The Journal of Systems & Software 175 (2021) 110910

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Product metrics for spreadsheets—A systematic review☆,☆☆

Birgit Hofer [a], Dietmar Jannach [b], Patrick Koch [b], Konstantin Schekotihin [b], Franz Wotawa [a,*]

[a] TU Graz, Austria
[b] University of Klagenfurt, Austria

ARTICLE INFO

Article history:
Received 27 April 2020
Received in revised form 30 September 2020
Accepted 18 January 2021
Available online 1 February 2021

Keywords:
Spreadsheet quality assurance
Spreadsheet metrics
Metrics catalog
Spreadsheet product metrics
Spreadsheet metrics survey

ABSTRACT

Software product metrics allow practitioners to improve their products and to optimize development processes based on quantifiable characteristics of source code. To facilitate similar benefits for spreadsheet programs, researchers proposed various product metrics for spreadsheets over the last decades. However, to our knowledge, no comprehensive overview of those efforts is currently available. In this paper, we close this gap by conducting a literature review of research works that either inherently or explicitly define product metrics for spreadsheets. We scanned five major digital libraries for scientific papers that define or use spreadsheet product metrics. Based on the identified 37 papers, we created a novel catalog of product metrics for spreadsheets. The catalog can be used by practitioners and researchers as a central reference for spreadsheet product metrics. In the paper, we *(i)* describe the proposed metrics in detail, *(ii)* report how often and for what purposes the metrics are used, *(iii)* identify significant discrepancies in the naming and definition of the metrics, and *(iv)* investigate how the appropriateness of the metrics was evaluated.

1. Introduction

Since their first appearance on the market in the late 1960s, spreadsheets have become a great success story. This may be due to the fact that spreadsheets allow their users to do calculations and numerical analyses in a very intuitive way similar to paper worksheets. Today, spreadsheets are used almost everywhere for storing data in a tabular form and for performing analysis like budget forecasts.

The outcomes of spreadsheet calculations often serve as a basis for decision-making in organizations. As a result, however, errors in these calculations can have tremendous effects, and there is a growing list of examples where the use of spreadsheets has led to substantial monetary losses, e.g., see Finextra (2020), Lee (2020). The occurrence of errors in spreadsheets is not a surprise. Panko (2008) estimated the probability of introducing a bug in a spreadsheet formula between 3% and 5%. When assuming stochastic independence of bugs a spreadsheet comprising 1000 formulas has a bug with a probability of more than 99.9%. As a consequence, there is, without any doubt, a strong need for applying quality assurance methods, techniques, and tools during the development of spreadsheets. Note that for ordinary programs from industry the average defect rate is expected to be between 1 and 25 bugs for every 1000 lines of code (see McConnell (2004), page 521), which is slightly better compared to the fault rate of spreadsheets. However, considering that commercial programs are huge in size comprising up to million lines of code, it is likely that there are no ordinary programs without bugs.

Researchers have already taken up the challenge of providing means for improving the overall quality of spreadsheets using a range of techniques for improved program comprehension, fault localization, and debugging (Jannach et al., 2014).

At their core, these techniques are using ideas and methods that originate in general software engineering, where the use of various *metrics* is a standard approach to improve both the development process and the final products. Such metrics are used to identify issues where improvements or corrections may be needed. Traditional and well-established metrics are, for example the following two *product metrics*, i.e., those that are related to characteristics of the software artifact itself, *lines of code* or *number of bugs per line of code*, which help to judge the complexity or the quality of a piece of software. These metrics-based assessments can then, for example, serve as a basis for automated approaches to fault prediction (Radjenovic et al., 2013).

The use of product metrics has been advocated for the spreadsheet domain for a number of years, e.g., in the context of fault

☆ Authors are listed in alphabetical order.
☆☆ Editor: [Andy Zaidman].
* Corresponding author.
E-mail addresses: bhofer@ist.tugraz.at (B. Hofer), dietmar.jannach@aau.at (D. Jannach), patrick.koch@aau.at (P. Koch), konstantin.schekotihin@aau.at (K. Schekotihin), fwotawa@ist.tugraz.at (F. Wotawa).

https://doi.org/10.1016/j.jss.2021.110910

prediction, refactoring, and risk assessment for audits. However, the spreadsheet metrics proposed in the literature, e.g., *number of cells* and *number of worksheets*, sometimes suffer from a number of shortcomings. First, the metrics were often defined without any formal specification of measurement instructions. This opens up the possibility of their misinterpretation when considered by others. In our previous work (Koch et al., 2019), we have for example noticed that the informal descriptions of the metrics are often not sufficient to reproduce measurements from the literature for a given spreadsheet or corpus. Second, there exists a considerable overlap between the metrics that were conceptualized by different researchers. These overlaps are however sometimes hard to discern, because many metrics are presented without formal definitions. Moreover, in cases where specific measurement procedures for the same metric concept were proposed by different researchers, those measurement procedures are sometimes in conflict with each other.

Besides these issues, the literature on product metrics for spreadsheets is generally scattered. With this paper, our goal is to aggregate and synthesize the existing knowledge about such product metrics through a systematic literature review. Based on these results of the survey we develop a comprehensive catalog of spreadsheet metrics, which we categorize along different dimensions, e.g., the intended purpose for which a metric was suggested. Furthermore, we discuss inconsistencies that we found regarding the definition of similar metrics in different papers. Finally, we analyze which of the suggested metrics were actually evaluated, i.e., if there is evidence that the metrics are suited for their intended purpose.

We organize this paper as follows. In Section 2, we outline the basic terminology behind spreadsheets, define our research questions, and provide details about the systematic literature review process. Afterwards, in Section 3, we introduce our catalog of spreadsheet product metrics as extracted from the papers obtained from the survey. We organize the metrics along five categories including size, coupling, complexity, structural properties (e.g., patterns) and other aspects. In Section 4, finally, we answer the research questions in detail.

2. Terminology, survey design and retrieval results

In this section, we first introduce the used terminology. Then, we provide details of the systematic literature search and report the statistics regarding the identified papers.

2.1. Terminology

We use the following terminology.

Worksheet and spreadsheet. A worksheet is a table consisting of $m \times n$ cells where m is the number of rows and n is the number of columns. Each cell has a unique identifier, which can be referenced in formulas.[1]

A spreadsheet (also known as workbook) is a list of worksheets. Each worksheet in a spreadsheet can be accessed by its index or name.

Corpus. A corpus is a collection of spreadsheets. Such corpora are used by researchers for analyzing, e.g., error patterns, and to evaluate methods and tools for spreadsheet quality assurance, e.g., fault localization.

[1] Spreadsheet tools usually support both relative and absolute references. For instance, Microsoft Excel supports R1C1 notation for the relative and A1 notation for the absolute references.

Metric. A metric is a function that maps software (or spreadsheet) data to a numerical value (IEEE, 1993). We can distinguish product and process metrics. Product metrics are directly derived from the software (or spreadsheet) and documentation. The number of lines of code of a certain program is one example of a product metrics. Process metrics are related to the development process. Examples of process metrics in software engineering are the person months required for delivering a certain amount of source code, developer attributes (Janvrin and Morrison, 2000) or error rates (Irons, 2008). In our review, we exclusively focus on product metrics. Process-related information about spreadsheets is usually not available, as spreadsheets are mostly not managed with version control systems.

Metric value. A metric value is the numerical value that is obtained when applying a metric (IEEE, 1993). The value can lie within a predefined numerical range or be a binary one (true, false).

Measure and measurement. In the context of this paper, to measure means to apply a metric. We use the term measurement both *(i)* to describe the process of measuring and *(ii)* to refer to the outcomes, i.e., the value, when applying a metric (IEEE, 1993).

Code smell. In software engineering, a code smell is any characteristic of the source code that indicates a potential problem or a risk, e.g., a fault or a code fragment that may be hard to maintain. To operationalize this concept, we assume that smells use metrics as their basis for deciding which parts of a spreadsheet are potentially risky. Smells use metrics with either a Boolean or a numeric output. Boolean metrics return *true* if the input set of cells matches some predefined pattern, and *false* otherwise. Smells using numeric metrics use predefined thresholds to determine whether an input set of cells is considered smelly given its metric value.

2.2. Survey design and retrieval results

For the literature survey we follow the guidelines introduced by Wohlin et al. (2012). In particular, we explicitly discuss the background and rationale behind the survey, its scope, the research questions, and the underlying methodology.

2.2.1. Literature survey design

Background and rationale. This literature survey aims to provide an overview of product metrics for spreadsheets defined in the literature, resulting in a categorized catalog of the identified metrics.

Scope. The survey, as mentioned above, focuses only on *product metrics*, i.e., those metrics that can be measured directly from a given spreadsheet file. Besides process metrics, we also exclude metrics that are *(i)* only applicable to specific tools or extensions to the basic spreadsheet paradigm, and *(ii)* metrics that were designed for specific fault localization approaches.

The reasons for excluding certain types of metrics are as follows. First, in contrast to other software, process metrics are usually not available for spreadsheets, as mentioned above. In principle, we could also apply certain existing metrics to spreadsheet extensions written, e.g., in Visual Basic. This would however tailor our catalog to certain tools like Microsoft Excel. Furthermore, traditional metrics designed for such imperative programming languages are not in the scope of our survey on spreadsheet-specific metrics. Other extensions to the basic spreadsheet paradigm provide additional high-level formalisms, e.g., ClassSheets (Cunha et al., 2013). Metrics for these extensions do however not apply for standard spreadsheets. Finally, a number of metrics in the literature were designed for specific fault detection approaches

such as CheckCell (Barowy et al., 2014), UCheck (Abraham and Erwig, 2007b), or dimension inference (Chambers and Erwig, 2009). These metrics often implement decision-making strategies that are closely tied to some internal heuristics designed to identify potentially problematic cells. Such specific solutions are however, not in the focus of our survey on general product metrics for spreadsheets.

Research questions. Our research questions are as follows:

- RQ1: Which spreadsheet product metrics are defined and used in the literature and for which purposes?
- RQ2: Are the definitions of the metrics used in the literature consistent?
- RQ3: How do researchers evaluate their metrics?

Research method. We searched for spreadsheet metric related publications by scanning the digital libraries of ACM, IEEE, Springer, Elsevier, and ArXiv for papers with the term 'spreadsheet' and one of the terms 'metric', 'measure', or 'smell'. We provide the exact search strings for the individual digital libraries in the Appendix. The returned papers were manually analyzed and filtered in a two-step process. In the first filtering step, one author of this paper excluded entirely irrelevant papers based on the title and the abstract, e.g., papers that were using spreadsheets to compute metrics for data of a different domain. In the second filtering step, two authors decided on the inclusion of the remaining papers based on the following inclusion criteria:

- Does a paper describe an approach based on some measurable characteristic of spreadsheets? All papers that assess quality of spreadsheets using only non-measurable characteristics were discarded.
- Can the measurements be done using spreadsheet file only? Since in this paper we focus on product metrics, we rejected all papers which were only considering measurements of any other aspect of the spreadsheet development process.
- Is the metric applicable beyond the approach presented in the paper? As discussed above, some metrics presented in the literature are specifically designed for certain decision-making procedures in which they were used. Since the applicability of these metrics in other scenarios is questionable, they were omitted.

Since the necessary information was often not provided in the abstract, the authors read the full paper to make a decision about the inclusion. Afterwards, we performed snowballing by recursively checking the references of the obtained papers to identify additional relevant publications. In a final step, we read all obtained papers to double-check the relevance of the selected research works. For each publication remaining in the collection, we documented all identified metrics and their definition in case one was provided. During this process, we co-evolved a classification schema and categorized the metrics according to this schema.

2.2.2. Retrieval results

Basic statistics. Table 1 gives an overview of the papers that were found through the advanced search functions of the considered digital libraries. The primary search resulted in 565 papers, including duplicates that were found in several libraries. From these 565 papers, we determined 30 as relevant for this survey. In the follow-up snowballing process, we found 7 additional papers. In these papers, we pinpointed 57 spreadsheet metrics. For many of these metrics various interpretations and implementations were found.

Table 1
Paper retrieval results.

Digital library	Number of papers:	
	Found	Relevant
ACM	60	8
ArXiv	33	10
Elsevier	240	1
IEEE	118	15
Springer	114	3
Total original papers	**565**[a]	**37**[a]
Unique original papers		30
Snowballing		7
Papers included in study		**37**

[a]Some papers where found in several digital libraries.

Historical development. Fig. 1 shows the numbers of relevant papers published per year. The first paper related to spreadsheet product metrics was published in 1988 and compared the complexity of spreadsheets to that of traditional programming languages such as Pascal, Fortran and Assembler using Halstead's complexity metrics (Kokol, 1988). After a long break, the next paper was published in the year 2000. This paper explored how auditors estimate the size and complexity of tax payer spreadsheets and lists five metrics that are assumed helpful for auditors (Butler, 2000). In 2012, the largest number of papers related to spreadsheet product metrics was published. Four of these papers dealt with spreadsheet smells (Hermans et al., 2012a,b; Cunha et al., 2012,b). In 2015, the second largest number of papers was published. Two of these papers were about spreadsheet smells (Hermans et al., 2015; Jansen and Hermans, 2015) and two of them described the newly created ENRON corpus (Hermans and Murphy-Hill, 2015; Jansen, 2015). More recent publications focused, for example, on the use of metrics as features in supervised machine learning approaches for fault prediction (Koch et al., 2018; Koch et al., 2019).

3. A catalog of spreadsheet metrics

We defined the following five product metric categories to structure our catalog. Our categorization is based on organizing the metrics in terms of what they measure.

- **Size Metrics** measure the size of a spreadsheet or corpus, e.g., the number of cells with formulas.
- **Coupling Metrics** are based on formula references, e.g., the number of cell references in a formula or the depth of computation chains.
- **Formula Complexity Metrics** are based on formula attributes other than references, e.g., the depth of a formula's parse tree.
- **Structure and Pattern-based Metrics** are based on inferred patterns, e.g., a structure in a worksheet that violates an established pattern and should therefore be refactored.
- **Environment Functionality-based Metrics** measure the level of use of particular features of individual spreadsheet tools, such as the incorporation of procedural code and specific libraries, or the use of cell formatting or protection features.

Using our five-item categorization scheme described above, the 57 metrics presented in our survey can be categorized as

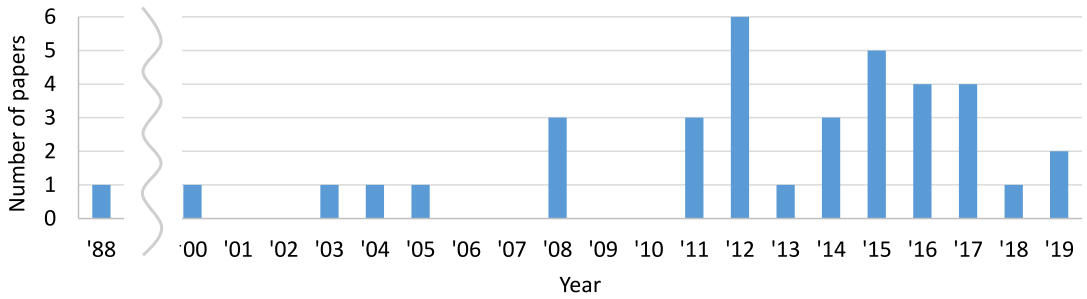

Fig. 1. Papers published per year.

shown in Fig. 2. The 14 Coupling Metrics form the largest category, followed by the category with 11 Structure and Pattern-based Metrics. The next two categories comprise 10 Size Metrics and Formula Complexity Metrics each. Finally, the smallest category is formed by 6 Environment Functionality-based Metrics. We discuss each of the 57 metrics in the following sections.

Alternative categorization schemes are possible as well. In previous works on spreadsheet smells, Hermans et al. for example distinguished between formula smells (Hermans et al., 2015) and inter-worksheet smells (Hermans et al., 2012a). In another work on smells, Cunha et al. listed *statistical smells*, *type smells* and *content smells* and *Functional Dependencies Based Smells* as their main categories (Cunha et al., 2012b). Differently from these previous approaches, our survey is not limited to smells. Furthermore, in our categorization we introduce various subcategories to formula-related metrics to distinguish between different types of metrics on a finer-grained level.

A different categorization method could be based on the application scope of the metrics (corpus, spreadsheet, worksheet, cell). Since some metrics can be applied on several levels (e.g., number of cells per worksheet or per spreadsheet), we refrained from using this aspect as a main criterion for the categorization. Yet another categorization could be based on the computation types of the metrics. There are *basic* metrics, which count certain spreadsheet features, *derived* metrics that combine two or more basic metrics, and *aggregated* metrics that, e.g., compute the sum or the average of other metrics for a spreadsheet. Such a categorization can, however, separate conceptually related metrics, which is why we did not choose the computation type as a main criterion.

3.1. Size Metrics

Size metrics are among the most frequently used metrics in the literature. They can be computed per corpus, spreadsheet, or worksheet.

3.1.1. Number of spreadsheets (NB-SPREADSHEETS)

The number of spreadsheets is used to describe the size of a corpora in empirical evaluations and to assess the size of spreadsheet applications in audits. There are four different types of this metric: *(i)* the *total number of spreadsheets* (Zhang et al., 2017; Dou et al., 2014, 2017), *(ii)* the *number of spreadsheets analyzed* (Hermans and Murphy-Hill, 2015; Dou et al., 2014), which indicates that not all spreadsheets of a corpus could be analyzed correctly, *(iii)* the *number of workbooks in an application* (Butler, 2000, 2008), which investigates connected spreadsheets, and *(iv)* the *number of spreadsheets with formulas* (Zhang et al., 2017; Dou et al., 2014; Hermans and Murphy-Hill, 2015). Note that with the *total number of spreadsheets* researchers sometimes actually refer to the *number of spreadsheets with formulas*, which would be more accurate, e.g., (Hermans et al., 2013). Moreover, a refined version of the last mentioned metric, namely the *number of spreadsheets with formulas that contain functions* (e.g., SUM, AVG), was proposed in (Hermans and Murphy-Hill, 2015).

3.1.2. Number of errors (NB-ERRORS)

The number of spreadsheets and cells with errors provides insight into the quality and maturity of corpora and spreadsheets. Cells with runtime errors such as `#DIV/0!`, `#N/A`, `#NAME?`, `#NUM!`, `#REF!`, and `#VALUE!` indicate that a spreadsheet is either still work in progress or contains faults. There are five different types of this metric: *(i)* the *number of spreadsheets with cells with error values* (Hermans and Murphy-Hill, 2015), *(ii)* the *number of formula cells that evaluate to an error value* (Hermans and Murphy-Hill, 2015; Koch et al., 2016; Fisher and Rothermel, 2005), *(iii)* the *number of input cells with errors* (Fisher and Rothermel, 2005), *(iv)* the *number of input cells that are referenced by other cells and contain an error value* (Fisher and Rothermel, 2005), and *(v)* the *number of output cells with errors* (Cunha et al., 2012a). Correia and Ferreira (Correia and Ferreira, 2011) measure the *number of unidentified values* per formula cell. Unfortunately, they do not explain the meaning of that term.

3.1.3. Number of Spreadsheets with Smells (NB-SPREADSHEETS_W_SMELL)

The number and percentage of spreadsheets within a corpus that suffer from a certain smell (Multiple References, Multiple Operations, Duplicated Formulas, Long Calculation Chain, Conditional Complexity, or Embedded Constant) (Hermans and Murphy-Hill, 2015) give an idea about the overall quality of a corpus. Examples of spreadsheet smells can be found in Hermans and Murphy-Hill (2015), Hermans et al. (2012b, 2015, 2012a), Jansen (2015), Jansen and Hermans (2015), Koch et al. (2019, 2018), Koch et al. (2019), Abreu et al. (2014a,b), Cunha et al. (2012b,?), Asavametha (2013).

3.1.4. Number of named ranges (NB-NAMED_RANGES)

Some researchers (Shubbak and Thorne, 2016; McKeever and McDaid, 2010) are skeptical of the benefits of named ranges since ambiguous names can be harmful and therefore spreadsheets with named ranges are considered as risky. There are three variants of this metric: *(i)* the *number of spreadsheets that contain named ranges* (Hermans and Murphy-Hill, 2015), *(ii)* the *number of named ranges* (Hermans and Murphy-Hill, 2015; Shubbak and Thorne, 2016), and *(iii)* the *distribution of named ranges* (e.g., the percentage of spreadsheets that use exactly one named range or less than five named ranges) (Hermans and Murphy-Hill, 2015).

3.1.5. File size (FILE_SIZE)

The *physical size of spreadsheets* measured in kilobytes or megabytes (Hermans et al., 2012a; Zhang et al., 2017; Colver, 2011) is intended as a complexity measure, but it is a controversial metric because the file size strongly depends on the underlying software (Colver, 2011). Thus, spreadsheets written with a different software or different versions of the same software cannot be compared with this metric. Furthermore, the size differs for spreadsheets that only link data from those that import data.

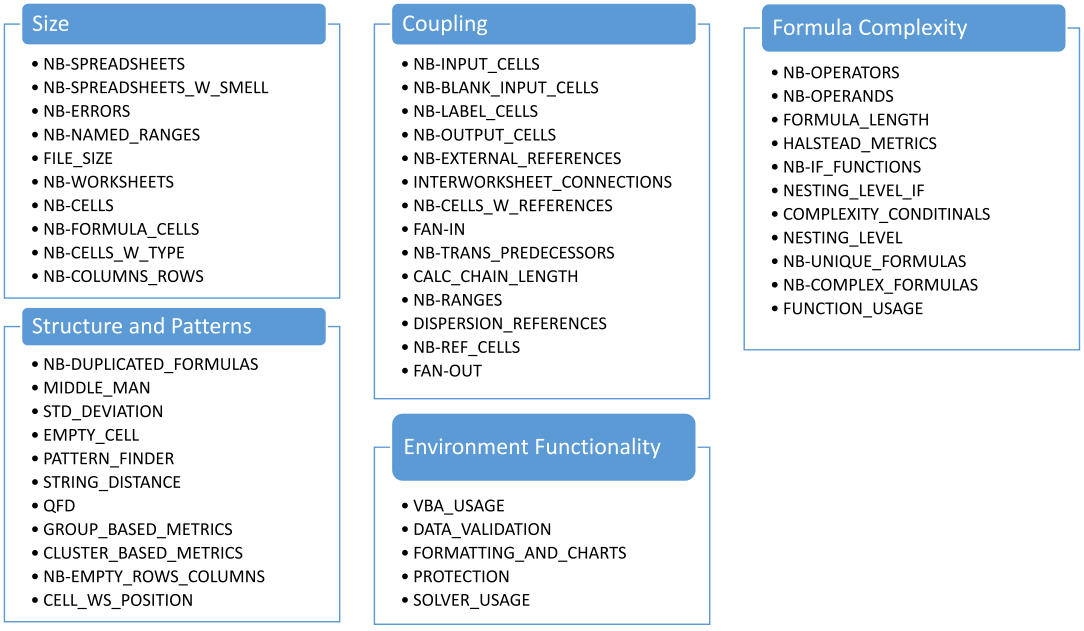

Fig. 2. Overview of 57 metrics organized in five categories.

3.1.6. Number of worksheets (NB-WORKSHEETS)

The *number of worksheets* metric is one of the most used metrics, probably because it is easy to measure. However, this metric can be problematic because there might be many empty worksheets because some spreadsheet programs (e.g. Microsoft Excel) automatically add three worksheets when a new workbook is created. Some researchers count the *number of all worksheets* (Butler, 2000; Hermans et al., 2012a; Jansen and Hermans, 2015; Hermans and Murphy-Hill, 2015; Jansen, 2015; Zhang et al., 2017; Dou et al., 2017; Butler, 2008; Correia and Ferreira, 2011; Colver, 2011; Hermans et al., 2011; Cheung et al., 2016), others in addition count the *number of non-empty worksheets* (Koch et al., 2016; Ballinger et al., 2003). A high *number of empty worksheets* is sometimes considered a sign of an unfinished spreadsheet (Cunha et al., 2012a).

3.1.7. Number of cells (NB-CELLS)

The *number of cells* metric is used to estimate the size of spreadsheets and corpora. There are different forms of this metric: *(i)* the *number of (used) cells* (Hermans et al., 2012a; Koch et al., 2016; Cunha et al., 2012a; Correia and Ferreira, 2011; Shubbak and Thorne, 2016; Hermans et al., 2011; Cheung et al., 2016; Ballinger et al., 2003; Hodnigg and Pinzger, 2015; Birch et al., 2014; Abraham and Erwig, 2007a), *(ii)* the *number of non-empty cells* (Jansen and Hermans, 2015; Hermans and Murphy-Hill, 2015; Jansen, 2015), and *(iii)* the *number of non-empty or referenced blank cells* (Koch et al., 2019; Hodnigg and Mittermeir, 2008). The later metric includes besides all non-empty cells also all empty cells that are referenced. While the terms *non-empty cells* and *non-empty or referenced blank cells* are unambiguous, the term *(used) cells* leaves room for interpretation whether it includes empty cells that are referenced. Considering empty but referenced cells in a metric can be useful, because such cells are used in an equation and potentially influence the computed output in the spreadsheet.

3.1.8. Number of formula cells (NB-FORMULA_CELLS)

The *number of formula cells* is an indicator of the size of spreadsheets (Butler, 2000; Hermans et al., 2012a; Jansen and Hermans, 2015; Jansen, 2015; Koch et al., 2019; Zhang et al., 2017; Dou et al., 2017; Butler, 2008; Koch et al., 2016; Cunha et al., 2012a; Shubbak and Thorne, 2016; Hermans et al., 2011; Cheung et al., 2016; Hodnigg and Pinzger, 2015; Hodnigg and Mittermeir, 2008; Jannach et al., 2019; Reschenhofer et al., 2017; Abraham and Erwig, 2007a). For corpora, the sum and average of all spreadsheets could be additionally computed (Hermans and Murphy-Hill, 2015; Zhang et al., 2017; Fisher and Rothermel, 2005). The ratio of formula cells to non-empty cells, also named *computational share*, is another metric of that type (Zhang et al., 2017; Reschenhofer et al., 2017).

Some researchers (Correia and Ferreira, 2011) measure the number and percentage of *formula cells* and the number and percentage of *calculation step cells*. A calculation step is a formula cell that is referenced and that does not serve as a proxy.[2] Since this metric does not count output cells and proxy cells, its measurement will be lower than that of the above mentioned metrics. Additionally, the authors of (Correia and Ferreira, 2011) measure the number and percentage of *data move cells*, i.e. proxy cells that are referenced.

3.1.9. Number of cells of a type (NB-CELLS_W_TYPE)

The number of cells with a certain type are used to describe corpora. The number of non-empty non-formula cells, input cells, and formulas that evaluate to error, Boolean, date, non-integer number, integer number, and string can be measured (Koch et al., 2016; Fisher and Rothermel, 2005) as well as the number of blank cells (Koch et al., 2016).

3.1.10. Number of non-empty columns and rows (NB-COLUMNS_ROWS)

The number of non-blank columns and rows can be used to estimate both the size of spreadsheets and copora (Correia and Ferreira, 2011; Hermans et al., 2011; Ballinger et al., 2003). The number of non-blank columns is additionally used to determine the "attractiveness" of worksheets. Worksheets with only a few columns are in some works considered more visually attractive than worksheets with many columns (Cunha et al., 2012a). The ratio of non-blank columns and rows allows one to assess whether a spreadsheet is organized vertically or horizontally (Correia and Ferreira, 2011).

[2] A proxy is a formula cell containing only a direct reference, e.g. `=A1`.

3.2. Coupling metrics

Coupling metrics are applied to individual formula cells (e.g., number of references) as well as to worksheets or the whole spreadsheet (e.g., references to other files). Individual cell measurements are often accumulated on the corpus or spreadsheet level using sum, median, average, minimal and maximum values.

3.2.1. Number of input cells and blank input cells (NB-INPUT_CELLS and NB-BLANK_INPUT_CELLS)

Different names are used in the literature to refer to a metric that counts how many cells contains input (in contrast to formula cells and other non-formula cells like labels): *(i)* the *data count* (Hodnigg and Pinzger, 2015; Hodnigg and Mittermeir, 2008) *(ii)* the *number of data source cells* (Correia and Ferreira, 2011) *(iii)* the *number input cells* (Birch et al., 2014; Jannach et al., 2019; Reschenhofer et al., 2017), *(iv)* the *number of non-empty non-formula cells* (Fisher and Rothermel, 2005), *(v)* the *number of manipulated numbers* (Butler, 2008), and *(vi)* the *number of root cells* (Ballinger et al., 2003).

The *data count* metric is also known as *fan-in* or *source* and counts the number of cells that are referenced, but that do not reference other cells (Hodnigg and Pinzger, 2015). Data source cells are non-blank, non-formula cells that are referenced (Correia and Ferreira, 2011). For the other metric names that are used to count input cells, no details are given in the original papers.

Besides absolute numbers, the *data centeredness* can be computed, which corresponds to the percentage of input cells to non-empty cells. The ratio of the *data centeredness* to the computational share (i.e., the number of formula cells) indicates whether the spreadsheet is data or formula centered (Correia and Ferreira, 2011; Birch et al., 2014; Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017).

Additionally, the *number of input cells with values of a certain type* (error, Boolean, date, non-integer number, integer number, string), the *number of input cells that are referenced by other cells*, and the *number of input cells that contain a certain type and that are referenced by other cells* can be computed (Fisher and Rothermel, 2005).

The number of blank cells (Cunha et al., 2012b; Koch et al., 2016; Cunha et al., 2012a; Correia and Ferreira, 2011) is considered as an indicator for a work-in-progress spreadsheet (Cunha et al., 2012a). It also serves as a basis for the *Reference to empty cells* smell (Koch et al., 2019; Cunha et al., 2012,b; Koch et al., 2018; Koch et al., 2019; Abreu et al., 2014a,b).

3.2.2. Number of labels (NB-LABEL_CELLS)

The number of labels (Butler, 2000, 2008; Correia and Ferreira, 2011; Ballinger et al., 2003; Hodnigg and Pinzger, 2015; Hodnigg and Mittermeir, 2008; Bregar, 2008) shows how well a spreadsheet is documented. Labels can be defined in three ways: *(i)* as cells that have neither incoming nor outgoing references (Ballinger et al., 2003), *(ii)* as unreferenced non-formula cells (Correia and Ferreira, 2011; Hodnigg and Pinzger, 2015; Bregar, 2008) and *(iii)* as constants (text & numeric cells) (Butler, 2000). Furthermore, the *information ratio* can be computed, which is the ratio or percentage of label cells to non-empty cells (Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008).

3.2.3. Number of output cells (NB-OUTPUT_CELLS)

Output cells are cells that reference other cells but that are not referenced (Hodnigg and Mittermeir, 2008). There are two ways to indicate the number of output cells: *(i)* absolute (Correia and Ferreira, 2011; Ballinger et al., 2003; Hodnigg and Pinzger, 2015; Hodnigg and Mittermeir, 2008), or *(ii)* as percentage of all cells (Correia and Ferreira, 2011).

3.2.4. Number of external references (NB-EXTERNAL_REFERENCES)

One has to invest more time to understand spreadsheets that contain external references (Butler, 2000). Therefore, the number of references to other spreadsheets (Butler, 2000; Jansen and Hermans, 2015; Jansen, 2015; Butler, 2008) may help to estimate the time required to inspect a spreadsheet.

3.2.5. Inter-worksheet connections (INTER_WORKSHEET_CONNECTIONS)

Spreadsheets whose worksheets are more interconnected are more difficult to understand and to maintain. There are several ways to measure the inter-worksheet connections: *(i)* the *total number of inter-worksheet connections* (Jansen and Hermans, 2015; Jansen, 2015; Cunha et al., 2012a; Butler, 2008), *(ii)* the *percentage of spreadsheets with inter-worksheets connections* (Jansen, 2015), *(iii)* the *number of references to another worksheet* for each cell (Koch et al., 2019; Jansen and Hermans, 2017), *(iv)* the *number of predecessors that lie in other worksheets* (Koch et al., 2019), *(v) Efferent coupling (independence)* (Birch et al., 2014), *(vi) Afferent coupling (responsibility)* (Birch et al., 2014), and *(vii) Instability* (Birch et al., 2014).

While the *number of predecessors* metric counts every referenced cell only once, the *number of references* metric counts multiple coinciding references (Koch et al., 2019). *Efferent coupling* is the number of worksheets referenced by cells of the given worksheet. The lower the value, the more independent the worksheet is. *Afferent coupling (responsibility)* is the number of worksheets that reference cells in the given worksheet. Worksheets with a high *afferent* value are difficult to adapt because many other worksheets rely on this worksheet. *Instability* is computed by dividing the Efferent Coupling metric value by the sum of the Afferent Coupling and the Efferent Coupling values (Birch et al., 2014).

There are different smells designed to identify worksheets that have strong connections to other worksheets. A worksheet suffers from the *Inappropriate Intimacy* smell when it references too many cells of another worksheet. A cell suffers from the *Shotgun Surgery* smell when it is referenced by many other formulas of different worksheets, because a change in this cell might result in changes in many other cells. The *Shotgun Surgery* smell can be expressed on the formula and on the worksheet level. If a formula is more "interested" in cells from another worksheet, it suffers from the *Feature Envy* smell and it would be better to move the formula to that worksheet (Hermans et al., 2012a).

3.2.6. Number of cells with references and Fan-In (NB-CELLS_W_REFERENCES and FAN-IN)

The number of formula cells referencing other cells were generally found as more difficult to read and comprehend by spreadsheet users (Hermans et al., 2012c).

The following different metrics were used: *(i)* the *number of formula cells that contain references to other cells* (Fisher and Rothermel, 2005; Cunha et al., 2012a), *(ii)* the *number of predecessors* per formula (Jansen, 2015; Koch et al., 2019), *(iii)* the *number of references* (also called fan-in or scope) (Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Koch et al., 2019; Cunha et al., 2012a; Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017; Bregar, 2008; Hermans et al., 2012c), *(iv)* the *number of range references* (Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Koch et al., 2019), *(v)* the *number non-local references* (Correia and Ferreira, 2011), and *(vi)* the *Valency* (Birch et al., 2014).

The *number of references* metric counts all references, while the *number of predecessors* metric counts several references to the same cell only once (Koch et al., 2019). The fan-in can be computed on the formula and the worksheet level (Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008). For ranges, one could

	A	B	C	D
1	1	=A1	=B1+B2	=A1+B1+C1
2	1	=A2		

Fig. 3. Spreadsheet example to illustrate CALC_CHAIN_LENGTH and NB-TRANS_PREDECESSORS.

additionally distinguish simple and combined range references (e.g., *A1:B3:C2*), and references in union form (e.g., *A1;B1*) (Koch et al., 2019). The number of references/ranges a formula is referring to is the basis for the *Multiple References* smell because a formula containing many references is difficult to understand (Koch et al., 2019; Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Koch et al., 2018; Abreu et al., 2014a,b). The *Valency* is the average number of cells each cell references and is referenced by Birch et al. (2014).

3.2.7. Calculation chain length and number of transitive predecessors (CALC_CHAIN_LENGTH and NB-TRANS_PREDECESSORS)

The length of a calculation chain is the number of cells that have to be traversed to reach the referenced cell that is farthest away (Hodnigg and Mittermeir, 2008; Bregar, 2008). Cells with a long calculation chain are considered as smelly (Koch et al., 2019; Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Jansen, 2015; Koch et al., 2018; Koch et al., 2019; Correia and Ferreira, 2011; Abreu et al., 2014a,b; Hermans et al., 2012c). The length of the chain of referencing cells can also be computed (Hodnigg and Mittermeir, 2008).

Related metrics in this context are *(i)* the *reachability* (Koch et al., 2019; Bregar, 2008), *(ii)* the *number of paths* (Correia and Ferreira, 2011; Bregar, 2008), and *(iii)* the *number of transitive predecessors* (Jansen and Hermans, 2015; Jansen, 2015; Correia and Ferreira, 2011).

The reachability of a cell is recursively computed as the sum of the reachability values of all referenced cells. The reachability of input cells is one. The *average reachability* is the sum of the reachability values of all cells (including input cells) divided by the number of cells (Bregar, 2008) .

Example 1. The values for cell D1 of the toy spreadsheet illustrated in Fig. 3 for these different metrics are as follows: CALC_CHAIN_LENGTH (D1)=3 (A1→B1→C1→D1), NB-TRANS_PREDECESSORS(D1)=5 (A1, A2, B1, B2, C1), reachability(D1)=4 (reachability(A1)=1 + reachability(A1)=1 + reachability(A3)=2).

3.2.8. Number of ranges (NB-RANGES)

The idea of this metric is based on the observation that formulas referring to one range of cells are less prone to error than the ones that refer to multiple ranges or individual cells, which are counted as singleton ranges. The *number of ranges* (Hermans et al., 2012c) metric returns the number ranges referenced in the input formula. Besides measuring the *number of ranges*, one could measure the *cumulative range height growth*, the *cumulative range width growth* of each formula cell, i.e., the sum of the row heights and/or column widths - 1 of the referenced ranges, and the number of references to (non-complex) ranges and combined range references (e.g., *A1:B2:C2*) (Koch et al., 2019).

3.2.9. Dispersion of references (DISPERSION_REFERENCES)

These metrics measure the physical distance of a cell to its referenced cells. The rationale behind these metrics is that faults are assumed to occur more frequently when the referenced cells are located in different rows, columns or worksheets. The following metrics are computed in this context: *(i)* the *dispersion of references* (Koch et al., 2019; Bregar, 2008) or *spreading factor* (Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017), *(ii)* the *row, column, and worksheet spans* of a cell's references (Koch et al., 2019; Bregar, 2008), *(iii)* the *percentage of reverse references* and *references in the same row/column* (Hermans et al., 2012c), *(iv)* the *number of distant references* (Koch et al., 2019; Hermans et al., 2012c), *(v)* the *number of distant and reverse predecessors* (Koch et al., 2019), and *(vi)* the *number of referenced cells that are in a different row, column or worksheet* (Koch et al., 2019).

The *dispersion of references* is the sum of the distances the cell has to its references (Bregar, 2008). The *row, column, and worksheet spans* for a formula cell are the distances between the left-most and right-most, the top-most and bottom-most, and the maximum worksheet distance of the referenced cells (Koch et al., 2019). A reference is considered as reverse if it is located in a worksheet that is right to the current cell's worksheet or if it is located in the same worksheet but right to and/or below the current cell. A reference is considered as distant if it is further away as 10 columns or 25 rows or if it is located in a different worksheet (Hermans et al., 2012c), or if it is further away than a previously defined number of columns and rows (Koch et al., 2019).

3.2.10. Number of referenced cells and Fan-Out (NB-REF_CELLS and FAN-OUT)

A large number of references in worksheet formulas make them hard to define, analyze, and verify. Therefore, measuring the number of references in a spreadsheet might be a good indicator of its complexity and, consequently, possibility to contain some quality issues. Several metrics focus on the number of references in a worksheet: *(i)* the *number of referenced cells* (Cunha et al., 2012a), *(ii)* the *number of referenced formula cells* (Fisher and Rothermel, 2005), *(iii)* the *fan-out/visibility* (i.e. how often a cell is referenced) (Koch et al., 2019; Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017; Bregar, 2008), *(iv)* the *number of ranges referring to a cell* (Koch et al., 2019), *(v)* how often a cell is referenced from other worksheets (Koch et al., 2019), *(vi)* the *number of successors* in total and in other worksheets (Koch et al., 2019), and *(vii)* the *number of cells* in total and in other worksheets *that refer to cells of this worksheet* (Koch et al., 2019). All these metrics are computed on the worksheet level and allow one to assess the complexity of a worksheet as a whole. Evaluation reports, e.g., (Koch et al., 2019; Hodnigg and Mittermeir, 2008), show that these metrics can efficiently separate simple spreadsheets that contain only few formulas from more complex ones that possibly require a more detailed analysis for quality issues.

3.3. Formula Complexity Metrics

Formula Complexity Metrics are applied to individual formula cells as well as to worksheets. The individual measurements are often accumulated on the corpus or spreadsheet level using sum, median, average, minimum, and maximum values.

3.3.1. Number of operators (NB-OPERATORS)

Complex formulas with multiple operators are widely recognized as a source of various quality issues. The following metrics are used to reason about the complexity of spreadsheets but also to identify individual complex formulas: *(i)* the *total number of operators* (Kokol, 1988; Thorne et al., 2008; Bregar, 2008; Hodnigg and Pinzger, 2015; Koch et al., 2018; Colver, 2011), *(ii)* the

number of distinct operators (Kokol, 1988; Thorne et al., 2008; Jansen, 2015), *(iii)* the *total number of function calls* (Reschenhofer et al., 2017; Colver, 2011), and *(iv)* the *unique number of function calls* (Reschenhofer et al., 2017; Jansen and Hermans, 2015; Colver, 2011)

The total number of functions used in a formula is used as basis for the *Multiple Operators* smell (Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Abreu et al., 2014a,b; Koch et al., 2019). The rationale of this smell is that formulas with many operators are difficult to understand. The average and maximal *number of functions per formula* and the average and maximal *number of distinct functions per formula* reflect the spreadsheet's lexical diversity (Reschenhofer et al., 2017) and they are one factor of spreadsheet auditing costs (Colver, 2011).

3.3.2. Number of operands (NB-OPERANDS)

Formulas with more operands are considered to be error-prone. The number of operands can be expressed as the *total number of operands* (Kokol, 1988; Thorne et al., 2008; Bregar, 2008; Hodnigg and Pinzger, 2015; Koch et al., 2019) and the *number of unique operands* (Kokol, 1988; Thorne et al., 2008).

3.3.3. Formula length (FORMULA_LENGTH)

The length of formulas can be measured in terms of the *number of characters* (Hodnigg and Pinzger, 2015; Jansen and Hermans, 2015; Jansen, 2015) or in terms of the *number of elements*, i.e., the number of operators and operands (Correia and Ferreira, 2011; Koch et al., 2019).

3.3.4. Halstead metrics (HALSTEAD_METRICS)

Several metrics in the literature were derived from famous Halstead measures (Halstead, 1977) developed in the context of general software engineering. These measures were one of the first attempts to identify measurable properties of general software artifacts. In particular, the goal of Halstead was to quantify the complexity of software artifacts and thus detect possible quality issues. Similarly, when applied to spreadsheets these metrics consider the total and unique number of operators and operands appearing in formulas of a spreadsheet. In particular, the metrics consider: *(i) Program length* ($n = n_1 + n_2$ with n_1 and n_2 being the number of distinct operators and operands) (Kokol, 1988; Reschenhofer et al., 2017), *(ii) Vocabulary* ($N = N_1 + N_2$ with N_1 and N_2 being the total number of operators and operands) (Kokol, 1988), and *(iii)* their variations *Estimated program length* ($N^* = n_1 * log_2 n_1 + n_2 * log_2 n_2$), *Program volume* ($V = N * log_2 n$), *Program level* $L = \frac{2}{n_1} * \frac{n_2}{N_2}$), *Language level* ($\lambda = L^2 * V$), *Complexity* ($C = \frac{2*n_1}{n_2*N_2}$) (Kokol, 1988; Thorne et al., 2008).

3.3.5. Number and nesting level of IF functions (NB-IF_FUNCTIONS and NESTING_LEVEL_IF)

Formulation of correct logic expressions is a complex and error-prone task. As a result, the use of conditionals increases the complexity of formulas (Hermans et al., 2012c) and, thus, the risk associated with a spreadsheet (Shubbak and Thorne, 2016). The following metrics can be used in this context (Hermans et al., 2012c; Zhang et al., 2017): *(i)* the *number of spreadsheets with IF functions*, *(ii)* the *number of IF functions* per spreadsheet or corpus, *(iii)* the *nesting level of IF functions*, and *(iv)* the *distribution of the nesting level*, i.e., the number of formulas that have 1–5, 6–10, 10–15, and 15–65 nested IF formulas.

3.3.6. Complexity of conditionals (COMPLEXITY_CONDITIONALS)

The complexity of conditionals is a good indicator for the complexity of formulas. Conditionals are Boolean expressions in a formula, i.e., AND, OR, XOR, NOT, IF, SWITCH, IFS, IFERROR, IFNA, COUNTIF, COUNTIFS, SUMIF, SUMIFS. The following metrics are used in this context: *(i)* the *number of conditionals* per formula (Reschenhofer et al., 2017; Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Abreu et al., 2014a,b; Hodnigg and Mittermeir, 2008; Correia and Ferreira, 2011; Koch et al., 2018; Koch et al., 2019), *(ii)* the *decision count* (Bregar, 2008; Koch et al., 2019), and *(iii)* the *complexity of a conditional construct* (Bregar, 2008).

The *decision count* is the number of simple conditions (conjunction and disjunction) within one formula. The *complexity of a conditional construct* considers the number of nested or preceding conditionals and the number of computational cascades without conditions (Bregar, 2008). A cell whose number of conditionals exceeds a certain threshold is considered as smelly (Koch et al., 2019).

3.3.7. Nesting level (NESTING_LEVEL)

The nesting level of a formula provides insight into the complexity of formulas. There are three variants of this metric: *(i)* the *depth of nesting* (Bregar, 2008; Reschenhofer et al., 2017; Hermans et al., 2012c; Jansen and Hermans, 2015; Jansen, 2015; Koch et al., 2019), *(ii)* the *nesting level of a token* (Bregar, 2008), and *(iii)* the *average nesting level* (Bregar, 2008; Reschenhofer et al., 2017; Koch et al., 2019; Hodnigg and Mittermeir, 2008; Hodnigg and Pinzger, 2015).

The depth of nesting is computed by measuring the depth of the formula's abstract syntax tree (AST). The nesting level of a token is the depth of an individual formula element in the AST. The average nesting level is the sum of the nesting levels of operands and operators divided by the number of operands and operators (Bregar, 2008).

3.3.8. Number of unique formulas (NB-UNIQUE_FORMULAS)

Quite often spreadsheet engines allow its users to automatically copy formulas by fixing their structure and changing only references to input cells. Such copied formulas can easily be recognized, since they are equivalent with respect to the relative R1C1 notation, i.e., these cells are *copy-equivalent*. Therefore, analysis of formulas in a spreadsheet can often be reduced to the inspection of the unique formulas from which all others were copied. The larger is the number of unique formulas in a spreadsheet, the harder it is for a user to identify any quality issues. This observation was used to define the *number of unique formulas* that can be computed: *(i)* per worksheet (Koch et al., 2019) *(ii)* per spreadsheet (Jansen and Hermans, 2015; Jansen, 2015; Colver, 2011; Koch et al., 2016; Hermans et al., 2012a; Jannach et al., 2019; Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017; Butler, 2008), and *(iii)* per corpus (Zhang et al., 2017; Hermans and Murphy-Hill, 2015; Fisher and Rothermel, 2005).

For corpora, the average value per spreadsheet can also be indicated (Zhang et al., 2017). Furthermore, variants of this metric such as the *percentage of unique formulas* (Correia and Ferreira, 2011), the *number of formulas that occur more than once* in a spreadsheet and the *number of times the most frequently occurring formula occurs* in a spreadsheet (Fisher and Rothermel, 2005) are considered in the literature.

3.3.9. Number of complex formulas (NB-COMPLEX_FORMULAS)

Complex functions that use logical criteria to retrieve data from columns and/or rows of the same or other spreadsheets can be hard to formulate and test for non-expert spreadsheet programmers. Therefore, spreadsheets that comprise numerous formulas using such functions are prone to different quality issues. To detect such issues Fisher and Rothermel (2005) suggest a metric that counts the number of formula cells that use SUMIF, COUNTIF, CHOOSE, HLOOKUP, INDEX, INDIRECT, LOOKUP, MATCH, OFFSET, and IF. Also, Cunha et al. (2012a) use the *number of complex formulas* as metric, but they do not define what they consider a complex formula.

3.3.10. Function usage (FUNCTION_USAGE)

Research results presented in Jansen (2015) indicate that in most formulas spreadsheet programmers use only a very limited number of functions, such as SUM, AVERAGE, ROUND, etc. Consequently, the programmers have less experience with rarely used functions and may introduce more quality issues in spreadsheets using them. Metrics focusing on these issues are *(i)* the *frequency of function usage per category* (e.g., logical, mathematical and trigonometry, database) (Ballinger et al., 2003; Jansen, 2015), *(ii)* the *15 most used functions* (Hermans and Murphy-Hill, 2015; Jansen, 2015) *and operators* (Hermans and Murphy-Hill, 2015), *(iii)* the ten most used vector functions (i.e. SUM, MIN, AVERAGE, MAX, PRODUCT, MATCH, OFFSET, VLOOKUP, INDEX, and CONCATENATE) (Barowy et al., 2014), and *(iv)* the *distribution of the number of used distinct functions* (Hermans and Murphy-Hill, 2015).

3.4. Structure and Pattern-based Metrics

The majority of these metrics are computed for individual cells. Only the *number of labeled empty rows/columns* are computed per worksheet and the *group-based metrics* are computed for blocks of connected cells.

We explain the following concepts and metrics by means of the worksheets shown in Fig. 4. These worksheets compute the total coffee consumption of three departments in order to make an informed decision for buying a new coffee machine.

3.4.1. Number of duplicated formulas (NB-DUPLICATED_FORMULAS)

Duplicated formulas are formula cells that belong to the same worksheet and whose abstract syntax trees (ASTs) derived from the R1C1 representation of each formula share at least one proper subtree, but that are not identical (Hermans et al., 2012b). As discussed in the previous section, in R1C1 representation formulas use relative references and therefore formulas automatically copied by a spreadsheet tool have similar ASTs. If ASTs of any two formulas in R1C1 notation have intersections, then it means that one of these formulas is manually modified copy of another. Since the "copy/paste" way of programming is a well known source of quality issues in the general software development, we can assume that the same holds also for spreadsheets. Therefore, when the number of cells with such "copy/paste" formulas exceeds a certain threshold, a worksheet is considered as smelly (Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Abreu et al., 2014a,b; Koch et al., 2019, 2018; Koch et al., 2019).

Example 2. The formulas from E9, E10, and E11 from worksheet Investment (Fig. 4(c)) are identical in R1C1 notation and the ASTs of other formulas in this worksheet (B3 and B5) do not share a common subtree. Therefore the metric NB-DUPLICATED_FORMULAS (Investment) has a value of 0.

	A	B	C	D	E	F
1	**Coffee Consumption: Department 1**					
2		**Employee**				
3	**Quarter**	**Anderson**	**Bourne**	**Connor**	**Dredd**	**Total**
4	1	35	1745	80	98	=SUM(B4:E4)
5	2	47	21	51	66	=SUM(B5:E5)
6	3	46	12	46	39	=SUM(B6:E6)
7	4	74	15	57	43	=SUM(B7:E7)
8	**Total**	=SUM(B4:B7)	=SUM(C4:C7)	=SUM(D4:D7)	=SUM(E4:E7)	=SUM(F4:F7)

(a) Department1

	A	B	C	D	E
2		**Department**			
3	**Quarter**	**Departement 1**	**Department 2**	**Depratment 3**	**Total**
4	1	=Department1!F4	=Department2!F4	=Department3!F4	=SUM(B4:D4)
5	2	=Department1!F5	=Department2!F5	=Department3!F5	=SUM(B5:D5)
6	3	=Department1!F6	=Department2!F6	=Department3!F6	=SUM(B6:D6)
7	4	=Department1!F7	=Department2!F7	=Department3!F7	=SUM(B7:D7)
8	**Total**	=Department1!F8	=Department2!F8	=Department3!F8	=SUM(B8:D8)

(b) Total

	A	B	C	D	E
1	**Investment Options**				
2	**Input**				
3	**Coffee / year**	=Total!E8			
4	**Service years**	3			
5	**Coffee total**	=B3*B4			
6					
7	**Comparison**				
8	**Type**	**Investment**	**Coffee price**	**Yearly service**	**Total**
9	Capsule	100	0,3	100	=B9+C9*B5+D9*B4
10	Drip	250	0,15	100	=B10+C10*B5+D10*B4
11	Automatic	800	0,1	200	=B11+C11*B5+D11*B4

(c) Investment

Fig. 4. Formula views of different worksheets used to explain concepts and metrics.

3.4.2. Middle man (MIDDLE_MAN)

Middle Man or *data move* cells are formula cells that consist of a single direct reference Hermans et al. (2012a). Chains of *Middle Man* cells make a spreadsheet more complicated than necessary and make changes more difficult. The number and percentage of *data move cells* point out spreadsheets suffering from this problem (Correia and Ferreira, 2011; Hermans et al., 2012a; Koch et al., 2019; Jansen, 2015; Koch et al., 2018; Koch et al., 2019).

Example 3. The cells of the area B4:D8 in the worksheet Total (Fig. 4(b)) as well as cell B3 in worksheet Investment (Fig. 4(c)) are data move cells. Therefore, the metric *number of data move cells* (Total)=12 and *number of data move cells* (Investment)=1. Since there are no chains of such data move cells, MIDDLE_MAN is zero.

3.4.3. Group-based metrics (GROUP_BASED_METRICS)

The application of metrics to *all* cells of a spreadsheet can sometimes be suboptimal. In some cases, the cells in different parts of a spreadsheet perform semantically quite different computations. For instance, a financial spreadsheet might first compute coefficients from client data and from market statistics and then combine the results of both computations to determine the credit rating for the client. Therefore, the results that are computed by metrics, e.g., smells, when applied to all cells of a spreadsheet might be inaccurate. To counteract this issue, researchers suggested methods to identify groups of cells by their functionality and then apply the metrics to such groups.

Koch et al. (2019) differentiate between input groups, formula groups, computation blocks, and headers. Each group is a set of cells from one column/row that are designed for the same purpose. Thus, input groups provide data values and do not

refer to any other cells. Formula groups are sets of formula cells that perform the same computations, e.g., determine a sum of some input values, which can be either input or formula groups. Input and formula groups can be used to define computation blocks, which are rectangular areas containing related value, formula, and empty cells. Examples where metrics are applied to groups of cells include the *Overburdened worksheet smell* (counting the number of computation blocks within a worksheet), the *Inconsistent formula group reference smell*, and the *Missing header smell* (Koch et al., 2019, 2018).

Example 4. The `Investment` worksheet (Fig. 4(c)) has seven different type-based groups ({A1:A5}, {B3}, {B4}, {B5}, {A7:A11; B8:E8}, {B9:D11}, and {E9:E11}) and two calculation blocks ({B3:B5} and {B9:E11}). The number of calculation blocks can be used to identify overburdened worksheets.

3.4.4. Pattern-based smells (STD_DEVIATION, EMPTY_CELL, STRING_DISTANCE, PATTERN_FINDER, and QFD)

Pattern matching is probably the simplest and the most intuitive way for spreadsheet developers to identify quality issues that occur frequently in spreadsheets. In general, each pattern is a heuristic rule that spreadsheet users derive from their experience gained while searching for faults in spreadsheets. For instance, cells in a spreadsheet that contain values or formulas are often grouped in square-like structures that usually have no empty cells between them. As a result, an empty cell surrounded by non-empty cells is very uncommon and can be used as an indicator for a developer for possible quality issues.

In the literature we could find a number of spreadsheet smells using pattern matching as an issue detection metric that maps a set of cells to Boolean values {0, 1} (Cunha et al., 2012b,?; Abreu et al., 2014a,b): *(i)* a cell with a numerical value suffers from the *Standard Deviation smell* when its value is more than two standard deviations away from the mean value of the values of numeric cells in the same row or column; *(ii)* an empty cell which is surrounded by non-empty cells suffers from the *Empty Cell smell*; *(iii)* a cell containing a string suffers from the *String Distance smell* when its string differs only slightly from a string of another cell; *(iv)* the *Pattern Finder smell* is used to find cells whose value type differs from the type of the surrounding cells; and *(v)* the *Quasi-Functional Dependency smell* detects deviations in data table entries.

The *Standard Deviation*, *Pattern Finder*, *String Distance*, and *Quasi-Functional Dependency* smells can be improved by performing the necessary comparisons within groups of cells instead of columns and rows (Koch et al., 2019). Moreover, the *Empty Cell* smell can be extended to take into account all empty cells of a computation block (Koch et al., 2019). The *Pattern Finder smell* can be reported separately for rows and columns (Koch et al., 2018; Koch et al., 2019).

Example 5. The value in cell C4 of the `Department1` worksheet (Fig. 4(a)) is extraordinary high compared to its surrounding numbers. Therefore, the *number of input cells suffering from the Standard Deviation smell* for worksheet Department1 is one (STD_DEVIATION (Department1)=1). The string distance finder would detect the typo in the header of cell D3 in worksheet Total (Fig. 4(b)). Therefore, STRING_DISTANCE (Total)=1.

3.4.5. Cluster-based metrics (CLUSTER_BASED_METRICS)

AMCheck (Dou et al., 2014) and CACheck (Dou et al., 2017) identify quality issues by checking if cells placed in a proximity to each other are also similar with respect to the computational semantics. The authors suggest to identify abnormalities in arrays of cells through the *ambiguous computation smell* that exploits discrepancies between the intended and the actual computational semantics of formula cells. The intended semantics is based on the fact that spreadsheet users usually do not mix formulas doing different computations in one row or column. Instead, they use some built-in functionality of their tools to copy/paste or auto-fill some formula into multiple cells thus creating arrays of cells performing similar computations. Therefore, positions of formulas can be used to determine if they are intended to perform similar computations. The actual semantics is determined by the analysis of a formula structure. That is, any two formulas are considered as semantically equivalent if they have the same references in R1C1 notation and do the same computations with respect to standard algebraic definitions. For instance, in CACheck the similarity of computations is verified using a satisfiability modulo theories (SMT) Solver Z3, which can identify that, e.g., $2 * (A1 + B1)$ and $2 * A2 + 2 * B2$ are equal.

CUSTODES (Cheung et al., 2016) is an unsupervised learning approach that automatically clusters cells to detect if a spreadsheet is suffering from the *missing formula smell* and the *dissimilar formula smell*. The tool first computes different features of spreadsheets parts and then use clustering algorithms to group similar cells with the goal of detecting outliers — missing or dissimilar formulas. CUSTODES uses a two-stage clustering technique which is based on strong and weak features of a spreadsheet. The strong features are extracted from groups of cells that are in a proximity to each other and the weak features are computed from cell labels, standard layouts, fonts, etc., that allow the tool to automatically adapt to varying tabulation styles.

3.4.6. Number of labeled empty rows and columns (NB-EMPTY_ROWS_COLUMNS)

The *number of labeled rows and columns that are empty* are indicators that a spreadsheet is still under development (Cunha et al., 2012a).

3.4.7. Position of cell in worksheet (CELL_WS_POSITION)

Quite often position of a cell in a worksheet can be used as an indicator for its intended purpose and functionality. For instance, headers are usually placed in top rows or leftmost columns, formulas finalizing the computations in the bottom rows, etc. Metrics extracting position features such as *(i)* the *column and row numbers of cells*, *(ii) column and row distances to the next cell that has the same type*, *(iii) column number of the right-most non-empty cell*, *(iv) row number of the bottom-most non-empty cell in a worksheet*, and *(v) position of the worksheet in the spreadsheets tab* can be used as features in a machine learning approach in order to predict faulty cells (Koch et al., 2019).

3.5. Environment Functionality-based Metrics

The following metrics are computed per spreadsheet or corpus.

3.5.1. Usage of VBA and macros (VBA_USAGE)

The development of VBA macros requires a high level of programming expertise from spreadsheet users. Since in most of the cases this expertise might be insufficient or even missing, even one macro can pose a significant threat to the overall quality of a spreadsheet (Cunha et al., 2012a; Shubbak and Thorne, 2016). As a result, a number of metrics were proposed in the literature to detect possible issues: *(i)* the *number of spreadsheets using VBA macros* (Fisher and Rothermel, 2005), *(ii)* the *number of spreadsheets that have user-defined functions* (Hermans and Murphy-Hill, 2015), *(iii)* the *number of user-defined functions* per spreadsheet (Hermans and Murphy-Hill, 2015), and *(iv)* the *frequency of usage* (Hermans and Murphy-Hill, 2015).

3.5.2. Use of data validation (DATA_VALIDATION)

The *use of data validation tools* such as drop-down lists and data validity constraints (Cunha et al., 2012a) and the *number and percentage of spreadsheets using test formulas* (Hermans and Murphy-Hill, 2015) fall in this category. Test formulas are IF formulas that have an OK message and an error message for the "then" and "else" branches and they are used to check, e.g., if percentage values sum up to 100%. These metrics indicate the maturity of spreadsheets because severe errors can be prevented or discovered early when using these techniques (Hermans and Murphy-Hill, 2015).

3.5.3. Formatting and charts (FORMATTING_AND_CHARTS)

While solving complex decision-making problems users might tend to create large spreadsheets that comprise all relevant data, computations, analysis of results, visualizations, etc. Such large spreadsheets are hard to analyze and therefore they are more prone to errors and different quality issues. Different metrics can help the users to navigate in such spreadsheets, understand their complexity, and detect possible issues. Thus, the *existence of different colors and/or different worksheets for input, output, and formula cells* can help spreadsheet users to understand and edit a spreadsheet. The *number of hidden rows and/or columns* represents a risk to the spreadsheet's security and understability (Cunha et al., 2012a). Furthermore, the *number of spreadsheets with charts* (Fisher and Rothermel, 2005) and the *number of charts per spreadsheet* (Shubbak and Thorne, 2016) can be used to assess the complexity and size of a spreadsheet.

3.5.4. Spreadsheet protection (PROTECTION)

The *number of protected cells for writing* and *password locking the workbook* are indicators of security (Cunha et al., 2012a).

3.5.5. Usage of the solver (SOLVER_USAGE)

Some spreadsheet tools, like Microsoft Excel, offer specific add-ins that extend standard functionality with various optimization methods for problems, such as linear programming or integer programming. Formulation and solution of problems in a spreadsheet and their solution with available solvers is highly complicated and should be done only by very experienced users. Therefore, in Shubbak and Thorne (2016) the authors propose a metric that identifies and reports usages of a solver as a quality risk indicator.

4. Answers to the research questions

In this section, we discuss and answer our three proposed research question in detail.

4.1. RQ1: Which spreadsheet product metrics are defined and used in the literature and for which purposes?

In the previous section, we explained the identified metrics in detail. In Table 2 we summarize how often these metrics and their variations are used or defined in literature. As can be seen in the table, Size Metrics, Coupling Metrics, and Formula Complexity Metrics are the most frequently used metric categories. Regarding individual metrics, NB-FORMULA_CELLS is the most frequently used one in the literature. NB-WORKSHEETS and NB-CELLS are often used Size Metrics. FAN-IN and CALC_CHAIN_LENGTH are the most used Coupling Metrics. NB-OPERATORS, COMPLEXITY_CONDITIONALS and NB-UNIQUE_FORMULAS are the most used Formula Complexity Metrics. The reason behind this observation might lie on the use of metrics in practice, or the ease of extracting these metrics from the spreadsheets. As we will see in the next paragraph, obtaining a measure for complexity is one of the main application areas where the metric categories Size Metrics, Coupling Metrics, and Formula Complexity Metrics seems to be appropriate.

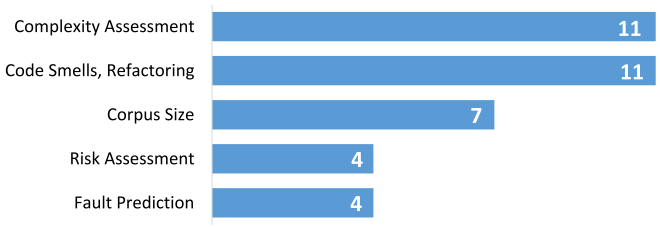

Fig. 5. Number of papers that use metrics for a certain purpose.

Fig. 5 illustrates for what purposes the metrics are used in the papers. To identify these different types of purposes, we analyzed the primary studies considered in our survey with respect to their main *research context*, i.e., in which particular problem setting the authors considered the use of spreadsheet metrics.[3]

As can be seen in Fig. 5, the majority of the papers, namely eleven, use metrics to assess the complexity of individual spreadsheets or to detect smells and/or propose cells that should be refactored. Seven papers use metrics to describe and/or compare corpora. Four papers use metrics to assess risks that come with spreadsheets and to decide which spreadsheets should be audited. Four papers use metrics to predict faulty cells in spreadsheets.

4.2. RQ2: Are the definitions of the metrics used in the literature consistent?

While the naming and definition of the metrics is generally intuitive in the literature, we identified misleading names and fundamental differences in some definitions, as will be shown below. Hence, **RQ2** can be answered with "no", i.e., the definitions of the metrics used in the literature are not always consistent.

Differences in naming. While the majority of the authors (Zhang et al., 2017; Jansen and Hermans, 2015; Jansen, 2015; Colver, 2011; Koch et al., 2016; Jannach et al., 2019; Hermans and Murphy-Hill, 2015; Koch et al., 2019; Correia and Ferreira, 2011; Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017) use the term *unique* to count identical formulas only once, Butler (2008) uses the term *unique* to count only those formulas which are not replicated in a worksheet. He uses the term *original* for formulas which are replicated in a worksheet. While Butler uses the term *external references* to count how often other spreadsheets are referenced, Jansen and Hermans (2017) use this term to count how often other worksheets are referenced.

Counting of worksheets. While some authors (Koch et al., 2016; Ballinger et al., 2003; Cunha et al., 2012a) explicitly mention that they count the *number of non-empty worksheets*, other authors (Zhang et al., 2017; Jansen and Hermans, 2015; Jansen, 2015; Hermans et al., 2011; Hermans and Murphy-Hill, 2015; Correia and Ferreira, 2011; Butler, 2008, 2000; Colver, 2011) do not mention whether they count all worksheets or only non-empty worksheets. Since some spreadsheet programs like Mi-

[3] Note that several metrics can in principle be used for more than one purpose. Our categorization here is, however, based on the main purpose as described in the primary studies.

Table 2
Metrics and their frequency of usage in literature.

Category	Metric	Frequency	Sum
Size Metrics	NB-SPREADSHEETS	● ● ● ● ● ● ●	7
	NB-SPREADSHEETS_W_SMELL	●	1
	NB-ERRORS	● ● ● ● ●	5
	NB-NAMED_RANGES	● ●	2
	FILE_SIZE	● ● ●	3
	NB-WORKSHEETS	● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	15
	NB-CELLS	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	16
	NB-FORMULA_CELLS	● ●	21
	NB-CELLS_W_TYPE	● ●	2
	NB-COLUMNS_ROWS	● ● ● ●	4
Coupling Metrics	NB-INPUT_CELLS	● ● ● ● ● ● ● ● ● ●	10
	NB-BLANK_INPUT_CELLS	● ● ● ● ● ● ● ● ● ●	10
	NB-LABEL_CELLS	● ● ● ● ● ● ●	7
	NB-OUTPUT_CELLS	● ● ● ●	4
	NB-EXTERNAL_REFERENCES	● ● ● ●	4
	INTER_WORKSHEET_CONNECTIONS	● ● ● ● ● ● ● ● ● ● ● ●	12
	NB-CELLS_W_REFERENCES	● ● ●	3
	FAN-IN	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	16
	NB-TRANS_PREDECESSORS	● ● ●	3
	CALC_CHAIN_LENGTH	● ● ● ● ● ● ● ● ● ● ● ● ●	13
	NB-RANGES	● ●	2
	DISPERSION_REFERENCES	● ● ● ● ●	5
	NB-REF_CELLS	● ● ● ●	4
	FAN-OUT	● ● ● ● ●	5
Formula Complexity Metrics	NB-OPERATORS	● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	15
	NB-OPERANDS	● ● ● ● ●	5
	FORMULA_LENGTH	● ● ● ● ●	5
	HALSTEAD_METRICS	● ● ●	3
	NB-IF_FUNCTIONS	● ● ●	3
	NESTING_LEVEL_IF	● ●	2
	COMPLEXITY_CONDITIONALS	● ● ● ● ● ● ● ● ● ● ● ●	12
	NESTING_LEVEL	● ● ● ● ● ● ● ●	8
	NB-UNIQUE_FORMULAS	● ● ● ● ● ● ● ● ● ● ● ● ● ●	14
	NB-COMPLEX_FORMULAS	● ●	2
	FUNCTION_USAGE	● ● ● ●	4
Structure and Pattern-based Metrics	NB-DUPLICATED_FORMULAS	● ● ● ● ● ● ● ●	8
	MIDDLE_MAN	● ● ● ● ● ●	6
	STD_DEVIATION	● ● ● ● ● ●	6
	EMPTY_CELL	● ● ● ● ●	5
	PATTERN_FINDER	● ● ● ● ● ●	6
	STRING_DISTANCE	● ● ● ● ●	5
	QFD	● ● ● ● ●	5
	GROUP_BASED_METRICS	● ●	2
	CLUSTER_BASED_METRICS	● ● ●	3
	NB-EMPTY_ROWS_COLUMNS	●	1
	CELL_WS_POSITION	●	1
Environment Functionality-based Metrics	VBA_USAGE	● ● ● ●	4
	DATA_VALIDATION	● ●	2
	FORMATTING_AND_CHARTS	● ● ●	3
	PROTECTION	●	1
	SOLVER_USAGE	●	1

crosoft Excel by default create three empty worksheets when creating a new spreadsheet, measuring the number of all workbooks vs. only the non-empty worksheets could yield considerably different results.

Counting of cells. While the terms *non-empty cells* (Jansen and Hermans, 2015; Jansen, 2015; Hermans and Murphy-Hill, 2015), *occupied cells* (Ballinger et al., 2003) and *non-empty or referenced blank cells* (Koch et al., 2019) are unambiguous, the terms *(used) cells* (Shubbak and Thorne, 2016; Cunha et al., 2012a; Koch et al., 2016; Hermans et al., 2011, 2012a; Hodnigg and Pinzger, 2015; Correia and Ferreira, 2011) and *cell count* (Birch et al., 2014) leave room for interpretation regarding whether or not they include empty cells that are referenced. Unfortunately, the papers do not provide any formal definitions to resolve this ambiguity.

Definition of identical formulas. In order to count the number of unique formulas, researchers have to define what they count as identical formulas. Jansen and Hermans (2015), Jansen (2015) define identical formulas using the R1C1 notation. Correia and Ferreira (2011) identify identical formulas as groups of copy-equivalent duplicates that are created via drag&drop. Reschenhofer et al. (2017) identify identical formulas by means of the AST (abstract syntax tree), where similar formulas have the same AST but different references. Reschenhofer et al.'s definition classifies more cells as identical than Jansen and Hermans' definition, which classifies more cells as identical than Correia and Ferreira's definition.

Definitions of input cells. Fisher and Rothermel (2005) count as input cells non-empty cells that do not contain formulas. Therefore, this definition counts label cells as input cells. Ballinger et al. (2003) and Hodnigg and Mittermeir (2008) measure the number of *root cells* / *data sources*, i.e. they count all cells that are referenced. Hodnigg and Pinzger (2015) define input cells as all cells that are referenced but do not reference other cells. This

definition includes formula cells that do not reference other cells as well as blank cells that are referenced. Correia and Ferreira (2011) measure the number of *data source cells* (non-blank, non-formula, referenced cells). Cunha et al. (2012a,b), Koch et al. (2016), and Correia and Ferreira (2011) additionally indicate the *number of blank cells referenced in formulas*. Fig. 6 illustrates the focus of the different authors.

Definition of operators. Kokol (1988) counts as operators arithmetic and logical operators, function calls to built-in functions, user defined function calls, the range operator and the symbols (), .., . . . , :, ., @. Thorne et al. (2008) count as operators arithmetic and logical operators and function calls to built-in functions. Bregar (2008) count as operators all function names and special symbols (e.g., +, -). Koch et al. (2019) count as operators binary operators ($+, -, *, /, \&, =, >=, >, <, <, <>, \hat{}$), unary operators ($+, -$), function calls to built-in functions, parenthesis pairs, and the percent operator.

Definition of operands. Kokol (1988) measures the *total occurrences of operands* by counting cell references, constants, names of external spreadsheets, literals and the addresses of the first, second and last cell in a range as well as the *number of unique operands*. Thorne et al. (2008) count cell references and numbers as operands, but do not mention how ranges are handled. Bregar (2008) counts the number of constants and references. Hodnigg and Pinzger (2015) measure the number of *parameters*, which could be constants and references. Both papers do not mention how ranges should be counted. Koch et al. (2019) count the number of constants as well as the number of operands. They define operands as constants and references to ranges, individual cells and names.

Definition of conditional complexity. Bregar (2008) counts simple conditions (conjunction and disjunction). Reschenhofer et al. (2017) count Boolean expressions in a formula, e.g., IF, COUNTIF, SUMIF. Koch et al. (2019) count AND and OR predicates (*decision count*) as well as conditional operators (AND, OR, XOR, NOT, IF, SWITCH, IFS, IFERROR, IFNA, COUNTIF, COUNTIFS, SUMIF, SUMIFS). Other researchers count the number of conditional operations (Abreu et al., 2014a,b; Hermans et al., 2012b, 2015; Jansen and Hermans, 2015), the number of conditional decisions (Hodnigg and Mittermeir, 2008) and the number of binary decisions (Correia and Ferreira, 2011).

Calculation chain length and transitive predecessors. For these metrics, we observe different definitions as well. Whereas Jansen (2015) does not count proxy cells, other authors (Hodnigg and Mittermeir, 2008; Bregar, 2008; Koch et al., 2019; Hermans et al., 2012b, 2015; Jansen and Hermans, 2015; Koch et al., 2018; Koch et al., 2019; Correia and Ferreira, 2011; Abreu et al., 2014a,b; Hermans et al., 2012c) count all directly and indirectly preceding cells.

Dispersion of references. Some authors (Bregar, 2008; Koch et al., 2019) compute the dispersion of references only for references that are located in the same worksheet. But others (Hodnigg and Mittermeir, 2008; Reschenhofer et al., 2017; Hermans et al., 2012c) handle also references to different worksheets, which make the resulting metric values incomparable.

4.3. RQ3: How do researchers evaluate their metrics?

To structure the answer of this research question, we first discuss the extent to which researchers evaluate the metrics they propose to use. Then we review the basis for such evaluations, and finally discuss evaluation procedures and results.

Table 3
Degree of evaluation in the investigated papers.

	None	Metrics applied	Metrics evaluated
Spreadsheet complexity	3	6	2
Corpus size	0	7	0
Fault Prediction	1	0	3
Code Smells And Refactoring	0	2	9
Risk Assessment and Auditing	1	2	1
Sum	5	17	15

4.3.1. Degree of evaluation

Table 3 provides insights about the extent of the evaluation in the literature. For five papers (Bregar, 2008; Hodnigg and Mittermeir, 2008; Hodnigg and Pinzger, 2015; Butler, 2008; Abreu et al., 2014a), no evaluation was performed. In 17 papers (Reschenhofer et al., 2017; Zhang et al., 2017; Jansen, 2015; Correia and Ferreira, 2011; Birch et al., 2014; Kokol, 1988; Koch et al., 2016; Fisher and Rothermel, 2005; Hermans and Murphy-Hill, 2015; Shubbak and Thorne, 2016; Hermans et al., 2011; Ballinger et al., 2003; Butler, 2000; Thorne et al., 2008; Jannach et al., 2019; Jansen and Hermans, 2017; Koch et al., 2019), metrics were applied but not evaluated. In particular metrics that describe the size of corpora and spreadsheets were often used without an evaluation of their effectiveness. In 15 papers (Jansen and Hermans, 2015; Hermans et al., 2015, 2012a,b; Colver, 2011; Hermans et al., 2012c; Cunha et al., 2012,b,a; Koch et al., 2018; Koch et al., 2019; Abreu et al., 2014b; Dou et al., 2017; Cheung et al., 2016; Dou et al., 2014), the usefulness of the metrics was systematically evaluated. This holds in particular for smells and for metrics that are used for fault prediction purposes.

Overall, the fact that in half of the papers no evaluation or no systematic assessment of the metrics was done suggests that more research is required to understand if the used metrics are truly appropriate and effective in their context of use.

4.3.2. Evaluation basis

In Table 4 we show the different types of spreadsheets and corpora used for applying or evaluating the presented metrics. Many research papers used more than one spreadsheet source, e.g., several different spreadsheet corpora, or one public corpus in combination with individual and private financial spreadsheets. In 19 papers (Reschenhofer et al., 2017; Hermans et al., 2015, 2012a,b; Jansen, 2015; Correia and Ferreira, 2011; Hermans et al., 2012c; Cunha et al., 2012; Koch et al., 2016; Fisher and Rothermel, 2005; Cunha et al., 2012b,a; Hermans and Murphy-Hill, 2015; Shubbak and Thorne, 2016; Koch et al., 2019; Koch et al., 2019; Dou et al., 2017; Cheung et al., 2016; Dou et al., 2014), the metrics were applied on the EUSES corpus or a subset of it. 13 papers (Zhang et al., 2017; Jansen and Hermans, 2015; Hermans et al., 2015, 2012a,b; Colver, 2011; Birch et al., 2014; Kokol, 1988; Hermans et al., 2011; Ballinger et al., 2003; Butler, 2000; Thorne et al., 2008; Jansen and Hermans, 2017) used non-public spreadsheets that are, for example, real-world business spreadsheets provided by the spreadsheet creators during a user study, or spreadsheets created for the purpose of a user study. In eight papers (Reschenhofer et al., 2017; Zhang et al., 2017; Jansen, 2015; Koch et al., 2016; Hermans and Murphy-Hill, 2015; Jannach et al., 2019; Koch et al., 2018; Koch et al., 2019), the spreadsheets from the ENRON corpus were analyzed.

4.3.3. Evaluation procedure, evaluated metrics, and results

In this section, we provide more details about papers where the used metrics were systematically evaluated. We structure this discussion distinguishing two types of empirical or experimental

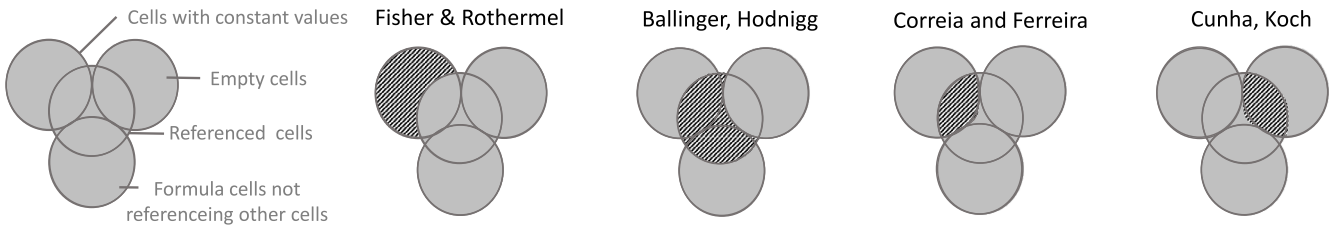

Fig. 6. Different counting of input cells.

Table 4
Frequency of usage of corpora/spreadsheets for applying and/or evaluating metrics.

Spreadsheets/Corpus	Usage
EUSES corpus (Fisher and Rothermel, 2005)	19
Non-public spreadsheets	13
Enron corpus (Hermans and Murphy-Hill, 2015)	8
Info1 corpus (Getzner et al., 2017)	2
Hawaii Kooker corpus (Aurigemma and Panko, 2010)	1

studies: *(i)* studies where user-perceived quality levels were compared with metric values or by manual annotations from human evaluators, and *(ii)* studies where the effectiveness of the metrics was judged indirectly, e.g., by the resulting fault prediction performance.

Studies considering user-perceived quality levels: Hermans et al. (2012c) evaluated to what extent certain complexity and placement metrics reflect the degree of spreadsheet understandability. In their evaluation, 40 professional spreadsheet users had to explain 15 formulas of selected spreadsheets from the EUSES corpus. Hermans et al. then compared the users' performance in terms of the perceived understandability, the observed understandability (i.e., the explanation of formulas), and the listing of direct and indirect references (all on a 4-point scale) with the metrics using the Spearman correlation. In terms of the results, the authors found a correlation between the perceived and observed understandability for the following metrics: the presence of conditional operations in a formula, the depth of the formula's parse tree, and the number of ranges in which the references are grouped. Furthermore, they identified a strong correlation of *Calculation Chain Lengths* and the participants' ability to list all references. The percentage of *Reverse References* and the percentage of *References in the Same Column* were also correlated with the perceived and observed understandability. In their study, the authors did, however, not find a correlation between the *Number of Direct References*, percentage of *References in the Same Row* and the percentage of *Distant References* and the users' performance.

Cunha et al. (2012a) computed metrics to automatically assess the quality of 30 spreadsheets from the EUSES corpus. The measurements obtained from the metrics were automatically mapped to a 0–5 star rating based on predetermined thresholds. Afterwards, Cunha et al. manually inspected the same spreadsheets, assigned a 0–5 star rating, and compared their ratings with the automatically created ones. The authors concluded that the used metrics were well suited to automatically assess the quality of spreadsheets. Only the characteristics *understandability* and *attractiveness* could not be properly assessed with their metrics.

Colver (2011) investigated whether certain size metrics (file size, number of worksheets, unique formulas, operators, function calls) are correlated with the auditing effort (measured in hours). He used 75 financial spreadsheets in his evaluation and could not identify a correlation between the effort and any of the metrics.

Hermans et al. quantitatively and qualitatively evaluated the inter-worksheet smells *Inappropriate Intimacy*, *Feature Envy*, *Middle Man*, and *Shotgun Surgery* (Hermans et al., 2012a) and the formula smells *Multiple Operations*, *Multiple References*, *Conditional Complexity*, *Long Calculation Chain* and *Duplicated Formulas* (Hermans et al., 2012b, 2015). They used the EUSES spreadsheet corpus and 10 real-world financial spreadsheets as a basis for their evaluation. The quantitative evaluation was used to determine how frequently the smells occur. The qualitative evaluation in form of semi-structured interviews was used to determine which smells have an impact on the spreadsheet understandability. The study participants confirmed that the inter-worksheet smells are a threat to the spreadsheet's quality and that refactoring the spreadsheet would help to increase its understandability and maintainability. The study participants agreed that formulas with multiple operations or multiple references are difficult to understand and error prone, but they did not perceive duplicated formulas and long calculation chains as a threat to the spreadsheet's quality. The conditional complexity smell did not occur often, but the study participants were aware of the risks that come with this smell.

Jansen and Hermans (2015) compared 54 pairs of financial spreadsheets w.r.t. the smell metrics *Conditional Complexity*, *Duplicated Formulas*, *Multiple Operations*, *Multiple References*, and *Long Calculation Chain*. Each spreadsheet pair consists of an original spreadsheet and an entirely redesigned spreadsheet that has the same functionality as the original one. The redesigned spreadsheets were created by financial consultants who were trained on spreadsheet design. Jansen and Hermans found out that the spreadsheets written by the consultants were less smelly. The occurrence of the smells *Multiple Operations*, *Multiple References*, *Conditional Complexity*, and *Duplicated Formulas* were reduced by 30%, 64%, 57%, and 93%. However, the *Long Calculation Chain* smell occurred 8% more often. Jansen and Hermans also noticed that the *Number of Inter-Worksheet Connections* dramatically increased by 239%. The measurements for the *Number of Non-Empty Cells*, *Worksheets*, *Formulas*, and *Unique Formulas* increased by 2%, 44%, 112%, and 43% respectively, but the measurements for *Formula Length*, the *Path Depth*, and the *Number of Transitive Predecessors* decreased by 12%, 18%, and 5%.

Cunha et al. (2012,b) evaluated their metric catalog with a subset of 180 randomly selected spreadsheets from the EUSES corpus. They manually inspected the automatically detected smells and classified them into one of the following four categories: no, low, medium, or high smell. As a result, they found that 78% of the automatically detected smells were not categorized as smells when manually inspected. This means that the overall precision is 22%. The precision of the metrics *Quasi-Functional Dependencies* and the *References to empty cells* is significantly higher with 41% respectively 52%. The precision values of the *String distance* and the *Standard deviation* smells are the lowest ones with 7% respectively 8%. The *Empty cell* and *Pattern finder* smells have a precision of 30% and 13%.

Studies judging quality indirectly: Abreu et al. (2014b) computed inter-worksheet, formula and pattern matching smells for the Hawaii Kooker Corpus, which consists of 73 similar spreadsheets, and manually investigated which smells actually pointed to faulty cells. The *Duplicated Formulas*, the *Multiple References* and the *Middle Man* are the most often detected smells with more than 300 detections per smell and they have a precision of 79%, 92%, and 64%. In addition, the *Long Calculation Chain* and the *Quasi-Functional-Dependencies* smells also have high precision with 87% and 82%. However, smells like *Inappropriate Intimacy*, *Pattern Finder*, *Conditional Complexity*, and *Multiple Operations* have a low precision with less than 36%. Moreover, six out of 15 smells suggested in the paper were unable to identify any issues in the evaluation spreadsheets. Among them the *Shotgun Surgery* and the *String Distance Smells* reported 1 and 21 false positives, respectively. The remaining four smells – the *Feature Envy*, the *Reference to Empty Cells*, the *Empty Cell*, and the *Standard Deviation* – were never detected in the corpus. Hence, it seems that some of the metrics are not very useful for smell detection. Furthermore, Abreu et al. showed that the combination of the best performing smells with spectrum-based fault localization had a precision of 66% and a recall of more than 70%. However, the evaluation was only based on a rather small data set comprising similar spreadsheets and results may not be generalizable.

Koch et al. (2018) evaluated the fault detection capabilities of 19 individual smells as well as of their combination in a machine learning approach using the Enron Error and the Info1 corpora. These corpora contain 26 and 119 spreadsheets with known faulty formulas. Through their study, they found that the individual smells lead to low precision (<20%) and recall (<40%). The best machine learning approach that combined the metrics finally had a precision of 30% and a recall of 95%.

In a follow-up work, Koch et al. (2019) extend their work by using not only smells but also other metrics. In total, they used a catalog of 64 metrics. Furthermore, they extended their evaluation to three corpora: the Enron Error corpus, the Info1 corpus and a modified version of the EUSES corpus. The precision of the individual metrics was again low, with 35% as highest value, but the recall was often above 80%. However, using the metrics as features in Random Forests resulted in an average F1 score of 79%, which suggests that the combination of metrics in a machine learning approach is suited for detecting faults.

The next three approaches AMCheck (Dou et al., 2014), CACheck (Dou et al., 2017), and CUSTODES (Cheung et al., 2016) represent cluster-based methods, see Section 3.4.5 for details. The AMCheck approach was evaluated in two steps. First, AMCheck was applied on the spreadsheets from the EUSES corpus. Only 40% of the spreadsheets in the EUSES corpus contain formulas and only 61% of the spreadsheets with formulas contain cell arrays, where AMCheck could be used. From the spreadsheets with cell arrays, 45% contain smells. 700 of the detected smells were manually investigated and 46% were classified as smells by the examiners. Second, ten real-life finance spreadsheets were analyzed using AMCheck and three officers were interviewed on smells that the tool has found in their spreadsheets. The obtained results indicate that the identified smells were often caused by copying and pasting formulas from other spreadsheets used in the past (Dou et al., 2014).

CACheck was evaluated on spreadsheets from the EUSES and Enron corpora. In a first step, the authors created a ground truth using 50 randomly selected spreadsheets from EUSES by manually identifying well-formed and smelly cell arrays. Then, CACheck and other fault localization approaches were applied on these spreadsheets. CACheck has a similar precision as AMCheck (90%), but a higher recall (90% vs. 72%). Afterwards, the authors ran AMCheck on the Enron corpus. From the reported smelly cell arrays, they manually examined 700 and found that 340 of these are true smells (Dou et al., 2017).

CUSTODES was evaluated on a randomly selected subset of 70 spreadsheets from the EUSES corpus. The spreadsheets were manually investigated and clusters and smelly cells were labeled in order to form a ground truth. CUSTODES clusters nearly all cells correctly with a precision of 91% and a recall of 89%. The smell detection capabilities of CUSTODES were compared to those of Microsoft Excel, AMCheck, UCheck and Dimension Inference. CUSTODES reported by far more cells as smelly than the other approaches and has a higher precision (65%) and recall (80%) (Cheung et al., 2016).

From the discussions, we can conclude that almost all papers introducing spreadsheet product metrics are also evaluating them on the basis of certain corpora. A little more than 50% of the papers are making use of well known and open-access corpora like Enron or EUSES for evaluation purposes, which makes the research results comparable and reproducible.

Discussion: Our analyses in the context of RQ3 – how researchers evaluate their metrics – revealed certain research gaps. In particular, we found that in more than half of the analyzed papers researchers did *not* systematically evaluate the metrics they proposed or applied. While certain metrics might generally appear intuitive or might have even been successfully applied for non-spreadsheet software artifacts, our observations call for more research efforts to validate the effectiveness of a number of proposed metrics in the context of spreadsheets.

On a more positive note, we could observe that in particular more recent works increasingly aim to provide empirical evidence for the value of certain spreadsheet metrics in their specific context of use. These empirical works are either based on studies that validate the correspondence of the metrics with human judgments or they are based on indirect measurements such as the fraction of correct fault predictions. In terms of human judgments, some works for example found that certain metrics from the literature correlate well with human perceptions in the context of spreadsheet *understandability*. Other works found that some metrics are perceived to be helpful for the *quality assessment* of spreadsheets. Some works, finally, could not establish a correlation between certain spreadsheet metrics and human perceptions (e.g., in terms of the *auditing effort*). Note that the publication of such non-supported hypotheses is important as well to advance our understanding of spreadsheet metrics. In terms of indirect assessments of spreadsheet metrics, we found that metrics can effectively be applied in the context of fault prediction, in particular when various metrics are combined in a machine learning approach.

Finally, considering methodological issues, we sometimes observed issues related to the replicability of existing studies. In user studies, for example, sometimes exact definitions of the examined constructs (e.g., *quality* or *maintainability*) are missing. In indirect quality assessments, on the other hand, researchers sometimes do not share the code and the data that was used in their computational experiments. In particular the first aspect, not sharing the code, may hamper progress in the field, because the analyses in our paper showed that it is not uncommon that different interpretations of certain metrics exist in the literature.

5. Summary and outlook

We presented the results of a systematic literature survey on spreadsheet product metrics proposed in the literature. Specifically, we discussed which metrics were proposed for which purpose, we identified inconsistencies in the literature, and we investigated to what extent researchers have validated the appropriateness and value of the used metrics. Our survey showed that

metrics are mainly used for complexity assessment, code smells, and the refactoring of spreadsheets. We hope that the catalog of metrics provided in this work will be used by practitioners and researchers alike, and serve as a central reference for spreadsheet product metrics. Practitioners can search for suitable metrics to be used to support spreadsheet quality assurance measures, and researchers can refer to the metrics of the catalog when evaluating new techniques, e.g., complexity assessments of spreadsheets or for fault detection.

From the analyses above, we derive the following future research directions. First, given the identified inconsistencies, any used metric should be formally defined, and it would also be helpful if public software tools or libraries were available to compute the metrics for a given spreadsheet. Second, there are a number of metrics for which there is no evidence yet that they fulfill their purpose, i.e., they have not been formally evaluated so far. Third, it seems promising to look for other metrics from software engineering, which were not applied yet for the domain of spreadsheets. Fourth, another direction of future research is to look at *process metrics* and other types of metrics, e.g., for certain extensions to the spreadsheet paradigm, which were beyond the scope of our current work. Fifth, it would be worth investigating if there are correlations between some of the metrics, which is another aspect which was not in the scope of our present work. This aspect is however particularly interesting, for example because there exists a mathematical relation between the derived Halstead metrics and their base metrics. In addition, there are other metrics that are interrelated. Shubbak and Thorne (2016), for example, detected a positive correlation between the number of cells and the number of formulas for 690 investigated financial spreadsheets. However, to the best of our knowledge, no detailed studies on such correlations between metrics exist in the literature.

CRediT authorship contribution statement

Birgit Hofer: Conceptualization, Resources, Writing - original draft preparation. **Dietmar Jannach:** Supervision, Conceptualization, Writing - reviewing and editing. **Patrick Koch:** Investigation, Resources, Writing - original draft preparation. **Konstantin Schekotihin:** Validation, Writing - reviewing and editing. **Franz Wotawa:** Supervision, Conceptualization, Writing - reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The work described in this paper has been funded by the Austrian Science Fund (FWF) project Interactive Spreadsheet Debugging under contract number P 32445.

Appendix. Paper retrieval process

We scanned the digital libraries of ACM, IEEE Xplore, Springer, Elsevier, and ArXiv for papers that include the term ‘spreadsheet’ and at least one of the following terms: ‘metric’, ‘measure’, or ‘smell’. Since the search functionality of the individual libraries are different, we list the exact search strings for each of them.

Library:	**ACM Digital Library**
URL:	https://dl.acm.org/advsearch.cfm?coll=DL&dl=ACM
Search date:	October, 24th, 2019
Search string:	`recordAbstract:(+spreadsheet) AND (metric measure smell)`
Number of results:	60 (after removing duplicates)

Library:	**ArXiv**
URL:	https://arxiv.org/search/advanced
Search date:	October, 24th, 2019
Search string:	`Query: order: -announced_date_first; size: 50; include_cross_list: True; terms: AND abstract=Spreadsheet; AND all=Metric OR Measure OR Smell`
Number of results:	33

Library:	**Elsevier Digital Library**
URL:	https://www.sciencedirect.com/search/advanced
Search date:	October, 24th, 2019
Search string:	`spreadsheet AND (metric OR measure OR smell)`
Number of results:	240

Library:	**IEEE Xplore**
URL:	https://ieeexplore.ieee.org/search/advanced/command
Search date:	October, 24th, 2019
Search string:	`((‘‘All Metadata’’:metric OR measure OR smell) AND ‘‘Abstract’’:spreadsheet)`
Number of results:	118 (after removing duplicates)

Library:	**Springer Link**
URL:	https://link.springer.com/advanced-search
Search date:	October, 24th, 2019
Search string:	with at least one of the words: `metric measure smell`, title contains: `spreadsheet`
Number of results:	114

References

Abraham, R., Erwig, M., 2007a. GoalDebug: A Spreadsheet Debugger for end users. In: 29th International Conference on Software Engineering (ICSE'07), pp. 251–260.

Abraham, R., Erwig, M., 2007b. UCheck: A spreadsheet type checker for end users. J. Vis. Lang. Comput. 18 (1), 71–95.

Abreu, R., Cunha, J., Fernandes, J.P., Martins, P., Perez, A., Saraiva, J., 2014a. Faultysheet detective: When smells meet fault localization. In: Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 625–628.

Abreu, R., Cunha, J., Fernandes, J.P., Martins, P., Perez, A., Saraiva, J., 2014b. Smelling faults in spreadsheets. In: Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 111–120.

Asavametha, A., 2013. Detecting Bad Smells in Spreadsheets (Master's thesis). Oregon State University, URL https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/m613n1906.
Aurigemma, S., Panko, R.R., 2010. The detection of human spreadsheet errors by humans versus inspection (auditing) software. CoRR arXiv:1009.2785.
Ballinger, D., Biddle, R., Noble, J., 2003. Spreadsheet visualisation to improve end-user understanding. In: Australasian Symposium on Information Visualisation, vol. 24, pp. 99–109.
Barowy, D.W., Gochev, D., Berger, E.D., 2014. CheckCell: Data debugging for spreadsheets. SIGPLAN Not. 49 (10), 507–523.
Birch, D., Liang, H., Kelly, P.H.J., Mullineux, G., Field, T., Ko, J., Simondetti, A., 2014. Multidisciplinary engineering models: Methodology and case study in spreadsheet analytics. In: Proceedings of the EuSpRIG Conference. arXiv:1401.4582.
Bregar, A., 2008. Complexity metrics for spreadsheet models. CoRR arXiv:0802.3895.
Butler, R.J., 2000. Is this spreadsheet a tax evader? How H.M. customs and excise test spreadsheet applications. In: Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS), pp. 1–6.
Butler, R.J., 2008. Risk assessment for spreadsheet developments: Choosing which models to audit. CoRR arXiv:0805.4236.
Chambers, C., Erwig, M., 2009. Automatic detection of dimension errors in spreadsheets. J. Vis. Lang. Comput. 20 (4), 269–283.
Cheung, S.-C., Chen, W., Liu, Y., Xu, C., 2016. CUSTODES: Automatic spreadsheet cell clustering and smell detection using strong and weak features. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 464–475.
Colver, D., 2011. Drivers of the cost of spreadsheet audit. CoRR arXiv:1111.5002.
Correia, J.P., Ferreira, M.A., 2011. Measuring maintainability of spreadsheets in the wild. In: Proceedings of the IEEE International Conference on Software Maintenance (ICSM), pp. 516–519.
Cunha, J., Fernandes, J.P., Martins, P., Mendes, J., Saraiva, J., 2012. SmellSheet detective: A tool for detecting bad smells in spreadsheets. In: Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 243–244.
Cunha, J., Fernandes, J.P., Mendes, J., Saraiva, J., 2013. Complexity metrics for classsheet models. In: Proceedings of the International Conference on Computational Science and Its Applications (ICCSA), pp. 459–474.
Cunha, J., Fernandes, J.P., Peixoto, C., Saraiva, J., 2012a. A quality model for spreadsheets. In: Proceedings of the International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 231–236.
Cunha, J., Fernandes, J.P., Ribeiro, H., Saraiva, J., 2012b. Towards a catalog of spreadsheet smells. In: Proceedings of the International Conference on Computational Science and Its Applications (ICCSA), pp. 202–216.
Dou, W., Cheung, S., Wei, J., 2014. Is spreadsheet ambiguity harmful? Detecting and repairing spreadsheet smells due to ambiguous computation, in: Proceedings of the International Conference on Software Engineering (ICSE), 2014, pp. 848–858.
Dou, W., Xu, C., Cheung, S.C., Wei, J., 2017. CACheck: Detecting and repairing cell arrays in spreadsheets. IEEE Trans. Softw. Eng. 43 (3), 226–251.
Finextra, 2020. Excel error leaves Barclays with unwanted Lehman assets. URL https://www.finextra.com/news/fullstory.aspx?newsitemid=19135. Last visited: March, 3rd 2020.
Fisher, M., Rothermel, G., 2005. The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. ACM SIGSOFT Softw. Eng. Notes 30 (4), 1–5.
Getzner, E., Hofer, B., Wotawa, F., 2017. Improving spectrum-based fault localization for spreadsheet debugging. In: Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 102–113.
Halstead, M.H., 1977. Elements of Software Science (Operating and Programming Systems Series). Elsevier.
Hermans, F., Murphy-Hill, E., 2015. Enron's spreadsheets and related emails: A dataset and analysis. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 7–16.
Hermans, F., Pinzger, M., van Deursen, A., 2011. Supporting professional spreadsheet users by generating leveled dataflow diagrams. In: Taylor, R.N., Gall, H.C., Medvidovic, N. (Eds.), Proceedings of the International Conference on Software Engineering (ICSE). pp. 451–460.
Hermans, F., Pinzger, M., van Deursen, A., 2012a. Detecting and visualizing inter-worksheet smells in spreadsheets. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 441–451.
Hermans, F., Pinzger, M., van Deursen, A., 2012b. Detecting code smells in spreadsheet formulas. In: Proceedings of the IEEE International Conference on Software Maintenance (ICSM), pp. 409–418.
Hermans, F., Pinzger, M., van Deursen, A., 2012c. Measuring spreadsheet formula understandability. CoRR arXiv:1209.3517.
Hermans, F., Pinzger, M., van Deursen, A., 2015. Detecting and refactoring code smells in spreadsheet formulas. Empir. Softw. Eng. 20 (2), 549–575.
Hermans, F., Sedee, B., Pinzger, M., van Deursen, A., 2013. Data clone detection and visualization in spreadsheets. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 292–301.
Hodnigg, K., Mittermeir, R.T., 2008. Metrics-based spreadsheet visualization: Support for focused maintenance. CoRR arXiv:0809.3009.
Hodnigg, K., Pinzger, M., 2015. XVIZIT: Visualizing cognitive units in spreadsheets. In: Proceedings of the IEEE Working Conference on Software Visualization (VISSOFT), pp. 210–214.
IEEE, 1993. IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1992, Institute of Electrical and Electronics Engineers, Inc.
Irons, R.J., 2008. The wall and the ball: A study of domain referent spreadsheet errors. CoRR arXiv:0804.0943.
Jannach, D., Schmitz, T., Hofer, B., Schekotihin, K., Koch, P.W., Wotawa, F., 2019. Fragment-based spreadsheet debugging. Automat. Softw. Eng. 26 (1), 203–239.
Jannach, D., Schmitz, T., Hofer, B., Wotawa, F., 2014. Avoiding, finding and fixing spreadsheet errors - A survey of automated approaches for spreadsheet QA. J. Syst. Softw. 94, 129–150.
Jansen, B., 2015. Enron versus EUSES: A comparison of two spreadsheet corpora. In: Proceedings of the Second Workshop on Software Engineering Methods in Spreadsheets, pp. 41–47.
Jansen, B., Hermans, F., 2015. Code smells in spreadsheet formulas revisited on an industrial dataset. In: Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 372–380.
Jansen, B., Hermans, F., 2017. The effect of delocalized plans on spreadsheet comprehension: a controlled experiment. In: Proceedings of the IEEE/ACM International Conference on Program Comprehension (ICPC), pp. 286–296.
Janvrin, D., Morrison, J., 2000. Using a structured design approach to reduce risks in end user spreadsheet development. Inf. Manage. 37 (1), 1–12.
Koch, P.W., Hofer, B., Wotawa, F., 2016. Static spreadsheet analysis. In: 7th IEEE International Workshop on Program Debugging (IWPD), ISSRE Workshop Proceedings, pp. 167–174.
Koch, P.W., Hofer, B., Wotawa, F., 2019. On the refinement of spreadsheet smells by means of structure information. J. Syst. Softw. 147, 64–85.
Koch, P., Schekotihin, K., Jannach, D., Hofer, B., Wotawa, F., 2019. Metric-based fault prediction for spreadsheets. IEEE Trans. Softw. Eng. Online http://dx.doi.org/10.1109/TSE.2019.2944604.
Koch, P., Schekotihin, K., Jannach, D., Hofer, B., Wotawa, F., Schmitz, T., 2018. Combining spreadsheet smells for improved fault prediction. In: Proceedings of the International Conference on Software Engineering: New Ideas and Emerging Results (ICSE NEAR), pp. 25–28.
Kokol, P., 1988. Spreadsheet language level: how high is it? SIGPLAN Not. 23 (6), 121–134.
Lee, J., 2020. Spreadsheet horror stories that will make you re-think your receivables management strategy. URL http://blog.anytimecollect.com/5-receivables-management-spreadsheet-horror-stories/, last visited: March, 3rd 2020.
McConnell, S., 2004. Code Complete, second ed. Microsoft Press.
McKeever, R., McDaid, K., 2010. How do range names hinder novice spreadsheet debugging performance? CoRR arXiv:1009.2765.
Panko, R.R., 2008. Thinking is bad: Implications of human error research for spreadsheet research and practice. CoRR arXiv:0801.3114.
Radjenovic, D., Hericko, M., Torkar, R., Zivkovic, A., 2013. Software fault prediction metrics: A systematic literature review. Inf. Softw. Technol. 55 (8), 1397–1418.
Reschenhofer, T., Waltl, B., Shumaiev, K., Matthes, F., 2017. A conceptual model for measuring the complexity of spreadsheets. CoRR arXiv:1704.01147.
Shubbak, M.H., Thorne, S., 2016. Development and experimentation of a software tool for identifying high risk spreadsheets for auditing. In: Proceedings of the EuSpRIG Conference. pp. 47–78, arXiv:1602.05231.
Thorne, S.R., Ball, D., Lawson, Z., 2008. A novel approach to formulae production and overconfidence measurement to reduce risk in spreadsheet modelling. CoRR arXiv:0803.1754.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., 2012. Experimentation in Software Engineering. Springer.
Zhang, J., Han, S., Hao, D., Zhang, L., Zhang, D., 2017. Automated refactoring of nested-IF formulae in spreadsheets. CoRR arXiv:1712.09797.

Birgit Hofer works as researcher at Graz University of Technology. She received a Ph.D. degree in Computer Science (2013) and a Master's degree (2009) from the same University. Her main research interests are automatic fault localization and correction in imperative and object-oriented software and spreadsheets, with a particular focus on spectrum-based fault localization, model-based debugging, and genetic programming approaches.

Dietmar Jannach is a full professor of Information Systems at the University of Klagenfurt, Austria. His main research area lies in the application of artificial intelligence technology to practical problems, with a particular focus on recommender systems in e-commerce, AI-based testing and debugging of spreadsheets, and on engineering problems of knowledge-intensive systems.

Patrick Koch obtained his Ph.D. from the University of Klagenfurt in 2019. He was a scientific project assistant at the University of Klagenfurt and afterwards at Graz University of Technology. He received his Master's Degree in Software Development and Business Management from Graz University of Technology, Austria, in 2016. His research is focused on static analysis and AI-based techniques for debugging and quality assurance of spreadsheets.

Konstantin Schekotihin is an associate professor of Intelligent Systems at the University of Klagenfurt, Austria. His research focus lies mainly on various aspects of knowledge representation and reasoning systemsincluding machine learning, knowledge acquisition and maintenance, reasoning techniques as well as their applications in, for instance, configuration, planning, and recommendation.

Franz Wotawa received an M.Sc. and a Ph.D. from Vienna University of Technology in 1994 and 1996 respectively. He is a full professor of software engineering at Graz University of Technology. His research interests include model-based and qualitative reasoning, theorem proving, mobile robots, verification and validation, and software testing and debugging.