Vol.2014-SE-183 No.20
2014/3/20

クラッシュレポートが不具合修正に与える影響の分析

小須田 光[1,a)]　亀井 靖高[1,b)]　伊原 彰紀[2,c)]　鵜林 尚靖[1,d)]

概要：ソフトウェアの不具合を解消するため，クラッシュレポートの収集を行う開発プロジェクトが増加している．クラッシュレポートは，ソフトウェアのクラッシュ時に自動作成されるレポートで，クラッシュ以前の実行環境に関する情報をまとめたものである．本研究の目標は，不具合修正のための情報をより多く集めることである．そのためにユーザがクラッシュレポートを送信することで得られるメリットを示し，クラッシュレポートの送信を促す．本稿では，不具合票とクラッシュレポートのデータとの間でなされている関連付けと，それらが不具合修正に与える影響に関して分析を行った．Firefox を対象に行ったケーススタディの結果，クラッシュレポートが関連付けられている不具合のうち，不具合票の登録よりも先にクラッシュレポートが送信されているものは，不具合票の登録よりも後にクラッシュレポートが送られてきたものに比べて，修正期間が中央値で約 80 日短縮されることがわかった．

キーワード：
クラッシュレポート，不具合修正, 実証的研究

1. はじめに

ソフトウェアの利用者の環境でソフトウェア欠陥による不具合が発生した場合，利用者に少なくはない時間的／金銭的損害を与えるだけではなく，開発プロジェクトの評判も落とすことになる [1]．そのため，全ての欠陥がリリース前の段階で発見され取り除かれていることが，利用者と開発者双方にとって望ましい．しかしながら，限られた開発工数と定められた納期の中では，開発者が全ての不具合を事前に発見することは難しい．

そこで，一部の開発プロジェクトでは，クラッシュレポートシステムを導入することで，不具合の発生状況を迅速に認識し，不具合を修正する仕組みを構築している [2]．クラッシュレポートシステムは，ユーザ環境下においてソフトウェアが予期しない停止をした際にスタックトレース [*1] やユーザ環境等，不具合の再現や修正に有用な情報をレポートにまとめ，開発プロジェクトへ送信するシステムである．

しかしながら，クラッシュレポートの送信には基本的にユーザの許可が必要なため，ユーザによって拒否され不具合解消に必要な情報が十分に得られない場合がある．送信が許可されない理由の一つは，クラッシュレポートを送信することでユーザが得られるメリットがユーザにとってわかりづらいことではないかと考えられる．

本稿の目的は，ユーザからのクラッシュレポートの送信を促進するために，クラッシュレポートの送信によって得られるユーザのメリットを明らかにすることである．これを達成するために，本稿ではオープンソースプロジェクトである Firefox プロジェクトを対象に分析を行った．具体的には，Firefox プロジェクトのクラッシュレポートシステム，及び，不具合管理システムに蓄積されたデータを用いて，1) 送信されたレポートの不具合修正への利用率，2) レポートが利用される不具合の種類，3) 不具合修正の完了率，および，4) 完了までの期間の変化に関して分析を行った．

以降，第 2 章では，クラッシュレポートシステム，および，これを利用した不具合修正のプロセスについて説明する．第 3 章では，リサーチクエスチョンを設け，第 4 章では，クラッシュレポートが不具合修正に与える影響について分析し，第 5 章では関連研究をまとめる．最後に，第 6 章で本稿のまとめと，今後の課題について述べる．

[1] 九州大学，福岡市
Kyushu University, Japan
[2] 奈良先端科学技術大学院大学，生駒市
Nara Institute of Science and Technology, Japan
[a)] kosuda@posl.ait.kyushu-u.ac.jp
[b)] kamei@ait.kyushu-u.ac.jp
[c)] akinori-i@is.naist.jp
[d)] ubayashi@ait.kyushu-u.ac.jp
[*1] クラッシュが起こった際のメソッドの呼び出し過程

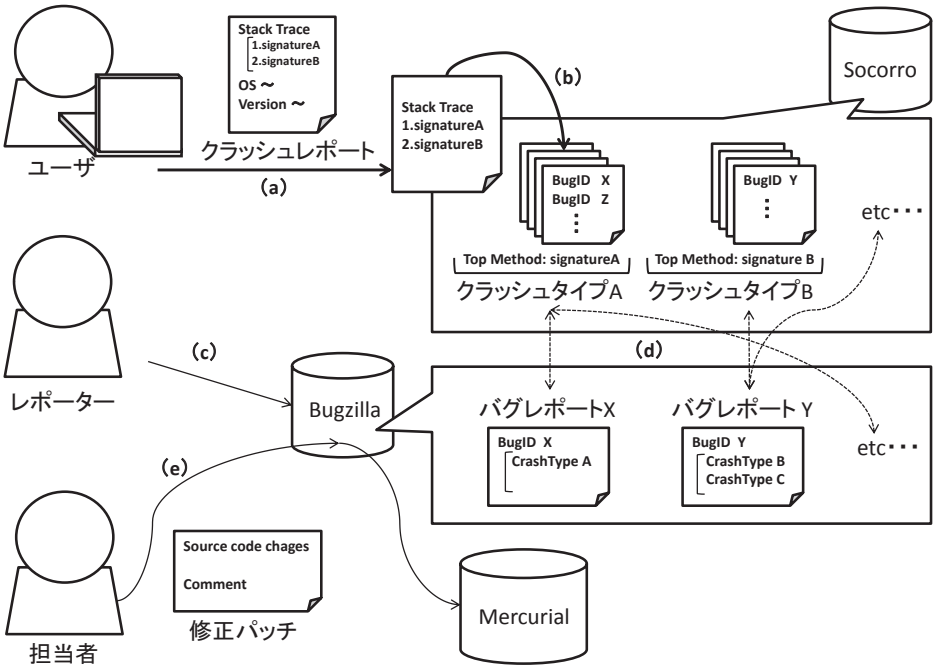

図 1　クラッシュレポートシステムの流れ [3]

2. クラッシュレポートシステムの仕組み

2.1 クラッシュレポートシステム

ユーザからのフィードバックを得るために，一部のソフトウェアシステムにはクラッシュレポートシステムが組み込まれている．ユーザがソフトウェアを利用している最中に，予期せぬ停止（クラッシュ）が発生すると，クラッシュレポートシステムが実行環境に関する情報をクラッシュレポートにまとめ，開発プロジェクトのクラッシュサーバへ送る（図 1（a））．

クラッシュレポートには，スタックトレースやユーザ環境（OS の種類やバージョン，当該ソフトウェアのバージョン）の情報が含まれている．スタックトレースは，メソッドの呼び出しの順序を記録したものであり，各スタックフレームに，順番，モジュール，メソッドシグニチャ，対応するソースコードへのリンクが記載されている．メソッドシグニチャは，単にメソッドの名前のみを表しているのではなく，メソッドの名前とそのメソッドの引数の組み合わせを表す．

クラッシュレポートは，開発プロジェクトのクラッシュサーバへ送信されると，レポートごとにユニークな番号が割り当てられる．その際，サーバでは，類似するクラッシュレポートを同一種類のレポートとしてまとめる．これは，クラッシュレポートシステムによって集められるクラッシュレポートの数が膨大であり，開発者がどういったクラッシュが頻繁に発生しているかを把握できないためである．同一種類のクラッシュレポートをまとめることは，頻出するクラッシュレポートの種類の把握を容易にし，開発者が優先的に取り組むクラッシュを決定する際に役立つ．Firefox プロジェクトの Socorro の場合，クラッシュレポートはスタックトレースのトップメソッドシグニチャに基づいてグループ分けされる（図 1（b））．

2.2 クラッシュレポートの報告からソースコードの修正までのプロセス

本節では，クラッシュレポートが送られてからソースコードが修正されるまでのプロセスを紹介する．まず，先にも述べたように，ユーザがソフトウェアを利用している最中に，クラッシュが発生すると，クラッシュレポートが開発プロジェクトのサーバへ送られる（図 1（a））．次に，クラッシュサーバは，大量のクラッシュレポートを類似するレポートごとにグループ化する（図 1（b））．

レポータと呼ばれる開発者が，クラッシュレポートを調査しており，もし，クラッシュレポートの原因である不具合が，不具合管理システム（例えば，Bugzilla）に起票されていない場合，レポータは，その不具合を不具合管理システムに新たな不具合として起票する（図 1（c））．そして，開発者は，クラッシュタイプと不具合票を関連付ける．クラッシュタイプと不具合は，多対多の関係で関連付けられ，一つのクラッシュタイプが複数の不具合と関連付けられる場合や，一つの不具合が複数のクラッシュタイプと関連付けられる場合もある（図 1（d））．開発者は議論を重ね，優先的に修正する不具合を選び，修正対象の不具合を決定後，その不具合に修正するための担当者を割り当てる [4]．

開発者は不具合を修正するためのソースコード（修正パッチ）を作成すると，不具合管理システムへ提出する（図

1（e)）．パッチの内容に誤りが無い場合，パッチは開発プロジェクトのソースコードへ統合される．

3. リサーチクエスチョン

本稿では，ユーザがクラッシュレポートを送信することによって，ユーザ自身が得られるメリットを明らかにする．目的が達成されることで，クラッシュレポートの送信を促し，開発プロジェクトに不具合の修正に必要な情報がより多く集まることが期待できる．本稿では，1) クラッシュレポートが不具合に関連付けられる確率，2) クラッシュレポートが関連付けられる不具合の種類，3) 不具合修正の完了率，および，4) 完了までの期間への寄与に関して調査を行う．下記の 4 つのリサーチクエスチョンに取り組む．

- **RQ1．送信されたクラッシュレポートのうち，不具合と関連付けられる割合は？**
  2 章で述べたように，開発プロジェクトでは，ユーザから送られてくるクラッシュレポートと不具合票とを関連付け，クラッシュレポートの内容も活用しながら，不具合の解決に取り組む．しかしながら，全てのクラッシュレポートが不具合票と関連付けられているわけではなく，全てが不具合解決に用いられているわけではない．本 RQ では，クラッシュレポートがどの程度役立てられているか（関連付けられているか）について示す．
- **RQ2．クラッシュレポートと関連付けられる不具合には共通した特徴があるのか？**
  ユーザの使用状況に密接に関わる特徴をもつ不具合がクラッシュレポートと関連付けられて修正されるのであれば，少なくとも特定の機能については，ユーザの使用状況の改善が見込める．
- **RQ3．クラッシュレポートを送信することで不具合の修正が完了される確率は高くなるのか？**
  ユーザ環境で発生した不具合の原因が解消されないままでいるのは，ユーザにとって不利益である．クラッシュレポートが送信されることで，不具合修正が完了される確率がどの程度高くなるかについて調査する．
- **RQ4．早い段階でクラッシュレポートとの関連付けが行われると，不具合の修正期間は短縮されるのか？**
  ユーザ環境での不具合が解消されたとしても，解消までに時間がかかりすぎるのは，ユーザにとって不利益である．クラッシュレポートが送信されることで，不具合修正に要する時間がどの程度短縮されるかについて調査する．

4. ケーススタディ

4.1 データセット

本稿では，ケーススタディの題材として，大規模なオープンソースプロジェクトである Mozilla Firefox プロジェクトを選んだ．本稿で対象とするクラッシュデータ，及び，不具合管理システムのデータを表 1 に示す．

表 1 データセットの概要

	クラッシュレポジトリ	不具合管理システム
対象期間	2010/02/25 ～ 2013/10/31	1998/04/07 ～ 2013/10/31
レポート数（不具合票数）	463,808,045	930,000

表 1 に示すようにクラッシュレポートの期間は，2010/02/25 から 2013/10/31 までである．クラッシュレポートは，Firefox プロジェクトのクラッシュサーバ [*2] から取得した．ただし，一部の日付のクラッシュレポートは存在せず，取得することができなかった [*3]．また，不具合管理システムは公開されている範囲（1998 年 4 月 7 日～2013 年 10 月 31 日）の期間を対象とした．2010 年以前の不具合管理システムのデータを用いる理由は，2010 年以前に不具合システムに登録されたものが，クラッシュレポートと関連付けられる可能性があるためである．

(RQ1) 送信されたクラッシュレポートのうち，不具合と関連付けられる割合は？

概要． 本 RQ では，クラッシュレポートがどの程度役立てられているか（関連付けられているか）について示す．

アプローチ． アプローチとして，全クラッシュレポートのうち，不具合に関連付けられているクラッシュレポートの割合を求める．不具合と関連付けられたクラッシュレポートの総数を全クラッシュレポート数で割ることで，不具合との関連付けがなされているクラッシュレポートの割合を求める．

各クラッシュレポートが不具合票に関連付けられているか否かは，Bugzilla の不具合票の *Crash Signature* という項目を参照し，その項目内に各クラッシュレポートの名前が記述されているかで判断した．

また，ユーザから頻繁に送信される種類のクラッシュレポートが優先的に不具合票と関連付けられているのかも調査した．含有するクラッシュレポートが多いクラッシュタイプの上位 20 位をトップクラッシュとし，トップクラッシュとそれ以外のクラッシュタイプとの間に関連付けされる割合の差が存在するのかを調査した．

結果． 結果として，まず，全クラッシュレポートを対象とした実験結果を述べる．全クラッシュレポートのうち，不具合に関連付けられているクラッシュレポートの割合は約 15.3%（70,884,417/463,808,0450）であった．クラッシュレポートが送信されたからといって，全てのクラッシュレポートが不具合票の解決に利用されている（つまり，関連

[*2] https://crash-analysis.mozilla.com/
[*3] 取得できなかった日付は，2010/3/14・4/12・4/12-4/15・6/11-6/15・6/17・7/23・8/5-6・9/15・2013/8/28/-9/2

表 2 (RQ1) レポート数上位 20 タイプと関連付けの有無

ランク	レポート数	全レポート数に占める割合 (%)	関連付けの有無
1	19,466,567	4.20	無
2	8,917,195	1.92	有
3	8,089,774	1.74	無
4	6,557,300	1.41	有
5	5,016,823	1.08	無
6	3,952,537	0.85	有
7	3,878,656	0.84	無
8	3,809,644	0.82	無
9	3,609,956	0.78	有
10	3,354,696	0.72	無
11	3,275,063	0.71	無
12	3,250,672	0.7	無
13	3,224,745	0.7	有
14	2,962,936	0.64	無
15	2,660,847	0.57	無
16	2,523,326	0.54	有
17	2,522,663	0.54	有
18	2,305,147	0.50	無
19	2,278,958	0.49	有
20	2,081,463	0.45	無

付けられている）わけではないことがわかった．

次に，トップクラッシュを対象とした実験結果を述べる．表 2 に含有レポート数上位 20 タイプと関連付けの有無についての調査の結果を示す．全クラッシュタイプのうち，クラッシュレポートの含有数の多いものから 20 種類と，各クラッシュタイプの含有レポートの全レポートに占める割合，不具合との関連付けの有無を示した．

トップクラッシュであるクラッシュレポートのうち，不具合票と関連付けられているクラッシュレポートの割合は約 35.8%であった．それに対して，トップクラッシュ以外に属するクラッシュレポートのうち，不具合票と関連付けられているクラッシュレポートの割合は約 10.1%であった．

そのため，含有するクラッシュレポートの数が多い，つまり，ユーザから多数送られてくるクラッシュタイプが優先的に不具合と関連付けられているものと考えられる．

送信されたクラッシュレポートが関連付けられている割合は約 *15%*である．トップクラッシュに着目すると，全トップクラッシュのうち，関連付けられているトップクラッシュの割合は約 *35.8%*である．送信された数の多いクラッシュレポートは優先的に関連付けられていることがわかった．

(RQ2) クラッシュレポートと関連付けられる不具合には共通した特徴があるのか？

概要. クラッシュレポートと関連付けられて修正される不具合には，ユーザの使用状況に密接に関わる特徴を持つものが多いか否かについて調査する．

アプローチ. 不具合票には関連項目を表すキーワードが任意の個数記載されている．これに着目して，クラッシュレポートが関連付けられる可能性の高い不具合の種類にはどのようなものがあるのか調査する．不具合票からキーワードを抽出し，それが複数ある場合には，その不具合はそれら全てのキーワードを持つものとして扱い，組み合わせを考慮しない．

あるキーワードを含む不具合票のうち，クラッシュレポートに関連付けられているものの数をそのキーワードを含む不具合票の数で割ることで，そのキーワードを含む不具合票がクラッシュレポートと関連付けられる確率を算出した．そのキーワードが付与される条件（つまり，各不具合の意味）は，Firefox プロジェクトが公開している情報 [*4] を参照した．

結果. 付与されている不具合が高確率でクラッシュレポートと関連付けられているキーワード上位 10 種類について調査した結果を表 3 に，関連付けされる確率が 0 であったものについての結果を表 4 に，それぞれ示す．個々の不具合の特徴を表すキーワードのみを記すため，表からは一部のキーワードを除外している．

不具合に付与されると，その不具合がクラッシュレポートに関連付けられる可能性が高くなっているキーワードは，その不具合が悪用可能である，多くのユーザへ波及する可能性が高い，クラッシュを引き起こす深刻なものである等の，早急な解決が求められることを示すものがある．また，解決のための再現の難易度等や，解決のために必要な情報等を示しているものなども存在する．表からは除外したが，ソフトウェアのどのバージョンで修正されたかを示しているものも数多く存在した．

一方で，付与された不具合とクラッシュレポートとの関連付けが全く行われていなかったキーワードには，その不具合が開発中に発見されたことを示すものや，特定の機能やコードの操作・変更時に発生したことを示すものが多く存在した．また，表から除外したキーワードには，特定の言語やアプリケーションを用いた際に発生したことを示すものやユーザエクスペリエンスに関するものが数多く存在した．傾向として，付与された不具合がクラッシュレポートに関連付けられている可能性の高いキーワードはその危険性や解決のための方法，解決した時期を示すものが多い．そして，関連付けられていないキーワードはその不具合が起きた条件に関するものが多い．

[*4] https://bugzilla.mozilla.org/describekeywords.cgi

表 3 (RQ2) 付与されている不具合の関連付けが高確率なキーワード

キーワード	関連付けられる確率 (%)	要約
steps-wanted	56.4	特定の人からの情報が不具合再現に際して重要
reproducible	49.6	再現可能な手順が存在する
crash	49.5	クラッシュを引き起こす深刻な不具合
sec-critical	25.2	悪用可能で，多くのユーザに危険を与える可能性のある脆弱性を持つ
testcase-wanted	24.3	既存及び追加のテストケースのにより有益な情報を得られる．より単純なテストケースにすることが望まれる
testcase	14.2	簡略化されたテストケースが導入されている
sec-high	13.6	訪問サイトでの通常のブラウジング範囲を超えない行動が，少数のユーザの危険につながる可能性のある悪用可能な脆弱性を持つ
regressionwindow-wanted	12.6	この不具合は回帰であり，それが起きた期間を絞り込む際に特定の人からの情報が重要になる
qawanted	11.4	より多くの情報が必要
regression	11.0	修正後に再発した

表 4 (RQ2) 付与されている不具合の関連付けが低確率なキーワード

キーワード	関連付けられる確率 (%)	要約
polish	0.00	ユーザーインターフェイスの改善のためのわずかな変更が必要な不具合
uiwanted	0.00	UI の設計支援を必要とする不具合
l12y	0.00	ハードコード文字列，サイズ情報，ハードコーディングされたフォントのようなローカライズの問題に関連
privacy	0.00	一般的なコンポーネント：セキュリティに属していないユーザーのプライバシーに関連
late-l10.00n	0.00	ローカライズのフリーズ（典型的には，ベータ版の後）
embed	0.00	開発作業の埋め込みを妨げる不具合
student-project	0.00	Mozilla プロジェクトで作業しようとしている学生に適切であろう機能拡張要求
css-moz	0.00	Mozilla の CSS の拡張機能の不具合

クラッシュレポートと関連付けられて修正される不具合には，ユーザの使用状況に密接に関わる特徴を持つものが多く，ユーザに大きな被害を与えかねないことを示すキーワードを含むものが多い．

表 5 RQ3,RQ4 におけるデータセット

	クラッシュレポジトリ	不具合管理システム
対象期間	2010/08/24 ～ 2013/05/04	2010/08/24 ～ 2013/05/04
レポート数（不具合票数）	352,833,331	278,653

(RQ3) クラッシュレポートを送信することで不具合の修正が完了される確率は高くなるのか？

概要. 不具合の原因が解消されないままでいるのは，ユーザにとって不利益であるため，クラッシュレポートが関連付けられた不具合は特に修正が完了することが望まれる．クラッシュレポートが関連付けられることで，不具合修正される確率が高くなるか否かを調査する．

アプローチ. 本 RQ では，クラッシュレポートと関連付けられている不具合票のうち，修正が完了しているものの数を関連付けられている不具合票全体の数で割ることで修正完了率を求める．また，クラッシュレポートと関連付けられていない不具合票に関しても同様に修正完了率を求める．

クラッシュレポートには，そのクラッシュレポートが属するクラッシュタイプが関連付けられている不具合の ID が bug list として記載されている．ここに記載されている不具合を対象に不具合の修正が完了したか否かについて調査を行うが，修正が完了しているかの判断は，不具合管理システムの不具合票の states を参照し，CLOSE となっているものを修正完了として扱う．

調査を行う直前に登録された不具合は修正可能な期間が他の不具合よりも短くなり，当然修正の完了率が下がることになるため，調査を行った 180 日前までに不具合票が登録されたもののみを対象とした．クラッシュレポートは 2010/02/25 以降のデータしかないため，それ以前にどのクラッシュレポートが送信されたかを知ることができない．そのため，2010/02/25 から 2010/08/24 までの 180 日の間に送られてきたクラッシュレポートを調査し，その中に含まれないものは，2010/08/24 以前には送られてきたことのないクラッシュレポートであると仮定した．これにより，2010/08/24 以降のクラッシュレポートのみを調査対象と

すれば，対象となるクラッシュレポート全てにおいて，各レポートが最初に送信されてきた時から観測することができるため，クラッシュレポートが不具合修正に与える影響を正確に分析することができる．また，不具合票はクラッシュレポートよりも古い期間のものも参照できる．これにより分析に偏りが生じることを避けるため，不具合票とクラッシュレポートの対象期間をそろえることとした．以上より，この分析におけるクラッシュレポートと不具合票の対象期間は表 5 の通りである．

結果． クラッシュレポートと関連付けられている不具合の修正完了率は，約 29%であり，関連付けられていない不具合の修正完了率は約 48%であった．クラッシュレポートは不具合の情報を提供するものである．このクラッシュレポートとの関連付けなしに登録されている不具合は，主に開発者が発見したものである．開発者が不具合を発見した場合，不具合の発生条件や再現方法を直接得ることができる．それらの情報を持った状態で修正を開始できるため，修正を完了できる確率が高くなるのではないかと考えられる．

> クラッシュレポートを送信したとしても，不具合修正が完了する確率は高くならない．

(RQ4) 早い段階でクラッシュレポートとの関連付けが行われると，不具合の修正期間は短縮されるのか？

概要． ユーザ環境での不具合が解消されるまでの間，ユーザは不利益を被ることになるため，不具合の原因となる不具合の迅速な修正が望まれる．本 RQ では，クラッシュレポートとの関連付けにより，不具合の修正期間は短縮されるのか調査する．

アプローチ． クラッシュレポートに関連付けられている不具合には，クラッシュレポートが送られてきた後に Bugzilla に登録された不具合と，元々登録されていた不具合に後から送信されてきたクラッシュレポートが関連付けられる不具合がある．これらのそれぞれに属する不具合票の修正期間の最小値，中央値，平均値，最大値を算出して比較する．また，この分析におけるクラッシュレポートと不具合票の対象期間は RQ3 と同じく，表 5 の通りである．

修正が行われた期間は，不具合票に記載されている「その不具合が起票された日時」から「最後に修正がなされた日時」までの期間とする．これらの差をとることで修正期間を算出した．

結果． 実験結果を図 2 に示す．箱ひげ図により，不具合とクラッシュレポートのどちらが先に登録又は送信されたかと，修正期間に関する数値を示す．修正期間の最小値・中央値・平均値はクラッシュレポートが不具合の登録より

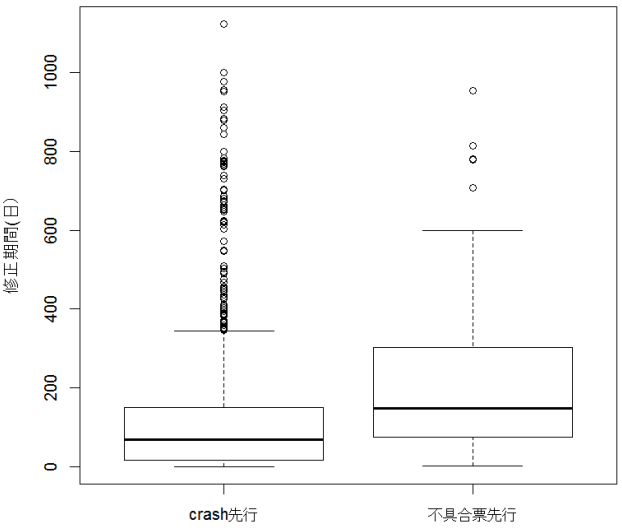

図 2 (RQ4) クラッシュレポート送信時期による修正期間の比較

も先に送られてきたものの方が修正期間は短かった．中央値に関しては，クラッシュレポートが送られていることで，80 日程度早期に解決されることがわかった．

> 早い段階でクラッシュレポートとの関連付けが行われると，不具合の修正期間は中央値で約 *80* 日短縮されることがわかった．

5. 関連研究

5.1 クラッシュレポートの分析

本稿と同様，クラッシュレポートの分析による，オープンソースプロジェクトの活動支援を目的とした研究が行われている [5][6][7][8]．例えば，Kim ら [8] は，クラッシュレポート全体の大半を占めている少数の種類のクラッシュレポート (トップクラッシュ) が存在することに着目し，それらの原因をできる限り早い段階で解決するため，製品のリリース後の短い期間で送られてきた少数のクラッシュレポートから，あるクラッシュレポートがトップクラッシュか否かを判定する予測モデルの構築を行い，Firefox で 75%，Thunderbird で 90%の精度を出している．

本稿では，トップクラッシュに着目した不具合修正が実際に行われているのかについて調査した点が新しい．

5.2 データソースの関連付け

複数のデータソース（リポジトリ）の関連付けにより，単体のそれでは得られない知見を得られる場合がある．そこで，2 つ以上のリポジトリの関連付けの手法に関する研究が行われている [9][10][4][11]．例えば，Śliwerski [4] らは，バージョン管理システムのコミットログに含まれる不具合 ID を正規表現により検出し，不具合票との関連付けを行っている．また，Khomh ら [7] は，クラッシュレポートによる修正の優先度判定の改善のために波及度（エント

ロピー）を利用した分析を行った際，その評価の方法として，そのクラッシュレポートと関連付けられている不具合票を利用した．この研究で，クラッシュレポートのエントロピーの高低が，関連付けられる不具合の個数や修正時間に影響を与えることが示されている．

従来研究では，クラッシュレポートシステムによる不具合修正の優先度の判定を行い，その評価にクラッシュレポートと不具合票との関連付けを用いている．一方で，本稿では，クラッシュレポートと不具合が関連付けられることにより，不具合の修正完了にどのような影響を持つのかについて分析した点が新しい．

6. おわりに

本稿では，クラッシュレポートが不具合修正に与える影響について調査した．そのために，クラッシュレポートが不具合票に関連付けられている割合，クラッシュレポートが関連付けられる確率の高い不具合の特徴，クラッシュレポートが関連付けられることによる修正完了率への影響，及び関連付けられているクラッシュレポートと不具合の登録の前後による修正期間への影響について分析を行った．

Firefox プロジェクトで 3 年間に蓄積された約 460,000,000 件のクラッシュレポート, 及び, Bugzilla で 15 年間に蓄積された 930,000 件の不具合票を対象としたケーススタディの結果得られた知見は，以下の通りである．

- 送信されたクラッシュレポートが関連付けられている割合は約 15%である．トップクラッシュに着目すると，全トップクラッシュのうち，関連付けられているトップクラッシュの割合は約 35.8%である．
- クラッシュレポートと関連付けられて修正される不具合には，ユーザの使用状況に密接に関わる特徴を持つものが多く，ユーザに大きな被害を与えかねないことを示すキーワードを含むものも多い．
- クラッシュレポートを送信することで不具合の修正が完了される確率は高くならない
- 早い段階でクラッシュレポートとの関連付けが行われると，不具合の修正期間は短縮される．

今後の課題として，不具合票に記載されている概要や，開発者同士のコメントのマイニングによる不具合票の分類があげられる．これにより，より詳細な特徴に関する分類が可能になるだけでなく，キーワードが付与されていないバグに関しても，分類の対象とすることができると考えられる．

謝辞 本研究の一部は，日本学術振興会 科学研究費補助金（若手 A：課題番号 24680003，挑戦的萌芽：課題番号 25540026）による助成を受けた．

参考文献

[1] Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K., Adams, B. and Hassan, A. E.: Revisiting Common Bug Prediction Findings Using Effort Aware Models, *Proc. Int'l Conf. on Software Maintenance (ICSM'10)*, pp. 1–10 (2010).

[2] Khomh, F., Dhaliwal, T., Zou, Y. and Adams, B.: Do faster releases improve software quality? An empirical case study of Mozilla Firefox, *Proc. Int'l Conf. on Mining Software Repositories (MSR'2012)*, pp. 179–188 (2012).

[3] 長本貴光，亀井靖高，伊原彰紀，鵜林尚靖：クラッシュログを用いたソースコード不具合箇所の特定に向けた分析，情報処理学会研究報告, ソフトウェア工学研究会，pp. 1–6 (2013).

[4] Śliwerski, J., Zimmermann, T. and Zeller, A.: When Do Changes Induce Fixes?, *Proc. Int'l Conf. on Mining Software Repositories (MSR'05)*, pp. 1–5 (2005).

[5] Dang, Y., Wu, R., Zhang, H., Zhang, D. and Nobel, P.: ReBucket: a method for clustering duplicate crash reports based on call stack similarity, *Proc. Int'l Conf. on Softw. Eng. (ICSE'12)*, pp. 1084–1093 (2012).

[6] Dhaliwal, T., Khomh, F. and Zou, Y.: Classifying field crash reports for fixing bugs: A case study of Mozilla Firefox, *Proc. Int'l Conf. on Software Maintenance (ICSM'11)*, pp. 333–342 (2011).

[7] Khomh, F., Chan, B., Zou, Y. and Hassan, A. E.: An Entropy Evaluation Approach for Triaging Field Crashes: A Case Study of Mozilla Firefox, *Proc. Working Conf. on Reverse Engineering (WCRE'11)*, pp. 261–270 (2011).

[8] Kim, D., Wang, X., Kim, S., Zeller, A., Cheung, S. C. and Park, S.: Which Crashes Should I Fix First?: Predicting Top Crashes at an Early Stage to Prioritize Debugging Efforts, *IEEE Trans. Softw. Eng.*, Vol. 37, No. 3, pp. 430–447 (2011).

[9] Bangcharoensap, P., Ihara, A., Kamei, Y. and Matsumoto, K.: Locating Source Code to Be Fixed Based on Initial Bug Reports - A Case Study on the Eclipse Project -, *Proc. Int'l Workshop on Empirical Software Engineering in Practice (IWESEP 2012)*, pp. 10–15 (2012).

[10] Servant, F. and Jones, J. A.: WhoseFault: automatic developer-to-fault assignment through fault localization, *Proc. Int'l Conf. on Softw. Eng. (ICSE'12)*, pp. 36–46 (2012).

[11] Zhou, J., Zhang, H. and Lo, D.: Where should the bugs be fixed? - more accurate information retrieval-based bug localization based on bug reports, *Proc. Int'l Conf. on Softw. Eng. (ICSE'12)*, pp. 14–24 (2012).