Finding Errors in the Enron Spreadsheet Corpus

Thomas Schmitz
TU Dortmund
44221 Dortmund, Germany
thomas.schmitz@udo.edu

Dietmar Jannach
TU Dortmund
44221 Dortmund, Germany
dietmar.jannach@udo.edu

***Abstract*—Spreadsheet environments like MS Excel are the most widespread type of end-user software development tools and spreadsheet-based applications can be found almost everywhere in organizations. Since spreadsheets are prone to error, several approaches were proposed in the research literature to help users locate formula errors. However, the proposed methods were often designed based on assumptions about the nature of errors and were evaluated with mutations of correct spreadsheets.**

In this work we propose a method and tool to identify real-world formula errors within the Enron spreadsheet corpus. Our approach is based on heuristics that help us identify versions of the same spreadsheet and our software helps the user identify spreadsheets of which we assume that they contain error corrections. An initial manual inspection of a subset of such candidates led to the identification of more than two dozen formula errors. We publicly share the new collection of real-world spreadsheet errors.

I. Introduction

Spreadsheets are used almost everywhere and at all levels of organizations [1]. They are often used for financial calculations and planning purposes so that errors in the calculations can have severe impacts for organizations [2]. Errors in spreadsheets are unfortunately not uncommon, in particular because spreadsheets are often developed by end-users with no education in software development. Already in the late 1990s a survey showed that in many studies on spreadsheet errors at least one fault[1] was found in every analyzed spreadsheet [4].

Different approaches to avoid spreadsheet errors are possible, starting with better training for end-users or defined quality procedures for spreadsheets. Over the years, also a variety of proposals for better *tool support* were made in the literature [3], ranging from visualization approaches [5], over environments that support systematic tests [6], to interactive debugging aids [7]. Many of the proposed error detection and correction tools focus on errors in individual *formulas*.

A common challenge when designing and evaluating such approaches is that not many real-world spreadsheets with known formula errors are available. Although larger collections of real-world spreadsheets exist, usually no information about the contained errors is given [8], [9]. To evaluate novel test and debugging techniques researchers therefore often inject errors into real-world or artificial spreadsheets using, e.g., the set of mutation operators for spreadsheets proposed in [10]. Such mutations can represent a useful approximation of the true errors that are made by users. Nonetheless, these mutation-based evaluations are based on certain assumptions about the types and frequency of different types of errors.

In 2015, Hermans and Murphy-Hill [11] published a new corpus of spreadsheets extracted from the publicly available emails of Enron, a huge US-company that went bankrupt in 2001 ("Enron scandal"). The new corpus comprises 15,770 spreadsheets that were created for productive use and of which 9,120 contain formulas. Again, however, no information is available about the errors that these spreadsheets contain.

In this paper, we therefore propose a method and publish a tool [12] to locate formula errors in spreadsheets of the Enron corpus. To find such errors, we first try to identify different *versions* of the same spreadsheet in the corpus, where one version contains a fix to a bug that existed in the previous version. We use different heuristics to detect such spreadsheet versions. In one strategy we reconstruct parts of the email conversations in which spreadsheets were exchanged and look for indicators in the email texts which suggest that the enclosed spreadsheet contains a bug fix. All spreadsheets that are attached in this conversation are then automatically checked for differences. In another approach we look for spreadsheets whose names are similar or slightly different and, e.g., contain a suffix like "_v2" or "_fixed". We then again compute the differences between these files. If only one or a few formulas were changed, these files represent *candidate* spreadsheets, which can then be manually inspected for errors.

Determining if a change of a formula represents a bug fix or rather implements an updated business logic is hard to automate as one has to understand the intended semantics of each formula. We therefore implemented a visual tool that automatically retrieves the different versions of a spreadsheet and supports the user in inspecting them. With the help of this tool we identified several spreadsheet errors of different types using only a limited set of heuristics. We publicly share our collection of errors to foster future research in the field [13].

II. Technical Approach

In this section we present how we reconstruct the email conversations and how we analyze differences in spreadsheets.

A. Reconstruction of Email Conversations

To identify emails that discuss errors in the attached spreadsheets, we propose to reconstruct the email conversations.

[1]In this paper we use the terms error and fault in an interchangeable manner. A discussion of the usage of the different terms can be found in [3].

978-1-5090-0252-8/16/$31.00 ©2016 IEEE

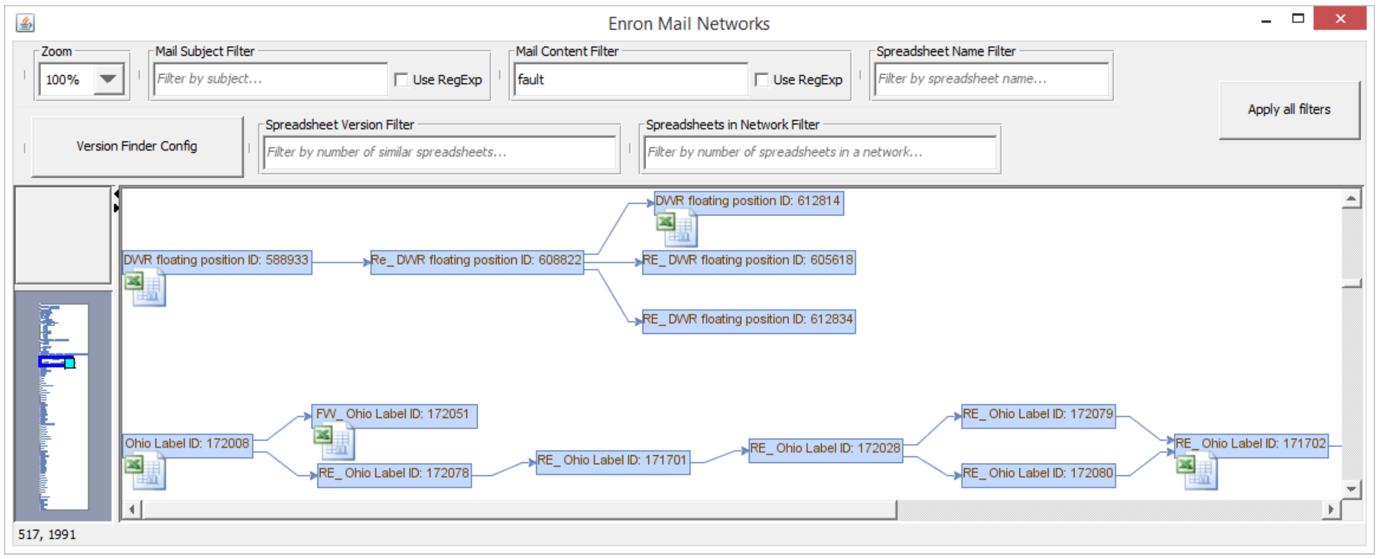

Enron Mail Networks
Zoom
100%
Mail Subject Filter
Filter by subject...
Use RegExp
Mail Content Filter
fault
Use RegExp
Spreadsheet Name Filter
Filter by spreadsheet name...
Apply all filters
Version Finder Config
Spreadsheet Version Filter
Filter by number of similar spreadsheets...
Spreadsheets in Network Filter
Filter by number of spreadsheets in a network...
DWR floating position ID: 612814
DWR floating position ID: 588933
Re_ DWR floating position ID: 608822
RE_ DWR floating position ID: 605618
RE_ DWR floating position ID: 612834
FW_ Ohio Label ID: 172051
Ohio Label ID: 172008
RE_ Ohio Label ID: 172078
RE_ Ohio Label ID: 171701
RE_ Ohio Label ID: 172028
RE_ Ohio Label ID: 172079
RE_ Ohio Label ID: 172080
RE_ Ohio Label ID: 171702
517, 1991

Fig. 1. A screenshot of our interactive tool for finding errors in the Enron corpus.

1) General Idea: Figure 1 shows the interactive visualization of such a conversation in our tool. Nodes in the graph correspond to emails and the edges represent that, for example, one email was sent in reply to another.

Our tool reconstructs such conversations using different heuristics. With the implemented heuristics we created 13,440 conversation graphs that had at least one spreadsheet attached. 1,100 of them consisted of two or more nodes. In our tool, individual keywords like “fix” or “error” as well as complex regular expressions can be used to filter those conversations that contain these keywords in the subject line, email text, or as part of a spreadsheet name.

The conversation and the attached spreadsheets can then be manually inspected one by one. To support the user in this manual process, the tool automatically determines and displays the exact differences between each spreadsheet of the conversation. If the number of differences between two files is very small and, e.g., only one single formula was changed, this might be an indicator of a possible bug fix.

Our approach of searching for certain terms in email conversations is inspired by [11], who found over 4,000 emails in the Enron corpus which had a spreadsheet attached and contained one of several keywords like *error* or *mistake*. Retrieving emails with certain keywords is however not sufficient for our purpose, as our goal is to find different versions of one spreadsheet to be able to identify possible errors.

2) Reconstruction Heuristics: Reconstructing the email conversations is not a straightforward process with the given data. The emails of the corpus unfortunately do not contain the two header fields called *references* and *in-reply-to* of the *Internet Message Standard*, which should contain unique message identifiers of previous messages.

Therefore, we used the email header information about the subject, sender, recipients and the timestamp of the message, as well as the message text itself to approximately reconstruct the conversations. Specifically, we inserted a link in a conversation graph – indicating that a message a is followed by a message b – whenever the following conditions were fulfilled.

(i) One of the recipients of a is the sender of b, i.e., the sender of b replied to a or forwarded a.
(ii) The subject lines of message a and b match (after removing prefixes like “Re:”) **or** the message text of b contains the entire text of a.
(iii) The timestamp of b is later than the one of a **and** there is no other email c with a timestamp that lies between a and b and for which conditions 1 and 2 are fulfilled.

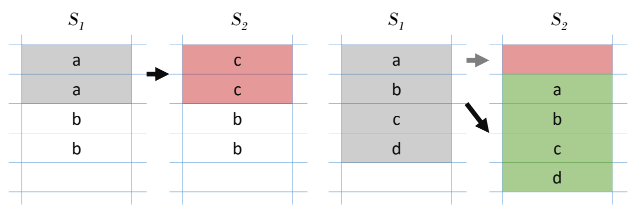
Checking these conditions again requires some heuristics-based approximations due to the noisiness of the data. The sender and recipient names, for example, are often set by the email client based on an integrated address book and do not contain email addresses but real names with no consistent ordering of first and last names. Therefore, we implemented a name matching technique that tries different orderings and uses the Jaro-Winkler distance to assess the similarity of different entries. We assumed the names to be identical if a certain threshold was surpassed.

B. Analyzing the Differences in Spreadsheets

Once we have determined a subset of spreadsheets that are presumably related, e.g., because they are in the same conversation graph or because they have similar names, our tool supports the user with an automated analysis of the differences between the files.

1) Detecting Modifications: Our analysis of differences focuses on changes in formulas. Changes only in number and text constants between two versions are not considered. We consider formula updates, insertions and deletions as changes between spreadsheet versions.

As mentioned above, spreadsheet versions that only have a limited number of differences are particularly relevant for us as



(a) Formulas were changed. (b) Formulas were moved.

Fig. 2. Analyzing differences of a spreadsheet.

it makes it easier to understand the modifications. A commonly used functionality in spreadsheet systems is to copy formulas to apply the same calculations on different rows or columns. In the Enron corpus a spreadsheet with formulas on average contains 2,223 formulas of which only 100 are unique [11]. If a bug fix concerns such a copied formula, we would therefore detect multiple formula changes.

In our calculation scheme for differences we account for such situations where so-called "copy-equivalent" formulas are changed. We achieve this through the use of the R1C1 notation in which copy-equivalent formulas have the same cell content. Figure 2a shows an example where in two copy-equivalent cells the formula was changed from a to c. According to our heuristic, this would only count as one difference.

2) Detecting Moved Cells: Another situation in which a naive approach to spot differences would lead to too many suspected changes is when new rows or columns are inserted as part of a change. Figure 2b shows such a situation where an empty row was inserted. The goal of the subsequently described heuristic is to detect when (blocks of) cells are moved. In the example in Figure 2b, our method should therefore report "no change" instead of a formula deletion in the topmost cell and a formula addition at the bottom.

To detect such movements we use heuristics regarding the surrounding of the changed cells. If we find the formula of the changed cell and an identical surrounding area of a specified size at a different location in the changed spreadsheet, we assume that the whole area was moved to this location.

Algorithm 1 sketches the idea of our corresponding spreadsheet difference analysis. The algorithm takes two spreadsheets S_1 and S_2 to be compared as input and maintains a list called *diffs* in which the found differences are stored. The main function examines all cells which contain a formula in at least one of the spreadsheets. For these cells, the function ISDIFFERENT is called, which checks if the content of the cell differs in the two spreadsheets. Internally, this method also checks if the same difference was already observed before for a copy-equivalent cell as we only want to count each difference once. In case a difference was found, i.e., one of the cells contains no formula or the formulas differ, the function WASMOVED is called, which returns true if we assume that a formula and its surroundings were moved. If the observed difference is not the result of a move, the cell c is stored as a difference in the set *diffs*.

The function WASMOVED checks if the formula in the given

Algorithm 1: FINDDIFFERENCES

```
Input: Two spreadsheets $S_1$, $S_2$; A minimum area size
       $minSize$ to recognize moved areas
Output: A set of cell positions $diffs$ for which
        differences were found between $S_1$ and $S_2$
1 foreach $c \in$ FORMULACELLS($S_1$)
      $\cup$ FORMULACELLS($S_2$) do
2     if ISDIFFERENT($c, S_1, S_2, diffs$) $\wedge$
          $\neg$WASMOVED($c, S_1, S_2, minSize$) then
3         $diffs \leftarrow diffs \cup \{c\}$;
4 return $diffs$;
  function WASMOVED($c, S_1, S_2, minSize$)
5     $candidates \leftarrow$ FINDSAMEFORMULAS($c, S_1, S_2$);
6     foreach $candidate \in candidates$ do
7         $areas \leftarrow areas \cup \{$FINDEQUIVALENTAREA($c$,
              $S_1, candidate, S_2$)$\}$;
8     return $minSize <$ MAXSIZE($areas$);
```

cell with the same surrounding area can be found elsewhere in the spreadsheet. The function first searches for all cells in S_2 that have the same formula as cell c in S_1. Then it iterates over all elements of this list called *candidates* and calculates the size of the area in S_2 that is equal to the area surrounding c in S_1. If a sufficiently large identical block – as specified by the *minSize* parameter – is found for at least one of the *candidates*, the algorithm assumes that the corresponding area was moved.

More complex heuristics or even exact pattern matching methods could of course be used but can come at the cost of higher computational complexity. We chose a simple heuristic as our goal is to support the parameterizable "on-demand" calculation of differences, e.g., in the context of email conversation graphs.

III. VALIDATION – DETECTING ERRORS IN THE CORPUS

To validate our general approach and the designed heuristics, we used the developed software tool to locate an initial set of real-world errors in the Enron corpus.

Our method supports two modes of operation to find spreadsheet versions: (a) based on the inspection of email conversations, (b) based on the similarity of file names.

A. Classifying Changes as Error Corrections

Determining whether a change from one spreadsheet version to another led to the correction or introduction of an error can in most cases only be done through a manual process[2]. Each identified error that we report here was therefore classified as such by at least two independent spreadsheet experts in a manual process. We adopted a conservative strategy and classified changes only as errors if the intended semantics of

[2]In our view, only very simple cases like the removal of a #DIV/0 error can probably be automatically detected with some confidence.

the calculations in the spreadsheet were understandable and the bug was obvious or even mentioned in the email text.

1) Example 1: We searched for email conversations that contained the words "error" and "spreadsheet" in the message text.[3] One filtered email contained the text "*Ron pointed out an error to me in my spreadsheet. The revised one is attached*". The sender pointed out that one calculation outcome was "*too low*". An automated comparison of the attached spreadsheet with other versions of it quickly led us to the change. In cell D6, the formula "=D4*1500" was changed by the sender of the email to "=D10*1500", i.e., a cell reference error was made in the original file, which led to the faulty (too low) outcome.

In that particular case the file names of the different versions of the spreadsheet attached to the emails were identical. This file and its different versions would therefore also be found by our tool when we only look for file versions without reconstructing the email conversations. The text of the email message however assures us that the change was actually an error and not a change of the business rules.

2) Example 2: When searching for files with similar names, our tool returned two versions of a multi-worksheet spreadsheet named CrackSpreadOptions.xls. The files contained six formula differences, which were however detected as changes to copy-equivalent formulas and counted as one. Specifically, the formulas in column M were changed from "=HEAT($B9;...;M$7)" to "=C9*HEAT($B9;...;M$7)" etc., i.e., the computation was extended with a multiplication factor that was forgotten in the previous version.[4] We were confident that this was truly a hard-to-detect omission error [14] because the updated spreadsheet also contained the comment "*Had to scale column M by the gas price!!!*".

B. An Initial Corpus of Errors in the Enron Corpus

So far, we have only conducted a few first sessions to build a corpus of spreadsheet errors with the help of our tool. We have inspected a few dozen of the email conversations with the above mentioned keywords manually to locate obvious errors as those reported above. Furthermore, we made a search based on identical filenames and limited the search to files which differed from each other in at most three formulas. From the returned spreadsheets we inspected about 200 files manually.

Overall, already through our initial search we could identify 28 occurrences which we classified as quantitative errors with high confidence. According to the classification of [15], we found 14 *mechanical* errors, 9 *logical* errors, and 5 *omission* errors. In addition to these errors, we found 8 *qualitative errors* [15], i.e., errors which do not directly lead to immediate failures but degrade the quality of the spreadsheet. Such qualitative errors for example include wrong labels for formulas. We are continuing to extend the corpus and provide all details on a public web site [13]. Our results so far confirm that all error types mentioned in the literature actually appear in real-world spreadsheets.

[3]The search with the two terms returned quite a number of irrelevant conversations as the word "error" was often part of email disclaimers.

[4]The function HEAT is part of an external library.

In the current corpus the majority of the problems was identified based on matching file names as this was the first technique that we explored. More than half of the errors could however have been found using either of our identification techniques (name-based or conversation-based). Specifically, for 19 of the 36 errors the email conversations included information about a corrected error or even its exact location.

IV. Related Work

Besides the Enron document corpus [11] used in this work, other collections of spreadsheets were published over the years to support error research for spreadsheets. Both the often-used EUSES corpus [8] (4,498 documents) and the more recent FUSE corpus [9] (249,376 documents) contain spreadsheets that were retrieved with the help of search engines. Many of the documents, however, contain no formulas at all. Furthermore, no additional information is available about potential errors in the spreadsheets or if they were in practical use.

Other spreadsheet collections were designed to include information about errors. The Hawaii Kooker Corpus for example comprises 75 spreadsheets (with 97 faults) that were created by undergraduate students [16]. A comparable corpus of spreadsheet documents created by students was presented in [17]. While these corpora obviously contain real errors made by humans, it is not fully clear if the spreadsheets and example calculations are representative for spreadsheets that are found in industry. Furthermore, spreadsheets that are created in exercises can be structurally quite diverse, hard to comprehend, or incomplete. Comparing a submitted solution with a reference solution can therefore be tedious.

Using email conversations as an additional source to detect errors in real-world spreadsheets has to our knowledge not been done before. Some works, however, exist that aim at automatically detecting differences in spreadsheets. *SheetDiff* [18], for example, uses a greedy technique to search for several types of differences which are then visually presented to the user. Later on, an approach called *RowColAlign* was proposed that uses a dynamic programming technique to address some shortcomings of *SheetDiff* [19]. In the current version of our tool the differences between spreadsheets are presented in a structured and compact text-based form. We see the integration of the ideas proposed in [18] or [19] to visualize the differences as a promising direction for our future work.

V. Conclusion

Research on error detection techniques for spreadsheets requires a solid understanding of the types of errors that users make when creating spreadsheets. In this work we have presented a method and tool to locate errors in the Enron spreadsheet corpus based on the identification of versions of the same spreadsheet. One particular novelty of our approach lies in the utilization of information from the email conversations in the company. Through a first manual inspection of a number of version candidates with our tool, we could develop an initial set of real-world spreadsheet errors which we plan to continuously extend in the future.

Acknowledgment

The work was funded by the Austrian Science Fund (FWF, contract I2144) and the German Research Foundation (DFG, contract JA 2095/4-1). Thanks to Tom-Philipp Seifert for implementation works.

References

[1] R. R. Panko and D. N. Port, "End User Computing: The Dark Matter (and Dark Energy) of Corporate IT," in *Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS 2012)*, Wailea, HI, USA, 2012, pp. 4603–4612.

[2] EuSpRIG, "Spreadsheet horror stories," Published online at http://www.eusprig.org/horror-stories.htm, Last accessed 2016.

[3] D. Jannach, T. Schmitz, B. Hofer, and F. Wotawa, "Avoiding, finding and fixing spreadsheet errors - a survey of automated approaches for spreadsheet QA," *Journal of Systems and Software*, vol. 94, pp. 129–150, 2014.

[4] R. R. Panko, "What We Know About Spreadsheet Errors," *Journal of End User Computing*, vol. 10, no. 2, pp. 15–21, 1998.

[5] F. Hermans, M. Pinzger, and A. van Deursen, "Supporting Professional Spreadsheet Users by Generating Leveled Dataflow Diagrams," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*, 2011, pp. 451–460.

[6] 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)*, 2006, pp. 43–50.

[7] D. Jannach and T. Schmitz, "Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach," *Automated Software Engineering*, vol. 23, no. 1, pp. 105–144, 2016.

[8] M. Fisher and G. Rothermel, "The EUSES Spreadsheet Corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms," *SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1–5, 2005.

[9] 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, Data Challenge*, 2015.

[10] R. Abraham and M. Erwig, "Mutation Operators for Spreadsheets," *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 94–108, 2009.

[11] F. Hermans and E. R. Murphy-Hill, "Enron's Spreadsheets and Related Emails: A Dataset and Analysis," in *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*, Florence, Italy, 2015, pp. 7–16.

[12] T. Schmitz and D. Jannach, "Enron Spreadsheet Error Finder," Published online at http://ls13-www.cs.tu-dortmund.de/homepage/spreadsheets/enron-spreadsheet-tool.shtml, last accessed 2016.

[13] ——, "The Enron Errors Corpus," Published online at http://ls13-www.cs.tu-dortmund.de/homepage/spreadsheets/enron-errors.htm, last accessed 2016.

[14] R. R. Panko and R. P. Halverson, "Are two heads better than one? (at reducing spreadsheet errors in spreadsheet modeling?)," *Office Systems Research Journal*, vol. 15, no. 1, pp. 21–32, 1997.

[15] ——, "Spreadsheets on Trial: A Survey of Research on Spreadsheet Risks," in *Proceedings of the 29th Hawaii International Conference on System Sciences (HICSS 1996)*, Wailea, HI, USA, 1996, pp. 326–335.

[16] S. Aurigemma and R. R. Panko, "The Detection of Human Spreadsheet Errors by Humans versus Inspection (Auditing) Software," in *Proceedings of EuSpRIG 2010 Conference*, London, United Kingdom, 2010.

[17] E. Getzner, "Improvements for Spectrum-based Fault Localization in Spreadsheets," Master's thesis, Graz University of Technology, http://spreadsheets.ist.tugraz.at/index.php/corpora-for-benchmarking/info1/, 2015.

[18] C. Chambers, M. Erwig, and M. Luckey, "SheetDiff: A Tool for Identifying Changes in Spreadsheets," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2010)*, Madrid, Spain, 2010, pp. 85–92.

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