B-007

組込みオープンソースソフトウェアの移植工程に対する信頼性評価ツールの開発と性能評価

Performance Evaluation of Reliability/Portability Assessment Tool for the Porting-Phase of Embedded Open Source Software

中道 徹* 田村 慶信† 山田 茂‡
Toru Nakamichi Yoshinobu Tamura Shigeru Yamada

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間での共同開発，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．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．近年，組込み向け OSS は， Android [4] 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 つ目はサポートへの不安である．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，移植工程におけるフォールト発生事象に基づいた信頼性評価法を提案する．これにより，組込み向け OSS の移植作業に対して，信頼性評価に関する何らかの評価基準を与えることができるものと考える．また，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・移植性評価結果を一覧表示する機能を追加し，その実行結果を示す．

2 ソフトウェア信頼度成長モデル

2.1 OSS に対する信頼性評価

OSS の開発環境とソフトウェア信頼性モデリングの観点から考えると，ソフトウェア内に潜在するフォールト数が有限であると仮定されたモデルよりも，むしろ無限回のソフトウェア故障が発生すると仮定されたモデルを適用する方が現実的である．すなわち，OSS の開発では常にフォールト修正やバージョンアップが繰り返されており，使用頻度や人気の高い OSS になるほどフォールト報告が頻繁に行われている．この運用形態は OSS の開発プロジェクトが無意味なものとみなされ解散するまで続けられる．したがって，1 つの企業組織内において，ある特定の使用目的に限定されたソフトウェアの開発を対象としている従来のソフトウェア信頼度成長モデル（software reliability growth model，以下 SRGM と略す）では，OSS の信頼度成長現象を十分に包括できないものと考える [5–8]．

これまでにも，数多くの SRGM が提案されている [9–11]．特に，その多くはソフトウェア内に潜在するフォールト数が有限であると仮定されたものである．潜在するフォールト数が無限大であると仮定された SRGM もいくつか提案されているが，信頼度退化を表現するために，それらの SRGM から導出される信頼性評価尺度から有益な情報を得ることは難しい．したがって，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を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]．

2.2 ハザードレートモデル

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がハザードレー

*山口大学大学院理工学研究科電子情報システム工学専攻
†山口大学大学院理工学研究科
‡鳥取大学大学院工学研究科

トモデルである．[11] このモデルでは，任意のテスト時刻におけるハザードレートは，その時点におけるソフトウェア内の残存フォールト数に比例すると仮定するものが多い．これらは，評価対象のソフトウェア故障発生時間間隔が，指数分布あるいはワイブル分布に従うものとする 2 種類のハザードレートに大別できる．本論文では，取り扱いも容易で単純な構造をもつ指数分布型ハザードレートモデルについて議論する．

3 組込み OSS の移植工程に対する信頼性評価法

3.1 モデルの記述

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)-式 (3) により表すことができるものと仮定する．

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

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

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

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．

3.2 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，

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

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，$\Pr\{A\}$ 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

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

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

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

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，

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

により与えられる．

4 Laplace Trend Test

一般的に，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発生す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，信頼度成長過程を表す評価尺度として Laplace Trend Test を用いる．Laplace Trend Test の検定統計量 $u(i)$ は

$$u(i) \equiv \frac{\frac{1}{i-1}\sum_{n=1}^{i-1}\sum_{j=1}^{n}\theta_j - \frac{\sum_{j=1}^{i}\theta_j}{2}}{\sum_{j=1}^{i}\theta_j\sqrt{\frac{1}{12(i-1)}}}, \tag{11}$$

により求めることができる．ここで，i は観測された j 番目のフォールト数を，θ_j は観測された j 番目のフォールト発見時間間隔を表す [17,18]．

5 モデルの適合性評価

本論文では，実測データへのモデルの適合性評価のために予測相対誤差を用いる．予測相対誤差 (predicted relative error) は，任意の時点 t_k において推定したときの，フォールト報告終了時点 t_K までに発見されるデータの予測値と実測値の相対誤差である．任意の時点 t_k における予測相対誤差を $PRE_k[t_k]$ とすると，

$$PRE_k[t_k] = \frac{\hat{y}(t_k;t_K) - y_K}{y_K}, \quad (12)$$

により計算される．$\hat{y}(t_k;t_K)$ は，任意の時点 t_k までの実測データを用いたフォールト報告終了時点 t_K でのデータの推定値，y_K は，フォールト報告終了時点 t_K までのデータの実測値である．

6 ツールの開発

6.1 要求仕様定義

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，最小二乗法を適用する．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 を用いる．
5. 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．
6. 全てのグラフは同時に表示する．
7. 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．
8. ツールの操作には GUI を使用し，マウスを用いて行う．
9. ツールの開発言語に Java を使用する．
10. グラフの描画には JFreeChart を使用する．
11. 数値計算の簡略化のため，以下のように定義する．

$$w_1 = pD, \quad (13)$$
$$w_2 = (1-p)\phi. \quad (14)$$

6.2 実行手順

本ツールを用いた OSS に対する信頼性・移植性評価ツールの実行手順を以下に示す．

1. 移植工程からフォールト発見時間間隔データを採取する．
2. フォールトデータと未知パラメータの初期値を CSV フォーマットで入力する．
3. 最小二乗法により未知パラメータを推定する．
4. 種々の信頼性・移植性評価尺度をグラフ表示する．
5. 全ての信頼性・移植性評価結果を一覧表示する．

7 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 [4] 上で BusyBox [19] 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 (3.1 参照)．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．

7.1 パラメータ推定

本ツールの実行結果として，まず，未知パラメータの推定結果を図 2 に示す．図 2 から，$\widehat{w_1} = 0.96910$，$\widehat{w_2} = 0.02519$，$\widehat{\alpha} = 0.04168$，$\widehat{N} = 99.9653$ と推定された．特に，$\widehat{\alpha}$ が正の値であることから，OSS の活動状態が安定していることが確認できる．

7.2 ソフトウェア信頼度

次に，式 (9) のソフトウェア信頼度の推定結果を図 3 に示す．ここで，横軸は時刻を表し，単位は日である．また，縦軸はソフトウェア信頼度を表す．図 3 から，フォールト報告終了以降において同様の移植環境で動作させた場合のソフトウェア信頼度は，2 日後には約 0.1 まで低下していることが分かる．したがって，今後も移植作業を継続して実施する必要があることが確認できる．

7.3 MTBF および予測相対誤差

式 (10) の MTBF の推定結果を図 4 に示す．ここで，横軸はソフトウェア故障数を表し，縦軸は平均ソフトウェア故障発生時間間隔を表す．また，赤色の折れ線は実測値を，青色の折れ線は推定値を表す．さらに MTBF に対する予測値と実測値の予測相対誤差を図 5 に示す．ここで，横軸はプロジェクト全体の進捗度，縦軸は予測相対誤差を表す．図 4 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．図 5 から，予測相対誤差の推定値は，移植工程が進むにつれ小さくなり，推定結果が安定している様子が分かる．

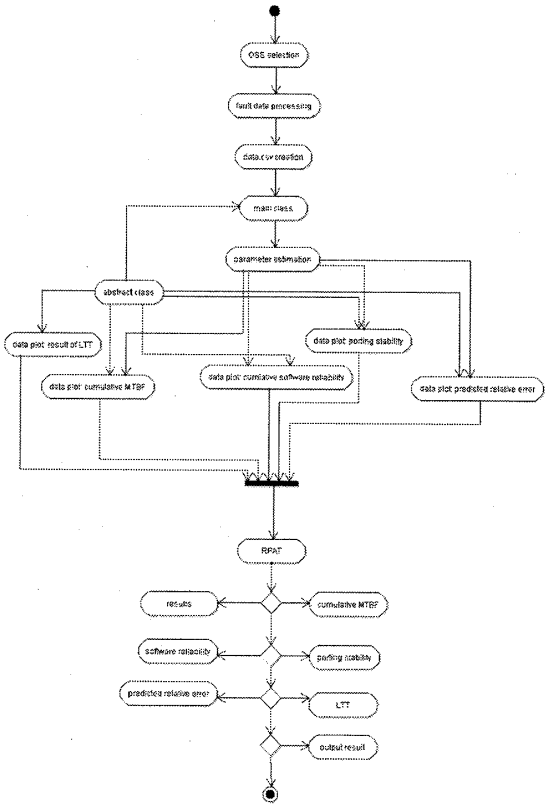

図 1： 本ツールのアクティビティ図.

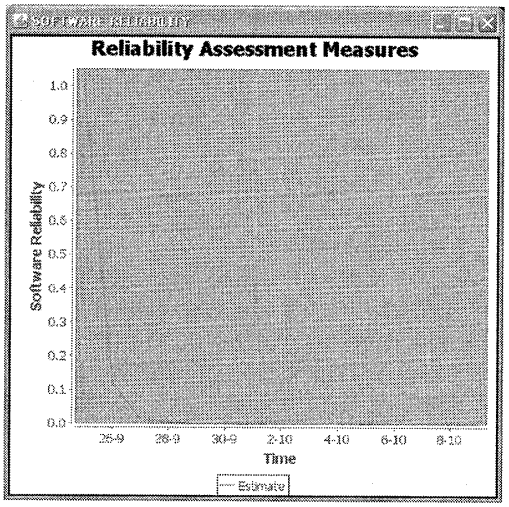

図 3： ソフトウェア信頼度の推定結果.

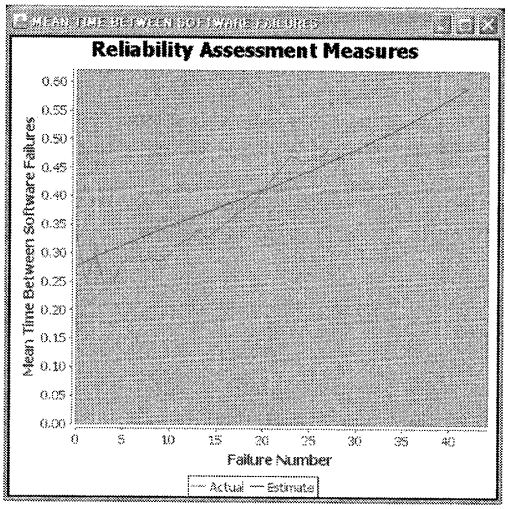

図 4： MTBF の推定結果.

7.4 移植性評価

ツール上で組込み OSS とコンポーネントの重要度を推定した結果を図 6 に示す．ここで，縦軸はコンポーネントの重要度を表す．棒グラフの左は組込み OSS の重要度を，棒グラフの右は独自に実装したソフトウェアコンポーネントの重要度を表す．図 6 から，移植対象となる組込み OSS の重要度が 97 %を占めていることが分かる．これは，組込みシステム開発における組込み OSS の重要度が大きいことを示す．このことから，本実行例においては，システム全体に対して，新たに移植するコンポーネントの重要度は低く，移植安定性が高いことが確認できる．

7.5 Laplace Trend Test

式 (11) に基づく Laplace Trend Test による推定結果を図 7 に示す．ここで，横軸は観測されたフォールトの番号

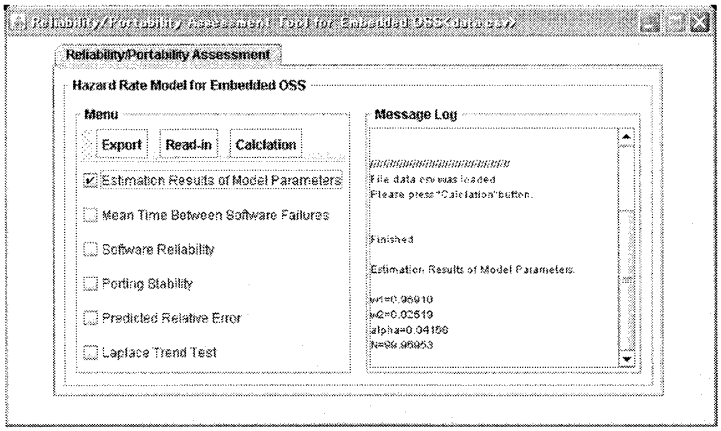

図 2： パラメータの推定結果.

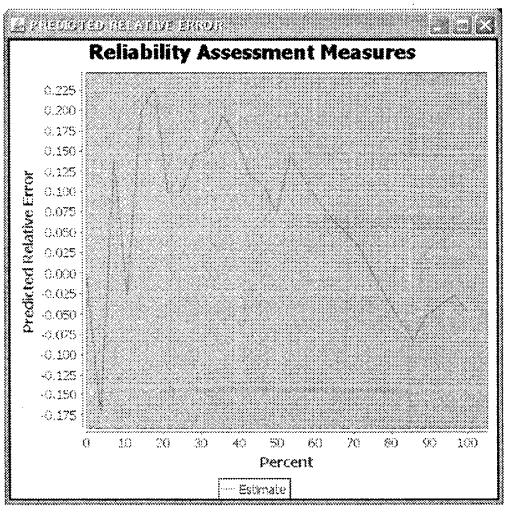

図 5： 予測相対誤差の推定結果.

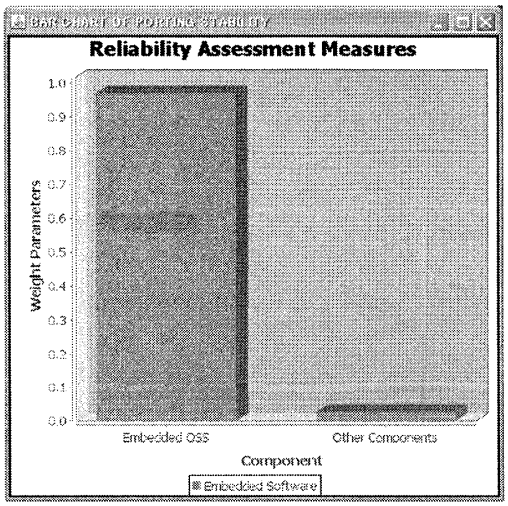

図 6： 移植性評価結果.

を，縦軸は Laplace Trend Test の検定統計量を示す．図 7 から，10 個目のフォールト発見数までの推定結果は正の値を示し，それ以降においては負の値を示している様子が確認できる．このことから，移植作業工程初期においては信頼度退化を示しているが，移植作業が進むにつれて信頼度成長に転じている様子が分かる．

7.6 推定結果の出力

信頼性・移植性評価の推定結果をテキストファイルおよび JPEG ファイルに出力し，HTML フォーマットで出力した結果を図 8 に示す．これにより，移植工程の開発管理者およびユーザにとって，現在の進捗状況について一覧表示したものを容易に配布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．

8 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．

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言える．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

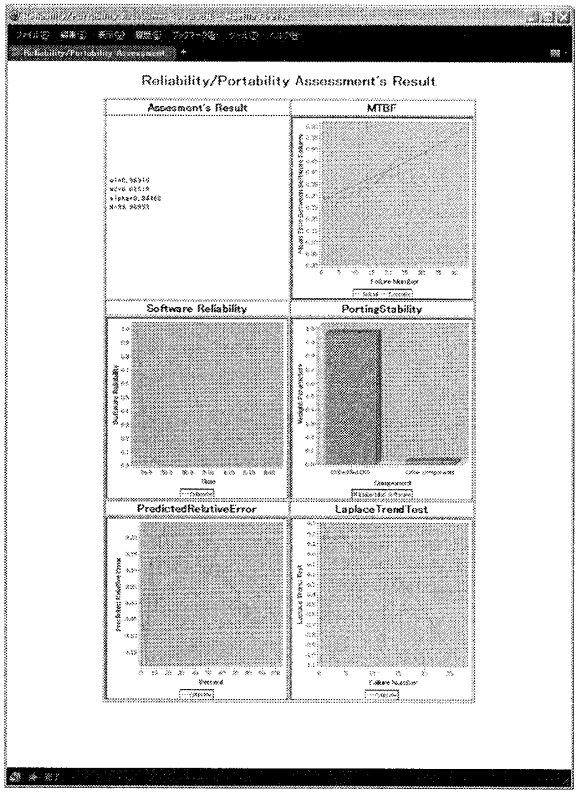

図 8： 信頼性・移植性評価の HTML による出力結果.

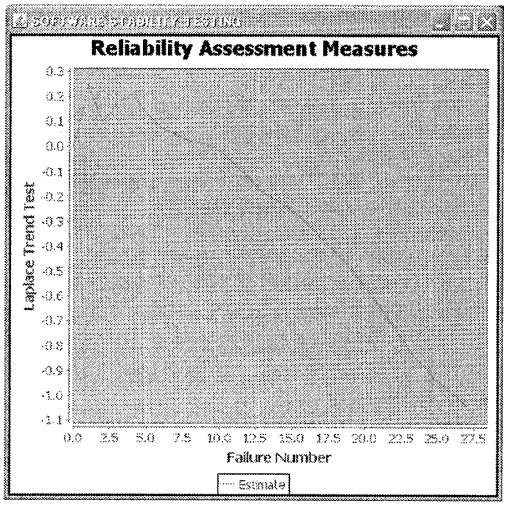

図 7： Laplace Trend Test の推定結果.

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であろう.

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．将来的には，上述したような最適リリース問題を定義し，最適リリース時刻推定機能をツールに組み込んで行く予定である.

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

[5] 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.

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

[7] 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.

[8] 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.

[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] 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.

[18] 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.

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