オープンソースプロジェクトにおける修正時間推移予測および実用可能性の検討

山口大学大学院・創成科学研究科 曽根 寛喜（Hironobu Sone）†
†Graduate School of Sciences and Technology for Innovation, Yamaguchi University
山口大学大学院・創成科学研究科 田村 慶信（Yoshinobu Tamura）††
††0Graduate School of Sciences and Technology for Innovation, Yamaguchi University
鳥取大学・名誉教授 山田 茂（Shigeru Yamada）†††
†††Emeritus Professor, Tottori University

1 はじめに

ソフトウェア開発を進める中でフォールトは発生する．フォールトが発見されないと運用時に個人や企業に大きな損失を与える可能性がある．ソフトウェア保守の目的は，性能の向上だけでなく，不具合の修正や機能拡張を行い，ソフトウェアの品質を向上させることにある．

近年，オープンソースソフトウェア（OSS）の利用が盛んになってきている．OSS は，誰もがアクセスできるように設計されたコードである．特定の企業ではなく，オープンソースのコミュニティによって開発されるため，プロプライエタリなソフトウェアよりも安価で，柔軟性があり，長寿命であることが多い．しかし，オープンソースコミュニティには誰でも参加できるため，プロジェクトメンバーのスキルや開発環境，活動期間などの違いにより，プロジェクトの進捗に高い不確実性が存在する．大規模なオープンソースプロジェクトでは，1 日あたり 100 件以上の不具合が報告されることがあり [1]，それらの不具合を全て迅速に修正することが困難である [2,3]．そのため，オープンソースプロジェクトの進捗を予測することは難しく，多くのユーザーや企業にとって，オープンソースプロジェクトの開発状況や運用状況を把握することは，OSS のバージョンアップや導入の判断材料として重要である．

本研究では，OSS を開発する中で数多く発生するフォールトの修正時間の特徴を予測・把握することでプロジェクトの状態を評価できるかを検討する．ステップとしては以下のようになる．

1. フォールト重要度別に修正時間の特徴を確認
2. どれくらいのデータを学習させることが可能か確認
3. ソフトウェア信頼度成長モデル（SRGM）に適用し，修正時間推移の予測結果を確認

また，本研究において使用するデータは Red Hat Enterprise Linux (RHEL) [4] のバージョン 8.0 である．RHEL は，Red Hat 社が開発している商用オープンソースの Linux ディストリビューションであり，本研究で使用しているデータは Bugzilla [5] からフォールト情報を入手することができる．今回使用したフォールトは 5068 個である．

2 使用データの確認

今回使用するオープンソースプロジェクトデータから修正時間推移予測をするにあたり，フォールトデータの特徴および学習可能なデータ量について検討する．これはステップ 1 および 2 に該当する．

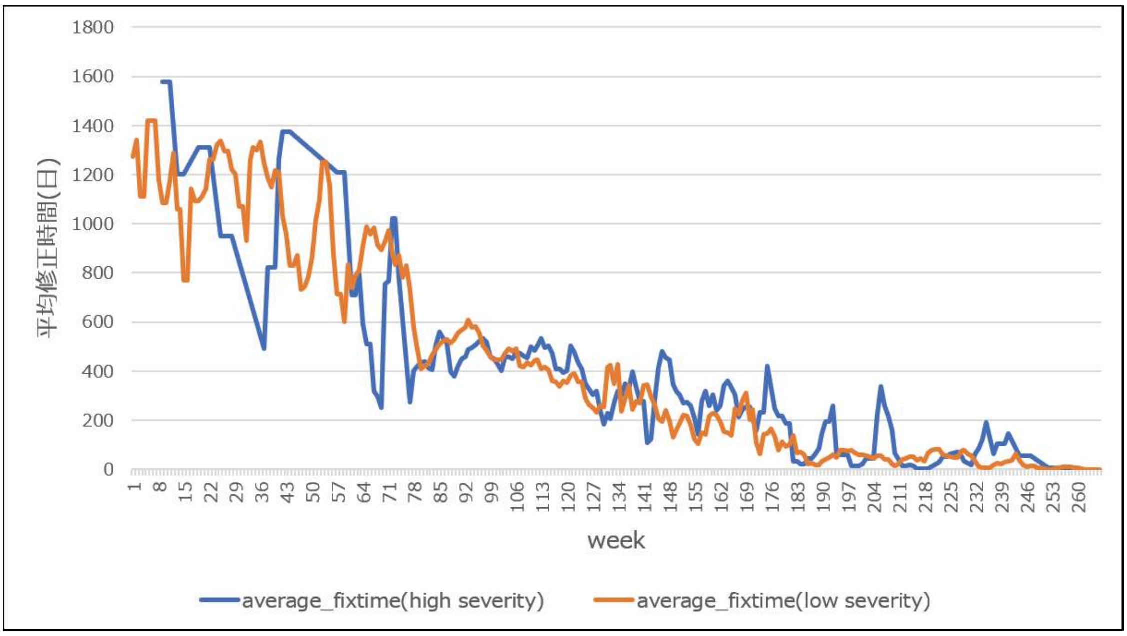

図 1： RHEL8.0 プロジェクトにおけるフォールト修正時間の 4 週間移動平均..

最初に，フォールト重要度別に修正時間の特徴を確認する．報告されるフォールトが重要度によって修正時間が異なる場合，重要度を層別したうえで修正時間推移を予測する．図 1 は，RHEL8.0 プロジェクトにおいて発生した 1 個当たりのフォールト平均修正時間を 4 週間の移動平均で取ったものである．重要度が高いフォールト（high severity）は最初の数週間においてフォールト報告がないため欠損値としているが，フォールト重要度によって修正時間推移に大きな違いはないため，重要度による層別は行わない．

次に， どれくらいのデータを学習させることが可能か確認する．直近に報告されたフォールト報告データは，まだ CLOSE されていないものや CLOSE と REOPEN を繰り返しているものがあるため，完全に CLOSE したフォールトデータを学習データとして取り扱いたい．図 2 は該当の週のフォールトは今回使用しているデータ取得日から何日前に CLOSE となったかを計算し，プロットしたものである．図より，概ね 100 週前後まではばらつきも大きいが，修正時間が長いフォールトが多く存在しており．REOPEN を繰り返している/修正に多くの工数を要したフォールトであると考えられる．一方で，200 週前後以降は修正時間が短く，ステータスが CLOSE となったとしても今後 REOPEN となるフォールトが複数存在する可能性がある．そのため，本研究では直近のフォールトは REOPEN となる可能性を考慮し，修正が完了し，安定してきている部分 (190 週) までを学習データとして予測する．

3 修正時間推移予測モデル

稼働時間 $t\,(t \geq 0)$ までのオープンソースプロジェクトへの投入開発時間を $\Omega\,(t)$ とし，$\Omega\,(t)$ は連続的な実数値をとる確率変数と仮定する．また，$\Omega\,(t)$ の時間変化率 $d\Omega\,(t)\,/dt$ は，現バージョンの OSS に必要とされる残存開発時間の影響を受けながら変化していくと考えるものとする．また，オープンソースプロジェクト特有の開発に関わる開発者やフォールト報告者の変化によるプロジェクトの進捗の不安定さを標準化された Gauss 型白色雑音によって近似的に表現する．よって，稼働時間 t における投入開発時間の変化率 $\beta\,(t)$ に，不規則性でモデル化する [6].

$$\frac{d\Omega\,(t)}{dt} = \{\beta\,(t) + \sigma\nu\,(t)\}\{\alpha - \Omega\,(t)\}, \tag{1}$$

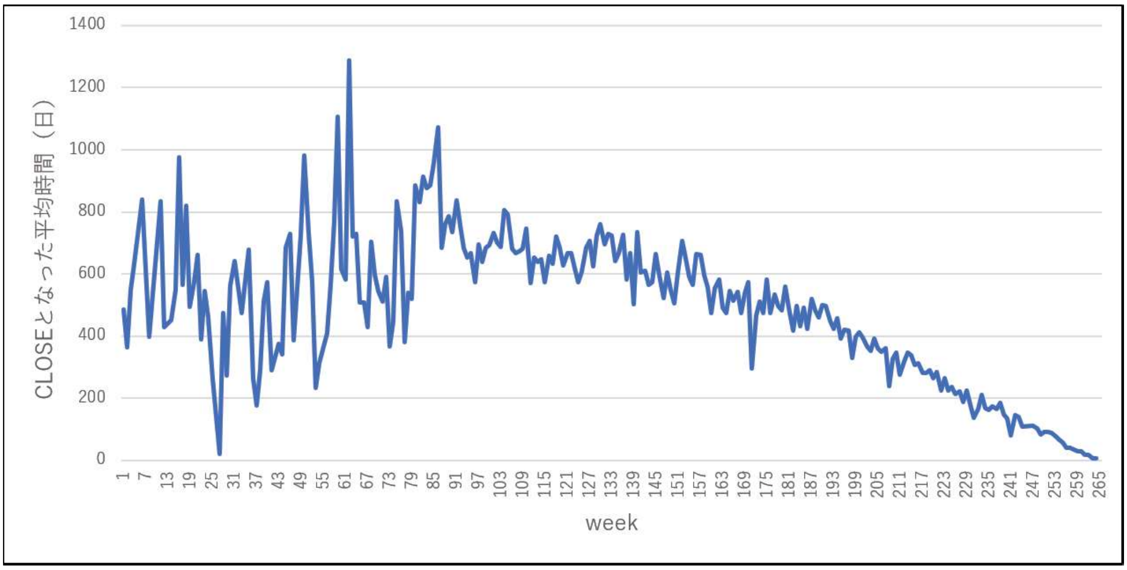

図 2： 7 × (データ取得週-week)-フォールト修正時間 (日)　の出力結果.

となる．ここで，α は OSS の特定バージョンの開発に必要とされる総開発時間，σ は非負数の定数パラメータであり，$\nu(t)$ は解過程の Markov 性を保証するための標準化された Gauss 型白色雑音である．$\Omega(0) = 0$ の条件のもと，式 (1) を解くために Itô の公式 [7] を用いると，

$$\Omega(t) = \alpha[1 - \exp\{-\int_0^t \beta(s)ds - \sigma\omega(t)\}], \tag{2}$$

ここで，$\omega(t)$ は白色雑音 $\nu(t)$ の時刻 t に関する積分として定義された Wiener 過程である．さらに，投入開発時間の増加率 $\beta(t)$ は以下のように定義される [8].

$$\int_0^t \beta(s)ds \doteq \frac{\frac{dF_*(t)}{dt}}{\alpha - F_*(t)}. \tag{3}$$

本研究では，累積開発時間を表す関数として，以下のソフトウェア信頼度成長モデル $F_*(t)$ を仮定する．

$$F_e(t) \equiv \alpha(1 - e^{-\beta t}), \tag{4}$$

$$F_s(t) \equiv \alpha\left\{1 - (1 + \beta t)e^{-\beta t}\right\}, \tag{5}$$

$$F_i(t) \equiv \frac{\alpha\{1 - \exp(-\beta t)\}}{1 + c \cdot \exp(-\beta t)}, \tag{6}$$

ここで，$\Omega_e(t)$ は $F_e(t)$ を用いた指数形ソフトウェア信頼度成長モデルを表し，$\Omega_s(t)$ は $F_s(t)$ を用いた遅延 S 字形ソフトウェア信頼度成長モデル，$\Omega_i(t)$ は $F_i(t)$ を用いた習熟 S 字形ソフトウェア信頼度成長モデルを表す．これら 3 種類のモデル式は，ソフトウェア信頼性評価において代表的なモデル式である [9].

よって，時刻 t までの累積保守時間は以下のように示せる．

$$\Omega_e(t) = \alpha[1 - \exp\{-\beta t - \sigma\omega(t)\}], \tag{7}$$

$$\Omega_s(t) = \alpha[1 - (1 + \beta t)\exp\{-\beta t - \sigma\omega(t)\}], \tag{8}$$

$$\Omega_i(t) = \alpha\left[1 - \frac{1 + c}{1 + c \cdot \exp(-\beta t)}\exp\{-\beta t - \sigma\omega(t)\}\right]. \tag{9}$$

これらのモデルにおいてパラメータ σ は，OSS 開発プロジェクトにおいていくつかの外部的要因によるノイズに依存していると仮定している．そのため，時刻 t までに要する累積保守時間の期待値は以下の

表 1： パラメータ推定結果.

		指数形モデル	遅延 S 字形モデル	習熟 S 字形モデル
parameter	α	1.102×10^5	1.044×10^5	1.037×10^5
	β	1.184×10^{-2}	2.494×10^{-2}	2.351×10^{-2}
	c	-	-	3.283
	σ	4.826×10^{-3}	6.394×10^{-3}	6.420×10^{-3}
AIC		2515.564	2514.653	2512.884

ように示せる.

$$\mathrm{E}[\Omega_e(t)] = \alpha\left[1-\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right], \tag{10}$$

$$\mathrm{E}[\Omega_s(t)] = \alpha\left[1-(1+\beta t)\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right], \tag{11}$$

$$\mathrm{E}\left[\Omega_i(t)\right] = \alpha\left[1-\frac{1+c}{1+c\cdot\exp(-\beta t)}\exp\left\{-\beta t+\frac{\sigma^2}{2}t\right\}\right]. \tag{12}$$

本研究では，修正時間推移を確認したいため，累積修正時間 $\Omega_*(t)$ を微分する.

$$\frac{d\Omega_e(t)}{dt} = \alpha\{\beta(t)+\sigma\nu(t)\}\exp\{-\beta t-\sigma\omega(t)\}, \tag{13}$$

$$\frac{d\Omega_s(t)}{dt} = \alpha(1+\beta t)\{\frac{\beta^2 t}{1+\beta t}+\sigma\nu(t)\}\exp\{-\beta t-\sigma\omega(t)\}, \tag{14}$$

$$\frac{d\Omega_i(t)}{dt} = \alpha\frac{(1+c)\left[\{-\beta t-\sigma\omega(t)\}\{1+c\exp(\beta t)\}+\beta c\exp(-\beta t)\right]}{\{1+c\exp(-\beta t)\}^2}\exp\{-\beta t-\sigma\omega(t)\}. \tag{15}$$

4　分析結果

本研究で使用する RHEL のフォールトデータを修正時間推移のとして使用する式 (13)-(15) に適用した際のパラメータを表 4 に示す．パラメータ推定には最尤推定法を使用しており，AIC の観点からどのモデルにおいても大きな差はないことが分かる．そのため，指数形モデルによる修正時間推移の予測結果を図 3 に示す．指数形モデルにおけるノイズは実際よりも大きく表示されたが，実際の修正時間推移におけるノイズの時間経過による変化と同様に次第にノイズの大きさが減少していることが分かる.

5　おわりに

本研究では，オープンソースプロジェクトにおけるフォールトの重要度を考慮したうえでフォールト修正時間の推移をソフトウェア信頼度成長モデルに適用し，予測した．フォールトの重要度によって修正時間の推移に大きな変化はないことから層別をしない状態で予測を行ったが，ノイズの大きさの推移については概ね実際のデータと同じ傾向を示すことが分かった．また，フォールトデータを入手した段階において直近に報告されたフォールトデータを除外することで REOPEN になる可能性のフォールトを減らすことにより，予測されたフォールト修正時間が REOPEN も考慮した最終的なフォールト修正時間と考えることができる．これにより，今後のオープンソースプロジェクトにおける安定性を検討する一つの材料になると考える.

今後は，他のオープンソースプロジェクトにも適用することで，今回の検討内容の検証や適切なモデル式について検討する必要がある.

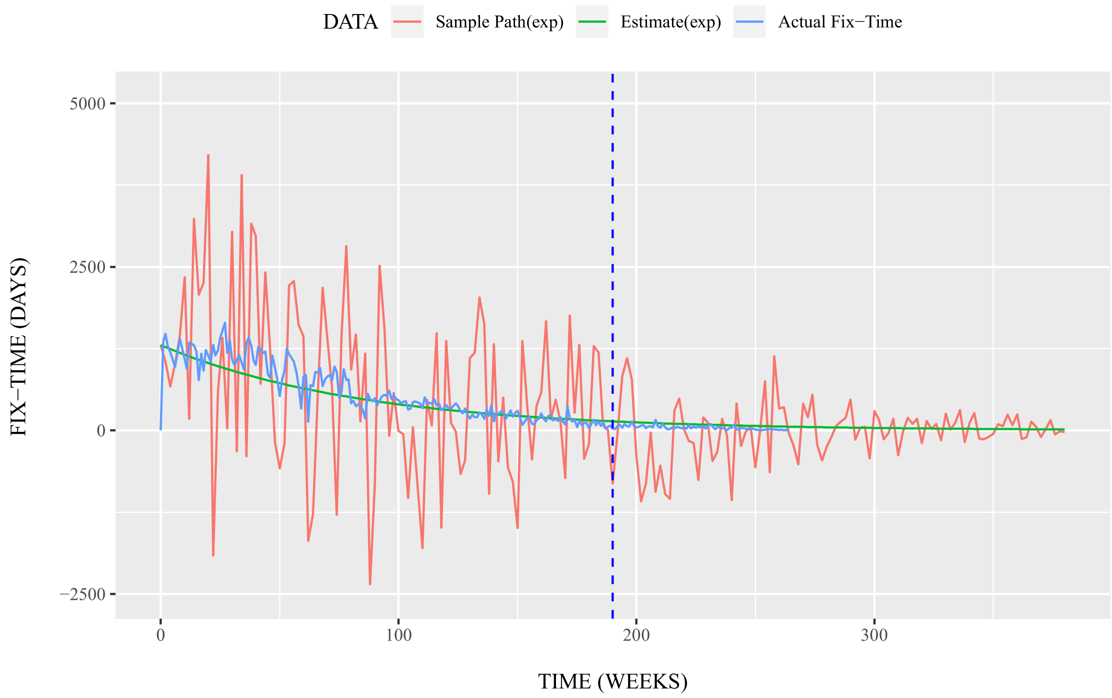

図 3： 指数形モデルにおける修正時間推移予測結果.

謝辞

本研究の一部は，JSPS 科研費基盤研究（C）（課題番号 20K11799）の援助を受けたことを付記する．

参考文献

[1] C. Sun, D. Lo, X. Wang, J. Jiang, D. Khoo, A discriminative model approach for accurate duplicate bug report retrieval. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE' 10)*, Cape Town. South Africa. 2-8 May 2010, pp.45-54.

[2] P. Hooimeijer, W. Weimer, Modeling bug report quality. *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering(ASE '07)*, Georgia. USA. 5-9 November, 2007, pp. 34-43.

[3] M. Nurolahzade, M. S. Nasehi, H. S. Khandkar, S. Rawal, The role of patch review in software evolution: an analysis of the mozilla firefox. *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops(IWPSE-Evol '09)*, Amsterdam. The Netherlands. 24-25 August 2009, pp.9-18.

[4] Red Hat, Red Hat Enterprise Linux, https://www.redhat.com/

[5] Bugzilla, The software solution designed to drive software development, https://www.bugzilla.org/

[6] E. Wong, *Stochastic Processes in Information and Systems*. McGraw-Hill, New York, 1971.

[7] L. Arnold,: *Stochastic Differential Equations-Theory and Applications*. John Wiley & Sons, New York, 1971.

[8] S. Yamada, M. Kimura, H. Tanaka, S. Osaki, Software reliability measurement and assessment with stochastic differential equations, IEICE Transactions on Fundamentals, vol. E77-A, no. 1, 109-116, 1994.

[9] K. Ranjan, K. Subhash, K. T. Sanjay, A study of software reliability on big data open source software, International Journal of System Assurance Engineering and Management, vol. 10, no. 2, pp. 242-250, 2019.