JabRef: BibTeX-based literature management software

Oliver Kopp, Carl Christian Snethlage, Christoph Schwentker

Abstract

JabRef is a comprehensive literature management tool built entirely around the BibTeX data format. JabRef's features include web search, import and export in various formats, file attachment management, and quality checks of entries. Moreover, JabRef supports conducting systematic literature reviews. As an open-source initiative, JabRef also provides a suitable platform for software engineering education, particularly for newcomers to the field.

1 Overview on JabRef

JabRef is a comprehensive, open-source, cross-platform reference management tool that offers a rich set of features. The tool has its roots dating back to a merge of Bibkeeper[1] and JBibtexManager[2]. Since its first release on November 30, 2003,[3] JabRef has come a long way in serving researchers and writers. JabRef not only boasts a user-friendly graphical interface,[4] but also provides a command line interface.[5] It empowers users to gather, organize, and share bibliographic references with ease. What sets JabRef apart from other reference management tools is its exclusive use of BibTeX as its internal file format [13]: All entries and metadata are stored inside a single BibTeX file, providing seamless integration with numerous typesetting systems.

Since JabRef 3.1, released on December 24, 2015, JabRef keeps the BibTeX entries unmodified by default. When a user changes an entry, JabRef applies its formatting rules to that entry and only modifies that single entry. All others remain untouched.[6] The ordering of entries in BibTeX files is also preserved. If a defined ordering is desired, a user can opt in to JabRef's feature to sort the file on save. All in all, JabRef can be used seamlessly with other third-party tools for editing or consuming BibTeX files.

Additionally, JabRef provides explicit support for BibLaTeX. This is visible to the user at several places. For instance, when adding a new entry, JabRef first asks for the entry type. That dialog is filled with the entry types available in BibTeX and BibLaTeX, respectively, depending on the library's setting. It is always possible to change the library mode between BibTeX and BibLaTeX at a later point. JabRef's functionality to check for non-standard field content is aware of BibLaTeX's field content handling. For instance, when page numbers are connected by a single hyphen, the integrity check warns only when in BibTeX mode.

To help guide readers through JabRef in more detail, the overall user interface of JabRef is presented in Section 2. The internal data model is discussed in Section 3. Subsequently, selected features are outlined in Section 4. Important facts on JabRef's open-source development and especially the onboarding of students are presented in Section 5. Collaboration is a hot topic and the current state of JabRef is discussed in Section 6. Finally, contribution possibilities are outlined in Section 7.

2 The UI of JabRef

Figure 1 shows the main window of JabRef.

The *search* bar ❶ in the toolbar enables searching in the library. Besides free text search, the search supports search in selected fields as well as a regular-expression based search. It is also possible to search in the attached PDFs.[7]

The *web search* ❷ side pane allows for searching for bibliographic entries out of more than 20 catalogs. For a concrete search, the user inputs a search query as a combination of keywords using the syntax of Apache Lucene[8]. This query is transformed to the syntax of the respective search catalog and subsequently used to query the corresponding catalog API. The returned entries are displayed as a list of bibliographic references, for some catalogs even with abstract. The user can now select those references to be imported into the library. All in all, the use of a general query syntax enables switching online resources transparently.[9]

The *groups* ❸ side pane allows for grouping references into hierarchical collections based on keywords, search terms, or manual assignment.[10]

[1] bibkeeper.sourceforge.net/

[2] ctan.org/pkg/jbibtexmanager

[3] sourceforge.net/p/jabref/news/2003/11/jabref-10-released/

[4] docs.jabref.org/getting-started

[5] docs.jabref.org/advanced/commandline

[6] Two exceptions: 1) If "save actions" is activated, JabRef rewrites entries updated by the save actions (Section 4.3). 2) Fields migrated during load. For instance, the update of the storage of JabRef's entry grouping, the upgrade of "markings" to JabRef groups, converting "special fields" from keywords to separate fields, and converting the field `review` to `comment`. The JabRef team is currently investigating the automatic update to provide improved transparency (JabRef#10370).

[7] docs.jabref.org/finding-sorting-and-cleaning-entries/search

[8] lucene.apache.org/core/2_9_4/queryparsersyntax.html

[9] docs.jabref.org/collect/import-using-online-bibliographic-database

[10] docs.jabref.org/finding-sorting-and-cleaning-entries/groups

doi.org/10.47397/tb/44-3/tb138kopp-jabref

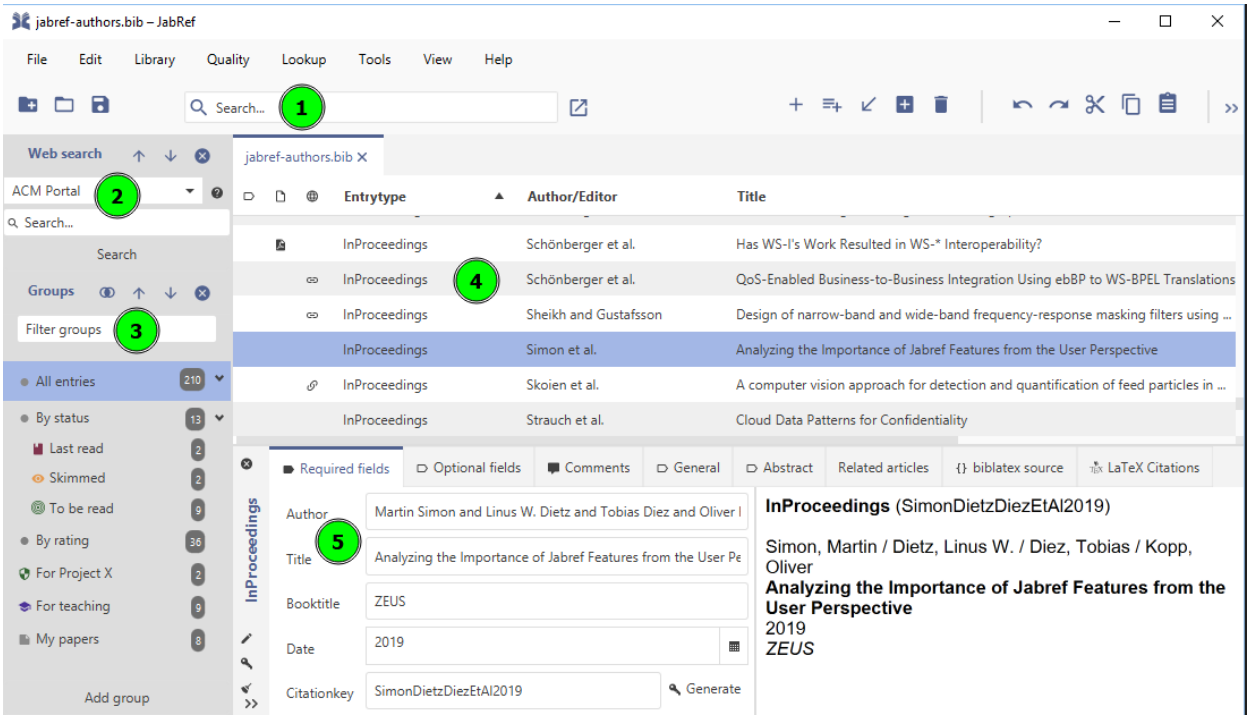

Figure 1: JabRef's main window

All entries are displayed in the *entry table* 4. The column headers represent field names. The contents of the table are an interpreted version of the field content: The displayed version is text that looks approximately as it would be output by BibTeX: Basic LaTeX formatting commands are interpreted and converted to Unicode to provide a better preview.

The *entry editor* 5 categorizes the BibTeX fields and provides editing capabilities for field values. The categorization is rendered as tabs in the entry editor. The distinction between required and optional fields is based on the field definition in the common BibTeX styles [11]. JabRef adds additional categories to support unique JabRef features.

The "Comments" tab lists comments to the entry. Starting with JabRef 5.10, user-specific comments are supported: If a BibTeX field name follows the format `comment-X`, it will be treated as a comment by user `X`.

The "General" tab groups general metadata of the entry. Examples are its DOI, keywords, groups of the entry, and a rating (1 to 5 stars) of the entry. Finally, attached files are listed here.

When PDFs are associated with an entry (using the `file` field), the "Annotations" tab displays highlighted text and PDF comments in a list format. Clicking on a particular item opens the PDF viewer at the respective location.

The "Abstract" tab is used to display the abstract field of the associated entry.

Occasionally, a BibTeX entry may contain fields that are not standard fields or recognized by JabRef. In such instances, these unrecognized fields are gathered and displayed under the "Other fields" tab.

The entry editor also offers literature recommendations, based on the currently selected article. Recommendations are provided by the Mr. DLib service [3], from a corpus with several million open access publications.[11] Mr. DLib uses several recommendation algorithms, including content-based filtering and collaborative filtering. The results are displayed inside the "Related articles" tab.

The "BibTeX source" tab displays the raw BibTeX code, which users can modify directly. Once the editing process is complete, JabRef parses the updated content and automatically populates the corresponding fields in the entry editor.

The "LaTeX citations" tab establishes a link between a LaTeX document and the citations used. JabRef recursively searches the directory of the BibTeX file for `*.tex` files and checks for `\cite` commands being included. If the given key matches the currently selected entry, JabRef displays the name of the `tex` file, with line and column number.

[11] `mr-dlib.org/`

The entry editor also provides a complete rendering of the entry. The layout can be configured either by a templating language custom to JabRef or by choosing a style from the Citation Style Language [19].[12] The custom template language utilized in JabRef comes with the significant advantage of being able to render fields that are not necessarily relevant to citations, such abstracts, comments, or other user-defined fields.

3 BibTeX-based internal data model

Since its inception, JabRef is implemented in Java and JabRef's internal data model is based directly on BibTeX.

On the technical side, the contents of a BibTeX entry is implemented as a map that links a `Field` to a `String`. A `Field` is a Java interface abstracting from concrete fields.[13] For instance, one concrete field class is `org.jabref.model.entry.field.StandardField`, which models all fields available in standard BibTeX and BibLaTeX. An example is `AUTHOR("author", FieldProperty.PERSON_NAMES)`. Based on the field property, the JabRef entry editor knows that the value editing component for that field should support person names. This is especially important for autocompletion. Besides `StandardField`s, there are `IEEEField`, `BiblatexSoftwareField`, and more. If JabRef does not recognize a field, it is stored in `UnknownField`, making JabRef extensible to unknown fields.

The same concept is implemented for entry types: The Java interface is `EntryType` with possible implementations `StandardEntryType`, `IEEETranEntryType`, `BiblatexSoftwareEntryType`, and `UnknownEntryType`.

4 Selected features

This section explains selected features of JabRef: Importing bibliographic data from files (Section 4.1) or the web (Section 4.2), quality assurance of entries (Section 4.3), and JabRef's feature for supporting systematic literature reviews (Section 4.4).

4.1 Importing MODS, EndNote, Citavi, and PDFs

JabRef can import other formats, such as MODS,[14] EndNote, and Citavi.

JabRef also offers the possibility to retrieve BibTeX data from PDF files, in four different ways:

1. reading embedded BibTeX data (e.g., generated by the `authorarchive` package [2]);
2. reading BibTeX text on the first page (e.g., generated by the `coverpage` package [8]);
3. interpreting the first PDF page and applying heuristics for the correct BibTeX data; and
4. using a self-hosted instance of the GROBID machine learning library [4].

In each case, the user drops a PDF between list entries in the main table, JabRef uses all these possibilities and merges the result using heuristics to retrieve the most complete BibTeX entry for the given PDF file. JabRef can also import a whole directory of PDF files and complete the current BibTeX library with the PDF files missing in that library.

4.2 Importing references using a web browser

JabRef provides a browser plugin that allows importing the reference represented by a website into JabRef.[15] An icon is displayed in the browser address bar. As soon as that icon is clicked, the plugin fetches the bibliographic data and sends it to JabRef. Then, JabRef presents an import inspection window, which allows for double-checking of the generated citation data and an assignment to one of the opened libraries.

4.3 Quality assurance of entries

JabRef offers three features to ensure the quality of entries: The integrity check,[16] the cleanup of entries,[17] and the automatic cleanup upon save.[18]

The *integrity check* checks the current BibTeX library for issues such as duplicate BibTeX keys, whether pages are connected via two hyphens in the case of BibTeX mode, and whether the field value contains HTML-encoded characters. Any issues are reported as a list in a separate window. When clicking on an issue, the respective entry is selected in JabRef's main table.

For some BibTeX issues, a cleanup is available: The user can start a cleanup dialog (Figure 2) and select actions to improve the quality of the bibliographic entries. For instance, some BibTeX entries have the DOI contained in the `note` field. JabRef

[12] `docs.jabref.org/setup/preview#layouts-styles`

[13] `github.com/JabRef/jabref/blob/refs/heads/main/src/main/java/org/jabref/model/entry/field/Field.java#L8`

[14] Metadata Object Description Schema by the Library of Congress: `www.loc.gov/standards/mods/`

[15] `docs.jabref.org/collect/jabref-browser-extension`

[16] `docs.jabref.org/finding-sorting-and-cleaning-entries/checkintegrity`

[17] `docs.jabref.org/finding-sorting-and-cleaning-entries/cleanupentries`

[18] `docs.jabref.org/finding-sorting-and-cleaning-entries/saveactions`

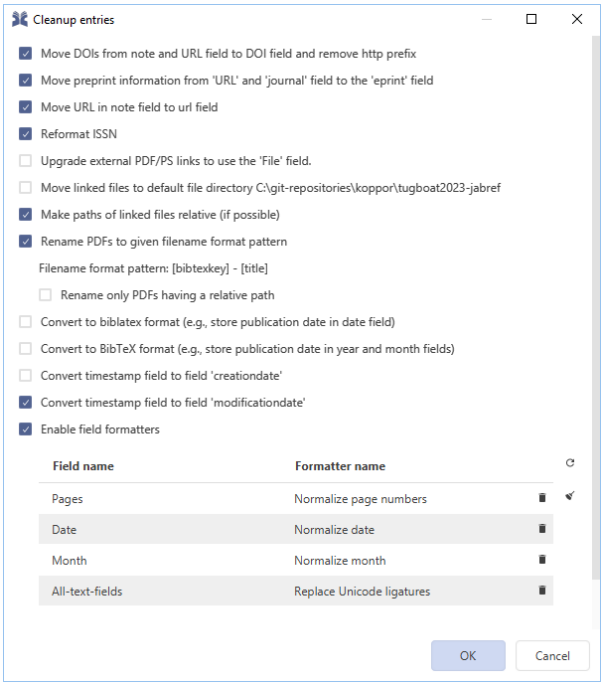

Figure 2: Options to clean up entries

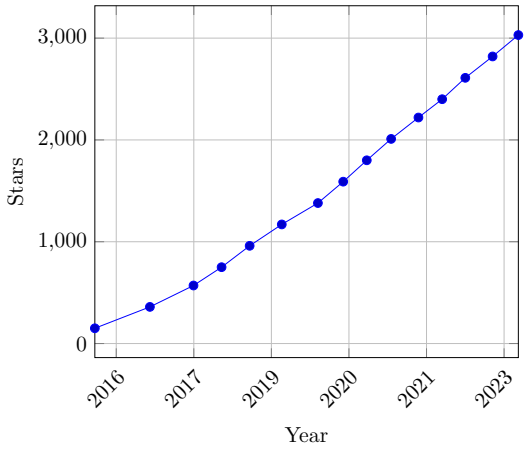

Figure 3: JabRef's stars on GitHub

can move the DOI from there to the `doi` field. Other cleanup actions include the normalization of page ranges and the normalization of name lists, which ensures that all names are formatted in the form "lastname, firstname".

Finally, the cleanup actions can also be triggered on save ("save actions"). For instance, the user can choose to always normalize the list of authors.

4.4 Systematic literature reviews

Systematic literature reviews (SLRs) are comprehensive examinations of existing research on a particular topic, conducted according to a well-defined and transparent methodology [7]. The most difficult challenge researchers face when conducting an SLR is during the search step [1, 16]. Commonly used e-libraries in the domain of computer science research, such as IEEE, arXiv, and ACM, do not support easy bulk access, which is key to the SLR method [1]. JabRef fills this gap with its special SLR feature [16]. The main concept of the implementation is to a) build on git and BibTeX to track the state of papers (excluded, included, ...); and b) transform a generic query syntax to the different query formats of each publisher.

JabRef provides an SLRwizard guiding the user through the creation of an SLR [16]. First, the user is asked for the intention of the SLR and in a next step JabRef transforms the queries and sends them to each catalog. The resulting BibTeX data is stored in one BibTeX file per catalog. The user then reviews the fetched entries and removes entries if they meet the exclusion criteria. This is recorded as a git commit. When fetching is re-triggered, the deletions are honored and only "real" new data is added to the SLR repository.

5 JabRef as an open-source project

JabRef has a long history as an open-source project. The first commit was on October 14, 2003. It moved to GitHub in 2015 and gained popularity (Figure 3). For the last 60 months, JabRef shows the following statistics:[19]

- 6 new committers per month on average,
- 16 total active committers per month on average,
- 37 percent of committers were new committers,
- 95 commits per month on average,
- 5.94 commits per committer per month on average, and
- Cumulative total committers grew by 402 (from 261 to 663).

The constantly increasing number of new contributors is most likely due to the use of JabRef in software engineering education in universities.[20] JabRef positions itself as a project that is more complex than programming exercises for beginners, but also less complex than distributed systems. This allows students to focus on the software engineering aspects of a real-world program. Examples of such aspects include: proper requirements specification and a proper design document, and also proper test cases

[19] The committer statistics have been gathered by git-pulse, available at `github.com/git-pulse/tools`.

[20] A list of courses is provided at `devdocs.jabref.org/teaching`.

Title	Size	Main Focus	Issue Understanding Effort	Implementation Effort	Testing Effort	Status
Integrate check-bib-for... #348	small	logic + UI	low	medium	low	Free to take
Add support for scite.ai #375	small	logic + UI	low	high	medium	Free to take
Move data to wiki data #49	small	logic	medium	low	low	Free to take

Figure 4: Issue board for selecting an issue to work on

and the process of using GitHub pull requests to contribute to an open-source project. In some courses, the schedule to contribute follows coordinated lectures, ensuring sustainability of the concepts taught (cf. [9]). We also try to make each project a success, which should also contribute to sustainability (cf. [17]).

One major issue for newcomers is to find suitable tasks [12]. In JabRef, we have tagged "good first issues" [6] for a long time. The intention of these issues was to provide the contributor with a self-guided tour through the concept of JabRef to be able to contribute more in a next issue. Since students participating in a course are primarily motivated to complete their assignment and then move on to other assignments, the required effort to understand JabRef was not usually made.

In 2022, we changed this approach. We moved away from recommending "good first issues" and turned to a more fine-grained categorization schema to present issues (Figure 4): We introduced a project board, where each issue is categorized by the size of the project, the main focus, the issue understanding effort, the implementation effort, the testing effort, and the status of the issue.

The size of a project is either small, medium, or large. We have seen that some students tend to like to work on the UI, while others prefer to do backend-like development. Thus, we introduced "main focus", which may be on one (or both) of the two main components of JabRef: The logic and the user interface.

Moreover, students tend to underestimate the task; in particular, they assume that they are able to understand a task in a very short time, implement it, and can get by with simple tests (if at all). Therefore, we aim to make these efforts more explicit.

The *issue understanding effort* captures the estimated effort to understand the context and the problem described in the issue. Although one might think that literature management is a simple task, there are details to think about. One example is adding PDF files that contain multiple papers.[21] Here, for instance, students need to understand that extracting BibTeX data from such a PDF is different from PDFs containing only one paper — and that this leads to significant code changes in JabRef.

The *implementation effort* captures the estimated effort to implement the main functionality. In general, effort estimates are not exact [10]. We merely view this category as a coarse-grained guideline for selecting an issue. For instance, an improved error message when downloading a file is quite different from developing and implementing a new concept for the entry editor in JabRef.

The *testing effort* is highlighted to stress that JabRef requires more than merely implementing a feature; it necessitates a comprehensive approach in line with the broader scope of software engineering, which goes far beyond writing code [18]. Furthermore, the inclusion of tests alongside new features is crucial for us, as it enables regression testing when subsequent changes may affect existing behavior.

Regarding other approaches, Vargovich et al. [14] take a different approach: they attempt to automatically label issues based on the domain of the APIs involved in the solution (e.g., User Interface, Test, Databases). Our manual approach distinguishes between "logic" and "UI" as we consider these domains to be the key distinguishing components (similar to backend and frontend development). We also distinguish according to the general size of the project and the difficulty of understanding an issue.

From a resource perspective, JabRef's software engineering education by JabRef is backed by the non-profit organization JabRef e.V., which provides the legal framework, the resources, and the network.

6 Collaboration

One of the most requested features is online collaborative work. Currently, JabRef offers collaboration based on the BibTeX file, because JabRef can detect if the file changed on disk. Thus, there

[21] github.com/JabRef/jabref/issues/8128

are currently two ways to collaborate directly:[22] 1) using version control systems such as `git`; 2) using file-synchronization services such as Dropbox or OneDrive. Both approaches require advanced users: regarding version control systems, not all LaTeX users use them [15]; and regarding file-synchronization services, each external update leads to a notification, which needs to be handled.

JabRef also offers collaboration using an SQL connection to a PostgreSQL database.[23] The changes are synchronized instantly to the server and then distributed to the other connected JabRef instances. It is possible to automatically write the current state to a local file, which can then be consumed by BibTeX tools. Additionally, JabRef offers connecting to a MySQL database. However, live synchronization is not supported there and the user has to push and pull changes manually.

Generally, there are several online services for collaboration with BibTeX: CiteDrive[24] provides a dedicated online service for BibTeX users, Overleaf [5] is the leading LaTeX online collaboration tool, and JabRef developers are also working on a web-based online collaboration platform.[25] In parallel, there are efforts to integrate JabRef with CiteDrive. Once accomplished, users will be able to utilize CiteDrive for BibTeX collaboration and take advantage of its Overleaf integration,[26] while also using JabRef for more advanced BibTeX management.

7 Contribution possibilities

JabRef offers multiple points of access for possible contributions. Besides direct code contributions, JabRef calls for improvements of the user documentation, which is currently maintained using Markdown in a repository on GitHub.[27] Lastly, there is a call for contributions page[28] describing even more possibilities to contribute.

Acknowledgements Our sincere gratitude goes to the users who have selected JabRef for their university work, demonstrating trust and confidence in our tool and our team. We extend heartfelt thanks to the dedicated volunteers who tirelessly support JabRef's users and code contributors. Their commitment to providing assistance in our online forum and meticulous review of pull requests is vital to our community. We also would like to thank all the users who constantly test the latest versions and give valuable feedback on the functionality. Lastly, we wish to express profound appreciation to all the Ph. D. students and postdocs who invest precious non-research time into JabRef. Their contributions enrich our project and reflect a commendable spirit of collaboration and innovation.

[22] `docs.jabref.org/collaborative-work/sharedbibfile`
[23] `docs.jabref.org/collaborative-work/sqldatabase`
[24] `www.citedrive.com/`
[25] `github.com/jabref/jabrefonline`
[26] `bibtex.eu/overleaf-citedrive/`
[27] `github.com/JabRef/user-documentation`
[28] `contribute.jabref.org/`

References

[1] A. Al-Zubidy, J.C. Carver. Identification and prioritization of SLR search tool requirements: an SLR and a survey. *Empirical Software Engineering*, 24(1):139–169, May 2018. `doi.org/10.1007/s10664-018-9626-5`

[2] A.D. Brucker. *The authorarchive package*, Feb. 2023. `ctan.org/pkg/authorarchive`

[3] S. Feyer, S. Siebert, et al. Integration of the Scientific Recommender System Mr. DLib into the Reference Manager JabRef. In *Advances in Information Retrieval*, Lecture Notes in Computer Science. Springer International Publishing, 2017. `doi.org/10.1007/978-3-319-56608-5_80`

[4] GROBID. `github.com/kermitt2/grobid`, 2008–2023.

[5] T. Hejda. TeX Live and Overleaf revisited. *TUGboat* 44(2), 2023. `doi.org/10.47397/tb/44-2/tb137hejda-overleaf-tl`

[6] H. Horiguchi, I. Omori, M. Ohira. Onboarding to open source projects with good first issues: A preliminary analysis. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021. `doi.org/10.1109/saner50967.2021.00054`

[7] B. Kitchenham, S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University, 2007.

[8] O. Kopp, M. Mühlich. *The coverpage package*, 2006. `ctan.org/pkg/coverpage`

[9] J. Ludewig. Erfahrungen bei der Lehre des Software Engineering. In *Software Engineering im Unterricht der Hochschulen, SEUH 11*. dpunkt, 2009.

[10] P. Matsubara, I. Steinmacher, et al. Trust yourself! Or maybe not: factors related to overconfidence and uncertainty assessments of software effort estimates. In *Brazilian Symposium on Software Engineering*. ACM, 2021. `doi.org/10.1145/3474624.3474643`

[11] F. Mittelbach, U. Fischer. *The LaTeX Companion*, ch. Bibliography Generation, pp. 375–468. Addison-Wesley, third ed., 2023.

[12] I. Steinmacher, C. Treude, M.A. Gerosa. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, 36(4):41–49, July 2019. doi.org/10.1109/ms.2018.110162131

[13] Technical University of Munich. Reference management software comparison – 9th update, 2022. doi.org/10.14459/9.COMPARISON_REFERENCE_MANAGEMENT_2022

[14] J. Vargovich, F. Santos, et al. GiveMeLabeledIssues: An open source issue recommendation system. In *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023. doi.org/10.1109/msr59073.2023.00061

[15] B. Veytsman. Using Overleaf for collaborative projects: First impressions and lessons learned. *TUGboat* 41(2):179–181, 2020. tug.org/TUGboat/tb41-2/tb128veytsman-overleaf.pdf

[16] D. Voigt, O. Kopp, K. Wild. Systematic literature tools: Are we there yet? In *ZEUS*, vol. 2839 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[17] M. Wetzel, H. Röder. Studentische Projekte: erfolgreich oder realistisch? In *Software Engineering im Unterricht der Hochschulen, SEUH 11*, pp. 17–28. dpunkt, 2009.

[18] T. Winters, T. Manshreck, H. Wright. *Software Engineering at Google: Lessons Learned from Programming Over Time*. O'Reilly Media, 2020.

[19] R.M. Zelle, B.M. Wiernik, et al. CSL 1.0.2 specification, 2015. docs.citationstyles.org/en/stable/specification.html

◇ Oliver Kopp
Sindelfingen, Germany
https://github.com/koppor
ORCID 0000-0001-6962-4290

◇ Carl Christian Snethlage
Zwiesel, Germany
https://github.com/calixtus

◇ Christoph Schwentker
Paderborn, Germany
https://github.com/Siedlerchr