2012 28th IEEE International Conference on Software Maintenance (ICSM)

Detecting Code Smells in Spreadsheet Formulas

Felienne Hermans, Martin Pinzger and Arie van Deursen
Delft University of Technology
Delft, the Netherlands
{*f.f.j.hermans, m.pinzger, arie.vandeursen*}*@tudelft.nl*

***Abstract*—Spreadsheets are used extensively in business processes around the world and just like software, spreadsheets are changed throughout their lifetime causing maintainability issues. This paper adapts known *code smells* to spreadsheet formulas. To that end we present a list of metrics by which we can detect *smelly* formulas and a visualization technique to highlight these formulas in spreadsheets. We implemented the metrics and visualization technique in a prototype tool to evaluate our approach in two ways. Firstly, we analyze the EUSES spreadsheet corpus, to study the occurrence of the formula smells. Secondly, we analyze ten real life spreadsheets, and interview the spreadsheet owners about the identified smells. The results of these evaluations indicate that formula smells are common and that they can reveal real errors and weaknesses in spreadsheet formulas.**

***Keywords*-spreadsheets; code smells; refactoring;**

I. Introduction

The use of spreadsheets is very common in industry, Winston [1] estimates that 90% of all analysts in industry perform calculations in spreadsheets. Spreadsheet developers are in fact end-user programmers that are usually not formally trained software engineers. There are many of those end-user programmers, more than there are traditional programmers, and the artifacts they create can be just as important to an organization as regular software. Technically, spreadsheets also have similarities to software. One could view spreadsheet formulas as little pieces of source code, since both consist of constants, variables, conditional statements and references to other parts of the software. It therefore seems logical to research what principles from software engineering are also applicable to spreadsheets.

In previous work [2] we have defined code smells between worksheets, such as high coupling between worksheets and *middle men* worksheets. The evaluation of those smells showed that they can indeed reveal weak spots in a spreadsheet's design. In this paper we follow that line of thought, but focus our attention on smells within spreadsheet formulas. To that end we present an set of *formula smells*, based on Fowler's code smells. We subsequently define metrics for each of the formula smells, to enable the automatic detection of the smells. We then describe a method to detect these formula smells. Our detection approach uses thresholds to divide the severeness of each formula smell into low, moderate, and high. The thresholds are based on the analysis of 4,233 spreadsheets from the EUSES corpus [3]. Thereon we address the issue of communicating identified smells to spreadsheet users. We choose to do this within the spreadsheet itself, with a spreadsheet *risk map*, a colored overlay on the spreadsheet, indicating risk in the spreadsheet formulas. Finally we evaluate the catalog of smells in two ways, with a quantitative and qualitative evaluation. We perform a quantitative evaluation on the EUSES spreadsheet corpus. The qualitative analysis was performed with ten real life spreadsheets and their developers from industry. With both studies we aim to answer the three research questions: R_1 What formula smells are most common, and why? R_2 To what extent do formula smells expose threats to spreadsheet quality? R_3 To what extent are risk maps an appropriate way to visualize formula smells?

The results of these evaluations show that formula smells can indeed reveal weaknesses, and even find real mistakes in a spreadsheet. The risk maps, although not yet perfect, are a good aid in helping to locate and understand formula smells.

II. Formula Smells

We define a number of *formula smells*, based on the existing work in the field of source code smells, initiated by Fowler [4]. Smells in source code indicate suspicious parts, that the developer might want to refactor to improve readability and minimize the chance of future errors. Formula smells are inspired by source code smells: they indicate formulas that are suspicious; not easy to read or error-prone. In the following we present our set of formula smells plus ways to refactor them.

A. Multiple Operations

One of the most well-known code smells is the Long Method. Inspired by this code smell, we define the formula smell *Multiple Operations*. Analogous to a long method, a formula with many different operations will likely be harder to understand than a shorter one. Especially since in most spreadsheet programs, there is limited space to view a formula, causing long formulas to be cut off.

A corresponding refactoring is the division of the Multiple Operations over multiple cells in a spreadsheet. For instance, instead of putting *SUM(A1:A6)*(B1+8)/100* in one cell, this could be split into two cells, one for the SUM, and one for the division, that are subsequently multiplied.

978-1-4673-2312-3/12/$31.00

B. Multiple References

Another code smell we use as a basis is the Many parameters code smell. A method that uses many input values might be split into multiple methods to improve readability. The formula equivalent of this smell occurs when a formula references many different other cells, like *SUM(A1:A5; B7;C18;C19;F19)*. In this case the essence of the formula is clear; some cells are summed. However locating the different cells that are contained in the sum can be challenging.

In this case there are several options to refactor. Firstly, like in the case of a formula with many operations, we could split the formula into several smaller formulas, each performing one step of the calculation. A second option is the relocation of the cells in the spreadsheet. One solution is to place the values in B7;C18;C19;F19 in A6 until A10, and to rewrite the formula as *SUM(A1:A10)*.

C. Conditional Complexity

Fowler states that the nesting of many conditional operations should be considered a threat to code readability, since conditionals are hard to read. Since spreadsheet systems also allow for the use of conditionals, spreadsheet formulas are at risk of this treat too. We hence consider a formula with many conditional operations as smelly, like the formula *IF(A3=1,IF(A4=1,IF(A5<34700,50)),0)*, because of the many conditional branches, the formula is hard to read.

To reduce conditional complexity of a formula, again it could be split into multiple steps, by putting each branch of the if in a separate cell, turning our example formula into *IF(A3=1,B1,B2)*, where cell B1 contains *IF(A4=1, IF(A5<34700, 50))* and B2 contains 0. B1 could again be refactored in this fashion. By separating the ifs, it is easier to understand what happens in each case. A different option is to combine multiple if formulas into one *SUMIF* or *COUNTIF* formula, by putting the branches of the if in separate cells.

D. Long Calculation Chain

Spreadsheet formulas can refer to each other, hence creating chains of calculation. To understand the meaning of such a formula, a spreadsheet user has to trace along multiple steps to find the origin of the data. Nardi and Miller [5] described that spreadsheet users find tracing a long calculation chain a tedious task.

To lower the number of steps of a calculation chain, steps of the chain could be merged into one cell. Note that there is a trade off between this metric and Multiple Operations and Multiple References. When they are lowered, this metric will be increased, and vice versa. Such trade offs occur in source code smells too.

E. Duplicated Formulas

Finally there is the *duplication* code smell, that indicates that similar snippets of code are used througout a class. This is a concept common in spreadsheets too, where some formulas are partly the same as others. Consider, for example, a worksheet that contains a cell with formula *SUM(A1:A6)+10%* and a second formula *SUM(A1:A6)+20%*. This formula exhibits *duplication*; the part *SUM(A1:A6)* is contained in more than one formula. Duplication is suspicious for two reasons. Firstly it poses a threat to maintainability, since when the duplicated part of the formula is adapted, this adaptation has to be performed at multiple places. This could be forgotten by the spreadsheet user, or a mistake could be made in some of the cases. Secondly, there is an impact on readability, when long parts of the formula are duplicated, it is hard to see how they differ from each other.

Duplicated formulas can be refactored by putting the shared subtrees in a separate formula and replacing the subtree with a reference to that formula.

III. Formula Metrics

To identify smell occurrences automatically, we make use of metrics, an approach common in the automatic detection of code smells [6]. We follow our approach outlined in [2] defining a metric to detect each of the formula smells in spreadsheets. This method entails the definition of a metric for each of the smells to indicate the presence of that particular smell.

1) Multiple Operations: We measure the length of a formula in terms of the total number of functions that the formula contains.

2) Multiple References: This metric is counted in terms of the number of ranges a formula is referring to.

3) Conditional Complexity: Conditional complexity is measured in terms of the number of conditionals contained by a formula.

4) Long Calculation Chain: This metric is defined as the length of the longest path of cells that need to be dereferenced when computing the value of the formula.

5) Duplicated Formula: For the localization of this smell more information about spreadsheet formulas is needed. Consider the spreadsheet in Figure 1. All the formulas in column E calculate the minimum of the four cells left to it, followed by the addition of a certain percentage. We argue that in this case, duplication should be detected. However, looking at the formulas, they do not look similar. We therefore will use the *relative R1C1 notation* when detecting duplication.

In the relative R1C1 notation, references to other cells are expressed relative to the cell containing the formula. *MIN(A2:D2)* in cell E2 is written as *MIN(RC[-4]:RC[-1])* in relative R1C1 notation. With this notation, all formulas in Figure 1 contain the subtree *MIN(RC[-4]:RC[-1])*, with

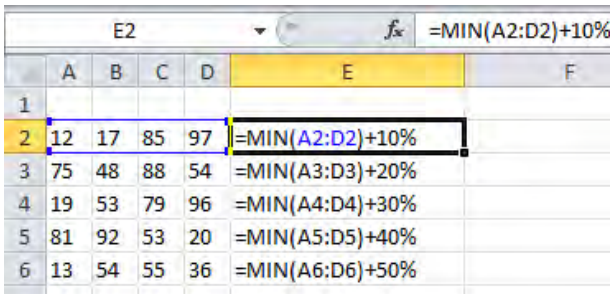


Figure 1. Formulas containing similar subtrees

different percentages. With this, we will measure the duplicate code smell as 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. We exclude the entire tree as subtree, since having the same R1C1 formula in an entire row or column is the usual way of defining a formula in spreadsheets.

IV. Determining Smell Thresholds

In order to use the metrics as smell indicators, we determine thresholds for each of the metrics. We do this by analyzing a large body of spreadsheets and based on the values for the metrics we find in that large body of spreadsheets, we set the thresholds for the metrics that indicate the smell.

The body of spreadsheets we use is the EUSES Spreadsheet Corpus [3]. This corpus consists of more than 4,223 real life spreadsheets, from all sorts of 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, for instance [7].

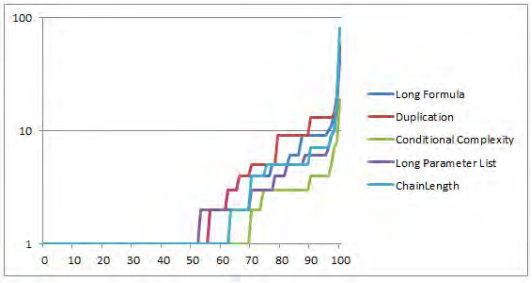


Figure 2. A quantile plot (% of formulas) for the five metrics for formulas in the EUSES corpus, with a logarithmic scale for the y axis

The total of 4,223 spreadsheets together contain 15,015 worksheets and 426,986 formulas. A spreadsheet however often contains many rows with formulas that are equal in the relative R1C1 notation, which we call *unique* formulas. We collect the metrics for all unique formulas, of which there are 55,736 in the EUSES corpus.

Figure 2 shows on overview of the metric values for all unique formulas in the EUSES corpus. As can been seen in this figure, the metrics all follow a power law like distribution, having most of their variability on the tail. We therefore propose to select the values at 70, 80 and 90% of the metric values, that will correspond to risk levels low, moderate and high. These are percentages that are also common in the analysis of source code smells [8]. Table I shows the thresholds that follow from the given selection for the five formula smells.

Smell	70%	80%	90%
Multiple Operations	4	5	9
Multiple References	3	4	6
Conditional Complexity	2	3	4
Message Chain	4	5	7
Duplication	6	9	13

Table I
THRESHOLDS FOR THE METRICS THAT INDICATE THE FORMULA SMELLS

V. Risk Maps

Having established a method to detect the formula smells, in this section, we investigate a way to communicate the located smells to spreadsheet users. We have chosen to show the formula smells inside the spreadsheet itself. We have seen in previous work that, when reasoning about formulas, spreadsheet users like to see the context of the formula [9]. We therefore use a colored overlay over the spreadsheet that indicates the formula smells, inspired by other spreadsheet tools like UCheck [10], [11], [12], combined with pop-ups showing what smell is detected, similar to those in UFix [13]. We call this the spreadsheet *risk map*.

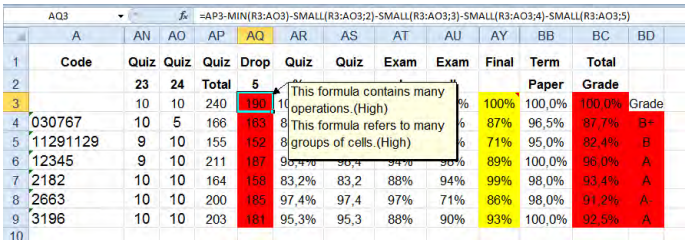


Figure 3. A spreadsheet with its risk map

We attach a comment to each colored cell, so when a spreadsheet user clicks a cell, an explanation about the located smell will pop up. The three different risk levels are marked with different colors; yellow for low; orange for moderate and red for high.

VI. Implementation

The generation of the risk maps is incorporated into the existing spreadsheet analysis system Breviz [14]. Breviz

is implemented in C# 4.0 using Visual Studio 2010. It utilizes the Gembox component to read Excel files. [1] Breviz reads an Excel file and calculates the metrics described above and subsequently generates the spreadsheet risk map. Furthermore the metric values are stored in a SQL Server 2008 database to perform the threshold detection.

VII. Evaluation

In order to evaluate the formula smells, metrics, thresholds, and risk map visualization, we perform two separate evaluations. In the first evaluation we turn our attention back to the Euses Spreadsheet corpus. We analyze all spreadsheets in the corpus and investigate the occurrence of the five spreadsheet smells. With this analysis, we seek to find a preliminary answer to research question R_1: what formula smells are common, and why.

For the second evaluation, aimed at a deeper understanding of R_1, plus answers to R_2 and R_3, we ask ten professional spreadsheet developers for access to their real life spreadsheets. We let our tool Breviz identify possible formula smells and show the participants the generated risk map. We thereon ask the spreadsheet users to reflect on the identified spreadsheet smells, in a semi-structured interview.

The following sections describe the two evaluations in more detail.

VIII. Smell Occurrences in the EUSES Corpus

A. Goal

The objective of the first study is to understand how common the five formula smells are, given the thresholds we have selected. While the thresholds were chosen such as percentages of formulas containing a smell, here we are interested in the distribution of *smelly* formulas across spreadsheets.

B. Setup

We use the Breviz tool to analyze the spreadsheets in the EUSES corpus for the presence of formula smells and their severity. Per spreadsheet the tool outputs the metric values for each of the five formula smells. We use this data to analyze the distribution of the formula smells over the three metric categories; above the 70%, 80% and 90% thresholds.This gives a first idea of the distribution of the formula smells over the spreadsheets.

C. Results

Table II shows the results of the first evaluation. As shown in this Table, Multiple Operations and Multiple References are most common in the EUSES Corpus. This is consistent with previous spreadsheet experiments, where it has been shown that spreadsheet are often adapted by their users [9]. In that process, spreadsheet formulas tend to get longer and more complex. As opposed to software development, where code is sometimes rewritten to improve readability or maintainability, the answers of the ten subjects of the second evaluation (see below) have taught us that this is not common practice among spreadsheet creators. Hence, when a formula has become long and complex, it is likely to remain that way.

Smell	$> 70\%$	$> 80\%$	$> 90\%$
Multiple References	23.8%	18.4%	6.3%
Multiple Operations	21.6%	17.1%	6.3%
Duplication	10.8%	7.1%	3.7%
Long Calculation Chain	9.0%	7.9%	3.3%
Conditional Complexity	4.4%	3.0%	1.1%
Any of the above smells	42.7%	31.4%	19.7%

Table II
Percentage of spreadsheets in the EUSES corpus that suffer from at least one of the five spreadsheet smells in EUSES corpus, for the three thresholds

Third and fourth come the Duplicated Formula and Long Calculation Chain. These two smells share the property that they are not immediately visible to the spreadsheet user. In most modern spreadsheet systems, when a cell is clicked, the formula it contains is shown. However in the case of Duplicated Formula and Long Calculation Chain, the formula does not reveal where the calculation chain of the formula ends, and with what formulas a cell shares subtrees. So it is interesting to see that around 10 percent of spreadsheets suffer from a smell that is not visible to a spreadsheet user that is not explicitly looking for it.

Conditional Complexity is the least common smell. This is surprising, since we have seen before that conditional operations are quite common in spreadsheets. We therefore dived into this phenomenon deeper. We found that of the total of 426,986 formulas in the corpus, 5,976 contain at least one conditional operation, this is 1.4% of all formulas. These formulas are divided over 380 spreadsheets, which amounts to 22.2% of the spreadsheets with formulas. We can hence state that the use of conditional operations is relatively common.

However, only 92 of the spreadsheets (5.3%) contain formulas with more than 2 conditional operations in one formula, adding up to only 695 formulas (0.2%). Evidently, the nesting of conditional formulas is not very common. We will hence devote attention to this fact in the second evaluation.

Regarding the thresholds, given our current choices a substantial percentage of the corpus suffers from spreadsheet smells, especially in the low category. In the second case study we will continue to investigate the thresholds, by observing how spreadsheet users from industry feel about formulas that are marked as smelly by these thresholds.

[1] http://www.gemboxsoftware.com/spreadsheet/overview

IX. Formula smells in an industrial case study

A. Goal

The aim of our second evaluation is investigating which of the formula smells actually poses a threat to spreadsheets (R_2), and whether risk maps help spreadsheet users find and understand formula smells (R_3). To determine this, we have surveyed 10 spreadsheet users and interviewed them about a spreadsheet that they often work with and that they found was large and complicated.

B. Setup

For the second evaluation we interviewed 10 professional spreadsheet users from the financial domain. We conducted our evaluation at Robeco, a Dutch asset management company with approximately 1600 employees worldwide, and over 130 billion Euro worth of assets under management.

We invited 27 participants of a previous study performed at Robeco [9] to participate in this evaluation, where they were asked to provide us with a spreadsheet that they worked with regularly. We explained to participants that we were going to analyze the quality of their spreadsheet, and discuss it with them. We provided subjects with a list of the five formula smells and a short explanation of each smell, so they could study this before the experiment. During each of the 10 case studies, the procedure was as follows. First we asked the subject to explain the spreadsheet to us. We then analyzed the spreadsheet and generated the spreadsheet risk map, which we showed to users in Excel 2010. We subsequently let subjects inspect the risk map, and asked them in a semi-structured interview setup, for each of the located formula smells 1) whether they understood the smell that was identified and 2) whether they thought this smell posed a threat to spreadsheet understandability, and if yes, why, and how severe the threat was, according to them. We finally asked them how the risk map and the pop-up helped them in understanding the formula smells.

C. Results

Table III shows an overview of the spreadsheets used in the case study in terms of the numbers of worksheets, cells, formulas, unique formulas and file size. The five final columns indicate the number of unique formulas in the spreadsheets suffering from the given smell, Multiple Operations (MO), Duplicate Formula (DF), Multiple References (MR), Conditional Complexity (CoC) and Long Calculation Chain (CaC). As can be seen from this table, formula smells occur frequently in the ten spreadsheets. The following describes the result of the case studies in detail.

1) General observations: When confronted with the most smelly formula cells, participants often needed time to explain the formula. In all cases the participants expressed statements like "what was this formula again?" or "let me just have a quick look". A commonality among the ten cases in the study is the fact that all participants immediately recognized the impact a smelly, hence complex formula could have on spreadsheet formulas understandability. When we discussed the most complex formulas in the spreadsheet with the participants, they indicated that it was going to be very hard to understand or adapt this formula for someone else than the spreadsheet's creator.

Most participants (8) had never considered the situation where another spreadsheet user had to understand their formulas that much. A previous study by Hermans *et al.* confirms the importance of such *transfer scenarios* [9]. What we found in our present study, however, is that participants did not realize prior to this study that keeping formulas short and simple would help future users of the spreadsheet understand it better and faster. A side effect of our study was increased awareness with our participants that they should take maintainability into account when building and adapting spreadsheets.

We found that the three thresholds and the corresponding coloring help subjects estimate severeness of the detected smell. One of the subjects compared this to the triangle function in Excel. This function marks potential errors, like calculations over empty cells with a small green triangle at the bottom of a cell. He stated:"That function is annoying, since many cells get colored. Because you have different shades, I can start inspecting the red ones, and ignore the yellow ones for now".

Regarding the values of the thresholds, we discussed each colored cell with the spreadsheet owner, systematically going through the worksheets. In all but one case the subjects agreed with the classification of the formulas. Only spreadsheet user S_3 stated that he felt that the system was too strict. His spreadsheet contained 3 cells with five different references and four operations. These cells were hence marked as having both the Multiple Operations and the Multiple References smell, while the user still found this acceptable. In the other formulas in his spreadsheet where these smells were located, he did found they were smelly, since the metric values for those formulas where higher than respectively 5 and 4. So from the perspective of this user the thresholds should be higher, however as stated above, he was the only one; the other nine subjects stated all marked formulas were indeed smelly.

2) Multiple Operations: **Findings** Multiple Operations were found in all ten spreadsheets, making them the number one common smell. In all cases we found that the subjects said that keeping the formulas short makes them easier to read and understand. Two of the subjects believed that formulas with many operations are often a cause of errors, saying "the chance of errors in such a long formula is so much bigger; when I find errors, it is almost always in long formulas". When asked for the reason that Multiple Operations where created, all subjects stated that this was an evolutionary process. Multiple Operations are hardly ever

ID	Spreadsheet Description	#Wrks.	#Cells	#Form.	#Un. F	Size(Kb)	MO	MR	DF	CaC	CoC
1	Calculate dividend	5	13,697	6,012	53	183	14	6	3	7	-
2	Investment strategies	5	21,600	3,031	98	605	9	8	4	3	-
3	Model energy companies	14	82,000	14,742	531	826	58	31	11	-	-
4	Valuation of swaps	8	31,485	5,034	67	1,690	17	-	4	-	21
5	P&L overview of all traders	10	17,263	9,152	142	4,261	26	23	17	7	-
6	Risk overview for different sectors	9	9,190	148	12	332	4	3	2	-	-
7	Comparing calculation models	14	24,580	3,388	39	348	17	14	23	-	-
8	Planning of trades	1	2,329	1,683	64	76	15	33	-	4	-
9	Report interest and liquidity risk	25	59,116	17,930	117	1,693	32	18	-	9	-
10	General ledger data for analysis	11	11,906	3,049	56	1,732	10	23	7	-	4

Table III
CHARACTERISTICS AND NUMBER OF SMELLS ABOVE THE 70% THRESHOLDS OF SPREADSHEETS USED IN THE CASE STUDY.

created at once. They are the result of the repeated adaptation of a formula, adding operations as the spreadsheet changes. As one of the subjects stated "Usually it starts with just a sum, but than you want to round it, add something and before you know it, the formula is two lines long". The two subjects above — the ones who had realized the error risk of Multiple Operations— did try to minimize formula length. However, sometimes, for instance, when a deadline was approaching, Multiple Operations were introduced anyway. There was no common practice among the spreadsheet users to restructure the spreadsheet after such a deadline. One of these two subjects mentioned "when it works, it works. No one really cares how it is done".

We found that the risk map helped in the case of Multiple Operations. In the case where this smell was located, the smelly cells were clearly marked with a color (yellow, orange or red). Hence, the reason why the smell was detected was immediately clear; many subjects stated something like "I understand why this formula is selected by your system, it is quite long."

Conclusions Spreadsheet users feel that Multiple Operations are more error prone than shorter formulas. Since Multiple Operations are harder to read, it is more difficult for users to spot an error, so formulas with multiple operations will less likely be corrected when they are wrong. Multiple Operations are often the result of changes to the spreadsheets, and the refactoring of complex formulas is not something that spreadsheet users do.

3) Multiple References: **Findings** Experiences with Multiple References were similar to those with Multiple Operations; when confronted with the smelly cells, it took the nine subjects a considerable amount of time, in the longest case even ten minutes, to explain the formula. This made them realize that it would be very hard for others to understand and adapt the formula, especially since locating the references can be a challenge. Excel supports users in locating the references by coloring the referenced cells. However, if there are many references and colors users find this feature to be more annoying than helpful as confirmed by nine of our participants. One of the subjects stated, when looking at a formula that referred to no less than 17 ranges "this formula is a real puzzle".

In this case, as opposed to the Multiple Operations smell, some participants did not immediately understand how to adapt this formula to make it less smelly. When asked, one of the participants even stated "but I need all that input to make the calculation". Splitting the formula into different steps seemed more difficult than with the Multiple Operations. In that case the formulas consist of different operations, and the splitting would consist of separating the operations. In this case however, we encountered formulas like SUM(A1:A5;B6;D7;E12), of which it was not immediately clear to the spreadsheet users how to improve it. It can be split into multiple steps, but what steps are logical is not so easy to determine for the user. We asked the nine participants to describe how they were going to split the formula, and only one was able to formulate a strategy. The other hesitated, one of them stated "I don't know where I should start, because I don't remember how I made this formula". As an alternative, cells could be moved, such that this formula will refer to one range. This made the participants hesitate even more. They clearly felt that moving formulas around was a tricky operation, since the effect of this move on other formulas is not clear. One of the subjects that tried to lower the references said "if I move this, what will happen to the other formulas? I would like to preview that".

For this smell again, the risk maps are a good way of conveying this smell. Formulas with many references were colored red, orange or yellow; and hence attracted the attention of the spreadsheet users. Clicking the formula revealed easily that the formula had too many references.

Conclusions Subjects found that formulas with many references are not easy to understand, since finding all references can be difficult. Even though the risk was understood, subjects found it hard to come up with the right refactoring to overcome this smell. This is partly caused by the fact that a long list of references can indicate that the placement of formulas is not optimal, and hence this smell can also reveal a weakness in the organization of the spreadsheet.

Refactorings to the location of formulas were found especially hard for the subjects, and support for this, like a preview, is definitely a promising avenue for future research.

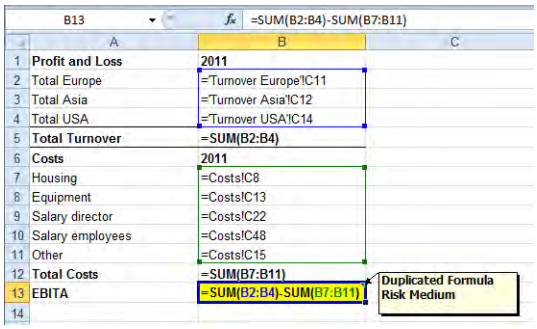
Finally we found it interesting that Excel's feature to color the cells referenced by a formula is only helpful in cases with few references (typically above 6 it got confusing for the participants). There is apparently a need for better support in locating references.

4) Duplicated Formula: **Findings** In the evaluation we found that the cases in which duplicated formulas are detected can be divided into two distinct categories.

- *Sharing Subtrees:* Different formulas are sharing a subtree, and there is an opportunity for refactoring.
- *Rare Formulas:* There is one formula that differs slightly from its neighbors, and therefore shares a subtree with these neighbors.

Figures 4 and 5 illustrate the two categories. In Figure 4 the highlighted formula (in cell B13) shares the subtree *SUM(B7: B11)* with the formula in cell B12. The same subtree occurs twice, so it might be better to replace *SUM(B7: B11)* in B13 with a reference to B12. In Figure 5 however something different is happening. The selected formula (E4) shares a subtree with the other formulas in the same row, each summing up the values of the three cells above it.

However, there is a small difference with the other formulas, which is the '+0.1', denoting the formula as *rare*, it is not like other formulas in the worksheet. Excel itself recognizes the risk of this type of formulas. This is one of the possible errors that Excel marks with a green triangle in case a formula in a row or column differs from its direct neighbors. Others are the referencing of an empty cell, and numbers formatted as text.



	A	B	C
1	**Profit and Loss**	**2011**	
2	Total Europe	='Turnover Europe'!C11	
3	Total Asia	='Turnover Asia'!C12	
4	Total USA	='Turnover USA'!C14	
5	**Total Turnover**	**=SUM(B2:B4)**	
6	**Costs**	**2011**	
7	Housing	=Costs!C8	
8	Equipment	=Costs!C13	
9	Salary director	=Costs!C22	
10	Salary employees	=Costs!C48	
11	Other	=Costs!C15	
12	**Total Costs**	**=SUM(B7:B11)**	
13	EBITA	=SUM(B2:B4)-SUM(B7:B11)	
14			

Figure 4. A formula with duplication

In our ten spreadsheets, we encountered two cases of a Rare Formula. In both of them, a formula was marked as having a lot of duplication, turned out to differ from the other formulas in its column, while the participants stated that this was actually wrong. Thus, the smell indicated an actual error in the spreadsheet.

Note that Excel was able to mark only one of these cases as possibly dangerous: Excel spots discrepancies between directly adjacent cells, whereas one of these errors involved cells disconnected from each other.

Opinions differed on the six cases in which sharing subtrees were encountered. Four of the subjects understood that having the references at one place made the structure of the spreadsheets better. However the remaining two saw no harm in the duplication of formulas. This is notable, since with source code many people agree that duplication should be avoided.

With respect to the risk maps, we noticed that the current implementation of the pop up as does not yet provide enough information: It only marks the formula that shares subtrees with many formulas, but does not indicate with what cells the subtrees are shared. This resulted in participants looking through formulas in the spreadsheet to find the formulas that shared a subtree. A possible solution could be to include the list of sharing formulas in the pop up, or create an Excel plug in that highlights the sharing formulas when a formula suffering form duplication is selected. We will address this in future work.

	A	B	C	D	E
1	Interest payable				
2	*- Group*	-20	-33.3	-28.9	-23.7
3	*=-- share of joint vent.*	-5.2	-7.2	-6.3	-3.7
4	Profit before taxation	=SUM(B1:B3)	=SUM(C1:C3)	=SUM(D1:D3)	=SUM(E1:E3)+0.1
5	Taxation	-22.1	-21	-14.7	-13.2
6	Taxation on exc. It.	-	-	-	-
7	Profit after taxation	=SUM(B4:B5)	=SUM(C4:C5)	=SUM(D4:D5)	=SUM(E4:E5)-0.1
8					
9					
10					

Figure 5. A rare formula

Conclusions Rare formulas can reveal true weaknesses and even errors in spreadsheets, and spreadsheet users agree with that.

However, the refactoring of duplicate pieces of formulas — in source code refactoring very common— is not considered to be am improvement to all spreadsheet users.

5) Long Calculation Chain: **Findings** This smell triggered most discussion with the five subjects whose spreadsheets were diagnosed with this smell.

The main reason was the fact that the risk maps do not provide enough information to understand this smell immediately. When a certain cell suffers from the Long Calculation Chain smell at the 70% level, this means that the path from this formula to the beginning of the longest calculation chain is at least 5 steps. The cells that are included in this calculation chain were not shown in the pop up. This led to spreadsheet users repeatedly stepping through formulas to check whether a formula indeed had a long calculation chain; and whether that was necessary.

Two of the subjects found that the calculation chain (in one case 10, in the other 17 steps) was indeed long, and that some restructuring would improve readability. The other three subjects found that, although the number of steps was

high, this was necessary in order to calculate the needed values. We did notice that it is easier to trace and understand the different calculation steps when they are located in the same row or column. When we asked the five subjects about this, they concurred. This means there is a need for an additional metric based on the location of the cells involved in the calculation chain. We will look into this in future research.

Furthermore there is the trade off between Multiple Operations and Multiple References on one the hand, and Long Calculation Chain on the other. When discussing this phenomenon with the five subjects, we learned that they felt in need of guidance where the right balance is. Hence, better support for managing this trade off is needed. This might be done with our risk maps or with other interfaces to help users to find the proper balance between the formula smells.

Conclusions Long Calculation chains are relatively common, but are difficult to refactor for spreadsheet users. Hence more support to help users to understand and refactor this smell is necessary.

6) Conditional Complexity: **Findings** This metric was the least common in the case study, similar to the finding in the evaluation of the Euses corpus. In the two spreadsheets in which it was located, the risk maps easily helped in locating the Conditional Complexity smell. When the users selected the cells suffering from the smell, they learned from the pop up that nested conditionals were found in the formula.

The two subjects understood and even apologized for the detected smell, stating "I know this formula is too hard, I was trying to get something to work, and then it just remained like that". Both subjects were well aware of the fact that nesting more than two conditional formulas was not such a good idea.

Conclusions The Conditional Complexity smell is in fact already known to spreadsheet users. Apparently there is some notion among the spreadsheet users that conditional operations are complex and should be handled with some care, probably explaining the low occurrence of this smell.

X. Answers to Research Questions

With the results of the EUSES analysis and the case studies, we revisit the research questions.

R_1 What spreadsheet smells are most common, and why? In both evaluations we have seen that Multiple Operations and Multiple References are the most common smells, and from the second evaluation we have learned that this is often caused by the modification of a spreadsheet, sometimes under time pressure. Since there is little awareness of the risks of Multiple Operations, spreadsheet users seem not to be concerned too much about maintainability of formulas. They keep extending formulas with more operations and more references, causing formulas to become long and complicated.

R_2 To what extent do formula smells expose threats to spreadsheet quality? We found two actual faults in a spreadsheet by looking at the Duplication Smell. With respect to the other smells, the concern caught is lack of understandability. Spreadsheet users found that our current smell detection strategies reveal the formulas that are the least maintainable. These formulas will be time consuming to change, and changes made will be more error prone.

R_3 To what extent are risk maps an appropriate way to visualize spreadsheet smells? The strengths of risk maps include their simplicity and the fact that the visualization is shown within the context of the spreadsheet. Seven subjects explicitly stated they liked the risk maps, posing statements like "these colors draw my attentions to the formulas that deserve a second check". Users furthermore appreciated the different levels of the smells, allowing them to inspect the worst formulas first. For the Long Calculation smell, however, additional interface elements are needed, in order to help spreadsheet users understand the cause of the smell.

Beyond risk maps, three of the subjects asked for a general score of the quality of their spreadsheet. Although we could provide them with the number of smells and their severity by looking into our database, an aggregation of the metrics is not provided by the current prototype. This could, for instance, be added by generating an extra worksheet in the spreadsheet in which overviews of all metrics are shown.

XI. Discussion

A. Named Ranges

In the current set of smells we have not taken into account *named ranges*, a spreadsheet feature allowing users to assign a name to a number of cells. We encountered named ranges in one of the case studies, where a formula that summed a named range, SUM(NamedRange), was marked as having the Many Reference smells. Initially the subject did not understand why it was marked as referring to many different ranges, since there was only one reference. The named range itself however consisted of several separate ranges. This raises the question whether we think this is smelly, and why. Note that the smells is in fact related to the named range itself —it is probably not a good idea to create a named range consisting of multiple ranges— rather than to the formula referencing the named range.

B. Applicability of the risk maps

Our risk map visualization exhibits limitations if the smell in question addresses concerns not exclusively contained in the smelly formula itself. This explains why some subjects were dissatisfied with the pop-ups of Long Calculation Chain and Duplicated Formulas, which essentially require information from cells outside the smelly formula itself. In future research we will explore how to present smells at different levels of abstraction in one integrated view.

C. Spreadsheet Evolution

While performing the case study, subjects shared that spreadsheets tend to undergo many changes during their life time (an observation also made in [9]), and that these changes can lead to a degradation of formula quality. This is an issue that warrants further investigation, calling for a longitudinal study of spreadsheet quality, and opening up the possibility of spreadsheet quality monitoring tools.

D. Threats to Validity

A threat to the external validity of our quantitative evaluation concerns the representativeness of the Euses Corpus spreadsheet set. This set, however, consists of 4223 spreadsheets covering 11 different domains. Furthermore, it has been used in many other studies and is collected from practice.

A threat to the external validity of our qualitative evaluation concerns the representativeness of the selected set of employees of Robeco and their spreadsheets. However other papers [15], [16] report on industrial spreadsheet stories similar to the ones we found at Robeco, so their practice seems representative. Further studies are however needed to generalize our findings.

With respect to internal validity, one of the threats is the fact that we did not pick a random sample of people. This effect can be decreased by using a larger test group in future experiments. We however believe the current test group serves as a good reference group, as the persons varied in age, function and daily tasks with spreadsheets. By working with practitioners we tried to maximize the realism of our evaluation, which unfortunately comes at the price of reduced repeatability.

XII. Related Work

Research efforts related to ours include papers that provide spreadsheet design guidelines. Raffensberger [17], for instance advises to merge references that occur only once. He furthermore states that unnecessary complex formulas with many operations and parenthesis should be avoided. Rajalingham *et al.* [18] also propose guidelines to improve spreadsheet quality, which they base on principles of software engineering.

Secondly, there are papers that address common errors in spreadsheets, like [19], [20], together with their causes. Powell *et al.* for instance [21] names conditional formulas (which is one of our smells) among the top three of commonly occurring spreadsheet error categories.

Furthermore there is related work on finding anomalies on spreadsheets, for instance the work on the UCheck tool [10], [11], [12]. UCheck determines the type of cells, and locates possible anomalies based on this type system. UCheck uses a similar visualization, with colors in the spreadsheet, to indicate found anomalies.

We ourselves have worked on spreadsheet smells in previous work [2]. In that paper we focused on detecting smells between worksheets, like high coupling. That paper followed our earlier work, in which we worked on the visualization of spreadsheets by means of class diagrams [22] and dataflow diagrams [9].

This paper differs from our previous work by focusing on detecting smells in spreadsheet formulas. Recently, other work on spreadsheet smells has been published [23], that aims at smells in values, such as typographical errors and values that do not follow the normal distribution. Other recent work by Badame and Dig [24] suggests an approach to support spreadsheet users in removing formula smells by refactoring.

XIII. Concluding Remarks

The goal of this paper is to investigate the applicability of code smells to spreadsheet formulas as a means to asses and improve spreadsheet quality.

To that end we have created a list of formula smells, based on our experiences with spreadsheets, related work in spreadsheet guidelines and literature on code smells.

We then defined a set of metrics for detecting five formula smells and presented the visualization of these smells with the spreadsheet risk map. We have evaluated the metrics and the risk map with a qualitative and quantitative evaluation. The quantitative evaluation was performed on the spreadsheets from the EUSES corpus. The qualitative evaluation was with spreadsheets from ten professional spreadsheet users from industry.

The key contributions of this paper are as follows:

- An analysis of the risky types of spreadsheet formulas
- An evaluation of these formula smells on the EUSES corpus and with ten professional spreadsheet users and their spreadsheets.

We have found that spreadsheet formula smells occur frequently, and can pose a real threat to spreadsheet understandability, and can even detect actual errors. Spreadsheet users in our qualitative evaluation found that the risk maps were a good way of indicating formula smells, and that the three thresholds helped them get a feeling of the importance of the located smells.

The current research gives rise to several directions for future work. Firstly, the definitions of the current set of metrics could be refined; as mentioned in the evaluation section, we could split the duplication metric, and add a metric for the location of cells in a long calculation chain. Secondly, some smells ask for a more elaborate visualization, for instance to indicate the balance between Multiple Operations and Long Calculation Chain. Finally, more support for formula refactoring is needed. We plan to investigate means to suggest such refactorings to the spreadsheet user, give them a preview of the result, or even perform them automatically.

REFERENCES

[1] W. Winston, "Executive education opportunities," *OR/MS Today*, vol. 28, no. 4, pp. 8–10, 2001.

[2] F. Hermans, M. Pinzger, and A. van Deursen, "Detecting and visualizing inter-worksheet smells," in *Proceeding of the 34rd international conference on Software engineering (ICSE 2012)*. ACM Press, 2012, pp. 451–460, to appear.

[3] M. Fisher and G. Rothermel, "The EUSES spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms," in *Proceedings of the Workshop on End-User Software Engineering*. ACM, 2005, pp. 47–51.

[4] M. Fowler, *Refactoring: improving the design of existing code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[5] B. Nardi and J. Miller, "The spreadsheet interface: A basis for end user programming," in *Proceeding of Tthe IFIP Conference on Human-Computer Interaction (INTERACT)*. North-Holland, 1990, pp. 977–983.

[6] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. L. Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Trans. Software Eng.*, vol. 36, no. 1, pp. 20–36, 2010.

[7] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proceedings of the 28th International Conference on Software Engineering (ICSE 2006)*. ACM, 2006, pp. 182–191.

[8] T. L. Alves, C. Ypma, and J. Visser, "Deriving metric thresholds from benchmark data," in *26th IEEE International Conference on Software Maintenance (ICSM 2010)*. IEEE Computer Society, 2010, pp. 1–10.

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

[10] R. Abraham and M. Erwig, "Ucheck: A spreadsheet type checker for end users," *Journal of Visual Languages and Computing*, vol. 18, pp. 71–95, February 2007.

[11] C. Chambers and M. Erwig, "Automatic detection of dimension errors in spreadsheets," *J. Vis. Lang. Comput.*, vol. 20, pp. 269–283, August 2009.

[12] M. Erwig, "Software engineering for spreadsheets," *IEEE Softw.*, vol. 26, pp. 25–30, September 2009.

[13] R. Abraham and M. Erwig, "How to communicate unit error messages in spreadsheets," in *WEUSE I: Proceedings of the first Workshop on End-User Software Engineering*. ACM Press, 2005, pp. 1–5.

[14] F. Hermans, M. Pinzger, and A. van Deursen, "Breviz: Spreadsheet visualization and quality analysis," in *Proceedings of the EuSpRIG 2011 Symposium*, 2011, pp. 63–72.

[15] D. G. Hendry and T. R. G. Green, "Creating, comprehending and explaining spreadsheets: a cognitive interpretation of what discretionary users think of the spreadsheet model," *International Journal of Human-Computer Studies*, vol. 40, no. 6, pp. 1033–1065, 1994.

[16] R. Panko, "Facing the problem of spreadsheet errors," *Decision Line*, vol. 37, no. 5, 2006.

[17] J. Raffensperger, "New guidelines for spreadsheets," *International Journal of Business and Economics*, vol. 2, pp. 141–154, 2009.

[18] K. Rajalingham, D. Chadwick, B. Knight, and D. Edwards, "Quality control in spreadsheets: a software engineering-based approach to spreadsheet development," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*. IEEE Comput. Soc, 2000, pp. 133–143.

[19] Y. Ayalew, M. Clermont, and R. T. Mittermeir, "Detecting errors in spreadsheets," in *Proceedings of EuSpRIG 2000 Conference*, 2000, pp. 51–62.

[20] R. Panko, "What we know about spreadsheet errors," *Journal of End User Computing*, vol. 10, no. 2, pp. 15–21, 1998.

[21] S. Powell, K. Baker, and B. Lawson, "Errors in operational spreadsheets: A review of the state of the art," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICCS 2009)*. IEEE Computer Society, 2009, pp. 1–8.

[22] F. Hermans, M. Pinzger, and A. van Deursen, "Automatically extracting class diagrams from spreadsheets," in *ECOOP 2010 - Object-Oriented Programming, 24th European Conference, Maribor, Slovenia, June 21-25, 2010. Proceedings*. Springer, 2010, pp. 52–75.

[23] J. Cunha, J. P. Fernandes, J. Mendes, and J. S. Hugo Pacheco, "Towards a Catalog of Spreadsheet Smells," in *The 12th International Conference on Computational Science and Its Applications*, ser. ICCSA'12. LNCS, 2012, to appear.

[24] S. Badame and D. Dig, "Refactoring meets spreadsheet formulas," in *IEEE 28th International Conference on Software Maintenance, ICSM 2012*. IEEE, 2012, to appear.