Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Tool Support for the Search Process of Systematic Literature Reviews

Dominik Voigt

Course of Study:	Informatik
Examiner:	Prof. Dr. Dr. h. c. Frank Leymann
Supervisor:	Karoline Wild, M.Sc.
Commenced:	May 4, 2020
Completed:	November 4, 2020

Abstract

Systematic Literature Review (SLR) is a popular research method with adoption across different research domains that is used to draw generalizations, compose multiple existing concepts into a new one, or identify conflicts and gaps in existing research. Within the Information Technology domain, researchers face multiple challenges during the search step of an SLR which are caused by the use of different query languages by digital libraries. This thesis proposes tool support that provides cross-library search by using a common query language across digital libraries. For this the existing digital library APIs and query languages have been analyzed and a new cross-library query language and transformation was developed that allows the formulation of cross-library queries that can be transformed into existing query languages. These concepts have been integrated within an existing reference management tool to provide integrated and automated cross-library search and result management to address the challenges faced by Information Technology researchers during the search step.

Contents

List of Figures

List of Tables

Acronyms

AST abstract syntax tree. 23

CS Computer Science. 29

EBSE Evidence-based Software Engineering. 15

FMC Fundamental Modeling Concepts. 43

IR Information Retrieval. 29

IT Information Technology. 13

SE Software Engineering. 14

SLR Systematic Literature Review. 7, 13

1 Introduction

The Systematic Literature Review (SLR) is an evidence-based research method that uses a systematical approach to answer research questions [Kit07]. Like other types of studies, SLRs follow a process with multiple steps. The main steps within that process are the search and selection steps that identify relevant studies, the data extraction step that extracts evidence from the identified studies, and the data synthesis step that uses the extracted data to answer the posed research questions. What makes the SLR a systematic approach is the fact that a predefined research protocol is followed. This research protocol is formulated during the planning of the SLR and describes how each step of the SLR is conducted. For the selection step, for example, it defines the criteria that are applied to each retrieved study to determine whether it is relevant or not. If the research protocol is faithfully followed it makes the research more repeatable, traceable, and less biased than non-systematic apporaches like a traditional literature review [KB13; Kit07]. These properties allow researchers to draw generalizations, compose multiple existing concepts into a new one, or identify conflicts and gaps in the existing research. Due to that, this method has seen a wide range of adoption across different research domains [Kit07].

But the systematic approach, based on a research protocol, also requires much more effort. The reason for this is that each step of the SLR process has to fulfill specific requirements to obtain the benefits of the SLR method. To reduce the effort required to conduct an SLR without reducing the benefits a large set of tools[1] exists. The most specialized tools are so-called SLR tools that try to support every step of an SLR and provide SLR-specific features, such as definition and application of selection criteria for the selection step.

However, especially within the domain of Information Technology (IT) research, researchers still face challenges during the conduction of SLRs [AC18; KB13]. Notably during the search step many challenges are encountered by IT researchers [AC18]. The goal of the search step is to identify all studies that are relevant to the research topic, i. e., all studies that might provide evidence relevant to the research questions. It is of utmost importance that this search is complete, and traceable, as otherwise relevant evidence that might influence the outcome of the SLR will be missed, or the result will not be replicable [KBL+11]. To achieve this completeness, and traceability the search has to be conducted systematically by following a search protocol. This protocol define the queries used in the search and all digital libraries that are searched [Kit07]. It is part of the research protocol and has to be included in the SLR study to make it replicable [KBL+11]. The main source of these challenges encountered in the search step within the domain of IT research is the fact that the digital libraries that contain IT-related studies do not provide adequate search features to satisfy the upper mentioned requirements of the search step [AC18; KB13]. Their main shortcomings are the use of different query languages by each digital library, providing no batch access to results, or providing

[1] A comprehensive catalog of support tools for SLRs is provided by the SLR Toolbox [MB15a] under http://systematicreviewtools.com/

no filtering of results [AC18; HCHA16; HCKH14]. Especially the use of different query languages poses a challenge during the search step. The reason for this is that for each query language the used query has to be adapted. This introduces a lot of additional manual effort to ensure the use of equivalent queries across different digital libraries. This additionally promotes unsystematic conduction of the search due to ad-hoc changes to the query string by the research team for specific libraries to achieve satisfactory search results [AC18].

To address these challenges tool support is required [AC18]. However, the existing IT specific SLR tools, which are all in fact Software Engineering (SE) specific that aim to support all steps within the SLR process fail to support the search process sufficiently by not addressing these challenges. Therefore, the objective of this thesis is to provide tool support to enable integrated and automated search across different digital libraries using the same query, and extraction and management of results to reduce the effort required during the search step. For this, the search features of the relevant digital library APIs have been analyzed and based on this analysis a cross-library query language was developed. With this, a transformation mechanism was developed that transforms cross-library queries into semantically equivalent queries in the targeted digital library query language. Finally, these two concepts are integrated as a so-called *Meta Searcher* into an existing reference management tool to provide the above-mentioned features.

The remainder of this thesis is structured as follows: Chapter 2 provides the background required to understand the rest of the thesis. This includes the basic SLR method, including all of the necessary steps, the challenges encountered during the search for literature, and the basic concept of query languages and transformation. In Chapter 3 related work is introduced. Here approaches are discussed to address the challenges encountered during the search process, SE specific SLR tools, and tools that provide an integrated search of IT-related digital libraries. Chapter 4 proposes a cross-library query language that can be used to formulate queries that can be transformed into semantically equivalent queries for IT-related digital libraries. In this chapter, an analysis of the capabilities of IT-related digital libraries is also provided. Chapter 5 proposes the Meta Searcher a tool component that leverages the cross-library query language proposed in Chapter 4. This includes a process model of the Meta Searcher supported search process, the component-based conceptual architecture of the Meta Searcher integrated in an existing reference management tool, and how each step of the introduced process model is supported by this integrated Meta Searcher. Chapter 6 discusses the implemented prototype, how the relevant aspects were realized, and the limitations of the prototype. Chapter 7 discusses the results of the thesis and discuss which further steps would need to be taken to advance the presented work.

2 Background and Fundamentals

This chapter introduces all fundamentals that are required to understand the concepts in this thesis. First, the general SLR process, which is the basis of this work, is introduced in Section 2.1.1. Then the literature search process is introduced in Section 2.1.2, as this is the process that will be supported within this thesis. This section also introduces the challenges associated with the search process.

In Section 2.2 the BibTeX, and BibLaTeX metadata formats, used to store bibliographic metadata, are introduced. These formats are the defacto standard to store and exchange metadata of publications. Finally, in Section 2.3 the basic concepts of query languages are introduced, including the common syntactic constructs, as well as their semantic interpretation.

2.1 Systematic Literature Review Process

An SLR is an evidence-based research method with interdisciplinary adoption [KDJ04]. An SLR allows a researcher to identify, aggregate, and interpret all research related to a specific topic or research question in an unbiased manner [Kit07]. Since an SLR examines existing research, called primary studies in this context, it is referred to as a secondary study [Kit07]. An SLR can include both quantitative and qualitative primary studies depending on the discipline and topic. In SE for example, most primary studies are qualitative [DDH07].

Each SLR follows a systematic process, defined as a protocol, that is specified before the review is conducted. This approach reduces the influence of researcher bias that can occur without a structured approach and makes these reviews repeatable, and therefore more impactful [BMNT05]. This allows researchers that conduct SLRs to (i) draw generalizations, (ii) identify gaps in existing research, (iii) find, and investigate conflicting results of primary studies [Kit07], and (iv) synthesize new theoretical concepts out of the concepts of primary studies [DDH07]. In Evidence-based Software Engineering (EBSE), where SLRs are a key research method, studies have shown that SLRs can achieve these results [BZ09; KBB+09]. These possibilities make SLRs a powerful complementary research method for primary studies [BMNT05].

2.1.1 SLR Process Model Overview

Since SLRs within a research domain follow a similar process, there exist domain specific guidelines for conducting SLRs. As this thesis focuses on SLRs within the domain of IT the guideline and process that is used in this domain is the basis of this work. One commonly used guideline within the IT domain is [Kit07], with some updates amended in [KB13]. This guideline originated from the SE domain, where it is one of the most used guidelines [BZ09]. However, it is also applicable to the field of IT research, as shown by recently published SLRs e. g., in [BBF+18; FML17; YBLW19].

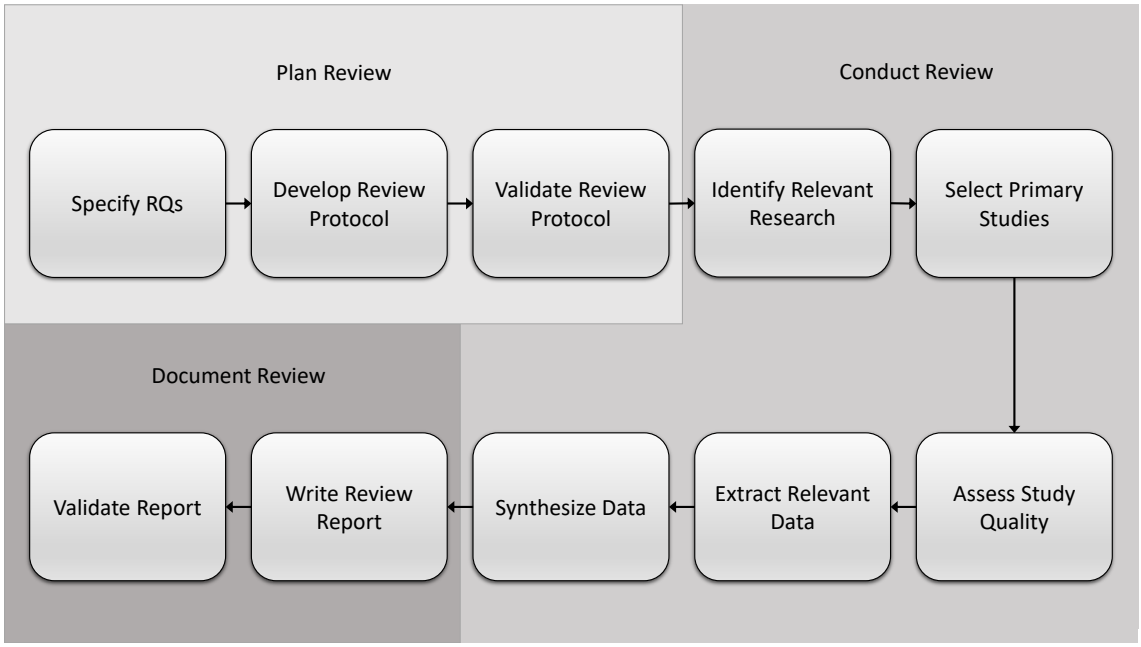

Figure 2.1: SLR process proposed by Brereton et al. [BKB+07]

To these guidelines, a corresponding SLR process model was proposed by Brereton et al. [BKB+07], which is co-authored by Kitchenham. Therefore, the proposed SLR process model by Brereton et al. [BKB+07] is the basis for this thesis. The process is depicted in Figure 2.1. In the following, each step is described in detail. Each of the ten steps is a sub-process. The detailed process of step four is introduced in the next section.

The actual first step is omitted by the model since first the research team should identify whether there exists a need for an SLR in the first place. This need exists if there either exists no SLR that fulfills the desired objective, or if the existing SLRs are of insufficient quality [Kit07]. If there exists a need for an SLR and the objective is of significant worth, a research team can begin the SLR process by defining its research questions in the first step (see Figure 2.1). These questions formulate the objective of the SLR. This step is important as it influence the rest of the SLR process [BKB+07]. Thus, the research questions should not be modified after the end of the planning phase. An exception to this would be if the researchers have to remove questions because there is not enough evidence to answer it.

In the second step, the review protocol is defined by the research team. In this protocol, the research team specifies how they will systematically conduct the review. It is used to define how the research team will execute each step in detail. The faithful execution of the research protocol enables SLRs to be replicable. The review protocol consists of the following parts defined by Kitchenham [Kit07]:

- Background: The rationale for the survey.
- The research questions that the review is intended to answer.
- Search Strategy: The strategy that will be used to search for primary studies including search terms and resources to be searched. Resources include digital libraries, specific journals, and conference proceedings. An initial mapping study can help determine an appropriate strategy.
- Study selection criteria: Study selection criteria are used to determine which studies are included in, or excluded from, a systematic review. It is usually helpful to pilot the selection criteria on a subset of primary studies.
- Study selection procedures: The protocol should describe how the selection criteria will be applied e.g. how many assessors will evaluate each prospective primary study, and how disagreements among assessors will be resolved.
- Study quality assessment checklists and procedures: The researchers should develop quality checklists to assess individual studies. The purpose of the quality assessment will guide the development of checklists.
- Data extraction strategy: This defines how the information required from each primary study will be obtained. If the data require manipulation or assumptions and inferences to be made, the protocol should specify an appropriate validation process.
- Synthesis of the extracted data. This defines the synthesis strategy. This should clarify whether or not a formal meta-analysis is intended and if so what techniques will be used.
- Dissemination strategy (if not already included in a commissioning document).
- Project timetable: This should define the review schedule.

Regarding the search strategy, Dieste and Padua [DP07] provides further insight into the aspects that should be considered when choosing a search strategy. Skoglund and Runeson [SR09] evaluated a reference-based search approach as the main search strategy and concluded that this might be a valid approach for specific topics. Overall it can be concluded that the reference-based search should be used complementary [KB13]. During the construction of the protocol, it might make sense to revise the research questions as the understanding of the research team regarding the problem increases [BKB+07].

During the third step, the protocol is validated. This is achieved by piloting of the review protocol to spot any issues with the procedures to collect studies or extract data to answer the research question [BKB+07]. If any issues are found these have to be addressed, especially if they are concerned with the answerability of the research questions [BKB+07; Kit07]. This shows that one aspect that is not captured by the process model is the fact that during all steps in the process, the review protocol is iteratively adapted if new knowledge over the problem is gained [Kit07].

After the third step is done, the review can be conducted. This second phase starts with the execution of the search strategy to collect relevant research in the fourth step. This step is also called the *search step* [KB13]. The systematic execution and completeness of the search process are essential to achieve the benefits of SLRs [AC18]. It is important to document the conducted searches of all sources. For each search of a digital library, it includes (i) the name of the digital library (ii) the search strings used (iii) the date of the search, and (iv) the years covered by the search. This documentation is necessary to make the search step and the achieved results transparent and repeatable [Kit07].

In the fifth step, after an iteration of the search process is completed, the previously defined inclusion/exclusion criteria are applied to the retrieved studies. This step is called the *selection step*. This can be a multi-tiered pruning process, e. g., [BBF+18], where selection criteria are applied sequentially. To reduce researcher bias it is important to conduct this study selection process by multiple researchers and discuss in the case of disagreements [Kit07].

In the sixth step, the selected studies are assessed for quality. This assessment can be used to remove studies that are not of sufficient quality or can be used to identify why certain studies might have contradictory results. An example of this is, in the case of qualitative studies, that one study misses an important study that leads to differentiating results [KBL+11]. Another reason for this assessment is to decide how reliable and impactful the result of a study is, and how much the result of the SLR should depend on this particular study result.

The assessment of quality should be based upon so-called quality instruments. These are checklists to evaluate each study's quality systematically. It is important to note that both qualitative as well as quantitative studies have to be assessed for quality using quality instruments. For both study types, separate quality instruments have to be used, as not all questions within the checklist apply to both types [Kit07].

During the seventh step, the researchers use the data extraction form to extract data from the studies to answer the research questions. The extraction form includes [Kit07]:

- Name of reviewer
- Date of data extraction
- Title, authors, journal, publication details
- Questions that have to be answered by the study to extract data
- Space for additional notes

The use of a form with the above-mentioned items makes the data extraction transparent. Especially the questions contained in the extraction protocol have to be piloted to ensure that they extract the data necessary to answer the posed research questions. To reduce bias, the extraction should be conducted by multiple researchers independently. In this case, the form is additionally used to spot inconsistencies in the extraction results. If due to a large number of papers it is not feasible to do the extraction by more than one researcher, Brereton et al. [BKB+07] propose one researcher to act as a data extractor, and one researcher to act as a data checker. This proposal reduces the effort

needed for data extraction and retains the consistency of the systematic data extraction process. As mentioned above, during the data extraction process, researchers might iteratively adapt the data extraction form [Kit07]

In the eighth step the data extracted from the selected primary studies are summarized and analyzed to synthesize an answer to the posed research questions. The methods used to achieve this depends on the type of primary studies used.

If the underlying primary studies are qualitative the review is called an integrative review [DDH07], and the synthesis is descriptive. In this case, the data extracted from the primary studies should be visualized in a tabular manner to show heterogeneity or homogeneity of results. If the underlying primary studies are quantitative the review is called an interpretive review, the synthesis is quantitative [DDH07]. In this case, the data has to be statistically analyzed and appropriately visualized.

In the ninth step the synthesis of results is concluded, and the research team formulates their final report. During this step, it is important to keep the target audience in mind, as the right venue has to be selected to publish the results, and reach the intended audience. Keeping this in mind during the formulation of the report is important because publication venues commonly impose formatting restrictions on the document that might be hard to fulfill when reporting an SLR. Size limitations for example are especially critical, as the report of the review protocol and systematic presentation of evidence requires considerable amounts of space. In the tenth step the report is reviewed by external researches to assess the quality and validity of the reported SLR [Kit07].

2.1.2 The Search Step Process

The search step is the first step conducted during an SLR. To introduce the search step in-depth a variant of the process of this step is introduced and discussed. As the search and selection of relevant studies go hand in hand these two steps are often planned together. Thus, the processes of these steps are often depicted in a concatenated manner. Therefore, the introduced process also contains the processes of both steps.

The search and selection process used by Kitchenham and Brereton [KB13] is a process that is close to the guidelines introduced in 2.1.1. Therefore, it can be seen as a best-practice process. However, due to the sophisticated nature of this process, it is often adapted in a simplified manner. An example of such a simplified adaption can be found in Francesco et al. [FML17] and Yussupov et al. [YBLW19]. This process is simpler but still sufficiently detailed to provide context for the tool supported search step that is discussed in Chapter 5. Therefore it is used for this thesis. This search and selection process used by Francesco et al. [FML17] and Yussupov et al. [YBLW19] is depicted in Figure 2.2. In this process, the authors search multiple digital libraries in the first step, which are recommended by guidelines, using the same search strings for each digital library. In both SLRs, a general search string was used to increase the sensitivity of the search. In the second step, the papers are filtered based upon the title and abstract. During this step also all publications that are not studies, e. g., textbooks, are discarded [FML17]. In the third step, the selected results from all digital libraries are de-duplicated and merged into the candidate set. In the fourth step, the selection criteria are applied to the papers in the candidate set. This step might also include the quality assessment of the paper if this is used as a selection criterion. This can be a multi-tiered pruning process, e. g., [BBF+18], where selection criteria are applied sequentially. In the fifth step, a reference-based search is used on the existing papers in the result set. With this other relevant

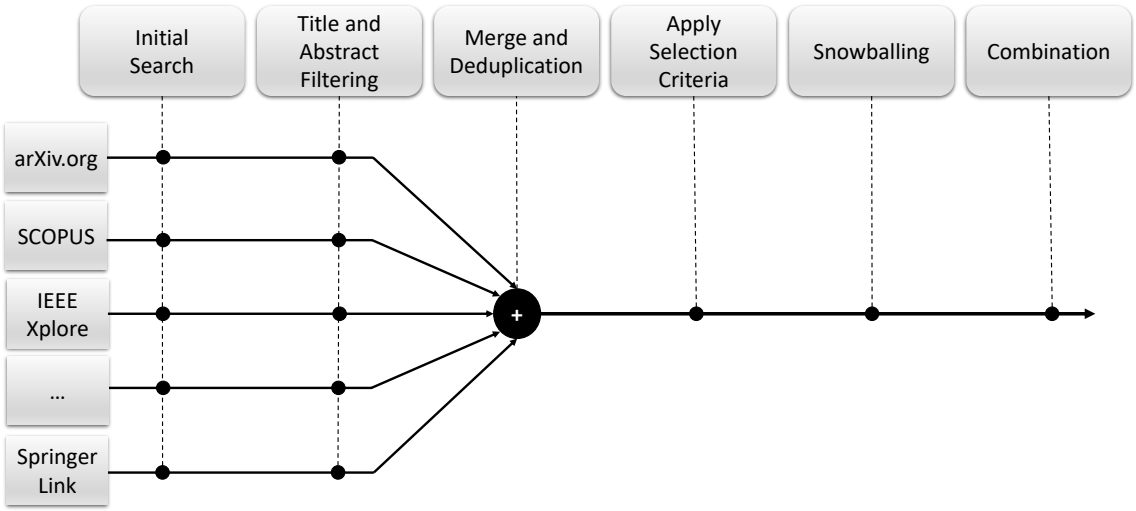

Figure 2.2: Search and selection process used by Francesco et al. [FML17] and Yussupov et al. [YBLW19]

papers can be identified that have been missed during the automated initial search. All new studies are then put through the selection process. In the sixth step, all papers that are reiterations of the same concept are combined and represented as its latest iteration.

In this process, the first and fifth steps are part of the search step. The de-duplication and merging can also be seen as part of the search step as it is part of the result management. The remaining steps are part of the selection step. This shows that the search and selection steps are often executed concurrently.

2.2 Metadata Formats for Literature Reference Management

Metadata formats are used to describe a resource in a consistent format. To achieve this they specify certain standardized fields that can be used to describe certain properties of the resource, e. g., a title field for the title of a publication [Key12]. Since the metadata format specifies what aspects of a resource can be described, the appropriate metadata format is dependent on the use case. A general modern metadata standard for a lot of different resources is the DCMI Metadata Terms [1]. For publications, such as books, articles, or studies, however, the BibTeX [2] and BibLaTeX [3], two bibliographic metadata formats, are the de-facto standard. This means that most digital libraries

[1] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[2] http://www.bibtex.org/
[3] https://www.ctan.org/pkg/biblatex

provide BibTeX/BibLaTeX metadata entries and reference management tools can work with these entries. Therefore, the Meta Searcher proposed in this thesis also supports the BibTeX/BibLaTeX metadata format to ensure interoperability with other tools within the domain.

The BibTeX format specifies a syntax that each of the entries has to follow, the BibTeX syntax is introduced here with an example entry shown in Listing 2.1. In this thesis only the BibTeX syntax is introduced, as BibLaTeX uses the same syntax and differentiates from a format viewpoint solely in the supported fields and entry types.

```
    @Article{Marshall2018,
      author = {Marshall, Chris and Kitchenham, Barbara and Brereton, Pearl},
      journal = {e-Informatica Vol. XII},
      title = {Tool Features to Support Systematic Reviews in Software Engineering - A Cross
Domain Study},
      year = {2018},
      doi = {10.5277/E-INF180104},
      publisher = {Institute of Applied Informatics, Wroclaw University of Technology, Wroclaw
},
    }
```

Listing 2.1: An Example BibTeX Entry

Each entry begins with an entry type specified after the "@". In Listing 2.1 this is the Article entry type. The entry types are used to determine the required and optional fields that are used to describe a type of publication. The required fields are specified to define which information is necessary to properly format the entry of the specified type. The optional fields contain further information that can be used to further specify the entry, but they are not necessary for proper formatting. For the depicted Article entry the required fields are the author, title, journal, and year fields[4].

After the entry type, the cite key is specified, here it is Marshall2018. The cite key is a value used to uniquely reference the specific entry within the local database of bibliographic entries. After that, each entry contains a comma-separated list of fields. As depicted in Listing 2.1, each field is a key-value pair. The BibTeX format allows the addition of arbitrary fields, e. g., the DOI[5] and publisher fields specified in Listing 2.1, these fields are ignored for formatting and can be used to store additional information.

2.3 Queries on Digital Libraries

For searching publications, different digital libraries exist. As most publishers offer their own digital libraries, e. g., Springer Nature offers Springer Link, it is essential to search multiple digital libraries to ensure that a search is complete. Indexing digital libraries that index other digital libraries, e. g., Google Scholar or Microsoft Academic, can be used to reduce the number of digital libraries, but cannot eliminate that need completely [KB13].

[4] https://www.openoffice.org/bibliographic/bibtex-defs.html
[5] https://www.doi.org/

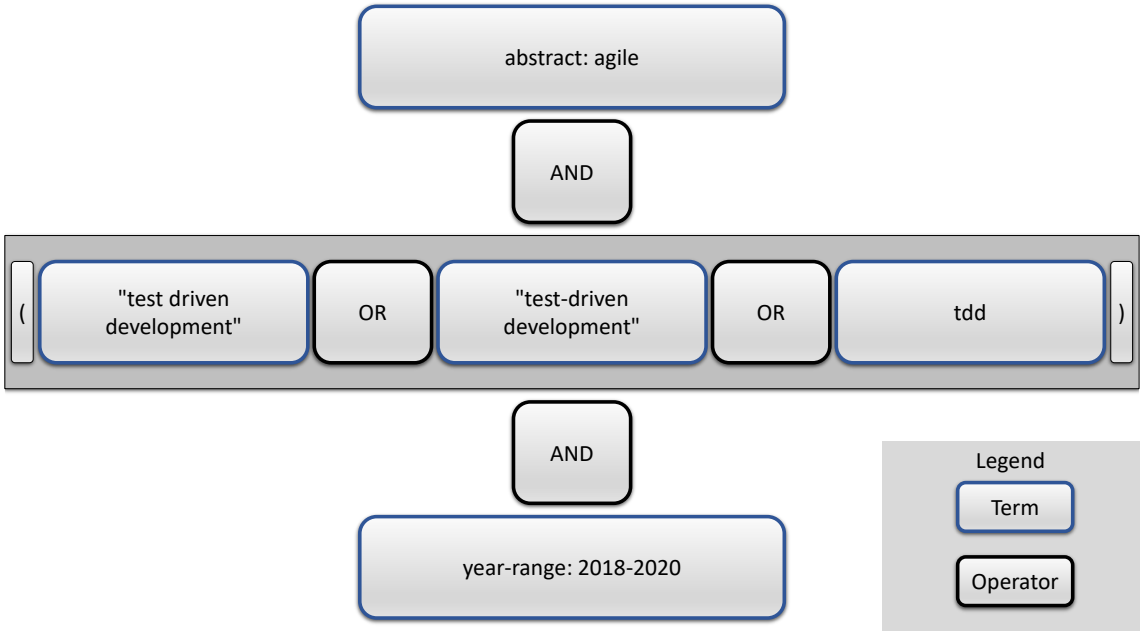

Figure 2.3: An Example Query

All digital libraries offer GUI interfaces that can be used to search for publications. To identify relevant publications the user provides the digital library with a query. Queries have to be tuned, and piloted, with regards to their sensitivity, and precision to deliver satisfactory search results [DP07; Kit07], a step essential to the SLR method. An example query is depicted in Figure 2.3. This can either happen manually over the GUI or automated over an API if available. As the digital library has to interpret the query it has to be formulated in a specific query language that is defined by the digital library (a *digital library-specific query language*). This query language defines the allowed syntactic constructs and their semantic interpretation for each query.

Each query consists of terms and operators. In the example query, the terms are the blue-rimmed boxes and the operators are the black-rimmed boxes. In most query languages terms can either be simple terms, or phrases that are commonly wrapped in quotes. In the example the fourth term is a simple term and the second and third are phrases. Phrases are necessary if a search should find publications that contain multiple terms in a specific sequence, e. g., in the case of "test driven development" the term matches all publications that contain the three terms (test, driven, development) in that sequence. Furthermore in most query languages exist so-called fielded terms. These are also terms that only match within a specific metadata field. The supported fielded terms and syntax for fielded search is different across digital libraries. One commonly used variant is depicted in the example. Here the first and last term are fielded terms, where the actual term is preceded by a field identifier and a ":". The fielded term "year-range: 2018-2020" in the example matches all publications that contain a value between 2018 and 2020 in the year/date field (BibTeX/BibLaTeX), e. g., the article depicted in Listing 2.1 would match the fielded term. Another special property that is exemplified by this year-range-fielded term is the fact that there exist range queries for comparable fields such as dates. This means that such a term matches any publication that contains a value in the specified field that lies within the specified range.

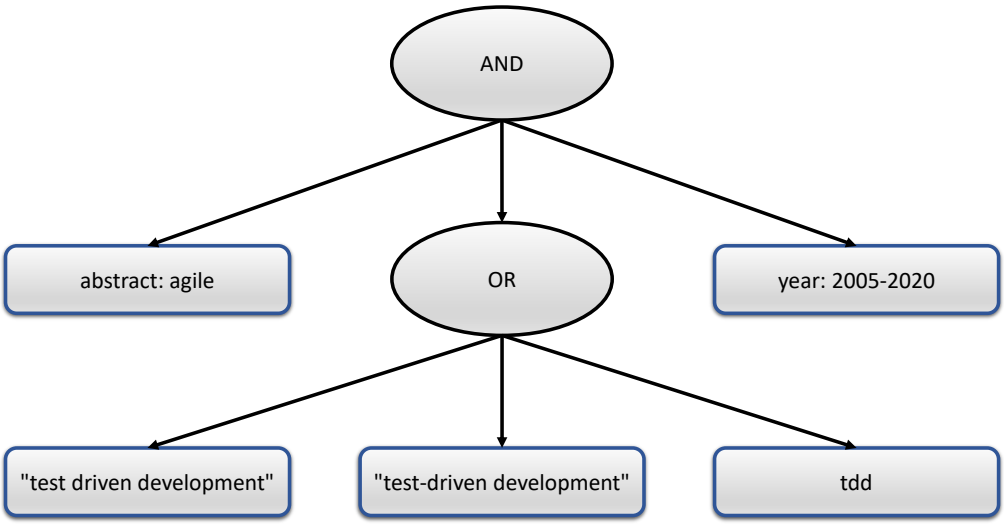

Figure 2.4: Simplified AST representation of the example query

Operators in the context of digital library-specific query languages are standard Boolean operators (AND, OR, NOT). These operators have a certain precedence that defines their order of evaluation, e. g., AND is evaluated before OR. It is important to note that different query languages support different operators fielded terms, and use a different query syntax to express the semantically same query.

Terms and operators can be grouped using so-called groupings. These groupings can modify the order of evaluation. The groupings and the operators define the logical structure of a query. This logical structure defines the order of evaluation for a query. In the example, the second row of the query is grouped using parentheses. Due to this grouping, the OR operators within the grouping are evaluated before the AND operators are evaluated even though normally the AND operator has higher precedence.

Due to this order of evaluation, it is essential to analyze the logical structure of a query for a transformation, as the structure of a query influences its semantic meaning. A query can be represented structurally as an abstract syntax tree (AST)[6]. The AST representation of the example query is depicted in Figure 2.4. The inner nodes of the AST are operators that are applied to all of its child nodes. The leaf nodes of the tree are terms.

The AST representation of the query is evaluated from the leaf nodes upwards, and thus each subtree in the AST represents a subexpression that is evaluated before its parent expression is evaluated. In the example query, the middle row was grouped using parentheses and is therefore a subexpression of the complete query expression. Therefore, this grouping is represented in the AST as the subtree that consists of the inner OR node in the middle and the corresponding term leaf nodes at the bottom.

[6]For an example see https://lucene.apache.org/core/8_6_1/queryparser/index.html

The use of such a structural representation makes groupings of terms and operators and the corresponding order of evaluation easily accessible within programs by traversing the tree.

2.4 Challenges during the Search Process

Even though the SLR method provides great benefits, conduction of such a study comes with several challenges and requires significant amounts of effort [AC18; CHHK13; HCKH14; KB13; RCCM10; RSSM10; SN07].

The main sources of effort are activities imposed by the SLR method, e. g., assessment of the quality of candidate studies. These activities are essential, as they add value to the method, by providing the benefits introduced in Section 2.1. The challenges encountered during the conduction of an SLR on the other hand should be addressed, as they introduce additional effort without providing any benefit to the SLR method.

Especially during the search step a significant amount of challenges are currently encountered [AC18]. Therefore this section discusses the challenges, and the resulting requirements to address them. This will be used to propose an appropriate tool support approach in Chapter 5.

Al-Zubidy and Carver [AC18] conducted an SLR and survey to identify the challenges encountered during the search process. Since Al-Zubidy and Carver [AC18] is the most recent SLR and survey regarding the search process and since challenges might change over time, it is the basis for this section.

In the SLR Al-Zubidy and Carver conducted, they identify a total of 29 unique challenges within literature. They report that 72% of the challenges are encountered during the conduction of an SLR. For example, Nidhra et al. [NYAT13] reports that a separate search string had to be used for IEEE Xplore due to search string length limitations, rather than challenges identified in the SLR process. Al-Zubidy and Carver [AC18] therefore argue that by addressing the identified challenges researchers would directly benefit from it. Moreover, they report that 85% of challenges are related to digital libraries, arguing that digital libraries have to adapt and that SLR tools are required to address some of these challenges. By mapping the sets of challenges identified in the survey and SLR onto each other Al-Zubidy and Carver [AC18] conclude that the challenges identified in literature are still up to date.

The top challenges identified are a result of the merging of the top ten challenges identified in the SLR, and survey are:

(i) Different syntax across digital libraries

(ii) Some digital libraries do not search full text

(iii) Better interfaces to search digital libraries are required (Inconsistent interfaces even within the same digital libraries)

(iv) Unable to download large batch of results automatically

(v) Limitation on complex strings in digital libraries

(vi) Limitation on the search string

(vii) Unable to conduct automated search on the interfaces of the digital libraries

(viii) Same digital library produce different results based on the order of the words

(ix) Digital libraries return a lot of irrelevant results

(x) Limited access to papers

(xi) Inconsistent terminology

(xii) Search string limitations

(xiii) Filter limitations

These challenges either have to be addressed, as otherwise the benefits provided by the SLR method cannot be ensured [KBL+11]. Al-Zubidy and Carver [AC18] identify the methods employed by SLR practitioners to address or circumvent these challenges. Al-Zubidy and Carver [AC18] also described the weaknesses of these approaches. The most common methods that have been used to address the challenges are [AC18]:

- **Iterative Search**: Iteratively modify search string until a satisfactory result is reached. The problem with this is the non-systematic nature of the method, leading to an unsatisfactory search result.
- **Alternative Sources**: Use of an alternative source if the original source has limitations with unsatisfactory implications, e. g., missing filters leading to greatly increased result set. The problem with this is that the alternative resource might not contain all relevant publications of the original resource.
- **Different Search Strings**: Use of digital library-specific search strings. The problem with this is that this results in a non-systematic search, and might lead to semantically in-equivalent search strings. This might compromise the quality of the search result, e. g., missing papers of a certain digital library due to an inconsistent, and insensitive digital library-specific search string.
- **Tool**: Use of tools to aid in the search process. The problem with this is that the documentation how exactly the tool was deployed often misses and that the existing tools do not provide adequate support to address the identified challenges.

Relevance	Requirement	Responsibility
(i)	Search multiple databases in standardized query	Digital library, Tools
(ii)	Removing duplicates	Tools
(iii)	Filtering capabilities	Tools
(iv)	Merging results from digital library	Digital library, Tools
(v)	Synonym recommendation	Tools
(vi)	Repository for papers	Digital library, Tools
(vii)	Standardized export	Digital library, Tools
(viii)	Download papers	Digital library, Tools
(ix)	Automatic determination of relevance	Digital library, Tools
(x)	Support snowballing	Digital library, Tools
(xi)	Automatic generation of search terms	Tools
(xii)	Support collaboration	Digital library, Tools
(xiii)	Papers categorization	Digital library

Table 2.1: Top requirements identified by Al-Zubidy and Carver [AC18], sorted by relevance

This shows that none of the existing workarounds can address the challenges encountered without reducing the benefits provided by the SLR method. This leads to the following issues:

(i) Synonyms have to be considered when constructing search strings due to the non-standardized terminology used by researchers [DP07].

(ii) Queries have to be transformed to conform to the digital library-specific query language [AC18].

(iii) Transformed queries might not be semantically equivalent due to inconsistent interfaces [AC18].

(iv) Transformed queries might lead to unexpected results as the Boolean operators, and the order of words is inconsistently evaluated [DDH07].

(v) Due to (iv), researchers conduct ad-hoc modifications to their search strings, to achieve satisfactory results, losing repeatability and traceability in the process [AC18; KBL+11].

(vi) Abstracts or full-text papers cannot easily be accessed over the provided API or GUI [DDH07; HCKH14].

(vii) Duplicates returned by multiple sources, have to be filtered [DDH07].

To develop and evaluate approaches to address these challenges a set of requirements is needed. Al-Zubidy and Carver [AC18] report a total of 162 requirements that have been provided by survey practitioners. The top results of this analysis are displayed in Table 2.1. In the left column, all requirements are listed that were reported more than three times. For each requirement, Al-Zubidy and Carver [AC18] specify whether a digital library, tool, or researcher is responsible for fulfilling the requirement. In the right column, the responsibility bearer for the requirement in the same row is listed. The requirements are arranged in descending order by the frequency of being reported.

Marshall and Brereton [MB15a] conducted a cross-domain survey to identify the current status of tool support for the SLR method in other research areas. The participants conducted research within the Healthcare or Social Science domain. These reported requirements are overall consistent with the ones identified by Al-Zubidy and Carver [AC18]. One differentiating aspect is the fact that collaboration was ranked higher by cross-domain researchers in the survey conducted by Marshall and Brereton [MB15a] than IT researchers in the survey conducted by Al-Zubidy and Carver [AC18]. Marshall et al. [MKB18] conducted a similar survey to Marshall and Brereton [MB15a] and also reached consistent results. Hassler et al. [HCHA16] conducted a community workshop to identify feature requirements for SLR tools. The participants voted on the importance of each feature, identifying similar feature requirements than the previous studies: (i) integrated search, (ii) Collaboration, and (iii) Traceability.

These studies identify a need for tool support that enables collaborative and traceable studies [AC18; HCHA16; MB15a]. It seems that the challenges encountered during the search process are still not sufficiently supported by SLR tools, as Al-Zubidy and Carver [AC18] and Hassler et al. [HCHA16] identify the same feature requirements to address these challenges. However, this does not seem to be because these challenges cannot be addressed, but rather are just not addressed by current tools, as Al-Zubidy and Carver [AC18] conclude in their study:

- The identified challenges are solvable. To achieve this the requirement bearers have to fulfill their identified requirements.
- SLR Tools still lack support for the search process. Especially features such as query transformation into the appropriate library syntax, standardized format for import/export of studies, and synonym recommendation.
- Certain challenges of the search process can be circumvented but might degrade the quality of the search. To achieve satisfactory results multiple search approaches have to be combined.
- SE specific SLR tools have been reported less than general-purpose tools, like spreadsheet tools. This might be since these tools have lacking or none search support. There still exists a need for tool support that fulfills the identified requirements. This argument is further supported by Hassler et al. [HCHA16], as they identify the integrated search feature as a key-feature for SLR tools.

3 Related Work

This chapter discusses scientific publications and tools in the context of support for the search process of the SLR method. In Section 3.1 approaches proposed in literature to address the challenges encountered during the search process are introduced and discussed.

Section 3.2 discusses existing SLR tools, focusing on their support for the search process. This section also discusses publications that compare and evaluate the different SLR tools are discussed. As the set of available support tools for SLRs is too large for an in-depth introduction[1], and my affiliation with Computer Science (CS), this thesis focus on tools that are targeted to support researchers in the field of CS and SE. Another reason is that tools from other disciplines do not support search for CS and SE digital libraries, and are therefore less suited to support the search process. Finally, in Section 3.3 tools that provide integrated search of digital libraries are discussed.

3.1 Existing Approaches

To address the challenges encountered during the conduction of SLRs (see Section 2.4), SLR authors propose different solutions. To reduce the effort when conducting a search of multiple digital libraries, Ramampiaro et al. [RCCM10], Riaz et al. [RSSM10], and Staples and Niazi [SN07] advocate for the development of a common search interface for all digital libraries. This approach also makes the search traceable using a search log. The issue with the proposal of Ramampiaro et al. [RCCM10] is that they fail to provide the concepts necessary to realize it. This thesis tries to provide a variation of the Meta Searcher that Ramampiaro et al. [RCCM10] propose, that includes the concepts required to realize it, as well as a discussion of how the proposed Meta Searcher supports the search process.

Other approaches include the use of Information Retrieval (IR) technology to reduce the effort required for the search process by making the data more structured and accessible. Felizardo et al. [FMMM12], Kitchenham and Brereton [KB13], Marshall and Brereton [MB13], and Ramampiaro et al. [RCCM10] propose clustering, and classification of publications by digital libraries to simplify the search effort, e. g., k-means [RCCM10]. Ramampiaro et al. [RCCM10] furthermore propose the use of automated text summarization to extract the core concepts of a paper, greatly reducing the effort required to decided whether to include the paper, while increasing the quality of the selection process. This is relevant for the search process as the text summarization should occur during the search process as part of the post-processing as the search process precedes the selection process (see Figure 2.2). With this, the effort required for manual and automated search and the

[1] A comprehensive catalog of support tools for SLRs is provided by the SLR Toolbox [MB15a] under http://systematicreviewtools.com/

following selection can be greatly reduced. Researchers could easily find relevant papers just by identifying a small set of papers through manual search. Furthermore, the number of false positives would be greatly reduced, and the application of inclusion/exclusion criteria would be simplified by the summarization. The problem with these approaches is the fact that this cannot be done by the research community itself but by the digital library providers. Nonetheless, this approach is promising and Microsoft Academic already incorporates some of these techniques.

Another important aspect of the SLR method is collaboration, as Al-Zubidy and Carver [AC18] and Hassler et al. [HCHA16] identify in their studies. This is especially important during steps that can be strongly influenced by researcher bias, such as the search and selection processes [Kit07]. To address this Carver et al. [CHHK13], Hassler et al. [HCHA16], and Staples and Niazi [SN07] propose the use of collaborative SLR tools.

It becomes apparent that there exists a multitude of challenges (see Section 2.4) that are encountered during the search process and that no currently used countermeasure delivers satisfactory results (see Section 2.4). There are different proposals on how to address these challenges, ranging from ideas that can be realized by individuals to ideas that can only be realized by cooperating with digital libraries. Another aspect to this is that some of these ideas are simpler to implement than others, e. g., a common search interface will be simpler to implement than a study summarization based on natural language processing.

This thesis proposes an automated cross-library search, similar in approach to the common search interface, that fulfills the requirements (i), (ii), (iii), (iv), (vi), (vii), (xii) (see Table 2.1) to address the challenges (i), (iii), (iv), (vii) of the challenges identified in Section 2.4. Additionally, this approach will address challenges of duplicate handling, and merging of results [AC18; DP07].

3.2 Software Engineering Specific SLR Tools

Al-Zubidy and Carver [AC18] and Marshall et al. [MBK14] conducted evaluations of SE specific SLR tools. While Marshall et al. [MBK14] evaluated the support for the complete SLR process Al-Zubidy and Carver [AC18] only evaluated the tools against the search requirements they identified (see Table 2.1).

Marshall et al. [MBK14] identified 4 tools in 2014 (i) SLuRp [BHB12], (ii) StArt [HZFT12], (iii) SLR-Tool [FBR10], and (iv) SLRTOOL [BRAC14]. Al-Zubidy and Carver [AC18] analyzed these four tools (i–iv) and identify two additional tools: (v) SESRA [MB15b], and (vi) Parsifal [2]. As the site provided by [FBR10] to download SLR-Tool (iii) was not available, the tool was not tested as it seems to be no longer supported. Due to that this thesis solely relies on the evaluations by Al-Zubidy and Carver [AC18] and Marshall et al. [MBK14] regarding SLR-Tool (iii). Al-Zubidy and Carver [AC18] evaluated the tools (i–iv) and report that only 6% of them are fulfilled by these tools. The tools (v–vi) support a limited automated search and reach a fulfillment rate of 20% [AC18]. All six tools aim to support each step of the SLR process instead of a single task. This separates them from the tools discussed in Section 3.3 The capabilities of the tools in the context of the search processes are:

[2] https://parsif.al/

(i) **SLuRp**: Allows for searching of individual digital libraries with library-specific queries [BHB12], i.e. no support of cross-library queries that can be automatically transformed into valid digital library-specific queries [AC18]. Marshall et al. [MBK14] evaluate this as automated search is not supported.

(ii) **StArt**: Does not provide any integrated search, but allows documentation and import of search results via so-called search sessions [AC18; HZFT12; MBK14].

(iii) **SLR-Tool**: Does not provide integrated search. Search results have to be imported [FBR10]. Al-Zubidy and Carver [AC18] give SLR-Tool points for supporting collaboration, Marshall et al. [MBK14] on the other hand report that no collaboration is possible. Due to the unavailability of the tool this contradiction was not further investigated.

(iv) **SLRTOOL**: Offers integrated search of Google Scholar, but does not allow to document this search within the tool, making the traceability of the search difficult [BRAC14; MBK14]. The tool supports limited collaborative functions [MBK14].

(v) **SESRA**: Al-Zubidy and Carver [AC18] give SESRA points for support of the search process and the use of standardized queries to search multiple databases. This is contradictory to the experiences made during a personal evaluation of SESRA (2020-09-28), as for each digital library source a separate search string has to be defined, and the automated search either does not return any results in the case of Springer, or the search process did not terminate in the case of IEEEXplore. The bulk import of search results is also supported but did not work without any indication of an error. SESRA supports collaboration of multiple researchers [MB15b].

(vi) **Parsifal**: Al-Zubidy and Carver [AC18] give Parsifal points for support of the search process and the use of standardized queries to search multiple databases. Parsifal allows the definition of a Base Search String that can be copied and adapted for other digital libraries. It does not support the use of a standardized query that can be transformed into valid digital library-specific queries. Parsifal offers integrated search of Scopus and Science@Direct, other digital libraries have to be searched manually and results have to be imported.

Another aspect that Al-Zubidy and Carver [AC18] investigated in their study is which tools the participants of the survey used to conduct their SLRs. The most-reported tools were in descending order: Spreadsheets, Digital Library Search Interfaces, Jabref and Mendeley, StArt, Endnote, and RefWorks. Any further tools were reported only two or fewer times. Of the SE specific SLR tools only StArt was mentioned more than once (6 times). The reason for this could be the fact that all six SE specific SLR tools only offer very limited support for the search process. Especially support for cross-library search with a standardized query language is currently not offered by any of these six tools. Therefore none of these tools address the challenges discussed in Section 2.4.

3.3 Integrated Search Tools

There exists a set of non-SLR tools that support integrated search. Crossref[3] is a tool that offers an integrated search of many different reference repositories of their members[4].

Another set of tools that provide integrated search are a couple of reference management tools, an overview of their capabilities has been published by Kopp et al. [KBM18][5]. This thesis discusses the reference management tools from the overview that support the search of multiple external, and popular digital libraries as these can be used by most researchers during the search process of an SLR. These are: (i) Citavi, (ii) wizdom.ai[6,7], (iii) JabRef, and (iv) Paperpile.

Citavi supports the use of standardized queries that can be automatically transformed into valid digital library-specific queries. A disadvantage is that Citavi is neither open-source nor free of charge. wizdom.ai supports integrated search of individual databases but does not support cross-library queries. A disadvantage is that wizdom.ai is neither open-source nor free of charge. JabRef supports integrated search of individual databases but does not support cross-library queries. An advantage is that it is open-source and free of charge. Paperpile supports a limited integrated search, as results retrieved when searching multiple digital libraries are not sorted by their source, and only free text search is supported. A disadvantage is that Paperpile is neither open-source nor free of charge.

All four tools (i–iv) allow the merging of results and identification of duplicates. Out of these tools, only Citavi offers a standardized query-like functionality. This functionality is offered over the GUI depicted in Figure 3.1. On the top left-hand side of the GUI, the user can select the set of targeted digital libraries. On the right-hand side of the GUI, a display shows which restraints exist regarding the selected set of digital libraries. These restrictions exist as the Citatvi cross-library query language only allows queries to use the search features that are supported by all selected digital libraries instead of providing best-effort transformation for the features that are not universally supported (see Section 4.1). This can, depending on the selected digital libraries, heavily restrict the available search features. A less restrictive approach would be to offer a best-effort approach that allows most search features and applies the supported ones to the corresponding digital libraries. For this reason, the cross-library search capability of Citavi is inferior to the cross-library search capability discussed in Chapter 4. Queries can be stored for future use but cannot be not exported as a search string which makes them hard to share between the research team and hard to document for the search process of an SLR.

An additional reference management tool that is not part of the overview is Yatta[8]. It is a command-line interface (CLI) reference management tool that provides integrated search of the arXiv, and Google-Scholar digital libraries. Like most of the other reference managers, it also does not provide any form of cross-library search.

[3] https://www.crossref.org/

[4] https://www.crossref.org/reporting/members-with-open-references/

[5] See https://ultimate-comparisons.github.io/ultimate-reference-management-software-comparison/ for a comprehensive comparison

[6] https://www.wizdom.ai

[7] Colwiz was merged into wizsom.ai

[8] https://www.npmjs.com/package/yatta

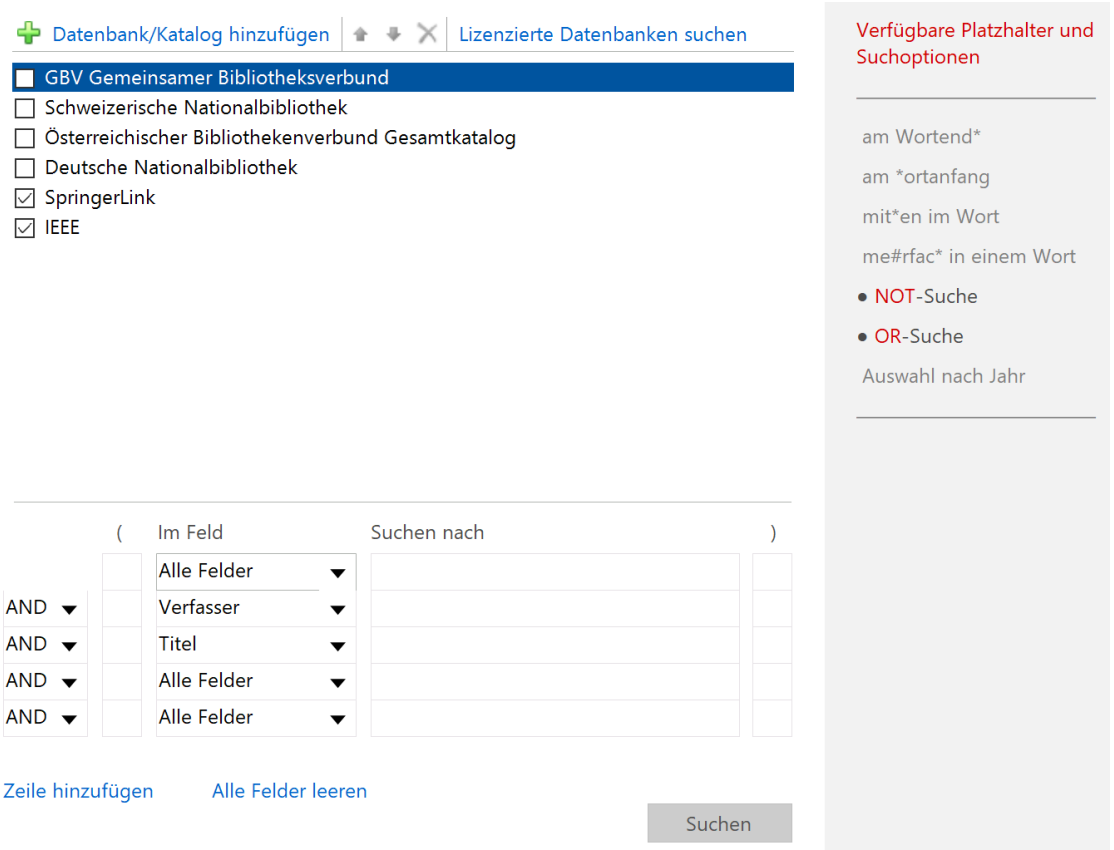

Figure 3.1: Citavi cross-library search GUI

4 Digital Library API Analysis and Cross-Library Query Language

The main requirement, as shown in Table 2.1, is to enable querying across multiple digital libraries. As each digital library supports a custom query language with different syntax and search features this is not a trivial task. Thus before the general search automation approach is introduced in Chapter 5, in this chapter, the results of an analysis of several digital library APIs are presented (see Section 4.1) and a cross-library query language is introduced (see Section 4.2). Section 4.2 also discusses how the transformation from the cross-library query language to a specific query language works.

4.1 Digital Library API Analysis

Most digital libraries use a custom query language that supports a set of search features. These search features consist of the supported Boolean operators, such as AND and OR, wildcard queries, and a set of searchable fields (see Section 2.3), such as the author or title field. For this analysis, the selected candidate set of digital libraries is based on the recommendations by [Gus18; KB13; KPB+10; PVK15] and digital libraries proposed by my advisor.

This candidate set consists of IEEE Xplore[1], ACM Digital Library[2], SCOPUS[3], Web of Science[4], Springer Link[5], arXiv[6], Google Scholar[7], BASE[8], zbMATH[9], The SAO/NASA Astrophysics Data System[10], and Microsoft Academic[11]. Out of these candidates all could be analyzed except Web of Science, since for the advertised Web of Science API Lite[12] no API endpoint or documentation could be found. For Google Scholar, and zbMATH no API documentation is available, all capabilities identified within the analysis have been reverse-engineered from advanced search queries that were created using the advanced search GUI of the digital libraries[9,13].

[1] https://ieeexplore.ieee.org/Xplore/home.jsp
[2] https://dl.acm.org/
[3] https://www.elsevier.com/solutions/scopus
[4] https://www.webofknowledge.com/
[5] https://link.springer.com/
[6] https://arxiv.org/
[7] https://scholar.google.com/
[8] https://www.base-search.net/
[9] https://zbmath.org/
[10] https://ui.adsabs.harvard.edu/
[11] https://academic.microsoft.com/
[12] https://clarivate.com/webofsciencegroup/solutions/xml-and-apis/
[13] https://scholar.google.com/

As some APIs offer specialized or non-standardized search features e. g., the capability to search the subject classification field in arXiv that uses custom codes for their subjects, this thesis identifies a set of frequently-used search capabilities across digital libraries by interviewing a JabRef Developer as an expert in the field of publication management. Only these identified search features have been analyzed. Additionally, some other capabilities that are relevant for the use of these APIs have been included. The result of the analysis is displayed in Table 4.1 and Table 4.2.

Features that have been reported in the documentation or have been identified through testing, are marked with ticks (✓), features that could not be found in the documentation or through testing are marked with crosses (✗). This is especially important to keep in mind for Google Scholar, and zbMATH as the features of these digital libraries could only be retrieved through testing. As a consequence, these APIs might offer certain features that were not found through testing. Therefore, certain features that are marked with a cross might exist but are not advertised towards the API consumer. For the non-search capabilities, some fields are marked as unknown as for these fields either no information was available, or an API key was required but could not be obtained. Within this analysis, there are two special cases discussed below. The ACM API provides a single fielded term to search the title and year field. Thus it does offer support for these fields, but it is not equivalent to the support of both fields individually. The Springer Link API does not offer native support for year-range field search, and even though it can be emulated through query construction it is marked with a cross as it is not natively supported. For any further argumentation that is based on the API features, ACM will be excluded, due to its missing support for the Boolean negation operator, which would prohibit the cross-library query language from supporting it. This is because the set of otherwise supported Boolean operators of AND and OR is not functional complete.

The analysis shows that:

(i) Boolean logic for queries is fully supported.

(ii) The supported fields vary between query languages, the only universally supported fields for search are the author and title fields.

(iii) Wildcard queries are supported by all digital libraries.

Regarding **(ii)**, full support cannot be provided for the non-universally supported fields, such as year, year-range, journal, etc. However, it is important to note that these capabilities only restrict the set of documents that are matched by a query, i.e. just reduce the number of false positives, and not the number of true positives. Because of that, supporting these search features for query languages that support these capabilities does not pose a risk to the completeness of the search. Therefore the cross-library query language can provide a best-effort approach to non-universally supported fields, i.e. transforming them in cases where the field is supported by the digital library-specific query language, and ignoring them otherwise.

Additionally the year, and journal field restrictions can be emulated within the search tool by post-processing the results from non-supporting sources based on the provided metadata. As the subject field contains non-standardized content, that cannot be mapped between query languages that support the subject field, it is not advisable to ever use this for an SLR related search, as it risks the completeness of the search. This field should only be used for the filtering of results in post-processing. For (iii) it is important to mention that not all digital libraries provide equal support of wildcard queries. The use of a wildcard as a suffix is supported universally, as a prefix or in the middle of the word it is not always supported.

Digital Library	IEEE Xplore [IEE20]	ACM Digital Library [ACM20]	SCOPUS [Els20]	Springer Link [Spr20]	arXiv [Cor20]
Search Capability					
Default Field	✓	✓	✓	✓	✓
Title Field	✓	✗	✓	✓	✓
Author Field	✓	✓	✓	✓	✓
Year Field	✓	✓	✓	✓	✗
Year Field (range)	✓	✗	✓	✗	✗
Abstract Field	✓	✗	✓	✗	✗
Keyword Field	✓	✗	✓	✓	✓
Subject Field	✓	✗	✓	✓	✓
Journal Field	✓	✗	✓	✓	✓
Boolean AND	✓	✓	✓	✓	✓
Boolean OR	✓	✓	✓	✓	✓
Boolean NOT	✓	✗	✓	✓	✓
Wildcard	✓	✗	✓	✓	✓
Other Capabilities					
Paginated Results	✓	✓	✓	✓	✓
Maximum Records	200	1000	200	100	2000
Returns Abstract	✓	✗	✓	✓	✓

Table 4.1: Digital Library API Analysis Part 1

4.2 Cross-library Query Language

Based on the analysis a cross-library query language with the following features is introduced:

(i) Support for complete Boolean logic with the following Boolean operators: AND, OR, NOT

(ii) Best-effort support for the search of the following fields: default, title, author, year, year-range, abstract, and journal field (under these names)

(iii) Support for a wildcard as a suffix, e. g., devel*

For the cross-library query language syntax, the default Lucene query language syntax[14] is used. The Lucene syntax was selected as it is a well established, and simple syntax that is already familiar to many users.

Note that the Lucene query language syntax includes support for Lucene query language-specific features that are not supported by the cross-library query language, such as regex and fuzzy search, and term boosting. These features and the related syntax should not be used for these queries,

[14] https://lucene.apache.org/core/8_6_1/queryparser/index.html

E-Library	Google Scholar	BASE [Bie20]	zbMATH	SAO/NASA ADS [Smi20]	Microsoft Academic [Mic20]
Search Capability					
Default Field	✓	✓	✓	✓	✓
Title Field	✓	✓	✓	✓	✓
Author Field	✓	✓	✓	✓	✓
Year Field	✓	✓	✓	✓	✓
Year Field (range)	✓	✓	✓	✓	✓
Abstract Field	✗	✓	✓	✓	✓
Keyword Field	✗	✗	✓	✓	✗
Subject Field	✗	✓	✓	✗	✓
Journal Field	✓	✗	✓	✓	✓
Boolean AND	✓	✓	✓	✓	✓
Boolean OR	✓	✓	✓	✓	✓
Boolean NOT	✓	✓	✓	✓	✓
Wildcard	✓	✓	✓	✓	✓
Other Capabilities					
Paginated Results	Unknown	✓	Unknown	✓	✓
Maximum Records	Unknown	125	200	2000	Unknown
Returns Abstract	✓	Unkown	Unknown	✓	✓

Table 4.2: Digital Library API Analysis Part 2

as they are not supported. The supported syntactical building blocks are Terms, Fields, Boolean Operators, Wildcard Searches (only suffix wildcards), and Grouping. These can be found under their names within the Lucene query language syntax specification [15].

Now that the cross-library query language is defined the transformation to a specific target query language can be discussed. This transformation uses the AST representation of the query formulated in the cross-library query language to construct a semantically equivalent query string in the target query language. How such an AST can be acquired for a cross-library query is discussed in Section 6.2. This transformation is realized by converting the nodes of the AST from the leaf nodes to the root node (bottom-up) into semantically equivalent syntactic constructs of the target query language. It is important to note that due to (ii) semantic equivalence in this case is only true in regards to the inclusiveness of the query, not the exclusiveness. The reason for this is that the best-effort support of these fields means that if these fields are contained within a cross-library query expression, that they will be transformed into a semantically equivalent representation if the target query language supports the field (for year, year-range, abstract, and journal fields). If a field is used that is not supported by the cross-library query language, or that is not supported by the target query language, the field will be ignored to ensure completeness of the search.

The properties of this best-effort transformation can be expressed mathematically:

[15] https://lucene.apache.org/core/8_6_1/queryparser/index.html

Let $G_{standard}$ be the context-free grammar of the cross-library query language (provided by Lucene).

Let G_{target} be the context-free grammar of the target query language.

Let D be the set of all possible publications and P_w be a predicate that describes for a publication $p \in D$ whether the query w matches the publication ($P_w(p) = 1$) or not: $P_w(p) = 0$

$$P_w : D \rightarrow \{0, 1\}$$

This models the semantic interpretation (which publications are matched) of a query w.

With this, the set of all publications that are matched by a query w P_w can be defined as:

$$P_w = \{p | p \in D \wedge P_w(p) = 1\}$$

The best-effort semantically equivalent mapping can then be expressed as a function t:

$$t : L(G_{standard}) \rightarrow L(G_{target}) \text{ with: } \forall w \in L(G_{standard}) : P_w \subseteq P_{t(w)}$$

To provide an example for such a transformation the example query depicted in Figure 2.3 is used as it is already formulated in the cross library language. The AST of the example query is depicted in Figure 2.4. The target query language for this example will be the query language used by the Springer Link API [Spr20].

The transformation begins by transforming the leaf nodes of the AST, which contain all the terms, into semantically equivalent syntactic constructs of the Springer Link query language. The terms within the grouping (the leaf nodes beneath the inner OR node) are directly transformed into equivalent simple terms and phrases. However, the first and last term cannot be transformed as they are not supported by the Springer Link query language. As the first term cannot be emulated in any way, it is discarded to ensure the completeness of the search. The last term on the other hand can be emulated as the Springer Link query language supports the year-fielded term (see Section 4.1). In this case the `year-range: 2018-2020` fielded term can be emulated through the following query string:

```
(date:2018-* OR date:2019-* OR date:2020-*).
```

The reason for the wild card at the end of each date field is that the API expects the ISO date format.

After that, the query string is constructed recursively by applying the operator of a node on all of the transformed child node strings. In the example first, the OR operator is applied to the three terms in the grouping. As the Springer Link query language also uses infix notation the resulting query string is:

```
(“test driven development” OR “test-driven development” OR tdd).
```

Then the AND operator is applied to the two remaining child nodes (as the abstract-fielded term is not supported). This results in the following query string:

```
(date:2018-* OR date:2019-* OR date:2020-*) AND
(“test driven development” OR “test-driven development” OR tdd)
```

With the cross-library query language and the transformation to any target query language, the automated cross-library search that is introduced in Chapter 5 can be realized to address the challenge of different syntax across digital libraries.

5 Automated Cross-library Search and Result Management

In this chapter, the Meta Searcher is introduced to support the search process in SLRs. It provides (i) cross-library search (see Chapter 4), (ii) result management of the retrieved studies in a standardized format and (iii) management of the targeted digital libraries and search queries using a machine-readable format. To achieve these features the Meta Searcher is integrated into an existing reference management tool that supports the de-facto BibTeX/BibLaTeX standard for references and leverages the cross-library query language and transformation introduced in Section 4.2.

Section 5.1 introduces the search process that leverages the features of the Meta Searcher, including the provided result management. Section 5.2 introduced the machine-readable format used to define the relevant search parameters. In Section 5.3 a conceptual architecture for the Meta Searcher is proposed. This section also discusses how the architectural components collaborate to support each step of the search process introduced in Section 5.1. In Section 5.4 the benefits of the limitations of this approach are discussed.

5.1 Automated Search Process Overview

To address the requirements summarized in Section 2.4 for search automation and result management in SLRs in this section the general automated search process and tool support using the proposed Meta Searcher is introduced. The Meta Searcher-supported search process is depicted in Figure 5.1. This process is based on the process introduced in Section 2.1.2. It contains the first and third step of the original process.

The first step is for the research team to formulate the queries in the cross-library query language (see Chapter 4), and select the targeted digital libraries in the machine-readable format introduced in Section 5.2. The result of this step is the so-called "SLR Definition" file, as it contains all information that is required for the retrieval of studies. After this step is done the Meta Searcher starts the automated cross-library search. The second step is to automatically transform the queries specified in the SLR definition into the target digital libraries query language. In this step, for each targeted digital library the corresponding fetcher transforms the query into a semantically equivalent query in the target query language (see Section 4.2). This step leverages the cross-library query language and transformation introduced in Section 4.2.

In the third step, the targeted digital libraries are automatically queried using the library-specific queries. This step includes the conversion of results into a de-facto standard format (BibTeX/BibLaTeX). This enables the direct export and use of the results in a wide variety of tools. To make the search process traceable, this querying and the corresponding results are documented by the Meta Searcher. The fourth step is to post-process the retrieved results. This includes the merging of

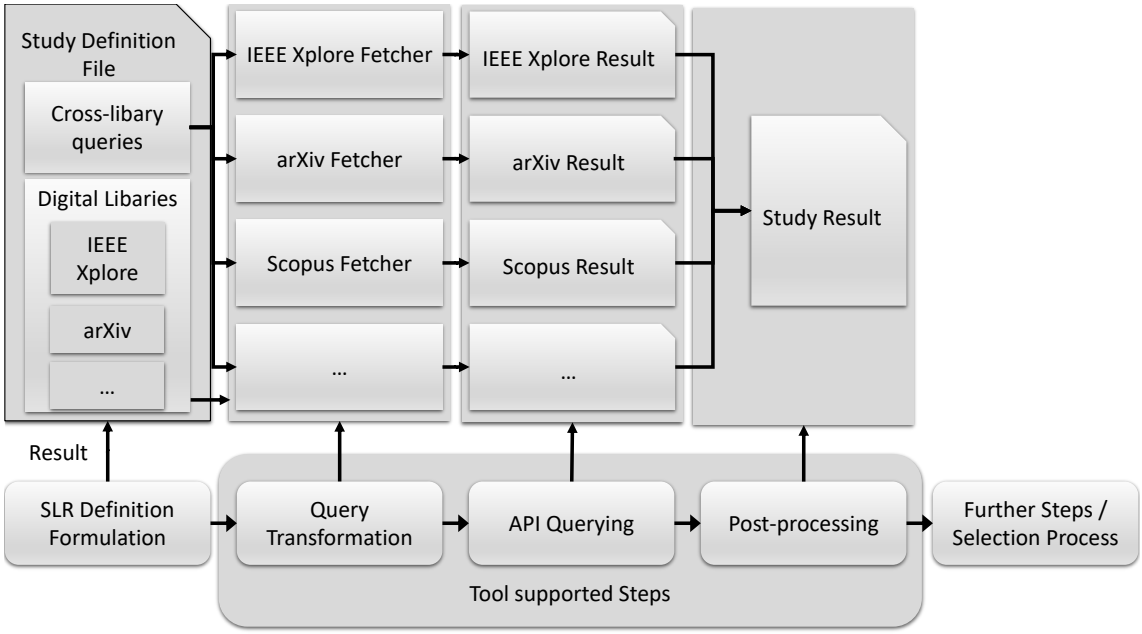

Figure 5.1: Tool Supported Search Process

results from different digital libraries into a single result, including the de-duplication of the result set. This third and fourth steps are part of the result management. These are realized through the integration of the Meta Searcher into an existing reference management tool that already provides most of these functionalities. After this step, the results are all available in a single bibliographic database file. All entries within this result database are either in BibTeX or BibLaTeX format. The fifth step is either to continue the search process with other steps, e. g., consultation of experts for further papers [Kit07], or either a focused [JS07] or a restricted [KBT+09] manual search. Or to begin the collaborative selection process. Therefore the Meta Searcher provides fully automated cross-library search and result management including de-duplication and merging of results.

5.2 SLR Definition File Format

During the Query Formulation step (see Figure 5.1) the conducting researchers have to define which digital libraries to include in the search and which queries should be used to identify relevant studies. As this information is essential to conduct a systematic search it has to be provided to the Meta Searcher to perform the automated search. This is achieved by formulating this information, and some metadata about the SLR, e. g., author or research questions, in a machine-readable format in a so-called “SLR Definition” file. As most reference management tools can read bibliographic databases (files with the .bib extension) that contains BibTeX/BibLaTeX entries, and as BibTeX syntax allows for custom entry types with custom fields this thesis defines the SLR Definition File as a bibliographic database and the contained information as entries in the BibTeX syntax.

An example SLR Definition file is depicted in Listing 5.1. In each SLR Definition file exists exactly one Study type entry that contains the `name`, `author`, `lastsearchdate`, and `researchquestions` terms. The name term contains the title of the SLR, the `author` term the SLR authors, and `researchquestions`

term the research questions that the SLR tries to address. The `lastsearchdate` term contains the date the last search was conducted. This date is stored in ISO format (yyyy-mm-dd). For each query, the file contains a `SearchQuery` type entry that contains the string representation of the query in the query term. The cite key for each query is query<id> where the id is a unique ID that is assigned to the query.

```
    @Study{v10,
      author            = {Dominik Voigt},
      lastsearchdate    = {2020-10-18},
      name              = {TestStudy},
      researchquestions = {ResearchQuestion1; ResearchQuestion2},
    }

    @SearchQuery{query1,
      query={abstract:agile AND ("test driven development" OR "test-driven development" OR tdd
) AND year:2005-2020},
    }

    @Library{library1,
      name = {Springer},
      enabled = {true},
      comment = {},
    }

    @Library{library2,
      name = {ArXiv},
      enabled = {false},
      comment = {},
    }
```

Listing 5.1: Example SLR Definition file

5.3 Conceptual Architecture

This Meta Searcher and the features it provides are realized using the architecture depicted in Figure 5.2. This conceptual architecture diagram uses the Fundamental Modeling Concepts (FMC) [KGT06] notation for block diagrams [KGT06]. In this visual notation, active components are depicted as rectangular boxes and passive components (Storage) as either ellipses or rectangular boxes with rounded corners. The edges depict an existing relation between connected components.

The diagram depicts how the Meta Searcher is integrated into an existing reference management tool. The Meta Searcher component itself consists of four active components, these are (i) the Meta Searcher API, (ii) the Digital Library Fetcher Controller, (iii) the Digital Library Fetchers, (iv) the Repository Manager, and a passive component the Version Controlled (VC) Local Repository. The Meta Searcher API (i) is a service layer component that provides an external interface and delegates the tasks that have to be fulfilled for an automated search to the Digital Library Fetcher Controller (ii), and the Repository Manager (iv). The Digital Library Fetcher Controller (ii) implements the meta-search and delegates any digital specific tasks, and the search of the digital library itself to the corresponding Digital Library Fetchers (iii). The Repository Manager is responsible for any

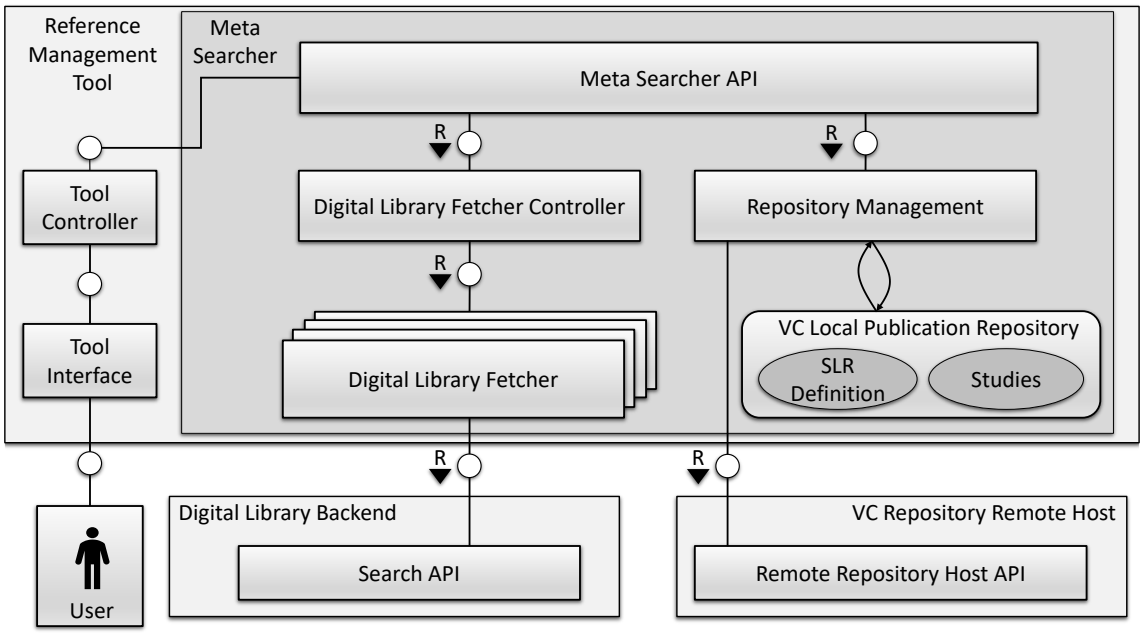

Figure 5.2: Conceptual Architecture of the tool within an SLR tool

repository related tasks such as persistence, and retrieval of information. It reads and writes the VC Local Repository. The VC Local Repository is a local file-based repository that contains all necessary information to conduct the search process and the search results. It tracks a remote VC repository to make the search process collaborative and traceable using the version history. This will be further discussed during the discussion about the collaboration of components.

Externally (outside of the system) the Meta Searcher component interacts with 3 active components: (vi) the user, (vii) the Search API of the digital library back end, and (viii) the Remote Repository Host API of the VC Repository Remote Host. The Search API (viii) provides the study references that match a provided query. The Remote Repository Host API represents the interface that is used to remotely create, access, and update a version-controlled remote repository. In the following subsections the realization of the different steps of the process introduced in Section 5.1 using the conceptual architecture is discussed.

5.3.1 SLR Definition Formulation

During the SLR Definition Formulation step (see Figure 5.1) the conducting researchers define the “SLR Definition” file. The Meta Searcher API then provides the Repository Management with the file. The Repository Manager then uses the parent directory of the file as the root of the VC local repository. During this step, a remote repository should be created. This allows multiple researchers to participate in the search process, or to just retrieve the results contained in the repository. Alternatively, if a VC remote repository already exists, the researcher should provide a URL to that repository and the Repository Manager will clone the remote repository by accessing the Remote Repository Host API and use the directory of the cloned repository as repository root.

5.3.2 Query Transformation and API Querying

The Query Transformation and API Querying step (see Figure 5.1) are executed together. The Meta Searcher API requests the targeted digital libraries and cross-library queries from the Repository Manager. The Repository Manager reads the SLR Definition file and provides the requested information. Then the Digital Library Fetcher Controller receives the queries formulated in the cross-library query syntax and the set of libraries that should be queried from the Meta Searcher API. The Controller then delegates the transformation of the queries into the digital library-specific query syntax, the querying of the digital libraries Search API, and the transformation of the references into a selected format to the digital library-specific Digital Library Fetchers (iii). The controller keeps track of which study references were returned for which query by which fetcher and returns this information to the Meta Searcher API.

5.3.3 Post-processing

In the post-processing step (see Figure 5.1) the Meta Searcher API delegates the persistence of the results it received during the third step from the Digital Library Fetcher Controller to the Repository Manager. The Repository Manager persists the received study references within the VC Local Repository in a structured manner as depicted in Figure 5.3. For each query, a folder is created with a unique name, and the results of each fetcher are stored individually in a file named after the fetched digital library. Additionally, the Repository Manager merges the results retrieved by all fetchers per query in a "Query result" file and merges these into an "SLR result" file that contains all retrieved study references. In case that any of these files already exist, the Repository Manager will merge the new results into the existing files. This merging has to ensure that no duplicates are created, addressing the challenge of duplicates within different digital libraries [CHHK13].

After the persistence was successful, the Repository Manager creates a new version using the Version Control system, and updates the remote repository with the new version. This allows anyone that has access to the repository to (i) see when a search was conducted, and (ii) see which new results were found, increasing the traceability of the search. During this step all other tasks could be initiated by the Meta Searcher API. Such as the post-filtering of results from digital libraries that do not support certain search features based on the metadata of the result entries.

This discussion shows that the provided architecture for the Meta Searcher component can support the search process introduced in Section 5.1. It also shows that the proposed Meta Searcher can provide an integrated and automated cross-library search with a standardized query for multiple libraries, addressing the main challenge identified in Section 2.4.

5.4 Benefits and Limitations

The Meta Searcher provides multiple benefits over existing search and result management approaches due to the features it provides. The automatic transformation of cross-library queries and the documentation of the search results lead to a fully automated, systematic cross-library search. This addresses the (i) requirement identified in Table 2.1 and ensures the systematic conduction of the search. The integration of the Meta Searcher into an existing reference management tool that

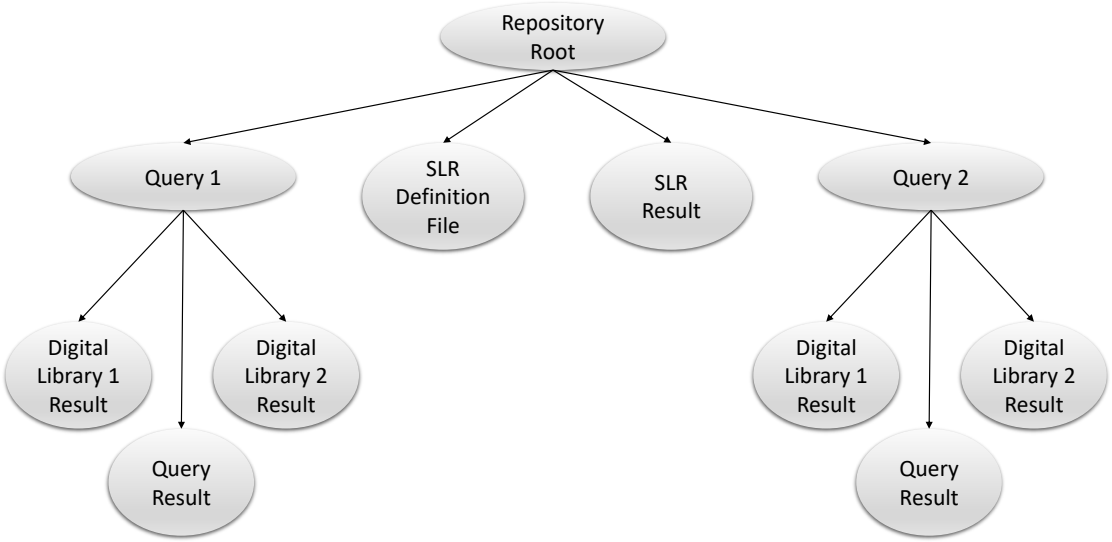

Figure 5.3: Directory Structure of the VC Local Repository

supports the BibTeX and BibLaTeX standard allows for result management of the results including the de-duplication and merging of results from different digital libraries. This addresses the (ii), (iii), (vi), and (vii) requirements (see Table 2.1), as it provides a single tool for search and management of studies. This also provides further features, e. g., automated citation key generation, that are provided by the reference management tool. The use of the BibTeX and BibLaTeX standard furthermore ensures easy export of the results to other compliant tools. Therefore it addresses the (i), (iii), (iv), and (vii) challenges of the search process identified in Section 2.4. Thus addressing many of the most relevant challenges that are encountered during the search process. However, even outside of the scope of the SLR method, the Meta Searcher can provide benefits to researchers by providing an automated and integrated search of multiple digital libraries and easy management and export of search results using the BibTeX and BibLaTeX formats.

But there are certain limitations to this approach, namely:

(i) Many of the digital libraries require an active subscription

(ii) Many of the digital library APIs require an API key that has to be requested

(iii) Many of the digital libraries have rate limits

(iv) All digital libraries need a manually developed fetcher and transformer to the digital library-specific query language adapted accordingly

(v) The transformer and the API capabilities are tightly coupled, i.e. if the API capabilities change the transformer has to change as well

(vi) The IEEE Xplore API only supports URL parameters for different fields of a query, which does not allow arbitrary Boolean queries with different fielded terms.

The first two limitations are not addressable, as these are just the current state of affairs. The third limitation might be solvable by trying to cooperate with the digital library providers, as Al-Zubidy and Carver [AC18] propose. The fourth and fifth limitations do not seem to be easily addressable either, as these challenges are induced by the fact that digital library APIs use specific, custom, and non-standardized query languages. To address these challenges there would be two possible collaborative solutions. One solution would be to resolve the need for transformation between query languages by using a standardized query language for all APIs of all large digital libraries. Another solution would be to provide a machine-readable API description by using an Interface Description Language (IDL), such as Swagger[1], and developing a query language transformer generator that uses the IDL description of a digital library API to create a transformer for that specific digital library-specific query language. To address the sixth limitation IEEE has to adapt their API. It can be concluded that most of these challenges are solvable, but require action on the side of the digital library providers.

[1] https://swagger.io/

6 Implementation

In this chapter, the implementation details of the Meta Searcher prototype that was developed during this thesis are discussed. As the prototype is integrated in JabRef it is also developed with Java. The architectural decisions made during the development of the prototype were documented with Markdown Architectural Decision Records[KA19] due to the benefits described by Kopp et al. [KAZ18]. Section 6.1 discusses the reasons why the prototype was developed and integrated within an existing tool, and why JabRef[1] was chosen. In Section 6.2 the implementation of a limited version of the cross-library query language (see Chapter 4) is introduced and discussed. Section 5.2 introduces the machine-readable format developed to define the queries and targeted digital libraries. Section 6.3 introduces how the persistence of the search results is realized. Section 6.4 discusses the limitations of the current prototype.

6.1 JabRef

As the Meta Searcher component only provides integrated search, it is essential to integrate it into an existing tool that provides additional functionalities to fully support the search process (see Chapter 3). As the references retrieved during the search have to be managed, integration of the prototype into a reference management tool that already offers features for the management of references is sensible, as this is an important feature [HCHA16]. Additionally, some reference management tools already provide integrated search functionality[2] that could be extended to develop the integrated cross-library search.

To be suitable for extension a tool has to fulfill certain requirements:

- To remove any monetary adoption barrier the extended tool should be free of charge.
- To enable the future enhancements to the developed prototype the selected tool should be open-source

There exist multiple open-source and free of charge reference management tools [KBM18]. JabRef was chosen from these as it (i) is already used by researchers to conduct SLRs [AC18], and (ii) provides integrated search for the large set of digital libraries, including the commonly used digital libraries within CS and SE (see Table 4.1 and Table 4.2). This set of supported digital libraries for integrated search is larger than that of the other free of charge and open-source alternatives[2] . Hence, JabRef was chosen to be extended to develop the prototype.

[1] https://www.jabref.org/

[2] See https://ultimate-comparisons.github.io/ultimate-reference-management-software-comparison/ for a comprehensive comparison

6.2 Cross-library Query Language and Transformation

To implement the Meta Searcher, the cross-library query language and transformation have to be implemented (see Section 4.2). For the transformation, the AST representation of a cross-library query is required (see Section 4.2). To generate an AST from a query string a parser is used. To create such a parser a parser generator can be used. A parser generator uses the provided context-free grammar, normally in Extended Backus–Naur Form (EBNF), that is used to define the query syntax to create the parser.

Thus for the cross-library query language, either a new query language based on an EBNF, and a semantic description can be defined, or an existing query language by an open-source search application can be used, if the AST of a query instance can be acquired within the code. In the first case, a parser that creates the parse tree and AST is needed. Such a parser can be generated by commonly used parser generators such as JavaCC[3], or ANTLR[4]. These use the EBNF of the context-free query language to create the parser.

For the second case, a project such as Lucene[5] can be used. Lucene provides a query language, a query parser, and access to the AST (called a QueryNode tree) as well as ways to natively interact with the nodes of the AST via node processors. Lucene offers a very sophisticated query language, the cross-library query language cannot support all of the Lucene query language features. Due to that, the proposed cross-library query language is a subset of the Lucene query language, and if it is used for query transformation certain Nodes of the Query Node tree will either need to be removed or ignored during transformation. Additionally, Lucene supports arbitrary fields out-of-the-box. Based on these options, the cross-library query language was chosen to be implemented using Lucene. As the traversal and transformation of the Query Node tree proved to be challenging, and IEEE Xplore only supporting URL parameters for different fields[6], the prototype only provides support for the Boolean AND operator for query construction, as it is the most commonly used operator for query formulation.

To transform a query from the cross-library query language into the target query language the Digital Library Fetcher Controller passed the string representation of the query to a Lucene query parser to create the AST. For this, the provided standard query parser is sufficient. Then the fielded terms are extracted from the AST and out of these terms a query object is created that aggregates all terms for each field in internal lists. This query object represents the provided query and is passed to the different Digital Library Fetchers of the targeted digital libraries for transformation and search. Each Digital Library Fetcher then transforms the provided query object into a semantically equivalent query in the digital library-specific query language. As IEEE Xplore only supports query parameters for the different fields of the query (see Section 5.4), the query object has to be directly transformed into a request URL. For all other digital libraries, a string representation of the query in the digital library-specific query syntax is created. Then the Fetchers make API calls against the search APIs of the corresponding digital library and convert the returned references in a common format, which is either BibTeX or BibLaTeX in the case of JabRef. This functionality was adapted

[3] https://javacc.github.io/javacc/
[4] https://www.antlr.org/
[5] https://lucene.apache.org/
[6] https://developer.ieee.org/docs/read/Metadata_API_details

from the existing Digital Library Fetchers that provide the original integrated search within JabRef. The Digital Library Fetcher Controller then creates an allocation of which results were returned by each digital library for each query. This is then passed to the Repository Manager for persistence.

6.3 Persistence

The Repository Manager is responsible for the management of the SLR Definition file, and the persistence of the results. It uses the directory that contains the SLR Definition file as the file-based repository. It extracts the information contained in the SLR Definition file using the existing functionality to parse ".bib" within JabRef and provides these to the Meta Searcher API. When it receives the results from the Meta Searcher API it creates the file structure presented in Figure 5.3. All results are stored as references within ".bib" files in BibTeX or BibLaTeX format. The results of each fetcher for each query are stored within the query folder under the name of the fetcher, e. g., arXiv.bib. The names for the folders for each query are generated from the normalized version of the string representation of the query and the query's id. The Repository Manager merges the results of all fetchers for each query into a file stored within the query folder (result.bib) and all results of all queries for the whole study into a file stored in the repository root (studyResult.bib). In the case that the repository already exists from a previous search, the Repository Manager merges the new results into the existing result files. To not introduce any duplicates during any of the merging tasks, the Repository Manager uses the duplication detection functionality of JabRef.

6.4 Limitations

Additionally to the conceptual limits (see Section 5.4) the prototype has further limitations. The developed prototype only supports a subset of the cross-library query language proposed in Chapter 4, as it only supports Boolean AND queries, where each term is connected using the Boolean AND operator. This is partial because the IEEE Xplore API currently does not support queries that use a single string representation, which does not allow to transform many queries into a semantically equivalent IEEE Xplore query (see Section 5.4). To support the full cross-library query language, the complete AST to query transformation has to be implemented as discussed in Chapter 4. Another limitation is that currently only IEEE Xplore, Springer Link, arXiv, and Google Scholar is supported as only for these digital libraries transformers were developed. This can be addressed by implementing transformers for all other libraries and merging them into the JabRef code base that contains the prototype, as JabRef is open-source. Additionally, the support for paginated search is currently missing but should be implemented to circumvent the limitations on the number of returned references as shown in Table 4.1 and Table 4.2 in the row named "Maximum Records". Other currently missing features are version-controlled and remote repositories that are required for collaboration and traceability of the search process (see Chapter 5).

7 Conclusion and Future Work

This thesis identified and addressed the current challenges (i), (iii), (iv), and (vii) (see Section 2.4) during the search process of an SLR. Based on existing works, challenges that were encountered by researchers during the conduction of the search process within an SLR could be identified (see Section 2.4). The studies provide strong evidence that the identified challenges still exist. The existing SLR tools that aim to support the researcher during all steps of the SLR process fail to address these challenges.

The main goal of this thesis was to address exactly these challenges that are currently not addressed satisfyingly by any other as relevant identified tool. One main contribution of this thesis is the digital library API capability analysis and the resulting cross-library query language that can be used to formulate queries that can be transformed into semantically equivalent queries in the digital library-specific query language.

Another contribution is the definition of a machine-readable format that can be used to define the queries and targeted digital libraries for an SLR. These contributions enabled the development of the Meta Searcher that can support the search process by providing integrated and automated cross-library search and result management. For this, a prototype was developed, that was introduced in Chapter 6, that supports a subset of the defined cross-library query language and a limited set of digital library fetchers that can transform the cross-library query language into the digital library-specific query language (see Section 6.4). It is integrated within JabRef, a reference management tool, to enable the result management feature using the features provided by JabRef. The developed prototype shows that the Meta Searcher integrated into a reference management tool, such as JabRef, can address the most relevant requirements (i), (ii), (iii), (vi), and (vii) (see Table 2.1). Making this a promising approach for future SLR and overall digital library search tools.

Outlook

This thesis provides the conceptual model for the Meta Searcher that provides cross-library search support and an implementation baseline for it. This prototype implementation should be extended to support the complete cross-library syntax and to provide the version-controlled repository system to enable easy collaboration of researchers within a team by providing a simple sharing mechanism for the search results and increasing the traceability through versioning. With this, the prototype can fully address the challenges currently faced during the search process of an SLR (see Section 2.4). The transformers would need to be adjusted accordingly. This would address the limitations discussed in Section 6.4. As there currently exists no way to decouple the transformer from the digital library-specific query language, the set of transformers has to be manually extended. These transformers additionally will have to manually maintained to address any changes made to the corresponding digital library search API.

Bibliography

[AC18] A. Al-Zubidy, J. C. Carver. “Identification and prioritization of SLR search tool requirements: an SLR and a survey”. In: *Empirical Software Engineering* 24.1 (May 2018), pp. 139–169. DOI: 10.1007/s10664-018-9626-5 (cit. on pp. 13, 14, 18, 24–27, 30, 31, 47, 49).

[ACM20] ACM. *ACM Digital Library API Documentation*. 2020. URL: https://github.com/CrossRef/rest-api-doc (visited on 08/26/2020) (cit. on p. 37).

[BBF+18] A. Bergmayr, U. Breitenbücher, N. Ferry, A. Rossini, A. Solberg, M. Wimmer, G. Kappel, F. Leymann. “A Systematic Review of Cloud Modeling Languages”. In: *ACM Computing Surveys* 51.1 (Apr. 2018), pp. 1–38. DOI: 10.1145/3150227 (cit. on pp. 15, 18, 19).

[BHB12] D. Bowes, T. Hall, S. Beecham. “SLuRp”. In: *Proceedings of the 2nd international workshop on Evidential assessment of software technologies - EAST ’12*. ACM Press, 2012. DOI: 10.1145/2372233.2372243 (cit. on pp. 30, 31).

[Bie20] Bielefeld University Library. *BASE API Documentation*. 2020. URL: https://www.base-search.net/about/download/base_interface.pdf (visited on 08/26/2020) (cit. on p. 38).

[BKB+07] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, M. Khalil. “Lessons from applying the systematic literature review process within the software engineering domain”. In: *Journal of Systems and Software* 80.4 (Apr. 2007), pp. 571–583. DOI: 10.1016/j.jss.2006.07.009 (cit. on pp. 16–18).

[BMNT05] J. Biolchini, P. G. Mian, A. C. C. Natali, G. H. Travassos. “Systematic review in software engineering”. In: *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES* 679.05 (2005), p. 45 (cit. on p. 15).

[BRAC14] B. S. Barn, F. Raimondi, L. Athappian, T. Clark. “Slrtool: A Tool to Support Collaborative Systematic Literature Reviews”. In: *Proceedings of the 16th International Conference on Enterprise Information Systems*. SCITEPRESS - Science, 2014. DOI: 10.5220/0004972204400447 (cit. on pp. 30, 31).

[BZ09] M. A. Babar, H. Zhang. “Systematic literature reviews in software engineering: Preliminary results from interviews with researchers”. In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, Oct. 2009. DOI: 10.1109/esem.2009.5314235 (cit. on p. 15).

[CHHK13] J. C. Carver, E. Hassler, E. Hernandes, N. A. Kraft. “Identifying Barriers to the Systematic Literature Review Process”. In: *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, Oct. 2013. DOI: 10.1109/esem.2013.28 (cit. on pp. 24, 30, 45).

[Cor20] Cornell University. *arXiv API Documentation*. 2020. URL: https://arxiv.org/help/api/user-manual (visited on 08/26/2020) (cit. on p. 37).

[DDH07] T. Dyba, T. Dingsoyr, G. K. Hanssen. "Applying Systematic Reviews to Diverse Study Types: An Experience Report". In: *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, Sept. 2007. DOI: 10.1109/esem.2007.59 (cit. on pp. 15, 19, 26).

[DP07] O. Dieste, A. G. Padua. "Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews". In: *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. IEEE, Sept. 2007. DOI: 10.1109/esem.2007.19 (cit. on pp. 17, 22, 26, 30).

[Els20] Elsevier. *Scopus API Documentation*. 2020. URL: https://dev.elsevier.com/sc_search_tips.html (visited on 08/26/2020) (cit. on p. 37).

[FBR10] A. M. Fernandez-Saez, M. G. Bocco, F. P. Romero. "SLR-TOOL - A Tool for Performing Systematic Literature Reviews". In: *5th International Conference on Software and Data Technologies*. 2010 (cit. on pp. 30, 31).

[FML17] P. D. Francesco, I. Malavolta, P. Lago. "Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption". In: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, Apr. 2017. DOI: 10.1109/icsa.2017.24 (cit. on pp. 15, 19, 20).

[FMMM12] K. R. Felizardo, S. G. MacDonell, E. Mendes, J. C. Maldonado. "A Systematic Mapping on the use of Visual Data Mining to Support the Conduct of Systematic Literature Reviews". In: *Journal of Software* 7.2 (Feb. 2012). DOI: 10.4304/jsw.7.2.450-461 (cit. on p. 29).

[Gus18] M. Gusenbauer. "Google Scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases". In: *Scientometrics* 118.1 (Nov. 2018), pp. 177–214. DOI: 10.1007/s11192-018-2958-5 (cit. on p. 35).

[HCHA16] E. Hassler, J. C. Carver, D. Hale, A. Al-Zubidy. "Identification of SLR tool needs – results of a community workshop". In: *Information and Software Technology* 70 (Feb. 2016), pp. 122–129. DOI: 10.1016/j.infsof.2015.10.011 (cit. on pp. 14, 27, 30, 49).

[HCKH14] E. Hassler, J. C. Carver, N. A. Kraft, D. Hale. "Outcomes of a community workshop to identify and rank barriers to the systematic literature review process". In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*. ACM Press, 2014. DOI: 10.1145/2601248.2601274 (cit. on pp. 14, 24, 26).

[HZFT12] E. Hernandes, A. Zamboni, S. Fabbri, A. D. Thommazo. "Using GQM and TAM to evaluate StArt – a tool that supports Systematic Review". In: *CLEI Electronic Journal* 15.1 (Apr. 2012). DOI: 10.19153/cleiej.15.1.2 (cit. on pp. 30, 31).

[IEE20] IEEE. *IEEE Xplore API Documentation*. 2020. URL: https://developer.ieee.org/docs/ (visited on 08/26/2020) (cit. on p. 37).

[JS07] M. Jorgensen, M. Shepperd. "A Systematic Review of Software Development Cost Estimation Studies". In: *IEEE Transactions on Software Engineering* 33.1 (Jan. 2007), pp. 33–53. DOI: 10.1109/tse.2007.256943 (cit. on p. 42).

[KA19] O. Kopp, A. Armbruster. “Generalized Markdown Architectural Decision Records: Capturing the Essence of Decisions (short paper)”. In: *Proceedings of the 11th Central European Workshop on Services and their Composition, Bayreuth, Germany, February 14-15, 2019*. Ed. by S. Kolb, C. Sturm. Vol. 2339. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 55–57. URL: http://ceur-ws.org/Vol-2339/paper11.pdf (cit. on p. 49).

[KAZ18] O. Kopp, A. Armbruster, O. Zimmermann. “Markdown Architectural Decision Records: Format and Tool Support”. In: *ZEUS*. 2018 (cit. on p. 49).

[KB13] B. Kitchenham, P. Brereton. “A systematic review of systematic review process research in software engineering”. In: *Information and Software Technology* 55.12 (Dec. 2013), pp. 2049–2075. DOI: 10.1016/j.infsof.2013.07.010 (cit. on pp. 13, 15, 17–19, 21, 24, 29, 35).

[KBB+09] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman. “Systematic literature reviews in software engineering – A systematic literature review”. In: *Information and Software Technology* 51.1 (Jan. 2009), pp. 7–15. DOI: 10.1016/j.infsof.2008.09.009 (cit. on p. 15).

[KBL+11] B. Kitchenham, P. Brereton, Z. Li, D. Budgen, A. Burn. “Repeatability of systematic literature reviews”. In: *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*. IET, 2011. DOI: 10.1049/ic.2011.0006 (cit. on pp. 13, 18, 25, 26).

[KBM18] O. Kopp, U. Breitenbücher, T. Müller. “CloudRef - Towards Collaborative Reference Management in the Cloud.” In: *ZEUS* (2018), pp. 63–68 (cit. on pp. 32, 49).

[KBT+09] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, D. Budgen. “The impact of limited search procedures for systematic literature reviews &#x2014 A participant-observer case study”. In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, Oct. 2009. DOI: 10.1109/esem.2009.5314238 (cit. on p. 42).

[KDJ04] B. Kitchenham, T. Dyba, M. Jorgensen. “Evidence-based software engineering”. In: *Proceedings. 26th International Conference on Software Engineering*. IEEE Comput. Soc, 2004. DOI: 10.1109/icse.2004.1317449 (cit. on p. 15).

[Key12] Keyser. *Indexing : from thesauri to the Semantic Web*. Oxford: Chandos, 2012. ISBN: 9781843342922 (cit. on p. 20).

[KGT06] A. Knopfel, B. Grone, P. Tabeling. *Fundamental Modeling Concepts: Effective Communication of It Systems*. WILEY, May 1, 2006. 334 pp. ISBN: 047002710X. URL: https://www.ebook.de/de/product/5740256/andreas_knopfel_bernhard_grone_peter_tabeling_fundamental_modeling_concepts_effective_communication_of_it_systems.html (cit. on p. 43).

[Kit07] Kitchenham. *Guidelines for performing Systematic Literature Reviews inSoftware Engineering*. Tech. rep. 2007 (cit. on pp. 13, 15–19, 22, 30, 42).

[KPB+10] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, S. Linkman. “Systematic literature reviews in software engineering – A tertiary study”. In: *Information and Software Technology* 52.8 (Aug. 2010), pp. 792–805. DOI: 10.1016/j.infsof.2010.03.006 (cit. on p. 35).

[MB13] C. Marshall, P. Brereton. “Tools to Support Systematic Literature Reviews in Software Engineering: A Mapping Study”. In: *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, Oct. 2013. DOI: 10.1109/esem.2013.32 (cit. on p. 29).

[MB15a] C. Marshall, P. Brereton. “Systematic review toolbox”. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering - EASE ’15*. ACM Press, 2015. DOI: 10.1145/2745802.2745824 (cit. on pp. 13, 27, 29).

[MB15b] J. S. Molléri, F. B. V. Benitti. “SESRA”. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering - EASE ’15*. ACM Press, 2015. DOI: 10.1145/2745802.2745825 (cit. on pp. 30, 31).

[MBK14] C. Marshall, P. Brereton, B. Kitchenham. “Tools to support systematic reviews in software engineering A Feature Analysis”. In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE ’14*. ACM Press, 2014. DOI: 10.1145/2601248.2601270 (cit. on pp. 30, 31).

[Mic20] Microsoft. *Microsoft Academic API Documentation*. 2020. URL: https://docs.microsoft.com/en-us/academic-services/project-academic-knowledge (visited on 08/26/2020) (cit. on p. 38).

[MKB18] C. Marshall, B. Kitchenham, P. Brereton. “Tool Features to Support Systematic Reviews in Software Engineering - A Cross Domain Study”. en. In: *e-Informatica Vol. XII* (2018), 2018, ISSN 1897–7979. DOI: 10.5277/E-INF180104 (cit. on p. 27).

[NYAT13] S. Nidhra, M. Yanamadala, W. Afzal, R. Torkar. “Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation”. In: *International Journal of Information Management* 33.2 (Apr. 2013), pp. 333–355. DOI: 10.1016/j.ijinfomgt.2012.11.004 (cit. on p. 24).

[PVK15] K. Petersen, S. Vakkalanka, L. Kuzniarz. “Guidelines for conducting systematic mapping studies in software engineering: An update”. In: *Information and Software Technology* 64 (Aug. 2015), pp. 1–18. DOI: 10.1016/j.infsof.2015.03.007 (cit. on p. 35).

[RCCM10] H. Ramampiaro, D. Cruzes, R. Conradi, M. Mendona. “Supporting evidence-based Software Engineering with collaborative information retrieval”. In: *Proceedings of the 6th International ICST Conference on Collaborative Computing: Networking, Applications, Worksharing*. IEEE, 2010. DOI: 10.4108/icst.collaboratecom.2010.9 (cit. on pp. 24, 29).

[RSSM10] M. Riaz, M. Sulayman, N. Salleh, E. Mendes. “Experiences Conducting Systematic Reviews from Novices’ Perspective”. In: BCS Learning & Development, Apr. 2010. DOI: 10.14236/ewic/ease2010.6 (cit. on pp. 24, 29).

[Smi20] Smithsonian Astrophysical Observatory. *The SAO/NASA Astrophysics Data System (SAO/NASA ADS) API Documentation*. 2020. URL: https://github.com/adsabs/adsabs-dev-api/blob/master/Search_API.ipynb (visited on 08/26/2020) (cit. on p. 38).

[SN07] M. Staples, M. Niazi. “Experiences using systematic review guidelines”. In: *Journal of Systems and Software* 80.9 (Sept. 2007), pp. 1425–1437. DOI: 10.1016/j.jss.2006.09.046 (cit. on pp. 24, 29, 30).

[Spr20] Springer. *Springer Link API Documentation*. 2020. URL: https://dev.springernature.com/docs (visited on 08/26/2020) (cit. on pp. 37, 39).

[SR09] M. Skoglund, P. Runeson. "Reference-based search strategies in systematic reviews". In: BCS Learning & Development, Apr. 2009. DOI: 10.14236/ewic/ease2009.4 (cit. on p. 17).

[YBLW19] V. Yussupov, U. Breitenbücher, F. Leymann, M. Wurster. "A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools". In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. ACM, Dec. 2019. DOI: 10.1145/3344341.3368803 (cit. on pp. 15, 19, 20).

All links were last followed on October 27, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature