Patrick Koch, BSc

Smelly Spreadsheet Structures: Structural Analysis of Spreadsheets to enhance Smell Detection

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Wotawa

Dipl.-Ing. Dr.techn. Birgit Hofer

Institute for Software Technology

Graz, April 2016

Abstract

The use of spreadsheets is the most popular form of end-user programming. As spreadsheets are in general not created and maintained by professional programmers, error rates are high. As a remedy, many approaches were proposed to avoid, find, and fix errors in spreadsheets. A number of these approaches, mostly proposing unit- and type inference techniques, take the structure of spreadsheets into consideration. However, in general, structural information is rarely used in established spreadsheet quality assurance (QA) techniques. This is attributable to structure inference being a difficult problem. Hence, no general process exists for the extraction of these structures. In this master's thesis, we present a novel analysis process to identify spreadsheet structures. Our analysis process follows three main concepts: First, our approach focusses on structural information provided by formula cells and formula relations. Second, wherever possible, we utilize one-dimensional areas for our analysis purposes, merging individual cells or partitioning two-dimensional areas. Third, in case of ambiguity during the analysis process, we opt for the interpretation which requires the least amount of assumptions. We evaluated the performance of our structural analysis process using the EUSES and ENRON spreadsheet corpora. We compared expected high-level structures, detected by manual inspection, with the results of our structural analysis process. Almost all of the expected structures could be found at least partly by our approach. Moreover, more than 80 % of structures could be inferred in their entirety. We conclude that these structures are well suited to enhance spreadsheet QA techniques. To demonstrate how structural information could be applied, we enhance the detection processes of spreadsheet smells (a QA technique) by following two approaches: First, we propose five updates to the catalogue of established smells making use of structural information. Second, we introduce seven new smells based on spreadsheet structures. The proposed enhancements reduce the runtime of smell detection routines, lower the number of false positive smell detections, and allow for the detection of further issues in form of new smells.

Kurzfassung

Kalkulationstabellen sind die gebräuchlichste Form von Programmierung durch Endnutzer. Sie sind jedoch fehleranfällig, da sie zumeist nicht von Fachkundigen erstellt werden. Um dem entgegenzuwirken wurde eine Vielzahl von Techniken zur Vermeidung, Aufsprüng, und Behebung von Fehlern in Kalkulationstabellen entwickelt. Einige dieser Techniken, zum Beispiel Unit- und Type-Inference, verwenden Strukturinformationen von Tabellen. Der Großteil der Techniken zur Qualitätssicherung (QS) von Kalkulationstabellen macht jedoch keinen Gebrauch von Strukturinformationen. Dies ist darauf zurückzuführen, dass die Ermittlung dieser Informationen ein schwieriges Problem darstellt. Es gibt folglich noch keinen allgemeinen Prozess, welcher solche Strukturen offenlegt. In dieser Masterarbeit stellen wir einen neuartigen Prozess zur Analyse von Strukturen in Kalkulationstabellen vor. Unser Prozess befolgt drei grundlegende Prinzipien: Erstens, wir verwenden Strukturinformationen die durch Formel-Relationen gedeckt sind. Zweitens, wir verwenden eindimensionale Zellbereiche. Einzelne Zellen werden hierzu zusammengefügt, und zweidimensionale Zellbereiche werden geteilt. Drittens, in Fällen die mehrere Interpretationsmöglichkeiten bieten, folgen wir jener Interpretation welche die wenigsten Annahmen voraussetzt. Wir haben die Funktion unseres Analyseprozesses unter Anwendung der EUSES und ENRON Kalkulationstabellen-Sammlungen begutachtet. Zu diesem Zweck wurden oberflächliche Strukturen, die durch manuelle Inspektion festgestellt wurden, mit den Ergebnissen unserer strukturellen Analyse verglichen. Nahezu alle erwarteten Strukturen konnten anteilsweise von unserem Ansatz entdeckt werden. Über 80 % dieser Strukturen konnten in ihrer Gesamtheit erschlossen werden. Wir folgern, dass diese Strukturen dazu geeignet sind um QS-Techniken zu verbessern. Wir demonstrieren eine solche Anwendung am Beispiel von Spreadsheet Smells, einer QS-Technik. Zum einen schlagen wir fünf Erweiterungen die durch Strukturinformationen ermöglicht werden für bestehende Spreadsheet Smells vor. Zum anderen etablieren wir sieben neue Spreadsheet Smells, deren Funktionsweise auf Strukturinformationen basiert. Die vorgeschlagenen Verbesserungen verringern die Laufzeit von Smell-Detektierung, verringern falsch-positive Smell Detektionen, und ermöglichen es neue Qualitätsmängel mit Hilfe von Spreadsheet Smells offenzulegen.

AFFIDAVIT

Eidesstattliche Erklärung

Acknowledgements

First of all, I would like to thank Professor Wotawa, whose leadership enabled and inspired me to take on this master's thesis in the first place. Furthermore, I would like to express my sincerest gratitude to Birgit Hofer, my guiding light in the academic twilight, who, despite personal time constrains, provided me with a near infinite amount of feedback and invaluable insights. Thank you for your guidance and support. A big thank you is also well deserved by Elisabeth, my resident proofreader, whose many and well-considered advices allowed me to push the language level of this work from half-decent to decent or beyond. Thank you for sharing your wisdom with me. Not least of all, I would like to express my deepest gratitude to my family and friends, who accompanied and supported me all the way throughout my studies and during my work on this thesis.
Thank you. All of you.

Table of Contents

1. Introduction

Spreadsheets are an established all-purpose tool for management and business applications. This is owed to the key strengths of the spreadsheet paradigm: intuitive operation, wide availability, and diverse applicability. An analysis of computer usage across 95 international organisations [Taylor, 2007] summarized that 7.4 % of computer working hours are spent using Microsoft Excel. 95 % of businesses utilize spreadsheets for financial accounting. The amount of individual spreadsheets in use is considerable. Panko and Port [2012] highlight some results of a 2010 spreadsheet census: For instance, a large government agency had 630,000 spreadsheets in use. A large health insurance company estimated to use about 2,000 spreadsheets. In reality, they had 42,000. A very large global bank had 8 to 10 million spreadsheets in use, while estimating to have one to two million. These examples indicate that spreadsheets are far more common and influential than widely believed.

Spreadsheets are used by many professions and in various levels of hierarchical structures. Entrepreneurs and white-collar workers are among spreadsheet users, as are financial experts. However, only 15 % of spreadsheet users are trained experts. Consequently, errors in spreadsheets are a common occurrence. In some cases, such errors can lead to costly consequences. The European Spreadsheet Risk Interest Group (EuSpRiG) lists economically significant incidents which are related to spreadsheet errors [2013]. In 2011, a calculation error of the financial advisory firm Ehelers resulted in a spreadsheet underestimating the total cost of a 10-year bond by $400,000. In the same year, a spreadsheet error lead the Norwegian Central Bank to issue wrong information regarding a change of rate hikes. Lastly, the 2010 paper by Harvard economists Carmen Reinhart and Kenneth Rogoff, 'Growth in a Time of Debt', was quoted to justify widespread international austerity measures. However, the spreadsheet employed by Reinhart and Rogoff contained wrong assumptions and calculation errors, each biasing the outcome of the calculations in favour of their result. Without these issues present, the result of their paper is inconclusive.

These and other incidents in recent years lead to a heightened awareness for the necessity of spreadsheet Quality Assurance (QA) techniques. Various approaches to prevent, detect, and fix faults within spreadsheets have been proposed. In particular, numerous approaches were made to adopt common techniques for code quality improvement into the spreadsheet domain. Jannach *et al.* [2014] provide a comprehensive overview of noteworthy approaches to improve spreadsheet quality. They also establish a classification of these approaches into six categories, based on their general working principle.

One of these categories is static spreadsheet analysis. Approaches within this category attempt to adapt established techniques of static code analysis to the spreadsheet domain. Two general types of approaches are part of this category. First, spreadsheet smells that detect suspicious characteristics based on particular calculated metrics. Second, unit and type inference approaches which attempt to relate cell values to labels and other information within a spreadsheet. Consequently, established relations can be used to infer unit information for related values. This unit information can then be applied to validate formula calculations.

Structural analysis is employed by some unit and type inference approaches. However, hardly any other established spreadsheet QA techniques we are aware of apply structural information in any significant amount. This is likely due to the fact that identification of meaningful structures within a spreadsheet is no trivial task. While spreadsheet users usually follow some conventional guidelines, spreadsheet structures in general do not follow specific rules. However, structural analysis could potentially provide valuable information to benefit spreadsheet QA.

As a consequence, we established a novel analysis approach to identify spreadsheet structures. During the process, we attempt to infer groups of related formula cells and referenced value cells. We further detect calculation blocks which denote separate areas containing groups of formula and value cells. Lastly, we infer relations of matching header cells to the columns and rows of detected calculation blocks.

Structural information inferred by our analysis process may benefit various spreadsheet QA techniques. We focus on enhancement of spreadsheet smell detection by following two basic approaches: First, we propose updates to existing smells to make use of structural information. A catalogue of spreadsheet smells was already established by the scientific community. We point out possibilities of enhancement for either groups of related smells, or specific individual smells. Second, we introduce new smells based on the results of our structural analysis process.

The main contributions of this master's thesis are (A) a novel analysis approach to infer structural information of a spreadsheet, (B) empirical evaluation of the performance of our structural analysis, (C) a set of proposals to update existing spreadsheet smells based on structural information, and (D) a set of new spreadsheet smells which utilize structural information in their detection mechanisms.

The remainder of this thesis is structured as follows: Section 2 provides preliminary descriptions and formal definitions used throughout the rest of the thesis. In Section 3, we introduce the principle of structural analysis, presenting the UCheck Header Inference process as an example. We outline the goals and concepts employed by our structural analysis approach. We then establish the general workflow of our novel analysis process, and explain each step within the workflow in greater detail. In Section 4, we present the spreadsheet corpora we used for our evaluation purposes and describe the filtering process we applied to each corpus. We then describe our evaluation process and present our findings. In Section 5, we introduce the notion of spreadsheet smells and present the catalogue of spreadsheet smells proposed by the scientific community. For each smell within the catalogue, we provide detailed information based on a defined set of criteria. We highlight similarities and interactions between specific smells and describe how the concept has been utilized throughout literature. Section 6 outlines possibilities to apply structural information to spreadsheet smells. We point out approaches to enhance existing spreadsheet smells, and also present a list of novel spreadsheet smells which utilize structural information. In Section 7, we summarize the relevant research related to spreadsheet smells. Section 8 concludes the thesis and points out opportunities for future work.

2. Preliminary Definitions

In this section, we provide preliminary descriptions and formal definitions used throughout the rest of the paper. In specific, we establish the terminology describing spreadsheets, spreadsheet components, and their interdependencies. We also establish a set of functions, providing a formal language to access and work with the previously defined terms.

Definition 1 (Cell)**.** A cell C is the basic building block of a spreadsheet. Each cell contains a specific item of information and is located at a specific position within a spreadsheet. Each cell can either be empty or contain one item of user-supplied content.

Definition 2 (Cell content)**.** Cell content refers to the information a user supplied for a specific cell. Each cell may either be empty, symbolized by the value ϵ, or contain one item of one of the following content types:

- **a Boolean value**, representing a logic or truth value. Allowed values for Boolean items are true and false.
- **a numeric value**, representing a number or quantity. Allowed values are integer as well as floating point numbers. Dates are also represented as a numeric value with special formatting.
- **a formula**, representing some form of calculation. Formulas allow for the calculation of values based on a number of supported spreadsheet functions and operators. Each formula is made up of a combination of operators and operands. Operators describe the applied calculations of the formula. Operands provide input values for operators. These are either references to values of other cells, or constant values.

- **a string value**, representing textual information stored within a character string. Strings are typically used as column and row headers to annotate data contained within the spreadsheet. However, strings may also be used as input values for spreadsheet formulas.

- **an error value**, representing an error which was detected during the evaluation of a formula. Various different error values are available to signal specific error circumstances (e.g.: division by zero).

The function *content*(cell C) provides access to the content of a specific cell. It is of the form C $\rightarrow$ *content*. Thus, providing the function with a cell C it returns an item of *content*. *Content* denotes either an empty result ϵ or the user-supplied content of cell C.

Definition 3 (Content view)**.** The cell contents of all cells in a spreadsheet make up the content view, or input view, of that spreadsheet. This view describes the state of a spreadsheet before any evaluation of cell content items has taken place. Thus, formulas in this view are represented by their formula string.

The content view is one of the two views, commonly employed when representing spreadsheets. Static analysis methods, like the ones we describe in this paper, use this view of a spreadsheet. The other established spreadsheet view is the value view, described in Definition 7.

Definition 4 (Cell type)**.** Each cell has a specific cell type that is based on the type of cell content it contains. The function *type*(cell C) provides access to the type of a specified cell. It is of the form C $\rightarrow$ *type*, and defined as

$$\mathit{type}(C) = \begin{cases} empty & \text{if } \mathit{content}(C) \equiv \text{the empty value } \epsilon \\ boolean & \text{if } \mathit{content}(C) \equiv \text{a Boolean value} \\ numeric & \text{if } \mathit{content}(C) \equiv \text{a numeric value} \\ string & \text{if } \mathit{content}(C) \equiv \text{a string value} \\ formula & \text{if } \mathit{content}(C) \equiv \text{a formula} \\ error & \text{if } \mathit{content}(C) \equiv \text{an error value} \end{cases}.$$

Definition 5 (Cell value)**.** Beside the static, user-supplied content, each cell also features a cell value. The value of a cell is determined via evaluation of the cell's content.

Cell values allow for value-based calculations in formula cells. The function *value*(cell C) provides access to the value of a specific cell. It is of the form C $\rightarrow$ *value*, whereby the resulting *value* depends on the cell's content. The function *value*(cell C) depicts this evaluation process and is defined as

$$value(C) = \begin{cases} \text{the numeric value 0} & \text{if } type(C) = empty \\ \{\text{boolean, numeric, string, error}\} & \text{if } type(C) = formula \\ content(C) & \text{otherwise} \end{cases} \cdot$$

Evaluation of an empty cell results in the numeric value 0. Evaluation of a formula cell results in either a Boolean, numeric, string, or error value, depending on the evaluated formula. The specific evaluation process of formula cells is not relevant for the type of static analysis we apply in this paper and is therefore not described in further detail. Evaluation of boolean cells, numeric cells, string cells, and error cells results in the respective cell's content as value.

Definition 6 (Value cells)**.** The cells of the types boolean, numeric, string and error are summarized into the set of value cells of a spreadsheet. These cells provide the input values for formula-based calculations within spreadsheets. Each value cell's value is equal to its cell content.

Definition 7 (Value view)**.** The value view of a spreadsheet encompasses non-empty cells within a spreadsheet, represented by each cell's value. Thus, it provides the state of a spreadsheet after evaluation, omitting values of empty cells for the purpose of clarity.

The value view is one of the two views commonly employed when representing spreadsheets. Spreadsheet environments like Microsoft Excel usually present the value view of a spreadsheet to their users. The other established spreadsheet view is the so-called content view, described earlier in Definition 3.

Definition 8 (Worksheet)**.** A worksheet W is a collection of cells within a spreadsheet. Each cell is contained in exactly one worksheet. The set of cells contained in a worksheet is returned by the function *cells*(worksheet W) such that

$$\mathit{cells}(W) = \{\mathit{cell}\ C \mid C \text{ is contained in } W\}.$$

Likewise, the worksheet a cell is contained in is returned by the function *worksheet*(cell C) such that

$$\mathit{worksheet}(C) = worksheet\ W \mid C \text{ is contained in } W.$$

Definition 9 (Coordinates)**.** Cells within a worksheet are arranged in a matrix. Columns and rows of this matrix are identified by an ascending index. Specific indexing practice depends on the writing system applied by the spreadsheet's author. In Left-To-Right (LTR) systems, indexing starts at the leftmost column, assigning the index one to it. In Right-To-Left (RTL) systems, indexing starts at the rightmost column, assigning the index one to it. Independent of the writing system, the topmost row always has the index one. Cell coordinates uniquely define the position of a cell within this matrix. The column coordinate matches the column index of the matrix. The row coordinate matches the row index of the matrix.

The functions *column*(cell C) and *row*(cell C) provide access to the coordinates of a specific cell. They are of the form C $\rightarrow$ *coordinate*, whereby *coordinate* denotes the column- or row-index of the cell within the worksheet.

Definition 10 (Coordinate notation)**.** Coordinates allow to identify a specific cell within a worksheet. A coordinate notation specifies the form these coordinates may be provided in. Within spreadsheet environments and scientific literature, two distinct coordinate notations are commonly used: the A1-notation and the R1C1-notation.

- The **A1-notation** is the standard notation used within spreadsheet environments. Using this notation, the coordinates of the leftmost, topmost cell of a spreadsheet created in a LTR writing system are stated in the Form "A1". The first part of the coordinate information comprises one or multiple letters which symbolize the column index. The letter A, being the first letter of the alphabet, thereby refers to the first column within the spreadsheet at index 1. The letter B refers to index 2, and so on. The second part of the coordinate information is a number which symbolizes the row-index within the spreadsheet.

- The **R1C1-notation** is mainly used to communicate formal properties and relationships within worksheets, as it allows for easy tracking and comprehension of formula references. Using this notation, the coordinates of a cell within a worksheet are stated in the form "R<row>C<col>". The <row> and <col> parts of each coordinate statement are integer values. The <row> value symbolizes the row index of the cell. The <col> value symbolizes the column index of the cell. For example, the coordinates R0C0 refer to cell A1, while the coordinates R1C1 refer to cell B2.
  When used in a formula, coordinate statements may also define a relative reference to another cell within the spreadsheet. To symbolize this, the <row> and <col> parts of the coordinate statement may take the form "[<offset>]". The value <offset> within the blocked quotes symbolizes an integer offset. To determine the position of the referred cell, <offset> should be added to the row or column coordinate of the formula cell which holds the reference accordingly. For example, the reference R[0]C[1] located at position A1 refers to the next neighbour within the same row, cell B1. The reference R[1]C[0] located at position A1 refers to the next neighbour within the same column, cell A2. Lastly, the reference R[2]C[2] located at position A1 refers to the cell two positions to the right and two positions below itself, cell C3.

Definition 11 (Area)**.** An area allows to identify a rectangular section of the cell matrix of a worksheet. Area information in spreadsheets is provided following the area-notation. This notation has the form "<start>:<end>", whereby both <start> and <end> are coordinates in either A1- or R1C1-notation. The referred rectangular section starts with the coordinates of <start> as left-upper corner, and ends with the coordinates of <end> as right-lower corner of the area. Each cell contained within the section of the cell matrix is regarded as part of the area.

Definition 12 (Spreadsheet)**.** A spreadsheet S is a non-empty collection of worksheets. Each worksheet within S is uniquely identified by an alphanumeric name.

WORKSHEETS is the set of all worksheets W contained in the spreadsheet S.
CELLS is the set of all cells C contained in the spreadsheet S.

Definition 13 (Position)**.** A position defines the location of a cell within a spreadsheet. The position of a cell is uniquely defined by the name of the worksheet W it is contained in and its coordinates within W. Position statements follow the notation "<worksheet>!<coordinates>".

The <worksheet> part of the notation refers to the name of the worksheet in S. The !-symbol acts as separator between worksheet and coordinate part of each statement. The <coordinates> part of the notation defines the coordinates of the cell within the named worksheet. This information may be supplied in either A1- or R1C1-notation.

Access to the worksheet, the coordinates of a particular position, and the cell which is described by this position is required for the description of our static analysis process, and provided by the following functions:

- The function *worksheet*(position P) provides access to the worksheet of a specific position. It is of the form P $\rightarrow$ *worksheet*, whereby *worksheet* denotes the worksheet of the position.
- The function *coordinates*(position P) provides access to the coordinates of a specific position. It is of the form P $\rightarrow$ *coordinates*, whereby *coordinates* denotes the coordinates of the position within its worksheet.
- The function *cell*(position P) is of the form P $\rightarrow$ *cell* and provides access to the cell that is defined by position P.

Definition 14 (Area Position)**.** An area position defines the location of an area within a spreadsheet. The position of an area is uniquely defined by the name of the worksheet W it is contained in and its start- and end-coordinates within W. Area position statements follow the notation "<worksheet>!<start>:<end>".

The <worksheet> part of the notation refers to the name of the worksheet in S. The !-symbol acts as separator between worksheet and coordinate part of each statement. The <start> and <end> parts of the notation define the start- and end-coordinates of the area within the named worksheet. This information may be supplied in either A1- or R1C1-notation. Area position references to areas within the same worksheet may omit the <worksheet>- and !-symbol-parts of the format.

Access to the worksheet, the area of a particular area position, and the set of cells that is described by this area position is provided by the following functions:

- The function *worksheet*(area position P_a) provides access to the worksheet of a specific area position. It is of the form $P_a \rightarrow$ *worksheet*, whereby *worksheet* denotes the worksheet of the position.
- The function *coordinates*(area position P_a) provides access to the area of a specific area position. It is of the form P $\rightarrow$ *area*, whereby *area* denotes the area of the area position within its worksheet.
- The function *cells*(area position P_a) is of the form P $\rightarrow \{cell\}$ and provides access to the set of cells that is defined by the area position P_a.

Definition 15 (Reference)**.** A reference is defined by a pair of positions within a spreadsheet: a base position and a target position. The base position identifies the cell containing the reference while the target position identifies the cell that the reference refers to. Consequently, the target position may contain relative coordinate information while the base position may not contain relative coordinate information. A reference indicates that the base cell contains a formula reference which points to the target cell.

References in formulas provide values stored within other cells of the spreadsheet to be used for calculations. Thus, a reference to a specific cell implies that the referencing cell is dependent on the referenced cell. Circular references as well as self-references are not allowed within the spreadsheet context.

The functions *basePosition*(reference R) and *targetPosition*(reference R) provide access to the positions of a specific reference. They are of the form R $\rightarrow$ *position*, whereby *position* denotes either the base or target position of the reference. The functions *isColRelative*(reference R) and *isRowRelative*(reference R) provide information about whether the target position is referred to in either absolute or relative fashion. They are of the form R $\rightarrow$ *Bool*, whereby the *Bool* value indicates whether the coordinate in question is relative.

Definition 16 (Area Reference)**.** An area reference is defined by a position and an area position within a spreadsheet. The position identifies the base cell of the reference while the area position identifies the target area of the reference. Consequently, the target area position may contain relative coordinate information while the base position may not contain relative coordinate information. An area reference indicates that the base cell contains a formula reference which refers to the target area.

Area references in formulas provide access to values stored within an area of the spreadsheet to be used within aggregating calculation functions. Thus, a reference to a specific area implies that the referencing cell is dependent on the referenced area.

The functions *basePosition*(area reference R_a) and *targetArea*(area reference R_a) provide access to the contents of a specific area reference. They are of the form $R_a \rightarrow$ *position* and $R_a \rightarrow$ *area position*, whereby *position* denotes the base position and *area position* denotes the target area of the reference.

Definition 17 (Predecessor)**.** A predecessor is a cell which a formula cell refers to. The function *predecessors*(cell C) identifies all predecessors of a specific formula cell. It is of the type C $\rightarrow \{C\}$ and is defined such that

$$\begin{aligned}
\mathit{predecessors}(C) = \{\text{cell } C_p \mid (\exists \text{ reference } R : \mathit{cell}(\mathit{basePosition}(R)) = C_p \\
\wedge\ \mathit{cell}(\mathit{targetPosition}(R)) = C) \\
\vee\ (\exists \text{ area reference } R_a : \mathit{cell}(\mathit{basePosition}(R_a)) = C_p \\
\wedge\ C \in \mathit{cells}(\mathit{targetArea}(R_a)))\}.
\end{aligned}$$

Definition 18 (Connection Set)**.** A tuple of cells (A,B) form a connection K if they adhere to the relation $A \in \mathit{predecessors}(B)$. Thus, cell A is a predecessor of cell B. **KS** denotes the set of all connections in a spreadsheet.

Definition 19 (Neighbour)**.** Neighbours of a cell C are cells which are contained in the same worksheet W and are located next to C. The set of neighbours of cell C is returned by the function *neighbours*(cell C) such that

$$\begin{aligned}
\mathit{neighbours}(C) = \{\mathit{cell}\ C_2 \mid \mathit{worksheet}(C) = \mathit{worksheet}(C_2) \\
\wedge\ |\mathit{column}(C) - \mathit{column}(C_2)| + |\mathit{row}(C) - \mathit{row}(C_2)| = 1\}.
\end{aligned}$$

3. Structural Analysis

As we pointed out in previous work [2015], structural analysis of spreadsheets is already an essential part of established spreadsheet QA methods. Spreadsheet smells rely on structural analysis data for their metric calculations. Unit-checking approaches require structural analysis to establish links between header and content cells. Rather than extracting a single metric for further processing, these analysis approaches attempt to externalize information which is provided within the content, structure, and relations of cells within a spreadsheet. Such information also may be employed to formulate further spreadsheet smells, as well as may be utilized to enhance profound refactoring-operations for spreadsheets.

We develop a static analysis approach to generate structural information and relations within worksheets. The analysis employed by the UCheck [2007] system serves as starting point for this approach. To that end, we discuss the structural analysis methods proposed as part of UCheck. Based on this insight, we distinguish which type of spreadsheet we target with our own analysis approach and which structural components we attempt to discern. Next, we present the core contribution of this master's thesis; We describe our structural analysis approach for spreadsheets. We conclude the chapter by discussing some notable edge cases we identified for our analysis approach.

3.1 Ucheck Analysis

UCheck is an error-detection system for spreadsheets which relies on unit information. It automatically processes header and unit inference and reports detected unit errors. Headers are string or value cells, which serve to annotate other cells within a worksheet. Units are related to the concept of units of measurement. A unit

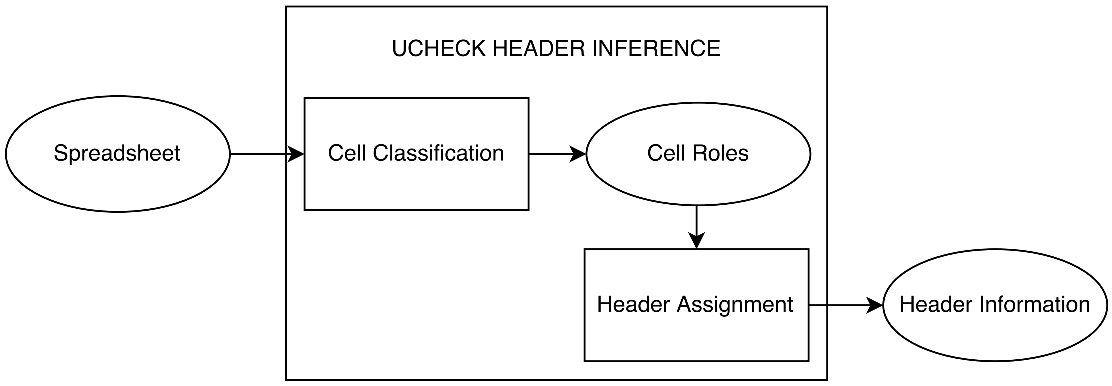


Figure 3.1: Workflow of UCheck Header Inference process.

provides context information to the value of a cell. Consequently, header relations can be applied to provide units to related value cells. The UCheck system relies on two static analysis phases, inferring header and unit information accordingly.

Figure 3.1 outlines the Header Inference process performed by UCheck. In a first step, the cells within a given spreadsheet are classified into predefined roles. Based on this categorization, in a second step, cells that were classified as potential headers are assigned to cells that are likely to be affected by those headers. Lastly, the inferred header information is aggregated and passed on to subsequent processing steps.

3.1.1 Cell Classification

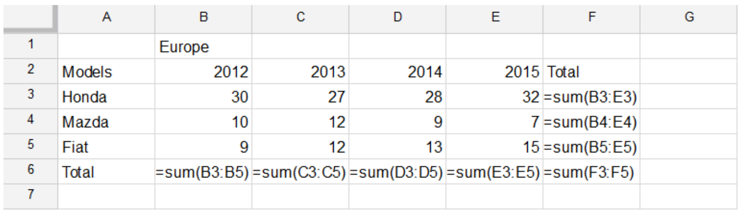
The analysis process of UCheck assumes that a spreadsheet consists of one or multiple tables. Based on this assumption about the spatial arrangement of cells, the system classifies the cells in a given spreadsheet using the following roles:

- *Header.* Cells which are used to label the data.
- *Footer.* Formulas which are typically placed at the end of rows or columns, containing some sort of aggregation function.
- *Core.* Data cells of a spreadsheet.
- *Filler.* Cells which are used to separate tables within the spreadsheet. Filler cells can either be blank or feature a special formatting.

To facilitate successful role-classification in a wide variety of scenarios, Abraham *et al.* introduced a framework to combine the results of several classification algorithms. Each algorithm assigns each cell within the spreadsheet a certain role and specifies a numeric confidence value for this assignment. The applied confidence value indicates the confidence in the classification. Confidence values may range from 1, indicating that the assigned role is not necessarily correct, to 10, indicating that the assigned role is most likely correct. Whenever a cell is classified in multiple categories by different algorithms, the classification with the highest confidence is picked.

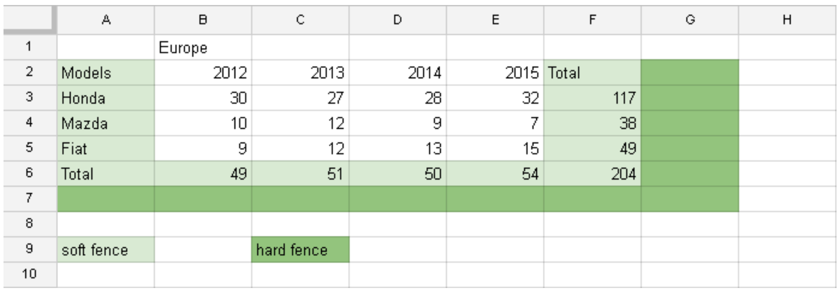
Abraham *et al.* established a number of strategies to determine initial classification values. Figure 3.2 provides an illustration of these strategies, when applied to a spreadsheet. Figure 3.2a depicts the value-view of the example spreadsheet. The sheet contains sales figures for various auto mobile brands over a number of years. The successive figures show the classification results of each strategy when applied to the provided example spreadsheet. The classification of a cell into a specific group is indicated by its background colour. The intensity of the colour represents the confidence level of the classification result. However, the strategy descriptions provided by Abraham *et al.* focus on the purpose and general working principle of each strategy. Consequently, the classification results we provide in Figure 3.2 are based on our interpretation of the strategies.

- Fence identification: Fences are columns or rows which form a boundary (top, bottom, left, or right) of a table within the spreadsheet. Fences are categorized in terms of hardness: Fences consisting only of blank cells are regarded as hard fences, any other fences are regarded as soft fences (e.g. in case of headers and footers of a table). Hard fences are classified with high confidence. Soft fences obtain a low confidence value. To enable detection of soft fences, fence identification may be applied after other classification algorithms already provided initial role-classification results for each cell. Figure 3.2b depicts the identified fences for the auto mobile-sales example. Cells on the border of the filled table area are categorized as soft fences, as they are non-blank. The borders surrounding this area are made up entirely out of blank cells. Consequently, these borders are categorized as hard fences.



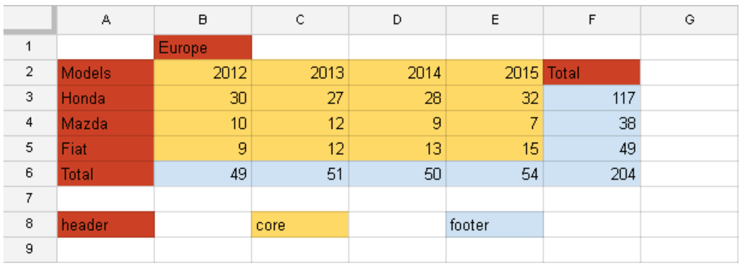
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B5)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							

(a) Formula-view of the auto mobile-sales example.



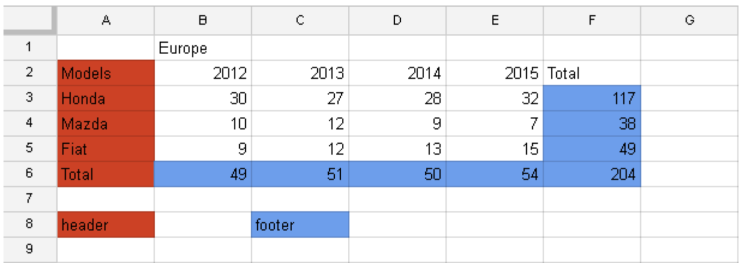
	A	B	C	D	E	F	G	H
1		Europe						
2	Models	2012	2013	2014	2015	Total		
3	Honda	30	27	28	32	117		
4	Mazda	10	12	9	7	38		
5	Fiat	9	12	13	15	49		
6	Total	49	51	50	54	204		
7								
8								
9	soft fence		hard fence					
10								

(b) *Fence identification* results.



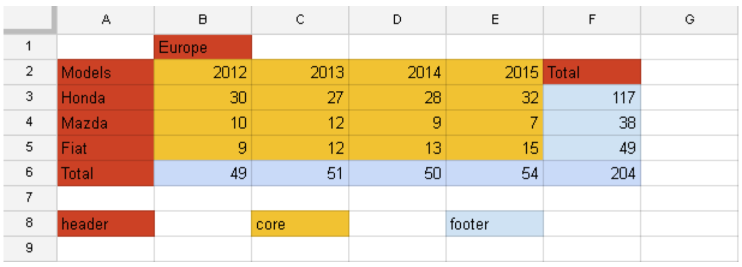
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							
8	header		core		footer		
9							

(c) *Content-based cell classification* results.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							
8	header		footer				
9							

(d) *Region-based cell classification* results.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							
8	header		core		footer		
9							

(e) *Footer-to-core expansion* results.

Figure 3.2: An example worksheet, demonstrating the UCheck Cell Classification process. The spreadsheet depicts sales numbers of various auto mobile brands. Background-colour indicates classification of a cell into a specific role. Colour hue indicates confidence of the classification.

- Content-based cell classification: Cells are classified into header, footer and core roles based on cell content. Following this algorithm, cells containing string values are classified as headers, cells containing numerical values are classified as core cells, and cells containing aggregation formulas are classified as footers. Classification results of this algorithm are assigned with a low level of confidence. Figure 3.2c depicts the results of content-based cell classification of the auto mobile-sales example. The aggregation formulas in column F and row 6 are correctly classified as footers. Each occurring string is correctly classified as header. However, the year dates in row 2 are incorrectly indicated as core cells, along with the correctly classified sales numbers.

- Region-based cell classification: This process employs knowledge about the extent of a table to classify roles with a higher level of confidence. Abraham *et al.* suggest to use identified fences to detect separate tables within a spreadsheet. Based on this information, if the top row or leftmost column of a table contains only strings, those cells are classified as headers with a high confidence level. Likewise, if the bottom row or rightmost column of a table contains aggregation formulas, those cells are classified as footers with a high confidence level. Figure 3.2d depicts the classification of header and footer cells in the auto mobile-sales example. Strings in column A are correctly classified as headers. Aggregation formulas in row 6 and column F are correctly classified as footers. However, header-classification for the remaining header cells of row 2 and cell B1 is failing.

- Footer-to-core expansion: This is a multi-step algorithm. Similarly to content-based cell classification, in a first step, cells containing aggregation formulas are classified as footers with a low confidence level. In the next step, cells referenced by the classified footer cells are classified as core cells with a high level of confidence. These cells are referred to as so-called *seed cells*. In the next step, cells that are immediate neighbours of and share the same type with seed cells are also classified as core cells and act as new seed cells for the analysis. Lastly, cells that are yet uncategorised are assigned a role: cells featuring any content are classified as header cells, whereas empty cells are classified as filler. Figure 3.2e depicts the results of footer-to-core expansion when applied to the auto mobile-sales example. Footer cells are correctly classified. However, expansion of the core-region wrongly encompasses the year dates in row 2. The remainder of the header cells is correctly classified.

3.1.2 Header Assignment

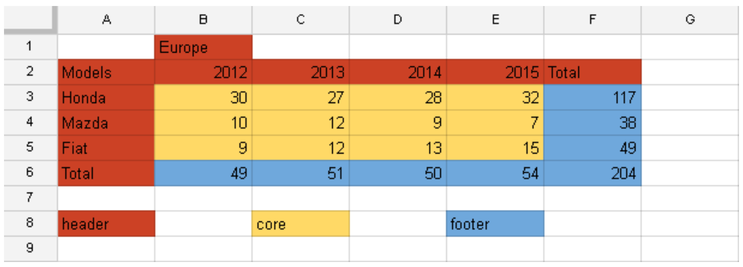
Based on the results of cell classification, UCheck assigns detected header cells to other cells they are related to. This assignment process is carried out in two steps:

1. First-level headers are assigned to core and footer cells. In this step, each core cell is assigned to the nearest header cells within the same row and column as first-level headers.

2. Higher-level headers are assigned to lower-level header cells. In this step, the original set of headers is separated into two sets depending on whether or not they have already been assigned in a header-relation. Some of the contents of the set containing unassigned headers might be candidates for higher-level headers. Other elements in the set might be comments or other additional string-values. Assignment of higher-level headers follow a set of restrictions, guaranteeing a well-formed header hierarchy.

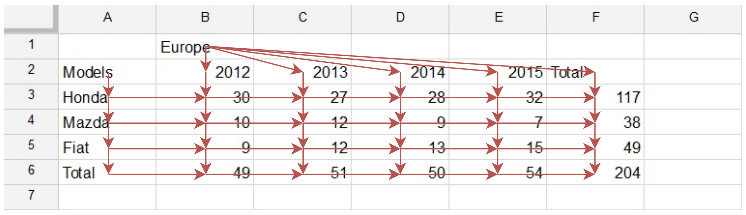
Figure 3.3 illustrates Header Assignment based on the previous example. Figure 3.3a presents the established cell roles for the worksheet. Figure 3.3b visualizes the inferred header relations. First-level Header Assignment establishes the first-level headers in column A and row 2. Higher-level Header Assignment relates cell A2 to the underlying column, and cell B1 to the first-level headers of the underlying row.

Discussion. The analysis of UCheck offers valuable insights into advanced structural analysis methods for spreadsheets. First off, it is important to be aware which type of spreadsheet is applicable for the intended analysis operation. For example, unit-inference may only be applied to spreadsheets that provide inherent unit information (e.g. in form of applicable column and row headers). Moreover, analysis methods require to be hand-crafted for detecting specific features within a spreadsheet. Thus, definition of expected structures and roles within processed spreadsheets should be the starting point of any advanced analysis approach. Lastly, the



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							
8	header		core		footer		
9							

(a) Cell Roles.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

(b) *Header Assignment* results.

Figure 3.3: Example of UCheck Header Assignment Process. Headers are assigned based on previously determined cell roles. First-level headers in column A and row 2 directly related to core and footer cells. Higher-level headers in cells A2 and B1 relate to other headers.

approach of using a combination of multiple classification algorithms and employing confidence values to form conclusive results appears to be good practice.

While our analysis of the UCheck system revealed valuable information, we would also like to point out some drawbacks we encountered: Provided descriptions of the employed cell classification methods are too general and lack a comprehensible example. Additionally, while the confidence values used for cell classification are stated for some of the presented classification algorithms, no reasoning for those specific values is provided. Lastly, classification results of the provided examples are partially incomprehensible. The results cannot directly be inferred by the provided rules. Cells B2, C2, D2, and E2 within Figure 3.3a illustrate one such example. Given the context of the spreadsheet, those cells are clearly intended to provide header information. This classification is even required for the subsequent header assignment process. However, none of the previously applied application steps, illustrated in Figure 3.2, would successfully categorize those cells as headers.

3.2 Analysis Targets

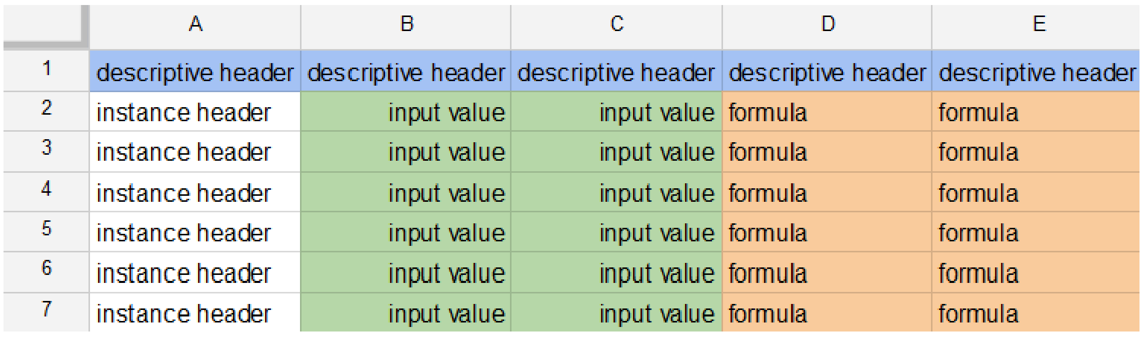
Worksheets harbour implicit structural information. The position of, content of, and formula relations between cells each offer cues to the overall structure of the worksheet. Structural analysis attempts to extract the structural information implied by these cues by means of static analysis. Results from structural analysis may be used to enhance problem detection within spreadsheets. Consequently, better suggestions how to solve those problems can be offered to users. Moreover, profound structural information is necessary to support automatic refactoring operations, as such operations often require structural alterations within a worksheet.

Spreadsheets may be used for a variety of purposes. Examples include the preparation of input-forms, the creation of number-based models, and data management. Our structural analysis approach focusses on the last use case. In general, we expect to analyse calculation-based spreadsheets. The aim of such spreadsheets is the collection of input-data and calculation of results based on the collected data. Thus, calculation-based spreadsheets' main features are formulas, input-data, and, optionally, labels for both.

As our analysis approach focusses on calculation-based spreadsheets, we expect relevant worksheets to contain:

- *formula groups*: groups of formula cells applying a specific calculation narrative to each part of the input-data.
- *input groups*: groups of input cells providing their data to formula groups.
- *descriptive headers*: identifying formula groups and input groups.
- *instance headers*: identifying single items within formula groups and input groups.
- *calculation blocks*: concise areas within the worksheet which aggregate related formula and input groups and allow assignment of matching labels.

Figure 3.4 highlights the basic structural blocks we expect within worksheets. The example features one concise calculation block, or area. This calculation block consists of two columns featuring input data (columns B and C), indicating input



	A	B	C	D	E
1	descriptive header	descriptive header	descriptive header	descriptive header	descriptive header
2	instance header	input value	input value	formula	formula
3	instance header	input value	input value	formula	formula
4	instance header	input value	input value	formula	formula
5	instance header	input value	input value	formula	formula
6	instance header	input value	input value	formula	formula
7	instance header	input value	input value	formula	formula

Figure 3.4: Example of expected worksheet structures.

groups. Moreover, two columns feature formulas (columns D and E), indicating formula groups. Each group is related to one header which is descriptive of the group in row 1. Each formula- and data-item within the group is related to one instance header in column A within the same row. The column of instance-headers in column A also indicates a group of its own and has a related descriptive header at cell A1.

By identifying expected content entities, we can draw conclusions about the purpose of each non-empty cell, and how it relates to other cells within the worksheet. Moreover, more elaborate analysis allows us to unite functionally similar and locally neighbouring cells into concise groups. For each of those groups, we draw conclusions about its purpose and relations to other groups within the worksheet. Lastly, we analyse the coupling between the various groups within a worksheet in order to determine separate functional sections within the worksheet.

3.3 Novel Analysis Approach

Our analysis intends to reveal the underlying structures which were provided by spreadsheet creators. On a basic level, a spreadsheet is simply a collection of cells. Each individual cell either provides an item of data or combines existing data items to generate new data. However, spreadsheet creators have an overarching goal in mind when creating and altering these cells. Usually, this goal takes shape as a thread of calculations which are simultaneously applied to multiple sets of input data. While each individual data item is different, the basic calculation procedure remains the same. Thus, we propose the following alternative interpretation of spreadsheets: A spreadsheet is a collection of cell groups. Each group is devoted to a specific purpose and connected to other cell groups by means of formula relations. The goal of our analysis approach is to detect these cell groups within a given spreadsheet. This approach provides a number of benefits. We provide a summary of possible areas of application in Chapter 8. The main benefits for this thesis are as follows:

- During group synthesis, we detect individual cells and groups that do not follow the expected calculation narrative. We report such outliers as suspicious and therefore smelly.
- We apply the area- and group-information resulting from our structural analysis to conceptualize new quality metrics. These metrics are utilized to formulate additional spreadsheet smells.
- While refactoring operations are usually applied to a limited set of cells, they tend to also affect a number of cells that are related to the initial targets. Structural information allows us to better estimate the effect of refactoring operations and, in consequence, offer better refactoring operations to users.

The following guidelines were applied during conception of our analysis process.

1. *Follow the formula.* Our analysis approach focusses on calculation-based spreadsheets. This focus enables us to ground the analysis process on structural information provided by formula cells and formula relations: Formula relations are the foundation of each of the grouping and blocking mechanisms employed. Likewise, the presented Header Assignment process only allows for the assignment of headers to groups of either formula cells, or cells referred to by formula cells. We do, however, acknowledge that other focal points are viable as well. We suggest those alternatives in Chapter 8.
2. *1D is key.* Areas can be categorized into one of three classes, depending on the dimensions of the area. Zero-dimensional (0D) areas encompass one cell. One-dimensional (1D) areas either encompass only one column but multiple rows, or one row but multiple columns. Two-dimensional (2D) areas encompass both, multiple columns and multiple rows. On various occasions during the analysis approach, we infer areas within a worksheet. Wherever possible, we

will opt to interpret 1D-areas. 0D-areas only contain a single cell. Thus, such areas are already the foundation of usual static analysis and do not yield additional benefit. Consequently, we attempt to merge 0D-areas into 1D-areas wherever possible. Merging of 1D-areas into 2D-areas results in potential loss of information. Therefore, we opt not to infer merged two-dimensional areas, choosing a set of neighbouring 1D-areas instead.

3. *Concrete & Concise.* Lastly, our analysis approach attempts to cover the actual state of the spreadsheet in question. However, even after applying the first two guidelines, opportunities may arise to interpret data in different ways. The drawn conclusions potentially influence results of follow-up analysis steps. Consequently, in case of ambiguity we opt for the interpretation that requires the least amount of assumptions while still allowing the process chain to continue.

With those guidelines in mind, we established the following analysis approach for spreadsheets. The overarching work-flow of our approach is depicted in Figure 3.5 and contains the following three processing steps:

1. *Grouping.* Synthesizes groups of related cells, based on various criteria.
2. *Blocking.* Aggregates groups into cohesive calculation blocks.
3. *Header Assignment.* Assigns header cells to calculation blocks and underlying groups.

Each step in the process chain relies on information provided by preceding processing steps, as well as additional information provided by the spreadsheet. In the following subsections, we will cover each of those steps in greater detail.

3.3.1 Grouping

Grouping represents the cornerstone of our analysis approach and detects cohesive groups of cells which conform with each other based on various cell attributes. Keeping the *Follow the Formula* principle in mind, the employed attributes are associated with formula content and formula relations within cells.

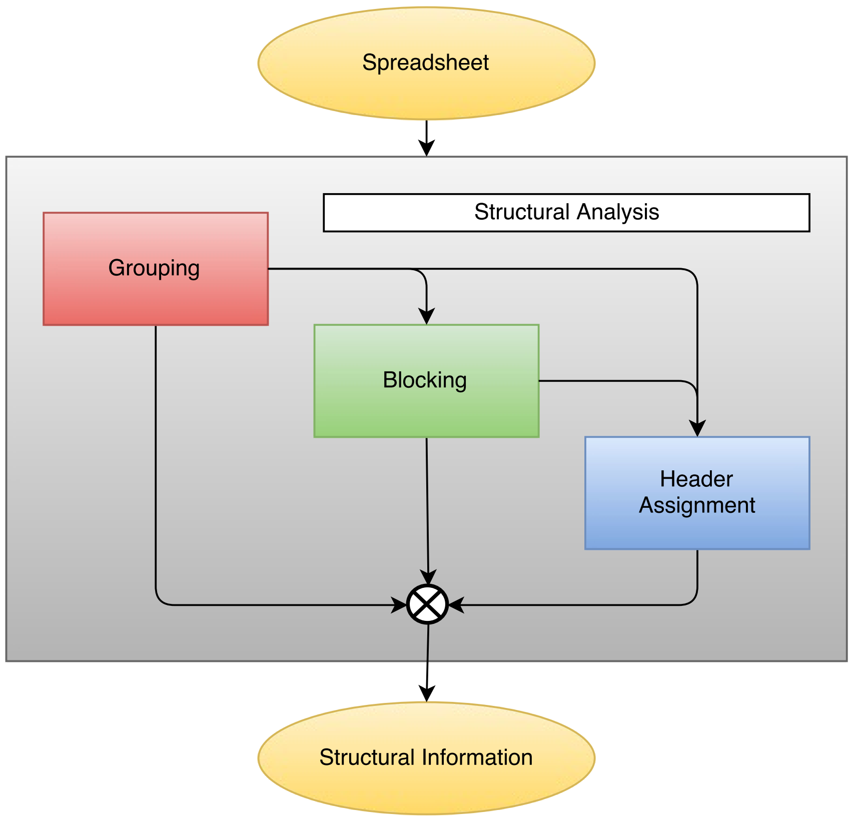


Figure 3.5: Workflow of the novel Structural Analysis process. Structural data of a provided spreadsheet serves as input data of the process and is supplied to each of its sub-processes. The sub-processes Grouping, Blocking, and Header Assignment supply their results to the successive follow-up processes. In addition, these sub-results are aggregated and supplied as result of the overall process.

We employ a bottom-up analysis approach whereby follow-up steps in the process chain utilize results of previous analysis steps. However, each step within the process chain also provides grouping information which is aggregated and serves as combined result of our grouping analysis. Figure 3.6 illustrates the process structure of our grouping approach, featuring the following three processing steps: *Type-based Grouping*, *Formula Group Partitioning*, and *Reference-based Grouping*.

Type-based Grouping

The Type-based Grouping step is employed to detect groups of cells that fulfil the same purpose. We utilize the cell type of each cell as criterion to form groups. In case of formula cells, the purpose of each cell is more strictly defined by its formula. Thus, in order to detect groups of formula cells fulfilling the same purpose, we need

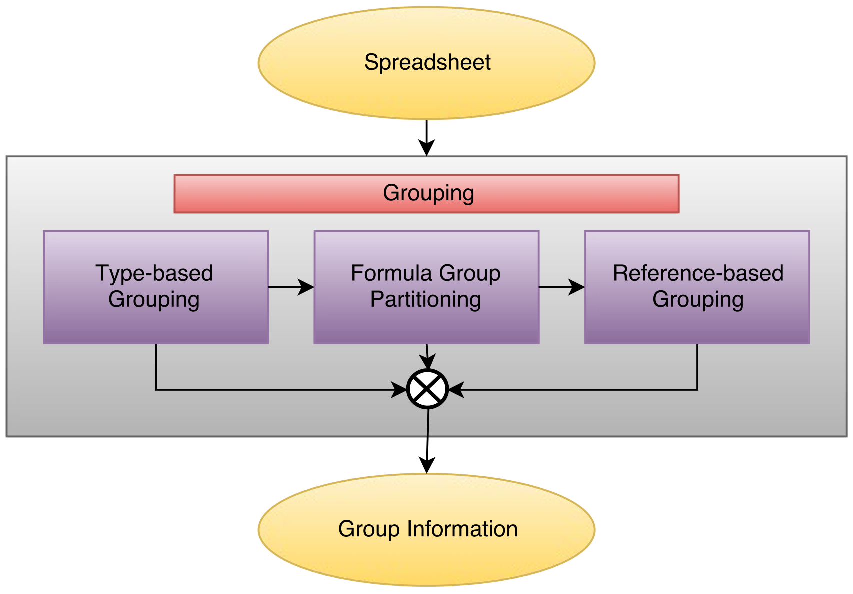


Figure 3.6: Workflow of novel Grouping process. Structural data provided by the spreadsheet serves as input for this process and is supplied to each step within the process chain. In addition, follow-up steps in the process chain utilize results of previous analysis steps. The steps Type-based Grouping, Formula Group Partitioning, and Reference-based Grouping are employed in the process. The results of each of those steps are aggregated to form the result of the Grouping process.

to match formula cells that contain the same formula. We strengthen the grouping criteria in this instance. Consequently, cells within a group are required to fulfil the following qualities:

- Each cell within the group features the same cell type.
- Each cell within the group is neighbour to at least one other member cell of the same group.
- For groups of formula cells, each cell within the group features the same formula in R1C1 representation.

By employing the R1C1-notation, the contained formula string remains independent of the position of the cell within the worksheet. This allows us to detect groups of formula cells that apply the same calculation to different input values. While type-based groups of other cell types may also be employed within further processing, the set of detected formula groups is the main result of this processing step. It represents the complete set of calculations employed within the provided spreadsheet.

Algorithm 1 describes the utilized grouping process. We consider a group G to be a set of cells $\{C\}$. Likewise, a map M is considered a set of mappings $(C \rightarrow G)$ from a cell C to a group G which also allows access to the set of mapped keys *M.keys*. Provided with a worksheet W, the algorithm determines the set of contained type-based groups *GroupSet* and the map, relating each cell of the worksheet to a matching group *CellGroupMap*. The function *determineTypebasedCellGroups*() in Line 1 iterates through each cell within the worksheet. For each cell, the function checks whether the provided cell has already been processed. To that end, the function verifies whether a mapping is already present in the *CellGroupMap*. If this is not the case, an empty group is initialized for the cell and the expansion of the group is started by a call to the function *expandGroup*(cell C, group G). Lastly, the new group is added to *CellGroupMap*, the set of known groups. The function *expandGroup*(cell C, group G) in Line 9 is a recursive function which first adds the cell to the provided group. Next, the function iterates the spatial neighbours of the cell. Neighbouring cells are accessed via the function *neighbours*(C) (see Definition 19). For each neighbouring cell, it is checked whether the neighbour is similar to the provided cell. Similarity checking is provided by the function *similar*(cell C, cell C_n) in Line 15. The checked similarity requirements follow the grouping requirements we stated beforehand. To check for similarity of formula cells, the R1C1-representation of the cells' formulas is accessed via calls to the function *R1C1formula*(cell C) in Line 18. If a similar neighbour is found, and the neighbouring cell is not yet part of any group, the group is expanded with the neighbour cell via call to *expandGroup*(cell C_n, group G).

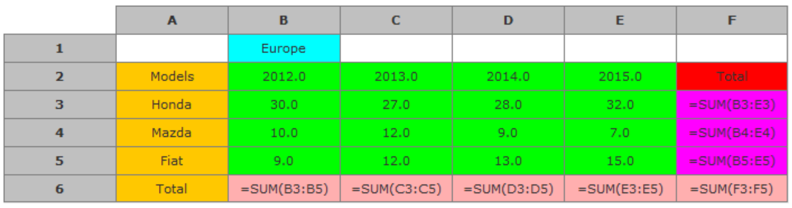
Figure 3.7 illustrates the result of our grouping algorithm when applied to the car sales example spreadsheet. Cells contained within the same group share the same background colour. The SUM-formulas in area B6:F6 were correctly matched in the light-red formula group. So were the formulas in area F3:F5. The groups in orange, cyan, and red indicate string groups. The green group indicates the only group containing numeric values within the spreadsheet. While the year dates in row 2 are used as headers within the example, type-based grouping matches them into the same numeric group as the input values of the spreadsheet in area B2:E5. This indicates a need for further processing.

Algorithm 1: DETERMINETYPEBASEDCELLGROUPS

Require: worksheet W

Ensure: set of groups in worksheet GroupSet; map relating cells to groups CellGroupMap

```
 1: procedure DETERMINETYPEBASEDCELLGROUPS
 2:     init GroupSet
 3:     init CellGroupMap
 4:     for all $C \in cells(W)$ do
 5:         if $C \notin$ CellGroupMap.keys then
 6:             init $G$
 7:             $expandGroup(C, G)$
 8:             GroupSet $\leftarrow$ GroupSet $\cup G$
 9: procedure EXPANDGROUP($C$, $G$)
10:     $G \leftarrow G \cup C$
11:     CellGroupMap $\leftarrow$ CellGroupMap $\cup (C \rightarrow G)$
12:     for all $C_n \in neighbours(C)$ do
13:         if $\{(C_n \notin$ CellGroupMap.keys$) \wedge similar(C, C_n)\}$ then
14:             $expandGroup(C_n, G)$
15: procedure SIMILAR($C$, $C_n$)
16:     if $type(C) \equiv type(C_n)$ then
17:         if $type(C) \equiv formula$ then
18:             return $(R1C1formula(C) \equiv R1C1formula(C_n))$
19:         else
20:             return $true$
21:     else
22:         return $false$
```



	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

Figure 3.7: Result of Type-based Grouping when applied to the car sales example. Cells indicated by the same background colour share the same type-based group. Groups indicated by the colours orange, cyan, and red are groups of string cells. The group indicated by the colour green is a group of numeric cells. The groups indicated by the colours purple and light-red are groups of formula cells that share the same formula in R1C1-representation.

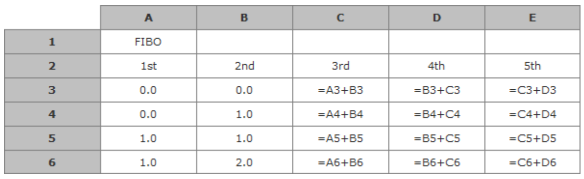
Formula Group Partitioning

The Formula Group Partitioning step is based on the results of the preceding Type-based Grouping. The preceding step provides formula groups which represent the calculations of the spreadsheet. However, those groups are solely defined by the individual cells contained within them. As recognized by our *1D is key* concept, this complicates further analysis steps. Hence, Formula Group Partitioning infers a set of area-based formula groups for each type-based formula group. Each area-based group is defined by a 1D-area within a worksheet. Partitioning of the reference-based group is processed such that each cell contained in the initial reference-based group is also contained within the areas of the resulting area-based groups. In addition, the partitioning process follows our *Concrete & Concise* concept. We keep the partition concrete, as all of the resulting area-based groups feature the same orientation. We keep the partition concise, as we choose the orientation that results in the least amount of area-based groups for the partition.

The type-based formula groups inferred for the car-sales example in Figure 3.7 already form one-dimensional areas. Hence, Formula Group Partitioning results in area-based groups each covering the same area. To illustrate the partitioning-effect of Formula Group Partitioning, Figure 3.8 presents another example. The illustrated worksheet depicts calculation of the Fibonacci series for different starting values. Each formula cell in the area C3:E6 features the same formula in R1C1-representation. To facilitate further processing, this area is partitioned into multiple 1D-areas. Following the *Concrete & Concise* principle, two valid options are available for partitioning: Horizontal partitioning leads to four horizontal groups, covering the areas C3:E3, C4:E4, C5:E5, and C6:E6. Vertical partitioning leads to three vertical groups, covering the areas C3:C6, D3:D6, and E3:E6. As vertical partitioning results in fewer resulting groups, this method is chosen by the process. The resulting partition is depicted in Figure 3.8c. Cyan, purple, and green cell backgrounds highlight the area-based groups resulting from the partitioning process.

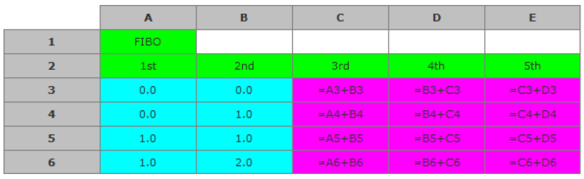
Reference-based Grouping

Like Type-based Grouping, the Reference-based Grouping process is employed to detect groups of cells that share the same purpose. Cells of a reference-based group



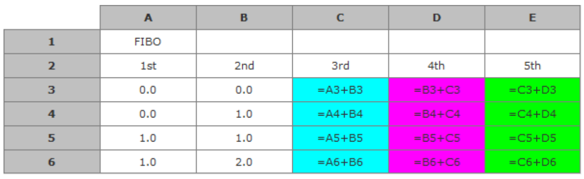
	A	B	C	D	E
1	FIBO				
2	1st	2nd	3rd	4th	5th
3	0.0	0.0	=A3+B3	=B3+C3	=C3+D3
4	0.0	1.0	=A4+B4	=B4+C4	=C4+D4
5	1.0	1.0	=A5+B5	=B5+C5	=C5+D5
6	1.0	2.0	=A6+B6	=B6+C6	=C6+D6

(a) Formula-view of the Fibonacci-calculation example.



	A	B	C	D	E
1	FIBO				
2	1st	2nd	3rd	4th	5th
3	0.0	0.0	=A3+B3	=B3+C3	=C3+D3
4	0.0	1.0	=A4+B4	=B4+C4	=C4+D4
5	1.0	1.0	=A5+B5	=B5+C5	=C5+D5
6	1.0	2.0	=A6+B6	=B6+C6	=C6+D6

(b) *Type-based Grouping* results.



	A	B	C	D	E
1	FIBO				
2	1st	2nd	3rd	4th	5th
3	0.0	0.0	=A3+B3	=B3+C3	=C3+D3
4	0.0	1.0	=A4+B4	=B4+C4	=C4+D4
5	1.0	1.0	=A5+B5	=B5+C5	=C5+D5
6	1.0	2.0	=A6+B6	=B6+C6	=C6+D6

(c) *Formula Group Partitioning* results.

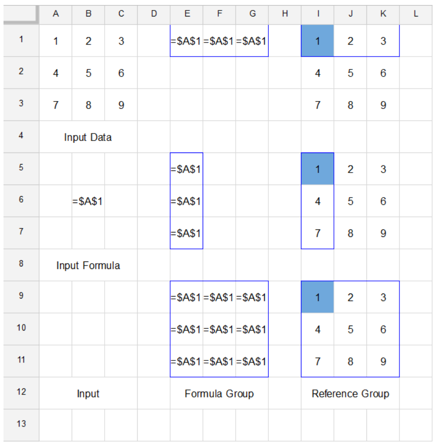
Figure 3.8: An example worksheet demonstrating the Formula Group Partitioning process. The table depicts the calculation of the Fibonacci series for various starting values. Cells of detected type-based and partitioned formula groups within the example share the same background-colour.

serve as input value for one of the calculations of the spreadsheet. To that end, the Reference-based Grouping process is based on the results of the preceding Formula Group Partitioning step. Formula Group Partitioning provides a set of formula groups fit for further processing. Reference-based Grouping iterates each entry in this set of groups. For each reference and area-reference contained in the formula of each group, the process attempts to determine either a single group, or a set of reference-based groups. The exact outcome depends on the type of the processed relation.

Cell references. Figure 3.9 illustrates the concept of Reference-based Grouping for cell-references. Each sub-figure follows the same structure. The input data for the example is provided in area A1:C3. The employed formula for the example is presented in cell B6. Columns E to G present the formula groups of the example. Formula groups are constructed by copying the base formula in either horizontal, vertical, or both orientations. Columns I to K represent the resulting reference groups within the input data. The blue border indicates the dimensions of the applied formula group for comparison. Cells with blue background-colour represent the resulting reference-based group. The reference-based groups resulting of the process depend on two key features: First, the relativity of the processed reference. Second, the orientation and extent of the processed formula group. In case of a fully-static reference, the resulting reference group always contains only 1 cell, independent of orientation and size of the formula group. This case is illustrated in Figure 3.9a. Figures 3.9b and 3.9c depict the results for column-static and row-static references. In case of a static row reference, horizontal and two-dimensional formula groups result in a horizontal reference group whereas a vertical formula group results in a 1-cell reference group. Likewise, in case of a static column reference, vertical and two-dimensional formula groups result in a vertical reference group whereas a horizontal formula group results in a 1-cell reference group. In case of an all-dynamic reference, the resulting group follows the form of the formula group. This case is illustrated in Figure 3.9d. A horizontal formula group results in a horizontal reference group. A vertical formula group results in a vertical reference group. A two-dimensional formula group would result in a two-dimensional reference group. However, the Formula Group Partitioning process only infers one-dimensional formula groups. Hence, no two-dimensional reference groups are created based on references.

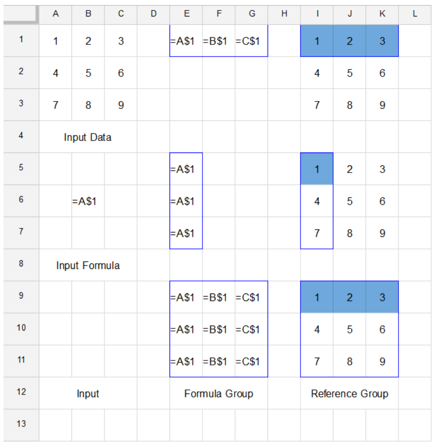
Area references. Reference-based Grouping for area-references poses a further problem. Depending on the dimensions of the referred area, different approaches are required: In case of a 0D-area, the area-reference targets a single cell. Consequently, the area-reference can be converted into a normal cell-reference, and the grouping process for cell-references can be employed. When handling 1D-areas, the grouping process is dependent on the orientation of the target area in relation to the orientation of the formula group that refers to the area. Handling of 2D-areas require a specific approach of their own.

1D area references in matching orientation. If the target area and the formula group share the same orientation (i.e. both are horizontal or both are vertical) the referred areas of each cell within the formula group overlap. These areas only differ by a translation of one cell in the orientation of the formula group. Thus, the resulting reference-based group can be assembled by aggregation of each of the individual area-references within the formula group. Figure 3.10 illustrates this process for



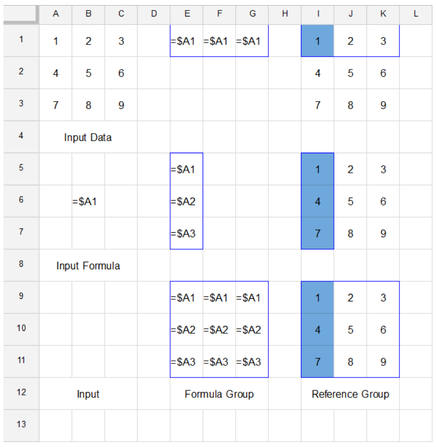
	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3		=$A$1	=$A$1	=$A$1		1	2	3	
2	4	5	6						4	5	6	
3	7	8	9						7	8	9	
4		Input Data										
5					=$A$1				1	2	3	
6		=$A$1			=$A$1				4	5	6	
7					=$A$1				7	8	9	
8		Input Formula										
9					=$A$1	=$A$1	=$A$1		1	2	3	
10					=$A$1	=$A$1	=$A$1		4	5	6	
11					=$A$1	=$A$1	=$A$1		7	8	9	
12		Input				Formula Group				Reference Group		
13												

(a) Static reference.



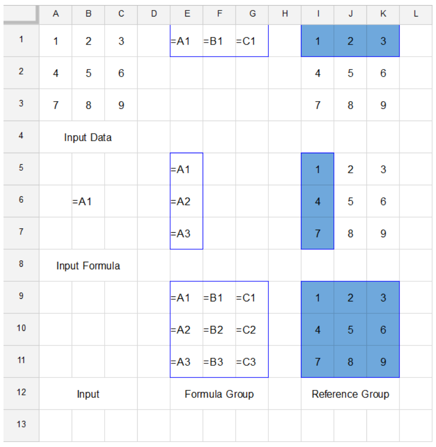
	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3		=A$1	=B$1	=C$1		1	2	3	
2	4	5	6						4	5	6	
3	7	8	9						7	8	9	
4		Input Data										
5					=A$1				1	2	3	
6		=A$1			=A$1				4	5	6	
7					=A$1				7	8	9	
8		Input Formula										
9					=A$1	=B$1	=C$1		1	2	3	
10					=A$1	=B$1	=C$1		4	5	6	
11					=A$1	=B$1	=C$1		7	8	9	
12		Input				Formula Group				Reference Group		
13												

(b) Static row reference.



	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3		=$A1	=$A1	=$A1		1	2	3	
2	4	5	6						4	5	6	
3	7	8	9						7	8	9	
4		Input Data										
5					=$A1				1	2	3	
6		=$A1			=$A2				4	5	6	
7					=$A3				7	8	9	
8		Input Formula										
9					=$A1	=$A1	=$A1		1	2	3	
10					=$A2	=$A2	=$A2		4	5	6	
11					=$A3	=$A3	=$A3		7	8	9	
12		Input				Formula Group				Reference Group		
13												

(c) Static column reference.



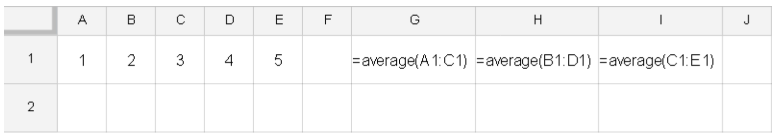
	A	B	C	D	E	F	G	H	I	J	K	L
1	1	2	3		=A1	=B1	=C1		1	2	3	
2	4	5	6						4	5	6	
3	7	8	9						7	8	9	
4		Input Data										
5					=A1				1	2	3	
6		=A1			=A2				4	5	6	
7					=A3				7	8	9	
8		Input Formula										
9					=A1	=B1	=C1		1	2	3	
10					=A2	=B2	=C2		4	5	6	
11					=A3	=B3	=C3		7	8	9	
12		Input				Formula Group				Reference Group		
13												

(d) Dynamic reference.

Figure 3.9: Concept of Reference-based Grouping when different expansion operations are applied to static, column-static, row-static, and dynamic cell-references within formulas.

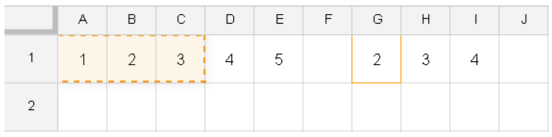
a horizontal formula group containing an area-reference to a horizontal area. The resulting reference-based group is one-dimensional in horizontal orientation as well, containing every cell that is referred to at least once by cells within the formula group. A formula group in vertical orientation referring to a 1D-area in vertical orientation leads to a similar result: a one-dimensional reference-based group in vertical orientation, containing every cell that is referred to at least once by cells within the formula group.

1D area references in non-matching orientation. If the orientation of the target area does not match the orientation of the formula group, the areas referred to by each individual cell of the formula group do not overlap. Rather, the areas align next to each other, forming a two-dimensional area. Figure 3.11a illustrates this



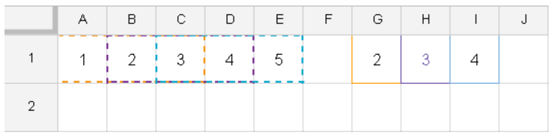
	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5		=average(A1:C1)	=average(B1:D1)	=average(C1:E1)	
2										

(a) Formula-view.



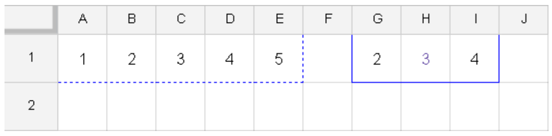
	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5		2	3	4	
2										

(b) Indication of one area-reference.



	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5		2	3	4	
2										

(c) Indication of all area-references.

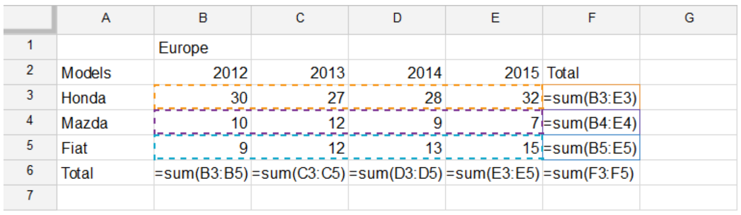


	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5		2	3	4	
2										

(d) Indication of the resulting area-reference.

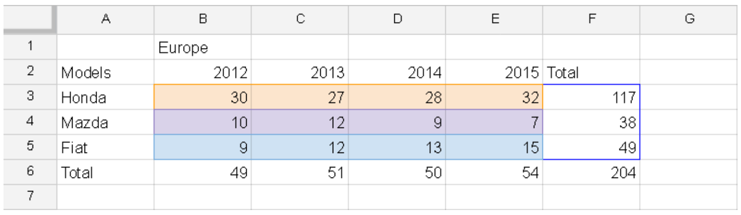
Figure 3.10: Concept of Reference-based Grouping for one-dimensional area-references in matching orientation.

scenario within the car-sales example worksheet. Area-references of formula group in area F3:F5 align next to each other and form the referred area B3:E5. The resulting 2D-area could already be employed as result for this grouping operation. However, our analysis process follows the *1D is key* principle. Thus, we partition the resulting 2D-area into a set of 1D-areas. Two different approaches can be employed for this partition operation. The first approach utilizes the areas referred to by each individual area-reference within the formula group as basis for the partition. Consequently, the operation results in a set of one-dimensional areas which are in opposite orientation than the referring formula group. This result option is depicted in Figure 3.11b. While this partition option is not chosen for our Reference-based Grouping approach, this result will be applied in the successive Formula Group Reference Matching step. The second approach considers individual cells the target area as basis for the partition. Partition takes place as if the target area was made up of individual cell-references that are combined by binary operators within the formula. As a consequence, applying the Reference-Based Grouping process to a formula group containing the formula *=SUM(A1:C1)* leads to the same result as applying the process to a formula group containing the formula *=A1+B1+C1*. The operation results in a set of one-dimensional areas with the same dimensions as the initial formula group. This result is depicted in Figure 3.11c. Following the *Concrete & Concise* guideline, this interpretation is chosen for our analysis approach, as it leads to consistent results for formulas that fulfil the same intent independent of the applied formula-format.



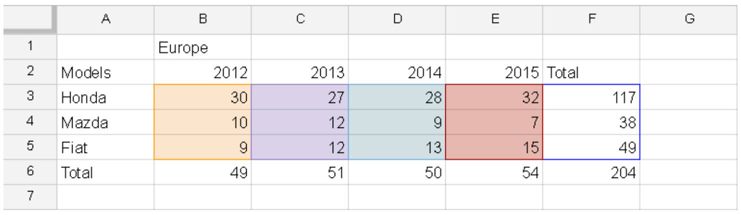
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B5)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							

(a) Indication of area-references.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

(b) Area-based result.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

(c) Area-element-based result.

Figure 3.11: Result of Reference-based Grouping for one-dimensional area-references in non-matching orientation. Each of the cells within the vertical formula group in area F3:F5 refers to the horizontal area left to it. The group as a whole refers to the area B3:E5. The resulting set of 1D reference-based groups depends on the applied area-partition. Area-based partition utilizes the areas provided by each individual reference of the formula group. Area-element-based partition aligns with results of single-cell references.

2D area references. Reference-based Grouping of a two-dimensional area-reference is not directly supported in our analysis process. Following our *1D is key* principle, we partition two-dimensional areas into a set of one-dimensional target areas before further processing. Each area within the resulting set is then applied to our analysis-process for 1D-areas. The orientation of the partitioning-operation is chosen with regard to further processing. We consider our analysis approach for 1D groups in non-matching orientation to be more meaningful than the analysis approach for matching orientation. Groups in matching orientation can be interpreted as a sliding window which applies the same calculation on groups which differ by only one cell, whereas groups in non-matching orientation can be interpreted as a sliding window which applies the same calculation on entirely different groups. Based on previous experience, we argue that spreadsheet users are more likely to apply calculations following the second principle. Consequently, in case of a horizontal formula group, the 2D-area is partitioned into a set of vertical 1D-areas. In case of a vertical formula group, the 2D-area is partitioned into as set of horizontal 1D-areas.

Algorithm 2 describes the process to detect reference-based groups in a worksheet. For purpose of this algorithm, the following prerequisites need to be fulfilled: Partitioned formula groups provide access to references and area-references of the group formula. To that end, the functions *references*(group G) and *areaReferences*(group G) return the references and area-references of a specific group. They are of the form G $\rightarrow$ $\{$*reference*$\}$ and G $\rightarrow$ $\{$*area-reference*$\}$, whereby *reference* and *area-reference* denote the references and area-references contained in the formula of the group. Moreover, partitioned formula groups and reference-based groups are both expressions of area-position-based groups. Such groups are defined by their area-position within the spreadsheet. The worksheet of the area-position defines the worksheet of the group. The area of the area-position defines the area of the group within the worksheet. This area position may be accessed by the function *areaPos*(area-position-based group G) of the form G $\rightarrow$ *area-position.* Lastly, a new reference-based group can be initialized, using an existing area-position as parameter.

The initial function of the algorithm, *determineReferencebasedCellGroups*(), initializes the data structures and iterates the provided set of partitioned formula groups. For each group within the set, the function iterates the sets of references and area references of the provided group and initializes new reference-based groups via the functions *initReferenceGroup*(group G, reference R) and *initReferenceGroup*(group G, area reference R_a). The initialized reference-based groups are registered within the data structures via a call to the function *registerNewGroup*(partitioned formula group G, reference-based group G_r) in Line 36. The

Algorithm 2: DETERMINEREFERENCEBASEDCELLGROUPS

```
Require: set of partitioned formula groups in worksheet FormulaGroups
Ensure: set of reference-based groups in worksheet GroupSet; map relating cells to
    reference-based groups CellGroupMap;
 1: procedure determineReferencebasedCellGroups(FormulaGroups)
 2:     init GroupSet
 3:     init CellGroupMap
 4:     for all $G \in$ FormulaGroups do
 5:         for all $R \in references(G)$ do
 6:             $G_r \leftarrow initReferenceGroup(G, R)$
 7:             $registerNewGroup(G, G_r)$
 8:         for all $R_a \in areaReferences(G)$ do
 9:             for all $G_r \in initReferenceGroups(G, R_a)$ do
10:                 $registerNewGroup(G, G_r)$
11: procedure initReferenceGroup($G$, $R$)
12:     init RefGroup
13:     $areaPos$(RefGroup) $\leftarrow processAreaPosForRef(G, R)$
14:     return RefGroup
15: procedure initReferenceGroups($G$, $R_a$)
16:     init RefGroups
17:     if $isZeroDimensional(targetArea(R_a))$ then
18:         RefGroups $\leftarrow$ RefGroups $\cup$ $initReferenceGroup(G, convertToRef(R_a))$
19:     else
20:         init TargetAreas
21:         if $isOneDimensional(targetArea(R_a))$ then
22:             TargetAreas $\leftarrow targetArea(R_a)$
23:         else
24:             TargetAreas $\leftarrow partitionArea(targetArea(R_a), G)$
25:         for all $A \in$ TargetAreas do
26:             if $orientation(A) \equiv orientation(G)$ then
27:                 init RefGroup
28:                 $areaPos$(RefGroup) $\leftarrow processAreaPosForMatchingAreaRef(G, A)$
29:                 RefGroups $\leftarrow$ RefGroups $\cup$ RefGroup
30:             else
31:                 for all $P_{targetArea} \in processAreaPosForPerpAreaRef(G, A)$ do
32:                     init RefGroup
33:                     $areaPos$(RefGroup) $\leftarrow P_{targetarea}$
34:                     RefGroups $\leftarrow$ RefGroups $\cup$ RefGroup
35:     return RefGroups
36: procedure registerNewGroup($G$, $G_r$)
37:     GroupSet $\leftarrow$ GroupSet $\cup$ $G_r$
38:     for all $C \in cells(G_r)$ do
39:         CellGroupMap $\leftarrow$ CellGroupMap $\cup$ $(C \rightarrow G)$
```

function *initReferenceGroup*(group G, reference R) in Line 11 initializes a new reference-based group based on the provided reference. The area position of the new group is determined by the function *processAreaPosForRef*(G, R) in Line 13. This function implements the grouping process for cell references illustrated in Figure 3.9. The function *initReferenceGroups*(group G, area reference R_a) in Line 15 initializes a set of new reference-based groups based on the provided area reference. It starts with initializing RefGroups, the set of resulting reference-based groups. The content of RefGroups depends on the dimensions of the target area of the provided area reference. The target area is accessed via the function *targetArea*(area reference R_a). In case of a 0D area reference, the area reference is converted into a cell reference, and the cell reference is further processed via the function *processAreaPosForRef*(G, R). The resulting reference-based group is added to RefGroups. In the case of either a 1D or 2D area reference, TargetAreas, a set of areas, is initialized. This set is filled depending on the dimensions of the area reference:

- *1D area reference.* The area of the area reference is added to TargetAreas.
- *2D area reference.* The referred area is partitioned into multiple 1D-areas via a call to the function *partitionArea*(area A, partitioned formula group G) in Line 24. Partitioning by the function follows the previously stated criteria. The resulting areas are added to TargetAreas.

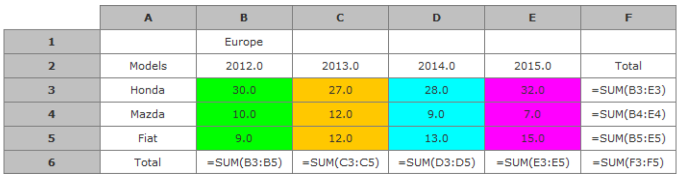
The process continues by iterating TargetAreas, the resulting set of 1D-areas. Each inferred area A is further processed depending on its orientation in relation to the orientation of the referring formula group G. Two different cases are possible:

- *A shares the orientation with G.* In this case, the process for areas in matching orientation is applied by the function *processAreaPosForMatchingAreaRef*(G, A) in Line 28. This function follows the guidelines illustrated in Figure 3.10 and returns a new area position. This resulting area position is used to initialize a new reference-based group. The resulting group is added to RefGroups, the result set of the function.
- *A does not share the orientation with G.* In this case, the process for areas in non-matching orientation is applied via the function *processAreaPosForPerpAreaRef*(G, A) in Line 31. This function follows the guidelines outlined in

Figure 3.11, and returns a set of inferred area positions. The area positions of this set are used to initialize a set of new reference-based groups. These groups are added to RefGroups, the set of resulting reference-based groups of the function.

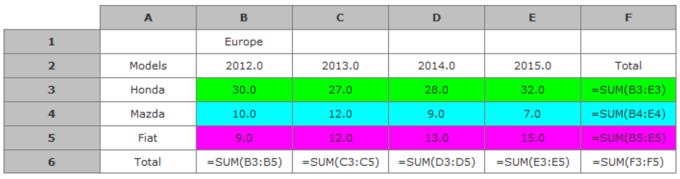
Lastly, RefGroups, the set of inferred reference groups for all cases, is returned as result of the function.

Figure 3.12 illustrates the result of Reference-based Grouping when applied to the car sales example. In contrast to Type-based Grouping and Formula Group Partitioning, each cell within a worksheet can be part of multiple reference-based groups. The vertical groups, indicated in Figure 3.12a are the result of processing of the formula group in column F. The horizontal reference-based groups illustrated in Figure 3.12b are the result of processing the formula group in row 6.



	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(a) Inferred vertical reference-based groups.



	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(b) Inferred horizontal reference-based groups.

Figure 3.12: Result of Reference-based Grouping when applied to the car sales example spreadsheet. Formulas in column F lead to the illustrated vertical reference groups. Horizontal reference groups are inferred based on formulas in row 6.

Extended Grouping

The analysis process we presented up until now is fit to detect type-based groups, partitioned formula groups, and reference-based groups within a worksheet. However, to enable the ensuing processing steps, further information about these detected groups is required. To that end, the basic grouping approach is extended by two additional processing steps:

- *Formula Group Reference Matching* provides information about references between partitioned formula groups.
- *Reference Group Merging* condenses results of the reference-based grouping process.

Figure 3.13 illustrates the relation of processing steps within the extended grouping process.

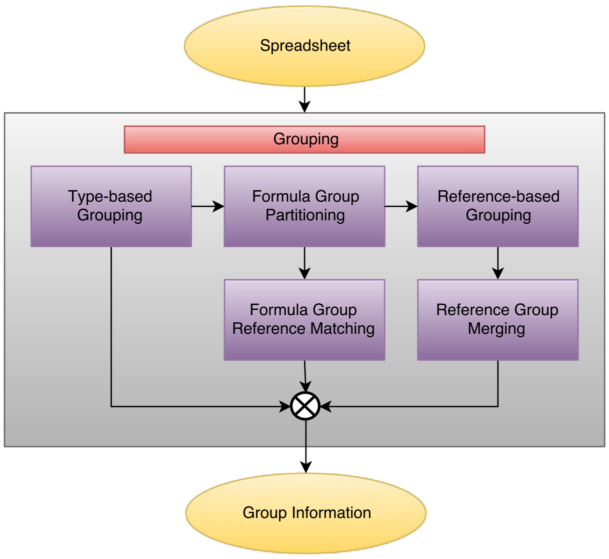


Figure 3.13: Workflow of the extended Grouping process. Results of the Formula Group Partitioning and Reference-based Grouping processes are refined by the introduction of additional processing steps: Partitioned Formula Group Matching and Reference Group Merging.

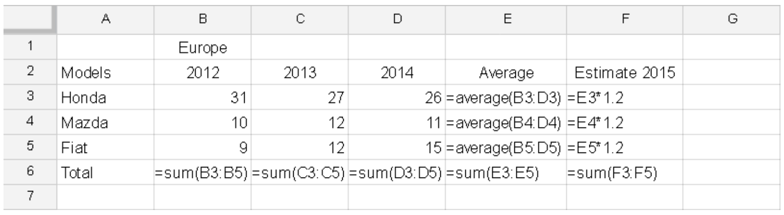
Formula Group Reference Matching. References between formula cells build a network which can be used to analyse connectivity of cells within a spreadsheet. Detected formula groups do not yet offer such connectivity information. The Formula Group Matching process aims to establish such relations between detected partitioned formula groups.

To that end, Formula Group Matching compares the areas that are referenced within a formula group to the areas encompassed by other formula groups. If the areas match, a relational link between the two formula groups is established. Reference-based Grouping already utilizes approaches to generate cell groups based on references of formula groups. The same basic grouping methods can be applied for Formula Group Matching. The working principle of these grouping processes are illustrated in figures 3.9, 3.10, and 3.11.

The reference-processing strategy employed by Formula Group Matching only differs in case of area references in non-matching orientation. In this scenario, the group reference can be processed into a 2D area within the worksheet. Following our *1D is key* principle, this area needs to be partitioned into a set of 1D areas. While the previous Reference-based Grouping employs area-element-based partitioning, Formula Group Matching employs area-based partitioning instead. This allows for preservation of the areas of each individual contributing area-reference. This is in line with the way spreadsheets are usually organized.

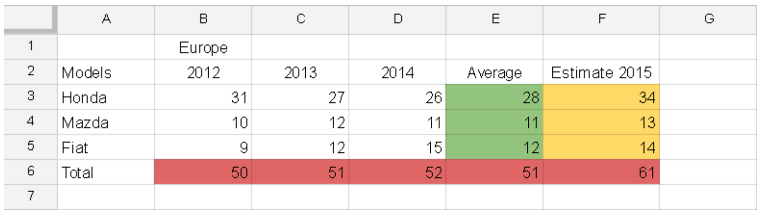
Figure 3.14 provides an example of the Formula Group Matching process when applied to a worksheet. The illustrated worksheet is an expansion of the previous car-sales example. Figure 3.14a provides the formula view of the updated worksheet. Column E calculates the average number of car sales per year. Column F offers an estimate for the next year, based on the calculated average. Figure 3.14b illustrates the detected formula groups for the worksheet. Figures 3.14e, 3.14d, and 3.14c present the calculated reference areas for each formula group. Whenever a reference area of a formula group matches the area of another formula group, a relational link between the two formula groups is established. Thus, the formula group in area B6:F6 refers to the formula groups in areas E3:E5 and F3:F5, as the areas of those groups match one of the reference areas of the group in B6:F6. Likewise, the formula group in area F3:F5 refers to the group encompassing area E3:E5. The reference areas of the formula group in area E3:E5 do not match any other formula group.

Reference Group Merging. The result of reference-based grouping is a set of 1-dimensional reference-based groups within a worksheet in both horizontal and vertical orientation. If multiple different formula groups refer to data within the same area of a worksheet, cells within this area may be part of multiple reference-based groups of different dimensions and orientations at the same time. However, to facilitate the following analysis processes, we require a more *Concrete & Concise* summary of the reference-based groups within a worksheet. In particular, we pose the requirement that each cell may be a member of only one horizontal, and only one vertical reference-based group. To fulfil this requirement, we merge overlapping groups of the same orientation.



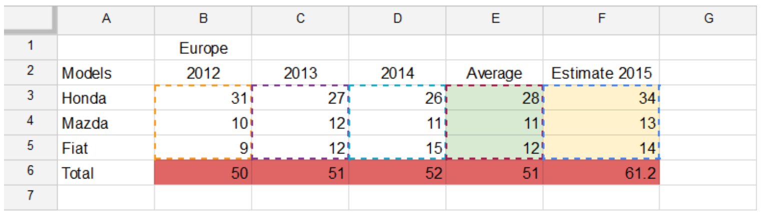
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	Average	Estimate 2015	
3	Honda	31	27	26	=average(B3:D3)	=E3*1.2	
4	Mazda	10	12	11	=average(B4:D4)	=E4*1.2	
5	Fiat	9	12	15	=average(B5:D5)	=E5*1.2	
6	Total	=sum(B3:B5)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							

(a) Formula view of example worksheet.



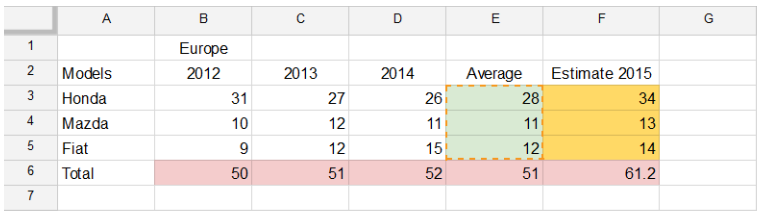
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	Average	Estimate 2015	
3	Honda	31	27	26	28	34	
4	Mazda	10	12	11	11	13	
5	Fiat	9	12	15	12	14	
6	Total	50	51	52	51	61	
7							

(b) Partitioned formula groups.



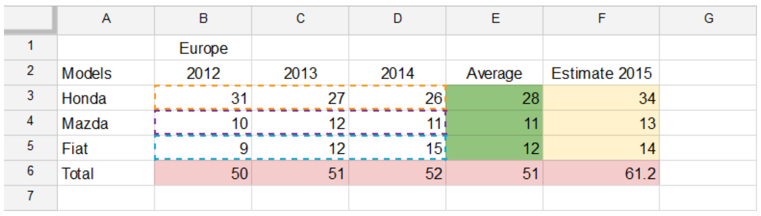
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	Average	Estimate 2015	
3	Honda	31	27	26	28	34	
4	Mazda	10	12	11	11	13	
5	Fiat	9	12	15	12	14	
6	Total	50	51	52	51	61.2	
7							

(c) References of formula group B6:F6.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	Average	Estimate 2015	
3	Honda	31	27	26	28	34	
4	Mazda	10	12	11	11	13	
5	Fiat	9	12	15	12	14	
6	Total	50	51	52	51	61.2	
7							

(d) References of formula group F3:F5.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	Average	Estimate 2015	
3	Honda	31	27	26	28	34	
4	Mazda	10	12	11	11	13	
5	Fiat	9	12	15	12	14	
6	Total	50	51	52	51	61.2	
7							

(e) References of formula group E3:E5.

Figure 3.14: Example of Formula Group Matching. The example worksheet is an extension of the car-sales example. Formula groups are indicated by a coloured background. Inferred reference areas for each group are indicated by hatched borders. Formula groups are related to other formula groups matching any reference area. Formula group B6:F6 refers to formula groups E3:E5 and F3:F5. Formula group F3:F5 refers to formula group E3:E5.

Algorithm 3 describes the process to condense reference-based groups in a worksheet. For the purpose of this algorithm, the following prerequisites need to be fulfilled: (1) Reference-based formula groups provide access to the orientation of the group. To that end, the functions *orientation*(group G) returns the orientation group G. It is of the form group $\Rightarrow$ *Orientation*, whereby *Orientation* is either *horizontal* or *vertical.* (2) The function *overlap*(reference-based group G, reference-based group G_2) provides information as to whether two provided reference-based groups overlap. It is of the form (G, G_2) $\rightarrow$ {*Boolean*} and defined as

$$\mathit{overlap}(G, G_2) = \begin{cases} \text{true} & \text{if } \{G \cap G_2 \neq \emptyset\} \\ \text{false} & \text{otherwise} \end{cases}.$$

(3) The function *mergeGroups*(G, G_2) allows to merge two reference-based groups. It is of the form (G, G_2) $\rightarrow$ {*reference-based group*}, whereby *reference-based group* denotes the resulting group which encompasses the union of the areas of group G and group G_2.

GroupSet, the set of all detected reference-based groups, serves as input for the algorithm. Groups within GroupSet are expected to be either zero-dimensional or one-dimensional in horizontal or vertical orientation. The algorithm iterates each reference-based group G and compares it to each other reference-based group G_2 in GroupSet. If the groups G and G_2 are not the same, but share the same orientation and overlap according to the *overlap*(G, G_2) function, both groups are removed from the set of known groups. Instead, a new reference-based group is initialized via a call to the function *mergeGroups*(G, G_2). This resulting group is consequently added to the set of merged reference-based groups.

Algorithm 3: MERGEREFERENCEBASEDCELLGROUPS

```
Require: set of reference-based groups in worksheet GroupSet
Ensure: set of merged reference-based groups in worksheet GroupSet;
1: procedure MERGEREFERENCEBASEDCELLGROUPS(GroupSet)
2:     for all $G \in$ GroupSet do
3:         for all $G_2 \in$ GroupSet do
4:             if $G \neq G_2 \wedge orientation(G) \equiv orientation(G_2) \wedge overlaps(G, G_2)$ then
5:                 GroupSet $\leftarrow$ GroupSet $\setminus (\{G\} \cup \{G_2\})$
6:                 GroupSet $\leftarrow$ GroupSet $\cup$ $mergeGroups(G, G_2)$
```

3.3.2 Blocking

Blocking allows for the detection of calculation blocks within a worksheet. As defined in Section 3.2, calculation blocks are concise areas within a worksheet which aggregate related formula groups and input groups. Detected blocks allow for assignment of matching labels to the block, and therefore to each individual group within the block. In perspective of the previous processing steps, this definition can be adapted as follows: Calculation blocks are concise areas within a worksheet which consist of neighbouring partitioned formula groups and reference-based groups. Partitioned formula groups combine neighbouring formula cells sharing the same R1C1-representation. Reference-based groups contain cells which are pointed to by a reference of a formula group. Detected blocks serve as basis for the successive Header Assignment process, assigning labels to matching calculation blocks. Consequently, each block fulfils the following requirements:

1. A block defines an area within a worksheet.
2. A block consists of a collection of area-based groups: partitioned formula groups or reference-based groups.
3. Each group within a block features a certain degree of proximity to at least one other group within the block.
4. Blocks may not contain potential label and header cells.

The blocking process we employ is based on the expansion of individual blocks. Block expansion follows the work-flow illustrated in Figure 3.15. A block is initialized for a specific area-based group. Successively, we check for presence of an eligible neighbour group. If one such neighbour is present, the validity of the possible expansion is checked. If expansion to the detected group results in a valid block, the expansion is conducted. If no neighbour for valid block expansion can be detected, expansion of the block concludes. This process is continued until each partitioned formula group and reference-based group of the worksheet is part of a block. While the presented blocking work-flow leads to sound results, a number of challenges remain to be discussed:

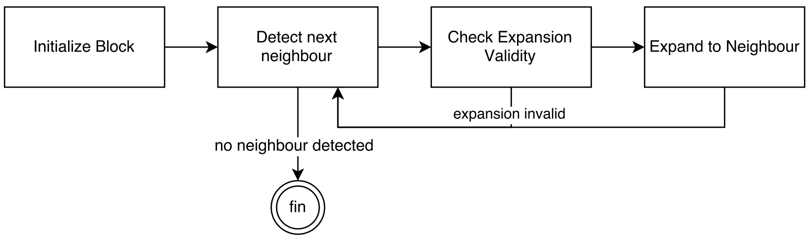


Figure 3.15: Workflow of Block Expansion process. A block is initialized for a specific area-based group. Successively, we check for presence of an eligible neighbour group. If one such neighbour is present, the validity of the possible expansion is checked. If expansion to the detected group results in a valid block, the expansion is conducted. If no neighbour for valid block expansion can be detected, expansion of the block concludes.

1. *How to determine whether a group is part of a block?* Blocks should only be initialized for groups that are not yet part of another block. To that end, we need a method to determine whether a group is already part of a block. Area-comparison can be employed for this check. As stated in our requirements, a block defines an area within a worksheet. Likewise, both partitioned formula groups and reference-based groups are defined by an area within the worksheet. Consequently, to determine whether a group is participating in a block, we can determine whether the area of the group is entirely located within the area of the block. If this is the case, the group in question is part of the block.

2. *How to detect eligible neighbours?* Eligible neighbours for a block are cell groups that are not yet part of the block and contain at least one cell that is in the neighbourhood of the current block. Neighbourhood implies location at either a vertical or horizontal border of the current block. Depending on which distance is acceptable for valid neighbours, a gap between the border of the block and the neighbouring group may be allowed. Figure 3.16 illustrates the neighbourhood-criteria for various employed distances. To confirm neighbourhood, at least one cell of the group to be checked needs to lie within one of the colour-indicated border areas.

3. *How to expand a block towards a neighbouring group?* If a valid neighbouring group is detected, we need to expand the block to contain the neighbour in question. As a block defines an area within a worksheet, we need to expand the area in such a way that the area of the neighbouring group is contained.

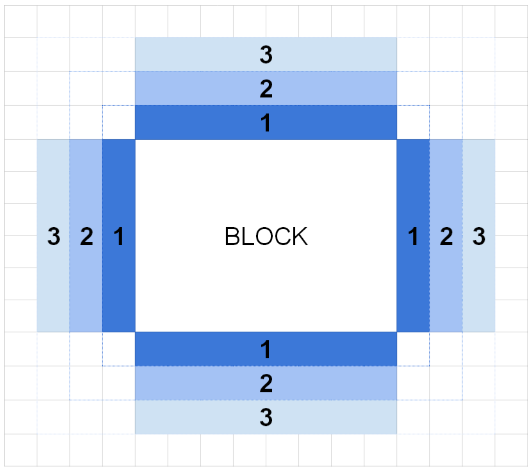
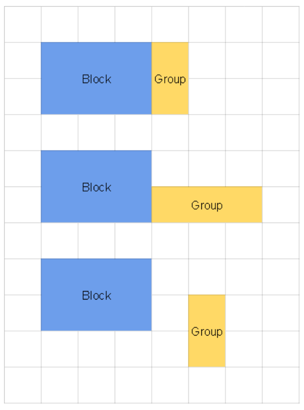


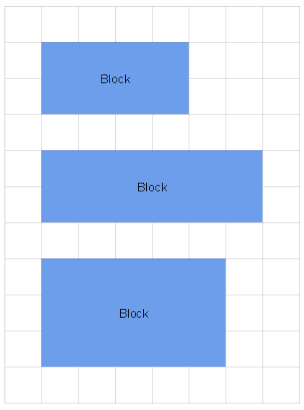
Figure 3.16: Concept of detection-areas for eligible neighbours of a block. Expansion of blocks is only allowed in direct vertical or horizontal neighbourhood based on the current dimensions of the block. Coloured areas at the border of the block indicate the cells which are checked for neighbouring groups. Depending on the distance which is employed to consider neighbouring groups as valid, the detection areas allow for greater expansion perpendicular to the border of the block.

Figure 3.17 illustrates block expansion for various scenarios. In each scenario, the neighbouring group is already deemed a valid expansion neighbour for the block. The blocks are expanded in such a way as to integrate the area of the neighbouring groups into the area of the block. Depending on the case, this may introduce cells into the block which were not part of the initial expansion group. This implies the need to check for validity of block expansions.

4. *How to check the validity of an expansion?* For each detected eligible neighbour, we need to process the validity of the possible expansion. Based on our requirements, each cell within the neighbouring group already is a valid expansion cell. However, as illustrated in Figure 3.17, expansion of a block may also introduce cells into the area of the block that are not part of the neighbouring group. Consequently, we need to check for validity of these cells as well. Only one requirement concerns validity of cells within a group: Labels and headers are not part of blocks. Thus, for each cell that is newly introduced to a block, we need to check whether those cells contain any labels or potential header cells. If this is the case, the block expansion is deemed invalid.



(a) Before expansion.



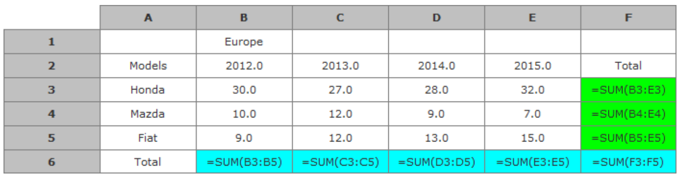
(b) After expansion.

Figure 3.17: Concept of Block Expansion in different neighbourhood scenarios. Valid neighbourhood for the neighbour groups in each scenario is already determined. Blocks are always expanded in such a way as to integrate the neighbour group into the block area. Additional cells that were not originally part of the neighbouring group may be introduced to the block as a result.

Figure 3.18 illustrates the result of the Blocking process, when applied to the car-sales example worksheet. Figures 3.18a, 3.18b, and 3.18c depict which groups serve as input for the blocking process. The resulting block is illustrated in Figure 3.18d. It encompasses each of the participating formula groups and reference-based groups. Surrounding blank cells, numeric cells and strings are not part of the block. The dimensions and position of the resulting block are only depending on the individual groups which form the block. The exact initial group selection and sequence of groups to expand the block do not influence the result.

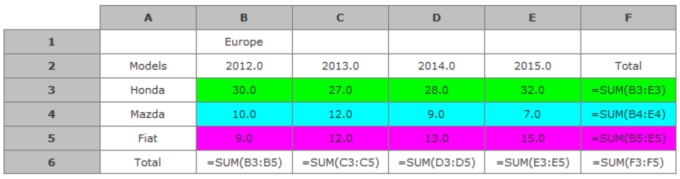
3.3.3 Header Assignment

Header Assignment is the final step of our novel structural analysis process. Header Assignment intends to detect header relations within a worksheet. A header in the worksheet scenario is a string or value cell which supplies contextual information for one or multiple related cells. Unlike formula-based relations, headers rely on proximity and positioning of the header cell in comparison with affected cells to establish the connection. Header-relations are required for certain spreadsheet QA techniques (e.g. unit-detection). Header information can also be applied to infer spreadsheet smells, as we demonstrate in Section 6.2. Most importantly, refactoring operations



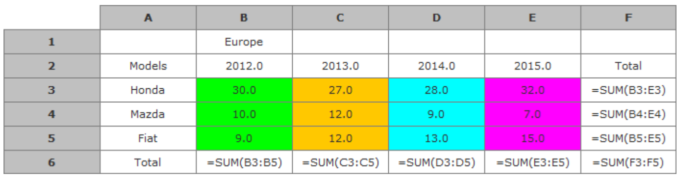
	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(a) Partitioned formula groups of worksheet.



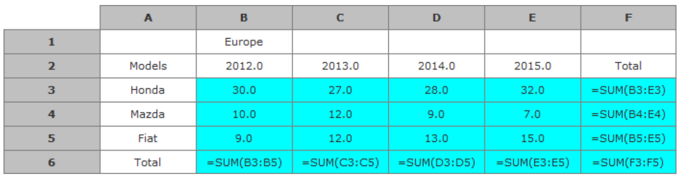
	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(b) Horizontal reference-based groups of worksheet.



	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(c) Vertical reference-based groups of worksheet.



	A	B	C	D	E	F
1		Europe				
2	Models	2012.0	2013.0	2014.0	2015.0	Total
3	Honda	30.0	27.0	28.0	32.0	=SUM(B3:E3)
4	Mazda	10.0	12.0	9.0	7.0	=SUM(B4:E4)
5	Fiat	9.0	12.0	13.0	15.0	=SUM(B5:E5)
6	Total	=SUM(B3:B5)	=SUM(C3:C5)	=SUM(D3:D5)	=SUM(E3:E5)	=SUM(F3:F5)

(d) Blocks of worksheet.

Figure 3.18: Blocking applied to the car-sales example worksheet. The resulting block in area B3:F6 encompasses every formula groups and reference-based group of the worksheet. The bottom edge of the area is defined by the formula group B6:F6. This group also expands the area to its leftmost and rightmost limits. The top border may be established by either the formula group F3:F5, one of the vertical reference-based groups or the topmost horizontal reference-based group.

to improve spreadsheet quality often require relocation of certain structures within the spreadsheet. To remain consistency, related headers need to be relocated as well.

The Header Assignment step works in accordance with our established analysis process. Consequently, instead of detecting general header relations, we attempt to detect headers based on our established blocking-concept. Headers provide contextual information for individual columns and rows of detected blocks. Hence, header cells are positioned in a way that implies a hierarchical relation to the column or row in question. Specific positioning depends on the writing system applied by the spreadsheet's author. In LTR systems, row headers are located next to the left border of a block. In RTL systems, row headers are located next to the right border of a block. Column headers are always positioned above the related block. With exception of blank cells, no other cells are allowed to be positioned between the edge of a block and a related header cell. Moreover, header cells may be affected by other header cells, forming a hierarchy of headers. In case of column headers, overlying header cells are positioned above subsequent headers. In case of row-headers, overlying header cells are positioned next to subsequent headers depending on the writing system. In LTR systems, overlying header cells are positioned left of, in RTL systems, overlying header cells are positioned right of subsequent headers. Lastly, a group of header cells on the same hierarchy level may be affected by an overlying header next to it. For instance, a layer of row headers may have a header cell above the layer which supplies context to the entire layer. As our evaluation data was created using the LTR writing system, our analysis approach expects row headers, higher-level row headers and layer-headers of column-header layers on the left side of the affected entity.

In summary, headers fulfil the following requirements:

- String cells and value cells that are not referred to by any formula are regarded as possible header cells.
- Valid header cells are located either above or left of the borders of a block.
- Only blank cells and other header cells may be positioned between a header cell and a related block.
- Headers may be organized in multiple layers.
- Lowest-level headers are related to either columns or rows of a block.
- Headers of higher layers affect subsequent headers in lower layers.
- Layer-headers are higher-level headers that are placed next to a header layer, affecting each header cell within the layer.

Based on these criteria, we propose the *Header Assignment* process illustrated in Figure 3.19 to identify block-based headers within a worksheet. The process as a whole requires access to the results of previous processing steps like identified blocks and established formula-relations. The work-flow of the process is separated into three successive steps. *Layer Detection* identifies potential header layers for blocks within a worksheet. *Header Propagation* establishes relations between header layers, lowest-level headers and the affected blocks. *Layer-Header Detection* identifies headers that affect an established header-layer of a block. Each step within the *Header Assignment* process supplies its results to successive processing steps.

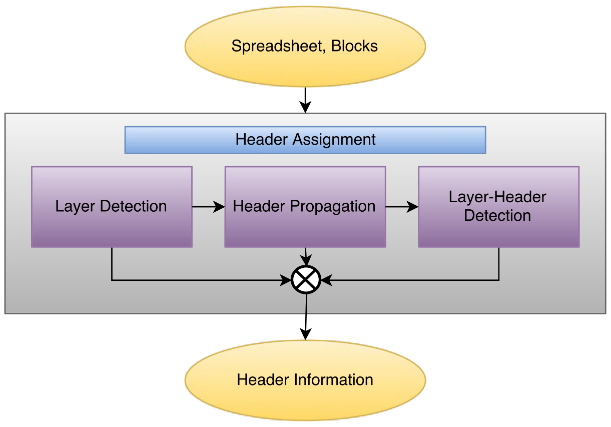


Figure 3.19: Workflow of Header Assignment process. Layer Detection identifies potential header layers for each block in the worksheet. Header Propagation establishes header-relations between different header layers, as well as between header layers and the related block. Layer-Header Detection identifiers higher-level headers situated next to a specific header layer.

Layer Detection

Layer detection detects layers of potential header cells in relation to a specific block. A header-layer is a set of cells that may provide header-information for a block. Consequently, a header layer may only contain either potential header cells in accordance to our requirements or empty cells. Each block may feature two different sets of header-layers: a set of layers of column-headers above the individual block, and a set of layers of row-headers, left of the individual block. Each header-layer shares the same dimensions as the border of the block it affects. A layer of column-headers is a horizontal group of cells which is as wide as the block it relates to. A layer of row-headers is a vertical group of cells which is as high as the block it relates to. Only blank cells and other header cells may be positioned between a header cell and a related block. Likewise, only valid header-layers may be positioned between a header-layer and a related block.

Figure 3.20 illustrates the working principle of Layer Detection. For each block, a set of successive layers of column-headers and row-headers is detected. To that end, an area matching the size of the border of the block is projected upwards and left of the block. If any potential header cells are contained in the projected area, a header-layer is initialized for this area. Header-layers are ordered based on the distance to the related block. This allows correct dependency propagation between different layers regarding the same block.

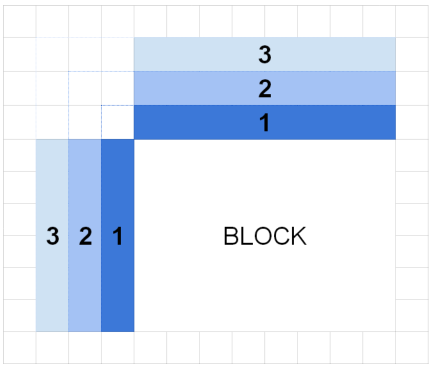


Figure 3.20: Detection-areas for header-layers of a block. Header-layers are detected via continuous projection of the top and left border-areas of a block. Detected layers are ordered ascending by the distance of the layer to the related block.

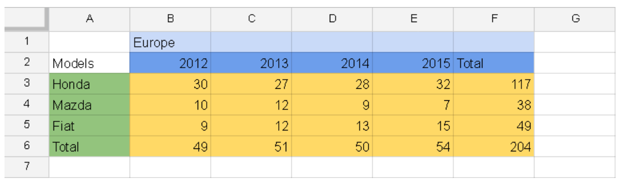
Based on these guidelines, Algorithm 4 describes the Header Detection process for a specific orientation of a specific Block B. The process first initializes the list to organize detected header-layers, LayerList. Consequently, the variable layerArea is initialized by a call to function *borderArea*(B) in Line 3. For column-header detection, this function returns the area of the top border of block B. For row-header detection, this function returns the area of the left border of Block B. The process continues by projecting the initialized area onto the worksheet by calling the function *traverseArea*(layerArea). In case of column-header detection, this function moves the provided area upwards by one cell. In case of row-header-detection, the function moves the provided area one cell to the left. The algorithm proceeds by checking whether the current area describes a valid header layer by a call to the function *isValidLayer*(layerArea) in Line 6. This function returns true if the area contains at least one potential header cell and no invalid cells (e.g. formulas). If a valid layer area was detected, a new layer is initialized by call to the function *intializeLayer*(layerArea) and the new layer is added to the list of detected layers

LayerList. This process repeats until a check by the function *isValidArea*(layerArea) in Line 8 fails. This function checks whether the current area has either reached the borders of the worksheet, or contains any cells that are not allowed to be positioned between header-layers and related blocks.

Algorithm 4: DETECTHEADERLAYERS

```
Require: block $B$
Ensure: ordered list of header-layers LayerList;
1: procedure DETECTHEADERLAYERS
2:     init LayerList
3:     layerArea ← $borderArea(B)$
4:     do
5:         layerArea ← $traverseArea$(layerArea)
6:         if $isValidLayer$(layerArea) then
7:             LayerList ← LayerList ∪ $intializeLayer$(layerArea)
8:     while $isValidArea$(layerArea)
```

Figure 3.21 illustrates the result of Layer Detection when applied to the car-sales example worksheet. Previous analysis revealed one calculation block in area B3:F6. Layer Detection results in a set of two layers of column-headers for this block. The first layer is located in area B2:F2. The second layer is located in area B1:F1. Analysis of the block also results in one layer of row-headers in area A3:A6. The header in cell A2 cannot be inferred by Layer Detection, as it is not located directly above or next to the related block. Instead, the Layer-Header Detection step will establish the relation for this header.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

Figure 3.21: Example of Layer Detection result, when applied to the car-sales example. Detected areas within the worksheet are indicated by background colour. The identified block of the worksheet is indicated yellow. Layer Detection detects two layers of column-headers for the block. The first layer of column-headers in row 2 is indicated by dark-blue, the second layer in row 1 by a light-blue colour. The one detected layer of row-headers in column A is indicated by green colour.

Header Propagation

Header Propagation intends to establish header relations between a set of header-layers, as well as between header-layers and the related block. Header-relations are ordered top-down: headers further away from the block influence underlying headers located between the position of the header and the block. Moreover, if no neighbouring higher-level header is present, an individual higher-level header may expand its influence and relate to a set of neighbouring lower-level headers. If no underlying header is present, a higher-level header may relate to a column or row of a block. However, even if no neighbouring header is present, an individual higher-level header may not relate to any column or row other than the column or row underlying the header. Lowest-level headers always relate exclusively to the underlying column or row of the block.

Figure 3.22 illustrates the working principles of header propagation for column-headers. Top-level headers influence underlying header cells. Lowest-level header cells are related to columns of the underlying block. If a higher-level header has no header cell next to it, it influences the underlying lower-level header as well as additional lower-level headers in direct neighbourhood. However, this influence expansion only works for right-hand neighbours in the case of column headers, and bottom-side neighbours in the case of row headers. Not every header and every column or row of a block is guaranteed to relate to a header cell. Lastly, headers in higher-level layers may directly relate to columns or rows of a block, if no lower-level headers are present in-between. However, even if no neighbouring higher-level header is present, the header may not relate to neighbouring columns or rows of the block. Each header may directly influence only one column or row of a block.

The process of layer-propagation is applied top-down based on the results of the layer-detection step. The process iterates each detected layer beginning with the layer with the greatest distance to the block. Each header cell contained in the current layer is propagated downward the successive layers. To that end, the process iterates every remaining layer and checks whether the successive layer contains a valid header cell at the same position, if that is the case, header-relations are established following the previously described guidelines. If no lower-hierarchy header

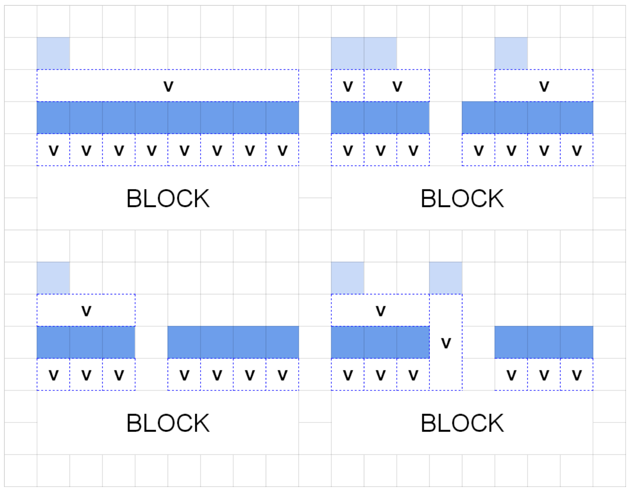


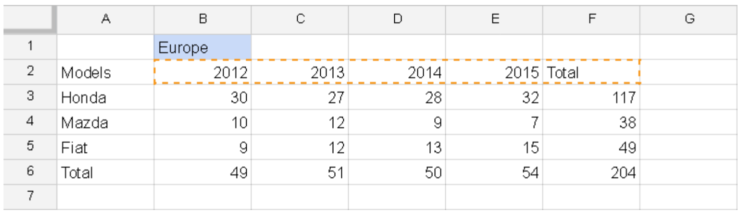
Figure 3.22: Working principles of Header Propagation. Present header cells within layers are indicated by background colour. Header cells in the first header-layer are indicated in dark-blue. Header cells in the second header-layer are indicated in light-blue. A **v** surrounded by a hatched border indicates connection between a header cell and an underlying headers or columns of the block.

could be detected, a connection between the current header and the underlying column or row of the block is established.

Figure 3.23 presents the results of Header Propagation when applied to the car sales example. The header in cell B1 is a higher-level header and affects the entirety of the header-layer in area B2:F2. Each individual header in header-layer B2:F2 refers to the underlying column of the block in B3:F6. Likewise, each individual header in the layer of row-headers A3:A6 refers to the row next to it in the block B3:F6.

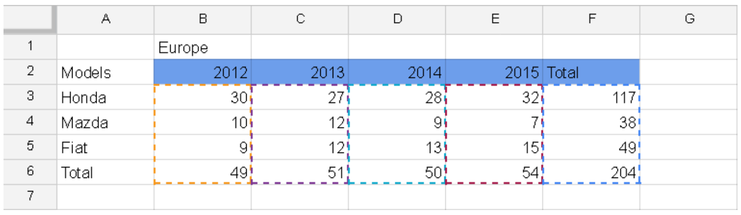
Layer-Header Detection

Layer-Header Detection is the last step of the Header Assignment process and aims at detecting headers next to header-layers. In many spreadsheets, the column- or row-headers are themselves related to an additional header next to the header group. To detect layer-headers of a block, the step iterates each header-layer of the block. In case of column-header layers, the process checks cells left of the current layer. In case of row-header layers, the process checks cells above the current layer. If any of the checked cells is a valid header cell with regard to our requirements, and the header is not yet part of any other header relation, Layer-Header Detection es-



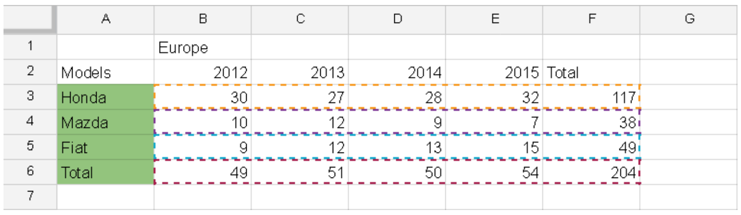
	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

(a) Relations inferred for column-header layer in row 1.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

(b) Relations inferred for column-header layer in row 2.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

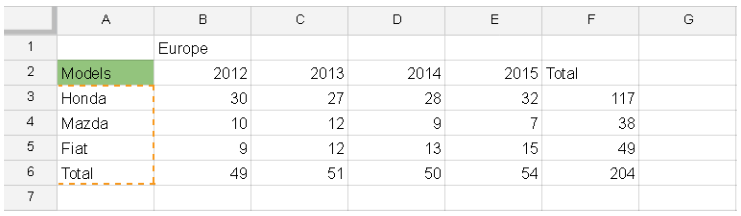
(c) Relations detected for row-header layer in column A.

Figure 3.23: Example for Header Propagation when applied to the car-sales worksheet. Each figure illustrates propagation of headers of one header-layer. The current layer is indicated by the same background-colour utilized during Layer Detection. Related areas are indicated by hatched borders of various colours.

tablishes a connection between the detected cell and the related header-layer. In cases where a layer-header may be applicable to both, a layer of row-headers and a layer of column-headers, we assign the layer-header to the layer of row-headers by default. We base this decision on our expectations regarding table-layout outlined in Section 3.2. Following this expectation, column headers act as descriptive headers for different characteristics of a set of instances. Row headers act as identifiers for single instances within the set. Consequently, a conflicting layer-header should be applied as descriptive header for the layer of instance headers. Figure 3.4 illustrates this behaviour. The cell A1 could be applied as layer-header for both the layer of descriptive headers in area B1:E1 and the layer of instance headers in area A2:A7. As we expect the header in A1 to provide descriptive information for the underlying column, we assign it to the underlying layer of row-headers. We acknowledge that our expectation regarding table-layout may not be met in all practical instances.

The results of a more elaborate blocking-analysis could provide insight in the preferred orientation within the current block. We point out ideas for such blocking-approaches in Chapter 8. This orientation information could be applied to resolve undecided cases, instead of relying on a default decision.

Figure 3.24 presents the result of Layer-Header Detection when applied to the car sales example. The string cell at Position A2 could be applied to both, the layer of column headers to the right of it, as well as to the layer of row headers below it. Following our established guideline, we conclude that the cell acts as layer-header for the layer of row-headers in area A3:A6.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

Figure 3.24: Example for Layer-Header Detection when applied to the car-sales worksheet. The identified layer-header in cell A2 is indicated by green background colour. The layer-header relates to the layer of row-headers in area A3:A6 which was identified by Layer Detection.

4. Evaluation

In this section, we present and discuss the methodology and results of our evaluation. In specific, we introduce the spreadsheet corpora our evaluation is based on. We establish which criteria evaluation data has to meet and how filtering guarantees these criteria for the provided corpora. In addition, we offer a comparison of the resulting filtered corpora, based on a number of quality metrics. We establish the evaluation methodology entertained in our work. Lastly, we state and discuss the results of our evaluation.

4.1 Evaluation Corpora

To evaluate the performance of our static analysis mechanisms, we analyse how effective our test implementation performs when applied to real-world spreadsheets. Thanks to the effort of various contributors, such input data is available in form of so-called evaluation corpora: collections of spreadsheets dedicated to scientific research and analysis. For our evaluation, we considered two spreadsheet corpora which are publicly available for research purposes. In this section, we present these corpora of interest, describe the necessary filtering-options we applied to them, and compare them by a number of performance characteristics. Our analysis operations were applied to spreadsheets which passed the previously mentioned filtering process and to non-empty worksheets contained in those spreadsheets.

EUSES
EUSES [Fisher and Rothermel, 2005] was the first extensive collection of spreadsheets which was accessible to the scientific community. Since suitable alternatives to the EUSES corpus were not available at the time this corpus was published, it was used as basis for evaluation of spreadsheet-related research projects for the

greater part of the last decade. The corpus contains about 4,500 spreadsheets which were mainly aggregated by a web search. Some contributions from numerous smaller evaluation- and validation-sets are contained as well. An in-depth analysis of various performance and structure metrics for the corpus is also available [2005]. Spreadsheets within the corpus are categorized into a set of labelled sub-categories. This allows researchers to choose which subset of spreadsheets best fits the needs of their evaluation processes.

ENRON

ENRON was introduced by Felienne Hermanns and Emerson Murphy-Hill [2015] as alternative to the popular *EUSES* corpus. The 15,770 Spreadsheets contained in this new corpus were extracted from the *Enron Email Archive*, a collection of emails containing inter-concern communications of the *ENRON* corporation, made publicly available during legal investigation of the corporation. As this corpus is still relatively new, no in-practice performance data is available yet. However, extensive analysis of the corpus published by Hermans and Murphy-Hill suggests that the corpus features similar characteristics to those of the EUSES corpus. An in-depth comparison of both corpora, conducted by Jansen [2015], came to the same conclusion.

4.1.1 Filtering of Spreadsheet Corpora

In this chapter, we seek to evaluate the performance-characteristics of the static analysis methods we proposed in our work. For this purpose, we apply our analysis methods to a collection of input files and statistically evaluate the aggregated analysis results. However, the employed spreadsheets originate from publicly available collections. As those collections do not follow any quality criteria, input files may not fulfil the functional requirements of our static analysis methods. Consequently, some form of filtering is required beforehand to ensure that utilized input data fulfils the following requirements:

1. Input files are correctly read by the employed Apache POI [2015] library.
2. Input files are correctly handled by our application, i.e. do not contain non-supported spreadsheet-components.

3. Input files contain at least one formula cell.

To ensure compliance of input data to these requirements, we introduced the filtering mechanism outlined in Figure 4.1. Input-data-collections are applied to a multi-tier filtering-process which ensures for passing files that each of the requirements is fulfilled. The first preprocessing step ensures that passing input files are readable by the employed POI library components. The second preprocessing step ensures that passing input files are readable and processable by our application. Thus, passing files do not contain any non-supported spreadsheet functionalities. Lastly, the third preprocessing step ensures that passing input files contain at least one formula cell. In addition, preprocessing operations may fail due to unforeseen reasons like insufficient memory capacity. In such cases, the relevant input file is moved to a designated quarantine folder to allow for further problem handling.

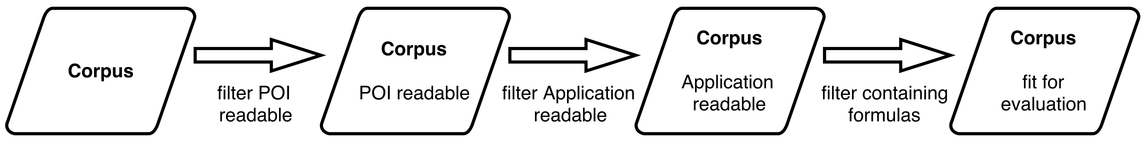


Figure 4.1: Multi-step filtering mechanism for spreadsheet corpora.

Table 4.1 illustrates the results of the filtering-process after application to the EUSES and ENRON corpora. In total, the EUSES corpus contains 4,495 files separated in 22 folders. Of those files, 323 spreadsheets cannot be opened by required POI components. 155 files contain non-supported spreadsheet components. 2,354 files do not feature any formula cells. 4 files could not be analysed correctly due to other issues. 1,659 spreadsheets are fit for evaluation. The ENRON corpus features 15,929 files contained in a single folder. All of those files, can be opened by required POI components. 2,527 files contain non-supported spreadsheet components. 6,656 files do not feature any formula cells. 55 files could not be analysed correctly due to other issues. The remaining 6,691 spreadsheets are fit for evaluation.

	Total	POI	Application	Formula	Other	Fit for evaluation
EUSES	4,495	323	155	2,354	4	1,659
ENRON	15,929	0	2,527	6,656	55	6,691

Table 4.1: Result of the filtering-process applied to the EUSES and ENRON corpora. Numbers indicate total and remaining file numbers, as well as how many files were blocked by each filtering step.

4.1.2 Corpora Comparison

We performed a set of two initial analysis operations on the spreadsheets contained in the filtered corpora. Those operations were chosen to provide a set of relevant performance metrics. These metrics are the basis of our comparison of both evaluation corpora. Moreover, these metrics allow us to adapt the follow-up evaluation steps applied to our static analysis methods, as well as to formulate performance-predictions for those methods. The first analysis operation gathered performance metrics on a spreadsheet basis. This operation was applied to each spreadsheet within the filtered corpora. Based on this operation, we gathered size and performance metrics for each spreadsheet. The second analysis operation gathered performance metrics on a worksheet-basis, and was applied to each non-empty worksheet within the filtered corpora. We present and compare the results of this operation based on the following focus-categories:

- Cell count: Number of cells for each cell type per worksheet.
- Reference count: Number of incoming and outgoing references for each cell type and based on occurrence-numbers per worksheet.
- Relation status: Number of input -, intermediate -, output -, and isolated cells per worksheet.
- Unique values: Number of unique occurring numbers, strings, and formulas based on occurrence per worksheet.
- Areas: Number of area references in formulas per worksheet.

The resulting metric data was aggregated for both corpora and is provided in table-form. In addition, we offer a discussion for the spreadsheet-based analysis result, as well as for each focus-category of the worksheet-based analysis results.

The first analysis operation focussed on spreadsheet-based evaluation. We analysed processing times for each spreadsheet, as well as how many worksheets and non-empty worksheets are contained within spreadsheets of the filtered corpora. Table 4.2 presents the results of the spreadsheet-based evaluation. Analysis of the EUSES corpus required about 43s of processing-time per file and indicates an average of 4.3 worksheets contained in each spreadsheet. On average, 3.5 of these worksheets are non-empty, for a total of 5,732 non-empty worksheets. Analysis of the filtered ENRON corpus on average required 89s of processing-time per file. This is about double the time required for the average file in the EUSES corpus. Worksheet sizes between ENRON and EUSES are similar, with 4.6 worksheets contained in each spreadsheet on average, and 3.7 of these worksheets being non-empty in the ENRON corpus. In total, the preprocessed ENRON corpus contains 24,330 non-empty worksheets.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES						
# worksheets	1	2	3	4.3	5	55
# non-empty worksheets	1	1	2	3.5	4	55
runtime in seconds	0.3	24.5	33.5	42.1	68.0	84.0
ENRON						
# worksheets	1	2	3	4.6	4	86
# non-empty worksheets	1	1	2	3.7	4	85
runtime in seconds	0.4	38.3	82.4	89.8	141.3	179.7

Table 4.2: Spreadsheet-based evaluation of filtered EUSES and ENRON corpora.

The first focus-category of the worksheet-based evaluation regards cell counts per worksheet. We tallied how many relevant cells in total, as well as how many cells of each specific cell type are contained within each non-empty worksheet. Table 4.3 presents the evaluation results of this category. For the EUSES corpus, worksheets between the 1st and 3rd quartile contain between 127 and 755 cells in total, including counted blank cells. This lies beneath the average of 1,775 cells, which is inflated due to a low number of worksheets containing significantly more than the average amount of cells. About 73 % of the cells contained in the filtered corpus are blank. For a blank cell to be counted in our evaluation, it has to either contain style information, or be referred to by any formula cell. 11 % of cells contain strings, followed by 10 % numeric values and 4.5 % formulas. Boolean and error cells are hardly used at all. In general, worksheets contained within the filtered corpus seem to contain mostly labelled numeric and string data, with a relatively few formulas for processing and analysis-purposes of this data. Worksheets of the ENRON corpus within the 1st and 3rd quartile contain between 240 and 1,589 cells in total. This lies beneath the average of 3,840 cells. Both values, however, indicate that the average worksheet within the ENRON corpus is twice the size of worksheets in the EUSES corpus. Type distribution of cells is similar within both corpora. About 72 % of the cells contained in the filtered ENRON corpus are blank. 9 % of cells contain strings, 12 % contain numeric values and 6 % formulas. Boolean and error cells are rarely used.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES						
total	1	128	303	1,755	755	289,430
blank	0	22	118	1,297	384	288,872
boolean	0	0	0	0.21	0	547
error	0	0	0	0.05	0	194
formula	0	0	1	82	52	20,490
numeric	0	3	30	176	105	121,582
string	0	21	44	200	88	206,053
ENRON						
total	1	240	537	3,840	1,589	468,278
blank	0	45	181	2,775	742	450,029
boolean	0	0	0	0.15	0	1,770
error	0	0	0	0.90	0	5,897
formula	0	2	36	242	103	141,664
numeric	0	9	53	464	258	182,448
string	0	21	46	358	116	115,599

Table 4.3: Worksheet-based cell count evaluation of filtered EUSES and ENRON corpora.

The second focus-category of the worksheet-based evaluation regards reference counts per worksheet. We tallied how many cells are referred to in total, as well as how many cells of each cell type are referred to in each worksheet. Moreover, we counted how many cells within a given worksheet are referred to once, twice, and three times. Similarly, we tallied how many references are contained within formulas of each worksheet in total, and how many of those references are targeting specific cell types. Moreover, we counted how many area references are contained in formulas of each worksheet. The results of this evaluation are presented in Table 4.4.

Worksheets within the filtered EUSES corpus contain a low number of references (median: 6 references per worksheet), and are only sparsely referenced themselves (median: 30 references per worksheet). The average numbers (184 and 526 respectively) of these indicators are, again, inflated by a low number of worksheets contributing a significant amount of references. With an average of 381 cells per worksheet, a surprising high amount of blank cells is target of references. This is

		Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES							
incoming	total	0	0	30	526	136	262,144
	blank	0	0	0	381	15	261,839
	boolean	0	0	0	0.19	0	459
	error	0	0	0	0.001	0	4
	formula	0	0	1	55	26	20,489
	numeric	0	0	2	79	38	40,546
	string	0	0	0	11	0	16,845
	referenced once	0	0	16	315	76	262,144
	referenced twice	0	0	0	51	11	12,606
	referenced thrice	0	0	0	12	0	12,526
outgoing	total	0	0	6	185	67	88,804
	blank	0	0	0	49	0	57,900
	boolean	0	0	0	0.42	0	943
	error	0	0	0	0.001	0	4
	formula	0	0	0	66	20	19,804
	numeric	0	0	0	59	15	23,938
	string	0	0	0	10	0	11,100
	area	0	0	1	20	9	9,173
ENRON							
incoming	total	0	11	104	1,052	343	230,415
	blank	0	0	2	698	48	131,422
	boolean	0	0	0	0.001	0	27
	error	0	0	0	0.024	0	211
	formula	0	0	17	146	91	130,676
	numeric	0	0	24	196	116	118,151
	string	0	0	0	13	0	14,334
	referenced once	0	5	48	797	118	156,675
	referenced twice	0	0	9	155	99	118,151
	referenced thrice	0	0	0	45	0	34,404
outgoing	total	0	0	21	490	144	204,218
	blank	0	0	0	125	1	97,746
	boolean	0	0	0	0.002	0	50
	error	0	0	0	0.007	0	1,688
	formula	0	0	2	163	53	158,137
	numeric	0	0	5	187	72	82,554
	string	0	0	0	14	0	23,790
	area	0	0	5	40	20	17,345

Table 4.4: Worksheet-based reference count evaluation of filtered EUSES and ENRON corpora. The category incoming counts the number of individual cells of specific types that are referred to by at least one formula. The category outgoing counts the number of formula-references which target cells of a specific type.

likely due to area references also containing blank cells in their areas and formulas referring to blank cells which are meant to be filled in by spreadsheet users as part of an input form. The first assumption is supported by the fact that a significantly lower average number of outgoing references (49) are targeting blank cells. In terms of non-empty cells, numeric cells are referred to the most (14 %), followed by formula cells (10 %). String cells are rarely referenced (2 %). References to error and Boolean cells are only contained in a small number of specific spreadsheets. 60 % of referenced cells are referred to only once. The rest of them are referred to multiple times. Worksheets within the filtered ENRON are about thrice as connected than worksheets in the EUSES corpus. The median for outgoing references lies at 21, and the median for incoming references lies at 104 references per worksheet. Average values are significantly higher. This is in line with previous observations. The trend of a high number of blank cells being target of incoming references continues in the ENRON reference. Also the distribution of incoming references to non-empty cells is similar. Numeric cells are referred to the most (18 %), followed by formula cells (14 %). String cells are rarely referenced (1 %). References to error and Boolean cells are only contained in a small number of specific spreadsheets. In comparison to EUSES' 60 %, 75 % of ENRON's referenced cells are referred to only once. The rest of them are referred to multiple times.

The third focus-category of the worksheet-based evaluation regards cell relation status counts per worksheet. We tallied how many cells within each worksheet can be categorized in the following categories:

- Input cells. Non-formula cells which are referred to at least once.
- Intermediate cells. Formula cells which are referred to at least once.
- Output cells. Formula cells which are not referred to.
- Isolated cells. Non-formula cells which are not referred to.

Table 4.5 presents the results of this evaluation. Even within the filtered EUSES corpus, more than a quarter of contained worksheets neither contain any references themselves, nor are target of referenced by external formulas. Even indicators between the median and 3rd quartile are significantly below the average of the respective values. In general, about 31 % of cells are concerned with any relations, whereas the remaining 69 % are isolated. 85 % of relation-relevant cells are input cells. 10 % contain intermediate calculations. About 5 % are formulas containing output values which are not further referenced. The average distribution numbers of relation-relevant cells within worksheets are similar between ENRON and EUSES. In general, about 30 % of cells are concerned with any relations, whereas the remaining 70 % are isolated. 80 % of relation-relevant cells are input cells. 12 % contain intermediate calculations. About 8 % are formulas containing output values which are not further referenced. However, in comparison with EUSES, a larger amount of worksheets contribute to the relation status averages, as 1st quartiles of the indicators for input and output cells feature values greater than 0, unlike the results of the ENRON dataset.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES						
input	0	0	18	473	88	262,144
intermediate	0	0	1	55	26	20,489
output	0	0	3	27	16	9,173
isolated	0	73	196	1,202	500	289,430
ENRON						
input	0	6	70	910	261	145,242
intermediate	0	0	17	146	91	130,676
output	0	1	7	96	26	27,504
isolated	0	94	308	2,692	971	468,270

Table 4.5: Worksheet-based relation status evaluation of filtered EUSES and ENRON corpora.

The fourth focus-category of the worksheet-based evaluation regards unique value counts per worksheet. For each worksheet, we tallied how many unique numeric and string values occur once, twice, three times, and in total. In addition, we counted how many unique formulas, based on the R1C1-representation of each formula, are contained in each worksheet. The results of this evaluation are presented in Table 4.6. Worksheets in the preprocessed EUSES corpus contain an average of 77 unique string and 69 unique numeric values. These indicators are significantly higher than the respective median values, due to an expected power-law distribution of worksheet sizes in general. However, this follows the trend, established by Table 4.3, indicating that worksheets contain on average more strings than numeric values. Occurrence frequencies for both string values and numerics are similar: About 80 %

of values only occur once, 10 % twice, 3 % thrice. Worksheets contain an average of 8 distinct formulas. Worksheets in the preprocessed ENRON corpus contain an average of 73 unique string and 134 unique numeric values. These indicators are significantly higher than the respective median values, due to an expected power-law distribution of worksheet sizes in general. However, while the indicators for string values are similar, the ENRON dataset includes significantly more unique numeric values than the EUSES corpus. Occurrence frequencies for both string values and numerics are similar to the EUSES results: About 80 % of values only occur once, 10 % twice, 3 % thrice. Worksheets contain an average of 15.5 distinct formulas. This is in line with our projections, as EUSES featured about eight and most size-related metrics are double in value when comparing EUSES and ENRON spreadsheets.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES						
numerics	0	2	17	69	54	16,833
numerics occurring once	0	0	11	54.68	41	15,969
numerics occurring twice	0	0	1	6.51	5	2,492
numerics occurring thrice	0	0	0	2.51	1	3,222
strings	0	17	33	78	61	27,767
strings occurring once	0	14	28	63.11	53	14,715
strings occurring twice	0	0	1	7.03	4	8,052
strings occurring thrice	0	0	0	2.60	1	5,526
formulas	0	0	2	8	8	960
ENRON						
numerics	0	4	25	135	79	50,966
numerics occurring once	0	2	16	104	54	38,783
numerics occurring twice	0	0	1	15	9	9,174
numerics occurring thrice	0	0	0	5	2	4,056
strings	0	15	32	7	62	25,133
strings occurring once	0	10	25	54	47	25,097
strings occurring twice	0	0	2	8	6	7,156
strings occurring thrice	0	0	0	3	2	2,340
formulas	0	1	4	16	10	730

Table 4.6: Worksheet-based unique value evaluation of filtered EUSES and ENRON corpora.

The last focus-category of the worksheet-based evaluation regards area counts per worksheet. For each worksheet, we tallied how many unique areas of the following area-categories are referred to in area-references of each worksheet.

- Single-cell: Areas encompassing only a single cell.

- Multi-column: Areas encompassing multiple columns, but only a single row.
- Multi-row: Areas encompassing only a single column, but multiple rows.
- Multi-dimensional: Areas encompassing multiple columns and multiple rows.

In addition, we counted how many references to each of the described area-types are contained in each worksheet. Lastly, we tallied how many unique areas are contained, in total, in area-references of each worksheet, as well as how many of those unique areas are referred to once and more than once. The results of this evaluation are presented in Table 4.7. Analysis of the results features a number of focal points:

- Analysis of the EUSES corpus shows that only 50 % of worksheets in the corpus contain any references. Most of the referred areas are referred to once. Merely 10 % of worksheets contain multiple references to the same area, for a total of about 5 % of all occurring areas. In comparison, 67 % of all worksheets in the ENRON corpus contain area references. About 95 % of the referred areas are referred to once. 6 % of worksheets contain multiple references to the same area, for a total of about 5 % of all occurring areas.
- In the EUSES corpus, multi-column areas appear to be more abundant than multi-row areas. However, the biggest worksheet already contributes about 17 % of all occurring multi-column areas. Moreover, multi-column areas only appear in 17 % of all worksheets, whereas multi-row areas appear in 44 %. Both area types are referred on average only slightly more than once. This is probably based on single instances, where a single area is repeatedly referred to many times, while most areas are referred to only once. In terms of usage, no generic favourite amongst users can be discerned for the EUSES corpus. In comparison, multi-row areas appear to be more abundant than multi-column areas in the ENRON corpus. Worksheets in general contain about 20 % more references to unique multi-row areas than to unique multi-column areas. Moreover, only 26 % of all worksheets contain references to any multi-column areas, whereas 64 % contain references to multi-row areas. Both area types are referred on average only slightly more than once. This

	Sum.	Median	Mean	Max	(>0)%
EUSES					
single-cell	1,046	0	0.18	500	1 %
multi-column	53,465	0	9.33	9,173	17 %
multi-row	34,624	0	6.04	792	44 %
multi-dimensional	632	0	0.11	40	4 %
single-cell references	1,046	0	0.18	500	1 %
multi-column references	59,815	0	10.44	9,173	17 %
multi-row references	44,268	0	7.72	792	44 %
multi-dimensional references	9,137	0	1.59	1,571	4 %
unique areas	89,767	1	15.66	9,173	50 %
unique areas #ref. =1	85,238	0	14.87	9,173	47 %
unique areas #ref. >1	4,529	0	0.79	144	10 %
ENRON					
single-cell	13,211	0	0.53	2,188	4 %
multi-column	360,894	0	14.63	8,784	26 %
multi-row	431,519	3	17.49	16,116	64 %
multi-dimensional	8,822	0	0.36	405	3 %
single-cell references	14,497	0	0.59	2,188	4 %
multi-column references	409,131	0	16.58	8,784	26 %
multi-row references	542,266	3	21.98	17,345	64 %
multi-dimensional references	35,806	0	1.45	1,488	3 %
unique areas	814,446	5	33.01	16,116	67 %
unique areas #ref. =1	770,706	4	31.24	16,116	67 %
unique areas #ref. >1	43,740	0	1.77	4,389	6 %

Table 4.7: Area references in formulas per worksheet of filtered EUSES and ENRON corpora. The table lists how often specific area types were contained in individual worksheets of a corpus, as well as how many references to those area types occur. Besides common statistical categories, the table features a column '(>0) %'. Values of this column indicate the percentage of worksheets that featured at least one area or reference of a specific type.

is probably based on single instances, where a single area is repeatedly referred to many times, while most areas are referred to only once. In terms of layout-preferences, multi-row references and coinciding table layouts seem to be favoured amongst the creators of the ENRON corpus.

- Multi-dimensional areas only appear in 4 % of all worksheets of the EUSES corpus. However, occurring multi-dimensional areas are referred to significantly more often than multi-row and multi-column areas. On average, each multi-dimensional area is referred to about 15 times. One likely explanation of this trend is infrequent usage of multi-dimensional areas as data storage. In such instances, multiple LOOKUP- and other analysis functions are required to work on the same dataset. In general, multi-dimensional areas are rare enough to be of no concern during most structural analysis approaches. However, if a worksheet contains multi-dimensional area references, those instances probably require adjusted analysis methods. In comparison, multi-dimensional areas only appear in 3 % of all worksheets of the ENRON corpus. Occurring multi-dimensional areas are referred to more often than multi-row and multi-column areas. On average, each multi-dimensional area is referred to about 4 times.

- Within the EUSES corpus, single-cell areas only occur in 1 % of all worksheets. One single worksheet contributes 50 % of those occurrences. For comparison, single areas occur in 4 % of worksheets of the ENRON corpus. Each individual area is on average referred to only slightly more than once. Thus, for structural analysis those instances of single-cell areas can be regarded as edge cases. Moreover, in most cases, those instances can be handled by converting the single-area reference into a usual cell reference during the analysis process.

Based on the analysed focal points, ENRON is comparable to the EUSES corpus. However, we did find some differences as well. Firstly, ENRON contains slightly more unique referred areas than ENRON, both in terms of average occurrence and in the number of worksheets containing any area reference. Secondly, while the occurrence of multi-column area references is similar between both corpora, ENRON contains significantly more multi-row area references. This is likely explained due to preferences in worksheet layouts. Spreadsheets of the ENRON corpus were created with the same business-context in mind, leading to a common preference of multi-row layouts employed in these spreadsheets. In contrast, the EUSES corpus contains a wide variety of spreadsheets fulfilling various purposes. Hence, no preferential spreadsheet layout could be established. Lastly, ENRON contains slightly more multi-dimensional area references than EUSES. However, each individual area is, on average, only referred to 4 times whereas EUSES' areas are referred to about 15 times on average. This is likely due to EUSES containing a collection of spreadsheets that are used as databases. Such spreadsheets contain expansive data-areas, allowing the application of multi-dimensional area references in LOOKUP- and similar analysis functions.

4.2 Evaluation Approach

In the following section, we describe the evaluation approach employed in this thesis. In particular, we determine the goal of our evaluation and derive the evaluation method we use in order to fulfil our evaluation goal. Lastly, we outline the process we apply to generate a sets of evaluation data for the EUSES and ENRON corpora.

4.2.1 Evaluation Goal

As main contribution of this thesis, we established a novel structural analysis process for spreadsheets. This process generates explicit structural information based on the inherent structure of a spreadsheet. Applicability of the analysis results depends on the quality of the inferred structure information. We argue that the quality of the detected high-level structures is an adequate indicator for the overall performance of the analysis process, because detection of these high-level structures is directly dependent on all previous processing steps. The high-level structures discerned by our analysis approach are calculation blocks and collections of either row- or column headers. Hence, we evaluate the ability of our novel analysis process to accurately discern blocks and headers of a spreadsheet.

4.2.2 Evaluation Method

Blocks and header collections are novel features within spreadsheets. To the best of our knowledge, no collection of spreadsheets is available that provides information about these or similar features. Consequently, evaluation by means of automatic comparison with an existing dataset is not applicable. Instead, a manual evaluation approach is required. We establish the following method for evaluation of our analysis process for using an individual spreadsheet:

1. We manually inspect the spreadsheet and infer blocks, collections of column headers, and collections of row headers.

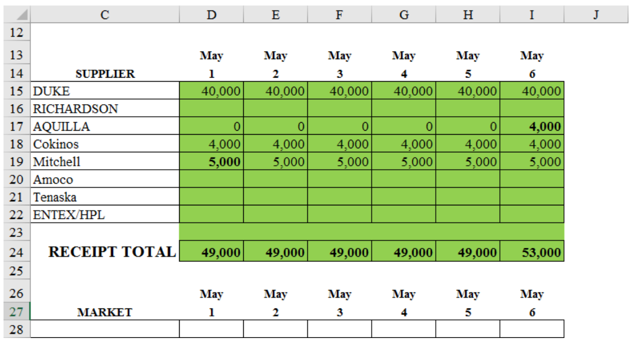
2. We apply our analysis on the same spreadsheet, and list the detected blocks, collections of column headers, and collections of row headers.
3. For each block and header collection we manually inferred, we check whether structural analysis was able to:
    (a) Detect the expected high-level structure.
    (b) Infer the area of the expected high-level structure.
    (c) Infer the dimensions of the expected high-level structure.

The conducted checks for each expected structure pose increasingly hard conditions on the detection performance of our analysis process. Requirement (a) checks whether the process is able to determine any block or header area located in the area of the expected block or header area. Requirement (b) checks whether the determined structure encompasses at least the same area as the expected structure. Requirement (c) checks whether the discerned block or header area matches the expected size of the related block or header area. Structures which encompass an area that is greater than the area of the expected structure do not fulfil this requirement.

Evaluation is limited to eligible worksheets within the chosen spreadsheet. Worksheet eligibility is based on two factors. First, eligible worksheets must contain at least one expected calculation block based on a formula within the worksheet. Second, spreadsheets often contain multiple worksheets which are structurally equivalent. Only one structurally equivalent worksheet within each spreadsheet is eligible for evaluation. The comparison-results for each structure type within each worksheet are tallied and serve as result of the evaluation of the worksheet.

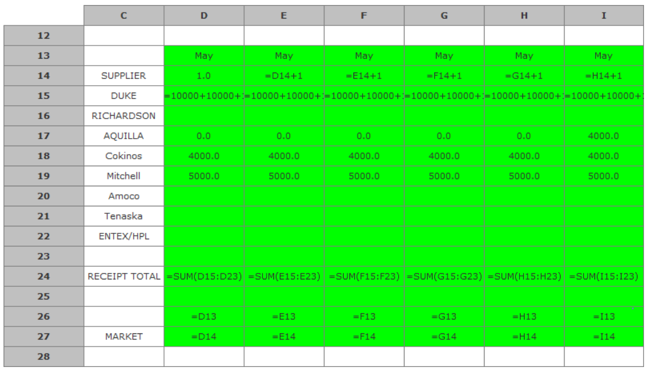
Figure 4.2 presents an example of the applied evaluation method. The example is based on an excerpt of a worksheet of the ENRON corpus. Sub-figure 4.2a illustrates the manual inspection of the worksheet. Based on this inspection, a block is expected to encompass area D15:I24 of the worksheet, highlighted in dark green. The result of process-based analysis of the same excerpt is presented by Sub-figure 4.2b. The actually detected block is highlighted in bright-green. Structural analysis is able to detect a block within the expected area. Furthermore, structural analysis is able to completely infer the area of the expected block. However, structural analysis proves unable to infer a block which correctly matches the dimensions of the expected block.

This is due to the fact that cells in rows 14, 26 and 27 contain formulas instead of static values. Consequently, these cells do not limit block expansion, but instead contribute to the size of the block; Block Expansion infers a block which exceeds the expected dimensions.



	C	D	E	F	G	H	I	J
12								
13		May	May	May	May	May	May	
14	SUPPLIER	1	2	3	4	5	6	
15	DUKE	40,000	40,000	40,000	40,000	40,000	40,000	
16	RICHARDSON							
17	AQUILLA	0	0	0	0	0	**4,000**	
18	Cokinos	4,000	4,000	4,000	4,000	4,000	4,000	
19	Mitchell	**5,000**	5,000	5,000	5,000	5,000	5,000	
20	Amoco							
21	Tenaska							
22	ENTEX/HPL							
23								
24	**RECEIPT TOTAL**	**49,000**	**49,000**	**49,000**	**49,000**	**49,000**	**53,000**	
25								
26		May	May	May	May	May	May	
27	MARKET	1	2	3	4	5	6	
28								

(a) Expected block.



	C	D	E	F	G	H	I
12							
13		May	May	May	May	May	May
14	SUPPLIER	1.0	=D14+1	=E14+1	=F14+1	=G14+1	=H14+1
15	DUKE	=10000+10000+	=10000+10000+	=10000+10000+	=10000+10000+	=10000+10000+	=10000+10000+
16	RICHARDSON						
17	AQUILLA	0.0	0.0	0.0	0.0	0.0	4000.0
18	Cokinos	4000.0	4000.0	4000.0	4000.0	4000.0	4000.0
19	Mitchell	5000.0	5000.0	5000.0	5000.0	5000.0	5000.0
20	Amoco						
21	Tenaska						
22	ENTEX/HPL						
23							
24	RECEIPT TOTAL	=SUM(D15:D23)	=SUM(E15:E23)	=SUM(F15:F23)	=SUM(G15:G23)	=SUM(H15:H23)	=SUM(I15:I23)
25							
26		=D13	=E13	=F13	=G13	=H13	=I13
27	MARKET	=D14	=E14	=F14	=G14	=H14	=I14
28							

(b) Detected block.

Figure 4.2: Example of the applied evaluation method. The example illustrates an excerpt of a worksheet of the ENRON corpus. Expected and inferred blocks in each sub-figure are indicated in green. The formula-view reveals that header information is provided via formula cells. Hence, the inferred block exceeds its expected dimensions.

4.2.3 Evaluation Data

We already established an evaluation method to apply to a certain worksheet. However, the spreadsheet corpora we intend to employ in our evaluation encompass a substantial number of spreadsheets. Hence, for the scope of this master's thesis, manual evaluation applied to each spreadsheet of the chosen evaluation corpora is not feasible. Instead, we propose a selection of representative spreadsheets for both evaluation corpora.

Spreadsheets of the EUSES corpus are categorized into one of eleven categories. After application of our filtering process, nine of these categories still contain spreadsheets. As representatives of the EUSES corpus, we select typical spreadsheets of each category. As measure of typicality, we apply the quotient of the number of formula cells to the overall number of cells contained in spreadsheets of each category. Up to five of the most typical spreadsheets within each category (the category 'personal' only features four spreadsheets) are picked for evaluation. In total, 44 spreadsheets in nine categories serve as evaluation data.

For spreadsheets of the ENRON corpus, we apply additional information provided by Dou *et al.* [2016]. Dou *et al.* presented VEnron, a spreadsheet corpus which features version information based on the ENRON corpus. To establish VEnron, Dou *et al.* clustered the spreadsheets of ENRON into 360 groups. Spreadsheets within the same group are different versions of the same spreadsheet. The authors of VEnron argue, that spreadsheet versions in the ENRON corpus were introduced by manual alteration by a user and successive re-issuing of the spreadsheet in form of an e-mail attachment. We therefore assume that groups featuring the most individual versions indicate spreadsheets that were of importance within the business processes of the ENRON company. We take advantage of this information by choosing the groups featuring the greatest version history as basis for our evaluation. To that end, we iterate each group within VEnron in order of cardinality. Each group whose spreadsheets meet the analysis criteria established in Subsection 4.1.1 is added to the pool of valid evaluation groups. We continue this process to establish the 30 most relevant evaluation groups. The most recent spreadsheet from each group is chosen as representative of the ENRON corpus.

4.3 Evaluation Results

We proceed to present the results of our evaluation of the structural analysis process. During this process, we applied our evaluation method to eligible worksheets of chosen spreadsheets of both, the EUSES and ENRON corpora. The results for the detection performance on individual worksheets were combined using two different evaluation modes:

- *Worksheet-based evaluation.* This evaluation mode is based on the percentage of expected structures that were found within each analysed spreadsheet. For example, manual appraisal of a worksheet results in five expected blocks. Our structural analysis process detects all five blocks; Four of them are complete; Three of them are correct. Consequently, worksheet-based evaluation indicates that 99 % of expected blocks were detected, 80 % of blocks were completely found, and 60 % of blocks were correct. Worksheet-based evaluation provides a statistical assessment of these detection rates for each analysed worksheet of both corpora.

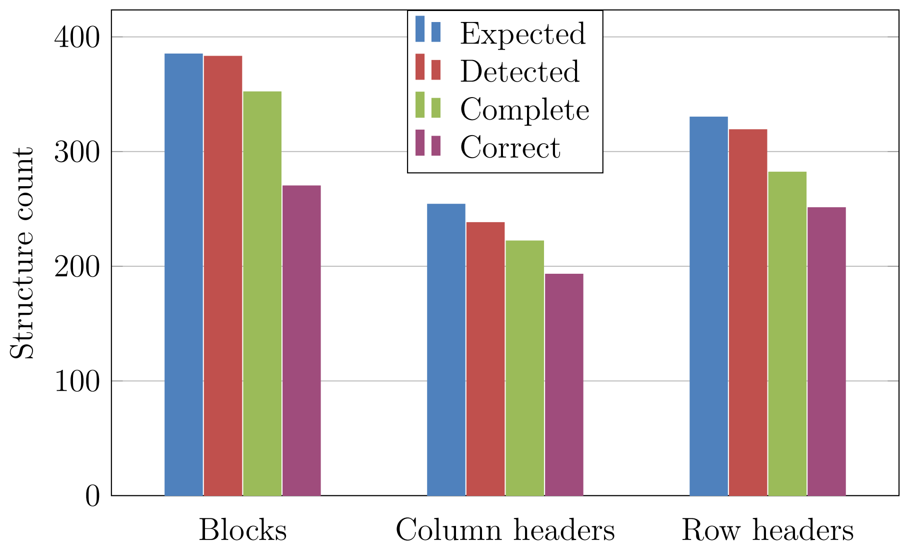
- *Corpus-based evaluation.* This evaluation mode is based on the total number of structures that were detected within all analysed spreadsheets of a specific corpus. Figure 4.3 provides an example for this evaluation mode based on the results of the ENRON corpus. Of the 205 blocks that were expected based on manual inspection, 204 were detected, 169 were completely inferred, and 93 were correctly inferred by our structural analysis process.

Table 4.8 summarizes the results of the worksheet-based evaluation mode. Analysis of both corpora resulted in an average block detection rate of 99 %. In average, most of the expected blocks could be completely inferred. Inference of complete blocks was slightly more successful in spreadsheets of the EUSES corpus (89 %) than in spreadsheets of ENRON (83 %). In both corpora, the expected blocks within three quarters of all spreadsheets could be completely inferred. The average rate of correctly deduced blocks in both corpora is noticeably lower than the detection rate of complete blocks. Again, inference of correct blocks for the EUSES corpus (69 %) was more successful than for ENRON (55 %). In more than one quarter of the analysed spreadsheets of the ENRON corpus, none of the corrected blocks could be inferred correctly. In both corpora, overall header detection was less successful than block detection. Detection of column headers (EUSES: 96 %, ENRON: 86 %) was more successful than detection of row headers (EUSES: 94 %, ENRON: 76 %). In both cases, most of the detected headers were completely inferred. Overall, header detection rates for both corpora were lower than related block detection rates. With exception of row headers in ENRON, the percentage of complete headers inferred in both corpora is comparable to the rates achieved for blocks. Detection rates of correct headers were higher in all cases than related detection rates for correct blocks.

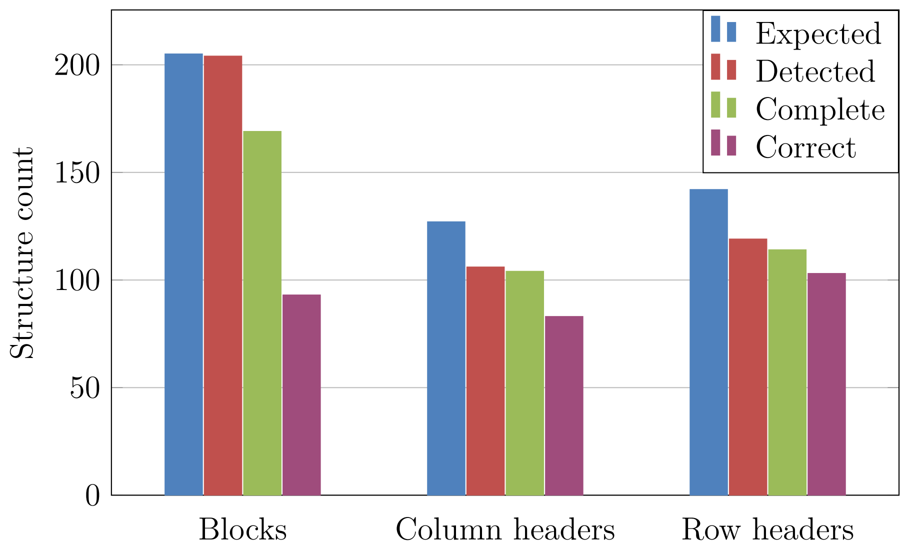
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
EUSES						
detected blocks	67	100	100	99	100	100
complete blocks	0	100	100	89	100	100
correct blocks	0	42	100	69	100	100
detected column headers	0	100	100	96	100	100
complete column headers	0	100	100	89	100	100
correct column headers	0	50	100	77	100	100
detected row headers	0	100	100	94	100	100
complete row headers	0	95	100	84	100	100
correct row headers	0	50	100	76	100	100
ENRON						
detected blocks	92	100	100	99	100	100
complete blocks	0	100	100	83	100	100
correct blocks	0	0	67	55	100	100
detected column headers	0	100	100	86	100	100
complete column headers	0	100	100	82	100	100
correct column headers	0	0	100	60	100	100
detected row headers	0	73	100	76	100	100
complete row headers	0	38	100	72	100	100
correct row headers	0	0	93	63	100	100

Table 4.8: Worksheet-based evaluation results. Metrics indicate success-rates to identify occurring high-level structures within a worksheet of the related corpus.

Figure 4.3 summarizes the results of the corpora-based evaluation. Most of the expected blocks could be detected by structural analysis. Most of expected block could be completely inferred. A noticeable drop is visible between detection rates for complete blocks and correct blocks. This drop is more pronounced for spreadsheets of the ENRON corpus. In general, detection rates for headers are slightly lower than block detection rates. Inference of complete blocks was slightly more successful for headers than for blocks. The rate of correctly inferred headers is noticeably higher than the rate of correctly inferred blocks.



(a) Result for EUSES corpus.



(b) Result for ENRON corpus.

Figure 4.3: Results of structure-based evaluation based on selected spreadsheets of EUSES and ENRON corpora. Bars illustrate how many blocks, column headers, and row headers were *Expected*, in comparison to how many of those structures were *Detected*, inferred *Complete*ly, and inferred *Correct*ly.

4.4 Discussion

The main purpose of this evaluation was to assess the ability of our analysis approach to detect high-level structures. In general, 99 % of the blocks as well as more than 80 % of expected headers could be detected. Most of the detected structures could be inferred featuring their complete dimensions. However, a noticeably lower percentage of expected structures could be inferred featuring their correct dimensions.

The key finding of our evaluation is that an average of 99 % of the expected blocks are detectable and most of detected blocks are complete. Hence, blocks inferred by our analysis can be used as basis for spreadsheet QA techniques with a high confidence. When applied to enhance spreadsheet smells, the main advantage of using blocks is limiting the number of cells which need to be analysed during smell detection. Completely inferred blocks fulfil this requirement, as they provide a smaller scope for smell detection. However, we acknowledge that non-correctly inferred blocks still introduce superfluous cells to smell detection processes and are therefore prone for improvement.

We observed that headers, in general, are less successfully detected than blocks of the same corpus. This is likely based on the employed definitions of high-level structures and our analysis process. Blocks are expected to emerge in areas within a worksheet that are related to calculations. Calculations are based on formulas, and blocks are inferred based on the position of formulas within a worksheet. Hence, blocks are reliably detected by our analysis process. However, detection of a header requires a cell to contain a value of a specific type, and be positioned in correct relation to a specific other block which was already inferred by the analysis process. Hence, more individual requirements need to be fulfilled for detection of an expected header. Consequently, a lower rate of successful detections is to be expected. Nevertheless, the rates of completely, and correctly inferred headers match and even exceed the rates for block detection. This is likely due to the fact that the total area and internal structure of header areas is lower than the total area internal structure of blocks. Hence, detection of header areas in their entirety is more reliable than block detection.

We found that the detection rates of structures (most pronounced for correct blocks) is noticeably higher for spreadsheets of the EUSES corpus than for those of the ENRON corpus. This correlates with our observation, that spreadsheets of the EUSES corpus feature varying amounts of structural complexity, whereas structural complexity of spreadsheets of the ENRON corpus is mostly similar. We reason that a common idea of spreadsheet structures might be established as part of company culture. However, further analysis is required to verify this finding.

During our evaluation, we noticed a number of reoccurring circumstances which lead to unsuccessful inference of the correct block area. These circumstances can be divided into two categories. First, circumstances which leads to a block being not completely detected: *Non-referred strings and decimals being placed within blocks.* Our analysis process does not allow for non-referred value cells to occur within a block. However, in some cases spreadsheet users supply additional information and commentary for values and areas within a calculation block by means of such cells. In these cases, blocks may are not expanded past the introduced value cells, leading to non-completely inferred blocks. Second, circumstances which lead to blocks being not correctly detected as consequence of expanding in unexpected areas of a worksheet:

- *Formulas in header cells.* Block headers are supplied either by formulas referring to the headers of other blocks, or by formulas calculating ascending indices. However, our analysis process expects any formula to be part of a block. Hence, depending on the position of the header cells, the areas containing such headers might be incorporated into neighbouring blocks, and the related blocks inferred featuring incorrect dimensions as a consequence.
- *Referred header cells.* Block headers are referred to by formulas within other parts of the spreadsheet. However, our analysis process expects any cell that is referred to by a formula to be part of a block. Hence, depending on the position of the header cells, the areas containing such headers might be incorporated into neighbouring blocks, and the related blocks inferred featuring incorrect dimensions as a consequence.
- *Lacking block separation.* Different blocks within a worksheet are positioned right next to each other, featuring only limited or no dividing space and no separating layer of headers. Distinct blocks might be separated by cell borders. However, borders are not part of our analysis process. Hence, such calculation areas are merged into a common block, and the block is featuring incorrect dimensions for both of the areas it is based on.

It is evident that detection rates for header areas are improvable, as well. During our evaluation, we discovered a number of reoccurring circumstances which limited the effectiveness of header detection. These circumstances can be divided into the following categories:

- *Formula-related header cells.* Header cells may either contain formulas or be referred to by other formulas. However, our analysis process does not consider formulas cells as well as cells that are referred to by any formula to be eligible header cells. Hence, such header cells cannot be inferred by the current analysis process.
- *Incorrectly inferred blocks.* Header detection infers layers of header cells based on the dimensions of detected blocks within a worksheet. Hence, incorrect detection of blocks which feature deviating dimensions leads to possible detection of header areas featuring the same deviating dimensions.
- *Ambiguous worksheet structure.* Header detection infers layers of header cells in the vicinity of of detected blocks within a worksheet. Hence, if any unrelated but valid header cell is positioned within such a vicinity, it is incorrectly detected as header of a block. Likewise, if any other cell that is not blank and not a valid header cell is positioned between an expected header cell and its related block, the relation for this header cell cannot be correctly established.

5. Spreadsheet Smells: State of the Art

The idea of spreadsheet smells is based on the notion of code smells, which was introduced by Martin Fowler [1999]. In his work, Fowler promoted refactoring as a method to improve the quality of object-oriented code. However, common guidelines for appropriate use of refactoring were unavailable at that time. Therefore, Fowler proposed the utilization of a new concept for that purpose: code smells. According to Fowler's definition, a smell is a certain circumstance within object-oriented source code which suggests a certain refactoring. As such, a smell is defined by a specific anticipated flaw within the structure of object-oriented code. Duplicated code sections in unrelated classes are an example of such a flaw. Fowler proposed a list of 22 smells concerning object-oriented source code.

Spreadsheet smells are a recent development within the field of static spreadsheet analysis. As the name suggests, this method revolves around the conception and detection of smells within a spreadsheet. Cells and worksheets that contain a smell are either more likely to contain a fault or harder to maintain than non-smelly ones. Therefore, spreadsheet smells define a quantifiable measure of suspiciousness towards specific cells and worksheets.

Code smells do not directly indicate faulty code statements by themselves. Instead, code smells indicate code segments that are hard to comprehend, hard to maintain, or error-prone. Likewise, spreadsheet smells in general do not indicate faulty cells by themselves, but rather highlight localized quality issues in a spreadsheet.

5.1 Known Spreadsheet Smells

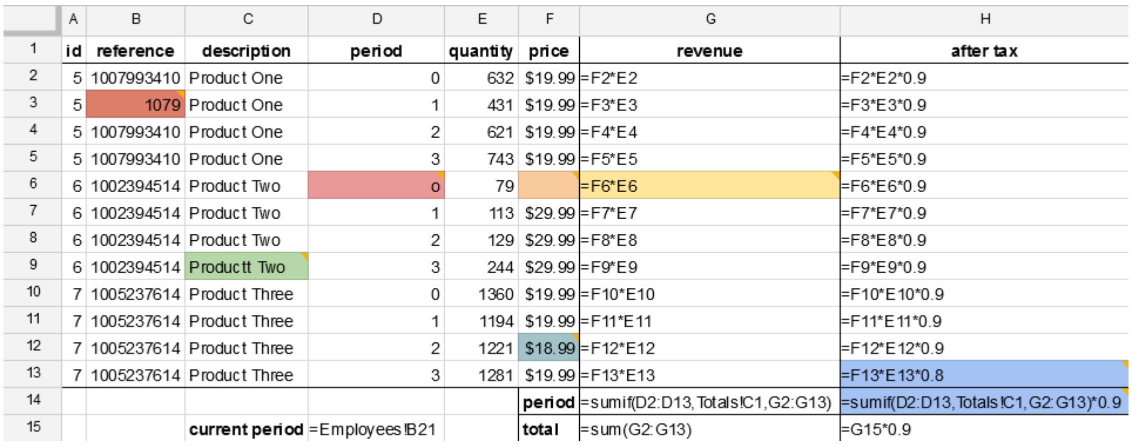
In recent years, a number of different advances have been made to adopt the notion of smells into the spreadsheet domain. These advances contributed to the set of spreadsheet smells known to the scientific community. In the following section, we present this collection of known spreadsheet smells. For each smell, we discuss the following aspects:

- *Origin & Intent:* We name the origin and intent of the smell.
- *Target:* We define which parts of a spreadsheet can contain the smell.
- *Detection:* We describe how the smell can be detected.
- *Example:* We provide an example, based on an exemplary spreadsheet presented in Figure 5.1.
- *Cause:* We state possible causes of the smell.
- *Consequences:* We elaborate on which subsequent issues may be indicated by the smell.
- *Alleviation:* We suggest ways to remove the smell.

The example spreadsheet in Figure 5.1 describes the basic internal processes of a warehouse company. Figure 5.1a depicts the *Sales* worksheet, aggregating product sales and revenues per period. Figure 5.1b illustrates the *Employees* worksheet, containing employee data and summarizing working hours. Lastly, Figure 5.1c presents the *Totals* worksheet, combining revenue and expenses to calculate total and periodic results as well as a productivity quota for the current period. Illustrated worksheets feature a number of highlighted cells. These highlights indicate detected smells that we will elaborate on throughout the section.

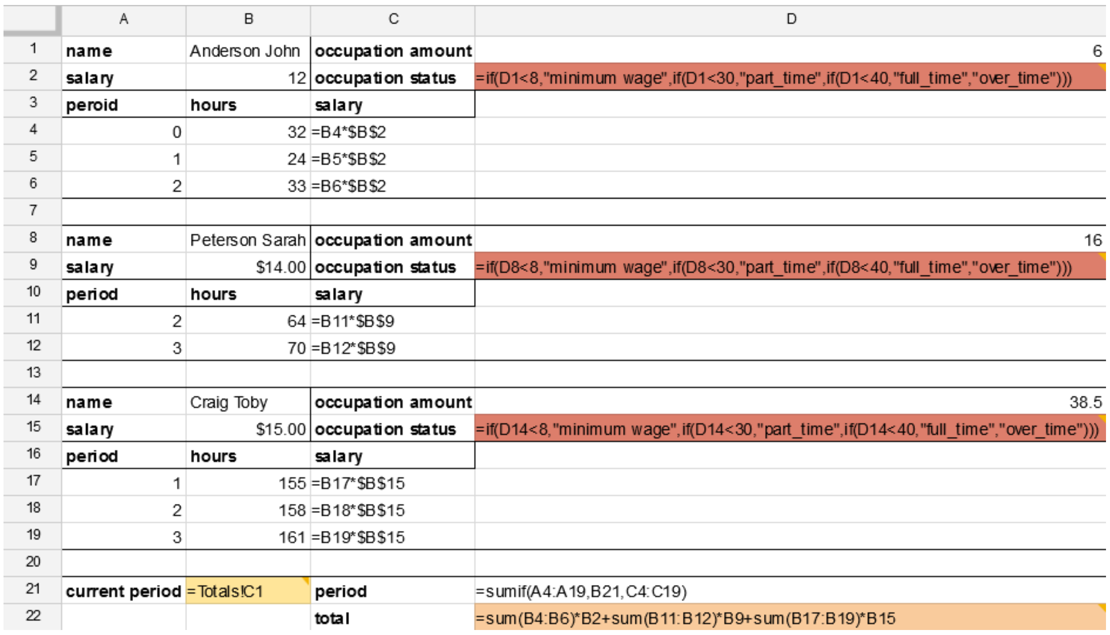
5.1.1 Standard Deviation

Origin & Intent. Standard Deviation was first proposed by Cunha *et al.* [2012b]. The smell is designed to detect statistical outliers within groups of numeric cells.



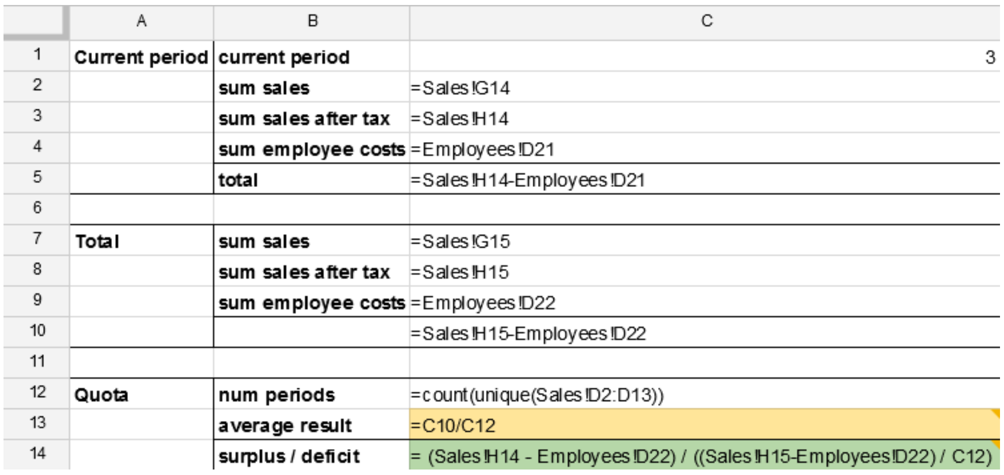
	A	B	C	D	E	F	G	H
1	**id**	**reference**	**description**	**period**	**quantity**	**price**	**revenue**	**after tax**
2	5	1007993410	Product One	0	632	$19.99	=F2*E2	=F2*E2*0.9
3	5	1079	Product One	1	431	$19.99	=F3*E3	=F3*E3*0.9
4	5	1007993410	Product One	2	621	$19.99	=F4*E4	=F4*E4*0.9
5	5	1007993410	Product One	3	743	$19.99	=F5*E5	=F5*E5*0.9
6	6	1002394514	Product Two	o	79		=F6*E6	=F6*E6*0.9
7	6	1002394514	Product Two	1	113	$29.99	=F7*E7	=F7*E7*0.9
8	6	1002394514	Product Two	2	129	$29.99	=F8*E8	=F8*E8*0.9
9	6	1002394514	Productt Two	3	244	$29.99	=F9*E9	=F9*E9*0.9
10	7	1005237614	Product Three	0	1360	$19.99	=F10*E10	=F10*E10*0.9
11	7	1005237614	Product Three	1	1194	$19.99	=F11*E11	=F11*E11*0.9
12	7	1005237614	Product Three	2	1221	$18.99	=F12*E12	=F12*E12*0.9
13	7	1005237614	Product Three	3	1281	$19.99	=F13*E13	=F13*E13*0.8
14						**period**	=sumif(D2:D13,Totals!C1,G2:G13)	=sumif(D2:D13,Totals!C1,G2:G13)*0.9
15			**current period**	=Employees!B21		**total**	=sum(G2:G13)	=G15*0.9

(a) *Sales* worksheet.



	A	B	C	D
1	**name**	Anderson John	**occupation amount**	6
2	**salary**	12	**occupation status**	=if(D1<8,"minimum wage",if(D1<30,"part_time",if(D1<40,"full_time","over_time")))
3	**peroid**	**hours**	**salary**	
4	0	32	=B4*$B$2	
5	1	24	=B5*$B$2	
6	2	33	=B6*$B$2	
7				
8	**name**	Peterson Sarah	**occupation amount**	16
9	**salary**	$14.00	**occupation status**	=if(D8<8,"minimum wage",if(D8<30,"part_time",if(D8<40,"full_time","over_time")))
10	**period**	**hours**	**salary**	
11	2	64	=B11*$B$9	
12	3	70	=B12*$B$9	
13				
14	**name**	Craig Toby	**occupation amount**	38.5
15	**salary**	$15.00	**occupation status**	=if(D14<8,"minimum wage",if(D14<30,"part_time",if(D14<40,"full_time","over_time")))
16	**period**	**hours**	**salary**	
17	1	155	=B17*$B$15	
18	2	158	=B18*$B$15	
19	3	161	=B19*$B$15	
20				
21	**current period**	=Totals!C1	**period**	=sumif(A4:A19,B21,C4:C19)
22			**total**	=sum(B4:B6)*B2+sum(B11:B12)*B9+sum(B17:B19)*B15

(b) *Employees* worksheet.



	A	B	C
1	**Current period**	**current period**	3
2		**sum sales**	=Sales!G14
3		**sum sales after tax**	=Sales!H14
4		**sum employee costs**	=Employees!D21
5		**total**	=Sales!H14-Employees!D21
6			
7	**Total**	**sum sales**	=Sales!G15
8		**sum sales after tax**	=Sales!H15
9		**sum employee costs**	=Employees!D22
10			=Sales!H15-Employees!D22
11			
12	**Quota**	**num periods**	=count(unique(Sales!D2:D13))
13		**average result**	=C10/C12
14		**surplus / deficit**	= (Sales!H14 - Employees!D22) / ((Sales!H15-Employees!D22) / C12)

(c) *Totals* worksheet.

Figure 5.1: Warehouse example spreadsheet.

Target. Standard Deviation detects smelly numerical values. Thus, it can only be applied to cells that contain such values. Moreover, the smell is usually applied to input cells only. Numeric results of formula cells are ignored in the detection process.

Detection. Detection of the Standard Deviation smell relies on the statistical properties of a group of cells. To that end, groups of neighbouring cells in either column or row orientation are formed. For each occurring group, the normal distribution model of its contained numeric values is calculated. Based on this model, cells within the group are marked as smelly, if their cell value deviates by two standard deviations from the calculated average of the group.

Example. An occurrence of the Standard Deviation smell can be seen in cell B3 of the *Sales* worksheet depicted in Figure 5.1a. We calculated the average of the values within column B to be 921,209,151 and the standard deviation of the values in the column to be 290,113,805. Therefore, values within the column are expected to lie within the range [340,981,541,1,501,436,762]. However, cell B3 contains the value 1,079. This lies outside the expected range. Thus, cell B3 is marked as smelly.

Cause. Standard Deviation indicates the occurrence of unexpected numeric cell values. Such values are usually introduced due to errors at the stage of data entry. Smelly numerical values may also be a result of ill-considered copy & paste operations. Furthermore, accidental alteration of cell values throughout the life cycle of a spreadsheet can introduce deviating values. Poor spreadsheet layout may also cause the smell. For example, a row or column may contain multiple successive groups of related numerical values to be computed alongside. However, such groups do not necessary follow the same mathematical distribution. Without some form of boundary between such value groups, some values within the row or column may be marked as smelly as a result.

Consequences. The Standard Deviation smell affects numerical cells. Spreadsheet users rely on the values provided by this type of cell to conduct various computations and statistic analyses. Therefore, a deviating input value will in most cases propagate throughout the spreadsheet and result in an error in at least one of the output values.

Alleviation. Correction of a cell that is tagged with the Standard Deviation smell depends on the circumstance that caused the cell to be smelly. In any case, the cell value should be examined at first. If a faulty cell value is detected, correcting it will usually also remove the smell. If the cell value was not faulty, or the cell is still indicated as smelly after correcting the value, structural considerations need to be applied. As described before, Standard Deviation may be detected due to successive groups of numeric values within the same row or column. In such cases, we suggest structural refactoring of the worksheet in question: Either introduce some form of boundary between the value groups, or split the groups into corresponding tables.

5.1.2 Empty Cell

Origin & Intent. The smell Empty Cell was proposed by Cunha *et al.* [2012b]. They proposed this smell to indicate cells that are empty, but occur within a context that suggests that the cell should contain a value.

Target. Empty Cell detects smelly empty cells. Consequently, it can only be applied to cells that do not contain any values, labels, formulas or errors.

Detection. This smell intends to detect empty cells occurring in a suspicious context. Such a context is described by a number of neighbouring cells that do not contain any other empty cells. Cunha *et al.* proposed to utilize windows of five cells for this purpose. During smell detection, for each row or column every possible window of 5 neighbouring cells is considered and verified whether it holds precisely one empty cell. If an empty cell only occurs within such groups, it is indicated as smelly.

Example. Cell F6 of the *Sales* worksheet, depicted in Figure 5.1a, is empty. Additionally, it is exclusively contained in 5-cell-neighbourhoods that do not contain any other empty cells. As a result, for this cell the Empty Cell smell is signalled. Cells in row 14 also contain empty cells. However, those cells occur in neighbourhoods with other empty cells. Subsequently, they are not marked as smelly.

Cause. The Empty Cell smell indicates empty cells that are surrounded by non-empty cells. Empty cells within a table are usually introduced by mistake. During data entry or spreadsheet creation, sometimes a cell is overlooked by accident. An empty cell within a table may also be the result of an ill-considered copy & paste operation. Lastly, the empty cell may have been introduced by an accidental delete operation later within the life cycle of the spreadsheet.

Consequences. Empty Cell indicates empty spots within the bulk of a worksheet. Cells within this area usually contain numeric input values or formulas. Those formulas either calculate final results or are themselves referenced by other formula cells. Thus, empty cells within those areas usually lead to missing or erroneous interim and final results.

Alleviation. In order to remove the Empty Cell smell, we suggest to check whether the cell in question should indeed be empty. If this is not the case, determine the missing content and insert it. Suggestions for the missing content may be automatically derived from the surrounding context and presented to the user.

5.1.3 Pattern Finder

Origin & Intent. The Pattern Finder smell was proposed by Cunha *et al.* [2012b]. It is an extension of the Empty Cell smell. Instead of focussing on empty cells, Pattern Finder attempts to detect more general deviations of expected cell types: for example, a string cell situated in a neighbourhood of cells containing numeric values.

Target. Pattern Finder attempts to detect unexpected deviations of occurring cell types within rows or columns of a worksheet. The focus thereby lies on the deviation itself, rather than the specific type. Thus, every cell within the active area of a worksheet is a potential target for the detection of the Pattern Finder smell.

Detection. The method of detecting the Pattern Finder smell is based on the methods that are applied to detect the Empty Cell smell. Detection of Pattern Finder relies on the inspection of the local neighbourhood of the cell in question. To that end, Cunha *et al.* proposed to form windows of 4 neighbouring cells for each row or column in the spreadsheet. For each of those windows, we need to check whether it contains exactly one cell containing a different type than the remainder of the group. If such a cell is detected, it is flagged as smelly.

Example. In the *Sales* worksheet, illustrated in Figure 5.1a, cell D6 contains the String "o". This cell thus has the cell type label. However, the surrounding cells in the same column contain number values. Therefore, D6 is indicated as smelly. It is likely that the cell should have contained the number 0 instead of the label "o": a typing error occurred.

Cause. Cells that are flagged with the Pattern Finder smell usually indicate some form of inconsistency within the spreadsheet. This inconsistency can might be introduced due to a mistakes during the creation or an update of the spreadsheet. Cells marked with the Pattern Finder smell may also contain temporary placeholder values, which were intended to be replaced eventually.

Consequences. Pattern Finder indicates inconsistencies within the bulk of a spreadsheet table. Cells within this area usually contain numeric input values or formulas that calculate interim results and are themselves referenced by other formula cells either directly or by use of area operations. Thus, inconsistencies within those areas usually lead to erroneous interim and final results.

Alleviation. To remove the Empty Cell smell, we suggest to check whether the indicated cell holds the correct value type. Otherwise, determine which type and value should be contained and replace the faulty value. Suggestions for the missing content may be automatically derived from the surrounding context and presented to the user.

5.1.4 String Distance

Origin & Intent. The String Distance smell was first introduced in [Cunha *et al.*, 2012b]. Cunha *et al.* noticed that typographical errors are frequently introduced during data input by typing. Consequently, they implemented the String Distance smell in order to detect typographical errors within spreadsheets. To that end, the smell indicates string cells that differ minimally from other strings contained within the same worksheet.

Target. String Distance is a spreadsheet smell that detects smelly string cells. As such, only cells containing string values may contain this smell. Other cell types, like formulas containing strings, are not taken into consideration during the detection of this smell.

Detection. In order to detect the String Distance smell, Cunha *et al.* proposed to utilize an algorithm introduced by Levenshtein [1966]. This algorithm takes two strings as input and calculates the number of single transformation operations that need to be applied to one of the strings in order to transform it into the other. During the detection process, this algorithm is applied to each pair of strings within a row or column. If such a pair of strings only differs by one transformation, the string in question is signalled as smelly. Cunha *et al.* suggest to limit the detection to strings that contain more than three characters. Moreover, String Distance may indicate ascending numeric and alphanumeric designations within neighbouring cells as smelly, e.g. cells in a row or column that contain the values *Product 1*, *Product 2* etcetera. Sequences like this are common within spreadsheets and should therefore be excluded from the detection of this smell.

Example. The cell C9 of the *Sales* worksheet, displayed in Figure 5.1a, contains the String Distance smell. This cell contains the label "Productt two". By removal of a 't' character, this string can be transformed to the label contained in cells C6 to C8. Removal of a character only amounts to a single operation. Consequently, cell C9 is indicated as smelly.

Cause. Cells that contain the String Distance smell contain string values that differ minimally from other occurring strings. Such instances usually indicate faults that were introduced by typing errors during data entry. A cell may also smell of String Distance as result of an accidental alteration of a cell value later on in the life cycle of a spreadsheet.

Consequences. String cells are typically used as labels and headers within a spreadsheet. In such cases, relating faults may predominantly lessen the accountability of a spreadsheet. However, string values can also be used as part of conditional branches of a formula, or to encode values within the same row or column using a LOOKUP operator. According faults directly affect the corresponding calculations and lead to erroneous results.

Alleviation. To remove the String Distance smell, check whether the string value contained in the indicated cell is correct. Otherwise, determine which string should be contained and insert it, replacing the faulty value. Suggestions as to which string should be contained may be automatically derived from the surrounding context and presented to the user, or even applied automatically.

5.1.5 Reference to Empty Cells

Origin & Intent. Reference to Empty Cells is a spreadsheet smell that was defined by Cunha *et al.* [2012b]. They pointed out that formulas that include references to empty cells are a typical source of errors in spreadsheets. Consequently, they introduced the Reference to Empty Cells smell to indicate such occurrences.

Target. The smell indicates formula cells that contain at least one reference to an empty cell. As such, only formula cells can contain this smell. Other cell types are not taken into consideration during the detection process of Reference to Empty Cells.

Detection. Detection of Reference to Empty Cells requires for every cell's formula within the spreadsheet to be analysed. For each formula, the cells that are referenced within the formula are determined. If one of those referred cells does not contain a value, the cell containing the formula in question is flagged as smelly with Reference to Empty Cells.

Example. Figure 5.1a depicts the *Sales* worksheet of our example. The formula at cell G6 of this worksheet contains a reference to the cell F6, which does not contain any value. As a result, G6 is marked as smelly.

Cause. Formula cells that are indicated to be smelly contain at least one reference that points to an empty cell. Such references may be introduced due to errors when entering formulas during spreadsheet creation. References to empty cells may also occur as a result of ill-considered copy & paste operations. When a formula cell containing a relative reference is copied, the reference indices are updated according to the position difference between base and target cells of the copy operation. However, it is not guaranteed that the value type of the newly referenced cell complies with the requirements of the formula. Lastly, accidental alterations during the life cycle of a spreadsheet may introduce references to empty cells into a formula or delete values from cells that are referenced by existing formulas.

Consequences. Formulas within spreadsheets are mainly utilized to conduct some form of calculation or statistical analysis based on provided input values. References to empty cells within a formula usually are not evaluated as errors. Instead, references to empty cells are interpreted as the numeric value 0, or an empty string, based on the related operator within the formula. However, formulas require meaningful input data at the location of their references to fulfil their intended function. Thus, a formula containing a reference to an empty cell may be syntactically valid, but will usually yield an erroneous result.

Alleviation. Removal of the Reference to Empty Cell smell requires analysis of the indicated formula as well as analysis of its references. In a first step, we suggest to check each reference as to whether it leads to an empty cell. If so, verify the correctness of the reference. If the reference is faulty, determine which cell should be pointed to instead and update the reference accordingly. If the reference itself is correct, examine the target cell in question. Determine which value or formula is missing and update the cell's content accordingly. Suggestions as to how the reference could be corrected may be automatically derived from the surrounding context and presented to the user.

5.1.6 Quasi-Functional Dependencies

Origin & Intent. The spreadsheet smell Quasi-Functional Dependencies (QFD) was originally proposed by Cunha *et al.* [2012b]. The idea for and the detection mechanism of this smell is based on the notion of Quasi-Functional Dependencies as described by Abraham and Erwig [2006]. In general, a quasi-functional dependency is established by values of multiple rows or columns that are functionally related to each other. Functional dependency occurs between two columns A and B, if multiple occurrences of the same value in column A always correspond to a specific other value in column B. The Quasi-Functional Dependencies smell indicates violations of such relations.

Target. QFD relies on the detection and cataloguing of functional dependencies between column or row values. Subsequently, only cells that contain values are relevant for this smell. Empty cells and cells that contain errors are not taken into consideration. Formula cells are ignored as well, as they do not contain constant cell values to form quasi-functional dependencies with.

Detection. Detection of the Quasi-Functional Dependencies smell requires the recognition of functional dependencies between at least two columns of a worksheet. Cunha *et al.* based their approach on previous work by Chiang and Miller [2008]. In this paper a more general version of the Functional Dependencies principle is presented and utilized to discern non-matching values. Smell detection involves collecting and matching the entirety of all occurring spreadsheet values. Based on the results of the matching step, quasi-functional dependencies are synthesized. Lastly, existing quasi-functional dependencies are evaluated, and cell values that deviate from expected results are indicated as smelly.

Example. An occurrence of QFD can be found in cell F12 of the *Sales* worksheet, depicted in Figure 5.1a. Values within the columns A, B, C, and F follow a Quasi-Functional Dependency. The value of one cell within those columns can be inferred from specific other values within the other columns. For example, in rows 10, 11, and 13 the columns always contain the respective values 7, 1005237614, Product Three, and $19.99. However, cell F12 does not follow this established relation. It contains the numeral value $18.99, while the values of the other columns infer the expected value of $19.99. This is suggestive of a typing error.

Cause. Cells that contain QFD usually indicate some form of inconsistency within the spreadsheet. A typing error or carelessness during data entry may be the cause of such faults. In addition, wrong or incomplete copy & paste may also lead to the introduction of values that smell of Quasi-Functional Dependencies.

Consequences. The Quasi-Functional Dependencies smell indicates an inconsistency within a row or column that contains input data of a spreadsheet. Such cells are usually referenced by formula cells either directly or by use of area operations. As a result, inconsistencies indicated by QFD usually lead to erroneous interim and final calculation results.

Alleviation. Removal of the Quasi-Functional Dependencies smell requires validation of the value contained in the indicated cell. If the value is faulty, determine which value should have been contained and replace the faulty content. As only one specific value that alleviates the smell can be inferred, this refactoring may be provided as automated process.

5.1.7 Multiple Operations

Origin & Intent. Multiple Operations is a spreadsheet smell which was introduced by Hermans *et al.* [2012b]. The smell is based on one of the most well-known code smells: the Long Method. Fowler [1999] proposed this code smell as a way to indicate methods that feature an excessive number of statements. Such methods usually do not fulfil a single, specific purpose, but rather combine multiple tasks. However, the combination of tasks within a single method renders the method in question less comprehensible and should therefore be avoided. Similarly, formulas within a spreadsheet environment should feature a limited number of operations in order to facilitate their accountability. Therefore, Hermans *et al.* introduced the Multiple Operations smell to indicate formulas that feature an excessive number of operations.

Target. The spreadsheet smell Multiple Operations intends to find formula cells that contain an excessive number of operations. As such, only cells that contain formulas are relevant for detecting this smell. Other cell types are not taken into consideration.

Detection. For the detection of the Multiple Operations smell, Hermans *et al.* suggest to count the number of operations each formula contains. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], cells should be indicated as smelly if a formula contains more than 4 operations.

Example. An occurrence of the Multiple Operations smell can be seen within cell D22 of the *Employees* worksheet, depicted in Figure 5.1b. The formula contained in this cell consists of 8 distinct operations. This number exceeds the suggested threshold for the detection of the Multiple Operations smell. Consequently, cell D22 is indicated as smelly.

Cause. Examination results of Hermans *et al.* suggest that the introduction of cells containing the Multiple Operations smell is an evolutionary process. Initially, formula cells do not contain excessive numbers of operations. However, spreadsheets are often target of recurring adaptations. These adaptations lead to additional operations to formulas as the need arises. Especially when working under time constraints, readability of a spreadsheet is often sacrificed in order to achieve a working solution.

Consequences. Cells that are flagged by the Multiple Operations smell contain a large number of operations. As the number of operations increases, the meaning of a formula becomes harder to understand. This circumstance is intensified by the fact that long formulas often are displayed cut-off, as they require more screen space than available.

Alleviation. In order to alleviate the Multiple Operations smell of a formula, we suggest refactoring: Split up the necessary calculations within the formula, distribute them over multiple cells and link them using references. As no additional knowledge or user input is required, this refactoring may be provided as automated process. In some cases, multiple used operations can be replaced by a single area-supporting operation like SUM.

5.1.8 Multiple References

Origin & Intent. The spreadsheet smell Multiple References was first proposed in [Hermans *et al.*, 2012b]. Hermans *et al.* adapted the notion from a similar code smell: Multiple References. This code smell was presented by Fowler [1999]. It indicates method definitions that feature an excessive number of parameters. Comprehensibility of a method definition declines as the number of parameters it requires increases. Likewise, the clarity of a formula declines as the number of references it features increases. Therefore, Hermans *et al.* suggested to utilize the Multiple References smell to indicate formula cells that require an excessive number of references.

Target. Multiple References points out formula cells that rely on a large number of different references. As a result, formula cells are exclusively relevant for the detection of this smell. Other cell types are not taken into consideration.

Detection. For detecting the Multiple References smell, Hermans *et al.* suggest to count the number of references to areas and other cells within a specific formula. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], cells should be indicated as smelly if they contain more than 3 references.

Example. Figure 5.1b depicts the *Employees* worksheet of our example. Cell D22 of this worksheet contains a formula referencing 6 unique areas and different cells. Consequently, it exceeds the suggested threshold for the detection of the Multiple References smell and is highlighted as smelly. Note that the Multiple Operations smell is detected for this cell as well.

Cause. The introduction of Multiple References usually follows the same evolutionary process that leads to the introduction of the Multiple Operations smell. Formula cells normally start out requiring a limited number of references. However, during the life cycle of the spreadsheet, adaptations are made to expand the functionality of those formulas which often require additional references.

Consequences. When contemplating spreadsheet formulas, the more single references to other cells they contain, the harder to comprehend they become. Similar to the Multiple Operations smell, this circumstance is intensified due to the fact that long formulas are usually not displayed completely, since they require more screen space than available.

Alleviation. In order to remove the Multiple References smell, we suggest to distribute the formula contained within the cell over multiple different cells, each containing a subset of the required operations and references. As no additional knowledge or user input is required, this refactoring may be provided as automated process. In some cases, multiple used operations may be replaced by a single area-supporting operation like SUM. In such cases, the corresponding references can be united using area references where appropriate. Additional references may be removed by relocating them next to and merging with an already referenced area.

5.1.9 Conditional Complexity

Origin & Intent. Conditional Complexity was introduced by Hermans *et al.* [2012b]. The spreadsheet smell is based on a notion by Fowler regarding conditional operations in object-oriented code. According to Fowler, readability of code declines in instances where multiple nested conditional operators occur. Similarly, multiple nested conditional operators within a spreadsheet formula are difficult to comprehend. Therefore, Hermans *et al.* proposed the Conditional Complexity smell to indicate formulas featuring an excessive number of nested conditional operators.

Target. The smell Conditional Complexity detects formula cells that contain an excessive number of nested conditional operators. Consequently, the detection of this smell relies solely on cells that contain formulas. Other cell types are not taken into consideration.

Detection. For detecting the Conditional Complexity smell, Hermans *et al.* suggest to count the number of nested conditional operators contained in a specific formula. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], cells should be indicated as smelly if they contain at least 3 nested conditional operators.

Example. Occurrences of the Conditional Complexity smell are contained within cells D2, D9, and D15 of the *Employees* worksheet, illustrated in Figure 5.1b. Each of those formulas contains 3 nested conditional operations. Thus, each cell exceeds the suggested threshold for the detection of Conditional Complexity and is therefore marked to contain the smell.

Cause. According to the evaluation by Hermans *et al.*, this smell rarely occurs. Spreadsheet users obviously have some notion that conditional operators are complex. Subsequently, formulas containing conditional operators are usually handled with care. However, especially when working under pressure, users are less reserved. In such circumstances, users are more likely to rely on nested conditionals.

Consequences. Conditional Complexity marks formula cells within spreadsheets that contain multiple conditional operators. However, even a single conditional operator within a spreadsheet formula can be hard to comprehend for end users. One relevant issue is that formulas are only afforded a limited amount of screen space to be displayed. This space usually does not suffice to show the entire conditional operator. Moreover, the syntax of conditional operators within spreadsheet environments does not emphasize the semantic function of each of its operands. This renders them hard to comprehend. Multiple consecutive conditional operators contained within a formula only worsen this effect.

Alleviation. One way to combat the Conditional Complexity smell is to divide the conditional operations over multiple cells. As no additional knowledge or user input is required, this refactoring may be provided as automated process. Alternatively, the SUMIF and COUNTIF operators can be used to aggregate single conditional operators. If applicable, the LOOKUP operator can be used instead. This operator allows to specify a search key as well as a cell range containing condition-value-pairs. If the key matches one of the conditions within the cell range, the corresponding value is displayed.

5.1.10 Long Calculation Chain

Origin & Intent. The spreadsheet smell Long Calculation Chain was first proposed in [Hermans *et al.*, 2012b]. It can be seen as the antithesis to the Multiple Operations smell. In the usual workflow of a spreadsheet, formula cells rely on results of other formulas for their calculations. As a result, chains of dependent calculations are formed. However, in order to verify the correctness of such a chain, a spreadsheet user needs to trace along multiple references to find the origin of the input values. The longer such a chain grows, the harder it becomes to comprehend. Therefore, Hermans *et al.* introduced this smell to indicate formula cells that rely on exceedingly long calculation chains.

Target. The spreadsheet smell Long Calculation Chain indicates formula cells that rely on a large number of successive dependent calculations. Consequently, only cells that contain formulas are relevant for the detection of this smell. Other cell types are not taken into consideration.

Detection. For detecting the Long Calculation Chain smell, Hermans *et al.* suggest to calculate the length of the longest path of successively referenced cells that need to be traced when evaluating a formula's value. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], cells should be indicated as smelly if the regarding metric is greater than 4.

Example. Cell C13 in the *Totals* worksheet, depicted in Figure 5.1c, is reliant on a vast number of cells. One of the longest calculation chains which can be formed is: Totals!C13 $\rightarrow$ Totals!C10 $\rightarrow$ Sales!H15 $\rightarrow$ Sales!G15 $\rightarrow$ Sales!G2 $\rightarrow$ Sales!E2. To calculate this chain, 5 dependencies need to be dereferenced. This number exceeds the threshold suggested by Hermans *et al.*. Thus, cell C13 of the worksheet *Totals* is indicated as smelly.

Cause. Formulas that rely on the interim results of other formulas for their calculations are a common practise within the spreadsheet domain. Consequently, the formation of calculation chains is a common occurrence. When expanding the functionality of a spreadsheet, users simply add new formulas that refer to existing calculation results. The overall structure of the utilized calculation chains is usually neglected during this process. Restructuring and regrouping of existing functionalities is not a common practice. Thus, long calculation chains are likely to occur, as the functionality of a spreadsheet expands.

Consequences. To understand the meaning and verify the correctness of a specific formula within a spreadsheet, a user needs to trace each reference within a given calculation chain. The longer a calculation chain grows, the higher the number of references and interim results it contains. Consequently, cells with long calculation chains are hard to comprehend and maintain.

Alleviation. In order to alleviate this smell while maintaining the functionality of the required calculations, we suggest to merge multiple steps along the calculation chain into a single formula which aggregates all the necessary operations. As no additional knowledge or user input is required, this refactoring may be provided as automated process. However, note that this approach leads to a trade-off between the intensity of this smell and the intensities of the Multiple Operations and Multiple References smells. In addition, if any cell within the calculation chain is referenced by another formula as well, this case has to be handled accordingly.

5.1.11 Duplicated Formulas

Origin & Intent. The Duplicated Formulas spreadsheet smell was introduced by Hermans *et al.* [2012b]. This spreadsheet smell is based on the *Duplicated Code* smell presented by Fowler [1999]. The related code smell indicates classes that contain multiple occurrences of similar code snippets. Likewise, different formula cells within a spreadsheet can contain equal sub-expressions. The code smell Duplicated Formulas indicates such cells within a worksheet.

Target. Duplicated Formulas points out different formula cells that feature equal sub-expressions within their formulas. Consequently, the detection of Duplicated Formulas is reliant on formula cells only. Other cell types are not taken into consideration.

Detection. For detecting the Duplicated Formula smell, Hermans *et al.* suggest to utilize the *relative R1C1 notation.* References depicted in this notation express their references to other cells relative to the cell that contains the formula. For example, the formula MAX(A1:A3) in cell A4 would be written as MAX(R[-3]C[0]:R[-1]C[0]) in this notation. Hermans *et al.* suggest to measure the number of cells within a worksheet that feature sub-expressions being either equal or share the same *relative R1C1 notation.* Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], cells should be indicated as smelly if they share the same sub-expression with at least 6 other cells. Nevertheless, in other work [2012b] they present an example whereby the Duplicated Formula smell is indicated for a cell that shares its sub-formula with merely a single other cell, casting a doubt on the provided threshold. Lastly, formulas that are entirely equal in *relative R1C1 notation* are not marked as containing the smell, as this is common practice among spreadsheet users. Abreu *et al.* [2014a] proposed an adapted approach for the detection of this smell. Instead of relying on the *relative R1C1 notation*, they directly compare occurring formulas. As an example, two cells containing the formulas SUM(A1:A3) * 1.1 and SUM(A1:A3) * 1.2 would have the first part duplicated and therefore be indicated as smelly.

Example. An occurrence of the Duplicated Formulas smell is demonstrated by cell H13 within the *Sales* worksheet, depicted by Figure 5.1a. The part of the formula referencing the cell values to the left of the formula is shared with other formulas within column H. However, the constant multiplication factor 0.9 of the formula within H13 differs from the remaining column. Thus, the cell is indicated as smelly. Another example is cell H14 of the same worksheet. The formula of this cell shares the SUMIF part with its neighbour, but expands it by an additional multiplication. Consequently, it is pointed out as containing the Duplicated Formulas smell.

Cause. Duplication and adaptation of formulas is common practice during the creation of a worksheet. It is not surprising that multiple occurrences of similar or equal sub-expressions are contained in various formula cells within the same worksheet. During their evaluation, Hermans *et al.* found that a substantial amount of spreadsheet users understand that formula duplication can lead to problems. However, they also observed that comprehension of the issue by users does not lead to fewer occurring duplications within the spreadsheets of those users. Moreover, some users do not see any harm at all in duplicating formulas.

Consequences. Duplication of formula parts may lead to a number of problems. If a large part of a given formula is duplicated from another one, it is hard to distinguish them within the program interface. Moreover, if a formula containing duplicated parts needs to be adapted, this adaptation has to be applied to each of the duplicated formulas. This poses a threat to the maintainability of the spreadsheet, as it is easy to overlook one of the duplicated formulas by mistake.

Alleviation. To remove the Duplicated Formulas smell, we suggest to extract the duplicated section from each related formula and move it into a single, designated formula cell. The base cells need to be updated with the reference to the new formula cell accordingly. This procedure usually increases readability and maintainability of a worksheet. As no additional knowledge or user input is required, this refactoring may be provided as automated process.

5.1.12 Inappropriate Intimacy

Origin & Intent. Inappropriate Intimacy is an inter-worksheet smell which was proposed by Hermans *et al.* [2012a]. According to their definition, the smell is primed to detect worksheets that are relying too heavily on the content of other worksheets. The smell is based on the identically named code smell presented by Fowler [1999].

Target. Inappropriate Intimacy intents to detect worksheets that feature an excessive number of references to different worksheets. Such references are only contained within formula cells of a spreadsheet. As a result, the detection of the Inappropriate Intimacy smell is solely reliant on analysis of formula cells. Other cell types are not taken into consideration.

Detection. In order to detect an inappropriately intimate worksheet, Hermans *et al.* introduced the so-called *Intimacy* metric. *Intimacy* between two spreadsheets is measured by the number of connections between them. The function, defined by this metric has the type $\mathbf{W} \times \mathbf{W} \rightarrow int$ and is defined by the formula

$$\text{Intimacy}(w_0, w_1) = |\{(c_1, c_0) \in \mathbf{KS} : c_0 \in w_0 \wedge c_1 \in w_1 \wedge w_0 \neq w_1\}|.$$

Intimacy counts the number of pairs (c_1, c_0) in $\mathbf{KS}$, the set containing all connections of the spreadsheet, for which holds true that c_0 is contained in worksheet w_0, c_1 is contained in worksheet w_1, and the two worksheets are different. Thus, it counts the number of references the worksheet w_0 contains that point to cells in the worksheet w_1. According to this definition, multiple references from one worksheet to a specific cell of another worksheet are counted repeatedly. The overall intimacy of a worksheet is calculated as

$$\mathrm{II}(w_0) = \max\{\mathrm{Intimacy}(w_0, w_1) : w_0, w_1 \in \mathbf{S}\}.$$

This metric indicates the maximum intimacy the worksheet w_0 has with any other worksheet. Hermans *et al.* suggest to utilize this metric to detect the Inappropriate Intimacy smell. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], the threshold for the detection of this smell is 8. Thus, a worksheet should be highlighted as smelly if it contains at least 8 references to another worksheet.

Example. The detection of the Inappropriate Intimacy smell is based on the maximal number of references to a different spreadsheet. Figure 5.1c depicts the *Totals* worksheet of our example, which contains this smell. By analysing formulas occurring in this worksheet, we see that it contains 6 references to the worksheet *Employees* and 9 references to the worksheet *Sales*. Therefore, the maximal number of references to a different worksheet for the *Totals* worksheet is 9. The value exceeds the suggested threshold for this metric. As a result, the *Totals* worksheet is indicated as smelly.

Cause. According to the evaluation by Hermans *et al.* [2012a], the Inappropriate Intimacy smell is quite common among typical spreadsheets. They observed that spreadsheet creators are usually not trained as programmers. Therefore, structuring spreadsheets in a logical way poses a difficult task. As a consequence, spreadsheet users rely excessively on cross-worksheet references within their formulas. This leads to the accumulation of the Inappropriate Intimacy smell in some cases. In particular, Hermans *et al.* identified two cases which repeatedly cause the Inappropriate Intimacy smell. One depicts the use of a so-called auxiliary worksheet in combination with a second one: The auxiliary worksheet contains data on which the other worksheet relies. The second case depicts two worksheets which repeatedly reference each other without any clear distinction of purpose between them.

Consequences. A worksheet containing the Inappropriate Intimacy smell indicates the existence of a strong semantic connection to at least one other worksheet. Such semantic connections are usually formed when logic to fulfil a coherent task is separated over multiple worksheets. However, such a split of functionality between multiple worksheets may weaken understandability of the spreadsheet as a whole. Moreover, when changes are made within the referenced worksheet, the reliant worksheet needs to be checked for correctness as well. This requires a spreadsheet user to continuously switch between the two worksheets.

Alleviation. Removal of the Inappropriate Intimacy smell of a worksheet requires to reduce its number of references to at least one specific other worksheet. However, this reduction usually requires a well thought-out restructuring process, as the functionality of the spreadsheet as a whole needs to remain unaltered. The correct refactoring strategy to apply depends on the individual situation. Wherever possible, neighbouring references to other worksheets should be merged using area references. In some cases, importing more extensive parts of functionality from the referenced worksheet may be required. In other cases, the best course of action is to merge two worksheets.

5.1.13 Feature Envy

Origin & Intent. Feature Envy is an inter-worksheet smell proposed in [Hermans *et al.*, 2012a]. Hermans *et al.* derived it from a corresponding smell for object-oriented code. Fowler [1999] introduced this code smell to indicate that a specific method is more reliant on the fields of another class than on fields of the class that contains the method. The same principle can be applied to spreadsheet formulas. The spreadsheet smell Feature Envy indicates formulas which are excessively reliant on foreign worksheets.

Target. Feature Envy detects and indicates formula cells that feature an excessive number of references to foreign worksheets. Therefore, only formula cells are analysed during the detection process of this smell. Other cell types are not taken into consideration.

Detection. In order to detect feature envious formula cells, Hermans *et al.* introduced the so-called *Enviousness* metric. *Enviousness* indicates how many references to cells of other worksheets a formula contains. The metric defines a function of the type $\mathbf{C} \rightarrow int$ and is defined by the formula:

$$\text{FE}(c_0) \equiv |\{(c_1, c_0) \in \mathbf{KS} \mid \exists w : c_0 \in w \wedge c_1 \notin w\}|$$

This metric counts the number of pairs (c_1, c_0) in **KS** for which holds true that c_0 is contained in the worksheet but the cell c_1 is not. Hermans *et al.* suggest to utilize this metric to detect the Feature Envy smell. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], a formula cell should be indicated as smelly if it contains at least 3 relations to other worksheets.

Example. An example of the Feature Envy smell can be found in cell C14 of the *Totals* worksheet, depicted in Figure 5.1c. The formula in cell C14 contains 4 distinct references to other worksheets. This value exceeds the threshold that is suggested for the detection of the Feature Envy smell. Thus, cell C14 of the worksheet *Totals* is indicated as smelly.

Cause. The introduction of this smell is based on the same principles that apply to the Inappropriate Intimacy smell. Hermans *et al.* [2012a] state that Feature Envy occurs quite often among spreadsheets. They reason that spreadsheet creators are usually not trained as programmers. Structuring spreadsheets in a logical way is not a trivial problem. As a result, such users rely heavily on cross-worksheet references within their formulas, resulting in the accumulation of the smell. Hermans *et al.* pointed out two distinct cases that repeatedly lead to the introduction of Feature Envy: One depicts the use of so-called auxiliary worksheets which contain data in combination with other worksheets that rely on the provided data. The second case depicts two worksheets that repeatedly reference each other without any clear distinction from each other.

Consequences. Feature Envy implies that a specific cell is excessively interested in cells from another worksheet, rendering it harder to understand. One likely reason for this is fact that spreadsheet users heavily rely on the highlighting feature for formula references. This feature draws borders around the cells and areas of referred cells of a selected formula. However, in most common spreadsheet programs this feature does not work across worksheets.

Alleviation. To alleviate the Feature Envy smell, we suggest to move the formula in question to the corresponding worksheet and link the result of the computation back to the initial worksheet. By carrying out this procedure, the formula in question is moved closer to the cell it is referring to. As a result, the comprehensibility of the worksheets should be improved.

5.1.14 Middle Man

Origin & Intent. Middle Man was introduced as an inter-worksheet smell in [Hermans *et al.*, 2012a]. Hermans *et al.* defined a middle man in the context of spreadsheets as a formula that only contains a single reference to another cell. Such formulas are used frequently as sources of information to be used in further calculations within a local worksheet. However, worksheets that contain an excessive number of such middle man cells are deemed smelly. The notion of a middle man was derived from an identically named code smell introduced by Fowler [1999].

Target. The Middle Man smell indicates worksheets that contain an excessive number of middle man cells. The detection of this smell is solely reliant on the analysis of formula cells within a specific worksheet. Other cell types are not taken into consideration.

Detection. In order to detect a worksheet that smells of the Middle Man smell, Hermans *et al.* introduced a special type of formula: the *middle man* formula. A middle man formula fulfils the sole purpose of fetching a value from another cell. Cells that contain such a formula are detected by the function *MMF*(cell c). This function has the form $\mathbf{C} \rightarrow \mathit{bool}$. However, Hermans *et al.* require that for the Middle Man smell to be detected for a worksheet, a calculation chain of two consecutive passing formulas needs to occur. To that end, they suggest the following metric:

$$\mathrm{MM}(w) \equiv |\{(c_1, c_0) \in \mathbf{KS} : c_1 \in w \wedge \mathit{MMF}(c_0) \wedge \mathit{MMF}(c_1)\}|$$

This metric counts the numbers of middle man formulas in a worksheet that are themselves used as reference by another middle man formula. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], Hermans *et al.*

suggest that a worksheet should be indicated as smelly if it contains at least 7 chained middle man formulas.

Example. An example related to the Middle Man smell can be found in cell B21 of the worksheet *Employees* (see Figure 5.1b). Analysis of the formula in cell C14 indicates that this cell is a middle man cell. Its sole purpose is to fetch the value contained in cell C1 of the *Totals* worksheet. However, the cell B21 itself is target of a middle man cell as well: cell B15 within the worksheet *Sales*. Thus, a chain of multiple middle man cells is established. This, in turn, increases the suspiciousness of the *Employees* worksheet towards the Middle Man smell. However, the suggested value of at least 7 linked middle man cells is not reached. Therefore, the worksheet does not contain the Middle Man smell.

Cause. Middle man cells are commonly introduced to pass along values and calculation results throughout different worksheets. The resulting values are often used for further calculations. However, middle man cells may also be used to keep an eye on specific values in different parts throughout a spreadsheet. In some cases, values are even passed along within the same spreadsheet for that effect. In either case, the method of value passing can follow one of two different approaches. Using the first approach, values are passed from the source to each middle man individually. Using the second approach, values are passed along by formation of a chain. However, this approach may introduce a number of problems.

Consequences. Middle man cells are commonly used to state results of calculations within other worksheets. However, a problem arises when a worksheet contains an excessive number of such middle man cells. Such occurrences impair general readability of a spreadsheet and are linked to bad spreadsheet design.

Alleviation. Which strategies should be applied to alleviate the Middle Man smell depends on the circumstances in which the middle man cells are applied. In cases where values are passed along a chain of multiple middle man cells, we suggest to adapt the corresponding formulas of each step along the chain to directly reference the source cell instead. As no additional knowledge or user-input is required, this refactoring may be provided as automated process. In cases where middle man cells relay values for further processing to different worksheets it might be favourable to simply relocate the spreadsheet logic in question to the vicinity of the data source, thus removing the requirement for value passing altogether.

5.1.15 Shotgun Surgery

Origin & Intent. Shotgun surgery is an inter-worksheet smell presented by Hermans *et al.* [2012a]. This smell is based on the corresponding code smell that indicates that a change within one class needs to be followed up by numerous little changes in several dependent classes. This code smell was presented by Fowler [1999]. The same principle can be applied to spreadsheets. Thus, the Shotgun Surgery smell is detected when cells of a specific worksheet are referenced by multiple formulas within other worksheets.

Target. The Shotgun Surgery smell indicates worksheets that are referenced by a large number of formulas that are contained in different worksheets. Therefore, only formula cells are required for the detection of this smell. Other cell types are not taken into consideration.

Detection. In order to detect worksheets that contain the Shotgun Surgery smell, Hermans *et al.* introduce two metrics, both of the type $\mathbf{W} \to int$:

$$\text{ChangingFormulas}(w) \equiv |\{(c_1, c_0) \in \mathbf{KS} : c_1 \in w \wedge c_0 \notin w\}|$$

The metric Changing Formulas counts the number of formulas outside of worksheet w. that refer to a cell in worksheet w. Based on an evaluation of the EUSES spreadsheet corpus [Fisher and Rothermel, 2005], a worksheet should be deemed smelly if it is referenced by at least 9 changing formulas.

$$\text{ChangingWorksheets}(w_0) \equiv |\{w_1 \in \mathbf{S} \mid \exists (c_1, c_0) \in \mathbf{KS} : c_0 \in w_1 \wedge c_1 \in w_0\}|$$

The metric Changing Worksheets counts the number of worksheets that contain references to any cell in the worksheet w_0. Based on the evaluation of the EUSES spreadsheet corpus, a worksheet should be indicated as smelly if it is referenced by at least 2 other worksheets.

Example. The *Employees* worksheet exhibits the Shotgun Surgery smell. As can be seen in Figures 5.1a and 5.1c, cells within this worksheet are referenced by both the *Sales* and the *Totals* worksheet. Therefore, the threshold to indicate the Shotgun Surgery smell for this worksheet is breached.

Cause. The Shotgun Surgery smell within a spreadsheet can usually be accounted to a lack of foresight during set-up of a spreadsheet. When working with extensive spreadsheets, references to cells of foreign worksheets are often used. Limiting the amount of such references requires a well-considered spreadsheet structure. However, the introduction of clean and comprehensible structures is typically of little concern to novice users. Consequently, instances of Shotgun Surgery naturally accumulate as the functionality of spreadsheets expands.

Consequences. The inter-worksheet smell Shotgun Surgery indicates that cells of a specific worksheet are referred to by many formulas of other worksheets. Naturally, if a highly-referred target cell changes, many of the formulas that refer to it need to be adapted as well. Thus, Shotgun Surgery is linked to maintainability of a spreadsheet. In addition, affected spreadsheets are more error-prone. Introduction of faults is more likely, as changes of individual cells may require adaptations in multiple other parts of the spreadsheet.

Alleviation. In order to cope with the Shotgun Surgery smell, we suggest to minimize the references to cells within the worksheet in question. Another approach to alleviate this smell is to move the reliant formulas to the worksheet in question.

5.2 Overview and Comparison

Spreadsheet smells can be arranged into groups based on which properties of spreadsheets they are related to. As an example, multiple smells are concerned with the complexity and accountability of formula cells. Thus, Long Calculation Chain, Multiple Operations, and Multiple References are connected to one another. When Long Calculation Chain is lowered, others are increased and the other way around. These trade-offs exist in source code smells too.

Empty Cell detection is a subset of the methodology to detect the Pattern Finder smell. However, Empty Cell detection usually allows for more general thresholds as to which part of the local neighbourhood is analysed.

Multiple occurrences of Feature Envy in relation to a specific worksheet imply Inappropriate Intimacy, but not the other way around. Middle Man may be seen as extreme form of Inappropriate Intimacy, referring to a single worksheet. Middle

Man is also an extreme case of Feature Envy, as corresponding formulas only refer to external cells. Shotgun Surgery may be caused by feature-envious or middle man cells, but is not required to be so. Shotgun Surgery can also be seen as counter to the Inappropriate Intimacy smell. Whereas Inappropriate Intimacy indicates that a given worksheet relies too heavily on a number of different worksheets, Shotgun Surgery indicates that other worksheets are too reliant on a specific cell within a single worksheet.

Every proposed inter-worksheet smell can be removed by simply placing all required spreadsheet logic into a single worksheet. However, a more structured design approach usually improves readability and maintainability of spreadsheets. Migration of spreadsheet logic into a single worksheet is only advisable up to a certain size. This behaviour is characteristic for an optimization process. Nevertheless, none of the proposed smells is adequate to be used as a counterbalance for such an optimization. This indicates a vacancy within the current spreadsheet smell catalogue.

In general, smells that indicate similar problems or have similar detection methods tend to influence one another. In order to highlight fundamental similarities and differences, Table 5.1 provides an overview of the spreadsheet smells summarized in this chapter. The table features the following properties:

Target	Spreadsheet parts that can contain the smell.
Oo-pendant	Code Smell this smell is based on, if any.
Cause	Stage of the spreadsheet life cycle at which the smell is usually introduced.
Consequences	Aspects of spreadsheet quality that are affected by the smell.
Alleviation	Options for smell removal.

Name	**Target**					**Oo-pendant**	**Cause**			**Consequences**		**Alleviation**		
	Empty cells	Numeric cells	String cells	Formula cells	Worksheets		Creation	Data entry	Expansion	Erroneous result	Impeded quality	Manual	Assisted	Automated
Std. Deviation		●				-		●		●		●		
Empty Cell	●					-	●	●		●			●	
Pattern Finder	●	●	●	●		-	●	●		●			●	
String Distance			●			-		●		●				●
Ref. to Empty Cells				●		-	●		●	●			●	
QFD		●	●			-		●		●				●
Multiple Operations				●		Long Method			●		●			●
Multiple References				●		Many Parameters			●		●			●
Cond. Complexity				●		-	●		●		●			●
Long Calc. Chain				●		-	●		●		●			●
Duplicated Formulas				●		Duplication			●		●			●
Inappr. Intimacy					●	Inappr. Intimacy	●		●		●	●		
Feature Envy				●		Feature Envy	●		●		●	●		
Middle Man					●	Middle Man			●		●	●		●
Shotgun Surgery					●	Shotgun Surgery	●		●		●	●		

Table 5.1: Comparison of Spreadsheet Smells

5.3 Utilization of Spreadsheet Smells

In the previous section, we summarized which spreadsheet smells have been proposed by the community and how those smells can be detected. In order to benefit from smell detection we need to apply the knowledge about detected smells in some form. Throughout the various examinations of the topic within the scientific community, a number of different approaches have been devised to that end. We organize these approaches into the following 3 categories:

- *Smell Indication* approaches directly provide feedback about detected smells to the user. Feedback may be provided either in-place or in form of additional documents. Users may then manually adapt their spreadsheets based on the additional information.
- *Smell Removal* approaches go a step further and attempt to infer procedures to remove perceived smells from a spreadsheet. Removal of smells usually requires some form of refactoring of the spreadsheet structure. Inferred changes that lead to the removal of smells may either be applied automatically or presented to the user in form of change suggestions.
- *Other Approaches* consult information about perceived smells and conduct further processing of the data to attain a specific goal. However, this goal is not directly related to the indication or removal of perceived smells.

In the following subsections, we describe each category in more detail and state exemplary existing approaches for each of them.

5.3.1 Smell Indication

The straightforward approach to apply the results of smell detection is to accumulate and provide feedback about perceived smells within the spreadsheet. Feedback may be provided either in-place via visual cues or in form of additional documents, diagrams, and charts. Most initial approaches towards the study of spreadsheet smells used such feedback mechanisms. The focus of those early works has been to establish and refine valid smells and detection processes. Smell utilization has been a minor concern at that stage. However, users may still benefit from those approaches. If applied, users may inspect, re-evaluate, and update their spreadsheets based on the additional feedback.

Hermans *et al.* [2012a] were among the first to venture into the scientific field of spreadsheet smells. Their work revolves around detecting and visualizing inter-worksheet smells within spreadsheets. In a previous work [Hermans *et al.*, 2011], they already established a process to extract data flow diagrams from spreadsheets as a suitable option to visualize the inherent structures of spreadsheets. Such diagrams usually provide visual aid to comprehend data dependencies and relationships between processes in information systems. Hermans *et al.* extended their computed diagrams to also indicate detected inter-worksheet smells. Following their established paradigm, boxes that represent worksheets are highlighted using colour if they contain a smell. The hue of the colour indicates the intensity of the detected smell. In addition, tool-tips explain which smell was detected and the concrete location of smell-inducing cells within the respective worksheet.

Another approach by Hermans *et al.* [2012b] focuses on the application of existing code smell principles to formula cells within spreadsheet environments. In contrast to their previous work [2012a], inter-worksheet dependencies are not considered. As a consequence, the authors chose a different visualization method for detected smells. Taking cues from related work [Abraham and Erwig, 2007], they adapted the notion of risk maps to indicate smells in specific formula cells. A 3-tiered, colour-coded overlay over the spreadsheet is used to highlight affected cells. The more intense the colour, the more likely a cell contains a smell. In addition, comments are added to each coloured cell, providing an explanation about the suspected smell.

Cunha *et al.* proposed another example that features the informative approach to smell utilization. They introduced the tool *SmellSheet Detective* [2012b] [2012a]. The purpose of this tool is to detect and indicate a predefined set of smells within provided spreadsheets. Both spreadsheets stored within the Google Docs platform as well as locally stored spreadsheets can be analysed by *SmellSheet Detective*. The tool itself is based on a modular and extensible Java library, which allows for easy incorporation of new smells into the detection process. The result of this process may be exported in the form of either *csv*, *Excel*, or LaTeX tables.

Lastly, Hermans *et al.* [2013] presented another approach in the field of smell detection. In this work, they attempt to automatically detect and highlight data clones in a provided spreadsheet. Data clones are the result of copy-paste operations within a spreadsheet. Not only single cells, but also groups of cells that are likely to be copied are indicated by their approach. Visualization is conducted based on a combination of the techniques, applied in the authors' previous approaches [2012a] [2012b]. A data flow diagram is created, which indicates data clone dependencies between worksheets via directed arrows. In addition, comments are added to affected cells and areas that explain either where specific values were copied to or where the source of specific values can be found within the spreadsheet.

5.3.2 Smell Removal

Although perceived smells do not always indicate errors, they at least point out some flaw that usually can be improved in some form. However, many spreadsheet smells are based on structural properties. Hence, removal of such smells requires changes to the structure of the spreadsheet. Such changes require substantial effort and may lead to the introduction of new issues during the process. Consequently, spreadsheet users often shy away from manually fixing those flaws. Approaches that fall into the removal category of smell utilization attempt to support users in these circumstances by automatically inferring the changes required to remove a specific smell. However, in some instances more than one course of action can be established to remove a perceived smell while none of the possibilities can be procedurally determined as preferred fix for the circumstance. In such cases, a list of possible fixes and/or additional information is generated and proposed to the user. The user may then choose which action to take.

Badame and Dig [2012] proposed an approach to automated smell removal. Their tool, dubbed *REFBOOK*, provides spreadsheet users with access to a suite of automatic refactorings, each removing one commonly encountered spreadsheet smell. The tool is available in form of a plug-in for Microsoft Excel. However, due to the multi-tiered architecture of their tool, it can be easily adapted for other spreadsheet environments as well. Refactoring options are provided to the user via a custom entry in the context menu. After selecting the target cell range of the operation and activating the context menu, a user simply has to choose the desired refactoring from the provided list. The plug-in handles communication of the selected command to the back-end of the tool as well as the application of the necessary changes to the spreadsheet. The back-end process is based on a system of generic spreadsheet entities. This allows the authors of the tool to expand it to other spreadsheet programs by simply supplying a matching add-on for the desired platform while the back-end does not need to be altered. The authors' evaluation of the tool indicates that users in general prefer the refactored output over the initial spreadsheet. They concluded that usage of the *REFBOOK* plug-in resulted in average time savings of more than 50 % in comparison to manual refactoring based on the same conditions. Moreover, Badame and Dig argue that manual refactoring frequently introduces new faults into the spreadsheet, whereas refactorings based on *REFBOOK* does not.

After their extensive work on smell detection, Hermans *et al.* [2014] proposed an approach that incorporates removal of detected smells as well. This approach is based on their previous work regarding smells affecting spreadsheet formulas, presented in previous work [2012b]. Rather than just visualizing detected smells, Hermans *et al.* also attempt to infer refactoring processes to remove those smells from the spreadsheet. Concrete refactoring suggestions are consequently added to the comments of each affected cell. Evaluation of their approach suggested that some detected smells can be reliably removed using the inferred refactoring suggestions. However, spreadsheet users might struggle to implement those refactorings themselves.

5.3.3 Other Approaches

The last category regards approaches that introduce other ideas to utilize the results of smell detection. Smell detection itself is not the focus of such approaches. Rather, information about detected smells is used as an interim result to base further processing on. For example, smell information may be combined with other static analysis techniques to improve accuracy or coverage ratings of existing spreadsheet fault location processes. Such approaches were of no significant interest to the research community as of yet. However, we suspect that more advances will make use of hybrid approaches involving spreadsheet smells in the future.

Abreu *et al.* [2014a] proposed an approach that revolves around further processing of smell detection results. They presume that even if only one cell is indicated as smelly, a number of other cells are likely to contribute to the suspicious circumstance as well. Following this notion, a subset of cells that was identified as smelly beforehand is provided as input for a fault localization algorithm. This algorithm determines a set of cells that are not necessarily smelly themselves, but are likely to cause the smells contained in the spreadsheet. Abreu *et al.* implemented this algorithm in a tool dubbed *FaultySheet Detective* [2014b]. It represents an extension of the *SmellSheet Detective* tool [Cunha *et al.*, 2012a]. In addition to the expanded detection process, *FaultySheet Detective* supports a larger set of spreadsheet smells as a basis of analysis. The tool supports analysis of spreadsheets stored within the Google Docs platform as well as locally stored spreadsheets. Suspicious cells within the spreadsheet that are provided as result by the tool are indicated via background colour. The intensity of the colour hue correlates with the number of faults which were found within the respective cell. In addition, a note is added to each coloured cell explaining which smells are detected for the cell as well as stating the result of the fault localization algorithm for the cell. Evaluation based on a faulty spreadsheet catalogue indicates that *FaultySheet Detective* is capable of identifying more than 70 % of faults within the tested spreadsheets.

5.4 Issues

While spreadsheet smells provide a convenient way to detect and highlight quality issues within spreadsheets, the current status of spreadsheet smells suffers from a number of drawbacks.
First, smells tend to interact with other smells that are based on similar quality metrics. This allows for optimization of various groups of smells against each other. However, as mentioned in Section 5.2, inter-worksheet smells are missing such a counterbalance for optimization. In particular, we require one or more smells which indicate at which point a worksheet should be split into multiple worksheets.
Additionally, spreadsheet smells are often specialized to indicate a specific quality issue. In order to do so, smells greatly narrow their focus of detected quality measures. Established smell ideas could be re-used with a wider focus, allowing for

detection of a wider range of quality issues. For example, the pattern-finder idea could be adapted and expanded to detect pattern mismatches within formula cells. Spreadsheet smells indicate potential quality impairments rather than specific errors. However, numeric thresholds for metric values at which cells are indicated as smelly are statically defined and ignore the current context of the spreadsheet in question. It would be favourable to establish a more dynamic and context-aware method to determine thresholds at which a spreadsheet part is considered smelly.
Various research teams have conducted a variety of in-depth studies related to common errors and quality issues within spreadsheets. However, only a small part of those identified issues are covered by the currently established spreadsheet smells. For example, no smell currently detects duplicated data blocks resulting from unaltered copy & paste operations. Likewise, a variety of useful common code smells and anti-patterns exist which could also be utilized to expand the current spreadsheet smell catalogue. Removal of spreadsheet smells frequently requires a considerable amount of restructuring within the spreadsheet. Manual refactoring therefore is often unattractive for end-users, or even introduces new errors into the spreadsheet. Moreover, the topic of smell removal for spreadsheets has mostly been neglected by the scientific community as of yet. In general, while research related to smell detection is numerous, only a few approaches have been made which allow end-users to directly benefit from detected smells.

6. Structure-Based Spreadsheet Smells

In previous chapters, we introduced the notion of spreadsheet smells, and presented a novel analysis process to infer structural information from a spreadsheet. In this chapter, we combine these established topics to create structure-based spreadsheet smells. While structural information may be applied to enhance various spreadsheet QA techniques, we focus on enhancing spreadsheet smell detection methods following two different approaches: First, we update existing smells by means of structural information. Second, we introduce new smells based on the results of our structural analysis process.

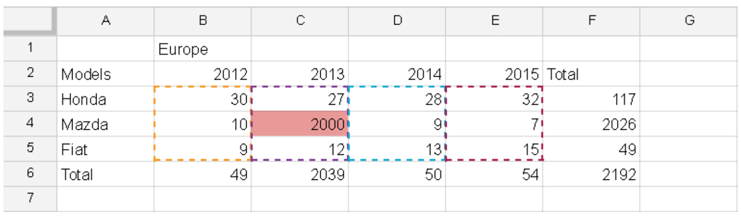
6.1 Updating Existing Smells

In this section, we present enhancements to established spreadsheet smells that can be applied to either groups of related smells, or specific individual smells. We provide examples, illustrating the enhanced detection methods in a specific case. Lastly, we discuss possible limitations of the proposed enhancements.

6.1.1 Sliding Window Smells.

Standard Deviation, Empty Cell, and Pattern Finder smells attempt to find abnormalities in areas within worksheets. The related detection processes utilize sliding windows to detect these abnormalities. For each position of the sliding window, smell detection attempts to establish a pattern. For windows where pattern recognition is successful, smell detection detects cells within the window whose values deviate from the established pattern. If any deviating cell is detected, this cell is reported as smelly. However, dimensions of the employed detection windows are not related to the characteristics of the analysed spreadsheet. Depending on each case, this may either lead to low positive detection rates or a high degree of false positive detections of these smells. To overcome this limitation, we propose to employ the same abnormality-detection processes to the groups established by structural analysis, instead. In particular, cells within partitioned formula groups and reference-based groups share a common functionality in regard to worksheet calculations. Thus, detected abnormalities within these groups are likely to be more meaningful than abnormalities within arbitrary-dimensioned sliding windows.

Figure 6.1 provides an example based on the car-sales worksheet. Cell C4 was altered and now contains an inflated value. Reference-based groups in vertical orientation are highlighted by hatched borders. Application of the Standard Deviation smell on the reference-based group in area C3:C5 detects the smell. Depending on the utilized checking-window, the conventional Standard Deviation detection method may not be able to detect the issue. The column header C2 is also a numeric value which could influence the statistic model used to check for deviating cell values. Likewise, conventional Standard Deviation detection might indicate header cells in the area B2:E2 as smelly, as these do not fit the statistical models established by underlying value cells. If cell A4 was altered to a Boolean value instead, application of the Pattern Finder detection method to the reference-based group in area C3:C5 would detect the issue. If cell A4 was altered to an empty cell, application of the Empty Cell detection method to the reference-based group in area C3:C5 would detect the issue.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	2000	9	7	2026	
5	Fiat	9	12	13	15	49	
6	Total	49	2039	50	54	2192	
7							

Figure 6.1: Example for enhanced sliding-window smells. The cell C4 contains an inflated value. Reference-based groups in vertical orientation for the worksheet are indicated by hatched borders. The Standard Deviation smell detection applied on the reference-based group in area C2:C5 detects the issue.

The structural analysis approach proposed in our work relies on formula relations within worksheets. Consequently, group-information for non-empty cells that are not part of such relations is not currently available. Ideas to infer structural information for neighbouring non-related value cells are presented in Section 8.

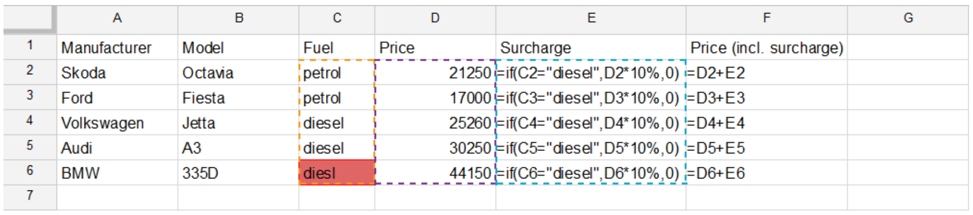
6.1.2 Similarity-Based Smells.

The Quasi-Functional dependencies and String Distance smells rely on dependencies between value cells located within various parts of a worksheet. QFD focusses on dependencies between columns of a worksheet. String Distance compares each pair of strings within a worksheet. However, different areas of a worksheet may be utilized for different purposes by spreadsheet users. Consequently, pattern matching for entire columns of a worksheet is not feasible in all circumstances. Similarly, comparison of strings located in structurally different parts of a worksheet is likely inconclusive.

To enhance the performance of these dependency-based smells, we propose to apply the specific analysis approaches with regard to groups established by structural analysis. This could potentially reduce the amount of false positive smell detections, as well as reduce analysis effort.

Figure 6.2 provides an example for the detection of the occurring String Distance smell. The worksheet illustrates a catalogue of car models. For each car, the manufacturer, model, fuel type, and price is listed. For cars using diesel fuel, a surcharge of 10 % is added to the total price of the car. Cell C6 contains an error, stating the string "diesl" instead of "diesel". This leads to the calculation of a wrong surcharge in Cell E6, and consequently a wrong total price in Cell F6. The faulty cell contains a string that is minimally diverging from the correct "diesel" strings in cells C4 and C5. Classical String Distance detection also detects this instance. However, classical detection requires to match strings of for each column and row of the worksheet. Thus, string detection would also analyse headers in row 1, as well as model and manufacturer names in columns A and B. Application of group-based String Distance detection also detects the issue, while only matching strings within the reference-based group C2:C6.

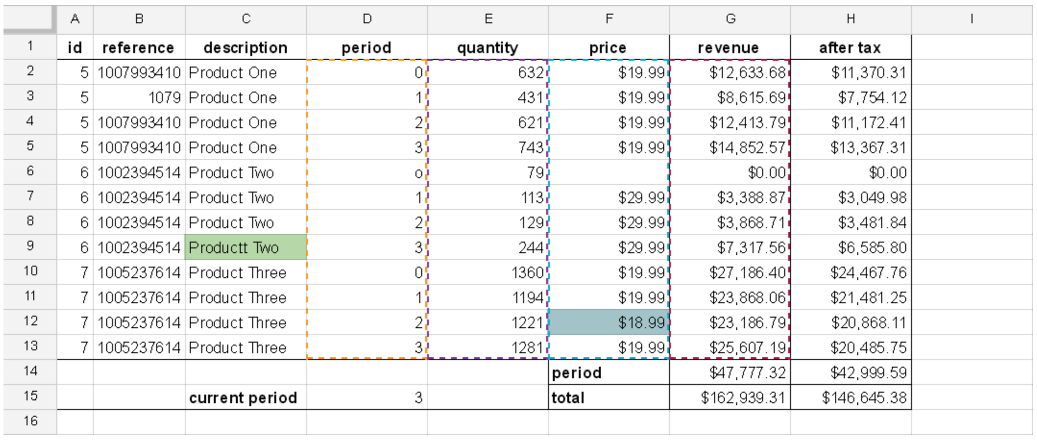
Figure 6.3 provides an example based on the sales worksheet. Cell C9 contains a minimally diverging string. Cell F12 contains a value which deviates from quasi-functional dependencies between columns F, B and C. Reference-based groups



	A	B	C	D	E	F	G
1	Manufacturer	Model	Fuel	Price	Surcharge	Price (incl. surcharge)	
2	Skoda	Octavia	petrol	21250	=if(C2="diesel",D2*10%,0)	=D2+E2	
3	Ford	Fiesta	petrol	17000	=if(C3="diesel",D3*10%,0)	=D3+E3	
4	Volkswagen	Jetta	diesel	25260	=if(C4="diesel",D4*10%,0)	=D4+E4	
5	Audi	A3	diesel	30250	=if(C5="diesel",D5*10%,0)	=D5+E5	
6	BMW	335D	diesl	44150	=if(C6="diesel",D6*10%,0)	=D6+E6	
7							

Figure 6.2: Example for enhanced String Distance smell. The worksheet illustrates a car catalogue. Surcharges are added based on the fuel type. Reference-based groups of the worksheet are indicated with hatched borders. Cell C6 is faulty, containing a wrong string. The cell is inferred as smelly by applying String Distance detection on the group in area C2:C6.

detected by our structural analysis process are indicated by a hatched border. Both deviating cells are detectable by established smell detection methods. However, QFD detection also includes header-strings in row 1 as well as footer strings and values in rows 14 and 15 into their matching process. Structural information allows to apply the matching processes on established structures instead. However, the structural analysis approach proposed within our work relies on formula relations within worksheets. Group-information for non-empty cells which are not part of such relations is not currently available. Consequently, the detection process to infer the smell in cell C9 cannot be enhanced by available structure information. This points out a shortcoming of the structural analysis process. An approach to correct this shortcoming is outlined in Section 8.



	A	B	C	D	E	F	G	H	I
1	**id**	**reference**	**description**	**period**	**quantity**	**price**	**revenue**	**after tax**	
2	5	1007993410	Product One	0	632	$19.99	$12,633.68	$11,370.31	
3	5	1079	Product One	1	431	$19.99	$8,615.69	$7,754.12	
4	5	1007993410	Product One	2	621	$19.99	$12,413.79	$11,172.41	
5	5	1007993410	Product One	3	743	$19.99	$14,852.57	$13,367.31	
6	6	1002394514	Product Two	o	79		$0.00	$0.00	
7	6	1002394514	Product Two	1	113	$29.99	$3,388.87	$3,049.98	
8	6	1002394514	Product Two	2	129	$29.99	$3,868.71	$3,481.84	
9	6	1002394514	Productt Two	3	244	$29.99	$7,317.56	$6,585.80	
10	7	1005237614	Product Three	0	1360	$19.99	$27,186.40	$24,467.76	
11	7	1005237614	Product Three	1	1194	$19.99	$23,868.06	$21,481.25	
12	7	1005237614	Product Three	2	1221	$18.99	$23,186.79	$20,868.11	
13	7	1005237614	Product Three	3	1281	$19.99	$25,607.19	$20,485.75	
14						**period**	$47,777.32	$42,999.59	
15			**current period**	3		**total**	$162,939.31	$146,645.38	
16									

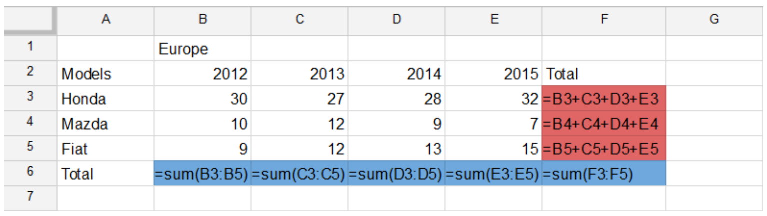
Figure 6.3: Example for enhanced similarity-based smells. The cell C9 is affected by the String Distance smell. The cell F12 is affected by the QFD smell. Reference-based groups in vertical orientation for the worksheet are indicated by hatched borders. Application of the related detection methods to grouped cell areas could provide higher detection accuracy while limiting processing demands.

6.1.3 Formula-Based Smells.

Multiple Operations, Multiple References, Conditional Complexity, and Feature Envy smells detect issues regarding formula cells of a worksheet. Consequently, the smell detection methods of these smells are applied to every formula cell within a worksheet. However, the applied smell detection methods are independent of the specific position of the formula, as well as independent of the specific position and the content of referred cells.

Spreadsheet users often utilize the same formula for each cell of a column or row within a worksheet. Consequently, the same calculation is applied to different input values. Our analysis process detects such calculation groups in form of formula groups. Hence, we propose to apply the same detection methods to the formulas of formula groups, instead. Following this approach, relevant smell detection methods need to be applied only once for the entire formula group. Analysing each individual cell is not required. Depending on the structure of the worksheet, this could provide significant benefits in regard to processing effort. Moreover, removal of formula-based smells either requires transformation of the related formula, or restructuring of the worksheet. In case of a required formula transformation, the suitable refactoring operation can be processed once and applied to each formula cell within the group. In case of a required worksheet restructuring, the suitable refactoring operation can be processed for the entire formula group, keeping the overall structure- and group-dependencies in mind.

Figure 6.4 provides an example based on the car-sales worksheet. The formulas in cells F3, F4, and F5 were altered such that they refer to each value in the same row individually. Each of these cells features more than three references to other cells. Consequently, the Multiple References smell is detected for these cells. The formula group in area F3:F5 is based on the same formula. Hence, the group is also featuring three group references. Group-based smell detection marks the entire group as smelly, but does not require checking each individual cell.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=B3+C3+D3+E3	
4	Mazda	10	12	9	7	=B4+C4+D4+E4	
5	Fiat	9	12	13	15	=B5+C5+D5+E5	
6	Total	=sum(B3:B5)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							

Figure 6.4: Example for enhanced formula-based smells. The cells F3, F4, and F5 are affected by the Multiple References Smell. Formula groups of the worksheet are indicated by a matching background-colour. Group-based detection would detect the smell for the formula group in area F3:F5.

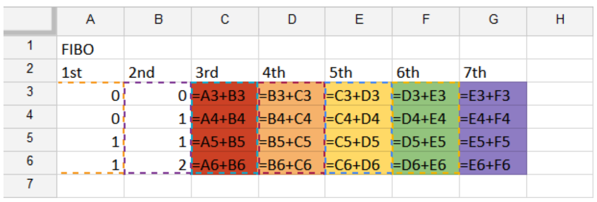
6.1.4 Long Calculation Chain.

The Long Calculation Chain smell detects formula cells which refer to a long chain of previous calculations. To detect this smell, reference chains are calculated and checked for each individual cell in the worksheet. Our structural analysis approach discerns formula groups and relations between formula groups of a worksheet. This allows us to establish calculation chains for formula groups of a spreadsheet.

Consequently, we propose to base detection of the Long Calculation Chain smell on formula group references. Cell-based detection of long calculation chains requires processing of all references of each formula cell. Likewise, group-based detection of the smell requires processing of all references of each formula group. However, the number of formula groups tends to be significantly lower than the number of formula cells. Moreover, processing of an area-reference of a formula cell establishes a reference from the formula cell to each individual cell within the target area. In contrast, processing of an area-reference of a formula group only establishes a group reference from the formula group to each of the groups inferred for the target area. However, the number of inferred groups for an area reference tends to be significantly lower than the number of individual cells within the area. Hence, depending on the structure of the spreadsheet, employing formula group analysis may lead to a significant reduction of the required processing effort. In the worst case scenario, each formula cell in the spreadsheet is part of an individual formula group, and group-based analysis of area-references infers a group for each cell within the referred area. In this scenario, group-based detection of the Long Calculation Chain smell is equivalent to the established cell-based detection process.

Removal of long calculation chain smells from a spreadsheet usually requires transformation of spreadsheet structures. Structural transformations, even regarding a single formula cell, require special attention to the surrounding groups. As an additional benefit of group-based detection, this structural context is already present for the smelly group.

Figure 6.5 provides an example based on the Fibonacci-calculation worksheet. The worksheet calculates numbers of the Fibonacci-series based on different starting values. Formula groups of the worksheet are indicated by matching background colour. Evaluation of the cells G3, G4, G5, and G6 requires checking of more than four individual cells. Consequently, the Long Calculation Chain smell is detected for these cells. Likewise, the formula group encompassing area G3:G6 requires processing of a chain of more than four individual group references. Consequently, the Long Calculation Chain smell is detected for the cells within the group. Based on this method, the same smelly cells can be deduced. However, cell-based smell detection requires reference processing for each individual formula cell of the worksheet. This amounts to a total number of 40 cell-references that need to be processed in this example. In contrast, group-based smell detection requires reference processing for each formula group of the worksheet. This amounts to a total number of 10 group-references that need to be processed in this example.



	A	B	C	D	E	F	G	H
1	FIBO							
2	1st	2nd	3rd	4th	5th	6th	7th	
3	0	0	=A3+B3	=B3+C3	=C3+D3	=D3+E3	=E3+F3	
4	0	1	=A4+B4	=B4+C4	=C4+D4	=D4+E4	=E4+F4	
5	1	1	=A5+B5	=B5+C5	=C5+D5	=D5+E5	=E5+F5	
6	1	2	=A6+B6	=B6+C6	=C6+D6	=D6+E6	=E6+F6	
7								

Figure 6.5: Example for enhanced Long Calculation Chain smell. The cells G3, G4, G5, and G6 are affected by the Long Calculation Chain smell. Formula groups of the worksheet are indicated by a matching coloured background. Reference-based groups of the worksheet are indicated by a hatched border. Group-based detection would detect the smell for the formula group in area G3:G6.

Processing of group-based references relies on the results of reference group matching. However, spreadsheet references do not always allow for neat matching of groups. For example, a group-reference may only partially match the area of another formula group. Such cases need to be handled accordingly. To highlight unresolved or inconsistent matching-results, we propose the novel Inconsistent Formula group References smell in Section 6.2

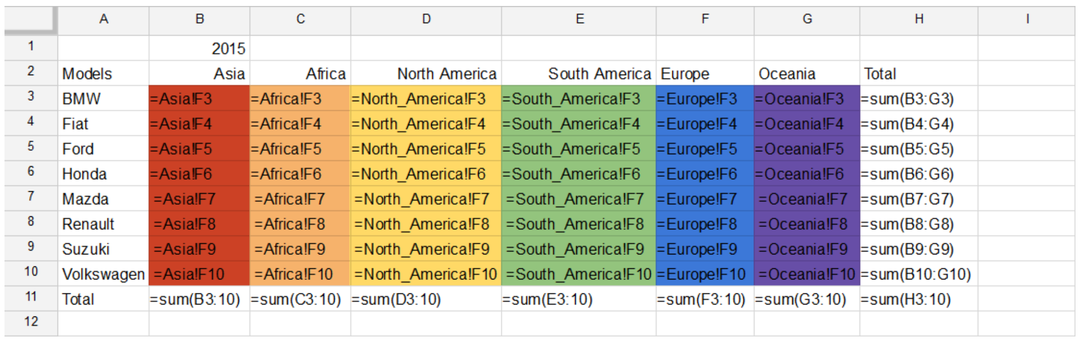
6.1.5 Inter-Worksheet Smells.

Inappropriate Intimacy, Middle Man and Shotgun Surgery smells rely on metrics which take inter-worksheet references of individual formulas into consideration. Detection of these smells is based on the number of cells that contain such references in a specific worksheet.

We propose to utilize inter-worksheet formula group relations as basis for smell detection instead. Hence, counting individual cells which refer to another worksheet is exchanged by counting the number of individual formula groups which refer to another worksheet. This would, however, alter the focus of the related smells. Cell-based analysis quantifies the raw number of inter-worksheet connections. In contrast, group-based analysis quantifies how many different purposes for connections exist between two worksheets. Each group-reference indicates a specific purpose in the calculation narrative of the worksheet.

Group-based analysis could also support consideration of the amount of individual connections. To that end, formula group reference analysis may be extended to also account for multiplicity of the base group and target groups of each reference. We outline our ideas for multiplicity quantification of formula group references in Section 8.

Figure 6.6 provides an example based on the car-sales worksheet. The worksheet summarizes car sales for the year 2015 for different continents. The sales numbers for 2015 of each continent are imported from individual worksheets for each continent. The cells in rows three to ten in columns B to G refer to matching columns containing the sales numbers for a specific continent. Each individual column features more than seven references to one other worksheet. Consequently, each individual column leads to the detection of the Inappropriate Intimacy smell for the worksheet. However, each cell of each individual column is part of a single formula group of this column. Group-based analysis would reveal that only one connection purpose exists between two individual worksheets. Consequently, the worksheet may not necessarily be indicated as smelly.



	A	B	C	D	E	F	G	H	I
1		2015							
2	Models	Asia	Africa	North America	South America	Europe	Oceania	Total	
3	BMW	=Asia!F3	=Africa!F3	=North_America!F3	=South_America!F3	=Europe!F3	=Oceania!F3	=sum(B3:G3)	
4	Fiat	=Asia!F4	=Africa!F4	=North_America!F4	=South_America!F4	=Europe!F4	=Oceania!F4	=sum(B4:G4)	
5	Ford	=Asia!F5	=Africa!F5	=North_America!F5	=South_America!F5	=Europe!F5	=Oceania!F5	=sum(B5:G5)	
6	Honda	=Asia!F6	=Africa!F6	=North_America!F6	=South_America!F6	=Europe!F6	=Oceania!F6	=sum(B6:G6)	
7	Mazda	=Asia!F7	=Africa!F7	=North_America!F7	=South_America!F7	=Europe!F7	=Oceania!F7	=sum(B7:G7)	
8	Renault	=Asia!F8	=Africa!F8	=North_America!F8	=South_America!F8	=Europe!F8	=Oceania!F8	=sum(B8:G8)	
9	Suzuki	=Asia!F9	=Africa!F9	=North_America!F9	=South_America!F9	=Europe!F9	=Oceania!F9	=sum(B9:G9)	
10	Volkswagen	=Asia!F10	=Africa!F10	=North_America!F10	=South_America!F10	=Europe!F10	=Oceania!F10	=sum(B10:G10)	
11	Total	=sum(B3:10)	=sum(C3:10)	=sum(D3:10)	=sum(E3:10)	=sum(F3:10)	=sum(G3:10)	=sum(H3:10)	
12									

Figure 6.6: Example for update of Inter-Worksheet smells. Formulas in rows three to ten in columns B to G refer to values in columns of various other worksheets. Formula groups of inter-worksheet formulas are indicated by a coloured background. Each column features more than seven relations to a specific other worksheet. Consequently, each individual row causes the Inappropriate Intimacy smell for the worksheet. Group-based analysis would reveal that only one purpose for references exists between each individual worksheet.

6.2 Introducing Novel Spreadsheet Smells

The established catalogue of spreadsheet smells was introduced without the possibilities of structural information in mind. Nevertheless, this additional information source provides a number of starting points to formulate new smells. In this section, we present various approaches to introduce new spreadsheet smells based on the results of structural analysis. Smell proposals follow the form established in Section 5.1, discussing the attributes *Origin & Intent*, *Target*, *Detection*, *Example*, *Cause*, *Consequences*, and *Alleviation*.

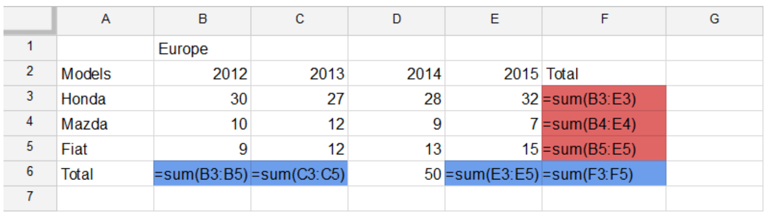
6.2.1 Duplicated Formula Groups

Origin & Intent. The Duplicated Formula Groups smell is based on the principle of the Duplicated Formulas smell. The Duplicated Formulas smell detects multiple occurrences of the same formula within the worksheet. Different interpretations of this smell have been established by various contributors. We agree on the basic principle of the idea shared by all of these. One specific interpretation regards the R1C1-representation of formulas as basis for smell detection. This interpretation states that formulas should be indicated as smelly if they share an identical R1C1-representation. Formula cells which share the same R1C1-representation apply identical calculations based on different input data. Neighbouring cells fulfilling identical calculations are a common occurrence in calculation-based spreadsheets. Therefore, we argue that this behaviour is intended when applying to neighbouring cells and should not be regarded as smelly. However, non-neighbouring cells fulfilling identical calculations may indicate bad spreadsheet structuring. Such formula cells are situated in different formula groups of a worksheet. Hence, we propose the Duplicated Formula Groups smell, which detects multiple formula groups within the same worksheet whose formulas share the same R1C1-representation.

Target. Duplicated Formula Groups detects quality issues based on the formula groups of a worksheet. Consequently, the detection process for this smell can only be applied to formula groups of a worksheet.

Detection. Detection of the Duplicated Formula Groups smell requires comparison of formula groups of a worksheet. Each formula group of a worksheet is compared with all other formula groups of the same worksheet. If two formula groups feature the same formula in R1C1-representation, both groups are indicated as smelly.

Example. An occurrence of the Duplicated Formula Groups smell can be seen in Figure 6.7. The worksheet is an adaptation of the previous car-sales worksheet. Cells belonging to a formula group are indicated by matching background colour. The cell D6 was overwritten by its cell value. Two separate formula groups are detected in row 6: one formula group in area B6:C6, and one formula group in area E6:F6. Both groups share the same formula in R1C1-representation. Consequently, Duplicated Formula Groups detection indicates both groups as smelly.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B5)	=sum(C3:C5)	50	=sum(E3:E5)	=sum(F3:F5)	
7							

Figure 6.7: Example for Duplicated Formula Groups smell. Formulas sharing the same representation in R1C1-notation share the same background-colour. Formula groups of areas B6:C6 and E6:F6 share the same formula in R1C1-representation and are therefore indicated as duplicate.

Cause. The occurrence of multiple groups using the same formula implies that multiple areas within the worksheet fulfil the same basic function. Such groups may occur following two different rationales: First, the two matching groups were initially part of the same group but were split because of a separating cell. Such cells may occur by users editing or deleting the content of one of the cells within the group, or by users introducing a new cell in the middle of the group. Second, the worksheet features separate areas which apply the same calculations on different sets of data. This may be intended by the creator of the worksheet. However, this could also indicate an error. The spreadsheet user may have intended to copy and edit the related formula group, but did not apply the edit operation after copying the cells.

Consequences. The smell points out instances of multiple formula groups that apply the same calculation. Possible consequences of such instances depend on the circumstances of their occurrence. Cases where matching groups were introduced unintentionally indicate faults in the worksheet. Cases where matching groups are the result of intentional editing may reduce comprehensibility and maintainability of a worksheet.

Alleviation. Correction of the Duplicated Formula Groups smell is possible following one of three approaches:

- *Merge both formula groups.* This approach is only applicable for matching groups created by separation of a cohesive group. In such occurrences, the operation which separated the base formula group can be reversed, resulting in a single, cohesive formula group. The smell in the previous example can be removed by this approach, fixing the fault in cell D6.
- *Edit one group.* Editing the group-formula of one of the matching groups removes the smell from both groups. This is applicable in cases where one of the matching formula groups was introduced by mistake, copying the base group but forgetting to edit the copied group.
- *Relocate one group to another worksheet.* Relocating one of the matching groups to another worksheet removes the smell for both groups. This is applicable in cases where both groups intentionally apply the same calculation.

6.2.2 Formula Group Distance

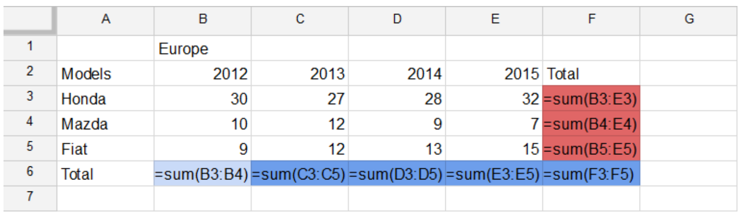
Origin & Intent. The Formula Group Distance smell is based on the principle of the String Distance smell. The String Distance smell detects string cells which are similar to other string cells within the same worksheet. Smell detection is based on the premise that the spreadsheet creator initially wanted to enter the exact same string, but made an error. Formula Group Distance applies the same idea to detect neighbouring similar formula groups within a worksheet. Such occurrences imply that the spreadsheet creator wanted to enter the exact same formula, but made an error. Thus, the Formula Group Distance smell detects neighbouring formula groups whose R1C1-representation is very similar.

Target. Formula Group Distance detects quality issues based on the formula groups of a worksheet. Consequently, the detection process for this smell can only be applied to formula groups of a worksheet.

Detection. Detection of the Formula Group Distance smell requires comparison of formula groups. Each formula group of a worksheet is compared with other formula groups of the same worksheet. If two formula groups are neighbours and feature similar formulas in R1C1-representation, both groups are indicated as smelly.

Example. An occurrence of the Formula Group Distance smell can be seen in Figure 6.8. The example is based on the car-sales worksheet. Cell B6 of the worksheet was altered to only refer to rows three and four of the same column. This cell creates a formula group of its own, encompassing the area B6:B6. Cells of the neighbouring area C6:F6 share the same formula in R1C1-notation. Thus, they create a formula group encompassing this area. The formula group in area B6:B6 is neighbour to the formula group in area C6:F6. The formulas of both groups only differ by a single coordinate-reference, implying similarity of the formulas. The groups are neighbouring and similar and therefore indicated to be affected by the Formula Group Distance smell.

Cause. The occurrence of neighbouring formula groups sharing similar formulas may occur following two different rationales: First, a spreadsheet user expands an existing formula group, but enters a different formula. The different formula may either be chosen intentionally or by mistake. Consequently, a new, neighbouring formula group is introduced which features a similar formula. Second, a spreadsheet



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B4)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							

Figure 6.8: Example for Formula group Distance smell. Cells within the same formula group share the same background-colour. The formula group in area B6:B6 is neighbour of the formula group in area C6:F6. The formulas of both groups are similar. Consequently, both formula groups are indicated as smelly of Formula Group Distance.

user edits the formula of a cell belonging to an existing formula group, introducing a minor change. Consequently, the edited cell establishes a new formula group of its own, featuring a similar cell formula. The new formula group is always located next to the remains of the initial formula group.

Consequences. Formula Group Distance points out instances of neighbouring formula groups which feature similar group formulas. Possible consequences of such occurrences depend on the circumstances of their introduction. Cases where similar neighbouring groups were introduced unintentionally indicate faults in the worksheet. Cases where such groups are the result of intentional editing or expansion may reduce comprehensibility and maintainability of a worksheet.

Alleviation. Correction of the Formula Group Distance smell is possible following one of three approaches:

- *Merge both formula groups.* Following this approach, the formula of one of the neighbouring groups is edited to match the formula of the other group. This action unifies both groups into a cohesive formula group. Instances where the smell was introduced by mistake can be alleviated by this approach.

- *Edit one group.* Following this approach, the group formula of one of the neighbouring groups is edited so that it is no longer similar to the formula of the neighbouring group. Instances where the smell was introduced by mistake can be alleviated by this approach. However, depending on the dimensions and locations of the groups, this may introduce the Unrelated Neighbours smell, which is presented in Subsection 6.2.3.

- *Relocate one group.* Following this approach, one of the groups is relocated either to another position within the worksheet, or to another worksheet. As the groups are no longer neighbouring, this removes the smell from both groups. This approach is applicable in cases where the smell was introduced by intentional changes to the worksheet.

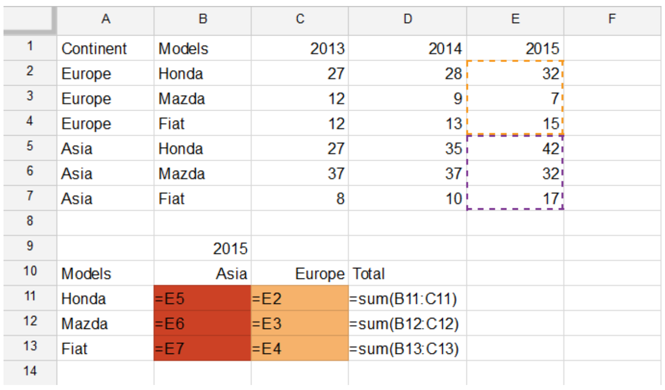
6.2.3 Unrelated Neighbours

Origin & Intent. The Unrelated Neighbours smell points out areas within a worksheet which are hard to comprehend. It is based on the results of our structural analysis process. During this process, we detect areas within a worksheet which fulfil specific purposes. Cells within areas encompassed by formula groups apply the same calculations. Cells within the areas encompassed by reference-based groups serve as input to specific formula groups. It is common convention that cells within visually distinct areas (e.g. within the same column of a calculation block) fulfil the same purpose within the worksheet. Placement of multiple areas with different purposes right next to each other without clear distinction violates this convention, impairing comprehensibility of the worksheet. However, detection of neighbouring but diverging areas within a worksheet is no easy task for spreadsheet users, as areas are only represented by the evaluated values of each cell. Thus, the Unrelated Neighbours smell detects pairs of formula groups or reference-based groups which share the same orientation and appear next to each other in this orientation. Neighbouring groups perpendicular to their orientation are common in worksheets and should therefore not be indicated as smelly (e.g. successive columns featuring different calculations). Likewise, single-cell formula groups next to other formula groups or reference-based groups are common in footer areas of worksheets and should also not be indicated as smelly.

Target. Unrelated Neighbours detects quality issues based on formula groups and reference-based groups of a worksheet. Consequently, the detection process for this smell is applied to formula-groups and reference-based groups of a worksheet.

Detection. Detection of the Unrelated Neighbours smell requires comparison of the detected groups of a worksheet. Each group of the worksheet is compared with each other group of the same worksheet. If two groups share the same orientation, and appear as neighbours in this orientation, and neither of them is a single-cell formula group, both groups are indicated as smelly.

Example. Figure 6.9 provides an example for the Unrelated Neighbours smell. The worksheet lists car sales of different car models in different continents over various years. In a separate area, the worksheet calculates the total of sold cars by car model for the year 2015. The calculation of totals in area D11:D13 refers to the values for each continent in areas B1:B13 and C11:C13. These values are provided by formulas referring to areas within the initial data set. The formula group in area B11:B13 refers to the area E2:E4, establishing a reference-based group for this area. The formula group in area C11:C13 refers to the area B5:B7, establishing a reference-based group for this area. Thus, the worksheet contains two reference-based groups at areas E2:E4 and E5:E7 which share the same vertical orientation and are neighbours in this orientation. Detection of the Unrelated Neighbours smell indicates both groups as smelly. The formula groups in areas B11:B13 and C11:C13 also share the same vertical orientation. However, they are not neighbours in vertical orientation. Therefore, the Unrelated Neighbours smell does not apply to these groups.



	A	B	C	D	E	F
1	Continent	Models	2013	2014	2015	
2	Europe	Honda	27	28	32	
3	Europe	Mazda	12	9	7	
4	Europe	Fiat	12	13	15	
5	Asia	Honda	27	35	42	
6	Asia	Mazda	37	37	32	
7	Asia	Fiat	8	10	17	
8						
9		2015				
10	Models	Asia	Europe	Total		
11	Honda	=E5	=E2	=sum(B11:C11)		
12	Mazda	=E6	=E3	=sum(B12:C12)		
13	Fiat	=E7	=E4	=sum(B13:C13)		
14						

Figure 6.9: Example for Unrelated Neighbours smell. Cells of relevant formula groups are indicated via matching background-colour. Relevant reference-based groups are indicated by a hatched border. The formula groups in areas B11:B13 and C11:C13 point to the reference-based groups in areas E2:E4 and E5:E7. These reference-based groups share the same orientation (vertical) and are neighbours in this orientation. Consequently, detection of the Unrelated Neighbours smell indicates both reference-based groups as smelly.

Cause. The main contributing factor for the occurrence of Unrelated Neighbours between reference-based groups is poor spreadsheet design. During creation of the spreadsheet, specific areas of a worksheet are planned to handle specific types of data. Unrelated Neighbours occur if any individual area is concerned with too many different types of data. Consequently, formulas that work on the data within these areas are required to refer to specific neighbouring sub-areas, instead of referring to the area as a whole. Occurrence of Unrelated Neighbours involving any formula group indicates another type of poor spreadsheet design. A formula group placed next to a reference-based group translates to multiple formula cells which apply the same calculation being placed next to multiple value cells within the same column or row. A formula group occurring next to another formula group translates to multiple formula cells which apply the same calculation being placed next to a set of other formula cells which apply another calculation.

Consequences. Unrelated Neighbours points out instances of neighbouring groups which are not related to each other. Such instances usually imply the existence of multiple sub-areas within the same column or row which fulfil different purposes. However, when working with spreadsheets, no indication of such groups is usually available. Groups are merely represented by the values of their individual cells. Hence, different sub-areas within the same column or row of a table are hard to distinguish. As a consequence, users are more likely to introduce faults when editing or expanding affected areas of a worksheet.

Alleviation. Correction of the Unrelated Neighbours smell depends on the circumstance which caused the smell. In cases where formula groups are located next to other groups, the relevant groups may be merged or one of the groups relocated. In cases where multiple reference-based groups are located next to each other, removal of the smell is usually not trivial. Such instances point out calculation blocks which fulfil too many purposes. Hence, removal of the smell requires separation of the calculation block into multiple different blocks, each handling a subset of the initial purposes. The example in Figure 6.9 illustrates such an instance. Removal of the smell requires the separation of the calculation block in area A2:E7 into two different calculation blocks, each relating to cars of either Europe or Asia.

6.2.4 Inconsistent Reference Dimensions

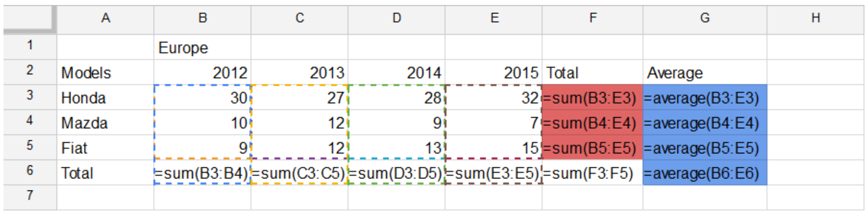
Origin & Intent. The Inconsistent Reference Dimensions smell points out inconsistent referenced areas within a worksheet. It is based on the results of our structural analysis process. During this process, we detect reference-based groups within a worksheet. The areas encompassed by these groups serve as input to specific formula groups. Multiple formula groups referring to overlapping areas within the worksheet introduce overlapping reference-based groups. Areas of two reference-based groups sharing the same orientation, overlapping each other, but not exactly matching one another, imply that individual cells of these areas fulfil diverging purposes. This can negatively effect the maintainability of the related worksheet.

Target. Inconsistent Reference Dimensions detects quality issues based on reference-based groups of a worksheet. Consequently, the detection process for this smell is applied to reference-based groups of a worksheet.

Detection. Detection of the Inconsistent Reference Dimensions smell requires comparison of the detected reference-based groups of a worksheet. Each group of the worksheet is compared with all other reference-based groups of the same worksheet. If two groups share the same orientation and have at least one cell in common, but do not encompass the same area, both groups are indicated as smelly.

Example. Figure 6.10 provides an example for the Inconsistent Reference Dimensions smell. The example is an adaptation of the car-sales worksheet. Column G calculates the average sales numbers for each car model over the entire timespan. However, cell G6 also calculates the average of the total sales numbers over the timespan. This is inconsistent with the present layout of the table. The formula group in area F3:F5 established a reference-based group encompassing rows three to five in columns B to E. The newly introduced formula group in column G introduces an additional reference-based group for each of these columns, encompassing rows three to six instead. Consequently, columns B to E each feature two reference-based groups, sharing the same orientation, overlapping in rows three to five, but not matching in row six. Hence, detection of the Inconsistent Reference Dimensions smell indicates each pair of reference-based groups in columns B to E as smelly.

Cause. Inconsistent Reference Dimensions may be introduced following one of two different narratives:



	A	B	C	D	E	F	G	H
1		Europe						
2	Models	2012	2013	2014	2015	Total	Average	
3	Honda	30	27	28	32	=sum(B3:E3)	=average(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	=average(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	=average(B5:E5)	
6	Total	=sum(B3:B4)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	=average(B6:E6)	
7								

Figure 6.10: Example for Inconsistent Reference Dimensions smell. Cells of relevant formula groups are indicated via matching background-colour. Relevant reference-based groups are indicated by a hatched border. Formula groups in areas F3:F5 and G3:G6 introduce two reference-based groups in each of the columns B, C, D, and E: One, encompassing rows 3 to 5, and one encompassing rows 3 to 6. Reference-based groups of each column share the same orientation, and encompass areas that overlap but do not match. Hence, detection of the Inconsistent Reference Dimensions smell indicates each pair of reference-based groups as smelly.

- *Inconsistent table layout.* During creation of the spreadsheet, specific areas of a worksheet are planned to handle specific types of data. Inconsistent Reference Dimensions occur if overlapping areas are concerned with diverging types of data. Consequently, formulas that work on the data within these areas are required to refer to overlapping sub-areas, instead of referring to the area as a whole.

- *Inconsistent formula use.* Even in cases where specific areas of the worksheet fulfil only a specific purpose, spreadsheet users may elect to refer to specific sections of these areas. This may lead to overlapping but non-matching areas, introducing the Inconsistent Reference Dimensions smell.

Consequences. The Inconsistent Reference Dimensions smell points out instances of overlapping reference-based groups. Such instances usually imply that the affected area attempts to fulfil diverging purposes. However, when working with spreadsheets, no indication of such groups is usually available. Groups are merely represented by the values of their individual cells. Hence, diverging sub-areas within the same column or row of a table are hard to distinguish. As a consequence, users are more likely to introduce faults when editing or expanding affected areas of a worksheet.

Alleviation. Correction of the Inconsistent Reference Dimensions smell depends on the circumstance which caused the smell. In cases where the smell is a result of inconsistent table layout, major restructuring is required to balance the purposes of the affected area. Such instances may requires separation of the related calculation block into multiple different blocks. In cases where the smell is a result of inconsistent formula use, adaptation of the referring formulas removes the smell. The example in Figure 6.10 illustrates such an instance. The formula group in area F3:F5 is inconsistent with the new formula group in area G3:G6. Changing the formula of cell F6 to "=SUM(B6:E6)" removes the smell in all instances while maintaining the calculation results of the worksheet.

6.2.5 Inconsistent Formula Group Reference

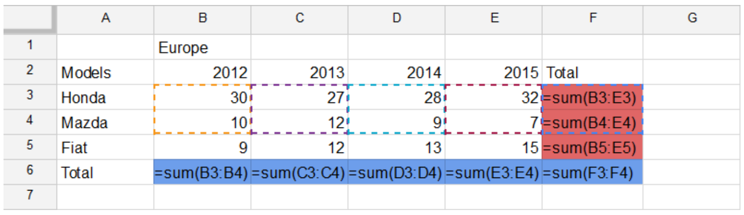
Origin & Intent. The Inconsistent Formula Group Reference smell points out inconsistent references between formula groups within a worksheet. It is based on the results of our structural analysis process. During this process, we detect formula groups within a worksheet. Each reference of a formula group refers to an area within the spreadsheet. If any other formula group encompasses this referred area, we establish a reference between these formula groups. However, inconsistencies may occur, whereby the area of the target group does not exactly match the size of the referred area. Hence, the base formula group either does not refer to all results of the target group, or the base formula group expects a greater number of result cells than the target group can provide. Either case may lead to wrong calculation results. Matching a reference area with a formula group that does not share the same orientation (e.g. a formula group as footer of a block summing up the results of formula groups in the columns of the block) will always result in a maximum of one overlapping cell. Consequently, this case is excluded from the smell detection process.

Target. Inconsistent Formula Group Reference detects quality issues based on formula groups of a worksheet. Consequently, the detection process for this smell is applied to formula groups of a worksheet.

Detection. Detection of this smell within a worksheet requires comparison of referred areas of each formula group with other detected formula groups of a worksheet. For each formula group of the spreadsheet, the referred areas of the group are processed. Each area is compared with each formula group of the worksheet.

If the referred area and the area encompassed by another formula group share the same orientation and overlap, but are not identical, this case is reported as smelly.

Example. Figure 6.11 provides an example for the Inconsistent Formula Group Reference smell. The example is an adaptation of the car-sales worksheet. The formula group in area B6:F6 contains a fault, only calculating the sum of the values in rows three and four. Matching of referred areas with formula groups attempts to match area F3:F4 with the formula group encompassing area F3:F5. The compared areas share the same vertical orientation, overlap, but do not match exactly. Hence, this instance is indicated as smelly.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B4)	=sum(C3:C4)	=sum(D3:D4)	=sum(E3:E4)	=sum(F3:F4)	
7							

Figure 6.11: Example for Inconsistent Formula group Reference smell. Cells of formula groups are indicated via matching background-colour. Reference areas of the formula group in area B6:F6 are indicated by a hatched border. The detection process matches the referred area F3:F4 with the formula group F3:F5. The areas share the same orientation (vertical), overlap, but do not match. Hence, this matching is indicated as smelly.

Cause. Inconsistent Formula Group Reference detects instances where a referring formula group either does not refer to all results of a referred formula group, or a referring formula group expects a greater number of result cells than the referred formula group can provide. We argue that this may occur following one of three scenarios:

- *Faulty formula initialization.* During creation of the spreadsheet, an error occurred at the initialization of the referring formula group.
- *Inconsistent edit (target).* A spreadsheet user alters the size of the referred group after the relation has already been established, but does not update the referring group to match the new size.
- *Inconsistent edit (source).* A spreadsheet user alters the reference of the referring group group after the relation has already been established, but does not update the referred group to match the new size.

Consequences. Inconsistent Formula Group Reference points out instances of formula groups inconsistently referring to another formula group. The base group either does not refer to every cell of the target group, or expects a greater number of result cells than is provided by the target group. In both cases, this is likely to lead to an erroneous calculation result within the source group.

Alleviation. In cases where the smell was introduced by inconsistent edit of the target group, reversal of this edit or adaptation of the source group to match the new dimensions removes the smell. In cases where the smell was introduced by inconsistent edit of the source group, reversal of this edit or adaptation of the target group to match the new referred area removes the smell. The example in Figure 6.11 illustrates such an instance. The formula group in area B6:F6 was altered and is now inconsistent with the referred group in area F3:F6. Either reversal of the edit by changing the formula of formula group B6:F6 to encompass row five or updating the target group by deleting cell F5 removes the smell.

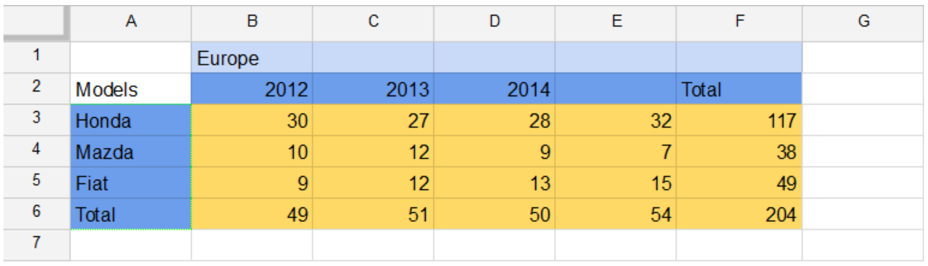
6.2.6 Missing Header

Origin & Intent. The Missing Header smell points out vacant headers for columns or rows of blocks within a worksheet. It is based on the results of our structural analysis process. During this process, we detect blocks within a worksheet. Blocks are concise areas within a worksheet which contain neighbouring groups of cells. Moreover, the analysis process attempts to infer headers for rows and columns of each detected block. If any row or column of a block is not related to a header, but headers for neighbouring columns or rows are available, the spreadsheet creator missed an opportunity to supply contextual information for the worksheet. This lowers comprehensibility of the related part of the worksheet.

Target. Missing Header detects quality issues based on blocks and block headers. Consequently, the detection process for this smell is applied to blocks and block headers of a worksheet.

Detection. Detection of the Missing Header smell requires analysis of the established header relations for each block of a worksheet. For each column and row of each block of the worksheet, we check whether any header relation is available for this column or row. If this is not the case, but a header relation for at least one other column or row of the same block is available, this instance is indicated as smelly. Missing headers are indicated at the lowest available header layer of the related block which features at least one other header for the block.

Example. Figure 6.12 provides an example for the Missing Header smell. The example is an adaptation of the car-sales worksheet. The header in cell E2 was removed. One calculation block encompassing area B3:F6 was discerned for the worksheet. The missing header cell E2 is part of the lowest layer of column-headers of the block. Other headers are available within the same layer. Hence, cell E2 is indicated as smelly.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014		Total	
3	Honda	30	27	28	32	117	
4	Mazda	10	12	9	7	38	
5	Fiat	9	12	13	15	49	
6	Total	49	51	50	54	204	
7							

Figure 6.12: Example for Missing Header smell. The discerned block of the worksheet is indicated by yellow background colour. Lowest level header layers of the block are indicated by dark blue, higher level headers by light blue. The header in cell E2 is missing. Other headers are present within the same header layer. Hence, the Missing Header smell is detected for cell E2.

Cause. Missing Header detects columns or rows of inferred calculation blocks which are not related to any header. We argue that missing headers may occur following one of three scenarios:

- *Incomplete initialization.* During creation of the worksheet, no header is provided by the spreadsheet creator.
- *Inconsistent edit (block).* A spreadsheet user expands the size of a block, but does not provide header information for the new introduced row or column.
- *Inconsistent delete (header).* A spreadsheet user deletes a header of a block, but does not delete the underlying column or row of the related block.
- *Unintentional related header.* No headers should be available for the columns or rows of a header. However, a potential header cell is unintentionally po-

sitioned in a way which can be interpreted as header. Consequently, Missing Header is detected for the remaining columns or rows of the block.

Consequences. Missing Header points out instances where either a column or row of an inferred block is missing a related header. This may indicate an error, whereby a user intended to delete the header as well as the underlying column or row of a block, but did not remove the related part of the block. Otherwise, such instances may be introduced by a spreadsheet creator or editor who did not provide contextual information for a part of the worksheet. Hence, future spreadsheet users may lack this information to correctly perform further updates of the spreadsheet.

Alleviation. Removal of the Missing Header smell may occur by one of three operations:

- *Supply missing header.* Insertion of the missing header information for the related part of the block removes the smell.
- *Remove underlying part of block.* This resolution is applicable in cases where a block was expanded by mistake, or missing header indicates an unsuccessful removal of a part of block.
- *Remove neighbouring headers.* This resolution is applicable in cases where a header relation was unintentionally established for columns or rows of a block.

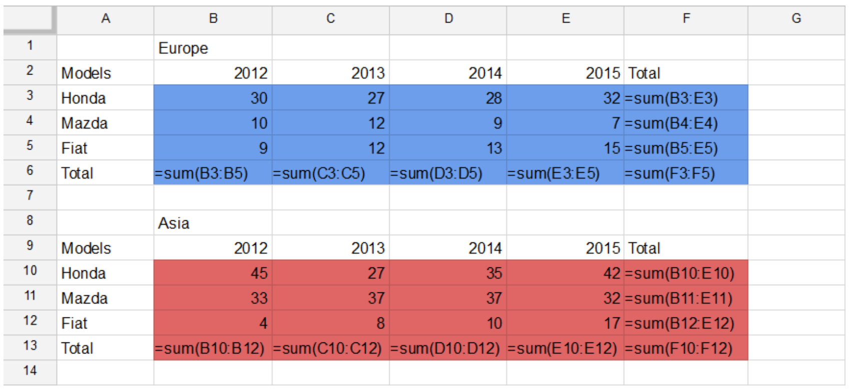
6.2.7 Overburdened Worksheet

Origin & Intent. The Overburdened Worksheet smell points out worksheets that are faced with too much responsibility. Calculation-based worksheets are comprised of two main types of components: components which apply calculations and components which provide data for those calculations. In the basic interpretation, components which apply calculations are formula cells, and components which provide data for those calculations are referred cells. The results of our structural analysis approach allow for an additional interpretation: components which apply calculations are formula groups, and components which provide data for those calculations are relation-based groups. Moreover, the Blocking process combines neighbouring formula groups and reference-based groups, providing a combination of both components. Independent of the interpretation, if any individual worksheet contains too many of either component, the worksheet becomes hard to comprehend and hard to maintain.

Target. Overburdened Worksheet detects quality issues based on the number of calculation components of a worksheet. Depending on the interpretation, the components may either be formula cells and referenced cells or formula groups and relation-based groups, or blocks of a worksheet.

Detection. Detection of the Overburdened Worksheet smell requires enumeration of the components which are applied for smell detection. In case of formula cells and referenced cells, the number of such cells contained in the worksheet is calculated. In case of formula groups and relation-based groups, the number of such groups contained in the worksheet is calculated. In case of blocks, the number of blocks in the worksheet is calculated. If any of these applied metrics exceeds a given threshold, the worksheet is indicated as smelly.

Example. Figure 6.13 provides an example for the Overburdened Worksheet smell. The example is an adaptation of the car-sales worksheet. The worksheet contains an additional calculation block, summarizing sales data for Asia. Depending on the employed detection strategy and threshold values, existence of a second calculation block of the same size could already indicate the worksheet as smelly.



	A	B	C	D	E	F	G
1		Europe					
2	Models	2012	2013	2014	2015	Total	
3	Honda	30	27	28	32	=sum(B3:E3)	
4	Mazda	10	12	9	7	=sum(B4:E4)	
5	Fiat	9	12	13	15	=sum(B5:E5)	
6	Total	=sum(B3:B5)	=sum(C3:C5)	=sum(D3:D5)	=sum(E3:E5)	=sum(F3:F5)	
7							
8		Asia					
9	Models	2012	2013	2014	2015	Total	
10	Honda	45	27	35	42	=sum(B10:E10)	
11	Mazda	33	37	37	32	=sum(B11:E11)	
12	Fiat	4	8	10	17	=sum(B12:E12)	
13	Total	=sum(B10:B12)	=sum(C10:C12)	=sum(D10:D12)	=sum(E10:E12)	=sum(F10:F12)	
14							

Figure 6.13: Example for Overburdened Worksheet smell. Cells of the two discerned calculation blocks of the worksheet are illustrated by blue and red background colour. The worksheet features multiple blocks. Based on the employed detection method, this may be indicated as smelly.

Cause. Overburdened Worksheet detects worksheets which contain too many calculation components. We argue that Overburdened Worksheet instances may occur following one of three scenarios:

- *Too much content.* The worksheet is simple in structure and does not feature many different groups. However, each group contains a vast number of individual items.
- *Too many calculation groups.* The worksheet features a high number of different groups, either in one concise or multiple blocks.
- *Too many calculation areas.* The worksheet features a high number of individual calculation areas (blocks).

Consequences. Overburdened Worksheet detects various circumstances which indicate worksheets that are faced with too much responsibility. Independent from the individual cause, worksheets which feature too many calculation components are hard to comprehend. Update operations applied to such worksheets take longer to implement and are more likely to introduce faults.

Alleviation. The Overburdened Worksheet smell may be removed by one of two operations:

- *Restructure worksheet.* In cases where the smell was detected as a result of a high number of individual groups or blocks, the worksheet may be restructured in a way which merges individual groups or blocks, thus lowering the related metric below its detection threshold.
- *Divide worksheet.* Individual parts of the worksheet are relocated to one or multiple other worksheets. Connectivity between divided worksheets is established via inter-worksheet-references. Blocking and Grouping results may be applied to infer parts to be relocated.

6.2.8 Overview

As we demonstrated for established smells, smells can be arranged into groups based on certain properties. In order to highlight fundamental similarities and differences, Table 6.1 provides an overview of the new spreadsheet smells proposed in this section. The table features the following properties:

Target	Spreadsheet parts that can contain the smell.
Cause	Stage of the spreadsheet life cycle at which the smell is usually introduced.
Consequences	Aspects of spreadsheet quality that are affected by the smell.
Alleviation	Options for smell removal.

Name	**Target**				**Cause**			**Cons.**		**Alleviation**		
	Formula Groups	Ref.-based Groups	Blocks	Headers	Creation	Data entry	Expansion	Erroneous result	Impeded quality	Manual	Assisted	Automated
Duplicated Formula Groups	●				●		●		●		●	
Formula Group Distance	●				●		●		●		●	
Unrelated Neighbours	●	●			●		●		●		●	
Incons. Ref. Dimensions		●			●		●		●		●	
Incons. Formula Group Ref.	●				●		●	●	●			●
Missing Header				●	●	●	●		●	●		
Overburdened Worksheet	●	●	●		●	●	●		●			●

Table 6.1: Comparison of novel Spreadsheet Smells

6.3 Discussion

In the previous sections, we proposed various approaches to apply the results of our structural analysis process in the form of spreadsheet smells. We showed that the application of structural information may provide a number of benefits. When applied to existing smells, structural information allows to limit the scope of the employed detection process. This results in lower processing requirements, and a lower number of false positives. When used to create new smells, structural information enables us to detect issues based on quality aspects which were inaccessible to established spreadsheet smells (e.g. inconsistent area dimensions).

However, smell detection based on structural information also suffers from a number of drawbacks. The initial analysis process requires a significant amount of processing. While the application of structural information benefits processing requirements of smell detection processes, these performance gains may not always outweigh the initial investment of processing time. Hence, structure-enhanced smell detection may take longer than the established detection methods. Another disadvantage is posed by the principle of our analysis process to infer groups based on formula relations. As a consequence of this principle, analysis is limited to the detection of groups of formula cells, groups of cells which are referenced by formulas, and headers for these detected groups. However, some smells also include other cell types in their detection criteria. For example, the QFD smell also includes columns and rows of non-referenced string cells in its detection process. To support such smells in their entirety, grouping of calculation-neutral cells is required. An approach to expand the current analysis process to infer groups for neighbouring cells is presented in Chapter 8. Lastly, applicability of the results of structural analysis to benefit smell detection requires that present issues, indicated by smells, do not prevent successful structural analysis to begin with. The structural analysis process proposed in this thesis is based on groups of neighbouring formula cells that share the same formula in R1C1-representation. Thus, smells which are not attributed to formula cells also do not affect our analysis process. However, issues regarding formula cells that are detected as smells might influence the results of our structural analysis process. Two types of such issues are prone to influence structural analysis, based on how the issue affects the formula group of the related formula cell:

- *The dimension of the group changes.* In this scenario, the cause of the smell either removes or adds a valid group cell at the edge of the group. This has a number of possible consequences. The dimensions of related reference-based groups may change. Formula Group Matching may establish new, faulty connections to other formula groups or might not be able to establish previously inferred connections. Reference Group Merging may condense previously unrelated reference-based groups, or result in a higher number of smaller, fragmented groups. This, in turn, may influence the result of the Blocking step. Lastly, diverging header-relations may be inferred as a result of different block dimensions.

- *A non-border cell is removed from the group.* This case occurs, if the cause of the smell either removes or edits a valid group cell within the group. As a consequence, the initial formula group is split into two separate groups. Related reference-based groups are split between the two resulting formula groups. Formula Group Matching remains mostly unaltered. Reference Group Merging may lead to a higher number of smaller, fragmented groups. Blocking may lead to the detection of smaller blocks, or even to the merging of previously separated blocks. Lastly, different header-relations may be inferred for new, incorrect blocks.

As we explained, issues which lead to smells might negatively affect the reliability of the analysis process. However, the process may still be applied and the resulting structural information utilized for smell detection. Based on these results, a number of different smells may identify the existing issues:

- *Duplicated Formula Groups* detects cases where a cell was removed or altered within a formula group. This leads to the separation of the formula group into two individual formula groups which share the same R1C1-formula.
- *Formula Group Distance* detects cases where a cell within or at the edge of a group was minimally edited. This introduces a separate formula group featuring a similar formula as neighbour of the remainder of the initial formula group.
- *Unrelated Neighbours* detects cases where two or more neighbouring cells were altered in the same way within a group. This introduces a new, unrelated formula group as neighbour of the remainder of the initial formula group. The smell may also detect cases of group expansion, if the resulting group is located next to another group.
- *Inconsistent Reference Dimensions* may detect any alteration which affects the areas inferred for the references of a formula group. The position of these areas depends on the related reference of the formula group. In case of a static reference, the dimensions of the area are constant. In case of a dynamic reference, the dimensions of the area are depending on the dimensions of the formula group. Hence, any alteration which affects a reference of a group affects the related reference-area. Any alteration which changes the size of the formula group affects the related reference-areas of dynamic references of the

group. If any of these affected areas is shared by another formula group, this is indicated as smelly.

- *Inconsistent Formula Group Reference* may detect any modification which affects the size of the group if the initial group was referenced by any other formula group, or if the initial group referred to any other formula group via a non-static reference.

- *Missing Header* detects an alteration that leads to a block expansion. In cases where no header is available for the new column or row of the block, this is indicated as smelly.

If one of these smells identifies the initial issue, then refactoring may restore the intended state of the worksheet. Consequently, the static analysis process can be applied to the corrected worksheet, resulting in structural information which is not impaired by any smell-causing issues. Hence, in most cases, structural analysis is either unconditionally applicable to smelly spreadsheets, or is able to indicate the issue which negatively affects the analysis results.

7. Related Work

In this chapter, we provide a summary of existing scientific approaches which are related to our work. In specific, we state approaches which contributed to the current status of the scientific spreadsheet smell catalogue, as well as major approaches to utilize spreadsheet smells in a way to benefit spreadsheet users. We also mention UCheck, which served as inspiration for our structural analysis process.

Cunha *et al.* [2012b] were the first to introduce the idea of a spreadsheet smell-catalogue. They also contributed a set of spreadsheet smells of their own. However, rather than base spreadsheet smells on existing code smells, they established a set of general guidelines to define smells for spreadsheets. Based on those guidelines, they defined a set of intuitively meaningful spreadsheet smells, generating an initial smell catalogue. Consequently, they evaluated this catalogue utilizing a large spreadsheet-repository. Based on this evaluation, they refined the initial catalogue, mainly improving its robustness. Cunha *et al.* also developed a tool to perform automated smell detection of spreadsheets: Smellsheet Detective. Cunha *et al.* [2012a] describe the Smellsheet Detective tool in greater detail and provide evaluation results of their implementation based on the EUSES corpus.

Hermans *et al.* [2012a] were among the main contributors to the current spreadsheet smell catalogue. Their first of two major contributions focussed on inter-worksheet smells for spreadsheets. In this work, Hermans *et al.* analysed the set of existing code smells presented by Fowler [1999] that are related to the interdependency of object-oriented classes. Based on this analysis, they adapted the working principles of these smells to the spreadsheet environment, substituting classes with worksheets. To facilitate automatic smell detection, Hermans *et al.* then defined metrics for each of the newly introduced spreadsheet-smells. For each of these metrics, they also provided a threshold, identifying a spreadsheet to be smelly if exceeded.

Hermans *et al.* proposed another contribution regarding smells based on spreadsheet formulas [2012b]. As with their previous work [2012a], they analysed existing code smells presented by Fowler [1999] in order to apply the basic idea of those smells to the spreadsheet paradigm. However, in this approach, Hermans *et al.* focussed on smells which affect spreadsheet formulas in particular. For each of the resulting smells, they, again, defined metrics and threshold values for each metric in order to facilitate automatic smell detection. Hermans *et al.* [2014] expanded on this approach. Besides a more detailed analysis and discussion, the main contribution of this extension is a catalogue of refactorings, which aim to resolve each of the presented formula smells. Hermans *et al.* also provide information regarding the implementation of their refactoring system as well as an exploratory study regarding the impact of spreadsheet refactorings.

Hermans *et al.* presented another approach towards automatic analysis of quality issues within spreadsheets [2013]. In particular, they proposed a method to automatically detect data clones within spreadsheets. Following their definition, a data clone occurs when the result of a formula-calculation is copied and pasted into another cell. Those copied formula-results may then be used as basis for further calculations. However, update of the initial calculation is not automatically applied to the copied value. Thus, data clones represent a risk in regard to spreadsheet maintainability and correctness. Data clones cannot strictly be categorized as static spreadsheet analysis method, as the approach requires evaluation of formulas within the analysed spreadsheet.

Badame and Dig recognized the trend to quality awareness regarding spreadsheets. In specific, they recognized that classic programming environments offer automatic refactoring options to improve code quality. However, despite being faced with similar quality issues, users of spreadsheet environments are not provided with similar options. Consequently, Badame and Dig asked different spreadsheet-user-groups about which automated spreadsheet refactorings would provide the greatest benefit. Based on this feedback, they created the REFBOOK tool [2012] as extension for Microsoft Excel, which allows users to automatically perform a specific set of refactorings.

Abreu *et al.* [2014a] attempted to introduce a new debugging system for spreadsheets by merging the spreadsheet smell metaphor with common fault localization techniques. They developed a system which automatically identifies cells that are likely to contribute to faults, based on detected spreadsheet smells. To that end, they first analysed the set of existing spreadsheet smells and categorized these smells into either of two categories: *fault-inducing* and non fault-inducing, or *perceived* smells. Each spreadsheet in question is then analysed in order to detect occurrences of fault-inducing smells. The cells which were inferred to contain one of the fault-inducing smells consequently act as input for a fault localization algorithm. The result of this algorithm is a set of cells which are likely to contribute to existing faults within the spreadsheet. Abreu *et al.* also developed an implementation of their debugging approach, resulting in the FaultySheet Detective tool [2014b].

Abraham and Erwig [2004] proposed an approach for automated header inference in spreadsheets. Throughout their work, they present a collection of algorithms to infer header information from a spreadsheet. Each of these algorithms analyses a different structural aspect, commonly found in spreadsheets. In addition, Abraham and Erwig provide a header inference framework which allows to employ a combination of the presented algorithms. By means of this framework, they evaluate and optimize different weighted combinations of header detection algorithms. In later work [2007], Abraham and Erwig applied their optimized header inference process by combining it with previous work into unit systems for spreadsheets [Erwig and Burnett, 2002]. The resulting system was implemented into a tool dubbed UCheck. The tool is provided as add-in for Excel which communicates with a header/unit inference engine implemented in Haskell. Once the system is put into action, it automatically infers headers and assigns and infers units. The tool indicates cells which contain detected unit errors via a red background color. Evaluation by Abraham and Erwig indicates that the header inference process rarely produces incorrect header assignments. Moreover, even in cases where incorrect header inference occurs, the follow-up unit inference does not report any illegal errors.

In this summary, we highlighted approaches that are closely related to our own work. However, the field of spreadsheet QA encompasses a wide variety of different techniques. For the interested reader, Jannach *et al.* [2014] provide a comprehensive overview of noteworthy approaches to improve spreadsheet quality.

8. Conclusions and Future Work

The main contribution of this thesis was to establish a novel structural analysis process for spreadsheets. Our analysis approach identifies groups of cells based on their role in calculations of a spreadsheet. We also detect areas that are encompassed by multiple neighbouring cell groups, indicating distinct calculation areas within a spreadsheet, dubbed blocks. Moreover, our analysis approach identifies header cells and relates them to groups and calculation areas of a worksheet. Due to the formula-based detection approach in combination with the generality of the detected structures, to the best of our knowledge, this process is the first of its kind.

We evaluated our analysis process based on spreadsheets of the EUSES and ENRON corpora. To that end, we applied our analysis approach on previously inspected spreadsheets and compared the results. The key finding of our evaluation is that an average of 99 % of expected blocks were found by our approach. Moreover, more than 80 % of detected blocks could be inferred in their entirety. Headers could be discerned in 76 % to 96 % of cases, based on the header type and the applied evaluation data. Most of detected headers could be inferred in their entirety. A drawback of our analysis approach could be identified regarding the dimensions of the detected structures. The correct size of overall detected structures could be inferred in 55 % to 77 % of cases. We identified a number of reoccurring circumstances which primarily cause these inference issues. Nevertheless, the evaluation results demonstrated the feasibility of using inferred spreadsheet structures to enhance spreadsheet QA techniques.

Structural information inferred by our analysis process may benefit various spreadsheet QA techniques. We demonstrated one such application by enhancing spreadsheet smells by means of structural information. We identified five issues in entries of the established smell catalogue, and demonstrated how these issues can be alleviated by applying structural information in the detection routines of the related smells. We also introduced seven new spreadsheet smells based on different aspects of structural information. The proposed enhancements reduce the runtime of smell detection processes, lower the number of false positive smell detections, and allow for the detection of further issues in form of new smells.

Application of structural information is not limited to spreadsheet smells alone. Our analysis approach provides a framework for future studies to incorporate structural information in all categories of spreadsheet QA techniques. As was demonstrated for spreadsheet smells, application of structural information likely provides similar benefits to other techniques as well. We would like to point out some interesting opportunities:

- *Unit Inference.* Our structural analysis approach was inspired by the analysis process employed by UCheck. However, the main focus of UCheck, along with a number of other approaches, is unit inference. Unit inference describes the process to infer unit information for cells within a worksheet. Unit information is based on the headers which affect each cell. Future work could apply such unit inference processes based on the header-relations detected by our analysis process. Moreover, existing unit-inference processes could be adapted to utilize additional structural information like blocks and formula-group relations.

- *Group-Editing Operations.* Future work could establish editing operations based on groups and blocks identified for a worksheet. Two types of operations would be required: First, operations affecting entire groups, supporting creation, deletion, and relocation of blocks and groups. Second, operations affecting individual elements of groups, supporting expansion of a group by an element, and deletion of an element from a group. Such operations could assist refactoring tools by allowing for structure-sensitive modification of worksheets. Moreover, those operations could be offered to spreadsheet-users in a context-sensitive way. This would establish an additional way to interact with spreadsheets.

- *Conceptual Knowledge.* Knowledge-databases are established platforms which provide knowledge-based analysis of terms. Provided with a term in string-representation, such systems generate a list of concepts to which this term is related to. Moreover, concepts are connected with other concepts, forming a concept network. By utilizing a knowledge-database, future work could introduce the notion of concepts for our analysis approach. This allows to assign concepts to string cells within a worksheet. This could provide a number of benefits:

  – Concepts could enhance header-assignation performance. Concepts at both ends of a header-relation could be checked for conformity, rejecting or flagging suspicious relations.
  – Concepts of header cells can be propagated to blocks, groups, and individual cells. This allows for context-based unit-checking, but also for sanity-checking of formula and formula group relations based on context information. For example, if a formula cell attempts to add two cells which do not share a unifiable context (e.g. fruit and hours) this could be detected as suspicious.
  – Concepts could be used as metrics for smell detection. For example, a *Too many Concepts* smell could indicate blocks or worksheets as smelly if they concern themselves with too many different contexts.
  – Contexts could aid in the determination of automatic refactoring operations. In many cases, a specific refactoring operations can be achieved in multiple different restructuring approaches within the worksheet. Context information could be applied to rank the impact of eligible restructuring operations, offering the least-impact operation to the user.

Our study can also serve as starting point for further approaches towards structural analysis of spreadsheets. In particular, future work could focus on improving the overall detection performance of structural analysis. Moreover, the detection process could be expanded to include further structures (e.g. structures for calculation-neutral value cells). We suggest the following approaches to adapt and expand the current analysis process:

- *Adapt to the circumstances.* During evaluation of our analysis process, we identified a number of reoccurring issues that caused failing or incomplete inference of structures. While some of these issues are based on questionable practices (e.g. the use of formula cells as headers), we need to acknowledge that those practices are applied in real-world spreadsheets. Consequently, the detection rule-set of the current analysis process should be expanded to account for these questionable but occurring circumstances.

- *Group Variety.* Our analysis approach employed the *Follow the Formula* principle. Consequently, wherever possible, grouping and blocking-operations relied on information provided by formula relations, to infer group areas. However, other criteria could also be applied to discern meaningful areas within a worksheet. Future work could employ such alternative focal points for group inference. For example, groups could be established for numerical values which establish and abide by a common statistical model. Likewise, groups could be established for string cells, featuring common occurrence of individual strings or minimal string-distance metrics. Position-based metrics like neighbourhood to other cells and the location of cells within the worksheet could also be reassessed.

- *Agree to 2D.* Our analysis approach employs the *1D is Key* principle. Following this principle, 0D-areas are collated into 1D-areas and 2D-areas are partitioned into sets of aligning 1D-areas. Further analysis processes are applied to these 1D-areas. Future work could assess the viability to utilize 2D-areas instead. In specific, blocking requires re-collation of previously partitioned sets of 1D-areas. The initial existing 2D-areas could be employed instead, potentially enhancing the results of blocking.

- *Diverse Group-Dependencies.* The current analysis approach constructs a hierarchy of formula groups based on group relations. However, relations within this hierarchy do not yet take types and multiplicity of source- and target groups of each reference into consideration. Future work could categorize formula group references based on these metrics. Consequently, follow-up analysis could employ these categories as precursors for relation-strength. For example, a relation between a 10-cell formula group and another 10-cell formula group has more impact on the overall structure of the worksheet than a relation between a 1-cell formula and a single value cell.

- *Block Variety.* Our analysis approach identifies blocks within a worksheet based on expansion of existing groups. Expansion takes place as long as any eligible group is in immediate range of the block, and inclusion of the new group does not lead to any invalid cells being part of the group. Future work could assess the viability of more restricted blocking processes. For example, block-building could introduce some form of sub-blocks, where sub-blocks within a block contain groups which share a common characteristic. Sub-blocks could be introduced to collate groups of the same orientation. Further, sub-blocks could be calculation-sensitive, only collating neighbouring blocks which are connected by a formula-relation.

- *Structure Propagation.* Our analysis process focusses on areas within spreadsheets which take part in any form of calculation. However, spreadsheets do not exclusively contain formula-related data. Nevertheless, in cases where unrelated data is located next to formula-related data, our analysis approach may still be applied. The resulting structural information may then be propagated to neighbouring areas of non-formula-related values. This propagation process may detect groups of independent value cells matching the predetermined structure. For example, a worksheet may contain a catalogue of items, organized in rows. Each column is dedicated to a specific item-related value. Most of these values are not part of any formula-relation. However, the catalogue also features the stock and price of each item, as well as formulas for value and price calculation. These formula-relations could provide structural shapes which can be applied to the neighbouring data. Thus, identified blocks and header-relations may be extended to non-calculation-relevant value cells.

In summary, while structural analysis poses a difficult problem and our proposed approach is not without its limitations, the benefits anticipated for our exemplary spreadsheet smell enhancements and our outlined perspectives for further areas of application demonstrate that examination of this problem proved to be a worthwhile effort in itself and also laid a possible foundation for future studies into this topic.

List of Figures

List of Tables

References

[Abraham and Erwig, 2004] Robin Abraham and Martin Erwig. Header and Unit Inference for Spreadsheets Through Spatial Analyses. In *2004 IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04)*, pages 165–172, 2004.

[Abraham and Erwig, 2006] Robin Abraham and Martin Erwig. Inferring Templates from Spreadsheets. In *Proceedings of the 28th International Conference on Software Engineering (ICSE'06)*, pages 182–191, 2006.

[Abraham and Erwig, 2007] Robin Abraham and Martin Erwig. UCheck: A spreadsheet type checker for end users. *Journal of Visual Languages & Computing*, 18(1):71–95, 2007.

[Abreu *et al.*, 2014a] Rui Abreu, Jácome Cunha, Joao Paulo Fernandes, Pedro Martins, Alexandre Perez, and Joao Saraiva. Smelling faults in spreadsheets. In *Proceedings of the 30th IEEE International Conference on Software Maintenance and Evolution (ICSME'14)*, pages 111–120, 2014.

[Abreu *et al.*, 2014b] Rui Abreu, Jácome Cunha, Joao Paulo Fernandes, Pedro Martins, Alexandre Perez, and Joao Saraiva. FaultySheet Detective: When Smells Meet Fault Localization. In *2014 IEEE International Conference on Software Maintenance and Evolution (ICSME'14)*, pages 625–628, 2014.

[Apache Software Foundation, 2015] Apache Software Foundation. Apache POI. `http://poi.apache.org/`, 2015. [Online; accessed 2015-08-31].

[Badame and Dig, 2012] Sandro Badame and Danny Dig. Refactoring meets spreadsheet formulas. In *28th IEEE International Conference on Software Maintenance (ICSM'12)*, pages 399–409, 2012.

[Chiang and Miller, 2008] Fei Chiang and Renée J. Miller. Discovering Data Quality Rules. *Proceedings of the VLDB Endowment*, 1(1):1166–1177, 2008.

[Cunha *et al.*, 2012a] Jácome Cunha, Joao Paulo Fernandes, Pedro Martins, Jorge Mendes, and Joao Saraiva. SmellSheet detective: A tool for detecting bad smells in spreadsheets. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'12)*, pages 243–244, 2012.

[Cunha *et al.*, 2012b] Jácome Cunha, João P. Fernandes, Hugo Ribeiro, and João Saraiva. Towards a Catalog of Spreadsheet Smells. In *Computational Science and Its Applications (ICCSA'12)*, number 7336 in Lecture Notes in Computer Science, pages 202–216. Springer Berlin Heidelberg, 2012.

[Dou *et al.*, 2016] Wensheng Dou, Liang Xu, Shing-Chi Cheung, Chushu Gao, Jun Wei, and Tao Huang. VEnron: A Versioned Spreadsheet Corpus and Related Evolution Analysis. In *Proceedings of the 38th International Conference on Software Engineering (ICSE SEIP'16)*, 2016.

[Erwig and Burnett, 2002] Martin Erwig and Margaret Burnett. Adding Apples and Oranges. In Shriram Krishnamurthi and C. R. Ramakrishnan, editors, *Practical Aspects of Declarative Languages*, number 2257 in Lecture Notes in Computer Science, pages 173–191. Springer Berlin Heidelberg, 2002.

[EuSpRIG, 2013] The European Spreadsheet Risks Interest Group EuSpRIG. EuSpRIG horror stories. `http://www.eusprig.org/horror-stories.htm`, 2013. [Online; accessed 2016-03-21].

[Fisher and Rothermel, 2005] Marc Fisher and Gregg Rothermel. The EUSES Spreadsheet Corpus: A Shared Resource for Supporting Experimentation with Spreadsheet Dependability Mechanisms. In *Proceedings of the First Workshop on End-user Software Engineering (WEUS'05)*, pages 1–5, 2005.

[Fowler, 1999] Martin Fowler. *Refactoring: improving the design of existing code.* Pearson Education India, 1999.

[Hermans and Murphy-Hill, 2015] Felienne Hermans and Emerson Murphy-Hill. Enron's spreadsheets and related emails: A dataset and analysis. In *Proceed-*

ings of the 37th International Conference on Software Engineering (ICSE'15), pages 7–16, 2015.

[Hermans *et al.*, 2011] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Supporting Professional Spreadsheet Users by Generating Leveled Dataflow Diagrams. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE'11)*, pages 451–460, 2011.

[Hermans *et al.*, 2012a] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting and Visualizing Inter-worksheet Smells in Spreadsheets. In *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*, pages 441–451, 2012.

[Hermans *et al.*, 2012b] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting code smells in spreadsheet formulas. In *28th IEEE International Conference on Software Maintenance (ICSM'12)*, pages 409–418, 2012.

[Hermans *et al.*, 2013] Felienne Hermans, Ben Sedee, Martin Pinzger, and Arie van Deursen. Data Clone Detection and Visualization in Spreadsheets. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*, pages 292–301, 2013.

[Hermans *et al.*, 2014] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineering (ESE'14*, pages 1–27, 2014.

[Jannach *et al.*, 2014] Dietmar Jannach, Thomas Schmitz, Birgit Hofer, and Franz Wotawa. Avoiding, finding and fixing spreadsheet errors – A survey of automated approaches for spreadsheet QA. *Journal of Systems and Software*, 94:129–150, 2014.

[Jansen, 2015] Bas Jansen. Enron versus euses: a comparison of two spreadsheet corpora. *arXiv preprint arXiv:1503.04055*, 2015.

[Koch, 2015] Patrick Koch. Smells and Units: An Overview of Selected Static Analysis Methods for Spreadsheets. `http://spreadsheets.ist.tugraz.at/index.php/papers/studentswork/`, 2015. [Online; accessed 2016-02-16].

[Levenshtein, 1966] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics Doklady*, volume 10, page 707, 1966.

[Panko and Port, 2012] Raymond R Panko and Daniel N Port. End user computing: the dark matter (and dark energy) of corporate IT. In *45th Hawaii International Conference on System Science (HICSS'12)*, pages 4603–4612, 2012.

[Taylor, 2007] Kevin Taylor. An Alaysis of Computer Use across 95 Organisations in Europe, North America and Australasia. 2007.