2001年度日本オペレーションズ・リサーチ学会
秋季研究発表会

不完全デバッグ環境を考慮した テスト空間依存型ソフトウェア信頼度成長モデル

01507374　富士通周辺機株式会社　*藤原隆次　FUJIWARA Takaji
01702425　鳥取大学　山田茂　YAMADA Shigeru

1　まえがき

様々な分野で重要な役割を担うソフトウェアシステムの高信頼化を図るための管理技術として，ソフトウェア開発における様々なテスト環境や運用環境を反映したソフトウェア信頼度成長モデル (以降，SRGM と略す) が提案されている．その多くのモデルは，完全デバッグ環境を仮定しているが，実際には，フォールト修正時に新規フォールトが作り込まれる不完全デバッグ環境を仮定する方がより実現的である．本研究では，テスト工程において投入されるテストケースによりソフトウェア内に形成されるテスト空間占有率の時間的推移が，ソフトウェアフォールト発見事象に影響するものと仮定し，さらにフォールト修正時に新規フォールトが作り込まれる**不完全デバッグ環境**を仮定した SRGM を提案する．

2　テスト空間の記述

ソフトウェア開発者は，テスト工程において多種・多様なテストケースを用いてソフトウェア内に潜在するフォールトの発見および修正・除去，つまりデバッグ作業を行っている．このとき，投入されたテストケースから影響を受けるソフトウェア内のモジュールおよび機能のテストパスに関する集合体が存在し，この集合体を総称して**テスト空間**と呼ぶ (図 1 参照) [1]．このテスト空間は，投入されるテストケースの増加とともに拡大し，これに伴ってデバッグ実施者の発見可能となるフォールト数も増加し，最終的にソフトウェアシステム全体に拡大する．

しかし，実際には，任意のテスト時刻 t においてソフトウェアシステム内に潜在するフォールト数が増加する可能性がある．また，テスト工程後期あるいは実際の運用・保守段階で発見されたフォールトを修正・除去する場合，テスト工程初期と比較して作り込まれる新規フォールト数が，指数関数的に増大するものと推測でき，テスト空間は拡大し続ける場合が考えられる．

以上のことを考慮して，ソフトウェア内におけるテスト空間占有率と，テストにより発見可能となるフォールト数との関係は以下の微分方程式で表すことができる．

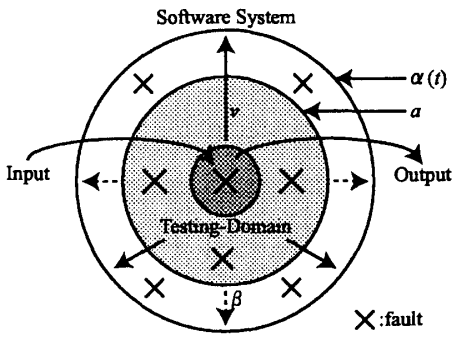

図 1: ソフトウェアのテスト空間概念図

$$\frac{du(t)}{dt} = v\left[\alpha(t) - u(t)\right] \ (v > 0), \qquad (1)$$

式 (1) において，v はテスト空間拡大率，$u(t)$ は任意のテスト時刻 t においてテスト空間内に潜在する累積フォールト数，$\alpha(t)$ は任意のテスト時刻 t においてソフトウェア内に潜在する累積フォールト数を表す．ここで，$\alpha(t)$ は次式で表すことができる [2]．

$$\alpha(t) = a \cdot \exp\left[\beta t\right] \ (a > 0,\ \beta > 0). \qquad (2)$$

式 (2) において，a はテスト開始前にソフトウェア内に潜在する総期待フォールト数，β は新規フォールト潜入率を表す．

したがって，微分方程式 (1) を，初期条件 $u(0) = 0$ の下で $u(t)$ に関して解くと次式を得る．

$$u(t) = \frac{av}{\beta + v}\left(\exp\left[\beta t\right] - \exp\left[-vt\right]\right). \qquad (3)$$

式 (3) は，ソフトウェア内におけるテスト空間占有率が図 2 のように拡大することを表し，不完全デバッグ環境を考慮したテスト空間関数 (I-D テスト空間) と呼ぶ．

さらに，単位テスト時間当りにデバッグ実施者の発見可能となるフォールトの増加数を表す**テスト空間成長関数** $\gamma(t)$ は，式 (3) から以下のようになる．

$$\begin{aligned}\gamma(t) &\equiv \frac{du(t)}{dt} \\ &= \frac{av}{\beta + v}\left(\beta \cdot \exp\left[\beta t\right] + v \cdot \exp\left[-vt\right]\right). \qquad (4)\end{aligned}$$

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

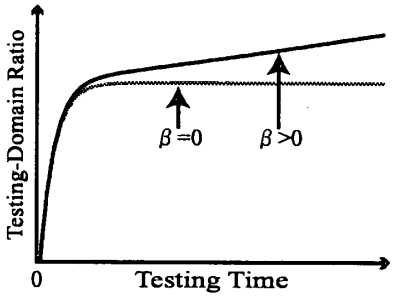

図 2: テスト空間占有率の時間的推移

3 ソフトウェア信頼度成長モデル

ソフトウェア開発のテスト工程におけるフォールト発見事象をモデル化するために，任意のテスト時刻 $t\,(t \geq 0)$ までに発見される総フォールト数を表す計数過程 $\{N(t), t \geq 0\}$ を導入する．この計数過程が平均値関数 $H(t)$ をもつ NHPP に従うものと仮定すると，SRGM は

$$\Pr\{N(t)=n\} = \frac{\{H(t)\}^n}{n!}\exp[-H(t)] \quad (n=0,1,2,\cdots),$$
$$H(t) = \int_0^t h(x)\,dx, \tag{5}$$

と定式化できる [3]．式 (5) において，$H(t)$ は $N(t)$ の期待値であり，テスト時刻 t までに発見される累積フォールト数を表す．さらに，$h(t)$ は NHPP の強度関数と呼ばれ，単位時間当りに発見されるフォールト数，つまり瞬間フォールト発見率を表す．

ここで，単位時間当りに発見されるフォールト数は，テスト空間内に残存するフォールト数に比例するものと仮定すると，式 (3) で表される I-D テスト空間関数 $u(t)$ から，平均値関数 $m(t)$ に関して次の微分方程式が成立する．

$$\frac{dm(t)}{dt} = b\,[u(t) - m(t)] \;\; (1 > b > 0). \tag{6}$$

式 (6) において，b はテスト空間内に残存するフォールト 1 個当りの発見率を表す．式 (6) の微分方程式を，初期条件 $m(0) = 0$ の下で $m(t)$ に関して解くと次式を得る．

$$m(t) = abv\left\{\frac{\exp[\beta t]}{(\beta+v)(\beta+b)} + \frac{\exp[-vt]}{(\beta+v)(v-b)} - \frac{\exp[-bt]}{(\beta+b)(v-b)}\right\} \quad (b \neq v). \tag{7}$$

また，式 (6) において，$\beta = 0$ および $v \to \infty$ のとき，Goel & Okumoto の指数形 SRGM に帰着する [3]．さらに，$\beta = 0$ および $v = b$ のとき，Yamada & Osaki の遅延 S 字形 SRGM に帰着する [3]．以上のことから，本モデルは指数形 SRGM および遅延 S 字形 SRGM の特徴を兼ね備えた SRGM であることが伺える．

さらに，瞬間フォールト発見率を表す NHPP の強度関数 $h(t)$ は，式 (7) から次式のようになる．

$$h(t) = abv\left\{\frac{\beta\cdot\exp[\beta t]}{(\beta+v)(\beta+b)} - \frac{v\cdot\exp[-vt]}{(\beta+v)(v-b)} + \frac{b\cdot\exp[-bt]}{(\beta+b)(v-b)}\right\} \quad (b \neq v). \tag{8}$$

4 適合性比較

適合性比較の評価基準として，平均偏差 2 乗和 (以降，ASSE と略す)[3] および赤池情報量基準 (以降，AIC と略す)[4] を用いる．比較対象 SRGM として，完全デバッグ環境を考慮したテスト空間依存型 SRGM(以降，基本形 T-DSRGM と略す) を取り上げる．比較結果を表 1 に示す．

表 1: 適合性比較結果

Data Set	Proposed Model	Basic Testing-Domain SRGM
ASSE	3.055	9.529
AIC	76.875	78.244

参考文献

[1] 藤原隆次, 山田茂：“テスト習熟性を考慮したソフトウェア信頼度成長モデルとその適合性評価に関する考察”，電子情報通信学会論文誌，vol.J83-A, no.2, pp.188-195 (Feb. 2000).

[2] S.Yamada, K.Tokunou and S.Osaki : “Imperfect debugging models with fault introduction rate for software reliability assessment”, *International Journal of Systems Science*, vol.23, no.12, pp.2241-2252 (Dec. 1992).

[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，東京 (1994).

[4] 赤池弘次：“情報量基準 AIC とは何か”，数理科学，no.153, pp.5-11 (1976).

© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．