品質改善プロセスの行列表現による記述モデル†

砂塚 利彦††

ソフトウェア生産の自動化が進んできているが，事務処理のような機能のパターン化が容易な分野以外では必ずしも普及しているとはいえない．したがって，現時点において高品質のソフトウェアを開発するには，人間側の努力がかなり必要となる．一般に品質向上の施策としてレビューと検査がある．検査の前までにいかに品質を作り込むか，また検査ではいかに残存エラーを少なくするかが問題となる．そこで，レビューや検査の施策に対する努力の投入量と，その効果との関係を知り，品質改善のプロセスの最適化をはかることを本研究の目的とする．本論文では，ソフトウェア開発における品質改善プロセスを開発期間と検査期間とに分け，エラーが作り込まれ，摘出されるプロセスを 2×2 の行列による記述モデルで表現する．開発プロセスは，（エラーの作り込み）と（エラーの摘出）が開発工程の数だけ繰り返されると考え，両方のモデルを定義した．また，検査プロセスは，（エラー摘出の軌跡）と考え，摘出エラー数の推移を離散型の信頼度成長モデルで記述した．これらのモデルに基づいて，開発・検査プロセスを定量的に管理・改善する方法を提案する．最適施策としては，累積エラー数が増すにしたがって，すなわち工程が進むにしたがってエラー発見率を大きくしていくことである．これにより検査開始時における含有エラー数が押えられる．検査工程では，本モデルは検査環境の変化によるパラメータの変化に柔軟に対応できることがわかった．

1. はじめに

近年ソフトウェアの生産技術が目覚しい進歩をとげ，設計支援ツール，コードジェネレータ，検査ツールなどの個別ツールをはじめ，一貫生産システムまで利用され始めている．これらによって生成されたプログラムの品質は，パターン化された機能の実現や再利用をする場合には非常に優れている．それ以外の場合には品質には影響を与えず，人手による作業が効率化されるにとどまる．なぜなら，システムは人間が考えたことをそのまま実現するので，人間の考え方が誤っていた場合には品質まで保証することはできない．現時点においては高品質のソフトウェアを開発するためには人間側の努力がかなり必要である．

一般に品質向上の施策としてレビューと検査がある．すなわち，検査の前までにいかにいいものを作るか，また検査ではいかに効率よくエラーを摘出し，残存エラーを少なくするかが問題となる．このほか作業の標準化，支援ツールの活用，教育などによっても品質の向上が期待できる．

本論文では，ソフトウェア開発における品質改善プロセスを開発期間と検査期間とに分け，エラーが作り込まれ，摘出されるプロセスを 2×2 の行列による記述モデルで表現する．開発プロセスでは，エラーの作り込みと摘出を記述し，検査プロセスでは，離散型の信頼度成長モデルを行列で表現した．最後に開発・検査プロセスを改善するための計量管理の方法を提案する．

2. 品質改善プロセス

ソフトウェアを開発し，検査し，リリースするまでのプロセスは，品質という視点から見ると，エラーの作り込みと摘出の繰り返しとみることができる．仕様書の作成やプログラミングではエラーを作り込み，レビューと検査とでエラーを摘出するのである．したがって，いかに作り込むエラーの数を減らすか，またいかに効果的にエラーを摘出するかでリリース時の品質が決まる．本章では，このメカニズムを開発プロセスと検査プロセスとに分けて定義する（**図1**参照）．

（1） 開発プロセス

開発プロセスは，（エラーの作り込み）と（エラーの摘出）が開発工程の数だけ繰り返されると考えられる．エラー数の変化を**図2**に示す．

（2） 検査プロセス

検査プロセスは，（エラー摘出の軌跡）と考えられる．ここでは，検査工程における摘出エラー数の推移を離散型の信頼度成長モデルで記述する．

3. 開発プロセスの記述モデル

はじめに列ベクトル $[y_{ij}, x_{ij}]'$ を定義する．

y_{ij}：工程 i，時点 j までの累積摘出エラー数

x_{ij}：工程 i，時点 j における残存エラー数

† Software Quality Improvement Process Model by Matrix Expression by TOSHIHIKO SUNAZUKA (Software Management Department, Software Engineering Development Laboratory, NEC Corporation).

†† 日本電気(株)ソフトウェア生産技術開発本部管理システム開発部

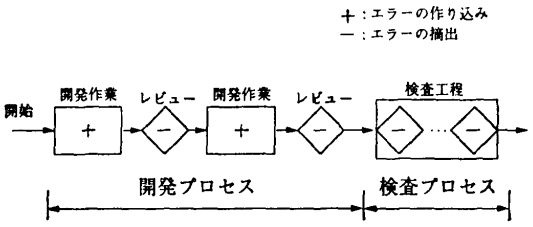

図 1 品質改善プロセスのメカニズム
Fig. 1 Mechanism of quality improvement process.

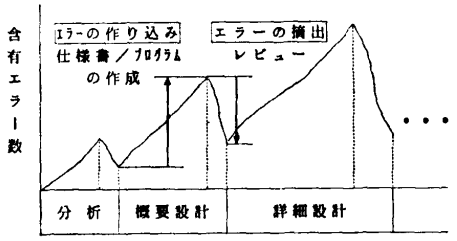

図 2 開発プロセスにおけるエラー数の推移
Fig. 2 Number of errors at each development phase.

ただし，時点 $j=1$ はレビュー前，$j=2$ はレビュー後を表す．

このとき y_{ij}, x_{ij} を次式で表すことを考える．

$$\begin{bmatrix} y_{ij} \\ x_{ij} \end{bmatrix} = \begin{bmatrix} a(i,j) & b(i,j) \\ c(i,j) & d(i,j) \end{bmatrix} \begin{bmatrix} y_0 \\ x_0 \end{bmatrix} = \boldsymbol{A}_{ij} \begin{bmatrix} y_0 \\ x_0 \end{bmatrix}$$

ただし，x_0 は開発を始める前の推定総エラー数とする．y_0 は累積摘出エラー数の初期値なので0である．

この関係を表す関数 a, b, c, d を定義していく．ここで $\boldsymbol{A}_{ij}$ はさらに次のように展開できる．

$$\boldsymbol{A}_{ij} = (\boldsymbol{Q}_i^{-1})^{2-j} \cdot \boldsymbol{Q}_i \cdot \boldsymbol{P}_i \cdot \boldsymbol{Q}_{i-1} \cdot \boldsymbol{P}_{i-1} \cdots \boldsymbol{Q}_1 \cdot \boldsymbol{P}_1$$

ただし，$(\boldsymbol{Q}_i^{-1})^0 = E$（単位行列）とする．

この式は，$\boldsymbol{P}$ でエラーの作り込み，$\boldsymbol{Q}$ で摘出を表し，これを繰り返すことを意味する．このとき，累積摘出エラー数 y_{ij} については次のことがいえる．

$$y_0 = 0$$

で，以後増加していく．また，ある工程でレビューが終了してから次の工程でレビューが開始されるまではエラーは摘出されないと考える．

$$y_{i,2} = y_{i+1,1}$$

残存エラー数 x_{ij} については，開発前における推定総エラー数を a とすると，

$$x_0 = a$$

となる．開発開始後はエラーが作り込まれると増加し，摘出されると減少する．

$$\cdots < x_{i,1} > x_{i,2} < x_{i+1,1} > \cdots$$

また，y_{ij} と x_{ij} には次の関係がある．

$$y_{i,1} + x_{i,1} = y_{i,2} + x_{i,2}$$

すなわち，同じ工程内では，エラー総数はレビューの前後で一定である．

$$y_{i+1,j} + x_{i+1,j} = y_{i,j} + x_{i,j} + a \cdot p_{i+1} \quad (0 < p_{i+1} < 1)$$

すなわち，次の工程では $a \cdot p_{i+1}$ だけエラーが作り込まれる．ただし，ここでは波及エラー（後述）を考慮していない．

（1） エラー作り込み/摘出モデル

ここで以下を仮定する．

① 過去の実績データから，それぞれの工程でのエラー作り込み比率が推定できる．

② 過去の実績データから，それぞれの工程でのレビューによるエラー発見率が推定できる．

これらの仮定から，エラー作り込み/摘出モデルは次のような行列で表現できる．

エラー作り込みモデル

最初の工程（$n=1$）は，推定総エラー数 x_0 のうちの比率 p_1 が作り込まれる．

$$\begin{bmatrix} y_{1,1} \\ x_{1,1} \end{bmatrix} = \begin{bmatrix} y_0 \\ p_1 \cdot x_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & p_1 \end{bmatrix} \begin{bmatrix} y_0 \\ x_0 \end{bmatrix}$$

ここで，$\boldsymbol{P}_1$ をエラー作り込みの行列とすると，

$$\boldsymbol{P}_1 = \begin{bmatrix} 1 & 0 \\ 0 & p_1 \end{bmatrix}$$

2番目以降の工程（$n \geqq 2$）は，直前の工程の残存エラーに，さらに推定総エラー数 x_0 のうちの比率 p_i が作り込まれる．

$$\begin{bmatrix} y_{i,1} \\ x_{i,1} \end{bmatrix} = \begin{bmatrix} y_{i-1,2} \\ x_{i-1,2} + p_i x_0 \end{bmatrix}$$

ここで，

$$x_{i-1,2} + y_{i-1,2} = \sum_{k=1}^{i-1} p_k x_0$$

（p_i：工程 i でのエラー作り込み比率（$\Sigma p_i = 1$））

という関係があるので，

$$= \begin{bmatrix} y_{i-1,2} \\ x_{i-1,2} + \dfrac{p_i}{\sum_{i-1} p_k} (x_{i-1,2} + y_{i-1,2}) \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0 \\ \dfrac{p_i}{\sum_{i-1} p_k} & 1+\dfrac{p_i}{\sum_{i-1} p_k} \end{bmatrix}\begin{bmatrix} y_{i-1,2} \\ x_{i-1,2} \end{bmatrix}$$

したがって，エラー作り込み行列 $\boldsymbol{P}_i$ は次のようになる．

$$\boldsymbol{P}_i=\begin{bmatrix} 1 & 0 \\ \dfrac{p_n}{\sum_{n-1} p_i} & 1+\dfrac{p_n}{\sum_{n-1} p_i} \end{bmatrix}$$

波及エラー（前工程での残存エラーが原因で新たに産み出されたエラー）も考慮に入れる場合には，次の行列を用いるとよい．

$$\boldsymbol{P}_i=\begin{bmatrix} 1 & 0 \\ \dfrac{p_n}{\sum_{n-1} p_i} & 1+\dfrac{p_n}{\sum_{n-1} p_i}+u_n \end{bmatrix}$$

u_n：波及エラー率

エラー摘出モデル

工程 i でのエラー発見率を q_i とすると，次のように表せる．

$$\begin{bmatrix} y_{i,2} \\ x_{i,2} \end{bmatrix}=\begin{bmatrix} y_{i-1,1}+q_i x_{i-1,1} \\ (1-q_i)x_{i-1,1} \end{bmatrix}$$

$$=\begin{bmatrix} 1 & q_i \\ 0 & 1-q_i \end{bmatrix}\begin{bmatrix} y_{i-1,1} \\ x_{i-1,1} \end{bmatrix}$$

したがって，エラー摘出行列は次のようになる．

$$\boldsymbol{Q}_i=\begin{bmatrix} 1 & q_i \\ 0 & 1-q_i \end{bmatrix}$$

例えば，開発時に作り込むエラー総数の推定値を 100 とし，工程 1 でその 30% を作り込み（$p_1=0.3$），レビューでそのうちの 60% のエラーを摘出する（$q_1=0.6$）の場合には次のようになる．

$$\begin{bmatrix} y_0 \\ x_0 \end{bmatrix}=\begin{bmatrix} 0 \\ 100 \end{bmatrix}\begin{matrix} \leftarrow \text{累積エラーの初期値は } 0 \\ \leftarrow \text{残存エラーの初期値は } 100 \end{matrix}$$

$$\begin{bmatrix} y_{1,1} \\ x_{1,1} \end{bmatrix}=\boldsymbol{P}_1\begin{bmatrix} y_0 \\ x_0 \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0 \\ 0 & 0.3 \end{bmatrix}\begin{bmatrix} 0 \\ 100 \end{bmatrix}=\begin{bmatrix} 0 \\ 30 \end{bmatrix}$$

工程 1 で 30 個のエラーが作り込まれた．

$$\begin{bmatrix} y_{1,2} \\ x_{1,2} \end{bmatrix}=\boldsymbol{Q}_1=\begin{bmatrix} y_{1,1} \\ x_{1,1} \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0.6 \\ 0 & 0.4 \end{bmatrix}\begin{bmatrix} 0 \\ 30 \end{bmatrix}=\begin{bmatrix} 18 \\ 12 \end{bmatrix}$$

レビューで 18 個のエラーを摘出し，まだ 12 個が残っている．

（2） QA 活動の効果表

エラーの作り込み/摘出の割合を正確に見積るためには過去の実績に裏付けられた p_i, q_i の効果表を作成するとよい．

表 1 q_i の効果（詳細設計工程での例）
Table 1 Sample effects of q_i (in detail design phase).

QA 活動 ＼ 効果のレベル	1 高い	2 やや高	3 やや低	4 低い
セルフチェック	0.25	0.20	0.15	0.10
クロスチェック	0.40	0.35	0.30	0.25
ストラクチャード・ウォークスルー	0.50	0.45	0.40	0.35
公式レビュー	0.80	0.70	0.60	0.50

例えば，W. Humphrey[6] のソフトウェア成熟度に合わせて p_i の係数を決めるとか，C. Jones[8] のレビューの実施のレベルに合わせて q_i の係数を決めるなどが考えられる．**表 1** に，q_i の効果の例を示す．

（3） 活用例

先に工程 1 でのエラー作り込み/摘出の例を示したが，この行列を利用することによって，目標の設定や結果の記述が可能になる．また，作り込み数が多かったり，発見率が低かったりした場合，新たな対策や効率アップが必要であることがわかる．

① エラー作り込み/摘出の例

先の例において，エラー総数を 20% 削減して 80 個という目標をたてる．作り込みの比率は同一の 30%（$p_1=0.3$）を用い，発見率を 10% アップの 70%（$q_1=0.7$）を目指すとする．このとき，工程 1 の終了時の残存エラー数は 12 から 7 へ大幅に削減することになる．

$$\begin{bmatrix} y_{1,2} \\ x_{1,2} \end{bmatrix}=\boldsymbol{Q}_1\boldsymbol{P}_1\begin{bmatrix} y_0 \\ x_0 \end{bmatrix}$$

$$=\begin{bmatrix} 1 & 0.7 \\ 0 & 0.3 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 0.3 \end{bmatrix}\begin{bmatrix} 0 \\ 80 \end{bmatrix}$$

$$=\begin{bmatrix} 17 \\ 7 \end{bmatrix}$$

② 波及エラーの例

前述の 2 つの例を使って，工程 2 でのエラー数の推移を調べてみる．ただし，作り込み比率，発見率は同じ値を用いる．また，波及するエラーは，残存エラーの 3 分の 2 が 1.5 倍になるものとする．したがって，$u_i=(1/3)\times 1+(2/3)\times 1.5-1=1/3$ となり，エラーが 33% 増えることを意味する．

摘出率が 0.6 のほうは次のようになる．

$$\begin{bmatrix} y_{2,2} \\ x_{2,2} \end{bmatrix}=\boldsymbol{Q}_2\boldsymbol{P}_2\begin{bmatrix} y_{1,2} \\ x_{1,2} \end{bmatrix}$$

$$=\begin{bmatrix}1 & 0.6\\ 0 & 0.4\end{bmatrix}\begin{bmatrix}1 & 0\\ 1^{+} & 7/3^{*}\end{bmatrix}\begin{bmatrix}18\\ 12\end{bmatrix}$$

$$=\begin{bmatrix}45.6\\ 18.4\end{bmatrix}$$

$^{+}$ $1=0.3/0.3$

* $7/3=1+(0.3/0.3)+1/3$

一方，摘出率が0.7のほうは，

$$\begin{bmatrix}1 & 0.7\\ 0 & 0.3\end{bmatrix}\begin{bmatrix}1 & 0\\ 1 & 7/3\end{bmatrix}\begin{bmatrix}17\\ 7\end{bmatrix}=\begin{bmatrix}40.5\\ 10.0\end{bmatrix}$$

となり，残存エラー数は工程2の終了時で，18.4と10.0のようにさらに差が開いていることがわかる．

4. 検査プロセスの記述モデル

ここでの検査プロセスは離散型の信頼度成長モデルであると定義する．したがって，検査の進行とともに推移する累積エラーを記述するモデルを提案する．代表的な指数型/S字型モデルと込山-砂塚モデルを行列を使って表現し，それぞれの発見率を算出する．

（1） モデルの一般形

検査プロセスにおけるエラーの摘出モデルは，マルコフ連鎖と考えることができる．**図3**に示すように，状態が2つ(「y：累積エラー」と「x：残存エラー」)あり，yはxへ遷移せず，xは検査時点に依存した摘出率分だけyへ遷移する．

したがって，以下のようなモデルが定義できる．

$$\begin{bmatrix}y_n\\ x_n\end{bmatrix}=\begin{bmatrix}1 & f(n;\boldsymbol{b})\\ 0 & 1-f(n;\boldsymbol{b})\end{bmatrix}\begin{bmatrix}0\\ a\end{bmatrix}$$

f：時点nでのエラー発見率

n：検査時点

a：検査開始時の残存エラー数

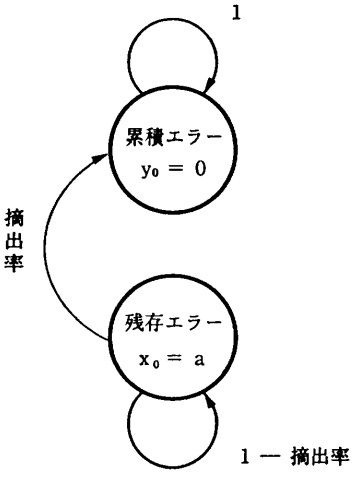

図3 検査プロセスのメカニズム
Fig. 3 Mechanism of test process.

$\boldsymbol{b}$：パラメータ群

この行列の性質については，付録Aで説明する．

（2） 指数型/S字型モデル

よく利用される2つのモデルについて累積エラー数，発見率，エラー1個当たりの発見率を求める．

指数型モデル

$$\begin{bmatrix}y_n\\ x_n\end{bmatrix}=\begin{bmatrix}1 & 1-(1-b)^n\\ 0 & (1-b)^n\end{bmatrix}\begin{bmatrix}0\\ a\end{bmatrix}$$

発見率：$-a(1-b)^n\log(1-b)$

エラー1個当たりの発見率：$-\log(1-b)$

本モデルの詳細は，山田/尾崎の論文[14]を参照されたい．

S字型モデル

$$\begin{bmatrix}y_n\\ x_n\end{bmatrix}=\begin{bmatrix}1 & 1-\{1-\log(1-b)^n\}(1-b)^n\\ 0 & \{1-\log(1-b)^n\}(1-b)^n\end{bmatrix}\begin{bmatrix}0\\ a\end{bmatrix}$$

発見率：$an(1-b)^n\{\log(1-b)\}^2$

エラー1個当たりの発見率：$\dfrac{n\{\log(1-b)\}^2}{1-n\cdot\log(1-b)}$

本モデル式の導出方法は，付録Bで説明する．また，パラメータの推定方法は付録Cで説明する．

（3） 込山-砂塚モデル

指数型/S字型モデルの両方の性質を含む著者らのモデル[9]を離散型で記述してみる．

$$\begin{bmatrix}y_n\\ x_n\end{bmatrix}=\begin{bmatrix}1 & 1-\left\{1+\dfrac{-\log(1-b)^n}{1-\log(1-b)^m}\right\}(1-b)^n\\ 0 & \left\{1+\dfrac{-\log(1-b)^n}{1-\log(1-b)^m}\right\}(1-b)^n\end{bmatrix}\begin{bmatrix}0\\ a\end{bmatrix}$$

ここで，mは初期習熟度補正パラメータ($m\geqq 0$)である．

発見率：$a(1-b)^n\{\log(1-b)\}^2\dfrac{n+m}{1-m\cdot\log(1-b)}$

エラー1個当たりの発見率：

$$\frac{(n+m)\{\log(1-b)\}^2}{1-(n+m)\log(1-b)}$$

本モデルの導出方法は，付録Dで説明する．なお，$m=0$のときS字型モデルとなり，$m=\infty$のとき指数型モデルとなる．

（4） 活用例

行列を用いた離散型モデルにおける最大の特長として，途中でパラメータの値が変化しても耐えられる点があげられる．

例えば，次のようなことは現場ではよく起こっている．検査の途中で投入労力が変わって，エラーの発見率が変化したとか，一通り検査データを流し終わったがエラーの摘出が不十分なため，さらにデータを追加

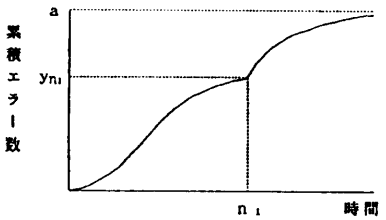

図 4 パラメータの値が変化した場合の累積エラー曲線
Fig. 4 Cumulative error curve in the case of parameter change at time n_1.

して検査を継続したなどである（**図 4** 参照）．この場合，パラメータばかりでなく，モデル自体が変わることさえ考えられる．

従来の連続型のモデルにおいては，このような場合適合度が著しく低下する．これは，すべてのデータを使って1つのモデルに当てはめているためである．

本モデルにおいては，パラメータが変化した（つまりテスト環境が変化した）時点 n_1 の前までの値を固定して，n_1 以降は，新たなパラメータあるいはモデルで推定を続ける．すなわち，以下のようにすればよい．

$$\begin{bmatrix} y_n \\ x_n \end{bmatrix} = \begin{bmatrix} 1 & f(n-n_1;\ \boldsymbol{b}_2) \\ 0 & 1-f(n-n_1;\ \boldsymbol{b}_2) \end{bmatrix} \begin{bmatrix} 1 & f(n_1;\ \boldsymbol{b}_1) \\ 0 & 1-f(n_1;\ \boldsymbol{b}_1) \end{bmatrix} \begin{bmatrix} 0 \\ a \end{bmatrix}$$

$$= \begin{bmatrix} 1 & f(n-n_1;\ \boldsymbol{b}_2) \\ 0 & 1-f(n-n_1;\ \boldsymbol{b}_2) \end{bmatrix} \begin{bmatrix} y_{n_1} \\ a-y_{n_1} \end{bmatrix}$$

ただし，$n \geqq n_1$ である．

このようにすれば，時点 n_1 以降のパラメータを推定するのに用いるデータ数は減少するという難点はあるが，打点の追従能力は向上すると考えられる．

さらに優れている点は，上式において行列は部品とみなせるため，モデル式自体が変化した行列に置き換えることも可能である．例えば，時点 n_1 まではS字型モデルであったが，n_1 以降は指数型モデルに変わった（f が変化した）などである．

5. 計量管理方法の提案

ソフトウェア開発における品質改善のプロセスを残存（含有）エラー数の推移で扱ってきた．この方法を有効に活用するためには，過去の実績データに基づいた目標/実績の管理が必要である．

（1） 改善目標，評価基準の設定

最終的な目標を「リリース時点における残存エラー数を最小にすること」とする．そのためには，検査時のエラー摘出数を増やすことである．また，検査のための努力が同じであれば，最終的な残存エラー数は検査開始時の含有エラー数に依存する．したがって，開発時に作り込むエラーをなるべく少なくし，レビューで摘出するエラーは逆に多くする．

評価基準は，開発プロセスにおいては「検査開始時の含有エラー数」とする．実際には，単位規模あたりのエラー数（件/KL）で表す．また，「開発日数」を短縮する（「開発工数」を削減する）ことも重要であるが，これは今後の課題とする．次に，検査プロセスにおいては「リリース時の残存エラー数」で評価する．また，「リリースまでの日数」も考慮する．

（2） 目標管理水準の設定

開発・検査の各フェーズにおいてエラーの作り込み数/摘出数を管理する．ここでは，当社の基本ソフトウェア開発本部をはじめとして多くの部門で活用している「品質会計」制度を利用する．「品質会計」では，摘出目標を負債とし，摘出実績を返済として，負債が0になれば次工程に進められると考えている．実際の作り込み数は計画には現れないが，推定する必要がある．これには，工程別エラーの作り込み，摘出のマトリクスが有効である（**表 2** 参照）．

（3） 開発プロセスの改善

改善のためのアプローチとして，①エラーの作り込み数を減らす，②摘出数を増やすの2つが考えられる．

要するにモデル内のパラメータの値を改善の方向に変化させるのである．3章の例を用いれば，作り込み数を20%削減し，摘出率を0.6から0.7に向上させるだけで，次工程への持越しエラー数が12から7へと40%も減っている．ただ，どの工程も一様に改善すると多大な工数が必要になるため，①，②に対する努力のバランスを検討する．以下，例に基づいて順に検討を進め，最後に理論的に裏付ける．

表 2 エラー管理表（例）
Table 2 Sample matrix for error making and detecting.

		摘出工程			
		概設	詳設	…	合計
作り込み工程	概	22/30	5/ 8	…	30
	詳	×	27/40	…	40
	⋮	×	×	…	⋮
	計（残）	22 (8)	32 (16)	…	100
エラー発見率		22/30 =0.73	32/48 =0.67	…	

表 3 パラメータの変化と効果との関係
Table 3 Relationship between parameter change and effects for error reduction.

	基準	ケース1	ケース2	ケース3	ケース4	ケース5
総数	100	100	100	80	80	70
摘出 q	0.6	0.7	0.8	0.7	0.8	0.7
残存数	12	9	6	7.2	4.8	6.3
効果(%)	—	25	50	40	60	48

(注1) 作り込み比率は $p=0.3$ とした．これは条件を同じにするためである．すなわち，このフェーズの比率を変えると他のフェーズに影響するためである．
(注2) 効果(%)とは，基準の例のエラー残存数に対する各ケースの削減率を表す．

表 4 波及エラー率とエラー発見率との相乗効果
Table 4 Interaction of spread rate of error and error detection rate to the number of errors.

		基準	ケース1	ケース2	ケース3	ケース4	ケース5
波及 u_i		0	1/3	1/3	1/3	1/3	1*
発見率	q_1	0.6	0.6	0.6	0.7	0.8	0.7
	q_2	0.6	0.6	0.7	0.7	0.7	0.7
	q_3	0.6	0.6	0.8	0.7	0.6	0.7
工程1	総数	30	30	30	30	30	30
	残存	12	12	12	9	6	9
2	総数	60.0	64.0	64.0	63.0	62.0	69.0
	残存	16.8	18.4	13.8	12.6	11.4	14.4
3	総数	100.0	112.8	111.3	109.2	107.1	129.4
	残存	22.7	26.9	12.2	17.6	22.6	22.4

(注1) エラー総数の初期値はすべて100とし，エラー作り込み比率は，$p_1=p_2=0.3$, $p_3=0.4$ とした．
(注2) *波及エラー率 $u_i=1$ というのは，例えば残存エラーの3分の2が2.5倍のエラーになるような場合である $(u_i=(1/3)\times1+(2/3)\times2.5-1)$.

まず，対策と効果との関係を数例計算してみると**表3**のようになる．

この表から，エラー発見率の向上がいかに重要であるかがわかる．つまり，効果的なレビューが必要であるということである．

次に波及エラーの影響とその対策について検討する．いくつか条件を変えて，総エラー数，残存エラー数の推移を**表4**に示す．

この表から，エラー総数を小さくするには，上工程において発見率を高めることが重要であることがわかる．また，残存エラー数を小さくするには，上工程はもとより，とくにエラー数が増えてきてからの発見数が決め手となる（ケース2）．

いくつかの制約条件を付けて，論理的にアプローチしてみる．制約条件として，次の2点をおく．

① 工程数を3 $(i=1\sim3)$ とする．すなわち，概要設計，詳細設計，製造の3工程で考えることにする．

② 波及エラーは考えないことにする．

これらから，検査開始時の含有エラー数は次のようになる．

$$\begin{bmatrix} y_{3,2} \\ x_{3,2} \end{bmatrix} = \boldsymbol{Q}_3\cdot\boldsymbol{P}_3\cdot\boldsymbol{Q}_2\cdot\boldsymbol{P}_2\cdot\boldsymbol{Q}_1\cdot\boldsymbol{P}_1\begin{bmatrix} y_0 \\ x_0 \end{bmatrix}$$

$$= \begin{bmatrix} [(p_1+p_2+p_3)+[\{p_1(1-q_1)+p_2\}(1-q_2)+p_3](1-q_3)]x_0 \\ [\{p_1(1-q_1)+p_2\}(1-q_2)+p_3](1-q_3)x_0 \end{bmatrix}$$

$$\therefore\quad x_{3,2}=[\{p_1(1-q_1)+p_2\}(1-q_2)+p_3](1-q_3)x_0$$

このとき，$x_{3,2}$ を最小にする作り込み比率 p，発見率 q は次のようになる．

p_1, p_2, p_3：任意

q_1, q_2：任意　　q_3：$=1$

すなわち，「過程でどのようにエラーを作り込み，摘出しようとも，最後の製造工程でのレビューですべてのエラーを発見すればよい」ということになる．しかしながら，これは現実的な解とはいえないので，さらに条件を加える．

③ エラー作り込み比率を指定する．ここでは，直前の例と同じ $p_1=p_2=0.3$, $p_3=0.4$ を用いる．

④ エラー発見率の値の範囲を指定する．ここでは，$0.6\leqq q_1, q_2, q_3\leqq0.8$ を用いる．

⑤ 総レビュー量に制限を付ける．④の制約だけならすべて0.8にすればよい．ここでは，$q_1+q_2+q_3=3\bar{q}$ $(0.6\leqq\bar{q}\leqq0.8)$ とし，どの工程のレビューを重視すべきかを検討する．

ここで再び $x_{3,2}$ を見直すと，

$$x_{3,2}=[\{p_1(1-q_1)+p_2\}(1-q_2)+p_3](1-q_3)x_0$$
$$=\{0.4+0.3(2-q_1)(1-q_2)\}(1-q_3)x_0$$

この値を小さくするには，$(2-q_1)(1-q_2)$, $(1-q_3)$ 共に小さくしなければならない．条件⑤から $q_1<q_2$ となる．さらに q_3 を含めた最適解は，**図5**から次のようになる．

$q_1<q_2<q_3$

ここでは，エラー修正のコストを考慮していないため，上工程が軽視されているように見えなくもないが，件数最小という評価関数のもとでは，条件③～⑤を加えたことにより，かなり現実的な解が得られたも

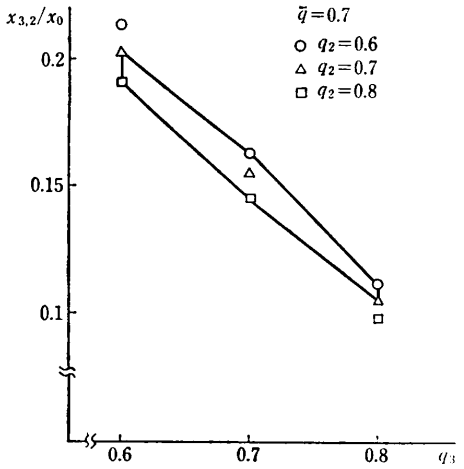

図 5 エラー発見率と残存エラー率との関係
Fig. 5 Relationship between error detection rate and residual error rate.

表 5 発見率とリリース時期との関係
Table 5 Relationship between error detection rate and delivery date.

	基準	ケース1	ケース2	ケース3	ケース4	ケース5
総エラー a	100	100	100	100	80	40
発見率 b	0.2	0.25	0.3	0.35	0.2	0.2
リリース日	22	17	14	12	22	22

(注1) S字型モデルを用い，最後までパラメータの値は一定とする．
(注2) リリース日とは検査開始からリリースまでの日数を表す．

のと信じる．

(4) 検査プロセスの改善

改善のためのアプローチとして，やはりエラー発見率を向上させることが考えられる．ここでは，発見率の変化とリリース時期の短縮との関係について評価する．ただし，リリースは発見エラーが95%を越えた時点とする．

また，検査開始時における残存エラー数もリリース時期に影響すると思われるので，合せて評価する．

表5から，エラー発見率がいかに重要であるかがわかる．発見率が2倍になると，リリースまでの日数は2分の1に短縮される．エラー発見率とリリースまでの日数とは反比例するのである．

また，この日数は総エラー数によって影響は受けないが，リリース時の残存エラー数は総エラー数に比例する．したがって，基準の残存数は5で，ケース4は4で，ケース5は2である．もし，残存数を2にしようとすると，基準は26日かかり，ケース4は25日かかる．このように検査開始時における残存エラー数の大きさは，絶対品質（規模当たりのエラー数）で評価すると大きな影響を与える．

(5) 最適施策

以上を総合すると，次の手順によって品質改善を進めるとよい．

① 現状把握

過去の実績データに基づいて，各工程のエラー数およびエラー発見率を算出する．

② 目標品質水準の設定

各工程での目標発見エラー数を設定する（含検査の目標）．

③ 開発プロセスの改善

目標達成のためのエラー発見率を算出し，品質向上施策と適用のレベルを検討する．最適施策としては，

1) 作り込むエラー総数を押えるためには，開発技術の水準を向上させ，経験を積むことである．高級な技法の使用経験を積み，作業の標準化を図り，チェックポイントを押える必要がある．また，波及エラーの影響を小さくするためには，上工程において発見率を高めることである．
2) 検査開始時の含有エラー数を少なくするためには，レビュー技術を向上させ，エラー発見率を大きくすることが重要である．SQMAT[1),12),16)]などの技術が有効である．また，レビューの工数が有限であることを考慮すれば，累積エラー数が増すにしたがって，すなわち工程が進むにしたがってエラー発見率を大きくしていくことが得策である．

④ 検査プロセスの改善

ここではエラー発見率を向上させるほかにない．施策として，もれの少ないテスト項目の作成，テスト技術の修得・活用などが考えられる．管理方法としては，まずリリース予定日から逆算して，エラー発見率と予測の累積エラー曲線を求める．次に，実績値を打点し，遅れが出るようであれば，回復に必要なエラー発見率を算出するとともに改善策を講ずる．

⑤ 経験の蓄積とフィードバック

今回のプロジェクトで得られた経験，例えば施策と効果の関係（表1参照），工程別エラー数と発見率（表2参照）などを次のプロジェクトでの改善に生かす．

6. おわりに

ソフトウェア開発における品質改善プロセスを記述するのに，2×2 の行列表現が有効であることを示した．開発プロセスではエラーの作り込み/摘出のメカニズムが記述でき，検査プロセスでは複数の信頼度成長モデルが記述できる．同時に，残存エラー数を最小にする最適施策についても提案した．

今後は，評価関数に開発工数やエラー修正コストも加えて，より現状のプロジェクト管理に近い施策を考えていくつもりである．

謝辞 本研究を進めるに当たっては，同部の込山俊博氏にレビューをはじめ，多大なご協力をいただきました．この場を借りてお礼申し上げます．

参考文献

1) 東，砂塚：ソフトウェア品質計測/保証技術(SQMAT)，品質，Vol. 16, No. 1, pp. 79-84 (1986).
2) Basili, V. and Selby, R. Jr.: Calculation and Use of an Environment's Characteristic Software Metric Set, *8th ICSE*, pp. 386-391 (1985).
3) Fujino, K.: Company-wide Software Quality Control (SWQC), *ICQC '87*, pp. 459-464 (1987).
4) Gilb, T.: *Principles of Software Engineering Management*, Addison-Wesley (1988).
5) Goel, A. and Okumoto, K.: Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, *IEEE Trans. Reliability*, Vol. R-28, No. 3, pp. 206-211 (1979).
6) Humphrey, W.: Characterizing the Software Process: A Maturity Framework, *IEEE Softw.*, pp. 73-79 (Mar. 1988).
7) Jones, C.: *Programming Productivity*, McGraw-Hill (1986).
8) Jones, C.: *Measuring Software Quality—A Survey of the State of the Art*, SPR (1989).
9) 込山，砂塚：テストの初期習熟度を考慮したソフトウェア信頼度成長モデル，ソフトウェア工学研究会報告，64-2 (1989).
10) Mizuno, Y.: Software Quality Improvement, *Computer*, Vol. 16, No. 3, pp. 66-72 (1983).
11) Musa, J., Iannino, A. and Okumoto, K.: *Software Reliability*, McGraw-Hill (1987).
12) Sunazuka, T., Azuma, M. et al.: Software Quality Assessment Technology, *8th ICSE*, pp. 142-148 (1985).
13) Sunazuka, T. and Basili, V.: Integrating Automated Support for a Software Management Cycle into the TAME System, UMIACS-TR-89-75, Univ. of Maryland (1989).
14) 山田，尾崎，成久：テストラン試行回数を考慮したソフトウェア信頼度成長モデルに関する考察，ソフトウェア工学研究会報告，31-1 (1983).
15) 山田：ソフトウェア信頼性評価技術，HBJ 出版局 (1989).
16) 吉澤，東，片山(編)：ソフトウェアの品質管理と生産技術，日本規格協会，pp. 15-41 (1988).
17) 国友：プログラムの品質管理(2)，*bit*, Vol. 14, No. 6, pp. 90-97 (1982).
18) 松坂：線型代数入門，岩波書店 (1980).
19) 森口(編)：ソフトウェア品質管理ガイドブック，日本規格協会，第17章 (1990).
20) 藤野，松尾谷：デバグ工学における精度配分問題，平成元年電気・情報関連学会連合大会 (1989).
21) 松尾谷：エラーのライフサイクル・モデル，ソフトウェア工学研究会報告，71-10 (1990).

付　録

付録A　信頼度成長モデルの行列の性質

$$\begin{bmatrix} 1 & f(n\,;\,\boldsymbol{b}) \\ 0 & 1-f(n\,;\,\boldsymbol{b}) \end{bmatrix} = \boldsymbol{F}_n$$

① 和を一定に保つ

$$[y', x']' = \boldsymbol{F}_n [y, x]'$$

このとき

$$y' = y + f(n\,;\,\boldsymbol{b}) \cdot x$$
$$x' = x - f(n\,;\,\boldsymbol{b}) \cdot x$$

となり，

$$y' + x' = y + x$$

である．

② 積がきれいに整理できる．

(1) $f(n\,;\,\boldsymbol{b}) = C$（定数）のとき

$$\boldsymbol{F}_2 \boldsymbol{F}_1 = \begin{bmatrix} 1 & 1-(1-C)^2 \\ 0 & (1-C)^2 \end{bmatrix}$$
$$\vdots$$
$$\boldsymbol{F}_n \cdots \boldsymbol{F}_2 \boldsymbol{F}_1 = \begin{bmatrix} 1 & 1-(1-C)^n \\ 0 & (1-C)^n \end{bmatrix}$$

となり，指数型モデルに収束した．

(2) $f(n\,;\,\boldsymbol{b})$ が定数でないとき

$f(n\,;\,\boldsymbol{b}) = f_n$ と記述する．

$$\boldsymbol{F}_2 \boldsymbol{F}_1 = \begin{bmatrix} 1 & 1-(1-f_1)(1-f_2) \\ 0 & (1-f_1)(1-f_2) \end{bmatrix}$$
$$\vdots$$
$$\boldsymbol{F}_n \cdots \boldsymbol{F}_2 \boldsymbol{F}_1 = \begin{bmatrix} 1 & 1-(1-f_1)(1-f_2)\cdots(1-f_n) \\ 0 & (1-f_1)(1-f_2)\cdots(1-f_n) \end{bmatrix}$$

と整理できる．

付録B　S字型モデルの導出方法

基本的には連続型モデルと離散型モデルを比較し，変数を変換することによって対応づける．

指数型モデルによって対応関係を導く.

累積エラー数:

$$m(n)=a\{1-(1-b)^n\}$$

とおく.

発見率:

$$m'(n)=-a(1-b)^n\log(1-b)$$

エラー1個当たりの発見率:

$$\frac{m'(n)}{a-m(n)}=-\log(1-b)$$

連続型のそれと比較すると，連続型の b が離散型の $-\log(1-b)$ に対応していることがわかる.

この結果に基づいて，(遅延) S字型モデルの式を変換する.

$$M(t)=a\{1-(1+bt)e^{-bt}\}$$

ここで,

$$b \rightarrow -\log(1-b),\quad t \rightarrow n$$

と変換すると,

$$M(n)=a[1-\{1-\log(1-b)^n\}(1-b)^n]$$

が得られる.

付録C　S字型モデルのパラメータの推定方法

$$M(n)=a[1-\{1-\log(1-b)^n\}(1-b)^n]$$

とし，データ群を $(n_1, x_1), (n_2, x_2), \cdots, (n_k, x_k)$ とする. n_k は検査時点，x_k は摘出エラー数である.

$$L=\exp\{-M(n_k)\}\cdot\prod_{j=1}^{k}\frac{\{M(n_j)-M(n_{j-1})\}^{x_j-x_{j-1}}}{(x_j-x_{j-1})!}$$

とすると，尤度関数は次のようになる.

$$\begin{aligned}\ln L&=-M(n_k)+\sum_{j=1}^{k}[(x_j-x_{j-1})\cdot\ln\{M(n_j)-M(n_{j-1})\}-\ln(x_j-x_{j-1})!]\\&=-a[1-\{1-\ln(1-b)^{n_k}\}(1-b)^{n_k}]+\sum_{j=1}^{k}[(x_j-x_{j-1})\cdot\ln\{\{1-\ln(1-b)^{n_{j-1}}\}(1-b)^{n_{j-1}}-\{1-\ln(1-b)^{n_j}\}(1-b)^{n_j}\}-\ln(x_j-x_{j-1})!]\end{aligned}$$

$\frac{\partial \ln L}{\partial a}=0$ より

$$\frac{x_k}{a}[=1-\{1-\ln(1-b)^{n_k}\}(1-b)^{n_k}]$$

$\frac{\partial \ln L}{\partial b}=0$ より

$$an_k(1-b)^{n_k-1}\ln(1-b)^{n_k}=\sum_{j=1}^{k}[\{(x_j-x_{j-1})\{n_{j-1}(1-b)^{n_{j-1}-1}\ln(1-b)^{n_{j-1}}-n_j(1-b)^{n_j-1}\ln(1-b)^{n_j}\}\}/\{\{1-\ln(1-b)^{n_{j-1}}\}(1-b)^{n_{j-1}}-\{1-\ln(1-b)^{n_j}\}(1-b)^{n_j}\}]$$

以上の連立方程式を数値的に解くことでパラメータ a, b の値が得られる.

付録D　込山-砂塚モデルの導出方法

付録B．S字型モデルの導出方法と同様にする.

$$b \rightarrow -\log(1-b),\quad t \rightarrow n$$

という変換を行えば,

$$M(t)=a[1-\{1+bt/(1+bm)\}e^{-bt}]$$

から

$$M(n)=a\left[1-\left\{1+\frac{-\log(1-b)^n}{1-\log(1-b)^m}\right\}(1-b)^n\right]$$

が，容易に導かれる.

(平成元年8月31日受付)
(平成2年4月17日採録)

砂塚　利彦（正会員）

1957年生．1980年早稲田大学理工学部工業経営学科卒業．1982年同大学院修士課程修了．同年日本電気(株)入社．以来，ソフトウェアの品質管理／品質保証技術の研究開発に従事．1987年8月より1年間メリーランド大学計算機科学科客員研究員．現在，ソフトウェア生産技術開発本部管理システム開発部主任．日本品質管理学会，IEEE各会員.