Supporting Spreadsheet Maintenance with Dependency Notification

Roy, Sohon; Hermans, Felienne; van Deursen, Arie

Publication date
2017

Document Version
Accepted author manuscript

Published in
IS-EUD 2017 6th International Symposium on End-User Development

Citation (APA)
Roy, S., Hermans, F., & van Deursen, A. (2017). Supporting Spreadsheet Maintenance with Dependency Notification. In J. V. Khan, & I. Soute (Eds.), *IS-EUD 2017 6th International Symposium on End-User Development: Extended Abstracts* (pp. 88-93). Technische Universiteit Eindhoven. https://pure.tue.nl/ws/files/69763287/IS_EUD2017_extended_abstracts.pdf#page=89

Supporting Spreadsheet Maintenance with Dependency Notification

Sohon Roy, Felienne Hermans, and Arie van Deursen

Delft University of Technology, Delft, The Netherlands,
{S.Roy-1, F.F.J.Hermans, Arie.vanDeursen}@tudelft.nl

Abstract. Spreadsheets in the industry are used by multiple employees in organizations, and they remain in use for several years. Maintenance of existing spreadsheets is thus common. One of the issues in maintaining spreadsheets is the fact that formulas create cell dependencies, and these dependencies are invisible to users. To address this, dependence tracing techniques have been developed, both commercially and in research. However, these techniques are effort consuming, and are designed as separate activities that force the users to leave the context of editing spreadsheets. As such, these techniques are not suitably supportive for usual spreadsheet maintenance tasks. In this extended abstract, we present our work in progress on a novel approach for notifying users of cell dependencies, integrated into the context of editing spreadsheets. We present a preliminary evaluation of the approach in the form of an exploratory user-study with seven employees of a financial modeling company. Results show that the approach has the potential to support industrial spreadsheet users in the context of spreadsheet maintenance, as indicated by the responses of six out of seven participants.

Keywords: Spreadsheets, End-user Software Maintenance

1 Introduction

Spreadsheets are popular end-user computing tools, but their use is not without issues. Hermans *et al.* found that spreadsheets can have a long life span, and they are often used by several users in an organization [3]. This makes spreadsheet *maintenance* a difficult task, software maintenance alike. One reason hampering the maintenance of spreadsheets is the fact that the *cell dependency graph*—the network of cells which refer to each other—is not visible directly. Users who are not familiar with the dependency structure of a given spreadsheet might find it difficult to grasp the entire structure of the spreadsheet. This can even result in a fear of further modification of the spreadsheet [2].

The problem of analyzing dependencies within spreadsheets is by no means new. Both in research and in practice, tools have been developed to support spreadsheet users in the activity of *dependence tracing.* We refer to our previous work [7] for a study of such tools [4, 8, 1, 3]. However, one important issue with existing tools is that they aim to support dependence tracing as a separate task; meaning that the user, while making changes to a formula, has to click buttons or use hot keys. As such, these tools support activities like auditing a spreadsheet, in which the users wants to deeply understand the dependency graph of a spreadsheet, but they do not support a user in making simple changes to formulas.

In summary, existing tools and techniques are not designed with the focus on supporting spreadsheet maintenance. Therefore, currently, the maintenance of *legacy* spreadsheets is not supported adequately, with users having to use dependence tracing tools which pose extra effort, and also force them to leave the context of spreadsheet editing. This latter deserves particular attention as context switching has been associated with cognitive difficulties faced during software maintenance [5, 9, 6].

The goal of this work therefore is to support users in editing formulas or values within a spreadsheet, whose underlying dependencies they might be unfamiliar with, without the need for them to leave the context of editing the spreadsheets. In this extended abstract, we introduce our tool *Aragorn*. Once enabled, Aragorn shows spreadsheet users the existence of dependents on every selected cell, ensuring the user is aware of them, and does not overlook making necessary changes to the dependents.

We conduct a preliminary evaluation of Aragorn, through an exploratory user-study with seven employees of a financial modeling company. Our findings show that Aragorn has a clear potential to help industrial spreadsheet users in the context of spreadsheet maintenance, by addressing the issue of hidden dependencies.

2 Background and Related Work

2.1 Cell Dependencies in Spreadsheets and Dependence Tracing

Formulas in spreadsheets can simply use values, like `=12+7`, but they may also refer to other cells within the spreadsheets. For this, spreadsheet users use cell references.

For example, if cell `A12` contains the formula `=B25+C25+K100`, then the value of `A12` depends on the values of `B25,C25` and `K100`; consequently `A12` is as a *dependent* of `B25,C25` and `K100`. Conversely, `B25,C25` and `K100` are the *precedents* of `A12`.

If a spreadsheet user wants to know the *precedents* of a cell, the user can simply read them from the formula. In the example `=B25+C25+K100`, the precedents are `B25,C25` and `K100`.

However, the converse is not true: when a spreadsheet user wants to know which are the *dependents* of a cell, i.e. which cells use its value, this is not possible without extra effort. The dependencies are hidden from spreadsheet users' (direct) view. To obtain an understanding of the dependencies in a spreadsheet, a user needs to use *dependence tracing* techniques. These are techniques for finding, visualizing, and navigating between dependents [7].

These techniques can be provided both as built-in features of spreadsheet applications, or as separate plug-in tools. For example, Microsoft Excel, has the *Audit toolbar* which provides an overlaid graph with the cells as nodes, and graph-edges shown with blue arrows depicting the cell dependencies.

2.2 Dependence Tracing Research

In addition to the built-in graph overlay feature of Excel, a number of techniques for dependence tracing have been proposed by researchers. For a summary, see our previous work [7]. While such proposed approaches all have their strengths and weaknesses, they share one important issue: they view dependence tracing as a separate activity. The spreadsheet user needs to navigate to a special menu created by the plugin [8, 1, 3], or enable a feature (Microsoft Excel, [4]) to start dependence tracing. While using such tools, users often find themselves spending dedicated time and effort solely for the purpose of understanding the dependencies inside a spreadsheet, making it an additional task on top of their normal work. This might be a reason why many of the dependence tracing techniques, have not found widespread adoption in industry yet [7].

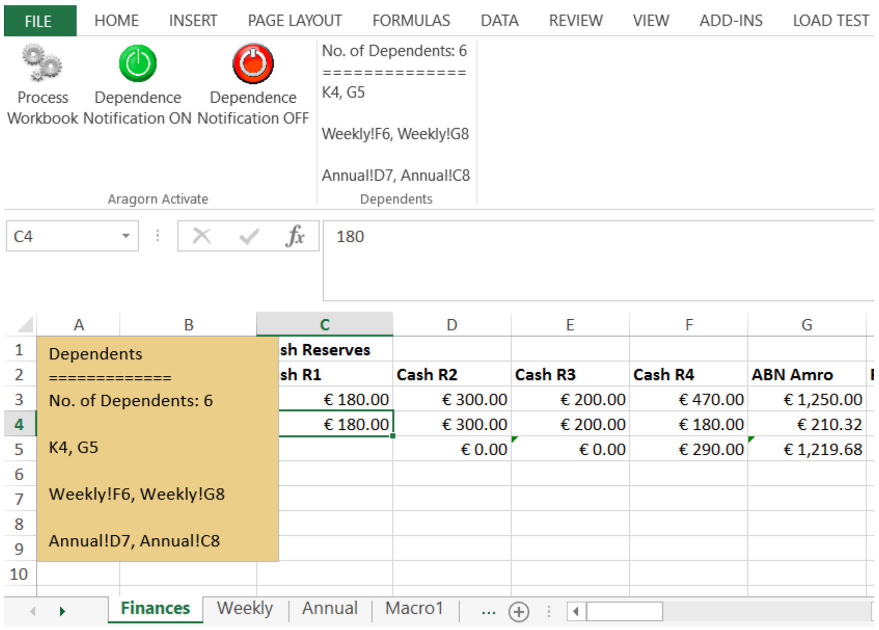
The goal of this work, therefore, is *to develop a technique that is specifically designed to support finding dependents in the context of spreadsheet maintenance.*

3 The Aragorn Approach

Aragorn supports spreadsheet users in maintaining spreadsheets whose dependency structures might be (partly) unfamiliar to them. It is simple and entirely integrated into the spreadsheet environment, to support dependence tracing as effortlessly as possible.

Aragorn notifies the spreadsheet users of the existence of dependents for the currently selected cell, by showing the users a popup. Figure 1 shows the user interface of Aragorn, with three features. Before a user can use Aragorn, they need to process the workbook, during which Aragorn constructs the dependence graph and keeps it in memory. The other two options the user has are to turn 'Dependence Notification' on and off. Figure 1 furthermore shows the popup shown to the user, once the 'Dependence Notification' is enabled, upon clicking $C4$ in worksheet 'Finances'. Dependents on the same worksheet are shown without a prefix (K4, G5), while dependents outside the worksheet are shown with their worksheet name prefixed (for example Weekly!F6). Apart from enabling the feature once, the user does not have to perform any other action. Aragorn shows the popups conveying dependence information about every cell that is selected, as the user navigates from cell to cell inside the spreadsheet.

In the popup, the user finds relevant information about dependents of the selected cell. It firstly shows the total number of direct dependents of the selected cell, which helps the user to obtain an idea how critical the selected cell might be. Secondly, it shows the locations of all direct dependents, both those that are in the same worksheet and those that are in a different worksheet; it displays



No. of Dependents: 6
==============
K4, G5

Weekly!F6, Weekly!G8

Annual!D7, Annual!C8

C4 | 180

Dependents
=============
No. of Dependents: 6

K4, G5

Weekly!F6, Weekly!G8

Annual!D7, Annual!C8

	A	B	C	D	E	F	G
1			sh Reserves				
2			sh R1	Cash R2	Cash R3	Cash R4	ABN Amro
3			€ 180.00	€ 300.00	€ 200.00	€ 470.00	€ 1,250.00
4			€ 180.00	€ 300.00	€ 200.00	€ 180.00	€ 210.32
5				€ 0.00	€ 0.00	€ 290.00	€ 1,219.68
6							
7							
8							
9							
10							

Fig. 1. The popup and the ribbon tab, showing dependents information when user clicks on cell C4 in the worksheet 'Finances'

the worksheet name and cell addresses. In addition to showing the dependence information in the popup, we also place it in the ribbon, as spreadsheet users are commonly used to obtaining information from the ribbon. Aragorn shows all of the above information for every cell the user selects, automatically. As such, the user does not have to perform any additional tasks, unlike other available dependence tracing techniques.

We have implemented the Aragorn approach in a .NET 4.5 based Microsoft Excel add-in written in C#. It is compatible with Office 2010 and 2013, and Windows 7 and 10. It uses the Infotron core engine to analyze the spreadsheets and obtain the list of dependents of all cells. The Infotron core engine is a commercialization of its predecessor Breviz— the spreadsheet analysis framework developed by Hermans *et al.* [3].

4 Preliminary Evaluation

4.1 Setup

In order to evaluate the usefulness of Aragorn, we conduct an exploratory user study with 7 professionals employed at F1F9— a financial modeling company based out of the UK. The main operations of F1F9 consists of analysis, development, auditing, and re-building of Excel based financial models.

We provide all the participants with the latest version of the Aragorn prototype, to install on their computers before the evaluation starts. They could then use the tool for a period of two weeks. After this period, we ask the participants to respond to a survey.

In the survey, we first provide the description of a scenario that sets up the spreadsheet maintenance context in which we intend to evaluate Aragorn. We subsequently ask six questions with one for identifying the participants, and the rest for evaluating Aragorn in the context of the described maintenance scenario.

4.2 Results

All seven of the participants agreed that they would be concerned in the described scenario, as the changes they make might inject errors into the spreadsheet, reaffirming the difficulty of spreadsheet modification related tasks.

On a five-point Likert scale, lack of dependency information was rated the second most important reason, slightly behind lack of familiarity about data, as probable reasons for the participants' concern about injecting errors. This result reaffirms that the problem of dependency is an important factor contributing to the difficulty of spreadsheet maintenance.

A total of six participants either agreed or strongly agreed that they would manually check dependencies for each cell they modified. This result points towards the lack of existing automated support for obtaining dependency related information during spreadsheet maintenance.

We asked if the participants could think of reasons for which they may prefer to skip checking of dependencies. This was an open question, and on analyzing the answers, we were able to identify five distinct groups of opinions the participants shared with us. We observe that two groups of opinions related to revelation of dependencies and time constraints, can be aimed to be addressed through supports like Aragorn, as Aragorn provides dependency information consuming minimal additional time. The other opinion groups are dependent on the type of maintenance or the nature of the spreadsheet to be modified, and as such are not possible to be addressed through support.

Finally, we asked the participants if they felt Aragorn will help making the task described in the scenario easier. Six of the participants agreed that Aragorn can make the task easier, and only one abstained.

From the results above, we can conclude that spreadsheet maintenance is difficult, and an important contributing factor to the difficulty is lack of dependency information. Normally, users are compelled to manually check for dependencies but dependency information being revealed to them or time constraints can be reasons due to which they may skip checking for dependencies. Addressing such reasons, Aragorn can help make the task of spreadsheet maintenance easier, as indicated by six out of the seven study participants.

In summary, we can state that based on responses from a set of seven industrial spreadsheet users, the overall preliminary evaluation of Aragorn, as an aid for spreadsheet maintenance that addresses the problem due to hidden dependencies, is promising.

5 Future Work

Our current work gives rise to several directions for future work. Firstly, there are some improvements to be made to the tool. For example, the performance of Aragorn could be improved to make it more feasible to use Aragorn in day to day spreadsheet use. Participants of our study also indicated a need for the ability to navigate to displayed dependents via hyperlinks. Secondly, we plan to perform a broader evaluation within a controlled setting, allowing for both qualitative as well as quantitative assessment of Aragorn, with the above improvements in place.

References

1. Ballinger, D., Biddle, R., Noble, J.: Spreadsheet visualisation to improve end-user understanding. In: Proceedings of the Asia-Pacific Symposium on Information Visualisation - Volume 24. pp. 99–109. APVis '03, Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2003), http://dl.acm.org/citation.cfm?id=857080.857093
2. Hermans, F.: Analyzing and Visualizing Spreadsheets. Ph.D. thesis, Delft University of Technology (2013)
3. Hermans, F., Pinzger, M., Van Deursen, A.: Supporting professional spreadsheet users by generating leveled dataflow diagrams. In: Proceedings of the 33rd International Conference on Software Engineering. pp. 451–460. ACM (2011)
4. Igarashi, T., Mackinlay, J.D., Chang, B.W., Zellweger, P.T.: Fluid visualization of spreadsheet structures. In: Visual Languages, 1998. Proceedings. 1998 IEEE Symposium on. pp. 118–125. IEEE (1998)
5. Lethbridge, T.C., Pak, J.: Integrated personal work management in tksee software exploration tool. In: Proc. of the 2nd Int. Symp. on Constructing Software Engineering Tools (CoSET'2000) (2000)
6. Parnin, C., Gorg, C.: Building usage contexts during program comprehension. In: 14th IEEE International Conference on Program Comprehension (ICPC'06). pp. 13–22. IEEE (2006)
7. Roy, S., Hermans, F.: Dependence tracing techniques for spreadsheets: An investigation. In: Proceedings of the 1st Workshop on Software Engineering Methods in Spreadsheets. p. 12 (2014)
8. Shiozawa, H., Okada, K.i., Matsushita, Y.: 3d interactive visualization for intercell dependencies of spreadsheets. In: Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on. pp. 79–82. IEEE (1999)
9. Zayour, I., Lethbridge, T.C.: A cognitive and user centric based approach for reverse engineering tool design. In: Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research. p. 16. IBM Press (2000)