Understanding Spreadsheet Evolution in Practice

Liang Xu
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences
Beijing, China
xuliang12@otcaix.iscas.ac.cn

***Abstract*—As a special kind of software, spreadsheets have been evolving during their life cycle. Understanding spreadsheet evolution can help facilitate spreadsheet design, maintenance and fault detection. However, understanding spreadsheet evolution is challenging in practice. There are many factors that hinder spreadsheet evolution comprehension, such as, lack of version information, complicated structure changes during evolution, etc. Thus, we propose this work to facilitate the understanding of spreadsheet evolution, including developing semi-automated technique to build versioned spreadsheet corpora, characterizing and understanding how spreadsheet templates are reused, developing automated tools for spreadsheet comparison, and new approaches for fault detection during evolution.**

Keywords—spreadsheet; evolution; version; empirical study; template; comparison

I. INTRODUCTION

As one of the most successful end-user programming platforms, spreadsheets have been widely used in various tasks in real life, such as data storage and analysis, financial reporting and so on [1]. In the U.S. alone, it is estimated that, in 2005, the number of people programming spreadsheets is 11 million, while there are only 2.75 million other professional programmers [2].

Spreadsheets usually have long life cycle. For example, the empirical study by Hermans et al. [3] pointed out that the average lifetime of spreadsheets is about 5 years, and 13 people work on a spreadsheet on average. Thus, spreadsheets have been evolving during their life cycle. In the field of software engineering, how software evolves has been well studied and many useful findings have been applied to software maintenance e.g., change prediction [4][5] and bug detection [6], etc. Inspired by this, studies on the spreadsheet evolution can benefit spreadsheet understanding, maintenance and fault detection. For example, we can compare different spreadsheet versions to find inconsistencies, and thus improve the quality of spreadsheets.

Existing studies on spreadsheets mostly focus on improving spreadsheet quality by applying software engineering approaches and techniques, such as testing [7][8], debugging [9][10][11], and error detection [12][13][14][15]. However, few studies were performed on spreadsheet evolution despite its importance. To the best of our knowledge, the only work [16] related to spreadsheet evolution is carried out on 54 pairs of spreadsheets. In each spreadsheet pair, the first spreadsheet is the original spreadsheet developed by the customer, and the second one was rebuilt by a company F1F9 [17]. Thus, our study focuses on spreadsheet evolution, and explores new opportunities to improve the quality of spreadsheets.

There are two challenges in understanding spreadsheet evolution. The first challenge is that there is no availability of industrial-scale spreadsheet corpora with version information. Because end users of spreadsheets rarely document the version information by hand or version control tools, such as SpreadGit [18] and SharePoint [19]. As a result, the version information is missing and different versions of a spreadsheet coexist as individual and similar spreadsheets. That is the main reason why there are so few studies on the spreadsheet evolution. The second challenge is lack of the detailed information about complicated structure changes during evolution. These information is the key to understand how spreadsheets evolve. However, existing spreadsheet comparison tools (e.g., DiffEngineX [20] and Synkronizer [21]) fail to identify those complicated structure changes accurately, such as inserting a cell down or right.

To better understand spreadsheet evolution and handle the above challenges, we aim to perform the following studies in this work. (1) To solve the first challenge, we build the first and public versioned spreadsheet corpus, VEnron [22][23]. We observe that spreadsheets are usually exchanged among people by emails. Thus, we extract spreadsheets attached in the emails from the Enron Email Archive [24], and use various information (e.g., the email subjects and contents) associated with the spreadsheets to recover version information among spreadsheets. (2) In order to understand how spreadsheets are created and modified, we carry out an empirical study on VEnron to characterize and understand how spreadsheet templates are used. We further identify the complicated structure changes during spreadsheet evolution. (3) To solve the second challenge, we plan to propose a greedy algorithm to compare two spreadsheets accurately, which can align two worksheets to minimize the number of changes in the cell level. We also plan to apply our comparison algorithm and version information to fault detection among evolution to improve the quality of spreadsheets.

The remainder of this paper is organized as follows. Section II describes our research questions. Section III shows our methodology and preliminary results. We discuss related work in Section IV. Finally, we summarize this paper in Section V.

II. RESEARCH QUESTIONS

As mentioned earlier in Section I, the unavailability of industrial-scale spreadsheet corpora with version information is the key obstacle to study spreadsheet evolution scientifically.

978-1-5386-0992-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSME.2017.38

Since the version information is missing and different versions of a spreadsheet coexist as individual spreadsheets. Thus, to create a new versioned spreadsheet corpus, we need to identify whether the spreadsheets are different versions of a spreadsheet, and further recover the version information between spreadsheets in the same evolution group. Here, in each evolution group, all spreadsheets are different versions of the same spreadsheet and sorted a straight-line historical orders. So our first research question (***RQ1***) is that: ***How can we recover version information from a set of spreadsheets?***

During spreadsheet development, users usually create a new spreadsheet by copying and modifying an existing one to process new data. The new created spreadsheet can be considered as an update version of the old one. Thus, the previous spreadsheet will be considered as the template of the later one. As the special spreadsheets or worksheets, those templates predefine the data layout and computational logic, and thus can save users' effort. To study spreadsheet evolution, we focus on the following question about spreadsheet evolution: how do users adopt spreadsheet templates to create new spreadsheets? However, it is challenging to know which spreadsheets can be templates and which ones can the instances of spreadsheets. Thus, we narrow down our research to understand the explicitly defined spreadsheet templates in VEnron [22]. We only focus on the spreadsheets marked by keyword "template", because this keyword indicates these spreadsheets can be used as templates to create new spreadsheets. So our second research question (***RQ2***) is that: ***How are spreadsheet templates used in the real world?***

The detailed information about complicated structure changes during evolution is missing and cannot be recovered accurately by existing comparison tools. This hinders the spreadsheet evolution related research, because recovering them manually is time-consuming. So our third research question (***RQ3***) is that: ***How can we compare two spreadsheets accurately?***

We observe that some errors have been introduced during spreadsheet evolution, and thus reducing the quality of spreadsheets. We expect to apply our comparison algorithm and version information to fault detection. So our fourth research question (***RQ4***) is that: ***How can we detect the introduced faults during spreadsheet evolution?***

We have accomplished RQ1 [22][23] and made some progress towards solving RQ2. We plan to address RQ3 and RQ4 in the near future. We give more detailed information about our methodology and preliminary results in next section.

III. Methodology and Preliminary Result

To answer RQ1, we first build the first versioned spreadsheet corpus based on the Enron Email Archive [24]. We first propose a filename-based spreadsheet clustering approach [22], and then propose a content-based spreadsheet clustering approach [23]. To answer RQ2, we filter out the templates with keywords and identify instances for each detected template. Then we compare the templates and corresponding instance to inspect the changes made during creation of instances. To answer RQ3, we plan to design a novel greedy algorithm to identify high-level changes that can be used to align two spreadsheets. To answer RQ4, we plan to combine our comparison algorithm with version information to detect inconsistent faults introduced during spreadsheet evolution.

A. RQ1: How can we recover version information from a set of spreadsheets?

We use the Enron Email Archive [24] as our research subject since it is public and contains lots of spreadsheets. The Enron Email Archive [24] contains 752,605 emails. These emails attach 16,189 unique spreadsheets according to MD5 file hashes. We manually inspect those spreadsheets and observe that users usually email their latest spreadsheets to others. Intuitively, the spreadsheet sent latter can be considered as an updated version of that was sent earlier. Thus, we can recover the relationship between spreadsheets according to the information hidden in the emails. Further, we observe that, without version control tools, users usually add the number or date string into the filenames (e.g., *May00_FOM_Req2.xls* and *Jun00_FOM_Req.xls*) to distinguish different versions of the same spreadsheet. This makes it possible to recover the version information based on the similarity of spreadsheet filenames.

Based on this observation, we propose the filename-based approach to recover the version information between spreadsheets. It works in four steps:

1) We extract the spreadsheets from Enron Email Archive, and keep the original filenames.
2) We calculate the shortened filename for each spreadsheet by removing version-related information, such as, numbers, date and special characters (e.g., "#", "-", and "*", etc.). Then we cluster spreadsheets into different groups according to whether they share the same shortened filenames. The spreadsheets in the same groups may be different versions of the same spreadsheet. For example, the spreadsheets *May00_FOM_Req2.xls* and *Jun00_FOM_Req.xls* share the same shortened name "*FOMReq*", they are clustered into the same group.
3) We manually validate whether the spreadsheets in the same group are different versions of a spreadsheet by checking whether they share similar worksheet names, table structures (including table titles, row / column labels and cell formulas) and email contents.
4) We recover the version order of the spreadsheets in each evolution group according to the version information extracted from filenames, worksheet names, spreadsheet contents and related emails (e.g., the sending time).

Result: we build the first versioned spreadsheet corpus, VEnron [22], which contains 360 evolution groups and 7,294 spreadsheets.

However, the applicability and accuracy of the filename-based approach is limited, because it relies on the assumption that all spreadsheets are well-named, which is not always true. This motivates us to propose a content-based algorithm, called SpreadCluster [23], to identify different versions of the same spreadsheet. Figure 1 gives the overview of SpreadCluster, we can see that it contains two phases: a training phase and a working phase. In the training phase, SpreadCluster extracts

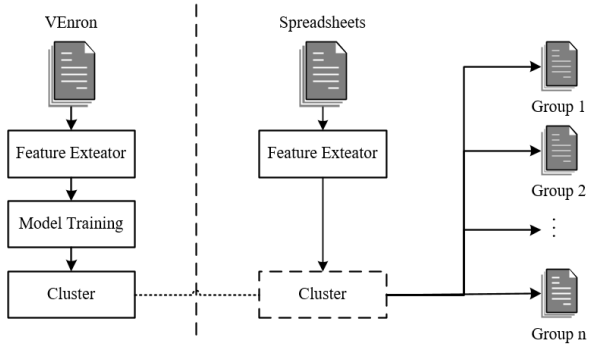

Figure 1. The overview of SpreadCluster, adopted from [23].

features (e.g., table headers and worksheet names) from each spreadsheet. Then, SpreadCluster calculates the similarity between spreadsheets based on the extracted features. Finally, SpreadCluster trains a clustering model using the training dataset created based on VEnron [22]. In the working phase, SpreadCluster extracts the same features from spreadsheets and calculates the similarity between them. Then, SpreadCluster uses the trained model to cluster spreadsheets into different evolution groups.

Result: we compare SpreadCluster with the filename-based approach on the spreadsheets extracted from the Enron email archive [24]. The experimental result indicates that SpreadCluster can identify evolution groups with high precision (78.5%) and recall (70.7%), while the filename-based approach [23] only achieves 59.8% precision and 48.7% recall.

To validate the applicability of SpreadCluster, we apply SpreadCluster on the other two spreadsheet corpora EUSES [25] and FUSE [26]. EUSES is the most frequently used spreadsheet corpus, and contains 4,037 spreadsheets extracted from World Wide Web. FUSE is a reproducible, internet-scale corpus, and contains 249,376 unique spreadsheets that were extracted from over 26.83 billion webpages. The spreadsheets in both two corpora are used in different domains. The experimental result shows SpreadCluster performs well on both two corpora with high precision (91.0% and 79.8%, respectively).

B. RQ2: How are the spreadsheet templates used in the real world?

We carry out this empirical study on the usage of templates in three steps:

1) We filter out the templates from VEnron [22] by checking whether filenames or worksheet names contain the keyword "template". Note that, not all spreadsheets and worksheets whose name contains the keyword "template" are templates, because users may keep the word "template" in the names of instances. We manually validate whether each detected spreadsheet or worksheet is a template or an instance.

2) We identify the instances for each validated template. For each worksheet template, we try to find its instances in the same spreadsheets. For each spreadsheet template, we try to find its instance in the same evolution group.

	Type-1	Type-2	Type-3	Type-4	Total
Worksheet Template	0	1	10	58	70
Spreadsheet Template	9	5	8	8	30

Table 1. The distribution of different types of templates.

3) We manually compare the templates with the corresponding instances in detail. We predefine twelve most frequently used types of high-level changes in practice (e.g., insert or delete rows / columns, move rows down or up, move columns left or right, insert a cell down or right, delete a cell up or left) and seven types of cell-level changes (header addition / refinement / modification / deletion, formula addition / deletion / modification) to measure the differences between the templates and corresponding instances.

Note that the accuracy of detection of changes between templates and instances in the step 3 will directly affect the correctness of the results of our study. However, we cannot find an available comparison tool that can identify the high-level changes accurately. Thus we have to align two spreadsheets manually and record the used high-level changes. After two worksheets are aligned, we get the changes in the cell-level by comparing them cell by cell.

Preliminary result: Table 1 shows our result, we in total identify 100 templates, including 70 worksheet templates (Worksheet Template) with 452 instances, and 30 spreadsheet templates (Spreadsheet Template) with 220 instances, involving 1,508 worksheets.

According to the content and purpose of these templates, we classify them into four categories:

- **Type-1: Fixed Data Layout.** The templates of this type only define the data layout and contain no formulas. When creating a new instance, users are forbidden to insert or delete rows (columns).
- **Type-2: Variable Data Layout.** The templates of this type only define the data layout and contain no formulas. But users are allowed to insert or delete rows (columns).
- **Type-3: Fixed Data Layout with Logic.** The templates of this type not only define the data layout and contain some computational logic. But users are forbidden to insert or delete rows (columns).
- **Type-4: Variable Data Layout with Logic.** The templates of this type not only define the data layout and contain some computational logic. But users are allowed to insert or delete rows (columns).

From Table 1, we can see that most worksheet templates (58/70) belong to the Type-4. That indicates the worksheet templates are usually to create complex worksheets.

We are ongoing comparing the templates with instances to collect the changes made during reusing. We want to answer more interesting questions, as follow:

1) What are the characteristics of spreadsheet templates? E.g., do the templates differ from instances with respect to the widely used metrics, such as size, level of coupling, and the use of functions?

2) What types of changes are made during creation of new worksheets or spreadsheets by templates?

3) What and how smells are introduced during creation of new worksheets or spreadsheets based on templates?

4) Are the templates well-designed? E.g., are new headers or formulas introduced into templates when creating new instances?

Furthermore, we observe that spreadsheet or worksheet templates are usually missing or not explicitly marked. Recovering the templates for a set of instances is helpful for users to understand and maintain their spreadsheets. We also plan to design a template recovery algorithm to recover the missing templates from a set of spreadsheet instances. We will evaluate this recovery algorithm by comparing its result with the real templates in our empirical study.

C. RQ3: How can we compare two spreadsheets accurately?

We plan to design an algorithm to accurately compare two spreadsheets. For a pair of worksheets, we try to mutate a worksheet based on a set of predefined high-level operations, and further align two worksheets and compare them cell by cell. We expect to find a set of high-level operations to minimize the differences between two aligned worksheets.

We in total predefine twelve most frequently used high-level operations in practice, which can change the data layout of a worksheet:

- **Insert or delete rows (columns)**: users may insert or delete one or more rows (columns), especially in usage of template Type-2 and Type-4.
- **Move rows down or up and move columns right or left:** users may reorder some rows or columns. Especially, users may swap two rows or columns.
- **Insert cells down or right**: when users insert one or more cells, the cells located below the insertion point (including the cells on the insertion point) are moved down, or the cells located on the right of the insertion point (including the cells on the insertion point) are moved right.
- **Delete cells up or left**: when users delete one or more cells, the cells below the deleted cells in the same columns are moved up or the cells on the right of the deleted cells in the same rows are moved left.

After applying a set of the above high-level operations, we try to find an optimal solution which can align two worksheets, and further compare worksheets and their related spreadsheets. However, enumerating all possible worksheet mutation solutions is very time-consuming. Thus, we plan to find out some heuristic rules to reduce the number of potential worksheet mutation solutions and find an optimal solution quickly.

D. RQ4: How can we detect the introduced faults during spreadsheet evolution?

During spreadsheet evolution, various changes may be introduced into spreadsheets. Improper changes may introduce faults in spreadsheets. Usually, the inconsistencies among different versions of a spreadsheet may indicate faults. However, not all inconsistences indicated faults. For example, a cell with a formula SUM(A1:A10) can be changed into SUM(A1:A11), when a new row is inserted before row 5. Although the cell's formula changes, it is correct. Thus, we plan to figure out a set of rules about which inconsistencies are wrong or correct. We further plan to figure out possible fixing suggestions from multiple other versions.

IV. Related Work

Spreadsheet corpora: The most widely used spreadsheet corpus is EUSES [25], containing 4,037 spreadsheets extracted from Internet. Enron [27], the first industrial spreadsheet corpus, contains more than 15,000 spreadsheet extracted from the Enron email archive [24]. FUSE [26] is the biggest spreadsheet corpus, containing 249,376 spreadsheets extracted from over 26 billion pages [28]. None of three spreadsheet corpora contains version information. In other words, their spreadsheets are independent and the relationships between them are missing. VEnron [22] is the first versioned spreadsheet corpus we built manually, containing 360 evolution groups and 7,294 spreadsheets. We proposed a content-based algorithm to automate it and built a much greater spreadsheet corpus than VEnron.

Spreadsheet evolution: Since the spreadsheet version information is missing, few work focus on spreadsheet evolution. The only one work we can find is carried out an evolution study on 54 pairs of spreadsheets [16]. However, the used corpus is not publicly available. The versioned spreadsheet corpus we built is publicly available. Based on this corpus, we carry out an empirical study on the usage of templates. We believe this corpus can be used to do more interesting research work.

Spreadsheet comparison: Some spreadsheet comparison techniques and tools have been proposed. SheetDiff [29] first detects the cell changes, and then optimizes cell changes into higher-level changes. DiffEngineX [20] and Synkronizer [21] are two commercial tools that can be used to compare and highlight the differences between two spreadsheets. However, the precision of those existing tools and techniques is not satisfying, especially in detecting high-level changes. In this proposal, we plan to propose a greedy algorithm to identify the high-level changes, then identify the changes in cell level by cell-by-cell comparison.

V. Conclusion

This paper is motivated by understanding the spreadsheet evolution in practice. We firstly propose two approaches to build the first versioned spreadsheet corpus, VEnron [22][23]. Based on this built corpus, we carry out an empirical study on the usage of templates to investigate how spreadsheets evolve. To make it easy to carry out further spreadsheet evolution related research, we plan to propose an algorithm to compare two spreadsheets accurately. We also plan to develop approaches to detect faults during evolution, and thus improve the quality of spreadsheets.

Acknowledgment

This work was supported in part by National Key Research and Development Plan (2016YFB1000803), Beijing Natural Science Foundation (4164104), and National Natural Science Foundation of China (61672506).

REFERENCES

[1] L. a. Kappelman, J. P. Thompson, and E. R. McLean, "Converging end-user and corporate computing," *Commun. ACM*, vol. 36, pp. 79–92, 1993.

[2] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2005, pp. 207–214.

[3] F. Hermans, M. Pinzger, and A. van Deursen, "Supporting professional spreadsheet users by generating leveled dataflow diagrams," in *Proceeding of the 33rd international conference on Software engineering (ICSE)*, 2011, pp. 451–460.

[4] S. Kim, Z. Thomas, E. J. Whitehead Jr, and A. Zeller, "Predicting faults from cached history," in *Proceedings of the 29th international conference on Software Engineering (ICSE)*, 2007, pp. 489–498.

[5] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *Trans. Softw. Eng.*, vol. 38, pp. 1276–1304, 2012.

[6] D. Kim, J. Nam, J. Song, and S. Kim, "Automatic patch generation learned from human-written patches," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2113, pp. 802–811.

[7] R. Abraham and M. Erwig, "AutoTest: a tool for automatic test case generation in spreadsheets," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2006, pp. 43–50.

[8] G. Rothermel, L. Li, C. DuPuis, and M. Burnett, "What you see is what you test: a methodology for testing form-based visual programs," in *Proceedings of the 20th International Conference on Software Engineering (ICSE)*, 1998, pp. 198–207.

[9] R. Abraham and M. Erwig, "GoalDebug: a spreadsheet debugger for end users," in *Proceedings of the 29th International Conference on Software Engineering (ICSE)*, 2007, pp. 251–260.

[10] J. Reichwein, G. Rothermel, and M. Burnett, "Slicing spreadsheets: an integrated methodology for spreadsheet testing and debugging," *ACM SIGPLAN Not.*, vol. 35, pp. 25–38, 1999.

[11] D. W. Barowy, D. Gochev, and E. D. Berger, "CheckCell: data debugging for spreadsheets," in *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages & Applications (OOPSLA)*, 2014, pp. 507–523.

[12] W. Dou, C. Xu, S. C. Cheung, and J. Wei, "CACheck: detecting and repairing cell arrays in spreadsheets," *Trans. Softw. Eng.*, vol. 43, pp. 226–251, 2017.

[13] W. Dou, S.-C. Cheung, and 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 (ICSE)*, 2014, pp. 848–858.

[14] W. Dou, S.-C. Cheung, C. Gao, C. Xu, L. Xu, and 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, pp. 787–798.

[15] S.-C. Cheung, W. Chen, Y. Liu, and 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 (ICSE)*, 2016, pp. 464–475.

[16] B. Jansen and F. Hermans, "Code smells in spreadsheet formulas revisited on an industrial dataset," in *Proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 372–380.

[17] "F1F9." [Online]. Available: www.f1f9.com/.

[18] "SpreadGit." [Online]. Available: https://www.crunchbase.com/organization/spreadgit.

[19] "SharePoint." [Online]. Available: https://products.office.com/zh-cn/sharepoint/collaboration.

[20] "Florencesoft DiffEngineX - compare Excel workbooks xlsx." [Online]. Available: https://www.florencesoft.com/.

[21] "Synkronizer Excel compare: compare, update and merge Excel files." [Online]. Available: http://www.synkronizer.com/.

[22] W. Dou, L. Xu, S.-C. Cheung, C. Gao, J. Wei, and T. Huang, "VEnron: a versioned spreadsheet corpus and related evolution analysis," in *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE)*, 2016, pp. 162–171.

[23] L. Xu, *et al.*, "SpreadCluster: recovering versioned spreadsheets through similarity-based clustering," in *Proceedings of the 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 158–169.

[24] "The Enron PST data set cleansed of PII by Nuix and EDRM." [Online]. Available: http://info.nuix.com/Enron.html.

[25] M. Fisher and G. Rothermel, "The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms," *ACM SIGSOFT Softw. Eng. Notes*, pp. 1–5, 2005.

[26] T. Barik, K. Lubick, J. Smith, J. Slankas, and E. Murphy-Hill, "FUSE: a reproducible, extendable, Internet-scale corpus of spreadsheets," in *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR)*, 2015, pp. 486–489.

[27] F. Hermans and E. Murphy-Hill, "Enron's spreadsheets and related Emails: a dataset and analysis," in *Proceedings of the 37th IEEE International Conference on Software Engineering (ICSE)*, 2015, pp. 7–16.

[28] "Common crawl data on AWS." [Online]. Available: http://aws.amazon.com/datasets/41740.

[29] C. Chambers, M. Erwig, and M. Luckey, "SheetDiff: a tool for identifying changes in spreadsheets," in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2010, pp. 85–92.