オープンソースソフトウェアに対するユーザ指向の信頼性評価ツールの開発

田村 慶信	肌附 康司	山田 茂	木村 光宏
広島工業大学情報学部	鳥取大学大学院工学研究科	鳥取大学工学部	法政大学工学部
情報工学科	社会開発システム工学専攻	社会開発システム工学科	経営工学科
tam@cc.it-hiroshima.ac.jp	b02t7041z@edu.tottori-u.ac.jp	yamada@sse.tottori-u.ac.jp	kim@k.hosei.ac.jp

概　要

現在のソフトウェア開発は，ネットワーク環境における分散開発が主流となっている．中でも，世界中の誰もが開発に参加できるという特徴をもつオープンソースプロジェクトの下で開発されたオープンソースソフトウェアは，分散型ソフトウェア開発形態の成功例として近年特に注目されている．特に，オープンソースソフトウェアの利用に関しては未だに多くの不安が残されており，サポートや品質上の問題は普及を妨げる大きな要因として考えられている．

本論文では，オープンソースソフトウェアのシステム全体を構成する複数のコンポーネントの信頼性に関する重要度を推定するために，ニューラルネットワークを適用した信頼性評価法を提案し，これを Java 言語によりツール化する．さらに，本ツールの適用可能性を評価するために，実際のオープンソースソフトウェアに対する信頼性評価を行う．本ツールがオープンソースソフトウェアの信頼性を定量的に評価するための指標を提示する手段として利用できるものと考える．

1　はじめに

ネットワーク環境を利用して開発された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，誰でも自由に改変することが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ている．また，OSS の代表格ともいうべき Linux[1]は市場でサーバ用途として有効であると認められ，この数年急激に普及してきている．

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および品質上の問題といった利用者側の不安である．第 2 に，OSS は本当にビジネスになるのか，オープンソースとして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のか，といった開発者側の不安である．特に，システム導入後のサポートおよび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いる [1]．OSS が急速に普及し始めている現在，OSS の信頼性に関するなんらかの評価指標を提示することが重要であると思われる．

本論文では，こうした OSS に対する信頼性評価法を提案するとともに，実際に公開されている OSS に対して本手法の適用可能性について考察する．特に，システム全体を構成する複数のコンポーネント，いわゆる各ソフトウェアコンポーネントの重要度を推定するためにニューラルネットワークを適用する．これにより，バグトラッキングシステム上から得られたデータに基づき，各コンポーネント間の相互作用を包括した信頼性評価法として利用できるものと考える．さらに，オブジェクト指向型言語である Java を用いて，本手法を OSS に対する信頼性評価ツールとして実装するとともに，本ツールの適用可能性を評価するために，実際の複数の OSS に対する信頼性評価の実行例を示す．本ツールにより，バグトラッキングシステム上から採取されたデータのみに基づいて，OSS の内部構造に熟知していないユーザの

[1]Linux は，Linus Torvalds の米国およびその他の国における登録商標あるいは商標である．

立場に立った OSS の信頼性に関するなんらかの指標を提示することが可能となるものと考える．

2　各コンポーネント間の相互作用

2.1　システム全体の信頼性に関する内部状態の把握

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．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，本論文ではニューラルネットワークを適用する．これにより，バグトラッキングシステム上から採取されたデータのみに基づいて機械的に各コンポーネントのシステム全体に与える重要度を推定することが可能となる．

2.2　ニューラルネットワークに基づく信頼性評価

OSS において，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を考えた場合，コンポーネントの規模，フォールト報告者のスキル，フォールト修正の状態，コンポーネントの開発時間，コンポーネント間のパスの数，コンポーネント間の入出力データ量といった様々な要因を考慮する必要がある．こうした複雑な状態を適当な仮定の下で物理的な意味からモデル化することは困難である．

これまでに，OSS を対象としたソフトウェア信頼性評価法として，AHP（Analytic Hierarchy Process）[2] とソフトウェア信頼度成長モデル（software reliability growth model，以下 SRGM と略す）を融合した手法が提案されている [3, 4]．しかしながら，これらは主に開発リーダの立場に立ったものであり，ユーザ側にとっては使用し難いという面があった．その主な理由は，オープンソースプロジェクトでは開発リーダの主観となる AHP の評価基準値の決定方法が難しいという問題である．ここでいう開発リーダとは，開発ボランティアの中でも最重要のメンバー，すなわち，中心的なソフトウェアを担当するコア開発者のことである．一方，周辺開発者やバグ報告者をユーザと定義する．OSS が急速に普及し始めている現在，こうしたユーザ側の立場に立った OSS の信頼性に関するなんらかの指標を提示することが重要であると思われる．

したがって，本論文では，各コンポーネント間の内部状態をブラックボックスとして捉えるためにニューラルネットワーク [5] を適用する．すなわち，入力と出力の関係から，その内部構造をニューラルネットワークにより学習させることによって，各コンポーネントがシステム全体の信頼性に与える影響度合いを推定する．これにより，バグトラッキングシステム上から採取されたデータのみに基づいた信頼性評価が可能となることから，実利用上においても容易に適用できるものと考える．本論文においては簡単のために 3 層ニューラルネットワークを適用する．

まず，$w_{ij}^1(i = 1, 2, \cdots, I; j = 1, 2, \cdots, J)$ を入力層と中間層の結合係数，また $w_{jk}^2(j = 1, 2, \cdots, J; k = 1, 2, \cdots, K)$ は中間層と出力層の結合係数とする．さらに，$x_i(i = 1, 2, \cdots, I)$ は正規化された入力データを表し，本論文では，フォールト報告者により致命的であると判断されたフォールト数，特定の OS において発見されたフォールト数，システムの内部構造に習熟した修正者のフォールト修正数，システムの内部構造に習熟した発見者のフォールト発見数とした．ここで，入力層，中間層，出力層におけるユニットの数を，各々I 個，J 個，および K 個とする．また，各々の層のユニットを示すインデックスを i, j, および k とする．ここで，各々の層のユニットの出力を h_j, y_k とすると，

$$h_j = f\left(\sum_{i=1}^{I} w_{ij}^1 x_i\right), \tag{1}$$

$$y_k = f\left(\sum_{j=1}^{J} w_{jk}^2 h_j\right), \tag{2}$$

となる．但し，$f(\cdot)$ はシグモイド型関数であり，

$$f(x) = \frac{1}{1 + e^{-\theta x}}, \tag{3}$$

として表される．ここで，θ はゲインと呼ばれる定数である．ネットワークの学習を行うために，誤差逆伝播法を用いる．ニューラルネットワークの出力層における値を $y_k(k = 1, 2, \cdots, K)$ とし，教師パターンを $d_k(k = 1, 2, \cdots, K)$ とすると，式 (2) の y_k の評価は次式で与えられる．

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^2. \tag{4}$$

ここで，教師パターン $d_k(k = 1, 2, \cdots, K)$ には，各コンポーネントにおける累積発見フォールト数データの正規化された値を採用する．すなわち，各コンポーネントにおける累積フォールト発見数データに基づいて，各コンポーネントの重み係数とそれに影響を及ぼす要因の結合状態の特徴をニューラルネットワークの結合係数に蓄積させ，ある時点における各コンポーネントの重要度の推定・予測が可能なモデルを考える．式 (4) の条件のもとに，各結合係数が最急降下法にて以下のように決定される．

$$\begin{aligned} w^2_{jk}(\sigma+1) &= w^2_{jk}(\sigma) \\ &+ \epsilon(y_k - d_k) \\ &\cdot f'\left(\sum_{j=1}^{J} w^2_{jk}(\sigma) h_j\right) h_j, \qquad (5) \\ w^1_{ij}(\sigma+1) &= w^1_{ij}(\sigma) \\ &+ \epsilon \sum_{k=1}^{K} (y_k - d_k) \\ &\cdot f'\left(\sum_{j=1}^{J} w^2_{jk}(\sigma) h_j\right) \\ &\cdot w^1_{ij}(\sigma) f'\left(\sum_{i=1}^{I} w^1_{ij}(\sigma) x_i\right) x_i. \qquad (6) \end{aligned}$$

ここで，σ は更新のサイクル，ϵ は学習の係数を表す．式 (5) および式 (6) の更新により求められた w^1_{ij} および w^2_{jk} から，式 (1) および式 (2) により，ニューラルネットワークの出力層における値 $y_k(k = 1, 2, \cdots, K)$ を算出することができる．これにより，各コンポーネントに対する重みパラメータ $p_i(i = 1, 2, \cdots, K)$ を次の式により導出できる．

$$p_i = \frac{y_i}{\sum_{i=1}^{K} y_i}. \qquad (7)$$

本論文では，ニューラルネットワークにおいて推定された各コンポーネントの重要度 p_i を，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と定義する．これは，あくまで信頼性に関する重要度であって，各コンポーネントの開発規模などを表すものではない．

3 システム全体に対する信頼性評価

従来から, ソフトウェアの信頼性を定量的に評価する手法として，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 1 つが，SRGM である [6]．中でも非同次ポアソン過程（nonhomogeneous Poisson process，以下 NHPP と略す）モデルは，実利用上極めて有効でありモデルの簡潔性が高いゆえにその適用性も高く，実際のソフトウェア信頼性評価に広く応用されている．この NHPP モデルは，所定の時間区間内に発見されるフォールト数や発生するソフトウェア故障数を観測して，これらの個数を数え上げる計数過程 $\{N(t), t \geq 0\}$ を導入し，以下の式で与えられる確率変数すなわちポアソン過程を仮定する SRGM である [6].

$$\begin{aligned} \Pr\{N(t) = n\} &= \frac{\{H(t)\}^n}{n!} \exp[-H(t)] \\ &\quad (n = 0, 1, 2, \cdots). \qquad (8) \end{aligned}$$

ここで，$\Pr\{\cdot\}$ は確率を表し，$H(t)$ は時間区間 $(0, t]$ において発見される総期待フォールト数，すなわち $N(t)$ の期待値を表し，NHPP の平均値関数と呼ばれる．

3.1 システム全体に対する信頼性評価

本論文では，検出可能フォールト数が無限であると仮定された非同次ポアソン過程に基づく対数型ポアソン実行時間モデルを適用する [6]．本モデルは，OSS に対する信頼性評価法に適用した結果から，短期的な予測精度に関して良好な結果を得ている [3]．したがって，本論文の数値例では，開発期間の短く歴史の浅い OSS を取り上げることから，システム全体に対する信頼性評価法について対数型ポアソン実行時間モデルを採用することとした．

このモデルでは，時間区間 $(0, t]$ で発見される総期待フォールト数を表す平均値関数 $\mu(t)$ は，

$$\begin{aligned} \mu(t) &= \frac{1}{\theta - P} \ln[\lambda_0(\theta - P)t + 1] \\ &\quad subject\ to\ \ (P - \theta) < \frac{1}{\lambda_0 t} \\ &\quad (0 < \theta,\ 0 < \lambda_0,\ 0 < P < 1), \qquad (9) \end{aligned}$$

により与えられる．ここで，パラメータ λ_0 は初期故障強度，パラメータ θ はソフトウェア故障 1 個当りの故障強度の減少率を表す．また，パラメータ P はシステム

全体に及ぼすコンポーネントの影響率を表す．これは，各コンポーネントに対してニューラルネットワークにより推定されたパラメータ y_i の平均値 $P = \sum_{i=1}^{n} y_i/n$ により表されるものとする．ここで，n はコンポーネント数を表す．さらに，モデルに含まれる未知パラメータの推定方法として最尤法を適用する．上記の NHPP モデルから，種々のソフトウェア信頼性評価のための定量的尺度を導出できる．

特に，OSS の開発環境について考えた場合，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 SRGM 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 [4]．本モデルにおけるパラメータ P により，OSS 開発の活動状態を定量的に表すことが可能となる．すなわち，OSS 開発において重要なのは，バグトラッキングシステムへのフォールト登録数やユーザ数の状態によって信頼度成長曲線が大きく変化することであり，こうした OSS 開発の活動状況を把握するための指標として有用であると考える．

3.2 モデルパラメータの推定

モデルに含まれる未知パラメータ λ_0 および θ の推定法として最尤法を適用する．このとき，一定の時刻 t_k までに発見された累積フォールト数 y_k に関する K 組のフォールト発見数データ $(t_k,\ y_k)(k = 1,\ 2,\ \cdots,\ K)$ が観測されているとすると，平均値関数 $\mu(t)$ をもつ NHPP モデルの対数尤度関数は，

$$\begin{aligned} \ln L &= \sum_{k=1}^{K} (y_k - y_{k-1}) \ln\left[\mu(t_K) - \mu(t_{k-1})\right] \\ &- \mu(t_K) - \sum_{k=1}^{K} \ln\left[(y_k - y_{k-1})!\right], \end{aligned} \tag{10}$$

となる．ここで，式 (9) の平均値関数 $\mu(t)$ をもつ対数型ポアソン実行時間モデルに含まれる未知パラメータ λ_0 および θ の最尤推定値を求めるために，λ_0 および θ について式 (10) を偏微分する．このとき，

$$\frac{\partial \ln L}{\partial \lambda_0} = \frac{\partial \ln L}{\partial \theta} = 0, \tag{11}$$

とおいて数値的に解くことにより，最尤推定値 $\hat{\lambda}$ および $\hat{\theta}$ を得る．具体的には，$\phi = \lambda_0(\theta - P)$ とおくと，式 (10) より，

$$\begin{aligned} \frac{\partial \ln L}{\partial \phi} &= \sum_{k=1}^{K} (y_k - y_{k-1}) \\ &\cdot \left\{ \frac{\left(\frac{t_k}{\phi t_k + 1}\right) - \left(\frac{t_{k-1}}{\phi t_{k-1} + 1}\right)}{\ln[\phi t_k + 1] - \ln[\phi t_{k-1} + 1]} \right\} \\ &- \frac{y_n t_n}{(\phi t_n + 1)\ln(\phi t_n + 1)} = 0, \end{aligned} \tag{12}$$

となる．ゆえに，式 (12) から求めた ϕ を使って，$\lambda_0 = \phi/(\theta - P)$ と，

$$y_n = \frac{1}{\theta - P} \ln(\phi t_n + 1), \tag{13}$$

の関係から，モデルパラメータ λ_0 および θ を求めることができる．

3.3 ソフトウェア信頼性評価尺度

式 (9) の平均値関数をもつ NHPP モデルから，種々のソフトウェア信頼性評価のための定量的尺度を導出できる．

瞬間フォールト発見率は強度関数により表すことができる．これは，単位時間当りに発見されるフォールト数として定義される．瞬間フォールト発見率は，式 (9) から以下のように導出できる．

$$\mu_d(t) = \frac{d\mu(t)}{dt}. \tag{14}$$

平均ソフトウェア故障発生時間間隔（mean time between software failures: MTBF）は，ソフトウェア 故障の発生頻度を表すのに有益な尺度である．また，MTBF が大きな値を取ることは，それだけフォールトが発見し難くなり，ソフトウェア信頼性が向上したと判断できることになる．任意の時刻 t における累積 MTBF（cumulative MTBF: $\mathrm{MTBF_C}$）は，以下のように導出できる．運用開始時点から考えたときの発見フォールト 1 個当りに要する発見時間の平均を意味する累積 MTBF は，

$$MTBF_C(t) = \frac{t}{\mu(t)}, \tag{15}$$

により表すことができる．

3.4 モデルの適合性評価

モデルの実測データに対する適合性を評価するために予測相対誤差を適用する．

予測相対誤差（predicted relative error）は，任意の時刻 t_k において推定したときの，フォールト報告終了時刻 t_K までに発見される累積発見フォールト数の予測値と実測値との相対誤差である．時刻 t_k における予測相対誤差を $PRE_k[t_k]$ とすると，

$$PRE_k[t_k] \quad = \quad \frac{\hat{y}(t_k;t_K) - y_K}{y_K}, \qquad (16)$$

により計算される．$\hat{y}(t_k;t_K)$ は，時刻 t_k までの実測データを用いたフォールト報告終了時刻 t_K での総期待発見フォールト数の推定値，y_K は，フォールト報告終了時刻 t_K までに発見された総フォールト数の実測値である．

4 OSS に対する信頼性評価ツールの開発

本ツールは，オブジェクト指向での分析・設計フェーズにおける表記法である Unified Modeling Language（以下 UML と略す）のアクティビティ図およびシーケンス図を用いて設計し，オブジェクト指向型言語の 1 つである Java を用いて開発した．図 1 は，本ツールの信頼性評価の処理手順をアクティビティ図で表現したものである．さらに，本ツールの動的な振舞いを記述したシーケンス図を図 2 に示す．

4.1 要求仕様定義

本ツールの要求仕様を以下に示す．

1. OSS に対する信頼性評価ツールとして，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て信頼性評価を行い，各推定結果をグラフ表示する．
2. システム全体を構成する複数のコンポーネントの重要度を推定するために，ニューラルネットワークを採用する．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，対数型ポアソン実行時間モデルを採用する．
3. 適用モデルに含まれる未知パラメータの推定，実測データに対するモデルの適合性評価，信頼性評価を行う．
4. ニューラルネットワークに基づき各コンポーネントの重要度を推定し，各コンポーネントの重要度を表示する．
5. 実測データと推定値との適合性比較規準としては，予測相対誤差を用いる．
6. ソフトウェア信頼性評価尺度として，総期待発見フォールト数，瞬間フォールト発見率，累積 MTBF を用いる．
7. すべてのグラフは，同時に表示することが可能であるとする．
8. 以上の操作は，GUI によりマウスボタンのクリックにより行う．
9. プログラム記述言語として，オブジェクト指向型言語の 1 つである Java を用いて開発する．

4.2 信頼性評価手順

以下に，OSS に対するソフトウェア信頼性評価手順を示す．

1. 信頼性評価の対象となる OSS を選択し，バグトラッキングシステムからフォールトデータを採取する．
2. 得られたデータから，ツールで読み込むために必要なデータファイルを作成する．
3. ニューラルネットワークにより各コンポーネントの重要度を推定する．モデルに含まれる未知パラメータを最尤法により推定する．
4. 信頼性評価尺度として，総期待発見フォールト数，瞬間フォールト発見率，および累積 MTBF を推定し，結果をグラフ表示する．また，予測相対誤差の推定結果をグラフ表示する．

4.3 ツールの実行例

実際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から採取されたフォールトデータを適用した本ツールの実行例を示す．本論文で提案された信頼性評価法の機能評価を行うために，Mozilla.org

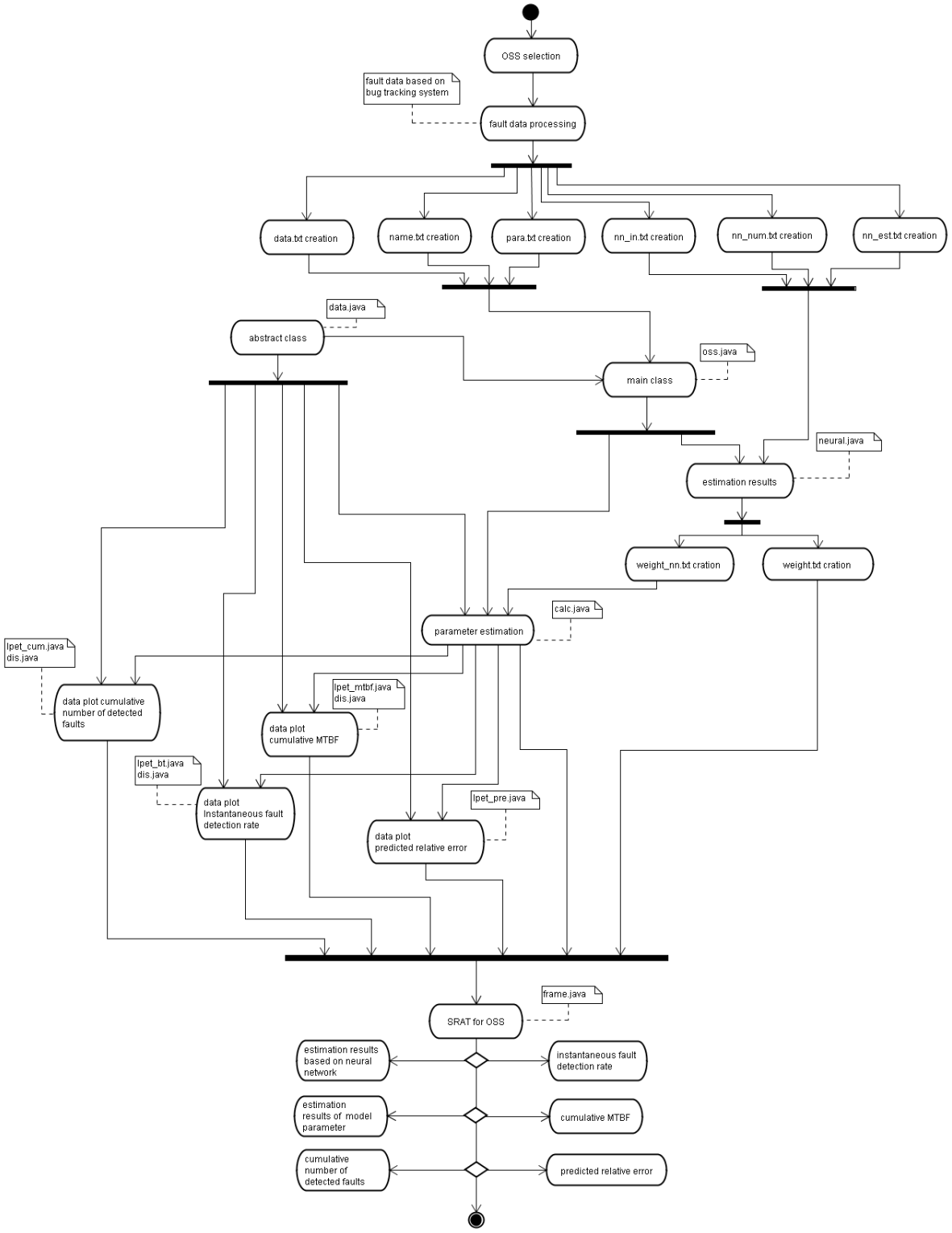

図 1： 本ツールにおけるアクティビティ図.

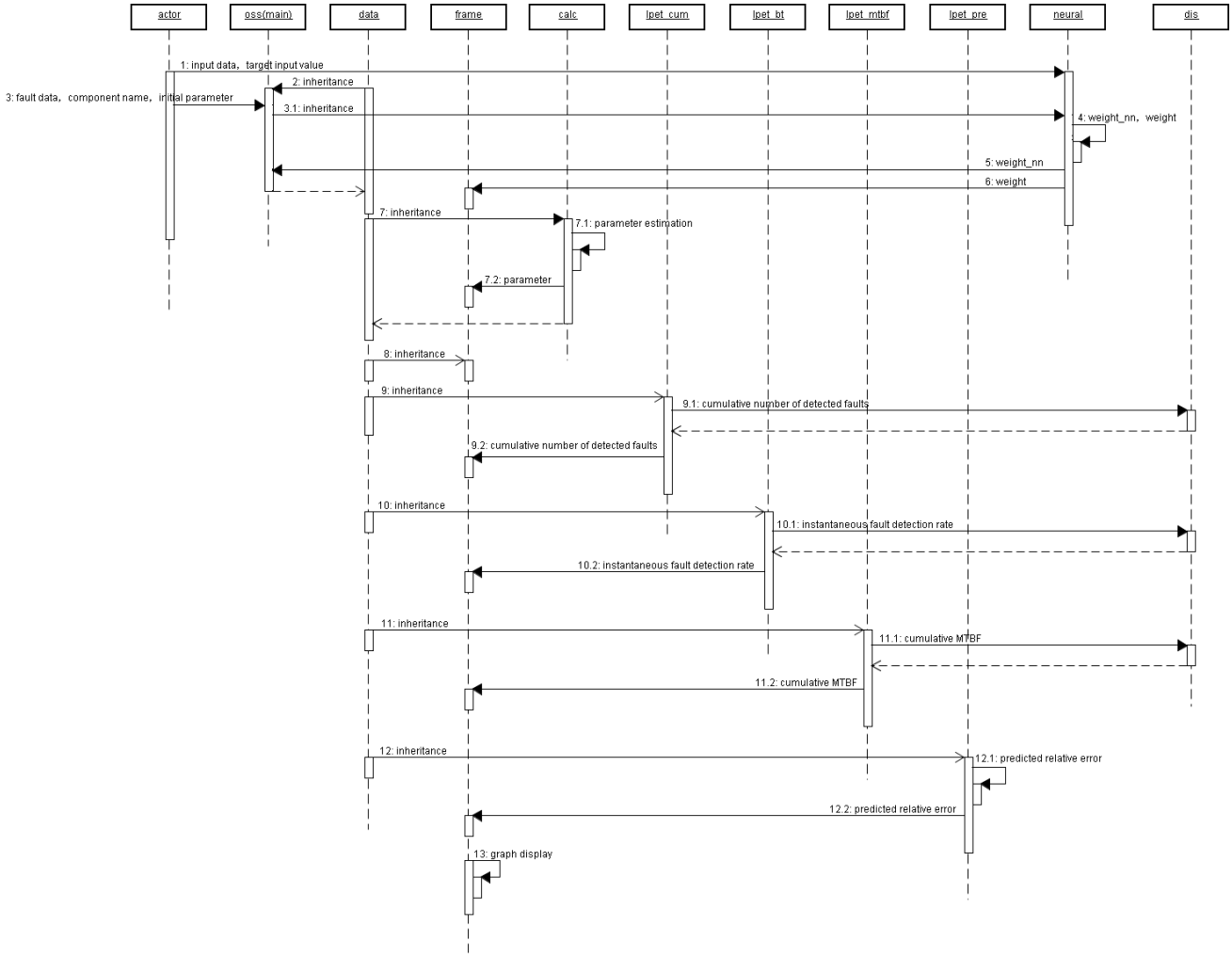

図 2： 本ツールにおけるシーケンス図.

プロジェクト[2]で開発が進められているスケジューラの Calendar[3] [7] を取り上げる.

ソフトウェアの信頼性を評価するために，所定の時間区間内に発見されるフォールト数またはソフトウェア故障発生数のデータに基づいた手法がとられているという歴史的経緯 [6] から，本論文におけるニューラルネットワークの出力データと教師信号には，各コンポーネントに対する累積フォールト発見数データを適用する．これにより，各コンポーネントの相互作用の状態をブラックボックスとして捉え，実際の OSS 開発環境においても適用可能性の高い手法として利用できるものと考える．

まず，ニューラルネットワークにより推定された各コンポーネントに対する重み係数の推定結果および式 (9) におけるモデルパラメータの推定結果を図 3 に示す．この結果から，Sunbird and Calendar-Extension Front End コンポーネントの信頼性に関する重要度が最大であり，Security コンポーネントの重要度が最低であることが分かる．これは，Sunbird and Calendar-Extension Front End コンポーネントの成熟度が高く，Security コンポーネントの成熟度が低いことを意味する．また，推定された式 (9) における累積フォールト発見数の期待値の推定値 $\widehat{\mu}(t)$ を図 4 に示す．また，式 (14) の瞬間フォールト発見率の推定値 $\widehat{\mu_d}(t)$，式 (15) における累積 MTBF の

[2]Mozilla と Mozilla のロゴは Mozilla Foundation の登録商標である.

[3]Calendar と Calendar のロゴは Mozilla Foundation の米国及びその他の国における商標または登録商標である.

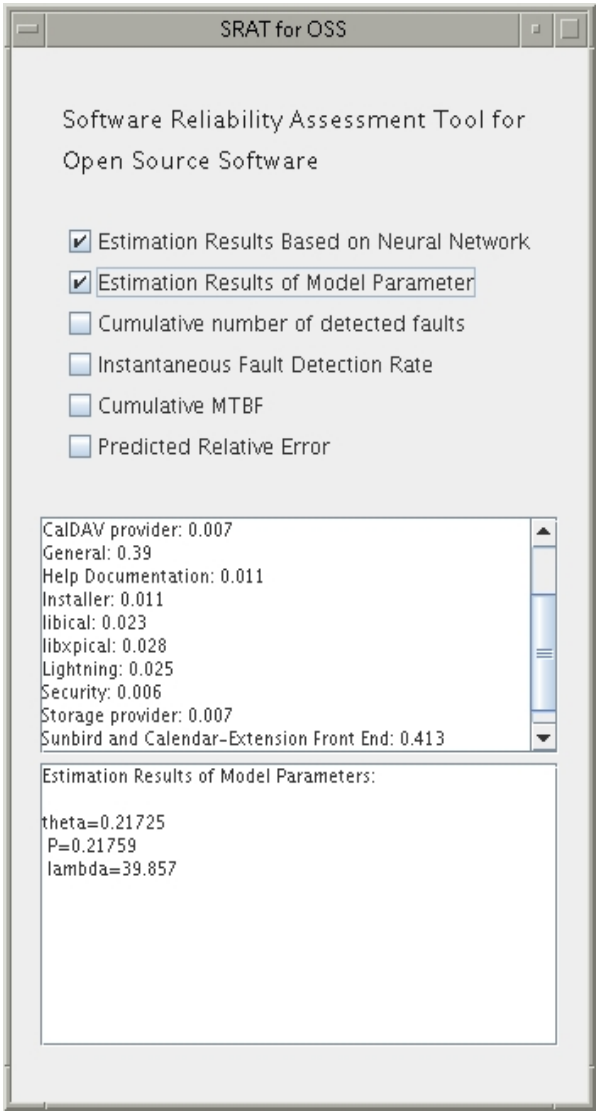

図 3： 推定された各コンポーネントに対する重要度とモデルパラメータの推定結果.

推定値 $\widehat{MTBF}_C(t)$ の推定結果を図 5 および図 6 に示す. これらの結果から，時間の経過とともに平均故障発生時間間隔が小さくなり，今後もソフトウェア故障が頻繁に発生することが確認できる．さらに，予測相対誤差の推定結果を図 7 に示す．図 7 から，実測値と推定値との誤差は，運用進捗率 20%以降において安定していることが確認できる.

同様に，その他の小規模なソフトウェアに適用した信頼性評価結果の一例を示す．Mozilla.org プロジェクト

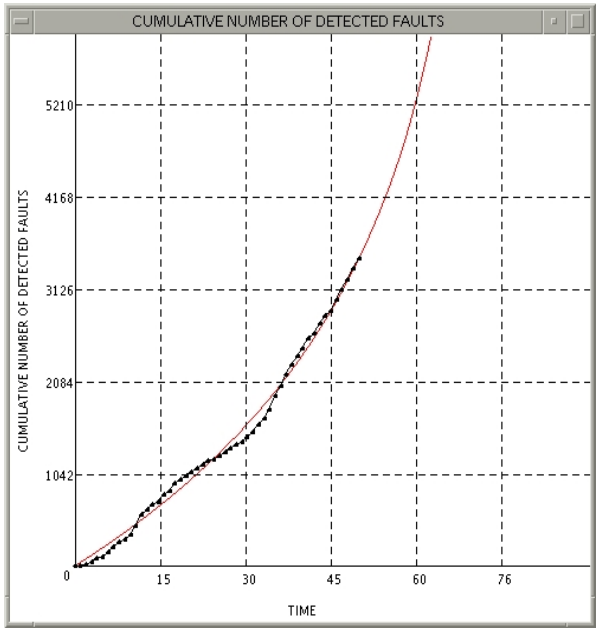

図 4： 推定された累積フォールト発見数の期待値，$\widehat{\mu}(t)$.

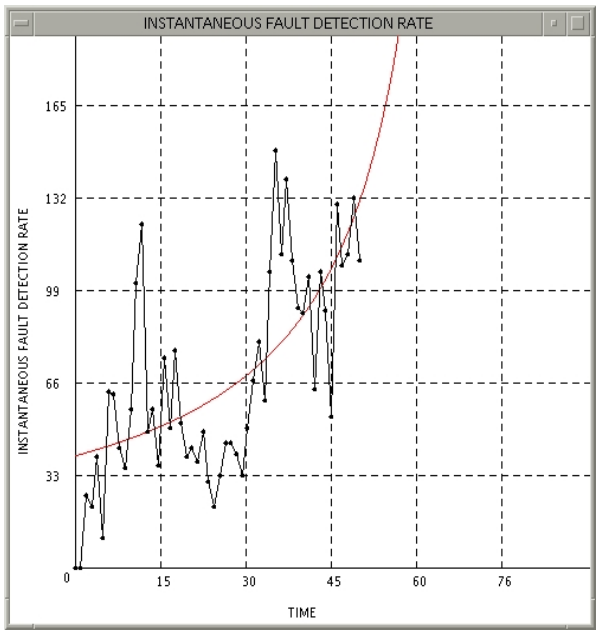

図 5： 推定された瞬間フォールト発見率，$\widehat{\mu_d}(t)$.

で開発が進められているメーラの Thunderbird[4] [8]，さらに仮想マシン環境を実現するためのソフトウェアであ

[4]Thunderbird と Thunderbird のロゴは Mozilla Foundation の米国及びその他の国における商標または登録商標である.

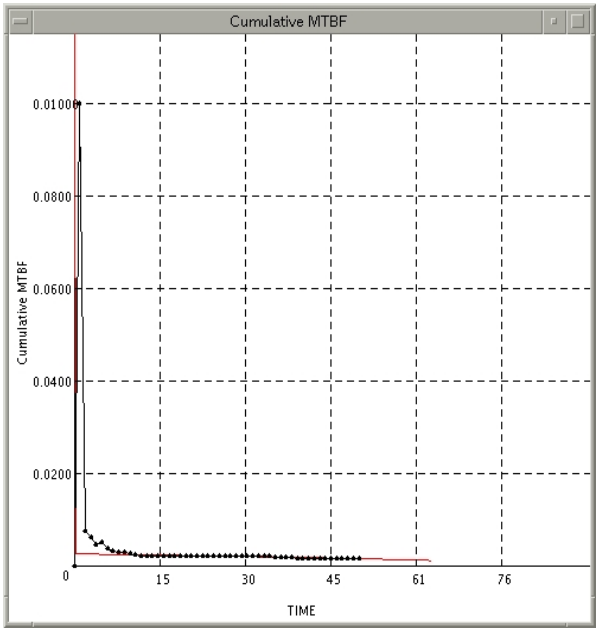

図 6： 推定された累積 MTBF，$\widehat{MTBF}_C(t)$.

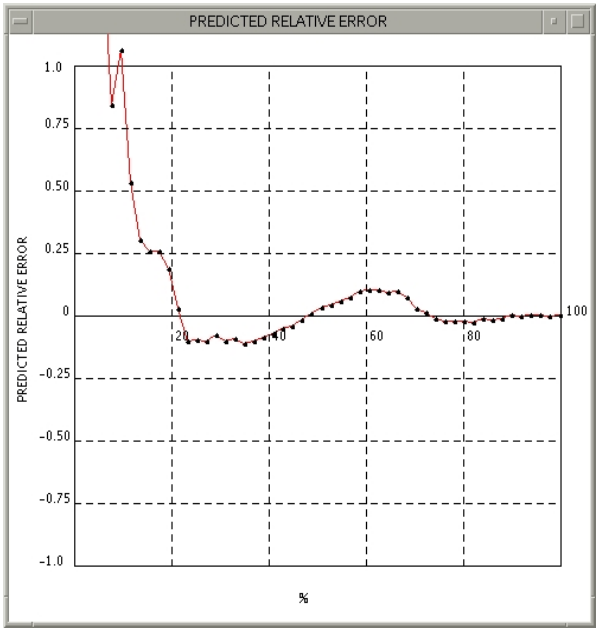

図 7： 推定された予測相対誤差.

り米 XenSource の下で開発されている Xen[5] [9] を取り上げて，その推定された式 (9) における累積フォールト

[5]その他の製品名および会社名は，各社の商標または登録商標である．

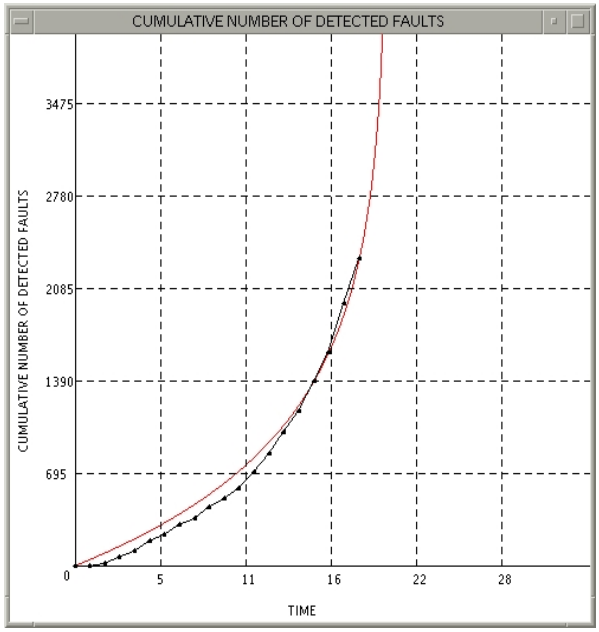

図 8： 推定された累積フォールト発見数の期待値，$\widehat{\mu}(t)$（Thunderbird）.

発見数の期待値の推定値 $\widehat{\mu}(t)$ を図 8 および図 9 に示す.

5 おわりに

オープンソースという開発スタイルは，今後，何らかの形で市場で大きな流れを作っていくものと考えられる [1]．この流れを阻害する大きな要因として，サポートや品質上の問題が挙げられる．

本論文では，こうした問題を解決するために，OSS に対する信頼性評価法を提案するとともに，本手法の数理モデルとしての中身を理解していなくとも容易に使用できるように OSS に対する信頼性評価ツールとして実装した．特に，ニューラルネットワークと SRGM を融合することにより，各コンポーネント間の相互作用を包括した信頼性評価法について議論した．また，実際にバグトラッキングシステムから採取されたフォールト発見数データに対する本ツールの実行例を示した．

OSS の開発は世界中に分散する誰もが開発に参加できる環境である一方，その信頼性向上に関する取り組みは，フォールト報告に基づいて修正作業を行うのみといったのが現状である．OSS では信頼性を動的かつ定量的に評価するという試みが行われていなかったことから，本

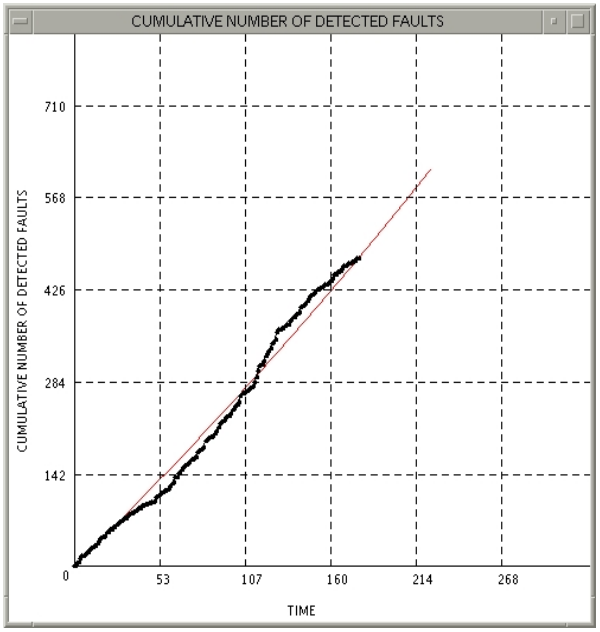

図 9： 推定された累積フォールト発見数の期待値，$\widehat{\mu}(t)$（Xen）．

論文において新たに提案された信頼性評価法を OSS に適用することによって，信頼性に関するなんらかの指標を提示する切っ掛けになるとともに，より高品質な OSS の開発に結びつくものと考える．

本論文では，信頼性評価の一例として小規模な OSS に対する信頼性評価例を示したが，今後は大規模な OSS に対する本手法の適用可能性についても考察する必要がある．大規模な OSS では，バグトラッキングシステム上に登録されているデータも膨大になることから，そのデータを処理するためのシステムを構築する必要がある．また，本研究ではバグトラッキングシステム上に登録された限られた情報だけに基づいた信頼性評価手法を提案しているが，その他にも，ユーザ数または人気度などを考慮した信頼性評価法を提案する必要があると考える．

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 18510124）および若手研究 (B)（課題番号 17700039）の援助を受けたことを付記する．

参考文献

[1] 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ェアの利用状況調査／導入検討ガイドラインの公表について, 東京, 2004.

[2] T. Satty, *The Analytic Hierarchy Process*, McGraw-Hill, New York, 1980.

[3] Y. Tamura and S. Yamada, “Comparison of software reliability assessment methods for open source software,” *Proceedings of the 11th IEEE International Conference on Parallel and Distributed Systems (ICPADS2005)–Volume II*, Fukuoka, Japan, July 20–22, 2005, pp. 488–492.

[4] 田村慶信, 山田茂, 木村光宏, “オープンソース共同開発環境に対するソフトウェア信頼性評価法に関する考察,” 電子情報通信学会論文誌，vol. J88–A，no. 7，pp. 840–847, 2005.

[5] E. D. Karnin, “A simple procedure for pruning back-propagation trained neural networks,” *IEEE Trans. Neural Networks.*, vol. 1, pp. 239–242, 1990.

[6] 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.

[7] Mozilla Calendar, Calendar, http://www.mozilla.org/projects/calendar/

[8] The Mozilla Thunderbird Mail Project, Thunderbird, http://www.mozilla.org/projects/thunderbird/

[9] XenSource, Inc., XenSource: The Art of Virtualization, http://www.xensource.com/