不具合修正における優先度と修正作業の関係理解のための分析

吉行　勇人[†1]　大平　雅雄[†1]

大規模な OSS プロジェクトには日々多くの不具合が報告されている．全ての不具合を次のリリースまでに修正することは困難であるため，プロジェクトの管理者は報告された不具合に優先度を設定する．高い優先度が設定された不具合は次のリリースまでに修正完了することが期待されているが，高い優先度が設定された不具合のうち約 20 %の不具合は次のリリース以降に修正完了している．そこで本研究では，優先度と修正作業が一致していない不具合に着目して分析を行う．

An Analysis for Understanding the Relationship between Priority of Bugs and Fixing Bugs in Bug Fixing Process

Hayato Yoshiyuki[†1] and Masao Ohira[†1]

A large number of bugs are reported to a large-scale open source software (OSS) project. Since fixing all the bugs before the next release is very difficult, a manager in the project needs to decide which bugs should be fixed preferentially. In this paper, bugs in two open source software projects are analyzed and classified according to the priority and whether they were fixed by the next release or not.

1. はじめに

近年，オープンソースソフトウェア (OSS) が広く利用されるようになり，大規模な OSS プロジェクトには日々多くの不具合が報告されている [1]．このような現状では，全ての不具合を即座に修正することは困難であるため，OSS プロジェクトでは報告された不具合に優先度を設定する．優先度は主に次のバージョンリリースを目安に設定され，次のリリースまでに修正を行いたい不具合や緊急性の高い不具合には高い優先度を，次のリリースに間に合わなくても構わない不具合や影響の小さい不具合には低い優先度を設定する．しかし，適切な優先度を設定するには報告された不具合を個別に精査する必要があるため，多大なコストがかかる．このような背景から，報告された不具合の情報を用いて優先度決定を自動化するための研究が行われている [3]．

しかし，不具合に設定された優先度は必ずしも不具合修正時間に影響を与える訳ではないことが知られている [2]．実際に OSS プロジェクトを対象とした予備調査の結果，高い優先度が設定された不具合のうち 20%～30%の不具合はリリース後に修正完了しており，設定された優先度と実際の修正作業が一致していない不具合が存在することが示唆された．

そこで，本研究では設定された優先度と実際の修正作業が一致していない不具合に注目して分析を行い，優先度決定の自動化の精度向上のための知見を得ることを目的とする．

2. 不具合の分類

設定された優先度と実際の修正作業が一致していない不具合を分析するために，本研究では不具合の「優先度の高低」と「次のリリースまでに修正完了したか否か」によって 4 種類の不具合に分類する．

まず,「優先度の高低」による分類について説明する．本研究の対象として用いる不具合管理システム JIRA では，Blocker, Critical, Major, Minor, Trivial の 5 段階の優先度が設定されている．不具合報告時の初期値は Major であり，記録されている不具合のうち約 80%の不具合は初期値のままとなっている．そのため，優先度が Major 以外である不具合は何らかの理由によりプロジェクトの管理者により優先度を設定，変更された不具合であると示唆される．従って，Major を除いた 4 種類の優先度のうち，Blocker, Critical を「優先度高」，Minor, Trivial を「優先度低」として不具合を分類する．

†1 和歌山大学
Wakayama University

表 1　分析結果の比較

Derby	件数		修正時間 (日)		変更ファイル数		変更行数	
	優先度高	優先度低	優先度高	優先度低	優先度高	優先度低	優先度高	優先度低
リリース前	68	649	10.08	5.21	6	6	44	24
リリース後	25	364	129.81(58.05)	284.32(194.37)	8	8	103	61
Qpid	件数		修正時間 (日)		変更ファイル数		変更行数	
	優先度高	優先度低	優先度高	優先度低	優先度高	優先度低	優先度高	優先度低
リリース前	152	488	5.66	1.80	6	4	22	22
リリース後	37	132	201.99(119.98)	165.61(97.26)	6	6	64	38

次に,「次のリリースまでに修正完了したか否か」による分類について説明する．不具合報告には報告された日付と，既に修正が完了していれば修正完了日が記録されている．OSS プロジェクトに記録されている過去のリリースの日付を参照して，各リリースの間に報告された不具合の修正完了日が次のリリース日よりも前か後かによって不具合を分類する．

3. ケーススタディ

本研究で行った不具合の分析について説明する．本分析では，Apache Derby プロジェクトおよび Apache Qpid プロジェクトの不具合管理システム JIRA に報告された不具合を分析の対象とする．対象不具合は，Apache Derby は 2004 年 9 月から 2014 年 10 月，Apache Qpid は 2006 年 9 月から 2014 年 10 月に報告された不具合のうち，既に修正が完了している不具合かつ修正のためにソースコードを変更している不具合とする．対象不具合の件数は Apache Derby は 1,106 件，Apache Qpid は 809 件とする．

分析内容として，前述の不具合の分類ごとに，不具合の件数，修正にかかった時間，修正時に変更したファイル数，修正時に変更したソースコードの行数をそれぞれ算出し，比較を行う．

4. 結果と考察

表 1 に分析結果を示す．件数以外の結果は各不具合の中央値で表している．修正時間のリリース後の値の括弧中はリリース日を超えてからどれだけの日数がかかったかを表している．

表 1 より，修正時間をみると，リリース後に完了した不具合はリリース前に完了した不具合よりも修正に時間がかかっていることが分かった．また，リリース日以降にかかった日数も長くなっているため，リリースに間に合わなかった不具合は急いで修正する必要が無くなり，修正が長期化する傾向にあると考えられる．

修正時に変更したソースコードの行数をみると，優先度高の不具合は優先度低の不具合よりも変更行数が多く，またリリース後に完了した不具合はリリース前に完了した不具合よりも変更行数が多いことが分かった．そのため，リリース前に修正完了するには，修正コストを正しく見積もり，修正担当開発者等を決める必要があると考えられる．

5. おわりに

本研究では，不具合修正プロセスにおける不具合の優先度と実際の修正作業の関係を理解するための分析を行った．分析の結果，リリース前に修正完了する不具合とリリース後に修正完了する不具合の特徴に違いがあることが分かった．

今後の課題は，不具合に関わる人に関する分析や不具合報告に含まれる情報を用いた分類の細分化を行い，優先度の自動決定や不具合が次のリリースまでに修正完了するか否か（不具合修正時間）の予測精度の向上を目指すことである．

ワークショップでは，時間の考慮について議論したい．例えばリリース間隔や次のリリースまでにどれだけ時間が残っているかによって優先度やリリース前に修正完了するか否かに違いがあると考えている．

謝辞　本研究の一部は，文部科学省科学研究補助金（基盤 (C): 24500041）による助成を受けた．また，独立行政法人情報処理推進機構が実施した「2013 年度ソフトウェア工学分野の先導的研究支援事業」の支援を受けた．

参考文献

1) Jeong, G., Kim, S. and Zimmermann, T.: Improving bug triage with bug tossing graphs, *Proceedings of ESEC/FSE '09*, pp. 111–120 (2009).
2) Syer, M. D., Adams, B., Zou, Y. and Hassan, A. E.: Studying the fix-time for bugs in large open source projects, *Proceedings of the PROMISE'11* (2011).
3) Tian, Y., Lo, D. and Sun, C.: DRONE: Predicting Priority of Reported Bugs by Multi-factor Analysis, *Proceedings of the ICSM '13*, pp.200–209 (2013).