社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－SE－161 (1)
2008／9／26

「非障害案件」の発見を目的としたバグ票の観察

石田 響子† 伏田 享平† 名倉 正剛† 川口 真司† 飯田 元†

† 奈良先端科学技術大学院大学 情報科学研究科
〒630-0192 奈良県生駒市高山町 8916-5
E-mail: †{ kyoko-i , kyohei-f, nag, kawaguti} @is.naist.jp, iida@itc.naist.jp

要旨

ソフトウェア開発では，ソフトウェアの個々の動作やその結果が定められた仕様に基づき，実装が行われる．しかしソフトウェアが利用される場面において，利用者が期待する動作と，開発者によって定義された仕様が異なる場合がある．そのために，ソフトウェアのある動作が，仕様通りの動作であるにもかかわらず，ユーザが不具合と認識することがある．本研究ではこのような案件を「非障害案件」と定義し，この発見を目的として，ユーザと開発者のやりとりを観察した．そして，オープンソースソフトウェア開発プロジェクトにおけるバグ管理システムでのやりとりを分析したところ，非障害案件と開発者の不具合への対応に関係があることが分かった．

Observations of Bug Reports for Detecting “Non Defective Issues”

KYOKO ISHIDA† KYOHEI FUSHIDA† MASATAKA NAGURA†
SHINJI KAWAGUCHI† and HAJIMU IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan
E-mail: †{ kyoko-i , kyohei-f, nag, kawaguti} @is.naist.jp, iida@itc.naist.jp

Abstract

On software developments, developers implement products according to the specifications about operational procedures and execution results. However, in using software, results supposed by users sometimes differs from the specifications defined by developers. In cases like this, the users will consider those results as bugs. In this paper, we define those cases as "Non Defective Issues". In order to detect non defective issues, we observed the communications between developers and users by analyzing bug tracking system. The results of this observation showed that non defective issues are related with developers' handlings to bug reports.

1 はじめに

ソフトウェア開発では，要求定義から導いた仕様に基づいて，設計や実装が行われる．しかし開発者が仕様通りに実装を行っていても，ソフトウェアが利用される場面では，ソフトウェアを実際に使用する者（以降「ユーザ」と記述する）にとって，その仕様が直感的でないことがある．このようなとき，ソフトウェアのある動作が，開発者にとっては仕様通りの動作であるにもかかわらず，ユーザに不具合と認識されうる．これは，仕様に記述されている事柄の認識や，仕様に明確な記述がない部分の認識が，ユーザと開発者で異なる場合に生じる．このような場面が多くみられるソフトウェアは，ユーザに対して使いにくい印象を与えてしまい，利用を敬遠される可能性がある．本研究では，ユーザに不具合と認識されたが，開発者にとっては不具合ではないと判断された案件を「非障害案件」と定義した．この中には，ユーザビリティの欠如に起因するものが存在すると考えられる．このため非障害案件を発見することで，ユーザビリティを向上することができる．

非障害案件は，ユーザと開発者の間のやりとりを分析することによって，発見可能である．本研究ではバグ管理システムに着目し，バグ票の報告におけるユーザと開発者のやりとりを分析した．まず開発者が個々のバグ票にどのように対応したかを分類することによって，開発者の対応と非障害案件の発生には何らかの関係があるという仮説を立てた．そして，その仮説を検証するために，4つのオープンソースソフトウェア開発プロジェクトを対象に，バグ管理システムのデータを分析した．

2 バグ管理システム

GNATS[*]やBugzilla[†]に代表されるバグ管理システムは，ソフトウェア開発プロジェクトにおいて，プロダクトの不具合報告や機能の追加要求を管理するために用いられる．オープンソースソフトウェア開発プロジェクトにおける多くのバグ管理システムでは，バグ報告はデータベースに記録され，Web ユーザインタフェースを介して自由に閲覧できるようになっている．

バグを報告する際には，多くの場合，発生した不具合やその再現手順を自然言語で記述する．システムによっては，対象モジュールを選択肢の中から指定する場合や，ファイルの添付が可能な場合もある．また，システム利用者はコメントを随時，自然言語で記述することができる．

オープンソースソフトウェア開発プロジェクトにおける，典型的な不具合修正のプロセスを以下に示す．

手順1. バグ票オープン：ユーザがプロダクトの不具合を発見し，バグ票を起票する．

手順2. 開発者割当て：バグ管理を受け持つ開発者が，起票されたバグ票に対して，不具合に対応する開発者の割当てを行う．

手順3. 不具合修正：割当てられた開発者が不具合を修正する．

手順4. バグ票クローズ:不具合の修正を確認し，対応完了としてマークする．

このようなバグ票のオープン，クローズなどの状態遷移や，割当てられた開発者の情報は，個々のバグ票に記録される．従って，バグ票を観察することにより，開発者の不具合への対応や，ユーザとのやりとりの状況を把握することができる．

3 非障害案件

3.1 非障害案件の定義

ソフトウェアが利用される場面において，ユーザが期待する動作と，開発者によって定義された仕様が異なる場合がある．このような場合，ユーザは，ソフトウェアに不具合があると判断することがある．たとえば，開発者が仕様として定めた事柄を，ユーザが誤って認識した場合は，ユーザはその部分の動作に不具合があると判断するだろう．同様に，仕様として明記していない部分を開発者自身の判断に基づいて実装した部分がユーザの直感的な基準と異なる場合も，ユーザはその部分の動作を不具合と判断するかもしれない．しかし開発者の観点からすると，これらの場合すべてにおいて，ソフトウェアは仕様通りに動作しており，開発者は不具合はないと主張するだろう．

本研究では，このような，ユーザにとっては不具合と認識されたが，開発者にとっては不具合でないと判断された案件を「**非障害案件**」と定義する．非障害案件の中には，ユーザビリティの欠落といった，ユーザに不利益をもたらすものが含まれると考えられる．非障害案件の例と，ユーザへの影響を次節で述べる．

3.2 非障害案件の例と影響

オープンソースソフトウェア FreeMind[‡]はマインドマップ[§]を記述するツールである．図1左のマインドマップに対して，FreeMind を用いてノードを編集する際には，図1右側のダイアログが表示される．ここでマインドマップのダイアログに着目すると，背景の色が白に固定されており，マインドマップのノードの背景と異なっている．この案件は FreeMind のユーザによって発見され，プロジェクトのバグ管理システムにバグ票が起票された．しかし開発者により「このダイアログの背景色は，ノードの背景色が反映されない仕様

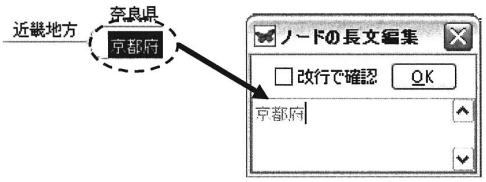

図 1: マインドマップのノードの編集（FreeMind）

[*] http://www.gnu.org/software/gnats/

[†] http://www.bugzilla.org/

[‡] http://freemind.sourceforge.net/wiki/

[§] Tony Buzan が開発したダイアグラム記述法．放射状にキーワードや図を配置する．主にブレーンストーミングや発想の整理に利用される．

である」と述べられ，バグ票はクローズされていた．

マインドマップのノードの文字色が白であれば，ダイアログでは白い背景に白い文字が表示されるため，編集が困難になる．このような非障害案件を発見することにより，見過ごされてしまった問題を見出すことができる．

3.3 非障害案件とバグ管理システム

非障害案件を発見するために，本研究ではバグ管理システムを解析して，開発者の不具合への対応やユーザとのやりとりの観察を行った．

3.1 節で述べたように，非障害案件はユーザにとっては不具合であると認識される．これはバグ管理システムにおいては，バグ票の起票（2 章で述べたプロセスにおける手順 1）が行われることから認識できる．

また非障害案件は，開発者にとって不具合ではないと判断される．バグ管理システムにおいては，バグ票に対する開発者の割当て（2 章で述べたプロセスにおける手順 2）が行われないままバグ票がクローズされる（2 章で述べたプロセスにおける手順 4）と考えられる．3.2 節で述べた FreeMind プロジェクトの非障害案件の例においても，ユーザが報告した不具合に対して，開発者によって修正を行わないと判断され，特定の開発者が修正のために割当てられないままバグ票がクローズされている．

本研究では，このような開発者が割当てられずにクローズされたバグ票には，非障害案件が含まれると仮定した．すなわち，ユーザが不具合と考えバグ票を起票したものの，開発者が修正不要と考え，修正のための開発者の割当てや対応を行わないままクローズされてしまった案件が含まれると考えた（**仮説 1**）．また開発者が割当てられずにクローズされたバグ票が多くなるほど，非障害案件が増加すると推測した（**仮説 2**）．次章では，この仮説を確認するために行った観察について報告する．

4 バグ票の観察

3.3 節で示した 2 つの仮説を確認するために，バグ管理システムにおいて，バグ票がクローズされた際に開発者が割当てられていないバグ票を対象に観察を行った．観察にあたり，クローズされたバグ票のうち，クローズの際に開発者が割当てられていないバグ票の割合を「**未割当てクローズ率**」と定義した．そして，ソフトウェア開発プロジェクトのバグ管理システムを分析し，未割当クローズ率を 1 ヶ月間ごとに算出した．次にバグ票がクローズされた動機を分類することにより，バグ票にどの程度の非障害案件が含まれるかを観察し，未割当てクローズ率と非障害案件の関係を調査した．

4.1 観察対象

SourceForge.net[‡]を利用してソフトウェア開発を行うプロジェクトを対象に，バグ管理システムに登録されたバグ票を観察した．SourceForge.net は，オープンソースソフトウェア開発に関するリポジトリを提供するサービスである．2008 年 4 月時点における，下記の 4 つのプロジェクトのバグ票を観察した．

- FreeMind
  FreeMind は知識の整理などに利用されるマインドマップを記述するソフトウェアである．階層構造の情報に特化した GUI を持つエディタである．多様なプラットフォーム・言語に対応しており，幅広い多くのユーザ層に利用されている．
- jEdit[§]
  jEdit は Java で書かれたプログラマ向けのテキストエディタであり，プラグインの利用によって様々なプログラミング言語に対応している．
  jEdit プロジェクトにおいては 3 つのバグ管理システムが運用されており，それぞれ jEdit のコア部分，ランチャー，プラグインにおける不具合を対象としている．非障害案件はどの部分においても発生し得ると考え，すべてのバグ管理システムのバグ票を対象とした．
- XOOPS Dynamic Web CMS[**]
  XOOPS Dynamic Web CMS（XOOPS と略す）はコンテンツ管理システムである．多くの言語に翻訳が行われており，世界中に数多くのユーザが存在する．
- phpMyAdmin[††]
  phpMyAdmin はデータベース MySQL をウェブ上で管理するツールである．多くの言語への翻訳も行われており，数多くのユーザに利用されている．

---

‡ http://sourceforge.net/
§ http://www.jedit.org/
** http://www.xoops.org/
†† http://www.phpmyadmin.net/

表 1: 観察対象プロジェクトの詳細

プロジェクト名	SourceForge.net に登録された年/月	開発者数	バグ票総数	バグ票がクローズされた回数		全観察期間における未割当てクローズ率
				未割当て	全て	
FreeMind	2000/08	8 人	641 件	262 回	343 回	76.4%
jEdit	1999/12	157 人	4235 件	2814 回	3704 回	76.0%
XOOPS	2001/12	136 人	1315 件	384 回	806 回	47.6%
phpMyAdmin	2001/03	8 人	3093 件	527 回	2761 回	19.1%

表 2: バグ票のクローズ動機の分類

グループ	クローズされた動機
修正済	開発者により修正が行われた.
非障害案件	開発者により不具合ではないと判断された.
ユーザ判断	ユーザ（起票を行った報告者）により不具合ではないと判断された.
重複	他のバグ票と内容が重複していた.
不適切	起票するバグ管理システムが不適切であった.
非再現	不具合を再現することができなかった.
ボット	長期間バグ票の更新が行われなかったため，バグ管理システムにより自動的にクローズされた.
不明	バグ票に修正が行われたかを示すコメントが記述されていない.
スパム	広告など不適切な起票であった.

4 つのプロジェクトの, 2008 年 4 月時点における開発者数やバグ票に関する詳細を表 1 に示す.

4.2 観察方法

観察にあたって，まず，各開発プロジェクトのバグ票のクローズ回数および未割当クローズ率を求めた. そして十分なデータを含まない期間を観察対象から除外することで, 着目する期間を導出した. 次に，それらの期間ごとに，バグ票がクローズされた動機を観察し，分類を行った. 以下に手順の詳細を示す.

4.2.1 未割当てクローズ率の算出と着目する期間の導出

まず，4 つのオープンソースソフトウェア開発プロジェクトのバグ管理システムにおいて，「バグ票クローズ回数」を月ごとに分割して計上した. そしてこの値が全観察期間の「バグ票クローズ件数」の平均より少ない月は，対象となるバグ票の数が少なく，観察に不向きであると判断し，観察の対象から除外した. 次に，残った期間の各月における「未割当てクローズ率」を計上した. その値を各プロジェクトの全観察期間における「未割当てクローズ率」と比較して，高い期間，低い期間，同程度の期間に分類し，それぞれの分類ごとに，ある 1 つの月を着目する期間として導出した.

4.2.2 バグ票のクローズ動機の分類

4.2.1 節で導出した各プロジェクトにおけるそれぞれの期間において，バグ票がクローズされた動機を表 2 に示す 9 つのグループに分類した.

なおバグ票によっては，複数のグループに分類されるバグ票も存在する. 例えば，起票された不具合を開発者が再現することができなかったため，報告者により詳細な情報を求めたが，ある一定の期間が過ぎてしまったためにバグ管理システムにより自動的にクローズされてしまう場合がある. このようなバグ票に対しては,「非再現」および「ボット」の両方のグループに重複して分類した.

4.3 観察結果

4.3.1 未割当てクローズ率の算出と着目する期間の導出

4.2.1 節の手順により，表 3 に示す各プロジェクトごとに，着目する期間を導出した.
FreeMind プロジェクトにおいては, 1 ヶ月間の未割当てクローズ率が, 全観察期間における未割当てクローズ率と比較して高い Fh，同程度の Fm，低い Fl を選出した. 他の 3 プロジェクトにおいても同様に選出した. ただし jEdit と phpMyAdmin プロジェクトに関しては, 全観察期間における未割当てクローズ率と同程度の未割当てクローズ率である期間が存在しなかったため, それらの期間を選出していない.

4.3.2 バグ票のクローズ動機の分類

4.2.2 節に示したように，着目したそれぞれの期間にクローズされた個々のバグ票について, クローズの動機を表 2 の 9 つのグループに分類した結果を表 4 に示す. 表 4 の各期間に対して，上段

表 3：各プロジェクトにおいて着目する期間

期間	年/月	クローズ回数		未割当てクローズ率
		未割当て	全て	
FreeMind				
Fh	2005/11	87 回	89 回	97.8%
Fm	2008/03	23 回	31 回	74.2%
Fl	2007/01	0 回	21 回	0.0%
jEdit				
Jh	2005/02	102 回	111 回	91.9%
Jl	2006/09	76 回	111 回	66.7%
XOOPS				
Xh	2007/03	35 回	38 回	92.1%
Xm	2006/05	39 回	80 回	48.8%
Xl	2006/10	7 回	17 回	41.2%
phpMyAdmin				
Ph	2002/01	24 回	48 回	50.0%
Pl	2003/09	6 回	44 回	13.6%

の値はバグ票のクローズが各グループに分類された回数を示し，下段の括弧内の値は分類された回数の全クローズ回数に対する割合を示す．

4.4 考察

表 4 より，Xh 以外の全ての期間において，クローズされたバグ票に非障害案件が含まれることが分かった．本章では，未割当てクローズ率と非障害案件の関係について分析する．まず，クローズされた動機を基準に，本分析の対象としないデータの除去を行った．次に，未割当てクローズ率と非障害案件の相関関係を求め，その妥当性について考察する．また非障害案件に関して，どのような対処を行うべきかについて述べる．

4.4.1 除去を行った期間

各期間の特徴から，本分析において対象としない期間を選出した．具体的にはバグ管理システムの開発者の割当て機能が使われていない期間や，

表 4：期間ごとのクローズ動機の分類

期間	未割当てクローズ率	分類された回数（全クローズ回数に対する割合）								
		修正済	非障害案件	ユーザ判断	重複	不適切	非再現	ボット	不明	スパム
FreeMind										
Fh	97.8%	49 回	13 回	1 回	15 回	1 回	9 回	1 回	2 回	0 回
		(55.1%)	(14.6%)	(1.1%)	(16.9%)	(1.1%)	(10.1%)	(1.1%)	(2.2%)	(0.0%)
Fm	74.2%	16 回	8 回	1 回	1 回	0 回	2 回	1 回	2 回	0 回
		(51.6%)	(25.8%)	(3.2%)	(3.2%)	(0.0%)	(6.5%)	(3.2%)	(6.5%)	(0.0%)
Fl	0.0%	5 回	2 回	0 回	0 回	0 回	2 回	19 回	12 回	0 回
		(23.8%)	(9.5%)	(0.0%)	(0.0%)	(0.0%)	(9.5%)	(90.5%)	(57.1%)	(0.0%)
jEdit										
Jh	91.9%	30 回	32 回	3 回	2 回	0 回	4 回	0 回	41 回	0 回
		(27.0%)	(28.8%)	(2.7%)	(1.8%)	(0.0%)	(3.6%)	(0.0%)	(36.9%)	(0.0%)
Jl	66.7%	48 回	28 回	2 回	5 回	0 回	15 回	11 回	14 回	0 回
		(43.2%)	(25.2%)	(1.8%)	(4.5%)	(0.0%)	(13.5%)	(9.9%)	(12.6%)	(0.0%)
XOOPS										
Xh	92.1%	0 回	0 回	0 回	0 回	0 回	1 回	1 回	0 回	37 回
		(0.0%)	(0.0%)	(0.0%)	(0.0%)	(0.0%)	(2.6%)	(2.6%)	(0.0%)	(97.4%)
Xm	48.8%	15 回	16 回	1 回	1 回	1 回	4 回	15 回	40 回	0 回
		(18.8%)	(20.0%)	(1.3%)	(1.3%)	(1.3%)	(5.0%)	(18.8%)	(50.0%)	(0.0%)
Xl	41.2%	11 回	1 回	0 回	1 回	0 回	3 回	4 回	0 回	0 回
		(64.7%)	(5.9%)	(0.0%)	(5.9%)	(0.0%)	(17.6%)	(23.5%)	(0.0%)	(0.0%)
phpMyAdmin										
Ph	50.0%	24 回	14 回	5 回	2 回	0 回	3 回	0 回	0 回	0 回
		(50.0%)	(29.2%)	(10.4%)	(4.2%)	(0.0%)	(6.3%)	(0.0%)	(0.0%)	(0.0%)
Pl	13.6%	34 回	1 回	1 回	1 回	0 回	2 回	0 回	5 回	0 回
		(77.3%)	(2.3%)	(2.3%)	(2.3%)	(0.0%)	(4.5%)	(0.0%)	(11.4%)	(0.0%)

開発者が意図していないクローズが多い期間を対象から除去する．

- **バグ管理システムにおいて開発者の割当て機能が使われていない期間（Fh）**

  FreeMind プロジェクトは，SourceForge.net へ登録が行われた 2000 年において，開発者は 1 人であった．ところが 2001 年以降に開発が途絶え，以後 2008 年 4 月現在まで開発に加わっていない．その後 2003 年頃から徐々に別の開発者が参加し，2005 年頃には開発者が 3 人となった．開発者が 1 人であった頃は，開発者の割当ての機能がほとんど使われていなかった．この習慣が Fh の期間にも残っていたため，開発者への割当てが行われなかったのではないかと考えられる．

- **広告がほとんどであった期間（Xh）**

  Xh の期間では，表 4 より「スパム」グループのバグ票の割合が非常に高かった．クローズが行われたバグ票のほとんどが，明らかに XOOPS の開発と関係のない広告であった．

- **「ボット」グループに分類された割合が 2 割以上であった期間（Fl，Xl）**

  Fl や Xl においては，保留とされていたバグ票が長期間更新されなかったために，バグ管理システムにより自動的にクローズが行われたものが多く見られた．保留に至った理由は，「不具合の修正が行われた」という開発者の報告の後にユーザから不具合がまだ残っているとコメントされたものや，開発者が不具合の再現を行うことができなかったものがほとんどであった．

以上により，まずバグ管理システムにおいて開発者を割当てる習慣がないことが，未割当てクローズ率に影響を与えることが分かった．また広告がほとんどであった Xh 以外の全ての期間においては，バグ票に非障害案件が確認された．これより，バグ管理システムにおいて開発者の割当てが行われているプロジェクトでは，広告に起因するバグ票を無視すれば，3.3 節の仮説 1 の傾向が成り立つことが確認された．一方，バグ管理システムにより自動的にクローズが行われた期間においては，開発者の意図に関わらず，未割当てのままクローズされるバグ票が多いことが分かった．このため，この場合には未割当てクローズ率が高い場合でも，非障害案件が多かったとは言い切れない．

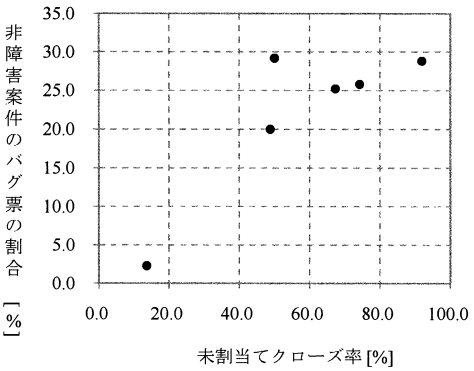

図 2: 未割当てクローズ率と
非障害案件に分類されたバグ票の割合の関係

4.4.2 未割当てクローズ率と非障害案件のバグ票の割合の関係

次に，未割当てクローズ率と非障害案件の数との間に相関関係があるかを検証する．なおこの際に，4.4.1 節に述べた Fh，Fl，Xl，Xh は除いた．

それぞれの期間における未割当てクローズ率と非障害案件のバグ票の割合を，図 2 に示す．これより，未割当てクローズ率と非障害案件の数の間には，正の相関関係があることが確認された．よって 3.3 節の仮説 2 にあるように，開発者が割当てられずにクローズされたバグ票が多くなるほど，非障害案件が増加する．なおこの相関係数は 0.848 であり，p 値は 0.069 であった．

4.4.3 妥当性の検証

4.4.2 で行った考察の妥当性を検証する．

- **サンプル数や対象プロジェクト数・性質**

  今回の観察では着目した期間の数が少なく，サンプル数が小さかった．また対象プロジェクトも 4 つのオープンソース開発プロジェクトであり，他のプロジェクトを加えると結果が異なる可能性がある．今回採用したソフトウェア開発プロジェクトは，GUI を用いて「編集」を行うもの（FreeMind および jEdit）や「設定」を行うもの（XOOPS や phpMyAdmin），また図形を扱うもの（FreeMind）やテキストデータを扱うもの（jEdit），対象としているユーザが異なるなど，プロジェクトの持つ性質は幅広い．しかしバグ報告対応の状況は，利用するバグ管理システムの違いをはじめとして，その他様々な要因に影響を受ける．このため，様々な見地から観察を行う必要がある．

- **プライベートマークが付与されたバグ票の影響**

  SourceForge.net のバグ管理システムには，個々のバグ票にプライベートマークを付与する機能がある．プライベートマークの付けられたバグ票は，起票を行った報告者またはプロジェクトに登録されている開発者だけしか閲覧することができない．よって今回の観察では，これらのバグ票を観察対象に含めることができなかった．

  ただし，プライベートマークの機能は，バグ票の不具合が重大な脆弱性を指摘する内容であった場合などに用いられることが多く，未対応のままクローズされるとは考えにくい．よってこのようなバグ票に非障害案件が含まれる可能性は低く，今回の分析に影響を与える可能性も低いと考えられる．

- **サンプルのばらつきや分類結果のぶれ**

  相関分析を行ったデータにおいて，未割当てクローズ率 20〜40%のサンプルが存在しなかった．この間の未割当てクローズ率の期間に対しての検証を試みる必要がある．なおバグ票のクローズ動機の分類を人手で行ったため，結果にぶれが生じている可能性がある．この点に関しても基準をより明確にするなどの検討が必要である．

4.4.4 今後の課題

- **「ボット」に分類されたバグ票や広告のバグ票が多かった期間の除去**

  未割当てクローズ率にもとづいて非障害案件の多寡を判別するためには，4.4.1 節で行った除去を半自動的に行う必要がある．

  「ボット」に分類されたバグ票が多い期間の除去については，バグ票のクローズ時の履歴から，半自動的に「ボット」グループへの分類を行うことが有効であると考えられる．今回観察対象とした SourceForge.net のバグ管理システムでは，ボットに該当するようなバグ票は，クローズを行ったユーザ名が特定の名前になるため，自動的に分類できる．

  同様に広告が多い期間については，迷惑メールフィルタリング技術を利用して，「スパム」グループへの分類をある程度は自動化できる．

- **非障害案件対処方法に関する検討**

  非障害案件に関して実際に FreeMind プロジェクトのバグ管理システムに起票されたバグ票を例に挙げ，どのように対処を行うべきかを考察する．

  3.2 節で例として用いた，マインドマップのノードの編集機能については，

  - ダイアログ内部では，常に文字色が黒，背景色が白になるようにする．
  - マインドマップの文字色と背景色の両方をダイアログに反映する．
  - マニュアルに注意書きを追加する．
  - 何も修正を行わない．

  など，さまざまな対処法が考えられる．このうちいずれの対処を採用するかについては，それぞれのソフトウェアが想定するユーザ像により異なる．よって非障害案件の対処に関しては，ユーザ像を開発者間で共有し，議論を行う必要がある．想定されるユーザのプロファイルを開発者間で共有することによってユーザビリティを向上する方法を「ペルソナ法」という [1]．また ISO13407 においても，「利用の状況の把握と明示」というプロセスの必要性が述べられている [2]．このような，ペルソナ法をはじめとするユーザビリティ向上策を，非障害案件対処時に適用する必要がある．

5 関連研究

バグ管理システムを対象として，様々な研究が行われている．本章では，まずバグ管理システムやそこに登録されたバグレポートを扱った研究を示す．また，オープンソースソフトウェア開発には，バグ管理システムが利用されていることが多い．そこで，オープンソースソフトウェア開発プロジェクトを対象とした研究についても紹介する．

Anvik らは，バグレポートの開発者への割当を自動的に行う手法を提案している [3]．Wang らは自然言語解析と実行時情報を利用した重複したバグレポートの検出手法を提案している [4]．Weiss らは修正完了の報告がなく，対応の完了していないバグレポートについて，修正工数を予測する手法を提案している [5]．Bettenburg らは Eclipse プロジェクトにおける，バグレポートの品質に関して調査を行っている [6]．Nichols らはオープンソースソフトウェアプロジェクトにおいて，ユーザビリティの問題に着目し，バグ管理システムにバグが登録されてからシステム改善が行われる過程を観察している [7]．このように，バグレポートを対象とした研究は多くなされているが，ソフトウェア利用者（ユーザ）の観点に

立って，バグレポートの報告内容に着目し観察した研究は，我々の知る限り存在しない．

バグ管理システムは，オープンソースソフトウェア開発においても利用されることが多い．そのため，オープンソースソフトウェアの開発プロジェクトを分析する際には，バグ管理システムや版管理システム，メーリングリストなどのソフトウェアリポジトリを対象とする場合が多い．Mockus らは，これまでオープンソースソフトウェア開発について指摘されていた事柄の正しさについて，CVS とバグ管理システムのデータを調査した[8]．Jensen らは，オープンソースソフトウェア開発者の役割の移り変わりなどについて，3 つの開発プロジェクトを比較し，その違いについて報告を行っている[9]．Matsumoto らはオープンソース開発において開発者とユーザとの橋渡しをおこなっているユーザの特定を，ネットワーク分析を用いて行っている[10]．これに対し，本研究では，ユーザの視点からバグ管理システムを独自の指標を用いて観察している．

6 おわりに

本研究では，ユーザにとっては不具合と認識されたが，開発者にとっては不具合でないと判断された案件を「非障害案件」と定義した．そして，この非障害案件を発見するために，バグ管理システムにおけるバグ票の観察を行った．観察の結果より，未割当てクローズ率と，非障害案件とされたクローズ回数には，正の相関関係が見られた．

この相関関係により，未割当てクローズ率を，非障害案件の発見に使うことができると考えられる．未割当てクローズ率の高い期間に着目することにより，多くの非障害案件を発見し，その対処を行うことでソフトウェアの品質向上を期待できる．そして非障害案件の対処については，それぞれの案件に適した対処を行う必要がある．

今後は，今回提案した検出結果に基づき，4.4.4 で述べたペルソナ法などのユーザビリティ向上策の適用について提案していきたい．また 4.4.3 で述べたクローズ動機の分類の自動化手法を提案する予定である．

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．

参考文献

1) Cooper, A.: The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity, Sams (1999).
2) International Organization for Standardization: ISO13407: 1999 Human-centered design processes for interactive systems, International Organization for Standardization (1999).
3) Anvik, J., Hiew, L., and Murphy, G.C.:Who Should Fix This Bug?, *Proc. 28th International Conference on Software Engineering (ICSE2006)*, pp.361-370 (2006).
4) Wang, X., Zhang, L., Xie, T., et al.:An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information, *Proc. 30th International Conference on Software Engineering (ICSE2008)*, pp.461-470 (2008).
5) Weiss, C., Premraj, R., Zimmerman, T., et al.: How Long will it Take to Fix This Bug?, *Proc. 4th International Workshop on Mining Software Repositories (MSR 2007)*, (2007).
6) Bettenburg, N., Just, S., Schroeter, A., et al.: Quality of Bug Reports in Eclipse, *Proc. 2007 OOPSLA Workshop on Eclipse Technology eXchange (eTX 2007)*, pp.21-25 (2007).
7) Nichols, D. M. and Twidale, M. B.: Usability Processes in Open Source Projects, *Softw. Process Improve. Pract.*, Vol.11, No.2, pp.149-162 (2006).
8) Mockus, A.,Fielding, R. T., and Herbsleb, J. D.: Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Trans. Softw. Eng. Methodol.*, Vol.11, No.3, pp.309-346 (2002).
9) Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proc. 29th International Conference on Software Engineering (ICSE2007)*, pp.364-374 (2007).
10) Matsumoto, S., Kamei, Y., Ohira, M., and Matsumoto, K.: A Comparison Study on the Coordination Between Developers and Users in Communities, Proc. Socio-Technical Congruence (STC 2008), (2008).