SmellSheet Detective: A Tool for Detecting Bad Smells in Spreadsheets

Jácome Cunha[†*], João Paulo Fernandes[†], Pedro Martins[†], Jorge Mendes[†], João Saraiva[†]
[*]Escola de Tecnologia e Gestão – Instituto Politécnico do Porto, Portugal
[†]HASLab / INESC TEC & Universidade do Minho, Portugal
{jacome,jpaulo,prmartins,jorgemendes,jas}@di.uminho.pt

***Abstract*—This tool demo paper presents *SmellSheet Detective*: a tool for automatically detecting bad smells in spreadsheets. We have defined a catalog of bad smells in spreadsheet data which was fully implemented in a reusable library for the manipulation of spreadsheets. This library is the building block of the *SmellSheet Detective* tool, that has been used to detect smells in large, real-world spreadsheets within the EUSES corpus, in order to validate and evolve our bad smells catalog.**

I. Introduction

Spreadsheets play a crucial role in modern society. They are inherently multi-purpose and widely used both by individuals with simple needs as well as by large companies as integrators of complex systems and as support for business decisions. In fact, it is estimated that 95% of all U.S. firms use them for financial reporting, that 90% of all analysts in industry perform calculations in them and that 50% of all spreadsheets are the basis for decisions. Effective mechanisms for error prevention, however, did not grow proportionally: up to 94% of real-world spreadsheets contain errors, which each year cause losses worth around 10 billion dollars [1]!

In this paper we seek to identify potential errors in spreadsheets in an automated way. We look for spreadsheet *smells*, a concept that was introduced on software by Martin Fowler [2] as a concrete evidence of a bad programming practice. A smell is not necessarily an error, but a characteristic that may cause problems understanding, updating or evolving a software artifact. An example of a software smell is the definition of long methods in an object-oriented program. In the context of spreadsheets, a (bad) smell is, for example, a reference to an empty cell within a spreadsheet formula. An extensive catalog of bad smells for spreadsheets is presented in [3].

To automatically perform the detection of spreadsheets' bad smells we have implemented a library for the analysis of spreadsheets. This library implements our full catalog of smells, that we also review in this paper. The library is reusable and can easily be extended to incorporate further analysis and smells, and it is also the basis for the development of the *SmellSheet Detective* tool. We have used this tool for detecting bad smells in large and real-world spreadsheets, namely by processing the large EUSES spreadsheet corpus to define and evolve our catalog of bad smells. The *SmellSheet Detective* automatically detects smells given a spreadsheet and the first preliminary results produced are promising: 22% of the smells automatically detected by our tool are real bad smells, according to a manual validation that we performed. Here, we present our library for manipulating spreadsheets and the *SmellSheet Detective* tool, which are the main contributions of this tool paper.

This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-010048 and FCOMP-01-0124-FEDER-020484. The four first authors were supported by FCT grants SFRH/BPD/73358/2010, SFRH/BPD/46987/2008, BI3-2011/PTDC/EIA/70271/2006 and BI4-2011PTDC/EIA-CCO/108613/2008, respectively. Part of this work was developed during a visit to Software Improvement Group (SIG), Amsterdam.

II. A Catalog of Spreadsheet Smells

The detection of errors in software systems is an important software engineering research field. Software errors cause programs not to behave as expected and are responsible for several accidents. Errors can have various sources, among them bad programming practices. The presence of bad smells in software code makes programs harder to understand, to maintain, and to evolve.

Despite Martin Fowler introduced the concept of program smells in the context of object-oriented programming, making it an important area of research, it can be applied to multiple other areas, among them spreadsheets. Orthogonal to all contexts, the detection of bad smells allows programmers to improve their solutions by eliminating bad programming practices, improving their readibility and robustness.

In this section we review the catalog of bad smells for spreadsheets introduced in [3]. This catalog was developed using a four steps methodology that we present next. In the first step, *catalog definition*, we used our experience developing and researching spreadsheets to propose an initial catalog of spreadsheets' bad smells. In the second step, *catalog validation*, we considered a large repository, the EUSES corpus [4], that contains more than 5000 spreadsheets, where we detected smells in a representative sub-set: 180 spreadsheets were randomly selected, which cover the different spreadsheet categories defined in this corpus. In the third step, *catalog evaluation*, we evaluated the results of the previous step, by manually inspecting all the identified bad smells. We classified the detected smells in four categories: *not a smell*, *low smell*, *medium smell* and *high smell*. In the fourth and last step, *catalog refinement*, using the results of the evaluation performed in the previous step, we adjusted the catalog by identifying wrong smells, by refining previously defined smells and by adding new smells that showed up when manually inspecting spreadsheets within the EUSES corpus. The result of this step is the catalog of spreadsheets' bad smells that we structured in four categories: *Statistical Smells* (standard deviation smell), *Type Smells* (empty cell and pattern finder smells), *Content Smells* (string distance and reference to empty cells smells) and *Functional Dependencies Based Smells* (Quasi-Functional Dependencies (QFD) smell).

III. The *SmellSheet Detective* tool

In order to automatically analyze sample spreadsheets we implemented *SmellSheet Detective*[1] that builds on our spreadsheets' manipulation library to detect the smells introduced in Section II. This implementation combines the Java programming language, the Google Web Toolkit (GWT), the Apache POI library and the Google libraries to work with spreadsheets within the Google Docs environment. We decided to support spreadsheets written in the Google Docs platform because the migration from desktop to online-based applications is becoming very common, and even the popular Microsoft Office suite has its online version. Nevertheless, we also support spreadsheets stored in local machines written using desktop applications, as can be seen in Figure 1.

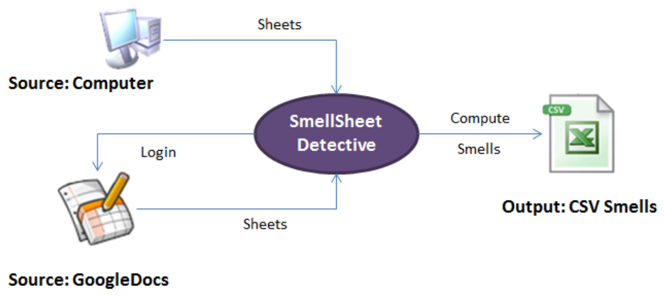

Fig. 1. *SmellSheet Detective* architecture.

When using the Google Docs version of our tool, a valid Google account login is required. Our tool allows the analysis of one spreadsheet at a time, but within it the user can select a single sheet or the full spreadsheet. When using the direct upload mechanism, the user can browse the spreadsheets in the local machine, with the *SmellSheet Detective* producing the results on *csv*, *Excel* and LaTeX tables.

Table I presents the results computed by the tool for the 180 spreadsheets considered when evolving our catalog. Each row represents a smell and each column represents the classification we gave to each smell found by the tool. As we can see, 22% of the smells found are in fact bad smells that deteriorate the quality of the spreadsheet.

The *SmellSheet Detective* was developed on top of a modular and extensible library written in Java. Next, we present the elegant implementation of the method that computes formulas that contain references to empty cells.

[1]Available at http://ssaapp.di.uminho.pt.

Smell \ Level	Low smells	Medium smells	High smells	Not smells	Total
Empty cells	115	4	0	274	393
Patterns	9	5	0	90	101
Std. Dev. cells	21	0	0	234	255
String Dist.	13	2	7	290	312
QFD	64	13	6	121	204
Ref2empty	9	9	8	24	50
Total	231	33	21	1033	1315

TABLE I
Results of running *SmellSheet Detective* in the EUSES corpus.

```
public TreeMap<String,String> ReferenceEmptyCells() {
  TreeMap<String,String> nullRefs =
      new TreeMap<String,String>();
    for(String sheet : spreadsheet.keySet())
      for(String cellRef :
          spreadsheet.get(sheet).keySet())      {
        TreeMap<String, String> oneNull =
          getNull(sheet, cellRef);
        for(String key : oneNull.keySet())
          nullRefs.put(key, oneNull.get(key)); }
    return nullRefs;                                    }
```

Given the extensibility feature of our library, extending it with new smells is very simple: we just need to add a new method implementing the smell. This is very important when developing a catalog, since new smells can easily be considered and added. For example, the smells from [5], [6] can easily be incorporated in our tool, which actually is undergoing work.

IV. Conclusion

In this paper we have presented the *SmellSheet Detective* tool and a Java-based library that implements an extensive catalog of bad smells for spreadsheets. The *SmellSheet Detective* automatically detects bad smells given a spreadsheet as input. We have used it to detect bad smells in a subset of the EUSES corpus with promising first results. The tool was developed using modular and extensible mechanisms, which allow a simple and fast implementation of new smells.

Code smells, as suggested by Fowler, are usually associated with refactorings that eliminate them. With this in mind, we are currently working in defining and implementing a set of domain specific spreadsheet refactorings that can be used to automatically eliminate detected smells.

References

[1] R. Panko, "Facing the problem of spreadsheet errors," *Decision Line, 37(5)*, 2006.
[2] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999.
[3] J. Cunha, J. P. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *12th Int. Conf. on Computational Science and Its Applications*, ser. LNCS, vol. 7336. Springer, 2012, pp. 202–216.
[4] M. F. Ii and G. Rothermel, "The euses spreadsheet corpus: A shared resource for supporting experimentation with spreadsheet dependability mechanisms," in *End-User SW Engineering Workshop*, 2005, pp. 47–51.
[5] S. Badame and D. Dig, "Refactoring meets spreadsheet formulas," in *Proceedings of the 28th IEEE International Conference on Software Maintenance*. IEEE Computer Society, 2012, to appear.
[6] F. Hermans, M. Pinzger, and A. van Deursen, "Detecting and Visualizing Inter-Worksheet Smells in Spreadsheets," in *Proc. of the 34rd International Conference on Software Engineering*. ACM, 2012, pp. 441–451.