Vol.2010-SE-169 No.4
2010/7/22

組込みOSSの移植工程に対する最適リリース問題とコスト削減に関する有効性評価

田村慶信[†1]　山田茂[†2]

近年，組込み機器に対してもAndroidやBusyBoxに代表される組込みOSS（open source software, 以下OSSと略す）が積極的に採用されつつある．特に，OSSを利用した組込みシステム開発では，自社で開発されたハードウェア上でOSSが動作するよう修正・再構築する作業である移植工程での品質評価が問題となっている．現在では，こうした移植可能性の問題から，OSSの導入へ踏み切れない企業が多く存在している．

本研究では，オープンソースプロジェクトの下で開発されているOSSを利用した組込みシステムの移植工程における信頼性を評価するために，組込みOSSを構成する主要コンポーネントと，それに影響を及ぼすコンポーネントとの特徴を同時に考慮したハザードレートモデルに基づく信頼性評価法を提案し，移植工程における最適リリース時刻を推定する．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を利用することによるコスト削減量について考察する．さらに，実際のOSSのソフトウェア故障発生時間間隔データに対する数値例を示すことにより，組込みOSSの移植性評価法について考察する．

Effectiveness Assessment of Cost Reduction and Optimal Release Problem for the Porting Phase of Embedded OSS

Yoshinobu Tamura[†1] and Shigeru Yamada[†2]

An embedded OSS (Open Source Software) known as one of OSS has since been gaining a lot of attention in embedded system area, i.e., Android, BusyBox, etc. However, the poor handling of quality and customer support prohibit the progress of embedded OSS. Also, it is difficult for developer to assess reliability and portability of the porting-phase in case of installing embedded OSS on single-board computer. We focus on the problems in such as the software quality, which prohibit the progress of embedded OSS.

In this paper, we propose a method of software reliability assessment based on a flexible hazard rate model considering several components such as device drivers. Also, we find the optimal release time of porting-phase by minimizing the total expected software cost. Moreover, we discuss the effectiveness assessment of cost reduction in case of using the OSS. Also, we analyze actual software failure-occurrence time-interval data to show numerical examples of software reliability assessment and optimal release time for the embedded OSS.

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，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，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，急激に普及が広まっている[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)]．特に，OSSはサーバ用途を主として，多くの分野において使用されているが，最近のOSSの傾向として，組込み機器に対してもAndroid[4)]やBusyBox[5)]に代表される組込みOSSが積極的に採用されつつある．

一方，OSSの利用に関しては，未だに多くの不安が残されている．まず第1に，システム導入後のサポートや品質上の問題といった利用者側の一般的な不安である．第2に，OSSは本当にビジネスになるのか，オープンソースのソフトウェアを事業化することによって自社製のソフトウェア商品までが市場を失うことにならないか，といった開発者側の不安である[3)]．特に，サポートや品質上の問題については，OSSの普及を妨げる大きな要因として考えられており，組込みOSSに関しては不安材料の大きな要因の1つとして考えられる．

また，OSSに対する現在の研究動向としては，設計工程や開発手法，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が[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

†1 山口大学大学院理工学研究科
Graduate School of Science and Engineering, Yamaguchi University
†2 鳥取大学大学院工学研究科
Graduate School of Engineering, Tottori University

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[12)–16)]．

本研究では，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（ポーティング）工程に対する信頼性および移植性評価法を提案する．特に，意思決定手法の 1 つである AHP 手法を用いて各コンポーネントに対する重要度を推定する．また，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．本研究では，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．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し，移植工程における最適リリース時刻を推定するために，総期待ソフトウェアコストを定式化し，その最適化問題を考える．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，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も定式化し，OSS を利用することによるコスト削減量について考察する．また，以上のことを，実際のバグトラッキングシステム上に登録されたソフトウェア故障発生時間間隔データに基づく信頼性評価例と共に示す．

2. 組込み OSS の移植工程に対する一般化ハザードレートモデル

本研究では，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_0 で A1 を，確率 p_i 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，

$$z_k(x) = p_0 \cdot z_k^0(x) + \sum_{i=1}^{m} p_i \cdot z_k^i(x) \tag{1}$$

$$(k = 1,\ 2,\ \cdots;\ p_0 > 0,\ p_i > 0,\ p_0 + \sum_{i=1}^{m} p_i = 1),$$

$$z_k^0(x) = D(1 - \alpha \cdot e^{-\alpha k})^{k-1} \tag{2}$$

$$(k = 1,\ 2,\ \cdots;\ -1 < \alpha < 1,\ D > 0),$$

$$z_k^i(x) = \phi_i\{N_i - (k-1)\} \tag{3}$$

$$(i = 1,\ 2,\ \cdots,\ m;\ k = 1,\ 2,\ \cdots,\ N_i;\ N_i > 0,\ \phi_i > 0),$$

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．

$z_k^0(x)$: A1 に対する X_k のハザードレート，
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 X_k のハザードレート，
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 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反映することができる．

Vol.2010-SE-169 No.4
2010/7/22

本モデルに含まれる式 (2) は，既存の Moranda モデル[15] 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[14] 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[12]．

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，

$$f_k(x) = \left\{ p_0 D(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\phi_i(N_i-k+1) \right\} \cdot \exp\left[-\left\{ p_0 D(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\phi_i(N_i-k+1) \right\} \cdot x \right], \tag{8}$$

となる．また，ソフトウェア信頼度は，

$$R_k(x) = \exp\left[-\left\{ p_0 D(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\phi_i(N_i-k+1) \right\} \cdot x \right], \tag{9}$$

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，

$$\mathrm{E}[X_k] = \frac{1}{p_0 D(1-\alpha\cdot e^{-\alpha k})^{k-1} + \sum_{i=1}^{m} p_i\phi_i(N_i-k+1)}, \tag{10}$$

により与えられる．

4. AHP に基づく各コンポーネントに対する重み係数の推定

1970 年代に開発された AHP（Analytic Hierarchy Process）は，主観的判断による意思決定支援に有効な方法として，欧米を中心に経営問題，エネルギー問題，政策決定，都市計画学など様々な分野で広く活用されている[17),18]．

ソフトウェアの信頼性評価手法の開発において，各コンポーネントでのデバッグの状況やその良し悪しが，システム全体の信頼性に与える影響を考慮しようとする場合，プログラムパス，コンポーネントの規模，フォールト報告者のスキルなどの，様々に絡み合った要因を捉える必要があると考えられる．しかしながらこれは困難であることが予想される．したがって本研究では，こうした複雑な状況下でシステム全体の信頼性に対する各コンポーネントの影響度合いを推定するために，一般には主観的判断の合理的合成方法として知られている AHP を利用し，システム全体の信頼性に対する各コンポーネントの重要度を表す重み係数の推定を行う．特に，適用される評価基準としては，各コンポーネントに対して発見されたフォールトの重要度，コンポーネントの規模，フォールト報告者のスキルといった要因が考えられる．各コンポーネントにおける AHP の評価基準に対するウェイトを，それぞれ $w_i(i=1,2,\cdots,m)$ とすれば，一対比較行列は，

$$A = \begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_m} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_m}{w_1} & \frac{w_m}{w_2} & \cdots & \frac{w_m}{w_m} \end{bmatrix}, \tag{11}$$

となる．この一対比較行列から各評価基準に対するウエイトを次式の幾何平均により求めることができる．

$$\alpha_i = \sqrt[m]{\prod_{i=1}^{m} w_i}. \tag{12}$$

以上のことから，各ソフトウェアコンポーネントに対するウエイトは，

$$p_i = \frac{\alpha_i}{\sum_{i=1}^{m} \alpha_i}, \tag{13}$$

により与えられる．

5. 数 値 例

5.1 組込み OSS

本研究では，携帯電話用 OS として開発・公開されている Android[4] 上で BusyBox[5] 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．

本研究では，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．

5.2 評 価 結 果

まず，**4** の AHP に基づく各コンポーネントに対する重みパラメータ p_0, p_1, および p_2 の推定結果を表 1 に示す．特に，評価基準としては，各コンポーネントに対するフォールトの重要度を取り上げた．表 1 から，Android コンポーネントに対する重要度が最も大きいことが分かる．一方，buildroot コンポーネントに対する重要度は最小であることが確認できる．

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，非線形最小二乗法の 1 つである Levenberg-Marquardt 法により，モデルに含まれる未知パラメータ D, α, ϕ_1, ϕ_2, N_1, および N_2 を推定した．まず，推定された MTBF を図 1 に示す．図 1 から，ソフトウェア故障が発生していく，すなわち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 2 に示す．図 2 から，0.25 日後のソフトウェア信頼度は約 0.1 であることが分かる．

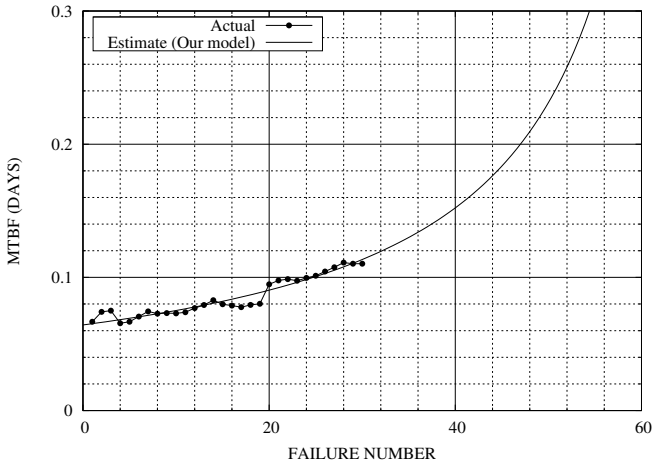

図 1 推定された MTBF.

6. モデルパラメータに対する感度分析

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 3 に示す．図 3 から，α の値が大きくなるにつれ，ソフトウェア信頼度が加速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，平均ソフトウェア故障発生時間間隔が小さくなり，すなわち信頼度成長が退化する様子が確認できる．このように，信頼度成長が退化する場合においては，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ことを意味する．

同様に，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

表 1 AHP に基づく各コンポーネント重要度の推定結果

	SLOC	Development History	Compatibility	Weight
Android	0.7854	0.7643	0.7854	0.5842
BusyBox	0.1488	0.1626	0.1488	0.2808
buildroot	0.0658	0.0730	0.0658	0.1350
Weighted Value	0.5803	0.2047	0.1582	0.0568

Component	SLOC	Development History	Compatibility	Weight	Total Weight Parameter $\widehat{p_i}$
Android	0.4558	0.1564	0.1242	0.0332	0.7696
BusyBox	0.0864	0.0333	0.0235	0.0160	0.1591
buildroot	0.0382	0.0150	0.0104	0.0077	0.0712

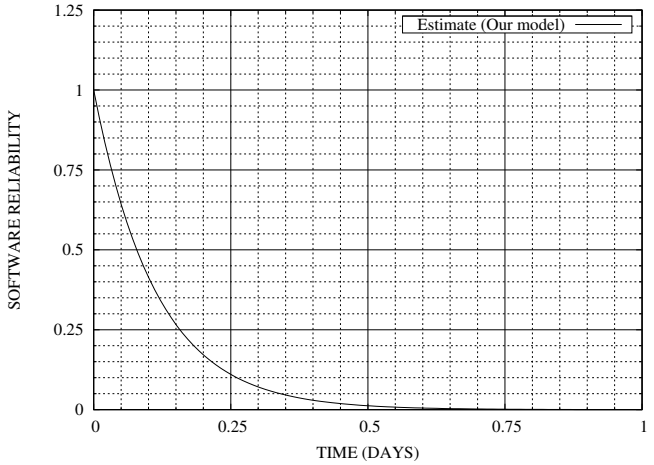

図 2 推定されたソフトウェア信頼度.

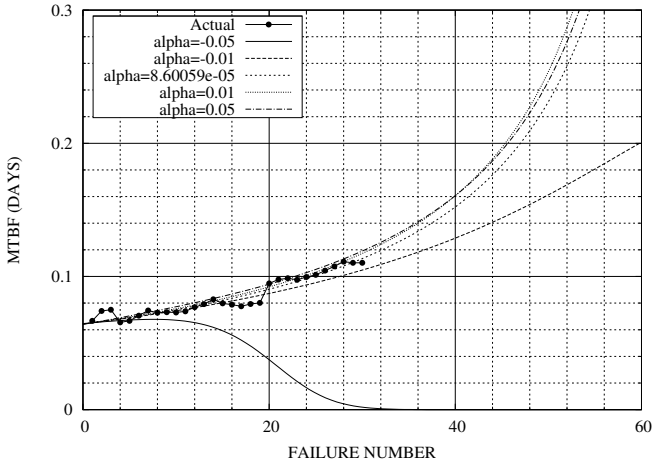

図 3 推定された MTBF のパラメータ α に対する感度分析結果.

ソフトウェア信頼度を図 4 に示す．図 4 から，パラメータ α が負の値をとる場合には，急速にソフトウェア信頼度が低下する様子が確認できる．

7. 最適リリース時刻の推定

7.1 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[19),20)] 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 $(c_3 > c_2)$.

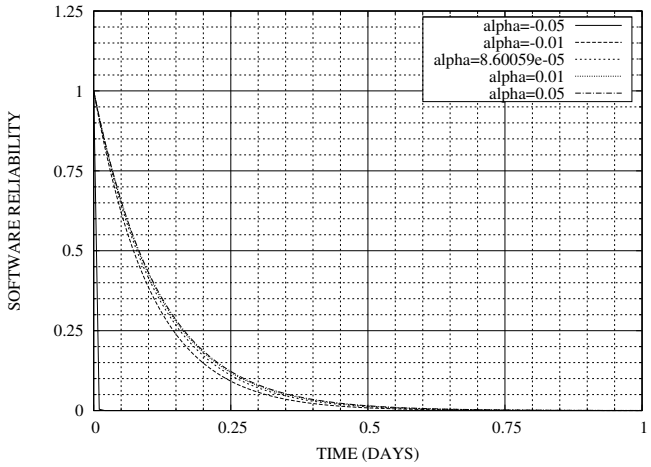

図 4 推定されたソフトウェア信頼度のパラメータ α に対する感度分析結果.

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．

$$C_1(l) = c_1 \sum_{k=1}^{l} \mathrm{E}[X_k] + c_2 l. \tag{14}$$

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．但し，$N > l$ と仮定する．

$$C_2(l) = c_3 \left(\sum_{i=1}^{m} N_i - l \right). \tag{15}$$

したがって，総期待ソフトウェアコストは，式 (14) および式 (15) より，

$$C_3(l) = C_1(l) + C_2(l), \tag{16}$$

のように表すことができる．この式 (16) を最小にする $l = l^*$ から，最適リリース時刻の期待値である $\sum_{k=1}^{l^*} \mathrm{E}[X_k]$ を求めることができる．

7.2 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．

$$C_4(l) = c_3 \left(\frac{p_0 \sum_{i=1}^{m} N_i}{\sum_{i=1}^{m} p_i} + \sum_{i=1}^{m} N_i - l \right). \tag{17}$$

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，

$$C_5(l) = C_1(l) + C_4(l), \tag{18}$$

のように表すことができることから，式 (16) および式 (20) より，ソフトウェアコストの削減量は，

$$C_5(l) - C_3(l) = \frac{c_3 \cdot p_0 \sum_{i=1}^{m} N_i}{\sum_{i=1}^{m} p_i}, \tag{19}$$

となる．

7.3 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 5 に示す．図 5 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 $\sum_{k=1}^{65} \mathrm{E}[X_k] = 12.928$ 日後となり，そのときの総期待ソフトウェアコストは 307.81 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 6 に示す．ここで，N_1 および N_2 のパラメータ推定結果

$$\widehat{N_1} = 99.971,\ \widehat{N_2} = 20.007,$$

と表 1 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 401 個となる．したがって，図 6 から，最適リリース時刻には変化はないが，そのときの総期待ソフトウェアコストは 1510.6 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 1202.8 であり，組込み OSS を利用することにより約 4 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．

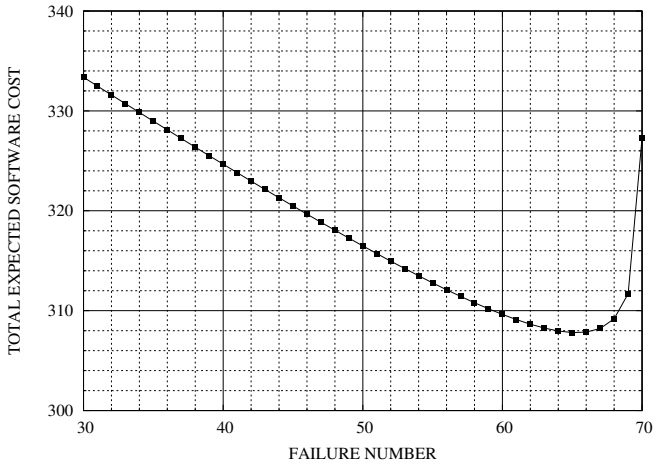

図 5 推定された総期待ソフトウェアコスト.

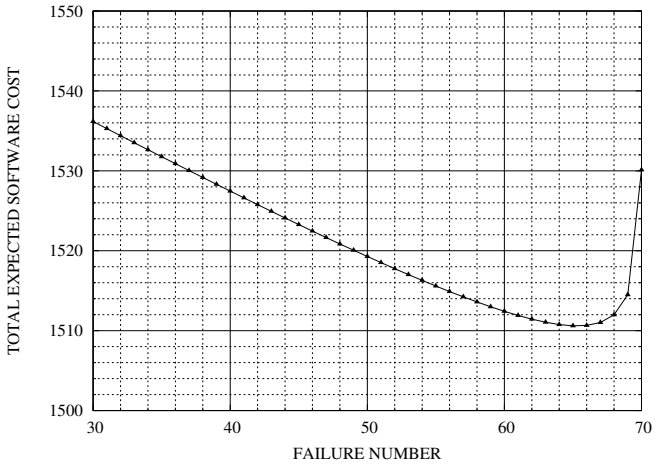

図 6 組込み OSS の残存フォールト数を考慮した推定された総期待ソフトウェアコスト.

8. むすび

本研究では，OSS を利用した組込みシステム開発の移植作業工程に対するコンポーネント重要度を考慮した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．本研究では，組込み OSS に対するハザードレートモデルを構築するとともに，組込みシステムを構成する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した．特に，各コンポーネントに対する重要度を表す重みパラメータに対して AHP 手法を適用した．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．

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，移植作業工程および運用段階における保守コストを考慮して，総ソフトウェアコストが最小となるように最適リリース時刻を求めるために，総期待ソフトウェアコストを定式化し，これ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 を利用することによるコスト削減量を定量化した．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 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．本研究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．本研究では，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．本研究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

用できるであろう．

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) Open Handset Alliance, Android, http://www.android.com/
5) Erik Andersen, BUSYBOX, http://www.busybox.net/
6) MacCormack, A., Rusnak, J. and Baldwin, C.Y.: 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) Zhoum, Y. and Davis, J.: 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) Li, P., Shaw, M., Herbsleb, J., Ray, B. and Santhanam, P.: Empirical evaluation of defect projection models for widely-deployed production software systems, *Proc. the 12th International Symposium on the Foundations of Software Engineering (FSE-12)*, 2004, pp. 263–272.
9) Norris, J.: Mission-critical development with open source software, *IEEE Software Magazine*, Vol. 21, No. 1, 2004, pp. 42–49.
10) Tamura, Y. and Yamada, S.: Software reliability assessment and optimal version-upgrade problem for open source software, *Proc. the 2007 IEEE International Conference on Systems, Man, and Cybernetics*, Montreal, Canada, October 7–10, 2007, pp. 1333–1338.
11) Tamura, Y. and Yamada, S.: A method of user-oriented reliability assessment for open source software and its applications, *Proc. the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan, Oct. 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) Schick, G.J. and Wolverton, R.W.: An Analysis of Competing Software Reliability Models, *IEEE Trans. Software Engineering*, Vol. SE–4, No. 2, pp. 104–120 (1978).
14) Jelinski, Z. and Moranda, P.B.: Software Reliability Research, in *Statistical Computer Performance Evaluation*, Freiberger, W.(ed.), pp. 465–484, Academic Press, New York (1972).
15) Moranda, P.B.: Event–altered Rate Models for General Reliability Analysis, *IEEE Trans. Reliability*, Vol. R–28, No. 5, pp. 376–381 (1979).
16) Xie, M.: On a Generalization of the J-M Model, *Proc. Reliability '89*, 1989, pp. 5 Ba/3/1–5 Ba/3/7.
17) 加藤 豊，小沢 正典：OR の基礎－ AHP から最適化まで－，実教出版株式会社，東京（1998）.
18) 木下 栄蔵：入門 AHP，日科技連出版社，東京（2000）.
19) Yamada, S. and Osaki, S.: Cost-reliability optimal software release policies for software systems, *IEEE Trans. Reliability*, Vol. R-34, No. 5, pp. 422–424 (1985).
20) Yamada, S. and Osaki, S., Optimal software release policies with simultaneous cost and reliability requirements, *European J. Operational Research*, Vol. 31, No. 1, pp. 46–51 (1987).