Beyond the Spreadsheet
Reflections on Tool Support for Literature Studies
Tell, Paolo; Cholewa, Jacob; Kuhrmann, Marco; Nellemann, Peter

Published in:
Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering

DOI:
10.1145/2915970.2916011

Publication date:
2016

Document version
Peer reviewed version

Document license
Unspecified

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 26. dec., 2018
Beyond the Spreadsheet: Reflections on Tool Support for Literature Studies

Paolo Tell, Jacob B. Cholewa
IT University of Copenhagen
Copenhagen, Denmark
{patel|jbec}@itu.dk

Peter Nellemann, Marco Kuhrmann
University of Southern Denmark
Odense, Denmark
{pnel|kuhrmann}@mumi.sdu.dk

ABSTRACT

Background: Even though a number of tools are reported to be used by researchers undertaking systematic reviews, important shortages are still reported revealing how such solutions are unable to satisfy current needs. Method: Two research groups independently provided a design for a tool supporting systematic reviews. The resulting tools were assessed against the feature lists provided by prior research. Results: After presenting an overview of the tools and the core design decisions taken, we provide a feature analysis and a discussion regarding selected challenges deemed crucial to provide a proper tool support. Conclusions: Although the designed solutions do not yet support the entire systematic review process, their architecture has been designed to be flexible and extendable. After highlighting the difficulties of developing appropriate tools, we call for action: developing tools to support systematic reviews is a community project.

CCS Concepts

•Software and its engineering → Software usability;
•Software prototyping: Software notations and tools; Designing software; Rapid application development;

Keywords

Design exploration, literature study, tool, prototyping

1. INTRODUCTION

Since Kitchenham’s work [4] in 2004, systematic reviews (SR; or systematic literature reviews, SLR) have emerged as a successful evidence-based approach in software engineering. However, performing literature studies challenges researchers, as: (i) they tend to be time-consuming due to both their intrinsic systematic nature and the amount of data they often require to handle, (ii) they are highly susceptible to errors as currently almost entirely depending on humans, and (iii) they require the cooperation of a team of researchers to ensure quality and avoid bias. Therefore, literature studies would greatly benefit from dedicated tool support, which is currently limited. So far, a number of tools have been used [9] of which few are specifically designed to support systematic reviews. Yet, researchers cannot rely on an adequate solution, even though a feature list has been published recently to guide future design endeavors [10]; and, the classic spreadsheet application seems to be the preferred fit due to its simplicity and flexibility.

Objective. Motivated by both the above challenge and personal needs, two research teams engaged in the creation of two independent and competing designs. Still respecting the features identified in [10], the goal of our endeavor was to push the boundaries of the currently available designs and to elaborate whether appealing solutions could be suggested.

Contribution. In this paper, we present our two solution proposals LiSA and ReviewIt. We aim to both present their design and stimulate a vivid discussion in the community that hopefully results in future collaborations to eventually provide us with a tool able to ease the burden of performing literature studies.

Outline. The remainder of this paper is organized as follows: in Sect. 2, we summarize related work and derive feature lists to properly support systematic reviews with tools. Sectio 3 describes the approach chosen to develop the tools presented in Sect. 4 (LiSA) and Sect. 5 (ReviewIt). Section 6 provides a discussion. We conclude the paper in Sect. 7.

2. RELATED WORK

Proper tool support for literature studies puzzled researchers now for years; thus, a number of tools have been reported to be used by researchers when engaging in literature studies [9, 11]. In the following, we provide a brief description of the major categories in which such tools could be grouped. Furthermore, we summarize the systematic review stages to be supported by tools and the features considered key for proper tool support.

Reference Managers. Representatives of this category are for instance EndNote, Mendeley, Papers, and Zotero, which allow researchers to store and manage bibliographies. Among other features that might be supported, maybe the most interesting relate to searching studies by querying digital libraries and tagging of individual articles.

Generic Support. This category captures those tools that we feel are the most frequently used: spreadsheet applications like Microsoft Excel. Because of their multi-purpose nature, these tools support researchers quite effectively, and
Table 1: Breakdown of SR stages presented in [7], including the assessment of LiSA and ReviewIt.

<table>
<thead>
<tr>
<th>Stage</th>
<th>LiSA</th>
<th>ReviewIt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification of the need for a review</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Commissioning a review (not mandatory)</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Specifying the research question(s)</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Developing a review protocol</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Evaluating the review protocol</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Conducting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification of research</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Selection of primary studies</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Study quality assessment</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Data extraction and monitoring</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Data synthesis</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Reporting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specifying dissemination mechanisms</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Formatting the main report</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Evaluating the report (not mandatory)</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Note: the support provided can be none (O), partial (anyak), or full (⬤).

Table 2: Prioritized features list presented in [11], including assessment of LiSA and ReviewIt.

<table>
<thead>
<tr>
<th>Feature</th>
<th>LiSA</th>
<th>ReviewIt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple users</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data extraction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Simple installation and setup procedure</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Document management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality assessment and validation</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Automated analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Study selection and validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meta-analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-use of data from past projects</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Search process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role management</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Development of review protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol validation</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Self-contained</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No financial payment</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Report validation</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Text analysis</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>Report write-up</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Note: the support provided can be none (O), partial (anyak), or full (⬤).

we believe much should be learned by this fact. Many of the other tools do not manage to be flexible enough to provide adequate support to researchers, so they eventually rely on a (scripted) spreadsheet.

**Systematic Review Support.** A small number of tools dedicated to support researchers performing systematic reviews have been proposed in the last years; notable examples are SLuRp [1], SESRA [12], and StArt [3]. These tools were analyzed in [10], yet those are not ranked with flying colors.

**Desired Support and Features.** Over the years, the initial guidelines [4] evolved, e.g., [7] and eventually [6]. The systematic review process defines different stages, which are summarized in Table 1 and that serve as input for directing the development of supporting tools. Furthermore, a set of features have been outlined by Marshall et al. [11]. The list was generated based on 13 semi-structured interviews, and the prioritized list is shown in Table 2. Taken together, they provide a high-level overview of features that provide a fairly large design space that should be carefully considered when designing tools. For space limitations, the two tables also include the results of the feature analysis of the tools herein presented, which will be discussed in Sect. 6.

Nevertheless, even though the key features for supporting tools are known and a number of special-purpose tools are available, still, spreadsheet tools seem to be the tools of choice. Spreadsheet tools are widely known and accessible and, furthermore, they allow researchers to tweak them as needed. In the following, we present our approaches to make the next step beyond the spreadsheet.

3. APPROACH

The driver for our work was to reflect on the design space of tool support for literature studies. In two teams, we independently developed two designs driven by the theme: "the silver bullet" for supporting literature studies. Our starting point was: (i) a shared requirements collection comprising 15 product- and five realization-related requirements mainly reflecting our personal needs emerging from conducting literature studies in several occasions, (ii) the requirements based on the common guidelines for systematic reviews [7], and (iii) additional related literature, e.g., [2, 14, 8, 13]. To focus our efforts, we agreed that: (i) experience overrules theory, but (ii) the guidelines from [7] form a baseline, and (iii) the features prioritization provided in [11] must not limit the development at this stage. Previous work, in particular [7,11], is used afterwards as an evaluation tool to assess the solutions via a feature analysis [5].

Practically, we set up three student projects: the first project was hosted at SDU in 2014 to investigate a set of meaningful requirements and demonstrate the realizability. In this project, students were provided with several reference studies, several Microsoft Excel files used to collect and document study data, and students were given the task to develop "something that is as flexible as the spreadsheet, but realized as a special-purpose application providing optimal support for researchers." This project was followed by the second project in which LiSA (Sect. 4) was developed in 2015. Also in 2015 and based on the requirements, the third project was hosted by ITU in which ReviewIt (Sect. 5) was developed. The projects 1, 2, and 3 only shared the requirements, but were carried out fully separated from each other (even the supervisors did not communicate in this period) to explore the design space for the problem given. In the beginning of 2016, tools and experiences were collected and are reported in the paper at hand.

4. LISA

The implementation of LiSA serves basically two purposes: (i) collect all information required to conduct a systematic review and organize this endeavor as a project, and (ii) provide means to coordinate distributed teams who collaboratively work on a systematic review. For this, LiSA is in principle designed as a distributed application, but also supports individual researchers. The main page of LiSA serves as landing page for the users to select the different options, such as configuring the project, evaluating articles in the database, adding further database items on demand, or generating reports. Hereby, LiSA implements a simple concept, which is shown in Figure 1.

5. REVIEWIT

ReviewIt is designed emphasizing one aspect that was deemed missing in all other surveyed tools: the enforcement of the protocol. The protocol contains the description of the workflow that must be followed to complete the review. It is through the analysis of such a document or its fragments in the final dissemination articles that we rely on the quality of a given literature study. Yet, no available tool allows a team to enforce such workflow.

![Figure 3: High-level conceptual model of ReviewIt.](image)

Figure 3 provides a pictorial description of how ReviewIt decomposes a literature study. A Study comprises three elements: a team, the Researchers involved in the study; the Acceptance Policy, either the quality assessment or the in-/exclusion criteria; and, a number of stages.

In ReviewIt, a Stage (e.g., “based on the title assess whether the paper is related to global software engineering”) is the computational unit that allows the designer of a study to instruct ReviewIt on which tasks must be created (encoded in Fields) and how these are to be handled (the Strategy).

A task is detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).

Besides additional information that are captured in the model like the title of a study, the research questions, and descriptions for each element in the model, ReviewIt supports researchers by providing a significant degree of automation. The most relevant automations happen: (i) during the data extraction from the bibliography file(s); (ii) in the task generation; and, (iii) in the application of acceptance policies.

First, any information contained in the input file(s) is automatically classified according to the data fields contained in such file(s). Second, units of work are generated based on the stages and their Strategy, hence, enforcing the protocol. To achieve this, in ReviewIt stages are detailed by describing which fields should be visible to and which ones should be requested from the user. Fields can be of two types but they are treated equally within ReviewIt: fields that are automatically extracted from the bibliography file(s) (e.g., title, abstract, etc.), which is the input to the system (e.g., a BibTeX file containing the result of the ‘identification of research’ stage [7]); and, user-defined ones, which allow the designer of a study to capture any kind of element ranging from the ones based on which acceptance policies must be assessed to those describing the data that must be extracted from the papers (e.g., “the study type and method reported in an article”).
detailed overview of the functionalities of the tools against the guidelines identified in [7] and the prioritized features list provided in [11] respectively. In the following, we will discuss selected key areas worth further analysis.

6.1 Are we really “systematic”?

The systematic review method requires each review to be planned and carried out following the plan detailed in the protocol and, therefore, we trust in the quality of systematic reviews. However, available tools provide limited support—if any—to ensure that the protocol is appropriately enacted and followed. We therefore argue that design efforts should focus on ensuring that additional support is provided to researchers to guarantee adherence to the plans described in protocol.

This aspect has been explored by both LiSA and ReviewIt leading to two different approaches: while ReviewIt provides extensive options to define a workflow and enforcing it afterwards via tasks generation, LiSA provides support to define stages and decision points, yet leaves the actual approach of reaching those open. Both tools allow to add/modify extra stages on-the-fly. Differently from LiSA, stages in ReviewIt can only be modified if not yet initiated by any team member or added after the already completed/initiated ones. We therefore see the two extremes: the choreographed work following a predefined workflow, and the free spreadsheet-like work pattern. Understanding which of the two, or any other approach for that matters, is preferable should be the topic of further investigation involving a broader experts group.

6.2 The Need for a Flexible Solution

Throughout our projects, it quickly became clear that predicting all requirements, which researchers might have regarding almost any aspect of the solutions, was not possible. Considerations as simple as “what kind of voting procedure to implement?”, “what conflict resolution mechanisms to support?”, or “which type of data fields to include?” hid many subtleties that got overlooked in the early stage of design. Even though the systematic review approaches have been extensively described in a number of publications, the nitty-gritty tweaks that are implemented in practice by individuals are hard to predict. Therefore, hard-coding (informed) personal decisions about the variations a tool should include is an approach prone to failure.

To provide a solution useful to a larger audience, we argue that the system must be flexible. However, flexibility is not simply obtained through an appealing user interface or via a multitude of options per feature, but must be considered at the architecture level. Researchers must be able to plug-in extra components to enhance support provided by the system to accommodate the needs and not to force them to fall back to spreadsheets. To provide a few examples, these could include support to aggregated data type capable of capturing an acceptance criteria based on combinations of multiple available data, specific types of automated analysis based, for instance, on meta-analysis or text analysis, opportunities to conduct long-term studies or study updates, tailored conflict resolution mechanisms, and interfaces to external analysis tools. Therefore, we argue that, even though the application domain is well-understood by the people involved in the design of tools dedicated to literature studies, solutions must be carefully designed by highly prioritizing flexibility.
6.3 A Call for Collaboration

One of our major intentions when engaging in this project was to stimulate a discussion in the community by challenging our belief that “available solutions are limited, and that a spreadsheet still represents the best and most flexible tool I can quickly get a hold on”. This is not necessarily the case going forward, we can do better!

We hope we managed to show that a single person/team will not be able to provide the comprehensive solution that we are all awaiting. Therefore, the solutions herein discussed represent a mere suggestion regarding how some focused aspects can be approached, and we expect that this paper will stimulate interesting discussion in the area of tool support for systematic reviews. Finally, we hope collaborators will join us to tackle a multitude of additional aspects required to design, develop, evaluate, and maintain a tool that we are all craving for by being available for expert opinion, evaluations, comparative analysis, as well as to speed up development.

6.4 Future Work

The paper at hand proposed two approaches to support researchers. These proposals add to the set of available tools (Sect. 2) and both explore ways of providing functionality to the users. As we mentioned in our discussion, many features still remain unexplored, as researchers developed a multitude of practices to compensate for missing tool support. Therefore, we plan to use the experiences collected in the Global Software Engineering (GSE) community to set up a joint development activity and to evolve the tools. Apart from evolving the tools presented, future work includes explorative as well as comparative case studies to (i) evaluate and compare the existing tools, (ii) improve and refine the requirements for proper tool support, and (iii) incrementally develop a consolidated platform to technically and methodically support systematic reviews.

7. CONCLUSIONS

In this paper, we have presented two competing designs for a tool able to improve the currently available solutions supporting literature studies. The resulting designs, which have deliberately focused on selected features and stages of systematic reviews, have shown that improvements can be achieved by carefully tackling quality attributes like flexibility and extensibility as well as properly supporting the ‘systematic’ nature of the systematic review method. However, the solutions discussed in the paper at hand have to be considered a mere suggestion, and we expect this paper to stimulate further discussion in the area of tool support for literature studies. This paper is a call for action, and we hope that researchers will join us in tackling the multitude of additional aspects required to get closer to a silver bullet.

Acknowledgments

This research has been funded by the Danish Agency for Science, Technology and Innovation under the project “Next Generation Technology for Global Software Development”, #10-092313.

8. REFERENCES