# Sharing Software-Evolution Datasets: Practices, Challenges, and Recommendations

DAVID BRONESKE, German Centre for Higher Education Research and Science Studies, Germany
SEBASTIAN KITTAN, Otto-von-Guericke University Magdeburg, Germany
JACOB KRÜGER, Eindhoven University of Technology, The Netherlands

Sharing research artifacts (e.g., software, data, protocols) is an immensely important topic for improving transparency, replicability, and reusability in research, and has recently gained more and more traction in software engineering. For instance, recent studies have focused on artifact reviewing, the impact of open science, and specific legal or ethical issues of sharing artifacts. Most of such studies are concerned with artifacts created by the researchers themselves (e.g., scripts, algorithms, tools) and processes for quality assuring these artifacts (e.g., through artifact-evaluation committees). In contrast, the practices and challenges of sharing software-evolution datasets (i.e., republished version-control data with person-related information) have only been scratched in such works. To tackle this gap, we conducted a meta study of software-evolution datasets published at the International Conference on Mining Software Repositories from 2017 until 2021 and snowballed a set of papers that build upon these datasets. Investigating 200 papers, we elicited what types of software-evolution datasets have been shared following what practices and what challenges researchers experienced with sharing or using the datasets. We discussed our findings with an authority on research-data management and ethics reviews through a semi-structured interview to put the practices and challenges into context. Through our meta study, we provide an overview of the sharing practices for software-evolution datasets and the corresponding challenges. The expert interview enriched this analysis by discussing how to solve the challenges and by defining recommendations for sharing software-evolution datasets in the future. Our results extend and complement current research, and we are confident that they can help researchers share software-evolution datasets (as well as datasets involving the same types of data) in a reliable, ethical, and trustworthy way.

CCS Concepts: • **Software and its engineering → Software libraries and repositories**.

Additional Key Words and Phrases: datasets, research artifacts, software evolution, artifact sharing, replicability, reproducibility, research data management

**ACM Reference Format:**
David Broneske, Sebastian Kittan, and Jacob Krüger. 2024. Sharing Software-Evolution Datasets: Practices, Challenges, and Recommendations. *Proc. ACM Softw. Eng.* 1, FSE, Article 91 (July 2024), 24 pages. https://doi.org/10.1145/3660798

## 1 INTRODUCTION

Sharing research artifacts has become a vital concern for most researchers in any field, and has gained major attention in software-engineering research, too [Baker, 2016, Baldassarre et al., 2023, Childers and Chrysanthis, 2017, Frachtenberg, 2022, Hermann, 2022, Heumüller et al., 2020, Mendez et al., 2020, Méndez Fernández et al., 2019]. Making artifacts (e.g., software, data, protocols) used

Authors' Contact Information: David Broneske, German Centre for Higher Education Research and Science Studies, Hannover, Germany, broneske@dzhw.eu; Sebastian Kittan, Otto-von-Guericke University Magdeburg, Magdeburg, Germany; Jacob Krüger, Eindhoven University of Technology, Eindhoven, The Netherlands, j.kruger@tue.nl.


ACM 2994-970X/2024/7-ART91
https://doi.org/10.1145/3660798

for a piece of research available promises many benefits for the research community. For instance, a shared artifact allows researchers to easily reuse it rather than needing to re-implement it, helps to validate the corresponding findings, and builds trust by contributing to open-science practices. Consequently, many researchers have started to investigate the practices, challenges, and benefits of artifact sharing within software engineering (cf. Section 6).

In parallel, venues like the *International Conference on the Foundations on Software Engineering* [Krishnamurthi, 2013] or *Empirical Software Engineering* [Méndez Fernández et al., 2019], publishers like the *Association for Computing Machinery*,[1] and funding institutions like the *European Union* [Schiltz, 2018] are pushing for more open science. For example, they have introduced reviews, awards, badges, or legal frameworks for sharing research artifacts. Also, ethical reviews have become more important and are offered or demanded by more and more universities and research institutes if the research conducted involves personal or otherwise critical data.

Despite such efforts and a general agreement on the pros of open science, artifact sharing also faces criticism, for example, because of the effort it takes to review artifacts, vague definitions of badges, or enforcing open-science practices on unfit types of artifacts (e.g., confidential data). So far, such research and practices focus on the accessibility and reusability of artifacts created by researchers themselves (e.g., software implemented by them, data measured during experiments). In contrast, other aspects of how artifacts are shared have often been neglected, for instance, what format an artifact is stored in, whether an artifact fulfills legal requirements (e.g., data protection, software licenses), or ethical concerns like data privacy [Baltes and Diehl, 2016, Gold and Krinke, 2020, 2022]. Such aspects are particularly relevant when researchers share artifacts that involve the data (e.g., software, personal information) of others.

A primary example for such data is version-control data extracted from software repositories, which is extensively used in empirical software engineering, for instance, in mining studies, case studies, or benchmarks. Software repositories (e.g., from Git, GitHub, BitBucket) exhibit various types of data on the evolution of a software system, for example, source code, developer names, mail addresses, natural-language comments, commits, pull requests, or documentation. The diversity of this data poses several challenges to researchers when sharing and reusing Software-Evolution Datasets (SEDs)—data extracted from a software repository's version-control and associated systems. For instance, researchers have to ask themselves: *How to best organize the data (e.g., relational database versus CSV files) in a way that enables others to reuse it (e.g., not requiring high-performance computer clusters)? How to ensure data privacy and anonymity (e.g., developer names, callouts in natural-language comments)? What is allowed to share under what license and legal requirements (e.g., data protection)?* To support researchers working with SEDs and improve current data-sharing practices, a more profound understanding of **how SEDs are and should be shared is required**.

Please note that we focus on SEDs in this article because they are particularly critical and subsume various other types of artifacts. SEDs themselves are important (1) because of the widespread use of the involved data in software-engineering research; (2) to proof the results reported in respective papers; (3) to allow other scientist to reuse that data without necessarily collecting the whole dataset again; and (4) to compare or even benchmark techniques working with the involved data. Consequently, it is important to ensure that SEDs are complete, documented, reusable, and adhere to legal regulations. For instance, a developer whose data is involved in a shared SED could ask for their data to be removed if the respective regulations are violated (e.g., regarding anonymization, data protection laws, compability of software licenses). In turn, this privacy-utility challenge would lose information and put additional efforts on the researchers. Understanding such problems and challenges is essential when sharing SEDs to avoid future problems. Still, despite the focus on SEDs,

[1] www.acm.org/publications/policies/artifact-review-and-badging-current

our findings and recommendations can also be transferred to datasets that involve the same types of data (e.g., source code, personal data), even if these do not represent a SED.

In this paper, we contribute to this understanding by reporting a meta analysis of 200 papers that have shared, modified, or used SEDs. For this purpose, we started collecting 41 relevant datasets and mining-challenge papers from recent iterations (2017–2021) of the International Conference on Mining Software Repositories (MSR). MSR is a prime venue for software-evolution research, since it focuses on mining evolution data from software repositories. It even has dedicated tracks for publishing and working on SEDs, which are also reused well beyond the MSR community itself (cf. Section 2.2). We forwards-snowballed [Wohlin, 2014] through the papers citing these SEDs to obtain a broader picture of how the SEDs have (not) been reused and what challenges researchers experienced while doing so. Then, we discussed our findings with an authority on research-data management and ethics reviews to elicit how SEDs can be shared in a responsible way. More specifically, we contribute the following in this paper:

- We provide an overview of 43 SEDs and the involved data's properties.
- We structure the challenges of creating, sharing, and reusing SEDs by analyzing 200 papers.
- We discuss the implications of our findings based on an expert discussion with an authority on research-data management and ethics reviews to provide recommendations for research on how to share SEDs in the future.
- We publish our dataset of 200 papers in a persistent open-access repository.[2]

Our results underpin the diversity of artifacts contributed via SEDs and the widespread use of such SEDs with their involved artifacts within the entire software-engineering community; even though we cover only a subset of all such datasets. We highlight and discuss various problems regarding the sharing of SEDs (e.g., data formats, topicality, data privacy) that are important for researchers to consider when sharing, but also when reviewing and using, SEDs. To address these problems, we provide recommendations for sharing SEDs, which we argue are particularly helpful for researchers without access to authorities on the matter that can support them. For instance, not all universities have yet implemented dedicated Ethics Review Boards or hired Data Stewards for software-engineering research to review the collection and sharing of SEDs. Overall, we hope that our contributions help the community improve their sharing of SEDs, and thus contribute to reliable open-science practices in software engineering.

The remainder of this article is structured as follows. First, we describe our methodology in Section 2. Then, we present the results of our literature review regarding temporary sharing practices in Section 3. In Section 4, we analyze the problems we identified from our literature review and present recommendations that are based on our discussion with the authority on research-data management. Afterwards, we discuss the threats to the validity of our study in Section 5. We then provide an overview of the related work and compare it to our own study in Section 6 before concluding this article in Section 7.

## 2 METHODOLOGY

Next, we introduce our research objectives and methodology, which we summarize in Figure 1. As we show, we started our meta-study with a literature review to elicit SEDs and papers that use these SEDs. From the resulting papers, we extracted data about the SEDs' properties and the problems reported. Then, we derived eight questions about sharing SEDs, which guided the semi-structured discussion of our findings with an authority on research-data management and ethics reviews. Based on this discussion, we clarified the outcomes of our meta study and specified recommendations for sharing SEDs. In the following, we describe the individual steps in detail.

[2] https://doi.org/10.5281/zenodo.11004148

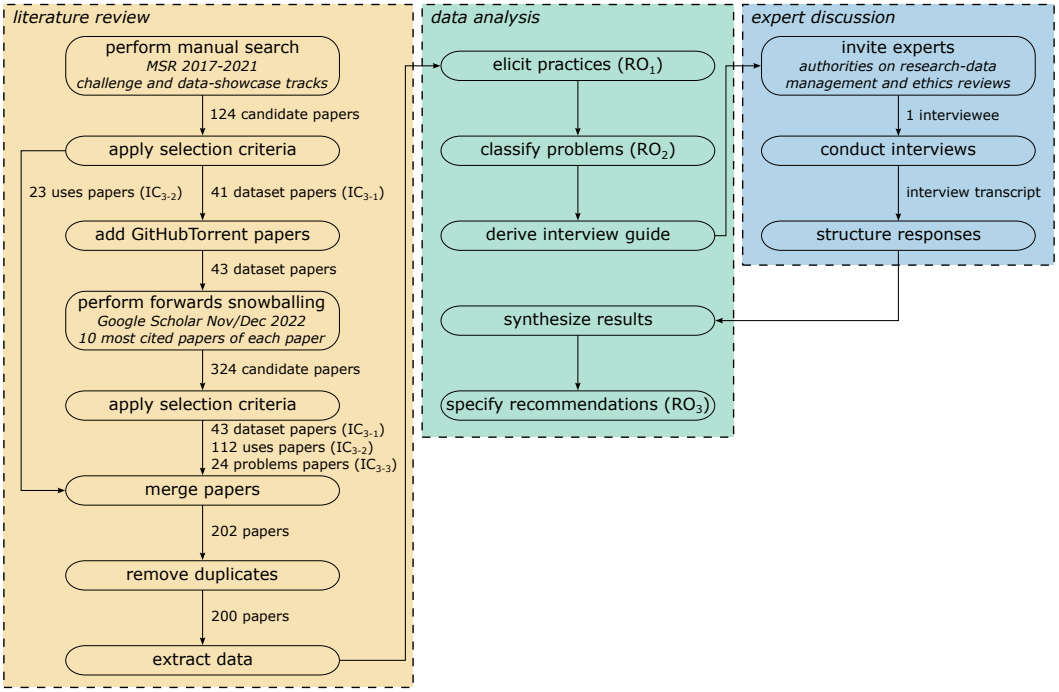


Fig. 1. Overview of our methodology.

## 2.1 Goal and Research Objectives

While artifact sharing has become widely established in research and is actively investigated (cf. Section 6), there are still many challenges associated to it. In particular, how to handle datasets that involve data about software evolution and the involved developers has rarely been studied, but poses many challenges related to copyright, anonymity, ethics, or feasible data storing. Our goal in this paper is to shed light into current practices as well as challenges of sharing such SEDs, and to provide recommendations for sharing SEDs in the future. For this purpose, we have defined the following three research objectives (ROs):

RO$_1$ *Elicit contemporary practices of sharing SEDs.*
First, we elicited how SEDs have recently been shared at a high-quality venue (i.e., that have undergone a peer review), analyzing their properties (e.g., data involved, size, where published) to gain an overview of recent sharing practices. The data we extracted is important to identify and understand practices that should be avoided to ensure legal (e.g., copyrighted data), ethical (e.g., data privacy), or analysis (e.g., performance issues due to size) problems. To tackle this objective, we conducted a manual search through five years of MSR (cf. Section 2.2), reviewing papers because these are also most likely the first references for researchers who want to use the respective SEDs.

RO$_2$ *Classify the problems of sharing and using SEDs.*
Second, we collected and structured the problems of sharing and using SEDs, which involve problems that we identified for the SEDs themselves and discussed with the expert as well as problems reported in the snowballed paper in which the SEDs were reused. Through this analysis, we contribute an overview of typical challenges that researchers and reviewers should keep in mind when working with SEDs. To tackle this objective, we analyzed all

200 papers using open-coding and open-card sorting to identify problems and classify those mentioned, using the expert discussion to confirm our analysis.

$RO_3$ *Specify recommendation for sharing SEDs.*
Finally, we specified recommendations for sharing SEDs by reflecting on the practices and problems we identified, incorporating the feedback of the expert discussion to propose solutions. So, we aim to contribute concrete recommendations that can help researchers share their SEDs in the future and can guide reviewers in assessing these. To tackle this objective, we synthesized solution strategies for the problems we identified in our meta analysis based on our experiences on research-data management, knowledge of relevant guidelines (e.g., FAIR principles [Wilkinson et al., 2016]), and the insights from the expert discussion.

These objectives define the scope of our meta analysis and expert discussion, which we report next.

### 2.2 Literature Review

To identify relevant papers for addressing our research objectives, we performed a two-step process based on guidelines and recommendations for searching papers for systematic literature reviews and mapping studies in software engineering [Brereton et al., 2007, Jalali and Wohlin, 2012, Kitchenham et al., 2015, Kitchenham and Charters, 2007, Krüger et al., 2020, Shakeel et al., 2018, Wohlin, 2014]. First, we conducted a manual search through the mining-challenge and dataset tracks of MSR, which are dedicated tracks at a high-quality conference for publishing datasets and solving challenges associated to these. Since MSR is focused on repository mining, and thus software changes (i.e., evolution), these tracks are an ideal opportunity to collect a sample of SEDs—which should be (and were) the primary focus of these tracks, too. Second, we performed a forwards snowballing on all relevant papers we identified to collect other papers for which the SEDs were (attempted to be) used and that report on the respective experiences. Please note that the snowballing was not limited to any venue and expands our sample to the broader software-engineering research community. Next, we describe our selection criteria before explaining these two steps in more detail.

**Selection Criteria.** To select papers, we defined three inclusion criteria (ICs):

$IC_1$ The paper is written in English.

$IC_2$ The paper (non-exclusive OR)
- $IC_{2\text{-}1}$ shares an SED (e.g., version-control data, issues, pull requests); or
- $IC_{2\text{-}2}$ uses an SED and does not just mention it, for instance, in the related work; or
- $IC_{2\text{-}3}$ describes or analyzes why an SED could not be reused.

Note that we also considered non-peer-reviewed papers during our snowballing (e.g., master theses, technical reports), since these are often longer and comprise more technical details. As a consequence, such papers helped us elicit problems of using SEDs that are often omitted in more space-restricted peer-reviewed papers. Regarding $IC_2$, we had to identify whether a publication shares an SED ($IC_{2\text{-}1}$); uses an SED, for instance, to evaluate a new technique ($IC_{2\text{-}2}$); or analyzes the problems of using an SED, for example, by attempting to reuse it or by comparing datasets ($IC_{2\text{-}3}$). Note that sharing in this context can also mean that an existing SED has been modified and re-shared, in which case a paper would fulfill both $IC_{2\text{-}1}$ and $IC_{2\text{-}2}$. We elicited papers fulfilling $IC_{2\text{-}3}$ to ensure that we do not only capture successful attempts of reusing an SED, but also challenges (with the SED) that prevented other researchers from that reuse.

**Manual Literature Search.** As our first step, we collected papers published at the MSR mining-challenge and dataset tracks; with MSR being one of the premier venues for research on software evolution and sharing corresponding datasets. We decided to search through MSR, because

(1) it has such dedicated tracks—in contrast to other flagship software-engineering venues covering research on software evolution (e.g., International Conference on Software Engineering; International Conference on Software Maintenance and Evolution);
(2) the primary topics of MSR relate to software-evolution data—which is why we expected a high level of expertise, quality, and best practices when sharing SEDs; and
(3) a manual search through dedicated MSR tracks promises a high ratio of relevant and high-quality papers, in contrast to an automated search that faces technical problems, is hard to replicate, and yields many irrelevant or lower quality papers.

Both tracks we considered involve papers that share SEDs, and the challenge track also involves papers reusing SEDs (i.e., solutions tackling the proposed challenge). Overall, we argue that MSR provided an ideal opportunity for eliciting papers that are relevant to address our research objectives, following a similar search idea as Gold and Krinke [2020, 2022] to keep our literature review focused and avoid the problems of automated searches (e.g., replicability, technical problems).

To identify relevant papers, we manually inspected the MSR entries in dblp.[3] Precisely, we analyzed papers from 2017–2021, which we considered a feasible time span, covering more recent practices (compared to older papers) while also providing time for the SEDs to be reused (compared to newer papers from 2022 when we conducted the search). We then identified all papers that are part of either the dataset or mining-challenge (proposal and solution) tracks. Unfortunately, the naming in dblp is inconsistent with these track names, for instance, in 2019 the dataset-track papers are listed under the publisher-provided categories "representations for mining" and "large-scale mining." To handle such inconsistencies, we verified that we identified the right and all papers of these tracks against the MSR conference website of each respective year.

Of the 124 papers in these five years and two tracks, we considered 64 relevant according to our ICs (i.e., 41 papers based on $IC_{2\text{-}1}$ and 23 according to $IC_{2\text{-}2}$). Please note that $IC_{2\text{-}3}$ was not relevant at this point, because there was no paper comparing different SEDs at the two tracks of MSR. We show an overview of the number of papers we included for each year and track in Table 1. As we can see, focusing on the two MSR tracks yielded the rather high ratio (51.61 %) of relevant papers we hoped for, drastically facilitating our search compared to an automated one. Moreover, it was easier to identify papers that did not fulfill our ICs. For instance, the challenge cases (and consequent solutions) of MSR 2018 on developer activities in IDEs [Proksch et al., 2018] and MSR 2019 on SOTorrent [Baltes et al., 2019] do not build on SEDs.

During a first analysis of these papers, we noted that many papers we identified to use an SED refer to or use parts of the GHTorrent dataset [Gousios, 2013, Gousios and Spinellis, 2012]. Since this dataset has been widely used in software-engineering research, we decided to add the two papers relating to it into our analysis, even though these papers are not in line with the time span we considered relevant. Please note that this is the only dataset that occurred several times, and we added only this one, due to its wide use in the community. So, we ended up with a total of 126 papers after our manual literature search, 43 of which share and 23 of which use an SED.

**Snowballing.** In November and December 2022, we performed a forwards snowballing using Google Scholar to extend our dataset. Particularly, we aimed to identify more papers that use one of the shared SEDs and that report on the challenges of doing so. To limit the effort of this process, we elicited for each of the 43 SED papers the ten most cited papers at that point in time. Since not all SEDs had been cited at least ten times when we performed the snowballing, we ended up with 324 snowballed papers (out of a theoretical maximum of 430). We then checked whether these 324 papers fulfilled our inclusion criteria, specifically whether they use ($IC_{2\text{-}2}$) and potentially re-share ($IC_{2\text{-}1}$) an SED or analyze problems of using one ($IC_{2\text{-}3}$). After this step, we ended up with 136

[3] https://dblp.org/db/conf/msr/index.html

Table 1. Number of papers we manually elicited from MSR.

| year | track | papers: all | included: SEDs ($IC_{2\text{-}1}$) | included: uses ($IC_{2\text{-}2}$) | discarded |
|---|---|---|---|---|---|
| 2017 | ds | 7 | 4 | 0 | 3 |
| | mc | 15 | 1 | 13 | 1 |
| 2018 | ds | 15 | 9 | 0 | 6 |
| | mc | 14 | 0 | 0 | 14 |
| 2019 | ds | 11 | 2 | 0 | 9 |
| | mc | 15 | 0 | 0 | 15 |
| 2020 | ds | 19 | 14 | 0 | 5 |
| | mc | 4 | 1 | 2 | 1 |
| 2021 | ds | 16 | 10 | 0 | 6 |
| | mc[1] | 8 | 0 | 8 | 0 |
| | **total** | 124 | 41 | 23 | 60 |

ds: data showcase – mc: mining challenge

[1] The challenge case of this year [Karampatsis and Sutton, 2020] was published as a data showcase at MSR 2020.

relevant papers, 112 that use a dataset ($IC_{2\text{-}2}$) and 24 that do not use a dataset but analyze or report problems that prevent such a use ($IC_{2\text{-}3}$). We identified and removed two duplicates that were part of MSR and the snowballing search. So, for our manual and snowballing search combined, we inspected 448 distinct papers (124 MSR, two GHTorrent, 324 snowballed, two duplicates removed), of which we included 200 as relevant to address our research objectives (44.64 %).

**Generalizability.** Regarding the generalizability of our sample, we would like to remark that only the initial set of papers we collected is from MSR only. We conducted forward snowballing to see whether other researchers reused these SEDs, which we did not limit to a specific venue. So, the initial SEDs themselves are only from MSR, but the snowballed papers (i.e., reshared SEDs, attempts to use the SEDs) span the whole software-engineering research community and are from venues like the International Conference on Software Engineering, Foundations on Software Engineering, Empirical Software Engineering, or Transactions on Software Engineering. Similarly, the authors of the SEDs and snowballed papers include active researchers across the community (e.g., Paul Raph, Michael Hilton, David Lo, Alexander Serebrenik, Andy Zaidnman, Georgios Gousios). We argue that this diversity in venues and authors shows that our sample and the results we derived from it are relevant and generalizable beyond MSR.

**Data Extraction.** We used text documents and a spreadsheet to collect all 200 papers and their bibliographic information (authors, publication year, title, venue). Then, we performed an open-coding process in which the second author (alone to ensure consistency) extracted relevant statements and data from each paper into individual text documents. For this purpose, he read through each paper in detail and focused on the parts that report the SED itself, how it has been used, and potential problems with sharing or using it. Moreover, he inspected the actual SED to verify the descriptions in the paper and to add details into the spreadsheet—if a link to the SED was available and still working.

Based on our research objectives, we defined the following data as relevant:

- A non-exclusive *categorization* for each paper, namely whether it shares an SED ($IC_{2\text{-}1}$), uses an SED ($IC_{2\text{-}2}$), or is relevant due to analyzing problems connected to SEDs ($IC_{2\text{-}3}$).
- An *SED's properties*, for which we also studied the actual SED if it was linked and available ($RO_1$): (1) name, (2) included data (e.g., repository metadata, (un)anonymized names), (3) sharing platform (e.g., Zenodo, GitHub), (4) storage format and supported queries (e.g., relational/graph database), (5) size, and (6) use cases for which it was suggested (e.g., automated program repair). This data helps understand temporary sharing practices by shedding light into what data (1) is shared how (2, 3, 4) and for what purposes (5). What data is shared

Table 2. Questions we designed to guide our semi-structured expert discussion.

| id | question | objective |
|---|---|---|
| $Q_1$ | For how many years have you been involved in reviewing/supporting data-sharing practices? | context |
| $Q_2$ | In how many software-engineering related ethics/data reviews have you been involved? | context |
| $Q_3$ | What are things to keep in mind or problems when sharing SEDs? | unbiased insights $RO_2$ & $RO_3$ |
| $Q_4$ | What types of data are particularly critical and why in SEDs? | unbiased insights $RO_2$ |
| $Q_5$ | How could we handle/share such critical types of data? | unbiased insights $RO_3$ |
| $Q_6$ | What (international) guidelines or references could researchers look into? | unbiased insights $RO_3$ |
| $Q_7$ | These are types of data we think are critical in the SEDs, what do you think about these? | feedback on data for $RO_2$ |
| $Q_8$ | How could we handle/share such critical types of data? | feedback on data for $RO_3$ |
| $Q_9$ | These are the problems the researchers mentioned, what do you think about these? | feedback on data for $RO_2$ & $RO_3$ |
| $Q_{10}$ | Would you like to add anything regarding our questions? | feedback on discussion |

is relevant to understand its criticality regarding ethics or legal regulations, for instance, whether developers can ask for their personal data or software to be removed later. How data is shared matters to understand its reusability, availability, as well as resource constraints. The use cases are important to understand why this data is needed in research, whether this is ethical, and what constraints exist (e.g., using high-performance clusters). Together, this data indicates the effort researchers are putting into SEDs and provides context for related challenges ($RO_2$) as well as the design of consequent recommendations ($RO_3$).

- The *problems* of sharing or using an SED as reported by the authors of a paper ($RO_2$).

After extracting this data, the first author of this paper performed a cross-validation, including an inspection of all 43 papers that share an SED and a random sample of the remaining papers. We found no errors in the extracted data, but added some refinements and details; particularly regarding the problems that were sometimes not described detailed enough in our spreadsheet. Furthermore, we revisited individual papers and SEDs during our analysis (cf. Section 2.4) to check for some context details, but did not identify errors.

### 2.3 Expert Discussion

To rely not only on our experiences and individual guidelines we were aware of to interpret our data, we aimed to discuss it with experts on research-data management and ethics reviews. For this purpose, we asked authorities we knew personally, one of which agreed to discuss our findings. Surely, more interviews would have been ideal to gain more insights. However, sharing datasets is bound to international regulations and guidelines that are typically very similar between countries. So, discussing the findings with one expert who knows about such regulations (particularly the strict ones of the European Union) should suffice to understand the most relevant problems and elicit helpful recommendations for sharing SEDs. Please note that the insights from our interview may be similar to those one would obtain when discussing SEDs with members of a dedicated Ethics Review Board. Nonetheless, our contributions (1) represent systematically collected insights to advance this line of research consistently and (2) provide insights for researchers who may not have access to such authorities.

To prepare the discussion, we defined the nine questions we display in Table 2 as a semi-structured guide and obtained ethics approval (Eindhoven University of Technology, ERB2023MCS32, August 25$^{th}$, 2023). First, we collected a minimum of background information to put the authority's expertise into context. In total, the authority has worked in their role for more then two years and has reviewed over 80 software-engineering-related data-sharing practices and ethics reviews. Please note that we refrained from collecting and reporting more data due to data-privacy concerns. After introducing the concept of SEDs, we asked four open questions ($Q_{3-6}$) related to problems, recommendations, and guidelines that are relevant in this context. To not bias the expert, we did

Table 3. Overview of the 41 SEDs we identified and the types of data they cover (five have two types). Please note that this overview serves as a lookup table and we only refer to each SED's name in other tables.

| type | SEDs | # |
|---|---|---|
| repository metadata | 20-MAD [Claes and Mäntylä, 2020], Andromeda [Opdebeeck et al., 2021], CROP [Paixão et al., 2018], Dockerfiles [Henkel et al., 2020], Duplicate Pull-Requests [Yu et al., 2018], Enterprise-Driven [Spinellis et al., 2020b], GE526 [Vagavolu et al., 2021], Git Archive [Markovtsev and Long, 2018], Git Repositories [Mockus et al., 2020], GHTorrent [Gousios, 2013, Gousios and Spinellis, 2012], Linux Kernel [Xu and Zhou, 2018], ManyTypes4Py [Mir et al., 2021], OCL expressions [Noten et al., 2017], Quantifying [Diamantopoulos et al., 2020], Repository Deduplication [Spinellis et al., 2020a], Sampling Projects [Dabic et al., 2021], Semantic Changes [Zhu et al., 2017], Shoulders of Giants [Zhang et al., 2020], Software Heritage Graph Dataset [Pietri et al., 2019, 2020a], Structured information [Schermann et al., 2018], UML models [Robles et al., 2017], Wonderless [Eskandani and Salvaneschi, 2021], World of code [Ma et al., 2019] | 23 |
| software quality | Andror2 [Wendland et al., 2021], Bugs.jar [Saha et al., 2018], C/C++ Code Vulnerabilities [Fan et al., 2020], Denchmark [Kim et al., 2021], ManySStuBs4J [Karampatsis and Sutton, 2020], QScored [Sharma and Kessentini, 2021], Software evolution [Yamashita et al., 2017], VulinOSS [Gkortzis et al., 2018] | 8 |
| human factors | CROP [Paixão et al., 2018], Enterprise-Driven [Spinellis et al., 2020b], Identity Resolution [Fry et al., 2020], Linux Kernel [Xu and Zhou, 2018], Mixed Graph-Relational Dataset [Ashraf et al., 2020], Software evolution [Yamashita et al., 2017] | 6 |
| testing and deployment | 50K-C [Martins et al., 2018], Duets [Durieux et al., 2021], JTeC [Corò et al., 2020], LogChunks [Brandt et al., 2020], TravisTorrent [Beller et al., 2017] | 5 |
| mobile apps | Android apps [Geiger et al., 2018], AndroidCompass [Nielebock et al., 2021], Andror2 [Wendland et al., 2021], AndroZooOpen [Liu et al., 2020] | 4 |

not reveal any of our data during these questions. Only then, we discussed this data and asked for clarifications that were not covered before ($Q_{7-9}$). Finally, we wrapped up the interview by discussing any things the expert wanted to add. The last author conducted the discussion with the expert using Microsoft Teams, which automatically transcribed the discussion.

### 2.4 Data Analysis

After extracting the data from the papers, we performed a collaborative open card-sorting-like process [Zimmermann, 2016] to identify common themes within that data. Specifically, all authors met in multiple discussion rounds to analyze the extracted data, agree on common themes, and connect these themes to our research objectives. For instance, we inspected the different types of data involved in the datastes and derived the types we summarize in Table 3 by agreeing on their boundaries in terms of relevant data. During this process, we also revisited the individual papers and datasets to check the correctness and level of detail of the data extraction. Since we did this step collaboratively and interactively, we cannot compute an inter-rater agreement. Then, one author coded the interview transcript, using the themes we already identified as codes and adding new ones if needed. Afterwards, we mapped both data sources via the codes, adding further explanations and recommendations of the expert to our previous insights.

## 3 $RO_1$: SHARING PRACTICES

In this section, we provide an overview of the paper data we collected to shed light into how SEDs have been shared in the past.

**Identified SEDs.** In Table 3, we display an overview of the 41 SEDs we collected from MSR. Note that two SEDs have two publications describing them, namely GHTorrent [Gousios, 2013,

Table 4. Data sources used to create the 41 datasets.

| **version-control data** | **30** | **existing datasets** | **13** | **other sources** | **9** |
|---|---|---|---|---|---|
| GitHub | 26 | GHTorrent | 11 | Jira | 2 |
| "other" Gits | 3 | World of Code | 2 | Google Big Query | 2 |
| GitLab | 3 | AndroZoo | 1 | security databases (CVE, NVD) | 2 |
| Subversion (SVN) | 2 | Pull-based development | 1 | Travis CI | 2 |
| Apache Git | 1 | | | DockerHub | 1 |
| BitBucket | 1 | | | Google Play | 1 |
| Gerrit | 1 | | | Linux Kernel Mailing List | 1 |
| Mercurial | 1 | | | package repositories (e.g., Debian, PyPi, NPM) | 1 |
| SourceForge | 1 | | | | |

Gousios and Spinellis, 2012] and the Software Heritage Graph Dataset [Pietri et al., 2019, 2020a]. Furthermore, we assigned five SEDs [Paixão et al., 2018, Spinellis et al., 2020b, Wendland et al., 2021, Xu and Zhou, 2018, Yamashita et al., 2017] to two types. For instance, Yamashita et al. [2017] combine evolving quality metrics (software quality) with developers' tasks and their dates (human factors). Consequently, the counts in the rightmost column do add up neither to the 41 SEDs nor the 43 publications. During our card sorting, we defined five high-level types of SEDs:

**Repository metadata** refers to SEDs that have collected and share typical version-control data (e.g., commits, pull requests). While such SEDs usually involve various additional types of data (e.g., software quality through bug reports or human factors through commit authors), we did not assign such SEDs to another type except if the authors explicitly enriched their SEDs for or with such other data (aiming to reduce redundancies).

**Software quality** refers to SEDs that involve data about the quality of evolving software systems. Such SEDs are enriched or narrowed down to focus on, for instance, bugs [Saha et al., 2018], code smells [Sharma and Kessentini, 2021], or code vulnerabilities [Fan et al., 2020].

**Human factors** refers to SEDs that focus on the stakeholders of a system. For example, Fry et al. [2020] share an SED on which they performed identity resolution via commit author identifiers.

**Testing and deployment** refers to SEDs that are concerned with the respective development activities. For instance, such SEDs include additional data on build logs [Brandt et al., 2020] or continuous integration [Beller et al., 2017].

**Mobile apps** refers to SEDs that collect software-evolution data related to mobile apps. For example, these SEDs represent collections of Android apps [Liu et al., 2020] or commits that touch Android compatibility checks [Nielebock et al., 2021].

Not surprisingly, we can see that SEDs are mostly concerned with typical repository metadata. Moreover, it is not surprising that various other important software-engineering research topics have led to dedicated SEDs (e.g., software quality for bug localization). Reflecting on the SEDs and their types, we argue that they seem representative for the broader research on software evolution.

**Data Sources.** As the first property of each SED, we investigated from what sources the researchers elicited the involved data—which we summarize in Table 4. Since we are studying SEDs, it is not surprising that all SEDs but one involve version-control data. The exception is the dataset of Linux mails by Xu and Zhou [2018], which refers to software evolution and version-control data within the mails. In detail, 26 of the 41 SEDs involve data extracted from GitHub, while 11 other SEDs have built on the GHTorrent data dump. Consequently, GitHub contributes to more than 90 % of the SEDs. A few other SEDs used different version-control systems or software-hosting platforms, such as BitBucket, Subversion, and SourceForge. Some SEDs (e.g., the Software Heritage Graph, World of Code) also combine data from various version-control systems. Only nine of the SEDs explicitly involve data from other sources, such as Jira, security databases, or different package repositories. Such other data sources do not only help enrich an SED with diverse projects, but support a certain

Table 5. Formats used to store data in the SEDs.

| format | (example) storage formats | # |
|---|---|---|
| spreadsheet | .csv, .xlsx | 19 |
| text files | .txt, .xml, .yaml, .json | 14 |
| relational database | MySQL, PostgreSQL | 11 |
| repository | GitHub repository | 4 |
| graph database | Neo4j, Google knowledge graph | 3 |
| document-oriented database | MongoDB | 2 |

use case and consequent type from Table 3 (e.g., using security databases for data on software quality). Still, SEDs that are systematically enriched with additional data are sparse in our sample.

**Sharing Platforms.** Next, we identified on what platform each SED has been shared (i.e., is hosted), with two dominating our sample: Zenodo with 18 and GitHub with 13 SEDs. Other sharing platforms are used by one or two SEDs only, specifically those platforms are OSF (2), BitBucket (1), and figshare (1). While it is a valuable trend that more SEDs are published in persistent repositories, we also found six instances in which SEDs are still shared on apparently personal or university websites. In October 2022, three of those six websites were not accessible anymore, namely those of AndroZooOpen [Liu et al., 2020], UML models [Robles et al., 2017], and TravisTorrent [Beller et al., 2017]. All other SEDs were still accessible via the link in the papers.

**Data Storing.** Besides the SEDs' location, we also investigated the formats researchers have used to store their SEDs. We display an overview of the primary format types and respective examples in Table 5. As we can see, we identified six primary formats, with spreadsheets (19) and text files (14) being the most common ones. While such formats are sometimes also a means to share data that can be imported into a database, some researchers provide their SEDs directly in an exported database scheme. Aligning to the previous two types, the relational scheme is most common (11), with graph (3) and document-oriented (2) databases being rarer. Four SEDs involve complete GitHub repositories. Note that some of the SEDs use multiple storage formats, which is why the sum of the rightmost column in Table 5 does not add up to the 41 SEDs. Interestingly, the SEDs span a variety of sizes of data, ranging from six to far more than 14 million software projects and including, for instance, plain version-control data (e.g., commits pull requests, issues), blobs, compiled binaries, additional documents, and metrics. As a consequence, the data storing of the SEDs is quite diverse.

**Use Cases.** Finally, we elicited nine use cases that have motivated researchers to create an SED (i.e., use cases mentioned in SED papers) or for which the SEDs were actually used (i.e., goals of the papers using an SED). We could summarize most use cases into one of two categories: analyzing software evolution (20 SEDs, 30 uses) or quality-issue detection and repair (26 SEDs, 20 uses). The former is concerned with any type of research that aims to improve our understanding of software evolution. The latter summarizes all research related to studying or resolving quality problems, including bug evolution, code smells, and automatic program repair. Other use cases occur less often, even though they are concerned with studying the developers of a system (12 SEDs, 13 uses; e.g., collaboration, interactions, gender), the processes employed (6 SEDs, 9 uses; e.g., regression testing, reviewing), or system analyses (6 SEDs, 5 uses; e.g., history slicing, feature location). Interestingly, human(-centered) aspects are mentioned and researched fewer times than plain software evolution or quality; but they also raise serious ethical and legal concerns (e.g., of identifying individuals). Furthermore, we identified a few more use cases, particularly in the papers using SEDs. These use cases include improving research (5 uses), constructing new datasets (5 uses), bot identification (3 uses), and others (3 SEDs). Logically, we can see that some use cases (e.g., bot identification) may be hard to anticipate due to technological advances, while it is surprising that some others are not really covered by dedicated SEDs (e.g., improving research). Lastly, we note

that we excluded 41 use cases that use an SED, because these were not concerned with software evolution and did not require an SED. Instead, these use cases built on the SEDs as a collection of subject systems, for instance, to evaluate techniques for scheduling in serverless software systems.

Insights $RO_1$: Dataset Sharing

- $I_1$ We identified 41 SEDs, which involve primarily repository metadata (23) and are sometimes enriched or narrowed down with respect to software quality (8), human factors (6), testing and deployment (5), or mobile apps (4).
- $I_2$ The primary source for SEDs is version-control data, either crawled directly (30) or reused from another dataset (13), that is sometimes (9) enriched with data from other sources.
- $I_3$ SEDs are often shared on established platforms like Zenodo (18) or GitHub (13), but six SEDs were shared on personal or institutional websites—of which three seem to be offline.
- $I_4$ The SEDs span a variety of data and sizes, resulting in different storage formats, primarily spreadsheets (19), text files (14), and relational databases (11).
- $I_5$ The most common use case are related to achieving insights on software evolution, quality and testing concerns, developers, or development processes; with alignment between use cases for which the SEDs have been suggested and actually used.

**Discussion.** Our insights hint at different concerns researchers should be aware of when it comes to sharing SEDs. As a conceptual challenge, most SEDs cover the same types of data ($I_1$), which is rarely enriched with data from other sources or of different types. On the one hand, this may limit the potential for novel research that requires additional data and it means that most SEDs may be highly similar in terms of what data they represent. So, working on new research directions will likely require the construction of dedicated SEDs, while picking most of the existing SEDs may induce threats to the external validity because mostly GitHub is covered (which, however, is the largest platform of diverse software projects). On the other hand, adding additional data to an SED can directly cause problems (e.g., privacy when adding developer characteristics) that may hamper the sharing of the SED. In this context, researchers need recommendations on how to deal with such problems to make their SEDs available, which is our goal in the remainder of this paper. Both views are further underpinned by most SEDs relying on the same data sources ($I_2$) and the fact that the use cases mostly cover well-established ones—while it is of course challenging to anticipate completely novel research directions ($I_5$). Similarly, most SEDs are being shared in (persistent) repositories ($I_3$), which is a good trend in terms of replicability. Nonetheless, this can also cause problems for researchers in terms of licensing and data privacy. Lastly, we feel that most SEDs are shared as spreadsheets or relational formats due to convenience ($I_4$). To scale analyses of large SEDs, other formats may be more appropriate (e.g., graph databases), but require further research to assess their pros and cons for relevant analyses, to benchmark their scalability, as well as to design appropriate analysis infrastructures. To summarize, our results for $RO_1$ reveal diverse practices for sharing SEDs, which hint at different challenges and potentially required solutions.

## 4 SHARING PROBLEMS AND RECOMMENDATIONS

In the following, we report and discuss the results of our meta-study and expert discussion that relate to the sharing problems we identified. Afterwards, we propose recommendations that can guide researchers when aiming to share an SED.

### 4.1 $RO_2$: Problems of Sharing

To answer $RO_2$, we elicited two sets of problems from our dataset. First, we analyzed the data shared within the 41 SEDs, aiming to understand to what extent it may be critical to share it. Second, we elicited the problems mentioned in the 158 snowballed papers that used (133) or analyzed (24) the

Table 6. Overview of users' data-privacy status that we elicited from the 41 SEDs' data.

| status | description | impacted SEDs | # |
|---|---|---|---|
| none | SEDs do not contain sensitive user data (note: we regard usernames in links to Git repositories as insensitive) | AndroidCompass, Andror2, Bugs.jar, C/C++ Code Vulnerabilities, Dockerfiles, Duets, Duplicate Pull-Requests, Enterprise-Driven, Git Archive, Git Repositories, JTeC, LogChunks, ManySStuBs4J, ManyTypes4Py, OCL Expressions, Repository Deduplication, Sampling Projects, Semantic Changes, Structured information, TravisTorrent, VulinOSS, Wonderless, World of Code | 23 |
| potential problem | SEDs contain sensitive user data and the paper does not discuss privacy concerns or their resolution | 50K-C, AndroZooOpen, Android Apps, CROP, Denchmark, GE526, GHTorrent, Linux Kernel, Mixed Graph-Relational Database, Quantifying | 10 |
| anonymization | the SED's paper explicitly states anonymization steps, some initial SEDs have been retrospectively replaced with anonymized versions | 20-MAD, Andromeda, Identity Resolution, Qscored, Shoulders of Giants, Software Evolution, Software Heritage Graph Dataset, UML models | 8 |

SEDs. The former provides insights into more conceptual problems of sharing datasets, while the latter focuses on reusability problems that should be considered while sharing. Also, we discussed general problems of sharing SEDs as well as our concrete results with the authority on data sharing. For consistency, we integrate the expert's comments as quotations in the following.

**Problems of Sharing SEDs.** When we conducted our analysis, we were particularly concerned with and found indicators for data-privacy problems (following GDPR)—which connects to the type of data involved and how this data is shared ($I_1$, $I_2$, $I_3$). In Table 6, we provide an overview for which SEDs we found such potential problems. We can see that a majority of 23 SEDs does not seem to involve sensitive user data and eight SEDs have been anonymized to account for data privacy. However, some of these eight SEDs were initially published in a different format, and later replaced with the anonymized version. Lastly, we found ten SEDs that involve sensitive user data, and the respective papers do not detail any anonymization steps or on what basis the SED has been published (e.g., consent). Consequently, these SEDs may be problematic considering ethical and data-privacy concerns, which has been discussed in the past in the context of spamming developers listed in GHTorrent without their consent [Baltes and Diehl, 2016]. The expert further noted that privacy-critical data does not only involve names, mail address, and other personal data of individuals, but can also touch, for instance, data related to their work because that may be used to evaluate their performance. Overall, the positive insight is that most SEDs seem to be concerned with data-privacy. However, some do not and we also stress that it would be helpful to report the handling of data-privacy concerns explicitly within the respective papers.

The expert raised two more problems that we noticed only occasionally in the papers. First,

> *"this is a bit tricky, but you also need consent for using people's data."*

Consent means that the individuals whose data is analyzed and shared agree to this processing and publishing. Since this clashes somewhat with the idea of mining software repositories, this situation poses an ethical and integrity problem because

> *"[...] people contributed to those software evolution or software datasets, not for the purpose for which you are using it. So, is that ethical privacy wise? For sure it's not. It's a Gray area [...]"*

So, the use of data and obtaining consent becomes somewhat of a gray area and subject to common sense. Ideally, the version-control system from which the data is collected would define precisely what data can be used for what purposes, but such information is often hard to find and interpret. For instance, GitHub specifies:

> *"You may use information from our Service for the following reasons, regardless of whether the information was scraped, collected through our API, or obtained otherwise:*

Table 7. Overview of the problems most commonly reported in papers (aiming to) use the 41 SEDs.

| problem | description | impacted SEDs | # |
|---|---|---|---|
| quantity and reliability | SEDs may not involve entries or a reliable ground-truth to serve as a feasible database | Android Apps, AndroidCompass, Bugs.jar, C/C++ Code Vulnerabilities, Duets, Enterprise-Driven, Git Archive, GHTorrent, ManySStuBs4J, Repository Deduplication, Sampling Projects, Semantic Changes, TravisTorrent, UML Models, Wonderless | 15 |
| missing data | SEDs may not involve the right data for an analysis | 20-MAD, Android Apps, Bugs.jar, Denchmark, Dockerfiles, Duplicate Pull-Requests, Git Archive, ManySStuBs4J, ManyTypes4Py, OCL Expressions, Semantic Changes, TravisTorrent, VulinOSS | 13 |
| redundant data | SEDs may involve example, toy, or stale-fork data | 20-MAD, CROP, Git Archive, LogChunks, ManySStuBs4J, ManyTypes4Py, OCL Expressions, Sampling Projects, Software Heritage Graph Dataset, TravisTorrent, Wonderless, World of Code | 12 |
| topicality | SEDs age and thereby may become outdated or even unavailable | Android Apps, AndroidCompass, Dockerfiles, Enterprise-Driven, Git Archive, GHTorrent, ManySStuBs4J, ManyTypes4Py, TravisTorrent, Repository Deduplication | 10 |
| faulty or invalid data | SEDs may involve manipulated or incorrectly extracted data | Android Apps, Duplicate Pull-Requests, Git Archive, GHTorrent, ManySStuBs4J, OCL Expressions, Software Heritage Graph Dataset, World of Code | 8 |
| accesibility | SEDs may exhibit complicated structures or require high-performance resources | C/C++ Code Vulnerabilities, Git Archive, ManySStuBs4J, Semantic Changes, Software Heritage Graph Dataset, TravisTorrent | 6 |
| others | SEDs may face other problems, such as missing tool support for analyses | Andromeda, AndroZooOpen, Qscored | 3 |
| | reused without problems (mentioned) | 50K-C, Andror2, Git Repositories, Identity Resolution, Linux Kernel, Mixed Graph-Relational Database, Quantifying, Shoulders of Giants, Software Evolution, Structured Information | 10 |
| | no reuse (attempted) | GE526, JTeC | 2 |

> - *Researchers may use public, non-personal information from the Service for research purposes, only if any publications resulting from that research are open access.*
> - *Archivists may use public information from the Service for archival purposes.*
>
> *Scraping refers to extracting information from our Service via an automated process, such as a bot or webcrawler. Scraping does not refer to the collection of information through our API. Please see Section H of our Terms of Service for our API Terms."*
>
> [https://docs.github.com/en/site-policy/acceptable-use-policies/github-acceptable-use-policies]

These rules allow researchers to mine public GitHub data, but they also enforce constraints like making any publication open-access. Consequently, we could assume that developers using GitHub should be aware that their data can be mined (which is also happening more and more often) and have given their consent by using GitHub. Still, as also noted by GitHub, there are privacy regulations that remain relevant, for instance, what data can be used for what purpose (e.g., forbidding spamming) and how to act on user requests (e.g., removal of data from a dataset). Second, the copyright of the software projects may pose problems [Ballhausen, 2019]. Specifically, if an SED shares actual source code, this may involve not only personal data (e.g., names in comments), but it would only be allowed if the SED and all source code in it are subject to compatible licenses. In summary, sharing a SED is subject to several ethical and legal problems that researchers have to assess in advance.

**Problems of Using SEDs.** For the problems other researchers had with using SEDs, we provide an overview of the six types of problems we identified in Table 8. In more detail, these problems are (with references to our related insights on the involved data and its sharing):

**Quantity and reliability:** We found 15 instances in which authors raised the problem that an SED is not large or representative enough to serve as a reliable ground-truth. Most often, these authors argued that an SED is too small or is not respected in a community. As a consequence, the researchers had to replicate the original queries, construct their own mining pipeline, or

extend the SED. As a concrete example, the C/C++ Code Vulnerability [Fan et al., 2020] SED has been considered too small to employ deep-learning models on it.

**Missing data ($I_5$):** In 13 instances, we identified the problem that an SED does not involve enough of the right data to properly represent the studied phenomenon. For instance, it has been argued that Denchmark [Kim et al., 2021] lacks version information for the included bug reports, which would be helpful in attribute for different analyses.

**Redundant data:** Quite on the opposite, we also found 12 instances in which authors argued that an SED involves too much redundant or irrelevant data. Particularly, the typical concern of toy and example projects (e.g., server blueprints) or stale forks (which most researchers consider not useful to analyze) have been raised. As a concrete example, some researchers aimed to use the World of Code [Ma et al., 2019] SED to conduct an expertise-identification analysis, but argued that developers occurring multiple times due to forks introduces bias.

**Topicality ($I_5$):** Since software evolution is continuous, any shared SED will become outdated if it is not regularly updated. Specifically, we found 10 instances in which researchers raised this concern and argued that they required more recent data (e.g., mining it themselves) or periodic updates of existing SEDs (e.g., as was the case with GHTorrent [Gousios, 2013, Gousios and Spinellis, 2012]). In the most extreme cases, SEDs may even become unavailable, not because they are outdated but simply because a sharing platform shuts down.

**Faulty or invalid data:** Faulty or invalid data threatens all scientific work, making the obtained results meaningless. We identified eight instances in which researchers raised the problem of incorrect metadata (e.g., timestamps). Two examples are GHTorrent, in which the authorship date of commits can be overwritten, and the Software Heritage Graph, in which duplicated commit identifiers are relabeled to make them unique. Even typical version-control systems and research tools [Hayashi et al., 2015] allow developers to manipulate recorded version histories. All such things threaten an SED and may make it useless for the intended analysis.

**Accessibility ($I_3$, $I_4$):** We found six instances in which researchers reported on accessibility problems, for instance, because the storage format and structure of an SED are complicated. Furthermore, software-evolution data is constantly growing, which means that researchers also need more and more computing power as well as hardware to analyze this data. For example, the Software Heritage Graph requires around 850 TiB storage, and even smaller parts of it that have been re-shared are often too large for typical computers.

**Others:** Lastly, we identified three individual problems that do not fit into the previous themes. For instance, we found that there is apparently no common tool support for semantic changes, and thus using the corresponding SEDs [Zhu et al., 2017] remains a problem.

Lastly, ten papers used a SED and did not report any problems, whereas two SEDs have not been used at the point in time when we performed the snowballing search.

When discussing these problems with the expert, they emphasized a few ones and provided additional explanations. In particular, missing, redundant, faulty, and wrong data can be the simple consequence of how repositories are mined:

> *"When you're crawling data, you don't know necessarily how accurate it is. How valuable is your data set then? And are you drawing conclusions on that?"*

As indicated by the expert, problems with collecting data can easily yield an SED that may be meaningless to other researchers because they cannot obtain reliable results:

> *"So that's the issue with using crawlers and you don't know the the quality and integrity of the data you're crawling [...] because they are public platforms and anybody can contribute and they can also contribute nonsense."*

Reflecting on the problems we identified with respect to using SEDs, we noticed that these are on a more technical level, particularly to those problems we discussed before for sharing.

$RO_2$: Problems of Sharing

- $I_6$ Sharing SEDs is subject to ethical and legal problems, particularly with respect to obtaining consent, data privacy, and copyright.
- $I_7$ Reusing shared SEDs faces particularly technical problems that must be considered before sharing the SED, namely: (1) ensuring quantity and reliability; (2) having the right data; (3) handling redundant data; (4) reasoning on topicality; (5) preventing faulty or invalid data; and (6) considering the accessibility of computing resources.

**Discussion.** Our insights show that researchers face two kinds of problems when creating and sharing SEDs. First, they have to resolve conceptual problems related to ethical and legal aspects ($I_6$). Second, they must consider technicalities of how they publish their SED so that it can be properly reused by other researchers ($I_7$). Both kinds of problems require detailed analyses by the researchers, and our insights to this point can guide them in the process. For instance, our previous insights into how SEDs are shared (cf. Section 3) in combination with the technical problems we found can help decide how to set up an SED. So, researchers should work towards resolving the problems we identified, for instance, by providing checklists for sharing SEDs and setting up tool infrastructures (e.g., for quality assessments). In the next section, we contribute a first step towards such goals by proposing an initial set of recommendations for sharing SEDs.

### 4.2 $RO_3$: Recommendations for Sharing SEDs

Based on our previous insights (referenced in the following), we aimed to define a set of recommendations that can help researchers when planning to share their SEDs. We refined and enriched these recommendations based on the expert discussion with the authority on research-data management and ethics reviews. Please note that this is only a first set of recommendations, which are, by nature, subject to changes, for instance, due to legal changes or changing ethical values. Moreover, we aimed to account more for rules defined by the European Union, since GDPR is a rather strict regulation that protects the personal data of all European citizens—so any researcher who collects their data must actually follow this regulation. Similar regulations exist within other countries and communities, and our recommendations may require corresponding adaptations. However, we argue that most recommendations are relevant and reasonable for software-engineering researchers anywhere on earth, as stated by the expert:

> *"Yeah, but if it's not GDPR, then it's of course another kind of privacy regulation that applies. And nearly all say something about collecting people's names and email addresses. So you always need to be worried about that."*

**Avoid Collecting (Personal) Data ($I_1$, $I_2$, $I_5$, $I_6$).** What data to collect and share is a key problem for creating and sharing SEDs [Gold and Krinke, 2020, 2022], since human factors are often of interest and scattered all across artifacts (e.g., commit messages, comments) but also particularly critical. There are two strategies for dealing with this problem: First, we can simply not collect the critical data, which is called *data minimization* (e.g., not collecting commit messages if commit dates suffice). Second, we can *anonymize* critical data (e.g., hashing names). Both strategies can have pros and cons, for instance, minimizing data reduces the risk of ethical concerns, while it may remove data that is needed to tackle a research question. Ideally, both strategies are combined and done already during the data collection:

> *"And yeah, [...] of course always anonymize, [and] don't collect the data you don't need."*

For instance, a crawler could skip or remove the data that shall be minimized or automatically hashes mails in natural-language text. Note that both strategies likely require some manual checks.

Despite the additional efforts and potential limitations regarding what research can be conducted, a fully anonymized SED has also various advantages. In particular, it ensures that the research is ethical and fulfills regulations like GDPR:

> *"[...] if it's truly anonymous, GDPR does not apply."*

Still, finding the balance between what data to minimize, anonymize, or handle differently can remain a challenging problem, even for experts:

> *"[...] about comments, the IT and I, we are also struggling with that. Is that copyrighted or is that a publication to which you should refer? You know, cite your references and that's it. Because it is on an open platform, anybody can access it. [...]"*

To check how to manage such data, the expert specifies to consult scientific-integrity guidelines, ethical guidelines, privacy regulations, agreed upon standards of the field, and common sense.

**Account for Different Software Licenses ($I_4$, $I_6$).** SEDs may combine source code from various, automatically crawled software projects. Consequently, copyright licenses of such projects become a critical concern [Ballhausen, 2019, Riehle and Harutyunyan, 2019]. Specifically, to share an SED involving code, it does not suffice that the license of each piece of code allows for redistribution, the licenses of all pieces of code within the SED must be compatible. One strategy to solve this problem is to:

> *"[...] filter out the licenses which you know are compatible."*

Since this may introduce sampling bias, multiple SEDs may be created to have different collections of source code with compatible licenses. Another strategy that is in line with open science and can also help tackle the previous problem is to have a fully replicable methodology and to provide only a dataset of weblinks to the actual source code or repository. This strategy does not only avoid sharing data, but it also accounts for the dynamic nature of software evolution:

> *"The dataset is dynamic, so it's only for that specific moment, but [this strategy] allows future researchers to test your theories at another point in time."*

**Ensure Quality and Reusability ($I_3$, $I_4$, $I_7$).** Building on our insights on how SEDs are shared (Section 3) and the problems we identified (Section 4.1), we want to stress the need for quality controlling an SED that shall be shared. Since we already discussed these problems in detail within the respective sections and other researchers have discussed the general quality of mined SEDs [Bird et al., 2009, Chatterjee et al., 2022, Kalliamvakou et al., 2014, 2016], we now sketch two directions for future research that can help mitigate the problems and that are persisting problems. First, we argue that we need to establish standardized analysis pipelines and frameworks. Those could incorporate means for ensuring data privacy, while also contributing to a consistent, reliable, and comparable analysis processes. Second, SEDs themselves lack standardization (e.g., regarding their formats). As a consequence, data cleansing becomes a key problem when reusing an SED, wasting time that is lost for the actual research. Interestingly, these problems should be well-known in research, but apparently researchers are experiencing them over and over again.

$RO_3$: Recommendations for Sharing SEDs

We have defined three recommendations for researchers to consider when sharing SEDs, including different proposals for implementing these.

## 5 THREATS TO VALIDITY

**Internal Validity.** Our data analysis of the existing SEDs is prone to interpretations. We employed open coding and card sorting to reduce the risks, but all categories remain our interpretation of the data. Similarly, we may have misunderstood statements in the papers or in additional documentation of the SEDs. Through our expert interview, we aimed to further confirm our interpretations and add complementary insights. While we only interviewed a single expert, which is also a threat to the internal validity, we argue that the nature of the discussion on regulations and the supportive data from our literature analysis mitigated this threat. Particularly, we argue that the insights we obtained at this point would not change strongly when interviewing multiple experts—which are also challenging to recruit due to the small population of experts on the matter. While we could not check for saturation, our qualitative interview provided in-depth insights into sharing SEDs and we followed a semi-structured guide to ensure a systematic and sound conduct. Lastly, we found further support for parts of our results and recommendations in the related work, which improves our confidence in our findings and their validity. To allow other researchers to verify and replicate our work, we share our dataset.[2]

**External Validity.** We have performed a manual search and snowballing starting from a single conference to collect our sample of SEDs. So, we may have missed highly relevant papers that share SEDs from other venues that could lead to different outcomes. However, MSR is a prime conference on software evolution, and is the only conference with dedicated tracks for publishing SEDs. For this reason, we argue that MSR represents a reliable overview of best practices on sharing SEDs, and that it is a reasonable strategy to start with a sample elicited from MSR tracks. Furthermore, while our SEDs come primarily from MSR, our snowballing on the use of the SEDs assured that experiences from a broad range of venues and researchers have been included in our analysis (cf. Section 2.2). Thus, our insights should be generalizable to other SEDs and also to other software-engineering datasets that involve the same types of data (e.g., software of other developers, profile pages or discussions with mail addresses, experimental data with free text calling out others). Lastly, the expert interview and related work providing supportive evidence further improve our confidence that our insights and recommendations are generizable beyond MSR and SEDs.

## 6 RELATED WORK

Even though sharing research artifacts is a long debated issue, researchers in software engineering have only recently started to investigate this topic systematically and in more detail. For instance, Timperley et al. [2021] have surveyed 153 software-engineering researchers to understand how artifacts are created, used, and reviewed. The authors aimed to understand current practices and derive recommendations for improving the quality of artifacts. Similarly, Hermann et al. [2020] have surveyed 257 researchers who participated in artifact-evaluation (i.e., reviewing) processes to understand their expectations for shared research artifacts and their evaluation. The authors identified specific quality expectations, but also inconsistencies in terminology as well as expectations that should be resolved. Heumüller et al. [2020] have investigated software tools shared at the International Conference on Software Engineering to analyze whether these were still available, identifying a positive trend over time. Other researchers have discussed the pros and cons of artifact sharing in the context of open science [Baldassarre et al., 2023, Hermann, 2022, Mendez et al., 2020], proposed or improved guidelines for sharing and reviewing artifacts [Damasceno and Strüber, 2021, Krishnamurthi, 2013, Krishnamurthi and Vitek, 2015, Méndez Fernández et al., 2019, Saucez et al., 2019, Winter et al., 2022, Zilberman and Moore, 2020], or further analyzed the incentives of artifact sharing and badges [Childers and Chrysanthis, 2017, Frachtenberg, 2022, Saucez and Iannone, 2018]—in software engineering and computer science in general. While

Table 8. Overview of the related work and its connections to our own study.

| topic | papers | contribution | relation to our work |
|---|---|---|---|
| artifact-sharing and open-science initiatives | [Baldassarre et al., 2023, Krishnamurthi, 2013, Krishnamurthi and Vitek, 2015, Mendez et al., 2020, Méndez Fernández et al., 2019] | create awareness for sharing software artifacts and open science in software engineering | motivation for our work, but no shared insights |
| studies on sharing software and other datasets in software engineering and related areas | [Childers and Chrysanthis, 2017, Damasceno and Strüber, 2021, Frachtenberg, 2022, Hermann, 2022, Hermann et al., 2020, Heumüller et al., 2020, Kotti et al., 2020, Kotti and Spinellis, 2019, Saucez and Iannone, 2018, Saucez et al., 2019, Timperley et al., 2021, Winter et al., 2022, Zilberman and Moore, 2020] | investigate artifact sharing regarding reviewing/creation standards, citations, incentives, or career benefits; identify lacks of community standards and quality requirements for sharing; do not focus on ethical concerns or SEDs | motivation for our work, but we (1) focus on SEDs instead of software or all datasets, (2) do not investigate metrics, impact, or reviewing guidelines, (3) aim to resolve the identified lacks for SEDs specifically |
| software licensing | [Ballhausen, 2019, Riehle and Harutyunyan, 2019] | review existing software licenses and licensing practices for creating new software | related to our insights and recommendation for checking software licenses when creating SEDs that comprise various software repositories |
| mining Git repositories | [Bird et al., 2009, Chatterjee et al., 2022, Kalliamvakou et al., 2014, 2016] | report first initiatives for creating SEDs by mining software repositories, emphasizing data-quality issues and deriving mining standards | related to our insights on quality problems of SEDs and the consequent recommendation |
| sharing and analyzing SEDs | [Baltes et al., 2019, Di Cosmo, 2018, Gousios, 2013, Gousios and Spinellis, 2012, Pietri et al., 2020b, Tiwari et al., 2017, Trautsch et al., 2020] | create and share well-known SEDs or tools for analyzing SEDs | motivation for our work, but we focus on problems around the matter that have not been discussed in these works |
| ethical dataset creation | [Baltes and Diehl, 2016, Gold and Krinke, 2020, 2022] | discuss ethical concerns of creating datasets, particularly when mining repositories | related to our insights and recommendation on legal and ethical issues, but we (1) build on a different research method (2) contribute a more in-depth study on SEDs as a whole, (3) and cover a broader perspective on the matter |

representing extensive research on artifact sharing, none of these works is concerned with the specific practices and challenges of sharing SEDs. We argue that SEDs face the inherent problem of involving personal or otherwise critical data that poses ethical concerns not investigated by any of the previous works. Moreover, we are not concerned with metrics or artifact-reviewing guidelines, which these works focus on. As a consequence, we are extending the scope of such works regarding how to share SEDs.

In the context of SEDs, additional concerns like copyright, licensing, and data privacy [Ballhausen, 2019, Riehle and Harutyunyan, 2019] are of utmost importance, since researchers are not sharing their own data, but data contributed by and including information of others (e.g., open-source developers). For instance, large-scale SEDs, such as GHTorrent or SoftwareHeritage [Baltes et al., 2019, Di Cosmo, 2018, Gousios, 2013, Gousios and Spinellis, 2012, Pietri et al., 2020b], or analysis frameworks for such SEDs [Tiwari et al., 2017, Trautsch et al., 2020] have been shared. In the past,

ethical concerns regarding the personal data in and certain practices of using such SEDs have been raised. The concern is that the impacted developers may loose their trust in and may stop supporting research, for instance, due to spam [Baltes and Diehl, 2016] or questionable studies that waste their time.[4] While several researchers have been concerned with the pros and cons of mining software-evolution data [Bird et al., 2009, Chatterjee et al., 2022, Kalliamvakou et al., 2014, 2016], they rarely discuss the ethical concerns or best practices of actually sharing the resulting SEDs. For this reason, it is not surprising that some released SEDs have also been criticized for ethical concerns. Unfortunately, the analyses of such concerns have been primarily anecdotal in the past. So, while there is some overlap in individual insights and such works have motivated us as well as support some of our results (e.g., regarding software licenses and the quality of SEDs), we provide a more systematic and in-depth overview of how to share SEDs that these works have not covered.

The most detailed research related to our study has been conducted by Gold and Krinke [2020, 2022], who discuss the ethics of mining software repositories in general. For this purpose, Gold and Krinke discussed the MSR mining challenges from 2006 until 2021 based on their personal experiences and conducted a community survey; using individual cases (including the creation of a SED) to showcase ethical concerns that exist in such cases. This research is closely related to ours (e.g., partial overlap in methodology and findings on personal data in SEDs), but we focused more on sharing SEDs as a whole—covering several additional challenges compared to Gold and Krinke (e.g., on reusing SEDs). Moreover, we employed a different methodology, building on published experiences and the expertise of an independent authority instead of our own opinions or a community survey. As a consequence, we provide a more in-depth overview of the practices and challenges of sharing SEDs, complementing and expanding upon these previous works. Finally, Kotti and Spinellis [2019] as well as Kotti et al. [2020] are concerned with the datasets published at MSR, but investigate their scientific impact (e.g., use, citations). Again, there is some overlap to our methodology, but we have a different focus, since we research the sharing practices surrounding SEDs. This represents a different goal and focuses on one type of dataset published at MSR.

In summary, we advance considerably upon the current state-of-the-art on sharing practices and challenges for SEDs, which have not been studied in this detail before. Particularly, we elicit what researchers are doing instead of surveying their opinions or what they claim to do by qualitatively analyzing papers instead of surveying researchers. By contributing an overview of the shared SEDs, synthesizing the problems of sharing and reusing these SEDs, as well as discussing these and potential solutions with an authority on the topic, we provide an in-depth understanding of sharing SEDs to guide researchers. Still, the similarities to the related work are supportive evidence for our findings, and partly highlight that these are persistent problems within the community.

## 7 CONCLUSION

In this paper, we reported a meta study of 200 papers that share or (attempt to) use an SED. We discussed our findings with an expert on the topic to obtain more in-depth insights and confirm our interpretation. Overall, we ($RO_1$) found that SEDs exhibit diverse properties, for instance, regarding data formats or sharing platforms (cf. Section 3); ($RO_2$) elicited ethical, legal, and technical problems (cf. Section 4.1); and ($RO_3$) derived three recommendations for researchers to reflect on (Section 4.2). We hope that our contributions help researchers prepare and share their SEDs in the future. Moreover, we discussed various directions for future work intended to improve the sharing of SEDs.

## DATA AVAILABILITY

Our data is available in a persistent open-access Zenodo repository.[2]

[4] https://www.theverge.com/2021/4/30/22410164/linux-kernel-university-of-minnesota-banned-open-source

## REFERENCES


Usman Ashraf, Christoph Mayr-Dorn, Alexander Egyed, and Sebastiano Panichella. 2020. A Mixed Graph-Relational Dataset of Socio-technical Interactions in Open Source Systems. In *MSR*. ACM.

Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. *Nature* 533, 7604 (2016).

Maria Teresa Baldassarre, Neil Ernst, Ben Hermann, Tim Menzies, and Rahul Yedida. 2023. (Re)Use of Research Results (Is Rampant). *Communications of the ACM* 66, 2 (2023).

Miriam Ballhausen. 2019. Free and Open Source Software Licenses Explained. *Computer* 52, 6 (2019).

Sebastian Baltes and Stephan Diehl. 2016. Worse Than Spam: Issues In Sampling Software Developers. In *ESEM*. ACM.

Sebastian Baltes, Christoph Treude, and Stephan Diehl. 2019. SOTorrent: Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets. In *MSR*. IEEE.

Moritz Beller, Georgios Gousios, and Andy Zaidman. 2017. TravisTorrent: synthesizing Travis CI and GitHub for full-stack research on continuous integration. In *MSR*. IEEE Computer Society.

Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, and Prem Devanbu. 2009. The Promises and Perils of Mining Git. In *MSR*. IEEE.

Carolin E. Brandt, Annibale Panichella, Andy Zaidman, and Moritz Beller. 2020. LogChunks: A Data Set for Build Log Analysis. In *MSR*. ACM.

O. Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *Journal of Systems and Software* 80, 4 (2007).

Preetha Chatterjee, Tushar Sharma, and Paul Ralph. 2022. Empirical Standards for Repository Mining. In *MSR*. ACM, 142––143. https://doi.org/10.1145/3524842.3528032

Bruce R. Childers and Panos K. Chrysanthis. 2017. Artifact Evaluation: Is It a Real Incentive?. In *2017 IEEE 13th International Conference on e-Science (e-Science)*.

Maëlick Claes and Mika V. Mäntylä. 2020. 20-MAD: 20 Years of Issues and Commits of Mozilla and Apache Development. In *MSR*. ACM.

Federico Corò, Roberto Verdecchia, Emilio Cruciani, Breno Miranda, and Antonia Bertolino. 2020. JTeC: A Large Collection of Java Test Classes for Test Code Analysis and Processing. In *MSR*. ACM.

Ozren Dabic, Emad Aghajani, and Gabriele Bavota. 2021. Sampling Projects in GitHub for MSR Studies. In *MSR*. IEEE, IEEE.

Carlos Diego Nascimento Damasceno and Daniel Strüber. 2021. Quality Guidelines for Research Artifacts in Model-Driven Engineering. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 285–296.

Roberto Di Cosmo. 2018. Software Heritage: Collecting, Preserving, and Sharing All Our Source Code. In *ASE*. ACM.

Themistoklis Diamantopoulos, Michail D. Papamichail, Thomas Karanikiotis, Kyriakos C. Chatzidimitriou, and Andreas L. Symeonidis. 2020. Employing Contribution and Quality Metrics for Quantifying the Software Development Process. In *MSR*. ACM.

Thomas Durieux, César Soto-Valero, and Benoit Baudry. 2021. Duets: A Dataset of Reproducible Pairs of Java Library-Clients. In *MSR*. IEEE, IEEE.

Nafise Eskandani and Guido Salvaneschi. 2021. The Wonderless Dataset for Serverless Computing. In *MSR*. IEEE.

Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. 2020. A C/C++ Code Vulnerability Dataset with Code Changes and CVE Summaries. In *MSR*. ACM.

Eitan Frachtenberg. 2022. Research artifacts and citations in computer systems papers. *PeerJ Computer Science* 8 (2022).

Tanner Fry, Tapajit Dey, Andrey Karnauch, and Audris Mockus. 2020. A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits. In *MSR*. ACM.

Franz-Xaver Geiger, Ivano Malavolta, Luca Pascarella, Fabio Palomba, Dario Di Nucci, and Alberto Bacchelli. 2018. A graph-based dataset of commit history of real-world Android apps. In *MSR*. ACM.

Antonios Gkortzis, Dimitris Mitropoulos, and Diomidis Spinellis. 2018. VulinOSS: a dataset of security vulnerabilities in open-source systems. In *MSR*. ACM.

Nicolas E. Gold and Jens Krinke. 2020. Ethical Mining: A Case Study on MSR Mining Challenges. In *MSR* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 265–276.

Nicolas E Gold and Jens Krinke. 2022. Ethics in the mining of software repositories. *Empirical Software Engineering* 27, 1 (2022).

Georgios Gousios. 2013. The GHTorent dataset and tool suite. In *MSR*, Thomas Zimmermann, Massimiliano Di Penta, and Sunghun Kim (Eds.). IEEE, IEEE Computer Society.

Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: Github's data from a firehose. In *MSR*. IEEE Computer Society.

Shinpei Hayashi, Daiki Hoshino, Jumpei Matsuda, Motoshi Saeki, Takayuki Omori, and Katsuhisa Maruyama. 2015. Historef: A tool for edit history refactoring. In *SANER*. IEEE.

Jordan Henkel, Christian Bird, Shuvendu K. Lahiri, and Thomas W. Reps. 2020. A Dataset of Dockerfiles. In *MSR*. ACM.

Ben Hermann. 2022. What Has Artifact Evaluation Ever Done for Us? *IEEE Security & Privacy* 20, 5 (2022).
Ben Hermann, Stefan Winter, and Janet Siegmund. 2020. Community Expectations for Research Artifacts and Evaluation Processes. In *ESEC/FSE*. ACM.
Robert Heumüller, Sebastian Nielebock, Jacob Krüger, and Frank Ortmeier. 2020. Publish or Perish, but do not Forget Your Software Artifacts. *Empirical Software Engineering* (2020).
Samireh Jalali and Claes Wohlin. 2012. Systematic Literature Studies: Database Searches vs. Backward Snowballing. In *ESEM*. ACM.
Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *MSR*. ACM.
Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An In-Depth Study of the Promises and Perils of Mining GitHub. *Empirical Software Engineering* 21, 5 (2016).
Rafael-Michael Karampatsis and Charles Sutton. 2020. How Often Do Single-Statement Bugs Occur?: The ManySStuBs4J Dataset. In *MSR*. ACM.
Misoo Kim, Youngkyoung Kim, and Eunseok Lee. 2021. Denchmark: A Bug Benchmark of Deep Learning-related Software. In *MSR*. IEEE, IEEE.
Barbara A. Kitchenham, David Budgen, and O. Pearl Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews*. CRC Press.
Barbara A. Kitchenham and Stuart Charters. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01. Keele University.
Zoe Kotti, Konstantinos Kravvaritis, Konstantina Dritsa, and Diomidis Spinellis. 2020. Standing on shoulders or feet? An extended study on the usage of the MSR data papers. *Empirical Software Engineering* 25, 5 (2020).
Zoe Kotti and Diomidis Spinellis. 2019. Standing on shoulders or feet? The usage of the MSR data papers. In *MSR*. IEEE.
Shriram Krishnamurthi. 2013. Artifact Evaluation for Software Conferences. *SIGSOFT Softw. Eng. Notes* 38, 3 (may 2013).
Shriram Krishnamurthi and Jan Vitek. 2015. The Real Software Crisis: Repeatability as a Core Value. *Commun. ACM* 58, 3 (feb 2015).
Jacob Krüger, Christian Lausberger, Ivonne von Nostitz-Wallwitz, Gunter Saake, and Thomas Leich. 2020. Search. Review. Repeat? An Empirical Study of Threats to Replicating SLR Searches. *Empirical Software Engineering* 25, 1 (2020).
Pei Liu, Li Li, Yanjie Zhao, Xiaoyu Sun, and John Grundy. 2020. AndroZooOpen: Collecting Large-scale Open Source Android Apps for the Research Community. In *MSR*. ACM.
Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of code: an infrastructure for mining the universe of open source VCS data. In *MSR*. IEEE / ACM.
Vadim Markovtsev and Waren Long. 2018. Public git archive: a big code dataset for all. In *MSR*. ACM.
Pedro Martins, Rohan Achar, and Cristina V. Lopes. 2018. 50K-C: a dataset of compilable, and compiled, Java projects. In *MSR*. ACM.
Daniel Mendez, Daniel Graziotin, Stefan Wagner, and Heidi Seibold. 2020. Open Science in Software Engineering. In *Contemporary Empirical Methods in Software Engineering*. Springer.
Daniel Méndez Fernández, Martin Monperrus, Robert Feldt, and Thomas Zimmermann. 2019. The open science initiative of the Empirical Software Engineering journal. *Empirical Software Engineering* 24 (2019).
Amir M. Mir, Evaldas Latoskinas, and Georgios Gousios. 2021. ManyTypes4Py: A Benchmark Python Dataset for Machine Learning-based Type Inference. In *MSR*. IEEE.
Audris Mockus, Diomidis Spinellis, Zoe Kotti, and Gabriel John Dusing. 2020. A Complete Set of Related Git Repositories Identified via Community Detection Approaches Based on Shared Commits. In *MSR*. ACM.
Sebastian Nielebock, Paul Blockhaus, Jacob Krüger, and Frank Ortmeier. 2021. AndroidCompass: A Dataset of Android Compatibility Checks in Code Repositories. In *MSR*. IEEE.
Jeroen Noten, Josh Mengerink, and Alexander Serebrenik. 2017. A data set of OCL expressions on GitHub. In *MSR*. IEEE Computer Society.
Ruben Opdebeeck, Ahmed Zerouali, and Coen De Roover. 2021. Andromeda: A Dataset of Ansible Galaxy Roles and Their Evolution. In *MSR*. IEEE.
Matheus Paixão, Jens Krinke, DongGyun Han, and Mark Harman. 2018. CROP: linking code reviews to source code changes. In *MSR*. ACM.
Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2019. The software heritage graph dataset: public software development under one roof. In *MSR*. IEEE / ACM.
Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2020a. The Software Heritage Graph Dataset: Large-scale Analysis of Public Software Development History. In *MSR*. ACM.
Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. 2020b. The Software Heritage Graph Dataset: Large-Scale Analysis of Public Software Development History. In *MSR*. ACM.

Sebastian Proksch, Sven Amann, and Sarah Nadi. 2018. Enriched Event Streams: A General Dataset for Empirical Studies on In-IDE Activities of Software Developers. In *MSR*.

Dirk Riehle and Nikolay Harutyunyan. 2019. Open-Source License Compliance in Software Supply Chains. In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*. Springer.

Gregorio Robles, Truong Ho-Quang, Regina Hebig, Michel R. V. Chaudron, and Miguel Angel Fernández. 2017. An extensive dataset of UML models in GitHub. In *MSR*. IEEE Computer Society.

Ripon K. Saha, Yingjun Lyu, Wing Lam, Hiroaki Yoshida, and Mukul R. Prasad. 2018. Bugs.jar: a large-scale, diverse dataset of real-world Java bugs. In *MSR*. ACM.

Damien Saucez and Luigi Iannone. 2018. Thoughts and Recommendations from the ACM SIGCOMM 2017 Reproducibility Workshop. *SIGCOMM Comput. Commun. Rev.* 48, 1 (apr 2018).

Damien Saucez, Luigi Iannone, and Olivier Bonaventure. 2019. Evaluating the Artifacts of SIGCOMM Papers. *SIGCOMM Comput. Commun. Rev.* 49, 2 (may 2019).

Gerald Schermann, Sali Zumberi, and Jürgen Cito. 2018. Structured information on state and evolution of dockerfiles on github. In *MSR*. ACM.

Marc Schiltz. 2018. Science Without Publication Paywalls: cOAlition S for the Realisation of Full and Immediate Open Access. *PLOS Medicine* 15, 9 (09 2018).

Yusra Shakeel, Jacob Krüger, Ivonne von Nostitz-Wallwitz, Christian Lausberger, Gabriel C. Durand, Gunter Saake, and Thomas Leich. 2018. (Automated) Literature Analysis - Threats and Experiences. In *SE4Science*. ACM.

Tushar Sharma and Marouane Kessentini. 2021. QScored: A Large Dataset of Code Smells and Quality Metrics. In *MSR*. IEEE.

Diomidis Spinellis, Zoe Kotti, Konstantinos Kravvaritis, Georgios Theodorou, and Panos Louridas. 2020b. A Dataset of Enterprise-Driven Open Source Software. In *MSR*. ACM.

Diomidis Spinellis, Zoe Kotti, and Audris Mockus. 2020a. A Dataset for GitHub Repository Deduplication. In *MSR*. ACM.

Christopher S. Timperley, Lauren Herckis, Claire Le Goues, and Michael Hilton. 2021. Understanding and Improving Artifact Sharing in Software Engineering Research. *Empirical Software Engineering* 26, 67 (2021).

Nitin M. Tiwari, Ganesha Upadhyaya, Hoan A. Nguyen, and Hridesh Rajan. 2017. Candoia: A Platform for Building and Sharing Mining Software Repositories Tools as Apps. In *MSR*. IEEE.

Alexander Trautsch, Fabian Trautsch, Steffen Herbold, Benjamin Ledel, and Jens Grabowski. 2020. The smartshark ecosystem for software repository mining. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*.

Dheeraj Vagavolu, Vartika Agrahari, Sridhar Chimalakonda, and Akhila Sri Manasa Venigalla. 2021. GE526: A Dataset of Open-Source Game Engines. In *MSR*. IEEE.

Tyler Wendland, Jingyang Sun, Junayed Mahmud, S. M. Hasan Mansur, Steven Huang, Kevin Moran, Julia Rubin, and Mattia Fazzini. 2021. Andror2: A Dataset of Manually-Reproduced Bug Reports for Android apps. In *MSR*. IEEE.

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, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR Guiding Principles for Sientific Data Management and Stewardship. *Scientific Data* 3, 1 (2016).

Stefan Winter, Christopher S. Timperley, Ben Hermann, Jürgen Cito, Jonathan Bell, Michael Hilton, and Dirk Beyer. 2022. A Retrospective Study of One Decade of Artifact Evaluations. In *ESEC/FSE* (Singapore, Singapore) *(ESEC/FSE 2022)*. ACM, New York, NY, USA.

Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *EASE*. ACM.

Yulin Xu and Minghui Zhou. 2018. A multi-level dataset of linux kernel patchwork. In *MSR*. ACM.

Aiko Yamashita, S. Amirhossein Abtahizadeh, Foutse Khomh, and Yann-Gaël Guéhéneuc. 2017. Software evolution and quality data from controlled, multiple, industrial case studies. In *MSR*. IEEE Computer Society.

Yue Yu, Zhixing Li, Gang Yin, Tao Wang, and Huaimin Wang. 2018. A dataset of duplicate pull-requests in github. In *MSR*. ACM.

Xunhui Zhang, Ayushi Rastogi, and Yue Yu. 2020. On the Shoulders of Giants: A New Dataset for Pull-based Development Research. In *MSR*. ACM.

Chenguang Zhu, Yi Li, Julia Rubin, and Marsha Chechik. 2017. A dataset for dynamic discovery of semantic changes in version controlled software histories. In *MSR*. IEEE Computer Society.

Noa Zilberman and Andrew W. Moore. 2020. Thoughts about Artifact Badging. *SIGCOMM Comput. Commun. Rev.* 50, 2 (may 2020).

Thomas Zimmermann. 2016. Card-Sorting: From Text to Themes. In *Perspectives on Data Science for Software Engineering*. Elsevier.