B-009

オープンソースプロジェクトにおける修正優先度を考慮したフォールト識別手法
Fault Identification Method Considering High Priority in Open Source Project

曽根 寛喜† 田村 慶信† 山田 茂‡
Hironobu Sone Yoshinobu Tamura Shigeru Yamada

1. はじめに

オープンソースソフトウェア（Open Source Software，以下 OSS と略す）とは，だれでも自由に利用，改変，再配布することができるソフトウェアである．近年，OSS は短納期，コスト削減，標準化といった観点から組込みシステムやサーバ用途として広く採用され，急速に普及している．特に，OSS はバザール方式と呼ばれる特有の開発形態をとっている．フォールトが発見されると世界中の開発者によって修正され，次のリリースに反映されるため，利用者や開発者が増えることにより，フォールトの発見・修正，リリースのサイクルを短くすることができる．特に，ソフトウェア開発ではフォールトの修正に関わる様々な情報を残し管理するためにバグ管理システム（Bug Tracking System，以下 BTS と略す）を用いた開発が主流となっている．

本研究では，BTS から得られる情報を用いることにより，新たに発生したフォールトが直近に修正されたフォールトとの相対評価により優先的に修正されるべきかを判断するための指標を提案する．また，実際に BTS のフォールトデータを用いることで数値例を示し，本提案手法の実用可能性について考察する．

2. 関連研究

近年盛んに OSS が開発・使用されるようになった一方で，大規模なオープンソースプロジェクトにおいてはフォールトの報告件数の増加により，フォールトの早期修正が困難になりつつあることや，修正されないフォールトが存在しているという問題がある[1][2]．このようなオープンソースプロジェクトにおける課題を解決するべく，OSS 開発者やオープンソースプロジェクトマネージャを支援するための研究は様々な観点から行われている[3][4]．例えば，BTS から得られるデータを用いることで報告されたフォールトの優先度や重要度を予測する研究[3] や，フォールトの修正時間を予測する研究[4]などがある．しかしながら，既存の手法によるフォールトの修正時間や優先度・重要度の予測結果が他のフォールトと同様の評価となった場合，フォールト修正担当者を決定する人物（Assignor）は，フォールトの修正優先順位を決めることが難しい．特に大規模プロジェクトでは報告されるフォールト数が多く，Assignor の負担が大きくなることから，容易にフォールトの修正状況を評価し，修正優先順位の把握ができるようになることが求められる．一方で，直近に修正されたフォールトの修正情報に基づき，新たに発生したフォールトを相対的に比較することにより評価することで，オープンソースプロジェクトの進捗状況を反映させた予測ができ，フォールトの修正優先順位も付けやすくなる．

本研究では，特定バージョンのオープンソースプロジェクトで発生したフォールトについて，直近に修正された

表 1　優先度および重要度のスコア対応表

優先度($prio$)	重要度($serv$)	スコア
P1	Blocker	5
P2	Critical	4
P3	Major	3
－	Normal	2.5
P4	Minor, Enhancement	2
P5	Trivial	1

フォールトと相対的に比較して優先度が高く，かつ修正時間がかかるフォールトを検出するための評価指標を提案する．さらに，実際の BTS 上のデータを用いて作成した指標をもとに予測することで本提案手法の有効性を検証する．

3. 提案指標

相対評価するフォールトの修正時間，優先度，および重要度を以下に示す．

$$T_i = t_{standard_i} \times (p_1 \times prio_i + p_2 \times serv_i). \quad (1)$$

ここで，$t_{standard_i}$ はi 番目のフォールト報告データの修正時間を正規化した値，$prio_i$はi 番目のフォールトの優先度，$serv_i$ はi 番目のフォールトの重要度，$p_1(0 \le p_1 \le 1)$および$p_2(0 \le p_2 \le 1)$は重みパラメータである．式(1)から，報告されたフォールトの修正時間に対し，T_i の値が大きくなるものはフォールトの修正時間，優先度，および重要度において問題があると判断することができる．特に，優先度と重要度が取りうる値に応じて，発生したフォールトの深刻さが大きく反映される．フォールトの優先度および重要度は，BTS に登録された情報を使用する．本研究では，取りうる値を[1,5]とした．表 1 に，優先度および重要度の対応表を示す．表 1 において，優先度の P1 は報告されたフォールトにおいて，主に開発者の観点から最も優先して修正すべきであることを示す．また，重要度の Blocker は報告されたフォールトにおいて，主にユーザの観点から最も優先して修正すべきであるとされる．

また，i 個のフォールト報告データの修正時間を正規化した値である式(1)の $t_{standard_i}$ は，次のように与えられる．

$$t_{standard_i} = \frac{t_i - min(t)}{max(t) - min(t)}. \quad (2)$$

ここで，t_i はi番目のフォールト修正時間である．修正時間の正規化により，$t_{standard_i}$ の取りうる値は[0,1]となる．

また，BTS に登録される優先度の高さは，フォールトの修正時間に影響を与えないという研究報告[2][5]がある．本研究では，優先度および重要度が修正時間に与える影響を考慮し，式(1)における重みパラメータp_1およびp_2の関係性を$p_1 < p_2$ と仮定し，$(p_1, p_2) = (0.4, 0.8)$とする．

† 東京都市大学　Tokyo City University
‡ 鳥取大学　Tottori University

表 3　予測結果

		OpenStack			Eclipse		
		再現率	適合率	F1 値	再現率	適合率	F1 値
閾値	μ	0.512	0.484	0.498	0.339	0.334	0.336
	$\mu+\sigma$	0.051	0.229	0.083	0.083	0.186	0.114
	$\mu+2\sigma$	0.004	0.143	0.008	0.065	0.081	0.072

4. 提案指標に基づくラベル付け

3.で示した評価指標T_i に基づき，学習データおよびテストデータに対して，以下の 1)~3) のように *High* および *Low* に対してラベリングを行った．

1) if $\mu \geq T_i$ then $High$ else Low
2) if $\mu+\sigma \geq T_i$ then $High$ else Low
3) if $\mu+2\sigma \geq T_i$ then $High$ else Low

ここで，μ はi 個あるフォールトデータにおける平均値，σ は標準偏差を表す．$High$と識別された場合は，修正時間やフォールトの優先度・重要度の観点から相対的に優先して修正すべきフォールトである． 2)および 3) の場合に *High* と識別された場合の方が，1)より優先して修正すべきと判断することができる．

5. 予測手法

本研究では，決定木を用いて集団学習を行う手法であるランダムフォレストを用いてテストデータにおける *High* もしくは *Low* を予測する．200 個のフォールトデータを学習し，その次に報告された 1 個のフォールトデータの *High* もしくは *Low* を予測することをフォールトの報告順に繰り返す．そのため，最初に報告された 200 個のフォールト以降をすべて予測することができ，フォールトデータがN 個の場合，$(N-200)$ 個の予測結果が出力される．

また，評価方法として，再現率，適合率，F1 値を使用し，10 回繰り返し予測した結果を平均した．本研究における再現率は，実際に *High* と識別されたフォールトのうち，予測モデルによって *High* と正しく予測することができたフォールトの割合を示す．また，適合率は予測モデルによって *High* と予測されたフォールトのうち，実際に *High* と識別されたフォールト数の割合を表す．さらに，F1 値は再現率と適合率の調和平均を示す．

6. 使用データ

本研究では，提案指標を評価するために，大規模なオープンソースプロジェクトの一つである，OpenStack[6]のバージョン 17(Queens) と Eclipse[7]のバージョン 4.7(Oxygen) を使用する．これらはオープンソースの BTS である Bugzilla から入手しており，OpenStack は 2284 個，Eclipse は 1800 個のフォールトデータを使用している．ただし，使用したデータは修正されたフォールトに限定している．

また，予測する際に使用した 11 種類の説明変数を表 2 に示す．これらは BTS から得られた情報である．特に，Component, OS, Reporter, Hardware, および Assignee は，200 個の学習データ内における出現率とする．

7. 予測結果

OpenStack と Eclipse のフォールト報告データに対して，ランダムフォレストを用いてフォールトの修正優先度を予測した結果を表 3 に示す．ここで，表 3 における閾値は，

表 2　説明変数について

説明変数	尺度	詳細
Component	比例	フォールトが発生したコンポーネントの出現頻度
Summary	比例	フォールトの詳細情報のワード数
OS	比例	フォールトが発生した OS の出現頻度
Weekday	名義	フォールト報告された日(平日，休日)
Reporter	比例	フォールト報告者の出現頻度
Type	名義	フォールト報告か機能追加要望か
Opened Interval	比例	前フォールト報告からの経過日数
Opened Month	間隔	フォールト報告された月
Hardware	比例	フォールトが発生したハードウェアの出現頻度
Cumulate Opened	比例	最初のフォールトからの累積日数
Assignee	比例	フォールト修正者の出現頻度

4.における *High* と *Low* の閾値を示す．

表 1 から，閾値の条件に応じて予測精度が向上することが分かった．また，BTS から入手可能なデータのみを説明変数とした場合，十分な予測精度を得ることは難しいことが分かった．特に，閾値が$\mu+\sigma$ や $\mu+2\sigma$ の場合の再現率と適合率ともに低い値であったため，実用化のためにはより予測精度の向上を図る必要がある．

8. おわりに

本研究では，オープンソースプロジェクトで発生したフォールトについて，直近に修正されたフォールトとの相対評価により優先的に修正されるべきかを判断するための指標を提案した．さらに，実データを用いて実用可能性について検討した．今後は，閾値の条件を厳しくした場合においても予測精度を向上させる必要がある．

参考文献

[1] P. Hooimeijer and W. Weimer, “Modeling bug report quality,” *Proceedings of the 22nd International Conference on Automated Software Engineering*, 2007, pp. 34-43.
[2] M. Nurolahzade, et al., “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 and Software Evolution Workshops*, 2009, pp. 9-18.
[3] H. M. Tran, et al., “An analysis of software bug reports using random forest,” *Proceedings of the 5th International Conference on Future Data and Security Engineering*, 2018, pp. 273-285.
[4] E. Giger, M. Pinzger, and C. H. Gall, “Predicting the fix time of bugs,” *Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52-56.
[5] D. M. Syer, et al., “Studying the fix-time for bugs in large open source projects,” *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, 2011.
[6] The OpenStack Foundation, The OpenStack project, http://www.openstack.org/
[7] The Eclipse Foundation, The Eclipse Project, http://www.eclipse.org/