A Dataset of Bug Patterns in JavaScript

Quinn Hanam
University of British Columbia
Vancouver, BC, Canada
qhanam@ece.ubc.ca

Fernando S. de M. Brito
Univ. Federal da Paraíba
João Pessoa, PB, Brazil
fernando@lavid.ufpb.br

Ali Mesbah
University of British Columbia
Vancouver, BC, Canada
amesbah@ece.ubc.ca

This artifact [2] provides the dataset for our FSE'16 paper *"Discovering Bug Patterns in JavaScript"* [3]. Its purpose is to enable replication of the evaluation and empirical study, and to allow users to explore some of the bug patterns that exist in JavaScript in greater detail than could be presented in the paper. It has multiple components including: an executable for reproducing or expanding the dataset, the list of subjects, the raw data, the list of basic change types (BCTs) and the list of change types. In addition to the raw data and executable, for each component we provide a graphical web interface for exploring the data.

1. EXECUTABLE AND SUBJECTS

We provide an executable and a list of git repositories for reproducing or expanding the results of our empirical study. The executable has two main classes: the first builds the raw dataset of $\langle commit, BCT \rangle$ relations by mining the repository histories; the second filters the data and clusters the commits into change types. Installation and useage instructions are available in the executable's README [1].

The list of git repositories is the same as what we used in our empirical study. Detailed information about these repositories is available on the artifact's web page [2].

2. RAW DATA

Our commit mining process created $\langle commit, BCT \rangle$ relations for 105,133 commits. This dataset is available as a CSV file which is downloadable from the artifact's web page. Each row contains the commit ID, the number of modified statements in the commit and a list of BCTs that are present in the commit. It can be used in lieu of regenerating the dataset with the executable or to perform alternate data mining tasks.

3. BASIC CHANGE TYPES

A BCT is the smallest unit of change in our method. The list of 582 unique BCTs in the raw data and the number of occurrences of each BCT is available through a searchable interface on the artifact's web page. This dataset is useful for looking up the relative frequency of BCTs. For example, the most frequently inserted call to a JavaScript API method is *replace*, which was inserted 269 times in our subjects. By contrast, the builtin method *eval* was only inserted 11 times.

4. CHANGE TYPES

Change types are groups of commits which share a similar set of BCTs and number of modified statements. The list of 219 change types discovered in our empirical study is available as an `.arff` file and through a searchable interface on the artifact's web page.

This dataset is useful for exploring bug patterns that exist for JavaScript. In our empirical study, we highlighted 12 bug patterns that we believe are good candidates for detection using static analysis tools. There are many more patterns that either occured less frequently in our dataset or that require project-specific or API-specific knowledge to diagnose. Because there are alternate applications for which these patterns may be relevant, we provide an interface for exploring change types in the same manner that we used in our empirical study.

We suggest the following method for investigating a particular class of bug:

1. Identify which BCTs might be included with the class of bug under investigation.
2. Use the interface on the artifact's web page to find change types that contain these BCTs.
3. For each change type, use the interface to inspect the commits inside the change type.

For example, assume we are interested in bugs associated with using callbacks. First, we identify relevant BCTs. In this case they are BCTs that change a `callback` convention token, such as the BCT *CV_MC_I_callback*. Next, we search for clusters that contain this BCT. Cluster #3 includes this BCT. Finally, we inspect the commits in cluster #3 by opening the commit summaries on GitHub using the provided links. We observe that in general, the commits in cluster #3 repair an error caused by unchecked state by returning control through a callback function.

5. EVALUATION DATA

The raw data used in the evaluation – both $\langle commit, BCT \rangle$ relations and change types – is available as a download from the artifact's web page. It can be used to replicate the evaluation.

6. REFERENCES

[1] BugAID: Source Code. https://github.com/saltlab/bugaid, 2016.

[2] BugAID: Toolset and Dataset. http://salt.ece.ubc.ca/software/bugaid/, 2016.

[3] Q. Hanam, F. S. d. M. Brito, and A. Mesbah. Discovering bug patterns in JavaScript. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016.

Artifact Scorecard

A Dataset of Bug Patterns in JavaScript

Quinn Hanam
University of British Columbia
Vancouver, BC, Canada
qhanam@ece.ubc.ca

Fernando S. de M. Brito
Univ. Federal da Paraíba
João Pessoa, PB, Brazil
fernando@lavid.ufpb.br

Ali Mesbah
University of British Columbia
Vancouver, BC, Canada
amesbah@ece.ubc.ca

1. INSIGHTFULNESS

How timely is the artifact? The analysis of JavaScript is currently a very active area in research and in practice. JavaScript is increasingly being used as a sever side language because of Node.js. The JavaScript specification is also currently undergoing a series of major updates after a long period of stagnation. Because of these ongoing changes to JavaScript, insight is needed to guide language development, tooling and use. We believe that the hard evidence of frequently occurring bug patterns that this artifact provides will be immediately useful.

How does it increase knowledge? The main purpose of our artifact is to allow researchers to explore common bug patterns that occur in JavaScript. We hope that this helps researchers understand the most problematic bug patterns by giving concrete examples.

We also believe that researchers will find our artifact useful for understanding some of the 'naturalness' of bug repairs. While change patterns have been studied in the past, to our knowledge this is the largest and first automatically generated dataset of commits that fix specific bugs.

2. USEFULNESS

Would it otherwise be difficult to reproduce? Our artifact provides a dataset that would require significant effort to reproduce. It would require re-implementing our tool, which can be done given the description of our method in the TR-track paper, but would be time consuming from a software engineering perspective. Significant effort went in to designing and testing our tool such that it can be easily extended to support additional languages and an expanded language model for JavaScript.

Our artifact also provides a web interface for inspecting results that is similar to the one we used. Inspecting change types without the aid of an automated script is particularily time consuming because for a given change type, details about the change type need to be aggregated and a description of each commit in the change type needs to be produced.

3. USEABILITY

Is it *"[e]asy to understand?"* Reading the TR-track paper is likely required for understanding what the artifact provides in order to use it effectively. However, we have split up and described each component of the artifact such that given an understanding of the TR-track paper it should be straightfoward to understand what each component provides, what it might be used for and how to use it.

Is it *"[a]ccompanied by tutorial notes?"* Providing the artifact is accepted to the artifact track, the pdf file which accompanies this scorecard will be appended to the TR-track paper. This accompanying document describes the artifact's components (executable, datasets and interfaces) and how to use them. Besides this, the artifact's web page provides a stand-alone description of each component.

3.1 Executable Component

Is it *"[e]asy to download, install, or execute?"* Our executable is available on GitHub and can be built and executed using Maven.

Is it *"[a]vailable in a virtual machine image?"* No. However, our executable runs on the JVM, so it should be portable to most platforms.

"Is the executable available online? Yes[1].

Is it *"[s]upported by configuration management tools (that) permit easy updates?"* The tool can be updated with git and Maven.

4. LICENSE

[1]https://github.com/saltlab/bugaid