BUMPER: A Tool for Coping with Natural Language Searches of Millions of Bugs and Fixes

Mathieu Nayrolles,
Software Behaviour Analysis (SBA) Research Lab
ECE, Concordia University
Montreal, Canada
m_nayrol@ece.concordia.ca

Abdelwahab Hamou-Lhadj
Software Behaviour Analysis (SBA) Research Lab
ECE, Concordia University
Montreal, Canada
abdelw@ece.concordia.ca

***Abstract*—In recent years, mining bug report (BR) repositories has perhaps been one of the most active software engineering research fields. There exist many open source bug tracking and version control 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 BUMPER (BUg Metarepository for dEvelopers and Researchers), a common infrastructure for developers and researchers interested in mining data from many (heterogeneous) repositories. BUMPER is an open source web-based environment that extracts information from a variety of BR repositories and version control systems. It is equipped with a powerful search engine to help users rapidly query the repositories using a single point of access. To demonstrate the effectiveness of BUMPER, we use it to build a large dataset from a variety of repositories. The dataset contains more than one million bug reports and fixes. Both BUMPER and the dataset are publicly available at https://bumper-app.com.**

I. Introduction

Program debugging, an important software maintenance activity, is known to be challenging and labor-intensive. Studies have shown that the cost of software maintenance can reach up to 70% of the overall cost of the software development process [1].

When facing a new bug, one might want to leverage decades of open source software history to find a suitable solution. The chances are that a similar bug or crash has already been fixed somewhere in another open source project. The problem is that each open source project hosts its data in a different data repository, using different bug tracking and version control systems. Moreover, 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 version control systems are not necessarily connected. The former follows the life of the bug, while the latter manages the fixes. As a general practice, developers create a link between the bug report system and the version control tool by either writing the bug #ID in their commits message or add a link towards the changeset as a comment in the bug report system. As a result, one would have to search the version control system repository to find candidate solutions.

Moreover, developers mainly use classical search engines that index specialized sites such as StackOverflow[1]. These sites are organized in the form of question-response where a developer submits a problem and receives answers from the community. While the answers are often accurate and precise, they do not leverage the history of open source software that has been shown to provide useful insights to help with many maintenance activities such as bug fixing [2], bug reproduction [3], fault analysis [4], etc.

In this paper, we introduce BUMPER[2] (BUg Metarepository for dEvelopers and Researchers), a web-based infrastructure that can be used by software developers and researchers to access data from diverse repositories using natural language queries in a transparent manner, regardless of where the data was originally created and hosted.

The idea behind BUMPER is that it can connect to any bug tracking and version control systems and download the data into a single database. We created a common schema that represents data, stored in various bug tracking and version control systems. BUMPER uses a web-based interface to allow users to search the aggregated database by expressing queries through a single point of access. This way, users can focus on the analysis itself and not on the way the data is represented or located. BUMPER supports many features including: (1) the ability to use multiple bug tracking and control version systems, (2) the ability to search very efficiently large data repositories using both natural language and a specialized query language, (3) the mapping between the bug reports and the fixes, and (4) the ability to export the search results in Json, CSV and XML formats.

II. Bumper Components

A. Bumper architecture

Figure 1 shows the overall architecture of BUMPER. BUMPER relies on a highly scalable architecture composed of two Virtual Private Servers (VPS), hosted on a physical server.

The first server handles the web requests and runs three distinct components: Pound, Varnish, and NginX. Pound is

[1] http://stackoverflow.com/

[2] A showcase of BUMPER is available at https://bumper-app.com/showcase.html

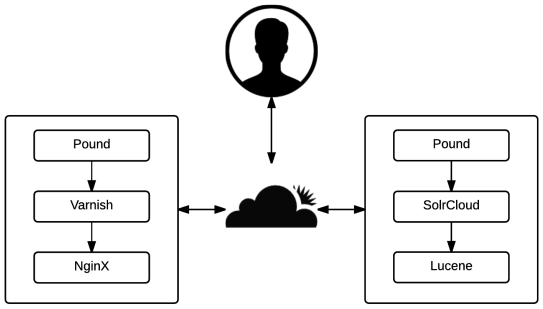

Fig. 1. Bumper Architecture

a lightweight open source reverse proxy program and application firewall. It also serves to decode https request to http. Translating an https request to http allows us to save the https decryption time required on each step. Pound also acts as a load-balancing service for the lower levels. The translated requests are then handled by Varnish. Varnish is an http accelerator designed for content-heavy and dynamic websites. It caches requests that come in and serves the Web requests from the cache if the cache is still valid. NginX (pronounced engine-x) is a web-server that was developed with a particular focus on high concurrency, high performances, and low memory usage. The second VPS also uses Pound for the same reasons and SolrCloud. SolrCloud is the scalable version of Apache Solr where the data can be separated into shards (e.g., chunks of manageable size). Each shard can be hosted on a different server, but still indexed in a central repository. Hence, we can guarantee a low query time despite the large size of the data. Finally, Lucene is the full text search engine powering Solr. This highly scalable architecture allows BUMPER to serve requests in less than a second in average.

B. BUMPER Metadata

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.

In addition, BUMPER has different indexes over the data in order to provide results in an efficient way. First of all, each feature of bug reports and bug fixes (number of files, number of hunks, etc.) have their own index. Furthermore, BUMPER has data indexes over the project name, the subproject name, the programming language, etc. Finally, two major indexes are built upon the concatenations of all the textual fields of the bug report and all the textual field (source code included) of a bug fix. They are named report_t and fix_t, respectively. These indexes are used when querying bug reports or bug fixes using natural language. Indexes can also be found in relational database management system (RDBMS). Most modern RDBMSs use binary trees to store indexes that keep data sorted and allow searches, insertions, and deletion in a logarithmic time. Nevertheless, an RDBMS can use only one index per table at the same time. This said, an RDBMS chooses only one index within the available ones–based on statistics–and uses it to complete the request. It is highly probable that thousands of records have to be scanned before the RDBMS finds the ones that match the query, especially if there is union or disjunction of records.

Unlike a traditional RDBMS, BUMPER relies on Apache Lucene and uses compressed bitsets to store indexes. Bitsets are one of the simplest—and older—data structure that contain only 0 and 1. BUMPER supports binary operations like intersection, AND, OR and XOR that can be performed in a snap even for thousands and thousands of records. As an example, if we wish to retrieve bug reports that contain the words "null pointer exception" and have a changeset containing a "try/catch", a binary intersection will be performed between the two sets of documents, which is much faster than selecting bug reports that match null pointer exception first and then checking if they have a changeset containing a try/catch as in the case of an RDBMS. This technique comes with a high overheadcompared to an RDBMS–for index update, but, in practice, information retrievals tend to be much faster. In our case, we want to provide a fast access to decades of open source historical data. We periodically update (at night) our indexes when a sufficient amount of new data has been downloaded from the bug tracking and version control systems that BUMPER supports.

C. Bumper Query Language and API

BUMPER supports two query modes: basic and advanced. The basic query insert users' inputs (YOUR TERMS) in the following query:

$$(type : \text{“}BUG\text{”}\ AND\ report\_t : \text{“}YOUR\ TERMS\text{”} \\ AND\ -churns : 0)$$

The first part of the query matches all bug reports (type:"BUG") that contain YOUR TERMS in the report_t index (report_t:"YOUR TERMS"). Finally, it excludes bug reports that do not have a fix (-churns:0). The result of this subquery will be merged with the following:

$$OR\ (\{!parent\ which = \text{“}type : BUG\text{”}\} type : \\ \text{“}CHANGESET\text{”}\ AND\ fix\_t : \text{“}YOUR\ TERMS\text{”}$$

This query selects all bugs' changeset–using a parent-child relationship ($\{!parent\ which = \text{“}type : BUG\text{”}\}\ type : \text{“}CHANGESET\text{”}$) – and that contain YOUR TERMS in the fix_t index (fix_t:"YOUR TERMS"). Finally, the results of the two previous queries will be added to the following subquery:

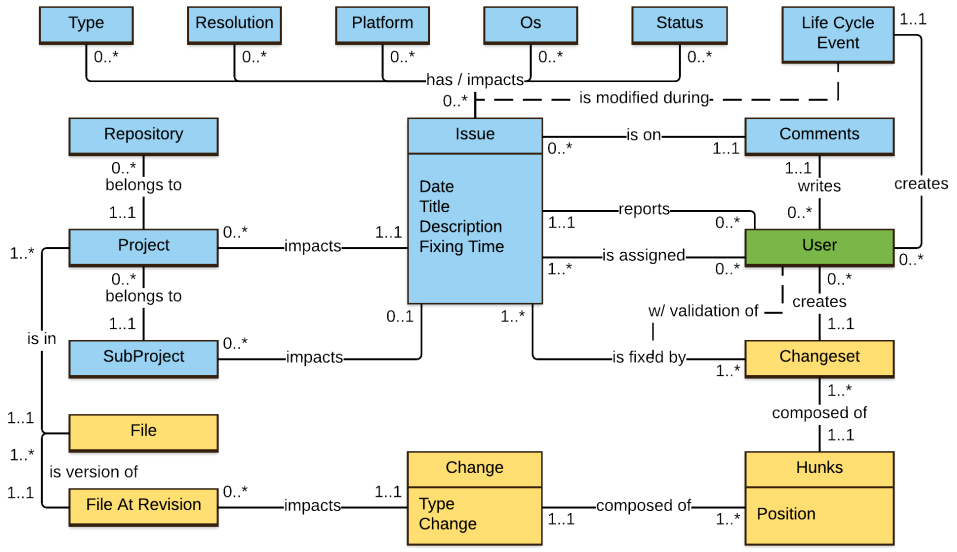

Fig. 2. BUMPER Metamodel

$$OR\ (\{!parent\ which = \text{“}type : BUG\text{”}\}$$
$$\{!parent\ which = \text{“}type : CHANGESET\text{”}\}$$
$$type : \text{“}HUNKS\text{”}\ AND\ fix\_t : \text{“}YOUR\ TERMS\text{”})$$

The query selects all the hunks that are children of changsets and grand-children of bug report $(\{!parent\ which = \text{“}type : BUG\text{”}\}\{!parent\ which = \text{“}type : CHANGESET\text{”}\}\ type : \text{“}HUNKS\text{”})$ and that contain YOUR TERMS in the fix_t index. The intent of this composed query is to search efficiently for YOUR TERMS in bug reports, commit messages and source code all together. The advanced query mode allows users to write their own queries using the indexes they want and the unions or disjunctions they need. As an example, using the advanced query mode, one could write the following:

$$(type : \text{“}BUG\text{”}\ AND\ report_t : \text{“}Exception\text{”}$$
$$AND\ (project : \text{“}Axis2\text{”}\ OR\ project : \text{“}ide\text{”})$$
$$AND\ (reporter : \text{“}Rich\text{”}\ OR\ resolution : \text{“}fixed\text{”})$$
$$AND\ (severity : \text{“}Major\text{”}\ OR\ fixing\_time : [10\ TO\ *])$$
$$AND\ -churns : 0)$$

This query finds all bug reports that contain Exception in the report_t index (first line) and belong to the Axis2 or the ide project (line 2) and have been reported by someone named Rich or have been fixed as a resolution (third line) and that have a Major severity or a fixing time greater than 10 days (fourth line) and have a fix (fifth line). More examples of the API's usage can be found at https://bumper-app.com/api.

D. Bumper Data Repository

Currently, BUMPER supports 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 supported by BUMPER to 1,930. These projects cover 16 years of development from 1999 to 2015. Gnome is a free desktop environment, 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 datasets, aggregated in BUMPER, are presented in Table I.

TABLE I
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

As we can see from the table, 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 sub-projects and unique contributors to these bug report systems.

III. Usage Scenarios

A. Bug Fixing

An example of a real-life scenario where BUMPER can be useful would be that of a developer trying to fix a bug. He

or she could copy/paste the faulty code, or the obtained error in the search bar of BUMPER. BUMPER will then return (in seconds) every bug report that contains references to the error at hand in the report or in the code developed to fix the bug in our dataset. The developer can then leverage BUMPERs search results to either find out that someone already fixed the exact same bug in another project and reuse the fix, analyse similar bugs—and their fixes— to design a solution for the problem, etc.

We conducted a preliminary study to assess the effectiveness of BUMPER to help with bug fixing tasks. We asked a group of developers to fill out an online survey[3] to see how much time it would take to fix the following made-up bug [4]:

When I run the CSVReader with a simple main like this:

Listing 1. Java Bug

```
CsvFileUtils csvFileUtils =                    1
  new CsvFileUtils("quebec.txt");              2
                                               3
for (int i = 0; i < 2000; i++) {               4
    printArray(csvFileUtils                    5
      .readOneLine());                         6
}                                              7
```

I got the following Exception: Exception in thread "main" java.lang.NullPointerException at CsvFileUtils.readOneLine(CsvFileUtils.java:22) at Main.main(Main.java:11). Can you please fix CsvFileUtils ?

The solution to this simple bug is straightforward as we only have to add a null check to Line 22 of CsvFileUtils [5].

Participants were asked to find a code snippet that can be slightly modified in order to fix a bug using BUMPER and Google. The goal of this preliminary study was to compare how fast a suitable fix can be found using BUMPER and Google. We send our survey to 20 participants and received 8 responses (40%) and asked them to report on their experience in terms of the time taken to find a fix and the number of Web pages that were browsed. Participants reported then, when using Google, it took, on average, 6.94 minutes by examining, in average, 7.57 online sources to find a fix. When using BUMPER, however, it only took around 4.5 minutes. The participants only needed to use BUMPER and not other website. The feedback we received from the participants was very encouraging. They mainly emphasized the ability for BUMPER to group many repositories in one place, making the search for similar bugs/fixes practical. We intend, in the future, to conduct a large scale (and more formal) experiment with BUMPER for a thorough evaluation of its effectiveness to help software developers in fixing bugs.

B. Studying bug-fix relationships

We see also BUMPER as an important tool for facilitating research in the area of mining bug repositories. 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 informations needed for adequate mining. BUMPER provides a straightforward interface to export bugs and their fixes into CSV, XML and JSON.

In short, BUMPER offers a framework 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 hope that the community contributes by adding more repositories to BUMPER. This way, BUMPER can become a unified environment that can facilitate bug analysis and mining tasks.

[3]http://goo.gl/forms/RvYFACkl7a

[4]https://github.com/MathieuNls/bumper-csv

[5]https://github.com/MathieuNls/bumper-csv/blob/master/src/CsvFileUtils.java

IV. Related work and conclusion

In this paper, we presented a web-based infrastructure, called BUMPER (BUg Metarepository for dEvelopers and Researchers), accessible at https://bumper-app.com. BUMPER allows natural language searches in bugs reports, commit messages and source code all together while supporting complex queries. Currently, BUMPER is populated with 1,930 projects, more than 732,406 resolved/fixed and with 1,645,252 changesets from Eclipse, Gnome, Netbeans and the Apache Software foundation. The speed of BUMPER allows developers to use it as a way to leverage decades of history scattered over hundreds of software projects in order to find existing solutions to their problems. There exist tools such as Codemine [5] and Boa [6] that enable the mining and reporting of code repositories. These tools, however, will require the download of all the data and process the mining for each installation. To the best of our knowledge, no attempt has been made towards building an unified and online datasets, from multiple sources, where all the information related to a bug, or a fix can be easily accessed by researchers and engineers in order to leverage decades of historical information. Moreover, the feedback we received from the users of BUMPER in a preliminary study shows that BUMPER holds real potential in becoming a standard search engine for bugs and fixes in multiple repositories.

References

[1] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan, 2005.

[2] 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)*. IEEE, feb 2014, pp. 144–153.

[3] M. Nayrolles, A. Hamou-Lhadj, S. Tahar, and A. Larsson, "JCHARMING: A bug reproduction approach using crash traces and directed model checking," *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 101–110, 2015.

[4] S. Nessa, M. Abedin, W. E. Wong, L. Khan, and Y. Qi, "Software Fault Localization Using N -gram Analysis," in *WASA*, 2008, pp. 548–559.

[5] J. Czerwonka, N. Nagappan, W. Schulte, and B. Murphy, "CODEMINE: Building a Software Development Data Analytics Platform at Microsoft," *IEEE Software*, vol. 30, no. 4, pp. 64–71, jul 2013.

[6] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: a language and infrastructure for analyzing ultra-large-scale software repositories," in *ICSE '13 Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, may 2013, pp. 422–431.