Artifact for Improving Fault Localization by Integrating Value and Predicate Based Causal Inference Techniques

Yiğit Küçük,
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, OH, USA
yxk368@case.edu

Tim A. D. Henderson,
Google Inc.
Mountain View, CA, USA
tadh@google.com

Andy Podgurski
Department of Computer and Data Sciences
Case Western Reserve University
Cleveland, OH, USA
podgurski@case.edu

***Abstract*—This work presents an overview of the artifact for the paper titled "Improving Fault Localization by Integrating Value and Predicate Based Causal Inference Techniques". The artifact was implemented in a virtual machine and includes the scripts for the UniVal algorithm for fault localization employing the Defects4J test suite. Technical information about the individual components for the artifact's repository as well as guidance on the necessary documentation for utilizing the software are provided.**

I. Introduction

This work introduces the artifact for our paper titled "Improving Fault Localization by Integrating Value and Predicate Based Causal Inference Techniques" [1]. The artifact facilitates the use of the *UniVal* algorithm for localizing software faults using the Defects4J (v2.0) test suite [2].
The artifact consists of the following:

- Scripts to run the *UniVal* fault localization technique.
- The implementation of our prototype tools detailed in our paper: *Predicate_Transformer* and *GSA_Gen*.
- Our implementations of the competing fault localization techniques included in our paper: *NUMFL-DLN/PLY [3], Elastic Predicates (ESP) [4], Baah2010 [5], Ochiai [6], D-Star [7]* and *Predicate Switching [8]*.
- Scripts for possible extensions which can be used to include additional automatically generated tests.

We designed the artifact to be an *Oracle VirtualBox* virtual machine image that can be downloaded and imported as an appliance using the VirtualBox software [9]. The operating system used for the virtual machine image is Ubuntu 20.04 LTS.

Users need no additional technological skills in order to set the virtual machine up and use it. Therefore, there are no requirements or software dependencies necessary to be able to run the artifact except the minimum requirements for the VirtualBox software [9]. Running the complete process of our algorithm is as simple as a command to a bash script (e.g. ∼UniVal Closure 62). The materials we make available in our artifact can be reused by researchers and practitioners in various ways – either to access the code for UniVal from our study on different subject programs and test suites or to augment the current code in the study and improve their own research and technologies.

The entire artifact is stored in a permanent Zenodo repository [10]. In addition to the virtual machine image, we included the following documents for guidance in the repository:

- Install.pdf - Explains how to download and set up the artifact virtual machine image from scratch.
- Readme.pdf - Comprehensive guide about all the scripts and directories in the virtual machine image.
- Paper.pdf - Copy of our accepted paper [1].

II. Technical Information

In this artifact, we assumed no additional programming or technology skills. Therefore, all the scripts are ready-to-use. After the installation, users of the artifact are recommended to assign at least 8 GB of RAM and 4 CPU Cores to the virtual machine for a smooth experience.

We used Java, R and Bash scripts to orchestrate the fault localization process. Our main script is named *UniVal.sh* which takes as arguments 1) a Defects4J project name, and 2) the project version [10] and outputs a suspiciousness list of program variables (*UniVal* algorithm transforms control statement predicates to variables) to be used in the fault localization process. Defects4J [2] consists of Java programs, therefore we implemented our instrumentation library in Java. Defects4J programs also use various Java versions, and consequently, we adapted the instrumentation script to work with the Java versions we encountered in our experiments (Java 5-8).

Our prototype tools both use Java 8 in their implementations. *Predicate_Transformer* changes the source code in place for predicates in control statements, mapping them to variables introduced by the tool. For *GSA_Gen*, we used Antlr (v4) [11] and overloaded parser methods to guide the instrumentation according to the *Gated Static Single Assignment* (GSA) form [12]. In addition to the instrumented classes for data profiling,

GSA_Gen also outputs the causal map that is later used for our causal inference based technique.

For implementations of the fault localization metrics and modeling and predicting counterfactual outcomes, we used R scripts. R is a powerful programming language for statistical operations and to have broader applicability of our technique, it was important to have a modular approach in designing the artifact. The resulting files are named after each metric compared, and they consist of a list of the program variables, and their suspiciousness scores.

Although we used Defects4J suite in our experiments for our paper, our tools are applicable to a wider spectrum of Java subject programs. The modular integration we have with instrumentation and the metrics calculations can be used for most Java programs with minor changes to the script such as paths to certain files.

In our paper, we did not include *String* type variables or additional automatically generated tests [2] in addition to the developer tests in our comparison and experiments due to the constraints of the techniques compared with *UniVal* [1]. However, since we support these aspects in *UniVal*, we decided to include the scripts in the virtual machine image. We modified a test generation script from the Defects4J repository to guide the additional tests in a directory and introduced a script for combining different test generation sources [2]. We created two optional parameters in the main Bash script of *UniVal* algorithm. We expect to extend this work using these aspects of comparison in a future study.

ACKNOWLEDGEMENT

This work was partially supported by NSF award CCF-1525178 to Case Western Reserve University. The authors would also like to thank Zhoufu Bai for providing scripts we used for including NUMFL into our evaluation.

REFERENCES

[1] Y. Kucuk, T. A. Henderson, and A. Podgurski, "Improving fault localization by integrating value and predicate based causal inference techniques," *arXiv preprint arXiv:2102.06292*, 2021.

[2] R. Just, D. Jalali, and M. D. Ernst, "Defects4j: A database of existing faults to enable controlled testing studies for java programs," in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. ACM, 2014, pp. 437–440.

[3] Z. Bai, G. Shu, and A. Podgurski, "Causal inference based fault localization for numerical software with numfl," *Software Testing, Verification and Reliability*, vol. 27, no. 6, p. e1613, 2017.

[4] R. Gore, P. F. Reynolds, and D. Kamensky, "Statistical debugging with elastic predicates," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2011, pp. 492–495.

[5] G. K. Baah, A. Podgurski, and M. J. Harrold, "Causal inference for statistical fault localization," in *Proceedings of the 19th international symposium on Software testing and analysis*. ACM, 2010, pp. 73–84.

[6] R. Abreu, P. Zoeteweij, and A. J. C. van Gemund, "On the accuracy of spectrum-based fault localization," in *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, 2007, pp. 89–98.

[7] W. E. Wong, V. Debroy, R. Gao, and Y. Li, "The dstar method for effective software fault localization," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 290–308, 2014.

[8] X. Zhang, N. Gupta, and R. Gupta, "Locating faults through automated predicate switching," in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 272–281.

[9] "Virtual box." [Online]. Available: https://www.virtualbox.org/

[10] Y. Kucuk, T. Henderson, and A. Podgurski, "Improving Fault Localization by Integrating Value and Predicate Based Causal Inference Techniques," Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4441439

[11] T. J. Parr and R. W. Quong, "Antlr: A predicated-ll (k) parser generator," *Software: Practice and Experience*, vol. 25, no. 7, pp. 789–810, 1995.

[12] K. J. Ottenstein, R. A. Ballance, and A. B. Maccabe, "Gated single-assignment form: dataflow interpretation for imperative languages," in *ACM SIGPLAN Symposium on Programming Language Design and Implementation*, 1990.