Empirical Software Engineering
https://doi.org/10.1007/s10664-020-09851-6

Publish or perish, but do not forget your software artifacts

Robert Heumüller[1] · Sebastian Nielebock[1] · Jacob Krüger[1,2] · Frank Ortmeier[1]

Published online: 08 October 2020

Abstract
Open-science initiatives have gained substantial momentum in computer science, and particularly in software-engineering research. A critical aspect of open-science is the public availability of artifacts (e.g., tools), which facilitates the replication, reproduction, extension, and verification of results. While we experienced that many artifacts are not publicly available, we are not aware of empirical evidence supporting this subjective claim. In this article, we report an empirical study on software artifact papers (SAPs) published at the International Conference on Software Engineering (ICSE), in which we investigated whether and how researchers have published their software artifacts, and whether this had scientific impact. Our dataset comprises 789 ICSE research track papers, including 604 SAPs (76.6 %), from the years 2007 to 2017. While showing a positive trend towards artifact availability, our results are still sobering. Even in 2017, only 58.5 % of the papers that stated to have developed a software artifact made that artifact publicly available. As we did find a small, but statistically significant, positive correlation between linking to artifacts in a paper and its scientific impact in terms of citations, we hope to motivate the research community to share more artifacts. With our insights, we aim to support the advancement of open science by discussing our results in the context of existing initiatives and guidelines. In particular, our findings advocate the need for clearly communicating artifacts and the use of non-commercial, persistent archives to provide replication packages.

Keywords Software · Open science · Open source · Artifacts · Publishing

Communicated by: Martin Monperrus

This paper has been awarded the Empirical Software Engineering (EMSE) open science badge.

The work of Jacob Krüger has been supported by the German Research Foundation (SA 465/49-3) and an IFI fellowship of the German Academic Exchange Service.

Robert Heumüller and Sebastian Nielebock contributed equally to the research reported in this article.

✉ Robert Heumüller
robert.heumueller@ovgu.de

Extended author information available on the last page of the article.

1 Introduction

Software-engineering research has always been driven by developing concepts and techniques to automate or facilitate the tasks of software developers (Wicks and Dewar 2007; Ossher et al. 2000). As a consequence, researchers have been building numerous *software artifacts*, ranging from analysis scripts for empirical studies, over prototypes to show the feasibility of a technique, to full-fledged tools that are used in practice. In the context of this article, software artifacts represent runnable (at least after compiling) programs, developed by researchers to obtain or analyze their results. This definition does not cover other types of artifacts (e.g., datasets), and thus is more specific in this regard than some existing guidelines for publishing artifacts (cf. Section 2).

Software artifacts are important for research, allowing other researchers to replicate results and to build on previous work, as well as for practical adoption, providing a means for practitioners to make use of research tools (von Nostitz-Wallwitz et al. 2018a, b; Diebold and Vetrò 2014; Garousi et al. 2016; Lo et al. 2015). This is additionally highlighted by numerous conferences adopting tool and demonstration tracks that focus on presenting such artifacts in a more interactive form than a scientific presentation. Moreover, in recent years, leading software-engineering venues, for instance, the *International Conference on Software Analysis, Evolution and Reengineering* (SANER)[1] or the *International Conference on Software Engineering* (ICSE),[2] explicitly welcomed contributions that replicated, reproduced, and critically discussed results derived with such artifacts (Monperrus 2014; Fu and Menzies 2017).

In parallel to pushing for open science in the context of open-access publications, several initiatives have started to promote publicly available software artifacts. For instance, the *Association for Computing Machinery* (ACM) introduced badges with defined criteria to motivate researchers to contribute their artifacts (Boisvert 2016).[3] Similarly, the *Journal on Empirical Software Engineering* (ESE) launched an open-science initiative (Méndez Fernández et al. 2019) for any artifact that is connected to its articles.

There are apparent benefits and motivations for making software artifacts available. However, during our daily work, we observed that it is sometimes complicated to obtain and reuse software artifacts from others. In the worst case, it is simply impossible to access artifacts, forcing us to re-implement them from scratch. This requires additional development time and raises the issue of how accurately we can reproduce that artifact. Unfortunately, even the best efforts of publishing artifacts may be in vain, considering that some platforms that are viewed as stable today may vanish or change their policies in the future. For instance, BitBucket decided to delete all of its Mercurial projects, due to their decreasing use in software development.[4] To avoid such removal of their software artifacts and instead keep them publicly available over a long period, researchers strive for platforms supporting open-science and long-term persistence, for example, as envisioned by the Software Heritage project (Di Cosmo 2018). Another issue with existing platforms is a missing, consistent procedure for publishing a software artifact, which hampers the ability to replicate or reuse that artifact. This problem is also stressed by numerous existing guidelines emphasizing related aspects for publishing artifacts, such as using a non-changeable (i.e., persistent) archive or adhering to "software-engineering practices" (cf. Section 2).

[1]https://saner2020.csd.uwo.ca/negativerestrack
[2]https://2019.icse-conferences.org/track/icse-2019-ROSE-Festival
[3]https://www.acm.org/publications/policies/artifact-review-badging
[4]https://bitbucket.org/blog/sunsetting-mercurial-support-in-bitbucket

All of the aforementioned issues indicate that the research community faces several open questions to scope and improve the current practices of publishing artifacts. In this article, we report an empirical analysis of research papers published at ICSE from 2007 to 2017. With our study, we answer the following research questions:

RQ$_1$ What is the prevalence of software artifact papers at ICSE?s
RQ$_2$ How have the artifacts from those papers been published?
RQ$_3$ What is the impact of publishing software artifacts?

To answer **RQ$_1$**, we analyzed the ratio of **software artifact papers** (SAPs) to all papers published at ICSE in the investigated time frame. We define SAPs as papers in which the authors report a software artifact that they developed specifically for the described research. While being only an indirect measure for the prevalence of SAPs at ICSE, answering this question provides insights into the importance and value of open-science, artifact publishing, and corresponding guidelines for the software-engineering community at ICSE. For **RQ$_2$**, we investigated four properties, which we derived from existing guidelines (cf. Section 2), for each SAP that we identified. The results are valuable to understand previous practices and current trends of publishing artifacts, indicating whether the software-engineering community does (in practice) accept the push for openness and advances in this regard. Finally, with **RQ$_3$**, we analyzed whether making artifacts publicly available has made a difference in terms of citations as a measure of scientific impact. This research question led us to our provocative title, but its main intention is to motivate researchers (similar to badges), highlighting that artifact availability has the potential to increase the quality and impact of research. From our observations, we derived four important lessons that we discuss in the context of existing guidelines.

Overall, we contribute the following with this article:

- We provide a comparison of existing open-science initiatives in terms of the properties they define for publishing software artifacts.
- We analyzed 789 papers that have been published at ICSE research tracks from 2007 until 2017. Based on this analysis, we address three questions, namely: (**RQ$_1$**) how many software artifacts have been reported in the papers; (**RQ$_2$**) whether and how these artifacts have been published; and (**RQ$_3$**) whether publishing software artifacts changed the impact (in terms of citations) of the corresponding papers.
- From our findings, we derive lessons learned that can help SAP authors avoid pitfalls, and which substantiate and complement existing artifact publishing guidelines (cf. Section 5.2).
- We provide a public replication package comprising our dataset and all analysis scripts on Zenodo.[5]

Our findings are important for research and practice alike. First, while most of the research community seems to already share the *feeling* that too few software artifacts are published, to the best of our knowledge, we are the first to provide *empirical evidence* from a large-scale study on this topic. Second, we provide insights on publishing artifacts that are currently not covered by existing initiatives on open-science, which could be improved based on our results. Third, we highlight how researchers can increase the impact of their research, motivating them to openly publish their software artifacts as far as possible. Fourth, with an increase of publicly available software artifacts, it will be easier to test

[5]https://doi.org/10.5281/zenodo.3935612

and build on research, facilitating the adoption of research in practice, and promoting trust between both areas.

Within this article, we compare eight existing guidelines for publishing software artifacts, from which we identified 14 properties that are considered important for artifact publishing (cf. Section 2). In Section 3, we describe our methodology for collecting and analyzing ICSE papers as well as their respective metadata. We explain how we classified papers based on their metadata and four of the properties we identified from guidelines. Using this classification, we investigated our research questions (cf. Section 4). In Section 5, we discuss possible reasons why software artifacts are not openly published based on the literature and our own experiences; and present our lessons learned to complement and refine existing guidelines. Then, we report possible threats to the validity of our study (cf. Section 6), describe related work (cf. Section 7), and conclude our main results (cf. Section 8).

2 Artifact Publishing Guidelines

Open-science initiatives, institutions, conferences, researches, and publishers have developed processes and guidelines for publishing software artifacts. For the purpose of this article, we classified six types of guidelines (based on eight concrete guidelines) that have been launched by major computer-science publishers, venues, and organizations, namely:

- The ACM badges[3] (Boisvert 2016)
- The ESE OpenScience initiative[6] (Méndez Fernández et al. 2019)
- The *Journal of Open Source Software* (JOSS) (Katz et al. 2018) and the *Journal of Open Research Software* (JORS)[7]
- The guideline by Wilson et al. (2017)
- The NASA Open Source Software Projects[8]
- The artifact-evaluation process of the *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS)[9] and the *International Conference on Computer-Aided Verification* (CAV)[10] as employed in 2019

We chose these guidelines because they represent recent initiatives in computer science and software engineering. In particular, these guidelines provide a publicly available description of their process and allow a broad insight into current efforts towards open science. Note that for simplicity, we have grouped the journals JOSS and JORS as well as the conferences TACAS and CAV, because they differ only in details with respect to the properties we analyzed. We emphasize that this list of guidelines is not complete. For example, several private organizations, such as Microsoft,[11] Facebook,[12] or Netflix,[13] publish their scientific tools as open-source artifacts. However, we were not able to find the respective guidelines, and, similar to BitBucket, private organizations may decide to remove or transfer these artifacts at any point in time for any reason.

[6]https://github.com/emsejournal/openscience/

[7]https://openresearchsoftware.metajnl.com/about/editorialpolicies/

[8]https://code.nasa.gov/

[9]https://conf.researchr.org/track/etaps-2019/tacas-2019-papers#Artifact-Evaluation

[10]http://i-cav.org/2019/artifacts/

[11]https://www.microsoft.com/en-us/research/tools/

[12]https://github.com/facebookresearch

[13]https://netflix.github.io/

Table 1 Overview of what properties the guidelines we consider in this article demand. We emphasize the properties that we were concerned with in our analysis in bold

Properties	**Guidelines**					
	ACM	**ESE**	**JOSS JORS**	**Wilson**	**NASA**	**TACAS CAV**
Runnable Software	●	●	●	●	●	●
Documentation	●	●	●	●	●	●
(Unique) Identifier	●	●	●	●	❍	❍
Accessible Archive	●	●	●	●	●	❍
Persistent Archive	❍	●	●	❍	❍	❍
Special Distribution	❍	❍	❍	❍	❍	●
License	❍	●	●	●	●	●
Experimental Data	●	●	❍	●	●	●
Small Test Data	❍	❍	●	●	❍	●
Open Issues List	❍	❍	❍	●	❍	❍
SE Practices	❍	❍	●	●	●	●
Law and Regulations	❍	❍	❍	❍	●	❍
List of Contributers	●	●	●	❍	●	●
Special Requirements	❍	❍	❍	❍	●	❍

Based on the guidelines, we derived four properties that we analyzed for each SAP. We display an overview of the main properties that software artifacts have to fulfill to adhere to each guideline in Table 1. This list of properties was compiled in two steps. First, a preliminary list was extracted by the second author from the guidelines' respective documents and web pages. Second, in a discussion based on this preliminary list, we identified the main properties by merging redundant properties and removing noise. Our goal was to ensure that the distilled properties were well-suited for depicting the requirements of the original guidelines. In the following, we discuss each guideline in detail, followed by a summary of all properties from Table 1, and an explanation on how we selected the four main properties for our analysis.

ACM Badges The ACM badges have been introduced as a means to award, and thus motivate, authors of accepted papers for making their software artifacts accessible and their results replicable. Several findings have shown that badges are an effective mechanism to increase the amount of open data in science, for example, in psychology (Kidwell et al. 2016) and medicine (Rowhani-Farid et al. 2017). First experiences from the *ACM Multimedia Systems Conference* (MSC), which applies badges since 2017, also support that observation for software artifacts (Thomee et al. 2018). Leading software-engineering venues, such as ICSE and the *Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (ESEC/FSE), have also introduced ACM badges.

Conferences can offer up to three categories of badges: "Artifacts Evaluated," "Artifacts Available," and "Results Validated." Papers awarded with the badge "Artifact Evaluated"—which further distinguishes between "Functional" and "Reusable"—provide complete and executable software, the experimental data, as well as appropriate documentation. In addition, the badge "Artifact Available" considers whether the software is actually accessible, meaning that it is uploaded to an archive and linked with a unique identifier, usually a Digital Object Identifier (DOI). The badge “Results Validated” considers whether the results of a paper have been “Replicated,” meaning that others produced the same results from the original artifact, or “Reproduced,” meaning that others produced the same results with an independent re-implementation. We remark that conferences that offer these badges can also adapt and strengthen the requirements. For example, conferences can enforce the publication of artifacts in a persistent (non-deletable) archive, such as Zenodo,[14] Figshare,[15] or Software Heritage (Di Cosmo 2018).

ESE OpenScience Initiative In 2019, ESE started an initiative to increase the number of open-source artifacts that are published based on the articles accepted at the journal (Méndez Fernández et al. 2019). In particular, ESE requires that authors publish the source code in a persistent archive with an open-source license. Furthermore, the authors have to adhere to the FAIR-principle,[16] meaning that artifacts should be **f**indable with a unique and persistent identifier (e.g., a DOI), **a**cccessible for machines and humans through interfaces, **i**nteroperable for machines to execute the artifact, and **r**eusable through a proper documentation and references to additional data. Similar to ACM, ESE offers badges (i.e, “Open Data” and “Open Material”) from the open-science framework.[17]

JOSS/JORS In 2016, the Journal of Open Source Software (JOSS) has been introduced to support the publication of software-artifact articles. For the review process, JOSS expects both, the article and the corresponding software, to be available via publicly accessible Git-based repositories. The reviewers then evaluate whether the described artifact is executable, documented, offers example data, and adheres to common software-engineering practices, for instance, the existence of automatic tests.[18] After acceptance, the artifact is stored in a persistent and publicly accessible archive, and is then referred to by its DOI.

The Journal of Open Research Software (JORS) published its first articles in 2013. It accepts so-called software meta-papers, which aim to complement research papers and to facilitate software reuse by describing artifacts in depth (e.g., their purpose, architecture). Particularly, JORS’ publicly available review criteria assume a runnable software to be persistently available in a suitable repository with some kind of unique identifier, such as a DOI or other URI. The journal further asks for an open license, small test data, and a minimal amount of software quality.

Guideline by Wilson et al. In their article, Wilson et al. (2017) describe practices for scientists conducting research with computing systems. They target not only the publication of artifacts, but describe also how to manage raw data, organize a software research project, and collaborate with other researchers. Regarding the publication of artifacts, the authors suggest that these as well as the experimental data should be given a unique identifier. Wilson et al. (2017) also emphasize the importance of structuring and testing the code, and providing useful documentation. In addition, they recommend using archives with version control and explicit numbers to retrieve particular versions. To provide confidence for other researchers, authors should provide a usage license as well as a public list of open issues, for example, in an issue tracking system.

[14] https://zenodo.org/
[15] https://figshare.com/
[16] https://www.force11.org/fairprinciples
[17] https://osf.io/tvyxz/wiki/1.%20View%20the%20Badges/
[18] https://joss.readthedocs.io/en/latest/submitting.html

NASA Open Source Software Projects Within their open-source initiative, researchers at NASA can publish their artifacts. For this purpose, they contact a software release authority (SRA) that checks the artifacts regarding specified constraints, and that supports the researchers in publishing their artifacts. The publishing procedure includes a compliance check with laws and regulations (e.g., proper licensing, export control) as well as further requirements (e.g., security concerns). Besides the source code, SRAs also check the existence and the quality of the documentation, the experimental data, whether all involved developers are mentioned, and whether the software was properly tested. Finally, the artifact is uploaded to a GitHub repository and included into NASA's software catalog.[19]

TACAS/CAV In 2019, TACAS and CAV included an artifact evaluation session. To prepare for this session, authors had to create a special replication package as a virtual machine (VM). Within this package, the authors had to include their code, a user manual, and all necessary dependencies. In particular, the artifact should be able to run solely within the VM and without a network connection. Since some calculations require a lot of computation power and time, authors should also provide smaller test data for fast replications. Moreover, TACAS demanded to specify a valid license.

Summary of Guideline Properties Based on the guidelines (cf. Table 1), we identified 14 different properties that are important when publishing software artifacts. All guidelines assume a *runnable software*, that is, the artifact can be executed by an external researcher or reviewer at least in a predefined environment in a described manner. Moreover, all guidelines expect some kind of *documentation*, varying from simple text files to full manuals, containing information on the installation, compilation, configuration, and usage of the artifact. A *unique identifier* describes some means of unambiguously identifying a software artifact. While, in theory, any unique "calling name" (e.g. "eclipse") for an artifact could be used, this uniqueness is hard to guarantee. Therefore, it is best to provide an absolute URI or even a DOI. Such a link allows researchers and reviewers to retrieve exactly that artifact with which the research results were obtained, without having to resort to internet searches. An *accessible archive* refers to a code repository that is publicly available through the world wide web. These may range from locally hosted version control systems up to systems hosted by private and public organizations. Orthogonal, a *persistent archive* is a code repository that ensures high degrees of longevity and immutability of artifacts. For example, Zenodo is hosted by the CERN Data Centre with the support of the EU and ensures accessibility as well as immutability based on DOIs. While all guidelines ask to publish an artifact as open source, some guidelines demand a special kind of *distribution* for an artifact, namely particular virtual machines. This is usually motivated by the purpose of reusing the artifact (e.g., for replication). A statement regarding the use of a specific *license* is usually required since developers and researchers need to know the conditions under which they are allowed to reuse the artifact. Moreover, it is good practice to add input data either as *small test data* or *experimental data*, such as benchmarks. Such data helps to replicate the results as well as to compare new artifacts against the same dataset. An *open issues list* describes a list of known problems (e.g., bugs or edge cases that are not handled) or potential extensions of the artifact. Contributors can use this list as starting point for improving or extending the software, to suggest new features, or to report bugs. So, an issue tracking system also fulfills this criterion. Some guidelines expect some degree of *software engineering practices* to

[19]https://software.nasa.gov/

be employed, ranging from code styles, over commenting, to automated test cases for validation. *Law and regulations* requirements are usually required by organizations and deal with aspects of licensing, ownership, and export control. A single guideline (i.e., NASA) requires a *list of contributors* comprising explicitly the developers of the software artifact, while the other guidelines expect a list of authors that can include both, contributors to the software artifact and the corresponding paper. This is useful to contact the respective researchers in cases of further questions. Finally, some *special requirements* may be relevant for an organization, such as security concerns. We can see that in particular the last three properties are concerned with legal issues and responsibilities, which are especially important for organizations—and thus not surprisingly part of the NASA guidelines.

Selection of Analyzed Properties During our analysis, we could not assess all identified properties, since some are hardly measurable (e.g., **appropriate documentation**), some would take too much time to evaluate for 789 papers (e.g., **runnable software**), and some were already extensively analyzed in previous works, for example, **datasets** (Poldrack and Poline 2015; Sicilia et al. 2017) and **licensing** (Schreiber and Haupt 2017; Almeida et al. 2017; Méndez Fernández et al. 2019). In particular, we did not check the quality of the artifacts, namely the existence of **runnable software** and the **documentation**. Since we are not, and cannot be, domain experts for all of these artifacts, we can hardly measure the quality or the level of replicability of all software artifacts. Usually, specialized researchers conduct such comparative studies against tailored benchmarks, for instance, as for code-smell detection (Fernandes et al. 2016) or code-clone detection (Bellon et al. 2007; Roy et al. 2009). Moreover, running artifacts requires knowledge of a variety of programming languages, build systems, and benchmarks as well as a correct parametrization of such artifacts. Several properties (i.e., **open issue list**, **SE practices**, **licensing**, **law and regulations**, **list of contributers**, **special requirements**) are also not related to the availability of the artifacts, which is why we did not investigate these.

We split the guidelines' **accessible archive** property into two properties for our analysis. So, we investigated: First, whether the authors **linked** to their artifact using some form of URI. Second, whether the artifact was actually **available** at the target of that URI (e.g., downloadable or usable as a web application). Ensuring this property is a precondition for any quality check, since an artifact that is unavailable cannot be assessed. We also analyzed the proportion of SAPs with **(unique) identifiers**, since these greatly facilitate identifying the exact artifact associated with an SAP. However, since most artifacts did not have a guaranteed unique identifier, we first counted *named* artifacts. Then, we approximated an upper boundary for the unique identifier property by counting the number of SAPs that provided a name or a URI. We further analyzed the types of websites used to see whether these changed over time and, in particular, to gain insights on the degree to which **persistent archives** were already in use. Finally, we analyzed the **distribution types**, because researches may have different motivations to reuse software artifacts, such as simple replications, comparisons to their own artifact, or extensions of the existing artifact. These use cases require different kinds of distributions, even though, in theory, source code should usually support all of them. In the next section, we explain the details of how we assessed these properties.

3 Methodology

In this section, we explain the methodology we employed for our analysis. We elaborate on the main steps of our *data acquisition*, *authorship-based community clustering*, and *classification*.

3.1 Data Acquisition

Our analysis is based on SAPs published at ICSE research tracks. Regarding our research questions, we decided to study ICSE publications in a representative period of eleven years from 2007 to 2017, deliberately excluding the most recent publications from 2018 and 2019 (at the time of the investigation, which started before ICSE 2019 took place), due to the likely bias regarding the citation analysis. To improve comparability and to eliminate other sources of bias, we limited our analyses to full papers in the main research track of ICSE. Using the *dblp computer science bibliography*,[20] the *Scopus API*,[21] and the *Crossref API*,[22] we automatically retrieved the meta data, abstracts, and citation counts of 792 papers. Later on, we removed three papers that were falsely attributed to the research track. We found two of these while comparing the dataset with original conference websites and final reports of the program committees. The third paper, we found during the validation step, which we describe in Section 3.3.

To investigate the four properties (i.e., accessible, persistent archive, unique identifier, and distribution type), we extracted information from all papers, which we used to classify them according to the criteria we define in Section 3.3. Particularly, we checked whether the papers claimed to have a software artifact (i.e., being an SAP). In such a case, we further collected the web-links (i.e., URIs) linking to the software artifacts (if provided). Using these web-links, we were able to check whether the linked archive was *accessible* and whether the artifact was *available* through that link. Note that we only accepted web-links linking to the artifact rather than to a general description of the paper or the artifact without access to it. Moreover, we also identified the *type of the web page* to assess the *persistence* of the archive. During our analysis, we noticed that most authors did not provide a *unique identifier* for their artifacts, but rather an *artifact name*. As unique identifiers have only recently been adopted for artifacts and are still not strictly enforced, we considered an artifact name as a weak surrogate for an identifier. For example, many research papers use the name of an artifact when referring to a particular paper or artifact, and therefore identify the respective artifact. Still, since, for example, simple scripts are usually not named, we also considered whether artifacts were linked via a URI, which we also considered as a unique identifier. We also documented in what form the artifact was *distributed*, if the artifact was available through the identified web-link.

For our envisioned citation impact analysis, we needed to establish comparability between different papers by normalizing the *Crossref* citation counts. Otherwise, the results would be heavily biased by the ages of the papers in the compared sets (older papers are more likely to have higher citation counts than more recent ones). To address this issue, we adapted the *average relative citation* (arc) metric (Hutchins et al. 2016; Piwowar et al. 2018). The *arc* score normalizes the citation count of an ICSE paper by the average citation count of all ICSE papers from that year. Let P_x be the set of papers published in year x of ICSE; then, for any paper $p \in P_x$, the *arc* score is defined as:

$$arc(p) = \frac{citationCount(p)}{\frac{\sum_{p' \in P_x} citationCount(p')}{|P_x|}}$$

[20] https://dblp.uni-trier.de/

[21] https://dev.elsevier.com/

[22] https://www.crossref.org/services/metadata-delivery/rest-api/

Consequently, an *arc* score greater than one indicates that a paper's citations are above average for that year, while values equal to or below one indicate average or below-average citations.

3.2 Authorship-Based Community Clustering

Before the actual analysis, we performed an authorship-based clustering of papers for two reasons: First, the clustering allowed us to analyze the publication behavior within sub-communities of ICSE, as opposed to ICSE in general. Second, we hypothesized that we could increase the efficiency of the manual classification by assigning clusters of related papers to the same analyst, facilitating the detection of similarities in paper structures, related software artifacts, and means of publishing artifacts—generally benefiting our manual analysis. While we did not evaluate to what degree the clustering actually sped up our analysis, the impressions of the analysts strongly supported this claim.

The choice of the similarity metric is crucial for any clustering since it has the greatest impact on what the detected clusters actually represent. As authorship-based similarity metric, we selected the Jaccard-similarity: Let A and B be the sets of authors of two papers, then the similarity is defined as:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Independent of the concrete clustering algorithm, this metric attempts to find clusters that share large proportions of authors between their papers.

Next, we evaluated two standard graph clustering algorithms, *Spectral Clustering* (von Luxburg 2007) and *Girvan Newman Clustering* (Girvan and Newman 2002), on our dataset, relying on the *Scipy, Scikit-Learn*, and *NetworkX* libraries for the implementations (Virtanen et al. 2020; Pedregosa et al. 2011; Hagberg et al. 2008). Using the *eigengap heuristic* (von Luxburg 2007), we estimated the number of clusters to be 209, which we then used to initialize both clustering algorithms. Using the *adjusted mutual information score* (Vinh et al. 2009), we compared the two resulting clusterings and determined a high correlation (*0.86*) between both algorithms. A qualitative inspection of the largest clusters supported this result. Due to the discovered similarity of both clusterings, in the remainder of the article, we only refer to the results of the Girvan Newman algorithm.

3.3 Classification

Next, we describe our process for manually classifying the 789 publications in our dataset. First, we split the dataset into three batches of approximately the same number of papers, one for each analyst (i.e., the first three authors of this paper). Instead of assigning papers to each analyst by random or by years, we assigned entire clusters to each analyst in a round-robin scheme. Then, we performed two rounds of classification to analyze the individual publications regarding the aforementioned properties. Finally, we validated the results of the classification, during which the three analysts crosschecked another batch of randomly selected papers for each year and each other analyst.

In the first round of classification, we analyzed only the papers from 2007 and 2017, and used the gathered insights to refine our method and agree on a final classification scheme. For example, one insight was that a distinction between *no artifact*, *preliminary artifact*, and *consolidated artifact* for the property *artifact maturity* was too vague—mainly due to researchers' inconsistent wording even within a single paper. So, we changed the value for this property to the binary *artifact claimed*. During the second round, we classified our entire dataset using the refined scheme, and the ACM Digital Library to retrieve the full-text papers.

Due to the total number of papers in this study, reading each paper completely would have been infeasible. Instead, we performed a heuristic search consisting of three steps. When a definitive answer regarding the properties could be derived from the first and second step, the third step was not executed. In the first step, the analysts carefully read the abstract. Second, they skimmed other likely sections, particularly the contributions, the evaluation, and the conclusion. Third, they performed a targeted search for keywords indicative of the *artifact claimed* and *link available* properties (we list these keywords in the property definitions at the end of this section). Afterwards, the analysts closely examined the context in which the keywords appeared to rule out false-positives, for example, when the use of the keyword "implementation" was not related to a software artifact belonging to the paper.

For validation, we conducted a crosscheck for which each analyst reviewed two additional, randomly selected papers per other analyst and year. Consequently, we validated a total of 132 classified papers (44 per analyst; 16.6% of all papers). If the two analysts disagreed, they discussed the issue and corrected the classification if necessary. In Table 2, we provide an overview of the classification errors we found during this validation step. Moreover, we found one paper to be falsely attributed to the research track, and we thus removed it from the following evaluation. In summary, the sample of papers that we validated suggests an error rate of less than 10 % for the classification.

In the following, we elaborate on the four properties derived from the guidelines, as well as two additional properties that do not originate from the existing guidelines (cf. Section 2). The first two, *paper type* and *artifact claimed*, represent properties of the SAP, rather than the artifact itself.

Paper Type Besides assessing whether a paper is an SAP (and thus implies an artifact), we also classified each paper into one of four categories:

- *Conceptual / Guideline* – Papers that introduce, for example, new concepts, guidelines, and best-practices, and reason about their effectiveness based on experiences or logic.
- *Empirical Study* – Papers that use empirical methods to improve the understanding of the status quo, rather than the effectiveness of some new technique. The latter, we considered to be technical contributions (for example, if a paper evaluates a new algorithm by applying it to existing benchmarks and comparing it to other state-of-the-art techniques).

Table 2 Classification errors we found during the validation

Property	**# Errors**	
Tool name	1	
	# False Negatives	# False Positives
Artifact claimed	4	1
Artifact linked	6	1
$\sum$	**13**	

- *Experience Report* – Papers that report on experiences in a subjective, non-empirical way. For example, these can be reports about the practical application of an existing method, process, or tool.
- *Technical Contribution* – Papers that focus on new methods or algorithms. If such a paper uses an empirical study to evaluate its contribution, we still labeled it as a technical contribution (cf. *Empirical Study*).

Artifact Claimed (Keywords for Searching: *tool, prototype, implementation*) For this category, we analyzed whether a paper claimed to have an associated *software artifact* or *not*. In this study, we considered a *software artifact* to be a runnable software program (at least after compiling) that researchers developed specifically to obtain or analyze their results. Thus, we did not consider other research artifacts, such as pure datasets, interviews, or guidelines as artifacts. As aforementioned, we aimed to further distinguish the implementation maturity, but most papers are vague on this distinction or even contradict either themselves or previous papers on the same software artifacts. For instance, the same paper may refer to an artifact as prototype and as industry-ready tool.

Artifact Name If a software artifact was named by the authors of a paper, we extracted that name as a proxy for the identifier property. Essentially, there are two categories for this property: We could either identify a *name* or *not*.

Artifact Availability (Keywords for Searching: *available, download, http, ftp, ://*) Within this category, we were concerned with the accessibility property, analyzing whether an SAP contains a link that references a web resource in the form of a URI for the software artifact. Note that this also encompasses DOIs, since they are a specific type of URIs. If present, we documented the links and categorized each SAP as follows:

- *Non-linked artifact:* The paper does not contain a link to its artifact.
- *Linked, but non-available artifact:* The paper contains a link to its artifact, but we could not download or use it from there.
- *Available artifact:* The paper contains a link and we could still download or use the artifact from the referenced archive.

Note that in contrast to artifact evaluation guidelines, such as of ICSE 2020,[23] we did not expect the artifacts to be uploaded to a certain persistent archive, but rather to be accessible via a web resource as determined in the following.

Type of Website In this category, we analyzed the persistence property. To this end, we documented the type of website on which a software artifact was available (or not), evaluating to what extent this had an impact on accessibility. We defined three categories:

- *Personal:* The website is a personal one (e.g., of an author or exclusively for making the software artifact available) that was not hosted by an institution or an organization.
  Example: https://www.jenn-doe.com/project
- *Academic / Institutional:* The website is hosted by a company, university, or a project (e.g., Eclipse Marketplace) and encompasses artifacts developed by this institution or within this project.
  Example: https://www.atlantis-university.edu/~jenndoe/project

[23] https://2020.icse-conferences.org/track/icse-2020-Artifact-Evaluation#Call-for-Submissions

– *Open-Source Repositories:* The website is an open-source repository hosted by an organization, such as GitHub or BitBucket, which provide a service to make source code publicly available for developers or organizations.
  Example: https://code-repo.com/jenndoe/project

Initially, we planned to have a fourth category identifying persistent archives, such as Zenodo. Such archives prohibit to change the artifact after publication and link it with a unique identifier. However, during our analysis, we found only a single artifact linked to such a persistent archive. Therefore, we integrated this paper into the class of *Open-Source Repositories*, since such persistent repositories provide a similar service for hosting artifacts.

Distribution Type Finally, we investigated the distribution property, for which we defined four ranked categories of how researchers distributed their artifacts (if the artifact was available):

1. *Source code:* The artifact was available as source code, allowing others to modify and build it.
2. *Binary:* The artifact was available as compiled binaries or as web application, but not as source code.
3. *Container:* The artifact was a container, such as a virtual machine or Docker container, from which the software artifact could be used.
4. *On demand:* The artifact was not available for download, but it was stated that interested researchers should contact an author.

If an artifact was provided in multiple distribution types, we classified it as the highest type.

Overall, we defined six properties to classify each paper. In Section 4, we describe the results of our analysis of the classified dataset, based on which we provide insights into our research questions.

4 Evaluation

The evaluation took place between May 20th 2019 and August 9th 2019. From the initial 792 papers, we excluded three that were falsely associated with the research track (one technical briefing, one short paper, and one keynote). So, our final dataset comprised 789 papers.

The following section is structured according to our three research questions. First, we analyze the proportion of SAPs (Section 4.1). Second, we investigate how authors have published their software artifacts and how this behavior evolved (Section 4.2). Finally, we evaluate the impact of publishing software artifacts by utilizing our citation analysis (Section 4.3).

4.1 Prevalence of Software Artifact Papers (RQ_1)

For the 11 years of ICSE we analyzed, 2007 had the lowest number of papers (49), and 2016 had the highest number (101). During the manual assessment of the papers, the analysts first discriminated the SAPs from the remaining papers (cf. Section 3). In Fig. 1, we show the proportion of SAPs and non-SAPs per year. Overall, a majority of 604 papers (76.6 %) are SAPs according to our definition, leaving a remainder of 185 papers without a software artifact. This also underpins the importance of software artifacts in the research area of

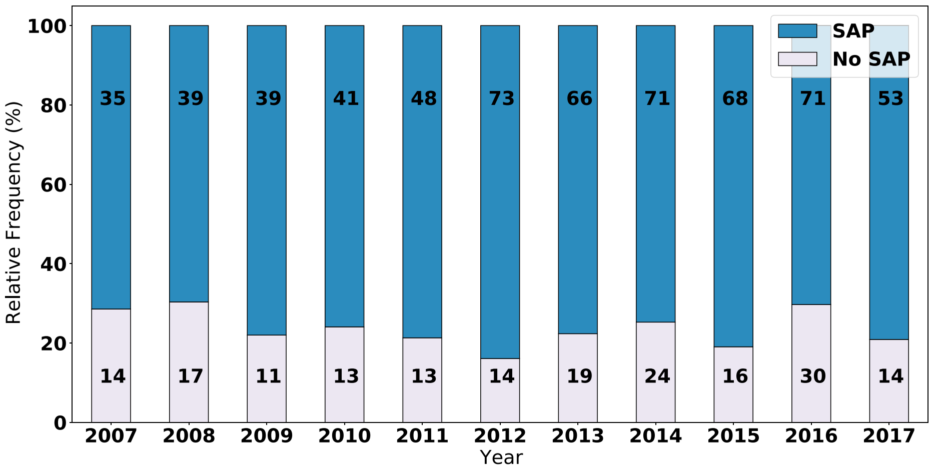

Fig. 1 Relative and absolute frequencies of SAPs per year

software engineering. Considering the proportion of SAPs, we can see that it remained fairly constant over the years, with SAPs always representing the majority—ranging from 69.6 % (in 2008) up to 83.9 % (in 2012). The lowest absolute number of SAPs (35) has been published in 2007. We conclude that for every year of ICSE we analyzed, the majority of authors claimed or implied that they created a software artifact for the purpose of their research.

Furthermore, we considered how SAPs are distributed between different paper types.

In Fig. 2, we can see that technical contributions are the most common paper type in our dataset (72.4 % of all papers). Also, they have by far the highest proportion of SAPs (550 / 571, 96.3 %) and the majority of SAPs originate from technical papers (91.1 %). Papers of the other types are far less commonly SAPs (ranging from 16.7 % for conceptional and guidelines to 25.9 % for empirical studies). We conclude that technical

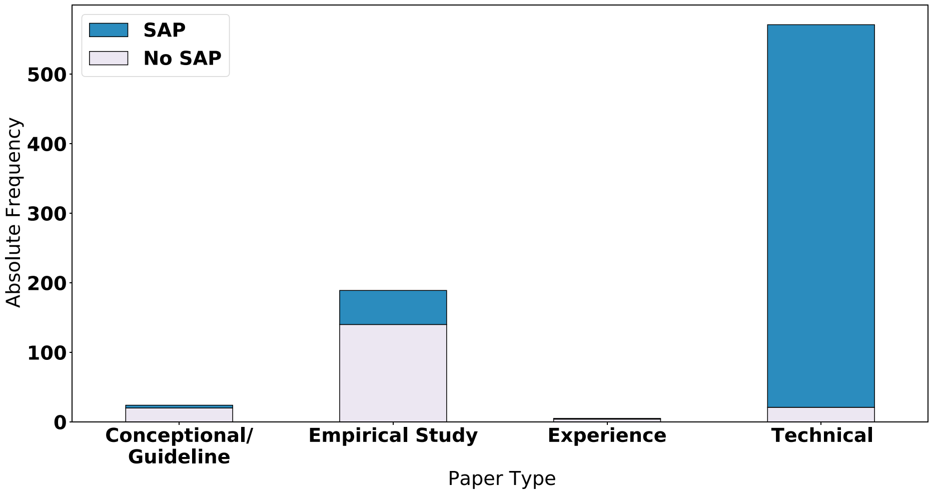

Fig. 2 Absolute frequency of SAPs per paper type

papers are the most frequent paper type for SAPs, with two other types of papers (i.e., conceptual/guidelines and experiences) being almost negligible.

Using our authorship-based clustering, we detected research communities within ICSE. We found that 165 of 209 automatically detected communities (78.9 %) published at least one SAP. We conclude that the majority of researchers have either actively taken part in writing SAPs or have likely come in contact with SAPs developed by their peers.

Insights for **RQ$_1$**

- Software artifacts are an essential part of ICSE papers
- Most software artifacts are contributed by technical papers
- The majority of sub-communities within ICSE published SAPs

4.2 How Software Artifacts are Published (RQ$_2$)

Since the majority of ICSE papers and communities apparently relies on creating software artifacts to produce their results, we next analyzed *if* and *how* these artifacts have been published.

First, we analyzed whether the authors of an SAP provide a link to their artifact in their paper. For each SAP, the analysts searched for the presence of a URI that points to an online resource for the respective artifact. In total, in 289 of 604 SAPs (47.8 %), the authors provided such a link. However, only 163 (27 % of all SAPs) of the artifacts were actually available through these links. We conclude that only slightly more than a quarter of ICSE SAPs have artifacts that are (still) readily available through their links, facilitating validation and allowing others to build on their results.

While the overall number of available artifacts is sobering, we determined a positive trend in recent years. In Table 3 and Fig. 3, we depict the absolute and relative frequencies of linked and available artifacts for SAPs per year. We can see that the proportion of linked artifacts increased from 8.6 % in 2007 to 71.7 % in 2017. The proportion of available artifacts increased from 5.7 % to 58.5 % during the same period. We approximated the

Table 3 Frequency of linked and available software artifacts described in SAPs per year

Year	#Software Artifact Papers	#Linked Artifacts	#Available Artifacts
2007	35	3	2
2008	39	13	5
2009	39	19	5
2010	41	15	7
2011	48	25	10
2012	73	33	20
2013	66	29	14
2014	71	38	21
2015	68	37	23
2016	71	39	25
2017	53	38	31
Σ	**604**	**289**	**163**

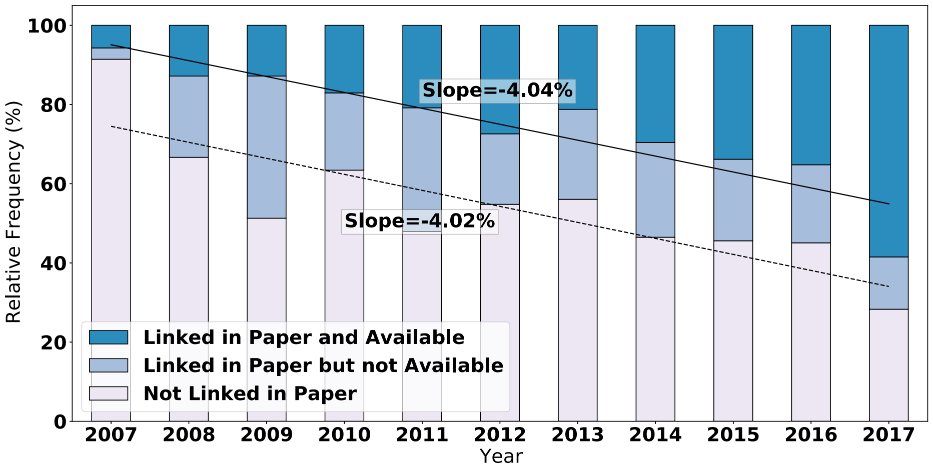

Fig. 3 Relative frequency of SAPs with linked/available software artifacts and linear regressions of the proportion of linked (dashed line) and available artifacts (solid line)

trends with the two linear regressions we display in Fig. 3. Both lines are almost parallel with slopes of approximately −4.02 % for linked artifacts (dashed line) and −4.04 % for available artifacts (solid line).

In our analysis, we noticed many broken links (i.e., the particular website was offline). However, the nearly identical slopes in Fig. 3 surprised us. Usually, we would have assumed some constant rate of link-decay similar to the "comatoseness" term introduced by Koehler (2002), which should lead to different slopes. We emphasize that we cannot directly map the results of the link decay study to the percentage of available software artifacts, since we did not check whether the link was accessible, but rather whether we could download (or use in case of web-apps) the software artifact from there. Further research is needed to clarify whether this is an indication of an unknown factor compensating the link decay or just a coincidence resulting from noisy data. Some linked websites simply described their research results without actually offering artifacts for download. Curiously, we also found one website that the authors had prepared for housing their artifacts, but which was apparently forgotten after the conference—which was more than five years ago. Nonetheless, we conclude that the awareness for publishing artifacts is increasing within the ICSE community.

We also considered the communities that we identified through our clustering (cf. Section 3.2). 123 of the 165 communities that published at least one SAP (74.5 %), had at least one SAP with a non-linked artifact and 141 (85.5 %) had at least one SAP with a non-available artifact. When considering only those communities that had at least two SAPs, the numbers change to 80 papers out of 91 (87.9 %) for linked artifacts and to 86 out of 91 (94.5 %) for available artifacts. We conclude that the problem of low artifact availability does unfortunately prevail throughout most research communities.

Note that this community analysis heavily relies on the qualitative insights we obtained while reviewing the papers, and during which we recognized that the authorship-based community clustering achieved a reasonable result in terms of identifying communities (i.e., collaborating authors). We deliberately do not name clusters, since our intention is not to point at any sub-community or even particular researchers, but rather to show that non-linked and not-available software artifacts appear in many sub-communities in software engineering—asking for the whole community to act and improve current practices. Moreover, we emphasize that this analysis should be replicated and extended in future work, using additional information apart from author names (e.g., session information from conferences) or more specialized techniques, such as topic modeling, to find more fine-grained communities.

For the 289 artifacts for which we identified a link, we further classified the type of website on which these artifacts were published. In general, the majority of artifacts are linked on academic or institutional websites (196), followed by open-source repositories (75), and personal websites (17). One paper provides its artifact's source code directly as an appendix to the paper. However, as we can see in Fig. 4, the proportion of open-source repositories has increased from 6.7 % in 2010 up to 65.8 % in 2017, while the proportion of academic and institutional websites has decreased. Note that we found only one artifact that is linked to a persistent storage archive, Zenodo, which we also classified as an open-source repository. We conclude that open-source repositories gain more and more attention towards publishing artifacts.

Regarding the distribution type, we considered those 163 artifacts that were available. Investigating the distribution type is important, as source code enables other authors to verify or adapt the implementation, while binary or container distributions may facilitate the replication of results. In general, the majority of artifacts that are still available have been published as source code (115), followed by binaries (38). Six artifacts seem to be available on demand, while only four are provided in a container. Over the years (cf. Fig. 5), the proportion of artifacts published as source code has always been highest, ranging from 50 to 100 %. So, we conclude that ICSE authors contribute their software artifacts mostly as source code.

Next, we analyzed to what extent authors named their artifacts, which makes it easier to refer to others' contributions and facilitates searching for the software artifacts online. In our analysis, we found that 413 of 604 (68.3 %) SAPs named their artifact using some "calling name". Note that except for 2007 (only 45.7 %), the majority of artifacts published

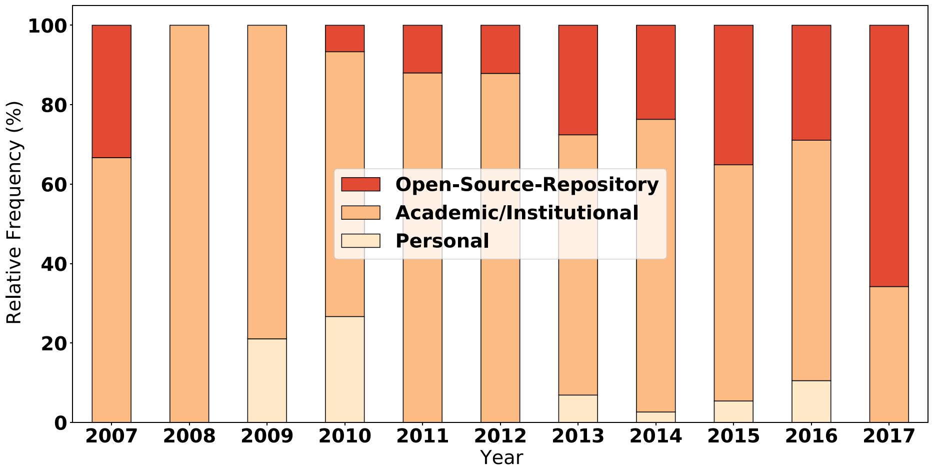

Fig. 4 Relative frequency of website types linked from SAPs

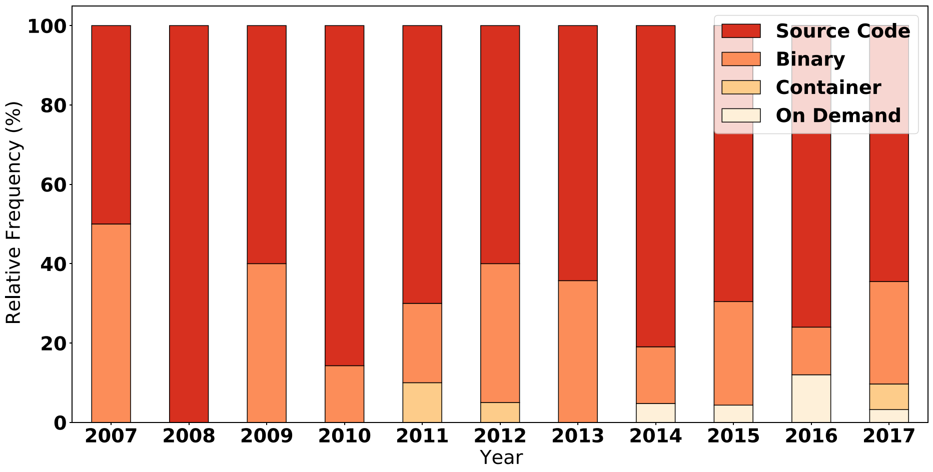

Fig. 5 Relative frequency of distribution types of available software artifacts

each year has a name. We also wanted to provide an upper bound for the state of implementation concerning the *unique identifier* property demanded by common guidelines. For this purpose, we considered "calling names" as well as provided URIs as surrogates for true unique identifiers. Using this definition, we found that 77.8 % of SAPs have an identifier that can be used to reference or search for the artifact. We emphasize that our definition of software artifacts has a significant influence on this statistic and it should be kept in mind when interpreting these numbers. While it is common to name fully-fledged tools or prototypes for new techniques, researchers rarely name their analysis scripts, which we also count as artifacts. Consequently, we conclude that most researchers at ICSE seem to intend to make their software artifacts uniquely identifiable.

Insights for **RQ$_2$**

- 52% of software artifacts in SAPs are not linked
- 73% of software artifacts were not available (either no link, or link did not lead to artifact)
- There is a positive trend of linked and available artifacts over the years
- Open-source repositories have gained track for publishing artifacts
- Most software artifacts are contributed as source code
- Most ICSE authors aim to make their artifacts uniquely identifiable

4.3 Impact of Publishing Artifacts (RQ$_3$)

We investigated whether different artifact-publishing behaviors could be associated with higher or lower scientific impact of the respective SAPs in terms of citations. For this purpose, we analyzed the average relative citation scores (*arc*) we introduced in Section 3. Using *arc* as a normalized citation count, we compared between groups of papers including different publication years. In particular, we were interested in possible differences between the following groups of papers, for which we depict distributions in the respective figures:

– Those with linked artifacts and those without (Fig. 6)

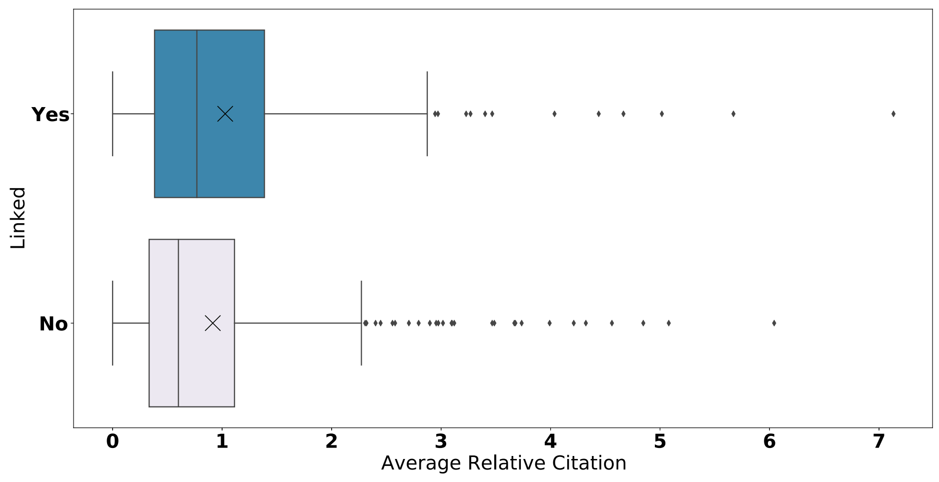

Fig. 6 Distribution of *arc* for linked artifacts (— → `Median`, X → `Mean`)

– Those with available artifacts and those without (Fig. 7)
– Those with named artifacts and those without (Fig. 8)

We considered these groups interesting, since linked and available artifacts facilitate reuse, which could increase the number of citations of those papers. Furthermore, naming an artifact simplifies finding software artifacts online, or requesting them from a corresponding author.

Based on the three box plots we depict, we cannot visually determine significant differences between the groups. However, the mean and median values of the groups (cf. Table 4) differ. The respective *arc* values for SAPs with linked, available, and named artifacts are consistently higher than their counterparts. We conducted a Kruskal-Wallis test (one-way

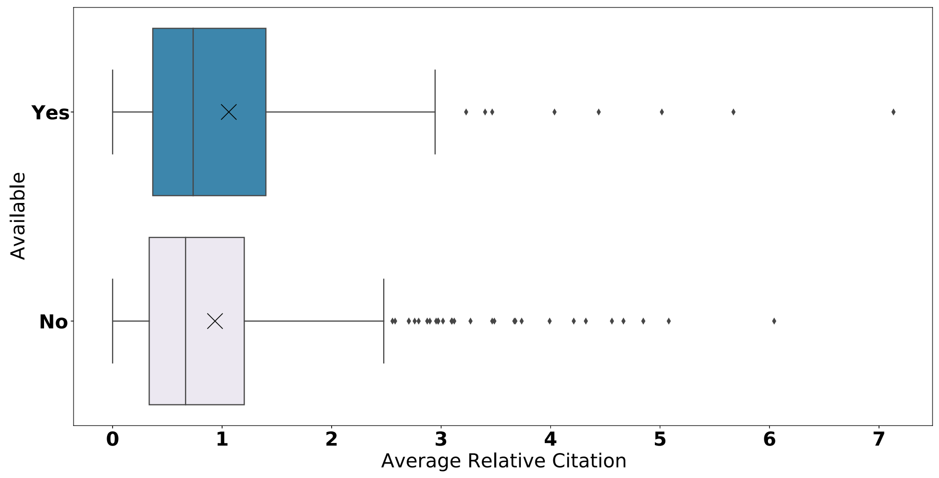

Fig. 7 Distribution of *arc* for available artifacts (— → `Median`, X → `Mean`)

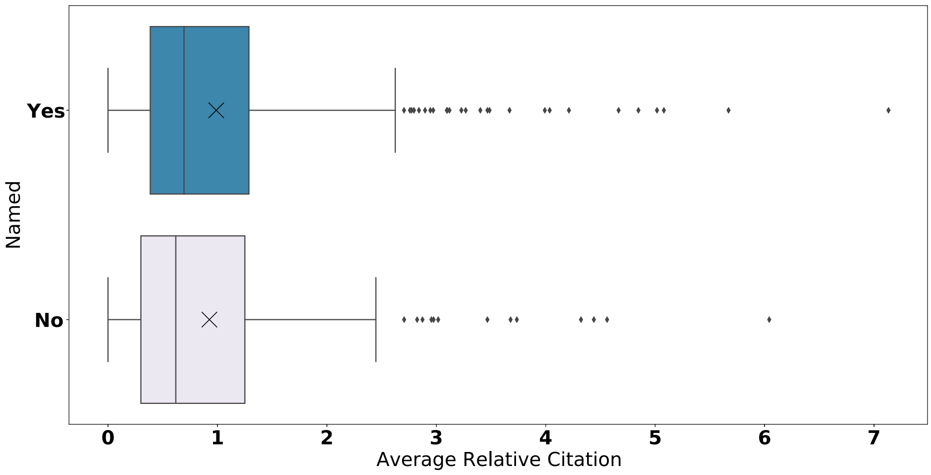

Fig. 8 Distribution of *arc* for named artifacts (— → `Median`, `X` → `Mean`)

ANOVA on ranks)[24] with a significance level of $\alpha = 0.05$ to check whether these differences are significant. The Kruskal-Wallis test is suitable for our analysis, as it does not require a normal distribution, and because we consider only one factor per test. We did not find any significant differences between the groups of available/non-available artifacts as well as for named/non-named artifacts. However, for the differences in the mean arc values of linked and non-linked artifacts, the test results show a significant difference. To quantify this effect, it is recommended to compute the effect size. Since we are dealing with potentially non-gaussian data, we used Cliff's δ and computed its confidence interval for $\alpha = 0.05$ (Hogarty and Kromrey 1999; Kitchenham et al. 2017). Cliff's δ is denoted as

$$\delta = Pr\{X > Y\} - Pr\{X < Y\}$$

where $Pr\{X > Y\}$ describes the probability of a randomly selected value of group X (e.g., arc of a paper with a linked artifact) being greater than a randomly selected value of another group Y (e.g., arc of a paper with a non-linked artifact)—and vice versa for $Pr\{X < Y\}$. In our case, this value is $\delta \approx 0.10$ with the 95 % confidence interval of [0.008, 0.192]. With regard to the respective guidelines (Kitchenham et al. 2017), this indicates a small, but positive effect on the arc, and thus on the scientific impact (with respect to the citation counts). This is also supported by the respective mean and median values of the *arc* of linked artifacts (cf. Table 4). We argue that this outcome is reasonable, considering that the link may not be working anymore, but most likely has worked when the paper first appeared and influenced other researchers. So, we conclude that making software artifacts publicly available, and providing other researchers with a link to guide them to the artifact, positively affects the impact of research.

[24]This test usually refers to the comparison of two or more independent groups. We also ran the more specific Mann-Whitney-U test, achieving the same results.

Table 4 Mean and median *arc* values of SAPs with linked, available, and named artifacts

	arc Artifact linked		*arc* Artifact available		*arc* Artifact named	
	mean	median	mean	median	mean	median
no	0.915	0.601	0.935	0.667	0.926	0.619
yes	1.028	0.770	1.061	0.735	0.989	0.694

Insights for **RQ$_3$**

- Providing links to the artifacts of SAPs is (slightly) positively correlated with these SAPs' scientific impact in terms of citations
- The current availability of an artifact seems not correlated to the impact
- The presence of names for artifacts seems not correlated to impact

5 Discussion

Our findings show that the vast majority of artifacts from SAPs published at ICSE in the studied years are not readily available. In the first part of this section, we discuss possible reasons and hurdles that prevent researchers from publishing artifacts. For reasons explained in Section 2, we did not gather data on why artifacts were not made available in our empirical study. Nevertheless, this is an important aspect when analyzing artifact publishing behavior, so we draw on related work as well as on our own experiences as researchers in the software-engineering domain.

In the second part of the section, we derive lessons learned based on our findings and impressions from this empirical study. These can serve two main purposes. First, they can help authors of SAPs by pointing out some pitfalls. Second, they underpin the importance of the properties demanded by artifact-publishing and open-science guidelines.

5.1 Reasons Not to Publish Artifacts

Open science and especially artifact availability suffer from a variety of similar issues as open-access publishing (Swan 2006; Haupt et al. 2018; Méndez Fernández et al. 2019).[25]

- Properly publishing an artifact requires a lot of initial (e.g., documentation, packaging) and maintenance effort (e.g., updating information when an author's affiliation changes). This requires much time and is usually not awarded (Méndez Fernández et al. 2019).
- Additional efforts occur when authors submit their software artifacts to venues with a double-blind review process. While double-blind reviews can reduce reviewing biases (Le Goues et al. 2018) and are preferred by a majority of authors (Prechelt et al. 2018),

[25]Carina Haupt also gave a talk (in German) on impediments regarding the publishing of software artifacts: https://media.ccc.de/v/gpn18-41-publish-your-research-warum-ffentlich-finanzierte-forschung-nicht-verffentlicht-wird.

the authors also have to anonymize their artifacts and data (Méndez Fernández et al. 2019).[26]
- Researchers may be unaware or uninterested in the benefits of publishing artifacts (Haupt et al. 2018).
- Selecting a suitable software license is simple when publishing artifacts that have been developed from scratch. However, this selection can become extremely complicated when several, potentially contradicting licenses of the artifact's components must be considered (Schreiber and Haupt 2017; Almeida et al. 2017; Méndez Fernández et al. 2019).
- Similarly, some authors and research institutions may have copyright concerns, which prevent them from publishing their artifact as open source.
- Other legal restrictions, such as conflicts with personal-data protection regulations like the General Data Protection Regulation (GDPR) in Europe, can be an additional obstacle (Méndez Fernández et al. 2019).[27]
- Researchers, especially in software engineering, may also be ashamed of their source code. While we are generally aware of what constitutes "good" software, prototypes and proofs-of-concept are seldom up to these standards. Researchers may, therefore, decide to not publish these "prototypical" software artifacts.
- Software artifacts are occasionally published after paper acceptance, for instance, because they may require additional consolidation and polishing to be understandable and useful to others. Due to the pressure of approaching submission deadlines, researchers may decide to postpone such "cosmetic" tasks until acceptance. While badges provide an incentive to publish artifacts immediately with the paper, the decision if and when to publish artifacts remains with the authors.
- Researchers may understandably withhold an artifact for some time if it is part of a larger project that has not yet been published in its entirety.

Usually, developing high-quality artifacts, providing user manuals, and proper documentation, as well as maintaining an artifact's availability is paired with a lot of effort. Some institutions, such as NASA (cf. Section 2), provide different resources to support these processes. However, not all universities and research institutions have the means to provide such support.

5.2 Lessons Learned

An important point we want to stress is that we *do not* blame any author for not publishing their artifacts. As we have shown in the previous section, there are several important and valid reasons for not publishing artifacts. Instead, we want to motivate the software-engineering community to further support initiatives like the ACM Badges or ESE OpenScience initiative. To support authors and such initiatives, we now discuss our lessons learned and their potential implications to improve current practices.

Clearly Communicate Artifacts Some authors did not clearly express whether they implemented an artifact or not. Although from our perspective, it was often very unlikely that

[26]Daniel Graziotin gave an example of how to disclose data for double-blind reviews at: https://ineed.coffee/5205/

[27]https://github.com/emsejournal/openscience/

the research was performed in a completely manual way, some papers failed to communicate whether a software artifact was implemented to produce the described results. A good way to provide this clarity is to name artifacts and to avoid vague terms like "approach" or "algorithm" when referring to software artifacts. If researchers want to express that an implementation is still in an early stage, we believe it should be consistently referred to as a prototype. Our findings show that more than two-thirds of all analyzed papers named their artifacts. However, software is very mutable, and simple naming does not guarantee that the correct version of an artifact can be retrieved. For this reason, and as suggested by several guidelines, we stress the usage of unique identifiers to refer to particular versions of software artifacts.

Your Weblinks will Break Even though we found a positive trend regarding the availability of artifacts at ICSE, we have also identified a group of linked artifacts (between 18.4 % in 2017 and 73.7 % in 2009) that were unavailable. Mostly, this was due to links being orphaned as authors changed their affiliations without properly forwarding to the new sites. While there is a trend towards publishing in open-source repositories, their long-term availability is questionable. Prominent examples are Google Code,[28] and, most recently, the discontinuation of BitBucket's Mercurial support.[4] Therefore, the software-engineering community and research institutions should—as suggested by guidelines, strive for persistent, optimally non-commercial archival repositories. As shown in previous work, such repositories can be expected to have a significantly longer half-life duration than ordinary URLs (Koehler 2004). In our study, we found only a single artifact that has been published in such a persistent archive, namely Zenodo. This, however, may merely be a result of our investigated time frame. Future studies should look at the adoption of persistent archives in more recent years.

Source Code, Binaries, or ...? We have seen that the majority of artifacts has been published as source code. However, within the guidelines exist varying priorities regarding the type of distribution. We believe that this should depend on the purpose for which an artifact is published. When aiming to replicate results, it is more important to have a prepared environment with all necessary dependencies (e.g., in a software container), as well as the compiled artifact and suitable test data. Having only the source code can impose additional effort for the compilation and resolution of dependencies. However, if the goal is to build on an existing artifact, access to the source code is usually the only viable solution. Thus, providing the artifact as source code, and pre-compiled binaries in a ready-to-use environment like a virtual machine or container would be the most flexible solution. In our analysis, we found that only a negligible proportion of software artifacts was made available as containers. We encourage the software-engineering community to further adopt such practices, as present in the related work (cf. Section 7).

Embrace Replications Several conferences and journals have introduced the concept of badges to indicate that authors made their artifacts available for replication, or that others were already able to replicate the results. A particular problem with such badges is that they only represent the state of the artifact at the time of publication or review. Therefore, we denote these badges as *static*. We have doubts concerning the longevity of such static badges, for example, because the respective programming languages or frameworks may

[28] https://opensource.googleblog.com/2015/03/farewell-to-google-code.html

become outdated and disappear. Moreover, researchers may find issues in artifacts a long time after the badges were granted. We argue that the community should further encourage and recognize papers that replicate research results, not only the paper that contributed the results in the first place. An alternative concept could be *living badges* that could become a valuable feature of persistent storage archives. These badges could be updated over time, for example, when a certain number of independent research groups could reliably replicate the results for an artifact or when positive results were achieved by applying the tool to different data sets. Such badges could also link to new versions of the artifact including new features or important bug fixes contributed by the original authors or other researchers. For the sake of replication, we believe that existing guidelines do not sufficiently address some questions of how the long-term replicability of research can be ensured. As an example, even if an artifact is available years after the SAP's publication, it may no longer be executable because of irretrievable dependencies to outdated compilers, frameworks, or operating systems. To avoid this, publishing artifacts and dependencies bundled in a container may be the best option.

6 Threats to Validity

In our analysis, we made assumptions that may threaten the validity of our results. Following a common guideline (Wohlin et al. 2012), we consider threats to construct, internal, and external validity.

6.1 Construct Validity

Construct validity regards problems in the methodology (e.g., that our metrics do not properly represent the properties we wanted to measure). Some potential threats concern the way we measured the availability of artifacts. Since we considered only ICSE publications, it may be possible that artifacts that were referenced, but not linked in an ICSE SAP, were actually introduced and linked to in papers published at other venues. Depending on how frequently this happens in practice, it may affect the validity of our results. However, in such cases, we would expect the authors to explicitly cite the relevant papers as well as the correct version information for the artifacts.

If no link is offered, researchers may still successfully search the web (e.g., on the website of the research group) or contact the corresponding author to ask for the artifact. To assess the efficiency of web searches, we checked a subset of 107 named artifacts that we classified as not available, using the complete set of one analyst and the papers from the years 2007, 2008, 2016, and 2017 of another analyst. By searching on GitHub, personal, and academic websites, we were able to retrieve 17 artifacts ($\approx$ 16 % of the 107 SAPs). As the search process was fairly unstructured and yielded few results, we did not include it in our evaluation. Furthermore, we refrained from contacting the corresponding authors of the 441 SAPs for which we did not find an artifact for practical reasons. Nevertheless, our results show that most artifacts cannot be found directly through links in the papers, but require additional search effort. Also, since there is usually no version information about the artifact in a paper, researchers cannot be certain whether they retrieved the exact artifact with which the results were obtained. Note that this may also be true for non-persistent archives, as these may keep several versions of one artifact.

Another threat is that we did not check all properties we found in existing guidelines (cf. Table 1). In particular, we did not check whether artifacts were runnable and did not replicate the results. Thus, our results only provide an upper bound; the real number of replicable results may be lower. Moreover, as we discuss in Section 7, there exist additional impact assessments besides citation counts. For example, we did not measure the industrial impact or the assessed impact in a qualitative manner. However, citation counts are generally accepted in research and can be efficiently determined for our dataset of 789 papers. Finally, our clustering method may not properly capture actual research communities, which would impact the validity of these analyses.

6.2 Internal Validity

When analyzing papers manually, there definitely is a subjective factor, because when human analysts interpret text, they may put emphasis on different factors or simply make mistakes. To mitigate this threat, we validated the classifications of 132 (16.6 %) papers as explained in Section 3.3. As expected, we made some mistakes during the classification. However, the error rate of less than 10 % in the validated set of papers indicates that our results are nevertheless likely to be valid.

Another threat to the internal validity is that we trusted the *dblp*, *Scopus*, and *CrossRef* databases to be correct as well as complete in their representation of ICSE. In terms of completeness, we validated the number of 789 ICSE research track publications listed in *dblp* by comparing it to the final reports of the program committees, the conference proceedings, and the conference programs. Combined, these sources indicate that the total number of relevant publications should have been 795. We cannot be sure whether our way of counting, *dblp*'s index, or the conference reports are at fault here. However, since this concerns only six potential papers, we believe that the difference is statistically negligible and did not necessitate further analyses. Since, for the vast majority of SAPs, we showed that their artifacts are not available, the relevance of our results is still apparent. Finally, our citation counts may be inaccurate, due to missing citations in *CrossRef*. However, this threat would also remain for any other source for the citation counts. *CrossRef* has the advantage that it is not limited, for example, to a single publisher.

6.3 External Validity

Since we analyzed only ICSE papers, we cannot directly adapt our results to other scientific areas outside of computer-science or even outside of software engineering. Considering software engineering, however, ICSE is arguably the most established conference with a high quality standard. Thus, our results may not be directly transferable, but—except for specialized journals—we would assume that ICSE is representative for our community. Nevertheless, we call for additional research on comparing artifact-publishing behavior across different software-engineering venues.

7 Related Work

Publishing and Using Software in Science Due to increased computation power and more intuitive user interfaces paired with a vast amount of data for analysis, several scientific disciplines employ software artifacts to produce their research results. This improves the ability to replicate results, clarify the solution, and may lead to more citations (Pradal et al. 2013; Lowndes et al. 2017). However, (Joppa et al. 2013) found that the majority of scientists select artifacts based on recommendations, well-respected authors, or simplicity of usage.

Only 8 % performed a solid validation, even though processes (Liu and Salganik 2019) and automatic support (Giannoulatou et al. 2014) for artifacts exist. When researchers use artifacts, they usually reference them. This is problematic if the original author does not clarify the exact version. Moreover, researchers are more likely to refer to citable papers than to the artifact website (Li et al. 2019). Simple availability of code and experimental data is not always sufficient to replicate results. So, researchers have suggested several technical solutions, such as, software containers like Docker (Boettiger 2015), cloud platforms (Trautsch et al. 2018), or electronic papers like ActivePapers (Hinsen 2014), which is similar to Jupyter notebooks.[29] In addition to the guideline of (Wilson et al. 2017), several other guidelines discuss how software artifacts should be developed and published (Jörg et al. 2016; Benureau and Rougier 2018; de Souza et al. 2019) or discuss necessary skills for researchers to publish their artifacts (Hampton et al. 2017; Johanson and Hasselbring 2018). Moreover, experience reports from other disciplines like geography, ecology, and bioinformatics provide insights into publishing or replicating results with software artifacts (Lowndes et al. 2017; Kim et al. 2018; Konkol et al. 2019).

Work on Open-Access Papers and Data Publishing software artifacts is closely related to the topics of open access and open data. Studies show that open-access papers are cited earlier (Kurtz and Brody 2006) and are correlated with higher citation counts (Antelman 2004; Piwowar et al. 2018; Lewis 2018). Surveys also determined that researchers have a positive view and are willing to provide their papers and data publicly (Swan 2006; Laakso and Polonioli 2018). This correlates with our detected trend of more and more artifacts being linked in the last decade of ICSE. Finally, there exist guidelines for publishing and managing open data (Kratz and Strasser 2014; Poldrack and Poline 2015), and studies that indicate the positive effect of badges on publicly accessible data (Kidwell et al. 2016; Rowhani-Farid et al. 2017). In an experience report on Zenodo, (Sicilia et al. 2017) provide insights on how communities work with such shared data.

Work on Impact and Transfer Analysis While the citation count is a common metric for research impact, it is not the only one. (Agarwal et al. 2016) present several metrics for assessing the impact of researchers, papers, and venues. Other researchers consider impact more qualitatively, as it cannot always be expressed purely by numbers. For example, (Morton 2015) introduced a framework tailored to social sciences. Some researchers define impact based on adoption in practice. Therefore, related work analyzed how research should be conducted and presented to be usable in industry (Schröter et al. 2017; von Nostitz-Wallwitz et al. 2018b).

Analysis of Software-Engineering Research Our work aligns to other studies on software-engineering research. These studies present critical self-reflections of researchers' practices and actions, and aim to rectify them. In the past, studies analyzed realistic empirical software-engineering studies (Sjøberg et al. 2002), the role of structured abstracts (Budgen et al. 2008), whether students represent a valid surrogate for professionals in empirical studies (Salman et al. 2015), usage of threats to validity (Siegmund et al. 2015; Schröter et al. 2017), the correctness of study results (Jørgensen et al. 2016), trending software-engineering research (Garousi and Mäntylä 2016), the effectiveness of double-blind reviews

[29] https://jupyter.org/

in software engineering (Le Goues et al. 2018), and negative factors in software-analytics research (Menzies and Shepperd 2019).

Despite all the aforementioned efforts, we are not aware of any paper that empirically analyzes to what extent researchers publish their software artifacts. Our impact measure (citation analysis) is similar to other works, but the scope and purpose of our study are unique. Consequently, our work is orthogonal to existing studies and provides valuable insights, especially for researchers.

8 Conclusion

In this article, we presented the results of the first large-scale empirical study on the evolution, status, and scientific impact of publishing software artifacts at ICSE. To this end, we classified 789 research-track papers from 2007 to 2017. We analyzed (1) the prevalence of SAPs, (2) *if* and *how* researchers made their artifacts available, and (3) the scientific impact of artifact publication.

Roughly 76.6 % of the publications we analyzed are SAPs according to our definition, and a similar ratio applies to each year of ICSE. While we have found that the number of linked and available artifacts seems to linearly increase, our results still show a large potential for improvement, as only 58.5 % of the artifacts in the most recent, analyzed year (2017) are actually available through a link. We found a statistically significant, but small, positive correlation between linking software artifacts in SAPs and the scientific impact of these papers in terms of citations. This indicates that making an artifact available, increases the chance of receiving more scientific attention.

In this study, we only looked at ICSE, as it is arguably the number one software-engineering conference and has been in this position for many years. However, we are certain that studying and comparing artifact availability at other venues and in particular with respect to recent developments could provide further valuable insights. For example, the adoption of artifact badges and persistent, archival repositories are particularly interesting, since their aim is to change the artifact publishing behavior in the scientific community. Another related subject for future research is the evolution of open-science guidelines and how these can be enhanced. Therefore, we plan to investigate the impact of these trends. Also, we see potential for expanding the community-based analysis with more refined clustering algorithms or with concepts such as LDA topic modeling. These could lead to more tangible interpretations of communities. Finally, we plan to analyze available repositories and archives to understand their support for long-term persistence and open science, ideally leading to a definition of requirements in the form of a checklist.

Acknowledgements We would like to thank the reviewers at ESE for their detailed and invaluable feedback, Prof. Daniel Katz for clarifications on the JOSS and JORS journals, as well as Dr. Christian Braune for his expertise in clustering and intelligent data analysis.

References

Agarwal A, Durairajanayagam D, Tatagari S, Esteves SC, Harlev A, Henkel R, Roychoudhury S, Homa S, Puchalt NG, Ramasamy R, Majzoub A, Dao Ly K, Tvrda E, Assidi M, Kesari K, Sharma R, Banihani S, Ko E, Abu-Elmagd M, Gosalvez J, Bashiri A (2016) Bibliometrics: tracking research impact by selecting the appropriate metrics. Asian J Androl 18(2):296–309. https://doi.org/10.4103/1008-682X.171582

Almeida DA, Murphy GC, Wilson G, Hoye M (2017) Do software developers understand open source licenses? In: Proc. 25th Int. Conf. Program Compr. (ICPC). IEEE, pp 1–11. https://doi.org/10.1109/ICPC.2017.7

Antelman K (2004) Do open-access articles have a greater research impact? Coll Res Libr 65(5):372–382. https://doi.org/10.5860/crl.65.5.372

Bellon S, Koschke R, Antoniol G, Krinke J, Merlo E (2007) Comparison and evaluation of clone detection tools. IEEE Trans Softw Eng 33(9):577–591. https://doi.org/10.1109/TSE.2007.70725

Benureau FCY, Rougier NP (2018) Re-run, repeat, reproduce, reuse, replicate: transforming code into scientific contributions. Front Neuroinform 11:69: 1–8. https://doi.org/10.3389/fninf.2017.00069

Boettiger C (2015) An introduction to docker for reproducible research. SIGOPS Oper Syst Rev 49(1):71–79. https://doi.org/10.1145/2723872.2723882

Boisvert RF (2016) Incentivizing reproducibility. Commun ACM 59(10):5–5. https://doi.org/10.1145/2994031

Budgen D, Kitchenham BA, Charters SM, Turner M, Brereton P, Linkman SG (2008) Presenting software engineering results using structured abstracts: a randomised experiment. Empir Softw Eng 13(4):435–468. https://doi.org/10.1007/s10664-008-9075-7

de Souza MR, Haines R, Vigo M, Jay C (2019) What makes research software sustainable? An interview study with research software engineers. In: Proc. 12th Int. Work. Coop. Hum. Asp. Softw. Eng. (CHASE). IEEE, pp 135–138. https://doi.org/10.1109/CHASE.2019.00039

Di Cosmo R (2018) Software heritage: collecting, preserving, and sharing all our source code. In: Proc. 33rd Int. Conf. Autom. Softw. Eng. (ASE). ACM, pp 1–2. https://doi.org/10.1145/3238147.3241985

Diebold P, Vetrò A (2014) Bridging the gap: SE technology transfer into practice: study design and preliminary results. In: Proc. 8th Int. Symp. Empir. Softw. Eng. Meas. (ESEM). ACM, pp 1–4. https://doi.org/10.1145/2652524.2652552

Fernandes E, Oliveira J, Vale G, Paiva T, Figueiredo E (2016) A review-based comparative study of bad smell detection tools. In: Proc. 20th Int. Conf. Eval. Assess. Softw. Eng. (EASE). ACM, pp 18:1–18:12. https://doi.org/10.1145/2915970.2915984

Fu W, Menzies T (2017) Revisiting unsupervised learning for defect prediction. In: Proc. 11th Eur. Softw. Eng. Conf./Found. Softw. Eng. (ESEC/FSE). ACM, pp 72–83. https://doi.org/10.1145/3106237.3106257

Garousi V, Mäntylä MV (2016) Citations, research topics and active countries in software engineering: a bibliometrics study. Comput Sci Rev 19:56–77. https://doi.org/10.1016/j.cosrev.2015.12.002

Garousi V, Petersen K, Ozkan B (2016) Challenges and best practices in industry-academia collaborations in software engineering: a systematic literature review. J Inf Softw Technol 79:106–127. https://doi.org/10.1016/j.infsof.2016.07.006

Giannoulatou E, Park SH, Humphreys DT, Ho JW (2014) Verification and validation of bioinformatics software without a gold standard: a case study of BWA and Bowtie. BMC Bioinform 15(16):S15. https://doi.org/10.1186/1471-2105-15-S16-S15

Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826. https://doi.org/10.1073/pnas.122653799

Hagberg AA, Schult DA, Swart PJ (2008) Exploring network structure, dynamics, and function using NetworkX. In: Proc. 7th Python Science Conf. (SciPy), pp 11–15

Hampton SE, Jones MB, Wasser LA, Schildhauer MP, Supp SR, Brun J, Hernandez RR, Boettiger C, Collins SL, Gross LJ, Fernández DS, Budden A, White EP, Teal TK, Labou SG, Aukema JE (2017) Skills and knowledge for data-intensive environmental research. Bioscience 67(6):546–557. https://doi.org/10.1093/biosci/bix025

Haupt C, Schlauch T, Meinel M (2018) The software engineering initiative of DLR: overcome the obstacles and develop sustainable software. In: Proc. 13th Int. Work. Softw. Eng. Science (SE4Science). ACM, pp 16–19. https://doi.org/10.1145/3194747.3194753

Hinsen K (2014) Activepapers: a platform for publishing and archiving computer-aided research. F1000Res 3(289):1–26. https://doi.org/10.12688/f1000research.5773.3

Hogarty KY, Kromrey JD (1999) Using SAS to calculate tests of Cliff's delta. In: Proc. SAS Users' Group Int. (SUGI), pp 1389–1393

Hutchins BI, Yuan X, Anderson JM, Santangelo GM (2016) Relative citation ratio (RCR): a new metric that uses citation rates to measure influence at the article level. PLOS Biol 14(9):1–25. https://doi.org/10.1371/journal.pbio.1002541

Johanson A, Hasselbring W (2018) Software engineering for computational science: past, present, future. Comput Sci Eng 20(2):90–109. 10.1109/MCSE.2018.108162940

Joppa LN, McInerny G, Harper R, Salido L, Takeda K, O'Hara K, Gavaghan D, Emmott S (2013) Troubling trends in scientific software use. Science 340(6134):814–815. https://doi.org/10.1126/science.1231535

Jörg F, Heiland J, Himpe C, Saak J (2016) Best practices for replicability, reproducibility and reusability of computer-based experiments exemplified by model reduction software. AIMS Math 1(3):261–281. https://doi.org/10.3934/Math.2016.3.261

Jørgensen M, Dybå T, Liestøl K, Sjøberg DI (2016) Incorrect results in software engineering experiments: How to improve research practices. J Syst Softw 116:133–145. https://doi.org/10.1016/j.jss.2015.03.065

Katz DS, Niemeyer KE, Smith AM (2018) Publish your software: introducing the. Journal of Open Source Software (JOSS). Comput Sci Eng 20(3):84–88. https://doi.org/10.1109/MCSE.2018.03221930

Kidwell MC, Lazarević LB, Baranski E, Hardwicke TE, Piechowski S, Falkenberg LS, Kennett C, Slowik A, Sonnleitner C, Hess-Holden C, Errington TM, Fiedler S, Nosek BA (2016) Badges to acknowledge open practices: a simple, low-cost, effective method for increasing transparency. PLOS Biol 14(5):1–15. https://doi.org/10.1371/journal.pbio.1002456

Kim YM, Poline JB, Dumas G (2018) Experimenting with reproducibility: a case study of robustness in bioinformatics. GigaScience 7(7):1–8. https://doi.org/10.1093/gigascience/giy077

Kitchenham BA, Madeyski L, Budgen D, Keung J, Brereton P, Charters SM, Gibbs S, Pohthong A (2017) Robust statistical methods for empirical software engineering. Empir Softw Eng 22(2):579–630. https://doi.org/10.1007/s10664-016-9437-5

Koehler W (2002) Web page change and persistence—a four-year longitudinal study. J Am Soc Inf Sci Tec 53(2):162–171. https://doi.org/10.1002/asi.10018

Koehler W (2004) A longitudinal study of web pages continued: a consideration of document persistence. Inf Res 9(2), http://InformationR.net/ir/9-2/paper174.html

Konkol M, Kray C, Pfeiffer M (2019) Computational reproducibility in geoscientific papers: insights from a series of studies with geoscientists and a reproduction study. O Int J Geogr Inf Sci 33(2):408–429. https://doi.org/10.1080/13658816.2018.1508687

Kratz J, Strasser C (2014) Data publication consensus and controversies. F1000Res 3(94):1–21. https://doi.org/10.12688/f1000research.3979.3

Kurtz M, Brody T (2006) The impact loss to authors and research. In: Jacobs N (ed) Open access: key strategic, technical and economic aspects, Chandos. https://eprints.soton.ac.uk/40867/

Laakso M, Polonioli A (2018) Open access in ethics research: an analysis of open access availability and author self-archiving behaviour in light of journal copyright restrictions. Scientometrics 116(1):291–317. https://doi.org/10.1007/s11192-018-2751-5

Le Goues C, Brun Y, Apel S, Berger E, Khurshid S, Smaragdakis Y (2018) Effectiveness of anonymization in double-blind review. Commun ACM 61(6):30–33. https://doi.org/10.1145/3208157

Lewis CL (2018) The open access citation advantage: does it exist and what does it mean for libraries? Inform Technol Libr 37(3):50–65. https://doi.org/10.6017/ital.v37i3.10604

Li K, Chen PY, Yan E (2019) Challenges of measuring the impact of software: an examination of the lme4 R package. J Informetrics 13(1):449–461. https://doi.org/10.1016/j.joi.2019.02.007

Liu D, Salganik M (2019) Successes and struggles with computational reproducibility: lessons from the fragile families challenge. Socius 5:1–21. https://doi.org/10.1177/2378023119849803

Lo D, Nagappan N, Zimmermann T (2015) How practitioners perceive the relevance of software engineering research. In: Proc. 10th Eur. Softw. Eng. Conf./Found. Softw. Eng. (ESEC/FSE). ACM, pp 415–425. https://doi.org/10.1145/2786805.2786809

Lowndes JSS, Best BD, Scarborough C, Afflerbach JC, Frazier MR, O'Hara CC, Jiang N, Halpern BS (2017) Our path to better science in less time using open data science tools. Nat Ecol Evol 1(6):0160: 1–7. https://doi.org/10.1038/s41559-017-0160

Méndez Fernández D, Graziotin D, Wagner S, Seibold H (2019) Open science in software engineering. arXiv:1904.06499

Méndez Fernández D, Monperrus M, Feldt R, Zimmermann T (2019) The open science initiative of the empirical software engineering journal. Empir Softw Eng 24(3):1057–1060. https://doi.org/10.1007/s10664-019-09712-x

Menzies T, Shepperd M (2019) Bad smells in software analytics papers. J Inf Softw Technol 112:35–47. https://doi.org/10.1016/j.infsof.2019.04.005

Monperrus M (2014) A critical review of automatic patch generation learned from human-written patches: essay on the problem statement and the evaluation of automatic software repair. In: Proc. 36th Int. Conf. Softw. Eng. (ICSE). ACM, pp 234–242. https://doi.org/10.1145/2568225.2568324

Morton S (2015) Progressing research impact assessment: a contributions approach. Res Eval 24(4):405–419. https://doi.org/10.1093/reseval/rvv016

Ossher H, Harrison W, Tarr P (2000). In: Proc. 22nd Int. Conf. Softw. Eng. (ICSE). ACM, pp 261–277. https://doi.org/10.1145/336512.336569

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Piwowar H, Priem J, Larivière V, Alperin JP, Matthias L, Norlander B, Farley A, West J, Haustein S (2018) The state of OA: a large-scale analysis of the prevalence and impact of open access articles. PeerJ 6:e4375. https://doi.org/10.7717/peerj.4375

Poldrack RA, Poline JB (2015) The publication and reproducibility challenges of shared data. Trends Cogn Sci 19(2):59–61. https://doi.org/10.1016/j.tics.2014.11.008

Pradal C, Varoquaux G, Langtangen HP (2013) Publishing scientific software matters. J Comput Sci 4(5):311–312. https://doi.org/10.1016/j.jocs.2013.08.001

Prechelt L, Graziotin D, Méndez Fernández D (2018) A community's perspective on the status and future of peer review in software engineering. J Inf Softw Technol 95:75–85. https://doi.org/10.1016/j.infsof.2017.10.019

Rowhani-Farid A, Allen M, Barnett AG (2017) What incentives increase data sharing in health and medical research? a systematic review. Research Integrity and Peer Review 2:4:1–10. https://doi.org/10.1186/s41073-017-0028-9

Roy CK, Cordy JR, Koschke R (2009) Comparison and evaluation of code clone detection techniques and tools: a qualitative approach. Sci Comput Program 74(7):470–495. https://doi.org/10.1016/j.scico.2009.02.007

Salman I, Misirli AT, Juristo N (2015) Are students representatives of professionals in software engineering experiments? In: Proc. 37th Int. Conf. Softw. Eng. (ICSE). IEEE, pp 666–676. https://doi.org/10.1109/ICSE.2015.82

Schreiber A, Haupt C (2017) Sharing knowledge about open source licenses at DLR. In: Proc. 13th Int. Symp. Open Collab. (OpenSym). ACM, pp 26:1–26:4. https://doi.org/10.1145/3125433.3125470

Schröter I, Krüger J, Ludwig P, Thiel M, Nürnberger A, Leich T (2017) Identifying Innovative documents: Quo vadis? In: Proc. 19th Int. Conf. Enterp. Inf. Syst. (ICEIS). ScitePress, pp 653–658. https://doi.org/10.5220/0006368706530658

Schröter I, Krüger J, Siegmund J, Leich T (2017) Comprehending studies on program comprehension. In: Proc. 25th Int. Conf. Program Compr. (ICPC). IEEE, pp 308–311. https://doi.org/10.1109/ICPC.2017.9

Sicilia MA, García-Barriocanal E, Sánchez-Alonso S (2017) Community curation in open dataset repositories: insights from Zenodo. Procedia Comput Sci 106:54–60. https://doi.org/10.1016/j.procs.2017.03.009

Siegmund J, Siegmund N, Apel S (2015) Views on internal and external validity in empirical software engineering. In: Proc. 37th Int. Conf. Softw. Eng. (ICSE). IEEE, pp 9–19. https://doi.org/10.1109/ICSE.2015.24

Sjøberg DIK, Anda B, Arisholm E, Dybå T, Jørgensen M, Karahasanovic A, Koren EF, Vokác M (2002) Conducting realistic experiments in software engineering. In: Proc. 1st Int. Symp. Empir. Soft. Eng. (ISESE). IEEE, pp 17–26. https://doi.org/10.1109/ISESE.2002.1166921

Swan A (2006) The culture of open sccess: researchers' views and responses. In: Jacobs N (ed) Open access: key strategic, technical and economic aspects, Chandos. http://eprints.soton.ac.uk/id/eprint/262428

Thomee B, Riegler M, Fd Simone, Simon G (2018) Sharing and reproducibility in ACM SIGMM. SIGMultimedia Rec 10(2):1:1–1:1. https://doi.org/10.1145/3264706.3264707

Trautsch F, Herbold S, Makedonski P, Grabowski J (2018) Addressing problems with replicability and validity of repository mining studies through a smart data platform. Empir Softw Eng 23(2):1036–1083. https://doi.org/10.1007/s10664-017-9537-x

Vinh NX, Epps J, Bailey J (2009) Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proc. 26th Int. Conf. Mach. Learn. (ICML). ACM, pp 1073–1080. https://doi.org/10.1145/1553374.1553511

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Jarrod Millman K, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey C, Polat I, Feng Y, Moore EW, Vand erPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, Contributors SciPy (2020) Scipy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods 17:261–272. https://doi.org/10.1038/s41592-019-0686-2

von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416. https://doi.org/10.1007/s11222-007-9033-z von Nostitz-Wallwitz I, Krüger J, Leich T (2018a) Towards improving industrial adoption: the choice of programming languages and development environments. In: Proc. 5th Int. Work. Softw. Eng. Res. Ind. Pract. (SER&IP). ACM, pp 10–17. https://doi.org/10.1145/3195546.3195548

von Nostitz-Wallwitz I, Krüger J, Siegmund J, Leich T (2018b) Knowledge transfer from research to industry: a survey on program comprehension. In: Proc. 40th Int. Conf. Softw. Eng. (ICSE). ACM, pp 300–301. https://doi.org/10.1145/3183440.3194980

Wicks MN, Dewar RG (2007) Controversy corner: a new research agenda for tool integration. J Syst Softw 80(9):1569–1585. https://doi.org/10.1016/j.jss.2007.03.089

Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017) Good enough practices in scientific computing. PLOS Comput Biol 13(6):1–20. https://doi.org/10.1371/journal.pcbi.1005510

Wohlin C, Runeson P, Höst M, Ohlsson MC (2012) Experimentation in software engineering. Springer, Berlin. https://doi.org/10.1007/978-3-642-29044-2

Robert Heumüller is a Ph.D. student at Otto-von-Guericke University of Magdeburg, where he also received his B.Sc. and M.Sc. degrees in Computer Science. In January 2016 he joined the Chair of Software Engineering, to research on automated software engineering, model-driven software development, and domain-specific languages.

Sebastian Nielebock is a Ph.D. student at the Otto-von-Guericke University of Magdeburg. He received his B.Sc. and M.Sc. degrees in Computer Systems in Engineering from there as well. Since October 2013 he is a member of the Chair of Software Engineering. His research focuses on empirical and automated software engineering, i.e., programming language analysis, automated error detection, and automated program repair.

Jacob Krüger is a PhD student and associated researcher at the Databases and Software Engineering work group of the Otto-von-Guericke University Magdeburg, currently visiting the University of Toronto in Canada. He received his M.Sc. degree in Business Informatics at the Otto-von-Guericke University in 2016, has been working as research associate at the Harz University of Applied Sciences Wernigerode, and visited Chalmers University of Technology — University of Gothenburg in Sweden. His research focuses on feature-oriented software development, with particular interests on software evolution, program comprehension, and human factors.

Frank Ortmeier is a full professor and head of the "Chair of Software Engineering (CSE)" at the Otto-von-Guericke University of Magdeburg, Germany. He received his Ph.D. degree from the University of Augsburg in 2005. After three years employed as a Post-Doc in Augsburg, he became an associate professor for "Computer Systems in Engineering" in Magdeburg in 2008. Since 2013 he is holding the Chair fo Software Engineering at OvGU. Currently, he is leading several research projects, coordinating the Bachelor' degree program "Computer Systems in Engineering" as well as the Master's degree program "Digital Engineering". He is a founding member of the university's Center for Digital Engineering, Management and Operations (CeDEMO). His research is driven by the idea of improving engineering tasks with methods from computer science – with a special focus on methods from Software Engineering, formal specification techniques, mobile assistance, and robotics.

Affiliations

Robert Heumüller[1] · **Sebastian Nielebock**[1] · **Jacob Krüger**[1,2] · **Frank Ortmeier**[1]

Sebastian Nielebock
sebastian.nielebock@ovgu.de

Jacob Krüger
jacob.krueger@ovgu.de

Frank Ortmeier
frank.ortmeier@ovgu.de

1 Otto-von-Guericke University Magdeburg, Magdeburg, Germany

2 University of Toronto, Toronto, ON, Canada