Incorporating software engineering practices with scientific computing to improve software sustainability

Authors: Sara Gosline (PNNL), Christopher Oehmen (PNNL), Asher Mancinelli (PNNL), Slaven Peles (ORNL)

Challenges: Scientific software is an important tool for scientific research, but in practice it is often treated as a scaffold -- meant to be dismantled after project completion. With a specific focus on a scientific question, the drive to sustain the project software through engagement of a broader community is lost. There exist many software engineering practices, such as documentation, unit testing, and code review, that can prevent this through building a software development infrastructure that encourages community development and code reuse.

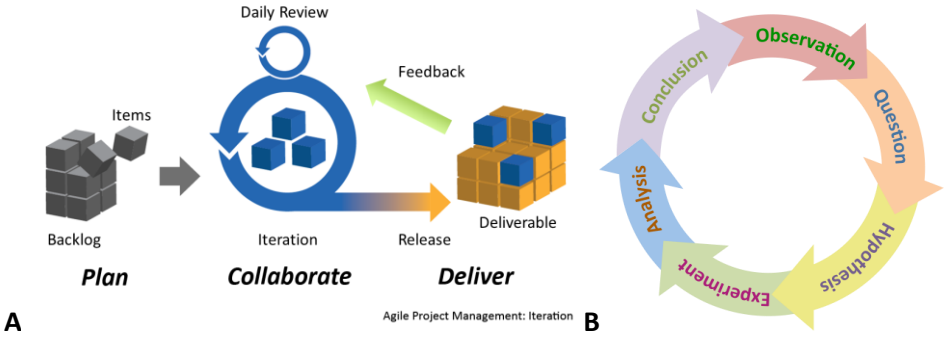

Figure 1. Mismatch between software development practices (A) and scientific method (B) showcase different priorities across fields.

Software engineering practices have been developed to encourage sound software development within broad, diverse communities. These practices ensure that tools can be used and reused over time by enabling many to contribute. For example, Agile software development[1], depicted in **Figure 1A** emphasizes iterative development (center) in which a team is creating, documenting and testing software in an iterative fashion before deployment. This approach produces final software that is able to be repurposed for outside projects or passed from one team to another.

Scientific computing generally follows the scientific method[2] (**Figure 1B)**. In this example, computing algorithms are developed and used under the 'experiment' and 'analysis' portion of the cycle with little emphasis on improving the software for future use. Since scientific computing methods often fail to adhere to the high standards set by the agile, or other, software engineering framework, projects that represent years of software development expertise fail to be repurposed across the longevity of the project or other projects.

Opportunities: This gap in software practices represents an opportunity for improved infrastructure requirements in future calls for software-heavy proposals. While many such standards exist for scientific data[3] and file formats, standards for software development infrastructure and code reuse are less enforced. Recent work in the life sciences has started to explicitly document these standards for reproducibility[4], but they are not broadly enforced.

Recent successes from the ExaSGD project from Exascale Computing family[5] have showcased how using infrastructure standards[6] can engage the broader research community and enable software sustainability. Additionally, we have been working to enforce these standards more broadly across PNNL to enable code sharing across scientific domains. For example, sustainable software practices in the National Security Directorate have enabled the use of a high performance hyper graph analytics library[7] to be used in the Earth and Biological Science Directorate to evaluate viral infection[8]. The ability to apply high performance compute algorithms across domains is currently underutilized due to the investment required, as these types of code reuse require solid software stewardship and sustainable practices to be successful across domains. In both of these examples, infrastructure standards and community building tools borrowed from software engineering can improve scientific outcomes.

Recommendations: The explosion of tools to enable automation of large-scale computing in the software engineering field can only really be captured with appropriate infrastructure standards to manage the use of these tools. Piecemeal implementation of cloud standards, coupled with poor regulation from individual agencies, will prevent sustainable projects from emerging from the scientific domain. As such we propose specific recommendations to develop and implement software engineering standards as part of any scientific computing proposal.

1-Develop infrastructure standards and requirements for proposal calls: At the core of any scientific computing endeavor there needs to be specific guidelines and metrics by which project implementation can be determined to meet sustainable software metrics. These can be carried out by borrowing practices from software engineering (**Table 1**)

Software engineering practice	Metrics to enforce practice	Use in software engineering	Benefit in scientific computing
Code review	Code linting software	Ensure code readability	Show value of code across interdisciplinary team
Documentation	Documentation checks	Enable others to use code and build on top of it	Facilitate publishing of algorithmic method, and ensuring code reuse
Unit testing	Test coverage	Make sure changes do not break build	Enable collaborative research and analysis
Continuous integration	Build success	Deploy code	Generate scientific figures/results
Portability	Tests across diverse platforms	Cloud deployment, heterogeneous hardware deployments	Code reuse, reproducible results
Scalability	Tests across scales	Grow with project needs	Ability to analyze more data

Table 1: Software engineering practices to be used for sustainable scientific computing

2-Investment in standards incentivization and coordination to support community involvement: In addition to incorporating these standards into future proposals, government agencies must delegate funds to ensure adequate enforcement of these standards across proposals. As such, grant recipients must be held to the standards they proposed. Funding should be made directly available to projects for standard enforcement (i.e. sustainable practices built) and revoked if specific metrics are not met. For example, each grant proposal can come with a shared "standard coordination center" that serves to ensure that all software and data standards are met across all funded institutions.

3-Proposals for code repurposing to further develop community engagement: One of the larger promises of computational science is the ability to solve a problem in one domain, such as the use of hypergraphs we described above[7,8]. This promise is rarely realized due to general failures in sustainable software development delineated above. We propose that explicitly incentivizing software projects that can be employed across domains will further motivate improved scientific practices and ultimately lead to more sustainable software.

Conclusion:

Here we describe the disconnect between the promise of scientific computing as it is implemented to date and the tools that are available to make software more sustainable. Specifically borrowing infrastructure standards and community coding practices from the software engineering community, we can make software more sustainable. We present numerous opportunities in this domain and specific recommendations for future funding ventures.

References:

1. Image: https://commons.wikimedia.org/wiki/File:Agile_Project_Management_by_Planbox.png
2. Image: https://commons.wikimedia.org/wiki/File:The_Scientific_Method_(simple).png
3. Heil, B.J., Hoffman, M.M., Markowetz, F. *et al.* Reproducibility standards for machine learning in the life sciences. *Nat Methods* **18,** 1132–1135 (2021). https://doi.org/10.1038/s41592-021-01256-7
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
5. https://www.exascaleproject.org/research-project/exasgd/
6. https://e4s-project.github.io/
7. Joslyn C.A. et al. (2020) Hypergraph Analytics of Domain Name System Relationships. In: Kamiński B., Prałat P., Szufel P. (eds) Algorithms and Models for the Web Graph. WAW 2020. Lecture Notes in Computer Science, vol 12091. Springer, Cham. https://doi.org/10.1007/978-3-030-48478-1_1
8. Feng, S., Heath, E., Jefferson, B. *et al.* Hypergraph models of biological networks to identify genes critical to pathogenic viral response. *BMC Bioinformatics* **22,** 287 (2021). https://doi.org/10.1186/s12859-021-04197-2