Interacting with Interactive Fault Localization Tools

Ferenc Horváth
hferenc@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary

Gergő Balogh
geryxyz@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary

Attila Szatmári
szatma@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary

Qusay Idrees Sarhan
sarhan@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary
Department of Computer Science, University of Duhok
Duhok, Iraq

Béla Vancsics
vancsics@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary

Árpád Beszédes
beszedes@inf.u-szeged.hu
Department of Software Engineering, University of Szeged
Szeged, Hungary

ABSTRACT

Spectrum-Based Fault Localization (SBFL) is one of the most popular genres of Fault Localization (FL) methods among researchers. One possibility to increase the practical usefulness of related tools is to involve *interactivity* between the user and the core FL algorithm. In this setting, the developer provides feedback to the fault localization algorithm while iterating through the elements suggested by the algorithm. This way, the proposed elements can be influenced in the hope to reach the faulty element earlier (we call the proposed approach Interactive Fault Localization, or iFL). With this work, we would like to propose a presentation of our recent achievements in this topic. In particular, we overview the basic approach, and the supporting tools that we implemented for the actual usage of the method in different contexts: iFL4Eclipse for Java developers using the Eclipse IDE, and CharmFL for Python developers using the PyCharm IDE. Our aim is to provide an insight into the practicalities and effectiveness of the iFL approach, while acquiring valuable feedback. In addition, with the demonstration we would like to catalyse the discussion with researchers on the topic.

CCS CONCEPTS

• **Software and its engineering** → Dynamic analysis; *Software maintenance tools*; *Integrated and visual development environments*; **Software testing and debugging**; • **Human-centered computing** → Interactive systems and tools.

KEYWORDS

spectrum-based fault localization, interactive fault localization, interactive debugging, testing, user feedback

A-TEST 2022, 17 - 18 November, 2022, Singapore

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Ferenc Horváth, Gergő Balogh, Attila Szatmári, Qusay Idrees Sarhan, Béla Vancsics, and Árpád Beszédes. 2022. Interacting with Interactive Fault Localization Tools. In *Proceedings of The 13th Workshop on Automating Test Case Design, Selection and Evaluation (A-TEST 2022).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

This work deals with *fault localization* (FL), a debugging subactivity in which the root causes of an observed failure are sought. In particular, we present a technique and several tools to aid Spectrum-Based Fault Localization (SBFL), a class of FL methods popular among researchers [17]. The benefit of SBFL is that it relies on two sets of information: detailed code coverage and test outcomes. These are typically readily available or easily obtainable in existing projects. Based on statistical information about the number of failing and passing test cases exercising different code elements of the system, elements are assigned various *suspiciousness scores* that can be used to *rank* the code elements, thus aiding the developer in the debugging activity.

There are barriers to the wider adoption of SBFL in programming practice, such as a high number of elements to investigate [12, 18], and other issues [8, 15]. A possibility to increase the practical usefulness of SBFL tools is to involve *interactivity* and hence improve one of the tool's most crucial performance properties, fault localization effectiveness [5, 14, 16].

In our approach, called Interactive Fault Localization (iFL), we involve the user's previous or acquired knowledge about the system. The developer interacts with the fault localization algorithm via the iFL ToolKit by giving feedback on the elements of the prioritized list. This way, the next proposed suspicious elements can be influenced in the hope to reach the faulty element earlier.

2 INTERACTIVE FAULT LOCALIZATION

2.1 Related Work

The developer typically has additional information about the system of which the SBFL engine is not aware. For example, Li *et al.* [10, 11] reuses the knowledge about passing parameter values in a debugging session, Hao *et al.* [4] asks for feedback about the execution trace, Gong *et al.* [3] asks only for a simple yes/no feedback for a given statement. Lei *et al.* [9] utilize test data generation techniques to produce feedback for interacting with fault localization techniques automatically. To our knowledge, however, contextual information about higher level entities (for instance, statement vs. enclosing function) has not yet been leveraged for interactive SBFL.

2.2 Our Approach

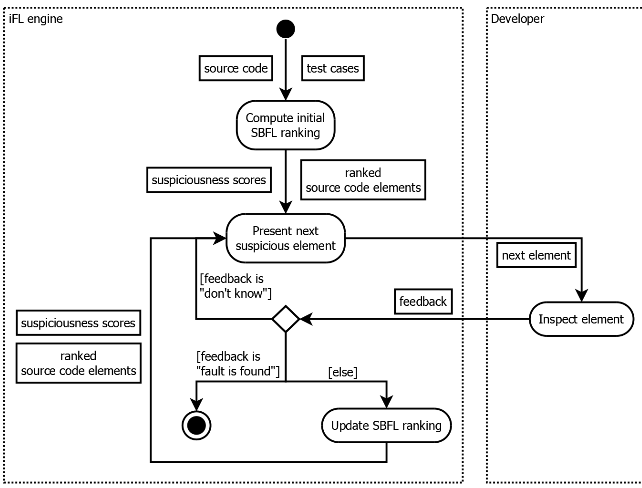

Figure 1: Basic process of Interactive Fault Localization

Figure 1 shows a conceptual overview of our approach. The process starts by calculating the initial ranking based on some traditional SBFL approach (e. g., Tarantula [7], but any other method could be used). The elements are then shown to the user starting from the beginning of the list, and the iFL engine is waiting for user feedback. In addition, the context of the elements is shown. The type of contextual data is implementation dependent. Related elements can be organized and presented based on various principles, for example, structural information, call-hierarchy, control-flow data, and others can be used. Next, the user investigates the recommended elements and their contexts, and is able to give feedback on the different groups of elements (practically the investigated element, and its context). Based on the feedback, the iFL engine performs various actions on the affected elements. It makes adjustments to the suspiciousness scores, recalculates the ranking and updates the list of elements, and the process continues with the next iteration.

3 IFL TOOLKIT

3.1 iFL4Eclipse for Java Developers

We present iFL4Eclipse [6], which is an Eclipse plug-in that supports iFL for Java projects developed in this environment. The plug-in reads the tree of project elements (classes and methods) and lists them in a view showing detailed information about those elements. This information includes, among others, the suspiciousness scores calculated using a traditional SBFL formula, such as Tarantula. This view also enables navigation to the source code elements and towards their contexts.

Interactivity between the tool and the programmer is achieved by providing the capability to send feedback to the FL engine about elements in the view. The interaction involves the *context* of the investigated element: in our case, Java classes and methods. This gives an opportunity to change the order of elements in the view and hopefully arrive at the faulty element more quickly.

iFL4Eclipse is a plug-in that supports Eclipse 2018-12 and later, and project written in Java 10 or later, and configured with Maven 3.6+, so it is part of the well-known workspace of developers. It is published via an update site and can be installed using common Eclipse functionality.

3.2 CharmFL for Python Developers

We also present CharmFL [1] [1], an Open-source fault localization tool for Python programs. The tool has many features that can help developers debugging their programs by providing a hierarchical list of ranked program elements based on their suspiciousness scores.

The front-end part of the tool, is a plug-in using the CharmFL engine for the PyCharm IDE. After installing the plug-in and opening the Python project in the IDE, the user can run the fault localization process to get the list of program suspicious elements. Additionally, the programmer may interact with the given list during debugging. For each statement the context: in our case Python methods, classes and static calls are provided to the developer. The wider range of information about the statements helps the developers to filter out parts of the list and find the faulty element more quickly.

The back-end of the framework that gathers and processes coverage and test results can be used as a stand-alone tool or integrated in other IDEs too as a plug-in. To obtain the code coverage, our tool uses the popular coverage measuring tool for Python, called `coverage.py` [2]. After collecting the coverage report, we run tests using `pytest` [13] to fetch the results. Finally, the tool calculates the suspiciousness score for each program element based on traditional SBFL methods.

4 DEMONSTRATION PLAN

During the first part of the hands-on session, we will start with a short presentation (10 minutes) that introduces the iFL approach to the participants, then we give a walk-through of all features of the aforementioned tools (approximately 25-25 minutes). The features will be presented on real open source projects to demonstrate the usefulness of the approach in actual fault-finding. For the remaining time – the second part of the session –, we prepare other examples on which the participants can try out our tools. Note that the second part could be scaled according to the final schedule of the event.

REFERENCES

[1] CharmFL 2022. CharmFL - homepage. https://sed-szeged.github.io/SpectrumBasedFaultLocalization/. (Accessed on 07/27/2022).
[2] Coverage.py 2022. Coverage.py - homepage. https://pypi.org/project/coverage/. (Accessed on 07/27/2022).

[1] Our tool paper "Interactive Fault Localization for Python with CharmFL" submitted to A-TEST(tool paper track) complements this hands-on paper

[3] Liang Gong, David Lo, Lingxiao Jiang, and Hongyu Zhang. 2012. Interactive fault localization leveraging simple user feedback. In *IEEE International Conference on Software Maintenance, ICSM*. IEEE, 67–76.

[4] Dan Hao, Lu Zhang, Tao Xie, Hong Mei, and Jia-Su Sun. 2009. Interactive Fault Localization Using Test Information. *Journal of Computer Science and Technology* 24, 5 (sep 2009), 962–974.

[5] Ferenc Horváth, Árpád Beszédes, Béla Vancsics, Gergő Balogh, László Vidács, and Tibor Gyimóthy. 2020. Experiments with Interactive Fault Localization Using Simulated and Real Users. In *Proceedings of the 36th IEEE International Conference on Software Maintenance and Evolution (ICSME'20)* (Adelaide, Australia (virtual event)). 290–300.

[6] iFL4Eclipse 2022. iFL4Eclipse - homepage. https://github.com/InteractiveFaultLocalization/iFL4Eclipse. (Accessed on 07/27/2022).

[7] James A. Jones and Mary Jean Harrold. 2005. Empirical evaluation of the tarantula automatic fault-localization technique. In *Proc. of International Conference on Automated Software Engineering* (Long Beach, CA, USA). ACM, 273–282.

[8] Pavneet Singh Kochhar, Xin Xia, David Lo, and Shanping Li. 2016. Practitioners' expectations on automated fault localization. In *Proceedings of the 25th International Symposium on Software Testing and Analysis - ISSTA 2016*. ACM Press, New York, New York, USA, 165–176.

[9] Yan Lei, Xiaoguang Mao, Ziying Dai, and Dengping Wei. 2012. Effective Fault Localization Approach Using Feedback. *IEICE Transactions on Information and Systems* E95.D, 9 (2012), 2247–2257.

[10] Xiangyu Li, Marcelo d'Amorim, and Alessandro Orso. 2016. *Iterative User-Driven Fault Localization.* Springer International Publishing, Cham, 82–98.

[11] Xiangyu Li, Shaowei Zhu, Marcelo d'Amorim, and Alessandro Orso. 2018. Enlightened Debugging. In *Proceedings of the 40th IEEE and ACM SIGSOFT International Conference on Software Engineering (ICSE 2018)* (Gothenburg, Sweden). ACM.

[12] Chris Parnin and Alessandro Orso. 2011. Are Automated Debugging Techniques Actually Helping Programmers?. In *Proceedings of the 2011 International Symposium on Software Testing and Analysis* (Toronto, Ontario, Canada). ACM, 199–209.

[13] pytest 2022. pytest - homepage. https://docs.pytest.org/. (Accessed on 07/27/2022).

[14] Qusay Idrees Sarhan and Árpád Beszédes. 2022. Effective Spectrum Based Fault Localization Using Contextual Based Importance Weight. In *Proceedings of the 15th International Conference on the Quality of Information and Communications Technology (QUATIC'22)* (Talavera de la Reina, Spain).

[15] Friedrich Steimann, Marcus Frenkel, and Rui Abreu. 2013. Threats to the Validity and Value of Empirical Assessments of the Accuracy of Coverage-based Fault Locators. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis* (Lugano, Switzerland). ACM, 314–324.

[16] Béla Vancsics, Ferenc Horváth, Attila Szatmári, and Árpád Beszédes. 2021. Call Frequency-Based Fault Localization. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 365–376. https://doi.org/10.1109/SANER50967.2021.00041

[17] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A survey on software fault localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.

[18] Xin Xia, Lingfeng Bao, David Lo, and Shanping Li. 2016. "Automated Debugging Considered Harmful" Considered Harmful: A User Study Revisiting the Usefulness of Spectra-Based Fault Localization Techniques with Professionals Using Real Bugs from Large Systems. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 267–278.