ソフトウェア工学 77−23
（１９９１. 2. 8）

ソフトウェア開発における進捗管理のためのモデルについて

高木千春 楠本真二 松本健一 菊野亨 鳥居宏次
大阪大学

概要

ソフトウェア開発管理の一つに工程管理がある．このうちテストの進捗管理では，ソフトウェア信頼度成長モデル（ＳＲＧＭ）などこれまでに多くのモデルが提案されている．しかし，プログラム作成の進捗管理の研究はあまり行われていない．本研究ではＳＲＧＭを応用したプログラム作成の進捗管理を考える．ＳＲＧＭでは，テスト時刻ｔまでに除去されるエラーの累積数を用いてエラーの除去される過程をモデル化している．これをプログラム作成過程に応用し，エラーの累積数の代わりにプログラム作成時刻ｔまでに行われる作業の累積値を用いてプログラム作成過程をモデル化する．モデル化においては遅延Ｓ字型ＳＲＧＭの考え方を用いている．

A Model for Progress Management of Software Development

Chiharu TAKAKI, Shinji KUSUMOTO, Ken-ichi MATSUMOTO
Tohru KIKUNO and Koji TORII
OSAKA University

Abstract

One major subject of software development management is progress management. Several models for testing progress management such as SRGM's have been proposed so far, but few studies on progress management model for implementation phase. In this paper, we discuss progress management on implementation phase using SRGM's. SRGM's modeled testing process using cumulative number of faults removed by testing time t. Applying this to implementation phase, we modeled it using cumulative quantity of work done by implementation time t instead of cumulative number of faults. We also discuss progress management on implementation phase using the concept of Delayed S-shaped Reliability Growth Model.

1 はじめに

ソフトウェアの大規模化と開発過程の複雑化にともなって, 開発計画の作成と計画に基づく開発過程の制御, すなわち開発管理の必要性が高くなってきている[4]. ソフトウェアの開発管理においては, 各開発工程における品質, コスト, および進捗が管理の対象である.

開発管理を効率良く行うためには, 管理対象をモデル化し, 管理対象を定量的, かつ客観的に評価することが重要である. しかし, 設計工程やプログラム作成工程 (コーディング工程) における開発管理モデルはほとんど提案されていない.

一方, テスト工程における開発管理モデルはいくつか提案されている. ソフトウェア信頼度成長モデル (Software Reliability Growth Model, 以下ではＳＲＧＭと略す) はテスト工程における品質管理のための代表的なモデルの一つである. ＳＲＧＭを利用すると, テスト時刻 t までに発見され, 除去されたフォールト数に基づいて, ソフトウェア中に残存するフォールト数を推定することができる. しかし, 発見され, 除去されたフォールト数がテスト工程における開発者の作業量を表すと仮定すると, ＳＲＧＭは単に品質管理のためのモデルではなく, テスト作業の進捗管理のためのモデルと考えることができる. しかもＳＲＧＭの持つ仮定に反しないように作業量を定義することができれば, テスト工程以外の工程の進捗管理に利用可能である.

本稿では, ＳＲＧＭをテスト工程における進捗管理のためのモデルとしてとらえる. その上で, プログラム作成の作業量を定義し, プログラム作成工程における進捗管理のためのモデルを提案する. 提案するモデルのもととなるのは, ＳＲＧＭの一つであり性能が良いとされている遅延Ｓ字型ＳＲＧＭ (Delayed S-Shaped SRGM) である. また, プログラム作成工程における作業量はプログラムコードの行数に基づいて表す.

提案するモデルの妥当性は大学の学生実験において収集したデータに基づいて議論する. 更に, 提案するモデルを利用すると, プログラム作成工程の早い段階でプログラム作成に要する総作業量を予測できる可能性のあることを示す.

2 ソフトウェア開発管理

ソフトウェア開発管理 (Software management) とは, ソフトウェア開発における経済的要求, 時間的要求, 品質要求を開発計画として具体化し, その計画に従ってソフトウェア開発の過程を最適に制御することである[1]. 進捗管理は工程管理の主要な管理内容の一つであり, 各開発工程の進み具合を管理するものである[1]. 進捗管理では, 予め作成された計画と実績との比較を行う. そして, 計画と実績の差が一定限度を越えた場合, 原因を究明し対策をたてる必要がある. 従って進捗管理では, ソフトウェア開発過程の早い時期に進捗の度合を知り, 将来の予測を行うことが重要である.

3 ソフトウェア信頼度成長モデル

3.1 定義

ＳＲＧＭは, テスト工程において, ソフトウエアの品質 (特に無欠陥性) を管理するためのモデルである.

ＳＲＧＭでは次の１., ２. を仮定する.

1. テスト段階において発見されたソフトウェア中のフォールトは全て修正され, 除去される.
2. テスト段階 (のフォールト修正, 除去の過程) において, 新たなフォールトがソフトウェア中に作り込まれることはない. 従って, テストの進行にともない, ソフトウェア中のフォールトは必ず少なくなる.

テスト段階において発見された総フォールト数を確率量としてモデル化し, その確率過程に対してNHPP を仮定する. このとき, テスト開始時からテスト時刻 t までに発見される総フォールト数 $N(t)$ は次式で表される[4],[5].

$$P_r\{N(t)=n\} = \frac{\{H(t)\}^n}{n!} \times \exp[-H(t)] \quad (n=0,1,2,\cdots) \tag{1}$$

$$H(t) = \int_0^t h(t)dx \qquad t \geq 0 \tag{2}$$

$H(t)$ をソフトウェア信頼度成長関数 (software reliability growth function), あるいは, ソフトウェア信頼度成長曲線 (software reliability growth

curve) と呼ぶ．また，$h(t)$ は時刻 t における単位時間当りの発見フォールト数の期待値を表す．

式 (1)，(2) より，テスト開始時にソフトウェア中に存在していたフォールト数（初期フォールト数）N_0 の期待値は次式で表される．

$$E(N_0) = H(\infty) \tag{3}$$

従って，ソフトウェアの信頼性を評価する尺度の1つであるソフトウェア中の時刻 t における残存フォールト数の期待値は次式で表される．

$$H(\infty) - H(t) \tag{4}$$

3.2 遅延S字型SRGM

SRGMは，ソフトウェア信頼度成長曲線 $H(t)$ の形状によって，(1) 指数型SRGMと (2) S字型SRGMに分類される[6],[7]．本研究では性能が良いとされる遅延S字型SRGMに注目する[3]．

遅延S字型SRGMは，故障の発生時刻からその故障の原因であるフォールトを除去する時刻までの時間的な遅れを考慮している[7]．

今，テスト時刻 t における単位時間当りの発生故障数の期待値 $h_f(t)$ は，その時刻のソフトウェア中の残存フォールト数に比例すると仮定する．このとき $h_f(t)$ は次式で表現される[7]．

$$h_f(t) = \frac{dH_f(t)}{dt} = \phi_f\{N - H_f(t)\} \tag{5}$$

ここで，N は初期フォールト数，$H_f(t)$ はテスト時刻 t までに発生する総故障数の期待値，ϕ_f は定常状態（すなわち，$t \to \infty$ としたとき）のフォールト1個当りの故障発生率をそれぞれ表す．

更に，テスト時刻 t における単位時間当りの発見フォールト数の期待値 $h(t)$ は，その時刻において故障の原因となっているがまだ発見されていないフォールト数に比例すると仮定する．このとき $h(t)$ は次式で表現される[7]．

$$h(t) = \frac{dH(t)}{dt} = \phi\{H_f(t) - H(t)\} \tag{6}$$

ここで，$H(t)$ はテスト時刻 t までに除去される総フォールト数の期待値，ϕ は定常状態（すなわち，$t \to \infty$ としたとき）のフォールト1個当りのフォールト発見率をそれぞれ表す．近似的に $\phi_f = \phi$ として，式 (5) および式 (6) を解くことにより次式を得る．

$$H(t) = N1 - (1 + \phi t)\exp^{-\phi t} \tag{7}$$

4 プログラム作成工程の進捗管理モデル

4.1 SRGMによる進捗管理

3.1で述べたようにSRGMはテスト工程における品質管理モデルである．しかしテスト作業とは，ソフトウェア中の残存フォールトを発見し，除去する作業であるとすると，発見され，除去されたフォールトの累積数はテスト作業量を表すことになる．そして，テスト開始時にソフトウェア中に存在していた総フォールト数と，発見され，除去されたフォールトの累積数を比較すれば，テスト作業の進み具合を知ることができる．従ってSRGMは品質管理モデルであると同時にテスト作業の進捗を表すモデルと考えることができる．

4.2 累積作業量

プログラム作成工程では仕様を満足するプログラムを作成するために (1) プログラムを新しく作成する作業（作成作業）と (2) プログラムを修正・変更する作業（修正作業）が行われる．ここでは，それぞれの作業の結果として作成されたプログラムコードの行数によってそれぞれの作業量が表されるものとし，それらの累積値の和を累積作業量 $S(t)$ と呼び，次のように定義する．

$$S(t) = \int_0^t C(x) + M(x)dx \tag{8}$$

ただし，$C(t)$ は時刻 t において新しく作成されたプログラムコードの行数，$M(t)$ は修正・変更作業によって時刻 t に作成されたプログラムコードの行数を表す．

4.3 モデルの導出

$S(t)$ は t に関して単調増加関数である．また $S(t)$ はプログラムが仕様を満足した時点である値 S をとり，それ以降変化しない．このことはテスト時刻 t までに発見され，除去されたフォールトの累積数，すなわち，SRGMの $H(t)$ は単調増加であり，テストが十分行われると $H(t)$ はテスト開始時にプログラム中に存在したフォールト数 N と等しくなることと同じである．したがってSRGMの $H(t)$ の代わりに $S(t)$ を使用したモデルによってプログラム作成工程における進捗管理が可能になると考える．

プログラム作成過程において実際にプログラムの作成や修正を行うためには，まず，作成・修正すべき機能や，作成・修正箇所を明確にしなければならない．

この考え方は3.2で述べた遅延S字型SRGMにおける故障の発生からその故障の原因であるフォールトの除去までに時間的な遅れが存在するという考え方に良く似ている．従って，ここでは遅延S字型SRGMの定義式 (7) に基づいて次式で表される新しいモデルを考案する．

$$S(t) = S_0(1 - (1 + \phi t)\exp^{-\phi t}) \qquad (9)$$

ここで S_0 は仕様を満足するプログラムを作成するために必要な総累積作業量，ϕ は仕事率をそれぞれ表す．

5 モデルの適用実験

5.1 実験概要

大阪大学基礎工学部情報工学科2年次の学生約40人の学生実験から得られたデータに対してモデルの適用実験を行った．この学生実験において学生の作成するプログラムは一種の在庫管理問題で，C言語を用いて作成することになっている．

データの収集はツールによって自動的に行われ，学生がプログラムを作成する期間中に行った全変更作業を，エディットの度にプログラム中の変更された箇所のみを抽出した差分リストの形で蓄積する[2]．また，プログラム開発に要した作業時間は端末利用時間で測定した．

データを収集した学生のうち，(1) 明らかに他人のプログラムを参照しているもの，(2) 指定された計算機以外の計算機を使用してプログラムを作成しているもの，(3) 締切までにプログラムが完成しなかったもの，(4) 差分リストの数が少な過ぎるもの (差分リスト数20以下)，を除いた結果，15名のデータを使用することになった．

各学生のデータの値を表1に示す．表1で，端末使用時間の欄は各学生がテスト開始までに使用した端末時間の合計を表す．累積作業量の欄の下のテスト前の欄はテスト開始までの累積作業量を，テスト中の欄はテスト期間中の累積作業量を，そして最終値の欄はプログラム完成時の累積作業量をそれぞれ表す．

表1 実験データ1

学生	端末使用時間(分)	累積作業量		
		テスト前	テスト中	最終値
#1	662	529	118	647
#2	1639	751	0	751
#3	431	476	51	527
#4	1432	687	90	777
#5	1106	608	0	608
#6	907	461	45	506
#7	1656	812	77	889
#8	819	715	101	816
#9	886	549	0	549
#10	1322	731	1	735
#11	1144	583	139	722
#12	588	345	0	345
#13	1980	788	56	844
#14	751	472	34	506
#15	942	517	56	573

表2 実験データ2

端末使用時間	作成作業量	修正作業量	累積作業量
0	0	0	0
10	5	0	5
14	5	1	6
34	13	1	14
49	13	13	26
54	13	19	32
58	14	20	34
85	17	28	45
102	17	51	68
152	17	82	99
177	18	93	111
222	20	93	113
264	54	100	154

5.2 実験データ

表2に，使用した実験データの一例を示す．ここには，ある学生が使用した端末時間とその時点までの累積作業量・修正作業量・累積作業量の値が記録されている．例えば，表2で上から6番目のデータは，この学生は5回目のエディットを行った時点での端末使用時間は54分で，その時点までの作成作業量，修正作業量，累積作業量はそれぞれ13行，19行，32行であることを示している．

5.3 分析

モデルが実際のプログラム作成過程に適合していることを確認するために，プログラムが完成した段階でモデルを適用し，モデルと実測データとの適合度を判定した．式(9)中のS_0とϕの推定は最尤推定法を用いて行った[6]．モデルと実測データとの適合度の検定には，Kolmogorov-Smirnov 検定を用いた．この結果，有意水準$\alpha = 1$ %で 15 名中 13 名のデータがモデルに適合した．

表 3 に推定結果を示す．適合度分析結果の欄は○がモデルに適合していることを，×がモデルに適合していないことを表す．推定誤差は次の式で表される．

$D = \frac{|S-S_0|}{S_0}$ ここでSはモデルによる最終累積作業量の推定値を，S_0は最終累積作業量の実測値をそれぞれ表す．

また，図1にモデルの適合例を示す．実線が累積作業量の実測値を表す．

6 モデルによる進捗管理

SRGMを用いたテストの進捗管理と同様に，このモデルを使用するとプログラム作成の早期の段階で最終累積作業量が予測できることから，モデルを使ってプログラム作成の進捗管理を行うことが考えられる．プログラム作成の進捗にともなう累積作業量の予測値の変化を図2に示す．この図からも分かるように，累積仕事量の予測値はプログラム作成過程の半ばで安定し，その予測値は実測値に十分近い．

最終累積作業量の予測を行う方法を述べる，まず差分リストが生成されるたびに累積作業量の最終値の予測を行う．予測値が安定した値を取り，かつ，モデルが実測データに適合するとき，この予測値は有効であるとみなす．具体的には，差分リストが生成されるたびに最尤推定法によって式(9)中のϕとS_0を推定する。次に，予測値が安定した値を取るということを，プログラム作成時刻t_iにおける累積作業量をS_iとすると，連続する6つの予測値$S_i,\cdots,S_{i+5}$が±5 %の誤差の範囲に収まるときと定める．これを式で表すと次のようになる．

$$\max\left(\frac{|s_k - s_n|}{s_n}\right) \leq 0.05 \qquad (n-5 \leq k \leq n-1)$$

次に,Kolmogorov-Smirnov 検定によってモデルと実測データとの適合度を判定する．モデルが実測データに適合しているとき，累積作業量の最終値の予測誤差を次の式で表す．

$D = \frac{|S-S_0|}{S_0}$

ここでSはモデルによる最終累積作業量の予測値を，S_0は最終累積作業量の実測値をそれぞれ表す．この結果，予測誤差Dは16 %以内に収まった．モデルを使った最終累積作業量の予測の例を図3に示す．実線は累積作業量の実測値を表す．

また，モデルによる最終累積作業量の予測の結果を表4に示す．この表にない＃3と＃5の学生は，モデルによる最終累積作業量が予測できなかった．この表の検定結果の欄で○はモデルに適合していることを×はモデルに適合していないことをそれぞれ表す．また予測時点での進捗度の欄は，最終累積作業量の予測が可能になった時点で，プログラム

表3 モデルを使った最終累積作業量の推定結果

学生	最終累積作業量			適合度検定結果
	実測値	推定値	誤差(%)	
#1	647	887	37	○
#2	751	913	22	○
#3	527	829	57	×
#4	777	805	4	○
#5	608	994	63	○
#6	506	513	1	○
#7	889	852	4	○
#8	816	946	16	○
#9	549	625	14	○
#10	735	837	14	○
#11	722	679	6	○
#12	345	372	8	○
#13	844	856	10	○
#14	506	529	5	○
#15	573	817	43	×

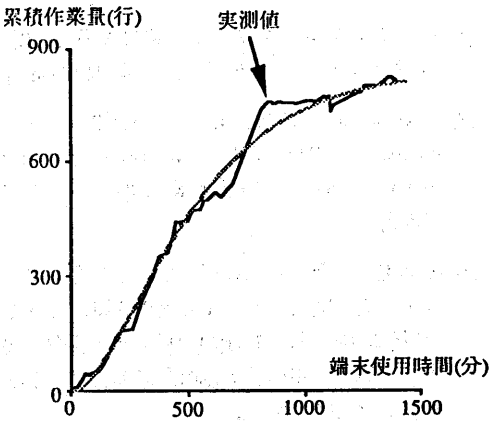

図1 モデル適合

作成の全作業工程の何%まで作業が進捗しているかを表す. 例えば#2の学生は全工程の81%の時点で最終累積作業量の予測ができ, 予測値(810行)と実測値(751行)との誤差は8%である.

7 おわりに

本研究では, ソフトウェア開発過程のうちプログラム作成工程のモデル化を行った. モデル化に当たって, SRGMの一つである遅延S字型SRGMの考え方を参考にし, プログラム作成作業の進捗状況を表す尺度として新しく累積作業量の概念を導入した. また, このモデルを大学の学生実験に適用した結果, このモデルがプログラム作成工程における進捗管理に使用できることが分かった.

参考文献

[1] 情報システムハンドブック編集委員会 (編): "情報システムハンドブック", 培風館 (1989).

[2] S.Kusumoto, K.Matsumoto, T.Kikuno and K.Torii: "GINGER: data collection and analysis system", 信学技報 SS90-5, pp.39-48 (1990).

[3] 松本健一, 菊野亨, 鳥居宏次: "S字型ソフトウェア信頼度成長モデルの大学環境における実験的評価 −推定精度の比較と習熟係数の決定−", 電子情報通信学会論文誌 D-I, J73-D-I, 2, pp.175-182 (1990).

[4] 宮本勲: "ソフトウェア・エンジニアリング:現状と展望", TBS出版会 (1982).

[5] J.D.Musa, A.Iannino and K.Okumoto: "Software Reliability: Measurement, Prediction, Application", McGraw-Hill (1987).

[6] M.Ohba: "Software reliability analysis models", IBM J.Res.Develop., 28, 4, pp.428-443 (1984).

[7] S.Yamada, M.Ohba and S.Osaki: "S-shaped reliability growth modeling for software error detection", IEEE Trans. Reliab., R-32, 5, pp.475-478 (1983).

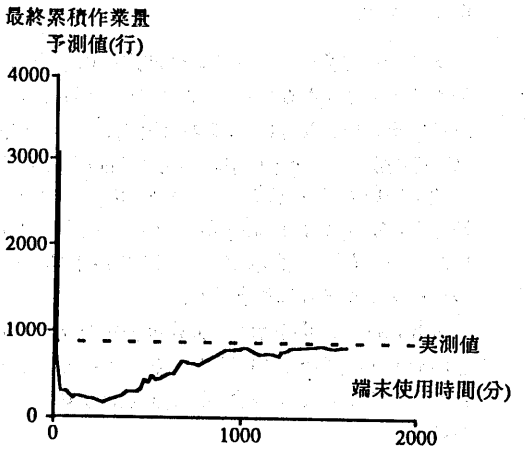

図2 モデルによる最終累積作業量の予測

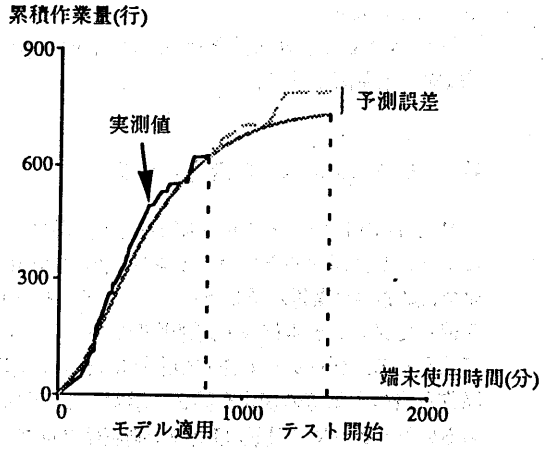

図3 モデルによる予測

表4 モデルを使った最終累積作業量の予測結果

学生	最終累積作業量		検定結果	予測時点での
	予測値	誤差(%)	$\alpha=1\%$	進捗度(%)
#1	333	46	×	48
#2	810	8	○	81
#4	242	69	×	30
#6	442	11	×	41
#7	851	4	○	51
#8	946	16	○	100
#9	622	12	○	90
#10	802	9	○	63
#11	428	41	×	45
#12	333	3	○	80
#13	717	15	○	44
#14	537	6	○	81
#15	799	39	○	98