Towards a New Peer Review Concept for Scientific Computing ensuring Technical Quality, Software Sustainability, and Result Reproducibility

Hartwig Anzt[1,2*], **Terry Cojean**[1], and **Eileen Kühn**[1]

[1] Karlsruhe Institute of Technology, Steinbuch Centre for Computing, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany

[2] University of Tennessee, Innovative Computing Lab, 1122 Volunteer Blvd, Knoxville, TN, 37996, USA

In this position paper we argue for implementing an alternative peer review process for scientific computing contributions that promotes high quality scientific software developments as fully-recognized conference submission. The idea is based on leveraging the code reviewers' feedback on scientific software contributions to community software developments as a third-party review involvement. Providing open access to this technical review would complement the scientific review of the contribution, efficiently reduce the workload of the undisclosed reviewers, improve the algorithm implementation quality and software sustainability, and ensure full reproducibility of the reported results. Using this process creates incentives to publish scientific algorithms in open source software – instead of designing prototype algorithms with the unique purpose of publishing a paper. In addition, the comments and suggestions of the community being archived in the versioning control systems ensure that also community reviewers are receiving credit for the review contributions – unlike reviewers in the traditional peer review process. Finally, it reflects the particularity of the scientific computing community using conferences rather than journals as the main publication venue.

1 Introduction

In academia, scholarly peer review has been established as the community's gold standard for ensuring quality norms in scientific publishing. Since Henry Oldenburg[1], who is often referred to as the father of the scientific peer review process, proposed the academic peer review in the 17th century, the process accrediting academic contributions for journal publication or conference presentation has remained mostly unchanged. The authors of unprecedented work submit their manuscript to the journal- or conference editor. The editor solicits the review of the work from scientific experts in the field, and based on the reviews the contribution is accepted, rejected, or subject to revisions. With scientific papers and the derived Hirsch-Index (H-Index) being the currency for promotion of a scientist, appointability to tenure, and academic reputation in general, scientists feel an increasingly high pressure to publish research findings – which results in an ever-increasing number of journal publications, workshop contributions, and conference proceedings [1, 3]. In consequence, the reviewing workload for established scientists becomes a significant fraction in the work portfolio, in most cases without rendering measurable benefits in terms of compensation or scientific reputation. It can be expected that if the status quo of the academic reputation primarily being based on the H-Index persists – which is very likely – the peer review concept as implemented will soon reach its capacity limits.

In this paper we argue for complementing the state-of-the-art review process that is implemented across all scientific disciplines with a new concept that 1) reduces the review workload for individual reviewers; 2) rewards review efforts with academic credit; 3) improves the sustainability of new software-, algorithm- and data analysis techniques; and 4) ensures full reproducibility and traceability of scientific computing contributions.

2 The challenges in the peer-review process for scientific computing

Particularly the growing field of scientific computing – the melting pot where domain scientists, mathematicians, data scientists, and machine learning enthusiasts team up to address scientific problems with sophisticated algorithms and compute power – experiences an explosion of scientific publications [2]. These publications typically present new strategies, implementations, parallelization efforts, machine learning frameworks, and data analysis algorithms for scientific applications ranging from social community analysis over earthquake prediction, DNA sequencing, tomography image analysis, individual medicine research, weather prediction, climate models, earth mantle simulations to wind turbine and oral airflow simulations. Even if assuming that all these contributions are worth reading, the pure amount of scientific papers and the associated review workload becomes a challenge to the scientific computing community. We refer to the unbearable review workload for established scientists as the **First Challenge** in the peer review process for scientific computing.

* Corresponding author: e-mail hartwig.anzt@kit.edu, phone +49 721 608 22756

[1] Henry Oldenburg, 1619–1677

The **Second Challenge** is that the review process itself does not or only indirectly contribute to the scientific reputation of the reviewer. Acknowledging that they may benefit in terms of learning about new ideas, the reviewing scientists are, especially if the contribution is not perfectly aligned with their own research, usually inclined to perform only a time-saving light review, and instead invest more effort into their own research. Only indirectly, if a thorough review catches the attention of a – typically more established – editor, a reviewer can get visibility. More recently, Publons[2] tries to reward the effort by providing review certificates, but this concept experiences sluggish acceptance as focusing on the review volume alone may even promote shallow reviews.

Complementary to the light review process, and in response to the academic pressure on increasing the paper count, scientists are also inclined to base publications on experimental code- and algorithm frameworks that enable the quick production of research findings instead of building sustainable simulation frameworks that can then be adopted by peers. While this approach is sure optimizing the benefit for the individual researchers, it is detrimental to the scientific community: Significant resources have to be invested to re-implement the proposed frameworks, and largely adopted community packages that are used in operational mode, e.g. for weather prediction, climate modeling, earthquake prediction, astrophysics applications, are often running behind in providing the cutting edge algorithms, simulation frameworks, and data analysis workflows. Obviously, this situation is unsatisfying and highly ineffective for the community as a whole. Part of the problem is the lacking acceptance of the research software engineer as academic entity, and that sustainable software development and code contributions do not reward the academic credit of scientific publications. We consider the resulting co-existence of fragile experimental simulation frameworks created with the unique goal of a single scientific publication on the one side and the sustainable software ecosystems adopted by the community and deployed on the largest computing facilities worldwide on the other side as the **Third Challenge** in the peer review process for scientific computing.

Closely related to that is the difficulty of reproducing research findings presented in scientific papers. With simulation frameworks remaining closed-source, the reviewers have no possibility to reproduce artifacts or check the correctness of the results presented. We consider the lacking traceability and reproducibility as the **Fourth Challenge** in the peer review process for scientific computing. While experimental result reproducibility turns out to be a weakness of the scientific peer review in general, it is in particular frustrating in the area of scientific computing where the reproduction of results typically does not require access to special laboratory equipment, large-scale research facilities, or proprietary technology, but only access to the algorithm implementations and the simulation software framework. In response to the situation, an increasing number of conferences and journals in the field are starting *reproducibility initiatives* where the authors of scientific contributions grant access to the developed algorithms, simulation codes, and data analysis frameworks [4, 5]. However, these initiatives are only slowly adopted, and even increase the workload for the reviewers that in this case not only assess the scientific content but also the technical realization.

One particularity of the field of scientific computing is that proceedings of top-tier conferences are considered at least as important as journal papers [**?**, 6], and the community using conferences rather than journals as the main publication venue [8]. One major reason for this is the nature of software development and the growth of compute technology needs a fast, but complete, review- and publication process. While fundamentally-different novel approaches may be able to tolerate a lengthy review process, incremental changes adopting novel hardware features need to be propagated quickly to keep pace with the 6–12-months development cycle of major hardware manufacturers [9].

Finally, the scientific computing community is heavily based on community interaction and community software efforts. In many cases, a new feature developed for a specific field can render similar benefits in a totally different field – without the developers even having considered this aspect. Given the agility and mobility of software, scientists reading a publication are not only interested in the ideas and findings presented in a paper, but even more in the tools enabling the results and the discussions on the design choices. Hence, in order to provide for a comprehensive documentation of a contribution, it is natural and useful in scientific computing to present scientific reviews and technical reviews side-by-side.

3 Professional software development

For identifying alternative review concepts for scientific computing, we follow ideas realized in the professional software development cycle as it is implemented by large software companies like Google, Apple, or Facebook, but also by established academic software efforts like Trilinos [10], FENICS [11] or ICON [12]. There, the software project is developed in a collaborative effort using a versioning system such as Git [13] that allows to orchestrate the collaborative development, synchronize contributions, and log the development history. In particular, these versioning systems archive every single contribution along with secondary information such as the contributor, the timestamp, and others. This allows at any time in the future to track who is responsible and creditable for a certain feature, change, or improvement. While this obviously also allows to track down the introduction of software bugs or controversial features, a significant part of these defects typically do not even make it into the software stack as every new contribution has to pass a rigorous review process imposed by the community: All new software contributions are coming as *pull requests* to the software ecosystem that need approval by reviewers to be integrated into the main stack. The reviewers are usually core developers of a software project, but technically anyone interested and

[2] https://publons.com

having expertise in the field can comment on a software contribution. This review concept is attractive for several reasons: 1) the reviewers choose themselves to review a pull request, which implies that they are likely experts in the particular field or interested in the feature; 2) listing the clear names of the reviewers gives them credit for their review effort, but also avoids bashing of the authors; 3) the review is public and archived by the versioning system, which implies anyone can at any time in the future revisit a review discussion; 4) a review process is not bound by time or feedback cycles, but every software contribution can be iterated until all technical flaws and discussions are resolved; 5) this review process becomes more valuable as not only reviewers but also potential users can participate in discussions on the functionality, use-cases, and performance aspects.

4 Implementing a new Peer Review Concept

Currently, the academic reputation of a researcher is primarily based on the number of scientific publications, the number of citations, and the H-Index combining the two metrics. We acknowledge that the status quo of the academic credit system is unlikely to change in nearby future. We also acknowledge that the peer review concept as implemented has proven to work – despite its flaws regarding reproducibility and the more recent rise of predatory journals [14]. However, given the challenges listed in 2, we are convinced that changes to the peer reviewing concept are necessary to maintain the quality-, reproducibility-, traceability-, and productivity-levels expected from the scientific community. But instead of radically changing the ecosystem, we propose to complement the well-established traditional review concept with a new concept that exploits the features and possibilities of collaborative software development and the related public software review process.

The idea is to allow peer-reviewed software patches contributing novel technology or features as conference contributions that are afterwards included as full paper in the post-conference proceedings. The anticipated workflow is that a scientist contributing a novel algorithm, new software feature, or simulation workflow marks this contributions in the pull request as a conference- or workshop submission. The reviewing software developers perform the usual code review, however keeping in mind the originality of the work. Either during the software review process or after the new feature is accepted to the software main stack, the author submits the patch as a conference contribution, linking to the pull request in the versioning system. The conference program committee may ask an external person for blind-reviewing of the submission. This independent reviewer can heavily benefit from the versioning system keeping track of the contribution and all reviewer comments: As the code reviewers are likely core developers that are very familiar with the software and/or experts in the specific field, the blind reviewer can leverage their feedback in the overall assessment. Ultimately, the blind reviewer's job is primarily to judge on whether the contribution is worth being presented at the conference and included in the post-conference proceedings. For a comprehensive overview about the functionality, performance, and quality of the contribution, we expect that the software patch comes with a detailed feature description, a user guide, potentially also a performance assessment, simulation results, and a survey how this contribution fits into the larger picture referencing related and orthogonal work. A good example of how a well-documented software patch submitted as conference contribution may look like is given in Anzt et al. [15]. The code reviewers' comments all being public, and likely improving the quality of the contribution, naturally request to acknowledge the names of the reviewers in the conference presentation and the paper proceedings. Presenting contributions to community packages at scientific conferences immediately offers the opportunity to structure the talk in a tutorial-like style such that attendees not only learn about the new feature, but also learn how to use it in their applications, data analyses, or simulation codes.

Complementing the traditional peer review concept in scientific computing with the new workflow comes with several benefits for the submitting authors, the reviewers, and the scientific community as a whole. For the authors of scientific contributions, submitting a software patch as conference proceeding

1. reduces the effort of publishing scientific papers, as submitting well-documented software patches rewards academic credit like writing a paper;
2. ensures all authors gets full credit for their work as the versioning system logs the author of every single line of code – removing tiring discussions about the author order in scientific papers;
3. allows to benefit from community comments on correctness, scope, performance, and functionality.

The reviewers, on the other hand, benefit in terms of

1. the code reviewers being able to iterate a code contribution until all discussions are resolved;
2. the code reviewers receiving credit for their work as their comments are archived by the versioning system and their names are acknowledged in the post-conference proceedings;
3. having full reproducibility and traceability of the contribution;
4. the undisclosed independent reviewers having a radically reduced review effort as they can access and leverage the code review of the community- and software experts;

Finally, the community as a whole benefits from this peer review concept as

1. it motivates high quality software contributions featuring detailed description and documentation;
2. the community can immediately access the novel technology by employing the code contributed to the open source software project;
3. users can access review discussions about functionality, scope, or artifacts;
4. users can immediately identify the main author to inquire technical details;
5. domain scientists can directly interact with software developers presenting the new features, results, or workflows at conferences.

Including the contributions in the post-conference proceedings ensures the authors are rewarded with the same academic credit as for scientific papers going through the traditional peer review process. We however argue that the novel workflow leveraging the code review concept established in the software development community not only helps to mitigate the pressure on the state-of-the-art peer review process that is indubitably hitting capacity limits, but also increases the efficiency and productivity of the scientific computing community. In particular, we are convinced that implementing this alternative peer review process will inevitably augment the quality, reproducibility, and sustainability of scientific contributions and community software efforts.

5 Summary

In this position paper, we propose to complement the traditional peer review process that has proven to be successful for centuries with an orthogonal approach where a blind reviewer can leverage the disclosed comments of a community software review archived by versioning systems. This enables researchers to contribute novel algorithms, features, or simulation workflows to community packages while still receiving the academic credit of a conference- or journal publication. While the primary goal of the alternative peer review process is to mitigate pressure on the review system and increase quality, reproducibility, and sustainability of scientific computing contributions, we hope that accepting high-quality software patches as conference contribution is another step in the direction of entrenching scientific software development as an academic field, and moving the academic evaluation system from traditional metrics (like the H-Index) towards a more community-advancing academic credit system.

Acknowledgements This work was supported by the “Impuls und Vernetzungsfond” of the Helmholtz Association under grant VH-NG-1241.

References

[1] UNESCO science report: towards 2030, ser. UNESCO Reference Works Series. UNESCO, 2015. [Online] https://en.unesco.org/unesco_science_report.
[2] P.P. Larsen and M. von Ins: The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index. Scientometrics. 2010;84(3):575–603.
[3] Svein Kyvik and Dag W. Aksnes (2015) Explaining the increase in publication productivity among academic staff: a generational perspective. Studies in Higher Education, 40:8, 1438-1453
[4] Association for Computing Machinery: Artifact Review and Badging, Reviewed April 2018. [Online] https://www.acm.org/publications/policies/artifact-review-badging.
[5] Supercomputing Conference: Reproducibility Initiative. [Online] https://sc19.supercomputing.org/submit/reproducibility-initiative/.
[6] M. Franceschet: The role of conference publications in computer science: a bibliometric view. Communications of the ACM, volume 53, no. 12, 129-132, 2010. [Online] https://users.dimi.uniud.it/~massimo.franceschet/publications/cacm10.pdf.
[7] J. Grudin: Conferences, Community, and Technology: Avoiding a Crisis. iConf, 2010. [Online] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/iConf2010.pdf.
[8] L. Fortnow: Time for Computer Science to Grow Up. Communications of the ACM, volume 52, no 8, 33-35, 2009. [Online] https://people.cs.uchicago.edu/~fortnow/papers/growup.pdf.
[9] Intel Corp: Manufacturing Process Technology – The Tick-Tock Model Through the Years. [Online] https://www.intel.com/content/www/us/en/silicon-innovations/intel-tick-tock-model-general.html.
[10] The Trilinos Project. [Online] https://trilinos.github.io/.
[11] The Fenics Project. [Online] https://fenicsproject.org/.
[12] ICON :: Icosahedral Nonhydrostatic Weather and Climate Model. [Online] https://code.mpimet.mpg.de/projects/iconpublic.
[13] The Git project. [Online] https://git-scm.com/.
[14] S. Ibba, F. E. Pani, J. G. Stockton, G. Barabino, M. Marchesi and D. Tigano: Incidence of predatory journals in computer science literature. Library Review, vol. 66, no 6/7, 505-522, 2017. [Online] https://doi.org/10.1108/LR-12-2016-0108.
[15] H. Anzt and G. Flegar: Are we doing the right thing? - A Critical Analysis of the Academic HPC Community. 20th International Workshop on Parallel and Distributed Computing (PDSEC), 2019. [Online] http://bit.ly/AreWeDoingTheRightThing.