SLuRp – A Tool to Help Large Complex Systematic Literature Reviews Deliver Valid and Rigorous Results

David Bowes
Science and Technology Research Institute
University of Hertfordshire
Hatfield, Herts
d.h.bowes@herts.ac.uk

Tracy Hall
Department of Information Systems and Computing
Brunel University
Uxbridge, UK
tracy.hall@brunel.ac.uk

Sarah Beecham
Lero – The Irish Software Engineering Research Centre
University of Limerick
Limerick, Ireland
sarah.beecham@lero.ie

ABSTRACT

Background: Systematic literature reviews are increasingly used in software engineering. Most systematic literature reviews require several hundred papers to be examined and assessed. This is not a trivial task and can be time consuming and error-prone.

Aim: We present SLuRp - our open source web enabled database that supports the management of systematic literature reviews.

Method: We describe the functionality of SLuRp and explain how it supports all phases in a systematic literature review.

Results: We show how we used SLuRp in our SLR. We discuss how SLuRp enabled us to generate complex results in which we had confidence.

Conclusions: SLuRp supports all phases of an SLR and enables reliable results to be generated. If we are to have confidence in the outcomes of SLRs it is essential that such automated systems are used.

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: *General*

General Terms

Management, Measurement, Documentation, Standardization.

Keywords

Systematic literature reviews, SLRs, collaboration tool, Open Source.

1. INTRODUCTION

Systematic literature reviews (SLRs) are increasingly established as an important aspect of software engineering research. The Journal of Information and Software Technology (IST) has a special section devoted to publishing SLRs. Top rated software engineering journals, such as IEEE Transactions on Software Engineering (TSE) and ACM's Transactions on Software Engineering and Methodology (TOSEM) have published numerous SLRs. These papers tend to have a high impact where as many as 334 citations have been recorded (e.g. [1]).

Although conducting SLRs has become popular they are difficult to execute well [2]. Performing an SLR in software engineering is a large, time consuming and complex task. Many hundreds or even thousands of papers can be identified as potentially relevant in the early stages. An extreme example of this is [3] where reviewers sourced over 3,000 papers, and only used 7 of them in their final review. In our own most recent SLR of 208 studies on fault prediction performance in software engineering [4], we initially identified 2,073 papers. Many pieces of information about all of these papers need to be accurately recorded, maintained, analysed and reported.

In addition to managing a large number of papers, performing an SLR requires many steps. All of these steps need to be implemented accurately for every paper.

In this paper we introduce a tool, 'SLuRp', to support the complex task of managing large numbers of papers, sharing tasks amongst a research team and following the arduous and rigorous SLR methodology recommended by Kitchenham and Charters [12]. Our contribution is to provide the research community with a support tool that increases the rigor and validity of our hitherto manual methods, while simplifying and shortening the time required to implement the many steps required to conduct a SLR.

Our own SLR [4], implemented twelve distinct steps. In most cases several reviewers are involved in an SLR. Our own SLR involved five different reviewers. The data each reviewer collects needs to be reliably stored and collated. This administrative complexity puts the quality of SLRs at risk, as the reliability and credibility of SLR conclusions are dependent on the quality of the SLR process used [5].

The recording and management of SLR data is often not reported in papers. Where the process is reported manual records are common [6, 7], as is the use of Endnote [8]. Both of these approaches are potentially problematic. Manual record keeping is time consuming and fault-prone. The functionality of Endnote supports only a limited number of SLR steps.

Other researchers have recognised the need to automate and simplify the SLR methodology for improved accuracy and speed of execution. According to Felizardo et al [13], their Systematic Mapping Visual Text Mining tool (SM-VTM) reduces the effort and time required to categorize and classify data in systematic mapping studies. However the SM-VTM approach has

questionable usability as it requires some prior experience and knowledge of text mining and visualization techniques. Malheiros et al [14] developed a similar VTM tool to specifically support the selection of primary studies in the systematic review process. VTM was shown to speed-up the selection process and improve quality of the selection process. While providing a potentially useful way to mine and cluster information from primary studies in a SLR, VTM does not provide the holistic, management of the whole SLR process, that requires tracking progress and storing of related data as in our SLuRp approach. Previous automated approaches focus on one aspect of the complex SLR process, leaving researchers to manage and integrate the new methods with their manual approaches. Eppi-reviewer[1] has been extensively used to support SLRs in other domains. It imports articles using reference manager databases and has a heavy focus on the synthesis aspect of literature reviews. However Eppi-reviewer is very generic and it is not obvious how to perform quality checks.

We could find no previous studies that look in detail at the way in which papers and data are managed and processed in existing SLRs. This is an important omission as noted by [9] who call for effective information retrieval tools to support performing systematic reviews given the "growth of the number of available papers and results published in the empirical software engineering field".

In response to the difficulties we experienced in managing and recording information in our first three SLRs [8, 10, 11] we developed SLuRp (Systematic Literature unified Review program). This is our own web enabled database for recording and managing all data necessary to perform an SLR. SLuRp is written in Java with an SQL database. It is open source and available for other reviewers to adapt and use[2]. SLuRp has a client side Java component which semi-automates the process of extracting information when querying online databases.

In the next section we describe the functionality of SLuRp. In Section Three discuss the advantages of SLuRp using examples from our own SLR. We conclude in Section Four.

2. THE FUNCTIONALITY OF SLuRp

SLuRp is designed around the SLR tasks required to conduct a review according to Kitchenham and Charters's [12] guidelines. How SLuRp supports each SLR task is discussed in this section.

2.1 Identify Relevant Research

All definitions required in the SLR must be established by the SLR research team. These include definitions of research questions, search terms, inclusion/exclusion criteria, quality check criteria etc. Once established all of these definitions are stored centrally on SLuRp.

2.2 Select Primary Studies.

a) SLuRp can apply pre-defined search terms to online databases (this is not permitted by some online databases e.g. ACM Portal).
b) SLuRp can semi-automatically extract papers from databases and save these.
c) SLuRp can semi-automatically store [PDF] copies of all papers locally if appropriate permissions exist.
d) SLuRp can record bibliographic details by importing BibTeX/RIS files from other citation management systems.
e) SLuRp prompts users to assign two+ reviewers to each paper. Each of the two+ reviewers independently applies the previously defined and stored inclusion and exclusion criteria to their assigned papers in turn.
f) SLuRp records assessment of each reviewer against the inclusion/exclusion criteria; and
g) SLuRp Records reason for rejection / acceptance.
h) SLuRp will identify differences in reviewer selections.
i) The need to reconcile disagreements between reviewers will be flagged by SLuRp.
j) Where reviewers cannot agree, SLuRp allows the user to assign an arbitrator. The paper remains in a 'needs moderation' state until a consensus is reached.
k) Once a decision has been recorded, SLuRp will remove rejected papers and record the reason (though SLuRp will change the status of papers to rejected, rather than remove them from the database). The frequency of disagreements for including or excluding papers is automatically generated by SLuRp. This allows inter-rater reliability scores to be easily produced.
l) Where reviewers have agreed to include the paper, SLuRp will store full copies of accepted papers (though SLuRp can be set up to store full copies of all papers).

2.3 Assess Study Quality

a) SLuRp allows reviewers to define a set of quality criteria. The application of which SLuRp supports in the same way as it deals with the application of inclusion/exclusion criteria. Steps e) to k) are repeated in the application of the quality criteria. SLuRp will: allocate each accepted paper to two reviewers; record results of quality assessment; flag differences in reviewer assessments; invite reviewers to reconcile differences between assessments; flag that a paper needs arbitration where no agreement is reached and record the results of quality assessment.

b) SLuRp allows reviewers to record all data extracted from included papers which have passed the quality check. It can record data about the study and the context of the study. For example dates of study, type of study, etc. It can also record data relating to the SLR's research questions. Quantitative and qualitative can be recorded. This is achieved by each reviewer completing a SLuRp form answering project defined questions. Answers can be in the form of categorical, numerical or free flow textual data. Usually drop down type answers are linked to recording categorical data, whereas free textual data is required for qualitative data extraction.

We also wanted to ensure the validity to the data extracted from included papers. Consequently SLuRp also allows two+ reviewers to extract data from each included paper. The results of these independent data extractions are compared and reconciled using the same process as applied during the application of inclusion/exclusion criteria and during the quality check (i.e. moderation between reviewers, followed by independent arbitration).

2.4 Synthesize Data

a) SLuRp aggregates quantitative findings and displays these in tables, or graphical form, e.g. box plot. All data is stored in a database which links the papers to the data extracted. This allows the data to be aggregated and statistically analysed

[1] http://eppi.ioe.ac.uk/cms/Default.aspx?alias=eppi.ioe.ac.uk/cms/er4

[2] https://bugcatcher.stca.herts.ac.uk/SLuRp

using SQL statements. SLuRp allows the results of the data analysis to be presented in two main forms:

- Tabular: results can be cross-tabulated and presented in HTML or LaTeX format for inclusion in papers.
- Graphical: SLuRp uses JFreeChart[3] to produce: box plots, pie charts, scatter plots, bar charts.

Each chart can be produced as: pdf, jpg, svg and png.

b) These graphical displays can highlight trends in results and allow the team to consider bias in results.

c) SLuRp supports the research team to synthesise qualitative data, according to the research questions (e.g report cross cutting themes across papers), by recording qualitative information which can then be aggregated and analysed in tabular and graphical format.

2.5 The Advantages of Using SLuRp

We have shown how SLuRp directly supports the implementation of SLR tasks. SLuRp also offers reviewers additional benefits we now discuss.

Centrally storing the SLR protocol and all associated instructions and criteria. The validity of SLR results is undermined if the protocol and associated instructions and criteria are not properly understood or applied consistently. SLuRp allows all SLR documentation to be centrally stored. This means that current versions of documents are always easily accessible to the whole team. This also means that current versions of all guidelines, instructions, definitions and criteria are those being used by everyone. In our previous SLRs such definitions change regularly and we have had problems ensuring that the right versions of definitions and instructions are being used by everyone. SLuRp's centralized storage of instructions, definitions and criteria ensures consistent application across papers and across reviewers.

Figure 1 is a screen shot of the first page SLuRp generates every time a user logs onto our own fault prediction SLR. At the top of the screen is the menu list, which has various options which include allowing access to the: instructions for all aspects of the SLR, full list of papers, editing facilities for any criteria used and the codes defined for data extraction etc.

This menu allows full access to all documentation (instructions, guidelines, definitions, criteria, code schemes etc.). All of this documentation can be edited via this menu. Editing access is determined by a variety of SLuRp user permissions.

Figure 1 is a screen shot of the first page SLuRp generates every time a user logs on; using our own fault prediction SLR as an example. The charts we implemented in our SLR dynamically track the progress of: individual reviewers, the current status of papers and the number of papers completed over time. The progress charts produced can be varied according to the specific needs of the SLR. These charts give immediate insight into the progress of individual reviewers and the team as whole.

[3] http://www.jfree.org/jfreechart/

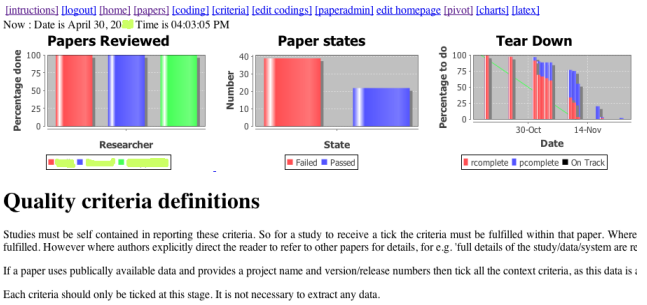

Figure 1. Screenshot of opening screen

Maintaining a central list of papers and their current status. A central list of all papers is always maintained by SLuRp. This list can be filtered according to the particular reviewer assigned to papers. This allows a reviewer to easily identify the papers assigned to them and access all relevant information regarding that paper, including the pdf of the full paper. This central list of papers can also be filtered according to the status of each paper. This allows sets of papers to be easily generated, for example, all those papers which have been included, or need to be moderated. Overall this central list of papers makes accessing the right papers easy and less error-prone.

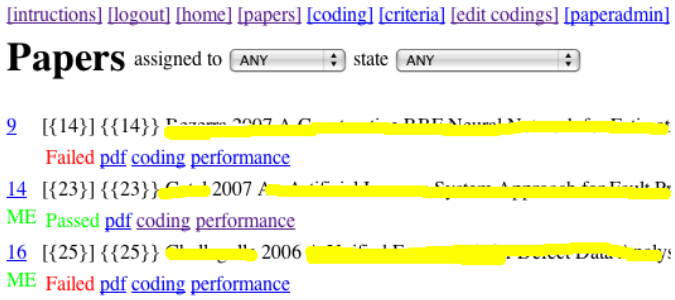

Figure 2. List of papers screenshot

Figure 2 shows the central list of papers maintained by SLuRp for our own fault prediction SLR. Figure 2 shows that every paper is listed. The details provided for each paper include:

- paper id number

- publication details (blanked out in Figure 2 to keep anonymous papers that did not pass our quality check)

- link to the pdf

- status in terms of passed or failed the quality check (colour coded, red for Failed, and green for Passed)

- links to pages for extracted data (coding) and performance data (a second set of data we extracted during our SLR).

Figure 2 also shows the filters that can be applied to this list of papers. For example, the list can be filtered according to the papers assigned to a particular reviewer (assigned to), and/or filtered to list only those papers which are included or which have passed the quality check (state). This allows a specific list of papers to be generated according to the particular task a particular reviewer is currently working on.

Controlling and managing the SLR process. SLuRp produces information showing the current status of every paper. For example, papers are labelled as either accepted, rejected, passed or failed the quality check, undecided or in need of moderation or arbitration. SLuRp also produces information showing the current

status of every reviewer. SLuRp quickly identifies if a reviewer is falling behind and needs extra support with their allocated reviewing tasks. This allows progress to be monitored and schedules to be managed. SLR process information is displayed in bar charts. These are dynamically updated and always current (examples of these are given in Figure 2).

Ensuring data validity. SLuRp allows at least two reviewers to be allocated to review each paper. Each reviewer 'votes' on whether the paper meets the inclusion and exclusion criteria. Each reviewer then 'votes' on whether the paper meets the quality criteria. Finally each reviewer 'votes' on what data is extracted from an included paper. SLuRp will highlight any conflicts. All conflicts will then go into the moderation process. This is where the reviewers themselves first try to resolve a disagreement. Should this prove impossible the paper will be labeled as needing arbitration. A new reviewer will then cast a deciding vote. This rigorous process of voting at three different SLR stages, secures the validity of the findings reported.

Reporting SLR results. The graphical charts and tables produced by SLuRp can be used directly in the final report and write up of the review. SLuRP can act as a LateX editor which incorporates the graphical results and tables directly into the final report. This allows changes to the raw data to be automatically included in the latest version of the SLR.

Producing data on the SLR process. In reporting an SLR a variety of data about the SLR itself needs to be produced. This may include the number of papers that meet the inclusion/exclusion criteria, the number of papers failing particular inclusion/exclusion criteria, the proportion of papers that failed the quality check, the number of decisions that required moderation. The inter rater reliability score for extracted data. SLuRp can produce all of this data easily.

3. CONCLUSION

Performing a rigorous SLR which reports reliable results is difficult but essential. Many of the difficulties are related to the administrative complexities involved with managing and controlling any large complex project. Our experience is that in order to produce reliable valid results, more than one reviewer is required. Maintaining large amounts of data in a team with several reviewers is time-consuming and error-prone. These errors are difficult to identify and eliminate without the use of a specific SLR tool like SLuRp.

4. ACKNOWLEDGMENTS

We thank the UK's Engineering and Physical Science Research Council, which supported this research at Brunel University under grant EPSRC EP/E063039/1. We also thank the Science Foundation Ireland, which partially supported this work at Lero under grant 3/CE2/I303_1. Finally we thank Thomas Shippey for writing the code to extract bibtex files from online databases.

5. REFERENCES

[1] Dybå, T. and T. Dingsøyr (2008). *Empirical studies of agile software development: A systematic review.* Information and Software Technology, 50(9–10): pp.833-859.

[2] Cruzes, D.S. and T. Dybå (2011). *Research synthesis in software engineering: A tertiary study.* Information and Software Technology, 53(5): pp.440-455

[3] Morais, Y., T. Burity, and G. Elias. *A Systematic Review of Software Product Lines Applied to Mobile Middleware.* in Sixth Int. Conf. on Information Technology. 2009.

[4] Hall, T., S. Beecham, D. Bowes, D. Gray, and S. Counsell (in press). *A Systematic Literature Review on Fault Prediction Performance in Software Engineering.* IEEE Transactions on Software Engineering (TSE).

[5] Brereton, P., B.A. Kitchenham, D. Budgen, M. Turner, and M. Khalil (2007). *Lessons from applying the systematic literature review process within the software engineering domain.* Journal of Systems and Software, 80(4): pp.571-583.

[6] Stol K, Ali Babar M, Avgeriou P, and Fitzgerald B, *A comparative study of challenges in integrating Open Source Software and Inner Source Software.* Information and Software Technology, 2011. 53(12): p. 1319-1336.

[7] Lane, S. and I. Richardson (2011). *Process models for service-based applications: A systematic literature review.* Information and Software Technology, 53(5): pp.424-439.

[8] Beecham, S., N. Baddoo, T. Hall, H. Robinson, and H. Sharp (2008). *Motivation in Software Engineering: A Systematic Literature Review.* Information and Software Technology (IST), Elsevier, 50 (9-10): pp.860–878.

[9] Ramampiaro, H., D. Cruzes, R. Conradi, and M. Mendona. *Supporting evidence-based Software Engineering with collaborative information retrieval* in Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on. 2010, 9-12 Oct. 2010.

[10] Hall, T., S. Beecham, N. Baddoo, H. Sharp, and H. Robinson (2009). *A Systematic Review of Theory Use in Studies Investigating the Motivations of Software Engineers.* ACM Transactions on Software Engineering and Methodology (TOSEM), 18(3).

[11] Zhang, M., T. Hall, and N. Baddoo (2011). *Code bad smells: a review of current knowledge.* Journal of Software Maintenance and Evolution: research and practice, 23(3): pp.179- 202.

[12] Kitchenham, B. and S. Charters (2007), *Guidelines for performing systematic literature reviews in software engineering (version 2.3)*, in *EBSE Technical Report*, Keele University, UK.

[13] Felizardo R K, Nakagawa E Y, Feitosa D, Minghim R, and Maldonado D C. *An Approach Based on Visual Text Mining to Support Categorization and Classification in the Systematic Mapping*. in *14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. 2010. Keele University, UK.

[14] Malheiros V D, Hohn E, Pinho R,. Mendonça M, and Maldonado J C. *A Visual Text Mining approach for Systematic Reviews. In: Empirical Software Engineering and Measurement*. in *First International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2007. Madrid, Spain: IEEE Computer Society.