数理解析研究所講究録
第1682巻 2010年 209-216

組込みオープンソースソフトウェアに対するハザードレートモデルに基づく移植性評価法

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

現在のソフトウェア開発環境は，分散共同開発が主流であり，同一企業内における開発形態から，複数のソフトウェアハウスや同一企業内，複数の企業間での遠隔地間共同開発，さらには，多くの開発者が協調しながら開発を行うオープソース・プロジェクトなどの様々な形態が存在する [1]．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，急激に普及が広まっている [2, 3]．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 [4]．

一方で，OSSの利用に関しては，OSSの普及を妨げる大きな要因として，サポートや品質上の問題が指摘されている．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．この指導的集団を形成している主要メンバーが主導的にテスト進捗度管理技術を導入することによって，より高品質なOSSの開発に結びつくものと考える．特に，一般企業において実践されているテスト進捗度管理技術をOSSの開発サイクルに組み込むことによって，従来よりもより高品質なOSSの開発が可能となると思われる．

また，最近のOSSの傾向として，組込み機器に対してもAndroid[5]やBusyBox[6]に代表される組込みOSSが積極的に採用されつつある．OSSの信頼性評価に関する特徴として，サーバおよびアプリケーションソフトウェアについては信頼度成長曲線に関して一定の傾向を示すものが多いが [7, 8]，こうした組込みソフトウェアについては，ハードウェアに依存するコンポーネントが含まれていることから，信頼性を評価することが難しくなってくる．

従来から，ソフトウェア製品の開発プロセスにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を

行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その代表的かつ古典的モデルの1つとして，ハザードレートモデルがある [9, 10, 11, 12, 13].

本論文では，こうしたオープンソースプロジェクトの下で開発されている組込みOSSを採用する際の移植作業工程に対する移植性評価法を提案する．また，実際に公開されている組込みOSSを採用した移植作業工程における信頼性評価の適用可能性について考察する．これにより，組込みOSSの普及を妨げる大きな要因として考えられている品質上の問題に対して，信頼性という観点からなんらかの定量的指標を提示することが可能となるものと考える．さらに，実際の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に基づく信頼性評価例を示す.

2 組込みOSSの移植工程に対するハザードレートモデル

本論文では，組込みOSSのポーティングにおける動的実行環境，すなわち独自に開発されたハードウェアに対する組込みOSSの移植作業中に生じるソフトウェア故障には，次の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り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区別はできないものとする．ここで，確率 p でA1を，確率 $(1-p)$ でA2のソフトウェア故障が発生するものとする．このとき，確率変数 X_k $(k=1,2,\cdots)$ により，$(k-1)$ 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ものとすると，X_k に対するハザードレートは，

$$z_k(x) = p\cdot z_k^1(x) + (1-p)\cdot z_k^2(x) \quad (1)$$
$$(k=1,\ 2,\ \cdots;\ 0\le p\le 1),$$
$$z_k^1(x) = D(1-\alpha\cdot e^{-\alpha k})^{k-1} \quad (2)$$
$$(k=1,\ 2,\ \cdots;\ -1<\alpha<1,\ D>0),$$
$$z_k^2(x) = \phi\{N-(k-1)\} \quad (3)$$
$$(k=1,\ 2,\ \cdots,\ N;\ N>0,\ \phi>0),$$

により表すことができるものと仮定する．ここで，各諸量を次のように定義する.

$z_k^1(x)$: A1に対するハザードレート,
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,
α : OSSの活動状態を表す形状パラメータ,
$z_k^2(x)$: A2に対するハザードレート,
N : 複数のコンポーネント内に潜在する総固有フォールト数,
ϕ : 複数のコンポーネント内における固有フォールト1個当りのハザードレート,
p : $z_k^1(x)$ に対する重みパラメータ.

式 (1) は，組込み OSS 内に潜在する総固有フォールトおよび独自に開発された複数のコンポーネントに内在するフォールトによるソフトウェア故障発生現象を，発生割合を表す p により陽に記述するものである．

本モデルに含まれる式 (2) は，既存の Moranda モデル [12] を組込み OSS の開発環境に合わせて修正したものであり，式 (3) は既存の Jelinski-Moranda(J-M) モデル [11] を表す．特に，式 (2) は，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が OSS の活動状況に応じて幾何級数的に減少するとともに，OSS の活動状態が指数関数的に増加するものと仮定している．

3 信頼性評価尺度

ポーティング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，

$$F_k(x) \equiv \Pr\{X_k \leq x\} \quad (x \geq 0), \tag{4}$$

により定義され，時間区間 $(0,\ x]$ で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

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

は，X_k の確率密度関数である．また，時間区間 $(0,\ x]$ で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は，

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

により定義される．式 (4) および式 (5) から，時間区間 $(0,\ x]$ で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 [9]．

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，

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

となる．また，ソフトウェア信頼度は，

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

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．

4 パラメータ推定

ポーティング作業中において，n 個のソフトウェア故障発生時間間隔 $X_k(k=1,\ 2,\ \cdots,\ n)$ が観測されたものとし，その n 個の観測データの組を $x^{(n)}=(x_1,\ x_2,\ \cdots,\ x_n)$ により表す．$x^{(n)}$ が与えられたときのモデルパラメータ p, D, α, ϕ, および N に対する対数尤度関数は，

$$\begin{aligned}\ln L &= \sum_{k=1}^{K}\ln\left[\left\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1}+(1-p)\phi(N-k+1)\right\}\right]\\ &- \sum_{k=1}^{K}\left\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1}+(1-p)\phi(N-k+1)\right\}\cdot x_k,\end{aligned}\tag{11}$$

となる．最尤推定値を求めるために，数値計算の簡略化を目的として，$pD=w_1$ および $(1-p)\phi=w_2$ とおき，w_1, w_2, α, および N について $\ln L$ をそれぞれ偏微分して得られる以下の同時対数尤度方程式を数値的に解くことにより最尤推定値 $\widehat{w_1}$, $\widehat{w_2}$, $\widehat{\alpha}$, および $\widehat{N}$ を推定することができる．

$$\frac{\partial \ln L}{\partial w_1}=\frac{\partial \ln L}{\partial w_2}=\frac{\partial \ln L}{\partial \alpha}=\frac{\partial \ln L}{\partial N}=0.\tag{12}$$

5 Laplace Trend Test

一般的に，OSS のフォールト発見過程と，企業組織の下で開発されたソフトウェアのフォールト発見過程の特徴を考えた場合，組込み OSS の移植工程では，ソフトウェア信頼度が一定して成長せず，信頼度成長と信頼度退化が同時に起こるような状況が発生することが考えられる．そのため，信頼度成長過程を定量的に評価し，慎重に進捗度管理を行う必要がある．本論文では，信頼度成長過程を比較する評価尺度として Laplace Trend Test を用いる．故障発生時間間隔データを用いた場合の Laplace Trend Test の検定統計量 $u(i)$ は

$$u(i) = \frac{\frac{1}{i-1}\sum_{n=1}^{i-1}\sum_{j=1}^{n}\theta_j-\frac{\sum_{j=1}^{i}\theta_j}{2}}{\sum_{j=1}^{i}\theta_j\sqrt{\frac{1}{12(i-1)}}},\tag{13}$$

により求めることができる [14, 15]．ここで，i は観測されたフォールト数を，θ_j は j 番目のフォールト発見時間間隔である．

6 数値例

6.1 組込み OSS

本論文では，携帯電話用 OS として開発・公開されている Android[5] 上で BusyBox[6] 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．

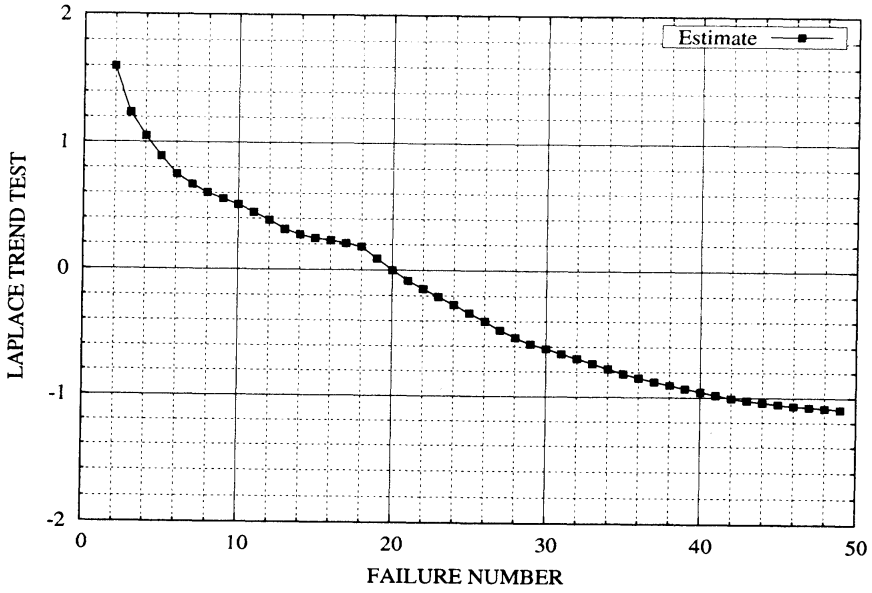

図 1： Laplace Trend Test により推定された検定統計量 $\widehat{u}(i)$.

6.2 信頼性評価

まず，本数値例で使用するデータに対して Laplace Trend Test を適用した評価結果を図 1 に示す．図 1 から，移植工程初期の段階では，信頼度が退化している様子が確認できる．一方，発見フォールト数が 20 個目以降においては，信頼度成長が起こっていることが分かる．

次に，推定された MTBF を図 2 に示す．図 2 から，フォールトが発見されるにつれて MTBF の値が増加していく，すなわち信頼度成長が起こっている様子が確認できる．さらに，ソフトウェア信頼度の推定値 $R_{30}(x)$ を図 3 に示す．図 3 から，0.25 日後の信頼度は約 0.1 であることが分かる．

6.3 移植性評価

本モデルの主要パラメータの一つである OSS の活動状態を表す形状パラメータ α を変化させた場合における推定された MTBF を図 4 に示す．図 4 から，α の値が大きくなるにつれ，信頼度が加速度的に成長する様子が確認できる．一方，α が負の値をとる場合には，平均故障発生時間間隔が小さくなり，すなわち信頼度が退化する様子が確認できる．このように，信頼度が退化する場合においては，将来的に移植作業が失敗に終わる可能性が高いことを意味する．

特に，パラメータ α が負の値をとる場合は，OSS の活動状態がプロジェクトを立ち上げたばかりの段階や，フォールト報告が非常に多くオープンソースプロジェクトが不安定な場合が想定される．また，パラメータ α が正の値をとる場合には，オープンソースプロジェクトが安定していると考えられる．

7 むすび

本論文では，オープンソースプロジェクトの下で分散共同開発されている組込み OSS の移植作業工程に対する信頼性評価法を提案した．また，実際の OSS のバグトラッキングシステムに登録されているフォールトデータに

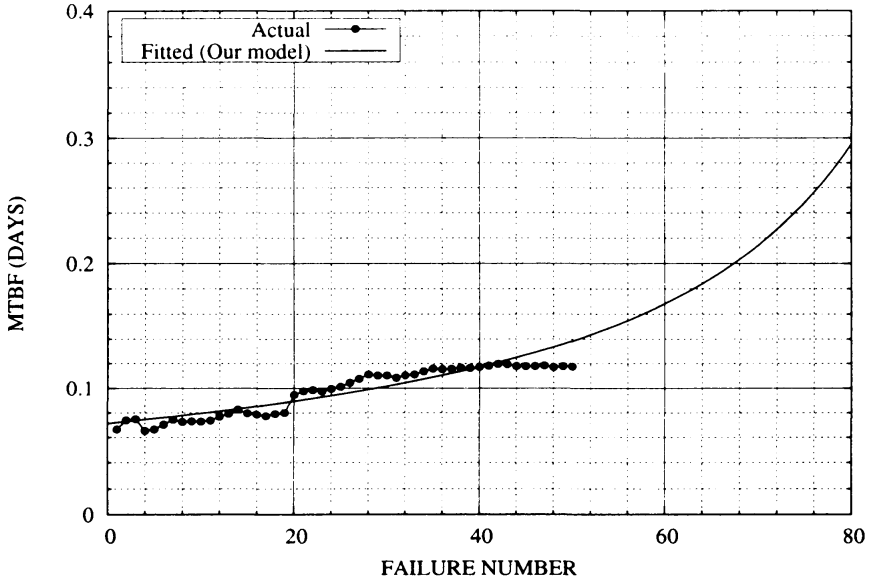

図 2： 推定された MTBF.

対して，信頼性評価尺度に関する数値例を示した．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 の故障発生時間間隔データに関しては，多くのフォールトが発見されるにつれて 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 とデバイスドライバのようなコンポーネントの 2 種類を想定したハザードレートモデルを提案した．

さらに，実際の移植作業工程を想定した数値例を示すとともに，提案されたハザードレートモデルに含まれる主要パラメータに対する感度分析結果を示した．組込み OSS が急速に普及し始めている現在，組込み OSS の信頼性に関する指標を提示することが重要であると考える．本論文で提案した信頼性評価手法を適用することにより，より高品質な組込み OSS の開発に結びつくものと考える．

組込み OSS の普及の流れを阻害する要因として，サポートや品質上の問題が挙げられる．本論文では，このような問題を解決するためにオープンソースプロジェクトの下で開発された組込み OSS の移植作業工程に対する信頼性評価法の 1 例を示した．本論文の数値例で取り上げた Android および BusyBox は，機器のネットワーク化，開発コスト削減，オープンソースといった点から組込み OS として近年注目されている．今後もオープンソースプロジェクトに基づく開発形態は急速に発展するものと考えられることから，こうした組込み OSS の信頼性および移植性評価法として利用できるものと考える．

謝辞

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

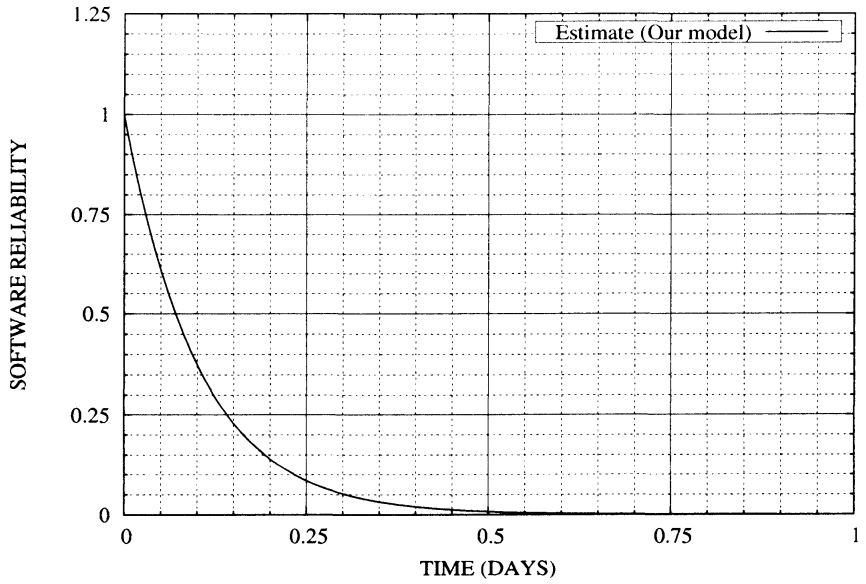

図 3： 推定されたソフトウェア信頼度.

参考文献

[1] A. Umar, *Distributed Computing and Client-Server Systems*, Prentice Hall, New Jersey, 1993.

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

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

[4] 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.

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

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

[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, pp. 2185–2190, 2006.

[8] 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, Oct. 7–10, pp. 1333–1338, 2007.

[9] 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.

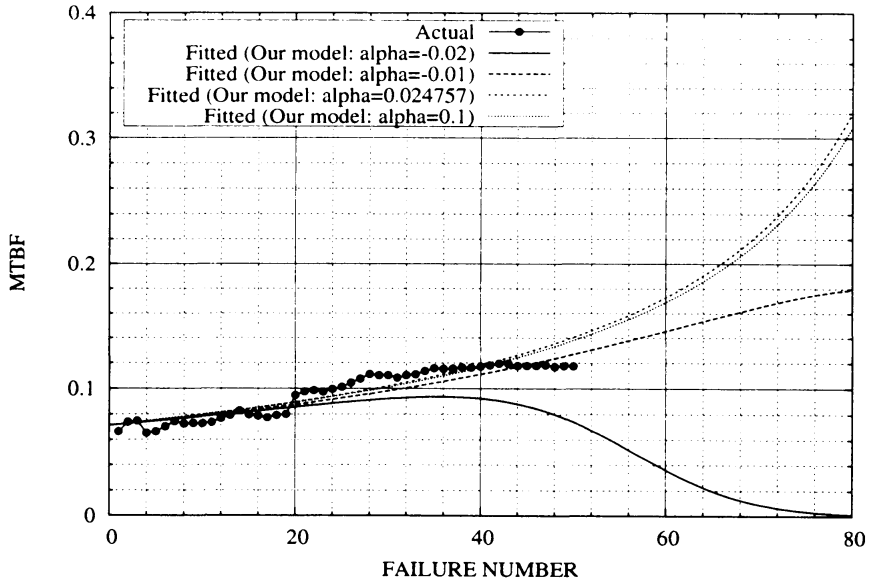

図 4： 推定された MTBF のパラメータ α に対する感度分析結果.

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

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

[12] P.B. Moranda, Event–altered Rate Models for General Reliability Analysis, IEEE Trans. Reliability, R–28 (5), pp. 376–381, 1979.

[13] M. Xie, On a Generalization of the J-M Model, Proc. Reliability '89, 5 Ba/3/1–5 Ba/3/7, 1989.

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

[15] V. Almering, M.V. Genuchten, G. Cloudt, and P.J.M. Sonnemants, Using software reliability growth models in practice, IEEE Software, Nov.–Dec. pp. 82–88, 2007.