解　説

ソフトウェア信頼度成長モデルの適用技術†

松本健一††　菊野　亨††　鳥居宏次††

1. まえがき

ソフトウェアの欠陥や故障が社会に大きな影響を与える今日，ソフトウェアの信頼性を定量的，かつ，客観的に評価する技術に大きな関心が集まっている[6),9),15),18),20)]．ソフトウェアのテスト工程，及び，運用段階におけるフォールト☆の発見，除去の過程をモデル化する理論的枠組みの一つとして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.) が知られている[3),8),17),23),24)]．

SRGM を実際のソフトウェア開発プロジェクトに適用するには次の二つの基本技術が必要となる．

(1) フォールトの発見，除去過程を表すデータの収集

(2) 収集したデータに基づくパラメータの推定

これらの基本技術は SRGM 上で導入された仮定，前提条件などと矛盾しないことが必要である．さらに，対象とするプロジェクトの実態にも適合していることが望ましい．

本稿では，こうした基本技術の現状を述べた後で，問題点を整理し，解決のための技術開発の可能性について議論する．ここでは，信頼性の評価に関する現状を次の三つの会社を例にして紹介する．

A社　モデルを用いた信頼性の定量的評価の重要性に気づき，“SRGM の初めての適用” を目指している．

B社　すでに SRGM 適用に関する実績をもっており，“SRGM のより充実した適用” を目指している．

C社　過去の実績の上に立って，プロジェクトの特性を考慮した “SRGM のより実用的な適用” を目指している．

容易に分かるように，これら3社における SRGM の各適用は時間の経過とともに，A社からB社へ，そしてB社からC社へと自然に移行可能である．

本稿では，将来の展望として，より高度な SRGM 適用のための応用技術についても述べる．

2. 三つの方式

2.1 シングル方式（A社）

A社では，現在，ソフトウェアの信頼性を定量的に評価するため SRGM をテスト工程で利用することを検討している．ソフトウェア開発においてモデルを用いた信頼性評価を行うのは初めてなので，既存の SRGM の一つを選び，適用することにした．

文献あるいは専門家の話などから，「アメリカでは指数型 SRGM がもてはやされているようだが，日本の企業のデータにはS字型 SRGM のほうがよく適合する」ことが分かったので，S字型 SRGM を選択した．

ここでは，ただ一つの SRGM を適用するという意味で，A社の SRGM の適用方法をシングル方式と呼ぶ．

2.2 マルチ方式（B社）

B社では，これまで，ある一つの SRGM を選び，幾つかのプロジェクトに適用するとともに，ツールの開発などデータ収集技術の確立に努めてきた．ただし，適用するプロジェクトの種類や数が増えるにともなって，実測データと適合しない事例も報告されてきた．

既存の SRGM で，あらゆるソフトウェア開発プロジェクトに適用可能で，かつ，高い推定精度を保証するモデルの存在しないことはよく知られている[15)]．B社の場合，OS から事務用のアプリケーションソフトウェアまで多くの種類のソフトウェアの開発を行っている．テスト方法もソフトウェアの種類や要求される

† Advanced Technologies for Practical Uses of Software Reliability Growth Models by Ken-ichi MATSUMOTO, Tohru KIKUNO and Koji TORII (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University).

†† 大阪大学基礎工学部情報工学科

☆ ソフトウェア開発において開発者がおかした誤り (error) がソフトウェア中に具体化したもの．故障 (failure) を引き起こす[15)]．障害とも呼ばれるが，障害は故障の同義語として用いられることがあるので，ここでは “フォールト” という用語を用いる．

品質によって大きく異なっている．したがって，全てのプロジェクトを一つの SRGM で評価するのは困難である．

そこでB社では，複数種類の SRGM を同時に適用しておいて，実測データに最もよく適合する SRGM を選び，その SRGM で信頼度を評価することにした．

ここでは，複数の SRGM を同時に適用するという意味で，B社の SRGM の適用方法をマルチ方式と呼ぶ．

2.3 インテリジェント方式（C社）

C社もB社と同様，多くの種類のソフトウェアの開発を行っているので，マルチ方式で SRGM を適用してきた．しかし，最近，ソフトウェアの種類によってはコードの自動生成や部品化，再利用が進んできた．また，開発するソフトウェアの大規模化にともなって，新人からベテランまで能力の異なる多くの人が開発に携わるようになった．時には，開発の一部を外部に委託することもある．

すなわち，一つのプロジェクトの中でも，開発されるソフトウェアの特性は均一ではなく，テスト方法，テスト要員の能力，さらに，テスト環境もさまざまである．

そこでC社では，単に実測データとの適合の度合だけで適用する SRGM を選択するのではなく，ソフトウェアやテスト工程の特性などを考慮して各プロジェクトの実態に合ったモデルを選択し，信頼度を評価することにした．

ここでは，プロジェクトの実態に合った SRGM を適用するという意味で，C社における SRGM の適用方法をインテリジェント方式と呼ぶ．

3. 必要な技術

3.1 データの収集

シングル方式は最も基本的な方法であり，A社のように初めて SRGM を利用する場合には適している．いわゆる“データ収集技術”はシングル方式の SRGM の適用にとって特に重要な技術である．なお，データ収集はB社，C社でも共通に必要な技術である．

通常，モデルの定義とモデルの利用の間には若干の立場の違いがある．たとえば，SRGM に関する論文などを見ると，モデル定義の考え方，仮定，条件などは明確にされているが，モデル適用のための具体的な方法や手順は明示されていないことが多い．特に，データ収集は省略されやすい．

データ収集を行うには，まず，収集すべきデータの定義を明確に与える必要がある．なお，収集すべきデータの定義に当たってはモデルの定義や適用上の仮定と矛盾していないことが要求される．

たとえば，運用段階への SRGM の適用を考えてみる．この場合，故障の発生とフォールトの発見を1対1に対応させることが多い．すなわち，各フォールトはそれぞれ異なる故障をただ1回だけ引き起こすとする．しかし，パッケージソフトウェアの場合，同時に複数のユーザによって使用されるため，あるフォールトが原因で引き起こされる故障は1回とは限らない．

次に必要となるのが，収集データの定義に基づくデータの収集方法の開発である．この開発で重要なのはできるかぎりテスト作業を妨げないことである．できれば，自動的にデータ収集が行えるようにすることが必要である．

SRGM の適用ではテスト工程がどのように行われたかを表す，いわゆるプロセスデータが必要となる．したがって，ソフトウェアそのものから収集されるプロダクトデータに比べるとその収集作業は一般に繁雑になる．たとえば，各テスト作業の開始時刻や終了時刻，各テスト作業で発見されたフォールト数や各フォールトの発見時刻などはテスト工程を通じて逐次的に収集する必要がある．さらに，データの収集作業はテスト実施者にとって余分な作業であるので，データ収集に要する労力が大きいと本来のテスト作業に支障をきたすだけではなく，収集データの信頼性をも低下させることになる．また，SRGM ではフォールトに関するデータを収集する．したがって，設計担当者やコーディング担当者がテスト実施者を兼ねているような場合，その申告に基づいて収集されたデータの信頼性は低くなることが多い．

4.1 では自動的なデータ収集技術の例として，現在開発が進められている GINGER システムについて述べる．

3.2 パラメータの推定

信頼度の推定精度は SRGM がもつパラメータの推定精度によって決定される．したがって，“パラメータの推定技術”の開発が重要になってくる．

B社が導入を計画しているマルチ方式とは，各 SRGM の推定精度を高めた上で，その中で最も精度の高い SRGM を選んで利用するものである．各モデルの推定精度は実測データとの適合度によって比較で

きる．適合度は，通常，偏差二乗和，χ^2 検定，Kolmogorov-Smirnov 適合度検定などによって定量的に評価される[22]．したがって，マルチ方式において本質的に重要なのは各 SRGM の推定精度の向上であり，パラメータの推定技術の確立である．

SRGM のパラメータは収集データから直接算出できるものと，最尤推定法などの統計的手法を用いて推定されるものに分類される．前者は主として，各 SRGM に固有なパラメータであり，たとえば，**4.2.1** で述べる習熟S字型 SRGM の習熟係数 r がある．一方，後者は多くの SRGM に共通なパラメータであり，たとえば初期フォールト数やフォールト発見率などがある．

各 SRGM に固有なパラメータの多くは，他の SRGM の適用には必要がなく，特別なデータに基づいて算出される．したがって，その算出だけを目的とした新たなデータ収集技術の開発が必要になる．しかし，与えられている定義に基づいてデータを収集することが困難な場合もある．たとえば，習熟係数 r の定義に基づいた値の算出に必要なデータの収集は非常に困難である[16]．

パラメータを定義式に基づいて求めることが困難な場合の対応について述べる．シングル方式であれば，是非ともその SRGM を適用したいという特別な理由がないかぎり，その適用はあきらめることになる．

一方，マルチ方式の場合，もしそうした SRGM の適用を止めるとすると，非常に基本的で，かつ，類似した SRGM しか適用できないことになる．すると，適用できる SRGM の数が限られてしまい，マルチ方式のよさがなくなってしまう．したがって，マルチ方式ではできるかぎり多種多様な SRGM の適用を可能とするため，パラメータの推定技術の確立が強く望まれる．

マルチ方式では複数の SRGM を同時に適用しているので，他の SRGM のために収集されたデータや推定されたパラメータの値を利用することもできる．**4.2.3** では，こうした利用の例を一つ示す．

3.3 特性に応じた適用

SRGM のパラメータは適用するプロジェクトの特性，すなわち，ソフトウェアの特性，テスト方法，テスト実施者の能力，テスト環境など，を表す．実測データに基づいたパラメータの推定はそうしたプロジェクトの特性の推定と等価である．

最尤推定法などの統計的手法は，その名のとおり，実測データに基づいて各パラメータに対して最ももっともらしい値を算出する方法である．しかし，プロジェクトの実態とかけはなれた値を推定する危険もある．

たとえば，小規模なプロジェクトの開発プロジェクトで，発見されるフォールトの数が比較的少ないものを考える．このプロジェクトに SRGM を適用し，最尤推定法によってパラメータの推定を行うと，主要なパラメータの一つである初期フォールト数が非常に大きな値，あるいは，∞となる場合がある[11),12)]．テスト対象の規模から判断すると，この推定結果は妥当でない．しかし，マルチ方式では，実測データとの適合度にのみ基づいて SRGM を選ぶため，このようにプロジェクトの実態とは合わない SRGM を利用する可能性がある．

こうした問題を解決するためには，「推定されたパラメータの値の妥当性を常に確認する」，あるいはさらに一歩進んで，「パラメータの値がプロジェクトの実態からかけはなれないことを常に保証する」ようなプロジェクトの“特性に応じたモデル適用技術”を確立することが必要である．

プロジェクトの特性に応じた SRGM の適用技術は次の三つに分類される．

(1) ソフトウェアやテスト工程の特性をパラメータとして組み入れた新しい SRGM の構築[21)]．

(2) 既存の複数個の SRGM を組み合わせた新しい SRGM の構築[5)]．

(3) プロジェクトの特性を考慮したデータの収集方法やパラメータの推定方法の工夫[8),15)]．

たとえば，超幾何分布モデルは(1)の一つの例である[21)]．超幾何分布モデルではテスト時間 t を定めるのに，テストプロセスの実際の処理時間だけでなく，新しくテストケース，テスト実施者数，テスト項目数，などをパラメータとして用いている．

(2)の例としては，指数型 SRGM と習熟S字型 SRGM を組み合わせた複合 SRGM がある[5)]．テスト工程の初期にはソフトウェア信頼度成長曲線の形状が指数型を示し，その後S字型となるという傾向に注目して，この複合 SRGM は提案されている．複合 SRGM の適用は発見されるフォールトの重要度の違いによって，テスト工程の途中でモデルを指数型からS字型に切り替える（**図-1** 参照）．

4.3 では，(3)の例として超指数型 SRGM に対する新しいモデル適用方法について述べる．

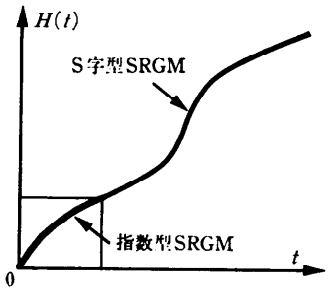


図-1 複合ソフトウェア信頼度成長モデル

4. 利用可能な技術の例

4.1 GINGER におけるデータ収集

4.1.1 GINGER の概要

GINGER システムはソフトウェア開発プロジェクトからソフトウェア，及び，その開発過程に関するデータを自動的に収集し，主にプログラマ性能[10),13)]を評価するために設計，開発されたシステムである[7)]．テスト工程で利用すれば SRGM 適用に必要なデータを収集することができる．

GINGER システムの構成とシステム内でのデータの流れを図-2 に示す．システムは次の 4 つの部分からなる．

- **データ収集部**…ソフトウェアとその開発過程に関するデータを収集する．開発者の負荷をできるだけ小さく抑え，かつ，収集データの信頼性を高くするため，データ収集は人手によらず，自動的に行う．
- **データ管理部**…収集データ，及び，分析結果をデータベースとして蓄積する．膨大なデータが取り扱えるようにデータ圧縮機能をもつ．
- **データ分析部**…収集データに基づいてプログラマ性能の評価，分析を行う．複数の尺度を同時に用いたり，適用環境に応じて尺度の選択が行える．
- **情報フィードバック部**…収集データ，及び，分析結果を，プログラマ性能に関する情報として開発者にフィードバックする．

GINGER のプロトタイプシステムは携帯性を高めるため UNIX 環境でC言語を用いて開発されている．さらに，複数のワークステーションが LAN (Ethernet) で結合された分散型環境への対応も考慮されている．

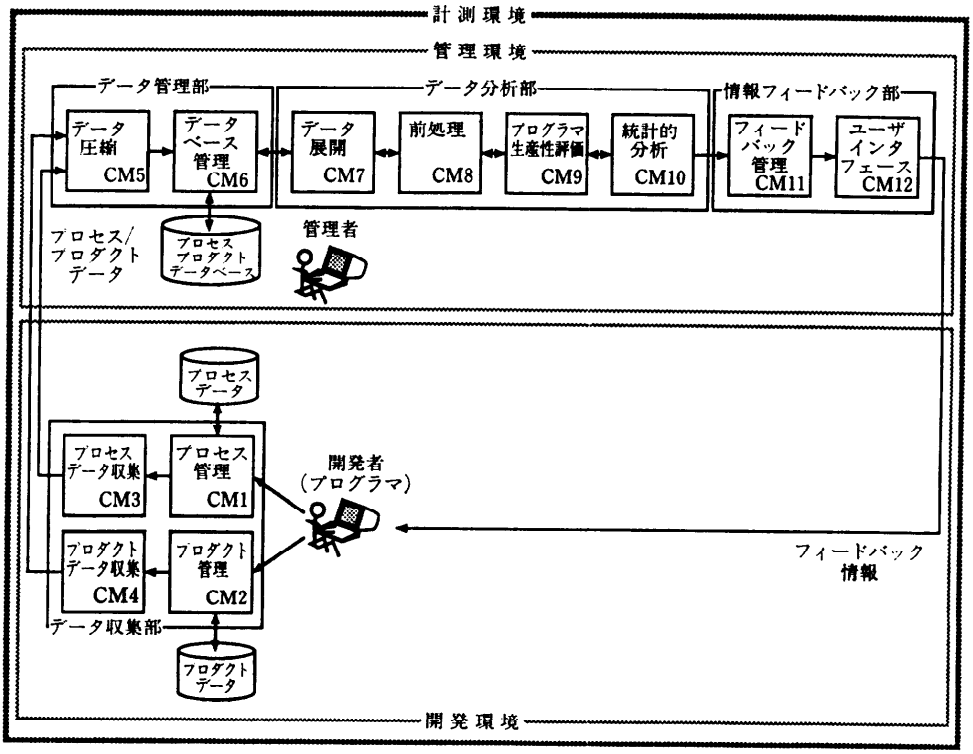


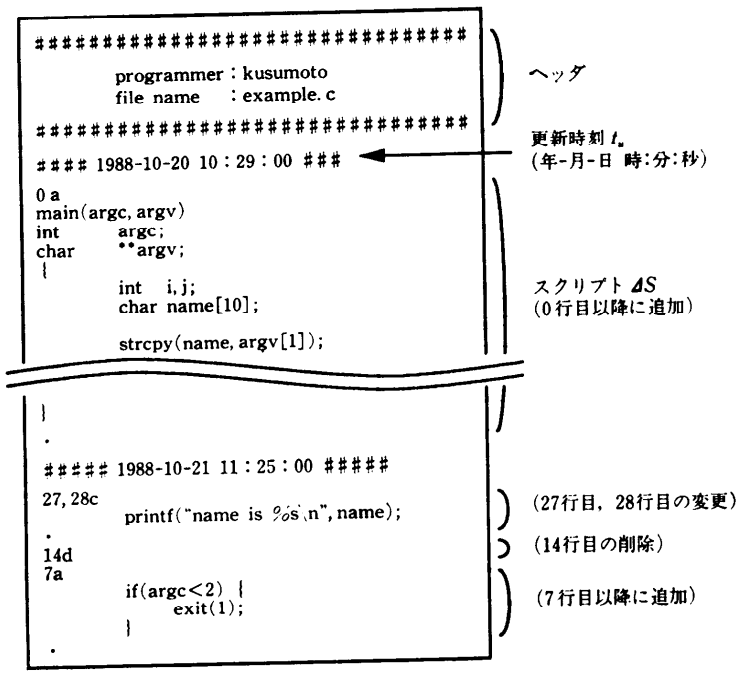
図-2 GINGER システムの構成

4.1.2 フォールト発見時刻の収集

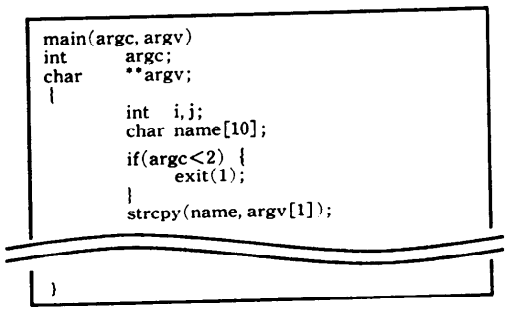
SRGM を適用するために必要な最も基本的なデータであるフォールトの発見，除去時刻を機械的に収集することは現時点では非常に困難である．しかし，その推定値を求めることは可能である．

たとえば，テスト工程におけるプログラムテキストの変更量と発見，除去されたフォールトの数には高い相関のあることが知られている[4]．プログラムテキストの変更量は，エディタなどによるテキストの編集作業においてテキストに追加された行数，他のテキストで置換された行数，及び，テキストから削除された行数に基づいて算出される．具体的には，プログラムテキストの更新履歴に基づいて算出する．

図-3 は GINGER のプロトタイプシステムで作成されたプログラムテキストの更新履歴の例である[7]．これは，各編集作業の前後のテキストを UNIX のファイル比較コマンド "diff" によって比較し，その差分を編集時刻とともに順次記録したものである．GINGER ではこのデータを基にフォールト数や各フォールトの原因となったエラーの寿命[10]を近似的に算出する．



(a) 更新箇所の差分リスト



(b) 最新バージョン

図-3 プログラムテキストの更新履歴の例

4.1.3 テスト時間の収集

テスト作業に費やされた工数を表すテスト時間もSRGMの適用に必要な基本的なデータの一つである．テスト時間として自動収集可能なデータには次のものが考えられる．

（1） カレンダー時間…通常の時間．計算機上，及び，机上においてテストに費やされた作業時間．毎日の作業時間が比較的一定していて把握しやすい場合や，テスト期間が長くて毎日の作業時間の変動を無視できる場合に用いる．

（2） 端末使用時間…テスト実施者が端末を使用している時間．机上での作業時間は含まれない．計算機上でのテストが主と考えられる場合や，毎日の作業時間の変動が大きく，カレンダ時間が利用できない場合に用いる．

（3） CPU 時間…ソフトウェアの実行に要した計算機の稼働時間．テスト対象となるソフトウェアを長時間連続して稼働させる場合や，出荷後の運用段階にモデルを適用する場合に用いる[15]．

（4） テスト回数…計算機上でのソフトウェアの実行回数．一回のテストによるソフトウェアの実行CPU 時間が非常にわずかで測定してもあまり意味がない場合に用いる[19]．

GINGER のプロトタイプシステムにおいて収集さ

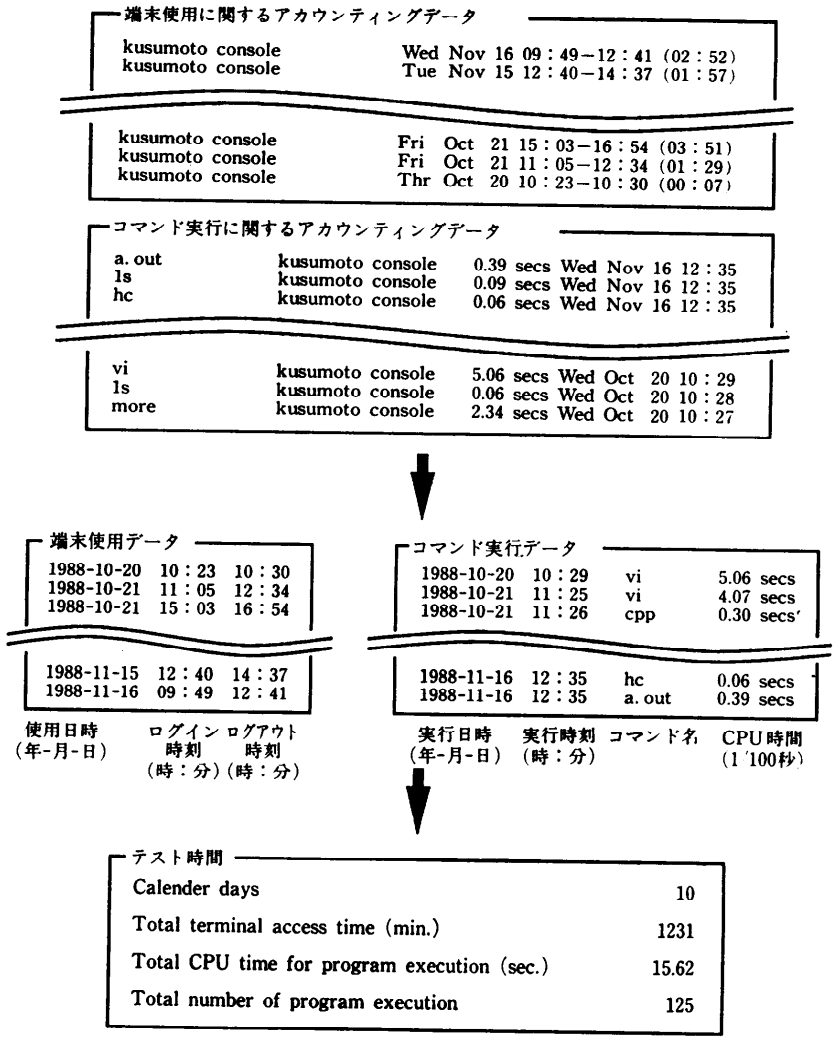


図-4 テスト時間の例

れるテスト時間の例を図-4 に示す[7]．これらの値はGINGER により全て自動収集可能である．しかし，テスト作業に要した工数を完全には表してはいない．それぞれが表しているものも少しずつ異なる．したがって，テスト作業の形態や要求されるデータの精度に応じてこれらを使い分ける必要がある．これらを組み合わせてテスト時間を新しく定義することも考えられる．

4.2 習熟係数 r の推定法

4.2.1 習熟係数 r

習熟S字型 SRGM の習熟係数 r は，文献16) によると，フォールトの総数に対する発見可能なフォールトの総数の比として解釈される．しかし，その値を計測することは一般に困難なため，経験に基づいて定めた値を用いているのが現状である．それに対して，収集が容易なデータのみに基づき，かつ，習熟S字型 SRGM の推定精度が最大となるような r の値の推定法が提案されている[11),14)]．

4.2.2 モデルの推定精度 $A(r)$ との関係

習熟係数 r と習熟S字型SRGMの推定精度 $A(r)=(N(r)-N_0)/N_0\times 100(\%)$ の関係を図-5 に示す[11),14)]．ここで，N_0 は初期フォールト数の実測値を，$N(r)$ は習熟係数の値を r としたときの習熟S字型 SRGM による初期フォールト数の推定値を，それぞれ表す．図-5 より，r と $A(r)$ の関係が右上がりのほぼ直線となっていることが分かる．

4.2.3 r の推定法

習熟S字型 SRGM の推定精度が最大となる（すなわち，$A(r)=0$ となる）r の最適値 r_{opt} に対する近似値を求める方法について述べる．準備として次の(1)，(2)を仮定する．

(1) $0<r\leq 1$ の範囲において r と $A(r)$ の関係が線形近似できる（図-5 参照）．

このとき，次の関係式が成立する．

$$A(r)=ar+b \tag{1}$$

$r\neq 0$ なので，0とほぼ等しい正の実数を表す記号 $\tilde{0}$ を導入する．このとき，a，b の値は

$$a=\frac{A(1)-A(\tilde{0})}{1-0}=A(1)-A(\tilde{0}) \tag{2}$$

$$b=A(\tilde{0}) \tag{3}$$

となる（図-6 参照）．

(2) 遅延S字型 SRGM を用いた初期フォールト数の推定精度は非常に高い[11),14)]．

今，N_0 を初期フォールト数の実測値とし，N_{DS} を遅延S字型 SRGM による初期フォールト数の推定値とするとき，次の関係式が成立する．

$$N_{DS}=N_0 \tag{4}$$

仮定(1)より，r の最適値 r_{opt} に対し次式が導かれる．

$$r_{opt}=-\frac{b}{a}=-\frac{A(\tilde{0})}{A(1)-A(\tilde{0})} \tag{5}$$

このとき，

$$A(r)=\frac{N(r)-N_0}{N_0} \tag{6}$$

であるから，式(5)より次式が求まる（図-6 参照）．

$$r_{opt}=\frac{N(\tilde{0})-N_0}{N(\tilde{0})-N(1)} \tag{7}$$

ここで，仮定(2)より次式が求まる．

$$r_{opt}=\frac{N(\tilde{0})-N_{DS}}{N(\tilde{0})-N(1)} \tag{8}$$

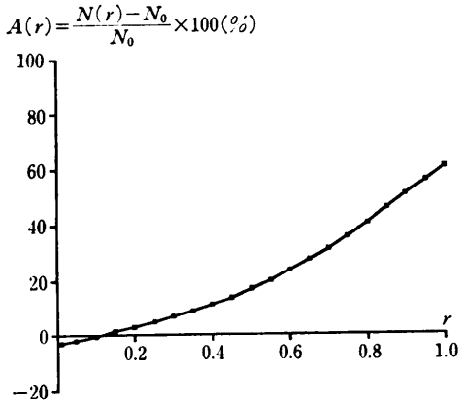


図-5 習熟係数 r と推定精度 $A(r)$ の関係

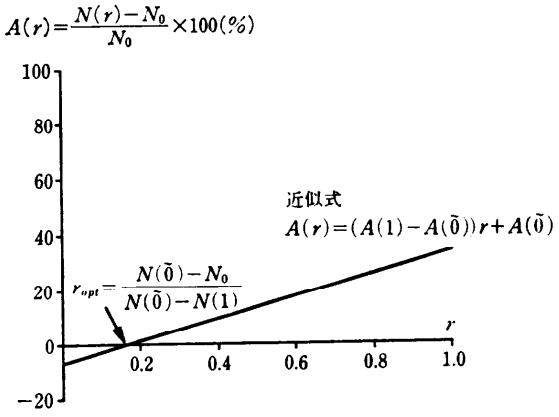


図-6 習熟係数 r の決定方法

この式(8)を利用するなら，遅延S字型 SRGM を1回と習熟S字型 SRGM を2回（たとえば，$r=0.01$ と $r=1$ に）適用することで r_{opt} の近似値が求まる．すなわち，習熟S字型 SRGM に注目してみると，$r=0.01$ と $r=1$ について試行した後，最適値 r_{opt} について適用することになる．

4.3 超指数型 SRGM の適用

4.3.1 超指数型 SRGM

超指数型 SRGM ではプログラムを幾つかのクラスタに分類し，クラスタごとに指数型 SRGM を適用する．こうして得られたソフトウェア信頼度成長曲線を単純に加え合わせてプログラム全体を評価する[16]．

超指数型 SRGM におけるソフトウェア信頼度成長曲線 $H(t)$ は次のように定義される．

$$H(t)=\sum_{i=1}^{n} H_i(t)$$

$$H_i(t)=N_i\{1-\exp(-\phi_i t)\} \qquad (9)$$

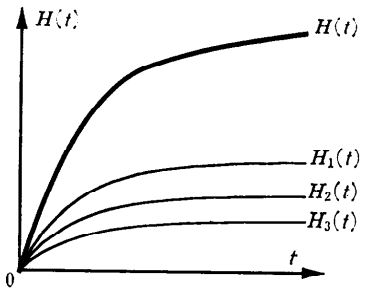
ここで n はクラスタ数を，$H_i(t)$ はクラスタ i のソフトウェア信頼度成長曲線を，N_i はクラスタ i の初期フォールト数を，ϕ_i はクラスタ i の定常状態のフォールト1個当たりのフォールト発見率を，それぞれ表す．

クラスタへの分類はプログラムの構造や機能に基づいて行うことも考えられる．しかし，SRGM を適用する立場からは初期フォールト数やフォールト発見率の差に基づいて行うほうがより効果的である．たとえば，新規に作成されたプログラムと自動生成や再利用されたプログラムでは初期フォールト数やフォールト発見率は大きく異なる[16]．すなわち，モジュールごとに自動生成率や再利用率を算出し，それに基づいてクラスタ分けすることが考えられる．

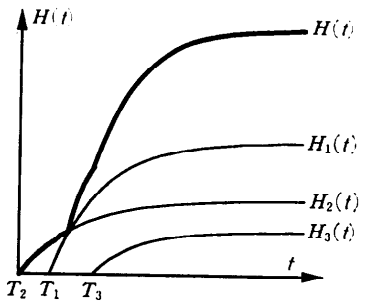
4.3.2 テスト開始時刻の差

クラスタごとに初期フォールト数とフォールト発見率を考慮する点で，超指数型 SRGM は指数型 SRGM よりもソフトウェアの特性を考慮したモデルである．しかし，定義式より各 $H_i(t)$ を単純に加え合わせており，$H_i(t)$ の原点は同一であると仮定している．すなわち，全てのクラスタのテスト開始時刻は同じである．

開発されるプログラムが小規模である場合には，この仮定は正しいかも知れない．しかし，プログラムの規模が大きくなってくると，その全てを一度にテストすること（あるいは全てを一度にテストできるようなテストデータを作成すること）は困難になる．したがって，全てのクラスタのテスト開始時刻が同一であると考えることはできなくなる．また，仮に小規模であっても，その構造上の特徴から各クラスタのテスト開始時刻が異なることも十分考えられる．



(a) 従来の適用方法



(b) テスト開始時刻の差を考慮した適用方法

図-7 適用方法の違いによるソフトウェア信頼度成長曲線 $\left(H(t)=\sum_{i=1}^{n} H_i(t)\right)$ の差

そこで，クラスタごとにテスト開始時刻が異なると仮定し，それぞれのテスト開始時刻が $H_i(t)$ の原点となると考える．この考えに基づいて超指数型 SRGM を再定義すると次のようになる[11]．

$$H(t)=\sum_{i=1}^{n} H_i(t)$$

$$H_i(t)=\begin{cases}0 & t<T_i\\ N_i\{1-\exp(-\phi_i(t-T_i))\} & t\geq T_i\end{cases} \qquad (10)$$

ここで T_i はクラスタ i のテスト開始時刻を表す．

適用方法の違いによるソフトウェア信頼度成長曲線の差を図-7に示す．図-7より，クラスタ間でテスト開始時刻に差がある場合，超指数型 SRGM のソフトウェア信頼度成長曲線はS字に近くなることが分かる．

5. 適用技術の将来

ここでは，A社，B社，C社のモデル適用計画を例に用いて，SRGM を適用するための基本技術の現状

と問題点を述べた．さらに，それらの基本技術の中で現在利用可能なものを幾つか紹介した．

本稿での議論は全て現行プロジェクトから収集したデータのみを利用するものであった．そのためSRGMの適用技術としては限界がある．

たとえば，SRGMを用いた初期フォールト数の推定を行う場合，テスト工程のできるだけ初期の段階に精度の高い推定を行えることが重要である．しかし，現行プロジェクトのデータのみを利用するかぎり，ほとんどのSRGMの予測性は非常に低い[12]．推定精度の高い遅延S字型SRGMであっても，テスト工程の2/3以上の時点まで進まないと，その精度は低い[12),22)]．

これを解決する一つの応用技術として，ここでは現行プロジェクトのデータに加えて，過去のプロジェクトでの収集データやSRGMの適用結果を利用することを提案する．この利用を可能とするには，フォールトデータだけを記録するのではなく，データを収集したプロジェクトの特性やSRGM適用の経緯などの情報も蓄積する必要がある．

メリーランド大学で開発の進められているTAMEシステムはこうした試みの一つである[1),2)]．Experience baseと呼ぶデータベースに各プロジェクトのソフトウェアの開発，及び，計測の経緯を記録している．

こうした過去のプロジェクトにおける“経験”の蓄積はSRGMの予測性の向上だけでなく，本稿で述べた各基本技術の質的な改良を生み出す可能性がある．たとえば，シングル方式におけるデータ収集の頻度の設定，マルチ方式におけるモデルの比較，さらに，インテリジェント方式におけるプロジェクトの特性の抽出などの各作業に参考となる情報の提供が可能になる．

参考文献

1) Basili, V.R.: Software Development: A Paradigm for the Future, Proc. of 13th International Computer Software and Applications Conference, pp. 471-485 (1989).
2) Basili, V.R. and Rombach, H.D.: The TAME Project: Towards Improvement-Oriented Software Environments, IEEE Trans. Softw. Eng., 14, 6, pp. 758-773 (1988).
3) Brocklehurst, S., Chan, P.Y., Littlewood, B. and Snell, J.: Recalibrating Software Reliability Models, IEEE Trans. Software Eng., 16, 4, pp. 458-470 (1990).
4) Dunsmore, H.E. and Gannon, J.D.: Analysis of the Effects of Programming Factors on Programming Effort, J. Syst. Softw., 1, pp. 141-153 (1980).
5) 岩佐 博：エラーの重要度およびエラー修正過程を考慮した複合ソフトウェア信頼度成長モデルの提案，情報処理学会論文誌，Vol. 29, No. 5, pp. 506-512 (1988).
6) 菊野，松本，鳥居：ソフトウェア信頼性の実現技術—現状と今後の展望—，電子情報通信学会誌，J73, 5, pp. 454-460 (1990).
7) Kusumoto, S., Matsumoto, K., Kikuno, T. and Torii, K.: GINGER: Data Collection and Analysis System, IEICE Technical Report, SS 90, pp. 39-48 (1990).
8) Levendel, Y.: Reliability Analysis of Large Software Systems: Defect Data Modeling, IEEE Trans. Software Eng., 16, 2, pp. 141-152 (1990).
9) Littlewood, B. (Ed.): Software Reliability: Achievement and Assessment, Blackwell Scientific Pub. (1987).
10) 松本，井上，菊野，鳥居：エラー寿命に基づくプログラマ性能の実験的評価—大学環境におけるプログラム開発—，電子情報通信学会論文誌D, J71-D, 10, pp. 1959-1965 (1988).
11) Matsumoto, K., Inoue, K., Kikuno, T. and Torii, K.: Experimental Evaluation of Software Reliability Growth Models, Proc. of 18-th International Symposium on Fault Tolerant Computing, pp. 148-153 (1988).
12) 松本，井上，菊野，鳥居：ソフトウェアの残存エラー数推定モデルの定量的比較，ソフトウェア・シンポジウム '88, pp. 161-170 (1988).
13) Matsumoto, K., Inoue, K., Kudo, H., Sugiyama, Y. and Torii, K.: Error Life Span and Programmer Performance, Proc. of 11th International Computer Software and Applications Conference, pp. 259-265 (1987).
14) 松本，菊野，鳥居：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, J 73-D-I, 2, pp. 175-182 (1990).
15) Musa, J.D., Iannino, A. and Okumoto, K.: Software Reliability: Measurement, Prediction, Application, McGraw-Hill (1987).
16) Ohba, M.: Software Reliability Analysis Models, IBM J. Res. Develop., 28, 4, pp. 428-443 (1984).
17) Ohba, M. and Chou, X.: Does Imperfect Debugging Affect Software Reliability Growth, Proc. of 11th International Conference on Software Engineering, pp. 237-244 (1989).
18) Rook, P. (Ed.): Software Reliability Handbook, Elsevier Science Pub. (1990).
19) 当麻喜弘：ソフトウェアの信頼性，情報処理，

Vol. 31, No. 4, pp. 508-517 (1990).
20) 当麻喜弘 (編)：フォールトトレラントシステム論，電子情報通信学会 (1990).
21) Tohma, Y., Tokunaga, K., Nagase, S. and Murata, Y.: Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution, IEEE Trans. Software Eng., 15, 3, pp. 345-355 (1989).
22) 上村，樋口，阿部，藤野：ソフトウェア信頼度成長モデルの検証—遅延S字型 NHPP モデルの基本ソフトウェアへの適用—，電子情報通信学会技術研究報告，R 88-4, pp. 17-22 (1988).
23) 山田 茂：テスト労力の投入量を考慮したソフトウェア信頼度成長モデルと適合性比較，電子情報通信学会論文誌D，J70-D, 12, pp. 2438-2445 (1987).
24) Yamada, S., Ohba, M. and Osaki, S.: S-Shaped Reliability Growth Modeling for Software Error Detection, IEEE Trans. Reliab., R-32, 5, pp. 475-478 (1983).
25) IEEE Standard Glossary of Software Engineering Terminology, IEEE, Rep. IEEE-Std-729-1983 (1983).

(平成2年9月10日受付)