数理解析研究所講究録
第1734巻 2011年 280-287

組込みOSSに対する一般化ハザードレートモデルに基づく開発コスト削減量に関する比較

山口大学大学院・理工学研究科 吉田 祐貴 (Yuki Yoshida) †
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

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, 誰でも自由に改変可能なソフトウェアである. そのため, 最近では組込みシステムやサーバ用途として広く採用され, 急激に普及している [1], [2]. また, OSS の最近の傾向として, 組込み機器に対しても Android[3] や BusyBox[4] などに代表される組込み OSS が積極的に採用されている.

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 [5]. 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 はウォーターフォールモデルに代表されるような一般的なソフトウェア開発モデルの下で開発が進められていないため, 開発工程には特に定められたテスト工程が存在せず, また, 開発から運用・保守に及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も行われていない.

OSS に対する現在の研究動向としては, 設計技法や開発手法, セキュリティを対象とした文献はいくつか提案されているものの [6]-[9], 動的解析に基づいた組込み OSS に対する信頼性評価に関する研究はほとんど行われていないのが現状である. さらに, 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 [10], [11], 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い.

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 [12].

本論文では, 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 (ポーティング) 工程に対する信頼性評価法を提案する. 特に, 企業組織において独自に開発された基板上へ組込み OSS を導入する際においては, 独自に開発されたデバイスドライバと組込み OSS との整合性を確認する作業も重要となる. 本論文では, こうしたソフトウェアコンポーネントと組込み OSS を同時に考慮したハザードレートモデルを提案するとともに, 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法の適用可能性について考察する. また, OSS 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とともに, 移植工程における総期待ソフトウェアコストの定式化することにより, 最適リリース問題について考察する. 特に, 総期待ソフトウェアコストに基づく OSS 導入時のコスト削減量および信頼限界の上限・下限値について議論する.

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.

2 OSS 移植工程に対する信頼性評価法

2.1 移植工程のための一般化ハザードレートモデル

本論文では，組込み OSS のポーティング時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.

ここで，ソフトウェア故障とは，ソフトウェア内に潜在するフォールトにより期待通りに動作しないことと定義する [12]. 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 A1 と A2 のどちらによるものか区別できないものとする.

ここで，A1 のソフトウェア故障は確率 p_0 で発生し，A2 のソフトウェア故障は確率 p_i で発生するものとする. このとき，$(k-1)$ 番目と k 番目のソフトウェア故障の時間間隔を確率変数 X_k とすると，X_k に対するハザードレート関数 $z_k(x)$ を，

$$z_k(x) = p_0 \cdot z_k^0(x) + \sum_{i=1}^{m} p_i \cdot z_k^i(x) \qquad (k=1,2,\cdots;0 \le p \le 1), \tag{1}$$

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

$$z_k^i(x) = \phi_i\{N_i-(k-1)\} \qquad (i=1,2,\cdots,m, k=1,2,\cdots,N_i, N_i>0, \phi_i>0), \tag{3}$$

のように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.

$z_k^0(x)$: 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_0 : 組込みソフトウェア全体に対する組込み OSS の開発労力の割合，
$z_k^i(x)$: A2 に対する i 番目のコンポーネントに対するハザードレート，
m : コンポーネント数，
N_i : i 番目のコンポーネント内に潜在する総固有フォールト数，
ϕ_i : i 番目の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，
p_i : 組込みソフトウェア全体に占める i 番目のコンポーネントの開発労力の割合.

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 i 番目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，発生割合を表すパラメータ p_0 および p_i により陽に記述するものである. p_0 および p_i には，ソースコード行数，開発コスト，開発時間に関する割合などを適用することができる. 本モデルに含まれる式 (2) は，既存の Moranda モデルを組込み OSS の開発環境に合わせて修正したものであり，組込み OSS のソフトウェア故障発生事象を表すハザードレートを意味する. また，式 (3) は既存の Jelinski-Moranda(J-M) モデルであり，独自のソフトウェアコンポーネントのソフトウェア故障発生事象を表すハザードレートを意味する. 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.

2.2 信頼性評価尺度

組込み OSS のポーティング時に検出される $(k-1)$ 番目と k 番目の間のソフトウェア故障の発生間隔を表す確率変数 X_k の分布関数は，

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

により定義され，時間区間 $(0, x]$ でソフトウェア故障が発生する確率を表す．ここで，確率 $\Pr\{A\}$ は事象 A が発生する確率を意味する．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が発生しない確率を意味する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，種々の信頼性評価尺度を導出できる．確率密度関数は，

$$\begin{aligned} f_k(x) &= \{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\cdot\phi_i(N_i - k + 1)\} \\ &\cdot \exp[-\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\cdot\phi_i(N_i - k + 1)\}\cdot x], \end{aligned} \tag{8}$$

となる．また，ソフトウェア信頼度は，

$$R_k(x) = \exp[-\{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\cdot\phi_i(N_i - k + 1)\}\cdot x], \tag{9}$$

と表すことができる．さらに式 (6) より，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 と略す）は，

$$\mathrm{E}[X_k] = \frac{1}{pD(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\cdot\phi_i(N_i - k + 1)}, \tag{10}$$

により与えられる．

3 数値例

3.1 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[3] 上で BusyBox[4] 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．

3.2 モデルパラメータの推定

モデルのパラメータ推定に際して，p_0 および p_i については，これまでに発見された累積発見フォールト数の割合を適用した．まず，Android については，Android Market introduced から Android 1.5 NDK，Release 1 までの累積発見フォールト数は 277 個であり，BusyBox に関しては，BusyBox 1.10.0 (unstable) から BusyBox

1.10.1 (stable) までの累積発見フォールト数は 21 個であった．本論文では，BusyBox の主要コンポーネントを buildroot および BusyBox と仮定し，その他の uClibc のようなコンポーネントをサブコンポーネントと仮定した場合を想定する．この場合，コンポーネント数は $m = 2$ となる．したがって，パラメータ p_0，p_1 および p_2 は以下のように与えられる．

$$\widehat{p_0} = 0.92953,\ \widehat{p_1} = 0.06040,\ \widehat{p_2} = 0.01007.$$

上述のように与えられた $\widehat{p_0}$，$\widehat{p_1}$，および $\widehat{p_2}$ から，モデルに含まれる未知パラメータ $\widehat{D}, \widehat{\alpha}, \widehat{\phi_i}, \widehat{\phi_2}, \widehat{N_1}$，および $\widehat{N_2}$ を推定する．

3.3 MTBF

推定された MTBF を図 1 に示す．図 1 より，フォールト発見数の増加とともに，MTBF の値が大きくなり，信頼度成長が起こっている様子が確認できる．

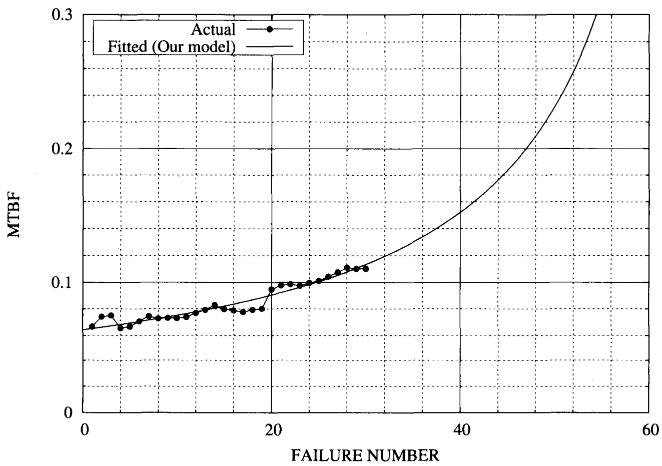


図 1： 推定された MTBF.

4 最適リリース時刻の推定

4.1 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 [13],[14] 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 ($c_3 > c_2$).

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，

$$C_2(l) = c_3 \left(\sum_{k=1}^{l} N_i - l \right). \tag{12}$$

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．

4.2 ソフトウェアコスト削減量の導出

組込み OSS を利用することによるソフトウェアコストの削減量を定量化するために，OSS の残存フォールト数を考慮した場合におけるリリース後の保守コストは以下のように定式化できる．

$$C_4(l) = c_3 \left(\frac{p_0 \sum_{i=1}^{m} N_i}{\sum_{i=1}^{m} p_i} + \sum_{i=1}^{m} N_i - l \right). \tag{14}$$

したがって，OSS の残存フォールト数を考慮した場合における総期待ソフトウェアコストは，

$$C_5(l) = C_1(l) + C_4(l), \tag{15}$$

のように表すことができる．このことから，式 (13) および式 (15) より，ソフトウェアコストの削減量は，

$$C_5(l) - C_3(l) = \frac{c_3 \cdot p_0 \sum_{i=1}^{m} N_i}{\sum_{i=1}^{m} p_i}, \tag{16}$$

となる．

4.3 最適リリース時刻およびソフトウェアコスト削減量に関する数値例

移植作業開始以降における推定された総期待ソフトウェアコストを図 2 に示す．図 2 から，最適リリース時刻の期待値は移植作業が開始されてから $\sum_{k=1}^{65} \mathrm{E}[X_k] = 12.93$ 日後であり，そのときの総期待ソフトウェアコストは 302.3 であることが確認できる．さらに，組込み OSS の残存フォールト数を考慮した場合における推定された総期待ソフトウェアコストを図 3 に示す．ここで，N_1 および N_2 のパラメータ推定結果

$$\widehat{N_1} = 98.815,\ \widehat{N_2} = 19.324,$$

と $\widehat{p_0}$, $\widehat{p_1}$, および $\widehat{p_2}$ から，組込み OSS に対する推定された残存フォールト数は約 1558 個となる．図 3 より，最適リリース時刻における総期待ソフトウェアコストは 4977.2 であることが確認できる．このことから，OSS を使用することによるコスト削減効果は 4674.9 であり，組込み OSS を利用することにより約 16 倍近くのソフトウェアコストが削減できることが分かる．

5 総期待ソフトウェアコストに対する信頼限界の上限・下限値

推定された MTBF に対する信頼限界の上限値および下限値は，

$$\left(\frac{2k}{\chi^2_{2k}\left(\frac{\epsilon}{2}\right)} \widehat{\mathrm{E}}[X_k], \frac{2k}{\chi^2_{2k}\left(1 - \frac{\epsilon}{2}\right)} \widehat{\mathrm{E}}[X_k] \right), \tag{17}$$

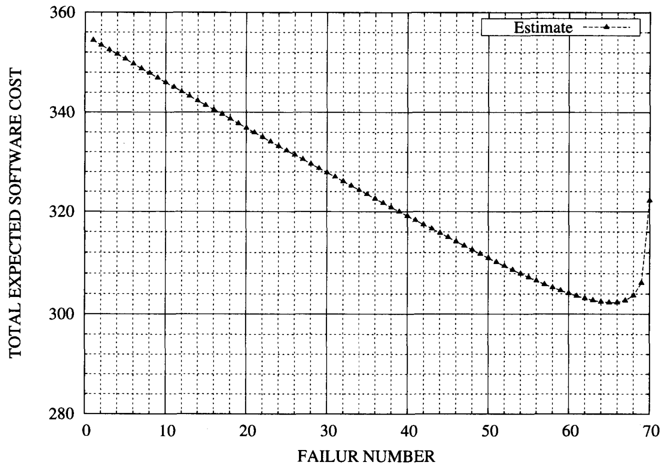
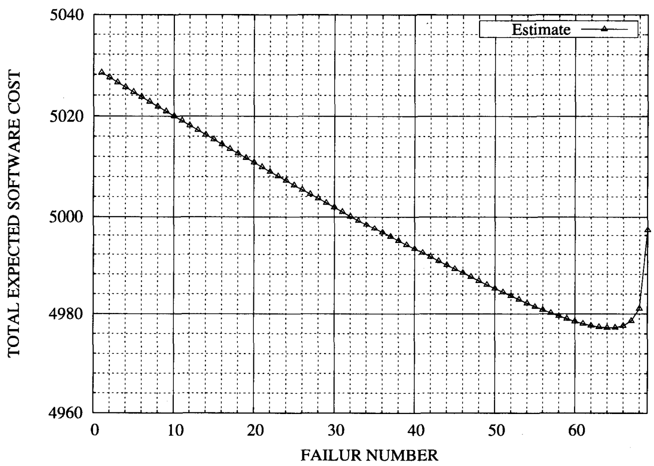


図 2： 総期待ソフトウェアコスト.



のように定義できる．ここで $\epsilon = (1-\gamma)$ は信頼係数を表す．また $\chi^2_\beta(\sigma)$ は上側確率 $100\sigma\%$，自由度 β における χ 二乗値を表す．式 (17) より，推定された総期待ソフトウェアコストに対する信頼限界の上限値および下限値を以下のように表すことができる．

$$C^U(l) \;=\; c_1 \sum_{k=1}^{l} \frac{2k}{\chi^2_{2k}\left(1-\dfrac{\epsilon}{2}\right)} \mathrm{E}[X_k] + c_2 l + c_3 \left(\sum_{i=1}^{2} N_i - l\right), \tag{18}$$

$$C^L(l) \;=\; c_1 \sum_{k=1}^{l} \frac{2k}{\chi^2_{2k}\left(\dfrac{\epsilon}{2}\right)} \mathrm{E}[X_k] + c_2 l + c_3 \left(\sum_{i=1}^{2} N_i - l\right). \tag{19}$$

推定された総期待ソフトウェアコストに対する信頼限界の上限値および下限値を図 4 に示す．図 4 より，推定された総期待ソフトウェアコストの上限値が最小となる発見フォールト数 l^* は 64 である．そのときの最適リリース時刻 $\sum_{k=1}^{l^*} \mathrm{E}[X_k]$ は 11.88 日であり，総期待ソフトウェアコスト $C^U(l^*)$ は 310.66 となる．一方で，推定された総期待ソフトウェアコストの下限値が最小となる発見フォールト数 l^* は 66 である．そのときの最適リリース時刻 $\sum_{k=1}^{l^*} \mathrm{E}[X_k]$ は 14.29 日であり，総期待ソフトウェアコスト $C^U(l^*)$ は 299.06 となる．したがって，本数値例

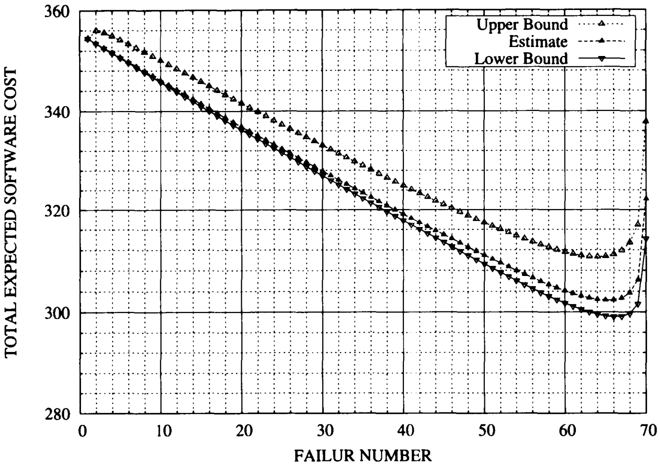


図 4： 総期待ソフトウェアコストの信頼限界.

においては組込みシステムを 11.83 日から 14.293 日の間にリリースすることにより，移植作業工程の総期待ソフトウェアコストを，信頼限界の範囲内で最小化することが可能となる．また，**4.2** と同様に，組込み OSS の残存フォールト数を考慮した場合における総期待ソフトウェアコストの信頼限界を推定することも可能となる.

6 おわりに

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 を採用した組込みシステム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 を採用した組込みシステム開発においては，移植作業の成否が組込み製品を出荷できるどうかに直接的に関係することから，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に比例して MTBF が増加する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，組込みシステムを構成する組込み OSS とデバイスドライバのような複数のコンポーネントを同時に考慮したハザードレートモデルを提案するとともに，実際の移植作業工程を想定した数値例を示した.

また，OSS を利用した組込みシステムの移植作業期間における最適リリース問題として，移植作業工程と運用段階におけるコストから総期待ソフトウェアコストの定式化を行い，総期待ソフトウェアコストを最小化するような最適リリース時刻を決定する問題について議論した．特に，組込み OSS を使用しない場合における総期待ソフトウェアコストを定式化し，OSS 導入時のコスト削減量を定量化した．また，総期待ソフトウェアコストに対する信頼限界の上限・下限値の数値例を示した．本手法により，組込み OSS の導入に踏み切るか否かの判断材料の一つとして利用できるものと考える.

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, Apache Software Foundation, http://httpd.apache.org/

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

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

[4] Eric Andersen, BusyBox, http://www.busybox.net/

[5] 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.

[6] A. MacCormack, J. Rusnak, and C.Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," Informs J. Management Science, vol. 52, no. 7, pp.1015–1030, 2006.

[7] Y. Zhoum, J. Davis, "Open source software reliability model: an empirical approach," Proc. the Workshop on Open Source Software Engineering (WOSSE), vol. 30, no. 4, 2005, pp. 67–72.

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

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

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

[11] Y. Tamura and S. Yamada, "A method of user-oriented reliability assessment for open source software and its applications," Proc. of the 2006 IEEE Intern. Conf. on Systems, Man, and Cybernetics, Taipei, Taiwan, October 8-11, 2006, pp. 2185–2190.

[12] 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.

[13] S.Yamada and S.Osaki, "Cost-reliability optimal software release policies for software systems," IEEE Trans. Reliability, vol. R-34, no. 5, pp. 422–424, 1985.

[14] S.Yamada and S.Osaki, "Optimal software release policies with simultaneous cost and reliability requirements," European J. Operational Research, vol. 31, no. 1, pp. 46–51, 1987.