数理解析研究所講究録
第1734巻 2011年 272-279

組込みOSSに対する信頼性/移植性評価ツールによる最適ソフトウェアリリース問題

山口大学大学院・理工学研究科 中道 徹 (Toru Nakamichi) †
山口大学大学院・理工学研究科 田村 慶信 (Yoshinobu Tamura) †
†Graduate School of Science and Engineering, Yamaguchi University
鳥取大学大学院・工学研究科 山田 茂 (Shigeru Yamada) ††
††Graduate School of Engineering, Tottori University

1 はじめに

現在の情報化社会では，車載情報システム，金融機関のオンラインシステム，医療機関のコンピュータによる個人情報の管理などのように多種多様な分野にわたってコンピュータに依存しており，システムにおいて，ひとたび障害や欠陥が表面化すると，日常生活に多大な損害を招くことになったり，人命に関わる事故を引き起こしたりする．特に，ネットワーク環境を利用して開発されるオープンソースソフトウェア (Open Source Software, 以下OSSと略す) は，世界中の誰もが開発に参加でき，ソースコードが公開され，誰でも自由に改変可能なソフトウェアであることから，組込みシステムやサーバ用途として広く採用され，急激に普及しつつある [1,2]．また，オープン規格やOSSを利用することによって，電子行政機関がプライバシーや個人の自由を保護するとともに，市民が電子政府と情報をやり取りできるようにするのに役立つことから，EU加盟国を中心に欧米においても政府関係機関がOSSを支持する動きが広がっている [3]．

OSSの開発環境を考えた場合，ユーザの使用により不具合が確認されるとバグトラッキングシステム上に不具合内容が報告され，その内容に基づきソースコードの修正作業を開発者が行い，修正されたOSSを再度，公表・配布するという開発サイクルで成り立っている．このように，OSSでは開発から運用保守におよぶ工程においてソフトウェアの品質・信頼性を評価するという試みが行われていなかった．オープンソース・プロジェクトのメンバー構成と動機付けの仕組みを考えた場合，中心にコアがあり，それを複数の周辺レベルが互いに混ざり合って取り囲む構造になっている．特に，開発ボランティアは，コア開発者と周辺開発者に分類される．最重要のメンバーは中心的なソフトウェアを担当する開発者であり，彼らは中心的なメーリングリストにアクセス可能となっている．従来のソフトウェア開発工程は，要求仕様定義，設計，コーディング，テスト，運用・保守という典型的なウォーターフォールモデルのように，なんらかの開発モデルが存在していた．しかしながら，オープンソースプロジェクトは，従来のソフトウェア開発プロセスとは異なり，テスト工程が存在しないという特徴をもつ [4–7]．近年，組込み向けOSSは， Android [8] などのパーソナルユースを意識したソフトウェアの登場により一層注目を集めつつある．特に，Androidでは開発環境を公開しており，今後のシステム開発やアプリケーション開発の活性化が期待されている．

一方，OSSを利用した組込み機器の開発にあたってはいくつかの問題点もある．1つ目は品質上の問題である．さらに，2つ目はサポート上の不安である．また，現在公開されているOSSを企業組織で開発された基板上に移植する際において，その移植作業が成功するか否かが不透明であるといった問題も挙げられる．これらの問題は今後組込みOSSが普及していく上での課題の1つである．

本論文では，組込みシステム上に組込みOSSを導入する移植工程における信頼性および移植性評価法を提案するとともに，本手法の適用可能性について考察する．特に，オブジェクト指向言語であるJavaを用いて信頼性・移植性を評価するツールを設計・開発し，実際のフォールトデータに対するツールの実行例を示す．さらに，開

発管理者やユーザがOSS移植工程における進捗状況を把握し易いように，信頼性・移植性評価結果だけではなく，ソフトウェア信頼度を同時に考慮した場合における最適リリース問題としての機能を追加し，その実行例を示す．

2 組込みOSSの移植工程に対する信頼性評価法

2.1 モデルの記述

これまでにも，数多くのソフトウェア信頼度成長モデルが提案されている [9–11]．本論文では，比較的構造も単純であり，従来から古典的なモデルとして知られているハザードレートモデルを適用する [12–16]．ハザードレートモデルとは，ソフトウェア故障の発生現象を，ソフトウェア故障率であるハザードレートにより記述するモデルである．

本論文では，組込みOSSへの移植作業中に生じるソフトウェア故障には，以下の2種類があるものと仮定する．

A1. 組込みOSSに潜在するフォールトにより引き起こされるソフトウェア故障．

A2. 独自に開発されたソフトウェアコンポーネントに内在するフォールトにより引き起こされるソフトウェア故障．

また，1つのソフトウェア故障は1個のフォールトによって引き起こされるものと仮定し，発生したソフトウェア故障の原因となるフォールトは，上記A1またはA2のいずれかであるかは判別できないものとする．このとき，$(k-1)$ 番目と k 番目の間のソフトウェア故障発生時間間隔を確率変数 $X_k(k=1,\ 2,\ \cdots,\ )$ とすると，X_k に対するハザードレートは，式(1)-(3)により表すことができるものと仮定する．

$$z_k(x) = p\cdot z_k^1(x)+(1-p)\cdot z_k^2(x) \qquad (k=1,\ 2,\ \cdots;\ 0\le p\le 1), \tag{1}$$

$$z_k^1(x) = D\{1-\alpha\cdot\exp(-\alpha k)\}^{k-1} \qquad (k=1,\ 2,\ \cdots;\ -1<\alpha<1,\ D>0), \tag{2}$$

$$z_k^2(x) = \phi\{N-(k-1)\} \qquad (k=1,\ 2,\ \cdots;\ N>0,\ \phi>0). \tag{3}$$

ここで，各諸量を以下のように定義する．

$z_k^1(x)$: A1に対するハザードレート，
α : OSSの活動状態を表す形状パラメータ，
D : 1番目のソフトウェア故障に対する初期ハザードレート，
p : 移植作業全体に対する組込みOSSの開発労力の割合，
$z_k^2(x)$: A2に対するハザードレート，
N : 独自に実装するソフトウェアコンポーネント内に潜在する総固有フォールト数，
ϕ : 独自に実装するソフトウェアコンポーネントに対する固有フォールト1個当りのハザードレート．

2.2 信頼性評価尺度

移植作業の際の動的実行環境において $(k-1)$ 番目と k 番目の間のソフトウェア故障発生時間間隔を表す $X_k(k=1,\ 2,\ \cdots,\ )$ の分布関数は，

$$F_k(x) \equiv \mathrm{P_r}\{X_k\le x\} \qquad (x\ge 0), \tag{4}$$

により定義され，時間区間 $(0,\ x]$ でのソフトウェア故障の発生する確率を表す．ここで，$\Pr\{\mathrm{A}\}$ は事象Aの生起確率を表す．したがって，$F_k(x)$ の導関数

$$f_k(x) = \frac{dF_k(x)}{dx}, \tag{5}$$

は，X_k の確率密度関数である．また，時間区間 $(0, x]$ において，ソフトウェア故障の発生しない確率を表すソフトウェア信頼度 $R_k(x)$ は

$$R_k(x) \equiv \mathrm{P_r}\{X_k > x\} = 1 - F_k(x), \tag{6}$$

により定義される．式 (4) および式 (5) から，時間区間 $(0, x]$ においてソフトウェア故障が発生していないときに，引き続く単位時間内にソフトウェア故障が発生する割合を意味するソフトウェア故障率 (ハザードレート) を

$$z_k(x) \equiv \frac{f_k(x)}{1 - F_K(x)} = \frac{f_k(x)}{R_k(x)}, \tag{7}$$

により与えることができる．

したがって，式 (1) のハザードレートモデルから，信頼性評価尺度を導出することができる．確率密度関数は

$$\begin{aligned} f_k(x) &= \{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + (1-p)\phi(N-k+1)\} \\ &\cdot \exp[-\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + (1-p)\phi(N-k+1)\}\cdot x], \end{aligned} \tag{8}$$

となる．また，ソフトウェア信頼度は，

$$R_k(x) = \exp[-\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + (1-p)\phi(N-k+1)\}\cdot x], \tag{9}$$

と表すことができる．さらに，式 (8) から，X_k の平均値すなわち k 番目のソフトウェア故障に対する平均ソフトウェア故障時間間隔（Mean Time between Software Failures，以下 MTBF を略す）は，

$$\mathrm{E}[X_k] = \frac{1}{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + (1-p)\phi(N-k+1)}, \tag{10}$$

により与えられる．

3 移植工程における総期待ソフトウェアコストの定式化

移植作業時における総期待ソフトウェアコストを，既存のソフトウェア最適リリース問題 [17] に基づき定式化し，総期待ソフトウェアコストを最小にする時刻を最適リリース時刻と定義する．まず，総期待ソフトウェアコストを定式化するために，以下のパラメータを定義する．

c_1 ：移植作業中における単位時間当りのテストコスト $(c_1 > 0)$,

c_2 ：移植作業中におけるフォールト 1 個当りの修正コスト $(c_2 > 0)$,

c_3 ：リリース後のフォールト 1 個当りの修正コスト $(c_3 > c_3)$.

よって，以下のような期待ソフトウェアコストが得られる．

$$C_1(l) = c_1 \sum_{k=1}^{l} \mathrm{E}[X_k] + c_2 l. \tag{11}$$

ここで，l は l 番目のソフトウェア故障発生回数を表す．一方，リリース後の保守コストは以下のように定式化できる．ただし，$N > l$ と仮定する．

$$C_2(l) = c_3(N-l). \tag{12}$$

したがって，総期待ソフトウェアコストは，式 (11) および式 (12) より，

$$C_3(l) = C_1(l) + C_2(l), \tag{13}$$

のように表すことができる．この式 (13) を最小にする $l = l^*$ から最適リリース時刻 $\sum_{k=1}^{l^*} \mathrm{E}[X_k]$ を求めることができる．

4 ツールの開発

本ツールの要求仕様を以下に示す．また，本ツールのアクティビティ図を図 1 に示す

1. OSS に対する信頼性・移植性評価に使用するデータはバグトラッキングシステムから採取された実測データを用いる．
2. バグトラッキングシステムから採取された実測データに基づいて信頼性評価を行い，各推定結果をグラフで表示する．
3. 提案されたハザードレートモデルに含まれる未知パラメータを推定するために，最尤法を適用する．
   システム全体に対する信頼性評価のために適用するモデルはハザードレートモデルを用いる．
4. 信頼性・移植性評価尺度として，MTBF，ソフトウェア信頼度，OSS の重要度，Laplace Trend Test [18,19] を用いる．
5. ソフトウェア信頼度を同時に考慮した場合における総期待ソフトウェアコストをグラフ表示する．
6. 推定値と実測値の適合性評価のために予測相対誤差を用いる．
7. 全てのグラフは同時に表示する．
8. 推定結果を HTML ファイル，テキストファイル，JPEG ファイルとして出力する．
9. ツールの操作には GUI を使用し，マウスを用いて行う．
10. ツールの開発言語に Java を使用する．
11. グラフの描画には JFreeChart を使用する．
12. 数値計算の簡略化のため，以下のように定義する．

$$w_1 = pD, \tag{14}$$

$$w_2 = (1-p)\phi. \tag{15}$$

5 ツールの実行例

本論文では，携帯電話用 OS として開発・公開されている Android [8] 上で BusyBox [20] が動作するシステムを構築する環境を想定し，Android が A1 に対するソフトウェア故障を，BusyBox が A2 に対するソフトウェア故障を表すものと仮定する．移植作業工程を想定するために，実際の Android および BusyBox のオープンソースプロジェクトにおけるバグトラッキングシステム上に登録されたフォールトデータを適用した数値例を示す．Android は携帯電話用 OS として知られ，BusyBox はテレビ，オーディオ，ブロードバンドルータ，小型サーバなど，家電製品を代表とした様々な組込み製品に利用されている．

本論文では，Android 1.5 NDK, Release 1 以降のデータを採用し，BusyBox については， BusyBox 1.10.1 (stable) 以降のデータを適用した数値例を示す．

5.1 MTBF

式 (10) の MTBF の推定結果を図 2 に示す．ここで，横軸はソフトウェア故障数を表し，縦軸はソフトウェア故障発生時間間隔を表す．図 2 から，フォールト発見数が増加するにつれ，MTBF が増加し信頼度成長が起こっている様子が確認できる．

5.2 最適リリース時刻の推定

ソフトウェア信頼度を同時に考慮した場合における最適リリース時刻の数値例を示す．まず，目標となるソフトウェア信頼度を以下のように仮定する．

$$R_k(1) = 0.5.$$

目標となるソフトウェア信頼度を0.5とした場合における総期待ソフトウェアコストを図3に示す．最適リリース時刻 $t^* = \sum_{k=1}^{51} \mathrm{E}[X_k] = 26.549$ の場合におけるソフトウェア信頼度は $R_{51}(1) = 0.41965$ となり，テストを延長する必要があることが分かる．したがって，図3から，目標となるソフトウェア信頼度 $R_k(1) = 0.5$ を考慮した場合における最適リリース時刻は $\sum_{k=1}^{55} \mathrm{E}[X_k] = 32.072$ となる．したがって，移植作業期間を $(t' - t^*) = 32.072 - 26.549 = 5.5238$ 日間延長する必要があることが分かる．また，そのときの総期待開発コストは4322.8であることが確認できる．

6 おわりに

本論文では，ハザードレートモデルに基づく組込向けOSSに対する信頼性・移植性評価ツールを開発した．また，本ツールの実行例として，実際のフォールトデータに対する信頼性・移植性評価結果を示した．本ツールにより，組込みシステム開発の移植工程に対して，品質上における何らかの指標を得ることができるものと考える．

組込みOSSを利用した組込みシステム開発においては，移植作業が成功するか否かが，組込み製品が出荷できるかどうかに直接的に関係してくることから，組込みシステムの開発工程の中でも移植工程を適切に管理することは非常に重要となる．特に，組込みOSSのソフトウェア故障発生時間間隔データに関しては，ソフトウェア故障発生数が多くなるにつれてMTBFが増加するという傾向があるものとそうでないものとが存在するため，それに応じた適切なハザードレートモデルを選択する必要がある．本論文では，組込みOSSに対するハザードレートモデルを適用した．さらに，信頼性・移植性評価結果を一覧表示することにより，移植工程の開発管理者およびユーザにとって，現在の進捗状況を容易に把握することが可能となり，移植工程のプロジェクト管理に役立つものと考える．

特に，組込みシステム開発の移植作業工程において，ある程度目安となるような適切な移植作業期間を推定することは，リリース後の信頼性維持や進捗度管理に役立つと考えられる．これまでにも，一般的な企業組織において開発されたソフトウェアシステムを対象としたソフトウェアの最適リリース問題 [11] が数多く提案されている．OSSを利用した組込みシステム開発においても，移植作業工程における最適リリース問題として総期待ソフトウェアコストを定式化することにより，最適な移植作業期間を決定することができるものと考える．本論文では，移植作業工程に対するソフトウェア信頼度を同時に考慮した最適リリース問題をツール化した．これにより，数理モデルに関する知識がなくとも，実際のソフトウェア開発者が迅速に最適リリース時刻および総期待ソフトウェアコストを推定することが可能となる．

謝辞

本研究の一部は，文部科学省科学研究費基盤研究 (C)（課題番号 22510150）および若手研究 (B)（課題番号 21700044）の援助を受けたことを付記する．

参考文献

[1] The Apache HTTP Server Project, The Apache Software Foundation, http://httpd.apache.org/

[2] Mozilla.org, Mozilla Foundation, http://www.mozilla.org/

[3] ソフトウェア情報センター研究会報告書, オープンソースソフトウエアの利用状況調査／導入検討ガイドラインの公表について, 東京, 2004.

[4] P. Li, M. Shaw, J. Herbsleb, B. Ray, and P. Santhanam, Empirical evaluation of defect projection models for widely-deployed production software systems, Proceedings of the 12th International Symposium on the Foundations of Software Engineering (FSE-12), 2004, pp. 263–272.

[5] J. Norris, Mission-critical development with open source software, IEEE Software Magazine, vol. 21, no. 1, pp. 42–49, 2004.

[6] Y. Tamura and S. Yamada, Software reliability assessment and optimal version-upgrade problem for open source software, Proceedings of the 2007 IEEE International Conference on Systems, Man, and Cybernetics, Montreal, Canada, October 7–10, 2007, pp. 1333–1338.

[7] Y. Tamura and S. Yamada, A method of user-oriented reliability assessment for open source software and its applications, Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, Oct. 8–11, 2006, pp. 2185–2190.

[8] Open Handset Alliance, Android,http://www.android.com/

[9] M.R. Lyu, ed., Handbook of Software Reliability Engineering, IEEE Computer Society Press, Los Alamitos, CA, 1996.

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

[11] 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.

[12] G.J. Schick and R.W. Wolverton, An Analysis of Competing Software Reliability Models, IEEE Transactions on Reliability Engineering, SE–4 (2), pp. 104–120, 1978.

[13] Z. Jelinski, P.B. Moranda, Software Reliability Research, in Statistical Computer Performance Evaluation, Freiberger, W.(ed.), pp. 465–484, Academic Press, New York, 1972.

[14] P.B. Moranda, Event–altered rate models for general reliability analysis, IEEE Transactions on Reliability, vol. R–28, no. 5, pp. 376–381, 1979.

[15] M. Xie, On a generalization of the J-M model, Proceedings of the Reliability '89, 1989, pp. 5Ba/3/1-5Ba/3/7.

[16] Y. Zhoum, J. Davis, Open source software reliability model: an empirical approach, Proceedings of the Workshop on Open Source Software Engineering (WOSSE), 2005, pp. 67–72.

[17] S. Yamada and S. Osaki, Cost-reliability optimal release policies for a software system, IEEE Trans. Reliability, vol. R-34, no. 5, pp. 422-424, Dec. 1985.

[18] P.A. Keiler and T.A. Mazzuchi, Enhancing the predictive performance of the Goel-Okumoto software reliability growth model, Proceedings of the Annual Reliability and Maintainability Symposium, 2000, pp. 106–112.

[19] V. Almering, M.V. Genuchten, G. Cloudt, and P.J.M. Sonnemants, Using software reliability growth models in practice, IEEE Software Magazine, vol.24, no 6, pp. 82–88, 2007.

[20] Erik Andersen, BUSYBOX, http://www.busybox.net/

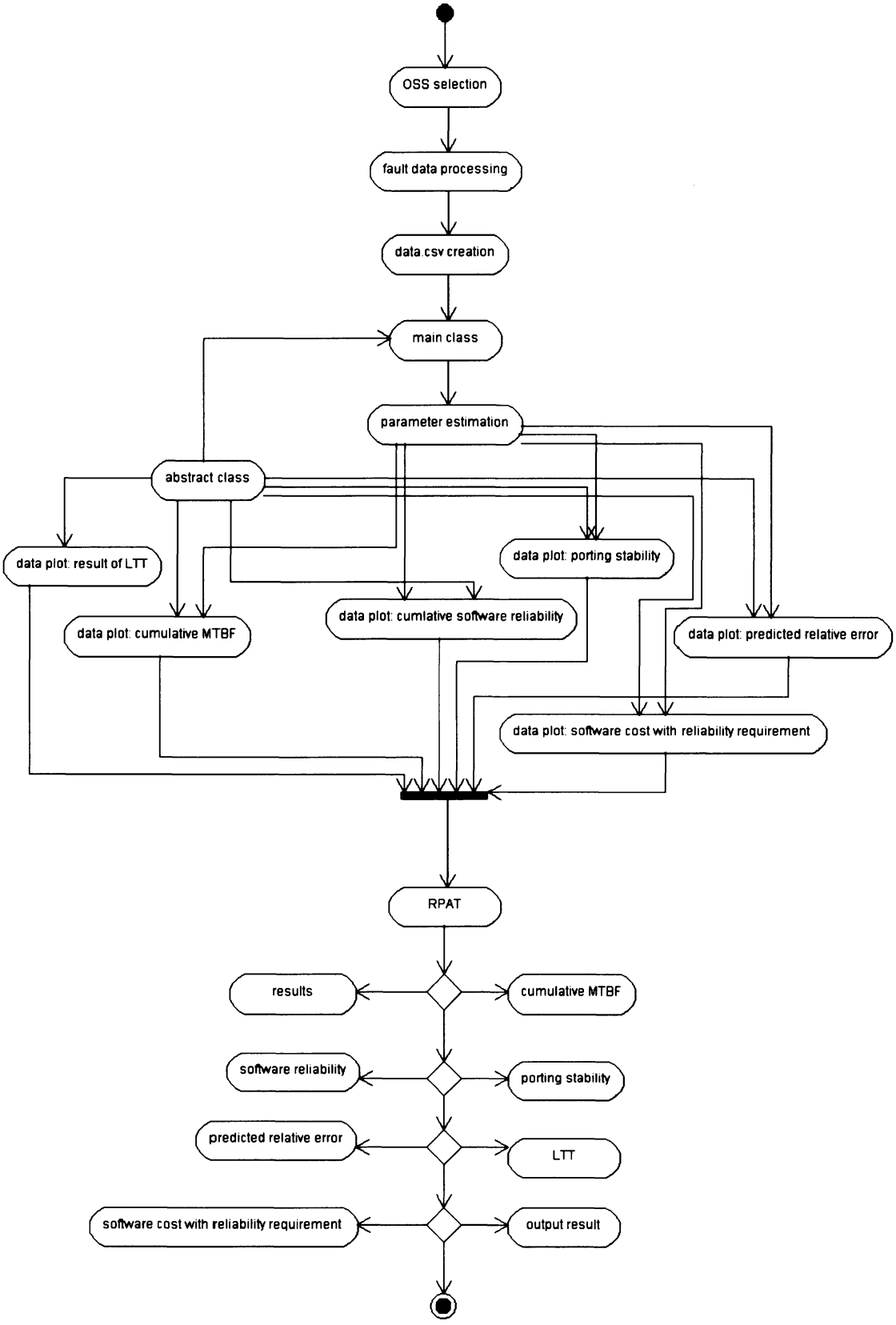

図 1： 本ツールのアクティビティ図.

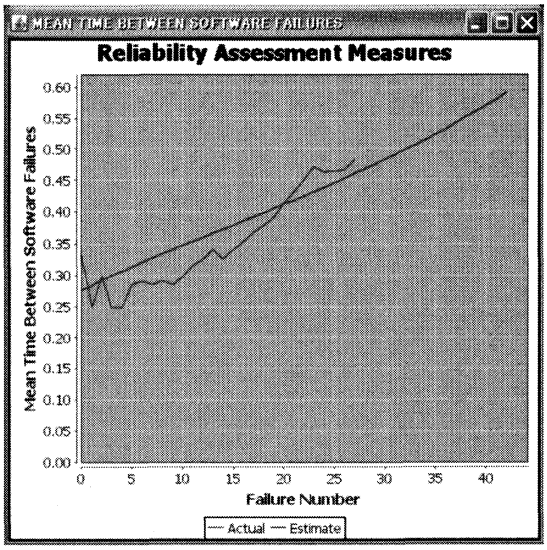

図 2： MTBF の推定結果.

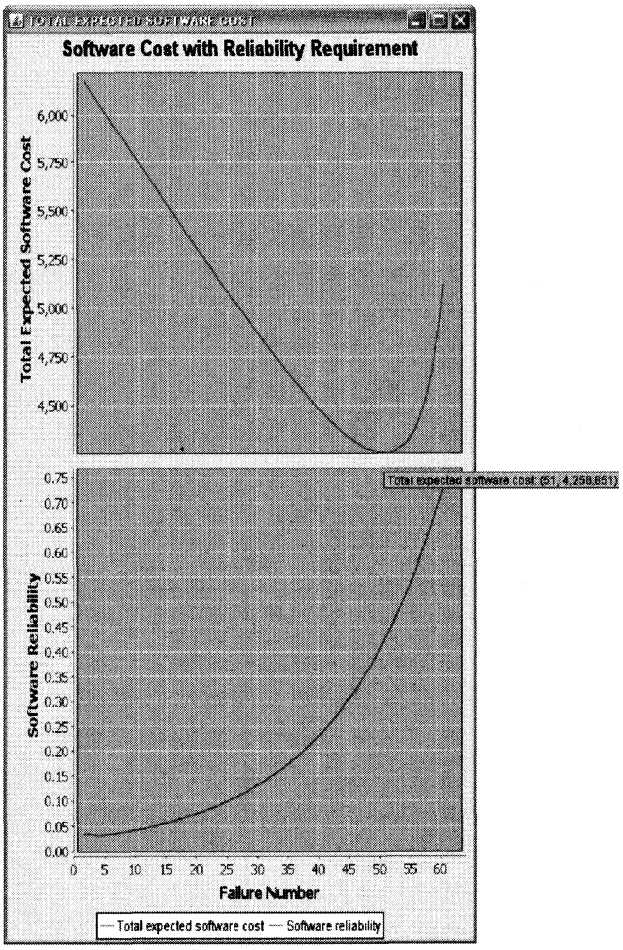

図 3： ソフトウェア信頼度を同時に考慮した場合における総期待ソフトウェアコストの推定結果.