Toward Reverse Engineering of VBA Based Excel Spreadsheet Applications

Domenico Amalfitano, Nicola Amatucci, Vincenzo De Simone, Anna Rita Fasolino, Porfirio Tramontana
Department of Electrical Engineering and Information Technologies
University of Naples Federico II
Via Claudio 21, Naples, Italy
Email: domenico.amafitano@unina.it, nicola.amatucci@unina.it, vincenzo.desimone2@unina.it,
anna.fasolino@unina.it, ptramont@unina.it

***Abstract*—Modern spreadsheet systems can be used to implement complex spreadsheet applications including data sheets, customized user forms and executable procedures written in a scripting language. These applications are often developed by practitioners that do not follow any software engineering practice and do not produce any design documentation. Thus, spreadsheet applications may be very difficult to be maintained or restructured. In this position paper we present in a nutshell two reverse engineering techniques and a tool that we are currently realizing for the abstraction of conceptual data models and business logic models.**

I. Introduction

Spreadsheets are interactive software applications for manipulation and storage of data. In a spreadsheet data are organized in worksheets, any of which is represented by a matrix of cells each containing either data or formulas that are automatically calculated at any variation of data. Modern spreadsheet systems (e.g. Microsoft Excel) are integrated with scripting languages (e.g., Microsoft Visual Basic for Applications). An end user can extend the presentation layer of the application with Visual Basic for Application (VBA in the following) by defining new *User Forms* and can develop new business functions by means of *Procedures*. The execution of these procedures is event-driven: they can be attached to events related to any element of the Excel Object Model. Whenever one of these events occurs, the attached event handler code is executed. VBA can also be used to programmatically access and modify the underlying Excel Object Model, i.e. the hierarchy of objects contained in Excel representing all its accessible resources.

Very often developers create complex spreadsheet applications without any documentation at design level making each task related to their maintenance very hard. In order to reconstruct design models of an existing spreadsheet application reverse engineering techniques and tools are needed. The research community has devoted great attention to analysis of formulas in a spreadsheet, while they leave out the analysis of the spreadsheets embedded code and the static and dynamic relationships between the code and the spreadsheet's cells.

In this paper we will present in a nutshell two reverse engineering techniques and a tool supporting the second technique allowing the abstraction of design models of an existing spreadsheet application. In details, in section II we will present a technique for abstracting conceptual data models by analyzing spreadsheets data while in section III we will present a tool supporting the abstraction of views representing the relationships between VBA procedures, user forms and spreadsheet cells. Finally, in section IV some conclusions and future works will be presented.

II. Data Model Reverse Engineering

The first technique presented in this paper regards the abstraction of conceptual data models from the analysis of the structure and the information included in an Excel spreadsheet by means of heuristic rules. This technique is based on heuristic rules automatically applicable on a spreadsheet. By means of these rules, set of candidate classes with attributes, relationships between them and the corresponding cardinalities are abstracted on the basis of the structure and of the properties of spreadsheets and of their components, such as sheets, cells, cell headers, etc.. In particular, the heuristics consider cells labels and data by looking for repeated data, synonyms and group of cells organized as arrays or matrixes.

The considered rules have been extensively presented in [1] and [2] where they have been used with success to abstract the conceptual data model underlying some complex spreadsheet applications used in the automotive context as informative systems. The abstracted data model represented a useful starting point in the process of migration of an existing spreadsheet-based informative system towards a Web application. Some of the rules were derived from works found in literature such as in [3] and [4]. These works are focused on spreadsheets used as calculation sheets. The proposed techniques are based on the analysis of formulas to abstract relationships between candidate classes. Instead, we focused on spreadsheets used as informative systems and proposed further rules based on the analysis of cell properties and values.

Figure 1 shows an example of a possible application of some of the proposed rules. First of all, the sheet of the spreadsheet shown in Figure 1 may be abstracted as a UML class. Moreover, the two distinct rectangular areas separated by a blank column and composed of labels and data may be abstracted as other two classes. Two composition relationships between these two classes and the class representing the sheet may be abstracted, too. The first rows of the two areas have

cells with bold texts and different background colors: they may be abstracted as attributes of the classes representing the areas. The texts of these cells may be abstracted as names of the attributes of the corresponding classes.

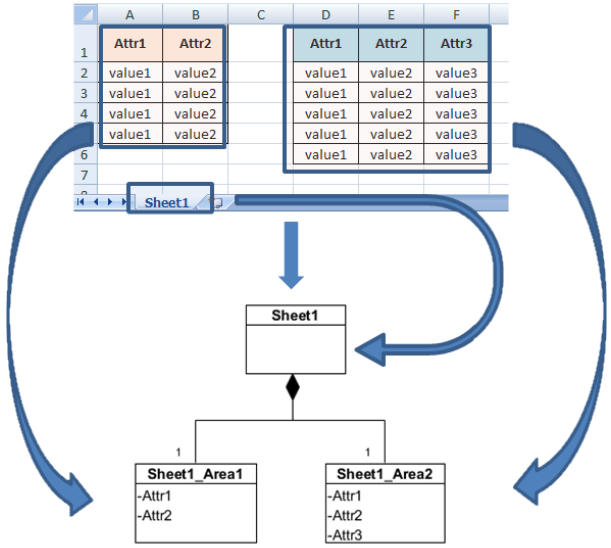

Fig. 1. An example of application of heuristic rules to abstract a conceptual data model of a spreadsheet

III. Business Logic Model Reverse Engineering

The second contribution presented in this position paper is related to a reverse engineering technique that we have realized to abstract models of the business logic of a spreadsheet application by statically analyzing sheets, user forms, VBA procedures and their inter-relationships. The technique is completely supported by an interactive Excel add-in called EXACT (EXcel Application Comprehension Tool) that we have realized and that is available at https://github.com/reverse-unina/EXACT. It provides features for the extraction of information and the abstraction of several views of an existing spreadsheet application.

The tool extracts information from a spreadsheet by exploiting the features offered by the Office Primary Interop Assemblies for the Excel Application that exposed the Microsoft Excel Object Library and by statically analyzing the source VBA code. The tool is able to reconstruct the set of elements composing the spreadsheet including sheets, user forms, event handlers, classes and procedures. Furthermore the tool recovers information related to the relationships between these elements such as procedure calls, relationships between events and the corresponding handlers, and dependencies between procedures and cells.

The EXACT tool provides several features of software visualization. It offers multiple views at different levels of detail and provides cross-referencing functions for switching between these views. For example, the left part of Figure 2 exemplifies the structural view proposed by EXACT for an Excel spreadsheet. The figure shows that the spreadsheet is composed of a single workbook with 4 worksheets, a VBProject including 6 code modules (that implement a total of 10 procedures) and a user form with 12 controls. In the right part of the figure there are some detailed information provided by the EXACT tool about a procedure selected by the user and a graph showing the dependencies between this procedure and the other application components. In particular, the considered procedure writes in 9 different groups of cells, reads the values of a group of cells and calls two procedures.

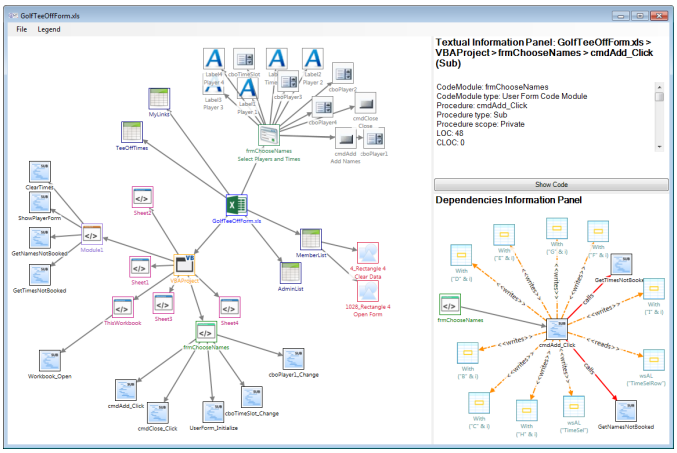

Fig. 2. Structural View of an Excel Spreadsheet

IV. Conclusions and Future Works

In this position paper we have presented in a very concise way two reverse engineering techniques and a tool abstracting conceptual data models and business logic models of Excel spreadsheets, taking into account both the data structure, the VBA code procedures, the User Forms and their inter-relationships. In future work, we plan to extend the EXACT tool by introducing the data model reverse engineering technique proposed in section II with the aim to make possible comprehension processes of Excel spreadsheet applications.

References

[1] D. Amalfitano, A. R. Fasolino, P. Tramontana, V. D. Simone, G. D. Mare, and S. Scala, "Information extraction from legacy spreadsheet-based information system - an experience in the automotive context," in *DATA 2014 - Proceedings of 3rd International Conference on Data Management Technologies and Applications, Vienna, Austria, 29-31 August, 2014*, 2014, pp. 389–398. [Online]. Available: http://dx.doi.org/10.5220/0005139603890398

[2] D. Amalfitano, A. Fasolino, V. Maggio, P. Tramontana, G. Di Mare, F. Ferrara, and S. Scala, "Migrating legacy spreadsheets-based systems to web mvc architecture: An industrial case study," in *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, Feb 2014, pp. 387–390.

[3] R. Abraham and M. Erwig, "Inferring templates from spreadsheets," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 182–191. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134312

[4] F. Hermans, M. Pinzger, and A. van Deursen, "Automatically extracting class diagrams from spreadsheets," in *Proceedings of the 24th European Conference on Object-oriented Programming*, ser. ECOOP'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 52–75. [Online]. Available: http://dl.acm.org/citation.cfm?id=1883978.1883984