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.

フィールド不具合数を許容値以下に抑えるための ソフトウェアテスト工数の推定モデルの提案

重松英二郎† 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
〒 560-8531 大阪府豊中市待兼山町 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 プロセス革新部
〒 525-0035 滋賀県草津市西草津 2-2-1

E-mail: †{sigematu,o-mizuno,kikuno}@ics.es.osaka-u.ac.jp, ††taka@biwa.kusatsu.omron.co.jp

あらまし 本研究では，ソフトウェアの最終的な品質をある一定の許容値以内に収めるために必要となるテスト工数の基準値を定める手法を提案する．これまでに，ある企業の開発プロジェクトを対象として，設計工数とレビュー工数の 2 つの工数の配分比が最終品質に与える影響の分析を行ってきた．しかし，設計工数とテスト工数の配分が最終品質に与える影響については十分に議論してきていない．そこで本報告では最終品質をある許容値以内に収めるために必要なテスト工数を求めることを目指す．提案法では，まず各工程に対する工数配分とフィールド不具合数の関係をモデル化した重回帰式を作成し，テスト工数を得るための式を導出する．そして，その式に対してフィールド不具合数の許容値，設計工数，レビュー工数を定めることで，品質を保証するために最低限必要なテスト工数の基準値を算出する．実際のプロジェクトデータを用いた適用実験の結果，提案法で求めた基準値以上のテスト工数があれば最終品質を保証できることを確認した．

キーワード ソフトウェア開発，プロジェクト計画作成，ソフトウェアテスト，品質保証

Estimation of Software Testing Effort to Assure Permissible Number of Field Defects

Eijiro SHIGEMATSU†, Osamu MIZUNO†, Tohru KIKUNO†, and Yasunari TAKAGI††

† Department of Information and Mathematical Science,
Graduate School of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka-shi, Osaka 560-8531, Japan
†† Social Systems Company, OMRON corporation
Nishi-kusatsu 2-2-1, Kusatsu-shi, Shiga 525-0035, Japan

E-mail: †{sigematu,o-mizuno,kikuno}@ics.es.osaka-u.ac.jp, ††taka@biwa.kusatsu.omron.co.jp

Abstract This paper proposes a method to determine the effort for testing phase in software development. In a certain company, we have performed several analyses of the assignment of efforts to design and review activities. However, the assignment of efforts to design and test activities was not discussed sufficiently. We therefore try to develop a method to determine the test effort needed to assure feasible quality. In the proposed method, we construct a multiple regression equation, which represents the relationship between the effort of each development phase and the quality of software product. We calculate the test effort by using the permissible number of field defects, the design effort, and the review effort. As a result of experimental evaluation using actual project data, we could assure the quality of software by spending more test effort than calculated value.

Key words software development, project planning, software test, quality assurance

1. まえがき

ソフトウェア開発においては，納期内に，低コストで，品質の高いプロダクトを開発することが求められている．近年，ソフトウェアの社会的な重要性がますます大きくなり，ソフトウェアの品質保証は特に重要な目標となっている [1].

組み込みソフトウェアを開発するある企業では，ソフトウェア開発プロジェクトの各工程（設計，レビュー，コーディング，テスト，デバッグ）に対する工数配分計画の作成について，ソフトウェア品質を考慮して取り組んでいる．我々は，その企業のソフトウェア開発プロジェクトを対象として，レビュー作業に費やす工数が設計工程全体に費やす工数に占める比率（レビュー比率）に着目することで，ソフトウェア品質を考慮した工数配分について研究してきた．具体的には，レビュー比率とプロダクトの最終品質の関係を統計的手法を用いて分析し，レビュー比率が高いプロジェクトではフィールド不具合数は比較的少ないということを示してきた [2], [3]．こうした分析に基づいて，その企業では，プロジェクト計画作成時点でレビュー比率が 15%以上となるように定めている．しかし，テスト作業にどれだけの工数を費やせば品質が保証されるかについては，これまで十分に議論されてきていない．

ソフトウェアの品質保証は重要な問題であり，出荷するプロダクトに不具合が発生することを避けるためにはテスト作業を十分に行う必要がある．しかし，一般にソフトウェア開発プロジェクトでは，プロダクトの納期が定められており，テスト工程に費やす工数を必要最小限に抑える必要がある．そのため，テスト工程に費やす工数の配分は大きな問題である．本研究では，設計工数とレビュー工数に対するテスト工数の配分を定めることを目的とする．そこで，フィールド不具合数をある許容値以内に収めるために必要なテスト工数の基準値を定める手法を提案する．

提案法では，まずテスト工数をモデル式で表現することを考える．そのために，各工程に対する工数配分とフィールド不具合数の関係をモデル化する．ここでは，フィールド不具合数を目的変数とし，設計工数，レビュー工数とテスト工数を説明変数とした重回帰式を作成する．そして，その重回帰式からテスト工数を得るための式を導出する．次に，作成した重回帰式とテスト工数を得るための式を用いて，テスト工数の基準値を設定することを考える．そのためにまず，過去のデータから重回帰式のパラメータを決定する．そして，フィールド不具合数の許容値，設計工数，レビュー工数を定めて，それらをテスト工数を得るための式に代入することで，テスト工数の基準値を得る．対象とする企業で実際に収集されたプロジェクトデータを用いた適用実験の結果，提案法で求めた値以上のテスト工数であれば最終品質が保証できることを確認した．

以降，2 章では，対象プロジェクトとそのプロジェクトにおける工数配分とソフトウェア品質の関係について説明する．3 章では，フィールド不具合数をある許容値以内に収めるために必要なテスト工数の基準値を設定する手法について説明する．4 章では，提案法の適用実験とその結果について述べる．最後に，5 章でまとめと今後の課題について述べる．

2. 準　　備

2.1 対象プロジェクト

本研究で対象とするプロジェクトは，ある企業の受発注システムを開発する部門における，既に終了した 39 個のプロジェクトである．対象プロジェクトの特徴を以下の (1)〜(3) にまとめる．

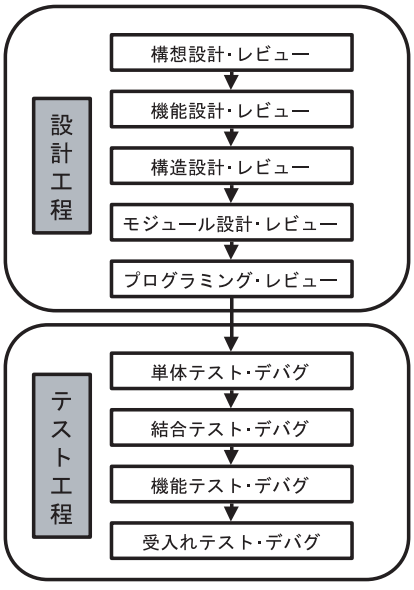
（1） 1996 年から 1997 年に開発を開始.

（2） 開発工数は 7〜35 人月（平均 14 人月）.

（3） 設計工数に対するレビュー工数の比率は 15%以上.

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．設計工程は，構想設計，機能設計，構造設計，

図 1　開発プロセス



モジュール設計，プログラミングの 5 つの作業から

構成されており，テスト工程は，単体テスト，結合テスト，機能テスト，受入れテストの 4 つのテスト作業とデバッグ作業から構成されている．

なお，設計工程の各作業の終了時にはレビュー作業が行われる．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作成した直後に，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．レビュー作業を行うことで，テスト工程におけるデバ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 [4]．この企業では，ピアレビュー [5] という 2〜3 人で行うレビューが実施されており，設計工程全体に費やす工数の少なくとも 15%の工数をレビューに割り当てるということなどを指示したガイドラインが作成されている [3].

テスト工程においては，テスト作業とデバッグ作業が行われる．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は必ず記録するように指示されている [6]．この企業では，ソフトウェア信頼度成長曲線 [7] などに基づいて，プロダクトに作り込まれた不具合がほぼ全て取り除かれたかどうかを経験的に判断することでテスト工程を終了するための基準としている．

2.2 メトリクスの定義

この節では，分析に用いるメトリクスを定義する．

まずはじめに，工数に関するメトリクス E_{design}, E_{review}, E_{test} を次のように定義する．

(1) E_{design}（設計工数）: 設計工程の設計作業に費やす工数（人日）.

(2) E_{review}（レビュー工数）: 設計工程のレビュー作業に費やす工数（人日）.

(3) E_{test}（テスト工数）: テスト工程に費やす工数（人日）.

設計工程は，図 1 に示すように数段階の設計作業とレビュー作業から構成される．設計作業には，設計ドキュメントを作成する段階とソースコードを作成する段階があるが，プログラムの設計作業とコーディング作業の区別はあいまいである [8]．そこで，数段階の設計作業に費やす工数の和を設計工数と定義し，数段階のレビュー作業に費やす工数の和をレビュー工数と定義する．

一般にテスト工程はテスト作業とデバッグ作業から構成される．本研究ではソフトウェア開発プロジェクトにおけるテスト工程の占める割合を問題としているので，テスト作業とデバッグ作業に費やす工数の和をテスト工数と定義する．

また，不具合数に関するメトリクス D_{field} を次のように定義する．

(4) D_{field}（フィールド不具合数）: プロダクトの出荷後 6ヶ月間に発見される不具合数（個）．プロダクトの品質を表す指標として用いる．

2.3 工数配分とソフトウェア品質

2.3.1 これまでの研究

対象としている企業では，品質を考慮した工数配分計画の作成に取り組んでいる．我々は，その企業のソフトウェア開発プロジェクトにおいて，設計作業とレビュー作業の工数配分に着目した分析を行ってきた．その分析においては，レビュー比率というメトリクスを導入して，レビュー比率とフィールド不具合数の関係を統計的手法を用いて分析してきた．レビュー比率とは，以下のように定義されるメトリクスであり，レビューの量を表す指標として用いられている．

$$\text{レビュー比率} = \frac{E_{review}}{E_{design} + E_{review}}$$

その分析の結果，レビュー比率が高いプロジェクトではフィールド不具合数は比較的少ないということを示した [2], [3]．こうした分析に基づいて，その企業では，プロジェクト計画作成時点でレビュー比率が 15%以上となるように定めている．

2.3.2 テスト工数の配分

プロダクトに作り込まれた不具合を発見・修正・除去するために実施されるテストは，ソフトウェアの品質・信頼性を高めて最終的な品質評価を行うために，開発プロセスにおける重要な最終工程となっている [7].

テスト工程を終了してプロダクトを出荷するのに最適な時期を見積もる方法として，テスト工程におけるソフトウェアの不具合の発見過程や発生現象をモデル化したソフトウェア信頼度成長モデルがある [7]. このモデルにより，ソフトウェアの信頼性の達成度合いや推移状況を確認したり，出荷品質を判定したり，さらにテストを終了して実際の運用段階へ移行するのに最適な時期，いわゆる最適リリース時期を見積もったりすることができる．しかし，プロダクトを出荷するのに最適な時期を見積もるためには，ソフトウェア信頼性の計測と評価を実施する必要があり，それらの情報はテスト作業をある程度進めないと得ることができない．

本研究で対象とした企業では，ソフトウェア信頼

度成長曲線などに基づいて，プロダクトに作り込まれた不具合が全て取り除かれたかどうかを経験的に判断することでテスト工程を終了している．しかし，テスト工程にどれだけの工数を費やせば品質が保証されるかについては十分に議論されてこなかった．そこで，本研究では設計工程に費やす工数に対するテスト工程に費やす工数の配分比を定めることを目的として，テスト工程にどれだけの工数を費やせば品質が保証されるかを問題とする．

3. テスト工数の基準値の設定手法

3.1 テスト工数の表現

設計工程に費やす工数に対するテスト工程に費やす工数の配分比を定めるために，まず各工程に対する工数配分とフィールド不具合数の関係をモデル化することを考える．ここでは，フィールド不具合数 D_{field} を目的変数とし，設計工数 E_{design}，レビュー工数 E_{review} とテスト工数 E_{test} を説明変数とした以下のような重回帰式を作成する．

$$D_{field} = \beta_0 E_{design} + \beta_1 E_{review} + \beta_2 E_{test} \quad (1)$$

このモデルは，設計作業で作り込まれた不具合がレビュー作業とテスト・デバッグ作業によって取り除かれる関係を，工数配分に着目して表している．

このモデルによってフィールド不具合数が設計工数とレビュー工数とテスト工数の配分によって決定される関係が表されていると考える．そこでテスト工数がその他の変数の関数となるように (1) 式を変形すると，以下の式が得られる．

$$E_{test} = (D_{field} - \beta_0 E_{design} - \beta_1 E_{review})/\beta_2 \quad (2)$$

(2) 式により，テスト工数 E_{test} をフィールド不具合数 D_{field}，設計工数 E_{design} とレビュー工数 E_{review} の関数として表すことができた．

3.2 設定手順の概要

提案法では，(2) 式を利用してテスト工数の基準値を設定する．テスト工数の基準値を設定する手順は，次の 4 ステップである．

Step 1 いくつかの過去のプロジェクトのデータから (1) 式の回帰係数の値を決定する．

$$\beta_0 = b_0, \quad \beta_1 = b_1, \quad \beta_2 = b_2$$

Step 2 適用対象のプロジェクトの特性からフィールド不具合数の許容値 γ_{field} を定める．すると，$D_{field} = \gamma_{field}$ となる．

Step 3 適用対象のプロジェクトの実績値から設計工数 C_{design} とレビュー工数 C_{review} を定める．

$$E_{design} = C_{design}, \quad E_{review} = C_{review}$$

Step 4 Step 1〜3 で定めた値を (2) 式に代入して得られた E_{test} の値を適用対象のプロジェクトのためのテスト工数の基準値 θ_{test} とする．

次の節以降で，各ステップの詳細を説明する．

3.3 パラメータ決定 (Step 1)

提案法では，テスト工数の基準値を過去のデータに基づいて定める．そこで，いくつかのプロジェクトのデータから (1) 式の回帰係数の値を最小 2 乗法により決定する．

$$\beta_0 = b_0, \quad \beta_1 = b_1, \quad \beta_2 = b_2$$

これらの係数の値を (1) 式に代入すると以下の式が得られる．

$$D_{field} = b_0 E_{design} + b_1 E_{review} + b_2 E_{test} \quad (3)$$

対象とするプロジェクトの各工程に対する工数配分とフィールド不具合数の関係を，(3) 式がうまくモデル化している度合を評価するために，ここでは重相関係数 R^2 を用いる．重相関係数 R^2 とは，重回帰モデルにおける説明変数が目的変数を説明している割合を表す 0 から 1 までの値であり，値が 1 に近いほど説明変数が目的変数をよく説明していることを表す [10].

3.4 フィールド不具合数許容値設定 (Step 2)

(3) 式の説明変数 (E_{test}, E_{design}, E_{review}) に工数データを代入することによって得られる D_{field} の値は，過去のデータに基づいて得られる推定値である．そのため，(3) 式の説明変数に工数データを代入した値と実測値にはいくらかの誤差が生じる．また，(3) 式によって得られる D_{field} の値は離散的ではなく連続的な値であることから，(3) 式によってフィールド不具合数を正確に推定することは難しい．そこで，フィールド不具合数の許容値を γ_{field} 未満と定め，$D_{field} = \gamma_{field}$ となるようなテスト工数を求めることで，その値をテスト工数の基準値と設定する．

3.5 設計工数とレビュー工数の設定 (Step 3)

このステップにおいて，適用対象のプロジェクト (つまり，テスト工数の基準値を設定しようとするプロジェクト) の設計工数とレビュー工数を定める．

$$E_{design} = C_{design}, \quad E_{review} = C_{review}$$

C_{design} と C_{review} は，プロジェクトの進捗状況によって次のように定める．設計工程が終了した後でテスト

工数の基準値を求める場合には，設計工数とレビュー工数の実測値をそれぞれ C_{design}, C_{review} とする．設計工程の途中あるいは，設計工程の前の工数計画段階においてテスト工数の基準値を求める場合には，設計工数とレビュー工数の見積り値をそれぞれ C_{design}, C_{review} とする．

3.6 テスト工数の基準値設定 (Step 4)

Step 1〜3 で定めた値を (2) 式に代入する．

$$\begin{cases} \beta_0 = b_0 \\ \beta_1 = b_1 \\ \beta_2 = b_2 \\ D_{field} = \gamma_{field} \\ E_{design} = C_{design} \\ E_{review} = C_{review} \end{cases}$$

代入した結果得られた E_{test} の値を，テスト工数の基準値 θ_{test} とする．

$$\theta_{test} = (\gamma_{field} - b_0 C_{design} - b_1 C_{review})/b_2 \quad (4)$$

プロジェクトのテスト工数が θ_{test} と等しければ $D_{field} = \gamma_{field}$ となるように (2) 式を作成している．したがって，プロジェクトのテスト工数が θ_{test} より多ければ，フィールド不具合数が γ_{field} 未満であることを保証できる．

4. 適 用 実 験

提案法の適用可能性を示すために，1996 年のプロジェクトデータを利用して，1997 年のプロジェクトにおけるテスト工数の基準値を設定するという実験を行う．ここでは，フィールド不具合を発生させない (すなわち，$\gamma_{field} = 1$ とする) ために必要なテスト工数の基準値を設定することを考える．

4.1 実 験 手 順

4.1.1 パラメータ決定 (Step 1)

まず，1996 年の 19 個のプロジェクトデータを用いて (1) 式の回帰係数を決定する．1996 年の各プロジェクトの工数データ (E_{test}, E_{design}, E_{review}) を表 1 に示す．

各プロジェクトの工数データとフィールド不具合数のデータから，回帰係数を以下のように決定した．

$$b_0 = 0.039, \quad b_1 = -0.069, \quad b_2 = -0.011$$

これらの係数の値を (3) 式に代入すると以下の式が得られる．

$$D_{field} = 0.039 E_{design} - 0.069 E_{review} - 0.011 E_{test} \quad (5)$$

(5) 式について，モデルの適合度を表す重相関係数 R^2 は 0.743 であった．すなわち，(5) 式の E_{design}, E_{review}, E_{test} によって D_{field} の変化の 74.3%を説明できている(注1)．

表 1　1996 年のプロジェクトの工数データ

No.	E_{design}	E_{review}	E_{test}
Y96-1	81.0	22.9	73.1
Y96-2	81.0	15.1	87.2
Y96-3	122.4	28.6	190.0
Y96-4	244.9	54.2	93.0
Y96-5	78.2	13.7	133.3
Y96-6	64.5	16.8	37.2
Y96-7	76.5	22.7	242.0
Y96-8	128.9	33.1	107.0
Y96-9	56.1	16.3	49.0
Y96-10	92.5	17.7	104.0
Y96-11	107.0	21.0	113.7
Y96-12	183.4	72.3	218.5
Y96-13	148.0	35.6	205.0
Y96-14	166.9	35.3	216.0
Y96-15	62.2	11.1	144.6
Y96-16	64.9	15.0	99.7
Y96-17	220.5	44.0	203.5
Y96-18	283.6	68.1	196.0
Y96-19	428.0	86.0	130.0

4.1.2 フィールド不具合数の許容値の設定 (Step 2)

次に，フィールド不具合数の許容値を定める．この実験では，フィールド不具合を発生させないために必要なテスト工数の基準値を設定することを考える．そのため，フィールド不具合数の許容範囲を 1 未満と定める．即ち，$\gamma_{field} = 1$ となるようなテスト工数を求めることで，その値をテスト工数の基準値と設定することを考える．

4.1.3 設計工数とレビュー工数の設定 (Step 3)

そして，テスト工数の基準値を定めようとするプロジェクトの設計工数とレビュー工数を定める．この実験では，1997 年の 20 個のプロジェクトに対して，提案法により得られる値がテスト工数の基準値として適用できるかどうかを調べる．そこで，各プロジェクトの設計工数とレビュー工数の実測値を利用することを考える．

1997 年の各プロジェクトの設計工数とレビュー工

(注1)：重相関係数 R^2 の値の評価は経験的に定められるが，文献 [9] において，フィールド不具合数を目的変数とし，定められた開発基準が満たされている度合いを説明変数としたフィールド不具合数推定モデルの重相関係数 R^2 は 0.28〜0.34 であった．その研究において作成されたフィールド不具合数推定モデルは，開発基準が満たされている度合いがフィールド不具合数に影響を与えることを示すことを目的としており，モデルの適合度についてあまり議論していないが，本研究のモデルの適合度は比較的高いことがわかる．

数の実測値を表 2 に示す.

表 2　1997 年のプロジェクトの設計工数とレビュー工数

No.	E_{design}	E_{review}
Y97-1	138.5	32.5
Y97-2	81.2	22.5
Y97-3	79.7	18.9
Y97-4	77.2	15.5
Y97-5	57.5	13.3
Y97-6	45.0	10.3
Y97-7	68.0	17.0
Y97-8	60.0	27.0
Y97-9	75.2	21.6
Y97-10	91.3	24.4
Y97-11	68.0	14.5
Y97-12	134.5	29.9
Y97-13	64.7	15.3
Y97-14	127.0	27.0
Y97-15	160.2	36.0
Y97-16	82.0	15.0
Y97-17	91.4	17.3
Y97-18	102.0	20.0
Y97-19	240.0	51.0
Y97-20	150.0	28.0

4.1.4　テスト工数の基準値の設定 (Step 4)
Step 1〜3 で定めた値を (5) 式に代入する.

$$\begin{cases} b_0 = 0.039 \\ b_1 = -0.069 \\ b_2 = -0.011 \\ \gamma_{field} = 1 \\ E_{design} = \text{各プロジェクトの実測値 } (C_{design}) \\ E_{review} = \text{各プロジェクトの実測値 } (C_{review}) \end{cases}$$

各プロジェクトについて，代入して得られた E_{test} の値をテスト工数の基準値 θ_{test} とする.

$$\theta_{test} = (1 - 0.039C_{design} + 0.069C_{review})/(-0.011)$$

4.2 実験結果

1997 年の 20 個の各プロジェクトについて，テスト工数の実測値と基準値 θ_{test} の比較結果とソフトウェア品質の対応を表 3 に示す.

この表において，E_{test} はテスト工数の実測値であり，θ_{test} はテスト工数の基準値である．テスト工数の実測値と基準値を比較して，実測値の方が大きいことを“ ○ ”で表し，基準値の方が大きいことを“ × ”で表す．ソフトウェア品質は，フィールド不具合が発生しなかったことを“ ○ ”と評価し，フィールド不具合が発生したことを“ × ”と評価する.

テスト工数の実測値が基準値より多いか，それとも基準値以下かということとフィールド不具合の有無について分類すると表 4 のようになった.

表 3　テスト工数の実測値と基準値の比較とソフトウェア品質

No.	E_{test}	θ_{test}	比較	品質
Y97-1	77.9	196.3	×	○
Y97-2	128.3	55.8	○	○
Y97-3	49.9	73.1	×	○
Y97-4	77.0	85.6	×	○
Y97-5	36.7	29.5	○	○
Y97-6	180.0	4.0	○	○
Y97-7	106.0	43.5	○	○
Y97-8	55.0	47.5	○	○
Y97-9	47.8	40.2	○	○
Y97-10	107.4	79.7	○	○
Y97-11	66.4	59.2	○	○
Y97-12	78.5	198.4	×	×
Y97-13	88.3	42.5	○	×
Y97-14	78.0	190.0	×	×
Y97-15	121.2	251.3	×	×
Y97-16	90.0	105.7	×	×
Y97-17	56.8	124.6	×	×
Y97-18	25.0	145.3	×	×
Y97-19	58.0	440.1	×	×
Y97-20	43.0	265.3	×	×

表 4　実験結果の分類

		フィールド不具合数		
		なし	あり	合計
テスト工数	基準値より多い	8	1	9
	基準値以下	3	8	11
	合計	11	9	20

分類の結果,

- テスト工数の実測値が基準値より多いプロジェクトは 9 個あったが，そのうち 8 個のプロジェクトではフィールド不具合は発生しなかった.
- テスト工数の実測値が基準値以下のプロジェクトは 11 個あったが，そのうち 8 個のプロジェクトではフィールド不具合が発生した.

フィッシャーの正確確率検定 [11] により,“ テスト工数の実測値が基準値より多いか基準値以下かということとフィールド不具合の有無が独立である ”という帰無仮説を棄却できるかどうかを検定した．統計的仮説検定の結果，帰無仮説は有意水準 $\alpha = 0.01$ で棄却された.

この結果から,「テスト工数の実測値が基準値より多いプロジェクトでは，フィールド不具合が発生することは少ないが，テスト工数の実測値が基準値以下のプロジェクトでは，フィールド不具合が発生することが多い」ということを統計的に示すことがで

きた．つまり，提案法により求めた値が，フィールド不具合を発生させないために最低限必要なテスト工数の基準値になっているといえる．

5. ま と め

本研究では，フィールド不具合数を許容値以内に収めるために必要なテスト工数の基準値を定める手法を提案した．提案法では，まずテスト工数をフィールド不具合数の観点からモデル化することを考えた．そのために，各工程に対する工数配分とフィールド不具合数の関係を表した重回帰式を作成して，その重回帰式からテスト工数を得るための式を導出した．次に，作成した重回帰式とテスト工数を得るための式を用いて，テスト工数の基準値を設定することを考えた．そのためにまず，過去のデータから重回帰式のパラメータを決定した．そして，フィールド不具合数許容値，設計工数，レビュー工数を代入することで，テスト工数の基準値を得るという手法を提案した．企業で実際に収集されたプロジェクトデータを用いた適用実験の結果，テスト工数の実測値が基準値より多いプロジェクトではフィールド不具合は発生しないが，テスト工数の実測値が基準値以下のプロジェクトでは，フィールド不具合が発生してしまうということが統計的に説明された．よって，提案法で求めた値が，フィールド不具合を発生させないために最低限必要なテスト工数の基準値になることが説明できた．

しかし，今回の結果は対象としている企業の一部のプロジェクト群についてのみ適用可能性を示したに過ぎない．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，必ずしも提案法により求めた値がテスト工数の基準値となるとはいえない．そのため，今後さらに多くのプロジェクトに対して適用実験を行い，提案法の有効性の検証を行わなければならない．

今後の課題としては，いくつかの段階に分けられているテスト工程のうち，どの段階のテスト作業が最もソフトウェア品質に影響を与えているのかということについて統計的手法を用いて分析することが考えられる．

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] 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.

[3] 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.

[4] M. E. Fagan, “Advances in software inspections,” IEEE Trans. Software Engineering, vol.12, no.7, pp.744–751, 1986.

[5] D. B. Bisant and J. R. Lyle, “A Two-Person Inspection Metnod to Improve Programing Productivity,” IEEE Trans. Software Engineering. vol.15, no.10, pp.1294–1304, 1989.

[6] 坂本啓司, 田中敏文, 楠本真二, 松本健一, 菊野亨, “利益予測に基づくソフトウェアプロセス改善の試み,” 電子情報通信学会論文誌 D-I, vol.J83-D-1, no.7, pp.740–748, 2000.

[7] 山田茂, 高橋宗雄, “ソフトウェアマネジメントモデル入門,” 共立出版, 1993.

[8] W. S. Humphrey, “A Discipline for Software Engineering,” Addison Wesley, 1995.

[9] M. S. Krishnan and M. I. Kellner, “Measuring process consistency: Implications for reducing software defects,” IEEE Trans. Software Engineering. vol.25, no.6, pp.800–815, 1999.

[10] 古谷野亘, “多変量解析ガイド,” 川島書店, 1988.

[11] 芝祐順, 渡部洋, 石塚智一, “統計用語辞典,” 新曜社, 1984.