不具合管理パターンが不具合修正に与える影響の分析

Analysis of the Impact of Bug Management Patterns on Bug Fixing

大平 雅雄[*] **大澤 直哉**[†] **アハマド ハッサン**[‡] **松本 健一**[§]

Summary. An efficient bug management process is critical for the success of software projects. Prior work has focused on improving this process, for example, by automating bug triaging, detecting duplicate bugs, and understanding the rationale for re-opening bugs. This paper continues this line of work by exploring the people who are involved in the bug management process. In particular we develop four patterns that distill the different relation between the people involved in the process: the reporter, triager, and fixer of a bug. Through a case study on the Eclipse Platform and JDT projects, we demonstrate that these patterns have an impact on the efficiency of the bug management process. For example, we find that using our patterns project personnel can improve their efficiency through better communication about bugs before assigning them.

1 はじめに

OSS (Open Source Software) プロジェクトの成功には，効率的な不具合管理プロセスが必要である．多くの OSS プロジェクトでは，不具合管理を効率良く行うために Bugzilla などのバグトラッキングシステム (BTS) を利用している．しかしながら，Eclipse や Mozilla プロジェクトなどの大規模な OSS プロジェクトは，現状の不具合管理プロセスの限界に直面している．近年の OSS 利用者の増加により，日々大量に報告される不具合のすべてに対処することが困難となりつつあるためである．

不具合管理を行う開発者（以降，**管理者**と呼ぶ）は，不具合報告の内容理解，重複する不具合報告の確認，適任の開発者の選別など，多岐に渡る作業を経て不具合修正タスクを他の開発者（以降，**修正担当者**と呼ぶ）に割り当てる．そのため，エンドユーザを含む多数の**報告者**から不具合が大量に報告される現状では，これら一連の作業が管理者にとって極めて複雑かつ困難なものとなっている．実際，Eclipse プロジェクトに報告される不具合の内，44%の不具合は，1 度の不具合修正作業では解決しないという状況が発生している [1]．管理者が不具合報告を正しく理解し，かつ，適任の開発者を探してタスクを割り当てることが困難なためである．

これまで，不具合管理プロセスの改善を目的として，不具合報告の記述方法の分析 [2] [3]，重複する不具合報告の検出 [4] [5]，タスクの再割当や再修正される不具合の原因分析 [1] [6] など，様々なアプローチで研究が行われてきた．本研究も不具合管理プロセスの改善を目指している点で先行研究と目的を共有している．特に本論文では，不具合管理におけるプロジェクト参加者（報告者，管理者，修正担当者）の関係を 4 つのパターン（不具合管理パターン）に分類し，不具合管理パターンが不具合修正の効率に与える影響を，不具合修正タスクの割当にかかる時間（**修正タスク割当時間**）と不具合の修正自体にかかる時間（**不具合修正時間**）の両面から分析する．

[*]Masao Ohira, 奈良先端科学技術大学院大学

[†]Naoya Osawa, 奈良先端科学技術大学院大学

[‡]Ahmed Hassan, Queen's University

[§]Ken-ichi Matsumoto, 奈良先端科学技術大学院大学

2　OSS 開発における不具合管理プロセスと不具合管理パターン

2.1　不具合管理プロセスに関与する参加者の役割

図 1 に，BTS を用いた一般的な不具合管理プロセスを示す．BTS を用いた OSS 開発では少なくとも，**報告者**，**管理者**，**修正担当者**という 3 種類の参加者の役割が存在する．**報告者**は，OSS プロダクトを利用／開発しているユーザ／開発者である．**管理者**は，不具合修正タスクを他の開発者に割り当てる権限を持ったプロジェクトの主要開発者である．**修正担当者**は，不具合修正タスクを管理者に割り当てられる開発者を指す．

不具合管理プロセスにおいて管理者は，重要な役割を担っている．不具合の早期解決のためには，報告された不具合の内容を素早く理解し，その不具合の修正に適任の担当者に修正タスクを割り当てることが必要になるためである．ここで，管理者は，報告者あるいは修正担当者を兼任することができる．この場合，不具合管理プロセスの効率は，別々の人物によってそれぞれの役割が担当される場合と比べ相当異なるものと予想される．そこで本研究では，不具合管理プロセスに関与する参加者の役割とその関係を 4 種類のパターン（**不具合管理パターン**）としてまとめ，それらが不具合管理プロセスに与える影響を明らかにすることを目的とする．

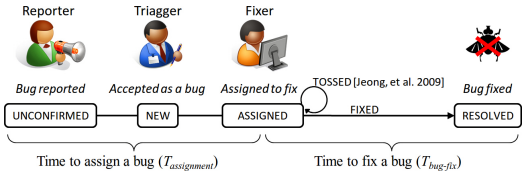

図 1　OSS 開発における不具合管理プロセス

3　不具合管理パターン

本章では，不具合管理プロセスに関与する報告者，管理者，修正担当者というプロジェクト参加者の関係を**不具合管理パターン**としてまとめ，不具合管理プロセスにおける「チーム」の特徴を整理する．図 2 に 4 種類の不具合管理パターンを示す．

パターン 1 ：報告者 = 管理者 = 修正担当者 [R=T=F]
パターン 1 では，すべての役割を 1 人の参加者（開発者）が担当する．不具合の原因を熟知し，かつ，不具合解決に自信がある開発者であると考えられるため，$T_{assignment}$ と T_{fix} は共に最も短いものになると予想される．

パターン 2 ：報告者 = 管理者 ≠ 修正担当者 [R=T≠F]
パターン 2 では，1 人の参加者 (A) が報告者と管理者を兼任し，別の参加者 (B) が修正担当者となる．(A) は不具合修正タスクを割り当てる権限を持った開発者であると想定できるため，$T_{assignment}$ は短くなるものと予想される．一方，(A) が不具合修正タスクを適任の (B) に割り当てることがどうかは保証されないため，T_{fix} は比較的長くなるものと予想される．

図 2　不具合管理プロセスにおける 4 つのパターン

パターン 3 ：報告者 ≠ 管理者 = 修正担当者 [R≠T=F]
パターン 3 では，1 人の参加者 (A) が不具合の報告者となり，別の参加者 (B) が管理者と修正担当者を兼任する．パターン 2 とは逆に，$T_{assignment}$ は比較的長くなり，T_{fix} は比較的短くなるものと予想される．

パターン 4 ：報告者 ≠ 管理者 ≠ 修正担当者 [R≠T≠F]
パターン 4 では，各役割を異なる参加者が担当する．報告者–管理者，管理者–修正担当者のそれぞれで不具合修正に関係する知識とスキルのミスマッチが生じやすいと考えられるため，$T_{assignment}$ と T_{fix} は共に長くなるものと予想される．パターン 4 は，OSS プロジェクトにおける協調作業の成否に深く関係している．現実には，OSS プロダクトはごく少数の参加者による多大なる貢献によって成り立っていることが知られており [7]，パターン 4 の効率を改善することが OSS 開発の今後の発展の鍵となると考えられる．

4 不具合管理パターンに関するケーススタディ

本章では，不具合管理パターンを用いて Eclipse platform と JDT を対象に行ったケーススタディについて述べる．

4.1 Data sets

表 1 に，本ケーススタディで用いたデータセットの概要を示す．収集した不具合報告データの内，約半数が 1 回で修正が完了する不具合であることが見て取れる．また，報告者は，管理者・修正担当者よりも約 10〜15 倍多いことが分かる．次に，1 回で修正が完了する不具合がどの不具合管理パターンによって解決されたかを図 3 に示す．Eclipse の両プロジェクトでは，パターン [R≠T=F] と [R≠T≠F] が主要な不具合管理パターンであることが見て取れる．2 つのパターンはほぼ同等の割合であり，不具合解決のために管理者に相当の負担がかかっていることが分かる．

表 1 Platform と JDT から収集したデータセットの概要

プロジェクト	分析期間	不具合報告数	修正不具合数（再修正有）	修正不具合数（再修正無）	報告者（人）	管理者（人）	修正者（人）
Platform	2007 年 1 月〜2009 年 12 月	21,308	8,434	4,133	811	54	85
JDT	2007 年 1 月〜2009 年 12 月	8,110	3,343	1,657	369	23	33

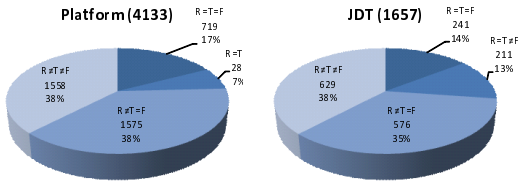

図 3 Eclipse プロジェクトにおける不具合管理パターンの割合

4.2 不具合管理パターンと修正タスク割当時間の関係

RQ1: 不具合管理パターンは修正タスク割当時間にどの程度影響を与えるか？

動機: $T_{assignment}$ は，報告者と管理者を兼任する参加者が存在するかどうかに依存していると考えられる．4 つの不具合管理パターンと $T_{assignment}$ の関係を調べることで，報告者と管理者を兼任する参加者の $T_{assignment}$ に対する影響をより明確に調べられると考えた．

表 2　Eclipse プロジェクトにおける不具合管理パターンと $T_{assignment}$ の分析結果

プロジェクト	不具合管理パターン	中央値（日）	標準偏差（日）	最大値（日）	最小値（日）	P 値		
						R=T≠F	R≠T=F	R≠T≠F
Platform	R=T=F	0.00	71.50	812.05	0.00	$< 0.01 **$	$< 0.01 **$	$< 0.01 **$
	R=T≠F	0.00	53.06	512.98	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	0.48	82.35	842.93	0.00	-	-	0.61
	R≠T≠F	0.53	51.00	842.93	0.00	-	-	-
JDT	R=T=F	0.00	75.48	713.66	0.00	0.59	$< 0.01 **$	$< 0.01 **$
	R=T≠F	0.00	25.18	237.06	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	0.69	93.67	927.05	0.00	-	-	$< 0.01 **$
	R≠T≠F	0.32	63.79	638.23	0.00	-	-	-

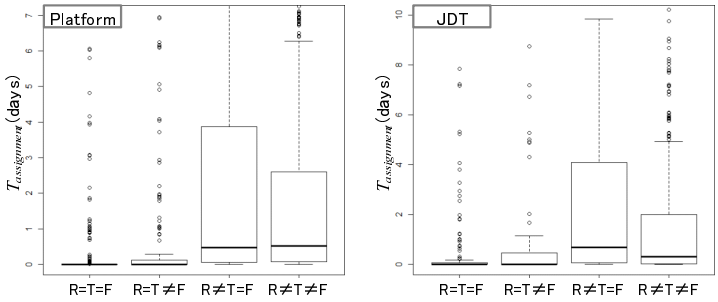

図 4　Eclipse プロジェクトにおける不具合管理パターンと $T_{assignment}$ の分析結果（箱髭図）

アプローチ:　$T_{assignment}$ の中央値（日）を求め，不具合管理パターンの間に $T_{assignment}$ に統計的有意差があるかどうかを検定した．検定には，Mann–Whitney の U 検定 ($\alpha = 0.05$) を用いた．平均値ではなく中央値を用いる理由は，外れ値（例えば，修正タスク割当に数百日かかるもの）が平均値を大きく歪めることが確認できたためである．

結果：表 2 と図 4 にそれぞれ，不具合管理パターンと $T_{assignment}$ との関係を分析した結果を示す．我々の予想通り，Platform では，[R=T=F] が最短の $T_{assignment}$ の結果を示している．JDT では，[R=T=F] と [R=T≠F] の $T_{assignment}$ には有意差がなく中央値も両者 0.00 日であることから，[R=T=F] はどちらのプロジェクトにおいても $T_{assignment}$ の観点では最適な構成であると言える．一方，我々の予想に反して，Platform プロジェクトでは，[R≠T=F] と [R≠T≠F] の $T_{assignment}$ に統計的に有意な差は認められなかった．さらに，JDT では，[R≠T≠F] の方が [R≠T=F] よりも有意に短い $T_{assignment}$ を示すことが分かった．[R≠T=F] という構成に対してこのような結果となった理由としては，管理者が適任の修正担当者を探すことができなかった，あるいは，修正担当候補者にタスクを依頼したものの断られるなどして最終的にタスクを管理者自身に割り当てた結果と考えるのが自然である．

4.3　不具合管理パターンと不具合修正時間の関係

RQ2: 不具合管理パターンは不具合修正時間にどの程度影響を与えるか？

動機: エンドユーザが報告者となった場合，管理者が不具合の内容を正しく理解することが困難なため，適切な修正担当者にタスクを割り当てることができず [2] [3]，T_{fix} が長引く可能性が高い．不具合修正パターンを用いて不具合管理プロセスを分析ことで，不具合修正の効率に関するより正確な理解が得られるものと期待した．

アプローチ: RQ1 と同様に，T_{fix} の中央値（日）を求め，不具合管理パターンの間に T_{fix} に統計的に有意な差があるかどうかを検定した．検定には，Mann–Whitney の U 検定 ($\alpha = 0.05$) を用いた．中央値を用いる理由は RQ1 と同様である．

結果：表 3 と図 5 にそれぞれ，不具合管理パターンと T_{fix} との関係を分析した結果を示す．両プロジェクトの [R≠T≠F] とその他のパターンの T_{fix} の間すべてに統計的有意差が認められる．このことは，参加者間のミスコミュニケーションやコ

表 3　Eclipse プロジェクトにおける不具合管理パターンと T_{fix} の分析結果

プロジェクト	不具合管理パターン	中央値（日）	標準偏差（日）	最大値（日）	最小値（日）	P 値		
						R=T≠F	R≠T=F	R≠T≠F
Platform	R=T=F	1.00	50.75	434.80	0.00	$< 0.01 **$	0.11	$< 0.01 **$
	R=T≠F	0.10	80.64	889.05	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	1.30	70.86	775.96	0.00	-	-	$< 0.01 **$
	R≠T≠F	7.13	115.19	988.21	0.00	-	-	-
JDT	R=T=F	0.64	36.92	377.84	0.00	0.19	0.32	$< 0.01 **$
	R=T≠F	0.83	32.98	281.03	0.00	-	0.49	$< 0.01 **$
	R≠T=F	0.79	45.23	583.11	0.00	-	0.49	$< 0.01 **$
	R≠T≠F	2.11	69.71	705.90	0.00	-	-	-

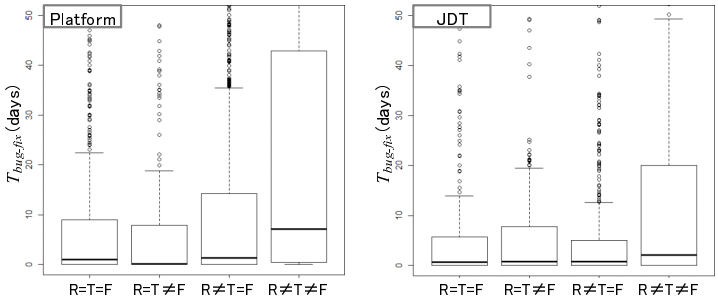

図 5　Eclipse プロジェクトにおける不具合管理パターンと T_{fix} の分析結果（箱髭図）

ミュニケーションオーバーヘッドにより，不具合管理プロセスにおける「チームワーク」は非常に困難であることを示唆している．

5　考察

5.1　結果のまとめ

プロジェクト参加者の役割と不具合修正プロセスに関して以下の知見を得た．

- 管理者が報告者を兼任する場合（[R=T=F], [R=T≠F]），他の場合（[R≠T=F], [R≠T≠F]）に比べ，修正タスク割当時間は約 7〜16 時間多く必要になる [RQ1].
- 異なる参加者が報告者･管理者･修正担当者をそれぞれ担当する場合（[R≠T≠F]），不具合修正時間は約 1〜7 日多く必要になる [RQ2] .

本ケーススタディの結果は，不具合管理プロセスにおける社会的側面と知識共有の側面の重要性を改めて強調するものであり，プロジェクト関係者の知識共有やコミュニケーション支援の必要性を示唆している．また，研究者が不具合解決時間等の予測モデルを構築する際に，参加者の役割と関係を考慮することでより良い予測精度をもたらす可能性があることを示している．

5.2　不具合管理プロセスへの参加者間の議論の効果

[R≠T≠F] は T_{fix} の観点では最も悪い構成であったが，図 5 の [R≠T≠F] には幅広い分布が認められる．[R≠T≠F] であっても，他のパターンよりも短い T_{fix} を実現する場合が存在することを示唆する．そこで我々は，Platform の [R≠T≠F] における 1,558 件の不具合報告から無作為に 100 件を抽出し，目視で内容を確認するとともに T_{fix} の長いものと短いものとに分けて比較した．その結果，修正タスク割当前の参加者間の議論が T_{fix} に大きな影響を与えることを見出した．

図 6 に，Platform での修正タスク割当前の議論（コメント数）と不具合修正時間との関係を示す．コメント数が増えるに従って T_{fix} が徐々に短くなっていることが見て取れる．コメント数の多い議論では，開発者らが不具合の解決方法や誰が修正担当者として最適かなどを議論していることも分かった．

このことから，プロジェクトチームとそのリーダーは，タスク割当前に十分なコ

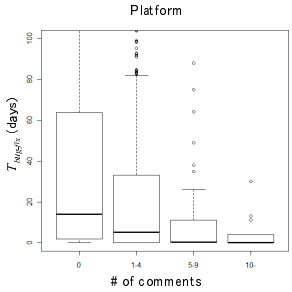

図 6　Platform の [R≠T≠F] における修正タスク割当前の議論（コメント）回数と T_{fix} の関係

ミュニケーションを図ることで不具合修正効率を改善できると言える．また，開発者間の議論は $T_{assignment}$ を長引かせる可能性を高めるものの，適任の担当者をより正し選ぶことができるため不具合の再修正を防止する効果が期待できる．

6　おわりに

本論文では，不具合管理パターンが不具合修正の効率に与える影響の分析を行った．その結果，不具合管理パターンが修正タスク割当時間と不具合修正時間に大きな影響を与えることが分かった．特に，3 種類の役割を別々の参加者が担当する場合，他のパターンよりも時間を要することが示された．ただし，タスク割当前に参加者間で議論を行うことで，不具合修正時間を短縮できることも分かった．

本論文では，修正担当者の再割当を必要とした不具合は分析の対象としなかった．再割当は様々な理由で必要となることが知られており [8]，不具合修正パターンとの関連を調べることは本研究の今後の課題である．また，得られた知見の妥当性を向上させるために，他のプロジェクトでも分析を行う必要がある．

謝辞　本研究の一部は、文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また、本研究の一部は、文部科学省科学研究補助費（基盤 B：課題番号 23300009，若手 B：課題番号 22700033）による助成を受けた．

参考文献

[1] G. Jeong, S. Kim, and T. Zimmermann, “Improving bug triage with bug tossing graphs,” ESEC/FSE’09, pp.111–120, 2009.
[2] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, “What makes a good bug report?,” SIGSOFT’08/FSE-16, pp.308–318, 2008.
[3] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, “Information needs in bug reports: improving cooperation between developers and users,” CSCW’10, pp.301–310, 2010.
[4] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, “A discriminative model approach for accurate duplicate bug report retrieval,” ICSE’10, pp.45–54, 2010.
[5] P. Runeson, M. Alexandersson, and O. Nyholm, “Detection of duplicate defect reports using natural language processing,” ICSE’07, pp.499–510, 2007.
[6] E. Shihab, A. Ihara, Y. Kamei, W.M. Ibrahim, M. Ohira, B. Adams, A.E. Hassan, and K. Matsumoto, “Predicting re-opened bugs: A case study on the eclipse project,” WCRE’10, pp.249–258, 2010.
[7] A. Mockus, R.T. Fielding, and J.D. Herbsleb, “Two case studies of open source software development: Apache and mozilla,” ACM Transactions on Software Engineering and Methodology, vol.11, no.3, pp.309–346, 2002.
[8] P.J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, ““not my bug!” and other reasons for software bug report reassignments,” CSCW’11, pp.395–404, 2011.