WARDER: Towards Effective Spreadsheet Defect Detection by Validity-based Cell Cluster Refinements

Yicheng Huang[a,b], Chang Xu[a,b,*], Yanyan Jiang[a,b], Huiyan Wang[a,b], Da Li[a,b]

[a]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
[b]*Department of Computer Science and Technology, Nanjing University, Nanjing, China*

Abstract

Nowadays spreadsheets are very popular and being widely used. However, they can be prone to various defects and cause severe consequences when end users poorly maintain them. Our research communities have proposed various techniques for automated detection of spreadsheet defects, but they commonly fall short of effectiveness, either due to their limited scope or relying on strict patterns. In this article, we discuss and improve one state-of-the-art technique, CUSTODES, which exploits spreadsheet cell clustering and defect detection to extend its scope and make its detection patterns adaptive to varying spreadsheet styles. Still, CUSTODES can be prone to problematic clustering when accidentally involving irrelevant cells, leading to a largely reduced detection precision. Regarding this, we present WARDER to refine CUSTODES's spreadsheet cell clustering based on three extensible validity-based properties. Experimental results show that WARDER could improve the precision by 19.1% on spreadsheet cell clustering, which contributed to a precision improvement of 23.3~24.3% for spreadsheet defect detection, as compared to CUSTODES (F-measure increased from 0.71 to 0.79~0.82). WARDER also exhibited satisfactory results on another practical large-scale spreadsheet corpus VEnron2, improving the defect detection precision by 10.7~21.2% over CUSTODES.

Keywords: Cell clustering, Defect detection, Validity property

---

*Corresponding author

Email addresses: njuhuangyc@outlook.com (Yicheng Huang), changxu@nju.edu.cn (Chang Xu), jyy@nju.edu.cn (Yanyan Jiang), cocowhy1013@gmail.com (Huiyan Wang), njulida@outlook.com (Da Li)

1. Introduction

Nowadays spreadsheets, as a popular application of end-user software, have been widely used in data storage, financial analyses, and quality control [49], with over 750 million users for representative Microsoft Excel alone [11]. However, in spite of this popularity, spreadsheets are found to be error-prone [46], and can cause catastrophic consequences, e.g., massive financial loss [1]. Such errors can exhibit in various forms, spreading from data cells to formula cells in spreadsheets, and existing studies [45] have suggested that the latter could commonly be the root causes. Within the scope of this article, we name the errors in formula cells by *defects* in spreadsheets (similar to bugs in traditional programs), and focus on effective techniques for detecting them automatically.

Detecting spreadsheet defects can be non-trivial. First, spreadsheets are typically maintained by non-programmer end-users (usually financial experts). Their behaviors can involve various unprofessional operations (from the point of view of programmers), e.g., overwriting a formula with a plain value or replacing it with another plausible formula, simply for ad hoc purposes (e.g., to fix a single incorrect data) [45]. Such operations can easily lead to a boosting of spreadsheet defects, since the original computational semantics is overwritten or altered accidentally. Second, auditing or tracking services are typically unavailable for common spreadsheet usages [37], and this fact results in the loss of clues on how spreadsheet defects have been introduced and where they are located. Third, semantic relationships among spreadsheet cells are typically hidden, and this fact results in the difficulties in automatically reasoning over spreadsheets for potential defects.

To address these challenges, our research community has proposed various spreadsheet defect detection techniques. For example, they rely on table header information to infer type inconsistencies in formula references (e.g., UCheck [4] and Dimension [12]), exploit specific patterns (e.g., rectangles) to recognize missing or inconsistent formulas (e.g., AmCheck [19] and CACheck [20]), or use adaptive learning to detect anomalies in formula cells (e.g., CUSTODES [13], Melford [52], and ExceLint [9]).

However, although producing promising detection results, these spreadsheet defect detection techniques have their own weaknesses. For the first category (type-based techniques), their inconsistency inference is vague and focuses on a limited scope, resulting in both low precision and recall results [59]. For the second category (pattern-based techniques), their relied patterns can be strict and precise for capturing certain features in spreadsheets (thus achieving a high precision, e.g., 71.9% for AmCheck and 86.8% for CACheck [20]), but not adaptable to varying styles in different spreadsheets (leading to a compromised recall, e.g., 60.3% for AmCheck and 71.0% for CACheck [20]). For the third category (learning-based techniques), they have become increasingly popular in recent years due to their adaptive learning abilities. We take CUSTODES [13] for example, as it has ever been considered to be the "best automated error finder" [9]. CUSTODES clusters spreadsheet cells according to their formula semantics (e.g., by abstract syntax trees), and at the same time restricts the impact caused by the dissimilarity of defective formulas and varying styles across spreadsheets by learning their features. By doing so, it largely increases the recall (up to 80% [13]), but at the same time compromises the precision (down to 65% [13]) when clustering irrelevant cells together.

Regarding the weaknesses of these pioneering techniques, we in this work propose a technique WARDER, based on the success of CUSTODES's adaptive learning of varying styles (features) across spreadsheet, but improve over its weakness when clustering both relevant and irrelevant cells together for spreadsheet defect detection. Our key observation is that the cell clustering, if accidentally involving irrelevant cells, would largely compromise the effectiveness of the technique's later defect detection (e.g., reducing the precision). Therefore, our main efforts in WARDER focus on automatically refining the cell clustering to make it more robust by filtering out irrelevant cells, and enhancing the refinements with relevance-oriented cell retrieval, while preserving CUSTODES's original adaptive learning ability.

Our targeted refinements are three-folded: (1) *Single-cell validity.* When adding cells into a cluster, WARDER would reject those data cells that can become invalid if cast to formulas for unification with other cells already in this cluster. For example, when the data in a cell is replaced by a formula for unification with other formula cells in the same cluster, this formula is found to be invalid for calculation (e.g., causing a wrong reference or citing a wrong place). Then this new data cell should not be added into this cluster. (2) *Multi-cell validity.* WARDER would also reject those data cells from being added into a cluster if these cells, once added, would violate important properties of the cells already existing in this cluster. For example, the references of existing cells in a cluster do not overlap each other before adding a new data cell, but this property would be violated if the cell is added and its contained data is replaced by a formula for unification. Then this new cell should also not be added into this cluster. (3) *Whole-cluster validity.* Other than cell-level validations, WARDER also examines the validity for all cells in a cluster as a whole. Since each cluster is formed for unifying common computational semantics and identifying few anomalies as defects in spreadsheets, it should be able to find a formula unifiable with most cells in this cluster. Otherwise, the cluster itself is not qualified since it lacks the common computational semantics (denying its original purpose of existence) [20], and should be canceled to avoid later misbehavior in spreadsheet defect detection.

With these dedicated refinements, WARDER aims for better cell clustering by filtering out irrelevant cells, thus improving its effectiveness in detecting spreadsheet defects. With different instantiations of these validity-based refinements, we propose two versions of WARDER (WARDER-ori and WARDER-ext, introduced later in Section 3). Both of them exhibit clear merits when compared to their predecessor CUSTODES, as well as to other existing spreadsheet defect detection techniques, including UCheck, Dimension, AmCheck, and CACheck. For example, regarding CUSTODES's own benchmark of 291 worksheets (basic units in spreadsheets) selected from the EUSES corpus [22], WARDER achieved a significant improvement as compared to CUSTODES. For cell clustering, WARDER boosted over 80% worksheets (80.1% for WARDER-ori and 81.6% for WARDER-ext) and 90% clusters (93.2% for WARDER-ori and 93.9% for WARDER-ext), either by improving the precision or already reaching the upper limit of 100%, with only a small sacrifice of less than 3% on the average recall (-2.9% for WARDER-ori and -2.4% for WARDER-ext). For defect detection, WARDER-ori's and WARDER-ext's cell clustering contributed substantially to its defect detection by achieving 23.3% and 24.3% precision improvements, respectively, as compared to CUSTODES. Combining the recall, this leads to an increase of F-measure on defect detection from 0.71 (for CUSTODES) to 0.79 (for WARDER-ori) and 0.82 (for WARDER-ext). Compared to other spreadsheet defect detection techniques, WARDER-ori and WARDER-ext also outperformed them on average by a precision of 88.2% and 89.2% and a recall of 72.0% and 75.4%, respectively, against the precision of 0.5–72.7% and the recall of 0.1–68.7% for other techniques. Besides, regarding an even larger-scale practical corpus VEnron2 [56], which contains 1,609 versioned groups refined from original 79,983 worksheets in the Enron corpus [28], both WARDER-ori and WARDER-ext exhibited their unique superiority over their predecessor CUSTODES, improving the defect detection precision by 10.7% and 21.2%, respectively.

In summary, this article makes the following contributions:

- We proposed the WARDER framework, which refines CUSTODES's cell clustering by three validity-based refinements and a cell retrieval enhancement for improving the effectiveness of spreadsheet defect detection.
- We realized the WARDER framework into two versions (WARDER-ori and WARDER-ext) with different instantiations of validity properties.
- We evaluated WARDER-ori and WARDER-ext with both existing benchmark spreadsheets and a practical large-scale spreadsheet corpus, and compared them to existing spreadsheet defect detection techniques.

Compared to its preliminary version (i.e., its conference paper [38]), the work in this article makes refinements and extensions to the original WARDER framework. In particular, WARDER-ext realizes more refinement instantiations on top of WARDER-ori along the three validity properties, and adds a validity-oriented cell retrieval enhancement into the framework. Besides, the work also enhances the evaluation with new experiments for WARDER-ext and new analyses for the cell clustering effectiveness and correlation study.

The remainder of this article is organized as follows. Section 2 introduces necessary background knowledge and terminologies on spreadsheet and its defect detection. Section 3 presents our WARDER framework with its extensible validity properties, and elaborates on its two versions, respectively. Section 4 experimentally evaluates WARDER with practical spreadsheets and compares it with existing spreadsheet defect detection techniques. Finally, Section 5 discusses the related work in recent years, and Section 6 concludes this article.

2. Background

In this section, we introduce necessary background knowledge on spreadsheet and its defect detection, as well as key terminologies (e.g., feature) used in the CUSTODES technique (and also in our WARDER technique).

2.1. Spreadsheet

Spreadsheet. A *spreadsheet* refers to a stand-alone spreadsheet file in a file system. For example, in Microsoft Excel, each opened Excel file is considered as a spreadsheet, which is named following the convention like A.xls or B.xlsx.

Worksheet. A *worksheet* refers to a single sheet page inside a spreadsheet. Normally, a spreadsheet contains multiple worksheets for different data storage and calculation purposes.

Cell. A worksheet contains multiple cells, each of which is referred to by a column number (e.g., A, B, and C) and a row number (e.g., 1, 2, and 3). A *cell* is the minimal information piece in a worksheet, which can contain a (numeric) data[1] (e.g., 200 or 11.5), formula (e.g., A1 + B2 or SUM(A1, C3)), or text string (e.g., "Fruit" or "----") for formatting purposes (e.g., as a table header or delimiter). In the scope of this article, we are interested in the former two, as they are also the main focus of many spreadsheet defect detection techniques. Regarding this, the former two cell types are also referred to as *data cell* and *formula cell*, respectively. Normally, a data cell contains a numeric data, which typically serves as the input to other formula cells, and a formula cell contains a formula, which can automatically update its corresponding value as calculated from other (data or formula) cells whose values serve as the input to this formula.

[1]Some work considers text strings also as data, i.e., their data cells can contain both numbers and text strings. These slightly different definitions would not affect our subsequent discussions.

Reference. References apply only to formula cells. When the data contained in a data cell serves as the input to a formula cell, we say that this formula cell (or simply this formula) *references* this data cell and the latter is referred to as a *referenced cell*.

2.2. Defect Detection

As mentioned earlier, spreadsheets can contain various defects in their formula cells. In the scope of this article, we focus on two major defect types that are dedicatedly targeted for each particular worksheet by existing spreadsheet defect detection techniques. Of course, there are also other defect types, e.g., table clone smells, which are detectable across worksheets [18]. Since they do not fall in our focus in this article, also not relevant to WARDER's comparison over CUSTODES, we do not cover them here. In the following, we introduce the two major defect types in spreadsheets, namely, *missing formula defect* and *inconsistent formula defect*.

Missing formula defect. This defect occurs when a cell contains a data instead of a formula, whose calculation result is supposed equal to this data value. For example, in a worksheet, suppose that cell C3 was originally defined by formula A1 + B2. Due to some unknown reason, its user overwrote this formula by a plain value 5. It is possible that this formula happened to have this value equal to its calculation result when the user conducted the overwriting. Then in this case, this defect became hidden for now, but would be triggered later when cell A1 or B2 has its value updated, since this value of 5 would no longer be automatically updated later.

Inconsistent formula defect. This defect occurs when a cell contains a formula different from those in its surrounding formula cells, but in fact it should not. For example, suppose that a column (e.g., C) of cells calculates the sum of its two left columns of corresponding cells (e.g., C1 = A1 + B1, C2 = A2 + B2, and so on). One formula might be mistakenly written as C2 = A2 − B2 or C2 = A2, but this defect could be hidden if cell B2 happened to contain a value of zero, although it would be triggered later when cell B2 has its value updated.

Such spreadsheet defects may not immediately trigger any visible error in concerned data values as explained above. Therefore, some pieces of work consider them as *smells* or *anomalies* [30], depending on their considered severity levels. Nevertheless, detecting such defects for timely fixing is important, as spreadsheet users are typically not programming experts and not sensible to such hidden defects, which can easily grow into catastrophic financial losses in the near future.

2.3. CUSTODES Technique

Our WARDER builds on CUSTODES [13]. The kernel part of CUSTODES is a clustering algorithm, which clusters together those cells of similar computational semantics. CUSTODES decides similarity by the notion of *feature*. Its considered features include both key ones (named *strong features*) and others (*weak features*). By strong features, CUSTODES forms clusters of cells that follow the same computational semantics. By weak features, CUSTODES adjust the formed clusters to allow their cells to suffer some noises, thus making possible to include defects into clusters for later detection. While this cluster-forming and -adjusting process will be further explained later when we integrate CUSTODES into WARDER, we introduce more details about strong feature and weak feature below.

Strong feature. CUSTODES considers two strong features in its cell clustering, namely *abstract syntax tree* and *cell dependency tree*, both of which apply to formula cells. The former models the computational semantics of a formula cell, while the latter represents how a formula cell depends on its referenced cells. CUSTODES considers the two features important in deciding whether two formula cells should belong to the same cluster.

Weak feature. Some cells suffering missing formula or inconsistent formula defects cannot be clustered successfully if one considers strong features *only*, since these cells' computational semantics are unexpectedly affected. CUSTODES adjusts its initially formed clusters to allow such cells by considering weak features, whose examples include cell address (i.e., row/column number), label (i.e., table header), and layout (e.g., font size and color). The cells that fail in the similarity comparison on strong features but win in that on weak features could still be clustered. By doing so, CUSTODES gains opportunities of looking into possible defects in these cells.

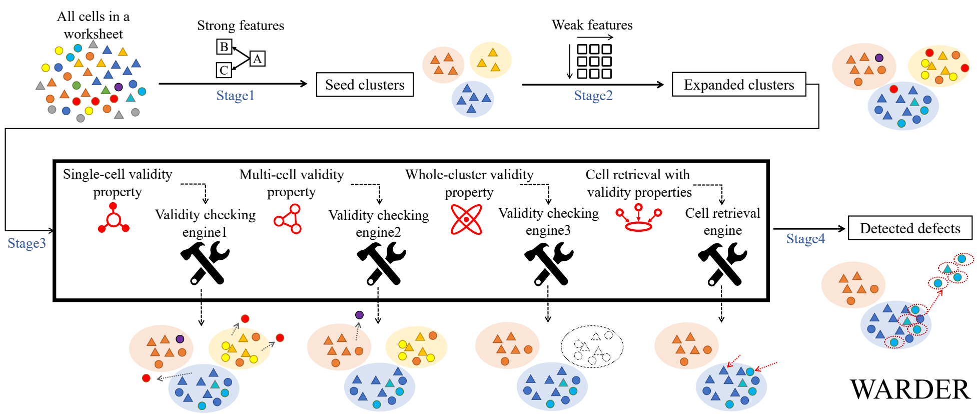


Figure 1: Workflow of the WARDER technique.

3. WARDER Technique

In this section, we present our WARDER technique for effective spreadsheet defect detection, in particular focusing on missing formula and inconsistent formula defects.

WARDER aims for refining CUSTODES's cell clustering for better defect detection effectiveness. Based on our previous work [38], we in this article present two WARDER versions, namely, WARDER-ori and WARDER-ext. The former is almost our previous work but with more details, while the latter extends the former with three refinements of more instantiations plus a cell retrieval enhancement. Both WARDER versions follow the same validity-property based framework, whose workflow is shown in Fig. 1.

In the following, we first introduce WARDER's workflow and its integration with CUSTODES. Then we present a motivating example to illustrate CUSTODES's limitations. After that, we propose WARDER and explain how its three validity-based refinements address the analyzed limitations. Finally, we revisit the motivating example to conclude WARDER's effectiveness.

3.1. WARDER Workflow

As shown in Fig. 1, WARDER, with CUSTODES integrated, follows a four-stage workflow to cluster relevant cells in a worksheet together, and then detect defects in each formed and refined cluster. Among these stages, the third stage (illustrated with a bold black rectangle) is the main work WARDER additionally conducts over what CUSTODES does for refining cell clusters for better defect detection.

First, WARDER uses CUSTODES to form an initial set of seed clusters, each of which contains cells of similar strong features (i.e., abstract syntax trees and cell dependency trees, as discussed earlier). This stage is for each initial cluster to own common computational semantics.

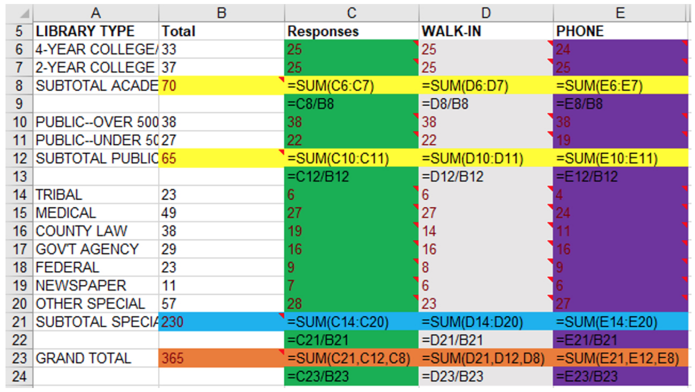
Second, WARDER uses CUSTODES to expand each seed cluster by adding remaining cells that were left from the first stage, as long as these added cells share similar weak features (e.g., cell address, label, and layout, as discussed earlier) with those already in the cluster. This stage is for retrieving back those cells that were left from seed clusters due to the their contained faulty formulas or varying styles across tables in the worksheet.

Third, WARDER refines all formed clusters by squeezing out those cells that violate three *validity properties* (i.e., single-cell, multi-cell, and whole-cluster ones, to be discussed later) associated with these clusters. In addition, WARDER also retrieves previously missed relevant cells back into these clusters, according to whether they follow the validity properties. This stage is for improving the quality of cell clustering by examining the validity of involved cells and formed clusters.

Fourth, WARDER uses CUSTODES to classify anomalies in refined cell clusters, and reports them as defects to spreadsheet users. This stage is for detecting defects (i.e., faulty cells that contain missing formula or inconsistent formula defects) in refined cell clusters with common computational semantics.

3.2. Motivating Example

Before we go into details of WARDER's refinements for improving CUSTODES's cell clustering, we use a motivating example to illustrate CUSTODES's limitations in adjusting its initial formed cell clusters. CUSTODES's original purpose is to retrieve those missed relevant cells back into initial clusters in order to improve its recall in spreadsheet defect detection. However, its retrieval is *aggressive* in that it could involve quite a few irrelevant cells, which instead impact the precisions of both its cell



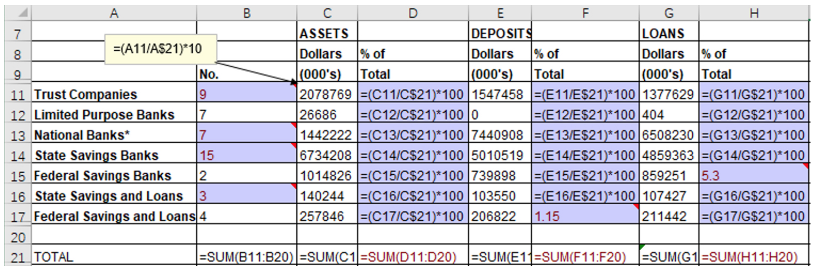
	A	B	C	D	E
5	LIBRARY TYPE	Total	Responses	WALK-IN	PHONE
6	4-YEAR COLLEGE/	33	25	25	24
7	2-YEAR COLLEGE	37	25	25	25
8	SUBTOTAL ACADE	70	=SUM(C6:C7)	=SUM(D6:D7)	=SUM(E6:E7)
9			=C8/B8	=D8/B8	=E8/B8
10	PUBLIC--OVER 500	38	38	38	38
11	PUBLIC--UNDER 50	27	22	22	19
12	SUBTOTAL PUBLIC	65	=SUM(C10:C11)	=SUM(D10:D11)	=SUM(E10:E11)
13			=C12/B12	=D12/B12	=E12/B12
14	TRIBAL	23	6	6	4
15	MEDICAL	49	27	27	24
16	COUNTY LAW	38	19	14	11
17	GOVT AGENCY	29	16	16	16
18	FEDERAL	23	9	8	9
19	NEWSPAPER	11	7	6	6
20	OTHER SPECIAL	57	28	23	27
21	SUBTOTAL SPECI/	230	=SUM(C14:C20)	=SUM(D14:D20)	=SUM(E14:E20)
22			=C21/B21	=D21/B21	=E21/B21
23	GRAND TOTAL	365	=SUM(C21,C12,C8)	=SUM(D21,D12,D8)	=SUM(E21,E12,E8)
24			=C23/B23	=D23/B23	=E23/B23

Figure 2: Motivating example - adapted from a worksheet in spreadsheet "VRSinventory01.xls" from the EUSES corpus, illustrating CUSTODES's cell clustering and defect detection results (6 clusters marked in different colors, and 37 defects annotated by red triangle marks)

clustering and defect detection negatively for spreadsheets. The example we discuss would exhibit how CUSTODES's effectiveness could be compromised.

This example was adapted from worksheet "Sheet1" in spreadsheet "VRSinventory01.xls" from the EUSES corpus [22], as shown in Fig. 2. For this worksheet, CUSTODES formed six cell clusters (marked in different colors) and detected a total of 37 defects (annotated by red triangle marks at top-right corners). Unfortunately, only 4 out of 37 defects (B8, B12, B21, and B23) are true positives (missing formula defects), leading to a very low precision of 10.8%. These false positives (33) were caused by CUSTODES's aggressive retrieval of irrelevant cells (e.g., C6–7, C10–11, and C14–20 for column C, and corresponding cells for columns D and E) into these clusters due to their similar weak features.

Our WARDER carefully considers this problem, and proposes to isolate those unqualified cells from entering the clusters, while still allowing qualified ones in. For this purpose, WARDER exploits the notion of *validity property* and conducts *validity-based refinements*, in order to improve the quality of cell clustering for effective spreadsheet defect detection. In the following, we first introduce WARDER-ori's three refinements over CUSTODES, and then present WARDER-ext's additional refinement instantiations and a cell retrieval enhancement over WARDER-ori.



	A	B	C	D	E	F	G	H
7			ASSETS		DEPOSITS		LOANS	
8	=(A11/A$21)*10		Dollars	% of	Dollars	% of	Dollars	% of
9		No.	(000's)	Total	(000's)	Total	(000's)	Total
11	Trust Companies	9	2078769	=(C11/C$21)*100	1547458	=(E11/E$21)*100	1377629	=(G11/G$21)*100
12	Limited Purpose Banks	7	26686	=(C12/C$21)*100	0	=(E12/E$21)*100	404	=(G12/G$21)*100
13	National Banks*	7	1442222	=(C13/C$21)*100	7440908	=(E13/E$21)*100	6508230	=(G13/G$21)*100
14	State Savings Banks	15	6734208	=(C14/C$21)*100	5010519	=(E14/E$21)*100	4859363	=(G14/G$21)*100
15	Federal Savings Banks	2	1014826	=(C15/C$21)*100	739898	=(E15/E$21)*100	859251	5.3
16	State Savings and Loans	3	140244	=(C16/C$21)*100	103550	=(E16/E$21)*100	107427	=(G16/G$21)*100
17	Federal Savings and Loans	4	257846	=(C17/C$21)*100	206822	1.15	211442	=(G17/G$21)*100
20								
21	TOTAL	=SUM(B11:B20)	=SUM(C1	=SUM(D11:D20)	=SUM(E1	=SUM(F11:F20)	=SUM(G1	=SUM(H11:H20)

Figure 3: Example 1 - Worksheet "Summary1201" for illustrating WARDER's single-cell validity refinement (one cluster is marked in purple by CUSTODES)

3.3. *WARDER's Refinements to CUSTODES's Cell Clustering*

In this subsection, we introduce WARDER-ori's three validity-based refinements to CUSTODES's cell clustering. The introduction is from the perspectives of single-cell, multi-cell, and whole-cluster properties, respectively. For ease of presentation, we refer to WARDER-ori by WARDER directly in this subsection.

3.3.1. *Single-cell Validity Refinement*

Our first refinement concerns, when WARDER expands CUSTODES's initial seed clusters with additional data cells, whether such cells to add are indeed valid themselves. Note that it may be difficult to tell whether these cells are valid or not directly, since they contain plain values only, without any visible relationship with other cells already existing in the concerned clusters. Nevertheless, since these cells to add and original cells in a target cluster are to be merged together, they should share common computational semantics according to the cell cluster's purpose. Then, these cells to add should be unifiable by some formula with those original cells. To validate this expectation, WARDER would test all formulas previously existing in the original cells in the cluster, to see whether any of them can fit in these cells to add. Here, "fit" means that such a formula, once replacing the plain value in a data cell to add, would still be computable. Otherwise, if all formulas are tested to be failed, e.g., citing a wrong place or causing a wrong reference, this data cell to add is probably problematic, and should be prevented from being added into this cluster, since it itself would become invalid by unification. This is known as the *single-cell validity refinement.*

Fig. 3 gives one example by worksheet "Summary1201", in which 25 cells (B11, B13, B14, B16, D11–17, F11–17, and H11–17) were clustered together (in purple) by CUSTODES. CUSTODES detected six defects, in which two (F17 and H15) are true positives (missing formula defects), and the other four (B11, B13, B14, and B16) are all false positives. The latter four data cells were retrieved into this cluster due to their shared weak features (e.g., similar headers and layouts) with the original cells in the cluster by CUSTODES. However, such retrieval is problematic according to WARDER's single-cell validity refinement. In fact, if any of the four data cells is added into the cluster, one has to unify its contained data with a formula unifiable with other existing cells in this cluster, and this unification would fail. For example, considering cell B11, the best candidate for its unifiable formula is "=(A11/A$21)*100", following the pattern shared by other cells in this cluster. However, this formula is non-computable, as both A11 and A$21 refer to a text string, which cannot participate into any arithmetic calculation. Similar problems occur to cells B13, B14, and B16, too. Therefore, WARDER would reject such data cells from being added into this cluster, guarding the precision for spreadsheet cell clustering.

3.3.2. Multi-cell Validity Refinement

The second refinement concerns, when WARDER expands initial seed clusters with additional data cells, whether the cells to add will not break existing multi-cell properties of the original cells in a cluster. Consider the references of a cell as an example, which are an important feature of spreadsheet cells and form the base of formula calculations. Suppose that the references of the original cells in a cluster never overlap with each other. Then one would expect that a new data cell to add should also not violate this property, when it is added into this cluster and its contained data is replaced by a formula for unification with those in other cells in this cluster. This expectation can also be expressed in a reversed way, i.e., references, if already overlapping with each other for the original cells in a cell cluster, should not encounter non-overlapping cases for new data cells to add. That is, this property should keep *consistent* for all cells in this cluster, and can be considered as an editing style across spreadsheet tables, which is also the focus of CUSTODES. Otherwise, the concerned

	A	AA	AB	AC	AD	AE	AF	AG	AH	AL	AM	AN
6	**Sponsor**	United B	The Clev			Northeas	City of C			National	ARMENI	
7	**Title**	Caring fc	Integratir	Total Foundation		Restorin	A Propo:	Total Local		New Me	MANOO	Total Other
8	**Total Amount**	5000	69231	=SUM(U8:AB8)		26487	58580	=SUM(AE8:AF8)		4500	450	=SUM(AL8:AM8)

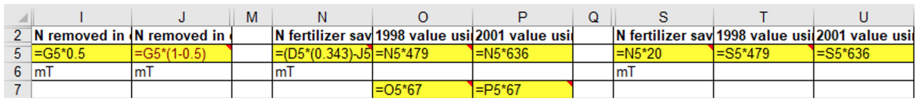
Figure 4: Example 2 - Worksheet "Detail for the College of A&S" for illustrating WARDER's multi-cell validity refinement. There is one cluster marked in yellow by CUSTODES, leading to six wrongly reported defects as annotated with red triangle marks, while these six cells would be filtered out from the cluster by WARDER's multi-cell validity refinement, and no longer be wrongly reported as defects.

data cell is considered to be problematic, and should be prevented from being added into this cluster. This is known as the *multi-cell validity refinement*.

Fig. 4 gives one example by worksheet "Detail for the College of A&S", in which nine cells (AA8, AB8, AC8, AE8, AF8, AG8, AL8, AM8, and AN8) were clustered together (in yellow) by CUSTODES. CUSTODES then detected six defects (AA8, AB8, AE8, AF8, AL8, and AM8) simply due to their contained plain values (missing formula defect), but all of them are false positives. On the other hand, WARDER would reject adding these six data cells (AA8, AB8, AE8, AF8, AL8, and AM8) into the cluster, and thus avoid detecting them as defects. In fact, the six data cells do not share any common computational semantics as the other three formula cells (AC8, AG8, and AN8). The former data cells refer to some specific values, which are directly from users, while the latter formula cells calculate the sums of several cells left to them. WARDER distinguishes the former from the latter by its multi-cell validity refinement: the references of the latter three formula cells do not overlap, but this property would be violated if merging the former six data cells with them together and replacing the data of the former with any formula from the latter cells. For example, when the data in cell AF8 is replaced by a formula "SUM(AD8:AE8)" by following the pattern of cell AG8, its references (AD8 and AE8) would overlap with cell AG8's references (AE8 and AF8). Similar problems occur to cells AA8, AB8, AE8, AL8, and AM8, too. As a result, WARDER would reject such data cells from being added into this cluster.

3.3.3. *Whole-cluster Validity Refinement*

The last refinement concerns the validity of each cell cluster as a whole, i.e., it focuses on cluster-level rather cell-level validity properties. It is expected that a cell cluster should follow common computational semantics in terms of a unified formula



	I	J	M	N	O	P	Q	S	T	U
2	N removed in (	N removed in (		N fertilizer sav	1998 value usi	2001 value usi		N fertilizer sav	1998 value usi	2001 value usi
5	=G5*0.5	=G5*(1-0.5)		=(D5*(0.343)-J5	=N5*479	=N5*636		=N5*20	=S5*479	=S5*636
6	mT	mT		mT				mT		
7					=O5*67	=P5*67				

Figure 5: Example 3 - Worksheet "World 1996" for illustrating WARDER's whole-cluster validity refinement (one cluster is marked in yellow by CUSTODES)

that can cover most cells in this cluster. Here, "cover" means that the formula in a concerned cell is equivalent to this unified formula, or the data in the cell can also be obtained by the calculation of this unified formula. WARDER would test all formulas available in this cluster as candidate formulas. If none of them can serve for this purpose, it would consider this cluster unqualified and choose to cancel it from further actions, in order to avoid misbehavior (e.g., mistakenly considering most cells as defects, which turn out to be many false positives) in later defect detection. This is known as the *whole-cluster validity refinement*.

Fig. 5 gives one example by worksheet "World 1996", in which ten cells were clustered together (in yellow) by CUSTODES. From this cluster, CUSTODES detected seven out of them as defects, but all of them are false positives. In fact, the ten cells contain almost totally different formulas (five patterns), which strongly indicate that they follow different computational semantics. By its whole-cluster validity refinement, WARDER would cancel this cell cluster. Note that WARDER needs a threshold value to control the judgment of "covering most cells". To play safe, WARDER chooses the threshold to be a conservative value of 50% to protect as many cell clusters as possible (as a comparison, CACheck [20] chooses a more aggressive value of 70%).

3.4. WARDER's Further Refinements and Cell Retrieval Enhancement

In this subsection, we first introduce WARDER-ext's further refinements by additional instantiations to the three validity properties. We then introduce WARDER-ext's cell retrieval enhancement for expanding cell clusters with validity-based quality guarantees. For ease of presentation, we refer to WARDER-ext by WARDER directly in this subsection.

3.4.1. *Single-cell Validity Refinement*

In the single-cell validity refinement, a non-computable formula can be due to various reasons. Text strings not being able to participate into arithmetic calculations are one of them, which is also WARDER's previous focus. We further extend WARDER's scope to include more restrictions. For example, an empty cell can be considered as one with a pending value to be filled in in future (currently, it is zero but with some intended computational semantics), or one simply for formatting purposes (no value, i.e., without any computational semantics). WARDER considers the former qualified for participating into formula calculations, but the latter unqualified, which can be judged from whether the concerned cell is associated with any table region (e.g., with table headers). Then WARDER would reject a data cell from being added into a cluster, if all the cells' unifiable formulas have to reference such unqualified empty cells. Empty cells are just one example. Considering spreadsheet users' diverse tabulating styles, WARDER collects and examines more content types in cells for the consideration of being qualified or unqualified for participating into formula calculations. For example, WARDER also considers cells with special symbols or words such as "#", "...", "x", and "NA" [50] as qualified (i.e., role of empty cells be to filled in future).

Fig. 6 gives one example from worksheet "apr02". Altogether 33 cells (F5−34, C42, F42, and I42) were clustered together (in green) by CUSTODES. CUSTODES detected 30 defects (F5−34) in this cluster, but all of them are false positives. They were retrieved into this cluster due to their shared weak features (e.g., layout) with the original cells in the cluster. However, such retrieval is problematic according to WARDER's single-cell validity refinement. Take cell F34 as an example. If cell F34 is considered into this cluster, it has to unify its contained data with formula "R35*S35" (the only candidate formula, following the pattern shared by cells C42, F42, and I42). However, WARDER considers this formula non-computable, because cell S35 belongs to those empty cells considered only for formatting purposes. To see it, cell S35 does not have a table header above it (in fact, all cells in column S are empty, simply for formatting). Similar analysis also applies to cells F5−33. As a result, WARDER would reject all these 30 cells (F5−34) from being added into this cluster.

	C	F	I	R	S
1					
2					
3	Apache/Cinergy	Williams	Total		
4	Supply	Supply	Supply	Total	
5	20000	25000	=+F5+C5	=C5*D5	
6	20000	25000	=+F6+C6	=C6*D6	
7	20000	25000	=+F7+C7	=C7*D7	
8	20000	25000	=+F8+C8	=C8*D8	
9	20000	25000	=+F9+C9	=C9*D9	
10	20000	25000	=+F10+C10	=C10*D10	
11	20000	25000	=+F11+C11	=C11*D11	
12	20000	25000	=+F12+C12	=C12*D12	
13	20000	25000	=+F13+C13	=C13*D13	
14	20000	25000	=+F14+C14	=C14*D14	
15	20000	25000	=+F15+C15	=C15*D15	
16	20000	25000	=+F16+C16	=C16*D16	
17	20000	25000	=+F17+C17	=C17*D17	
18	20000	25000	=+F18+C18	=C18*D18	
19	20000	25000	=+F19+C19	=C19*D19	
20	20000	25000	=+F20+C20	=C20*D20	
21	20000	25000	=+F21+C21	=C21*D21	
22	20000	25000	=+F22+C22	=C22*D22	
23	20000	25000	=+F23+C23	=C23*D23	
24	20000	25000	=+F24+C24	=C24*D24	
25	20000	25000	=+F25+C25	=C25*D25	
26	20000	25000	=+F26+C26	=C26*D26	
27	20000	25000	=+F27+C27	=C27*D27	
28	20000	25000	=+F28+C28	=C28*D28	
29	20000	25000	=+F29+C29	=C29*D29	
30	20000	25000	=+F30+C30		
31	20000	25000	=+F31+C31		
32	20000	25000	=+F32+C32		
33	20000	25000	=+F33+C33		
34	20000	25000	=+F34+C34		
35					
42	=P43*Q43	=R43*S43	=U43*V43		

Figure 6: Example 4 - Worksheet "apr02" for illustrating WARDER's extended single-cell validity refinement (one cluster is marked in green by CUSTODES; other clusters are not marked for ease of presentation)

3.4.2. Multi-cell Validity Refinement

In the multi-cell validity refinement, the cells in a cluster can have various properties. While WARDER previously focused on the reference-overlapping property among multiple cells, we extend WARDER to consider more properties (e.g., cell type and layout style). Take cell type as an example. WARDER now additionally enforces the type consistency property for references among multiple cells. Suppose that the references of the original cells in a cluster have consistent types for their referenced cells (e.g., all being formula or data cells). Then one would expect that a new data cell to add should not violate this property, when it is added into this cluster and its contained data is replaced by a formula for unification with those in other cells in this cluster. That is, this formula should have consistent types for its referenced cells as those in the existing cells. Otherwise, the concerned data cell should be prevented from being added into this cluster. Still, considering that a cell cluster can contain potential defects, this property might be affected when some cells in this cluster do

	A	D	E	F	G	I	M
14	**Sub-Total**	=SUM(D9:D13)	=SUM(E9:E13)	=SUM(F9:F13)	=SUM(G9:G13)	=SUM(I9:I13)	0.179279643868879
32	**Sub-Total**	=SUM(D18:D31)	=SUM(E18:E31)	=SUM(F18:F31)	=SUM(G18:G31)	=SUM(I18:I31)	0.029
33							
34	**Grand Total**	=(D14)+(D32)	=(E14)+(E32)	=(F14)+(F32)	=(G14)+(G32)	=(I14)+(I32)	0.081

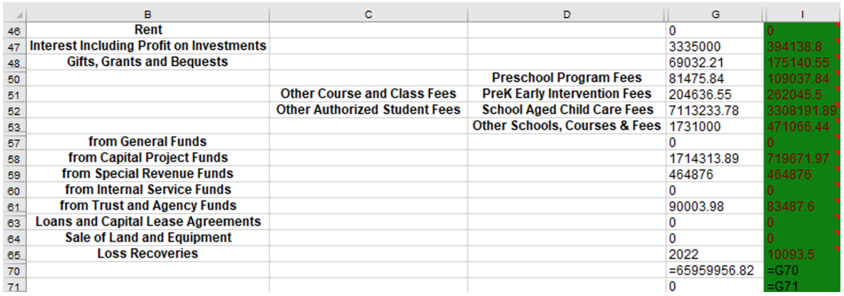
Figure 7: Example 5 - Worksheet "feeder' for illustrating WARDER's extended multi-cell validity refinement (one cluster is marked in yellow by CUSTODES; other clusters are not marked for ease of presentation)

not carry their supposed types (e.g., some data cells suffering missing formula defects should be formula cells instead). To avoid filtering out relevant cells accidentally as irrelevant cells from a cluster due to such issues, WARDER considers all data cells to be added as a whole (when their total number is not trivially only one), and refrains itself from applying this refinement if over half these data cells would thus be filtered out, in order to play safe.

Fig. 7 gives one example by worksheet "feeder", in which six cells (D34, E34, F34, G34, I34, and M34) were clustered together (in yellow) by CUSTODES. A defect (M34) was detected by CUSTODES, but it is a false positive. We observe that the former five cells (D34, E34, F34, G34, and I34) reference corresponding cells in rows 14 and 32, respectively, and these referenced cells are all formula cells. When considering cell M34, if it is added into this cluster, its contained data would be replaced by a formula for unification with data in other cells in the cluster. The unifiable formula would be "M14+M32" (the only candidate formula, following the pattern shared by cells D34, E34, F34, G34, and I34). However, both M14 and M32 are data cells, inconsistent with other referenced cells (formula cells) for existing cells in this cluster. This violates the type consistency property. As a result, WARDER would now reject cell M34 from being added into this cluster.

3.4.3. Whole-cluster Validity Refinement

In the whole-cluster validity refinement, the cells in a cluster can be formula cells or data cells, and WARDER previously focused on formula cells only, i.e., it tests the possibility of unifying formulas in at least 50% formula cells in the cluster. We now extend WARDER's test to data cells as well, i.e., it additionally tests the possibility of unifying both formulas and data in at least 50% formula and data cells. For any cluster that fails to pass either test, WARDER would cancel this cluster.



	B	C	D	G	I
46	Rent			0	0
47	Interest Including Profit on Investments			3335000	394138.8
48	Gifts, Grants and Bequests			69032.21	175140.55
50			Preschool Program Fees	81475.84	109037.84
51		Other Course and Class Fees	PreK Early Intervention Fees	204636.55	262045.5
52		Other Authorized Student Fees	School Aged Child Care Fees	7113233.78	3308191.89
53			Other Schools, Courses & Fees	1731000	471066.44
57	from General Funds			0	0
58	from Capital Project Funds			1714313.89	719671.97
59	from Special Revenue Funds			464876	464876
60	from Internal Service Funds			0	0
61	from Trust and Agency Funds			90003.98	83487.6
63	Loans and Capital Lease Agreements			0	0
64	Sale of Land and Equipment			0	0
65	Loss Recoveries			2022	10093.5
70				=65959956.82	=G70
71				0	=G71

Figure 8: Example 6 - Worksheet "General Rev 3" for illustrating WARDER's extended whole-cluster validity refinement (one cluster is marked in green by CUSTODES)

Fig. 8 gives one example by worksheet "General Rev 3", in which altogether 17 cells were clustered together (in green) by CUSTODES. From this cluster, CUSTODES detected 15 (all data cells) out of them as defects, but all of them are false positives. In fact, the 17 cells include two formula ones and 15 data ones. The two formula cells (I70 and I71) happened to share common computational semantics (both referencing the cell immediately to its left), and thus WARDER previously would fail to cancel this cluster (property holding). For extended WARDER, it considers both formula and data cells, for testing the possibility of unifying all formulas and data found in these cells. Then, after replacing the data with formulas of references to their left cells (following the pattern of I70 and I71), the unification would fail for 9 out of 15 data cells (i.e., cannot cover these cells). In fact, for the remaining six data cells that have been covered, five of them happen to contain a plain value of zero (the only one that is really covered is cell I59). Even if one counts formula and data cells together, the whole coverage is still less than 50% (property violated). As a result, WARDER would now cancel this cluster, avoiding numerous false positives detected later.

3.4.4. Cell Retrieval Enhancement

WARDER's validity-based refinements improve the quality of cell clusters from CUSTODES only by filtering out irrelevant cells. We further extend it to retrieve back those relevant cells that fall outside the consideration of CUSTODES. Note that this can accidentally add back plausible cells as relevant ones, and thus needs to be careful. Regarding this, WARDER proposes a cell retrieval enhancement based on our earlier

	D	E	F	L	AF	AG
8	**Total Population**	**Total Housing Units**	**Total**	**Total**	**Private**	**Not Enrolled**
9	836	335	674	11	0	477
10	1773	972	1743	18	56	1273
11	2663	1719	2632	38	69	2182
111	0	0	0	0	0	0
112	0	0	0	0	0	0
113	1341	888	1267	15	18	1101
114	1138	955	1158	6	26	1005
115	128283	64251	=+SUM(F9:F114)	=+SUM(L9:L114)	=+SUM(AF9:AF114)	=+SUM(AG9:AG114)

Figure 9: Example 7 - Worksheet "Sheet1" for illustrating WARDER's cell retrieval enhancement (one cluster is marked in green by CUSTODES)

discussed validity properties for quality guarantees.

WARDER requires that all cell clusters should have passed the aforementioned single-cell, multi-cell, and whole-cluster validity-based refinements before their cell retrieval enhancement. This is for setting up the quality criteria for later cell retrieval to reference, since these refinements have filtered out irrelevant cells from the clusters as many as possible. Then WARDER conducts the cell retrieval by examining cells surrounding each cluster in a depth-first search manner, and adds them into the concerned cluster if they satisfy the following conditions:

(1) The cell under consideration should have not been contained in any cluster. That is, this cell is still free now. This condition guarantees that one cluster will not steal any cell from other clusters.

(2) The cell should not violate the aforementioned single-cell and multi-cell validity properties of the target cluster that is considering retrieving this cell. In addition, if the cell could be retrieved into this cluster, the cluster's whole-cluster validity properties should also not be violated.

(3) The cell should share common computational semantics with the cells already in the target cluster. That is, the content contained in this cell can be covered by the formula that has unified existing cells in the cluster. To avoid coincident equivalence in the unification judgment, the cell under consideration should not contain trivial values (e.g., zero) or be unifiable by trivial formulas (e.g., simply referencing another cell without any calculation).

With the above three conditions, WARDER aims for maintaining the high quality of cell clusters, even with added new cells, with the help of previously proposed validity-based refinements. For the illustration of its usefulness, we give one example, as shown in Fig. 9.

This example is based on worksheet "Sheet1", in which altogether 23 cells (F115 and L115−AG115) were clustered together (in green) by CUSTODES (some columns are hidden for saving space). For this cluster, CUSTODES missed two true defects (D115 and E115), since they are not in the cluster at all (although immediately adjacent). For this case, we observe that CUSTODES's formed cell clusters could be not complete in containing relevant cells. Even WARDER's refinements to these clusters only improve their quality by filtering out irrelevant cells, but can do nothing with the two left-aside cells. Nevertheless, WARDER's cell retrieval enhancement can successfully retrieve the two cells into the concerned cluster, since: (1) both cells are not contained in any cluster, (2) they do not violate any single-cell, multi-cell, and whole-cluster validity property associated with the target cluster, and (3) the data contained in them (128,283 and 64,251) can be unified by formula "+SUM(D9:D114)" and "+SUM(E9:E114)" (value equivalence), respectively. With the two cells added back, this cluster becomes both complete (i.e., relevant cells retrieved) and precise (i.e., validity properties still holding), contributing to later spreadsheet defect detection.

3.5. Motivating Example Revisited

We revisit our earlier motivating example (Section 3.2) to illustrate how WARDER improves CUSTODES's cell clustering and addresses its limitations in this example. As shown in Fig. 10, WARDER successfully filtered out 33 irrelevant cells (marked by the blue cross) from the six clusters formed by CUSTODES, preventing these cells from being identified as false defects later.

We take 11 cells (C6−7, C10−11, and C14−20) as an example to explain what WARDER does. These 11 cells, together with the other four (C9, C13, C22, and C24), were previously considered in the same cluster by CUSTODES (Fig. 2), which later caused 11 false defects. WARDER can successfully identify them for filtering out based on its validity-based refinements.

First, cell C6 violates the single-cell validity and is thus filtered out. The reason is that cell C6's unified formula is "C5/B5" (only candidate), which is, however, non-

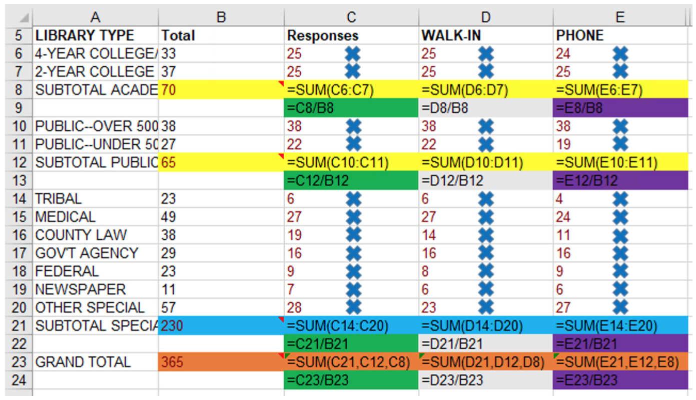


Figure 10: Motivating example revisited - the same worksheet as in Fig. 2, illustrating WARDER's cell clustering and defect detection results (6 clusters marked in different colors, and 4 defects annotated by red triangle marks, with 33 irrelevant cells marked by the blue cross filtered out from CUSTODES's clustering results)

computable, since this formula refers to two text strings ("Responses" and "Total") for an impossible arithmetic calculation (division).

Second, nine cells (C10−11 and C14−20) violate the multi-cell validity and are thus filtered out. The reason is that the references of original cells (i.e., C9, C13, C22, and C24) existing in this cluster do not overlap with each other or other cells, but the nine cells, if added into this cluster, would violate this property. For example, cell C10's unified formula "C9/B9" references cells C9 and B9, causing existing cell C9 overlapped.

Third, the last cell C7 violates our extended multi-cell validity refinement and is also filtered out. The reason is that cell C7's unified formula is "C6/B6" (only candidate), but both referenced cells (C6 and B6) are data ones. This fact is inconsistent with that of original cells existing in this cluster, whose referenced cells are one formula and one data.

The other 22 irrelevant cells in columns D and E can be similarly prevented from being added into this cluster. By doing such validity-based refinements, WARDER effectively improves the quality for CUSTODES's cell clustering.

Table 1: Statistics of our experimental subjects

# spreadsheets	# worksheets	# cells	# formula cells	# cell clusters	# defects (faulty cells)
70	291	189,027	26,716	1,610	1,974

4. Evaluation

In this section, we experimentally evaluate our WARDER technique (both versions, WARDER-ori in Section 3.3 and WARDER-ext in Section 3.4), and compare it to existing spreadsheet defect detection techniques.

We implemented WARDER in Java, and used Apache POI [2] to manipulate spreadsheets. Integrated with CUSTODES, WARDER contains a total of 9,800 lines of Java code, with about 5,000 lines are newly added to, or modified from CUSTODES. Working as the same style as CUSTODES does, given a spreadsheet for analysis, WARDER annotates cells in the spreadsheet by means of comments, indicating which cells can be clustered together with common computational semantics and what defects could be associated with the commented cells (e.g., missing formula defects or inconsistent formula defects).

4.1. Research Questions

In the evaluation, we aim to answer the following three research questions:

- **RQ1 (Effectiveness):** *How effective is WARDER in clustering relevant cells and detecting defects in the clusters, as compared to existing spreadsheet defect detection techniques?*
- **RQ2 (Correlation):** *Does WARDER's improved cell clustering contribute to its spreadsheet defect detection, in particular, on the detection precision?*
- **RQ3 (Individual impacts):** *How do WARDER's three validity-based refinements individually contribute to its effectiveness on the spreadsheet defect detection?*

4.2. Experimental Design and Setup

Subjects. To facilitate WARDER's comparison with its predecessor CUSTODES, we first selected CUSTODES's own benchmark as our experimental subjects. The benchmark was originally sampled from the EUSES corpus [22], and contains 70 spreadsheets and 291 worksheets, as shown in Table 1. The 291 worksheets contain 189,027 cells, among which 26,716 are formula cells. The benchmark also contains ground truths to facilitate follow-up research in the spreadsheet field. The ground truths annotate 1,610 cell clusters, which contain 1,974 defects (faulty cells with missing or inconsistent formula defects), for evaluation purposes.

Techniques. We compared our WARDER (both versions) with the aforementioned five spreadsheet defect detection techniques, namely, UCheck, Dimension, AmCheck, CACheck, and CUSTODES (special focus). For comparison purposes, we obtained the five techniques' implementations from their respective authors. We compared WARDER with these five techniques on their spreadsheet defect detection effectiveness. We also additionally compared WARDER's cell clustering effectiveness with that of CUSTODES (WARDER's improvement focus).

To study individual impacts (RQ3) of WARDER's three validity-based refinements on its effectiveness, we configured WARDER to enable these refinements individually in the experiments, which are named WARDER-sc (with only single-cell validity refinement enabled), WARDER-mc (with only multi-cell validity refinement enabled), and WARDER-wc (with only whole-cluster validity refinement enabled), respectively. Then the base configuration with all the three validity refinements enabled is named WARDER-full or WARDER directly.

Metrics. Regarding the effectiveness on spreadsheet defect detection (applicable to all studied techniques), we first measured the number of detects both reported by a technique and in the ground truths (true positives or TP), that reported but not in the ground truths (false positives or FP), and that in the ground truths but not reported (false negatives or FN). Then based on these measurements, we calculated

$$precision_d = \frac{\text{TP}}{\text{TP} + \text{FP}},\ recall_d = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

Table 2: Defect detection results for the seven spreadsheet defect detection techniques/versions

Technique	**Detected (#)**	**TP (#)**	**FP (#)**	$precision_d$	$recall_d$	$F\text{-}measure_d$
UCheck	204	1	203	0.5%	0.1%	0.00
Dimension	1,842	14	1,828	0.8%	0.7%	0.01
AmCheck	2,343	1,322	1,021	56.4%	67.0%	0.61
CACheck	1,866	1,356	510	72.7%	68.7%	0.71
CUSTODES	2,380	1,545	835	64.9%	78.3%	0.71
WARDER-ori	1,612	1,421	191	88.2%	72.0%	0.79
WARDER-ext	1,669	1,488	181	89.2%	75.4%	0.82

and

$$F\text{-}measure_d = 2 \cdot precision_d \cdot \frac{recall_d}{precision_d + recall_d},$$

which is the harmonic mean of $precision_d$ and $recall_d$ (subscript "d" represents defect detection). These three metrics measure the technique's effectiveness on spreadsheet defect detection.

Regarding the effectiveness on spreadsheet cell clustering (applicable to WARDER and CUSTODES only), we followed CUSTODES's pair-wise similarity comparison [13] to calculate TP (number of pairs of relevant cells clustered together), FP (number of pairs of irrelevant cells clustered together), and FN (number of pairs of relevant cells not clustered together). We then calculated a technique's effectiveness on spreadsheet cell clustering by $precision_c$, $recall_c$, and $F\text{-}measure_c$ in a similar way (subscript of "c" represents cell clustering).

Environment. All experiments were conducted on a PC Station (ThinkStation) with an Intel® Xeon® CPU E5 1620 v4 @3.50GHz and 64GB RAM, which was installed with Microsoft Windows 10 Professional and Oracle Java 8.

4.3. Experimental Results and Analyses

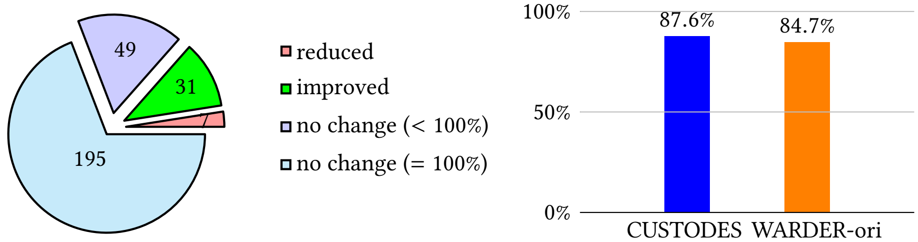
In the following, we report and analyze the experimental results, and answer the three research questions in turn.

1) RQ1: Effectiveness. We first evaluate WARDER's (both versions') effectiveness on spreadsheet cell clustering and defect detection, and then compare it to that of the other five defect detection techniques.

Regarding spreadsheet defect detection, Table 2 [2] compares the detection results for all the seven techniques/versions, which include the precision, recall, and F-measure values, as well as the statistics of the detected defects and their contained true positives and false positives. From the table, we observe that: (1) UCheck and Dimension obtained only very low scores (less than 1% for both the precision and recall, and no more than 0.01 for the F-measure) due to their limited analysis scopes, echoing earlier studies [59]; (2) AmCheck and CACheck obtained much higher scores (56.4–72.7% for the precision, 67.0–68.7% for the recall, and 0.61–0.71 for the F-measure) due to their effective analysis patterns (e.g., cell arrays); (3) CUSTODES happened to obtain an equal score on F-measure as CACheck, but with a focus on the recall (78.3% against 68.7% for CACheck); (4) WARDER-ori, as expected, focuses on improving the precision for CUSTODES's defect detection and obtained a striking improvement from 64.9% to 88.2%, leading to a large jump on the F-measure from 0.71 to 0.79, by a small sacrifice on the recall of about 6%, as against its predecessor CUSTODES; (5) WARDER-ext exceeded WARDER-ori on both the TP (67 more) and FP (10 less) measurements, realizing the highest precision of 89.2% among all and a recall bonus of 3.4% over WARDER-ori (75.4% vs. 72.0%), resulting the highest F-measure of 0.82 among all techniques/versions. In the comparisons, one may concern that CUSTODES detected more true positives than WARDER (e.g., 124 more than WARDER-ori and 57 more than WARDER-ext), but these results were accompanied with much more false positives (e.g., 644 more than WARDER-ori and 654 more than WARDER-ext), which can be overwhelming for manual inspection.

Regarding spreadsheet cell clustering, Fig. 11 compares the clustering results between CUSTODES and WARDER-ori, and Fig. 12 compares those between CUSTODES and WARDER-ext (as mentioned earlier, clustering comparisons not applicable to

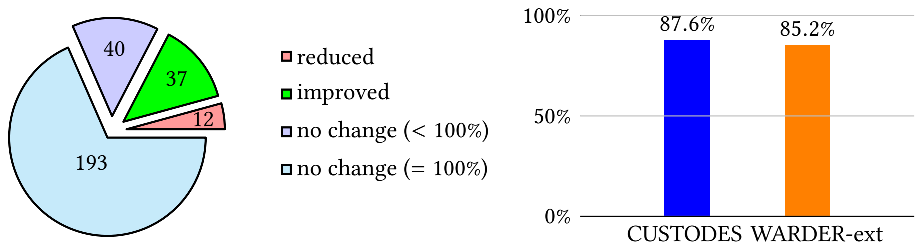
[2]We note that for fair experimental comparisons, we have (re-)conducted all experiments. There are some minor changes to the data as presented in the previous work [38], but the changes are slight (caused by fixing a counting flaw in the data collection), and do not affect our subsequent discussions.



(a) Precision comparison (in terms of affected worksheets)

(b) Recall comparison (overall)

Figure 11: Cell clustering results for CUSTODES and WARDER-ori



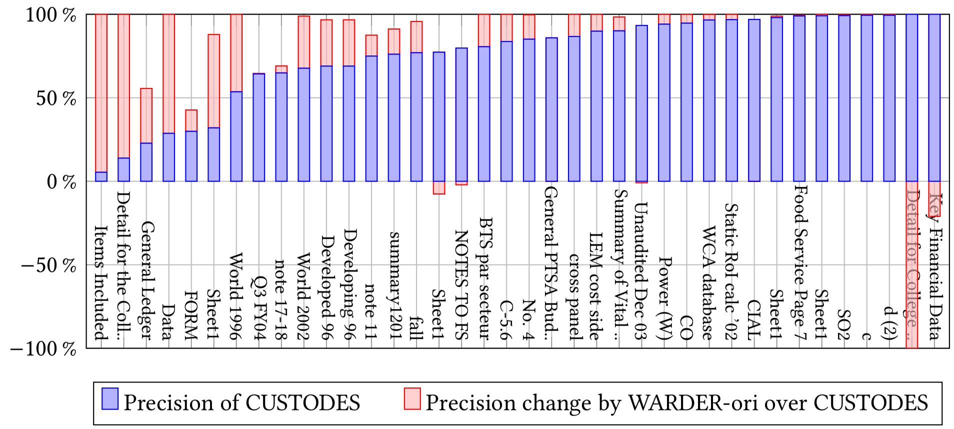
(a) Precision comparison (in terms of affected worksheets)

(b) Recall comparison (overall)

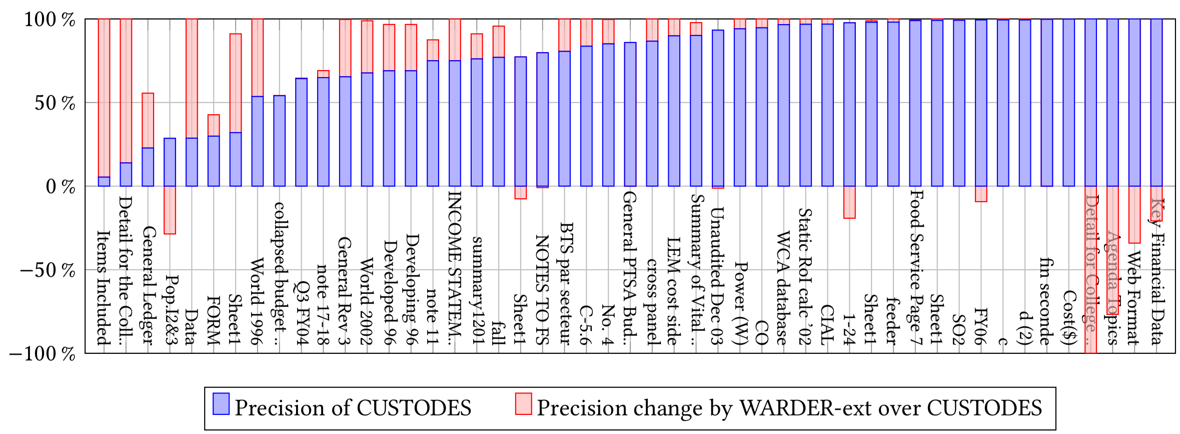
Figure 12: Cell clustering results for CUSTODES and WARDER-ext

other techniques). The comparisons concern both the clustering precision and recall.

We first study the clustering precision from the perspective of worksheets. For this purpose, we partition the 282 worksheets containing at least one cluster out of the total of 291 ones into four categories (Fig. 11a and Fig. 12a). We observe that: (1) WARDER-ori improved the precision for 31 worksheets, and reduced for 7 ones; (2) The precision kept unchanged for 244 worksheets, in which 195 ones already reached 100% (i.e., upper limit). In other words, WARDER-ori improved the cell clustering for 226 worksheets (226/282 = 80.1%), either by improving the precision or already reaching the upper limit of 100%. We further look into details for all 38 worksheets with precision changes (Fig. 13a), and observe that WARDER-ori improved the cell clustering precision by 0.3% to 94.6% (20.7% on average). The improvement gains are significant, much more than those lost due to few reduced precisions. For a complete picture, we note that WARDER-ori's large improvement on the clustering precision



(a) Precision changes from CUSTODES to WARDER-ori (38 worksheets)



(b) Precision changes from CUSTODES to WARDER-ext (49 worksheets)

Figure 13: Worksheets with clustering precision changes from CUSTODES to WARDER

	A	N	O	V	W	Y	Z	AC	AD	AQ	AR
9	Spon	U.S. Dep	Total Federal	Thoma	Total Foundation	Clevela	Total Lo	Cente	Total Other	Ohio D	Total State
10	Title	Modeling		Citizen		Educat		Fee a		Readin	
11	Total	423331	=SUM(B11:N11)	9720	=SUM(Q11:V11)	86070	=+Y11	5000	=SUM(AB11:AC11)	80851	=SUM(AF11:AQ11)

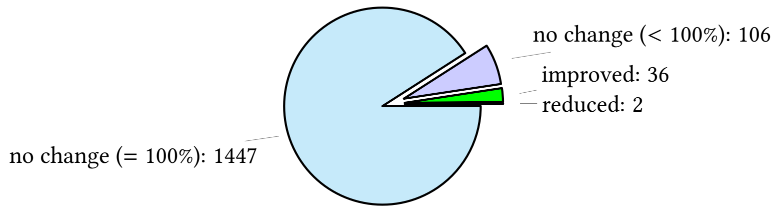
Figure 14: Worksheet "Detail for College of Education" (one cluster marked in green)

came only with a marginal loss on the clustering recall of 2.9% (from 87.6% to 84.7%, in Fig. 11b).

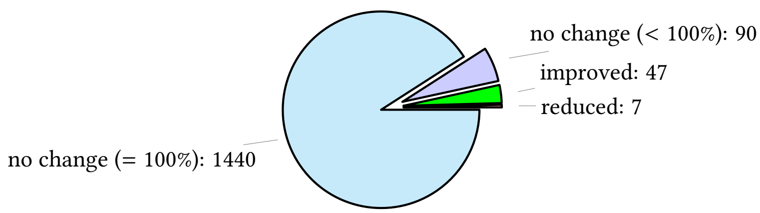
As a comparison, WARDER-ext behaved comparably or better than WARDER-ori on spreadsheet cell clustering. For example, it improved the cell clustering, either by improving the precision or already reaching the upper limit of 100%, for 230 worksheets (230/282 = 81.6%), which are slightly more than that of WARDER-ori (230 vs. 226, or 81.6% vs. 80.1%). We also look into details of all 49 worksheets with precision changes (Fig. 13b), and observe that WARDER-ext improved the clustering precision by 0.1% to 94.6% (19.1% on average), comparable with that of WARDER-ori. Regarding the clustering recall, WARDER-ext also improved over WARDER-ori by slightly reducing the latter's loss against CUSTODES on the recall (-2.4% vs. -2.9%, or 85.2% vs. 84.7%).

Both Figs. 13a and 13b disclose an exceptional worksheet "Detail for College of Education", for which WARDER's (both versions) clustering precisions dropped from 100% to zero (against CUSTODES). We further look into this case. The ground truths suggest that cells {O11, W11, Z11, AD11, AR11} should be clustered together, as marked in green (Fig. 14). CUSTODES "correctly" clustered these cells together, but WARDER did not. However, we found that these five cells actually contain different formulas, and thus violate WARDER's whole-cluster validity property (there is no common computational semantics shared among most of these cells). This explains why WARDER rejected this cluster (we conjecture that it could be a clustering flaw in the ground truths). In fact, this cell cluster indeed does not contain any defect, as the ground truths suggest. Therefore, this precision dropping on spreadsheet cell clustering did not affect WARDER's spreadsheet defect detection at all.

For a more fine-grained analysis, we then study the clustering precision from the perspective of clusters themselves, since each worksheet can contain a varying num-



(a) Precision comparison for CUSTODES and WARDER-ori



(b) Precision comparison for CUSTODES and WARDER-ext

Figure 15: Cell clustering results for CUSTODES and WARDER in terms of affected clusters

ber of clusters and the earlier studied precision associated with a worksheet may not precisely represent that with each cluster in this worksheet. To align the comparisons, the analysis focuses on those cell clusters reported by both CUSTODES and WARDER, and compare them to those in the ground truths to examine the precision changes. The number of such cell clusters is 1,591 for the comparison between CUSTODES and WARDER-ori, and 1,584 for that between CUSTODES and WARDER-ext. Then we partition these clusters into four categories to illustrate the precision changes (Fig. 15a and Fig. 15b). Considering the effectiveness as either increasing the clustering precision or already reaching the upper limit of 100%, WARDER-ori is effective in improving the cell clustering on 93.2% analyzed clusters $(1,483/1,591)$, and this percentage is 93.9% for WARDER-ext $(1,487/1,584)$. Therefore, both WARDER versions are effective in refining cell clustering for a higher quality, with WARDER-ext behaving slightly better.

In summary, to answer research question RQ1, we conclude that: *WARDER is effective in both spreadsheet cell clustering and defect detection (WARDER-ext has further improvement over WARDER-ori); it greatly improved the defect detection precision (by*

Table 3: Correlation study for WARDER against CUSTODES on the precision changes between cell clustering and defect detection in terms of worksheets (↑: precision improved, ↓: precision reduced, →: precision unchanged)

Category	**Δ precision (cell clustering)**	**Δ precision (defect detection)**	**# worksheets**	**Sum of each category**
Correlation supported	↑	↑	13	115 (82.1%)
	↓	↓	4	
	→	→	98	
Correlation unsupported	↑	→	6	16 (11.4%)
	↑	↓	3	
	↓	→	4	
	↓	↑	3	
Unknown	→	↑	8	9 (6.4%)
	→	↓	1	
Total	-	-	140	140 (100.0%)

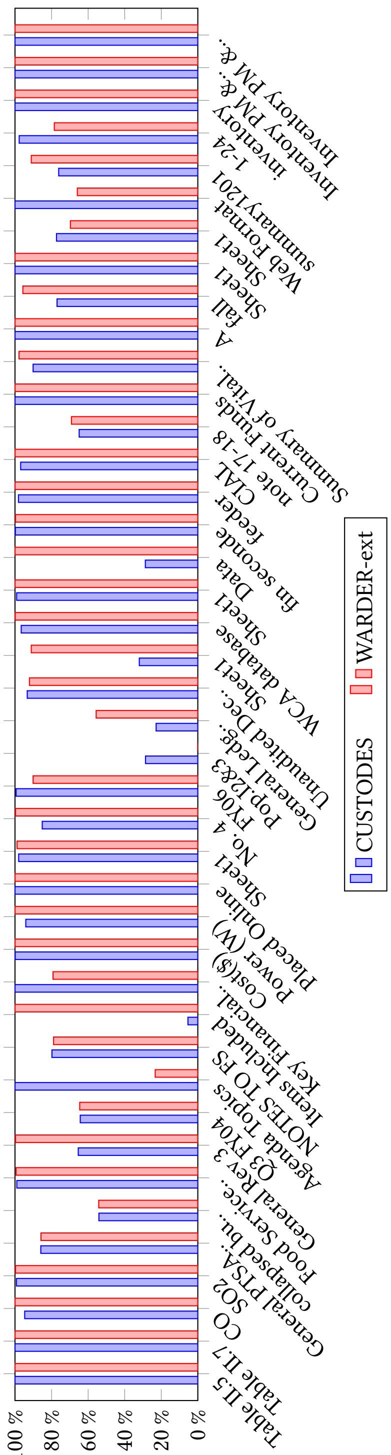
16.5–88.7%), and achieved the best precision (89.2%) and F-measure (0.82) values among all studied spreadsheet defect detection techniques.

2) RQ2: Correlation. We then study the correlation between WARDER's precision improvement over CUSTODES on spreadsheet cell clustering and that on defect detection. Since we in RQ1 have validated both WARDER versions' effectiveness and they follow the same framework, in the following we conduct experiments with WARDER-ext for the correlation study. For ease of presentation, we refer to WARDER-ext by WARDER directly in this part.

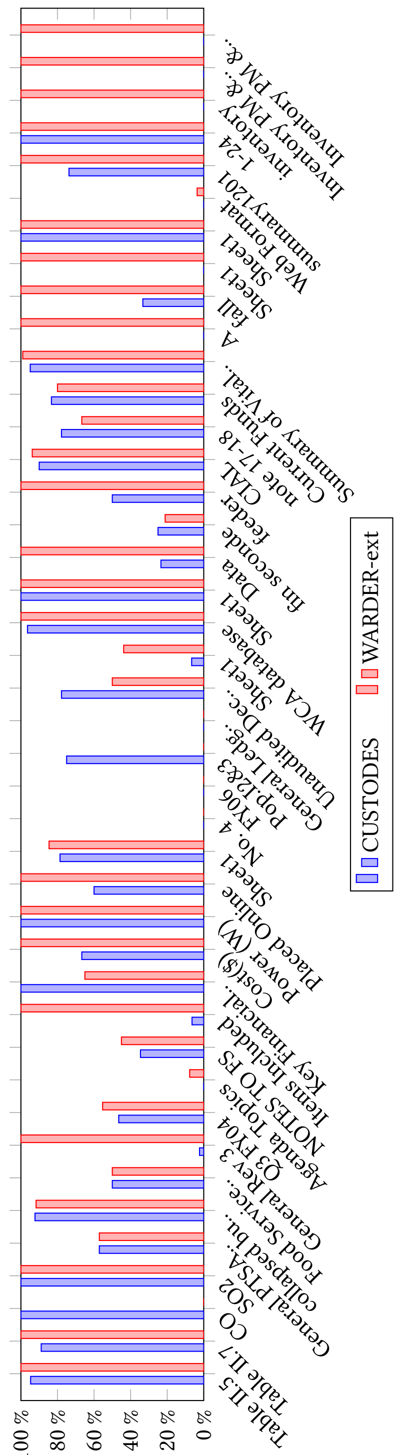
We use three symbols ↑, ↓, and → to represent the precision improved, precision reduced, and precision unchanged, respectively. Then, we partition 140 worksheets containing at least one defect in the ground truths out of the total of 291 ones into three categories, as shown in Table 3: (1) the "correlation supported" category indicates that when WARDER has its precision improved, reduced, or unchanged on spreadsheet cell clustering as compared to CUSTODES, that on spreadsheet defect detection behaved the same way; (2) the "correlation unsupported" category indicates that when WARDER has its precision improved on cell clustering, that on defect detection kept unchanged or was even reduced, or when having its precision reduced on cell clustering, that on defect detection kept unchanged or was even improved; (3) finally, the "unknown" category lists the remaining combinations, which neither supports nor does not support the correlation. As a whole, we observe that the first category dominates (82.1%), and thus suggests that WARDER's focused precision improvement on spreadsheet cell clustering indeed brings about its corresponding improvement on spreadsheet defect detection.

Fig. 16 shows more details about the precision comparison between WARDER and CUSTODES on spreadsheet cell clustering (Fig. 16a) and defect detection (Fig. 16b). To be focused, we removed those 98 worksheets having their precisions unchanged for both spreadsheet cell clustering and defect detection, and listed only the remaining 42 ones. One can observe from the figure detailed precision changes, as well as their change correlations between cell clustering and defect detection in most cases.

One may notice one exception for worksheet "CO" (the third bar), where WARDER's defect detection precision dropped from 100% to zero, although it improved the cell clustering precision. We further looked into this case. The ground truths suggest that cells {B11, E11} (in green) and {C11, F11} (in orange) should form two clusters, as shown in Fig. 17, and cells C11 and F11 are both defects. CUSTODES detected the two defects *accidentally* by clustering the four cells together. This result is an accident because the four cells actually do not share any common computational semantics (the two green cells calculate the largest value, while the two orange cells calculate the second largest value). CUSTODES considered the two orange cells as defects simply because they contain plain values only (missing formula defect). On the other hand, WARDER clustered {B11, E11} only together and thus did not detect any defect in them. It missed the two orange cells because they do not contain any formula and should not form a cluster. Without any additional evidence (e.g., more cells together and some contain formulas that can unify values in other cells), WARDER played safe by choosing not to form such clusters (otherwise, more false positives could result).

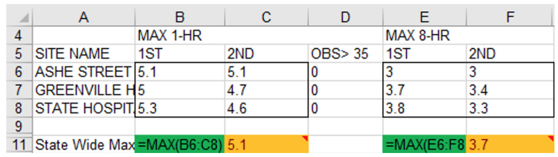


(a) Precision on cell clustering between CUSTODES and WARDER



(b) Precision on defect detection between CUSTODES and WARDER

Figure 16: Precision details in spreadsheet cell clustering and defect detection between WARDER and CUSTODES for worksheets with changed precisions



	A	B	C	D	E	F
4		MAX 1-HR			MAX 8-HR	
5	SITE NAME	1ST	2ND	OBS> 35	1ST	2ND
6	ASHE STREET	5.1	5.1	0	3	3
7	GREENVILLE H	5	4.7	0	3.7	3.4
8	STATE HOSPIT	5.3	4.6	0	3.8	3.3
9						
11	State Wide Max	=MAX(B6:C8)	5.1		=MAX(E6:F8	3.7

Figure 17: Worksheet “CO” (two clusters marked in green and orange, respectively)

Similarly, besides the worksheet-based analysis for the correlation, we also look into the cluster-based correlation analysis. The analysis involves 205 cell clusters, which were reported by both CUSTODES and WARDER, and contain at least one defect both in the ground truths and detected by both techniques (so that the precision correlation between cell clustering and defect detection can be studied). We partition these cell clusters into three categories, as shown in Table. 4, and also observe that the first category (correlation supported) dominates with an even more significant percentage of 96.6% (vs. 82.1% for the worksheet-based analysis). This result further validates the close correlation between WARDER’s improvement on spreadsheet cell clustering and its improvement on spreadsheet defect detection with this finer-grained analysis.

In summary, to answer research question RQ2, we conclude that: *WARDER’s improved spreadsheet cell clustering over CUSTODES indeed contributes to its improved spreadsheet defect detection, and this correlation was supported by 82.1% (worksheet-based) and 96.6% (cluster-based).*

3) RQ3: Individual impacts. Finally, we study the individual impact of WARDER’s three validity-based refinements on its effectiveness in detecting spreadsheet defects. Similarly, we conduct experiments with WARDER-ext for the impact study. For ease of presentation, we also refer to WARDER-ext by WARDER directly in this part. In the experiments, WARDER was configured with each validity-based refinement enabled only (named WARDER-sc, WARDER-mc, WARDER-wc, as mentioned earlier), and compared to the full-fledged WARDER (named WARDER-full). Note that WARDER’s cell retrieval enhancement is also based on its validity properties, and thus can be accordingly split into its different configurations.

Table 5 compares spreadsheet defect detection results for CUSTODES and WARDER’s

Table 4: Correlation study for WARDER against CUSTODES on the precision changes between cell clustering and defect detection in terms of clusters (↑: precision improved, ↓: precision reduced, →: precision unchanged)

Category	**Δ precision (cell clustering)**	**Δ precision (defect detection)**	**# clusters**	**Sum of each category**
Correlation supported	↑	↑	4	198 (96.6%)
	↓	↓	0	
	→	→	194	
Correlation unsupported	↑	→	1	3 (1.5%)
	↑	↓	1	
	↓	→	1	
	↓	↑	0	
Unknown	→	↑	4	4 (2.0%)
	→	↓	0	
Total	-	-	205	205 (100.0%)

Table 5: Defect detection results for CUSTODES and WARDER configured with different validity-based refinements

Technique	**Detected**	**TP**	**FP**	$precision_d$	$recall_d$	$F\text{-}measure_d$
CUSTODES	2,380	1,545	835	64.9%	78.3%	0.71
WARDER-sc	2,311	1,625	686	70.3%	82.3%	0.76
WARDER-mc	2,271	1,575	696	69.4%	79.8%	0.74
WARDER-wc	1,924	1,532	392	79.6%	77.6%	0.79
WARDER-full	1,669	1,488	181	89.2%	75.4%	0.82

four configurations. We observe that: (1) WARDER's each validity-based refinement is useful, and individually improved the precision for defect detection by 4.5−14.7% over CUSTODES, with a recall comparable to that of CUSTODES (in the range of $[-0.7, +4.0]$), leading to an eventual improvement on the F-measure from 0.71 to 0.74~0.79; (2) when combining all the three validity-based refinements together, WARDER-full achieved the highest precision (89.2%) and F-measure (0.82), which are also echoed earlier in Table 1.

In summary, to answer research question RQ3, we conclude that: *WARDER's three validity-based refinements are all useful by individually contributing to its effectiveness on spreadsheet defect detection, and achieve the best effectiveness when combined together.*

4.4. Case Study

Besides the preceding controlled experiments, we also evaluate our WARDER's effectiveness practically in detecting spreadsheet defects using an even larger-scale corpus VEnron2 [56]. VEnron2 contains 1,609 versioned groups, refined from original 79,983 real-life worksheets in the Enron corpus [28]. We chose the latest spreadsheet file from each versioned group, i.e., totally 1,609 spreadsheets, which correspond to a total of 7,140 worksheets as the subjects of our case study. We fed these worksheets to different spreadsheet defect detection techniques/versions to compare their effectiveness on practical spreadsheet defect detection. In the case study, we selected two techniques, namely, CUSTODES and WARDER's both versions (WARDER-ori and WARDER-ext) for comparisons, since they form an evolving family (WARDER-ext extends WARDER-ori, and WARDER-ori extends CUSTODES). In order to facilitate our experimental comparisons and make them fair, we removed some worksheets for which at least one technique/version failed to run normally (e.g., causing an unexpected crash or exception, or exceeding our controlled time limit of five minutes for handling each individual worksheet, so as to avoid being trapped into dead locks or unknown errors). This treatment left us a total of 6,478 worksheets for our case study.

One trouble is that VEnron2 does not contain ground truths for evaluating a spreadsheet defect detection technique's effectiveness (e.g., recall and F-measure can-

Table 6: Defect detection results for the three spreadsheet defect detection techniques/versions on VEnron2 (for sampled 449 worksheets)

Technique	**# defects**	**# TP**	**# FP**	**Precision**
CUSTODES	4,568	1,367	3,201	29.9%
WARDER-ori	2,767	1,123	1,644	40.6%
WARDER-ext	2,460	1,257	1,203	51.1%

Table 7: Defect detection results for the three spreadsheet defect detection techniques/versions on VEnron2 (for all 6,478 worksheets)

Technique	**# reported worksheets**	**# reported defects**	**Time cost (min)**
CUSTODES	1,446	16,279	583
WARDER-ori	1,272	10,665	560
WARDER-ext	1,273	10,352	594

not be calculated). Therefore, we focus mainly on the precision comparison for the three techniques/versions. Considering the large number of all worksheets, although we could run each technique/version on all these worksheets (e.g., for measuring their time costs), we had to use worksheet sampling and manual inspection for measuring the precision. Among all the 6,478 worksheets, 1,498 worksheets were reported to contain defects by at least one studied technique/version. Based on them, we randomly sampled 30% (rounded to 449) worksheets from them for manual inspection, which decided whether each reported defect is a true one or not. Based on the inspection results, Tables 6 and 7 compare the three techniques/versions for their defect detection results.

From Table 6, we observe that: (1) Compared to CUSTODES, WARDER (both versions) achieved higher spreadsheet defect detection precisions, outperforming CUSTODES by 10.7% (for WARDER-ori) and 21.2% (for WARDER-ext), accompanied with much fewer false positives, which echo WARDER's focus on improving the precision over CUSTODES; (2) Among the three techniques/versions, WARDER-ext achieved the highest defect detection precision (i.e., 51.1%), indicating its additional benefits in improving the precision over WARDER-ori (from 40.6% to 51.1%); (3) Although WARDER-ori and WARDER-ext reported relatively fewer true positives (1,123 and 1,257, respectively), which were accompanied with much fewer false positives (1,644 and 1,203), which are 1,557 and 1,998 fewer than that of CUSTODES (3,201), and this feature can be quite useful since all spreadsheet defects have to be manually verified later in practice.

From Table 7, we observe that, similar to the sampled 449 worksheets, WARDER (both versions) reported fewer spreadsheet defects (10,665 for WARDER-ori, and 10,352 for WARDER-ext), as compared to 16,279 for CUSTODES. Considering that WARDER-ext achieved the highest precision, its report quality is expected to be high (e.g., in 2,460 defects WARDER-ext detected 1,257 true positives, while in 4,568 (about 1.9 times) defects CUSTODES detected only 1,367 true positives (about 1.1 times)). Regarding the efficiency (time cost), we note that WARDER was based on CUSTODES and improved its spreadsheet cell clustering only (one stage of the total four, as shown in Fig. 1). Therefore, WARDER's time costs were close to that of CUSTODES, e.g., 560 minutes for WARDER-ori and 594 minutes for WARDER-ext, as compared to 583 minutes for CUSTODES. It is understandable that WARDER-ori cost less time than CUSTODES, since WARDER-ori filtered out irrelevant cells and unqualified clusters, thus reducing unnecessary workloads associated with these cells and clusters in later stages in the workflow (Stages 3 and 4, as shown in Fig. 1). Although WARDER-ext conducted more refinements, which can further reduce the time cost as WARDER-ori did, it also conducted cell retrieval enhancement, which on one hand itself cost more time, and on the other hand added some cells into its clusters and thus increased the workload for later stages. Still, WARDER's both versions took comparable time as CUSTODES (only 3.9% less and 1.9% more, respectively).

As a conclusion, *WARDER is satisfactory in detecting defects for practical spreadsheets. Both versions (WARDER-ori and WARDER-ext) achieved higher precisions (40.6% and 51.1%), outperforming CUSTODES by 10.7% and 21.2%, respectively. Their time costs are comparable to CUSTODES (within the 4% difference).*

In the following, we discuss some aspects that may threaten the validity of our experimental conclusions.

One threat concerns the calculation of spreadsheet cell clustering metrics (i.e., $precision_c$, $recall_c$, and $F\text{-}measure_c$) introduced in Section. 4.2. They are based on the TP, FP, and FN measurements calculated from CUSTODES's pair-wise similarity comparisons [13]. We note that such calculations count the numbers of spreadsheet cell pairs on whether they belong to the same cluster or different clusters. They are thus different from those measuring detected spreadsheet defects, since the latter can be counted naturally one by one. As a result, studying the correlation between WARDER's cell clustering and its spreadsheet defect detection could be affected to some extent. To alleviate this threat, we studied the correlation by both worksheet-based and cluster-based analyses. We observed that over 80% worksheets and 90% clusters support our studied correlation, and this suggests that WARDER's improved cell clustering indeed generally contributes to its spreadsheet defect defection, as we expected earlier.

One may notice that WARDER still has room for improvement, considering that it failed to detect few spreadsheet defects, as we analyzed earlier. The major reason is that WARDER has focused on improving spreadsheet cell clustering, by filtering out irrelevant cells and unqualified clusters, but it itself does not refine the anomaly detection part, which directly relates to spreadsheet defect detection. As a result, WARDER would suffer the same problems with the anomaly detection part inherited from CUSTODES due to the latter's focused scope. Nevertheless, we observed in experiments that WARDER already outperformed CUSTODES largely, and this suggests that WARDER has focused on a dominating factor for the effectiveness improvement on defect detection. Still, the above analysis points out new directions that might deserve future efforts.

We note that we attempted but did not manage to compare WARDER to the other two learning-based techniques, Melford [52] and ExceLint [9] in our experiments. For the former, we did not find its tool available. For the latter, we found its tool but encountered problems in the experiments. First, ExceLint's scope is very different from those of the other seven spreadsheet defect detection techniques/versions studied in our experiments, in that it focuses on detecting part of inconsistent formula defects that have been caused by wrong references. Second, ExceLint skips detecting missing formula defects, since they may not trigger errors immediately. However, all the other techniques in our experiments consider such defects harmful and detect them, since such defects can trigger unexpected errors when the concerned spreadsheets undergo future maintenance. In fact, missing formula defects are common in practical spreadsheets (e.g., 79–81% in the VEnron2 corpus by different techniques in our experiments). Therefore, directly comparing WARDER with ExceLint could be unfair and possibly seriously underestimate ExceLint's effectiveness. Besides, we also encountered problems when running ExceLint as it lacked a specifically-annotated ground truth as its runtime support. Therefore, we had to leave out its comparison in our experiments.

Similarly, some other techniques share a different focus as we studied in this article, and thus we did not compare them experimentally. For example, some techniques focus mainly on spreadsheet smells (e.g., formula smells [30, 31], input smells [16, 15], inter-worksheet smells [29], and their unions [6, 5]), which concern syntactic issues like code smells [23] with spreadsheets, which differ from our focus of missing formula and inconsistent formula defects (essentially semantic issues). This is echoed by AmCheck's [19], CACheck's [20], and CUSTODES's [13] analyses in their experimental designs. Nevertheless, we consider that the two lines of work are both beneficial for spreadsheet quality assurance, and can be combined as collaborative assistance to spreadsheet users.

5. Related Work

Spreadsheet quality issues are common. Spreadsheets can contain various defects [46, 47, 42, 44], and these defects can cause catastrophic losses to human daily lives [48, 1, 43]. Galletta et.al [24] conducted an empirical study on spreadsheets, and reported that even spreadsheet experts cannot significantly outperform novices in identifying spreadsheet defects. This result suggests that identifying spreadsheet defects can be a non-trivial research problem.

Empirical evidences also support that even simple spreadsheet auditing tools can be quite useful in practice. For example, Nixon and O'Hara [41] reported a positive assistance by supporting auditing in spreadsheet maintenance tasks. Later, Anderson [7] confirmed the usefulness of such assistance, but also raised a concern for numerous missed spreadsheet defects. To better understand spreadsheet cell relations and maintain the spreadsheet quality, Mittermeir et al. [14, 40] proposed three types of "logical areas" for clustering those formula cells that satisfy three forms of equivalences, namely, copy, logical, and structural equivalences, respectively. Such clustering can help spreadsheet users better understand conceptual models behind spreadsheets, and avoid or inspect defective cells more easily.

Theoretically, each formula cell in a spreadsheet can be regarded as a piece of software (i.e., an end-user domain-specific language based program). Thereby, finding errors in formula cells (i.e., defects studied in this work) is a typical software validation and verification task. On the other hand, the nature of spreadsheet formulas [47] brings new research opportunities of effectively and efficiently identifying and diagnosing spreadsheet defects for end users [44].

Spreadsheet defect detection and prediction. The heart of spreadsheet defect detection and prediction tasks traces back to the idea of metric-based defect prediction [10] (a.k.a., code smell, as introduced by Fowler [23]) and "bugs as deviant behavior" [21]. Both have motivated a series of subsequent software bug-hunting work, by assuming that: (1) buggy formulas in spreadsheets have their distinctive characteristics, and (2) most formula cells are correct, ill-looking or deviant formulas can be identified as potential defects.

We have witnessed rapid research developments for better quantitative characterization of spreadsheet formulas by metric-based analyses. UCheck [3, 4] and Dimension [12] are probably two representative pioneers on this aspect. Based on unit or dimensional information derived from spreadsheet tables, they verified the correctness of formula calculations by checking whether there exists any illegal combination of incompatible units. Then, Hermans et al. implemented portable tools to detect and visualize several types of spreadsheet defects by focusing on inter-worksheet smells [29], data clones [32], and formula smells [30, 31]. They are the first to adapt the concept of code smell in conventional programs to the spreadsheet domain. Almost meanwhile, Cunha et al. extended the category of smells, by focusing on statistical smells, type smells, content smells, and functional dependencies based smells [16, 15]. After that, Abreu et al. [6, 5] integrated various categories of spreadsheet smells, and used a generic spectrum-based strategy to localize defects and improved the localization precision and recall for spreadsheet smells. Recently, Koch et al. [35] further refined some of the existing spreadsheet smells with structural analysis.

Hofer et al. [33], [34] further studied the impact of different similarity coefficients on the accuracy of spectrum-based spreadsheet defect localization. Meanwhile, AmCheck [19] and its follow-up extension CACheck [20] were proposed to support effective defect detection by focusing on spreadsheet flaws caused by ambiguous computation semantics based on the notion of cell array. Similarly, Xu et al. [57] proposed to detect defective empty cells in spreadsheets by analyzing the context of empty cells.

The WARDER technique, proposed in this work, is similarly based on the adaptive learning mechanism in CUSTODES [13], using formula-related spreadsheet cell clustering and anomaly-based defect detection. CUSTODES also provided a spreadsheet benchmark to facilitate follow-up research evaluations. Two follow-up techniques built on this benchmark are Melford [52] and ExceLint [9]. The former used network-based modeling to detect missing formula defects, while the latter used statistical calculations to measure the likelihood of a spreadsheet defect based on the entropy and layout of its associated references for detecting inconsistent formula (or reference) defects. Similar inconsistency issues can also raise concerns for general software in many fields, e.g., context inconsistency detection for adaptive applications [54], inconsistency management for software engineering [53], etc.

Realizing that cell clustering plays a central role in effectively identifying spreadsheet defects, WARDER makes attempts to improve spreadsheet defect detection by refining CUSTODES's cell clustering based on cell-level and cluster-level validity properties, as we presented in this article.

Spreadsheet defect fixing and prevention. Spreadsheet defect detection and prediction techniques identify anti-patterns or deviations as defect evidences, which can also be used as fixing suggestions. For example, besides detecting spreadsheet smells, CACheck [20] also proposed to automatically repair its detected smells by synthesizing and recovering intended computational semantics in terms of formulas for the concerned cells.

As spreadsheet data and formulas are continually evolving over time, the maintenance of spreadsheets is also an important issue. To better understand the evolution of spreadsheets, Harutyunyan et al. [27] proposed to automatically identify differences between spreadsheet versions so that maintenance can be more reliably conducted. Badame and Dig [8] proposed to obtain different measures on spreadsheet formulas, so that these formulas can be better refactored for maintenance tasks. Luckey et al. [39] attempted to prevent spreadsheet defects by supporting safer spreadsheet evolution with correctness guarantee.

In WARDER, the formula differences between a defective cell and its associated cluster can be used as defect fixing or prevention hints. With such hints, search-based program repairing [55] may be incorporated for synthesizing fixing suggestions.

Other spreadsheet-related research. There are also some pieces of work focused on other spreadsheet-related research (e.g., understandability, programming environment, and formula synthesis). For example, Zhang et al. [58] proposed an automated approach to improving the expression for nested-IF formulas in spreadsheets by removing logic redundancy, so that high-level formula semantics can be more easily identified for end users' better understanding. Cunha et al. [17] aimed at helping build a more reliable spreadsheet programming environment.

Finally, program synthesis techniques are becoming increasingly popular in the spreadsheet domain, and many pieces of research are increasingly proposed for solving spreadsheet-specific problems, such as automating table transformation [26], string synthesis [25], and number transformation [51] tasks by a programming-by-example approach. Essentially, program synthesis is a powerful tool for deducing hidden relations and structures in spreadsheets, e.g., inferring formulas from data. Samuel et al. [36] modeled common spreadsheet formulas and relations through predicates and expressions, and used a two-stage approach to generating and testing spreadsheets by constraint solving, so that constraints across spreadsheet cells can be synthesized.

Synthesized formulas can also be used as a reference in a wide variety of spreadsheet analysis tasks, e.g., providing reduced canonical form of formulas. This could be a promising research direction for spreadsheet quality assurance.

6. Conclusion

In this article, we studied the problem of spreadsheet defect detection. We presented WARDER for refining CUSTODES's cell clustering in order to improve its effectiveness in detecting spreadsheet defects. WARDER is based on our key observations that rely on our identified three validity properties to prevent problematic clusters from being formed, which can involve irrelevant cells and unqualified clusters. These properties concern different levels of cluster validities, from single-cell, multi-cell, to whole-cluster validities, whose uses effectively contribute to the improved precision of spreadsheet cell clustering and defect detection. In addition, WARDER's validity-based cell retrieval enhancement further strengthens its effectiveness. Our experimental evaluation with a known benchmark and a case study with a large-scale spreadsheet corpus have confirmed WARDER's effectiveness in detecting spreadsheet defects, in particular on the detection precision.

WARDER leaves some opportunities for further improvement with future extensions. For example, its validity framework is flexible, allowing for further extensions with more instantiations of validity properties. Besides, currently WARDER focuses on improving spreadsheet cell clustering, by filtering out irrelevant cells and unqualified clusters, and it can be extended for improving the anomaly detection part by referring to better algorithms as we discussed earlier. We are working along these lines.

Acknowledgement

The authors wish to thank the editors and anonymous reviewers for their valuable comments on improving this article. This work was supported in part by National Key R&D Program (Grant #2017YFB1001801) and National Natural Science Foundation (Grants #61932021, #61690204, #61802165) of China. The authors would also like to thank the support of the Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China.

References

[1] `http://www.eusprig.org/index.htm`, [Online; accessed 8-March-2019], 2019.

[2] `https://poi.apache.org/`, [Online; accessed 8-March-2019], 2019.

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

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

[5] R. Abreu, J. Cunha, J. P. Fernandes, P. Martins, A. Perez, J. Saraiva, FaultySheet Detective: When Smells Meet Fault Localization, in: 2014 IEEE International Conference on Software Maintenance and Evolution, IEEE, 625–628, 2014.

[6] R. Abreu, J. Cunha, J. P. Fernandes, P. Martins, A. Perez, J. Saraiva, Smelling faults in spreadsheets, in: 2014 30th IEEE International Conference on Software Maintenance and Evolution, IEEE, 111–120, 2014.

[7] W. Anderson, A comparison of automated and manual spreadsheet error detection, Ph.D. thesis, Master thesis, Massey University, 2004.

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

[9] D. W. Barowy, E. D. Berger, B. Zorn, ExceLint: Automatically Finding Spreadsheet Formula Errors, vol. 2, ACM, New York, NY, USA, ISSN 2475-1421, 148:1–148:26, 2018.

[10] V. R. Basili, L. C. Briand, W. L. Melo, A validation of object-oriented design metrics as quality indicators, IEEE Transactions on Software Engineering 22 (10) (1996) 751–761.

[11] P. Carey, K. Berk, Data Analysis with Microsoft Excel, Brooks/Cole, 1997.

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

[13] S.-C. Cheung, W. Chen, Y. Liu, C. Xu, CUSTODES: automatic spreadsheet cell clustering and smell detection using strong and weak features, in: Proceedings of the 38th International Conference on Software Engineering, ACM, 464–475, 2016.

[14] M. Clermont, Auditing large spreadsheet programs, in: In International Conference on Information Systems Implementation and Modelling, Citeseer, 306–315, 2003.

[15] J. Cunha, J. P. Fernandes, P. Martins, J. Mendes, J. Saraiva, SmellSheet detective: A tool for detecting bad smells in spreadsheets, in: Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC, IEEE, 243–244, 2012.

[16] J. Cunha, J. P. Fernandes, H. Ribeiro, J. Saraiva, Towards a catalog of spreadsheet smells, in: Proceedings of the 12th International Conference on Computational Science and Its Applications, Springer Berlin Heidelberg, 202–216, 2012.

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

[18] W. Dou, S.-C. Cheung, C. Gao, C. Xu, L. Xu, J. Wei, Detecting table clones and smells in spreadsheets, in: Proceedings of the 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2016), Seattle, WA, USA, 787–798, Nov 2016.

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

[20] W. Dou, C. Xu, S.-C. Cheung, J. Wei, CACheck: detecting and repairing cell arrays in spreadsheets, IEEE Transactions on Software Engineering 43 (3) (2017) 226–251.

[21] D. Engler, D. Y. Chen, S. Hallem, A. Chou, B. Chelf, Bugs As Deviant Behavior: A General Approach to Inferring Errors in Systems Code, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles, SOSP '01, ACM, New York, NY, USA, ISBN 1-58113-389-8, 57–72, 2001.

[22] M. Fisher, G. Rothermel, The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms, ACM SIGSOFT Software Engineering Notes 30 (4) (2005) 1–5.

[23] M. Fowler, Refactoring: Improving the design of existing code, Addison-Wesley, 1999.

[24] D. F. Galletta, D. Abraham, M. El Louadi, W. Lekse, Y. A. Pollalis, J. L. Sampler, An empirical study of spreadsheet error-finding performance, Accounting, Management and Information Technologies 3 (2) (1993) 79–95.

[25] S. Gulwani, Automating string processing in spreadsheets using input-output examples, ACM SIGPLAN Notices 46 (1) (2011) 317–330.

[26] W. R. Harris, S. Gulwani, Spreadsheet table transformations from examples, ACM SIGPLAN Notices 46 (6) (2011) 317–328.

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

[28] F. Hermans, E. Murphy-Hill, Enron's spreadsheets and related emails: A dataset and analysis, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, IEEE, 7–16, 2015.

[29] F. Hermans, M. Pinzger, A. v. Deursen, Detecting and visualizing inter-worksheet smells in spreadsheets, in: Proceedings of the 34th International Conference on Software Engineering, IEEE Press, 441–451, 2012.

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

[31] F. Hermans, M. Pinzger, A. van Deursen, Detecting and refactoring code smells in spreadsheet formulas, Empirical Software Engineering 20 (2) (2015) 549–575.

[32] F. Hermans, B. Sedee, M. Pinzger, A. van Deursen, Data clone detection and visualization in spreadsheets, in: 2013 35th International Conference on Software Engineering, ICSE, IEEE, 292–301, 2013.

[33] B. Hofer, A. Perez, R. Abreu, F. Wotawa, On the empirical evaluation of similarity coefficients for spreadsheets fault localization, Autom. Softw. Eng. 22 (1) (2015) 47–74.

[34] B. Hofer, A. Riboira, F. Wotawa, R. Abreu, E. Getzner, On the empirical evaluation of fault localization techniques for spreadsheets, in: Fundamental Approaches to Software Engineering - 16th International Conference, FASE 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings, 68–82, 2013.

[35] P. Koch, B. Hofer, F. Wotawa, On the refinement of spreadsheet smells by means of structure information, Journal of Systems and Software 147 (2019) 64–85.

[36] S. Kolb, S. Paramonov, T. Guns, L. De Raedt, Learning constraints in spreadsheets and tabular data, Machine Learning 106 (9-10) (2017) 1441–1468.

[37] B. R. Lawson, K. R. Baker, S. G. Powell, L. Foster-Johnson, A comparison of spreadsheet users with different levels of experience, Omega 37 (3) (2009) 579–590.

[38] D. Li, H. Wang, C. Xu, F. Shi, X. Ma, J. Lu, WARDER: Refining cell clustering for effective spreadsheet defect detection via validity properties, in: Proceedings of the 19th IEEE International Conference on Software Quality, Reliability, and Security (QRS 2019), 139–150, Sofia, Bulgaria, Jul 2019.

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

[40] R. Mittermeir, M. Clermont, Finding high-level structures in spreadsheet programs, in: Ninth Working Conference on Reverse Engineering, 2002. Proceedings., IEEE, 221–232, 2002.

[41] D. Nixon, M. O'Hara, Spreadsheet auditing software, arXiv preprint arXiv:1001.4293 .

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

[43] R. Panko, What we don't know about spreadsheet errors today: The facts, why we don't believe them, and what we need to do, arXiv preprint arXiv:1602.02601 .

[44] R. R. Panko, Spreadsheet errors: What we know. What we think we can do, arXiv preprint arXiv:0802.3457 .

[45] R. R. Panko, S. Aurigemma, Revising the Panko–Halverson taxonomy of spreadsheet errors, Decision Support Systems 49 (2) (2010) 235–244.

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

[47] K. Rajalingham, D. R. Chadwick, B. Knight, Classification of spreadsheet errors, arXiv preprint arXiv:0805.4224 .

[48] C. M. Reinhart, K. S. Rogoff, Growth in a Time of Debt, American Economic Review 100 (2) (2010) 573–78.

[49] J. Sajaniemi, Modeling spreadsheet audit: A rigorous approach to automatic visualization, Journal of Visual Languages & Computing 11 (1) (2000) 49–82.

[50] A. O. Shigarov, A. A. Mikhailov, Rule-based spreadsheet data transformation from arbitrary to relational tables, Information Systems 71 (2017) 123–136.

[51] R. Singh, S. Gulwani, Synthesizing number transformations from input-output examples, in: International Conference on Computer Aided Verification, Springer, 634–651, 2012.

[52] R. Singh, B. Livshits, B. Zorn, Melford: Using neural networks to find spreadsheet errors, Tech. Rep. .

[53] G. Spanoudakis, A. Zisman, Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals, chap. Inconsistency management in software engineering: Survey and open research issues, World Scientific, 329–380, 2001.

[54] H. Wang, C. Xu, B. Guo, X. Ma, J. Lu, Generic adaptive scheduling for efficient context inconsistency detection, IEEE Transactions on Software Engineering, 2019 .

[55] W. Weimer, T. Nguyen, C. Le Goues, S. Forrest, Automatically finding patches using genetic programming, in: Proceedings of the 31st International Conference on Software Engineering, ICSE, 364–374, 2009.

[56] L. Xu, W. Dou, C. Gao, J. Wang, J. Wei, H. Zhong, T. Huang, SpreadCluster: recovering versioned spreadsheets through similarity-based clustering, in: Proceedings of the 14th International Conference on Mining Software Repositories, IEEE Press, 158–169, 2017.

[57] L. Xu, S. Wang, W. Dou, B. Yang, C. Gao, J. Wei, T. Huang, Detecting faulty empty cells in spreadsheets, in: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 423–433, 2018.

[58] J. Zhang, S. Han, D. Hao, L. Zhang, D. Zhang, Automated refactoring of nested-IF formulae in spreadsheets, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 833–838, 2018.

[59] R. Zhang, C. Xu, S.-C. Cheung, P. Yu, X. Ma, J. Lu, How effectively can spreadsheet anomalies be detected: An empirical study, Journal of Systems and Software 126 (2017) 87–100.