Data Clone Detection and Visualization in Spreadsheets

Felienne Hermans, Ben Sedee, Martin Pinzger, and Arie van Deursen
Software Engineering Research Group
Delft University of Technology, Infotron
Netherlands
f.f.j.hermans@tudelft.nl, b.m.w.sedee@student.tudelft.nl, m.pinzger@tudelft.nl, arie.vandeursen@tudelft.nl

***Abstract*—Spreadsheets are widely used in industry: it is estimated that end-user programmers outnumber programmers by a factor 5. However, spreadsheets are error-prone, numerous companies have lost money because of spreadsheet errors. One of the causes for spreadsheet problems is the prevalence of copy-pasting.**

In this paper, we study this *cloning* in spreadsheets. Based on existing text-based clone detection algorithms, we have developed an algorithm to detect *data clones* in spreadsheets: formulas whose values are copied as plain text in a different location.

To evaluate the usefulness of the proposed approach, we conducted two evaluations. A quantitative evaluation in which we analyzed the EUSES corpus and a qualitative evaluation consisting of two case studies. The results of the evaluation clearly indicate that 1) data clones are common, 2) data clones pose threats to spreadsheet quality and 3) our approach supports users in finding and resolving data clones.

***Index Terms*—spreadsheets, clone detection, spreadsheet smells, code smells**

I. Introduction

Spreadsheets are heavily used within companies in many domains, ranging from financial to medical and from educational to logistics. It is estimated that 90% of desktops have Excel installed [1] and that the number of spreadsheet programmers is bigger than that of software programmers [2]. Because of their widespread use, they have been the topic of research since the nineties [3]. However, most papers focus on analyzing and testing the *formulas* in a spreadsheet.

The impact of data on spreadsheet calculations has been somewhat overshadowed by this interest in formulas. However, problems with data can pose threats to a spreadsheet's integrity too. A paper by Ballou *et al.* [4] phrases the problem as follows "...errors in the operational data can influence the determination of the most appropriate forecasting model" and "The manager is unlikely, however, to study the implications of errors in the data that are being projected. Clearly such errors have an impact, but it is not necessarily obvious which are potentially serious and which less". Although this paper is 25 years old, the problem statement is still very valid. In 2003, TransAlta lost US$24 Million because of a copy-paste error in a spreadsheet[1]. More recently, the Federal Reserve made a copy-paste error in their consumer credit statement which, although they did not make an official statement about the impact, could have led to a difference of US$4 billion[2]. These stories, although anecdotal, underline the fact that copy-paste errors in spreadsheets can greatly impact spreadsheet quality.

[1]http://bit.ly/cQRoy8

[2]http://bit.ly/6XwN9t

In this paper we focus on the occurrence of copy-pasting in spreadsheets by analyzing how the detection of *data clones* can help spreadsheet users in finding errors and improving the quality of their calculations. To that end we study related work in the field of clone detection in source code and come up with an approach to detect data clones in spreadsheets. In addition to exact clones, we also detect *near-miss clones*, those where minor to extensive modifications have been made to the copied fragments [5].

Our approach is based on existing text-based clone detection techniques [6], we use cell values as fingerprints and remove values that do not occur as formula and plain text. Subsequently, we group values that occur in multiple places into *clone clusters*, to detect groups of cells that are possibly copied.

Detected clones are visualized in two ways. Firstly, we generate a dataflow diagram that indicates how data is cloned between two worksheets, by drawing an arrow between boxes that represent those worksheets. This way, users can see how data is copied through worksheets and files. Secondly, we add pop-up boxes within the spreadsheet to show where data is copied, and in the case of near-miss clones, what cells differ.

This approach is subsequently validated both quantitatively and qualitatively. Firstly, we analyze the EUSES corpus [7] to calculate the precision and performance of our algorithm and to understand how often clones occur. Secondly, we perform two case studies: one with a large budget spreadsheet from our own university and a second one for a large Dutch non-profit organization, for which we analyzed 31 business critical spreadsheets.

From these three evaluations, we conclude that 1) data clones in spreadsheets are common, 2) data clones in spreadsheets often indicate problems and weaknesses in spreadsheets and 3) our algorithm is capable of detecting data clones quickly with 80% precision and supports spreadsheet users in finding errors and possibilities for improving a spreadsheet.

II. Related Work

As stated in the Introduction, Ballou *et al.* [4] described the problem of data quality in spreadsheets. More recently, O'Beirne [8] states that "...much present use of spreadsheets is as data manipulation and reporting tools used to bypass the controls around IT development. " And that this "ad hoc integration, transformation, or simple cobbling together is done by the user to get what they need when they need it. This gives rise to many extracted copies of corporate data as imports or query links in spreadsheet files. These personal data stores are often referred to as 'data shadows' or 'silos' or 'spreadmarts' giving rise to 'multiple versions of the truth'." [8] He furthermore cites evidence that "errors in the transfer of data from the field officer forms through to the DEFRA spreadsheet equating to an error rate of 14 percent over the year."

Clone detection in source code has been researched extensively and resulted in numerous clone detection techniques and tools. Bruntink *et al.* [9] give the following overview:

Text-based techniques perform little or no transformation to the raw source code before attempting to detect identical or similar (sequences of) lines of code. Typically, white space and comments are ignored [6], [10].

Token-based techniques apply a lexical analysis (tokenization) to the source code and, subsequently, use the tokens as a basis for clone detection [11], [12].

AST-based techniques use parsers to obtain a syntactical representation of the source code, typically an abstract syntax tree (AST). The clone detection algorithms then search for similar subtrees in this AST [13].

PDG-based approaches go one step further in obtaining a source code representation of high abstraction. Program dependence graphs (PDGs) contain information of a semantical nature, such as control and data flow of the program. Kommondoor and Horwitz [14] look for similar subgraphs in PDGs in order to detect similar code. Krinke [15] first augments a PDG with additional details on expressions and dependencies, and similarly applies an algorithm to look for similar subgraphs [14], [15].

Other related efforts are Roy [5], [16], who created NiCad, a parser-based and language specific tool that detects both exact and near-miss clones with high precision and recall. Roy and Cordy [16] compared several open source systems and study, among other properties, the occurrence of near-miss clones versus exact clones. They detected significantly higher numbers of near-miss clones than exact clones in the systems under evaluation. For more information on existing code clone detection techniques and tools, we refer the reader to the comprehensive survey by Roy *et al.* [17].

Our approach is text-based and is most similar to that of Johnson [6], where we use cell values as fingerprints.

Recently, there have been other efforts to apply clone detection to artifacts other than source code, Alalfi *et al.* [18] have successfully applied clone detection to SimuLink models. For this purpose they adapted NiCad and were able to efficiently find exact, renamed and near-miss clones in SimuLink models. To the best of our knowledge, no approach to detect data clones in spreadsheets exists. Most related are the efforst of Burnett *et al.* [19] that also compute similarity between cells, albeit in a quite different way and for the purpose of supporting the testing of spreadsheets. Also relevant is the work of Gold *et al.* that describe an approach to detect clones in dataflow languages [20].

Finally, there is our own work on spreadsheet analysis. Previously, we have worked on an algorithm to visualize spreadsheets as dataflow diagrams [21], and subsequently on detecting inter-worksheet smells in those diagrams [22]. Recently we have also worked on detecting smells in spreadsheet formulas [23]. This paper is a continuation of our research on smells in spreadsheets, however we shift our focus to detecting and visualizing clones in spreadsheet data. A tweet-sized paper on clones in spreadsheets was recently accepted in Tiny Transactions on Computer Science [24].

III. Motivation

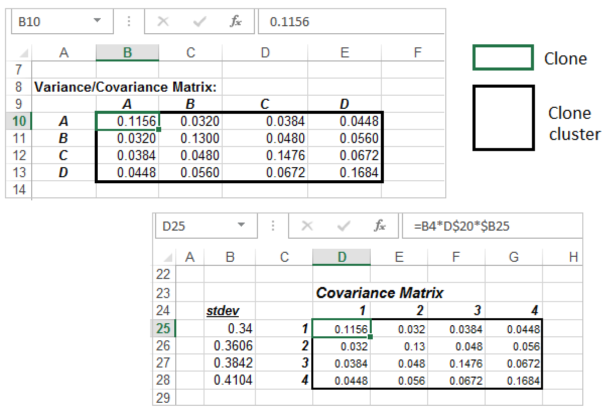
In our work with spreadsheet users, we often see that they copy and paste data from one spreadsheet to the other, from to worksheet to the other, and even within worksheets. When data is copy-pasted, or *cloned*, the spreadsheet might become more prone to errors, similar to the effect clones have on a software system.

Research in the field of source code analysis has analyzed the negative effect of clones on quality and maintenance. Mayrand *et al.* [25] showed that duplicated code fragments can increase maintenance effort. Furthermore a study by Jurgens *et al.* that analyzes industrial code shows that inconsistent changes to code duplicates are frequent and lead to severe unexpected behavior [26]. However, not all clones are harmful, Kapser en Godfrey [27] show that clones can also have a positive impact on maintainability.

Strictly, copy-pasting data in spreadsheets is not necessary: most spreadsheet systems have a feature where data can be linked from one worksheet to another and even from one file to another. In Excel, this can be done by either typing the location of the file in a formula, followed by the name of the worksheet and the cell(s), or by opening both worksheets or spreadsheets and creating the link by clicking, just as one would do with any formula.

So why would spreadsheet users resort to copy-pasting data from one spreadsheet file to the other, if most spreadsheet systems have a 'better' way to do this? In our experience, this practice can have several reasons. Firstly, sometimes users are not aware of a way to link spreadsheets, they do not know how to use the link-formulas. Secondly, users are often unaware of the risks of copy-pasting data and it seems the easiest way.

We do not aim at changing the spreadsheet users's behavior, since that would, most likely, involve changing the process around a spreadsheet and that would be hard to implement in a company. We rather follow an approach that allows users to proceed as they normally would, but mitigate the risks by detecting and visualizing the copy-paste relationships. This



B10 0.1156

	A	B	C	D	E	F
7						
8	Variance/Covariance Matrix:					
9		A	B	C	D	
10	A	0.1156	0.0320	0.0384	0.0448	
11	B	0.0320	0.1300	0.0480	0.0560	
12	C	0.0384	0.0480	0.1476	0.0672	
13	D	0.0448	0.0560	0.0672	0.1684	
14						

D25 =B4*D$20*$B25

	A	B	C	D	E	F	G	H
22								
23				Covariance Matrix				
24		stdev		1	2	3	4	
25		0.34	1	0.1156	0.032	0.0384	0.0448	
26		0.3606	2	0.032	0.13	0.048	0.056	
27		0.3842	3	0.0384	0.048	0.1476	0.0672	
28		0.4104	4	0.0448	0.056	0.0672	0.1684	
29								

Fig. 1. Cells B10 and D25 form a clone. Since all their non-empty neighboring cells are also clones, we detect two clone clusters in this example.

enables users to take the appropriate measures in a similarly pragmatic way.

Therefore the aim of this paper is to *both understand the impact of copy-pasting, and to develop a strategy to automatically detect data clones in spreadsheets*. We refine this goal into three research questions.

R1 How often do data clones occur in spreadsheets?

R2 What is the impact of data clones on spreadsheet quality?

R3 Does our approach to detect and visualize data clones in spreadsheets support users in finding and understanding data clones?

IV. Data Clones

The type of clones we are interested in are formula results that are copied as data for other parts of the spreadsheet. This type of copying is easy in Excel: with 'paste special' a user can copy values only.

We hypothesize that this type of cloning is risky: when formulas or their input is updated and their values change, this results in an extra maintenance task: updating the copies of the formulas too. If this is forgotten, errors could occur. We therefore look for tuples of a formula cell and a constant cell (a cell with a fixed value in it) that contain the same value. We call such a tuple of cells a *clone* and corresponds to what is called a *clone pair* in the clone detection literature.

It can, of course, happen by coincidence that a formula and a cell contain the same value. However, if a group of neighboring cells are all copies, or are all copied, we are probably dealing with data that has been copied by the user. We call such a group a *clone cluster*. In more detail: a clone cluster consists of cells that are all either formula cells or constant cells, and all of them are contained in a clone.

We call two clone clusters *matching* if they contain the same values. Figure 1 shows an example of a clone, in the small green rectangle, and two matching clone clusters in the gray rectangles.

Finally, we distinguish between clones clusters of which all values match and *near-miss* clone clusters, which contain cloned cells, but also cells that differ from each other. Near-miss clone clusters can occur when by a user updates a copied cell, but not does not update the original cell.

V. Data Clone Detection

In this section we describe the algorithm with which we detect clone clusters. This algorithm consists of 5 steps, as shown in the overview in Figure 2.

A. Algorithm

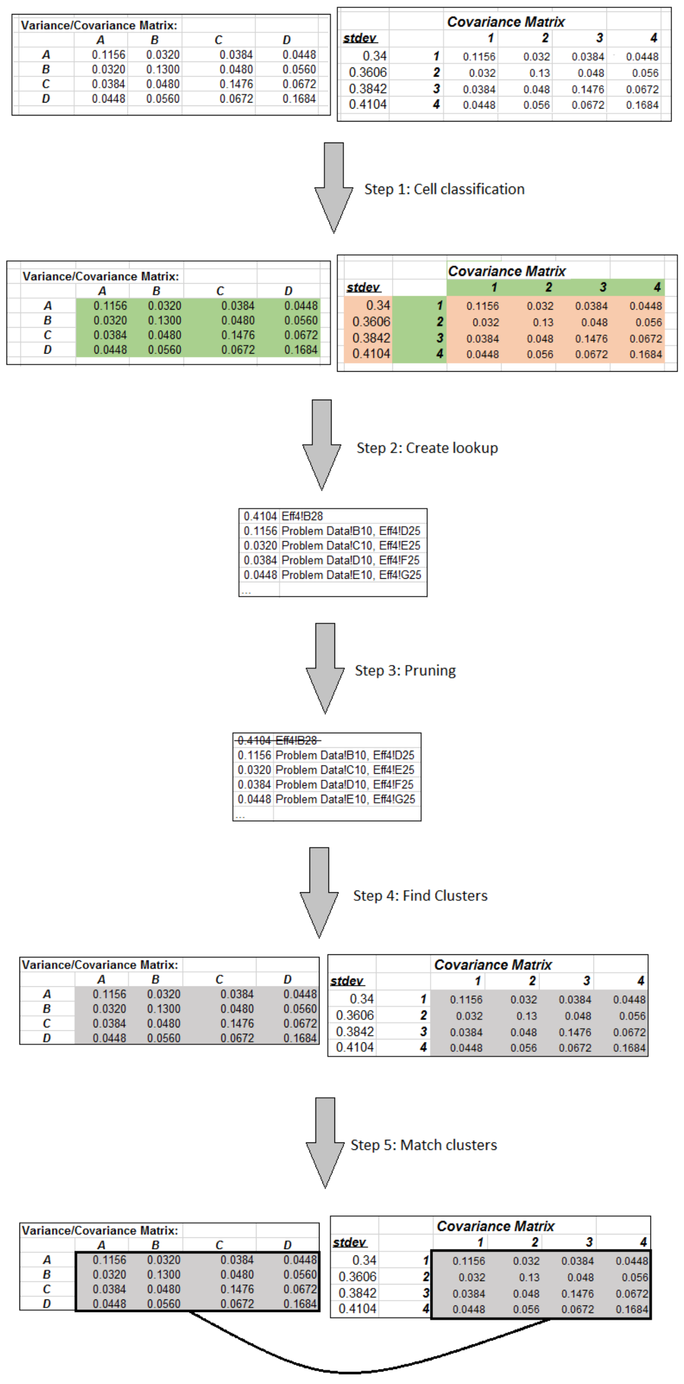
In the first step, *cell classification*, we divide the cells into data cells, formula cells and empty cells. For data cells, we only consider cells containing a number. Although strings can be the result of a formula, such as string concatenation or a lookup function, we do not take them into consideration, since strings are usually used for labels and descriptions in spreadsheets. In Figure 2 formula cells are colored pink and data cells containing a number are colored green. Note that this classification differs slightly from the cell classification algorithms we have used in previous papers [21], [28]. In our current version, all cells containing a number are considered data cells, and not only cells that are used as input for formulas. Since we are looking for clones, the data cells do not necessarily have to be used in calculations.

In the second step, *lookup creation*, a lookup table of all cells is created, with the cell value as key and a list of locations as the value. Shown in Figure 2 the value 0.4101 occurs only in Eff4!B28 and 0.1156 occurs in Problem Data!B10 and Eff4!D25. This step is similar to the creation of fingerprints in Johnson's clone detection approach [6].

The third step, *pruning*, removes all values from the lookup table that do not occur both in a formula and a constant cell, since these cells can never be part of a clone, conform our definition. In the example shown in Figure 2, 0.4101 is removed, since it only occurs in a formula and not in a constant cell.

In the subsequent fourth step called *cluster finding*, the algorithm looks for clusters of neighboring cells that are all contained in a clone, and that are all either formula cells or constant cells. The clusters are found by starting with a cell that is still contained in a value of the lookup table. In Figure 2, we start with cell Problem Data!B10 that contains 0.1156. Subsequently, this cell's neighbors are inspected. If these neighbors are contained in the lookup table too, the cluster is expanded and their neighbors are inspected. The fourth step of the algorithm results in a list of formula clusters and a list of constant clusters.

In the fifth and final step, *cluster matching*, each formula cluster is matched with each constant cluster. For two clusters to match, they have to contain the same values, i.e. all values in one cluster also have to occur in the second cluster. If one cluster is bigger than the other, all values of the smaller cluster have to be found in the second cluster. Furthermore, there may not be a formula link between the formula cells and the cloned



Variance/Covariance Matrix:				
	A	B	C	D
A	0.1156	0.0320	0.0384	0.0448
B	0.0320	0.1300	0.0480	0.0560
C	0.0384	0.0480	0.1476	0.0672
D	0.0448	0.0560	0.0672	0.1684

		Covariance Matrix			
stdev		1	2	3	4
0.34	1	0.1156	0.032	0.0384	0.0448
0.3606	2	0.032	0.13	0.048	0.056
0.3842	3	0.0384	0.048	0.1476	0.0672
0.4104	4	0.0448	0.056	0.0672	0.1684

Variance/Covariance Matrix:				
	A	B	C	D
A	0.1156	0.0320	0.0384	0.0448
B	0.0320	0.1300	0.0480	0.0560
C	0.0384	0.0480	0.1476	0.0672
D	0.0448	0.0560	0.0672	0.1684

		Covariance Matrix			
stdev		1	2	3	4
0.34	1	0.1156	0.032	0.0384	0.0448
0.3606	2	0.032	0.13	0.048	0.056
0.3842	3	0.0384	0.048	0.1476	0.0672
0.4104	4	0.0448	0.056	0.0672	0.1684

0.4104	Eff4!B28
0.1156	Problem Data!B10, Eff4!D25
0.0320	Problem Data!C10, Eff4!E25
0.0384	Problem Data!D10, Eff4!F25
0.0448	Problem Data!E10, Eff4!G25
...	

~~0.4104~~	~~Eff4!B28~~
0.1156	Problem Data!B10, Eff4!D25
0.0320	Problem Data!C10, Eff4!E25
0.0384	Problem Data!D10, Eff4!F25
0.0448	Problem Data!E10, Eff4!G25
...	

Variance/Covariance Matrix:				
	A	B	C	D
A	0.1156	0.0320	0.0384	0.0448
B	0.0320	0.1300	0.0480	0.0560
C	0.0384	0.0480	0.1476	0.0672
D	0.0448	0.0560	0.0672	0.1684

		Covariance Matrix			
stdev		1	2	3	4
0.34	1	0.1156	0.032	0.0384	0.0448
0.3606	2	0.032	0.13	0.048	0.056
0.3842	3	0.0384	0.048	0.1476	0.0672
0.4104	4	0.0448	0.056	0.0672	0.1684

Variance/Covariance Matrix:				
	A	B	C	D
A	0.1156	0.0320	0.0384	0.0448
B	0.0320	0.1300	0.0480	0.0560
C	0.0384	0.0480	0.1476	0.0672
D	0.0448	0.0560	0.0672	0.1684

		Covariance Matrix			
stdev		1	2	3	4
0.34	1	0.1156	0.032	0.0384	0.0448
0.3606	2	0.032	0.13	0.048	0.056
0.3842	3	0.0384	0.048	0.1476	0.0672
0.4104	4	0.0448	0.056	0.0672	0.1684

Fig. 2. Overview of the approach consisting of the five different steps: cell classification, lookup creation, pruning, cluster finding and cluster matching.

value cells, since we do not consider a link (i.e. =Sheet2!B1) to be a clone. In Figure 2, the two gray clusters match, since all values of the left cluster match with values on the right.

B. *Parameters*

The algorithm takes four parameters as input:
StepSize is used in the fourth step of the algorithm and indicates the search radius in terms of numbers of cells. Setting it to 1 means we are only looking for direct neighbors, with step size 2, a 'gap' of 1 cells is allowed in a cluster.
MatchPercentage is used in the final step, when clusters are matched. This percentage indicates what percentage of the cells has to match. Setting it to 100% means the values have to match exactly, lower percentages allow for the detection of near-miss clones.
MinimalClusterSize sets the minimal number of cells that a cluster has to consist of. Very small clusters might not be very interesting, hence we allow for a minimal threshold.
MinimalDifferentValues represents the minimal number of different values that have to occur in a clone cluster. Similar to small clusters, those clusters consisting of a few different values will be of less interest.
Furthermore, a user of the algorithm can indicate whether clones are found *within* worksheets, *between* worksheets, between spreadsheets or a combination of those.

VI. Clone visualization

We visualize the detected clones in two ways. Firstly, we generate a dataflow diagram that shows the relationship between worksheets that contains clones. Secondly, we add a pop-up to both parts of a clone indicating the source and the copied side of a clone.

The two visualizations serve a different purpose. The dataflow diagram is aimed at *understanding* the relationship between worksheets and show how data is copied between worksheets in a spreadsheet or between multiple spreadsheets.

The pop-ups within the spreadsheet, on the other hand, are meant for support when maintaining a spreadsheet. Whether it is updating the copied side of a clone or refactoring the copy into a link, the pop-ups support the spreadsheet user in selecting the right cells.

A. *Dataflow diagrams*

In previous work [21], [22] we have developed a tool to generate a dataflow diagram from a spreadsheet to represent dependencies between worksheets. In this visualization, worksheets are visualized as rectangles, while arrows are used to indicate a formula dependency between two worksheets.

We consider the copying of data from one worksheet to another as a dependency between worksheets and hence we decided to show this dependency in our original dataflow diagrams too. We show data clone dependencies with a dashed arrow to show the difference with formula dependencies which are shown with solid arrows.

Figure 3 shows the dataflow diagram corresponding to the spreadsheet hw4a.xls shown in Figure 1. In this spreadsheet

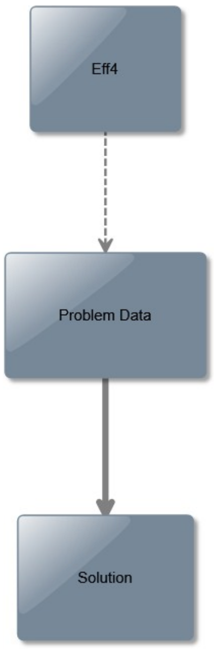
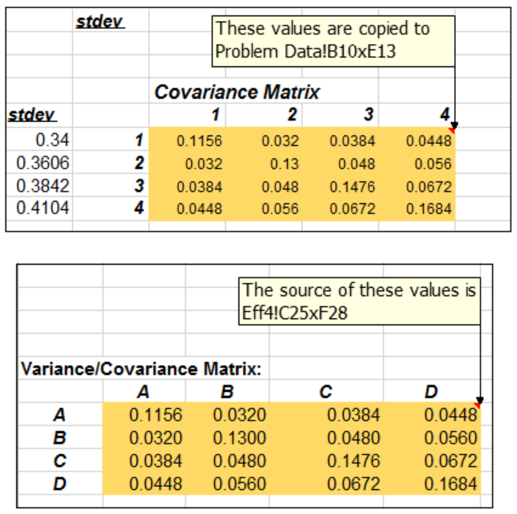


Fig. 3. Screenshot of the clone detection dataflow diagram corresponding to our running example hw4a.xls



stdev

These values are copied to Problem Data!B10xE13

Covariance Matrix

stdev		1	2	3	4
0.34	1	0.1156	0.032	0.0384	0.0448
0.3606	2	0.032	0.13	0.048	0.056
0.3842	3	0.0384	0.048	0.1476	0.0672
0.4104	4	0.0448	0.056	0.0672	0.1684

The source of these values is Eff4!C25xF28

Variance/Covariance Matrix:

	A	B	C	D
A	0.1156	0.0320	0.0384	0.0448
B	0.0320	0.1300	0.0480	0.0560
C	0.0384	0.0480	0.1476	0.0672
D	0.0448	0.0560	0.0672	0.1684

Fig. 4. Screenshot of the clone detection pop-up showing the copy-paste dependency for our running example hw4a.xls

data is copied from formulas in the worksheet *Eff4* to the worksheet *Problem Data*, this is shown in the diagram with a dashed arrow.

B. *Pop-ups*

As described above, we add pop-ups to the spreadsheets to both the source and the copied side of the clone. This warns the user that data has been copied, so he can update the copy in case of a change to the formulas. Furthermore provides an easy way for the user to improve the design of the spreadsheet, by changing the copy-paste relationship into a link. By creating a link, the dependencies are made explicit and future changes will automatically be propagated.

Figure 4 shows an example of a pop-up indicating a detected clone cluster. On the formula side, we show where the data is copied and on the data side, we indicate the source.

VII. Implementation

Our current approach for the detection of data clones in spreadsheets is implemented into our existing spreadsheet analysis tool Breviz [21], [22]. Breviz is implemented in C# 4.0 using Visual Studio 2010. It utilizes the Gembox component to read Excel files.[3] Breviz reads an Excel file and executes the above described clone detection algorithm, either within a spreadsheet or among multiple files, and subsequently generates the dataflow diagram and a copy of the spreadsheet with pop-ups.

Breviz, including the data clone analysis, is available as-a-service, by uploading a spreadsheet to Infotron's website.[4]

[3]http://www.gemboxsoftware.com/spreadsheet/overview
[4]http://app.infotron.nl

VIII. Evaluation Overview

To evaluate our approach, we performed both a quantitative and a qualitative analysis. First, we analyzed a subset of the EUSES corpus [7] to determine how well our algorithm performs and to learn how often data clones exist in this corpus. The corpus consists of more than 4000 spreadsheets from 11 different domains collected from practice.

Secondly, we studied two different real-life cases. The first case study was conducted at the South-Dutch foodbank, where employees keep track of all logistics using spreadsheets. For the second case study we evaluated a spreadsheet used by our university to calculate the budget for a large (> €25 Million) research proposal. With the qualitative analyses we aim to determine whether detected data clones actually pose a threat to spreadsheet quality.

IX. Quantitative evaluation

A. *Goal*

The aim of the first evaluation is to answer research question 1 "How often do data clones occur in spreadsheets?" and to preliminarily evaluate the performance of our algorithm both in terms of execution time as in terms of the precision of the algorithm.

B. *Background*

In this evaluation, we used spreadsheets from the EUSES corpus [7]. This corpus contains real-life spreadsheets from 11 different domains, ranging from educational to financial, and from inventory to biology. It was created in 2005 and has since then been used by several researchers to evaluate spreadsheet algorithms, among which [29] and [30]. Of the 4223 spreadsheets in the corpus, 1711 spreadsheets contain formulas.

C. Setup

To reach our goal, we ran our data clone detection algorithm on those 1711 spreadsheets, for different values of the MinimalClusterSize and MinimalDifferentValues parameter.

Since we do not have a validated benchmark, we focus on matching exact clones. Evaluating the correctness of near-miss clones without the owners of the spreadsheets would leave too much room for speculation. Hence we set MatchPercentage to 100% for the quantitative study. In the qualitative studies, we will take near-miss clones into consideration. Furthermore, we do not search for clones between the files of the EUSES corpus. Since the spreadsheets are unrelated, matches between spreadsheets would always be false positives.

For each detected clone, we manually determine whether this is a real clone or a false positive. We do this by inspecting clones and determining whether 1) the detected clone clusters indeed share the same data values, 2) one of the detected clone clusters consists of formulas and the other of constant cells and 3) headers of the found clones match to decide whether the clones indeed concern the same conceptual data. This way we calculate the precision of our approach. We calculate this precision as the percentage of spreadsheets in which we verified a clone, divided by the total number of spreadsheets in which a clone is detected by the algorithm, rather than measuring it as the number of verified clones divided by the number of detected clones. We do this because we found that some spreadsheets contain as many as 25 clones, all of which are very similar and this could skew the results.

Since we do not know what spreadsheets in the corpus contain clones, we cannot not analyze the recall of our algorithm at this point. It would be both time-consuming and speculative to inspect all 4000+ spreadsheets in the corpus by hand to check whether they contain data clones. We plan to analyze recall in a future study on a smaller set of spreadsheets of which we can contact the creators.

The results of our analysis are all available online in the FigShare corpus[5], to enable other researchers to replicate our results.

D. Findings

1) Precision: Using MinimalClusterSize 5 and MinimalDifferentValues 3, which we consider the lowest meaningful values, our algorithm detects 157 spreadsheet files in the EUSES corpus that contain clones. Manual inspection showed that of these detected files, 86 contain verified clones. This 86 is highlighted in Table II. 86 files out of 157 detected files with clones leads to a precision of 54.8%, as highlighted in Table I.

In this table, one can find the precision for different values of MinimalClusterSize and MinimalDifferentValues. Combinations where MinimalDifferentValues is bigger than MinimalClusterSize are not allowed, since there cannot be more different values in a clone cluster than cells.

[5]http://figshare.com/authors/FelienneHermans/98650

As illustrated by Table I, the precision rises for higher values of the two parameters, especially the parameter MinimalDifferentValues is of influence, as we suspected. Highest precision (81.7%) is obtained with both parameters set to 9, this value is also highlighted in Table I. In that case we still detect 49 clone files, which amounts to 57% of all 87 spreadsheets that contain verified clones in the EUSES test set (highlighted in Table II).

2) False positives: The biggest category of false positives is a group of data that happen to occur at multiple places in a spreadsheet. For instance in a spreadsheet used for grades, we detect several groups of the numbers 1 to 10. If some are calculations and others are input data, this is detected as a clone. Especially for low values of MinimalClusterSize and MinimalDifferentValues, both below 6, this occurs frequently, since chances that small groups of values are found in multiple places are higher. A second category of false positives is a clone that is actually a header: spreadsheet users use formulas to describe their data, such as a department code or a year. If in one case they use a formula and in another case they use a constant value, this is detected as a clone. Another type of false positives are clones consisting of array formulas that have the same value as other formulas in the worksheet. Gembox, the third party library we use to read spreadsheets, is not able to recognize array formulas, so they are read as being values.

3) Performance: Running the clone detection algorithm over the 1711 spreadsheet files in the EUSES corpus which contain formulas total took 3 hours, 49 minutes and 14 seconds (13,754 seconds in total). This means analyzing one file takes an average of 8.1 seconds.

4) Clone occurrence: In total, there are 1711 spreadsheets in the EUSES corpus that contain formulas, which means around 5% of all spreadsheets with formulas contain verified clones. Although not directly comparable, papers on clone analysis on source code estimate that 10 to 30% of lines of code are duplicated. For instance, Baker [12] reported that around 13% - 20% of large code bases can be clones. Lague *et al.* [31] found that, when considering function clones only, between 6.4% - 7.5% of code is cloned. Baxter et al. [13] have reported 12.7% cloning and Mayrand et al. [25] have estimated that industrial source code contains between 5% and 20% duplication.

5) Observations: While we cannot yet conclude something about the impact of data clones on spreadsheet quality, we noted several interesting similarities in this evaluation.

Firstly, we saw that a common pattern for cloning is the use in overviews and reports. In this scenario, spreadsheet users use one worksheet to calculate values, and copy them to a second worksheet to create a summary, a graph or a report sheet. Since many of these spreadsheets did contain links between worksheets, we do not think this use is due to the fact that the user did not know how to create links.

Secondly, we saw that copies are used to sort. In this scenario, a whole column is copied, sometimes directly next to the column with values, but the copy is sorted. This use might be due the way sorting in combination with links is

TABLE I
RESULTS OF THE EUSES EVALUATION SHOWING THE PERCENTAGE OF SPREADSHEETS CONTAINING A CLONE

	Minimal Different Values										
Minimal Size	3	4	5	6	7	8	9	10	11	12	13
5	**54.8%**	59.1%	63.7%	-	-	-	-	-	-	-	-
6	54.2%	59.2%	62.9%	70.1%	-	-	-	-	-	-	-
7	53.8%	59.1%	62.5%	69.5%	70.9%	-	-	-	-	-	-
8	56.1%	60.2%	63.6%	70.1%	71.6%	72.9%	-	-	-	-	-
9	56.6%	60.6%	64.3%	71.2%	72.9%	74.6%	**81.7%**	-	-	-	-
10	55.1%	58.6%	62.3%	69.7%	71.4%	73.3%	80%	79.2%	-	-	-
11	56.3%	57.7%	60.9%	68.3%	70.2%	71.4%	78.4%	77.6%	78.3%	-	-
12	56.6%	58.1%	60.6%	67.8%	69.6%	70.9%	78%	77.1%	77.8%	81%	-
13	56.9%	57.4%	61%	66.7%	69.2%	70.6%	76.6%	75.6%	76.7%	80%	80%

TABLE II
THE NUMBER OF SPREADSHEETS IN EUSES CONTAINING A DATA CLONE, FOR VARYING VALUES OF MINIMALDIFFERENTVALUES AND MINIMALSIZE

	Minimal Different Values										
Minimal Size	3	4	5	6	7	8	9	10	11	12	13
5	**86**	81	72	-	-	-	-	-	-	-	-
6	77	74	66	61	-	-	-	-	-	-	-
7	70	68	60	57	56	-	-	-	-	-	-
8	64	62	56	54	53	51	-	-	-	-	-
9	60	57	54	52	51	50	**49**	-	-	-	-
10	54	51	48	46	45	44	44	42	-	-	-
11	49	45	42	41	40	40	40	38	36	-	-
12	47	43	40	40	39	39	39	37	35	34	-
13	41	39	36	36	36	36	36	34	33	32	32

implemented in Excel. When one sorts a column that has links, the formulas get sorted too. Users might prefer to make a copy to keep their formulas intact.

Finally, an unexpected observation was that in some cases the format of the cells that were clones did not match. For instance, the original formulas were typed currency, while the copied cells were typed as a percentage. Even without knowing the context of this spreadsheet, we can conclude that one of the cell formats must be wrong. This practice is error-prone, especially in the case of dates. When a date is typed as a number, Excel will show a the number of days this day is removed from January 1, 1900, since Excel uses the 1900 date system. This way a user can easily overlook the fact that this value represents a date. In future work we plan to work on the detection of these mismatching clones.

X. THE CASE STUDIES

After we performed the quantitative evaluation and we were convinced of both the applicability of our approach and the frequency with which clones occur in practice, we conducted two case studies to investigate the implications of data clones in spreadsheets.

A. Goal

The goal of the two case studies is to answer research questions 2 and 3: to learn more about the impact of data clones and to evaluate our data clone detection and visualization approach.

B. Setup

To reach this goal, we have analyzed real-life spreadsheets in both case studies: we ran the data clone detection algorithm and subsequently we presented the results to the spreadsheet owners. Next, we went over all detected clones with them and asked them the following questions:

- Is this a real clone, in other words: did you copy this data?
- Did this clone lead to errors or problems?
- Could this clone be replaced by a formula link?

Furthermore, we asked them the following questions about clones and about our approach:

- Do you know why no direct links were used initially?
- How did the pop-ups help you in understanding the found data clones?
- How did the dataflow diagrams help you in understanding the found data clones?

C. Background

The following describes the background of the two case studies.

Foodbank A foodbank is a non-profit organization that distributes food to those who have difficulty purchasing it. We ran our case study at the distribution center of the foodbank, that supplies 27 local foodbanks. In 2011 the distribution center processed an average of 130.000 kilograms of food per month. To keep track of this process, they use a set of spreadsheets. The figures of incoming food from sponsor and food sent out to local foodbanks should balance, since no food is supposed to remain in the distribution center.

In January 2012, the distribution center of the foodbank approached us and asked whether we could help them improve their spreadsheet administration, since they observed that the total result did not balance and food remained in the center, or went missing.

Initially, we did not know what caused their problems, but when we learned about the copy-pasting practice that was used, we suspected that clone detection might help to locate the errors. We asked the foodbank whether they would be interested in participating in a study of a new feature we were developing, with which they agreed.

Subsequently, we received 31 spreadsheet files from the foodbank, to check whether clones might be the source of problems. One of those spreadsheets was the distribution list, while the other 30 were lists of a specific region.

Delft University In April of 2012 Delft University of Technology participated in a grant proposal, for which a budget spreadsheet had to be created. One of the authors of this paper was involved in this proposal and got this spreadsheet from a financial employee of the university.

This particular spreadsheet calculates, among other things, the salary cost of different types of employees. These salaries are raised every year, because of inflation, and the creator of this spreadsheet calculated the salaries once and copied them to different places in the spreadsheet.

The author involved in this proposal noticed this duplication and asked this employee whether he would want to participate in a study on cloning in spreadsheets and this employee agreed.

D. Findings

In this subsection we describe the results of both case studies.

Foodbank In the first study, we searched for data clones in the test set of the foodbank by running the prototype tool over the 31 spreadsheets. We used parameters 9 for MinimalClusterSize and MinimalDifferentValues, since they were the optimal choice in the quantitative analysis. Furthermore we set MatchingPercentage to 80% and StepSize to 2 to enable the detection of near-miss clones. Running the algorithm took 3 hours, 9 minutes and 39 seconds, which amounts to 6 minutes per file. Performance was worse than in the EUSES case, since in this analysis, all clones of all files had to be compared with each other, since we were searching for clones between files too.

With these settings, we detected 145 clones, of which 61 were near-miss clones, in other words, they had a matching percentage of less than 100%. Furthermore, in this case we only searched for clones between spreadsheets, since we knew that there would only be clones between files. We discussed the found clones one by one with three employees of the foodbank and checked whether the found clones were actual clones. Only one of the found clones was identified as a false positive: in that case, by coincidence two files contained similar values.

Subsequently, we studied the near-miss clones in more detail: were they really errors that had been made in copy-pasting? We found that all cases were 'wrong' in the sense that the values should match. The employee that we discussed the results with stated "these should always match, I don't understand why they do not." However, in many cases, the updates made to the copies were the right values, but in 25 of the detected 61 near-miss clones were actual errors that the employees were not aware of before our analysis. In the other cases, the copies were not mistaken. For instance, in some cases only half of the column was copied, because for the other items, the amounts from the previous month remained in the distribution center. While checking the near-miss clones, we also found that one of the exact clones was actually an error: here the data had been copied into the wrong column. The foodbank employees stated that all found clones could, in theory, be replaced by direct links. No direct links were used initially, since the employee who created the spreadsheets, was not very proficient in spreadsheets at that time. She started with a few spreadsheets and copying the values would not be much of a problem. When the collection of spreadsheets got bigger, it became increasingly more difficult to make the change.

Later on, another employee was put in charge of handling the spreadsheets. She stated: "Since I did not build all the sheets, I am always a bit afraid to adapt formulas. Since I can see the links in the pop-ups that you created, I feel more safe, since I know it will do the right things."

This sentiment is shared even by the original creator of the spreadsheets, saying "The problem with managing multiple sheets is that you never know if changing one cell will mess up other sheets or files." Especially for the current maintainer of the spreadsheets, seeing files that were not linked was insightful. "I assumed this region was already copied into the total sheet, but in the diagram I see it is not. I should fix that right away." After the employees fixed the clones that we found, the overall results balanced as they should, meaning less food is wasted on a monthly basis, which we consider a very good result and strong evidence that data clones can indeed be the cause of errors.

Delft University In the case study for the Delft University, we studied the budget spreadsheet, consisting of 15 worksheets. We again used MinimalClusterSize 9 and MinimalDifferentValues 9 and set the MatchingPercentage to 80% and StepSize to 2. In this case study, we searched for clones between worksheets, since there was just one spreadsheet. Running the algorithm took 3 seconds.

We found 8 exact clones, which all turned out to be real clones, i.e. they were copied by the spreadsheet creator. When we asked him why he used the clones instead of links, he stated that the spreadsheet was a draft version and that it seemed easier to simply copy the values. Although these clones did not lead to problems in this case, the spreadsheet owner did state that if he had to share the spreadsheet with others, he thought he should replace the clones with links. He stated that our analysis would be very useful in improving the spreadsheet and removing the clones: "This scan is very useful. You can just copy-paste and the system indicates where you should make links". In this case study, the visualization was interesting, since there were two worksheets that were both connected by a solid arrow, indicating formula dependencies, and a dashed arrow which shows a clone dependency. We consider this as extra risky, because the spreadsheet's user might think all values are linked upon seeing one of the formula dependencies.

XI. The research questions revisited

In this section, we revisit the research questions.

R1: How often do data clones occur in spreadsheets? We learned from the both EUSES case and the case studies that clones occur often in spreadsheets. Around 5% of all spreadsheets in the EUSES corpus contain clones.

R2: What is the impact of data clones on spreadsheet quality? From the two case studies, we learned that clones can have two different types of risks. We learned that clones matching 100% mainly impact the users perspective of spreadsheets ("I did not know these values were copied from that source"), while near-miss clones really causes trouble ("this value should have been updated months ago").

R3: Does our approach to detect and visualize data clones in spreadsheets support users in finding and understanding data clones? In both studies we saw that employees were not always aware of what values were copied from what sheets to what others. Even creators of the spreadsheets did not know all relations by heart. The dataflow visualizations aided users in quickly getting an overview of the, otherwise very hidden, copy dependencies ("the system indicates where you should make links").

XII. Discussion

Our current approach to finding clones helps spreadsheet users to understand how data is copied throughout worksheets and spreadsheets and furthermore supports them in improving erroneous relations. In this section, we discuss a variety of issues that affect the applicability and suitability of the proposed approach.

A. Relative settings for parameters

In the current evaluations, we have used fixed settings for the parameters: we set MinimalDifferentValues and MinimalClusterSize both to 9, irrespective of the spreadsheet. However, it could improve performance of the algorithm if these parameters were calibrated using properties of the spreadsheets. For instance, in a spreadsheet with only 8 rows, no clones will ever be found. Although the evaluations showed that using fixed settings leads to useful results, taking spreadsheet properties into account might improve the algorithm even further.

B. Headers

In previous work, we have worked on the extraction of meta data from spreadsheets [28]. Other authors have worked on this as well [32], [33]. We could use extracted header information, such as column or row names to gain more confidence in detected clones. If clones are found below column headers with the same name, chances are bigger that clones are are 'real' clones and concern the same conceptual data.

C. Copied data

In addition to copying the results of formulas, a spreadsheet user can also copy data to different places of the spreadsheet. This is a different type of cloning in spreadsheets that we have not yet considered for this paper. We hypothesize that copy-pasting of data might also be error-prone, however this calls for more research. Furthermore, there is be the challenge of detecting the origin of the clone, similar to origin analysis in source code [34], [35]. We see this as an interesting avenue for future research.

D. Clone genealogy

The current approach analyzes cloning as it occurs in a spreadsheet at a given point in time. However, it would also be very interesting to understand how clones are created and adapted. When spreadsheets are be placed under version control, such as provided by Microsoft's Sharepoint, for example, it will be possible to also track the history of a set of clones, similar to clone genealogy work in source code analysis [36], [37].

E. Spreadsheet maintenance support

We learned from the case study at the foodbank that, when spreadsheets become larger and more complex, their users become more anxious to make structural changes. Although this does not relate to cloning directly, updating a clone into a formula link is one of those changes that users fear might mess up the entire sheet. This means that spreadsheets need better support for previewing a change, such as some refactoring tools offer, or the option to calculate what cells will be affected by a certain change.

F. Threats to validity

A treat to the internal validity of our quantitative evaluation is the fact that we validated the precision of the algorithm manually. We have however described our approach and made our results publicly available, so our results can be replicated. A threat to the external validity of our quantitative evaluation concerns the representativeness of the EUSES corpus. However, the EUSES corpus is a large set that is collected from practice and has been used for numerous spreadsheet papers. The external validity of the qualitative evaluation might suffer from this threat, however these spreadsheets too are collected from practice and available online to enable other researchers to replicate our results.

XIII. Concluding remarks

The goal of this paper is to investigate the risks that data clones pose to spreadsheets. To that end we have defined data clones and described a way to detect and visualize them. We have subsequently evaluated data clones in two ways, with a quantitative evaluation on the EUSES corpus and two real-life case studies in which we found that data clones are common and can lead to real errors.

The key contributions of this paper are as follows:

- The definition of data clones in spreadsheets (Section IV).

- An approach for the automatic detection (Section V) and visualization (Section VI).
- An implementation of that approach into our existing spreadsheet analysis toolkit Breviz (Section VII).
- A quantitative evaluation of the proposed clone detection algorithm on the EUSES corpus (Section X).
- A real-life evaluation with 31 spreadsheet from a Dutch non-profit organization and 1 from academia (Section IX).

The results of our evaluations show that around 5% of spreadsheets contain data clones and that these clones lead to actual errors such as in the case of the foodbank. The current research gives rise to several directions for future work. Firstly , the algorithm could be improved to get a better precision. Also, in a new study, we will analyze the recall of the algorithm and on detecting clone that do not match in their number format, since these might be even more error-prone than the data clones we detect currently. Furthermore, the case study learned us that impact analysis of changes in spreadsheets could help to increase a spreadsheet user's confidence, either before the change or directly after. This is a very interesting avenue for further research.

Finally, taking the meta data into account might strengthen the clone detection algorithm.

REFERENCES

[1] L. Bradley and K. McDaid, "Using bayesian statistical methods to determine the level of error in large spreadsheets," in *Proc. of ICSE '09, Companion Volume*, 2009, pp. 351–354.

[2] C. Scaffidi, M. Shaw, and B. A. Myers, "Estimating the numbers of end users and end user programmers," in *Proc. of VL/HCC '05*, 2005, pp. 207–214.

[3] D. Bell and M. Parr, "Spreadsheets: A research agenda," *SIGPLAN Notices*, vol. 28, no. 9, pp. 26–28, 1993.

[4] D. P. Ballou, H. L. Pazer, S. Belardo, and B. D. Klein, "Implication of data quality for spreadsheet analysis," *DATA BASE*, vol. 18, no. 3, pp. 13–19, 1987.

[5] C. K. Roy, "Detection and analysis of near-miss software clones," in *Proc. of ICSM '09*, 2009, pp. 447–450.

[6] J. H. Johnson, "Identifying redundancy in source code using fingerprints," in *Proc. of CASCON '93*, 1993, pp. 171–183.

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

[8] P. O'Beirne, "Information and data quality in spreadsheets," in *Proc. of Eusprig '08*, 2008.

[9] M. Bruntink, A. van Deursen, R. van Engelen, and T. Tourwé, "On the use of clone detection for identifying crosscutting concern code," *TSE*, vol. 31, no. 10, pp. 804–818, 2005.

[10] S. Ducasse, M. Rieger, and S. Demeyer, "A language independent approach for detecting duplicated code," in *Proc. of ICSM '99*, 1999, pp. 109–118.

[11] T. Kamiya, S. Kusumoto, and K. Inoue, "CCfinder: A multilinguistic token-based code clone detection system for large scale source code," *TSE*, vol. 28, no. 7, pp. 654–670, 2002.

[12] B. S. Baker, "On finding duplication and near-duplication in large software systems," in *Proc. of WCRE '95*, 1995, pp. 86–95.

[13] I. D. Baxter, A. Yahin, L. M. de Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proc. of ICSM '98*, 1998, pp. 368–377.

[14] R. Komondoor and S. Horwitz, "Using slicing to identify duplication in source code," in *Proc. of SAS '01*, 2001, pp. 40–56.

[15] J. Krinke, "Identifying similar code with program dependence graphs," in *Proc. of WCRE '09*, 2001, pp. 301–309.

[16] C. K. Roy and J. R. Cordy, "Near-miss function clones in open source software: an empirical study," *Journal of Software Maintenance*, vol. 22, no. 3, pp. 165–189, 2010.

[17] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol. 74, no. 7, pp. 470–495, 2009.

[18] M. H. Alalfi, J. R. Cordy, T. R. Dean, M. Stephan, and A. Stevenson, "Models are code too: Near-miss clone detection for simulink models," in *Proc. of ICSM '12*, 2012, to appear.

[19] M. Burnett, A. Sheretov, B. Ren, and G. Rothermel, "Testing homogeneous spreadsheet grids with the "what you see is what you test" methodology," *TSE*, vol. 28, no. 6, pp. 576–594, 2002.

[20] N. Gold, J. Krinke, M. Harman, and D. Binkley, "Issues in clone classification for dataflow languages," in *Proc. of IWSC '10*, 2010, pp. 83–84.

[21] F. Hermans, M. Pinzger, and A. van Deursen, "Supporting professional spreadsheet users by generating leveled dataflow diagrams," in *Proc. of ICSE '11*, 2011, pp. 451–460.

[22] ——, "Detecting and visualizing inter-worksheet smells in spreadsheets," in *Proc of ICSE '12*, 2012, pp. 441–451.

[23] ——, "Detecting code smells in spreadsheet formulas," in *Proc of ICSM '12*, 2012, to appear.

[24] F. Hermans, "Exact and near-miss clone detection in spreadsheets," *TinyToCS*, vol. 1, no. 1, 2012.

[25] J. Mayrand, C. Leblanc, and E. Merlo, "Experiment on the automatic detection of function clones in a software system using metrics," in *Proc. of ICSM '96*, 1996, pp. 244–.

[26] E. Jürgens, F. Deissenboeck, B. Hummel, and S. Wagner, "Do code clones matter?" in *Proc. of ICSE '09*, 2009, pp. 485–495.

[27] C. J. Kapser and M. W. Godfrey, ""cloning considered harmful" considered harmful: patterns of cloning in software," *Empirical Software Engineering*, vol. 13, no. 6, pp. 645–692, 2008.

[28] F. Hermans, M. Pinzger, and A. van Deursen, "Automatically extracting class diagrams from spreadsheets," in *Proc. of ECOOP '10*, 2010, pp. 52–75.

[29] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proc. of ICSE '06*, 2006, pp. 182–191.

[30] J. Cunha, J. Saraiva, and J. Visser, "Discovery-based edit assistance for spreadsheets," in *Proc. of VL/HCC '09*, 2009, pp. 233–237.

[31] B. Laguë, D. Proulx, J. Mayrand, E. Merlo, and J. P. Hudepohl, "Assessing the benefits of incorporating function clone detection in a development process," in *Proc. of ICSM '97*, 1997, pp. 314–321.

[32] R. Abraham and M. Erwig, "Header and unit inference for spreadsheets through spatial analyses," in *Proc. of VL/HCC '04*, 2004, pp. 165–172.

[33] M. Erwig and M. M. Burnett, "Adding apples and oranges," in *Proc. of PADL '02*, 2002, pp. 173–191.

[34] M. Godfrey and Q. Tu, "Tracking structural evolution using origin analysis," in *Proc. of IWPSE '02*, 2002, pp. 117–119.

[35] L. Zou and M. W. Godfrey, "Detecting merging and splitting using origin analysis," in *Proc. of WCRE '03*, 2003, pp. 146–154.

[36] M. Kim and D. Notkin, "Using a clone genealogy extractor for understanding and supporting evolution of code clones," in *Proc. of MSR '05*, 2005.

[37] T. Bakota, "Tracking the evolution of code clones," in *Proc. of SOFSEM '11*, 2011, pp. 86–98.