Scientific Software Development Collaboratory

Jeffrey Carver, University of Alabama* Nasir Eisty, Boise State University†

1 Introduction

Software development is a complex socio-technical activity that incorporates tools, languages, development environments, processes, measurement, and people, all of which must be studied and understood. When considering scientific software development, one could add aspects that may be less relevant to traditional software development, including complex problem domains, advanced computing platforms, and unique developer cultures. The scientific software development ecosystem is complex now, and it is only becoming more complex with the convergence of HPC, AI, and HPDA combined into workflows to enable new scientific discovery. Studying the development of software in a traditional setting is challenging enough, however with the addition of these factors related to scientific software development, the challenge increases.

One of the key challenges to scientifically studying scientific software development (or software development of any kind) is a collaborative environment in which software engineering researchers can connect with software development teams to identify and study real problems. Scientific software development teams, like any software development team, have aspects of their software development process, tools, or teams that are sub-optimal. These deficiencies, if left unaddressed, could result in low-quality software and ultimately incorrect scientific results. However, in many cases, these teams do not have the time, resources, or knowledge to address these deficiencies. Conversely, many software engineering researchers do not have the knowledge of these real problems or access to the developers to study those problems.

Therefore we can describe the two-fold challenge as:

- **Scientific software teams, who have needs, do not have an easy way to connect with software engineering researchers who would like to study those needs.**
- **Software engineering researchers who would like to support scientific software developers do not have an easy way to identify the real problems or study those problems in context.**

To mitigate the challenges above, we propose a virtual laboratory that can act as a collaboration hub of scientific software developers, software engineering researchers, and software developers in general. The following section describes the proposed laboratory.

2 Proposal for Overall Scope of Effort

A *virtual laboratory* that connects scientific software development teams with software engineering researchers for mutual benefit would help address the challenges above. Such a laboratory would provide a venue where scientific software development teams can describe their software development challenges and their willingness to participate in research to develop and evaluate solutions to those challenges. Examples of the types of challenges include: understanding code, finding reviewers, using proper metrics and benchmarks, using proper testing techniques and frameworks, CI issues, and long-term sustainability management. In addition, software engineering researchers interested in scientific software development can monitor these challenges to identify those they are interested in addressing. For example, software engineering researchers might produce training materials, develop new tools and techniques specifically for scientific software, formalize processes, or create guidelines. After this "matchmaking", the researcher and the scientific software development team can develop a plan. The formalization of this type of laboratory will allow for the following benefits:

*carver@cs.ua.edu
†nasireisty@boisestate.edu

- Multiple software engineering researchers can propose solutions to the same challenge and use research studies to identify the strengths and weaknesses of each approach.
- Different scientific software teams who face the same, or similar, challenges will be able to identify each other and work together with researchers on solutions.
- The laboratory will provide the opportunity to systematize the challenges, the solutions, and the results from research studies in a manner that can scientifically advance our understanding of scientific software development to develop new metrics, tools, and development processes, and training materials.
- Institutionalizing this laboratory will allow Ph.D. students and postdocs to directly affiliate with this laboratory and dive deep into this growing research area.

We model this proposal on the successful *Software Engineering Laboratory*, a collaboration between NASA, Computer Sciences Corporation, and the University of Maryland (in which Carver participated as a Ph.D. student). This 25-year collaboration focused on understanding software development principles in context, the factors that affect the development of software, and the types of problems encountered all within the context of scientific research software focused on space flight. In the DOE software context, our proposed laboratory builds on the experiences of this laboratory, as described in a paper reflecting on its history [2].

This type of virtual laboratory will also provide the infrastructure necessary to help answer many important questions. Some examples of the types of questions that such a laboratory will help answer include, but are not limited to:

- What software engineering methods either uniquely apply or uniquely do not apply to the development of scientific high-performance software?
- How can researchers study changes in culture of scientific high-performance software development and use that affect the approaches used to develop that software?
- What unique changes in the development process or toolchain for scientific high-performance software improve productivity, accuracy, trust, reliability, and/or sustainability?
- How can research results motivate changes in the scientific high-performance computing community to improve practices for software development and use?
- How do various static and dynamic analysis methods affect the development of scientific high-performance software?

3 Conclusion

The scientific software ecosystem for HPC is always in flux, but is now approaching a new height of diversity and complexity with the convergence of HPC+AI+HPDA. A single best practice cannot provide the solutions needed, rather a long-term institutionalized relationship is needed to chart this complex landscape. There is a growing interest in how the use of appropriate software engineering practices can support the development and sustainability of software for scientific and high-performance computing. Therefore, the time is ripe for the creation of this laboratory to enable the community to better understand which practices are most important in this domain. As more developers of scientific high-performance software realize the benefits of software engineering and seek out the most appropriate methods, the benefits provided by this proposed laboratory will be readily recognized by many. The authors have already begun addressing some of the questions around this important topics [1, 3, 4, 5, 6, 7], but there is a need for additional work.

References

[1] Elvira-Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Jeffrey C. Carver. Software engineering practices for scientific software development: A systematic mapping study. *Journal of Systems and Software*, 172:110848, 2021.

[2] Victor R. Basili, Frank E. Mcgarry, Rose Pajerski, and Marvin V. Zelkowitz. Lessons learned from 25 years of process improvement: The rise and fall of the nasa software engineering laboratory. In *of the NASA Software Engineering Laboratory, IEEE Computer Society and ACM International Conf. on Soft. Eng., Orlando FL*, pages 69–79. ACM Press, 2002.

[3] Nasir .U. Eisty and Jeffrey C. Carver. Developers perception of peer code review in research software development. *Empirical Software Engineering*, 27, 2022.

[4] Nasir U. Eisty, George K. Thiruvathukal, and Jeffrey C. Carver. A survey of software metric use in research software development. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 212–222, 2018.

[5] Nasir U. Eisty, George K. Thiruvathukal, and Jeffrey C. Carver. Use of software process in research software development: A survey. In *Proceedings of the Evaluation and Assessment on Software Engineering*, EASE '19, page 276–282, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Dustin Heaton and Jeffrey C. Carver. Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, 67:207–219, 2015.

[7] A. Nanthaamornphong and J. C. Carver. Test-driven development in hpc science: A case study. *Computing in Science & Engineering*, 20(5):98–113, Sep./Oct. 2018.