のための研究開発」の委託に基づいて行われた．

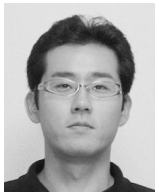
参 考 文 献

1) Blackburn, J., Scudder, G. and Wassenhove, L.: Improving Speed and Productivity of Software Development: A Global Survey of Software Developers, *IEEE Trans. Softw. Eng.*, Vol.22, No.12, pp.875–885 (1996).
2) Cohen, J.: *Statistical power analysis for the behavioral sciences* (*2nd Edition*), p.567, Lawrence Erlbaum Associates, Associates Mahwah, NJ (1988).
3) Field, A. and Hole, G.: *How to design and report experiments*, p.384, Sage Publications, London (2003).
4) 冨士　仁，古山恒夫，菅野文友：ソフトウェアの欠陥密度に影響を与える複雑さの特質とその尺度の分析，品質，Vol.26, No.3, pp.91–101, 日本品質学会 (1996).
5) 古山恒夫，菊地奈穂美，安田　守，鶴保征城：ソフトウェア開発プロジェクトの遂行に影響を与える要因の分析，情報処理学会論文誌，Vol.48, No.8, pp.2608–2619 (2007).
6) 本田紘介，土肥　正，岡村寛之：モジュールサイズの分布に基づいたソフトウェア欠陥密度の評価に関する考察，電子情報通信学会論文誌，Vol.J86-A, No.6, pp.713–717 (2003).
7) 菅　民郎：Excel で学ぶ多変量解析入門，p.266，オーム社 (2001).
8) Knab, P., Pinzger, M. and Bernstein, A.: Predicting defect densities in source code files with decision tree learners, *Proc. International Workshop on Mining Software Repositories*, pp.119–125 (2006).
9) 小室　睦，男澤　康，木村好秀：開発現場の実態に基づいたピアレビュー手法改善と改善効果の定量的分析，*SEC journal*, Vol.1, No.4, pp.6–15 (2005).
10) Lee, N. and Litecky, C.: An Empirical Study of Software Reuse with Special Attention to Ada, *IEEE Trans. Softw. Eng.*, Vol.23, No.9, pp.537–549 (1997).
11) Little, R. and Rubin, D.: *Statistical Analysis with Missing Data*, *2nd ed.*, p.408, John Wiley & Sons, New York (2002).
12) 御園謙吉，良永康平（編）：よくわかる統計学 II 経済統計編，p.228，ミネルヴァ書房 (2007).
13) Munson, J. and Khoshgoftaar, T.: Regression Modeling of Software Quality: Empirical Investigation, *Information and Software Technology*, Vol.32, No.2, pp.106–114 (1990).
14) Nagappan, N., Williams, L., Hudepohl, J., Snipes, W. and Vouk, M.: Preliminary Results On Using Static Analysis Tools For Software Inspection, *Proc. 15th International Symposium on Software Reliability Engineering*, pp.429–439 (2004).
15) Nagappan, N. and Ball, T.: Static Analysis Tools as Early Indicators of Pre-Release Defect Density, *Proc. 27th International Conference on Software Engineering*, pp.580–586 (2005).
16) 中野裕也，水野　修，菊野　亨，阿南佳之，田中又治：コードレビューの密度と効率がコード品質に与える影響の分析，*SEC journal*, Vol.2, No.4, pp.10–17 (2006).
17) 小野寺孝義，山本嘉一郎（編）：SPSS 事典：BASE 編，p.280，ナカニシヤ出版 (2004).
18) Runeson, P. and Wohlin, C.: An Experimental Evaluation of an Experience-Based Capture-Recapture Method in Software Code Inspections, *Empirical Software Engineering*, Vol.3, Issue 4, pp.381–406 (1998).
19) Strike, K., El Eman, K. and Madhavji, N.: Software Cost Estimation with Incomplete Data, *IEEE Trans. Softw. Eng.*, Vol.27, No.10, pp.890–908 (2001).
20) Tabachnick, B.G. and Fidell, L.S.: *Using Multivariate Statistics* (*3rd Edition*), p.880, Harper Collins College Publishers, New York (1996).
21) 高田義広，松本健一，鳥居宏次：ニューラルネットを用いたソフトウェア信頼性予測モデル，電子情報通信学会論文誌 D-I，Vol.J77-D-I, No.6, pp.454–461 (1994).
22) Takahashi, M. and Kamayachi, Y.: An Empirical Study of a Model for Program Error Prediction, *IEEE Trans. Softw. Eng.*, Vol.15, No.1, pp.82–86 (1989).
23) 高橋良英：C 言語ソフトウェア保守工程における Halstead のソフトウェアサイエンス計測と障害密度との関係の分析，電子情報通信学会論文誌 D，Vol.J82-D1, No.8, pp.1017–1034 (1999).
24) 田中　豊，垂水共之（編）：Windows 版統計解析ハンドブック多変量解析，p.240，共立出版 (1995).
25) 田中　豊，垂水共之（編）：Windows 版統計解析ハンドブックノンパラメトリック法，p.164，共立出版 (1999).

(平成 20 年 4 月 30 日受付)
(平成 20 年 12 月 5 日採録)

角田　雅照（正会員）

1997 年和歌山大学経済学部卒業．2007 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学同研究科特任助教．博士（工学）．ソフトウェアメトリクス，ユビキタスコンピューティングの研究に従事．電子情報通信学会，ヒューマンインタフェース学会，IEEE 各会員．



玉田　春昭（正会員）

1999 年京都産業大学工学部情報通信工学科卒業．2001 年同大学院博士前期課程修了．2006 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学産学官連携研究員．2007 年同大学情報科学研究科特任助教．2008 年京都産業大学コンピュータ理工学部助教．博士（工学）．ソフトウェアプロテクション，エンタープライズアプリケーションの研究に従事．電子情報通信学会，IEEE 各会員．

森崎　修司（正会員）

2001 年奈良先端科学技術大学院大学情報科学研究科博士課程修了．同年（株）IIJ にてプロダクトマネージャ，RFID ソフトウェアの国際標準策定に従事．2005 年奈良先端科学技術大学院大学研究員，2007 年同大学情報科学研究科助教にて，ソフトウェアレビュー，ソフトウェア計測を中心としたエンピリカルソフトウェア工学，ソフトウェアタグの研究に従事．博士（工学）．IEEE 会員．

松村　知子

2004 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年より同大学産学官連携研究員．博士（工学）．ソフトウェアのフォールト検出技術，ソフトウェア開発プロジェクト管理，エンピリカルソフトウェア工学の研究に従事．電子情報通信学会，IEEE 各会員．

黒崎　　章

2006 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．2007 年同大学情報科学研究科特任助教．2008 年大阪芸術大学入職．博士（工学）．ヒューマンファクタ，ヒューマンインタフェースの研究に興味を持つ．化学工学会，自動車技術会各会員．

松本　健一（正会員）

1985 年大阪大学基礎工学部情報工学科卒業．1989 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．1993 年奈良先端科学技術大学院大学情報科学研究科助教授．2001 年同大学教授．工学博士．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集/利用支援の研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．