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.

ソフトウェア開発における不具合発見履歴と最終品質の関係に対する統計的分析

天嵜　聡介† 　吉富　隆† 　水野　修†† 　菊野　亨†† 　高木　徳生††,†††

† 大阪大学 大学院基礎工学研究科 情報数理系専攻
†† 大阪大学 大学院情報科学研究科 情報システム工学専攻
〒 560–8531 大阪府豊中市待兼山 1–3
††† オムロン株式会社 ソーシアルシステムズカンパニー
E-mail: †amasaki@ics.es.osaka-u.ac.jp,††{o-mizuno,kikuno}@ist.osaka-u.ac.jp,†††taka@kusatsu.biwa.omron.co.jp

あらまし　本研究では，ソフトウェア開発プロジェクトのコードレビュー以降の工程における工程ごとの不具合発見履歴に注目し，ソフトウェアプロダクトの最終的な品質との関係を実際のプロジェクトデータを用いた統計的分析によって明らかにする．具体的には，まず，コードレビュー以降の工程の不具合発見履歴によってプロジェクトを 4 つのクラスに分類した．次に，分類したクラス間でそれぞれのソフトウェアプロダクトの最終品質に有意な差があることを統計的仮説検定によって示した．最後に，不具合発見履歴を特徴づける要因を特定するために，コードレビュー以前の工程で収集できるメトリクスデータに対する分析を行った．ロジスティック回帰分析を用いた分析の結果，幾つかのメトリクスデータが不具合発見履歴に影響を与えていることを統計的に確認した．

キーワード　ソフトウェアテスト，ソフトウェア品質， 統計的分析

Statistical analysis of relationship between fault detection trend and field quality of software product

Sousuke AMASAKI†, Takashi YOSHITOMI†, Osamu MIZUNO††, Tohru KIKUNO††, and Yasunari TAKAGI††,†††

† Graduate School of Engineering Science, Osaka University
†† Graduate School of Information Science and Technology, Osaka University
1–3 Machikaneyama, Toyonaka-shi, Osaka 560–8531, Japan
††† Social Systems Company, OMRON Corporation
E-mail: †amasaki@ics.es.osaka-u.ac.jp,††{o-mizuno,kikuno}@ist.osaka-u.ac.jp,†††taka@kusatsu.biwa.omron.co.jp

Abstract　In this paper, we clarify a relationship between the fault detection trend (that is, a change of the number of faults detected by four successive testings) and the field quality of software product. The relationship is originally found and believed by developers in a certain company. For this purpose, we firstly define four types of trends using rank correlation coefficient, and classify the projects into four classes. Next, as a result of comparison of field quality, we derived a proposition that software product developed by project with the trend of decreasing type has high software quality. Furthermore, in order to find the factors that have strong influence on the fault detection trend, we performed the logistic regression analysis. As a result, we got two principal metrics which may affect the fault detection trend with statistical significance.

Key words　software testing, software quality, statistical analysis

1. まえがき

近年，ソフトウェアの社会的な位置づけがますます重要性を帯びてくると共に，その品質を保証することが重要な目標となっている [1]〜[3], [11], [20]．ソフトウェアの品質は，通常そこに含まれる不具合の数によって評価される．つまり，ソフトウェア開発においてソフトウェアプロダクト中に作り込まれた不具合の数と，検出され修正された不具合の数によって決定される．従って，設計・コーディング工程において作り込まれる不具合を少なくし，テスト・デバッグ工程において検出・修正される不具合を多くすることが高品質なソフトウェアを作成することにつながる．

特に，品質を決定する最後の作業がテスト・デバッグ工程であることから，テスト・デバッグ工程で発見される不具合と最終品質の関係は長く研究されている．その成果として幾つかのソフトウェア信頼度成長モデルが提案され [20]，実際のプロジェクトに適用した事例も多く報告されている [15].

しかし，ソフトウェア開発工程を詳細に見ると，テスト・デバッグ工程はいくつかのサブ工程で段階的に構成されている [6], [9]〜[11]．また，コーディング工程のコードレビューにおいてもテスト・デバッグ工程と同様に，ソフトウェアプロダクトから不具合が発見され修正されている [9]．このような開発プロセスにおいてはテスト・デバッグ工程ではなく，例えばコードレビュー以降の工程全体で発見される不具合について分析する必要がある．しかし，ソフトウェア信頼度成長モデルでは複数の工程にまたがった適用を想定していない．このことから，開発プロセスの工程全体を見渡す必要があるような場合にはソフトウェア信頼度成長モデルの適用が困難となる．そのため開発プロセス全体にわたって不具合の発見に関するデータを計測しても，それらを最終品質の向上に十分に活用できないことがある．

我々は組み込みソフトウェアの開発を手がけているある企業の協力の下，ソフトウェア開発プロセスの改善に関する研究を行なってきた [7], [8], [12], [13]．この企業でも，最終品質の保証あるいは向上のために，開発プロセス全体にわたって計測した不具合発見データをいかに利用するかが問題となっている．特に，この企業では，不具合発見履歴の減少傾向（増加傾向）と最終品質の間に次のような関係があることが経験的に知られている．すなわち，

「発見される不具合の数が開発工程の進捗と共に減少傾向にあれば最終品質は高く，発見される不具合の数が開発工程の進捗と共に増加傾向にあれ最終品質は高くない」

ということである．

本研究では，コードレビュー以降の不具合発見履歴に着目し，この企業が過去において行なったソフトウェア開発プロジェクトのデータを使用して，経験的に得られた知識を統計的分析によって検証する．そして，そのような傾向を特徴づける要因の特定を設計・コーディング工程において記録されたメトリクスデータの統計的分析によって行なう．

以降，2 節では本研究の目的，対象となるプロジェクト，分析に用いるメトリクスの定義について述べる．3 節では，不具合発見履歴を分類する基準を定め，実際のプロジェクトに適用し，その結果を示す．続く 4 節では 3 節での分類に従って，最終品質の比較を行ない，その結果が有意であることを示す．

そして 5 節では主に設計・コーディング工程のメトリクスについて，不具合履歴による分類との関連性を分析し，分類に影響を及ぼしている要因の特定を行なう．最後に 6 節でまとめを述べる．

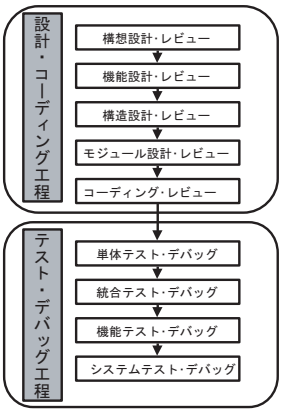


図 1　開発プロセス

2. 準　　備

2.1 研究の目的

本研究では，実際の企業から得られたデータに対する統計的分析により，次の 2 点を明らかにすることを目的とする．

（1） ある企業が経験的に得ている「発見される不具合の数が開発工程の進捗と共に減少傾向にあれば最終品質は高く，発見される不具合の数が開発工程の進捗と共に増加傾向にあれ最終品質は高くない」という知識の妥当性をテスト・デバッグ工程の不具合発見履歴に対する分析から導くこと

（2） テスト・デバッグ工程における不具合発見履歴（つまり，不具合検出頻度の増加・減少傾向）を特徴づける要因が設計・コーディング工程中に存在することを導くこと

2.2 対象プロジェクト

本研究で対象とするプロジェクトは，ある企業で実施されたプロジェクトのうち同種のシステムを開発する部門における，既に終了した 134 個のプロジェクトである．対象プロジェクトは次の基準で選ばれている．

（1） 1995 年から 1998 年の間に実施されている

（2） テスト・デバッグ工程全体で検出された不具合数がある一定値以上となっている

（3） コードレビューからシステムテストまでの 3 つ以上の工程で品質データの記録が残されている

対象プロジェクトの開発プロセスは，図 1 に示すような標準的なウォーターフォールモデルとなっている．設計・コーディング工程は，構想設計，機能設計，構造設計，モジュール設計，コーディングの 5 つの作業から構成されており，各作業の終了時にはレビュー作業が行なわれる．レビュー作業とは，設計ドキュメントやソースコードなどのプロダクトを構成した直後に，そのプロダクトに作り込まれた不具合を静的解析により発見し，修正

する作業である．レビュー作業を行なうことで，テスト・デバッグ工程におけるデバッグ作業を削減することができ，開発全体に費やす工数を削減できることが知られている．なお，コーディングでのレビュー作業のことを特にコードレビューと呼ぶ．

テスト・デバッグ工程はモジュールテスト，統合テスト，機能テスト，システムテストの 4 つの作業からなり，各作業ごとにテスト作業とデバッグ作業が行なわれている．テスト作業によって検出され，デバッグ作業によって除去された不具合についてはその作業内容の詳細を必ず記録するように指示されている．

2.3 メトリクスの定義

本研究で用いるメトリクスを以下のように定義する．

（1） 不具合検出頻度

不具合検出頻度を表すメトリクス *DF*（Defect Frequency）を次のように定義する．

$$DF = \frac{\text{不具合検出数 [個]}}{\text{レビュー工数またはテスト・デバッグ工数 [人日]}}$$

このメトリクスを用いて各工程の不具合検出頻度を次のように定義する．

- DF_{review}:構想設計レビューからモジュール設計レビューまでの作業に対する *DF*
- DF_{PGR}:コードレビューに対する *DF*
- DF_{MB}:モジュールテスト・デバッグに対する *DF*
- DF_{SB}:統合テスト・デバッグに対する *DF*
- DF_{FB}:機能テスト・デバッグに対する *DF*
- DF_{CB}:システムテスト・デバッグに対する *DF*

この *DF* の値が低くなる原因としては

- 作り込まれた不具合が少ない
- レビューの効率が悪い
- その工程の作業が困難である

などが考えられる．

（2） 開発規模

開発規模を次の式で定義する．

$$\text{開発規模 [}KLOC\text{]} = \frac{\text{プログラムの行数 [}LOC\text{]}}{1000}$$

通常は開発規模が大きくなるにつれ，プロジェクトは複雑になる．よって，十分に品質への配慮がなされていないとフィールド不具合密度や不具合検出頻度が高くなる．

（3） フィールド不具合密度

フィールド不具合密度を次のように定義する．

$$\text{フィールド不具合密度} = \frac{\text{フィールド不具合数 [個]}}{\text{開発規模 [}KLOC\text{]}}$$

ここでフィールド不具合数とはプロダクトを出荷後の 6ヶ月間に報告された不具合の数を表す．

（4） 生産性

生産性を表すメトリクスを次のように定義する．

$$\text{生産性} = \frac{\text{開発規模 [}KLOC\text{]}}{\text{コーディング工数 [人日]}}$$

ここでの開発規模とはコーディング終了時の開発規模を指すものと仮定する．このメトリクスは設計・コーディング段階におけるプロジェクトの困難さを示す一つの指標として用いる．

表 1　順位相関係数によるクラス分類

順位相関係数τ	クラス
$\tau = -1$	d
$-1 < \tau < 0$	d'
$0 \leqq \tau < 1$	i'
$\tau = 1$	i

生産性が低くなる原因としては次のことが考えられる．

- 人員が能力不足である
- その工程での作業が困難である

（5） レビュー工数比率

レビュー工数比率を次のように定義する．レビュー工数比率は設計・コーディング工程の全ての作業に対してレビューが占める割合を示す．

$$\text{レビュー工数比率} = \frac{\text{レビュー工数 [人日]}}{\text{設計工数 [人日]} + \text{レビュー工数 [人日]}}$$

ここで設計工数とは構想設計からコーディングまでの設計・コーディング工程全体の工数である．レビュー工数比率の値が低いと，設計段階において十分に不具合が除去されていない可能性が高くなる．

（6） コードレビュー工数比率

コードレビュー工数比率を次のように定義する．このメトリクスはコードレビューが占める割合を表す指標として用いられている．

$$\text{コードレビュー工数比率} = \frac{\text{コードレビュー工数 [人日]}}{\text{コーディング工数 [人日]} + \text{コードレビュー工数 [人日]}}$$

このメトリクスはコーディング工程のみに注目した時のレビュー工数比率であるといえる．よって，コードレビュー工数比率はレビュー工数比率と同様に，値が低いと不具合が除去されていない可能性が高くなる．

3. 不具合発見履歴によるプロジェクトの分類

我々は以前よりある企業の協力の下，ソフトウェア開発プロセスの改善に関する研究を行なってきた．この企業では不具合発見履歴の増加・減少傾向と最終品質の間に次のような関係があることが経験的に分かっていた．

「発見される不具合の数が開発工程の進捗と共に減少傾向にあれば最終品質は高く，発見される不具合の数が開発工程の進捗と共に増加傾向にあれ最終品質は高くない」

ここでは，まずコードレビュー以降，つまりコーディング作業終了時に実行するレビューからシステムテストまでの間の不具合発見の履歴を調べ，それに基づいて対象プロジェクトを分類する．

3.1 分類の方法

テスト工程における各作業での不具合発見履歴の増加・減少傾向を調べるためにどのような基準を用いるかは大きな問題である．傾向性の指標はいくつかあるが，単調増加・減少傾向の指標

表 2　不具合発見履歴による分類の結果

クラス	プロジェクト数（全体に占める割合）	フィールド不具合密度の順位平均
d	51(38.1%)	44.5
d'	64(47.8%)	46.6
i'	16(11.9%)	78.6
i	3(2.2%)	108.7

としては順位相関係数が最も一般的である [4], [19].

本研究では順位相関係数として Kendall の順位相関係数 τ の値を用いてプロジェクトを 4 つのクラス d, d', i', i 分類することにした．その基準を表 1 に示す．また，各クラスの例を図 2 に示す．

図 2（a）に示すように，$DF_{PGR} > DF_{MB} > DF_{SB} > DF_{FB} > DF_{CB}$ を満たすプロジェクト，つまり，テスト工程全体を通して，不具合検出頻度 DF の値が常に減少しているプロジェクトでは順位相関係数が $\tau = -1$ となり，d クラスに分類される．逆に，図 2（d）に示すように，$DF_{PGR} < DF_{MB} < DF_{SB} < DF_{FB} < DF_{CB}$ を満たしていて，テスト工程全体を通して，不具合検出頻度 DF の値が常に増加するプロジェクトでは順位相関係数が $\tau = 1$ となり，i クラスに分類される．

3.2 分 類 結 果

順位相関係数 τ を用いて 134 個の対象プロジェクトを分類した結果を表 2 に示す．同表より，約 40%が d クラス，約 50%が d' クラスであることが分かる．従って，最終品質も十分に高いことが期待される．さらに各プロジェクトのフィールド不具合密度の順位をクラス内で平均をとり，順位平均として示している．なお，この値は，フィールド不具合率を表しているのではなく，あくまでも順位であることに注意されたい．この順位平均でみる限り，d' クラスと i' クラスの間に大きな差が存在することが読みとれる．

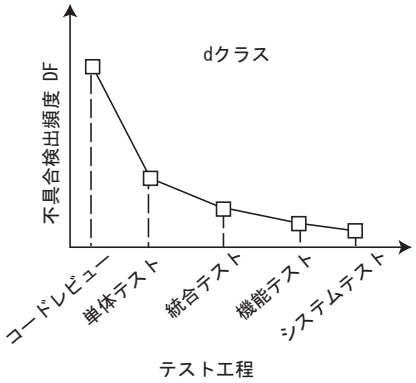
4. 不具合発見履歴と最終品質の関係

4.1 分析の目的

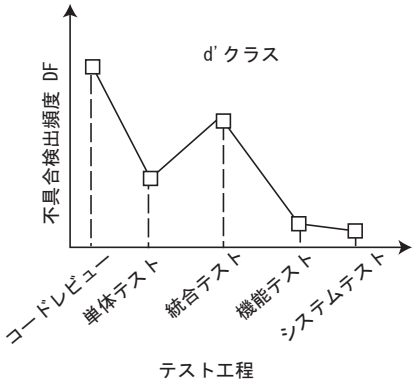
表 2 に示す各クラスに属するプロジェクトに関しては次のような解釈が成り立つと考えられる．

- d クラス:初めの工程で十分に不具合が除去され，終わりの工程ではあまり不具合が検出されなくなっている．そのため，残存する不具合の数が少なくなっている．
- d' クラス:全体の傾向としては d クラスと同じ減少傾向にあるため，残存する不具合の数は比較的少なくなっている．
- i クラス:初めの工程で十分に不具合が除去されず，終わりの工程で多くの不具合が検出されている．そのため残存する不具合の数が多くなっている．
- i' クラス:全体の傾向としては i クラスと同じ増加傾向にあるため，残存する不具合の数は比較的多くなっている．

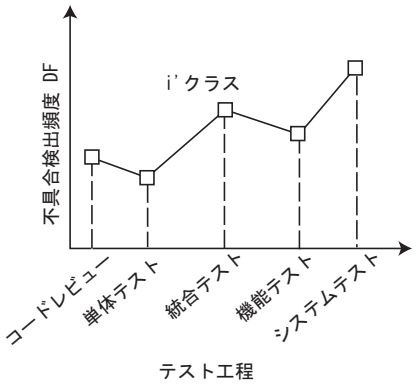
以上のような考察の結果として，d クラス，d' クラス，i' クラス，i クラスのフィールド不具合密度の位置母数(注1)をそれぞれ



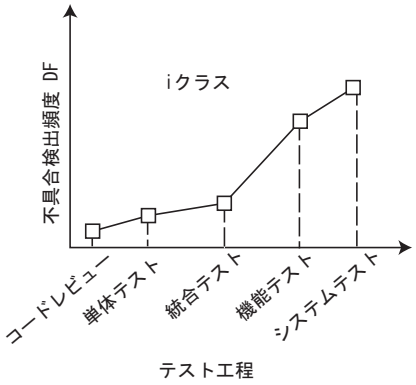
(a) d クラス



(b) d' クラス



(c) i' クラス



(d) i クラス

図 2　クラスの例

(注1)：位置母数とは平均値や中央値，最頻値のように度数分布の位置を示す母数である．本研究では位置母数として順位平均を用いる．

表 3　Ryan の対比較の結果

Step	名義的有意水準 α'	比較クラス	p 値
1	$\alpha' = 0.0166$	(d, i)	0.017
2	$\alpha' = 0.025$	(d, i')	0.016
		(d', i)	0.024
3	$\alpha' = 0.05$	(d, d')	0.43
		(d', i')	0.026
		(i', i)	0.089

$\theta_d, \theta_{d'}, \theta_{i'}, \theta_i$ とすると，表 2 より，次式が成り立つと予想できる．

$$\theta_d \leqq \theta_{d'} \leqq \theta_{i'} \leqq \theta_i$$

4.2 分析手順

不具合発見履歴と最終品質の関係の分析のために，次の帰無仮説 H_0 とそれに対する対立仮説 H_1 を検定する．有意水準は $\alpha = 0.05$ とする．

H_0:クラス間にフィールド不具合密度の差はない

$$\theta_d = \theta_{d'} = \theta_{i'} = \theta_i$$

H_1:クラス間でフィールド不具合密度に差があり，次式が成立する（ただし，全ての等号が同時に成り立つことはない）

$$\theta_d \leqq \theta_{d'} \leqq \theta_{i'} \leqq \theta_i$$

この仮説の検定には Jonckheere 検定を用いる．Jonckheere 検定は多群間の位置母数の差について，傾向性を持つ対立仮説に対して用いられる手法である [5].

Jonckheere 検定により有意な差があることが示されただけでは，各群間に有意な差が認められるかまでは明らかにならない．そこで，さらに詳しく分類を行なうために多重比較を行なう．多重比較とは，多群間の検定において全体として有意な差が認められた場合に，どの群間の差が有意であるかを分析する手法である [18].

ここでは群間に傾向性の順序があるので，Ryan の方法を用いる．Ryan の方法とは，群を順番に並べ，2 群間の対比較を段階的に繰り返す手法である [14]．Ryan の方法では，2 群の対比較の際に使用する有意水準の値は，全体の有意水準 α から導かれる名義的有意水準 α' で定められる．本研究では全体の有意水準を $\alpha = 0.1$ とする．

4.3 分析結果

Jonckheere 検定によって有意確率 $p = 0.02 (< 0.05)$ となり，有意水準 5% で帰無仮説 H_0 は棄却された．よって，フィールド不具合密度に関して，クラス間には全体として有意な差があることが統計的に示された．つまり，関係式

$$\theta_d \leqq \theta_{d'} \leqq \theta_{i'} \leqq \theta_i$$

が導かれたことになる．

次に，Ryan の対比較による結果を表 3 に示す．この表の p 値が名義的有意水準 α' を上回らない比較クラスについて有意さが確認できる．よって，表 3 より次のクラス間の比較である程度の有意差が確認された．

$$\theta_{d'} < \theta_{i'}$$

この結果を観察すると，4 つのクラス d, d', i', i の隣接するクラス間で有意な差が認められるのは，d' クラスと i' クラスの間のみである．よって，全体的には次式が成り立つといえる．

$$\theta_d \leqq \theta_{d'} < \theta_{i'} \leqq \theta_i$$

これは，d クラス，d' クラス，i' クラス，i クラスの順にフィールド不具合密度は大きくなり，この並びのうち，特に d' クラスと i' クラスの間に大きな差があることを示している．

3.2 の分類結果において，d' クラスは d クラスに準じ，i クラスは i' クラスに準じ，d' クラスと i' クラスの間に大きな差があるとの予想を示した．したがって，この予想と Ryan の対比較の結果が合致したことになる．以上より，d クラスと d' クラス，i クラスと i' クラスのそれぞれの間には有意な差がないと考えることができる．

そこで，以降の分析では，有意な差の見られなかった d クラスと d' クラスを D クラス，i' クラスと i クラスを I クラスに統合する．

表 4　ロジスティック回帰モデルによる分類

		モデル式 $p(z)$ による分類	
		D クラスの数	I クラスの数
順位相関係数 τ の値による分類	D クラスの数	113	2
	I クラスの数	15	4

5. 不具合発見履歴を特徴づける要因

5.1 分析の目的

これまでの結果より，不具合発見履歴と最終品質の間に関連があることが示された．しかし，品質の高いプロダクトを常に保証するためには，不具合発見履歴を特徴づける要因の分析が必要である．ここでは設計・コーディング工程に対して定義されるメトリクスの中から見つけることを考える．

5.2 分析方法

テスト・デバッグ工程における不具合発見履歴を特徴づける要因の候補として次の 5 つのメトリクスを考える．

- 不具合検出頻度（DF_{review}）
- 開発規模
- 生産性
- レビュー工数比率
- コードレビュー工数比率

まず，これら 5 つのメトリクスについてロジスティック回帰分析を行ない，モデル式の要素としての有意性を検証する．次に，有意であると認められたメトリクスを用いてロジスティック回帰モデルを作成する．

最後にモデル式による分類と順位相関係数 τ による分類の結果から，フィッシャーの正確確率検定 [16] によって検証を行なう．正確には，帰無仮説「順位相関係数 τ によって D クラスと I ク

ラスとに分類されることと，ロジスティック回帰モデルによって D クラスと I クラスに分類されることとは独立である」が棄却できるかどうかの検定を行なう．なお，有意水準は $\alpha = 0.05$ と定める．

5.3 分析結果

候補となる要因である 5 つのメトリクスにステップワイズ法を適用し，ロジスティック回帰分析を行なった．その結果，次の 2 つのメトリクスについてだけ有意性が認められた．

- コードレビュー工数比率
- 不具合検出頻度（DF_{review}）

これらの要素を用いて作成されたロジスティック回帰モデルは次式の通りである．

$$p(z) = \frac{\exp(-19.142z_1 - 0.149z_2 + 0.852)}{1 + \exp(-19.142z_1 - 0.149z_2 + 0.852)}$$

ここで，$p(z)$ はプロジェクトが I クラスに分類される確率を表す．また，z_1, z_2 はそれぞれ，コードレビュー工数比率，不具合検出頻度（DF_{review}）を表す．

この回帰モデルを用いてプロジェクトの分類を行なった結果を表 4 に示す．この表を用いてフィッシャーの正確確率検定を行なった結果，有意水準 $\alpha = 0.004(< 0.05)$ で帰無仮説は棄却された．よって，コードレビュー工数比率と不具合検出頻度（DF_{review}）は D クラス，I クラスを特徴づける要因と見なすことができる．

6. まとめ

本研究では，コードレビュー以降の不具合発見履歴について，企業が経験的に得ていた最終品質についての知識が妥当であることを統計的分析により明らかにした．また，コードレビュー工数比率と不具合検出頻度（DF_{review}）が不具合発見履歴を特徴づける要因として妥当であることを統計的に示した．従って，進行中の開発プロジェクトのコードレビュー工数比率と設計・コーディング工程での不具合検出頻度（DF_{review}）の値をモニターしておけば，テスト・デバッグ工程の様子と最終品質の良さがある程度予測できる可能性を示していることになる．

文献

[1] V.Basili and J.Musa, “The future engineering of software: A management perspective”, IEEE Computer,vol.24, no.9,pp.90–96, 1991

[2] D.N.Card and R.L.Glass, “Measuring Software Design Quality”, PTR Prentice-Hall,Inc., 1990

[3] R.G.Ebenau and S.H.Strauss, “Softwawre Inspection Process”, McGraw-Hill,Inc., 1993

[4] M.Kendall and J.D.Gibbons, “Rank Correlation Methods”, Edward Arnold,5th Edition, 1990

[5] E.L.Lehmann, “Nonparametrics: Statistical Methods Based on Ranks”, Holden-Day,Inc., 1975

[6] B.Marick, “The Craft of Software Testing”, PTR Prentice-Hall, 1995

[7] O.Mizuno, T.Kikuno, K.Inagaki, Y.Takagi, and K.Sakamoto, “Statistical analysis of deviation of actual cost from estimated cost using actual project data”, Information and Software Technology,vol.42,pp.465–473, 2000

[8] O.Mizuno, T.Kikuno, Y.Takagi, and K.Sakamoto, “Characterization of risky projects based on project managers’ evaluation”, Proc. of 22nd International Conference on Software Engineering(ICSE2000),pp.387–395, 2000

[9] G.J.Myers 著，長尾 真 訳，“The Art of Software Testing （ソフトウェア・テストの技法）”, 第 1 版，株式会社近代科学社，2000

[10] M.L.Shooman 著，箱崎 勝也 訳，“Software Engineering — Design, Reliability, and Management（ソフトウェアの性能とテスト）”，第 1 版，マグロウヒル株式会社，1989

[11] I.Sommerville, “Software Engineering”, Addison-Wesley,4th edition, 1992

[12] Y.Takagi, T.Tanaka, N.Niihara, K.Sakamoto, S.Kusumoto, and T.Kikuno, “Analysis of reviews’ effectiveness based on software metrics”, Proc. of 6th International Symposium on Software Reliability Engineering,pp.34-39, 1995

[13] T.Tanaka, K.Sakamoto, S.Kusumoto, K.Matsumoto, and T.Kikuno, “Improvement of software process by process description and benefit estimation”, Proc. of 17th International Conference on Software Engineering(ICSE95),pp.123–132, 1995

[14] 岩原信九郎，“新しい教育・心理統計 ノンパラメトリック法”，株第 2 版，式会社日本文化科学社，1968

[15] 大場充，“ソフトウェア品質保証のためのソフトウェア・プロジェクトの実績データ収集・分析技法”，第 1 版，株式会社ソフトウェア・リサーチ・センター，1993

[16] 芝祐順，渡部洋，石塚智一，“統計用語辞典”，新曜社，1984

[17] 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

[18] 長田靖，吉田道弘，“統計的多重比較法の基礎”，第 1 版，株式会社サイエンス社，1933

[19] 武藤眞介，“統計解析ハンドブック”，第 1 版，株式会社朝倉書店，1995

[20] 山田茂，高橋宗雄，“ソフトウェアマネジメントモデル入門”，第 3 版，共立出版株式会社，1999