Code Smells in Spreadsheet Formulas Revisited on an Industrial Dataset

Bas Jansen
Delft University of Technology
The Netherlands
b.jansen@tudelft.nl

Felienne Hermans
Delft University of Technology
The Netherlands
f.f.j.hermans@tudelft.nl

***Abstract*—In previous work, code smells have been adapted to be applicable on spreadsheet formulas. The smell detection algorithm used in this earlier study was validated on a small dataset of industrial spreadsheets by interviewing the users of these spreadsheets and asking them about their opinion about the found smells. In this paper a more in depth validation of the algorithm is done by analyzing a set of spreadsheets of which users indicated whether or not they are smelly.**

This new dataset gives us the unique possibility to get more insight in how we can distinguish 'bad' spreadsheets from 'good' spreadsheets. We do that in two ways: For both the smelly and non smelly spreadsheets we 1) have calculated the metrics that detect the smells and 2) have calculated metrics with respect to size, level of coupling, and the use of functions. The results show that indeed the metrics for the smells decrease in spreadsheets that are not smelly. With respect to size we found to our surprise that the improved spreadsheets were not smaller, but bigger. With regard to coupling and the use of functions both datasets are similar. It indicates that it is difficult to use metrics with respect to size, degree of coupling or use of functions to draw conclusions on the complexity of a spreadsheet.

I. Introduction

Spreadsheets could be considered the most successful end-user programming platform, with an estimated 55 million people using them in the US alone. Because spreadsheets are so widely used as programming tools, it is plausible to apply methods from software engineering to them in order to improve them. This has been done in previous work, among others by Hermans *et al.* [1] who translated some of Fowler's code smells [2] to the realm of spreadsheets.

In their paper, a method for the detection of *spreadsheet smells* is described, including for example a long list of referenced cells or deeply nested conditional formulas. To evaluate their smell detecting algorithm, Hermans *et al.* detected smells in ten spreadsheets created by employees of an investment bank, and subsequently interviewed the users of these spreadsheets, asking their opinion about the found smells. This study had two obvious limitations: firstly, the dataset used in this evaluation was very small, consisting of only 10 spreadsheets stemming from one company. Secondly, it was not known in advance if these spreadsheet were suffering from smells; users were just asked for 'complex spreadsheets'.

This paper presents a more extensive investigation of spreadsheet smells on an entirely new dataset, obtained from financial modeling company F1F9. Employees of F1F9 develop financial models in Excel for customers, often based upon the customer's existing spreadsheet models. We have obtained 54 pairs of spreadsheets consisting of the original model developed by the customer and the rebuilt model created by F1F9 employees. Customers in general reach out to F1F9 because they cannot maintain their spreadsheets models anymore, in other words: they are smelly. As such, these pairs of smelly and non-smelly versions of the same spreadsheet provide ample opportunity for us to investigate what characterizes a smelly spreadsheet.

To do so, we have performed an evaluation in which we detected smells for both smelly and non-smelly spreadsheets. We have applied both the Wilcoxon Signed-Ranks Test for Paired Samples and the Wilcoxon-Mann-Whitney test to see if there is significant difference between the two types of spreadsheets. We find that indeed the rebuilt spreadsheets contain smells less frequently. In addition to calculating smells, we also calculated size and coupling metrics of the two types of spreadsheets and investigated their use of functions. Surprisingly enough, the rebuilt sheets are not smaller, but bigger, and seem very similar in terms of coupling and function use. Hence, these metrics do not offer value when trying to distinguish maintainable from smelly spreadsheets.

With our work, we improve upon the existing, preliminary, study in two ways. Firstly, our dataset is bigger, consisting of 108 spreadsheets. More importantly, our set is based on pairs of spreadsheets, one being an original, smelly spreadsheet, and the other being a rebuilt, well-structured model, allowing us to pair-wise compare the models.

The remainder of this paper is structured as follows: in the next section we give background information about the smells that we used to analyze the spreadsheets. In Section III we describe the setup of our analysis. We explain the content of the dataset and the procedure we followed to calculate the different metrics. The results of the analysis are presented in section IV. In Section V we discuss the results in more detail and put them in the context of the FAST standard that was used by F1F9 to rebuild the financial models. Several issues that affect the applicability and suitability of the findings are discussed in section VI and we finish the paper with related work (Section VII) and the concluding remarks (VIII).

978-1-4673-7532-0/15 © 2015 IEEE

II. Background

Hermans *et. al.* [1] introduced 5 smells in spreadsheet formulas:

- **Multiple Operations**: Inspired by the code smell Long Method, this smell indicates the length of the formula. It measures the total number of operations that a formula contains. Figure 1 shows a formula that is suffering from this smell with a total of 15 unique operations within the formula.

```
=IF(AND(NOT(I10);NOT(AND(Assumptions!$E$401>=I24;Assumptions!
$E$402<=I24));I7>Assumptions!$E$399);MAX(-I15*IF(I24=0;0;INDEX(
debt_cf; MATCH(I24; debt_cf_dates;1)))*(SUM(Quarters!H128:I129));-(
I33+I34+(libor+Assumptions!$E$382+rac)*((I28-H28)/365)*(H33+I33)*I18)
);-H36-I36)*I29
```

Fig. 1. Example of Multiple Operations Smell

- **Multiple References**: Another well known code smell is Many Parameters. The spreadsheet formula equivalent is Multiple References. It counts the number of ranges a formula is referring to. An example of this smell is a formula with 79 references that is shown in Figure 2.

```
=MIN(0;R$587;R$587+S$587;R$587+S$587+T$587;R$587+S$587+T$587+
U$587;R$587+S$587+T$587+U$587+V$587;R$587+S$587+T$587+U$587+
V$587+W$587;R$587+S$587+T$587+U$587+V$587+W$587+X$587;
R$587+S$587+T$587+U$587+V$587+W$587+X$587+Y$587;R$587+
S$587+T$587+U$587+V$587+W$587+X$587+Y$587+Z$587;R$587+
S$587+T$587+U$587+V$587+W$587+X$587+Y$587+Z$587+AA$587;
R$587+S$587+T$587+U$587+V$587+W$587+X$587+Y$587+Z$587+
AA$587+AB$587;R$587+S$587+T$587+U$587+V$587+W$587+X$587+
Y$587+Z$587+AA$587+AB$587+AC$587)*Q$803
```

Fig. 2. Example of Multiple References Smell

- **Conditional Complexity**: Many nested conditional operations are considered as a threat to code readability [2]. The same is true for spreadsheet formulas. The Conditional Complexity smell measures the number of conditionals contained by a formula. Figure 3 shows a formula with 7 nested IFs functions. This was the maximum number of nested IFs that was allowed up to Excel 2003.

```
=(IF(A7<='Flags(mth)'!$D$21;0;IF(A7<=Inputs!$E$19;IF(A7<Inputs!$E$19;(
(A7-MIN(A6;Inputs!$C$13))/(Inputs!$D$55-Inputs!$C$13))*LOOKUP(
Inputs!$D$55;Notes!$E$8:$CB$8;Notes!$E$553:$CB$553)+C6;IF(A7=
Inputs!$E$19;((A7-MAX(A6;Inputs!$C$13))/(Inputs!$D$55-Inputs!$C$13))
*LOOKUP(Inputs!$D$55;Notes!$E$8:$CB$8;Notes!$E$553:$CB$553);IF(
Repayment!A7=Inputs!$E$19;OFFSET(Notes!$I$553;0;Repayment!#REF!);
IF(A7=Inputs!$D$61;OFFSET(Notes!$I$553;0;Repayment!#REF!);IF(A7>
Inputs!$D$61;0;OFFSET(Notes!$I$553;0;Repayment!#REF!)/2)))))))*
1000000
```

Fig. 3. Example of Conditional Complexity Smell

- **Long Calculation Chain**: In spreadsheets, it is common that formulas refer to other formulas. Therefore, one could say that a spreadsheet consists of a collection of calculation chains. Tracing a long calculation chain is considered by users as a tedious task. The smell is measured by the length of the longest path of cells that need to be referenced when computing the value of the formula.
- **Duplicated Formulas** The equivalent of the Duplicate Code smell in spreadsheets is the *Duplicated Formulas* smell. The smell as described in [1] occurs at formulas that are partially the same as others. The smell is measured by the number of formulas, located in the same worksheet and expressed in relative R1C1 notation, with which a formula shares at least one proper subtree. In the original study, it was found that users found this smell hard to understand. Therefore we changed the definition for this smell. It is now measured by the number of identical formulas that are located in the same worksheet and having at least one function or operator. Row 39 in Figure 4 shows an example of four identical formulas.

	V	W	X	Y
1				
2	42010	42377	42743	43108
39	=$F39/$E39	=$F39/$E39	=$F39/$E39	=$F39/$E39
40	=$F40/$E40	=$F40/$E40	=$F40/$E40	=$F40/$E40
41	=$F41/$E41/2	=$F41/$E41	=$F41/$E41	=$F41/$E41
42	=$F42/$E42/2	=$F42/$E42	=$F42/$E42	=$F42/$E42
43		=$F43/$E43/2	=$F43/$E43	=$F43/$E43
44		=$F44/$E44/2	=$F44/$E44	=$F44/$E44
45			=$F45/$E45/2	=$F45/$E45/2
46			=$F46/$E46/2	=$F46/$E46/2
47	=SUM(V33:V46)	=SUM(W33:W46)	=SUM(X33:X46)	=SUM(Y33:Y46)

Fig. 4. Example of Duplicated Formula Smell

We received the dataset that we use in this paper from F1F9[1]. They are the world largest financial model building firm. They build their models using spreadsheets, according to the FAST standard. The FAST standard [3] was first developed by employees of F1F9. It is now maintained by the FAST Standard Organization[2]. The standard is primarily concerned with good spreadsheet design. Its acronym stands for Flexible, Appropriate, Structured, and Transparent. It aims to support spreadsheet designers to build spreadsheets that: Are free of fundamental omissions; Have no logical errors; Can be created under short lead times; Can be easily used and reviewed; Are readily adaptable when circumstances change.

III. Experimental Setup

The dataset we use for this paper consists of 54 pairs of spreadsheets. For every pair one spreadsheet was created by a client of F1F9, the other one an improved version built by consultants of F1F9 according to the FAST standard. Both spreadsheets have the same functionality and deliver the same results for identical input. However, the models within the spreadsheets are completely different. F1F9 built their version

[1]To protect the confidentiality of the models we only had access to the dataset on F1F9s premises and then only indirectly whereby our software automatically generated and stored only the necessary survey statistics. At no point did we have direct access to the models, nor did our software extract any commercial data from the models.

[2]http://www.fast-standard.org/

completely from scratch. All spreadsheets contain financial models.

When we received the dataset, it consisted of a total of 146 spreadsheets. However, we discovered that the set contained some duplicates and that for some client files the matching F1F9 file was missing and vice versa. So after an initial cleaning, a set of 130 files remained. Subsequently, we analyzed these spreadsheets with the Spreadsheet Scantool, developed at Delft University of Technology. The tool runs on the previously developed Breviz core that was made for spreadsheet visualization and smell detection [4]. Some of the remaining files were password protected, corrupt or otherwise unreadable by the scantool and were therefore excluded from the dataset. Of course if a client file was unreadable, we also had to exclude the matching F1F9 file. Eventually we ended up with 108 scanned files.

The spreadsheets in the Client set were perceived by their users as problematic. It was because of this reason, that they asked F1F9 to rebuild these models. What makes the dataset interesting for our research is that we have one set of spreadsheets that are perceived by their users as problematic and a matching set of spreadsheets that, according to professional model builders, are easier to understand, less error-prone and less difficult to maintain.

In earlier studies regarding smells in spreadsheets the smells were validated by either asking users about their opinion about the smelliness [1] or by manual inspecting the detected smells to see if they were actually smelly [5]. What was missing in these validations, was ground truth about the smelliness of a spreadsheet. Fortunately, we can solve this now with the above mentioned dataset.

One would expect that a spreadsheet that is considered easier to understand, less error-prone and less difficult to maintain contains less smells than a spreadsheet that is perceived as problematic. This brings us to the main question for this research:

R1 Do spreadsheets that are perceived as easier to understand, contain fewer smelly cells than spreadsheets that are perceived as problematic?

In earlier work [6], metrics for size, coupling and the use of functions were used to characterize spreadsheets in large corpora. We will use these metrics to analyze the differences between the client and the F1F9 spreadsheets. Table I gives an overview of these metrics.

Most of these metrics are self-explanatory, however a few deserve some further explanation [6]:

- **Number of unique formulas per spreadsheet (s4)**: It is common practice in spreadsheets to define a formula in once cell and then copy it down or right to other cells. As a consequence, many of the formula cells in a spreadsheet contain the same formula except for the references to the other cells. Therefore, we also measure the number of unique formulas in the spreadsheet. We determine the unique formulas by looking at the relative R1C1 notation of the formula. As described by Sajaniemi [7], this notation stays the same even if you copy a formula down or right.
- **Path depth (c3), transitive precedents (c4), and number of preceding cells (f3)**: In most cases formulas receive input from other cells. This is what we measure with the number of preceding cells per formula. However, these precedents could be formulas themselves that, in turn, have their own precedents. The number of transitive precedents is calculated by tracing along these precedents until on all branches a cell is reached without any precedents. The path depth is the longest calculation chain within the tree of precedents. See also Figure 5.
- **Parse tree depth (f2)**: This metric indicates how nested a formula is. The formula A1 + A2 has a parse tree depth of 2, the formula (A1 - A2) / (A3 * SQRT(A5)) a parse tree depth of 5.

TABLE I
OVERVIEW OF METRICS

Dimension	Metric
Size	s1 # non-empty cells per spreadsheet
	s2 # worksheets per spreadsheet
	s3 # formulas per spreadsheet
	s4 # unique formulas per spreadsheet
	s5 length of formula in characters (measured per cell)
Coupling	c1 % external links per spreadsheet
	c2 # interworksheet connections per spreadsheet
	c3 path depth per formula
	c4 total number of transitive precedents per formula
Use of functions	f1 Number of unique functions per formula
	f2 Parse three depth per formula
	f3 Number of preceding cells per formula

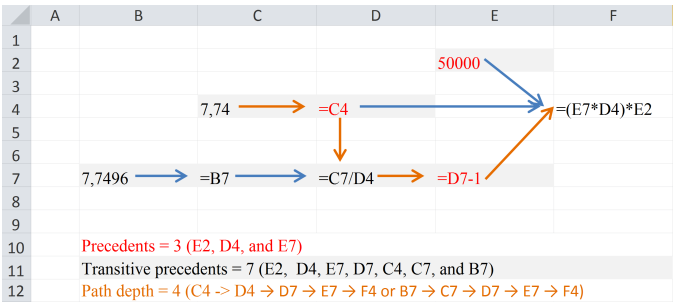

Fig. 5. Precedents, Transitive Precedents, and Path Depth ([6])

Calculating these metrics for the spreadsheets will enable us to answer the second research question:

R2 What are the differences with respect to size, level of coupling and the use of functions between spreadsheets that are perceived as easier to understand and spreadsheets that are perceived as problematic?

To answer both research questions, we calculated for each file the metrics that indicate one of the five smells that were described in Section I and the metrics with respect to size, coupling, and use of functions.

IV. Results

A. Smells

Figure 6 displays the results for the smell metrics. We use the radar chart to visualize all metrics in a single figure. The chart shows the relative score for each smell. The F1F9 scores (red line) are represented as a percentage of the Client scores (blue line) (Client = 100%).

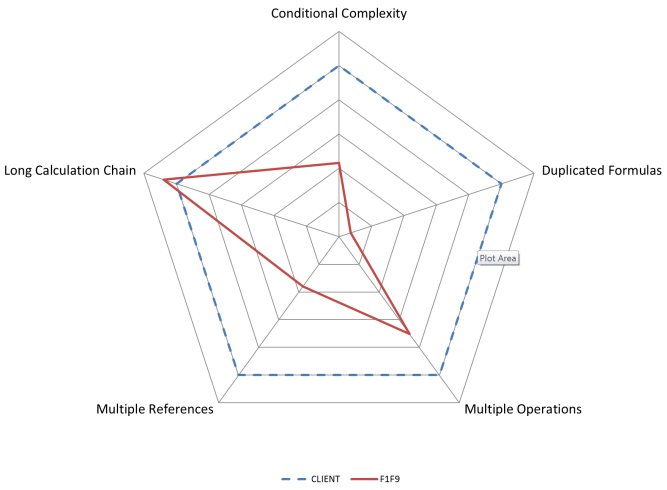

Fig. 6. Relative score of Smells

For the exact figures see Table II. It shows for each smell the median of the number of times the smell occurred in each spreadsheet. The last column gives the score for the F1F9 spreadsheets as a percentage of the score of the Client sheets and were used in Figure 6.

TABLE II
Overview of Smells

Metric	Client	F1F9	F1F9 (%)
Multiple operations	101.0	71.0	70.3%
Multiple references	136.5	49.0	35.9%
Conditional complexity	36.0	15.5	43.1%
Long calculation chain	412.0	444.5	107.8%
Duplicated formulas	296.0	21.5	7.2%

We can see that overall the number of occurrences of the smells decrease. We observe a dramatic decrease of the occurrence of the *Duplicated formula* smell and also see a clear difference for the *Multiple references* and *Conditional complexity* smells. In Section V, we will explain some possible causes for these findings. *Long calculation chain* forms an exception because the number of occurrences of this smell is slightly higher in the F1F9 sheets. We have analyzed this in more detail. Figure 7 shows the boxplot for this smell for both the Client and the F1F9 sheets. The boxplot displays the minimum, 1st quartile, median, 3rd quartile, and maximum value. In the Client data set, there is one spreadsheet with a calculation chain of 9,995 cells, that can be considered as an outlier. We have excluded it from the boxplot because we wanted to visualize the difference in interquartile distance between F1F9 and the Client, which is not affected by the outlier. It shows that although the median for the number of occurrences of the *Long calculation* smell is slightly higher for F1F9 than the Client, the 3rd quartile and maximum value were decreased dramatically. There are fewer spreadsheets that suffer in a high degree from the *Long calculation* smell in the F1F9 dataset.

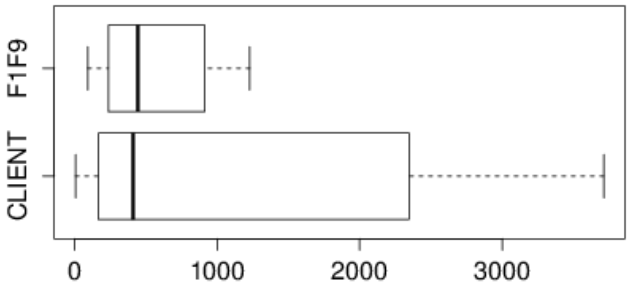

Fig. 7. F1F9 has a slightly higher median, but a much smaller interquartile distance

B. Size, Coupling and Use of Functions

The metrics for the dimension size have been summarized in Figure 8 and the exact figures can be found in Table III. It turns out that almost every size metric has increased for the F1F9 spreadsheets. Only the length of the formulas decreases as compared to the Client sheets. Notable is also the number of formulas. This metric has increased much more than the other size metrics.

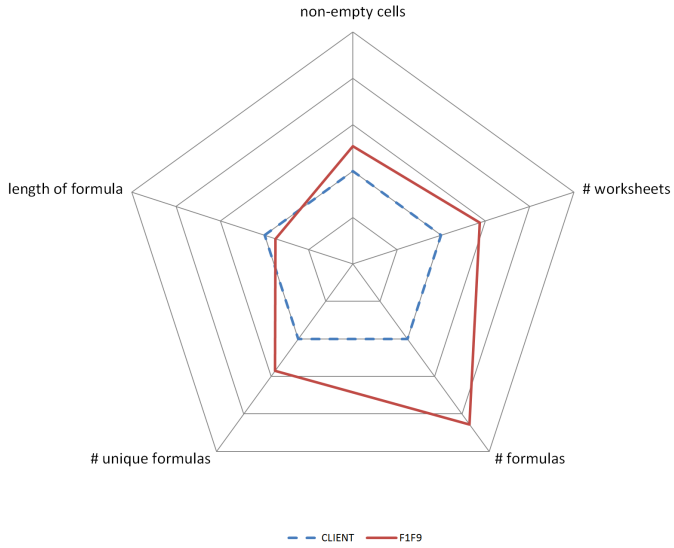

Fig. 8. Relative score on dimensions of size

To measure the level of coupling, we have analyzed both the external (to other spreadsheets) and internal (within the same spreadsheet) links between worksheets, the path depth per formula and the total number of transitive precedents of a formula. The results of this analysis are visualized in Figure 9 and the exact figures summarized in Table IV.

TABLE III
OVERVIEW OF SIZE METRICS

Metric	Client	F1F9	F1F9 (%)
s1 # non-empty cells	170,202.0	215,998.5	126.9%
s2 # worksheets	19.5	28.0	143.6%
s3 # formulas	92,954.5	198,711.0	213.8%
s4 # unique formulas	1,467.0	2,094.0	142.7%
s5 formula length	32.0	28.0	87.5%

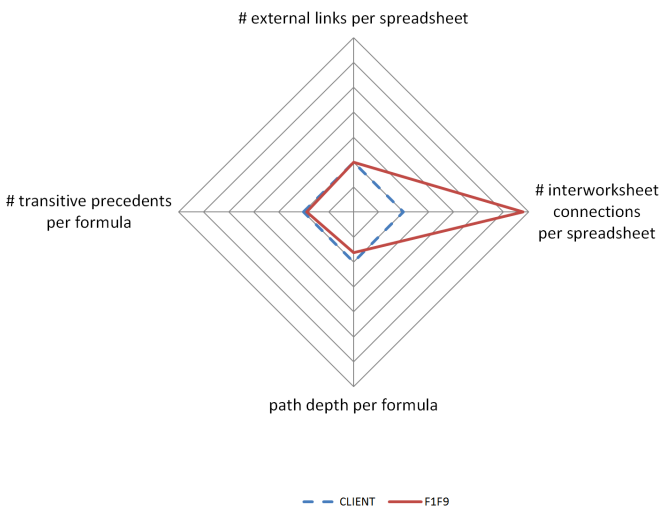

Fig. 9. Relative score on dimensions of coupling

TABLE IV
OVERVIEW OF COUPLING METRICS

Metric	Client	F1F9	F1F9 (%)
c1 # external links	0.0	0.0	100.0%
c2 # interworksheet connections	60.5	205.0	338.8%
c3 Path depth	16.0	13.0	81.3%
c4 # transitive precedents	135.0	127.0	94.1%

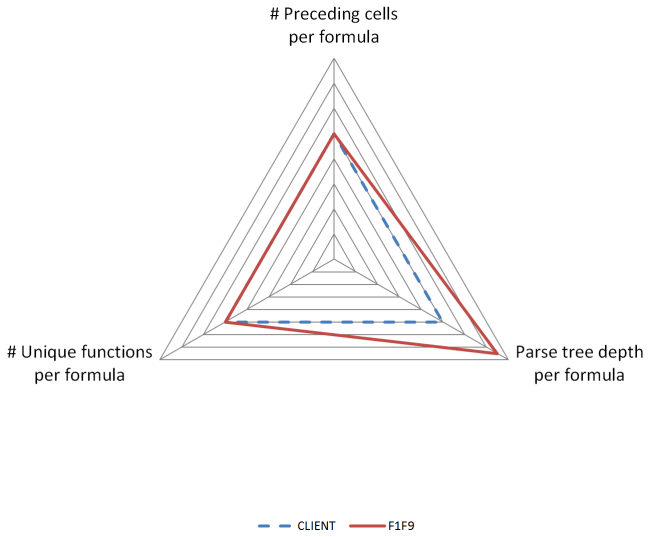

Fig. 10. Relative score on Use of Functions

From the results, it seems that both datasets are almost identical on the degree of coupling, except for the number of interworksheet connections. These are much higher within the F1F9 dataset.

Finally, we have analyzed the use of functions in both datasets, by looking at the number of unique functions used, the parse tree depth, and the number of preceding cells per formula. Figure 10 summarizes the results for these metrics. Exact figures can be found in Table V.

With respect to the use of functions, both datasets are very similar. The only difference is the median for the parse tree depth, which is 3 for F1F9 and 2 for the Client sheets.

C. Significance of the Differences

These results give us an indication of the differences between smelly and non-smelly spreadsheets. However, we do not know yet if these differences are statistically significant. Because we have pairs of smelly and non-smelly spreadsheets we can do a paired group comparison to determine if there is a significant difference. To do so, we used the Wilcoxon Signed-Ranks Test for Paired Samples. However, we can do this test only for the metrics on spreadsheet level (because we have pairs of spreadsheets). If the metric is a characteristic of a formula, a paired comparison is not possible. We do not have pairs of formulas that we can compare. For these metrics we have to test if the distribution of the two datasets is different. We calculated that using the Wilcoxon-Mann-Whitney test. Both tests give us a p-value that can be found in Table VI. We have denoted the metrics on spreadsheet level with an 's' and the metrics that are characteristics of formulas with an 'f'.

If there was a significant difference, we have calculated the effect with the Cliff's Delta d. For both metric s2 (number of Worksheets) and metric c2 (number of interworksheet connections) the effect is large ($d \geq 0.47$). For *Duplicated formulas*, metric c1 (number of external links), and metric c4 (number of transitive precedents) the effect is medium ($0.33 \leq d < 0.47$). For *Conditional complexity*, *Multiple references*, metric s1 (number of non-empty cells), metric s3 (number of formulas) the effect is small ($0.147 \leq d < 0.33$). For the other metrics the effect is negligible.

V. INTERPRETATION

In the previous section, we have described the results of our analysis. We found that the F1F9 sheets were less smelly, but, to our surprise, also bigger. With regards to coupling and the way functions were used, the sets appear more similar. In this section, we will further discuss some of these results. Although we were not able to interview the users of the spreadsheets, we assume that they perceived the F1F9 spreadsheets as easier to maintain and less error-prone. If this was not the case, F1F9 would have gone out of business a long time ago. But what is causing the difference between the F1F9 and the Client spreadsheets? Of course the employees of F1F9 build complex spreadsheet models for a living, but maybe even more importantly they make use of the FAST standard to build these models. A further explanation of some of the concepts

TABLE V
METRICS ON THE USE OF FUNCTIONS

Metric	Client	F1F9	F1F9 (%)
f1 # unique functions	1	1	100.%
f2 parse tree depth	2	3	150%
f3 # preceding cells	3	3	100%

TABLE VI
STATISTICAL ANALYSIS OF CLIENT AND F1F9 DATASETS

Dimension	Metric	Level	p-value	d
Smells	Conditional complexity	s	<0.01	0.158
	Duplicated formulas	s	<0.01	0.412
	Multiple Operations	s	<0.01	0.030
	Multiple References	s	<0.01	0.310
	Long calculation chain	s	<0.05	0.055
Size	s1 # non-empty cells	s	<0.01	0.228
	s2 # worksheets	s	<0.01	0.536
	s3 # formulas	s	<0.01	0.257
	s4 # unique formulas	s	>0.05	-
	s5 formula length	f	<0.01	0.078
Coupling	c1 # external links	s	<0.01	0.400
	c2 # interworksheet conn.	s	<0.01	0.835
	c3 path depth	f	<0.01	0.037
	c4 # transitive precedents	f	<0.01	0.379
Use of Functions	f1 # preceding cells	f	<0.01	0.071
	f2 parse Tree Depth	f	<0.01	0.096
	f3 # unique functions	f	<0.01	0.023

and terminology of the FAST standard will help us to better understand the found results.

The FAST standard divides a spreadsheet in different hierarchical levels to organize their guidelines. The highest level is the workbook itself, followed by the individual worksheets. According to the FAST standard there can be many worksheets within a workbook but each worksheet always fits in one of the following four functional classes:

1) **Foundation**: The basis for the financial model. These worksheets contain all the inputs, timing rules, assumptions and indexation.
2) **Workings**: The engine of the model. All calculations necessary for the final result of the model are made on these sheets.
3) **Presentation**: The output of the model, usually made up of financial statements, charts and summaries. These sheets are used for decision making by the users of the model.
4) **Control**: These sheets assist the builder during the process of creating the model. It normally contains list of pending changes, version control, table of contents, error checking, etc. Furthermore, if scenario planning or sensitivity analysis is used, they are controlled from this sheet.

Figure 11 shows several worksheets of a financial model that was built according to the FAST standard. The sheets InpC, INpS, and Time are examples of a Foundation sheet; Ops, Asset, and Finstats are examples of a Workings sheet.

In the FAST standard a worksheet is divided in several calculation blocks. A calculation block can be considered as an autonomous paragraph on a worksheet and is always

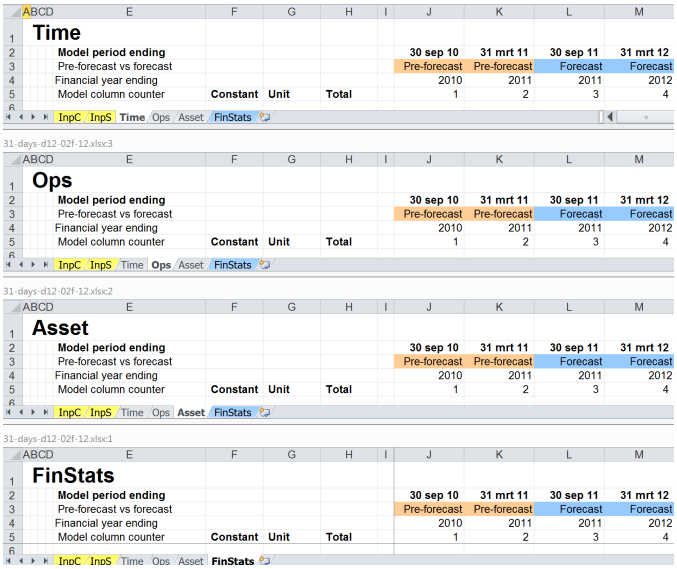

Fig. 11. An example of a consistent column structure that has been maintained across all sheets

responsible for a single calculation. Rows 44 through 51 in Figure 12 show an example of a calculation block. The calculation block itself consists of the ingredients for the calculation (row 47: Domestic charter landings trough 50: International scheduled landings) and the actual calculation (row 51: Total landings).

The lowest level of a financial model is formed by the individual line items (for example row 49 'International charter landings' in Figure 12). It's defined as a unit of information displayed on a row or column, of its own with its own label.

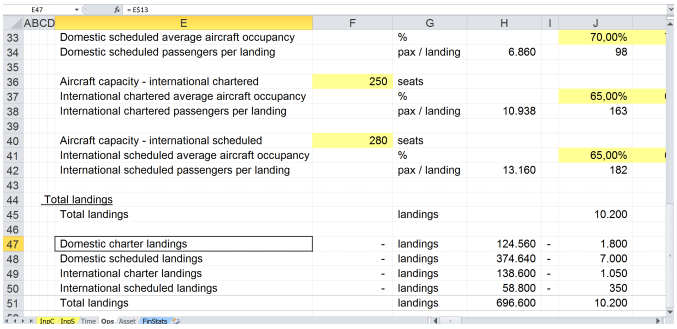

Fig. 12. Screenshot of a FAST model that shows an example of a calculation block

In the remainder of this section, we will analyze the possible effects of the FAST guidelines on the metrics and smells we have calculated for the F1F9 sheets. We will first discuss the smells and at the end of the section focus on the metrics for size, coupling, and the use of functions.

A. Smells

First of all FAST strongly advises to create short and easy to understand formulas. This explains why we find less formulas that suffer from the *Multiple Operations* and *Multiple*

References smells. Furthermore, the standard discourages the use of IF and even prohibit the use of nested IFs. We see that reflected in the lower occurrence of the *Conditional Complexity* Smell.

FAST also dictates that a calculation should only be made once. If the result of the calculation is needed somewhere else in the model, one should link back to the original calculation. Furthermore, formulas should be consistent along the row or column axis, meaning that the formula should be created once and than dragged to the right or the bottom. This is illustrated in Figure 13. It displays the formulas in the R1C1 notation to show that all the formulas on a single row have the same structure. However, they are not identical. The formulas differ in their references to other cells. In this example a single formula was copied to 70 columns. If a spreadsheet is designed in accordance with this rule it means that to understand the sheet we do not have to inspect 71 cells to check if the formulas differ somewhere. We just have to inspect the cell in (in this case) the 10th column. In addition, because the column structure across the different worksheets is consistent this holds true for every worksheet. The guidelines of consistent formulas and defining a calculation only once, explain the dramatic decrease in the occurrence of the Duplicate formula smell.

	9	10	11	12	13	14	15
29		0,6	0,6	0,6	0,6	0,7	0,7
30		= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C
31							
32							
33		0,7	0,7	0,7	0,7	0,7	0,7
34		= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C	= R[-2]C6 * R[-1]C
35							

	I	J	K	L	M	N	O
29		0,6	0,6	0,6	0,6	0,7	0,7
30		= $F28 * J29	= $F28 * K29	= $F28 * L29	= $F28 * M29	= $F28 * N29	= $F28 * O29
31							
32							
33		0,7	0,7	0,7	0,7	0,7	0,7
34		= $F32 * J33	= $F32 * K33	= $F32 * L33	= $F32 * M33	= $F32 * N33	= $F32 * O33
35							

Fig. 13. Example of consistent formulas. The formulas in row 30 and 33 are displayed both in R1C1 (top) and A1 (bottom) notation. Although the formulas differ in normal notation, the R1C1 notation shows that they are actually identical.

The number of smell occurrences decreases for four of the five smells. However, the median for the *Long Calculation Chain* in the F1F9 dataset is slightly higher than in the Client dataset. This is not unexpected, trying to minimize the *Multiple References* and *Multiple Operations* smells (ie breaking long formulas in shorter parts), inevitable leads to longer calculation chains.

B. Size, Coupling, and Use of Functions

Based on the size metrics, we can conclude that the F1F9 models grew in size. Within the FAST standard there are several guidelines that could explain this increase.

- *Separate worksheets per functional class*: the standard dictates a strict separation between input (Foundation), calculation (Workings), output (Presentation), and control, which causes more worksheets per spreadsheet.
- *Maintain consistent column structure across all sheets*: Most financial models are time-related. For example, to calculate the business case for a major investment it is necessary to predict the future cash flows over the life span of the investment (which could easily be 30 years). The FAST standard prescribes to model the time-dimension along the columns of a worksheet. Because of this guideline, these time series are repeated on every worksheet even if that means that on some worksheets these columns are unused. Figure 11 shows an example of such a consistent column structure across different sheets.
- *Construct all calculations in a separate calculation block*: A calculation block (see Figure 12 for an example) consists of all the ingredients (inputs) that are necessary for a calculation. Ingredients can also be used in other calculations. The standard dictates that in such a case the ingredients are repeated (by direct linking) to form a new calculation block. This increases the number of non-empty cells on a worksheet.

With respect to coupling, the number of interworksheet connections was the only metric in the F1F9 sheets that differed from the Client sheets. This could be caused by the way calculation blocks are constructed according to the FAST standard. As we already saw, a calculation block consists of the necessary ingredients and the calculation itself. The ingredients are often input values that are coming from another (Foundation) worksheet and thus creating an interworksheet connection. If the same ingredient is necessary for another calculation, it will be repeated. However, the FAST standard forbids a series of linked links, so called daisy chains. We illustrate this concept with a small example in which for a certain calculation a start date is needed. The start date is coming from the sheet 'InpC' (which is a Foundation sheet) and is located in cell F11. The start date is an ingredient for a calculation block and it is put in cell F12 on the sheet 'Time'. The formula for this cell becomes:

F12: = InpC!F$11

This same start date is also needed as ingredient in a second calculation block (in for example cell F21) on the same sheet. In general users tend to solve this with:

F21: = F12

However, this creates a daisy chain. The value from cell F11 on the sheet 'InpC' is retrieved via cell F12 on the sheet 'Time'. To prevent daisy chaining the formula should be:

F21: = InpC!F$11

Applying this guideline will create an additional interworksheet connections every time an input value is re-used in a different calculation block. It explains why the number of interworksheet connections in the F1F9 sheets is higher than in the Client sheets.

The use of functions is similar in both datasets. In earlier work [6] we saw that in the Enron Corpus the majority of formulas only make a direct reference to a few other cells, are hardly nested and within the formula only one function (not being an operator) is used. Despite the fact that both the Client and the F1F9 dataset consists of complex financial models, we see the same kind of metrics with respect to the complexity of the formulas. Users tend to create simple formulas. Complex and large formulas do exists but are an exception.

VI. Discussion

In the previous sections, we have analyzed the occurrences of smells and metrics with respect to size, coupling and use of functions in both the Client and the F1F9 spreadsheets. In this section, we discuss some topics that could affect the applicability and suitability of the approach used and the results found.

A. Threats to Validity

The dataset we received from F1F9 gave us a unique opportunity to work with complex, real-life, industrial spreadsheets to investigate what characterizes a smelly spreadsheet. Unfortunately, this real-life dataset comes with the price of reduced repeatability. We are strong believers of open data, but because the spreadsheets contain confidential information, we were only allowed to analyze them automatically and we are not able to share them.

A threat to the external validity of our analysis is the representativeness of the provided dataset. Half of the spreadsheets were created by a single company. However, the Client spreadsheets are from different clients. More important is the fact that all spreadsheets are financial models. Consequently the findings of our analysis can only be applied to spreadsheets within the specific domain of financial modeling.

B. Pivot Tables, Charts and VBA code

In our analysis, we limited ourselves to the described smells and metrics. However the improvements that were made by F1F9 could also affect the use of more elaborate structures like Pivot tables, charts and VBA code. In future research, we plan to specifically analyze these constructs.

C. Calculation Chains

All the smells that we have analyzed are calculated on the formula level. The same is true for the following metrics that we used to analyze the size, level of coupling and use of functions:

- s5 length of formula in characters
- c3 path depth per formula
- c4 total number of transitive precedents per formula
- f1 number of unique functions per formula
- f2 parse three depth per formula
- f3 number of preceding cells per formula

We took the single formula as the object of analysis. We considered it as the equivalent of a line of code. However, in a spreadsheet it is always possible to take a formula and split it over more than one cell. What we consider a line of code is actually an arbitrary decision of the user. To see and understand the complete code for a certain calculation, you need to look at the complete calculation chain. In future research, we plan to extend the analysis of smells and metrics to the level of the calculation chain.

VII. Related Work

Our work builds upon the work of Hermans *et. al.* [1], in which the concept of spreadsheet smells at the formula level was introduced. However, for their evaluation they used a small dataset of which it was not known in advance whether it contained smelly spreadsheets. In addition to that paper, Hermans also worked on other types of spreadsheet smells, for example focusing on detecting smells between worksheets, rather than within [8]. Other work on spreadsheet smells was done by Cunha *et al.* who aim at detecting smells in values, such as typographical errors and values not following the normal distribution [5].

A second category of related work aims at defining spreadsheet metrics. Bregar developed a catalog of spreadsheet metrics based on software metrics [9]. He however did not evaluate his metrics in practice. Hodnigg and Mittermeir [10] also proposed several spreadsheet metrics of which some are similar to Bregar's. Their metrics are divided into three categories: general metrics, such as the number of formulas and the number of distinct formulas; formula complexity metrics, such as the number of references per formula, and the length of the longest calculation chain; and finally metrics, such as the presence of scripts in, e.g., Visual Basic for Applications (VBA), user defined functions and external sources. Hole *et al.*[11] propose an interesting approach to analyze spreadsheets in terms of basic spreadsheet metrics, such as the number of functions used, the presence of charts and the complexity of program code constructs with the specific aim of predicting the level of the spreadsheet creator.

A research direction related to smell detection is spreadsheet refactoring, which also has been addressed by researchers in recent years. The first to present a semi-automated approach were Badame and Dig [12], whose refactorings unfortunately were not directly based on smells. A generalization of this idea was presented by Hermans and Dig [13].

VIII. Concluding Remarks

This paper describes the analysis of a new spreadsheet dataset. This set consists of 54 pairs of spreadsheets, which both implement the same functionality, but are either smelly and hard to maintain (client) or well-structured (F1F9).

For each spreadsheet, we calculated the metrics that indicate formula smells and extended this analysis with additional metrics for size, coupling and the use of functions. For each metric we determined whether there was a significant difference between the client and the F1F9 sheets. If a difference was found, we calculated the effect size.

Based on this analysis we answered our two research questions:

R1 Do spreadsheets that are perceived as easier to understand, contain fewer smelly cells than spreadsheets that are perceived as problematic?

R2 What are the differences with respect to size, level of coupling and the use of functions between spreadsheets that are perceived as easier to understand and spreadsheets that are perceived as problematic?

Our analysis reveals two interesting points. Firstly, the F1F9 spreadsheets indeed suffer from smells to a much lower extent than the client sheets. We observed for example that the F1F9 sheets contain fewer duplicated formulas and that formulas have fewer references to other cells. Secondly, size and coupling metrics, obvious candidates to measure spreadsheet complexity, do not succeed in differentiating between the both parts of the datasets.

Our current analysis gives rise to ample directions for future work. In this paper we did a pairwise comparison on spreadsheet level. Because F1F9 rebuilt the models from scratch, it was not possible to do this on formula level. In future research, we are planning to analyze the effect of refactoring spreadsheet formulas in existing models. In such a case pairwise comparison on formula level is possible.

A spreadsheet that contains fewer smells should be easier to understand and maintain. In this study, we saw indeed that the spreadsheets improved by F1F9 contain fewer smells. However, it was not possible to interview the users of these spreadsheets to obtain their opinion about the understandability and maintainability of the spreadsheets. We believe this is a promising avenue for future research. We will perform a case study with users to test if they are able to understand and maintain a refactored spreadsheet with less effort.

REFERENCES

[1] F. Hermans, M. Pinzger, and A. Deursen, "Detecting code smells in spreadsheet formulas," *Proceedings of the International Conference on Software Maintenance (ICSM)*, 2012.

[2] M. Fowler, *Refactoring : improving the design of existing code*. Reading, MA: Addison-Wesley, 1999.

[3] FAST Standard Organisation. (2015) The fast standard - practical, structured design rules for financial modelling, version fast02a. [Online]. Available: http://www.fast-standard.org/

[4] F. Hermans, "Analyzing and visualizing spreadsheets," Ph.D. dissertation, PhD thesis, Software Engineering Research Group, Delft University of Technology, Netherlands, 2012.

[5] J. Cunha, J. P. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *Computational Science and Its Applications–ICCSA 2012*. Springer, 2012, pp. 202–216.

[6] B. Jansen, "Enron versus euses: A comparison of two spreadsheet corpora," in *Proceedings of the 2nd Workshop on Software Engineering Methods in Spreadsheets, Florence, Italy*, 2015.

[7] J. Sajaniemi, "Modeling spreadsheet audit: A rigorous approach to automatic visualization," *Journal of Visual Languages & Computing*, vol. 11, no. 1, pp. 49–82, 2000.

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

[9] A. Bregar, "Complexity metrics for spreadsheet models," in *Proc. of EuSpRIG '04*, 2004, p. 9.

[10] K. Hodnigg and R. Mittermeir, "Metrics-based spreadsheet visualization: Support for focused maintenance," in *Proc. of EuSpRIG '08*, 2008, p. 16.

[11] S. Hole, D. McPhee, and A. Lohfink, "Mining spreadsheet complexity data to classify end user developers." in *DMIN*, 2009, pp. 573–579.

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

[13] F. Hermans and D. Dig, "Bumblebee: a refactoring environment for spreadsheet formulas," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 747–750.