プログラム品質を考慮した試験工程進捗モデル

金井 敦[†] 古山 恒夫[††]

本論文では，大規模なソフトウェア開発における試験工程の進捗を管理するためのモデルを提案する．本モデルは，時間に対する試験項目の消化度合いを試験進捗と考え，その進捗度合いが試験項目あたりのバグ発見数（プログラム品質）に強く影響を受けることに着目したモデルである．ソフトウェア開発の実測データを用いて，プログラム品質を考慮しない場合よりも21%程度正確に試験進捗を計算できることを示し本モデルの妥当性を検証する．さらに，本モデルの応用として，検証用に用いたものとは異なる実測データを用いて試験期間予測法について検討する．

The Test Progress Model Based on Program Quality

Atsushi Kanai[†] and Tsuneo Furuyama[††]

This paper proposes the model to evaluate directly test phase progress. This model is based on the point that test phase progress is strongly affected by detected bugs number per a test item (program quality). The evaluation of this model using actual software development project data shows the test phase progress can be evaluated approximately 21% more accurately than the case without considering program quality. Farthermore, as an example of application of this model, this paper shows based on the other actual project data that test phase period can be predicted.

1. まえがき

試験工程はソフトウェアの開発工程の中でも開発資源を多く使用する工程であり，この工程管理を適切に行うことは非常に重要である．試験工程の計画および進捗管理は，試験項目の数やバグ発見率などから経験的に進捗の管理を行っていることが多い．一般的な試験工程の手順は，1) 試験項目の設計，2) 設計した試験項目の実施計画策定，3) 試験実施およびデバグという順で行われる．試験実施中は消化試験項目数やバグの発見状況を測定しながら，進捗や品質が管理される．

試験実施中の管理については，従来からソフトウェア信頼度成長モデルを用いたソフトウェア品質の管理手法が多く提案されている．ソフトウェア信頼度成長モデルは試験進捗にともなう残存バグの総数とその発見確率の変化をベースに構築されており，これまでの経験から時間に対する累積発見バグ数（信頼度成長曲線）の多くはS字形や指数形の曲線になることが知られている[1)~4)]．

一方，試験工程の進捗を直接的に把握するためには，ソフトウェアの品質よりも試験項目の消化率を指針にすべきであるが，直接的に試験項目の消化過程を扱うモデルとしては死滅過程を適用したモデル[5)]が提案されているにとどまる．

本論文では，消化試験項目数を直接扱うことを目的として，進捗に大きな影響を与えるプログラム品質（試験項目あたりのバグ発見数）を考慮し的確に試験工程進捗を表現するモデルを提案する．具体的には，あらかじめ設計された試験項目がどのように消化されていくかという観点から，時間に対する試験項目消化の様子を発見バグ数を考慮しモデル化する．大規模なソフトウェア開発においては，試験工程は一般的に，単体試験，結合試験，総合試験，運用試験と順に実施される場合が多いが[6)]，本モデルは計画的かつ網羅的にシステム全体の試験を行う結合試験工程および総合試験工程を適用対象とする．

2章で，モデルを構築し，3章でそのモデルを検証する．4章ではこのモデルの応用例として試験期間の予測手法を考察し評価する．

† NTT 情報流通プラットフォーム研究所
Information Sharing Platform Laboratories, Nippon Telegraph and Telephone Corporation
†† 東海大学開発工学部
School of High-Technology for Human Welfare, Tokai University

2. モデル構築

2.1 モデル構築の考え方

大規模なソフトウェア開発における試験工程の試験項目作成においては，作成者によって信頼性がばらつかないように，作業標準によって試験項目抽出基準やプログラム記述量あたりの標準試験項目数などが決められている場合が多い．このように，同じ作業標準を用いているプロジェクトにおいては試験項目抽出基準が統一されている．また，試験項目は，なるべく試験期間を短くし効率的に試験を実施することができるように，全試験項目がプログラムのすべての部分をカバーし，個々の試験項目のカバー範囲がなるべく重ならないように設計されている．したがって，試験工程の進捗度合いを見るために試験項目数の消化の程度を尺度とすることは自然であり，実際に利用されている管理指標でもある．ただし，この尺度は，全試験作業に対する現在の作業の進捗度合いを示しているだけで，いわゆるソフトウェア信頼度の成長度合いは直接には示していないことに注意する必要がある．

最近は，コンピュータシステムのダウンサイジングも進み試験を行ううえでのコスト上の大部分を人件費が占めるようになってきている．試験工程においても，試験を実施する要員が必要とするハードウェア環境は問題なく確保されている場合が多い．そこで，他のリソースに比較して工数が進捗に与える影響が最も支配的であると本モデルでは仮定する（仮定 1）．また，工数には試験項目 1 件あたりの実施工数と発見したバグ 1 件あたりの処理工数があり，それぞれあまりばらつきがないと仮定する（仮定 2）．この仮定は，発見されたバグが開発工程全般に影響を与えるような（設計工程にまで遡って対応が必要な場合など）重大なものではなく，局所的に対応可能であるようなバグのみが発見されている状況であるといえる．したがって，本モデルは，開発工程全般に影響を与えるようなバグが発見されたケースは適用対象外とする．

本モデルの適用可能となる開発工程の条件を整理すると以下となる．

- 試験工程の作業標準があり，試験項目設計については管理された状態である．
- 結合試験工程および総合試験工程である．
- 要員数（工数）が試験実施の速さを決める主要因である．
- 発見されたバグが局所的に対応可能である．

本論文では，単位期間（毎週，毎日など）に実施される試験項目数を p_i とし，試験工程の進捗度合いを表す基本的な指標とする．ただし，i は工程を測定する場合に最初の開始単位期間を 1 として数えたときの期間番号である．たとえば，試験期間 i における試験項目数 p_i などと呼ぶこととする．これらの条件の下で，期間 i において投入された工数 m_i は次のように表すことができる．

$$m_i = \alpha e_i + \beta p_i. \tag{1}$$

ただし，e_i は期間 i に発見されるバグ数，p_i は期間 i に実施される試験項目数（以後，試験進捗と呼ぶ），α はバグを 1 件処理するのに必要な試験工程全体の平均工数，β は試験項目を 1 件処理するのに必要な試験項目全体の平均工数である．

ε_i を期間 i において試験項目 1 件あたりに発見されるバグ数（以後，プログラム品質と呼ぶ）とすると以下が成り立つ．

$$e_i = \varepsilon_i p_i. \tag{2}$$

式 (2) を用いて式 (1) を変形すると p_i に関する以下の式が求まる．

$$p_i = \frac{m_i}{\alpha\varepsilon_i + \beta}. \tag{3}$$

一方，式 (1) をすべての試験期間 $(i = 1, \ldots, N)$ にわたって加算すると，全試験期間に渡り投入された全工数 M が得られる．

$$M = \sum_{i=1}^{N} m_i \tag{4}$$

$$= \alpha \sum_{i=1}^{N} e_i + \beta \sum_{i=1}^{N} p_i \tag{5}$$

$$= \alpha E + \beta P. \tag{6}$$

ただし，E はバグ総数であり，P は試験項目総数である．ここで，α と β の比を δ（デルタ値）とおいて式 (4) をそれぞれ α と β について解くと次の式が得られる．

$$\alpha = \frac{\delta M}{\delta E + P}. \tag{7}$$

$$\beta = \frac{M}{\delta E + P}. \tag{8}$$

式 (7) と式 (8) を式 (3) に代入し以下を得る．

$$p_i = \frac{m_i}{M} \frac{\delta E + P}{\delta\varepsilon_i + 1}. \tag{9}$$

式 (9) において，全試験期間にわたり投入された全試験工数で正規化（全工数を 1 とした割合）した工数分布を λ_i とすると $\lambda_i = m_i/M$ であり，式 (9) は以下の式となる．

$$p_i = \lambda_i \frac{\delta E + P}{\delta\varepsilon_i + 1}. \tag{10}$$

さらに，上式の両辺を P で割ると，進捗率（試験項目総数を 1 とした試験項目数の割合）$\rho_i(=p_i/P)$ に関する，以下の式を得ることができる．

$$\rho_i = \lambda_i \frac{\delta\bar{\varepsilon}+1}{\delta\varepsilon_i+1}. \quad (11)$$

ただし，$\bar{\varepsilon}$ は試験期間全体の平均プログラム品質（$=E/P$）である．

2.2 モデルの分析

本モデルは，試験進捗，工数，プログラム品質の関係を定式化したものである．本節では，このモデルの持つ性質を分析する．

2.2.1 パラメータの性質

本モデルにおいて，試験進捗 p_i を得るために必要なパラメータは式 (10) あるいは式 (11) に基づいて以下となる．

- P：試験項目総数
- E：バグ総数（あるいは，$\bar{\varepsilon}$：平均プログラム品質）
- ε_i：期間 i のプログラム品質
- δ：デルタ値（バグ 1 件対応するのに必要な平均工数と試験項目 1 件に対応するのに必要な工数の比）
- λ_i：期間 i の正規化投入工数（全試験工数に対する期間 i での投入工数の割合）

以下に，プロジェクト実施中にその値を取得する容易さの観点から各パラメータの性質を示す．

レベル 1：実施前に確実に決まっているパラメータ

- P：試験項目総数
  これは，試験が実施される前に設計されており，確実に実施前に決められている．試験進捗度合いの目標となる値である．

レベル 2：実施前にあらかじめ想定されている必要があるパラメータ
  これは，実際の値は実施されてみなければ分からないが，実施前に管理上想定されている必要のある値かあるいは想定に基づいて計算可能なパラメータである．

- E：バグ総数 および ε_i：期間 i のプログラム品質
  類似モジュールの開発経験などの類推から，発生バグ数などをある程度予測することができる．
- δ：デルタ値
  本モデル以外では直接利用されていないが，この値を求めるのに必要な工数は通常管理上の観点から想定可能である．
- λ_i：期間 i の正規化投入工数
  投入工数は工程に入る前に計画されているはずである．

本モデルで使用するレベル 2 パラメータは，試験開始前に推定することが可能であるものと仮定する．

2.2.2 モデルの性質

（1） プログラム品質が一定の場合

プログラムの品質がまったく均質である場合は，ε_i は定数となり，したがって，式 (10) および式 (11) は以下となる．

$$p_i = P\lambda_i. \quad (12)$$

$$\rho_i = \lambda_i. \quad (13)$$

上式より，プログラム品質が一定の場合は，工数分布と試験進捗が同一の形となることが分かる．また，様々なケースについて全体的な傾向をとらえると，個々の期間ごとのバグが平均化されて見えるため，様々なケースの全体としての傾向から工数分布の傾向を得ることができる．

（2） 工数が平均して投入されている場合

工数が平均して投入されている場合は λ_i が定数 $(1/N)$ となる．したがって，式 (10) および式 (11) は以下となる．

$$p_i = \frac{\delta\bar{e}+\bar{p}}{\delta\varepsilon_i+1}. \quad (14)$$

$$\rho_i = \frac{1}{N}\frac{\delta\bar{\varepsilon}+1}{\delta\varepsilon_i+1}. \quad (15)$$

ただし，$\bar{e}$ と $\bar{p}$ はそれぞれ単位期間あたりの平均発生バグ数と平均試験進捗である．上式から，工数が平均して投入されている場合は，プログラム品質の変動に応じて，品質が悪くなると進捗が落ち，良くなると進捗が上がりその関係がほぼ反比例であることを示している．

3. モデルの検証

3.1 検証の考え方

本モデルの理想的な検証は，すべての実測パラメータを与えて，式 (10) から試験進捗を計算し実測値と一致するかどうかを評価することである．ところが，本論文で使用する開発データは実測データとして理想的な検証に必要なすべての情報がそろっているわけではないため，本モデルの検証にあたってはその手法をまず検討する必要がある．

今回検証に利用する実測データには，必要なパラメータのうち，工数分布 λ_i，およびデルタ値 δ が存在しないが，それ以外のパラメータを計算するために必要な情報はすべて存在している．式 (10) から，期間 $i=1,\ldots,k$ について k 本の方程式が得られるが，未知数は $\lambda_i(i=1,\ldots,k)$ と δ の $k+1$ 個であるため，

このままでは一般に λ_i および δ は自由に選ぶことができる．そこで，ここでは，実際の工数分布がそうであるように工数分布 λ_i はある一定の関数形に従うという制約を設ける（一般に工数は突然変更し難く計画的に滑らかに変化するものであるという仮定に基づく）．極端な例としてつねに一定の同一工数がかけられている場合は式 (14) で示したように，各期間 i のプログラム品質を与えるだけで試験進捗を計算できる．この考え方に従い，λ_i を関数値 $\tilde{\lambda}(t_i; a_1, \ldots, a_m)$ で近似することとする．ただし，$\tilde{\lambda}(t; a_1, \ldots, a_m)$ はある滑らかな関数，t_i は期間 i を代表する開始からの経過時間，$a_1, \ldots, a_m$ は具体的な関数を決定するパラメータである．したがって，式 (10) の λ_i を $\tilde{\lambda}(t_i; a_1, \ldots, a_m)$ で置き換えた以下の式を評価用のモデルとして用いる．

$$\tilde{p_i} = \tilde{\lambda}(t_i; a_1, \ldots, a_m) \frac{\tilde{\delta} E + P}{\tilde{\delta} \varepsilon_i + 1}. \tag{16}$$

ただし，$\tilde{p_i}$ は期間 i で実施される試験項目数の推定値である．モデルを決めるパラメータ $a_1, \ldots, a_m$ および $\tilde{\delta}$ を決定する制約条件（アンド条件）は以下となる．

• 実測値との誤差最小条件

モデル計算値と実測値の誤差を最小にするように決定するための条件である．誤差の評価方法としては，各試験期間での試験進捗 p_i の誤差を最小化する方法と累積試験進捗 $\sum p_i$ を最小化する方法が考えられる．最終目標に向かった現在の進捗状況把握という点では，最終目標に対してどれくらい進んでいるかという評価が重要となるため，ここでは累積試験進捗の誤差を最小にするような，以下の条件を満たすものとする．

$$E = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} \tilde{p_j} - \sum_{j=1}^{i} p_j \right)^2 \to \min. \tag{17}$$

• 総試験項目数の一致条件

総試験項目数が一致することは，試験進捗を把握する目的からみて重要な条件である．誤差を最小になるように各パラメータを決定したとしても，一般には総試験項目数が一致するとは限らない．そこで，以下の要件を満たす必要がある．

$$\sum_{i=1}^{N} \tilde{p_i} = P. \tag{18}$$

ここで，$\tilde{\lambda}(t_i; a_1, \ldots, a_m)$ および $\tilde{\delta}$ は想定値であって実際の値ではないが，たとえこれらの値が想定値であったとしても試験進捗に関する実測値と計算値が一致していることはモデルの正当性の必要条件を満たしているといえる．逆に，実測値から逆算したモデルではあってもモデルが誤っている場合は実測値と計算値は一致しなくなる．この方法を用いることにより，工程管理の適用例で述べるように，工数情報を直接用いずにモデルの計算値が実測値とよく一致することも検証できる．すなわち，工数情報がない状態でも過去の試験進捗とプログラム品質から将来の進捗状況を予測することが可能となる．

3.2 検証の手順

式 (17) を満たす $a_1, \ldots, a_m$ および $\tilde{\delta}$ は，$\partial E / \partial a_i = 0 (i = 1, \ldots, m)$ および $\partial E / \partial \delta = 0$ を満たす方程式から求めることができる．しかし，式 (17) を見れば容易に分かるように，これらの方程式は少なくとも非線形，多くの場合は超越方程式となって，一般には解析的に解を求めることができず，数値計算で解を求めることになる．本論文はモデルの提案と検証が主目的であるため，数値計算で解を求める効率的な手法は議論しない．ここでは次に示す方法でパラメータを近似的に求め，実測値との誤差を評価することとする．

パラメータは次のように求める．

(1) ステップ 1：仮の $\tilde{\delta}$ の設定と仮の $\tilde{a_1}, \ldots, \tilde{a_m}$ の計算

まず仮の $\tilde{\delta}$ を決め，試験進捗の実測値 p_i から式 (10) を用いて λ_i を計算する．次に，式 (19) を満たすように $a_1, \ldots, a_m$ を求める．

$$E = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} \tilde{\lambda}(t_j; a_1, \ldots, a_m) - \sum_{j=1}^{i} \lambda_j \right)^2 \to \min. \tag{19}$$

(2) ステップ 2：仮の $\tilde{\delta}$ の再計算

式 (18) を満たすように $\tilde{\delta}$ の値を修正する．

(3) ステップ 3：各パラメータの仮り決め

(4) ステップ 4：誤差 E の評価

式 (17) で示す誤差 E を評価し，収束点と思われる値となるまでステップ 1 からの操作を繰り返す．

上のステップで $\tilde{\delta}$ と $a_1, \ldots, a_m$ を確定する．したがって，試験進捗計算に際して確定していない値は各試験期間のプログラム品質 ε_i のみとなる．この値の実測値を試験期間ごとに与えることにより各試験期間の試験進捗 $\tilde{p_i}$ を計算し，式 (17) における誤差 E を評価し，その大きさを対照モデルと比較することにより妥当性を検証する．

なお，一般に条件 (18) の制約条件がない場合のほうが，より誤差が小さくできることから単純に条件 (17) のみでパラメータを求める評価よりも厳しい評価であるといえる．

3.3 検　　証

3.3.1 検証に用いる開発データ

モデルを検証するためのデータとして，実際のソフトウェア開発現場で収集された 9 種類のソフトウェア開発の試験工程進捗状況管理データを利用した（サンプル 1 からサンプル 9）．開発ソフトウェアは言語処理プログラム，ソフトウェア設計支援システム，データベース管理システムや各種アプリケーションなど様々である．総規模は，延べ 400 K ライン程度である．試験項目総数は，約 63,000 項目，バグ数は約 6,000 件に及ぶ．測定されたデータは実施試験項目数およびバグ数であり，週単位で集計されている．工数については測定されていない．また，発見バグの種類や重大度のデータも収集されていない．このため 2 章で示した本モデル適用上の仮定のうち仮定 2 が成り立っているサンプルかどうか確認することができない．そこで，今回は上記すべてのサンプルについてまず本モデルを適用することにする．大部分がモデル計算値と一致すればモデルは正しいと判断する．一致しないサンプルについては，仮定が成り立っていない可能性について検討することとする．

また，4 章においては，あえて上記データとはまったく異なるデータである文献 7) で紹介されたデータを用いて上記データと同様な検証を試みるとともに，適用例を示す（サンプル 10）．

3.3.2 近似曲線の準備

前章で示したように，プログラム品質が一定の場合は試験進捗と工数分布 $\tilde{\lambda}(t : a_1, \ldots, a_m)$ が一致するため，一次近似として試験進捗と同一な関数形で工数分布を表現できると考えられる．また，3.1 節で述べたのと同様な理由から累積試験進捗曲線により関数形を決定する．**図 1** に，最大値を 1 に正規化した累積試験進捗曲線をサンプル 10 を除くすべてのサンプルについて示す．図より，おおよそ S 字曲線型や指数形となっていることが分かる．このような曲線で近似すると比較的一致する傾向にあるのは，試験開始直後はまずメインルートの正常系を確認してから枝葉の正常系さらには異常系と試験が進んでいくために，確保している試験要員が実施できる試験項目に比べて最初は実施できる試験項目が少ないが，中盤では非常に多くなり，後半では徐々に少なくなるためであると思われる．

ここでは，S 字曲線から指数形の曲線まで広くカバーする統合モデル[8),9)] を用いる．このモデルを用いることにより特定の形の曲線に依存しない汎用モデルとすることができる．統合モデルの一般解には，統合モデルを表す微分方程式のパラメータの範囲により，3 つの形式のものが存在する．このうち，(1) 指数形と S 字形の曲線を 1 つの式でカバーすることができる，(2) $t = 0$ で $y = 0$ となる解を持つ，(3) 一般解に含まれるパラメータ数が最も少ないという 3 つの条件を満たす次の形式のもの[10)] をここでは用いることとする．

$$y = C_3(1 - \exp(-C_1 t))^{C_2}. \qquad (20)$$

ここで，C_1 は時間的な伸び具合を，C_2 は曲線の曲がりの程度を意味している．C_2 の値の変化により曲線の形が S 字形から指数形まで連続的に変化する．

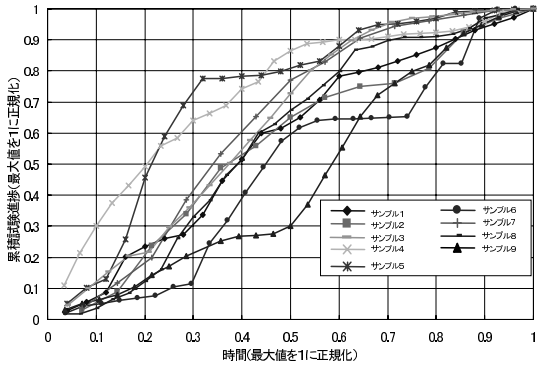

図 1 累積試験進捗実測データ（時間軸と試験進捗を最大値 1 に正規化）
Fig. 1 Actual data vs. estimated values.

本モデルでは試験期間 N ですべての試験項目が終了しているため，試験期間 N での累積工数の値は 1 にならなくてはならない．また，本モデルは離散系のモデルであるが，試験期間の単位を 1 ずつ増えるとしているため，連続モデルの t を i/N に置き換えるのみで，λ_i を表現できる．さらに $t = 1$ で $y = 1$ という正規化条件を考慮することにより，累積工数として以下の式を得る．

$$\sum_{j=1}^{i} \tilde{\lambda}(t_j; C_1, C_2) = \frac{(1-\exp(-C_1 \frac{i}{N}))^{C_2}}{(1-\exp(-C_1))^{C_2}}. \qquad (21)$$

3.3.3 評　　価

前節で示した手順に従って，パラメータ C_1，C_2 と δ を求める．またプログラム品質を考慮しない比較モデルとして，累積試験進捗曲線を滑らかな関数，すなわち式 (21) で示される関数形を仮定して直接式 (17) で近似した場合の C_1，C_2 を求める．9 サンプルのうち，サンプル 9 を除いた 8 例が一見して形が良く一致している．サンプル 9 については，比較手法と比べてより相対誤差率が小さくなるようにすることができなかった．サンプル 9 を除く両者のパラメータの値と相対誤差率（絶対値（計算値 − 実測値）/実測値）の平均値を**表 1** に示す．また，**図 2** に実測値と両者の計算

表 1 パラメータ値と誤差
Table 1 Parameters and errors.

サンプル番号		1	2	3	4	5	6	7	8
本モデル	C_1	2.6	2.8	1.9	4.8	3.2	4.6	4.4	4.1
	C_2	1.6	2.0	0.8	2.1	2.2	3.5	3.6	2.3
	δ	2.2	1.1	2.6	1.3	52	5.7	28	22
	相対誤差率（%）	6.1	8.2	3.1	8.4	18.2	7.6	7.9	10.1
最小二乗法	C_1	2.6	2.2	3.4	5.4	2.2	5.2	4.6	4.4
	C_2	1.9	1.7	0.9	2.1	2.5	3.9	4.1	3
	相対誤差率（%）	7.3	11.7	4.4	9.3	22.9	8.2	10.6	13.3

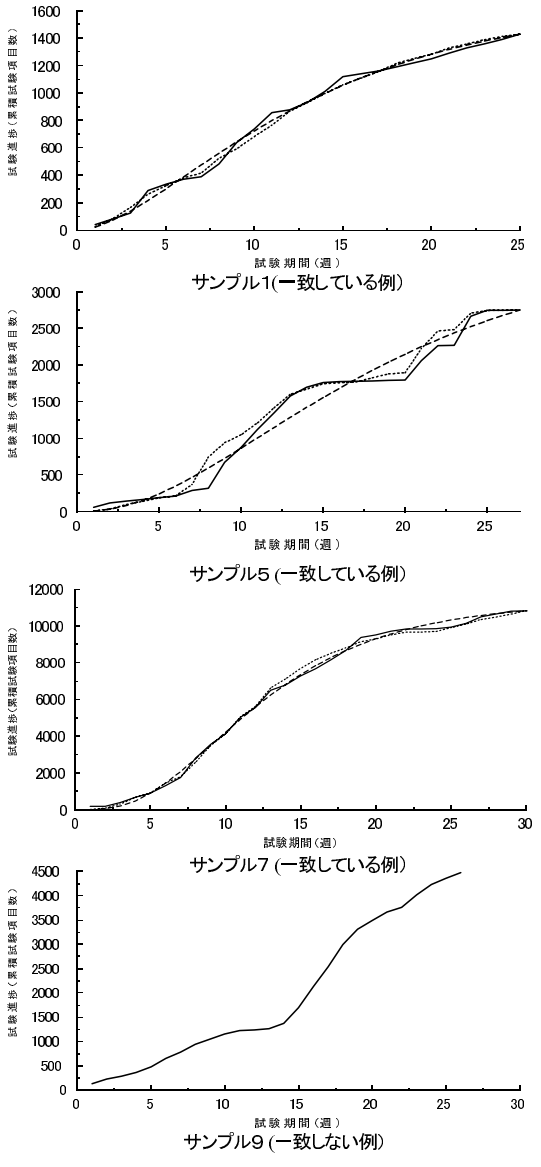

図 2 実測値と計算値の例
Fig. 2 An example of practical values and estimated values.

値をプロットしたグラフをよく一致している 3 例と一致しなかった 1 例について示す．一致しなかった原因は，おそらく試験工程半ば（大きく進捗が落ち込んでいるところ）で予想を大きく超える重大なバグが発見されその対応に工数の大部分がとられたために，プログラム品質とは直接関係なく試験進捗が落ち込んだものと思われる．サンプル 9 を除く相対誤差率の平均は本モデルが 8.7%，比較手法が 10.96%であり，21%程度より正確に一致していることが分かる．また，サンプル 5 のケースのように一見して凹凸が一致していることも分かる．

ところで，デルタ値は 1.1 から 52 までかなりばらつきがある．これは，試験項目を 1 件処理する工数に対して，発見したバグを 1 件処理する工数が 1 から 50 倍程度のばらつきがあることを示している．この値は，試験項目の処理工数の平均と発生バグの処理工数の平均によって決まる．これについては，次のことが考えられる．1 つの試験項目の内容はプロジェクトや使用している作業標準によって様々に異なる．たとえば，大きな機能単位（関数単位）に管理上の試験項目番号を割り振る場合もあれば，関数の入力データのバリエーション 1 つ 1 つに試験項目番号を割り振る場合もあり，プロジェクトによるばらつきが大きいと考えられる．このばらつきに加え，バグの処理工数のばらつきが重なるためデルタ値もばらつきが大きくなったものと考えられる．実際，今回のサンプルでは，KL あたりの試験項目数が 14 倍程度開きがあり，上の仮説を裏付けている．

4. 工程管理への適用例

本モデルの応用は様々考えられるが，ここでは，試験工程終了時期の予測法について考察する．

本モデルを用いることにより，試験工程実施中に試験工程がどのくらいで終了することができるか（試験項目をすべて終了することができるか）を予測する方法の例を示す．例としては，あえてモデルの検証には用いなかった，文献 7) で紹介されたデータを用いる

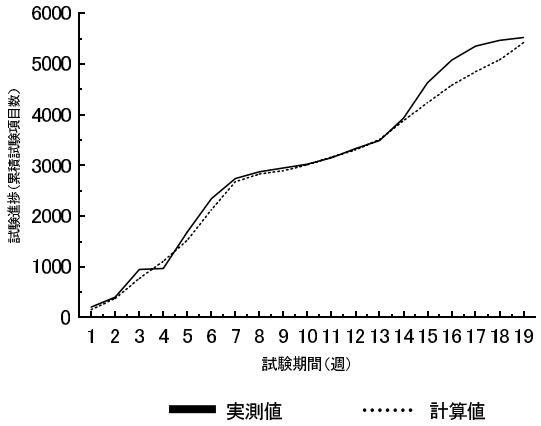

図 3 サンプル 10 の実測値と計算値
Fig. 3 Practical values and estimated values of sample 10.

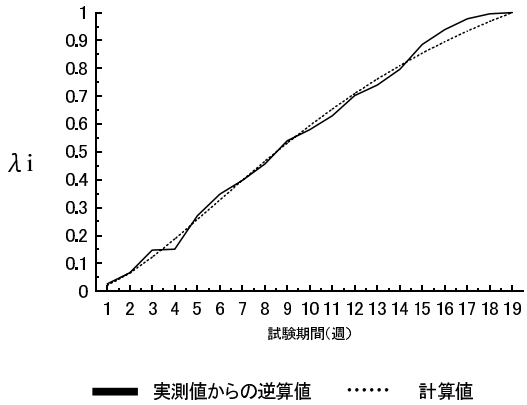

図 4 サンプル 10 の工数分布計算値 λ
Fig. 4 λ of sample 10.

（サンプル 10）．このデータは本論文で検証用に用いたデータと同様に週単位で実施試験項目数とバグ項目数が集計されている（工数データは紹介されていない）．なお，本データの最初の週は進捗 0 のため考えず，第 2 週目を試験期間 1 として最終週を 19 週として表した．

4.1 予 測 方 法

前章で示したように，累積試験進捗曲線はプログラム品質に強く影響されるため，より精度良く試験期間を予測するためには，プログラム品質にあまり影響されない曲線をベースにその曲線を推測するのがよい．要員の確保は通常試験実施前に行われており，要員の変動はコストに最も影響するため，状況が非常に大きく変わらない限り最初に確保した要員の範囲内で試験が実施されることが多い．このため，工数分布はプログラム品質に直接的には影響しないと考えられる（もちろんバグが予想以上に発生した場合などは要員を増やすなどの対応策がとられる場合もあると考えられるが，例外的事項と考えて本モデルの対象外とする）．

ここでは，式 (10) から逆算により正規化工数分布を求め，3.2 節で示した手順で，C_1，C_2，N を求め，N を予測試験期間とする．なお，工数分布の計算にあたってはデルタ値 δ および平均プログラム品質 $\bar{\varepsilon}$ を知る必要があるが，ここでは，便宜的に，デルタ値についてはあらかじめ前章と同様の方法で求め，平均プログラム品質については全期間から平均を計算してその値を用いた．

4.2 シミュレーション

図 3 に実測値と本モデルの計算値を示す．各試験期間のプログラム品質は実測値を与え前章と同様な方法で計算値を求めた図である．また，**図 4** に工数分布

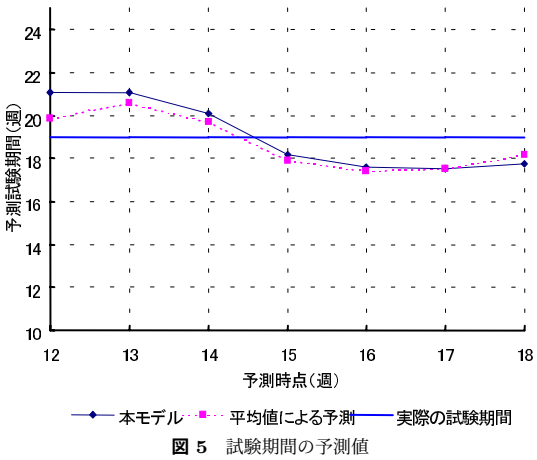

図 5 試験期間の予測値
Fig. 5 An example of testing period evaluation.

λ_i の式 (10) からの実測データによる逆算値と近似値による計算値を示す．これらより，本モデルの計算値が実測値とよく一致していること，および工数分布の変動が試験進捗曲線よりもなだらかな変化となっていることが分かる．各予測時点で計算した試験期間（終了予定時期）を本手法を用いた場合と総試験項目数をその時点の平均試験進捗で割ることにより予測した場合（平均値による予測）の値を**図 5** に示す．

サンプル 10 の場合，15 週，16 週および 17 週では本手法の予測誤差の方が少なかったが，他の週では平均値による予測の方が予測誤差が少なかった．これは，サンプル 10 の累積試験進捗分布が S 字形ではなく直線系に近かったためであり，S 字度合いが大きくなるほど本手法の予測精度が高くなると思われる．

5. む す び

本論文では，従来から行われている試験項目数とそ

の予定処理時間からの予測と比較してより正確な試験工程管理のベースとなる試験進捗モデルを構築し，実際の開発データをもとに検証した．プログラム品質を考慮した本モデルは考慮しない場合に比べてより精度良く試験進捗状況を表現できることを示した．本モデルを適用することにより，試験項目消化数とバグ発見数というどのようなプロジェクトでも必ず測定している基礎的なデータをもとに，簡単により正確な進捗管理が可能となる．

本モデルの適用については試験期間予測の一例を提示したが，本例で示した試験期間予測法の一般的性質，適用領域，他の手法と比較した場合の優位性などについては今後さらに明確化する予定である．

参 考 文 献

1) 大場 充：ソフトウェア信頼性モデル入門，情報処理，Vol.31, No.12, pp.1623–1630 (1990).
2) 当麻喜弘：超幾何分布に基づくソフトウェア残存フォールト数推定モデル，情報処理，Vol.31, No.12, pp.1641–1646 (1990).
3) 山田 茂：ソフトウェア信頼性モデル—基礎と応用，日科技連出版社 (1994).
4) Yamada, S., Ohba, M. and Osaki, S.: S-S haped reliability growth modeling for software error detection, *IEEE Trans. Rel.*, Vol.R-32, No.5, pp.475–478 (1983).
5) 木村光宏，山田 茂：テスト消化過程に基づくソフトウェアテス進ちょく度評価モデルに関する考察，信学論（D-I），Vol.J79-D-I, No.12, pp.1211–1217 (1996).
6) 藤野喜一，花田収悦：ソフトウェア生産技術，社団法人電信通信学会 (1985).
7) 芝田寛二：ソフトウェア製品の生産計画と工程管理，情報処理，Vol.21, No.10, pp.1035–1042 (1980).
8) Furuyama, T. and Nakagawa, Y.: A manifold growth model that unifies software reliability growth Models, *International Journal of Reliability, Quality and Safety Engineering*, Vol.1, No.2, pp.161–184 (1994).
9) 古山恒夫，中川 豊：ソフトウェア成長曲線に関する統合モデルと有効性の検証，情報処理学会ソフトウェア工学研究会報告，Vol.97, No.10, pp.73–80 (1994).
10) 古山恒夫：ソフトウェア信頼度成長モデルに関する統合モデルの解析的パラメータ推定法，情報処理学会論文誌，Vol.37, No.12, pp.2326–2333 (1996).

(平成 12 年 8 月 29 日受付)
(平成 13 年 6 月 19 日採録)

金井 敦（正会員）

昭和 55 年東北大学工学部通信工学科卒業．昭和 57 年同大学院工学研究科情報工学教室博士前期課程修了．同年日本電信電話公社横須賀電気通信研究所入所．以来，ソフトウェア移植モデル，クロスコンパイラ，分散開発環境，通信ソフトウェア開発法，インターネットサービス開発技術の研究に従事．現在，NTT 情報流通プラットフォーム研究所アプリケーションプラットフォームプロジェクト主幹研究員．電子情報通信学会，IEEE 各会員．

古山 恒夫（正会員）

昭和 20 年生．昭和 43 年東京大学工学部計数工学科卒業．昭和 48 年同大学院博士課程修了．同年日本電信電話公社入社．横須賀電気通信研究所で，拡張型言語，Ada，Common LISP などの言語処理プログラムの研究実用化に従事．日本電信電話（株）ソフトウェア研究所でソフトウェアプロジェクト管理法，ソフトウェア品質保証法，ソフトウェア見積り法などの研究実用化に従事．平成 8 年東海大学開発工学部情報通信工学科教授．平成 6 年度情報処理学会山下記念研究賞受賞．IEEE，電子情報通信学会，ヒューマンインタフェース学会各会員．工学博士．