Bidirectional Paper-Repository Tracing in Software Engineering

Daniel Garijo
Universidad Politécnica de Madrid
Madrid, Spain
daniel.garijo@upm.es

Miguel Arroyo
Universidad Politécnica de Madrid
Madrid, Spain
miguel.arroyo.marquez@alumnos.upm.es

Esteban Gonzalez
Universidad Politécnica de Madrid
Madrid, Spain
egonzalez@fi.upm.es

Christoph Treude
The University of Melbourne
Melbourne, Australia
christoph.treude@unimelb.edu.au

Nicola Tarocco
CERN
Geneva, Switzerland
nicola.tarocco@cern.ch

ABSTRACT

While computer science papers frequently include their associated code repositories, establishing a clear link between papers and their corresponding implementations may be challenging due to the number of code repositories used in research publications. In this paper we describe a lightweight method for effectively identifying bidirectional links between papers and repositories from both LaTeX and PDF sources. We have used our approach to analyze more than 14000 PDF and Latex files in the Software Engineering category of Arxiv, generating a dataset of more than 1400 paper-code implementations and assessing current citation practices on it.

CCS CONCEPTS

• **Applied computing** → **Document analysis**; • **Software and its engineering** → **Software libraries and repositories**.

KEYWORDS

Research software, article analysis, software citation, Open Science

ACM Reference Format:
Daniel Garijo, Miguel Arroyo, Esteban Gonzalez, Christoph Treude, and Nicola Tarocco. 2024. Bidirectional Paper-Repository Tracing in Software Engineering. In *21st International Conference on Mining Software Repositories (MSR '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643991.3644876

1 INTRODUCTION

Research software (i.e., the scripts and tools developed to support the results described in scientific publications) [11] is increasingly recognized as a key asset in academic curricula.[1]

In order to provide researchers with the appropriate credit, the scientific community has proposed software citation principles [21] and formats [5] that have been adopted by popular code platforms such as GitHub.[2]

While studies have started assessing the relationship between code repositories and academic publications [4, 7, 22] to date, it remains challenging 1) to establish a clear link between research papers and their corresponding software implementations; and 2) to identify whether best citation practices for software are being followed both from articles and their corresponding code repositories.

In this paper, we propose a method to connect scientific papers with their corresponding code repositories (out of all the tool URLs used in a paper). Our approach searches both articles and code repositories for mutual references, creating a bidirectional link when they are found. Our contributions include:

(1) Two lightweight and reusable article-code repository extraction pipelines, both for Latex and PDF sources.
(2) A **dataset**[3] **of 1485 bidirectional correspondences** between articles and code implementations in the *Software Engineering* category in Arxiv (CS.SE).
(3) A study of the software citation best practices in our extracted dataset.

Ultimately, we aim to promote the importance of traceability, emphasizing its significance to both the academic and developer communities.

The rest of the paper is structured as follows: Section 2 describes initiatives related to our work. Section 3 describes our extraction pipelines, while Section 4 and Section 5 describe our dataset and study, respectively. Section 6 discusses the benefits and limitations of our work, and Section 7 concludes the paper.

All code used to calculate the results of this paper can be found on Github[4] (v0.0.1) [3].

2 RELATED WORK

A number of initiatives have explored the role of research software in scientific publications [4, 7, 22] and the importance of academic credit in software development [14]. Platforms like PapersWithCode[5] list the implementations associated with paper publications, based on human feedback, while works like Heumüller et al. [13]

[1] https://sfdora.org/read/

[2] https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-citation-files
[3] https://doi.org/10.5281/zenodo.10307603
[4] https://github.com/oeg-upm/bidirectional_paper_repository_scripts
[5] https://paperswithcode.com/

MSR 2024, April 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0587-8/24/04…$15.00
https://doi.org/10.1145/3643991.3644876

rely on manual curation analysis of hundreds of papers in computer science engineering to assess the availability of their implementations. These efforts result in high-quality paper-repository implementation datasets, but do not assess code citation practices and require significant manual effort to generate and curate.

Other analyses such as Hata et al. [12] focus on describing the challenges of link-paper traceability, while Wattanakriengkrai et al. [22] look at GitHub code repositories to find links pointing back to papers, in order to quantify the frequency of these links. At a more granular level, Inokuchi et al. [15] also search for citation links in code comments. The scope of these works is different from ours, where we detect and study the bidirectional relationships between papers and code.

In recent years, initiatives like Softcite [6] have started detecting tool names from research articles. Istrate et al. [16] build on Softcite to generate a large extraction of tool mentions in millions of publications. The main focus of these works is finding tool names to detect citation networks, rather than detecting tool implementations associated with each publication.

Schindler et al. [20] go a step further by assessing the role of a tool mention in a publication (i.e., whether it was created, reused, or referenced). However, to date, this remains an open research challenge.[6]

Finally, recent work [18] proposes an automated method for extracting implementations of publications and assessing their README files to label their metadata (similar to Kelley and Garijo [17]). However, the implementation and training data associated with this effort are not openly accessible, and the tool used for PDF to text generation presents limitations when detecting footnotes (a common means for referring to software repositories). Instead, our approach reliably detects tool-paper implementations, which we use to assess current citation practices.

3 EXTRACTING CODE REPOSITORIES FROM RESEARCH PAPERS

Our aim is to propose a lightweight, precise method to create a dataset with the main code implementations of a corpus of research publications. This may be challenging, since many code repositories may be mentioned in a paper. Our rationale is to select those code repositories mentioned in a paper that include a citation back to the paper itself, i.e., where we find a bidirectional paper-code repository link. Note that a paper may have more than one associated code repository.

Figure 1 shows an overview of the two methods used to detect the bidirectionality between a paper and a code repository. First, if a code repository (e.g., GitHub, GitLab) is mentioned in a paper, we access it and look in the corresponding README file and citation files (Bibtex, CITATION.CFF) whether there is a link back to the paper. This link is often found through a Digital Object Identifier (DOI), but in some cases the article may have been accepted in another venue, and hence the DOI does not correspond to Arxiv anymore. In those cases, we match the papers with their corresponding title (exact match).

[6]Community efforts include https://github.com/karacolada/SoftwareImpactHackathon2023_SoftwareCitationIntent

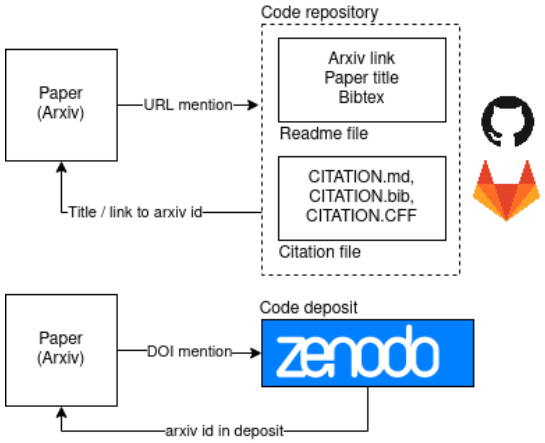

Figure 1: Methods used to detect bidirectionality links between papers and code

Second, if a paper includes a link to a software deposit such as Zenodo, we download the contents and look for a link back in the description and README file of the deposit.

Given that papers may be available in different formats, we propose two main means for link extraction: from Latex sources and from PDF. We briefly describe them below:

- Our *Latex extraction pipeline*[7] [10] takes as input the Latex sources of a project, e.g., from Arxiv, and produces as output a CSV with the bidirectional links found by using regular expressions. All latex sources are merged into a single file, which is the one used for the extraction. With all candidate URLs, we use the GitHub API to retrieve paper metadata and the README file of a target repository.

- Our PDF pipeline[8] [2] takes as input the PDF file of a publication and converts it into text using Apache Tika,[9] a lightweight PDF to text converter. Tika was chosen ahead of other popular options such as GROBID [1] due to its speed and its ability to correctly extract the text in footnotes. Similar to the Latex pipeline, we use regular expressions to retrieve the list of candidate URLs from the resultant text. We also use SOMEF [17, 19], a Software Metadata Extraction framework that looks for citation content in the whole code repository, not just the README file.

The rationale for the development of both pipelines was to assess the robustness of our proposed method and to take into account any potential links that may be missing. For example, in some cases, the PDF to text extraction breaks links in several lines (making them hard to detect), while in others the only available source is PDF, not Latex.

In order to assess our extraction pipelines, two of the authors annotated a gold corpus of 150 Arxiv papers by hand.[10] The corpus includes several types of bi-directionality paper-repository links (e.g., repositories including just the title of the paper, Arxiv URLs or DOIs in the README file, using citation files, repositories in different code platforms, etc.). In the rare cases where annotators

[7]https://github.com/ctreude/SoftwareImpactHackathon2023_BiDirectional
[8]https://github.com/SoftwareUnderstanding/RSEF
[9]https://github.com/chrismattmann/tika-python
[10]https://doi.org/10.5281/zenodo.10316689

Pipeline	Precision	Recall	F1
Latex	1	0.7	0.83
PDF	1	0.94	0.97

Table 1: Precision, recall and F1 score for the bidirectional link extraction pipelines

disagreed about the presence of an Arxiv URL or DOI of the paper in a repository, a third annotator, also an author, reviewed and joined the discussion to reach a decision on the final value of the bi-directionality. Table 1 shows the results of both pipelines on our gold standard. Both pipelines are very precise (no incorrect links detected) with a high recall (0.7 for Latex and 0.94 for PDF files). Some of the errors are derived from heterogeneous PDF and latex formatting errors, and article-repository title mismatches as discussed in Section 6.

4 BIDIRECTIONAL PAPER-REPOSITORY DATASET

We used the Arxiv API[11] to download all available papers from the Computer Science - Software Engineering category (CS.SE, the one closest to our expertise), in order to assess our approach. In total, we downloaded 14760 pdf papers that range from 1998 until Nov 2023. 10826 Latex sources were found for those papers and downloaded as well.

In total, our Latex and PDF pipelines found 1410 articles with 1485 bidirectional links to code repositories (some articles contain more than one bidirectional implementation). Of them, 592 papers were found by both pipelines, while an additional 780 were found only by the PDF pipeline. This makes sense, since nearly four thousand more papers were available in PDF for analysis. Interestingly, the Latex pipeline detects 46 links that were not found in its PDF counterpart. After a manual inspection, it turns out that many of these links are not available in the corresponding PDFs, but are commented out in the Latex files. We consider these links to be correct, as the code repositories exist and link back to their respective publications.

Our bidirectional article-repository dataset is available online [9].

5 STUDY: A BIDIRECTIONAL ANALYSIS IN SOFTWARE ENGINEERING

In order to demonstrate the usefulness of our dataset, we conducted a small study to assess the Open Science best practices in the Software Engineering papers in Arxiv. We propose the following research questions:

- RQ1: In all papers from the cs.SE category on ArXiv, how many provide direct links to GitHub and GitLab repositories or Zenodo archives?
- RQ2: How many of the provided links to GitHub or Zenodo in the cs.SE papers on arXiv are reciprocated with links from the platforms back to the paper?
- RQ3: What are the current citation practices in bidirectional repositories? Are best practices for software citation being adopted?

[11] https://info.arxiv.org/help/api/index.html

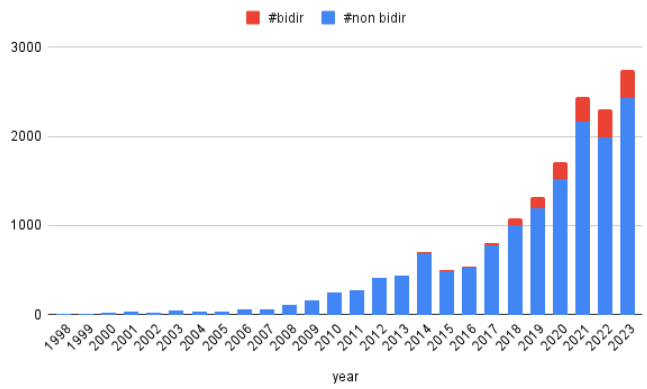

Figure 2: Number of papers with bidirectional links per year (bidir) in the CS.SE category until Nov 2023

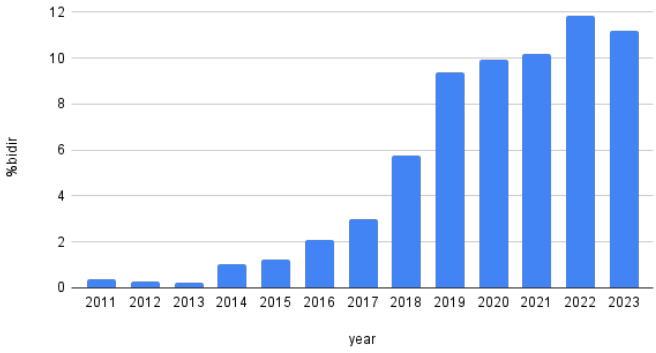

Figure 3: Percentage of papers with bidirectional links per year in the CS.SE category until Nov 2023

To answer RQ1, around 80% of detected links point to GitHub repositories. Table 2 shows the number of papers that include links, along with their platform. In our study, we restrict ourselves to GitHub, GitLab, and Zenodo links, as these are among the most popular platforms for depositing and versioning code [7]. Remarkably, an average of 1.5 links per paper are reported, which has grown considerably in the most recent years following the trends for Open Science.

As for RQ2, Figure 2 shows the number of papers available in Arxiv through the years, compared with the number of bidirectional links found in them (in red). Figure 3 drills down in the comparison by showing the percentage of papers with bidirectional links per year (note that the dataset covers until Nov 2023, hence the last year may not be fully complete). Overall, less than 15% of the papers report an implementation with a bidirectional link. Another fact to consider is that GitHub, GitLab, and Zenodo appeared after 2008, with the first bidirectional links appearing in 2014.

Finally, regarding RQ3, Table 3 shows the citation practices followed by all code repositories found. Since the Findable, Accessible, Interoperable and Reusable (FAIR) principles for data [23] were proposed in 2016, the Software Citation Principles [21] and the Citation

Pipeline	GitLab	Zenodo	GitHub	Total
PDF	7	273	1159	1439
Latex	0	40	638	678

Table 2: Type of links extracted by each pipeline. Some papers may have more than one bidirectional link

Citation practices	description	Bibtex	CFF	title
# Bidirectional papers	759	307	49	353

Table 3: Citation practices among the bidirectional dataset. We distinguish whether the DOI is included in the README file, whether the match is done against a citation done in Bibtex, whether a CFF file exists, or whether the title of the paper is included in the README file.

File Format [5] have been developed to aid in research software citation. Table 3 shows a summary of the citation methods followed by the code repositories linking back to papers. Despite current recommendations, it seems like a majority of researchers tend to include citation details directly in their README description or code repository title. Bibtex is still the preferred method to propose a software citation, while CFF is last (it was proposed in 2021, and its adoption may not yet be widespread). In many cases, researchers add a link to their papers using plain text in the README file. With the support for CFF in GitHub, we expect this number to grow in the following years.

6 DISCUSSION

With the adoption and continued development of the FAIR principles and Open Science best practices, there is a need for detecting research outputs associated with scientific publications. Our dataset and extraction pipelines are a unique resource for aiding in this task, since 1) they can be used to detect additional bidirectional implementations of research publications and 2) since our results are very precise, they can be used to create new corpora of annotations (paragraphs, footnotes, and references) to train ML models for citation intent classification.

As for limitations, in this paper we aimed at creating lightweight, precise pipelines that detect bidirectional links between papers and their code implementations. However, we do not address the detection of uni-directional links, i.e., those papers that link to a repository with their implementation, but where there is no link back to the paper. Addressing this issue would require additional text analysis to identify the sentiment of the paragraph describing the tool mention, similar to the approach proposed by Lin et al. [18]. Additionally, our approach is sensitive to changes in titles and DOIs: if an Arxiv paper links to a repository that links back to a paper with different DOI and title, we will not find a match. Comparing additional metadata (e.g., authors) may help address this issue.

As for the scope of our analysis, our dataset and pipelines focus on detecting links to GitHub, GitLab, and Zenodo. However, a number of software registries (e.g., ASCL,[12] SciCrunch,[13] etc.) may also be mentioned in domain-specific applications. Supporting the detection of new platforms is possible with approaches similar to Escamilla et al. [8], but detecting the link back to the corresponding paper from those applications requires specific API adapters.

Finally, the results shown in the study may be affected by different citation practices followed in research communities, countries and journals. Extending the scope of this study in other domains may uncover new citation practices.

7 CONCLUSIONS AND FUTURE WORK

In this paper we propose a dataset of bidirectional links between papers in the Software Engineering (cs.SE) category in Arxiv and their corresponding code implementations (GitHub, GitLab, and Zenodo), along with the Latex and PDF pipelines used for their extraction. Our extraction pipelines are very precise, lightweight, and can be easily adapted to other areas to detect the current software citation practices in different communities. We have illustrated the usefulness of our dataset by presenting a small study with three research questions, showing how software citation best practices are slowly being adopted by the software engineering community (although there is still room for improvement). An additional benefit of our dataset is its potential to generate corpora of software mentions, extending the current state of the art [6].

The dataset we have compiled and presented in this paper opens many avenues for future research, particularly to understand and enhance the synergy between academic publications and software development in the field of software engineering. A particularly interesting area of investigation is the co-evolution of academic papers and their associated software artifacts. Researchers may use this dataset to study how updates to Arxiv papers and corresponding code repositories are interlinked over time, shedding light on the iterative process of academic and software development. The dataset may also be used to investigate the impact of software on academic citation metrics: Do papers with associated open-source software attract more citations or academic attention? A comparative analysis of citation practices across different subfields within software engineering may reveal discipline-specific trends and practices. There is also an opportunity to examine the geographical and institutional distribution of these software projects, particularly in the context of Open Science policies. Are regions with more proactive Open Science mandates more likely to produce papers with associated software repositories? This line of inquiry may reveal how policy decisions at the governmental level influence the adoption of Open Science practices within the academic and software development communities.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Chan Zuckerberg Initiative for hosting the workshop that led to the development of this work. This work is supported by the Madrid Government (Comunidad de Madrid - Spain) under the Multiannual Agreement with Universidad Politécnica de Madrid in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the VPRICIT, and through the call Research Grants for Young Investigators from Universidad Politécnica de Madrid.

[12] https://ascl.net/about
[13] https://scicrunch.org/

REFERENCES

[1] 2008–2023. GROBID. https://github.com/kermitt2/grobid. swh:1:dir:dab86b296e3c3216e2241968f0d63b68e8209d3c

[2] Miguel Arroyo, Daniel Garijo, and Esteban González Guardia. 2023. Research Software Extraction Framework (RSEF). https://doi.org/10.5281/zenodo.10306762

[3] Miguel Arroyo, Daniel Garijo, and González Guardia. 2024. Scripts and data used in the paper the evaluation (v0.0.1). https://doi.org/10.5281/zenodo.10577754

[4] Domhnall Carlin, Austen Rainer, and David Wilson. 2023. Where is all the research software? An analysis of software in UK academic repositories. *PeerJ Computer Science* 9 (2023), e1546.

[5] Stephan Druskat, Jurriaan H. Spaaks, Neil Chue Hong, Robert Haines, James Baker, Spencer Bliven, Egon Willighagen, David Pérez-Suárez, and Olexandr Konovalov. 2021. *Citation File Format.* https://doi.org/10.5281/zenodo.5171937

[6] Caifan Du, Johanna Cohoon, Patrice Lopez, and James Howison. 2021. Softcite dataset: A dataset of software mentions in biomedical and economic research publications. *Journal of the Association for Information Science and Technology* 72, 7 (July 2021), 870–884. https://doi.org/10.1002/asi.24454

[7] Emily Escamilla, Martin Klein, Talya Cooper, Vicky Rampin, Michele C. Weigle, and Michael L. Nelson. 2022. The Rise of GitHub in Scholarly Publications. In *Linking Theory and Practice of Digital Libraries*, Gianmaria Silvello, Oscar Corcho, Paolo Manghi, Giorgio Maria Di Nunzio, Koraljka Golub, Nicola Ferro, and Antonella Poggi (Eds.). Springer International Publishing, Cham, 187–200.

[8] Emily Escamilla, Lamia Salsabil, Martin Klein, Jian Wu, Michele C Weigle, and Michael L Nelson. 2023. It's Not Just GitHub: Identifying Data and Software Sources Included in Publications. In *International Conference on Theory and Practice of Digital Libraries*. Springer, 195–206.

[9] Daniel Garijo, Miguel Arroyo, Esteban Gonzalez, Christoph Treude, and Nicola Tarocco. 2023. *Bidirectional dataset.* https://doi.org/10.5281/zenodo.10307603

[10] Daniel Garijo, Miguel Arroyo, Esteban Gonzalez, Christoph Treude, and Nicola Tarocco. 2023. Bidirectional paper-repository traceability. https://doi.org/10.5281/zenodo.10303607

[11] Morane Gruenpeter, Daniel S. Katz, Anna-Lena Lamprecht, Tom Honeyman, Daniel Garijo, Alexander Struck, Anna Niehues, Paula Andrea Martinez, Leyla Jael Castro, Tovo Rabemanantsoa, Neil P. Chue Hong, Carlos Martinez-Ortiz, Laurents Sesink, Matthias Liffers, Anne Claire Fouilloux, Chris Erdmann, Silvio Peroni, Paula Martinez Lavanchy, Ilian Todorov, and Manodeep Sinha. 2021. Defining Research Software: a controversial discussion. https://doi.org/10.5281/zenodo.5504016

[12] Hideaki Hata, Jin L. C. Guo, Raula Gaikovina Kula, and Christoph Treude. 2021. Science-Software Linkage: The Challenges of Traceability between Scientific Knowledge and Software Artifacts. arXiv:2104.05891 [cs.SE]

[13] Robert Heumüller, Sebastian Nielebock, Jacob Krüger, and Frank Ortmeier. 2020. Publish or perish, but do not forget your software artifacts. *Empirical Software Engineering* 25, 6 (Nov. 2020), 4585–4616. https://doi.org/10.1007/s10664-020-09851-6

[14] James Howison and James D. Herbsleb. 2013. Incentives and Integration in Scientific Software Production. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work* (San Antonio, Texas, USA) *(CSCW '13)*. Association for Computing Machinery, New York, NY, USA, 459–470. https://doi.org/10.1145/2441776.2441828

[15] Akira Inokuchi, Yusuf Sulistyo Nugroho, Supatsara Wattanakriengkrai, Fumiaki Konishi, Hideaki Hata, Christoph Treude, Akito Monden, and Kenichi Matsumoto. 2020. From Academia to Software Development: Publication Citations in Source Code Comments. arXiv:1910.06932 [cs.SE]

[16] Ana-Maria Istrate, Donghui Li, Dario Taraborelli, Michaela Torkar, Boris Veytsman, and Ivana Williams. 2022. A large dataset of software mentions in the biomedical literature. *arXiv preprint arXiv:2209.00693* (2022).

[17] Aidan Kelley and Daniel Garijo. 2021. A Framework for Creating Knowledge Graphs of Scientific Software Metadata. *Quantitative Science Studies* (11 2021), 1–37. https://doi.org/10.1162/qss_a_00167

[18] Jialiang Lin, Yingmin Wang, Yao Yu, Yu Zhou, Yidong Chen, and Xiaodong Shi. 2022. Automatic analysis of available source code of top artificial intelligence conference papers. *International Journal of Software Engineering and Knowledge Engineering* 32, 07 (2022), 947–970.

[19] A. Mao, D. Garijo, and S. Fakhraei. 2019. SoMEF: A Framework for Capturing Scientific Software Metadata from its Documentation. In *2019 IEEE International Conference on Big Data (Big Data)*. 3032–3037. https://doi.org/10.1109/BigData47090.2019.9006447

[20] David Schindler, Felix Bensmann, Stefan Dietze, and Frank Krüger. 2022. The role of software in science: a knowledge graph-based analysis of software mentions in PubMed Central. *PeerJ Computer Science* 8 (Jan. 2022), e835. https://doi.org/10.7717/peerj-cs.835

[21] Arfon M Smith, Daniel S Katz, and Kyle E Niemeyer. 2016. Software citation principles. *PeerJ Computer Science* 2 (2016), e86. https://doi.org/10.7717/peerj-cs.86

[22] Supatsara Wattanakriengkrai, Bodin Chinthanet, Hideaki Hata, Raula Gaikovina Kula, Christoph Treude, Jin Guo, and Kenichi Matsumoto. 2022. GitHub repositories with links to academic papers: Public access, traceability, and evolution. *Journal of Systems and Software* 183 (2022), 111117. https://doi.org/10.1016/j.jss.2021.111117

[23] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* 3, 1 (2016), 1–9. https://doi.org/10.1038/sdata.2016.18