SGUARD: A Feature-based Clustering Tool for Effective Spreadsheet Defect Detection

Da Li[†§], Huiyan Wang[†§], Chang Xu[†‖], Ruiqing Zhang[¶‡‡], Shing-Chi Cheung[‡††], Xiaoxing Ma[†‖]
[†]State Key Lab. for Novel Software Tech. and Dept. of Comp. Sci. and Tech., Nanjing University, Nanjing, China
[¶]Search Tech. Center Asia, Microsoft, Suzhou, China
[‡]Dept. of Comp. Sci. and Engr., The Hong Kong University of Sci. and Tech., Hong Kong, China
[§]{lidanuaa, cocowhy1013}@gmail.com, [‖]{changxu, xxm}@nju.edu.cn, [‡‡]zhangrq3216@163.com, [††]scc@cse.ust.hk

***Abstract*—Spreadsheets are widely used but subject to various defects. In this paper, we present SGUARD to effectively detect spreadsheet defects. SGUARD learns spreadsheet features to cluster cells with similar computational semantics, and then refine these clusters to recognize anomalous cells as defects. SGUARD well balances the trade-off between the precision (87.8%) and recall rate (71.9%) in the defect detection, and achieves an F-measure of 0.79, exceeding existing spreadsheet defect detection techniques. We introduce the SGUARD implementation and its usage by a video presentation (https://youtu.be/gNPmMvQVf5Q), and provide its public download repository (https://github.com/sheetguard/sguard).**

***Index Terms*—Cell clustering, Defect detection**

I. INTRODUCTION

Nowadays, spreadsheets are being widely used, and have become one of the most popular end-user environments [1]. However, despite the popularity, spreadsheets are found to be error-prone [2], especially for their formula-related cells. We name these errors in the concerned formulas *defects* in spreadsheets, which are also found to be root causes for many other errors [3], e.g., errors in data cells. This calls for effective spreadsheet defect detection, which is still challenging for two major reasons. First, spreadsheets are typically maintained by non-programmer end-users, and are usually not supported by auditing and tracking services [4]. Second, spreadsheet cells own hidden semantic relationships among them, which lead to difficulties in reasoning for spreadsheet defects.

To address these challenges, various techniques have been proposed [5–8]. Some rely on header information in spreadsheets to infer type inconsistencies in their formula references, e.g., UCHECK [5] and DIMENSION [6], but the inference focuses only on a limited scope, leading to an unsatisfactory precision and recall rate [9]. Some others exploit specific patterns (e.g., rectangle areas) to recognize missing or inconsistent formulas and achieve a much better precision in spreadsheet defect detection, e.g., AMCHECK [7] and CACHECK [8], but their relied patterns are static and do not adapt to varying styles in different spreadsheets, leading to a compromised recall rate [8].

In this paper, we present SGUARD, an effective tool for detecting spreadsheet defects. SGUARD builds on our previous efforts, CUSTODES [10] (ever considered as "best automated error finder" [11]) and WARDER [12], and can adaptively learn varying styles (features) of different spreadsheets for cell clustering, from which it identifies anomalous cells as defects. SGUARD consists of three main components, i.e., *cell clustering*, *cluster refinement*, and *defect detection*. The first component uses a two-stage clustering technique to cluster cells with similar computational semantics based on their shared strong (e.g., formula semantics) and weak features (e.g., layout information), aiming for a high recall rate. The second component refines the clustering results by squeezing out irrelevant cells from clusters and removing unquantified clusters, aiming for a high precision. The third component analyzes the refined clusters and detect outliers in each cluster as defects to be presented to end users, aiming for providing useful bug information. The SGUARD tool is fully automated with one button-click execution.

We experimentally evaluated SGUARD on the widely-used EUSES corpus [13]. Regarding defect detection, SGUARD achieved a precision of 87.8% and recall rate of 71.9%, against 0.5–72.3% and 0.1–68.4% for other techniques, respectively. We also evaluated SGUARD on the latest spreadsheet corpus VEnron2 [14]. The results show that SGUARD exhibited its unique superiority over existing techniques (41.3% against 16.3–34.3% on the precision of spreadsheet defect detection).

In the following, we present an illustrative example (Section II), elaborate on SGUARD's methodology (Section III), introduce its implementation (Section IV), evaluate its performance (Section V), and concludes the paper (Section VI).

II. ILLUSTRATIVE EXAMPLE

Fig. 1 illustrates part of an example worksheet (a page in a spreadsheet) adapted from the EUSES corpus [13]. It contains three cell clusters, each of which follows a specific computational semantics, as annotated by three colors (green, orange, and blue). Among them, five cells contain faulty formulas (defects), namely, D15, D19, F13, F16, and F19, as annotated by red triangles. To facilitate the discussion, we specify eight blocks (containing one or multiple cells by red dashed lines) by letters from A to H. We consider different spreadsheet defect detection techniques below [1].

[1]We exclude UCHECK and DIMENSION from the discussion since they neither have the cluster concept nor report any defect for this example.

	A	B	C	D	E	F
7			DEPOSITS/SHAR		LOANS	
8			Dollars	% of	Dollars	% of
9		No.	(000's)	Total A	(000's)	Total B
10	Trust Companies	9	1547458	=(C10/C$19)*100	1377629	=(E10/E$19)*100
11	National Banks*	7	7440908	=(C11/C$19)*100	6508230	=(E11/E$19)*100
12	State Savings Banks	15	5010519	=(C12/C$19)*100	4859363	=(E12/E$19)*100
13	Federal Savings Banks	2	739898	=(C13/C$19)*100	859251	5.3
14	State Savings and Loans	3	103550	=(C14/C$19)*100	107427	=(E14/E$19)*100
15	Federal Savings and Loa	4	206822	1.15	211442	=(E15/E$19)*100
16	State Credit Unions	15	711205	=(C16/C$19)*100	568652	3.5
17	Federal Credit Unions	63	2127767	=(C17/C$19)*100	1735908	=(E17/E$19)*100
18				C		
19	TOTAL	=SUM(B10:B18)	=SUM(C10:C18)	100	=SUM(E10:E18)	=SUM(F10:F17)
20		D	E	F	G	H
28	Out-of-State Ownership	4	3782155	=(C28/C30)*100	2823577	=(E28/E30)*100

CUSTODES's FP

CACheck's FP

Fig. 1: An illustrative example adapted from EUSES

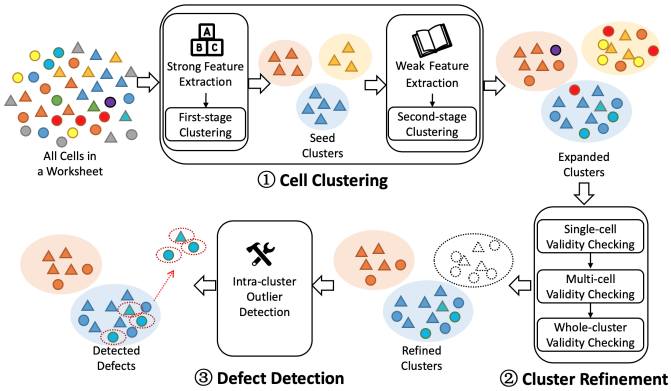

Fig. 2: Workflow of SGUARD

When one applies AMCHECK, it relies on fixed rectangle patterns to recognize clusters and reports one cell cluster (block C), missing the other two (blocks A+B and F+H). As a result, it detects only two defects (D19 and F19), missing the other three (false negatives). When one applies CACHECK, it is extended with generic rectangle patterns and reports four cell clusters (A, B, C, E+F+G+H). Note that it considers A and B as two separate cell clusters, and wrongly considers blocks E and G. As a result, CACHECK wrongly reports C28 and E28 as defects (false positives). When one applies CUSTODES (SGUARD's predecessor), it learns varying features in spreadsheets to cluster cells and does not restrict to rectangle areas. It reports three cell clusters (A+B, C, D+F+H), which are much more precise. Still, as affected by the irrelevant cell B28 (block D), CUSTODES wrongly reports B28 as defect (false positive). Finally, when one applies SGUARD (integrating CUSTODES + WARDER), it correctly recognizes three precise cell clusters, and detect five true defects exactly.

III. SGUARD METHODOLOGY

Fig. 2 illustrates SGUARD's three main components, namely, cell clustering, cluster refinement, and defect detection, as aforementioned, and we explain their details below.

A. Component 1: Cell Clustering

The first component uses a two-stage cell clustering technique, aiming to cluster cells with similar computational semantics together based on their shared features.

In the first stage, SGUARD forms cell clusters according to these cells' shared strong features (e.g., cell formulas and their reference relations). These clusters are also known as seed clusters to be expanded later. Specifically, when considering strong features, all cell formulas are parsed into two tree structures, namely, abstract syntax tree (AST) and cell dependency tree, with each node containing its referenced cells, values, or operations. Based on such structures, two cell formulas can be compared by their tree similarity measurement according to a tree editing distance algorithm [15]. During the clustering process, each formula cell initially forms a cluster with itself and then iteratively joins other clusters based on a standard hierarchical agglomerative clustering algorithm [16] according to their measured similarities. SGUARD considers thus obtained clusters with at least two formula cells as seed clusters.

In the second stage, the seed clusters are expanded to include those cells that are yet not in any cluster (data cells and remaining formula cells), as long as these cells share similar weak features with cells already in some seed cluster. SGUARD considers six weak features, namely, cell address, label [5], alliance [17], table [18], rectangle cell range [7], and gap template [18]. SGUARD considers a cell cluster to be expanded with more cells according to a distance measurement based on a bootstrapping algorithm as inspired by existing work [19]. By doing so, SGUARD both identifies cells with similar computational semantics and retrieves back more cells that were previously isolated due to their contained defects, expecting for a high recall rate in later defect detection.

B. Component 2: Cluster Refinement

The second component refines the cell clusters obtained from cell clustering, considering that the clustering could be too aggressive and accidentally include irrelevant cells or form unqualified clusters. SGUARD refines these cell clusters based on three validity properties, namely, single-cell, multi-cell, and whole-cluster validities.

For the single-cell property, SGUARD focuses on the quality of each cell in a cluster, and requests the cell to be valid (e.g., any cell should not cite a wrong place or contain an invalid reference). For the multi-cell property, SGUARD focuses on cell relations in each cluster and its newly added cells, and requests all concerned cells to stick to similar characteristics (e.g., cell reference overlapping relations among cells in a cluster should not be violated by any newly added cell). For the whole-cluster property, SGUARD focuses on the quality of each whole cell cluster, and requests most cells in it to share a similar computation semantics [8] (e.g., any cell cluster should be able to generate a unique formula expression that fits most of its included cell formulas).

SGUARD applies these three properties to refine cell clusters, and filter out those irrelevant cells or unqualified clusters (when violating any property). By doing so, SGUARD expects for a high precision in later spreadsheet defect detection.

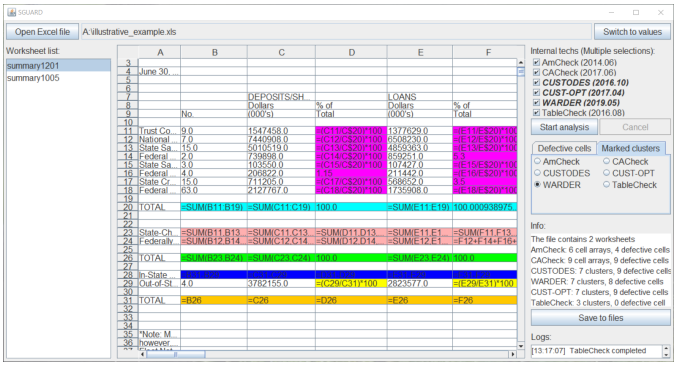

Fig. 3: SGUARD usage screenshot

C. Component 3: Defect Detection

The last component performs outlier detection to identify defects in each cell cluster and report them to end users.

Specifically, in each formed cluster, any data cell is reported to contain a missing formula defect since it can be unified with other cells in the same cluster by the same formula expression, and all formula cells are measured by a local outlier detection algorithm [20] with AST structures to classify outlier ones as containing inconsistent formula defects. All detected defects are reported to end users with fine-grained annotations (e.g., clusters marked in different colors and defects marked by red triangles in Fig. 1). Such annotations are helpful in assisting later manual confirmation and bug-fixing.

IV. IMPLEMENTATION AND USAGE

We implemented the SGUARD tool in Java. It uses Apache POI [21] to manipulate spreadsheets. Its implementation contains a total of 10,500 lines of code, including 7,300 lines of core code and 3,200 lines of graphical interface related code. Fig. 3 gives a screenshot of using SGUARD for detecting spreadsheet defects in an interactive way.

We briefly introduce SGUARD's usage. First, to select a specific Excel spreadsheet file for analysis, a user clicks the "Open Excel File" button in the menu bar and then all its associated worksheets would be listed in the left "Worksheet list" area for watch. After that, the user can select one or multiple spreadsheet defect detection techniques from the right "Internal techs" panel to analyze the worksheets in the opened spreadsheet. Here, note that SGUARD not only implemented its own CUSTODES ("CUSTODES" for its published version in [10] and "CUST-OPT" for its latest version [22]) and WARDER [12] techniques as we mentioned before, but also integrated three existing popular defect detection techniques, i.e., AMCHECK [7], CACHECK [8], and TABLECHECK [23], so that its user can easily choose different techniques for comparison. Then, with a worksheet selected from the left panel and (multiple) technique(s) selected from the right panel, the user can click "Start analysis" to start SGUARD's defect detection process. When the analyses are complete, the results (i.e., worksheet with colored annotations) would be shown in the central panel for the selected worksheet. Then one can choose to inspect certain results (cell clusters annotated by different colors and defects annotated by red colors specially) by clicking "Defective cells" or "Marked clusters" buttons in the right middle panel. Moreover, users can also easily save the detection results (worksheets with annotations) into files by clicking the "Save to files" button for later analyses or reuse. Some logging information would also be shown during the SGUARD execution (e.g., "Info" and "Logs" panels) for deeper investigation.

SGUARD is fully automated. Everything a user needs is almost one button click.

TABLE I: Defect detection results for the six spreadsheet defect detection techniques on EUSES [12]

Technique	Detected	TP	FP	*precision*	*recall*	*F-measure*
UCHECK	204	1	203	0.5%	0.1%	0.00
DIMENSION	1,824	14	1,828	0.8%	0.7%	0.01
AMCHECK	2,372	1,316	1,030	56.1%	66.7%	0.61
CACHECK	1,866	1,350	516	72.3%	68.4%	0.70
CUSTODES	2,380	1,539	841	64.7%	78.2%	0.71
SGUARD	1,612	1,415	197	87.8%	71.9%	0.79

V. EVALUATION

A. Experimental Setup

We evaluate our SGUARD and compare it to existing techniques, including UCHECK, DIMENSION, AMCHECK, CACHECK, and CUSTODES (SGUARD's predecessor). We note that we used CUSTODES's latest version CUST-OPT in our experiment. We selected a refined sample set [10] from the EUSES corpus [13] as our evaluation benchmark, which contains 70 spreadsheets and embedded 291 worksheets. These worksheets contain 189,027 cells, among which 26,716 are formula cells. The benchmark also contains ground truths, which annotate 1,610 cell clusters and among them 1,974 defects (faulty cells with missing or inconsistent formulas). We also compare four leading techniques (AMCHECK, CACHECK, CUSTODES, and SGUARD) on a larger corpus VEnron2 [14] (containing 6,258 worksheets after selecting the latest spreadsheet in its each evolution group) to investigate their practical usefulness in spreadsheet defect detection. All experiments were conducted on a commodity PC with an Intel Core™ i7-6700 CPU @3.41GHz with 64GB RAM, installed with Microsoft Windows 10 Professional and Oracle Java 8.

B. Experimental Results

Table I compares the defect detection results for all the six techniques on the selected EUSES spreadsheets. We observe that UCHECK and DIMENSION achieved a quite unsatisfactory precision and recall rate, leading to a very low F-measure (0.00–0.01), due to their limited analysis scopes. The other four techniques have different strengths in spreadsheet detect detection. For example, AMCHECK and CACHECK significantly improved the detection effectiveness by cell array based patterns (F-measure of 0.61–0.70), and CACHECK additionally improved the precision to 72.3% over that of its predecessor AMCHECK (56.1%), but their recall rates were

TABLE II: Defect detection results for the four spreadsheet defect detection techniques on VEnron2 [12]

	For all 6,258 worksheets		For sampled 300 worksheets		
Technique	**# reported defects**	**Time cost (min)**	**# defects**	**# TP**	**Precision**
AMCHECK	20,280	21	3,316	540	16.3%
CACHECK	12,953	372	1,559	534	34.3%
CUSTODES	14,102	537	2,334	629	26.9%
SGUARD	9,462	518	1,240	512	41.3%

restricted to a 66.7–68.4% level. CUSTODES and SGUARD, instead, achieved a higher recall rate of 71.9–78.2% by their feature-based cell clustering, and SGUARD additionally improved the precision to 87.8% over that of its predecessor CUSTODES (64.7%) by its dedicated cluster refinement via validity properties. Altogether, SGUARD achieved the highest F-measure of 0.79 against 0.00–0.71 of other techniques.

Table II compares the defect detection results for four leading techniques (AMCHECK, CACHECK, CUSTODES, and SGUARD) on the VEnron2 spreadsheets, which consists of two parts: complete 6,258 worksheets and randomly sampled 300 worksheets for precision comparison [12]. From the second part, we observe that: (1) SGUARD achieved the highest precision (41.3%), outperforming the others by 7.0–25.0%; (2) although SGUARD reported a little less true positives (512), which was accompanied with much fewer false positives (728), which are 297–2,048 fewer than those of the other three techniques, and this feature can be very useful since all spreadsheet defects have to be manually inspected later. Besides, from the first part, we observe that SGUARD reported fewest defects (9,462), as compared to 20,280 for AMCHECK and 12,953 for CACHECK. Considering that SGUARD achieved the highest precision, its report quality is expected to be high, e.g., in 1,240 defects SGUARD detected 512 true positives, while in 3,316 (over 2.6 times) defects AMCHECK detected only 540 true positives (only marginally more). Regarding the efficiency (time cost), SGUARD took 518 minutes for analyzing all 6,258 worksheets, with an average of 5.0 seconds for each worksheet. This cost is higher than AMCHECK and CACHECK, but slightly lower than its predecessor CUSTODES by its cluster refinement.

As a summary, we recommend SGUARD for practical spreadsheet defect detection, considering its higher precision and F-measure, but with somewhat more time cost. Besides, SGUARD is flexible for integrating with other spreadsheet detect detection techniques, as aforementioned.

VI. CONCLUSION

In this paper, we present SGUARD, an effective clustering-based tool for detecting spreadsheet defects. SGUARD learns spreadsheet features to cluster cells with similar computational semantics, from which it detects faulty cells with formula defects. SGUARD aims for both precision and recall rate in spreadsheet defect detection and achieves the best F-measure among existing techniques. Currently, SGUARD has been implemented as a stand-alone automated tool with user-friendly GUI, capable of manipulating Excel spreadsheets, with one button-click execution for defect detection with colorful annotations. It is also extensible for other spreadsheet defect detection plugins. The links for its video presentation and download repository are https://youtu.be/gNPmMvQVf5Q and https://github.com/sheetguard/sguard, respectively.

ACKNOWLEDGEMENT

This work was supported by National Key R&D Program (Grant #2017YFB1001801) and National Natural Science Foundation (Grants #61932021 and #61690204) of China.

REFERENCES

[1] P. Carey and K. N. Berk. *Data Analysis with Microsoft Excel*. Brooks/Cole, 1997.
[2] S. G. Powell, K. R. Baker, and B. Lawson. A critical review of the literature on spreadsheet errors. *DSS*, 2008.
[3] R. R. Panko and S. Aurigemma. Revising the panko–halverson taxonomy of spreadsheet errors. *DSS*, 2010.
[4] B. R. Lawson, K. R. Baker, S. G. Powell, and L. Foster-Johnson. A comparison of spreadsheet users with different levels of experience. *Omega*, 2009.
[5] R. Abraham and M. Erwig. UCheck: A spreadsheet type checker for end users. *JVLC*, 2007.
[6] C. Chambers and M. Erwig. Automatic detection of dimension errors in spreadsheets. *JVLC*, 2009.
[7] W. Dou, S. C. Cheung, and J. Wei. Is spreadsheet ambiguity harmful? detecting and repairing spreadsheet smells due to ambiguous computation. In *ICSE*, 2014.
[8] W. Dou, C. Xu, S. C. Cheung, and J. Wei. CACheck: detecting and repairing cell arrays in spreadsheets. *TSE*, 2017.
[9] R. Zhang, C. Xu, S. C. Cheung, P. Yu, X. Ma, and J. Lu. How effectively can spreadsheet anomalies be detected: An empirical study. *JSS*, 2017.
[10] S. C. Cheung, W. Chen, Y. Liu, and C. Xu. CUSTODES: automatic spreadsheet cell clustering and smell detection using strong and weak features. In *ICSE*, 2016.
[11] D. W. Barowy, E. D. Berger, and B. Zorn. ExceLint: Automatically finding spreadsheet formula errors. *OOPSLA*, 2018.
[12] D. Li, H. Wang, C. Xu, F. Shi, X. Ma, and J. Lu. WARDER: Refining cell clustering for effective spreadsheet defect detection via validity properties. In *QRS*, 2019.
[13] M. Fisher and G. Rothermel. The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *SEN*, 2005.
[14] L. Xu, W. Dou, C. Gao, J. Wang, J. Wei, H. Zhong, and T. Huang. SpreadCluster: recovering versioned spreadsheets through similarity-based clustering. In *MSR*, 2017.
[15] M. Pawlik and N. Augsten. RTED: a robust algorithm for the tree edit distance. *VLDB Endowment*, 2011.
[16] W. H. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *JC*, 1984.
[17] Y. Ayalew, M. Clermont, and R. T. Mittermeir. Detecting errors in spreadsheets. *arXiv*, 2008.
[18] R. Abraham and M. Erwig. Inferring templates from spreadsheets. In *ICSE*, 2006.
[19] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *ACL*, 2006.
[20] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *ICMD*, 2000.
[21] https://poi.apache.org/. [Online; accessed 19-June-2019].
[22] http://sccpu2.cse.ust.hk/custodes/. [Online; accessed 19-June-2019].
[23] W. Dou, S. C. Cheung, C. Gao, C. Xu, L. Xu, and J. Wei. Detecting table clones and smells in spreadsheets. In *FSE*, 2016.