Tool-supported fault localization in spreadsheets: Limitations of current evaluation practice

Birgit Hofer
Graz University of Technology
8010 Graz, Austria
bhofer@ist.tugraz.at

Dietmar Jannach
TU Dortmund
44221 Dortmund, Germany
dietmar.jannach@udo.edu

Thomas Schmitz
TU Dortmund
44221 Dortmund, Germany
thomas.schmitz@udo.edu

Kostyantyn Shchekotykhin
University Klagenfurt, Austria
kostya@ifit.uni-klu.ac.at

Franz Wotawa
Graz University of Technology
8010 Graz, Austria
wotawa@ist.tugraz.at

ABSTRACT

In recent years, researchers have developed a number of techniques to assist the user in locating a fault within a spreadsheet. The evaluation of these approaches is often based on spreadsheets into which artificial errors are injected. In this position paper, we summarize different shortcomings of these forms of evaluations and sketch possible remedies including the development of a publicly available spreadsheet corpus for benchmarking as well as user and field studies to assess the true value of the proposed techniques.

Categories and Subject Descriptors

H.4.1 [**Information Systems Applications**]: Office Automation—*Spreadsheets*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Debugging aids*

General Terms

Spreadsheets, Debugging, Fault Localization

1. INTRODUCTION

Locating the true causes why a given spreadsheet program does not compute the expected outcomes can be a tedious task. Over the last years, researchers have developed a number of methods supporting the user in the fault localization and correction (debugging) process. The techniques range from the visualization of suspicious cells or regions of the spreadsheet, and the application of known practices from software engineering like spectrum-based fault localization (SFL) or slicing, to declarative and constraint-based reasoning techniques [1, 3, 6, 7, 9, 11, 12, 16].

However, there is a number of challenges common to all these approaches. Unlike other computer science sub-areas, such as natural language processing, information retrieval or automated planning and scheduling, no standard benchmarks exist for spreadsheet debugging methods. The absence of commonly used benchmarks prevents the direct comparison of spreadsheet debugging approaches. Furthermore, fault localization and debugging for spreadsheets require the design of a user-debugger interface. An important question in this context is: what input or interaction can realistically be expected from the user? Finally, the main question to be answered is whether or not automated debugging techniques actually help the developer as discussed in [14] for imperative programs.

In this position paper, we discuss some limitations of the current research practice in the field and outline potential ways to improve the research practice in the future.

2. LACK OF BENCHMARK PROBLEMS

To demonstrate the usefulness of a new debugging technique, we need spreadsheets containing faults. Since no public set of such spreadsheets exists, researchers often create their own suite of benchmark problems, e.g., by applying mutation operators to existing correct spreadsheets [2]. Unfortunately, these problems are only rarely made publicly available. This makes a comparative evaluation of approaches difficult and it is often unclear if the proposed technique is applicable to a wider class of spreadsheets.

In some papers, spreadsheets from the EUSES corpus[1] are used for evaluations. As no information exists about the intended semantics of these spreadsheets, mutations are applied in order to obtain faulty versions of the spreadsheets. The spreadsheets in this corpus are however quite diverse, e.g., with respect to their size or the types of the used formulas. Often only a subset of the documents is used in the evaluations and the selection of the subset is not justified well. Even when the benchmark problems are publicly shared like the ones used in [10], they may have special characteristics that are advantageous for a certain method and, e.g., contain only one single fault or use only certain functions or cell data types.

A corpus of diverse benchmark problems is strongly needed for spreadsheet debugging to make different research approaches better comparable and to be able to identify shortcomings of existing approaches. Such a corpus could be incrementally built by researchers sharing their real-world and artificial benchmark problems. In addition, since it is not always clear if typical spreadsheet mutation operators truly correspond to mistakes developers make, insights and practices from the Information Systems field should be better integrated into our research. This in particular includes the use of spreadsheet construction exercises in laboratory settings that help us identify which kinds of mistakes users make and what their debugging strategies are, see, e.g., [4].

[1] `http://esquared.unl.edu/wikka.php?wakka=EUSESSpreadsheetCorpus`

3. USABILITY AND USER ACCEPTANCE

Spreadsheet debugging research is often based on offline experimental designs, e.g., by measuring how many of the injected faults are successfully located with a given technique, see, e.g., [5]. In some cases, plug-ins to spreadsheet environments are developed like in [1] or [11]. Similar to plug-ins used for other purposes, e.g., spreadsheet testing, the usability of these plug-ins for end users is seldom in the focus of the research. The proposed plug-ins typically require various types of input from the user at different stages of the debugging process. Some of these inputs have to be provided at the beginning of the process and some can be requested by the debugger during fault localization. Typical inputs of a debugger include statements about the correctness of values/formulas in individual cells [10], information about expected values for certain cells [1, 3], specification of multiple test cases [11], etc.

In many cases, it remains unclear, if an average spreadsheet developer will be willing or able to provide these inputs since concepts like test cases do not exist in the spreadsheet paradigm. Therefore, researchers have to ensure that a developer interprets the requests from the debugger correctly and provides appropriate inputs as expected by the debugger. One additional problem in that context is that user inputs, e.g., the test case specifications, are usually considered to be reliable and most existing approaches have no built-in means to deal with errors in the inputs.

Overall, we argue that offline experimental evaluations should be paired with user studies whenever possible as done, e.g., in [8] or [11]. Such studies should help us validate whether our approaches are based on realistic assumptions and are acceptable at least for ambitious users after some training. At the same time, observations of the users' behavior during debugging can be used to learn about their problem solving strategies and to evaluate whether the tool actually helped to find a fault.

Again, insights and practices both from the fields of Information Systems and Human Computer Interaction should be the basis for these forms of experiments.

4. FIELD RESEARCH

In addition to user studies in laboratory environments, research on real spreadsheets as suggested in [15] is required to determine potential differences between the experimental usage of the proposed debugging methods and the everyday use of such tools in companies or institutes. Error rates and types found in practice could differ from what is observed in user studies whose participants in many cases are students. In [13], e.g., a construction exercise with business managers was done to determine error rates. In addition, the user acceptance of fault localization tools could vary strongly because of different expectations of professional users with respect to the utilized tools. To ensure the usability for real users, existing spreadsheets can be examined and questionnaires with users can be made, as done, e.g., in [7].

5. CONCLUSIONS

A number of proposals have been made in the recent literature to assist the user in the process of locating faults in a given spreadsheet. In this position paper, we have identified some limitations of current research practice regarding the comparability and reproducibility of the results. As possible remedies to these shortcomings we advocate the development of a corpus of benchmark problems and the increased adoption of user studies of various types as an evaluation instrument. As experimental settings differ from real-life, we additionally propose to use field studies to obtain insights on how debugging methods are used in companies.

6. REFERENCES

[1] R. Abraham and M. Erwig. GoalDebug: A Spreadsheet Debugger for End Users. In *Proc. ICSE 2007*, pages 251–260, 2007.

[2] R. Abraham and M. Erwig. Mutation Operators for Spreadsheets. *IEEE Trans. on Softw. Eng.*, 35(1):94–108, 2009.

[3] R. Abreu, A. Riboira, and F. Wotawa. Constraint-based debugging of spreadsheets. In *Proc. CibSE'12*, pages 1–14, 2012.

[4] P. S. Brown and J. D. Gould. An Experimental Study of People Creating Spreadsheets. *ACM TOIS*, 5(3):258–272, 1987.

[5] C. Chambers and M. Erwig. Automatic Detection of Dimension Errors in Spreadsheets. *J. Vis. Lang. & Comp.*, 20(4):269–283, 2009.

[6] J. Cunha, J. a. P. Fernandes, H. Ribeiro, and J. a. Saraiva. Towards a catalog of spreadsheet smells. In *Proc. ICCSA'12*, pages 202–216, 2012.

[7] F. Hermans, M. Pinzger, and A. van Deursen. Supporting Professional Spreadsheet Users by Generating Leveled Dataflow Diagrams. In *Proc. ICSE 2011*, pages 451–460, 2011.

[8] F. Hermans, M. Pinzger, and A. van Deursen. Detecting and Visualizing Inter-Worksheet Smells in Spreadsheets. In *ICSE 2012*, pages 441–451, 2012.

[9] F. Hermans, M. Pinzger, and A. van Deursen. Detecting Code Smells in Spreadsheet Formulas. In *Proc. ICSM 2012*, pages 409–418, 2012.

[10] B. Hofer, A. Riboira, F. Wotawa, R. Abreu, and E. Getzner. On the Empirical Evaluation of Fault Localization Techniques for Spreadsheets. In *Proc. FASE 2013*, pages 68–82, 2013.

[11] D. Jannach and T. Schmitz. Model-based diagnosis of spreadsheet programs - A constraint-based debugging approach. *Autom. Softw. Eng.*, to appear, 2014.

[12] 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*, to appear, 2014.

[13] F. Karlsson. Using two heads in practice. In *Proc. WEUSE 2008*, pages 43–47, 2008.

[14] C. Parnin and A. Orso. Are Automated Debugging Techniques Actually Helping Programmers? In *Proc. ISSTA 2011*, pages 199–209, 2011.

[15] S. G. Powell, K. R. Baker, and B. Lawson. A critical review of the literature on spreadsheet errors. *Decision Support Systems*, 46(1):128–138, 2008.

[16] J. Reichwein, G. Rothermel, and M. Burnett. Slicing Spreadsheets: An Integrated Methodology for Spreadsheet Testing and Debugging. In *Proc. DSL 1999*, pages 25–38, 1999.