Vol.2013-SE-179 No.12
2013/3/11

# 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

長本 貴光[1,a)]　亀井 靖高[1,b)]　伊原 彰紀[2,c)]　鵜林 尚靖[1,d)]

**概要**：ソフトウェアの品質向上を目的として，クラッシュレポートを収集する開発プロジェクトが増えている．クラッシュレポートは，ソフトウェアがクラッシュした際に自動的に開発プロジェクトに送信され，その内容は実行環境に関する情報（バージョン，OS，メモリ情報等）をまとめたものである．本研究では，不具合の修正箇所の特定にかかる労力を軽減するために，クラッシュレポートからソースコードの不具合箇所を自動的に特定することを最終的な目標としている．本稿では，その前段階として，クラッシュレポートからソースコードの修正箇所をどのように関連付けられるかについて分析を行った．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のバグ含有率，及び，リンク率は，全バージョンの平均でそれぞれ 62.26%と 15.4%であった．

## 1. はじめに

ソフトウェアシステムのリリース後に，利用者の環境でバグによる不具合が発生した場合，当該開発プロジェクトの評判を落とすだけでなく，利用者にも少なくはない時間的／金銭的損害を与える [3]．そのため，リリースの前段階で，全てのバグが取り除かれていることが望ましい．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，開発者が全てのバグ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し，利用者からのフィードバックを自動的に集めることで，できる限り素早くバグの発生を認識し，修正する仕組みを構築している [5]．クラッシュレポートシステムは，ユーザーのシステムが予期していない停止状態（クラッシュ）に陥った際に，スタックトレース[*1]やユーザー環境の情報等をクラッシュレポートにまとめ，開発プロジェクトのサーバへ自動的に送るシステムである．

しかしながら，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は非常に膨大であり，例えば，オープンソースプロジェクトの 1 つである Firefox の場合は，1 日に 250 万件ものクラッシュレポートが集められる [4]．Firefox プロジェクトでは，クラッシュレポートシステムを用いることで，クラッシュレポートを Socorro [9] というサーバに集め，類似するクラッシュレポートをひとまとめにするなどの工夫を行っているものの，多数のクラッシュレポートに対してどのソースコードファイルを修正すればよいかを手動で特定するには大きな労力を要するという問題がある．

本研究の最終的なゴールは，ソースコードの不具合箇所の特定にかかる労力を軽減するために，クラッシュレポートからソースコードの不具合箇所を自動的に特定することである．このゴールを達成するために，本稿ではその前段階として，クラッシュレポートからソースコードの修正箇所をどのように関連付けられるかを分析する．まず，クラッシュレポートからソースコードの修正箇所を関連付ける方法を示し，その後，Firefox プロジェクトをケーススタディの題材として，どの程度の割合でクラッシュレポートをソースコードに関連付けられるのかを示す．

以降，2 章でクラッシュレポートシステム，及び，クラッシュレポートシステムを用いた開発プロセスについて説明する．3 章では，ケーススタディの概要と結果について述べ，4 章では関連研究をまとめる．最後に，5 章で本稿のまとめと，今後の課題について述べる．

## 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

ユーザー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．ユーザーがソフトウェアを利用している

[1] 九州大学
Kyushu University, Japan
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology, Japan
[a)] nagamoto@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 [6]

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ー環境（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を組み合わせて決められる．

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，ユーザー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

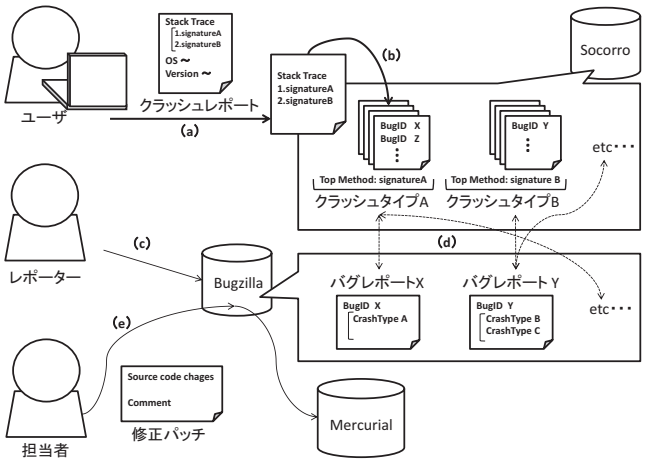


図 1 クラッシュレポートシステムの流れ

表 1 データセットの概要

| | クラッシュレポジトリ | バージョン管理システム |
|---|---|---|
| 対象期間 | 2010/02/25 ～ 2012/10/31 | 2010/02/25 ～ 2012/12/26 |
| レポート数（コミット数） | 326,941,554 | 74,883 |

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バグが，バグ管理システム（例えば，Bugzilla）に起票されていない場合，レポータは，そのバグをバグ管理システムに新たなバグとして起票する（図 1（c））．そして，開発者は，クラッシュタイプとバグ票を関連付ける．クラッシュタイプとバグは，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バグと関連付けられる場合や，一つのバグ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，優先的に修正するバグを選び，修正対象のバグを決定後，そのバグに修正するための担当者を割り当てる [8]．

開発者はバグ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，バグ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 [4]．

### 2.3 本稿で取り組む課題とアプローチ

本研究ではバグの修正にかかる労力を軽減するために，クラッシュレポートからソースコードの不具合箇所を自動的に特定するということを目的にしている．図 1 を用いて目的を説明すると，図 1 の (a) から (e) までの修正過程のデータを用いることで，新たなクラッシュレポートが開発プロジェクトに送られた際（図 1(a)）に，そのバグに起因するソースコードの修正箇所を自動的に特定する（図 1 (e)）．

**課題**：修正箇所を自動的に特定する研究を行うためには，クラッシュレポートと，そのクラッシュレポートの原因であるバグの修正のために変更されたソースコードとが関連付けられたデータセットを準備する必要がある．このデータセットを用いることで，クラッシュレポートの特性値（クラッシュの発生時期やスタックトレース等）から，ソースコードの修正箇所を自動的に特定できるか，を研究することができる．そのため，本稿では，クラッシュレポートからソースコードの修正箇所を関連付ける方法を示す．

また，ソフトウェア工学分野のデータは，分析を行うためにデータをクリーニングする過程で，たとえ元のサンプル数が大きかったとしても，最終的なサンプル数が小さくなるケースがある．例えば，バグ管理システムのバグ報告を対象に，一度修正完了と判断されたバグ報告が実は正し

表 2　バージョンごとのクラッシュタイプ数とレポート数

| Ver. | リリース日 | サポート終了日 | サポート期間 | クラッシュタイプ数 | クラッシュレポート数 |
|---|---|---|---|---|---|
| 3.0 | 2008/06/17 | 2010/03/30 | 93 週間 | 327,501 | 8,777,133 |
| 3.5 | 2009/06/30 | 2011/06/21 | 103 週間 | 503,319 | 15,016,027 |
| 3.6 | 2010/01/21 | 2012/04/24 | 118 週間 | 2,365,896 | 129,457,295 |
| 4 | 2011/03/22 | 5 のリリース | 13 週間 | 799,297 | 39,874,791 |
| 5 | 2011/06/21 | 6 のリリース | 8 週間 | 332,971 | 11,712,949 |
| 6 | 2011/08/16 | 7 のリリース | 6 週間 | 305,582 | 10,456,612 |
| 7 | 2011/09/27 | 8 のリリース | 6 週間 | 303,962 | 10,352,390 |
| 8 | 2011/11/08 | 9 のリリース | 6 週間 | 336,786 | 12,233,255 |
| 9 | 2011/12/20 | 10 のリリース | 6 週間 | 289,928 | 9,880,646 |
| 10* | 2012/01/31 | 11 のリリース | 6 週間 | 260,329 | 7,443,703 |
| 11 | 2012/03/13 | 12 のリリース | 6 週間 | 355,910 | 10,528,612 |
| 12 | 2012/04/24 | 13 のリリース | 6 週間 | 368,074 | 11,492,466 |
| 13 | 2012/06/05 | 14 のリリース | 6 週間 | 308,124 | 10,907,497 |
| 14 | 2012/07/17 | 15 のリリース | 6 週間 | 353,139 | 15,735,857 |
| 15 | 2012/08/28 | 16 のリリース | 6 週間 | 368,427 | 14,034,281 |
| 16 | 2012/10/09 | 17 のリリース | 6 週間 | 335,796 | 7,113,407 |
| 17* | 2012/11/20 | 18 のリリース | 6 週間 | 75,649 | 1,550,559 |
| 18 | 2013/01/08 | 継続中 | | 20,128 | 318,571 |
| 19 | 2013/02/19 予定 | | | 6,414 | 55,503 |

* バージョン 10，及び，17 は法人向けサポート版もあり，そちらは 54 週間サポートされる．

く修正されておらず再開されるという事例（Reopen）を分析した研究では，初期のサンプル数は約 18,000 件あったものの，条件等を絞り込んだ結果，最終的に 1 割未満（約 1,500 件）になった [7]．本稿では，クラッシュデータを扱うため，初期のサンプル数は非常に大きいものの，どの程度の件数を関連付けられるかはあらかじめ調査しておく必要がある．

**アプローチ**：クラッシュリポジトリに含まれるクラッシュレポートには，ソースコードの変更箇所は示されておらず，また，バージョン管理システムのソースコードの変更履歴にも，クラッシュレポートの ID は示されていない．

そこで，本稿では，バグ管理システムによって管理されているバグ ID に着目する．レポータがクラッシュタイプとバグ票を関連付けた場合，同じクラッシュタイプのクラッシュレポートがサーバに送信されると，以前に関連付けられたバグ ID が自動的に付与される．

一方，開発プロジェクトでは，ソースコードを修正し，バージョン管理システムにコミットする際，コミットの目的をコメント中に付与することが多い．例えば，バグの修正を目的とする場合，”Fix #10000”といったように，どのバグを修正したかを明記する．実際にこの経験則に基づいた様々な研究が行われている [8]．

本稿では，まず，クラッシュレポートに付与されているバグ ID を抽出し，そのバグ ID が修正されている編集履歴が見つかるか否かで，ソースコードとの関連付けができるかを判断した．

## 3. ケーススタディ

### 3.1 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，バージョン管理システムのデータを表 1 に示す．

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 2012/10/31 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]．本研究で対象としたバージョンの一覧と，各レポート数の内訳を表 2 に示す．各バージョンは系列でまとめており，例えば 3.0.1 や 3.0a1 や 3.0pre といったものをまとめてバージョン 3.0 としている．バージョン 4.0 以降は，Firefox でリリースのポリシーが変わったため [5]，マイナーバージョンごとに分けずにひとつにした．今回の分析対象期間（2012/10/31）以降にリリースされたバージョンの 17，18，19 に対してクラッシュレポートが存在するのは，それらのバージョンのアルファ版がリリースされているためである．

ソースコードの変更履歴は，Firefox が開発に用いているバージョン管理システム・Mercurial からクローンを行い取得した[*4]．その後，log コマンドにより，変更履歴の

[*2] https://crash-analysis.mozilla.com/
[*3] 取得できなかった日付は，2010/3/13・4/11・4/16・6/10・6/16・7/22・8/4・9/14
[*4] hg clone http://hg.mozilla.org/mozilla-central/ src

表 3　クラッシュタイプのリンク率

| Ver. | A. クラッシュタイプ数 | バグ ID を含むレポート | | バグ ID を含まない クラッシュタイプ | バグ ID 含有率 (%) (B+C) / A | リンク率 (%) B / A |
|---|---|---|---|---|---|---|
| | | B. リンク可 | C. リンク不可 | | | |
| 3.0 | 327,501 | 212 | 1,840 | 325,449 | 0.63 | 0.06 |
| 3.5 | 503,319 | 242 | 2,047 | 501,030 | 0.45 | 0.05 |
| 3.6 | 2,365,896 | 450 | 3,090 | 2,362,356 | 0.15 | 0.02 |
| 4 | 799,297 | 867 | 3,559 | 794,871 | 0.55 | 0.11 |
| 5 | 332,971 | 634 | 2,766 | 329,571 | 1.02 | 0.19 |
| 6 | 305,582 | 655 | 2,786 | 302,141 | 1.13 | 0.21 |
| 7 | 303,962 | 678 | 2,750 | 300,534 | 1.13 | 0.22 |
| 8 | 336,786 | 701 | 2,809 | 333,276 | 1.04 | 0.21 |
| 9 | 289,928 | 686 | 2,808 | 286,434 | 1.21 | 0.24 |
| 10 | 260,329 | 779 | 2,928 | 256,622 | 1.42 | 0.30 |
| 11 | 355,910 | 718 | 2,877 | 352,315 | 1.01 | 0.20 |
| 12 | 368,074 | 734 | 2,859 | 364,481 | 0.98 | 0.20 |
| 13 | 308,124 | 762 | 2,747 | 304,615 | 1.14 | 0.25 |
| 14 | 353,139 | 929 | 3,130 | 349,080 | 1.15 | 0.26 |
| 15 | 368,427 | 972 | 3,057 | 364,398 | 1.09 | 0.26 |
| 16 | 335,796 | 896 | 2,741 | 332,159 | 1.08 | 0.27 |
| 17 | 75,649 | 650 | 2,010 | 72,989 | 3.52 | 0.86 |
| 18 | 20,128 | 514 | 1,277 | 18,337 | 8.90 | 2.55 |
| 19 | 6,414 | 251 | 665 | 5,498 | 14.28 | 3.91 |

コメントを取得し，そのコメントに対して正規表現を適用してバグの修正か否か，及び，そのバグ ID を抽出した．

### 3.2　結果

#### 3.2.1　クラッシュタイプのリンク率

各バージョンにおける，クラッシュタイプ数，バグ ID を含むクラッシュタイプの数とその内，Mercurial と関連付けられるクラッシュタイプの数，クラッシュタイプ数に対してどれぐらいの割合でバグ ID が付与されているか（バグ ID 付与率），付与されているバグ ID が Mercurial のコミットログのバグ ID とリンク可能かどうか（リンク率）を調べた（表 3）．バグ ID 付与率は，バグ ID を含むクラッシュタイプの数を，各バージョンの総クラッシュタイプ数で割った数である．リンク率は，Mercurial のコミットログのバグ ID と関連付けられたクラッシュタイプの数を総クラッシュタイプの数で割った数である．

クラッシュタイプの数は，各バージョン内で同じクラッシュタイプを 1 件として集計を行った．同じクラッシュタイプでも，複数のバグ ID を含むクラッシュレポートやバグ ID を含まないクラッシュレポートを同時に含む場合がある．本研究では最終的にクラッシュレポートとソースコードの関連付けを行うことが目的であるため，同じクラッシュタイプのいずれかのクラッシュレポートにバグ ID が含まれている場合，バグ ID が付与されているとみなし，それがコミットログのバグと関連付けられる場合，リンク可能として扱った．

表 4 に示すように，クラッシュタイプのバグ ID 含有率は，全バージョンの平均で 2.20%，製品版が出ていない 17 から 19 を除いた場合はおおよそ 1%程度であった．リンク率は，平均で 0.55%であり，ほとんどのクラッシュタイプは，ソースコードの修正箇所と関連付けることはできなかった．その一方で，アルファ版やベータ版のみが提供されているバージョン 17 から 19 に関しては，他のバージョンと比べて，バグ ID 含有率，リンク率ともに大きい値を示した．これは，アルファ版やベータ版が，開発者がバグを修正することを目的としたリリースであり，集中的にバグ出しをして，それを修正しているためと考える．

> クラッシュタイプのバグ含有率，及び，リンク率は，全バージョンの平均でそれぞれ *2.20%*と *0.55%*であった．

#### 3.2.2　クラッシュレポートのリンク率

クラッシュレポートにおけるリンク率を表 4 に示す．クラッシュタイプの結果の表と同様で，バグ ID 付与率は，バグ ID を含むクラッシュレポートの数を，各バージョンの総クラッシュレポートの数で割った数である．リンク率は，Mercurial のコミットログのバグ ID と関連付けられたクラッシュレポートの数を総クラッシュレポートの数で割った数である．クラッシュレポートに複数のバグ ID が含まれていた場合，そのうちの 1 つが関連付けられる場合，リンク可能として扱った．

表 4 に示すように，クラッシュレポートのバグ ID 含有率は，クラッシュタイプの結果と比べて，非常に大きく，平均で 62.26%であり，リンク率は 15.40%であった．非常に

表 4　各バージョンにおけるクラッシュレポートのリンク率

| Ver. | A. クラッシュレポート数 | バグ ID を含むレポート | | バグ ID を含まないクラッシュレポート | バグ ID 含有率 (%) (B+C) / A | リンク率 (%) B / A |
|---|---|---|---|---|---|---|
| | | B. リンク可 | C. リンク不可 | | | |
| 3.0 | 8,777,133 | 385,264 | 2,817,682 | 5,574,187 | 36.49 | 4.39 |
| 3.5 | 15,016,027 | 1,342,166 | 4,537,079 | 9,136,782 | 39.15 | 8.94 |
| 3.6 | 129,457,295 | 11,442,095 | 47,703,381 | 70,311,819 | 45.69 | 8.84 |
| 4 | 39,874,791 | 5,144,835 | 16,330,815 | 18,399,141 | 53.86 | 12.90 |
| 5 | 11,712,949 | 1,293,016 | 5,325,084 | 5,094,849 | 56.50 | 11.04 |
| 6 | 10,456,612 | 1,540,748 | 4,558,076 | 4,357,788 | 58.33 | 14.73 |
| 7 | 10,352,390 | 2,096,317 | 4,353,578 | 3,902,495 | 62.30 | 20.25 |
| 8 | 12,233,255 | 1,261,188 | 6,157,159 | 4,814,908 | 60.64 | 10.31 |
| 9 | 9,880,646 | 1,008,109 | 5,338,823 | 3,533,714 | 64.24 | 10.20 |
| 10 | 7,443,703 | 1,098,977 | 3,808,192 | 2,536,534 | 65.92 | 14.76 |
| 11 | 10,528,612 | 1,092,322 | 5,629,454 | 3,806,836 | 63.84 | 10.37 |
| 12 | 11,492,466 | 1,227,856 | 6,012,947 | 4,251,663 | 63.00 | 10.68 |
| 13 | 10,907,497 | 1,002,618 | 6,583,490 | 3,321,389 | 69.55 | 9.19 |
| 14 | 15,735,857 | 1,261,075 | 10,376,023 | 4,098,759 | 73.95 | 8.01 |
| 15 | 14,034,281 | 1,323,022 | 8,683,846 | 4,027,413 | 71.30 | 9.43 |
| 16 | 7,113,407 | 752,374 | 4,171,035 | 2,189,998 | 69.21 | 10.58 |
| 17 | 1,550,559 | 344,732 | 811,714 | 394,113 | 74.58 | 22.23 |
| 18 | 318,571 | 192,195 | 69,913 | 56,463 | 82.28 | 60.33 |
| 19 | 55,503 | 20,356 | 19,691 | 15,456 | 72.15 | 36.68 |

大きく，平均で 62.26%であり，リンク率は 15.40%であった．特に，アルファ版やベータ版のみが提供されているバージョン 17 から 19 に関しては，クラッシュタイプと同様に，他のバージョンと比べて，バグ ID 含有率，リンク率ともに大きい割合を示した．クラッシュタイプのリンク率が 1%を切っているにもかかわらず，クラッシュレポートのリンク率が 15.4%もあった理由は，開発者たちは出現頻度の高いクラッシュタイプを優先的に修正する傾向が高い [2][6] ためであると考える．

バージョン 4 から 16 に関しては，リンク率は，おおよそ 10%に収束していた．これは，ソフトウェアがリリースされた後，一定期間経過すると，そのバージョンのバグが優先的に修正されずにリンクできないレポート数が増加しているためであると考える．そして，最終的にリンク率は約 10 %に収束していると考える．

本稿の目的の 1 つは，クラッシュリポジトリとバージョン管理システムのデータの加工を行う上で，最終的にどの程度の件数を関連付けられるかを調べることである．Reopen を分析した研究では，初期のサンプル数は約 18,000 件あったものの，条件等を絞り込んだ結果，最終的に 1 割未満（約 1,500 件）であった [7] が，本研究では，1 割以上のサンプルを関連付けることができたため，予測研究を行うで一定の数はあるものと考える．

> クラッシュレポートのバグ含有率，及び，リンク率は，全バージョンの平均でそれぞれ *62.26%*と *15.40%*であった．この割合は，従来研究と比較した場合，今後の予測研究にとって一定の割合であるものと考える．

#### 3.2.3　クラッシュレポートに含まれるバグ ID の数

本稿では，クラッシュレポートに複数のバグ ID が含まれていた場合，そのうちの 1 つが関連付けられる場合，リンク可能として扱った．この仮定がどの程度正しいかを調べるために，バグ ID を含んでいたクラッシュレポートを対象にクラッシュレポート 1 件につき含まれるバグ ID の個数を調べた．その結果，約半数のバージョンで，50%以上のクラッシュレポートには 1 つのバグ ID しか含まれておらず，約半数のバージョンで 50%以上のクラッシュポートには 2 つのバグ ID しか含まれていなかった．そのため，本研究の分析に用いた仮説は妥当であると考える．

## 4.　関連研究

### 4.1　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を分析し，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 [1][2][4][6]．例えば，Khomh らは，優先的に修正することが望ましいクラッシュタイプの特定方法を提案している [4]，従来手法ではクラッシュタイプの出現頻度のみが用いられている一方で，Khomh らの提案手法では，頻度に加えてクラッシュがどの程度のユーザに分布しているかを考慮している．Firefox のベータ版である Firefox4.0b1-4.0b10 を用いて実験を行った結果，彼らの提案手法は，開発者によって実際にバグ修正に取り組まれているクラッシュをより正確に特定できることを示した．

Dhailiwal らは，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クラッシュレポートを分類する従来手法では，本来は類似していないクラッシュレポートを同一グ

ループに分類してしまうという問題に取り組んだ [2]．彼らの提案する手法は，まず，従来手法で大まかにクラッシュレポートを分類し，さらに同一グループのスタックトレースの類似性をレーベンシュタイン距離*5によって細分化する手法である．Firefox のベータ版である Firefox4.0b1-4.0b10 を用いて実験を行い，提案手法の有用性を示した．

従来研究では，クラッシュリポジトリに着目してクラッシュタイプ，及び，クラッシュレポートをどのように細分化するかについて研究している．その一方で，本稿では，クラッシュリポジトリを別のリポジトリと関連付けを行うことに着目しており，その点が従来研究との大きな違いである．従来研究を用いることは，より正確な関連付けに寄与する可能性があるので，今後の研究で行う予定である．

### 4.2 データソース（リポジトリ）のリンキング

複数のリポジトリをうまく関連付けることで，単体のリポジトリだけでは得られない知見を得られることがある．そのため，2 つ以上のリポジトリをどのように関連付けするかという研究が行われている [8][10]．例えば，Śliwerski [8] 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バグ ID を正規表現で検出し，バグ管理システムに登録されているバグとの関連付けを行っている．

また，Zhou ら [10] は，Bugzilla などのバグ管理システムにバグ報告が行われた際に，その報告に記載されたキーワードから修正対象である可能性の高いソースコードファイルを自動的に検出する手法を提案している．4 つのオープンソースプロジェクトを対象に評価実験を行った結果，提案手法は効率的に修正対象のソースコードファイルを検出できたと報告している．結果の一つとして，Eclipse プロジェクトの約 3,000 件の障害報告それぞれに対して，提案手法によって提示される修正候補ファイルの上位 10 件を調べたところ，62%の精度で実際に修正されたファイルが含まれていたことがわかった．

従来研究では，バグ管理システムとバージョン管理システムの関連付けが主に行われている一方で，本稿では，クラッシュリポジトリからバージョン管理システムへの関連づけを行った点が新しい．また，どの程度の割合で関連付けられるかをデータを用いて分析した点も本稿の大きな貢献の 1 つである．

## 5. おわりに

クラッシュレポートからソースコードの不具合箇所を自動的に特定するために，本稿ではその前段階として，クラッシュレポートからソースコードの修正箇所をどのように関連付けられるか分析した．Firefox プロジェクトで 2 年間に蓄積された，約 320,000,000 件のクラッシュレポート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．

- クラッシュタイプのバグ含有率，及び，リンク率はわずかであり，全バージョンの平均でそれぞれ 2.20%と 0.55%であった．
- クラッシュレポートのバグ含有率，及び，リンク率は，全バージョンの平均でそれぞれ 62.26%と 15.40%であった．
- クラッシュレポートのリンク率は，従来研究と比較した場合，今後の予測研究にとって一定の割合であるものと考える．

今後の課題として，トップクラッシュを対象とした分析，及び，バグ管理システムのデータを併用することが挙げられる．また，今回のデータを用いて，ソースコードの不具合箇所を予測するための特徴を調べることも今後の重要な課題である．

**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）による助成を受けた．

*5 文字列の類似度を測るための距離

### 参考文献

[1] 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).

[2] 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).

[3] 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).

[4] 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).

[5] 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'12)*, pp. 179–188 (2012).

[6] 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).

[7] Shihab, E., Ihara, A., Kamei, Y., Ibrahim, W., Ohira, M., Adams, B., Hassan, A. E. and Matsumoto, K.: Studying re-opened bugs in open source software, *Empirical Software Engineering*, pp. 1–38 (2012).

[8] 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).

[9] Socorro: `http://socorro.readthedocs.org/`.

[10] 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).