Open Source Research Software

Wilhelm Hasselbring, Kiel University

Leslie Carr, University of Southampton

Simon Hettrick, Software Sustainability Institute and University of Southampton

Heather Packer and Thanassis Tiropanis, University of Southampton

For good scientific practice, research software should be open source. It should be both archived for reproducibility and actively maintained for reusability.

In computational and computer science, research software is a central asset for development activities. For good scientific practice, the resulting research software should be open source. Established open source software licenses provide sufficient options for granting permissions such that it should be the rare exception to keep research software closed.

Proper engineering is required for obtaining reusable and sustainable research software. This way, software engineering methods may improve research in other disciplines. However, research in software engineering and computer science itself will also benefit when programs are reused. To study the state of the art in this field, we analyzed research software publishing practices in computer and computational science and observed significant differences: computational science emphasizes reproducibility, while computer science emphasizes reuse.

SOFTWARE ENGINEERING FOR SUSTAINABLE RESEARCH SOFTWARE

Research software is employed during the scientific discovery process, and it can be an object of study itself. Computational science (also known as *scientific computing*) involves the development of research software for model simulations and data analytics that facilitate the understanding of natural systems, answering questions that neither theory nor experiment alone is equipped to answer. Computational science is the application of computer science and software engineering principles to solving technical problems, whereas computer science is the study of computer hardware and software design.

Despite the increasing importance of research software to the scientific discovery process, well-established software engineering practices are rarely adopted in

Digital Object Identifier 10.1109/MC.2020.2998235
Date of current version: 30 July 2020

0018-9162/20©2020IEEE

EDITOR **DIRK RIEHLE**
Friedrich Alexander-University of Erlangen Nürnberg;
dirk.riehle@fau.de

FROM THE EDITOR

This month's column on open source takes us into the world of scientific research, where being open is (or should be) the default. This applies not only to data but to the software used for research as well the software that is the result of research. What is more natural than publishing such software as open source? Wilhelm Hasselbring and his colleagues bring us the state of the art on this important subject. We will pick up open source governance again with the next issue to finish the theme on how to use open source in products and the enterprise. As always, happy hacking, and stay safe! — *Dirk Riehle*

computational science.[3] Computer science—in particular, software engineering—may help with reproducibility and reuse to advance computational science. Publishing research software as open source[4] is an established practice in science; a popular open source platform is GitHub. Researchers also disseminate the code and data for their experiments as virtual machines on repositories such as DockerHub. Modular architectures facilitate a collaborative development process for open source research software.[5]

RELATING RESEARCH SOFTWARE TO PUBLICATIONS

To analyze the current state of research software publication, we conducted an initial study of it and the relationship between research software and research periodicals. For this analysis, research software is identified by either of the following:

- research publications that cite software repositories
- software repositories that cite research publications.

Research software is analyzed in our study through a combination of research publication and software repository metadata.

Assumptions

We have to make certain assumptions for analyzing the relationships between research software and research publications. First, we presume that a research publication refers to some GitHub repository for the related research software. Second, we suppose that somewhere in a GitHub repository a publication identifier [a digital object identifier (DOI)] is available. We do not assume bidirectional links. We are well aware that these assumptions restrict the coverage of our analysis, but the study becomes tractable and repeatable with these suppositions.

Analysis data set

More than 5,000 GitHub software repositories have been identified as research software, according to the criteria explained previously: either a research publication referenced the software repository or the software repository referenced a research periodical. As research publications, we include papers from arXiv and the Association for Computing Machinery (ACM) digital library.

Covered research areas

Our first interesting observation is that our three data sets cover quite different research areas:

- The GitHub research software set is drawn mainly from the computational sciences, particularly the life sciences [Figure 1(a)]. This is determined by resolving the DOI to obtain publication metadata at datacite.org and classifying the dissemination venue (for example, a journal or conference) in which the paper appeared.
- The ACM research software set is dominated by software engineering, information systems, social and professional topics, and human-centered computing [Figure 1(b)].
- The arXiv research software set is monopolized by computer science topics [Figure 1(c)], which mainly concern artificial intelligence (AI) topics [computer vision, machine learning, and computational linguistics; Figure 1(d)]. Thus, these computer science subareas on arXiv seem to emphasize a "publish and share as early as possible" attitude, which is encouraged by the unreviewed-publication repository on the site.

Sustainability of research software

For this study, we consider research software as sustainable if it has a long life span and remains live. We consider a repository as live if some activity occurred during the past year; otherwise it is viewed as dormant. The "life span" of a software repository is the length of time between its first and last commit activities. To analyze the sustainability of research software, we divide the software repositories between "live" and "dormant."

As presented previously, publications cited from GitHub repositories belong mainly to computational science, for which we observe an even split between live and dormant software depositories. Publications from the ACM digital library belong chiefly to computer science; for their cited software repositories, we also observe an even split between live and dormant depositories. However, the life spans of computer science software repositories are

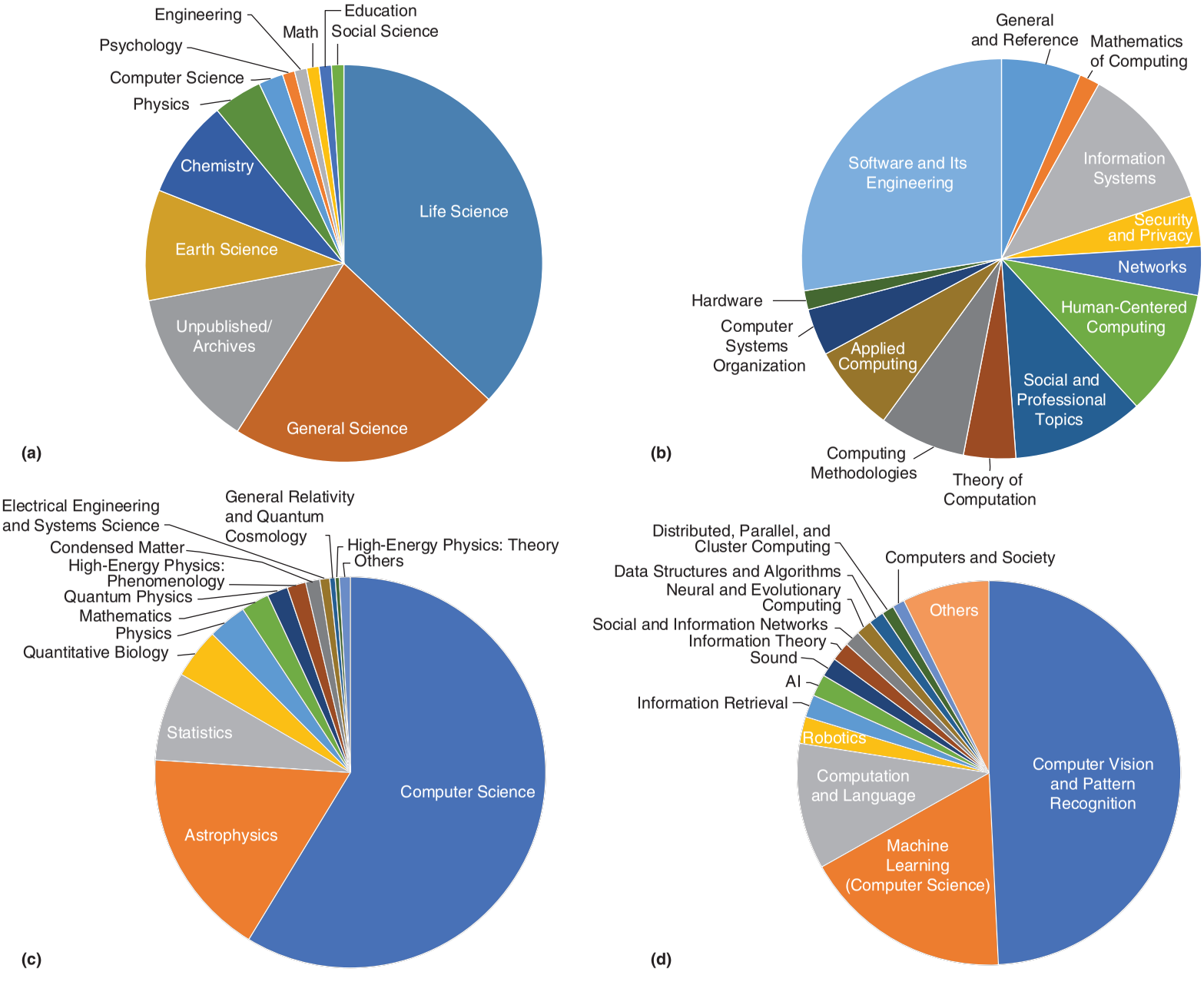

Figure 1. Research areas of publications with related GitHub repositories: (a) research areas of publications cited from GitHub repositories, (b) ACM computer science publications citing GitHub repositories, (c) arXiv publications citing GitHub repositories, and (d) computer science publications in arXiv from (c) refined into subareas.

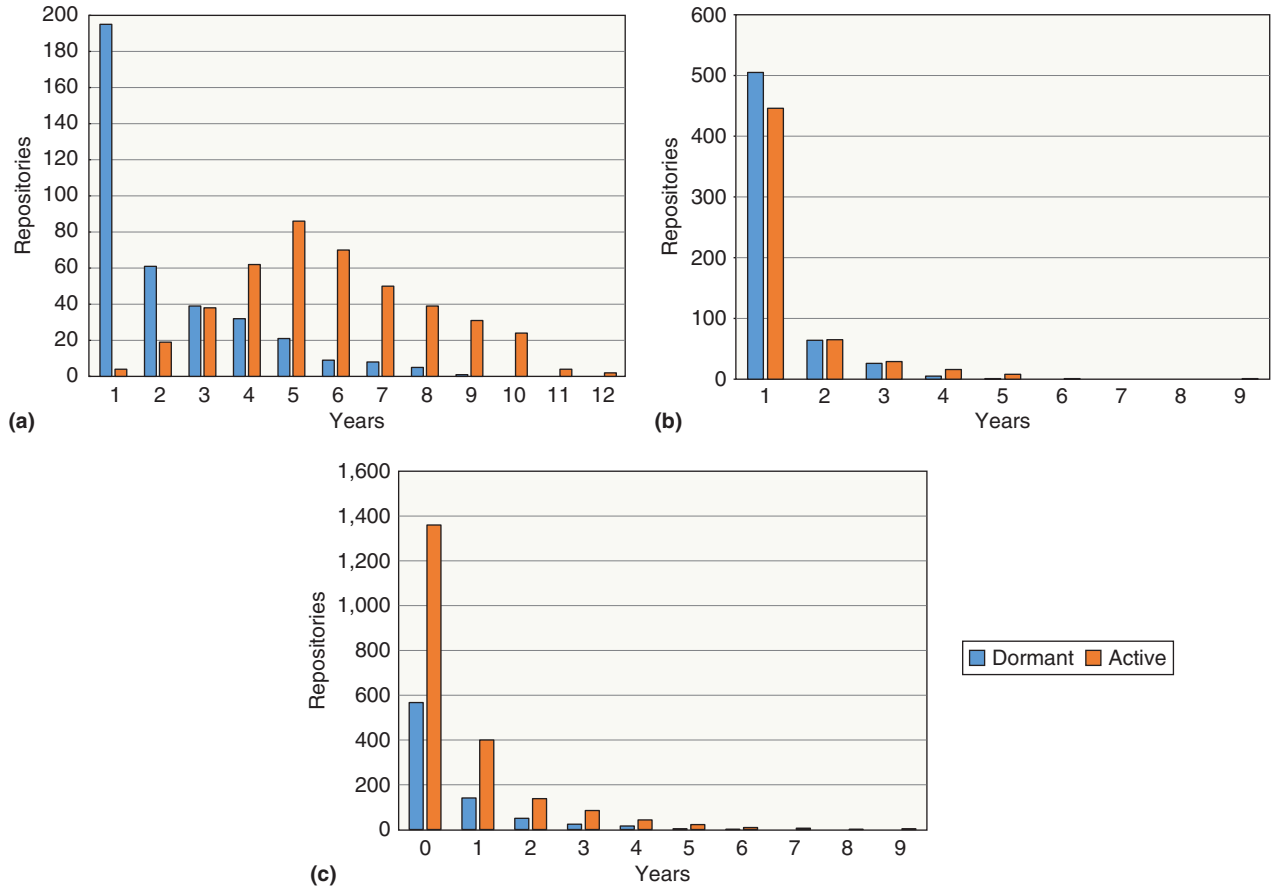

Figure 2. The life span of software repositories in years. The life span of GitHub repositories (a) cited in ACM computer science publications, (b) citing publications via DOIs, and (c) cited in arXiv publications.

hugely longer than those of the computational science software archives.

As Figure 2(a) shows, the computer science software repositories' life spans are distributed with a median of five years. Our hypothesis is that, in computer science research, commercial open source software frameworks are often employed. They are maintained through long time spans by employees of the associated companies.

According to Figure 2(b), the computational science software repositories' life spans have a distribution with a median life span of 15 days. One-third of these repositories remain live for under one day. We hypothesize that, in computational science research, research software is often published only when a corresponding paper has been distributed. The software is not further maintained on GitHub but at some private place, as before (if it is kept up at all).

Examining Figure 2(c), we see that the arXiv repositories are somewhere in between, with a median life span of eight months. Furthermore, 75% of the arXiv repositories are live. We conjecture that in parts of the AI community, the attitude of publishing as early as possible motivates researchers to develop their investigative software openly from the beginning of projects.

Relationships and categories of research software

In addition to the life span, it is interesting to take a closer look at activity related to repositories, such as the number of commits per time unit. We observe different categories and relationships between research publications and research software, including

- software as an output of research, collaboratively constructed and maintained through an active open source community
- software as an output of research, privately developed but published openly and abandoned after dissemination
- software itself as an object of study or analysis
- software that leads to a fork (in GitHub) that is independently developed as a research output and published openly (if successful, it may be fed back into the original project via GitHub pull requests)

- software used as a tool or framework to perform the research.

Besides these relationships, software is cited as related work, background, and examples. GitHub repositories are also used to publish data and reference lists to collections of software.

OPEN ISSUES AND FUTURE WORK

As mentioned, we had to make assumptions for this initial study. To make our analysis tractable and repeatable, we presume that a research publication refers to some GitHub repository for the related software or that somewhere in a GitHub repository a publication identifier is available. We are well aware that these suppositions restrict the coverage of our analysis, but even with this limited scope, we have already observed interesting differences in software publication behaviors in different research domains. In our future work, we intend to extend and refine this analysis, for instance, to perform a deeper survey of the repository activities.

Research software is not always cited with a link to a GitHub repository. It could also be published, for instance, in Bitbucket or GitLab repositories. Alternative citations may refer to papers, manuals, and books that introduce the software. As additional research publications, we will include papers from more publishers, such as the IEEE. Our initial analysis does not cover such citation links. To facilitate a more comprehensive study of the relationships between research software and publications, specific research software observatories could provide appropriate citation links and graphs.[2]

For good scientific practice, software artifacts should be published with preservation in mind. GitHub, for example, does not directly support the conservation of software "snapshots" that were used to achieve some specific research results. This may, for instance, be achieved via taking a snapshot from GitHub to be archived on Zenodo.org. GitHub supports the use, reuse, and active involvement of researchers. Zenodo facilitates the archiving and reproducibility of published research results. Thus, for the reproducibility and reusability of research software, specific-solution approaches are required. Our study revealed highly varying publication behavior in different scientific disciplines: computational science emphasizes reproducibility, while computer science stresses reuse.

Compared to research data, research software should be both archived for reproducibility and actively maintained for reusability. The combination of Zenodo (for archiving and reproducibility) and GitHub (for maintenance and reuse) may be used to achieve this. Furthermore, research software should be open source. Established open source software licenses[1] provide options that obviate the need to keep research software closed. In the vast majority of cases, some existing license will be appropriate. Only in exceptional cases and for very good reasons should research software be closed.

REFERENCES

1. M. Ballhausen, "Free and open source software licenses explained," *Computer*, vol. 52, no. 6, pp. 82–86, June 2019. doi: 10.1109/MC.2019.2907766.
2. W. Hasselbring, L. Carr, S. Hettrick, H. Packer, and T. Tiropanis, "From FAIR research data toward FAIR and open research software," *Inform. Technol.*, vol. 62, no. 1, pp. 39–47, Feb. 2020. doi: 10.1515/itit-2019-0040.
3. A. Johanson and W. Hasselbring, "Software engineering for computational science: Past, present, future," *Comput. Sci. Eng.*, vol. 20, no. 2, pp. 90–109, Mar. 2018. doi: 10.1109/MCSE.2018.021651343.
4. D. Riehle, "The innovations of open source," *Computer*, vol. 52, no. 4, pp. 59–63, Apr. 2019. doi: 10.1109/MC.2019.2898163.
5. C. Zirkelbach, A. Krause, and W. Hasselbring, "Modularization of research software for collaborative open source development," in *Proc. 9th Int. Conf. Advanced Collaborative Networks, Systems and Applications (COLLA 2019)*, 2019, pp. 1–7.

WILHELM HASSELBRING is a full professor of software engineering in the Department of Computer Science, Kiel University, Germany. Contact him at hasselbring@email.uni-kiel.de.

LESLIE CARR is a full professor of web science in the Department of Electronics and Computer Science, University of Southampton, United Kingdom. Contact him at lac@ecs.soton.ac.uk.

SIMON HETTRICK is deputy director of the United Kingdom's Software Sustainability Institute and a full professor in the Department of Electronics and Computer Science, University of Southampton, United Kingdom. Contact him at sjh@ecs.soton.ac.uk.

HEATHER PACKER is a New Frontier fellow in the Department of Electronics and Computer Science, University of Southampton, United Kingdom. Contact her at hp3@ecs.soton.ac.uk.

THANASSIS TIROPANIS is an associate professor in the Department of Electronics and Computer Science, University of Southampton, United Kingdom. Contact him at t.tiropanis@southampton.ac.uk.