SAND2021-14621C

Short-term results versus long-term impact: Applying software development best practices to scientific software

Kelley Ruehl, Sandia National Laboratories, kmruehl@sandia.gov
Kate Klise, Sandia National Laboratories, kaklise@sandia.gov

Challenge

Research projects often require development of custom scientific software, or research software, to address novel research questions and advance state-of-the-art methods. Scientific software is often written for a very specific purpose. It is commonly used to prototype ideas and generate results quickly. Generally, the software is not intended for use outside of the main research group. Consequently, software development best practices, like writing documentation and testing the software, are often not prioritized. This focus on quick results can be driven by many factors, such as an immediate deadline, limited budget, and team dynamics. This can be great for short-term success, but there are often long-term costs.

This can present several challenges, including verification, validation, reproducibility, and software maintenance [1,2,3]. With a growing number of software options available for research, it is important that software not be trusted blindly. Both scientific and open source software packages should be vetted before use. In the early stages of a project, the research team must decide how the custom software will be developed and maintained and which software dependencies will be included. Too often, software development starts without fully considering how these software architecture decisions will affect short-term results and long-term impact.

What starts as a software prototype can become the basis of a multi-year research project and eventually evolve into an open source software package. Research teams can quickly grow and evolve, and software can easily grow from a thousand to millions of lines of code. When speed is prioritized over best practices, the result is often unreadable code, undocumented features, possible errors in the software, and single-developer knowledge of features. Many times, these challenges are only identified once the project is over and key developers have moved on. These factors make long-term development and support of application-based scientific software very challenging.

Opportunity

Similar to the impact FAIR (Findable, Accessible, Interoperable and Reusable) data principles have had on data management, software development best practices can drastically change the long-term impact of a project [4]. Software development best practices include version control, documentation, quality assurance, source code management, and a software development plan. These aspects of software development can seem burdensome, especially for scientific software developed for an immediate research need. However, writing clean code, documenting features, generating tests, and completing rigorous review by team members enables long-term success. In addition to producing a more cohesive software product, they will enable higher quality research and ensure reproducibility by the research team and by external members of the scientific community. This initial investment can save time and budget in the long-term, resources that can then be used to advance future research.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

The increased use of open source software within research projects has improved adoption of software development best practices. This applies to both scientific software that is intended for open source distribution and to software that is never intended to be publicly distributed. Adoption of best practices can be attributed to software development tools like GitHub and GitLab that support version control, continuous integration testing, and documentation. Well established style guides, like PEP 8, also contribute to more uniformity in the software development approach. Integrating best practices into scientific software development requires allocating resources to make critical software design decisions and maintaining this focus throughout the project lifecycle. While it can be challenging to reserve funding for this purpose, the long-term benefits can far outweigh the initial cost. Once best practices become part of the research culture, they can be leveraged to benefit multiple projects, and used to establish a research community with a common software development style.

While standards for highly regarded open source software packages are not always appropriate or realistic for scientific software development projects, there is much to be learned from those examples. Open source software often prioritizes best practices over speed and efficiency. This relates to both the software development timeline, and the way the software is written. Developers might choose to write clean and readable code over a computationally faster implementation because the highest priority is the software's usability. This disproportionate focus on usability is part of what makes open source software unique. However, open source software faces its own challenges, like funding. While open source software projects have the potential for broader application and long-term impact, they have limited appeal in the short-term. Funding agencies are less inclined to commit to the long-term investment required to support an open source software package's continued maintenance and support. Application-based scientific software driven by a desire for quick results can be a more palatable investment for funding agencies. In short, decisions that are well suited for open source software may not be appropriate for scientific software, but a balance must be established to ensure high quality research.

Timeliness/Maturity

Source control, testing platforms, and style guides have been available for a long time, but the current abundance and transparency of open source software provides the scientific research community countless high-quality examples to guide their development. Adopting well established norms and integrating with highly regarded open source software packages enables research teams to focus on custom features for their scientific software. Leveraging existing styles and packages can also free time and budget to create documentation and tests. In the long run, this adds credibility to the research by ensuring reproducibility and building a user community.

References

[1] Goble, C., 2014. Better software, better research. *IEEE Internet Computing*, *18*(5), pp.4-8.
[2] Joppa, L.N., McInerny, G., Harper, R., Salido, L., Takeda, K., O'Hara, K., Gavaghan, D. and Emmott, S., 2013. Troubling trends in scientific software use. *Science*, *340*(6134), pp.814-815.
[3] Stodden, V., McNutt, M., Bailey, D.H., Deelman, E., Gil, Y., Hanson, B., Heroux, M.A., Ioannidis, J.P. and Taufer, M., 2016. Enhancing reproducibility for computational methods. *Science*, *354*(6317), pp.1240-1241.
[4] Wilkinson, M., Dumontier, M., Aalbersberg, I. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3,** 160018 (2016). https://doi.org/10.1038/sdata.2016.18

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.