ソフトウェア開発におけるテスト労力の最適配分問題

山田　茂，一森　哲男

1. まえがき

ソフトウェア開発の最終段階であるテスト工程では，開発作業の結果として，ソフトウェア内に多くの人為的誤りや欠陥などのソフトウェアフォールト（以下フォールトと略す）が潜入しているので，これらを発見･修正する必要がある．さらに，一般的な大規模ソフトウェア開発プロジェクトでは，総開発コストの約50 %をテスト工程で費やされていると言われる [1]．このテスト工程は，ユーザの要求する機能が満たされているかどうかを確認するために，単体テスト，統合テスト，総合テストという一連の段階がある．したがって，ソフトウェアのテストが効率的に行われるためには，開発資源を有効に用いてテスト工程を管理する必要がある．例えば，与えられたテスト資源をどのようにソフトウェアプロジェクトの各サブプロジェクトに配分すれば，意図するソフトウェアの品質／信頼性目標を満たすことができるかということは重要な問題である．ここで，テスト資源とは，テストにより費やされたCPU時間や工数などのテスト労力である．このようなテスト工程での信頼性評価結果に基づく合理的なテスト労力の配分問題は，従来はほとんど議論されていない．また，情報ネットワーク技術の進展により，複数プロジェクトで分散協調的に行う分散ソフトウェア開発形態が望まれていることから，計画されたテスト労力量を適切に各プロジェクトに配分する必要性が出てきた

本稿では，まず，テスト工程での信頼性評価手法としてのソフトウェア信頼度成長モデルと，これに基づく最適な総テスト時間を見積るための最適リリース問題について議論する．次に，テスト終了後の信頼性を確保するために，いくつかのテスト工程管理上の制約を考慮して，代表的な非線形数理計画問題として定式化されるテスト労力の最適配分問題について考察する．ここで，投入されるテスト労力と達成されるソフトウェア信頼性を関係づけるソフトウェア信頼度成長モデル [2],[3] を適用する．

やまだ　しげる　鳥取大学工学部
〒680-8552 鳥取市湖山町南 4-101
いちもり　てつお　大阪工業大学情報科学部
〒573-0196 枚方市北山 1-79-1

2. ソフトウェア信頼度成長モデル

ソフトウェアいわゆるコンピュータプログラムは人間の作った知的生産物であるため，その開発工程において人為的誤りや欠陥などのフォールト（バグとも呼ばれる）が潜入することは避けられない．そこで，ソフトウェア開発の最終工程におけるテスト工程や実際の運用･保守段階において，フォールトの発見･修正が行われる．このようなフォールト発見事象を記述し，テスト工程や運用・保守段階におけるソフトウェアの信頼性評価を行うために，時刻 t までに発見される総フォールト数 $N(t)$ を確率変数として導入し，これを計数過程 (counting process) とみなして $\{N(t), t \geq 0\}$ とする．この $N(t)$ に対し，非同次ポアソン過程 (non-homogeneous Poisson process，以下NHPPと略す) を導入し，$N(t)$ の期待値を $H(t)$ とすれば，NHPPに基づくソフトウェア信頼度成長モデル [4]-[6] は，

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

$$H(t) = \int_0^t h(x)dx, \tag{2}$$

と定式化され，ソフトウェアの信頼性評価において特に実用性の高さから有望視されている．ここで $H(t)$ は，NHPPの平均値関数 (mean value function) と呼ばれ，時刻 t までに発見される総期待フォールト数を表す．また，$h(t)$ は瞬間フォールト発見率を表し，NHPPの強度関数 (intensity function) と呼ばれる．

© 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．

本稿では，テスト工程において投入される単位テスト労力当りに発見されるフォールト数が，その時点において残存するフォールト数に比例するものと仮定して構築されたテスト労力依存型ソフトウェア信頼度成長モデル (testing-effort dependent software reliability growth model) [2],[3] を考える．すなわち，式 (2) の平均値関数 $H(t)$ に対して

$$H_T(t) = a(1 - \exp[-rW(t)]) \qquad (a > 0, \quad 0 < r < 1), \quad (3)$$

と仮定するものである．ここで，パラメータ 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，パラメータ r は任意のテスト時刻における 1 個当りのフォールト発見率，$W(t)$ はテスト工程において時刻 t までに費やされるテスト労力の累積消費量を表す．式 (3) から，任意のテスト時刻 t においてソフトウェア内に残存する総期待フォールト数は

$$z(t) \equiv a - H_T(t) = a \cdot \exp[-rW(t)], \quad (4)$$

により与えられる．また，テスト終了後，時間区間 $(0, s]$ でのソフトウェア信頼度，すなわちソフトウェア障害の発生しない確率は

$$R(s) = \exp[-\gamma a \cdot \exp[-rW(t)]s] \quad (\gamma > 0, s \geq 0), \quad (5)$$

により表すことができる．ここで，γ は定数パラメータである．

3. ソフトウェアの最適リリース問題 [7],[8]

ソフトウェアのテスト工程では，それまでの開発作業により作り込まれたソフトウェア内に潜在するフォールトの発見･修正作業が行われている．このとき，テスト時間を費やせば費やすほどソフトウェア内の残存フォールト数は少なくなるので，ソフトウェアの信頼性は向上する．しかし，テスト時間が長くなり過ぎるとユーザへのリリース時期が遅れる結果となる．また逆に，テスト時間が短か過ぎるとユーザへのリリース時期（納期）に対する要求は満足できるが，ソフトウェア内に残存するフォールトが数多く存在することになるために，リリース後のソフトウェアの信頼性は低く，運用段階で発生するフォールトの保守コストは増加する結果となる．よって，ソフトウェアのテスト工程から運用段階に移行するのに最適なリリース時期（あるいは総テスト時間）を求める問題が存在する．

以上のことから，関連する総期待ソフトウェアコストを定式化するために，次のコストパラメータを定義する．

c_1 :テスト工程において発見されるフォールト 1 個当りの修正コスト

c_2 :運用段階において発見されるフォールト 1 個当りの修正コスト $(c_2 > c_1 > 0)$

c_3 :テストに要する単位時間当りのコスト $(c_3 > 0)$

上記のコストパラメータを用いて，ソフトウェアの最適リリース問題に関する総期待ソフトウェアコストが導かれる．このとき，式 (3) の平均値関数 $H_T(t)$ を使って，テスト工程における総テスト時間を T 時間とすると，ソフトウェアのテスト工程および運用段階において費やされる期待コストの総計，つまり総期待ソフトウェアコスト $C(T)$ は，

$$C(T) = c_1 H_T(T) + c_2\{a - H_T(T)\} + c_3 T, \quad (6)$$

となる．よって，式 (6) の $C(T)$ を最小にするような総テスト時間 $(T = T^*)$ を求めることがソフトウェアの最適リリース問題である．

4. 最適テスト労力配分問題

2. で議論したテスト労力依存型ソフトウェア信頼度成長モデルを使って，代表的な最適テスト労力配分問題について議論する．

4.1 基本的テスト労力配分問題 [9],[10]

次のテスト労力配分問題を考える．

(1) テスト工程での単体テストにおいて，ソフトウェアシステムを構成する独立な M 個のモジュールあるいはソフトウェア部品をテストする（または，あるソフトウェアプロジェクトが M 個の独立なサブプロジェクトから構成され，それらを独立にテストすると考えてもよい）．

(2) 各ソフトウェアモジュール内の残存フォールト数は，テスト労力依存型ソフトウェア信頼度成長モデルにより推定する．

(3) 総テスト労力量 Q は予め計画値として与えられ，その制約内で各モジュールにテスト労力を配分する．

(4) ソフトウェア内の残存フォールト数をできるだけ少なくするために，所定の計画値 Q を各モジュールに適切に配分する．

© 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．

単体テスト工程で，モジュール$i(i=1,2,\cdots,M)$のテストにq_iのテスト労力量が費やされたものとする．モジュールi内の期待残存フォールト数は，式(4)から

$$z_i = a_i \cdot \exp[-r_i q_i] \quad (i=1,2,\cdots,M), \tag{7}$$

と推定される．但し，a_iは単体テスト開始前のモジュールi内に潜在する総期待フォールト数，r_iはモジュールiに対する単位テスト労力当りのフォールト発見率，さらにq_iはモジュールiに配分されたテスト労力量とする．上記のことから，テスト資源の最適配分問題を定式化すると，

$$\text{minimize} \sum_{i=1}^{M} w_i a_i \cdot \exp[-r_i q_i], \tag{8}$$

$$\text{subject to} \sum_{i=1}^{M} q_i \le Q, q_i \ge 0 (i=1,2,\cdots,M), \tag{9}$$

となる．ここでw_iは，モジュールiの重要度やプログラムの複雑度などを考慮した正の重みである．したがって，式(9)の制約条件式の下で，式(8)を最小化する各モジュールへの最適テスト労力配分量$q_i^*(i=1,2,\cdots,M)$を求めることが目的となる．

以上のことから，式(9)の制約条件式の下で，式(8)を目的関数とする最適化問題を解く．ここで，制約条件である式(9)はQ以下の不等式であるが，目的関数である式(8)は凸でかつ単調減少であるので，テスト労力量を使用すればするほど良い結果を生む．よって，テスト労力量Qを全て使用することになる．

ここで，ラグランジュ乗数λを導入する．このとき，

$$L = \sum_{i=1}^{M} w_i a_i \cdot \exp[-r_i q_i] + \lambda \left[\sum_{i=1}^{M} q_i - Q \right], \tag{10}$$

を得る．非線形計画法の理論[11]より，この問題が最適解をもつための必要十分条件は，$q_i \ge 0$であるので，

$$\left.\begin{array}{ll} \frac{\partial L}{\partial q_i} = -w_i a_i r_i \cdot \exp[-r_i q_i] + \lambda \ge 0 & \\ q_i \cdot \frac{\partial L}{\partial q_i} = 0 & (i=1,2,\cdots,M) \end{array}\right\}, \tag{11}$$

となる．さらに，$A_i = w_i a_i r_i (i=1,2,\cdots,M)$とし，各モジュールに対して，次のように順序付けされているものとする．

$$A_1 \ge A_2 \ge \cdots \ge A_{k-1} \ge A_k \ge A_{k+1} \ge \cdots \ge A_M. \tag{12}$$

式(12)は，各モジュールに配分されたテスト労力量に対して，フォールトを発見しやすい順に並べてあることを意味する．ここで，$A_k > \lambda \ge A_{k+1}$のとき，式(11)を解くと，

$$q_i = \max\left\{0, \frac{1}{r_i}(\ln A_i - \ln \lambda)\right\},$$

つまり，

$$\left.\begin{array}{ll} q_i = (\ln A_i - \ln \lambda)/r_i & (i=1,2,\cdots,k) \\ q_i = 0 & (i=k+1,\cdots,M) \end{array}\right\}, \tag{13}$$

となる．このとき，式(9)および式(10)より，

$$\ln\lambda = \frac{\sum_{i=1}^{k} \frac{1}{r_i} \ln A_i - Q}{\sum_{i=1}^{k} \frac{1}{r_i}}, \tag{14}$$

となる．式(14)を満たすλの値をλ_kとすると，

$$\lambda_k = \exp\left\{ \frac{\sum_{i=1}^{k} \frac{1}{r_i} \ln A_i - Q}{\sum_{i=1}^{k} \frac{1}{r_i}} \right\} (k=1,2,\cdots,M),$$

となる．

したがって，最適ラグランジュ乗数λ^*は，$\lambda^* \in \{\lambda_1, \lambda_2, \cdots, \lambda_M\}$となり，容易に$\lambda^*$が求まる．よって，テスト労力の最適配分量は，

$$q_i^* = \max\left\{0, \frac{1}{r_i}(\ln A_i - \ln \lambda^*)\right\}, \tag{15}$$

となる．

4.2 信頼度制約を考慮したテスト労力配分問題[10],[12]

4.1の基本問題に信頼度要求を付加した次のテスト労力配分問題を考える．

(1)〜(3) 4.1の基本問題と同一．

(4) 単体テスト終了後の各モジュールの信頼度の達成値が最低限$R_0(0 < R_0 \le 1)$以上であるという制約の下で，ソフトウェア内の残存フォールト数をできるだけ少なくするために，所与の計画値Qを各モジュールに適切に配分する．

以上のテスト労力量の最適配分問題を定式化すると，式(5)を用いて

$$\text{minimize} \sum_{i=1}^{M} w_i a_i \cdot \exp[-r_i q_i], \tag{16}$$

$$\text{subject to} \sum_{i=1}^{M} q_i \le Q, q_i \ge 0 (i=1,2,\cdots,M), \tag{17}$$

$$R(s) = \exp[-\gamma_i a_i \cdot \exp[-r_i q_i] s] \ge R_0, \tag{18}$$

© 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．

となる $(i=1,2,\cdots,M)$．つまり，式(17)および式(18)の制約条件式の下で，目的関数である式(16)を最小化する各モジュールへの最適テスト労力配分量 $q_i^*(i=1,2,\cdots,M)$ を求めることが目的となる．

式(18)の制約条件式を変形すると，

$$q_i \geq -\frac{1}{r_i}\ln\left[-\frac{\ln R_0}{\gamma_i a_i s}\right] (i=1,2,\cdots,M), \tag{19}$$

となる．ここで，式(19)の右辺を d_i とおく．この d_i は，モジュール i に対して信頼度の達成値を最低限維持するために費やされるテスト労力量である．$e_i = \max\{0, d_i\}$ とおくと，式(17)および式(18)の制約条件式は，

$$\sum_{i=1}^{M} q_i \leq Q, \tag{20}$$

$$q_i \geq e_i \qquad (i=1,2,\cdots,M), \tag{21}$$

となる．ここで，$x_i = q_i - e_i$ と置き直せば，式(16)-式(18)は，

$$\text{minimize} \sum_{i=1}^{M} w_i a_i \cdot \exp[-r_i e_i]\exp[-r_i x_i], \tag{22}$$

subject to

$$\sum_{i=1}^{M} x_i \leq Q - \sum_{i=1}^{M} e_i,\ \ x_i \geq 0\ (i=1,2,\cdots,M), \tag{23}$$

となる．これは，次のような変換を行えば，4.1の基本問題に帰着できる．

$$\left.\begin{array}{l} Q \leftarrow Q - \sum_{i=1}^{M} e_i \\ w_i \leftarrow w_i \cdot \exp[-r_i e_i] \\ q_i \leftarrow x_i \end{array}\right\}. \tag{24}$$

4.3 コスト評価基準を考慮したテスト労力配分問題[13],[14]

フォールトの発見・修正のために投入されるテスト労力のコスト要因を考慮して，次のテスト労力配分問題を考える．

(1)〜(3) 4.1の基本問題と同一．

(4) 単体テスト終了後の各モジュールの信頼度の達成値が最低限 $R_0(0 < R_0 \leq 1)$ 以上であるという制約の下で，総期待ソフトウェアコストを最小にするために，所与の計画値 Q を各モジュールに適切に配分する．

この最適配分問題を解くために，3.の最適リリース問題におけるコストパラメータ c_3 は単位テスト労力量当りのコストとする．このとき，式(6)から，

$$\begin{aligned} C(T) &= c_1 a(1-\exp[-rW(T)]) \\ &\quad + c_2 a \cdot \exp[-rW(T)] + c_3 W(T), \end{aligned} \tag{25}$$

となる．式(25)を用いて，各モジュールに対して投入されるテスト労力と，各モジュールのテストに必要な期待コストとの関係を定式化し，ソフトウェアシステムの単体テストに対する総期待ソフトウェアコストを導出する．そこで，各モジュール $i(i=1,2,\cdots,M)$ に配分されるテスト労力量 q_i を用いると，各モジュール i の単体テストにおける総期待コストは

$$v_i = c_1 a_i(1-\exp[-r_i q_i]) + c_2 a_i \cdot \exp[-r_i q_i] + c_3 q_i, \tag{26}$$

により与えられる．ここで，コストパラメータの意味は次のように理解することになる．

c_1 :単体テスト工程において発見されるフォールト1個当りの修正コスト

c_2 :単体テスト工程で未発見となった（すなわち，次のテスト段階である統合テストに持ち越される）フォールト1個当りの修正コスト $(c_2 > c_1 > 0)$

c_3 :単体テストの作業に必要な単位テスト労力当りのコスト $(c_3 > 0)$

以上のことから，総期待ソフトウェアコストに基づいて，信頼度制約下におけるテスト労力の最適配分問題を定式化すると

$$\begin{aligned} \text{minimize}\ \ & c_1 \sum_{i=1}^{M} a_i(1-\exp[-r_i q_i]) \\ & + c_2 \sum_{i=1}^{M} a_i \cdot \exp[-r_i q_i] + c_3 \sum_{i=1}^{M} q_i, \end{aligned} \tag{27}$$

$$\text{subject to} \sum_{i=1}^{M} q_i \leq Q, q_i \geq 0 (i=1,2,\cdots,M), \tag{28}$$

$$R(s) = \exp[-\gamma_i a_i \cdot \exp[-r_i q_i]s] \geq R_0 \quad (i=1,2,\cdots,M), \tag{29}$$

となる．つまり，式(28)および式(29)の制約条件式の下で，式(27)の目的関数を最小化することが目的となる．4.2と同様にして，d_i（=式(19)の右辺）および $e_i(=\max\{0,d_i\})$ $(i=1,2,\cdots,M)$ を用いると，上記の配分問題は次のように書き直せる．

$$\begin{aligned} \text{minimize}\ \ & c_1 \sum_{i=1}^{M} a_i(1-\exp[-r_i e_i]\exp[-r_i x_i]) \\ & + c_2 \sum_{i=1}^{M} a_i \cdot \exp[-r_i e_i]\exp[-r_i x_i] + c_3 \sum_{i=1}^{M}(x_i + e_i), \end{aligned} \tag{30}$$

© 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．

subject to

$$\sum_{i=1}^{M} x_i \leq Q - \sum_{i=1}^{M} e_i, \tag{31}$$

$$x_i \geq 0 \quad (i = 1, 2, \cdots, M). \tag{32}$$

以上のことから，式 (31) および式 (32) の制約条件の下で，式 (30) の目的関数を最小にする最適化問題を考えればよく，4.1 の基本問題と同様にして解くことができる．

5. むすび

本稿で議論した非線形計画問題 (文献 [15] も参照されたい) として定式化されたテスト労力配分問題以外にも，想定するテスト工程での管理的問題に応じて，目的関数と制約式を適当に設定して種々の問題が考えられる [16],[17]．また，単体テストやコンポーネントレベルのテストの次段階である統合テストや総合テストに応用できる問題に拡張していく必要がある．

参考文献

[1] 松本正雄, 小山田正史, 松尾谷徹 : "ソフトウェア開発・検証技法", 電子情報通信学会, 東京 (1997).

[2] 山田茂, 大寺浩志 : "ソフトウェアの信頼性～理論と実践的応用～", ソフト・リサーチ・センター, 東京 (1990).

[3] Yamada, S., Ohtera, H. and Narihisa, H. : "Software reliability growth models with testing-effort", *IEEE Trans. Reliability*, Vol.R-35, No.1, pp.19-23 (1986).

[4] Lyu, M. R. (ed.) : "*Handbook of Software Reliability Engineering*", IEEE Computer Society Press, Los Alamitos, CA (1995).

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

[6] 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).

[7] Okumoto, K. and Goel, A. L. : "Optimum release time for software systems based on reliability and cost criteria",*J. Systems and Software*, Vol.1, No.4, pp.315-318 (1980).

[8] 山田茂, 木村光宏 : "ソフトウェア保守コストモデルにおける最適リリース問題", オペレーションズ・リサーチ, Vol.40, No.7, pp.353-358 (1995).

[9] Ohtera, H. and Yamada, S. : "Optimal allocation and control problem for software testing-resources", *IEEE Trans. Reliability*, Vol.R-39, No.2, pp.171-176 (1990).

[10] Yamada, S., Ichimori, T. and Nishiwaki, M. : "Optimal allocation policies for testing-resource based on a software reliability growth model", *Mathematical and Computer Modelling*, Vol.22, No.10-12, pp.295-301 (1995).

[11] Bazaraa, M. S. and Shetty, C. M. : "*Nonlinear Programming: Theory and Algorithms*", John Wiley & Sons, New York (1979).

[12] 西脇正敏, 山田茂, 一森哲男 : "高信頼性ソフトウェア開発におけるテスト資源の最適配分問題", 日本経営工学会誌, Vol.45, No.4, pp.325-329 (1994).

[13] 田中正敏, 山田茂, 一森哲男 : "納期を考慮したソフトウェア最適リリース問題に基づく最適テスト労力配分方策", 日本応用数理学会論文誌, Vol.5, No.3, pp.97-107 (1995).

[14] Nishiwaki, M., Yamada, S. and Ichimori, T. : "Testing-resource allocation policies based on an optimal software release problem", *Mathematica Japonica*, Vol.43, No.1, pp.91-97 (1996).

[15] Ibaraki, T. and Katoh, N. : "*Resource Allocation Problems ; Algorithmic Approaches*", MIT Press, London (1988).

[16] Czuchra, W. : "Software development decisions based on simple resource allocation algorithms", *Joint Proceedings of Wyzsza Szkola Morska Gdynia and Hochschule Bremerhaven*, No.10, pp.13-24 (1997).

[17] Kubat, P. and Koch, H. S. : "Managing test procedures to achieve reliable software", *IEEE Trans. Reliability*, Vol.R-32, No.3, pp.299-303 (1983).

© 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．