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 and proprietary bug tracking and source code version 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 source code versioning 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 online at https://bumper-app.com.

I. INTRODUCTION

Debugging programs, an important software maintenance activity, is challenging and labour-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 yet unknown bug, one might want to leverage decades of open-source software history in order to find a 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 insight to help with many maintenance activities such as bug fixing [2], bug reproduction [3], fault analysis [4], etc.

In this paper, we introduce BUMPER (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 were 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, generated stored in 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 support 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.

Finally, we asked developers to solve a bug using habitual habit and using BUMPER. The results indicate that our approach reduces the amount of data-sources to be visited in order to find a suitable solution for the bug at hand and therefore, speeding up the debugging and maintenance processes.

II. BUMPER COMPONENTS

Figure 1 shows BUMPERs main interface. Currently, BUMPER supports 380 projects, more than 732,000 resolved/-
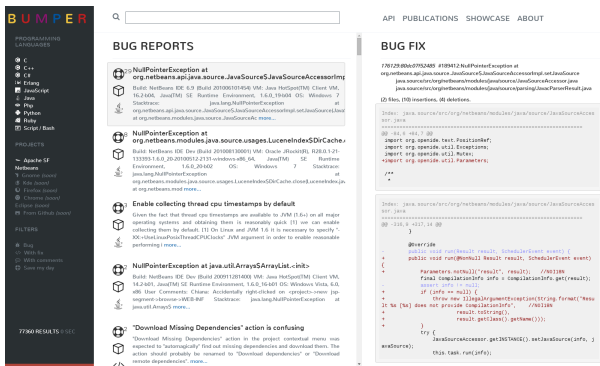
---

[1]http://stackoverflow.com/
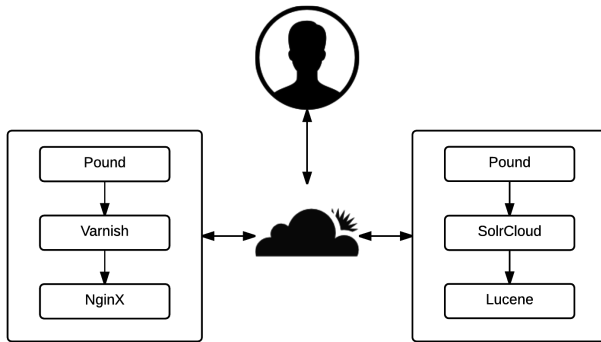
Fig. 1. Bumper Interface

Fig. 2. Bumper Architecture

fixed and with 1,645,252 changesets from Eclipse, Netbeans, the Apache Software Foundation and Gnome software systems. It can readily be extended to support other systems.

A. Bumper architecture

Figure 2 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 a lightweight open source reverse proxy program and application firewall. It also serves to decode https to 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 is 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 chunk 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 while exponentially increasing the data. Finally, Lucene is the full text search engine powering Solr. This highly scalable architecture allows Bumper to serve requests under .5 second, in average.

B. BUMPER Metadata

Figure 3 shows the core BUMPER metamodel, which captures the common data elements used by 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 sub-project 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 RDBMS uses 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 a 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 for 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 a try/catch as in the case of an RDBMS. This technique comes with a high overheadcomparing to an RDBMS–for index update, but, in practice, information retrievals are orders of magnitude faster. In our case, we want to provide a fast access to decades of open source history. We periodically update (at night) our indexes
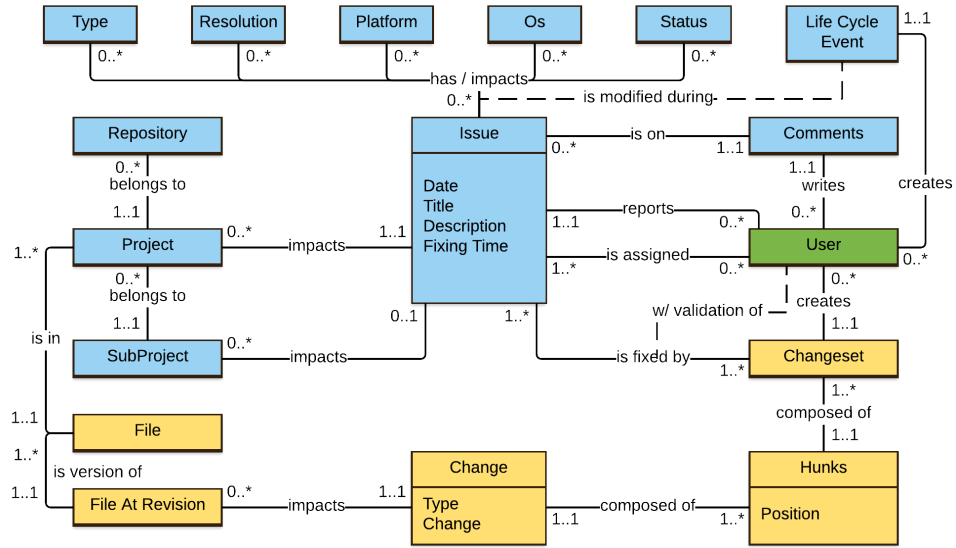
Fig. 3. BUMPER Metamodel

when sufficient amount of new data has been downloaded from the bug tracking and source versioning systems 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:

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

The query selects all the hunks that are child of changesets and grand-child 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 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 query 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

First, we extract the raw data from bug report management systems of five different open source institutions used in this study: Gnome, Eclipse, Netbeans, Apache Software Foundation and Github. Each institution is composed of many open source projects: 512, 190, 39 and 349 , respectively. Summing up to 1,930 different open source projects.

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 source versioning system. Indeed, Gnome, Eclipse and Apache Software Foundation projects are based on Git (or have git based mirrors) while Netbeans uses Mercurial.

Gnome is a free desktop environment, mainly developed in c and c++ by volunteers and paid contributors both.

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

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 characteristics of the five datasets used in this paper are presented in Table I. Cumulatively, these datasets span from 1999 to 2015.

In summary, our consolidated dataset contains 732,406 bugs, 1,645,252 changesets, 457,663 files that have been modified to fix the bugs and 1930 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.

We choose these five datasets because they expose a great diversity in terms of the programming languages used to develop the applications, the size of the development teams, localization, utility and maturity. Moreover, the used different tools, i.e. Bugzilla, JIRA, Git and Mercurial, and therefore, Bumper is ready to host any other datasets that use a composition of these tools.

Bumper users can follow the same process to add other repositories from various bug tracking and version control systems such as Eclipse, Mozilla Foundation datasets. The BUMPER architecture can then be used to organize and access multiple repositories in a unified manner.

III. USAGE SCENARIOS

A. Bug Fixing

An example of a real-life scenario where BUMPER can be useful would be that 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 the matter of 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 1,930 projects and 732,406 bugs. 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 bugsand their fixesto 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 ask a group of developers to find a suitable solution of the presented bug online as would

have in a classical debugging / maintenance session and repeat the operation using BUMPER. Finally, participants were asked to report how long it took to find a suitable solution with both methods and to rate their experience using BUMPER.

We presented participants of our online survey[2] with the following made up bug[3]: *When I run the CSVReader with a simple main like this:*

Listing 1. Java Bug
```
public static void main(String[] args)    1
  throws IOException{                       2
                                            3
  CsvFileUtils csvFileUtils =               4
    new CsvFileUtils("quebec.txt");         5
                                            6
  for (int i = 0; i < 2000; i++) {          7
                                            8
    printArray(csvFileUtils                 9
      .readOneLine());                      10
  }                                         11
}                                           12
```

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 straight forward as we only have to add a null check to line 22 of CsvFileUtils[4].

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 the experiment 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 their experience in terms of time taken to find a fix and webpages 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 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 fixes practical. We intend in the future to conduct a large scale 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 insight 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. As an example, if one wants to extract the features of a bug report that get fixed quickly (this type of analysis is performed in [11]), the following query can be used:

$$(type: “BUG” AND fixing\_time : [0TO10] \ churns : 0)$$
$$OR(type: “BUG” AND fixing\_time : [200TO*]$$
$$churns : 0)\&facet = true\&facet.field = project$$

The query returns bugs that have been fixed in less than 10 days and bugs that have been fixed in more than 200 days. Moreover, the $\&facet = true\&facet.field = project$ part of the query provides a count of each category (e.g., less than 10 days and more than 200 days) by projects. Finally, researchers can press the XML, CSV or JSON button, shown in Figure 1. to export their data in the desired format. 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.

IV. RELATED WORK

Codemine [5] and Boa [6] are two well-known projects that enable the mining and reporting of code repository, including bug report and related fixes. Both 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. Providing a unified, online, and easily accessible datasets have been shown to be effective for learning new APIs [7], [8] and BUMPER aims to achieve a similar goal, which is empowering developers by giving access to multiple data repositories that they can use to find how similar bugs have been fixed throughout the history of hundreds of well known software projects.

V. CONCLUSION

In this paper, we presented an online tool named 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 and contain 1,930 projects, more than 732,406 resolved/fixed and with 1,645,252 changesets that were involved in fixing them from Eclipse, Gnome, Netbeans and The Apache Software foundations software.

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. Furthermore, users found a solution to a simple bug 40% faster using Bumper than Google.

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] N. Chen, "Star: stack trace based automatic crash reproduction," Ph.D. dissertation, 2013.

[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] C. Bizer, T. Heath, and T. Berners-Lee, *Linked data-the story so far. Semantic Services, Interoperability and Web Applications: Emerging Concepts*. IGI Global, 2011.

[7] J. E. Montandon, H. Borges, D. Felix, and M. T. Valente, "Documenting APIs with examples: Lessons learned with the APIMiner platform," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, oct 2013, pp. 401–408.

[8] M. M. Rahman, S. Yeasmin, and C. K. Roy, "An IDE-based context-aware meta search engine," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, oct 2013, pp. 467–471.