数理解析研究所講究録
第2078巻 2018年 147-153

ソフトウェア信頼性評価と最適リリース時刻推定のための NW.jsに基づくアプリケーションソフトウェアの開発

東京都市大学・知識工学部 田村 慶信（Yoshinobu Tamura）†
†Faculty of Knowledge Engineering, Tokyo City 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

一般的に，ウォーターフォールモデルに代表されるように，要求仕様定義・設計・コーディング・テスト・運用保守の流れでソフトウェアの開発および保守が行われている．特に，ソフトウェア開発のテスト工程は，ソフトウェア品質を定量的に評価するだけでなく，ユーザに対するリリース（出荷）時期の決定にも関係することから，非常に重要な工程となる．従来から，テスト工程におけるソフトウェアの信頼性を評価するための数理モデルとして，これまでに数百におよぶソフトウェア信頼性モデルが提案されてきた [1–3]．本論文では，ソフトウェア信頼度成長モデル（software reliability growth model，以下SRGMと略す）に基づくソフトウェアツールの開発を通して，NW.jsを適用したアプリケーションソフトウェアの実装事例について紹介する．同様のWebアプリケーション開発フレームワークとして統計言語Rによる Shinyが存在する．しかしながら，Shinyの場合には，Rのソースコード部分にグラフ描画に必要なコードを記載する必要がある．さらに，大規模なデータを扱うような場合においては，処理速度などの問題からネイティブアプリケーションとして実行する方が望ましいケースもある．NW.jsは，オープンソースソフトウェア（open source software，以下OSSと略す）[4] のアプリケーション開発フレームワークとして知られており，HTML，CSS，およびJavaScriptで書かれたWebアプリケーションを，Windows・Mac・Linuxをターゲットとしたデスクトップ向けにパッケージングし，ネイティブアプリケーションとして実行するためのクロスプラットフォーム実行環境である．本開発ツールの紹介を通して，研究において利用してきた統計言語Rのソースコードを有効かつ迅速に再利用できることを示す．

特に，ソフトウェア開発のテスト工程を動的に評価するための数理モデルとして，数多くのソフトウェア信頼性モデルが提案されてきた [2]．また，ソフトウェア開発支援ツールも，いくつか提案されている．本論文では，SRGMに基づくソフトウェアツールの開発を通して，最新のアプリケーション開発技術であるNW.jsと，統計言語Rのggplot2およびplotlyライブラリを利用したソフトウェアツールの実装事例について議論する．さらに，実際のフォールトデータを利用したソフトウェアツールの実行例を示す．

2 SRGMに基づくソフトウェア信頼性評価と最適リリース問題の概要

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，実際に企業組織などにおける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過程すなわちポアソン過程を仮定する SRGM である [2].

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

ここで，$\Pr\{\cdot\}$ は確率を表し，$\Lambda(t)$ 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．

本論文において開発されたアプリケーションソフトウェアに採用した SRGM を以下に示す．

- 指数形 SRGM

$$\Lambda(t) \equiv E(t) = a(1-e^{-bt}) \qquad (a>0,\ b>0). \tag{2}$$

ここで，$E(t)$ は時間区間 $(0,t]$ において発見される累積フォールト数の期待値を表す．パラメータ a は最終的に発見される総期待フォールト数，パラメータ b はフォールト 1 個当りのソフトウェア故障発見率またはフォールト発見率を表す．

- 遅延 S 字形 SRGM

$$\Lambda(t) \equiv S(t) = a\left\{1-(1+bt)e^{-bt}\right\} \qquad (a>0,\ b>0). \tag{3}$$

指数形 SRGM の場合と同様に，$S(t)$ は時間区間 $(0,t]$ において発見される累積フォールト数の期待値を表す．

- 対数型ポアソン実行時間モデル

$$\Lambda(t) \equiv \mu(t) = \frac{1}{\theta}\ln\left[\lambda_0\theta t+1\right] \qquad (0<\theta,\ 0<\lambda_0). \tag{4}$$

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．

上述した SRGM から，最適リリース問題として，テスト工程におけるコスト評価基準に基づく最適リリース時刻を推定することが可能となる [5,6]．なお，モデルの性質上，総期待ソフトウェアコストについては，指数形 SRGM および遅延 S 字形 SRGM について議論する．まず，以下のようなコストパラメータを定義する．

c_1: 総合テスト工程におけるフォールト 1 個当りの修正コスト，

c_2: 総合テスト工程における単位時間当りのテストコスト，

c_3: リリース後のフォールト 1 個当りの修正コスト．

このとき，総合テスト工程におけるソフトウェアコストは，以下のように定式化できる．

$$C_1(t) = c_1\Lambda(t) + c_2 t. \tag{5}$$

また，リリース後におけるソフトウェアの修正コストは次式で与えられる．

$$C_2(t) = c_3\left\{a-\Lambda(t)\right\}. \tag{6}$$

上記から，総期待ソフトウェアコストは，以下のように定式化できる．

$$C(t) = C_1(t) + C_2(t). \tag{7}$$

式 (7) を最小にする時刻 t^* が最適リリース時刻となる．

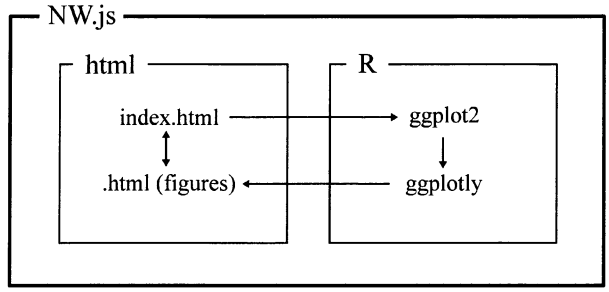

図 1： アプリケーションソフトウェアの構造.

3 アプリケーションソフトウェアの開発

本論文において開発されたソフトウェアツールの特徴は，研究成果を迅速に実社会における現場へ提供できる点にある．特に，様々な分野において利用されている研究用のツールとして，統計言語 R が良く知られている．この統計言語 R において提供されているグラフ描画のためのパッケージとして ggplot2 がある．さらに，この ggplot2 の付属パッケージとして JavaScript グラフライブラリとして知られる plotly が存在する．plotly は，ggplot2 によって作成されたグラフを JavaScript を含む html 形式に出力する．出力されたグラフには，拡大，縮小，特定項目の抽出，画像ファイルとしての出力など，様々なインタラクティブな機能を有している．この 2 種類のライブラリと NW.js を組み合わせることにより，ネイティブアプリケーションを容易に開発することが可能となる．本論文におけるアプリケーションソフトウェアの概念図を図 1 に示す.

本論文において開発されたアプリケーションソフトウェアには，SRGM に基づく信頼性評価尺度および最適リリース問題を考慮するために，以下のような定量的評価尺度を採用する.

- 期待累積発見フォールト数
- 期待残存フォールト数
- MTBF（平均ソフトウェア故障発生時間間隔）
- 総期待ソフトウェアコスト

また，SRGM の実測データに対する適合性を評価するために，赤池情報量規準（Akaike's information criterion，以下 AIC と略す）を用いる.

4 開発されたツールの実行例

実際のソフトウェア開発プロジェクトにおけるフォールトデータ [7] に基づくアプリケーションソフトウェアの実行結果を示す.

本ツールのメイン画面を図 2 に示す．また，本ツールのメニュー画面を図 3 に示す．まず，実際のソフトウェア開発プロジェクトにおけるフォールトデータを分析したアプリケーションソフトウェアの実行例として，AIC の推定結果を図 4 に示す．図 4 から，指数形 SRGM の実測データに対する適合性が，他の SRGM と比較して良好であることが確認できる．また，ソフトウェア信頼性評価尺度および総期待ソフトウェアコストの推定例を図 5〜図 8 に示す．図 5〜図 7 から，全てのモデルにおいて，テスト時間の経過とともに信頼度が成長している様子が確認できる．さらに，図 8 から，指数形 SRGM における最適リリース時刻はテスト開始から 43ヶ月目であり，そのときの総期待ソフトウェアコストは 6534.9 であ

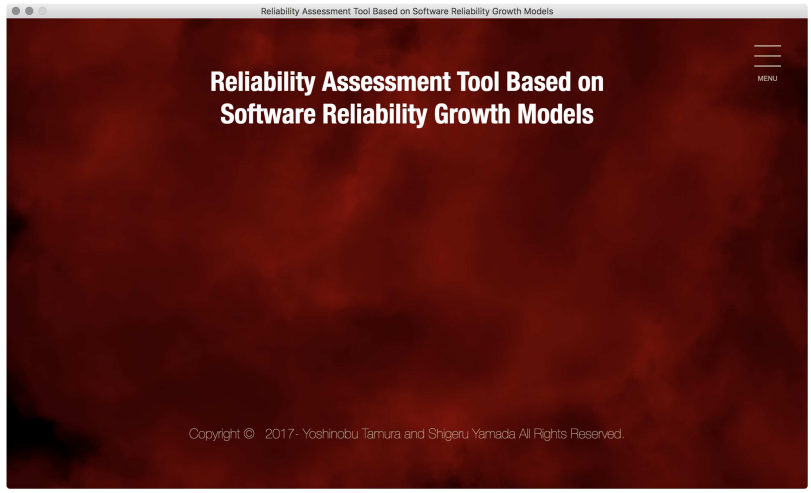

図 2： 開発されたツールのメイン画面.

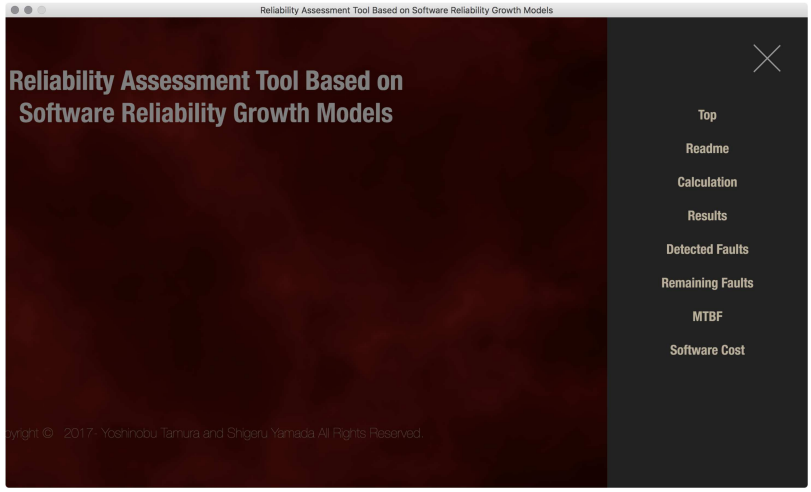

図 3： 開発されたツールのメニュー画面.

ることが分かる．同様に，遅延 S 字形 SRGM における最適リリース時刻はテスト開始から 22ヶ月目であり，そのときの総期待ソフトウェアコストは 5567.1 であることが分かる．

5　おわりに

現在，多くのソフトウェア開発現場で，OSS が利活用されている．特に，OSS を利用することにより，コスト削減，単納期，標準化などが可能となることから，様々なソフトウェア開発プロジェクトにおいても OSS が積極的に採用されている．同様に，研究環境についても，OSS を利用することで，低コストで研究成果を実社会における現場へ迅速に提供することが可能となる．

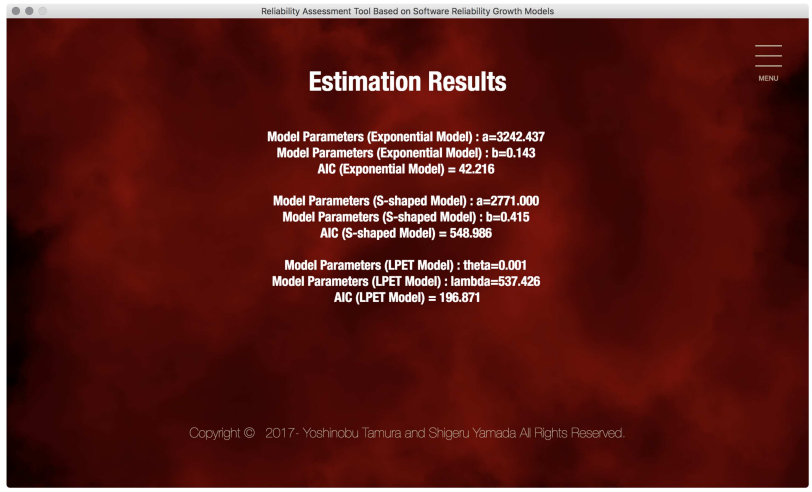

図 4： AIC の推定結果.

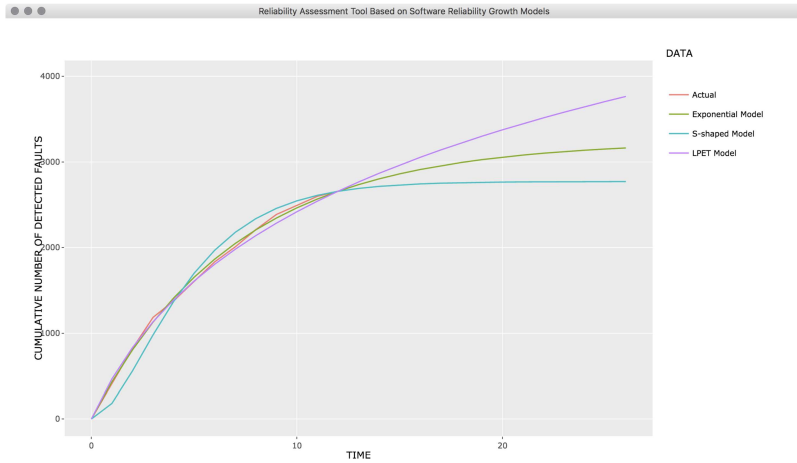

図 5： 推定された累積発見フォールト数.

本論文では，ソフトウェア開発プロジェクトのテスト工程において，ソフトウェア品質を定量的に評価するために広く応用されている SRGM に基づく信頼性評価ツールを，OSS に基づくアプリケーションソフトウェアとして実装した．特に，開発されたアプリケーションソフトウェアの実装手法には，統計言語 R の ggplot2 および plotly ライブラリが利用されており，これらのライブラリと NW.js を組み合わせることで，統計言語 R を利用した研究用資産を有効利用できるだけでなく，研究成果を迅速に実社会におけるソフトウェア開発現場へ提供できるものと考える．

本論文で実装されたアプリケーションソフトウェアには，NW.js，統計言語 R，ggplot2，および plotly という JavaScript ライブラリが利用されている．これらは，OSS として開発および公開されており，無

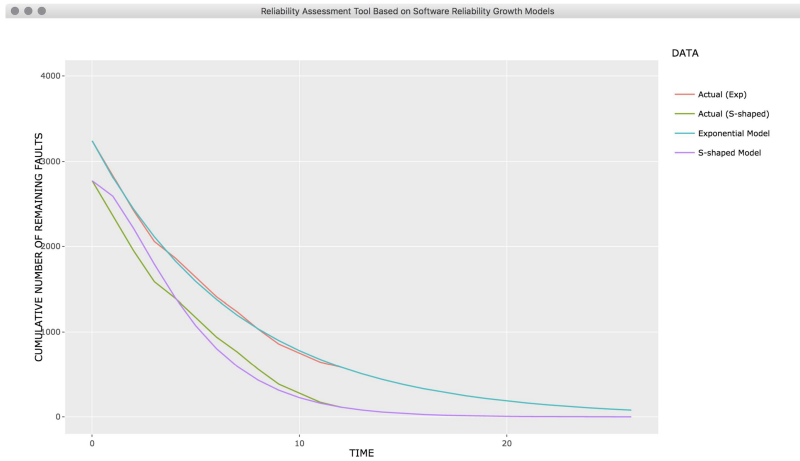

図 6： 推定された残存フォールト数.

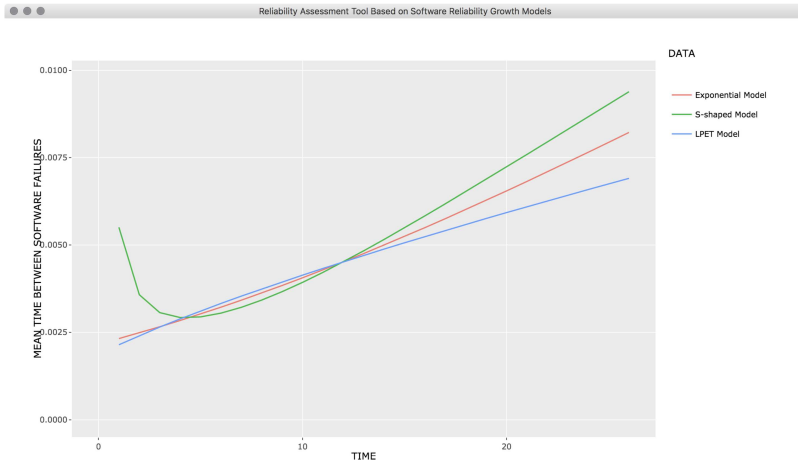

図 7： 推定された MTBF.

料で迅速に実装することが可能となる．特に，本実装方法により，研究による提案手法などの内容を詳細に理解していなくとも，誰でも利用可能な形式で頒布することができる．本論文における実装方法は，研究者にとっても，GUI のみの実装を行うだけで済むことから，研究成果を社会へ迅速に還元することが可能となる参考例として役立つものと考える．なお，開発されたアプリケーションソフトウェアは以下のサイトから利用できる．

http://tam.ims.tcu.ac.jp/

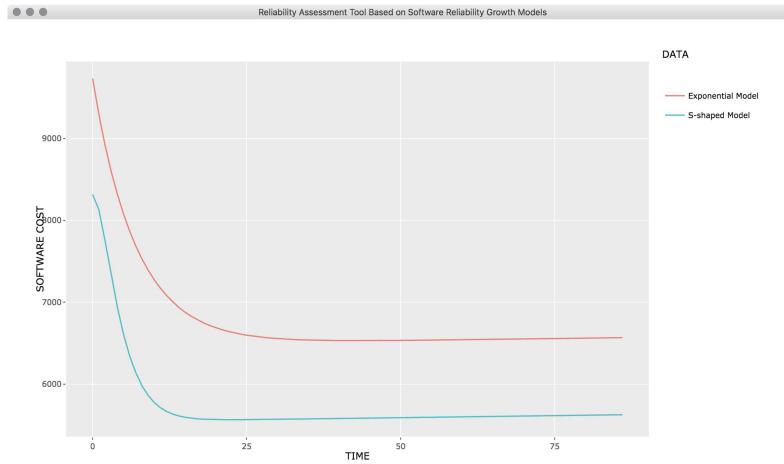

図 8： 推定された総期待ソフトウェアコスト.

謝辞

本研究の一部は，JSPS 科研費基盤研究（C）（課題番号 15K00102 および 16K01242）の援助を受けたことを付記する.

参考文献

[1] M.R. Lyu, ed., *Handbook of Software Reliability Engineering*, IEEE Computer Society Press, Los Alamitos, CA, 1996.

[2] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*, Springer–Verlag, Tokyo/Heidelberg, 2014.

[3] P.K. Kapur, H. Pham, A. Gupta, and P.C. Jha, *Software Reliability Assessment with OR Applications*, Springer–Verlag, London, 2011.

[4] S. Yamada and Y. Tamura, *OSS Reliability Measurement and Assessment*, Springer–Verlag, London, 2016.

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

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

[7] W.D. Brooks and R.W. Motley, Analysis of Discrete Software Reliability Models, Technical Report RADC–TR–80–84, Rome Air Development Center, Berlin, 1980.