2017 IEEE 28th International Symposium on Software Reliability Engineering Workshops

The Future of Automated Debugging – Focus on the Niches first

Franz Wotawa
Institute of Software Technology
Graz University of Technology, Graz, Austria
wotawa@ist.tugraz.at

***Abstract*—There is a gap between research and practice of automated and algorithmic debugging. In order to close this gap, we claim to search for niches where research techniques and methods can be more easily applied and tested under real-world development conditions. In particular, we discuss the niche of spreadsheet fault localization as one very promising area and present the current state of research in applying model-based debugging for spreadsheets.**

I. Introduction

Automation has been the key driver of economy leading to increased productivity and competitiveness. In the software industry automation is currently mainly applied in areas like verification and validation but there are almost no tools available supporting or automating fault localization and repair once a failure has been detected. It is interesting to note that despite of the fact that algorithmic and automated debugging has gained a lot of interest in research over the last 35 years, the number of available at least semi-automated debugging tools that can be applied in todays software development processes and programming languages is rather limited. There are mainly extended research prototypes available, e.g., Zoltar [1], that can be more or less easily integrated into current integrated development environments.

On the other side there have been a lot of research papers on automated debugging published, presenting a variety of methods and techniques ranging from program slicing, algorithmic program debugging, model-based debugging (MBD), spectrum-based fault localization (SFL), to mutation-based and genetic-based debugging. See [2] for a recent and rather complete overview on algorithmic and automated debugging. So why is there this rather huge gap between research and application of automated debugging? This gap can be hardly explained with a limited importance or relevance of debugging support in practice. Especially when searching for faults in programs that were originally written by someone else, or where there have been some months between writing the source code and finding a failure, the debugging task is time consuming and intricate. At least in software maintenance there is a need for tools supporting debugging in practice. Therefore, there must be other reasons.

Parnin and Orso [3] stated the important question whether (current) automated debugging techniques actually help programmers. In their paper, the authors report on an experimental study they have carried out using a SFL tool. Their paper offers very interesting observations and insights, like that *expert programmers speed up debugging when using the tool*. In addition, tools help to *correct faults instead of simply patching failures*. But programmers – in order to use tools effectively – *want values, overviews and explanations*. Hence, there is hope that programmers use semi-automated debugging tools on a regular basis during development. In order to achieve this, tools have to be improved and more research is needed. Parnin and Orso mentioned some of the future research challenges, which mainly apply for SFL techniques.

Another explanation for the gap between debugging research and practice may rely in the way (debugging) research is carried out. In research, it is sufficient to show that the new technique or method improves the older ones using studies based on a limited number of well-known example cases. There is no need for considering a lot of different examples from practice. Moreover, researchers tackle usually an abstracted problem, and ignore more complex situations or specific particularities coming from considered programming language or development tools. In addition, after providing a method, technology or programming-language changes that occur when time elapses, are hardly considered any more. But in order to be successful, debugging tools and methods have to be maintained to cope with changes. Therefore, there are only a few options for bringing debugging tools intro practice. Two of them are: (i) a company adopts a certain method or technology and comes up with a industrial-grade tool, or (ii) research focuses on areas where there are less changes and where research prototypes can be more easily be integrated and used.

We claim that *there are niches for automated debugging where new debugging methods and tools can be easily integrated and used in order to further promote research in fault localization*. In the following, we briefly outline current research activities on debugging of spreadsheets purely focusing on fault localization based on models.

II. Debugging support for end-user programming

In end-user programming people, with less experience in software engineering, write programs. Spreadsheet programming is by far the most prominent example of end-user programming. Because spreadsheets are often used as basis for decision making in companies and public bodies, the

978-1-5386-2387-9/17 $31.00 © 2017 IEEE
DOI 10.1109/ISSREW.2017.16

quality of spreadsheets is of utmost importance. Hence, there have been a lot of research in the area of quality assurance for spreadsheets (see e.g. Jannach et al. [4]). Besides quality measures based on patterns, or tools for testing support, it is also important to consider fault localization support in the spreadsheet domain. Let us consider again [3] stating that often users patch failures but do not correct the corresponding faults, which would be even more important when debugging spreadsheets. There are some requirements when dealing with spreadsheet debugging tools. First, tools have to provide results ideally in an interactive manner, and second, the interaction with the tool has to be as simple as possible. Spreadsheets also come with beneficial facets like stability of their environment and underlying programming language, simplicity of the programming language paradigms, and the availability of tools for accessing spreadsheets programmatically, which makes them an ideal testbed for debugging research.

Interestingly, spreadsheets have been discovered as niche for fault localization quite late. The first publications adopting slicing were published more than 15 years ago. Jannach and Schmitz [5] are the first applying MBD for locating bugs in spreadsheets. There the idea is to map the formulae stored in cells into a constraint representation including a particular variable ab_c stating that the formula of cell c is abnormal, i.e., not correct. All the constraints have the form $ab_c \vee f_c$, where f_c the constraint representation of cell c's formula. The cell's corresponding constraints state that either the cell is incorrect or that its formula is valid and thus constraining the relation between cells and values. A constraint solver can be used to determine the values for all the ab variables. A cell where ab is assigned to be true, is a diagnosis.

The direct representation of formulae as constraints is appealing especially for spreadsheets because it is easy to obtain. Unfortunately, when dealing with constraints over numbers and large spreadsheets the original model-based approach can hardly be used interactively, because of the runtime, which likely exceeds several seconds or even minutes. Therefore, there is a need for more efficient debugging support. Hofer et al. [6] made use of MBD but instead of using a direct constraint representation, the authors introduce the use of abstract models. This kind of models only considers correct or incorrect cell values and thus decreases required computational resources. The abstract model comes up with good diagnostic accuracy and fast diagnosis runtime allowing to use the approach in an interactive way. In order to overcome the limitation of classifying values as correct or incorrect only, [7] introduced a more sophisticated model allowing to classify values as smaller, equivalent, or larger than expected. Such a qualitative deviation model comes with a similar low computational footprint and a better diagnostic accuracy. Hence, we see an increasing number of fault localization methods and techniques for spreadsheets that are very much close of being used in practice.

III. Conclusions

In this abstract, we claimed that there is a huge gap between debugging research an its use in regular practice. We discussed some reasons behind and came up with using niches for applications of debugging methods and techniques. We belief that niches having aspects like stability of underlying programming languages and environments, and simplicity of the underlying programming language paradigms are of particular interest for automated debugging. In order to support this belief, we briefly discussed the application of MBD for spreadsheet fault localization, where challenges like interactivity and ease of use seems to be solved. What is still missing are automated debugging tools to be integrated within spreadsheet environments and experimental studies to be carried out. Niches like spreadsheets might be the ideal starting point for bringing debugging research into daily practice. Furthermore, research and improvements, e.g., handling object-oriented languages, dealing with large programs, and integrated different debugging methods, are needed when going beyond niches.

Acknowledgment

The work described in this paper was funded by the Austrian Science Fund (FWF) under contract number I2144 and the Deutsche Forschungsgemeinschaft (DFG) under contract number JA 2095/4-1.

References

[1] T. Janssen, R. Abreu, and A. J. van Gemund, "Zoltar: a spectrum-based fault localization tool," in *Proc. of the 2009 ESEC/FSE Workshop on Software Integration and Evolution and Runtime (SINTER '09)* USA: ACM, 2009, pp. 23–30.

[2] 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, Aug 2016.

[3] C. Parnin and A. Orso, "Are automated debugging techniques actually helping programmers?" in *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, USA: ACM, 2011, pp. 199–209.

[4] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa, "Avoiding, finding and fixing spreadsheet errors - A survey of automated approaches for spreadsheet QA," *Journal of Systems and Software*, vol. 94, pp. 129–150, 2014.

[5] D. Jannach and T. Schmitz, "Model-based diagnosis of spreadsheet programs - A constraint-based debugging approach," *Automated Software Engineering*, pp. 1–40, 2014.

[6] B. Hofer, A. Hoefler, and F. Wotawa, "Combining models for improved fault localization in spreadsheets," *IEEE Trans. Reliability*, vol. 66, no. 1, pp. 38–53, 2017.

[7] B. Hofer, I. Nica, and F. Wotawa, "Ai for localizing faults in spreadsheets," in *Proc. of the 29th IFIP International Conference on Testing Software and Systems (ICTSS)*, St. Petersburg, Russia, October 2017.