Software Engineering for Sustainable Scientific Software

A Postion Paper for the 2019 Collegeville Workshop on Sustainable Scientific Software (CW3S19)*

Jeffrey Carver
Department of Computer Science
University of Alabama
carver@cs.ua.edu

Use of *appropriate* software engineering (SE) practices is critical for improving the quality, reliability, maintainability, and sustainability of scientific software. This position paper describes my work in two fields directly relevant to this workshop.

Software Engineering for Science The discipline of SE focuses on developing (and evaluating) techniques and tools to assist developers in efficiently building high-quality, maintainable, and sustainable software. Contrary to what many in the scientific community believe or have experienced in their interactions with software engineers, SE does not always refer a monolithic, process-heavy software development approach. Many of the more modern SE approaches are lightweight and agile, characteristics that make them more applicable to scientific software development.

One of the biggest barriers to more widespread application of SE practices in scientific software development is a mismatch (or perceived mismatch) between traditional SE and the specific needs of scientific software developers. There are many reasons for this perceived mismatch including: (1) the lack of a complete understanding of the specific needs of the scientific domain in question, (2) the lack of tailored SE practices, and (3) the lack of good examples of effectively using SE practices in scientific software. Regardless of the reason, a scientific developer needs to be convinced of the value of any SE practice before he or she will allocate any precious time to employing that practice.

To encourage the use of appropriate SE practices, we need close collaborations between software engineers and developers of scientific software. These collaborations should focus on identifying specific needs and developing (or tailoring) practices to address those needs. To be successful, these collaborations require effort and input from both parties. Scientific software developers must provide information about the specific problems and constraints in their environment and must be open to trying out new SE practices. Software engineers must be willing to listen to scientific developers and develop or tailor practices to fit their needs and constraints.

Based on my extensive experience in Software Engineering for Science, including my own case studies [5, 8, 9, 12, 15, 14] and surveys [6, 7, 13], as well as a workshop series I co-orgainze (http://www.SE4Science.org/workshops), I believe that appropriate, lightweight, modern software engineering practices can be of great value to the scientific software community.

Increasing the level of participation and quality in Free/Libre, Open Source Software (FLOSS) Building a vibrant FLOSS community around a software package requires focused effort. In my previous work, I have studied how the social networks formed through activities like peer code review affect community building and peer impression formation for virtual teams [1,

*Adapted from a position paper submitted to the first workshop on Geospatial Software: Connecting Big Data With Geospatial Discovery And Innovation (https://gsi.cigi.illinois.edu/workshop/position-papers/)

2, 3]. I have also studied the barriers that prevent One-Time Contributors (i.e. those who have successfully contributed 1 code patch) from continuing with the project [4, 10, 11]. As one of the topics of interest for this workshop is the development and use of open source software, a better understanding of the social dynamics and of the methods for attracting more participants are highly relevant.

References

[1] A. Bosu, J. Carver, R. Guadagno, B. Bassett, D. McCallum, and L. Hochstein. Peer impressions in open source organizations: A survey. *Journal of Systems and Software*, 94:4 – 15, 2014.

[2] A. Bosu and J. C. Carver. *How Do Social Interaction Networks Influence Peer Impressions Formation? A Case Study*, pages 31–40. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[3] A. Bosu and J. C. Carver. Impact of developer reputation on code review outcomes in oss projects: An empirical investigation. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, pages 33:1–33:10, 2014.

[4] A. Bosu, J. C. Carver, C. Bird, J. Orbeck, and C. Chockley. Process aspects and social dynamics of contemporary code review: Insights from open source development and industrial practice at microsoft. *IEEE Transactions on Software Engineering*, 43(1):56–75, Jan 2017.

[5] J. Carver. Development of a mesh generation code with a graphical front-end: A case study. *Organizational and End User Computing*, 24, 2014.

[6] J. Carver, D. Heaton, L. Hochstein, and R. Bartlett. Self-perceptions about software engineering: A survey of scientists and engineers. *Computing in Science Engineering*, 15(1):7–11, Jan 2013.

[7] N. U. Eisty, G. K. Thiruvathukal, and J. 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, Oct 2018.

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

[9] D. W. Heaton. *Software Engineering for Enabling Scientific Software Development.* PhD thesis, University of Alabama, 2015.

[10] A. Lee and J. C. Carver. Are one-time contributors different? a comparison to core and periphery developers in floss repositories. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10, Nov 2017.

[11] A. Lee, J. C. Carver, and A. Bosu. Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: A survey. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 187–197, May 2017.

[12] A. Nanthaamornphong, J. Carver, K. Morris, H. Michelsen, and D. Rouson. Building cliime via test-driven development: A case study. *Computing in Science Engineering*, 16(3):36–46, May 2014.

[13] A. Nanthaamornphong and J. C. Carver. Test-driven development in scientific software: a survey. *Software Quality Journal*, pages 1–30, 2015.

[14] 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.

[15] A. Nanthaamornphong, J. C. Carver, K. Morris, and S. Filippone. Extracting uml class diagrams from object-oriented fortran: Foruml. *Scientific Programming*, 2015:15, 2015.