The Journal of Systems and Software 94 (2014) 129–150

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Avoiding, finding and fixing spreadsheet errors – A survey of automated approaches for spreadsheet QA

Dietmar Jannach [a,*], Thomas Schmitz [a], Birgit Hofer [b], Franz Wotawa [b]

[a] TU Dortmund, Germany
[b] TU Graz, Austria

ARTICLE INFO

Article history:
Received 20 November 2013
Received in revised form 25 February 2014
Accepted 18 March 2014
Available online 27 March 2014

Keywords:
Spreadsheet
Quality assurance
Tool support

ABSTRACT

Spreadsheet programs can be found everywhere in organizations and they are used for a variety of purposes, including financial calculations, planning, data aggregation and decision making tasks. A number of research surveys have however shown that such programs are particularly prone to errors. Some reasons for the error-proneness of spreadsheets are that spreadsheets are developed by end users and that standard software quality assurance processes are mostly not applied. Correspondingly, during the last two decades, researchers have proposed a number of techniques and automated tools aimed at supporting the end user in the development of error-free spreadsheets. In this paper, we provide a review of the research literature and develop a classification of automated spreadsheet quality assurance (QA) approaches, which range from spreadsheet visualization, static analysis and quality reports, over testing and support to model-based spreadsheet development. Based on this review, we outline possible opportunities for future work in the area of automated spreadsheet QA.

1. Introduction

Spreadsheet applications, based, e.g., on the widespread Microsoft Excel software tool, can nowadays be found almost everywhere and at all levels of organizations (Panko and Port, 2012). These interactive computer applications are often developed by non-programmers – that is, domain or subject matter experts – for a number of different purposes including financial calculations, planning and forecasting, or various other data aggregation and decision making tasks.

Spreadsheet systems became popular during the 1980s and represent the most successful example of the End-User Programming paradigm. Their main advantage can be seen in the fact that they allow domain experts to build their own supporting software tools, which directly encode their domain expertise. Such tools are usually faster available than other business applications, which have to be developed or obtained via corporate IT departments and are subject to a company's standard quality assurance (QA) processes.

Very soon, however, it became obvious that spreadsheets – like any other type of software – are prone to errors, see, e.g., the early paper by Creeth (1985) or the report by Ditlea (1987), which were published in 1985 and 1987, respectively. More recent surveys on error rates report that in many studies on spreadsheet errors at least one fault was found in every single spreadsheet that was analyzed (Panko, 1998). Since in reality even high-impact business decisions are made, which are at least partially based on faulty spreadsheets, such errors can represent a considerable risk to an organization[1].

Overall, empowering end users to build their own tools has some advantages, e.g., with respect to flexibility, but also introduces additional risks, which is why Panko and Port call them both "dark matter (and energy) of corporate IT" (Panko and Port, 2012). In order to minimize these risks, researchers in different disciplines have proposed a number of approaches to avoid, detect or fix errors in spreadsheet applications. In principle, several approaches are possible to achieve this goal, beginning with better education and training of the users, over organizational and process-related measures such as mandatory reviews and audits, to better tool support for the users during the spreadsheet development process. In this paper, we focus on this last type of approaches, in which the spreadsheet developer is provided with additional software tools

* Corresponding author at: TU Dortmund, 44221 Dortmund, Germany. Tel.: +49 231 755 7272; fax: +49 231 755 7269.
E-mail address: dietmar.jannach@tu-dortmund.de (D. Jannach).

[1] See http://www.eusprig.org/horror-stories.htm for a list of real-world stories or the recent article by Herndon et al. (2013) who found critical spreadsheet formula errors in the often-cited economic analysis of Reinhart and Rogoff (2010).

http://dx.doi.org/10.1016/j.jss.2014.03.058
0164-1212/

and mechanisms during the development process. Such tools can for example help the developer locate potential faults more effectively, organize the test process in a better structured way, or guide the developer to better spreadsheet designs in order to avoid faults in the first place. The goals and contributions of this work are (A) an in-depth review of existing works and the state-of-the-art in the field, (B) a classification framework for approaches to what we term "automated spreadsheet QA", and (C) a corresponding discussion of the limitations of existing works and an outline of perspectives for future work in this area.

This paper is organized as follows. In Section 2, we will define the scope of our research, introduce the relevant terminology and discuss the specifics of typical spreadsheet development processes. Section 3 contains our classification scheme for approaches to automated spreadsheet QA. In Sections 4–9, we will discuss the main ideas of typical works in each category and we will report how the individual proposals were evaluated. Section 10 reviews the current research practices with respect to evaluation aspects. In Section 11, we point out perspectives for future works and Section 12 summarizes this paper.

2. Preliminaries

Before discussing the proposed classification scheme in detail, we will first define the scope of our analysis and sketch our research method. In addition, we will briefly discuss differences and challenges of spreadsheet QA approaches in comparison with tool-supported QA approaches for traditional imperative programs.

2.1. Scope of the analysis, research method, terminology

Spreadsheets are a subject of research in different disciplines including the fields of Information Systems (IS) and Computer Science (CS) but also fields such as Management Accounting or Risk Management, e.g., Creeth (1985) or Galletta et al. (1993).

Scope. In our work, we adopt a Computer Science and Software Engineering perspective, focus on tool support for the spreadsheet development process and develop a classification of automated spreadsheet QA approaches. Examples for such tools could be those that help the user locate faults, e.g., based on visualization techniques or by directly pointing them to faulty cells, or tools that help the user avoid making faults in the first place, e.g., by supporting complex refactoring work. Spreadsheet error reduction techniques from the IS field, see, e.g., Thorne (2014), and approaches that are mainly based on "manual" tasks like auditing or code inspection will thus not be in the focus of our work.

Research on spreadsheets for example in the field of Information Systems often covers additional, more user-related, or fundamental aspects such as error types, error rates and human error research in general, the user interface, cognitive effort and acceptance issues of tools, user over-confidence, as well as methodological questions regarding the empirical evaluation of systems, see, e.g., Reinhardt and Pillay (2004), Galletta et al. (1996), Powell et al. (2008), Howe and Simkin (2006), Olson and Nilsen (1987), Panko (1998). Obviously, these aspects and considerations should be the basis when designing an automated spreadsheet QA tool that should be usable and acceptable by end users. In our work and classification, we however concentrate more on the provided functionality and the algorithmic approaches behind the various tools. We will therefore discuss the underlying assumptions for each approach, e.g., with respect to user acceptance or evaluation, only as they are reported in the original papers. Still, in order to assess the overall level of research rigor in the field, we will report for each class of approaches how the individual proposals were evaluated or validated.

These insights will be summarized and reviewed in Section 10. In this section, we will also look at the difficulties of empirically evaluating the true value of spreadsheet error reduction techniques according to the literature from the IS field.

Regarding tool support in commercial spreadsheet environments, we will briefly discuss the existing functionality of MS Excel and comparable systems in the different sections. Specialized commercial auditing add-ons to MS Excel usually include a number of QA tools. As our work focuses more on advanced algorithmic approaches to spreadsheet QA, we see the detailed analysis of current commercial tools to be beyond the scope of this paper. Finally, we will also not cover fault localization or avoidance techniques for the imperative programming extensions that are typically part of modern spreadsheet environments.

Research method. For creating our survey, we conducted an extensive literature research. Papers about spreadsheets are published in a variety of journals and conference proceedings. However, there exists no publication outlet which is only concerned with spreadsheets, except maybe for the application-oriented EuSpRIG conference series.[2] In our research, we therefore followed an approach which consists both of a manual inspection of relevant journals and conference proceedings as well as searches in the digital libraries of ACM and IEEE. Typical outlets for papers on spreadsheets which were inspected manually included both broad Software Engineering conferences and journals such as ICSE, ACM TOSEM, or IEEE TSE. At the same time, we reviewed publications at more focused events such as ICSM or the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC). In addition, major IS journals and events such as Information Systems Research, ACM TOIS or ICIS were considered in our research.

When searching the digital libraries, we started by looking for papers containing the term "spreadsheet" in the title or abstract. From the 400 to 500 results returned by the search engines of the libraries, we manually inspected the abstracts. Provided their scope was relevant for our research, we categorized them according to the categorization framework described in Section 3, and followed the relevant references mentioned in the articles.

Terminology. Regarding the terminology used in the paper, we will use the terms "spreadsheet", "spreadsheet application", or "spreadsheet program" more or less interchangeably as often done in the literature. When we refer to the underlying software system to create spreadsheets (e.g., Microsoft Excel), we will use the term "spreadsheet environment" or "spreadsheet tool". In some papers, the term "form-based visual languages" is used (Rothermel et al., 1998) to describe the more general family of such systems. In our work, we will however rely on the more widespread term "spreadsheet".

There are a number of definitions of the terms "error", "fault", and "failure" in the literature. According to IEEE standards for Software Engineering an "error" is a misapprehension on side of the one developing a spreadsheet caused by a mistake or misconception occurring in the human thought process. A "fault" is the manifestation of an "error" within a spreadsheet which may be causing a "failure". A "failure" is the deviation of the observed behavior of the spreadsheet from the expectations. In the literature on spreadsheets, in particular the terms "fault" and "error" are often used in an interchangeable manner. Surveys and taxonomies of spreadsheet problems like (Panko and Halverson, 1996; Rajalingham et al., 2001), or (Panko and Aurigemma, 2010), for example, more or less only use the term "error". In our review, we will – in order to be

[2] http://www.eusprig.org.