iFL for Eclipse – A Tool to Support Interactive Fault Localization in Eclipse IDE

Gergő Balogh Victor Schnepper Lacerda Ferenc Horváth Árpád Beszédes
Department of Software Engineering, University of Szeged
{geryxyz,lacerda,hferenc,beszedes}@inf.u-szeged.hu

I. Interactive Fault Localization

We present a tool to aid Spectrum-Based Fault Localization (SBFL) [1]. SBFL provides a ranked list of suspicious code elements to the user based on statistical analysis of test execution outcomes and code coverage. Recent studies highlighted some barriers to the wide adoption of SBFL, including a high number of suggested elements to investigate [2], and other issues [3]. A possibility to increase the practical usefulness of SBFL tools is to involve interactivity. In our approach, called *iFL*, we involve the user's previous knowledge about the system: the developer interacts with the fault localization algorithm by giving feedback on the elements of the prioritized list. Contextual knowledge of the user about the ranked items, classes of methods in our case, is used to reposition larger code entities in their suspiciousness, thus aiding the FL process. The benefits of developer's additional knowledge have already been explored. For example, Li *et al.* [4] reuse the knowledge about passing parameter values, while Gong *et al.* [5] ask only for a simple yes/no feedback for a given statement. To our knowledge, contextual information about higher level entities has not yet been leveraged for interactive SBFL.

II. Benefits and contributions

FL is a debugging activity in which, by definition, the programmer interacts with the source code of the software being debugged, which can be performed most effectively through the IDE itself. *iFL for Eclipse* 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 table with detailed information. This includes the suspiciousness scores calculated using a traditional SBFL formula such as Tarantula. Interactivity between the tool and the programmer is achieved by providing the possibility to send feedback to the FL engine about the table elements and their context. The user can choose from the following options: (1) the item is faulty (the process stops), (2) it is not faulty nor its context (they are moved lower in the rank), or (3) it is not faulty but the context is suspicious (they are ranked higher).

III. Technical details

iFL for Eclipse is a plug-in supporting Java 10 and later, and Eclipse 2018-12 and later. It is published via an update site (at the time of submission, it is available as a prototype but the first release will be made open source before the demonstration). The tool uses JDT to detect the methods from the source code from an Eclipse Java project. In the present phase, the scores are loaded from an external FL tool. User interaction is session-based and is tied to one active project. The main UI is an Eclipse graphical panel, serving as the front end. The purpose of the back-end is the update of scores and the recalculation of the rank list based on user input.

Current state: At the present state of our research agenda, the demonstrated tool serves our research purposes: to investigate the feasibility and effectiveness of *iFL*. As such, it is in a prototype state, not thoroughly tested and validated. The tool will be made open source, along with the results of associated experiments, in order to enable other researchers its independent validation and further development. In terms of functionality, currently it includes the basic features but there are many possibilities for further development. Our primary plans are to increase usability and flexibility in terms of user feedback actions and the underlying FL computation. Ongoing research is to perform user studies to investigate the effectiveness of the proposed *iFL* approach and the tool itself.

IV. Demonstration plan

During the live demonstration, we will give a walkthrough of all features of the tool. The main features will be first presented on a simple example, and then a more complex (open source) project will be used to demonstrate the usefulness of the approach in actual fault finding. The video showing the planned demonstration is available at:
https://youtu.be/AYkcjmSgkA0.

References

[1] W. E. Wong, R. Gao, Y. Li, R. Abreu, and F. Wotawa, "A survey on software fault localization," *IEEE Transactions on Software Engineering*, vol. 42, no. 8, pp. 707–740, 2016.
[2] C. Parnin and A. Orso, "Are automated debugging techniques actually helping programmers?" in *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, 2011, pp. 199–209.
[3] P. S. Kochhar, X. Xia, D. Lo, and S. Li, "Practitioners' expectations on automated fault localization," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 165–176.
[4] X. Li, S. Zhu, M. d'Amorim, and A. Orso, "Enlightened debugging," in *Proceedings of the 40th IEEE and ACM SIGSOFT International Conference on Software Engineering (ICSE 2018)*, 2018, pp. 82–92.
[5] L. Gong, D. Lo, L. Jiang, and H. Zhang, "Interactive fault localization leveraging simple user feedback," in *IEEE International Conference on Software Maintenance, ICSM*, 2012, pp. 67–76.

This work was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.