The Journal of Systems and Software 000 (2016) 1–14

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

How effectively can spreadsheet anomalies be detected: An empirical study

Ruiqing Zhang[a,b], Chang Xu[a,b,*], S.C. Cheung[c], Ping Yu[a,b], Xiaoxing Ma[a,b], Jian Lu[a,b]

[a] State Key Laboratory for Novel Software Technology, Nanjing Unversity, Nanjing, China
[b] Department of Computer Science and Technology, Nanjing Unversity, Nanjing, China
[c] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China

ARTICLE INFO

Article history:
Received 27 October 2015
Revised 17 March 2016
Accepted 21 March 2016
Available online xxx

Keywords:
Spreadsheet
Anomaly detection
Empirical study

ABSTRACT

While spreadsheets are widely used, they have been found to be error-prone. Various techniques have been proposed to detect anomalies in spreadsheets, with varying scopes and effectiveness. Nevertheless, there is no empirical study comparing these techniques' practical usefulness and effectiveness. In this work, we conducted a large-scale empirical study of three state-of-the-art techniques on their effectiveness in detecting spreadsheet anomalies. Our study focused on the precision, recall rate, efficiency and scope. We found that one technique outperforms the other two in precision and recall rate of spreadsheet anomaly detection. Efficiency of the three techniques is acceptable for most spreadsheets, but they may not be scalable to large spreadsheets with complex formulas. Besides, they have different scopes for detecting different spreadsheet anomalies, thus complementing to each other. We also discussed limitations of these three techniques. Based on our findings, we give suggestions for future spreadsheet research.

1. Introduction

Spreadsheets are widely used by end users for computation tasks, such as numerical analysis, statistical analysis, decision making, financial accounting, and so on. Despite their wide adoption, spreadsheets are error-prone (Powell et al., 2008). Spreadsheet development is rarely driven by good software engineering practices. For example, there are few comments on spreadsheets to facilitate their changes and maintenance. Spreadsheets can easily become complicated and disordered due to constant evolution in their life cycles, which also worsens the problem. Research has reported that spreadsheet errors have induced huge financial loss to many organizations (Panko, 2006).

To address this problem, various techniques have been proposed to improve the quality of spreadsheets by avoiding (Luckey et al., 2012; Cunha et al., 2014; Badame and Dig, 2012), detecting (Burnett and Erwig, 2002; Hermans et al., 2012a, 2013) and fixing (Abraham and Erwig, 2008, 2009) errors in spreadsheets. Although these techniques claim to be beneficial, there is little empirical evidence that they can improve the quality of practical spreadsheets. Absence of comprehensive and persuasive evaluation on these techniques makes their effectiveness unclear and indistinguishable to end users (Jannach et al., 2014).

This paper presents an empirical study to evaluate three influential techniques and compare their performance in detecting spreadsheet *anomalies* (including smells and errors). The three techniques under study are AmCheck (Dou et al., 2014), UCheck (Abraham and Erwig, 2007) and Dimension (Chambers and Erwig, 2009). All of them can detect anomalies in spreadsheets without assuming test oracles or parameter thresholds in advance. Such anomalies are not those readily visible by Excel's checking rules.

In our study, we selected two large-scale spreadsheet corpora and one academic corpus as our subjects. They are the EUSES Spreadsheet Corpus (Fisher and Rothermel, 2005), Enron Spreadsheet Corpus (Hermans and Murphy-Hill, 2015) and Hawaii Kooker Corpus (Aurigemma and Panko, 2010). The EUSES Spreadsheet Corpus (Fisher and Rothermel, 2005) has been widely used (but typically by its small samples) for spreadsheet evaluation and research since its creation in 2005 (Dou et al., 2014; Hermans et al., 2012b; Außerlechner et al., 2013). It contains 4,037 real-life spreadsheets from 11 categories. The Enron Spreadsheets Corpus consists of 15,929 spreadsheets extracted from the Enron Email Archive, which is an archive of email messages of the Enron corporation (Hermans and Murphy-Hill, 2015). The Hawaii Kooker Corpus contains 74 spreadsheets created from a word problem by students at the University of Hawaii (Aurigemma and Panko, 2010). The

* Corresponding author at: Department of Computer Science and Technology, Nanjing Unversity, Nanjing, China. Tel.: +86 2589680919; fax: +86 83593283.
E-mail addresses: zhangrq3216@163.com (R. Zhang), changxu@nju.edu.cn (C. Xu), scc@cse.ust.hk (S.C. Cheung), yuping@nju.edu.cn (P. Yu), xxm@nju.edu.cn (X. Ma), lj@nju.edu.cn (J. Lu).

http://dx.doi.org/10.1016/j.jss.2016.03.061
0164-1212/

anomalies in these spreadsheets were made by students naturally. To our best knowledge, no existing research on spreadsheets has been evaluated using such large-scale subjects in a full manner.

Our experimental results show that AmCheck outperforms UCheck and Dimension in detecting spreadsheet anomalies. We found that the precision and recall rate for spreadsheet anomaly detection by UCheck and Dimension are low. Nevertheless, they could detect certain anomalies, respectively, which other techniques could not detect. We found that all the three techniques completed processing most spreadsheets in the study within an acceptable time limit. However, their efficiency became poor for some large-scale spreadsheets with complex formulas. We also found that the three techniques detected very different sets of spreadsheet anomalies. This suggests that they complement each other for improving the coverage of spreadsheet anomaly detection. Finally, we present some common patterns with illustrative examples, which could not be processed correctly or satisfactorily by the three techniques, to explain the limitations of the three techniques. Based on them, we point out potential research opportunities for better spreadsheet anomaly detection.

Overall, this paper makes the following contributions:

1. A comprehensive evaluation of three spreadsheet anomaly detection techniques on spreadsheets of a significantly large volume.
2. An in-depth comparison of the three techniques about various aspects concerning their performance, which include precision, recall rate, efficiency and scope.
3. A careful analysis of the three techniques' limitations and proposal of corresponding suggestions for follow-up research in this field.

The remainder of this paper is organized as follows. Section 2 introduces existing spreadsheet anomaly detection techniques and explains our selection of the three techniques for our study. Section 3 presents our experimental design and puts forward research questions for study. Section 4 describes our experimental procedures. Section 5 analyzes our experimental results and answers research questions. Section 6 presents our analyses of the limitations of the three studied techniques and their results for different spreadsheet categories. Section 7 analyzes the threats in our experiments, which are followed by a discussion of recent related work in Section 8. Finally, Section 9 concludes this paper.

2. Background

In the section, we introduce the background to spreadsheet anomaly detection, as well as our selection of AmCheck, UCheck and Dimension for comparing their performance in detecting spreadsheet anomalies.

2.1. Selection of techniques

Spreadsheets are used for various purposes in organizations everywhere, but they are particularly prone to errors and cause huge financial loss. For this case, various techniques have been proposed to detect spreadsheet anomalies during the last decade. However, many of them are inappropriate for direct comparisons in our study.

In our study, we are concerned about two types of anomalies. One type is *error*, which indicates mistakes. For example, we consider that a cell contains an error if its formula is incorrect and has led to a wrong value. The other type is *smell*, which likely turns into an error with the evolution of spreadsheets. For example, a cell *D*1's formula should be "=*A*1+*B*1+*C*1", but its actual formula is "=*A*1+*B*1". If cell *C*1 is empty, cell *D*1's value is correct at the moment. However, we consider that cell *D*1 contains a smell because *C*1 may be filled with a value later, which would result in an error (incorrect *D*1 value). Both types of anomalies concern spreadsheet cells' computational semantics, and they are the focus of our study in this paper.

We exclude some spreadsheet anomaly detection techniques from the consideration in our study if they rely on any user-provided subjective treatment (e.g., threshold parameter) or extra resource (e.g., test case or oracle). This is because such treatment or resource would cause a technique's performance subject to variety or comparisons between techniques not on a fair base. For example, Hermans et al. (Hermans et al., 2012a, 2012b) adapted the concept of code smell to spreadsheets and presented formula smells in spreadsheets. They defined metrics to detect syntactic issues, such as multiple operations and multiple references in formulas in spreadsheet cells. However, users must provide a threshold to decide whether a formula is an anomaly, which is subjective. Jannach and Schmitz (Jannach and Schmitz, 2016) proposed translating spreadsheet checking to a constraint satisfaction problem and using classical diagnosis algorithms to detect anomalies in spreadsheets. It requires extra test cases for solving constraints. Hofer et al. (Hofer et al., 2013) adapted spectrum-based fault localization techniques to detect spreadsheet anomalies. It requires failing test cases to locate cells with faults. Abreu et al. (Abreu et al., 2014) proposed a technique to automatically pinpoint potential smells in spreadsheets, e.g., empty cells and multiple operations. However, such smells are mostly syntactic ones in formula writing, and require specific thresholds to decide. For example, the smell of multiple operations requires users to provide a threshold to determine how many operations are too many for a proper formula. Since it does not satisfy our criterion for selecting spreadsheet anomaly detection techniques in the study as mentioned above, and some of its targeted smell types can already be detected by Excel (e.g., reference to empty cells), we choose to exclude it from further comparison. These techniques require oracles or test cases to work, which may not be generally available for spreadsheet corpora in practice.

With the above consideration, we select AmCheck (Dou et al., 2014), UCheck (Abraham and Erwig, 2007) and Dimension (Chambers and Erwig, 2009) for our study, as they particularly focus on detecting spreadsheet anomalies and do not rely on extra information.

2.2. Introduction to the three techniques

In the following, we introduce the three selected spreadsheet anomaly detection techniques.

AmCheck is a recent technique that detects and repairs ambiguous computation smells in spreadsheets. It is based on the observation that formula cells grouped together in a row or column usually share the same computational semantics, which are called a *cell array*. AmCheck first extracts cell arrays that should follow the same computational semantics and identifies those from these cell arrays that contain inconsistent formulas. Then, it extracts constraints of formulas from these cell arrays, and uses them to recover or synthesize formula patterns of cell arrays. At last, it checks all cells in a cell array and reports those cells with inconsistent formulas as anomalies. AmCheck claims to be able to detect two kinds of ambiguous computation smells: *missing formula smell* and *inconsistent formula smell*. The former occurs when some cells in a cell array do not contain any formula, while the latter occurs when some cells contain inconsistent (non-equivalent) formulas. Taking the table in Fig. 1 as an example, AmCheck can extract the cell area [*D*2:*D*6] as a cell array first. Then, it infers a formula "=RC[−2]+RC[−1]" as the formula pattern of this cell array. At last, cells *D*4 and *D*5 are reported as anomalies because they do not contain correct formulas. To be specific, cell *D*4 contains a missing

Please cite this article as: R. Zhang et al., How effective can spreadsheet anomalies be detected: An empirical study, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.03.061