A Bug-Fix Metarepository for Developers and Researchers

Mathieu Nayrolles
Software Behaviour Analysis Research Lab
ECE, Concordia University,
Montréal, Canada
mathieu.nayrolles@concordia.ca

Abdelwahab Hamou-Lhadj
Software Behaviour Analysis Research Lab
ECE, Concordia University,
Montréal, Canada
wahab.hamou-lhadj@concordia.ca

ABSTRACT

In recent years, mining bug reports (BR) and their related fixes has perhaps been one of the most active software engineering research fields. There exists many open source and proprietary bug tracking and source code versioning systems that developers and researchers can use to examine bug reports so as to reason about software quality. The issue is that these repositories use different interfaces and ways to access and represent data, which hinders productivity and reuse. To address this, we introduce a large dataset of 700,000 fixed bugs belonging to 1,900 different projects. This dataset follows a clear infrastructure and allows developers and researchers interested in mining data from different (and heterogeneous) repositories to do so easily.

Keywords

Bug tracking systems; Source code versioning systems; Bug Reports; Mining software repositories

CCS Concepts

•**Software and its engineering** → *Software libraries and repositories;*

1. INTRODUCTION

Bug tracking systems such as Bugzilla and Jira have grown to contain hundreds of thousands of bugs, providing a vast amount of data to several active research fields including bug reproduction, bug triaging, and empirical studies. The analysis of BRs can provide useful insight that can help with many maintenance tasks such as bug fixing [20, 3, 19], bug reproduction [2, 4, 8, 18], bug prediction [5, 10, 12], and fault analysis [6].

Today, there exist many bug tracking and source code versioning tools (e.g., Bugzilla, Jira, SVN, Git, etc.) that can be used by practitioners and researchers to conduct large-scale studies on the causes and distribution of bugs in software systems. The problem is that these systems have different interfaces to access data. The data is not represented in a uniform way either. This is further complicated by the fact that bug tracking tools and code versioning systems are not necessarily connected. The former follows the life of the bug itself and not its fixes, which are managed by the latter. Analyzing the bugs and their fixes from different sources require going back and forth between diverse tools, creating parsers, mapping data from a repository to another, etc. These tasks are not only time consuming but add no value to the analysis itself.

For the time being, our dataset, which is available for download at https://bumper-app.com/msr16, aggregates bug reports and fixes from Eclipse[1], Gnome[2], the Apache Software fundation[3] and Netbeans[4]. Moreover, our dataset associates each bug report (bug description, reporter, assignee, comments, ...) to its related fixes. These systems use Bugzilla[5] or Jira[6] as their bug report system and Git[7] or Mercurial[8] as their source code versioning engine.

2. DATA COLLECTION

Figure 1 illustrates how we extracted data from various bug report tracking and code versioning systems. In this example, we extract raw data from two bug tracking systems, Bugzilla and Jira with their corresponding source code versioning systems, Git and Mercurial. The extracted data is consolidated into one database where we associate each bug report with its fix. Note that this association is based on the general practice where developers create a link between the bug tracking system and their source versioning tool by either writing the bug #ID in their commit message or adding a link towards the changeset as a comment in the bug report system. Algorithms for linking these two entities in the absence of explicit linkage can also be used such as the one presented by Wu et al. [21] in their tool, Relink.

Currently, our dataset includes five bug report management systems, namely, Gnome, Eclipse, Netbeans and the Apache Software Foundation that are composed of 512, 190, 39 and 349 projects respectively, bringing the total of projects to 1,930. These projects cover 16 years of development from 1999 to 2015. Gnome is a free desktop environment,

[1] https://www.eclipse.org/
[2] https://www.gnome.org/
[3] http://www.apache.org/
[4] https://netbeans.org/
[5] https://www.bugzilla.org/
[6] https://www.atlassian.com/software/jira
[7] https://git-scm.com/
[8] https://www.mercurial-scm.org/

MSR '16 May 14–15, 2016. Austin, USA
© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

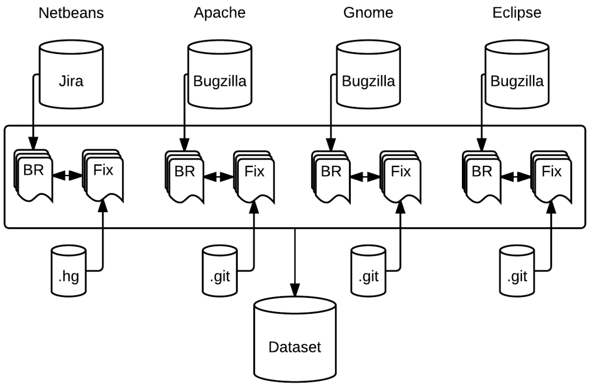

Figure 1: BUMPER Data Collection

Table 1: RESOLVED/FIXED BUG (R/F BR), CHANGESETS (CS), AND PROJECTS BY DATASET

Dataset	R/F BR	CS	Files	Projects
Gnome	550,869	1,231,354	367,245	512
Netbeans	53,258	122,632	30,595	39
Apache	49,449	106,366	38,111	349
Eclipse	78,830	184,900	21,712	190
Total	732,406	1,645,252	457,663	1,930

mainly developed in C and C++. Eclipse and Netbeans are integrated development environments (IDEs) for developing with many programming languages, including Java, PHP, and C/C++. Finally, the Apache Software Foundation (ASF) is a non-profit public charity established in 1999, that provides services and support for many like-minded software project communities of individuals who choose to join the ASF. The extracted data is consolidated in one database where we associate each bug report with its fix. The fixes are mined from different types of version control systems. Gnome, Eclipse and Apache Software Foundation projects are based on Git (or have git-based mirrors), whereas Netbeans uses Mercurial. The characteristics of the five aggregated datasets are presented in Table 1.

We chose to use these systems to have data coming from diverse code versioning and bug tracking systems. Also, Gnome, Eclipse, Netbeans and the Apache Software Foundation exhibit a great deal of diversity in terms of the programming languages used to build applications, development teams, location of the development teams, utility, and maturity. Moreover, they use different tools, Bugzilla, JIRA, Git, and Mercurial. This said, we can and plan to integrate other datasets that use any combination of these tools.

As we can see from Table 1, our consolidated dataset contains 732,406 bugs, 1,645,252 changesets, 457,663 files that have been modified to fix the bugs and 1,930 distinct software projects belonging to four major organizations. We also collected more than two billions lines of code impacted by the changesets, identified tens of thousands of sub-projects and unique contributors to these bug report systems.

3. DATASET DESCRIPTION

Figure 2 shows the core BUMPER metamodel, which captures the common data elements used by most existing bug tracking and control version systems. An issue (task) is characterized by a date, title, description, and a fixing time.

Issues are reported (created) by and assigned to users. Also, issues belong to a project that is in a repository and might be composed of subprojects. Users can modify an issue during life cycle events which impact the type, the resolution, the platform, the OS and the status. Issues are resolved (implemented) by changeset that are composed of hunks. Hunks contain the actual changes to a file at a given revision, which are versions of the file entity that belongs to a project.

Our dataset contains information found in many common bug reporting and source code versioning systems. More particularly, it revolves around three main entities (1) Bug reports, (2) Changesets, and (3) Hunks. Bug reports can contain zero or many changesets—some bug reports such as duplicates do not require changesets. A changeset may contain one or many hunks. We carefully examined a large set of tools to define the characteristics of each entity so as to provide support for various types of analyses. A bug report is characterized by the following features:

- **ID**: A unique identifier
- **Dataset**: The dataset where the bug is extracted from
- **Date**: The bug submission date
- **Title**: The title of the bug report
- **Description**: The description of the bug
- **Project**: The project affected by this bug
- **Sub_project**: The sub-project that this bug affects.
- **Version**: the version of the project that this bug affects
- **Impacted_platform**: The platform that this bug affects
- **Impacted_OS**: The operating system that this bug affects
- **Bug_status**: The status of the bug
- **Resolution**: A description on how the bug was resolved
- **Reporter_pseudo**: The pseudonym of the person who report the bug
- **Reporter_name**: The name of the person who reported the bug
- **Assigned_to_pseudo**: The pseudonym of the person who has been assigned to fix the bug
- **Assigned_to_name**: The name of the person who has been assigned to fix the bug
- **Bug_severity**: The severity of the bug
- **Fixing_time**: The time in minutes it took to fix the bug
- **Comment_nb**: The number of comments posted on the bug report system for that bug

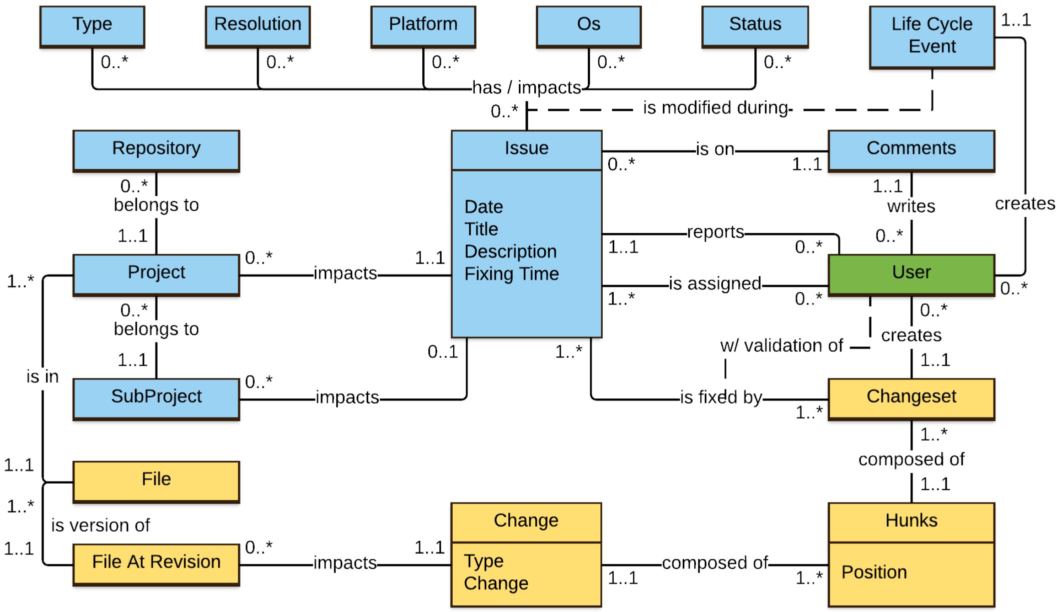

Figure 2: Metamodel

- **Comment**: The comments associated with the bug.
- **File**: The name of the source code file that have been modified to fix a bug.

A changeset is represented using the following features:

- **ID**: The unique identifier, represented as an SHA-1 hash
- **User**: The name and email of the person who submitted the commit
- **Date**: The date at which this commit has been fixed
- **Summary**: The commit message entered by the user
- **File**: The fully qualified name of a file modified on that commit. A changeset can have one or many files.
- **Number_files**: The number of files that have been modified in that commit
- **Insertions**: The number of inserted lines
- **Deletions**: The number of deleted lines
- **Churns**: The number of modified lines
- **Hunks**: The number of consecutive lines that have been changed
- **Parent_bug**: The id of the bug this changeset belongs to.

4. HOW TO USE THIS DATASET

In this section, we present how developers and researchers have and could use our dataset.

4.1 Finding bug-fix

BUMPER [16, 17][9] is a search engine for bug-fix in the same way as Google is a search engine for websites. BUMPER uses Apache Solr [15] to provide natural language searches of bug reports and fixes. This can help developers, engineers and computer science students not only to find a patch to a given bug but also discover how experienced open-source developers write fixes.

4.2 Studying bug-fix relationships and patterns

We see our dataset as a way to facilitating research in the area of mining bug repositories and more particularly, bug-fix relationships and patterns [11, 19, 14]. Studying software repositories to gain insights into the quality of the code is a common practice. However, this task requires time and skills in order to download and link all the pieces of information needed for adequate mining. Moreover, our dataset is available in JSON[10], CSV[11] and XML[12] so one can choose his favorite format to work with.

4.3 Comparing Approaches

In bug reproduction [4, 9, 18], prediction [10, 3], triaging [1, 7, 13] and other related fields of software maintenance and evolution, comparing approaches is often taxing as datasets might not be publicly available and the way data are gathered can be unclear. With our dataset, approaches could be more easily compared to each other.

In short, our dataset that can be used by software practitioners and researchers to analyse (efficiently) bugs and their fixes without having to go from one repository to another, worry about the way data is represented and saved, or create tools for parsing and retrieving various data attributes. We

[9] https://bumper-app.com
[10] https://bumper-app.com/msr16.json
[11] https://bumper-app.com/msr16.csv
[12] https://bumper-app.com/msr16.xml

hope that the community embraces this dataset and evolve it with even more bug and projects.

5. CONCLUSION

In this paper, we presented a dataset that we built to offer developers and researchers the ability to access various bug-related data from diverse repositories in a single dataset. This paper contributes also with a large dataset with 732,406 bugs, 1,645,252 changesets, 457,663 files that have been modified to fix the bugs and 1,930 distinct project related to Netbeans, the Apache Software foundation's software, Eclipse and Gnome. The dataset is publicly available at https://bumper-app.com/msr16.

As future work, we want to improve our dataset by adding more projects such as projects hosted on Github[13] and the Mozilla Foundation[14] datasets. Also, we intend to add other features such as the number of times a bug is reopened, the number of times a bug has been duplicated, etc.

6. REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceeding of the 28th international conference on Software engineering - ICSE '06*, page 361, New York, New York, USA, may 2006. ACM Press.

[2] S. Artzi, S. Kim, and M. D. Ernst. Recrash: Making software failures reproducible by preserving object states. In *Proceedings of the 22nd European Conference on Object-Oriented Programming*, pages 542–565, 2008.

[3] P. Bhattacharya and I. Neamtiu. Bug-fix time prediction models. In *Proceeding of the 8th working conference on Mining software repositories - MSR '11*, page 207, New York, New York, USA, may 2011. ACM Press.

[4] N. Chen. *Star: stack trace based automatic crash reproduction.* PhD thesis, The Hong Kong University of Science and Technology, 2013.

[5] M. D'Ambros, M. Lanza, and R. Robbes. An extensive comparison of bug prediction approaches. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 31–41. IEEE, may 2010.

[6] M. Hamill and K. Goseva-Popstojanova. Exploring fault types, detection activities, and failure severity in an evolving safety-critical software system. *Software Quality Journal*, 23(2):229–265, apr 2014.

[7] G. Jeong, S. Kim, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, page 111, New York, New York, USA, aug 2009. ACM Press.

[8] W. Jin and A. Orso. BugRedux: Reproducing field failures for in-house debugging. In *Proceedings of the 34th International Conference on Software Engineering, IEEE*, pages 474–484. Ieee, jun 2012.

[9] W. Jin and A. Orso. F3: fault localization for field failures. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, pages 213–223, New York, New York, USA, 2013. ACM Press.

[10] Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan. Revisiting common bug prediction findings using effort-aware models. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10. IEEE, sep 2010.

[11] D. Kim, J. Nam, J. Song, and S. Kim. Automatic patch generation learned from human-written patches. pages 802–811, may 2013.

[12] S. Kim, H. Zhang, R. Wu, and L. Gong. Dealing with noise in defect prediction. *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, page 481, 2011.

[13] H. Kong. CosTriage: A Cost-Aware Triage Algorithm for Bug Reporting Systems. In *Proceeding of the Association for the Advancement of Artificial Intelligence*, 2011.

[14] M. Martinez, W. Weimer, and M. Monperrus. Do the fix ingredients already exist? an empirical inquiry into the redundancy assumptions of program repair approaches. In *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, pages 492–495, New York, New York, USA, may 2014. ACM Press.

[15] M. Nayrolles. *Mastering Apache Solr - A Practical Guide to Get to Grips with Apache Solr.* 2014.

[16] M. Nayrolles and A. Hamou-Lhadj. BUMPER: Bug Metarepository Search Engine for Developers and Researchers. In *Consortium for Software Engineering Research*, 2015.

[17] M. Nayrolles and A. Hamou-Lhadj. BUMPER: A Tool for Coping with Natural Language Searches of Millions of Bugs and Fixes. In *23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering*, 2016.

[18] M. Nayrolles, A. Hamou-Lhadj, T. Sofiene, and A. Larsson. JCHARMING : A Bug Reproduction Approach Using Crash Traces and Directed Model Checking. In *Proceedings of the 22nd International Conference on Software Analysis, Evolution, and Reengineering, IEEE, 2015*, pages 101–110, 2015.

[19] R. K. Saha, S. Khurshid, and D. E. Perry. An empirical study of long lived bugs. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 144–153. IEEE, feb 2014.

[20] C. Weiß, T. Zimmermann, and A. Zeller. How Long will it Take to Fix This Bug ? In *Fourth International Workshop on Mining Software Repositories (MSR'07)*, number 2, page 1, 2007.

[21] R. Wu, H. Zhang, S. Kim, and S. Cheung. Relink: recovering links between bugs and changes. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering.*, pages 15–25, 2011.

[13] https://github.com

[14] https://bugzilla.mozilla.org/