How Effective 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] and Jian Lu[a,b]

[a] *State Key Lab 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*

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.

Keywords: Spreadsheet, Anomaly detection, Empirical study

Email addresses: zhangrq3216@163.com (Ruiqing Zhang[a,b]), changxu@nju.edu.cn (Chang Xu[a,b]), scc@cse.ust.hk (S.C. Cheung[c]), {yuping,xxm,lj}@nju.edu.cn (Ping Yu[a,b], Xiaoxing Ma[a,b] and Jian Lu[a,b])

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 [26]. 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 [25].

To address this problem, various techniques have been proposed to improve the quality of spreadsheets by avoiding [24][13][8], detecting [9][19][20] and fixing [3][4] 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 [23].

This article 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 [14], UCheck [2] and Dimension [11]. 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 [15], Enron Spreadsheet Corpus [17] and Hawaii Kooker Corpus [6]. The EUSES Spreadsheet Corpus [15] has been widely used (but typically by its small samples) for spreadsheet evaluation and research since its creation in 2005 [14][18][7]. 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 [17]. The Hawaii Kooker Corpus contains 74 spreadsheets created from a word problem by students at the University of Hawaii [6]. The 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 article 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 article 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 article.

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 $D1$'s formula should be "=A1+B1+C1", but its actual formula is "=A1+B1". If cell $C1$ is empty, cell $D1$'s value is correct at the moment. However, we consider that cell $D1$ contains a smell because $C1$ may be filled with a value later, which would result in an error (incorrect $D1$ value). Both types of anomalies concern spreadsheet cells' computational semantics, and they are the focus of our study in this article.

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. [19] [18] 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 [22] 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. [21] adapted spectrum-based fault localization techniques to detect spreadsheet anomalies. It requires failing test cases to locate cells with faults. Abreu et al. [5] 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 [14], UCheck [2] and Dimension [11] 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 [D2:D6] 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 D4 and D5 are reported as anomalies because they do not contain correct formulas. To be specific, cell D4 contains a missing formula smell because it is overridden with a plain value. Cell D5 contains an inconsistent formula smell because this formula is non-equivalent to the inferred formula pattern for the cell array it belonging to. The authors conducted an empirical study on the EUSES Spreadsheet Corpus and a case study on real-life spreadsheets to evaluate this technique's performance. However, no comparison with existing techniques has been conducted.

UCheck uses a unit reasoning system that exploits the information of headers in spreadsheets to check the consistency of formulas in spreadsheets. It detects anomalies according to the fact that incorrect formulas often exhibit inconsistent unit information. UCheck is based on two static analysis phases that infer header and unit information [1], respectively, for all cells in spreadsheets. It first

	A	B	C	D
1		**Apple**	**Orange**	**Total**
2	**Febrary**	1	3	=B2+C2
3	**March**	2	4	=B3+C3
4	**April**	3	5	8
5	**May**	4		=B5
6	**June**	3	6	=B6+C6

Figure 1: Detecting anomalies by AmCheck: example spreadsheet and cell array

	A	B	C	D
1		Fruit		
2	Month	Apple	Orange	Plum
3	May	4	5	6
4	June	7	7	8
5	July	6	5	0
6	Total	=SUM(B2:B4)	=SUM(C3:C5)	=SUM(D3:D5

Figure 2: Detecting anomalies by UCheck: example spreadsheet

	A	B	C	D
1		Fruit		
2	Month	Apple	Orange	Plum
3	May	4	5	6
4	June	7	7	8
5	July	6	5	0
6	Total	=SUM(B2:B4)	=SUM(C3:C5)	=SUM(D3:D5

Figure 3: Detecting anomalies by UCheck: header inference

infers header information (used to label spreadsheet data and indicate data's meanings) for spreadsheets according to their layouts and structures. Then, it assigns units to all spreadsheet cells on the basis of the inferred header information. After that, it tries to simplify the unit information to a normal form according to its built-in rules. Cells with units that cannot be simplified to well-formed ones are reported as anomalies. Taking the table in Fig. 2 as an example, UCheck would infer its header information first, as shown in Fig. 3. We can observe that "Fruit" is the header of cell B2, C2 and D2, "Apple" is the header of cell B3, B4, B5 and B6 in Fig. 3. Then, UCheck assigns unit information to each cell. For example, cell B2's unit is "Fruit", cell B3's unit is "Month [May] & Fruit [Apple]", and cell B6's unit is "Fruit | Month [May] & Fruit [Apple] | Month [June] & Fruit [Apple]". Then, UCheck simplifies cell B6's unit to

	A	B	C	D
1		Time	Speed	Length
2	driver1	3	200	=B2*C2
3	driver2	4	180	=B3*C3
4	driver3	5	190	=B4+C4
5	driver4	6	210	=B5-C5

Figure 4: Detecting anomalies by Dimension: example spreadsheet

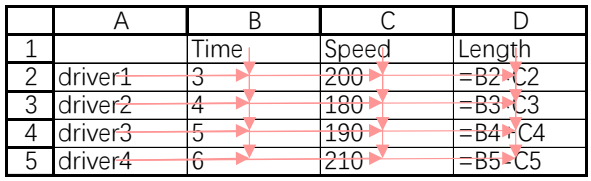

Figure 5: Detecting anomalies by Dimension: header and dimension inference

"Fruit | (Month [May | June] & Fruit [Apple])", which is not well-formed since it violates UCheck's fourth rule for meaningful unit expressions. Hence, UCheck would report cell B6 as an anomaly. UCheck has been evaluated on two small sets of spreadsheets. One set consists of 10 spreadsheet examples from a book and another consists of 18 spreadsheets developed by students. We consider that these spreadsheets are too few and they might not be representative.

Dimension uses a reasoning system that checks the consistency of spreadsheet formulas based on inferred dimension information (units for measuring spreadsheet data contained in cells) [10]. It also analyzes spreadsheets' spatial structures to infer the header information as UCheck does. Then it analyzes the inferred labels[1] and assigns them to potential dimensions. At last, it propagates the dimension information through spreadsheet formulas and detects anomalies during this propagation process. If a cell's dimension information is invalid or two incompatible dimensions are added, then Dimension reports an anomaly. Taking the table in Fig. 4 as an example, Dimension infers its header informa-

[1]UCheck and Dimension's authors used two different terms for different meanings. They used "header" to emphasize relationships between cells, and used "label" to refer to contents in cells (header cells).

tion first, as shown in Fig. 5, just like UCheck. Then, it analyzes labels "Time", "Speed" and "Length" and assigns them to potential dimensions. For example, "Second (s)" will be assigned as the default measurement to cells B2, B3, B4 and B5 because their headers are "Time". Dimension will report cell D4 as an anomaly because it adds cell B4, whose measurement is "Second (s)", to cell C4, whose measurement is "m/s", and this is regarded as illegal. Similarly, cell D5 is also reported as an anomaly. Dimension has been evaluated on a small subset of the EUSES Spreadsheet Corpus, containing only 40 spreadsheets.

We note that these three techniques can detect non-trivial spreadsheet anomalies, i.e., not those that can already be detected by Excel's checking rules. Besides, all of them can work without user-provided subjective thresholds or extra resources. Currently, there is no existing comprehensive study investigating whether the anomalies the three techniques detect are common and how these techniques are compared to each other.

3. Experimental Design

In the section, we explain the design of our experiments, including research questions, dependent and independent variables and experimental subjects.

3.1. Research Questions

The performance of a spreadsheet anomaly detection technique includes its precision and recall rate. Besides, efficiency is also an important concern. Therefore, our experiments study the following three research questions concerning precision, recall rate and efficiency, respectively:

- RQ1: *Which of the three techniques has the lowest false positive rate in reporting anomalous cells (best precision)?*
- RQ2: *Which of the three techniques can detect the most anomalous cells (best recall rate)?*
- RQ3: *Which of the three techniques is the most efficient?*

Since all the three techniques report certain cells in spreadsheets as anomalies, the precision or recall rate is measured with respect to anomalies in cells (or anomalous cells). For example, the precision of a technique is measured by the proportion of real anomalous cells against all anomalous cells reported by the technique. We call a cell *anomalous cell* if it contains an anomaly (error or smell), and worksheets containing anomalous cells *anomalous worksheets*. Unless otherwise specified, "anomaly" mentioned later represents "anomalous cell" for convenience.

The three techniques take advantage of different properties of spreadsheets to detect anomalies. Specifically, AmCheck relies on the extraction of cell arrays. UCheck utilizes the header and unit information of cells. Dimension checks the validity of dimension information assigned from label information. The three techniques thus detect anomalies with different assumptions and therefore their scopes of spreadsheet anomaly detection may differ.

- RQ4: *Are the scopes of the three techniques the same? If not, which one has a wider scope?*

We conjecture that the three techniques can detect different anomalies because they leverage different spreadsheet properties for anomaly detection. We check whether the anomalies they detect overlap. Therefore, we also study the following research question:

- RQ5: *Do the anomalies detected by the three techniques have any overlapping?*

At last, we discuss whether the three techniques are subject to any limitation. We attempt to make suggestions for improvement as well as inspiring future research. Therefore, the last research question is:

- RQ6: *What are the limitations of the three techniques?*

We conduct controlled experiments to answer the above six research questions.

3.2. Variables

There are four dependent variables in our experiments:

- **Precision.** The precision is measured by the proportion of real anomalies against all anomalies reported by a technique.
- **Recall rate.** The recall rate is measured by the proportion of real anomalies detected by a technique against all existing real anomalies. However, we note that one cannot know all existing real anomalies in advance due to the lack of oracle for determining anomalies in spreadsheets. Therefore, we sampled a subset of spreadsheets and inspected them manually for the ground truth.
- **Efficiency.** The efficiency is measured by the time a technique spends on processing spreadsheets.
- **Scope.** The scope is measured by the number of spreadsheets/worksheets that meet the assumptions of a technique for detecting anomalies (a spreadsheet contains one or more worksheets as different pages). For example, AmCheck requires a spreadsheet to own cell arrays, UCheck requires a spreadsheet to be able to pass its header and unit inference, and Dimension requires a spreadsheet to contain dimension information, as aforementioned.

The only independent variable in our experiments is the technique used for detecting spreadsheet anomalies. The three treatments for this variable are AmCheck, UCheck and Dimension, as mentioned earlier.

3.3. Experimental Subjects

As mentioned, we selected the EUSES Spreadsheet Corpus [15], Enron Spreadsheet Corpus [17] and Hawaii Kooker Corpus [6] as our experimental subjects. We obtained a large collection of worksheets from these spreadsheet corpora, but ignored those that violate either of the following two criteria:

- The worksheet's owner spreadsheet can be manipulated by the Apache POI[2].
- The worksheet contains at least one formula.

A worksheet is a sheet (or page) of rows and columns in a spreadsheet, and each spreadsheet can contain multiple worksheets. Take the EUSES Spreadsheet Corpus as an example, the number of worksheets in a spreadsheet ranges greatly (from 1 to 45). Therefore, we consider a worksheet as a unit in our study. We used the Apache POI as the interface to process spreadsheet files. So, those spreadsheets that cannot be manipulated by the Apache POI were ignored. The latter criterion requires that a worksheet should contain at least one formula because all the three techniques detect anomalies on the basis of formulas. So, those worksheets without formulas were ignored (they actually contain plain data only).

4. Experimental Procedures

In this section, we introduce our experimental procedures, which consist of three phases: preparation, detection and inspection. In the preparation phase, we adapted the three spreadsheet anomaly detection tools (techniques) to make them able to process spreadsheet subjects from the three corpora in a batch mode. In the detection phase, we applied the three tools to detect spreadsheet anomalies in turn. We recorded the detected anomalies, spent time and other relevant data. In the inspection phase, we sampled a subset of all processed worksheets and inspected them manually for the ground truth. Fig. 6 gives an overview of our whole study process.

4.1. Tool Adaptation and Spreadsheet Preparation

We used original algorithm or tool implementations of the three spreadsheet anomaly detection techniques for experiments, except that we only re-

[2]http://poi.apache.org/.

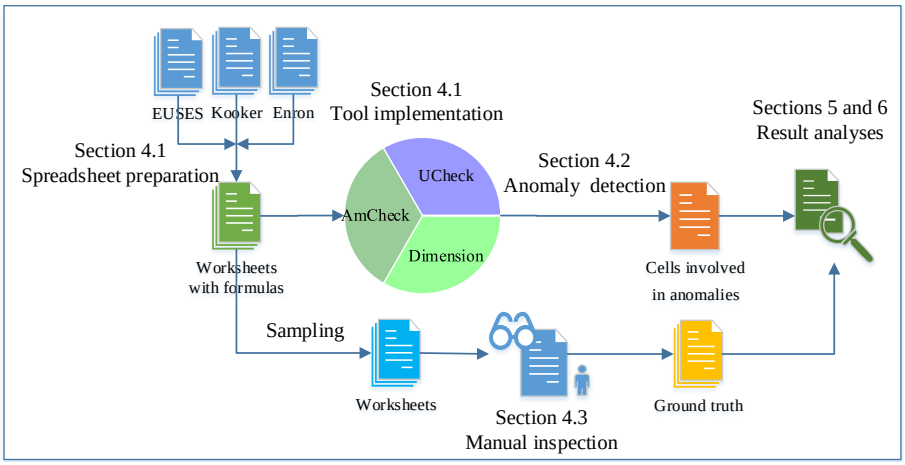

Figure 6: Overview of the study process. We first preprocessed the three corpora and implemented interfaces for manipulating the three tools (techniques). Then we used the adapted tools to detect anomalies in worksheets and recorded cells and worksheets involved in anomalies. At last, we inspected sampled worksheets for the ground truth and analyzed experimental results.

Table 1: Preprocessed results of spreadsheet subjects (Apache POI: numbers of spreadsheets that can be manipulated by the Apache POI; With formulas: numbers of worksheets with formulas; Removing redundance: numbers of the remaining worksheets after removing redundance)

	Spreadsheets	Apache POI	Worksheets	With formulas	Removing redundance
EUSES	4,036	3,745	13,661	4,054	4,054
Kooker	74	74	583	150	76
Enron	15,928	15,859	68,730	36,907	15,529

implemented their I/O interfaces for facilitating spreadsheet manipulation.

Regarding spreadsheet preparation, we kept those worksheets that can be manipulated by the Apache POI and contain formulas, as mentioned earlier. The preprocessed results of the three spreadsheet corpora are given in Table 1. Column "Spreadsheets" represents the numbers of all spreadsheets in a corpus. Column "Apache POI" represents the numbers of spreadsheets that can be

manipulated by the Apache POI. Column "Worksheets" represents the numbers of all worksheets in spreadsheets that can be manipulated by the Apache POI. Column "With formulas" is a refinement of "Worksheets" with the "containing at least one formula" constraint. Column "Removing redundance" represents the numbers of the remaining worksheets after removing redundance, which is explained below.

For the EUSES Spreadsheet Corpus, 3,745 out of 4,036 spreadsheets can be manipulated by the Apache POI. These spreadsheets contain 13,661 worksheets, and 4,054 of them contain at least one formula, which is our focus in the study. For the Hawaii Kooker Corpus, all its 74 spreadsheets can be manipulated by the Apache POI. They contain 583 worksheets, but most of them are blank sheets and only 150 worksheets contain formulas. Among these worksheets, most of them are paired, i.e., one is named "Original" and the other is "Annotated". They contain the same contents except that the later is labeled with some extra annotations as comments. We thus ignored such redundant worksheets and kept only 76 "Original" worksheets for experiments. The situation of the Enron Spreadsheet Corpus is more complex. We found mass redundant worksheets, which are from the same developers and share almost exactly the same contents. To avoid analysis bias to certain worksheets, we removed such redundance before experiments. We observed that spreadsheets in the corpus have been renamed by their collectors according to certain rules, and we can derive their developer and original spreadsheet names from their corresponding file names. We distinguish "spreadsheet name" from "spreadsheet file name" for this particular corpus to avoid confusion. We found that a set of spreadsheets with the same spreadsheet name and created by the same developers typically share almost the same content, and this is a strong indicator of redundancy. They are probably different versions of the same spreadsheet with minor changes. To remove redundance, we randomly selected one of them for experiments due to lack of other evidence showing which one is better. Besides, there is another major source of redundance that some spreadsheets are from the same developers and contain almost the same contents but their spreadsheet names

differ slightly. We conjecture that this is because their developers intended to change their names for certain reasons. Such redundance also occurs to some worksheets in one spreadsheet, i.e., these worksheets were renamed after slight content changes. We took a unified, heuristic way to remove such redundance: if some worksheets are from the same developers and contain the same number of formula cells, text string cells and numerical cells, we consider that they are redundant and kept only one copy of them in a random way. Altogether, 15,529 of 36,907 worksheets in this corpus were finally reserved for our experiments.

In summary, we obtained/selected 4,054, 76 and 15,529 worksheets for experiments from the three corpora, EUSES Spreadsheet Corpus, Hawaii Kooker Coupus and Enron Spreadsheet Corpus, respectively.

4.2. Anomaly Detection

In this phase, we applied the three spreadsheet anomaly detection tools (techniques) to the selected worksheets to detect anomalies. We ran the three tools on the same platform, which consists of six virtual machines from a computer server. The server was equipped with an Intel(R) Xeon(R) CPU, consisting of 24 cores working at 2.4GHz and with 48 GB memory. Each virtual machine was configured to run with two cores and 6 GB memory.

For each selected worksheet (we may directly call it "each worksheet" for convenience later when not causing confusion), we counted the number of anomalies detected by each tool and recorded their specific locations for further analysis. We recorded the execution time of each tool on each worksheet for comparing efficiency. We also recorded the number of worksheets that meet the assumptions of each technique, to be specific, number of worksheets with cell arrays for AmCheck, number of worksheets that could pass the unit inference for UCheck, and number of worksheets that could pass the dimension inference for Dimension, to compare their scopes.

During the anomaly detection, we found that some worksheets could not be processed within even 24 hours, which was set as our timeout limit. Since we used the original implementations of the three techniques, such timeout case

Table 2: The numbers of worksheets that could be handled by the three techniques (Within timeout: numbers of worksheets that could be processed by all the three techniques within the timeout limit; AmCheck/UCheck/Dimension: numbers of worksheets that could be processed within the timeout limit and met each technique's assumptions; A&U&D: numbers of worksheets that could be processed within the timeout limit and met assumptions of all the three techniques)

	All worksheets	Within timeout	AmCheck	UCheck	Dimension	A&U&D
EUSES	4,054	4,049	2,101 (51.89%)	1,141 (28.18%)	1,626 (40.16%)	504 (12.45%)
Kooker	76	76	35 (46.05%)	31 (40.79%)	31 (40.79%)	20 (26.32%)
Enron	15,529	15,332	9,682 (63.15%)	2,231 (14.55%)	3,695 (24.10%)	999 (6.52%)

might be due to these tools' internal defects (e.g., infinite loops or dead locks). We removed these worksheets that could not be processed within the timeout limit from our subjects. To be specific, for the EUSES corpus, all worksheets could be processed within the timeout limit by AmCheck, while five worksheets could not by UCheck and two worksheets could not by Dimension. In total, five worksheets (4,054 − 4,049) from the EUSES corpus could not be processed by at least one technique within the timeout limit. For the Hawaii Kooker Corpus, all worksheets could be processed by all the three techniques within the timeout limit. For the Enron corpus, 14 worksheets could not be processed by AmCheck within the timeout limit, while 165 worksheets could not by UCheck and 158 worksheets could not by Dimension. In total, 197 worksheets (15,529 − 15,332) from the Enron corpus could not be processed by at least one technique within the timeout limit. At last, 4,049 worksheets from the EUSES corpus, 76 worksheets from the Hawaii Kooker Corpus and 15,332 worksheets from the Enron corpus could be processed by all the three techniques within the timeout limit.

Table 3: The numbers of anomalous worksheets and anomalous cells detected by the three techniques

	Anomalous worksheets			Anomalous cells		
	AmCheck	UCheck	Dimension	AmCheck	UCheck	Dimension
EUSES	614	132	688	10,012	3,124	12,602
Kooker	32	21	29	71	97	164
Enron	3,999	291	1,820	147,212	7,757	74,104

Even if a tool can run for a given spreadsheet, it may return no result (detected anomalies) if this spreadsheet does not meet this tool's (technique's) assumption. For example, AmCheck requires the spreadsheet to own cell arrays, UCheck requires the spreadsheet to be able to pass its header and unit inference, and Dimension requires a spreadsheet to contain dimension information, as mentioned earlier. We could obtain such information from tool outputs to decide whether a worksheet in experiments met a certain tool's assumption. Altogether, 504 worksheets from the EUSES corpus, 20 worksheets from the Hawaii Kooker Corpus, and 999 worksheets from the Enron corpus met the assumptions from all the three techniques.

It took us extremely long time to complete all experiments, especially for the Enron Spreadsheet Corpus, which was the largest. We totally spent about three months (24-hour running every day with six virtual machines). We give the demographics of worksheets that could be processed by the three techniques in Table 2. Column "All worksheets" represents the numbers of all worksheets selected as subjects. Column "Within timeout" represents the numbers of worksheets that could be processed by all the three techniques within the timeout limit. Column "AmCheck", "UCheck" and "Dimension" denote the numbers of worksheets that could be processed within the timeout limit and met each technique's own assumption. The last column "A&U&D" represents the numbers of worksheets that could be processed within the timeout limit and met assumptions for all the three techniques.

In the following, we further study anomaly detection results for those worksheets that could be processed within the timeout limit by all the three techniques. In Table 3 we present the number of anomalous worksheets and anomalous cells detected by each tool for each corpus. For all 4,049 worksheets from the EUSES Spreadsheet Corpus, 614 of them were considered to contain anomalies by AmCheck. Specifically, AmCheck reported 10,012 anomalies in these worksheets. UCheck detected 132 worksheets with anomalies, and among them pinpointed 3,124 cells with anomalies. Dimension detected 688 worksheets and 12,602 cells with anomalies. It is clear that AmCheck and Dimension reported much more anomalies than UCheck, no matter in the number of worksheets or that of cells. Table 3 also lists anomaly detection results for the other two corpora.

To study the nature of these detected anomalies, we partitioned them into seven categories (in terms of cells with anomalies), as in Table 4. The first column "A&U&D" represents the numbers of anomalies that were detected by all the three techniques. The following three columns represent anomalies that were detected by two techniques but not the remaining one. For example, column "A&U" represents the numbers of anomalies that were detected by both AmCheck and UCheck, but not by Dimension. The last three columns represent anomalies that were detected by one technique only. From Table 4, we observe that the anomalies that were detected by more than one technique are very few. For example, only three anomalies were detected by all the three techniques in the EUSES Spreadsheet Corpus.

4.3. Manual Inspection

The three spreadsheet anomaly detection techniques pinpointed quite a few anomalies. However, we could not precisely know whether they are real anomalies due to practical difficulties (e.g., no oracle and unable to contact original spreadsheet developers). Therefore, we have to inspect the concerned worksheets and cells manually for the ground truth.

It is practically impossible for us to inspect all worksheets because of their

Table 4: The numbers of anomalous cells detected by the three techniques (A&U&D: numbers of anomalies detected by all the three techniques; A&U/A&D/U&D: numbers of anomalies detected by two techniques; A/U/D: numbers of anomalies detected by one technique only)

	A&U&D	A&U	A&D	U&D	A	U	D
EUSES	3	0	50	279	9,959	2,842	12,270
Kooker	6	0	1	44	64	47	113
Enron	15	1	729	1,254	146,467	6,487	72,106

huge volume. So, we sampled them and inspected samples carefully. To avoid sampling bias, we set two criteria. First, a sampled worksheet must meet the assumptions of all the three techniques. This is because we want to compare performance for the three techniques when they can indeed work. Second, a sampled worksheet must have been reported to contain anomalies by at least one technique. Otherwise, there is nothing to compare. We totally sampled 100 worksheets from the EUSES Spreadsheet Corpus and Enron Spreadsheet Corpus, respectively. We sampled (selected) all the 20 worksheets from the Hawaii Kooker Corpus since this number is not large.

Our worksheet sampling process went this way. At first, two postgraduates randomly sampled worksheets together, but inspected them for the ground truth individually. From their ground truth results, they picked up different (inconsistent) items for further diagnosis. Then, two professors (coauthors of this article) inspected these different ground truth items individually, and gave their opinions and explanations. When these opinions and explanations were merged, we checked whether they were consistent. If yes, the concerned ground truth items were decided. If no, the concerned worksheets were discarded and more worksheets were sampled until they reached our target numbers (i.e., 100 worksheets for the EUSES corpus and 100 worksheets for the Enron corpus) for the study. For the Hawaii Kooker Corpus, we sampled and inspected all its worksheets (20) successfully. By doing so, we tried our best to alleviate potential bias from a single person. During the manual inspection, each person

inspected each worksheet in varying time, which ranged from several minutes to about half an hour. The total time cost, including individual inspection, cross-checking and final decision on all selected 220 worksheets, took us about 20 days.

5. Result Analyses

In this section, we present and analyze experimental results. We also answer our earlier raised research questions.

5.1. Automatic Anomaly Detection Results

As mentioned earlier, the three techniques have different assumptions on spreadsheets/worksheets to detect anomalies. Only worksheets with cell arrays can be processed by AmCheck for detecting anomalies. UCheck can detect anomalies in worksheets that can pass its header and unit inference. Dimension requires worksheets that can provide header and dimension information. We have recorded and compared the numbers of worksheets that meet the assumptions of the three techniques earlier in Table 2. We observe that AmCheck can detect anomalies for the most worksheets, while UCheck can detect anomalies for the least for all the three corpora.

We have also compared the numbers of detected anomalous worksheets and anomalous cells for the three techniques in Table 3, and partitioned detected anomalous cells into seven categories in Table 4.

Besides, we have recorded and compared spreadsheet processing time taken by the three techniques. Fig. 7 and Fig. 8 show accumulative percentages of worksheets that could be processed within given time by the three techniques for the EUSES Spreadsheet Corpus and Enron Spreadsheet Corpus, respectively. We did not study the processing time for the Hawaii Kooker Corpus since its contained spreadsheets are too few. The horizontal axis for Fig. 7 and Fig. 8 represents the given processing time (in seconds under a logarithmic scale), and the vertical axis represents the percentage of worksheets that could be

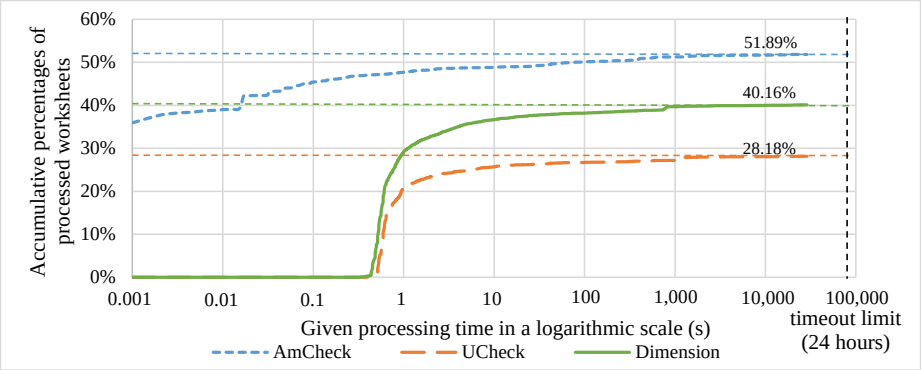

Figure 7: Accumulative percentages of worksheets that could be processed within given time in a logarithmic scale for the EUSES Spreadsheet Corpus.

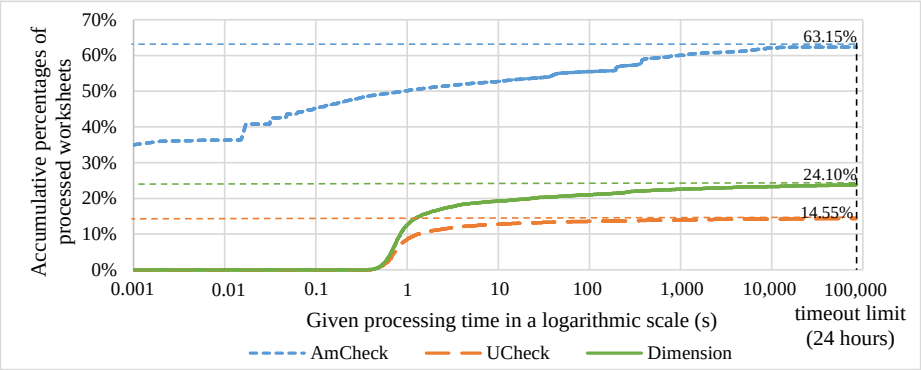

Figure 8: Accumulative percentages of worksheets that could be processed within given time in a logarithmic scale for the Enron Spreadsheet Corpus.

processed within the given time against the total worksheets. We note that each technique can process only a portion of the total worksheets as mentioned earlier, and therefore these techniques' limits do not reach 100%. From the figures, we observe that AmCheck processed clearly the most worksheets and thus its scope is the largest, while UCheck processed the least. We also observe that UCheck and Dimension behave similarly except the latter processed more worksheets. Although the processing time of each worksheet varies, we can still roughly observe that AmCheck processed worksheets much faster than UCheck and Dimension (2.2 seconds per worksheet vs. 120.9 and 107.8 seconds per

Table 5: The results of manual inspection and true positives for the three techniques on the three corpora (in terms of the number of anomalous cells)

	Ground truth	AmCheck		UCheck		Dimension	
		Reported	True	Reported	True	Reported	True
EUSES	611	332	188	301	2	2,597	58
Kooker	51	46	16	81	8	126	18
Enron	3,851	2,829	1,957	1,218	2	2,477	43

worksheet on average for EUSES, and 16.1 seconds vs. 441.2 and 851.8 seconds for Enron).

Besides, we also observe that AmCheck could process more than 35% worksheets within only 0.001 seconds for both corpora. This portion of worksheets (35%) is comparable to, or already more than, all worksheets UCheck and Dimension could process (28.15% and 40.11% for EUSES; 14.36% and 23.78% for Enron).

5.2. Manual Inspection Results

We manually inspected 100 worksheets sampled from the EUSES Spreadsheet Corpus and Enron Spreadsheet Corpus, respectively, and all the 20 worksheets from the Hawaii Kooker Corpus for the ground truth. Based on it, we calculated true positives, false positives and false negatives for the three techniques.

Table 5 presents the results of manual inspection and true positives. Column "Ground truth" represents the numbers of anomalies we identified (as real anomalies) from the sampled worksheets during the manual inspection. For each technique, column "Reported" denotes the numbers of anomalies detected by this technique for these sampled worksheets, and column "True" denotes the numbers of true positives in these detected anomalies.

We then calculated the precision and recall rate for the three techniques based on the ground truth in Fig. 9. We observe that AmCheck has the highest precision and recall rate in almost all three corpora. The precision and recall

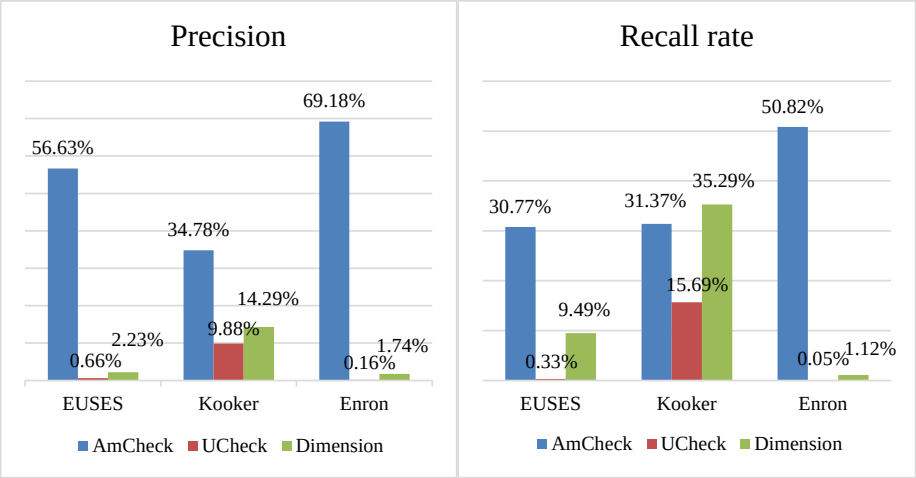

Figure 9: Precision and recall rate results for the three techniques on the three corpora.

rate of Dimension are higher than UCheck. We also observe that AmCheck has the best performance (precision and recall rate here) for the Enron Spreedsheet Corpus, while UCheck and Dimension have the best for the Hawaii Kooker Corpus. We consider that the Enron Spreadsheet Corpus was archived from the Enron corporation and it should better represent practical spreadsheets. On the contrary, the Hawaii Kooker Corpus was created by students and its contained spreadsheets share many similarities. As a summary, AmCheck can be more suitable for practical spreadsheets in terms of precision and recall rate in anomaly detection.

5.3. Answering Research Questions

In the following, we answer the research questions RQ1-6 based on our experimental results and analyses.

RQ1. Research question RQ1 concerns anomaly detection precision. The three techniques' precision comparison has been illustrated in Fig. 9. For the EUSES Spreadsheet Corpus, AmCheck's precision is as high as 56.63%, which is clearly higher than UCheck's precision, 0.66%, and Dimension's precision, 2.23%. The situation for the Enron Spreadsheet Corpus is similar. However,

things are a bit different for the Hawaii Kooker Corpus. AmCheck performed much worse for the Hawaii Kooker Corpus than for the other two corpora, while UCheck and Dimension performed much better (although AmCheck's precision is still the highest, as compared to those of UCheck and Dimension). In our manual inspection, we found that the spreadsheets from this corpus indeed share the same topic (as for the same problem) and their structures are also similar. Since UCheck and Dimension highly rely on table structures and their units to infer anomalies, they behaved similarly for all these spreadsheets. These spreadsheets are all small-sized and their layouts are simple. This helps increase the success rate for the header inference. Therefore, UCheck and Dimension both behaved relatively satisfactorily for spreadsheets from this corpus. Overall, AmCheck has the highest precision in spreadsheet anomaly detection. For practical scenarios, AmCheck's precision is relatively acceptable, but still has large room for improvement. UCheck and Dimension's precisions seem too low.

RQ2. Research question RQ2 concerns recall rate, which has also been illustrated in Fig. 9. We observe that the recall rate behaves similarly as the precision for all the three techniques. In general, AmCheck has the highest recall rate while UCheck has the lowest. AmCheck has the best recall rate for the Enron Spreadsheet Corpus and it has similar relatively low recall rates for the EUSES Spreadsheet Corpus and Hawaii Kooker Corpus. In contrast, UCheck and Dimension have the best recall rate for the Hawaii Kooker Corpus and the worst for the Enron Spreadsheet Corpus. To be specific, AmCheck's recall rate is 30.77% for the EUSES Spreadsheet Corpus, much higher than UCheck's 0.33% and Dimension's 9.49% for the same corpus. For the Enron Spreadsheet Corpus, AmCheck's recall rate reaches 50.82% but UCheck's is just 0.05%, which is too low.

As a whole, AmCheck has the best performance in both precision and recall rate, although their values are still far from satisfactory. On the contrary, UCheck and Dimension do not seem to perform well for the EUSES and Enron Spreadsheet Corpus. However, they have a much better performance for the Hawaii Kooker Corpus.

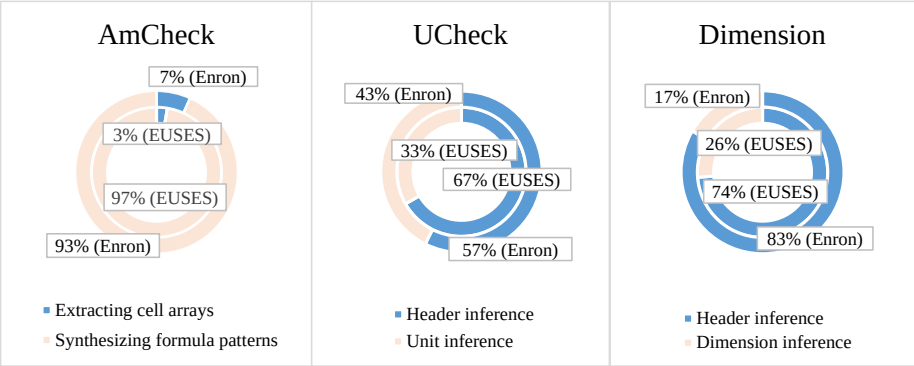

Figure 10: Time consumed by different modules of the three techniques.

RQ3. Research question RQ3 concerns efficiency. Due to reasons like implementation languages and file operations as we explained later, the three techniques cannot compare their efficiency precisely. Experimental results in Fig. 7 and Fig. 8 only generally suggest that the implementation of AmCheck is more efficient than those of UCheck and Dimension in detecting spreadsheet anomalies.

We note that all the three techniques encountered cases where they could not return any result even after processing a worksheet more than 24 hours. We inspected some of them and found that most of such worksheets are indeed large-scale or contain complex formulas. So, we consider that all the three techniques do not generally scale well to large-scale spreadsheets with complex formulas.

We are also wondering which parts of the three techniques are time-consuming. AmCheck consists of two modules, extracting cell arrays and synthesizing formula patterns. UCheck is composed of the header inference and unit inference. Dimension consists of the header inference and dimension inference. Fig. 10 illustrates the percentages of time consumed by different parts of each technique for the EUSES Spreadsheet Corpus and Enron Spreadsheet Corpus, respectively. We observe that the formula pattern synthesis cost the majority of processing time for AmCheck (93–97%). For UCheck, its header inference required more

time than its unit inference (57–67%), but for Dimension, its header inference consumed a clear majority of the total processing time (74–83%). So, we conjecture that the formula pattern synthesis in AmCheck and the header inference in UCheck and Dimension are probably performance bottlenecks and require improvement. For example, we suggest that AmCheck could deploy a more efficient constraint solver for its formula pattern synthesis, and that UCheck and Dimension may need to implement efficient algorithms to realize its heuristic rules for header inference, which consumes the most time cost.

RQ4. Research question RQ4 concerns scope. Table 2 presents the numbers and percentages of worksheets that meet the assumptions of the three techniques, respectively, and we use them to discuss the three techniques' scopes. Taking the EUSES Spreadsheet Corpus as an example, AmCheck can detect anomalies for 2,101 (51.89%) worksheets, while UCheck and Dimension can detect anomalies for 1,141 (28.18%) and 1,626 (40.16%) worksheets, respectively. The three techniques have similar results for the other two corpora. AmCheck clearly can detect anomaly for more worksheets than UCheck and Dimension. It indicates that the concept of cell array can be more common to spreadsheets, and we consider that AmCheck has the widest scope for spreadsheet anomaly detection (or it has the weakest assumption on spreadsheets for anomaly detection). UCheck processed the fewest worksheets and Dimension behaved in between for all the three corpora.

RQ5. Research question RQ5 concerns the overlapping of anomalies detected by the three techniques. From Table 4, we observe that the anomalies detected by at least two techniques (2,382) are much less than the anomalies detected by only one technique (250,355). This suggests that the anomalies detected by the three techniques rarely overlap. Nevertheless, this also suggests that the three techniques complement to each other as they focus on different sets of spreadsheet anomalies.

RQ6. Research question RQ6 concerns limitations of spreadsheet anomaly detection techniques. The major limitation for the three techniques is their low precisions and recall rates (Fig. 9), especially for UCheck and Dimension. A

closer study in the next section will disclose that AmCheck may extract incorrect cell arrays and this commonly results in false positives. UCheck and Dimension may infer incorrect header information and this leads to significantly reduced precisions and recall rates. This suggests that all the three techniques need improvement on their heuristic rules for correctly extracting cell arrays or inferring header information. Another limitation is their low efficiency. The three techniques all cost a large amount of time in processing spreadsheet subjects from the three corpora. Even some worksheets could not be processed within 24 hours, which is intolerable. All the three techniques do not generally scale to large-scale spreadsheets with complex formulas. Results also suggest that the formula pattern synthesis in AmCheck and the header inference in UCheck and Dimension are the main reasons for the inefficiency.

To improve the spreadsheet anomaly detection's precision and recall rate for UCheck and Dimension, we suggest that one key is to design better heuristic rules for inferring headers and analyzing labels, especially for complex spreadsheets. On one hand, such rules can be better obtained from systematic mining of more spreadsheet subjects, and the rules can also be adaptive, depending on specific features of spreadsheets under processing, when general rules do not widely exist. On the other hand, it is also suggested to establish standards (or programming paradigms) for spreadsheet development and guide users to follow in order to avoid some anomaly-inducing anti-patterns, which either cause spreadsheets prone to anomalies themselves, or reduce the effectiveness of certain spreadsheet detection techniques.

In summary, AmCheck performs better than UCheck and Dimension in scope, precision and recall rate. Although AmCheck has a wider scope than UCheck and Dimension, the latter two also detect spreadsheet anomalies AmCheck cannot detect. Regarding efficiency, all the three techniques are very time-consuming, and they are not scalable to large-scale spreadsheets with complex formulas.

	A	B	C	D
1		**Fruit**		
2	**Month**	**Apple**	**Banana**	**Total**
3	January	200	0	=B3+C3
4	February	220	0	=B4+C4
5	March	220	200	=B5+C5
6	April	160	300	=B6+C6
7	May	0	170	=B7+C7
8	June	0	210	=B8+C8

Figure 11: Example 1-0: A well-formed spreadsheet

	A	B	C	D
1		**Fruit**		
2	**Month**	**Apple**	**Banana**	**Total**
3	January	200	0	=B3
4	February	220	0	=B4
5	March	220	200	=B5+C5
6	April	160	300	=B6-C6
7	May	0	170	=B7
8	June	0	210	=B8

Figure 12: Example 1-1: A spreadsheet with anomalous cells

6. In-depth Analyses

In the section, we present some in-depth analyses of experimental results and illustrate the limitations of the three techniques using some spreadsheet examples.

6.1. Limitations of the Three Techniques

Both precisions and recall rates of the three techniques, especially, UCheck and Dimension, are not high or even quite low according to the results in Fig. 9, and we try to analyze the reasons. We sampled some worksheets and reviewed them carefully. We found that some common patterns had caused either false positives or false negatives to the three techniques. We illustrate them by some simple spreadsheet examples in the following and we believe that our analyses explain why they all received not high or even low precision and recall rate scores.

6.1.1. Limitations of UCheck

We first introduce a simple spreadsheet example in Fig. 11. The example has been widely discussed and its several variants have been referred to in existing

	A	B	C	D
1		**Fruit**		
2	**Month**	**Apple**	**Banana**	**Total**
3	January	200		=B3
4	February	220		=B4
5	March	220	200	=B5+C5
6	April	160	300	=B6-C6
7	May		170	=B7
8	June		210	=B8

Figure 13: Example 1-2: A spreadsheet with blank cells

	A	B	C	D
1				
2		**Apple**	**Banana**	**Total**
3	January	200	0	=B3
4	February	220	0	=B4
5	March	220	200	=B5+C5
6	April	160	300	=B6-C6
7	May	0	170	=B7
8	June	0	210	=B8

Figure 14: Example 1-3: A spreadsheet without high-level headers

work [2] [14]. This simple spreadsheet records the sales of two kinds of fruits, "Apple" and "Banana", in different months. Column D represents the total sales of the two fruits and its cells' formulas are the sum of corresponding cells in columns B and C. However, developers may introduce anomalies by mistake when editing this spreadsheet, as illustrated by Example 1-1 in Fig. 12. The sale of banana was "0" in January, and thus developers may override cell $D3$ with a seemingly simpler formula "=B3", which causes an anomaly because $D3$'s value will no longer be updated automatically when cell $C3$'s value changes. Similar anomalies may occur to cells $D4$, $D7$ and $D8$. Besides, cell $D6$ also contains an anomaly (error) because developers miswrote its operator "+" as "−". For this example, UCheck reported cells $D3$, $D4$, $D7$ and $D8$ as anomalies correctly. However, it missed cell $D6$ since its unit information does not violate any rule in UCheck.

We then modify the example in Fig. 12 slightly and obtain new examples in Fig. 13 and Fig. 14. Example 1-2 omitted the values in cells $C3$, $C4$, $B7$ and $B8$ since they are all "0". Such omissions are reasonable for this particular example and has been very common in practical spreadsheets, especially, in spreadsheets

	A	B	C	D	E	F	G
1		**Fruit**					
2	**Month**	**Apple**	Brand1	Brand2	**Banana**	Brand1	**Total**
3	January	=C3+D3	200	0	=F3	230	=B3+E3
4	February	=C4+D4	220	20	=F4	160	=B4+E4
5	March	=C5+D5	220	100	=F5	200	=B5+E5
6	April	=C6+D6	160	150	=F6	300	=B6+E6
7	May	=C7+D7	0	180	=F7	170	=B7+E7
8	June	=C8+D8	0	210	=F8	210	=B8+E8

Figure 15: Example 1-4: A spreadsheet whose multi-level headers are placed in the same row

with sparse contents. However, after such a slight change, UCheck will no longer be able to assign header information to blank cells like $C3$, and as a result it cannot detect any anomaly in this spreadsheet. The other example 1-3 omitted two higher-level headers, "Month" and "Fruit", as compared to Example 1-1, and now no higher-level header can be assigned to cells. As a result, UCheck cannot detect any anomaly in this spreadsheet similarly. Examples 1-2 and 1-3 suggest that false negatives can be serious for UCheck when the concerned spreadsheets undergo slight changes that do not follow UCheck's rules (implicit assumptions). For example, consider the 100 worksheets we sampled from the EUSES Spreadsheet Corpus. About 50% of them suffered the blank cell problem and 28% suffered the missing higher-level header problem.

Another common pattern in practical spreasheets is that multi-level headers are specified in the same row or column but with different fonts or styles, as Example 1-4 in Fig. 15. In the example, each kind of fruit may have one or more brands. For example, fruit "Apple" has two brands, "Brand1" and "Brand2", while fruit "Banana" has only one brand, "Brand1". Then the formulas of cells in column B should be the sum of corresponding cells in columns C and D, and the formulas of cells in column E should be a direct reference to corresponding cells in column F. Besides, the formulas of cells in column G should be the sum of corresponding cells in columns B and E. In fact, the spreadsheet in Fig. 15 is well-formed and contains no anomaly. However, UCheck pointed out all cells in columns E and G as anomalies because it could not infer correct headers for them in such a case. This example suggests that false positives can be serious for UCheck when the concerned spreadsheets have such features that

	A	B	C	D	E
1			**Fruit**		
2	**No.**	**Month**	**Apple**	**Banana**	**Total**
3	1	January	200	0	=C3+D3
4	=A3+1	February	220	0	=C4+D4
5	=A4+1	March	220	200	=C5+D5
6	=A5+1	April	160	300	=C6+D6
7	=A6+1	May	0	170	=C7+D7
8	=A7+1	June	0	210	=C8+D8

Figure 16: Example 1-5: A spreadsheet with auto-number cells

cause UCheck unable to correctly infer their header information. For example, about 26% of the worksheets sampled from the EUSES Spreadsheet Corpus suffered this multi-level header problem.

From the preceding examples, we consider that the effectiveness of UCheck depends much on how spreadsheets are prepared, e.g., whether they contain few blank cells, whether they have proper headers, whether these headers are placed in right places, and so on. However, in our study, we found that many spreadsheets do not follow such strong restrictions in their preparations. Many practical spreadsheets are complex and contain casual carelessness, and this caused trouble to the header inference in UCheck, which then behaved unsatisfactorily in spreadsheet anomaly detection. Hence, how to obtain the correct header information reliably may be the key to improve the performance of UCheck. Researchers can find more patterns of header settings from a large amount of spreadsheet subjects and apply them to the header inference. Furthermore, the rules to infer headers should be adaptive to specific features of spreadsheets.

6.1.2. Limitations of Dimension

Dimension has similar limitations as UCheck. For Example 1-4 in Fig. 15, Dimension also reported all cells in column G as anomalies because it inferred incorrect dimension information based on misleading headers. Dimension reported that the formulas in column G tried to add two different dimensions, and this is considered illegal in Dimension.

Besides trouble from header information, this technique's dimension inference itself is also unreliable and can easily cause many false positives. Example

	A	B	C	D	E
1	**Hour**	**Speed**	**Length**	**Distance**	
2	20	14	=A2*B2	1000	=D2-C2
3	21	15	=A3*B3	1200	=D3-C3
4	20	13	=A4*B4	1000	=D4-C4
5	23	14	=A5+B5	1100	=D5-C5
6	25	15	=A6/B6	1250	=D6-C6
7	26	16	=A7*B7	1050	=D7-C7

Figure 17: Example 2: A spreadsheet with clear dimension information

1-5 in Fig. 16 presents such a scenario. Column A "No." represents the number of each Record. Many developers are inclined to use formulas, like "=A3+1" for cell $A4$, to auto-number these cells. Even if Dimension could infer correct headers, all formula cells in column A were still reported as anomalies. Dimension considered that it is illegal for the formula in cell $A4$ to add one cell $A3$, which indicated a certain dimension, and "1", which indicated no dimension, together. However, this case is very common in reality, and also for our studied three corpora. For example, about 28% of the worksheets sampled from the EUSES Spreadsheet Corpus were reported to contain a large number of such cases during the dimension inference, in particular, four of them shared exactly the same pattern (auto-numbered cells) as in Example 1-5.

Another example in Fig. 17 exposes other limitations of Dimension. Column A represents the time (in hours), column B represents the speed and column C represents the length, which is the product of time and speed. Column D represents the total distance and column E represents the remaining distance, which is the difference between distance and length. Marked cells were anomalies reported by Dimension. In fact, cells $C5$ and $C6$ contain anomalies because their common operator "*" was miswrote as "+" and "/", respectively. However, Dimension could only detect cell $C5$ but omitted cell $C6$ because it can only determine the addition of two different dimensions to be illegal, but cannot judge whether the division of them is illegal according to its rules. Besides, it misreported all cells in column E as anomalies, and this caused a large number of false positives. This is because Dimension derived different dimensions for labels "Length" and "Distance", whose dimensions are the same actually, and

	A	B	C	D
1		**Fruit**		
2	**No.**	**Apple**	**Banana**	**Total**
3	1	200	0	=B3+C3
4	=A3+1	220	0	=B4+C4
5	=A4+1	220	200	=B5+C5
6	=A5+1	160	300	=B6+C6
7	=A6+1	0	170	=B7+C7
8	=A7+1	0	210	=B8+C8

Figure 18: Example 1-6: A spreadsheet with auto-number cells adjacent to plain data

then determined the formulas of cells in column E, which subtract one dimension from another "different" dimension, to be illegal.

From these examples above, we can find two factors affecting the performance of Dimension. On one hand, inferring the header information can be unreliable and one should try to improve it as suggested in Section 6.1.1. On the other hand, the deficiency of dimension knowledge may influence the derivation of dimensions. Hence, we suggest researchers to enrich the dimension knowledge to adapt to more spreadsheets from different domains.

6.1.3. Limitations of AmCheck

Example 1-5 in Fig. 16 also discloses a weakness of AmCheck. If an auto-numbered column, such as column A, is adjacent to other columns with pure data, AmCheck would probably extract incorrect cell arrays and then cause further false positives. Example 1-6 in Fig. 18 presents such a case. AmCheck extracted cell ranges "[A4:C4]", "[A5:C5]", and so on, as five cell arrays by mistake, which should be "[A3:A8]" instead. This is because the formulas in cells in column "A" refer to other cells in the same column, and thus AmCheck regarded each of them as one cell in another row-based cell array, which comprises consecutive cells in a row, according to its built-in cell array detection rules. With such incorrectly extracted cell arrays, AmCheck further misreported all cells in the range of "[B3:C8]" as anomalies (missing formula smells). For such cases, AmCheck's cell array extracting rules need improvement. For example, 19% of the worksheets sampled from the EUSES Spreadsheet Corpus suffered this formula-data adjacency problem.

	A	B	C	D	E
1		**Fruit**			
2	**Month**	**Apple**	**Banana**		**Total**
3	January	200	0		=SUM(B3:C3)
4	February	220	0		=SUM(B4:C4)
5	March	220	200		=SUM(B5:D5)
6	April	160	300		=SUM(B6:D6)
7	May	0	170		=SUM(B7:D7)
8	June	0	210		=SUM(B8:D8)

Figure 19: Example 1-7: A spreadsheet for which AmCheck inferred an incorrect formula pattern

AmCheck may also induce false positives and false negatives when recovering or synthesizing formula patterns for cell arrays, even if it has extracted correct cell arrays, such as in Example 1-7 in Fig. 19. AmCheck extracted the correct cell array "[E3:E8]", but reported cells $E3$ and $E4$ as anomalies incorrectly. This is because AmCheck inferred "=SUM(Bi:Di)" as this cell array's formula pattern, rather than "=SUM(Bi:Ci)", which is more suitable. AmCheck did so since it observed more occurrences of the former, but did not notice that the former refers to meaningless cells (blank cells). For example, 15% of the worksheets sampled from the EUSES Spreadsheet Corpus contain such formulas that refer to incorrect (blank) cells.

We suggest improving AmCheck from two aspects, cell array extraction and formula pattern inference. AmCheck may extract incorrect cell arrays for some specific patterns, such as in Example 1-6. We suggest researchers to ameliorate the existing cell array extraction rules. At least, one may add special processing for these specific patterns. AmCheck may also report incorrect anomalies even with correctly extracted cell arrays for which it infers incorrect formula patterns. Research may consider some properties of spreadsheets when inferring formula patterns. For example, the header information may help suggest more suitable formula patterns, as in Example 1-7.

6.2. Anomaly Detection for Different Spreadsheet Categories

Spreadsheets can belong to different categories, affecting their topics and structures. For example, some spreadsheets are mainly for recording data and

Table 6: Some properties of spreadsheets from different categories in the EUSES corpus (Worksheets: the numbers of worksheets with formulas for different categories; Cells: average numbers of cells in each worksheet; String cells: average numbers of string cells in each worksheet; Numerical cells: average numbers of numerical cells in each worksheet; Formula cells: average numbers of formula cells in each worksheet)

	Worksheets	Cells	String cells	Numerical cells	Formula cells
cs101	8	104	26	35	24
database	566	1,971	606	282	199
fibly	4	122	47	15	17
financial	926	1,385	79	90	108
forms3	30	3,169	355	151	12
grades	611	1,328	115	266	115
homework	602	1,077	89	151	117
inventory	804	1,489	166	132	138
jackson	0	-	-	-	-
modeling	467	1,211	247	187	185
personal	36	4,599	1,197	2,656	607

thus contain few formulas, while others care more about data statistics and thus contain a large number of mathematic formulas. Therefore, we finally compare the effectiveness of the three techniques in detecting anomalies when spreadsheets are projected to different categories. All spreadsheets in the Hawaii Kooker Corpus share almost the same topics, and thus we ignored this corpus. Besides, the Enron Spreadsheet Corpus has the categories of its spreadsheets mixed and hard to distinguish, and we also had to leave it. Fortunately, the EUSES Spreadsheet Corpus naturally provides its category information, and therefore we further studied this corpus.

Although the EUSES Spreadsheet Corpus provides 11 categories naturally, we are curious about the variance of spreadsheets from these different categories.

Table 7: The projected results on seven spreadsheet categories in the EUSES corpus for the three techniques

	Ground truth	AmCheck		UCheck		Dimension	
		Reported	True	Reported	True	Reported	True
cs101	10	6	6	6	0	6	0
database	343	107	100	137	0	1,156	3
financial	107	6	6	7	1	387	22
grades	82	93	60	1	0	686	18
homework	19	28	1	30	1	76	2
inventory	32	67	10	43	0	185	13
modeling	18	25	5	77	0	101	0

So, we collected some properties of spreadsheets from different categories and present the results in Table 6. Column "Worksheets" denotes the numbers of worksheets with formulas for different categories. Column "Cells" denotes the average numbers of cells in each worksheet, including blank cells. Columns "String cells" and "Numerical cells" denote the average numbers of string cells and numerical cells in each worksheet, respectively. Column "Formula cells" denotes the average numbers of formula cells. We can observe that the scales of spreadsheets from different categories vary significantly. Spreadsheets from "cs101" and "fibly" contain just more than 100 cells per worksheet, while spreadsheets from "forms3" and "personal" contain more than 3,000 cells per worksheet. The proportions of "string cells", "numerical cells" and "formula cells" for each worksheet also differ greatly for spreadsheets from different categories. For example, the most cells of spreadsheets from "personal" are numerical ones, but from "database" the most cells are string ones. We can also observe that spreadsheets from "financial" and "forms3" contain plenty of blank cells (more than 80%). It is clear that spreadsheets from different categories indeed have distinctive features or properties, which may relate to their category names.

The 100 worksheets we sampled from the EUSES Spreadsheet Corpus for

manual inspection were from seven different categories. We present their projection results in Table 7 and make some interesting observations. For AmCheck, its precision is pretty high for spreadsheets from categories "cs101", "database" and "financial" (93.46–100%), but quite low for spreadsheets from categories "homework", "inventory" and "modeling" (3.57–20%). The anomalies detected by UCheck are so few that we cannot obtain any reasonable conclusion. For Dimension, its precision for spreadsheets from categories "financial" and "inventory" is relatively higher (5.68–7.03%) than other categories. On the contrary, its precision for spreadsheets from categories "cs101", "database" and "modeling" is exceptionally low (0–0.26%). It is clear that AmCheck and Dimension both have varying precisions in detecting anomalies for spreadsheets from different categories. Similarly, we also calculated these techniques' recall rates and found that they also vary greatly for different spreadsheet categories (5.26–73.17% for AmCheck and 0–40.63% for Dimension; not calculated for UCheck as it has too few data).

As a conclusion, the three techniques performed quite differently for different spreadsheet categories. We conjecture that spreadsheets from one category might share some common but hidden features that "favor" or "hate" certain spreadsheet anomaly detection technique. Currently, we have tried to disclose properties associated with different categories, but whether and how these properties relate to the performance of spreadsheet anomaly detection techniques needs further research.

7. Threats to validity

We adapted three tools for batch processing of spreadsheets based on existing technique implementations from their original authors. Although this tried to respect original implementations, it itself might introduce other bias. First, the three techniques were implemented by different people in different programming languages, and this might affect our efficiency measurement. Second, UCheck and Dimension connect their core algorithms to file I/O interfaces via

sockets, while AmCheck reads spreadsheet files directly. This might also affect the efficiency measurement. To alleviate such threats, our experimental analyses mainly focus on effectiveness comparison for spreadsheet anomaly detection. Regarding efficiency, we concern mostly the processing time distribution, which is fair for each technique individually, in this study.

There are also threats in our manual inspection. We sampled worksheets and checked them manually to mark anomalies for the ground truth. Deciding whether a cell contains any anomaly can be subjective. To alleviate this threat, we focused our attention particularly on those cells with formulas, cells referred to by formulas and their neighboring cells. These cells are more likely to contain anomalies, and they are also the focus of the three techniques. We attempted to understand the intention of spreadsheet developers by analyzing the concerned formulas and labels. For better confidence, we sometimes compared worksheets across spreadsheets when they share similar topics and structures. Besides, we cross checked the ground truth between two postgraduates and two professors to avoid subjectivity. For other researchers to repeat our experiments, we have prepared a link[3] to access required materials in our study.

8. Related Work

In this work we conducted a study of comparing the performance of three spreadsheet anomaly detection techniques, namely, AmCheck, UCheck and Dimension. The authors of AmCheck have earlier conducted an empirical study on the EUSES Spreadsheet Corpus [14]. However, it has not been compared with other techniques. Besides, our studied subjects are more comprehensive than the EUSES Spreadsheet Corpus only. UCheck and Dimension have been evaluated on only 83 spreadsheets from the EUSES Spreadsheet Corpus [12], which are also much less than our subjects. In their experiments, there was no error

[3]Our experimental setup, sampled worksheets, and instructions on how to access spreadsheet corpora and spreadsheet tools used in our experiments can be obtained from the link: http://cs.nju.edu.cn/changxu/temp/spreadsheets.rar.

reported by both UCheck and Dimension, and this echoes our results in this study. However, UCheck's and Dimension's precisions were reported very high in their experiments, and this does not conform with our results in this work. Since we do not have the details of their experimental setup and procedure, we cannot repeat the same experiments. We conjecture that it is possibly due to different sets of spreadsheets for experiments, as well as the instability of their provided implementations, as we observed from our experiments. Dimension has also been evaluated in work [11], in which only 40 spreadsheets were sampled from the EUSES Spreadsheets Corpus. Unlike our fully automatic processing, they adjusted spreadsheet headers manually after inference, and their reported precision from experiments was higher than ours. We notice such differences in measured performance between existing work and our work in this article. To make our results more useful and convincing, we additionally analyzed underlying reasons why these techniques did not perform well by concrete examples, aiming to give substantial improvement suggestions.

There are also other techniques or tools proposed for avoiding, finding or fixing anomalies in spreadsheets. Hermans et al. [19] adapted the concept of code smell to spreadsheets and presented the concept of formula smell for spreadsheets. They defined metrics for each formula smell for detecting spreadsheet smells automatically. Later, they further proposed a technique for detecting specific data clone smells in spreadsheets [20]. Similarly, Hofer et al. [21] adapted spectrum-based fault-localization techniques to detect spreadsheet errors. Jannach and Schmitz [22] proposed translating spreadsheet checking to a constraint satisfaction problem and using classical diagnosis algorithms to detect errors in spreadsheets. Recently, Abreu et al. [5] composed a catalog of spreadsheet smells with a generic spectrum-based fault localization technique to detect potential faults in spreadsheets automatically.

Besides detecting smells or errors in spreadsheets, some work aims at improving spreadsheet structures to prevent potential errors. For example, Luckey et al. [24] presented a model-based approach to supporting correct spreadsheet evolution. Cunha et al. [13] proposed extracting relational models from

spreadsheets and embedding them back to spreadsheets for building a reliable spreadsheet programming environment. Other work supports spreadsheet developers or users in the maintenance process. For example, Harutyunyan et al. [16] presented an algorithm for automatically identifying differences between spreadsheet versions for easier maintenance. Badame and Dig [8] proposed seven measures for refactoring spreadsheet formulas for easier maintenance in future.

9. Conclusion

In this article, we have empirically evaluated and compared three influential spreadsheet anomaly detection techniques. To the best of our knowledge, this is the first empirical study of spreadsheet anomaly detection on large-scale and comprehensive subjects. We have studied the three techniques in the precision, recall rate, efficiency and scope, and observed that AmCheck has the best precision and recall rate, but UCheck and Dimension can find different types of anomalies.

The three techniques have different strengths and limitations. They are all amenable to automation, which is appreciated. Besides, they do not rely on any oracle or test case, and this makes them applicable to potentially all existing spreadsheets. Some noticeable limitations include: incomplete preprocessing rules in these techniques, not scalable to large-scale spreadsheets with complex formulas, and so on. Based on our results, we have given some suggestions for improving the three techniques. For example, AmCheck is suggested to enhance its rules for improving the accuracy of cell array extraction. UCheck and Dimension are suggested to consider more complex spreadsheet layouts to infer headers better.

Besides, we have used some examples to illustrate common patterns that cannot be well handled by the three techniques, which may inspire future spreadsheet research. We have also discussed whether spreadsheets from different categories are subject to certain properties, and explored whether this affects the

three techniques' performance preliminarily. We are interested in whether such properties can be exploited to enhance or extend capabilities for spreadsheet anomaly detection, which is a potential research direction.

Acknowledgement

The authors wish to thank anonymous reviewers for their valuable comments on improving this article. This work was supported in part by the National Basic Research 973 Program (Grant No. 2015CB352202), and National Natural Science Foundation (Grant Nos. 61472174, 91318301, 61321491) of China, and by the Research Grants Council (611811) of Hong Kong. All correspondence should be addressed to Chang Xu (email: changxu@nju.edu.cn, tel: +86-89680919, fax: +86-83593283).

References

[1] Abraham, R., Erwig, M., 2004. Header and unit inference for spreadsheets through spatial analyses. In: Visual Languages and Human Centric Computing, 2004 IEEE Symposium on. IEEE, pp. 165–172.

[2] Abraham, R., Erwig, M., 2007. Ucheck: A spreadsheet type checker for end users. Journal of Visual Languages & Computing 18 (1), 71–95.

[3] Abraham, R., Erwig, M., 2008. Test-driven goal-directed debugging in spreadsheets. In: Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on. IEEE, pp. 131–138.

[4] Abraham, R., Erwig, M., 2009. Mutation operators for spreadsheets. Software Engineering, IEEE Transactions on 35 (1), 94–108.

[5] Abreu, R., Cunha, J., Fernandes, J. P., Martins, P., Perez, A., Saraiva, J., 2014. Smelling faults in spreadsheets. In: Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on. IEEE, pp. 111–120.

[6] Aurigemma, S., Panko, R. R., 2010. The detection of human spreadsheet errors by humans versus inspection (auditing) software. arXiv preprint arXiv:1009.2785.

[7] Außerlechner, S., Fruhmann, S., Wieser, W., Hofer, B., Spork, R., Muhlbacher, C., Wotawa, F., 2013. The right choice matters! smt solving substantially improves model-based debugging of spreadsheets. In: Quality Software (QSIC), 2013 13th International Conference on. IEEE, pp. 139–148.

[8] Badame, S., Dig, D., 2012. Refactoring meets spreadsheet formulas. In: Software Maintenance (ICSM), 2012 28th IEEE International Conference on. IEEE, pp. 399–409.

[9] Burnett, M., Erwig, M., 2002. Visually customizing inference rules about apples and oranges. In: Human Centric Computing Languages and Environments, 2002. Proceedings. IEEE 2002 Symposia on. IEEE, pp. 140–148.

[10] Chambers, C., Erwig, M., 2008. Dimension inference in spreadsheets. In: Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on. IEEE, pp. 123–130.

[11] Chambers, C., Erwig, M., 2009. Automatic detection of dimension errors in spreadsheets. Journal of Visual Languages & Computing 20 (4), 269–283.

[12] Chambers, C., Erwig, M., 2010. Reasoning about spreadsheets with labels and dimensions. Journal of Visual Languages & Computing 21 (5), 249–262.

[13] Cunha, J., Mendes, J., Saraiva, J., Visser, J., 2014. Model-based programming environments for spreadsheets. Science of Computer Programming 96, 254–275.

[14] Dou, W., Cheung, S.-C., Wei, J., 2014. Is spreadsheet ambiguity harmful? detecting and repairing spreadsheet smells due to ambiguous computation. In: Proceedings of the 36th International Conference on Software Engineering. ACM, pp. 848–858.

[15] Fisher, M., Rothermel, G., 2005. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In: ACM SIGSOFT Software Engineering Notes. Vol. 30. ACM, pp. 1–5.

[16] Harutyunyan, A., Borradaile, G., Chambers, C., Scaffidi, C., 2012. Planted-model evaluation of algorithms for identifying differences between spreadsheets. In: Visual Languages and Human-Centric Computing (VL/HCC), 2012 IEEE Symposium on. IEEE, pp. 7–14.

[17] Hermans, F., Murphy-Hill, E., 2015. Enron's spreadsheets and related emails: A dataset and analysis. In: Proceedings of the 37th International Conference on Software Engineering-Volume 2. IEEE Press, pp. 7–16.

[18] Hermans, F., Pinzger, M., Deursen, A. v., 2012. Detecting and visualizing inter-worksheet smells in spreadsheets. In: Proceedings of the 2012 International Conference on Software Engineering. IEEE Press, pp. 441–451.

[19] Hermans, F., Pinzger, M., van Deursen, A., 2012. Detecting code smells in spreadsheet formulas. In: Software Maintenance (ICSM), 2012 28th IEEE International Conference on. IEEE, pp. 409–418.

[20] Hermans, F., Sedee, B., Pinzger, M., Deursen, A. v., 2013. Data clone detection and visualization in spreadsheets. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, pp. 292–301.

[21] Hofer, B., Riboira, A., Wotawa, F., Abreu, R., Getzner, E., 2013. On the empirical evaluation of fault localization techniques for spreadsheets. In: Fundamental Approaches to Software Engineering. Springer, pp. 68–82.

[22] Jannach, D., Schmitz, T., 2014. Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. Automated Software Engineering, 1–40.

[23] Jannach, D., Schmitz, T., Hofer, B., Wotawa, F., 2014. Avoiding, finding and fixing spreadsheet errors–a survey of automated approaches for spreadsheet qa. Journal of Systems and Software 94, 129–150.

[24] Luckey, M., Erwig, M., Engels, G., 2012. Systematic evolution of model-based spreadsheet applications. Journal of Visual Languages & Computing 23 (5), 267–286.

[25] Panko, R., 2006. Facing the problem of spreadsheet errors. Decision Line 37 (5), 8–10.

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