鳥取環境大学紀要
第 4 号（2006.2）pp.129 －137

オープンソースソフトウェア開発における信頼性評価：モデルと応用

Reliability Assessment in Open Source Software Development: Model and Application

田村　慶信　　**山田　茂***

TAMURA Yoshinobu　　YAMADA Shigeru

和文要旨：最近のソフトウェア開発は、クライアント／サーバ・システムや Web プログラミング、オブジェクト指向開発、ネットワーク環境での分散開発といったネットワーク技術を基盤とする新しい開発形態が多用されるようになってきている。ネットワーク型開発形態の代表的な成功例として、オープンソースソフトウェア（open source software、以下 OSS と略す）が挙げられる。これは、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り、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。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たオープンソースソフトウェアに対する信頼性評価法について議論する。特に、AHP またはニューラルネットワークとソフトウェア信頼度成長モデルを融合した信頼性評価法について議論する。さらに、実際のオープンソースソフトウェアに対する本手法の適用可能性と今後の課題について考察する。
【キーワード】信頼性、オープンソースソフトウェア、AHP、ソフトウェア信頼度成長モデル

Abstract ： Recent software development environment has been changing into new development paradigms such as concurrent distributed development environment and the so-called open source project by using network computing technologies. In order to consider the effect of each software component on the reliability of an entire system under such open source project, we discuss a new approach to software reliability assessment by creating a fusion of AHP and neural network with a software reliability growth model. Moreover, we consider the effeciency and effectiveness of the software reliability assessment method for an actual open source project.
【Keywords】 Software reliability, open source project, AHP, reliability growth model

１．はじめに

21 世紀は情報技術（IT）の世紀であると言われている。環境に負荷の少ない社会経済を実現するためには、企業内通信網によるペーパーレス化、企業活動での生産・調達・運用支援統合情報システムの導入による生産・流通の効率化など、環境改善に向け様々な分野で情報システムの開発が重要となる。このように、ネットワーク技術を基盤とする情報システムは持続可能な社会の実現のためにも必須である。特に、近年はインターネットの普及により世界中が同時に新しい情報を得ることができるようになっている。こうした状況から、ネットワーク技術を基にしたソフトウェア製品の分散開発、およびソフトウェアそのものの分散化がさらに拡大してきた。特に、最近のソフトウェア開発は、クライアント／サーバ・システムや Web プログラミング、オブジェクト指向開発、ネットワーク環境での分散開発といった新しい開発形態が

＊ 鳥取大学工学部社会開発システム工学科　教授

多用されるようになってきている[1,2]。現在、分散ソフトウェア共同開発は、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ンソースプロジェクトなどの様々な形態が存在する[3]。こうしたソフトウェアの分散化が進む一方で、現在のところ分散開発されたソフトウェアシステムに対する有効なテスト管理方法は提案されていない。

特に、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 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[4]。特に、クライアント用途では商用ソフトウェアが市場シェアの多くを占めているが、サーバ用途に限ってみると OSS が圧倒的な市場シェアを占めている。その一例として、GNU/Linux オペレーティングシステム[1]や Apache ウェブサーバなど[2]が挙げられる。2005 年 7 月の活動的ウェブサーバの市場シェアの統計では、Apache72.1%、Microsoft 22.26%、Zeus 0.79%、Netscape 0.62%、WebSTAR 0.34%、WebSite 0.10%、その他 3.78% と、Apache が支配的であると示されている[5]。さらに、組込みシステム用 OS のシェアについては、トロン協会 ITRON 部会により実施されているリアルタイム OS の利用動向と ITRON に関するアンケート調査結果によれば、自社用の OS（半数弱が ITRON 仕様 OS）が 22.4%、半導体メーカ製の OS（大半が ITRON 仕様 OS）が 9.0%、その他（主にソフトウェアメーカ製の OS であり、その大半が ITRON 仕様 OS）が 19.8% となっている。オープンソース OS として知られている ITRON が、組込みシステムの分野において多くの市場シェアを占めていることが分かる[6]。現在、OSS はサーバ用途や組込みシステムとして利用されているだけではなく、アプリケーションソフトウェアなどの様々な分野で広く利用されている。

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[4]。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いる。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 OSS に対して、テスト管理に関する問題の 1 つとして信頼性評価法について議論するとともに、実際に公開されている OSS に対する信頼性評価法の適用可能性と、その有効性について考察する。

本論文では、信頼性評価を実施する際に、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モジュールの重要度を推定するにあたり、AHP（Analytic Hierarchy Process）[7]およびニューラルネットワーク[8]を適用する。これにより、システム全体に与える各コンポーネントの信頼性に関する影響度合いを推定することが可能となる。また、システム全体に対しては従来から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）に基づいた信頼性評価法を適用する。これにより、実際のオープンソースプロジェクトに対して、開発リーダまたはユーザの立場に立った信頼性評価法として利用できるものと考える。

2．開発者指向に基づく各コンポーネントに対する信頼性評価

2－1　SRGM に基づく信頼性評価

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、SRGM による方法がとられている。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のため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関数すなわちポアソン過程を仮定する SRGM である[9]。

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

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。

2 その他記載している会社名、商品名は一般に各社の商標または登録商標である。

ここで、Pr{・}は確率を表し、$H(t)$は時間区間 (o,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。
本論文では、各コンポーネントについて累積発見フォールト数データの成長曲線の形状により、以下に示す NHPP モデル[9]のうち最適なモデルを適用する。

▶　指数形 SRGM

▶　習熟 S 字形 SRGM

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、ソフトウェア信頼性評価のために有用な種々の定量的尺度を導出できる。

2 － 2　AHP に基づく重み係数の推定

1970 年代に開発された AHP 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[7]。

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把握す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。したがって本論文では、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ω_i (i=1,2,・・・, n)とすれば、一対比較行列は、
となる。各評価基準に対するウエイトを次式の幾何平均

$$A = \begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n} \end{bmatrix}, \qquad (2)$$

により求めることができる。

$$\begin{aligned} \alpha_i &= \sqrt[n]{\prod_{j=1}^{n} x_{ij}}, \\ x_{ij} &= \frac{w_i}{w_j}. \end{aligned} \qquad (3)$$

以上のことから、各評価基準に対するウエイトは、
により与えられる。

$$\beta_i = \frac{\alpha_i}{\sum_{i=1}^{n} \alpha_i}, \qquad (4)$$

このβ_iは各評価基準同士のウエイトを表す。同様にして、式(4)のウエイトβ_iから、各コンポーネントの重要度を表す重み係数 pi を推定することができる[10]。

3.　ユーザ指向に基づく各コンポーネントに対する信頼性評価

2 － 2 で議論した AHP を使用することによりソフトウェア開発者の主観を考慮した信頼性評価法[10]は、オープンソースプロジェクトにおいては、実質上のリーダが特定しづらいことから、主観的判断に基づく信頼性評価結果が曖昧となり、実利用上において適用し難いことが考えられる。

本論文では、こうした複雑な状況下において、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度合いを考慮するために、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。すなわち、各コンポーネントの状態がシステム全体の信頼性に影響を及ぼすことを前提として、各コンポーネント間の相互作用をブラックボックスとして捉える。つまり、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、各コンポーネントのシステム全体の信頼性に関する影響度合いを推定する。

本論文においては簡単のために図 2 に示すような 3 層ニューラルネットワークを適用する。まず、ω_{ij}^{1}(i = 1, 2, ・・・, I ; j =1, 2,・・・, J)を入力層と中間層の結合係数、またω_{ij}^{2}(j =1, 2, ・・・, J ; k = 1, 2,・・・,K)は中間層と出力層の結合係数とし、x_i (i = 1, 2,・・・, I)は正規化された入力データを表す。

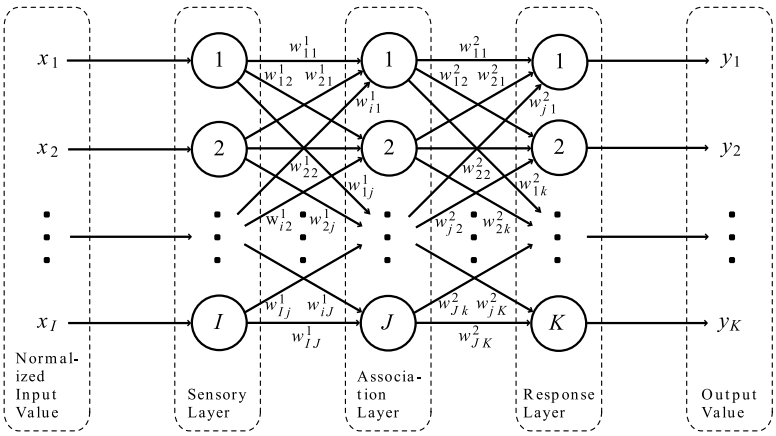

図 1　本論文における 3 層ニューラルネットワークの構造

ここで、入力データ x_i には、バグトラッキングシステムから採取されたデータを適用する．バグトラッキングシステム上に登録されている内容としては、フォールト報告の日付、フォールトレベル、フォールト修正者、フォールト報告者、フォールトの修正状態、各フォールトに対するコンポーネント名、バージョン情報、障害内容がある。本論文では、この中でも特にシステム全体の信頼性に関して影響を与えるとともに、データとしての取り扱いやすさの観点から、致命的であると判断されたフォールト数、フォールト発見時における特定 OS の数、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、中間層と出力層のユニットの出力を h_j , y_k とすると、

$$h_j = f\left(\sum_{i=1}^{I} w_{ij}^1 x_i\right), \tag{5}$$

$$y_k = f\left(\sum_{j=1}^{J} w_{jk}^2 h_j\right), \tag{6}$$

となる。但し、f(・) はシグモイド型関数であり、

$$f(x) = \frac{1}{1+e^{-\theta x}}, \tag{7}$$

として表される。ここで、θ はしきい値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, ・・・ ,K) とし、教師パターンを d_k(k = 1, 2, ・・・ ,K) とすると、式(6) の y_k の評価は

$$E = \frac{1}{2}\sum_{k=1}^{K}(y_k - d_k)^2, \tag{8}$$

により評価される。ここで、教師パターン d_k (k = 1, 2, ・・・,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、各ソフトウェア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結合状態の推定・予測が可能なモデルを考える。式(8) の条件のもとに、結合係数が最急降下法にて決定される。

4. 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 [9]。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 [11]。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。故障強度はテストにおいて発生するソフトウェア故障数に関して指数関数的に減少するものと仮定すると、

$$\frac{d\mu(t)}{dt} = \lambda_0 e^{-(\theta - P)\mu(t)}, \tag{9}$$

と表される。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。式(9) から、初期条件 $\mu(0) = 0$ の下で $\mu(t)$ に関して解くと、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、

$$\begin{aligned}\mu(t) &= \frac{1}{\theta - P}\ln[\lambda_0(\theta - P)t + 1] \\ & \quad subject\ to\ \ (P-\theta) < \frac{1}{\lambda_0 t} \\ & \quad (0 < \theta,\ 0 < \lambda_0,\ 0 < P < 1), \qquad (10)\end{aligned}$$

により与えられる。ここで、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、開発者指向に基づく場合には、各コンポーネントに対して推定されたパラメータ b_i と 2-2 の AHP 手法により推定された重みパラメータ p_i との重み付き平均により表されるものとし、

$$P = \sum_{i=1}^{n} p_i \cdot b_i, \qquad (11)$$

により定義する。ここで、n はソフトウェアシステムのコンポーネント数を表す。さらに、p_i は各コンポーネントに対する重みパラメータを表し、システム全体に対する各コンポーネントの重要度を表す。また、b_i は i 番目のコンポーネントに対する指数形 SRGM および習熟 S 字形 SRGM に含まれるフォールト発見率を表す定数パラメータを表す。一方、ユーザ指向に基づく場合には、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重みパラメータ $p_i(= y_k)$ の平均値により表されるものとする。

4 － 1 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、

$$\begin{aligned}\ln L &= \sum_{k=1}^{K}(y_k - y_{k-1})\ln\left[\mu(t_K) - \mu(t_{k-1})\right] \\ & \quad - \mu(t_K) - \sum_{k=1}^{K}\ln\left[(y_k - y_{k-1})!\right], \qquad (12)\end{aligned}$$

となる。ここで、式(10) 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(12) を偏微分する。このとき、

$$\frac{\partial \ln L}{\partial \lambda_0} = \frac{\partial \ln L}{\partial \theta} = 0, \qquad (13)$$

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(12) は、

$$\begin{aligned}\frac{\partial \ln L}{\partial \phi} &= \sum_{k=1}^{K}(y_k - y_{k-1}) \\ & \quad \cdot \left\{\frac{\left(\frac{t_k}{\phi t_k + 1}\right) - \left(\frac{t_{k-1}}{\phi t_{k-1} + 1}\right)}{\ln[\phi t_k + 1] - \ln[\phi t_{k-1} + 1]}\right\} \\ & \quad - \frac{y_n t_n}{(\phi t_n + 1)\ln(\phi t_n + 1)} = 0, \qquad (14)\end{aligned}$$

となる。ゆえに、式(14) から求めた ϕ を使って、$\lambda_0 = \phi/(\theta - P)$ と、

$$y_n = \frac{1}{\theta - P}\ln(\phi t_n + 1), \qquad (15)$$

の関係から、モデルパラメータ λ_0 および ϕ を求めることができる。

4 － 2 ソフトウェア信頼性評価尺度

式(10) 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(10)から以下のように導出できる。

$$\mu_d(t) = \frac{d\mu(t)}{dt}. \qquad (16)$$

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 における瞬間 MTBF（instantaneous MTBF: $MTBF_I$）および累積 MTBF（cumulative MTBF: $MTBF_C$）は、以下のように導出できる。

任意の時刻 t における瞬間的なフォールト発見間隔の平均を意味する瞬間 MTBF は、

$$MTBF_I(t) = \frac{1}{d\mu(t)/dt}, \qquad (17)$$

表 1　Xfce の AHP に基づく各コンポーネントに対する重み係数の推定結果

Component	Weight parameter p_i
general	0.055219
other	0.44820
xfce4	0.091727
xfdesktop	0.18165
xffm	0.11970
xfwm	0.10351

となる。

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、

$$MTBF_C(t) = \frac{t}{\mu(t)}, \tag{18}$$

により表すことができる。

5. 数値例

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数値例を示す。

5 － 1 開発者指向に基づく信頼性評価結果

Xfce[12] と呼ばれる OSS のフォールトデータを適用した数値例を示す。本論文で用いたデータは、6 つのコンポーネントから構成された Xfce におけるバグトラッキングシステムから採取されたものである。

2. 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 $p_i(i = 1, 2, \cdots, n)$ 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、other 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、general 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。

次に、式(10) における累積フォールト発見数の期待値の推定値 $\hat{\mu}(t)$ を図 2 に示す。さらに、式(10) の平均値関数から導出される瞬間フォールト発見率の推定値 $\hat{\mu}_d(t)$、瞬間 MTBF の推定値 $\widehat{MTBF}_I(t)$ および累積 MTBF の推定値 $\widehat{MTBF}_C(t)$ の推定結果を図 3 から図 5 に示す。これらの結果から、瞬間 MTBF および累積 MTBF は両者ともに、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。

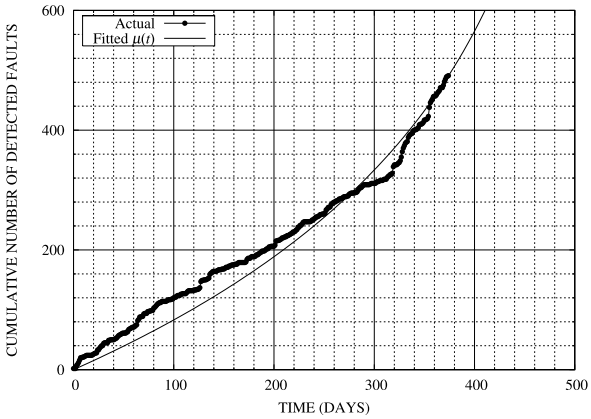

図 2　推定された累積フォールト発見数の期待値, $\hat{\mu}(t)$

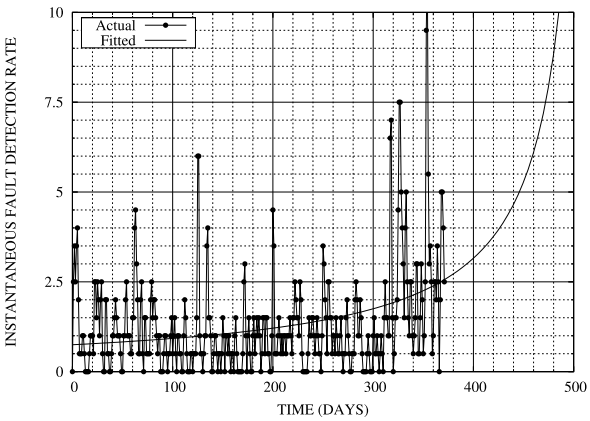

図 3　推定された瞬間フォールト発見率, $\hat{\mu}_d(t)$

5 － 2　ユーザ指向に基づく信頼性評価結果

11 個の主要コンポーネントから構成される Thunderbird[13)3] と呼ばれる Mailer の OSS を取り上げる。各コンポーネントに関して、システム全体の信頼性に影響を及ぼすと考えられる要因別データ一覧を表 2 に示す。ニューラルネットワークを適用するにあたり、表 2 におけるデータを入力データとした。**3.**のニューラルネットワークに基づく各 OSS の各コンポーネントに対する重みパラメータ $p_i(i = 1, 2, \cdots, n)$の推定結果を表 3 に示す。表 3 から、Mail Window Front End 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、Help Documentation 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。

次に、システム全体に対する信頼性評価結果の一例を

3 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す。

表 2　Thunderbird の各コンポーネントに対する要因別データ

Component Name	Number of Fault Level (Critical)	OS (Windows)	Assigned to (mscott)	Reporter (floeff)	Total Number of Faults
Account Manager	9	88	127	1	130
Address Book	4	59	102	3	105
Build Config	1	3	14	0	15
General	72	293	538	3	558
Help Documentation	0	1	3	0	5
Installer	3	28	33	1	33
Mail Window Front End	58	578	982	6	1013
Message Compose Window	15	127	227	6	233
Migration	3	36	51	0	51
Preferences	2	37	92	5	96
RSS	3	45	76	0	78
Total	170	1295	2245	25	2317

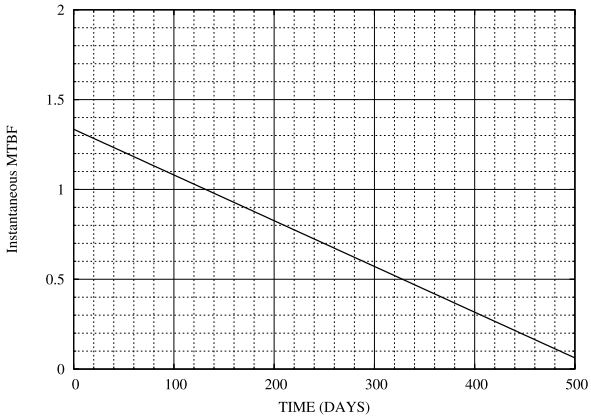

図 4　推定された瞬間 MTBF，$\widehat{MTBF_I}(t)$

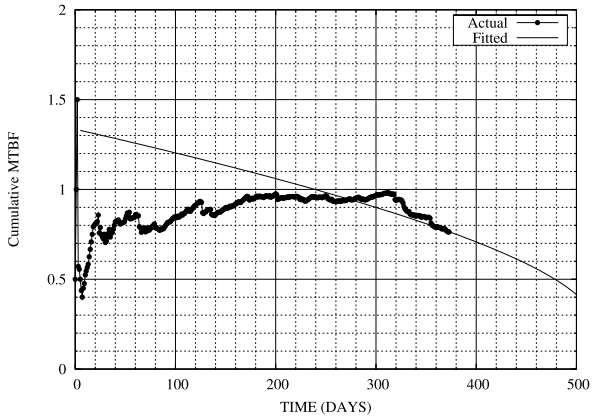

図 5　推定された累積 MTBF，$\widehat{MTBF_C}(t)$

示す。式(10) における累積フォールト発見数の期待値の推定値 $\mu_d(t)$を図 6 に示す。さらに、式(10) の平均値関数から導出される瞬間フォールト発見率の推定値 $c\ \mu_d(t)$、瞬間 MTBF の推定値 $MTBF_I(t)$ および累積 MTBF の推定値 $MTBF_C(t)$ の推定結果を図 7 から図 9 に示す。これらの結果から、瞬間 MTBF および累積 MTBF は両者ともに、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。

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 OSS に対する信頼性評価法について議論した。特に、開発者指向として意思決定手法の 1 つである AHP 手法を適用し、ユーザ指向としてニューラルネットワークを適用することにより、各コンポーネントに対する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、実際の OSS の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数値例を示した。

開発者指向の信頼性評価法の特徴としては、開発者しか知り得ない情報、すなわち開発者の主観を取り込んだ信頼性評価法として利用できることが挙げられる。しかしながら、OSS の開発では主要な開発メンバー（開発リーダ）が複数いることから AHP の評価基準値の与え方が問題となる場合が多い。一方、ユーザ指向の信頼性評価法については、実際に OSS を使用するユーザ側の視点に立った信頼性評価法として、ニューラルネットワークを使用することにより機械的かつ容易にコンポーネントの重要度を推定することが可能となる。しかしながら、ニューラルネットワークの各ユニットにコンポーネント数を割り当てているため、コンポーネント数が多くなると、学習の収束に影響がある。

このように、各手法に関してメリットとデメリットが

表 3　Thunderbird の各コンポーネントに対する重み係数

Component Name	Weight parameter
Account Manager	0.0537
Address Book	0.0487
Build Config	0.0091
General	0.2377
Help Documentation	0.0054
Installer	0.0138
Mail Window Front End	0.4347
Message Compose Window	0.0951
Migration	0.0266
Preferences	0.0425
RSS	0.0327

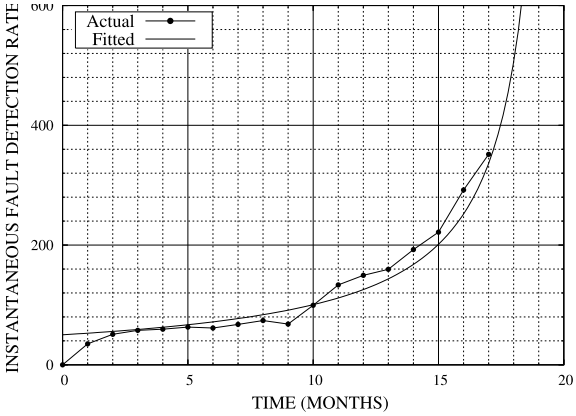

図 7　推定された瞬間フォールト発見率，$\widehat{\mu_d}(t)$

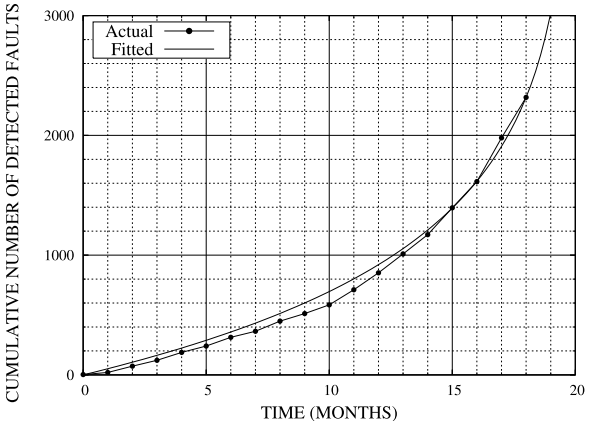

図 6　推定された累積フォールト発見数の期待値 $\widehat{\mu}(t)$

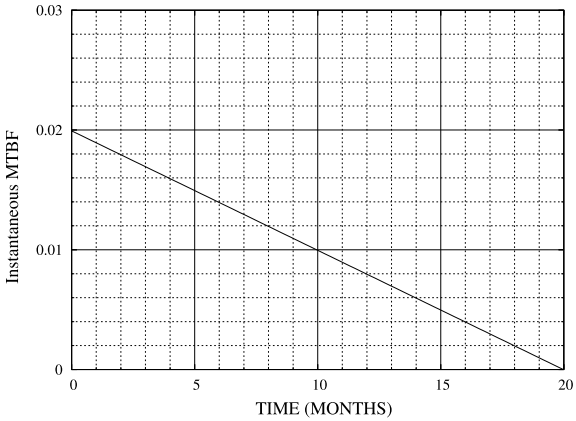

図 8　推定された瞬間 MTBF，$\widehat{MTBF_I}(t)$

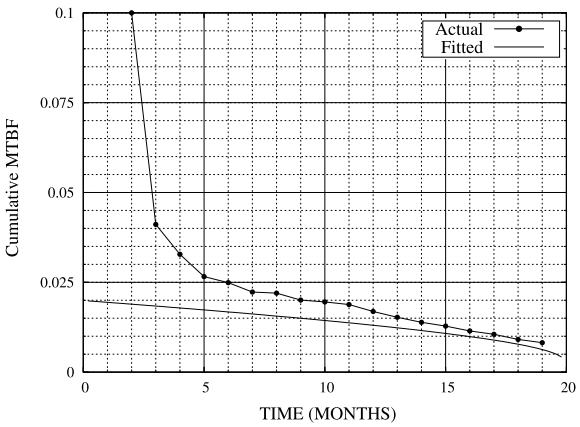

図 9　推定された累積 MTBF，$\widehat{MTBF_C}(t)$

あるが、両者ともにシステム全体の信頼性に与える各コンポーネントの重要度を定量的に導出することが可能となり、導出された各コンポーネントのウエイトから各コンポーネントに対する発見フォールト数の推定が可能となることから、OSS の信頼性を定量的に把握するための手法として有効であると考える。

ソフトウェアは、その内容の質の高低によって採否が決められるべきものであり、オープンソースであるか否かによってその優劣が決まるものではない。しかしながら、ソフトウェアが、開発者やユーザなどの様々な立場を超えて知見を結集する必要があるものであることを考えると、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 [4]。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、オープンソースプロジェクトの下で開発された OSS に対する信頼性評価法の 1 例を示した。

将来的には、本手法をソフトウェアツールとして実装することにより、多くのユーザに対して容易に扱うことが可能な信頼性評価ツールとして提供していくことが重要であると考える。これまで、OSS では信頼性を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、本論文において新たに提案された信頼性評価手法を OSS

に適用することによって、より高品質な OSS の開発に結びつくものと考える。

謝辞

本論文の一部は、文部科学省科学研究費基盤研究(C)(2)（課題番号 15510129）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) A. Umar(1993), Distributed Computing andClient-Server Systems, Prentice Hall, New Jersey.
2) 松本正雄、 小山田正史、 松尾谷徹(1997)「ソフトウェア開発検証技法」『電子情報通信学会』
3) 赤羽豊和(1998)「クライアント/サーバ・システムのテスト技法」『ソフト・リサーチ・センター』
4) 「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」『ソフトウェア情報センター研究会報告書』(2004)
5) E-Soft Inc., Internet Research Reports, http://www.securityspace.com/s survey/data/index.html
6) トロン協会 ITRON 仕様検討グループ ITRON Project Archive, http : //www. sakamura-lab. org/TRON/ITRON/ home-j. html
7) T. Satty(1980) *The Analytic Hierarchy Process*, McGraw-Hill, New York.
8) E. D. Karnin(1990) “A simple procedure forpruning back-propagation trained neural net-works,” *IEEE Trans. Neural Networks.*, vol. 1, pp. 239-242.
9) 山田茂(1994)『ソフトウェア信頼性モデル－基礎と応用－』日科技連出版社
10) 田村慶信、 山田茂、 木村光宏(2005)「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.
11) Y. Tamura and S. Yamada(2005) “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, pp. 488-492.
12) Olivier FOURDAN, Xfce - Desktop Environment, http://www.xfce.org/
13) The Mozilla Thunderbird Mail Project, Thunderbird, http://www.mozilla.org/projects/thunderbird/

（2006 年 1 月 11 日受理）