Moving Forward Together: How a Software Engineering Department Can Impact Developer Productivity in a Research Organization

A whitepaper for the 2020 Collegeville Workshop on Scientific Software, focusing on Developer Productivity.

Jim Willenbring, Reed Milewicz; *Sandia National Laboratories*

Introduction

In the past decade, the requirement to productionize software formerly considered research software has become more common. The requirement of producing production quality software has increased the demand for Research Software Engineers, scientific programmers, DevOps staff and other software-focused professionals within research organizations. Ten years ago, there were only a few such staff members in the Center for Computing Research at Sandia National Laboratories. That number slowly grew until last year when a department reorganization created the Software Engineering and Research Department. This reorganization gathered up the software-focused staff members from around the center and put them in a single department focused not only on enabling software technologies, but also on software engineering research tailored specifically to scientific software.

Background

While our department is relatively new (formed in 2019), it has been several years in the making. A few years prior to its creation, our center assembled the Software Engineering, Maintenance, and Support (SEMS) project. SEMS was a project staffed by software practitioners split across several departments, similar in purpose to the current department except that there was no research component. SEMS was exceptionally successful in meeting a demand for useful tools, training, and support, and this spurred motivation among leadership to create a department around the work of SEMS. While SEMS was considered successful, and the project still continues with a modified scope under a different name, there are several advantages to having an official department.

Communication and collaboration: Software-focused staff have the opportunity to interact more closely with like-minded people, opening up more opportunities for collaboration. This was also the case to some extent with SEMS, but not all software-focused staff were involved, and trying to involve a large number of people at a small fraction of their time was often inefficient.

Viable career paths: In a research organization, it can be challenging to establish career paths for software-focused staff without an intentional plan. Computational science and mathematics organizations tend to value software only insofar as software advances the science, which makes it hard to build a rewarding career based around software itself. Creating an official department has made software engineering a first-order concern for the parent organization, and carves out a space for people in software-related roles to advance their careers.

Strategic focus and prioritization: A new department also added a manager with a software focus. Prior to the reorganization, several managers recognized the importance of software-focused efforts, and some hiring slots were even used to hire staff to fill these needs. However, no manager's primary focus was software, which limited strategic planning and the ability to prioritize in this area.

Hiring and retention: When trying to hire the best software professionals available, it is much better to describe a department with a specific mission, staffed with people who do work similar to the work the potential employee would be doing, with a focus that is strongly supported by the Center and its management,

than to try to hire someone into a department with a mission only tangentially related to their skillset and staffed with people who have completely different backgrounds.

Research Focus As mentioned above, SEMS did not have a research focus, but as it evolved into a department within a research organization it acquired this focus. Software engineering research (also known as software science) is the study of software systems and their development, operation, and maintenance. Research and practice in software engineering have always been deeply interconnected, and it was both necessary and naturally expected that we would bring the two together under the same roof. In that sense, our department not only provides software engineering as a service to scientific software teams but also seeks to improve upon the practice of software engineering itself.

In Sandia's nomenclature, the Software Engineering and Research department is also known as department 1424, a number which describes its purpose and place in the laboratories. For the benefit of the uninitiated, this means the department exists within the Advanced Science & Technology Division (1000), within that the Center for Computing Research (1400), and finally within that the Extreme-scale Computing Group (1420). That is to say that our team supports the scientific computing mission of Sandia through R&D activities in the area of software engineering, especially for software meant to run on highly scalable and performant computing architectures. As the name "Software Engineering *and* Research" might imply, ours is a hybrid organization composed of both software engineering practitioners and researchers. This is common in the software industry, where the line between the engineer doing research and the researcher doing engineering are often deliberately blurred.

What We Offer

Infrastructure: Our department firmly believes that robust, scalable, and sustainable infrastructure for software projects is vital to the scientific computing mission of our center. Our infrastructure work carries forward the original objectives of SEMS, such as developing and deploying tools and resources. Case in point, we maintain Jenkins-based build/test farms to provide CI/CD capabilities to software projects across multiple Sandia networks. We also develop and support a dependency management system that creates a common environment for developers on architectures ranging from personal laptops to cutting-edge testbed servers. In terms of off-the-shelf tools, our department supports an Atlassian tool suite including Jira and Confluence. We also create tailored software solutions to meet project needs, such as an auto-testing framework for pull requests (Autotester), a repository traffic monitoring framework to measure community engagement and impact (Repometer), custom plugins for Jenkins to measure the performance of subroutines (Watchr), and a Jenkins pipeline scripting library (SPiFI).

Training and Outreach: We aim to cultivate a community of practice around software engineering in our center. We regularly offer courses on software engineering tools and techniques, and we collect feedback from participants to provide course material that is relevant to their needs. Meanwhile, within our department, we have a long-running round table series on best practices in software engineering where team members come together to deliberate and discuss the processes and principles that lead to high-quality software.

Research: Our department is building a program of fundamental and applied research in software engineering. Our research is focused on the science and engineering of research software in the domains of interest to Sandia and the ongoing delivery of production software to our stakeholders. In our research activities, we seek to derive generalizable insights that benefit not just Sandia but the scientific community at large. We strive to publish at major software engineering and computational science conferences to engage with the broader community, and we bring that community to Sandia through a campaign of invited talks known as the Sustainable Software Engineering Seminar Series (SSESS). Additionally, our research mission directly serves the Center 1400 community through consultation with the aim of providing research-as-a-service (RaaS).

Software Development: A key element of our work is in teaming with application and algorithm researchers to provide embedded development, maintenance, and support. In fact, the majority of our department's funding comes through direct work with individual projects. One important type of software development services provided by our department is scientific programming. This type of development requires scientific

domain-specific knowledge and skills. For example, familiarity with core algorithms and experience using MPI, OpenMP, and/or CUDA. The department also provides software development services for core software engineering needs, such as testing, build systems, and other areas that are not scientific-domain specific. Many of these contributions are outside of the training and interest of the center's scientific research staff. In these ways, we are able to apply our software engineering expertise to our center's software projects, raising the bar on quality and rigor in scientific software development.

Our Impact on Developer Productivity

In all of our diverse efforts, our goal is to empower domain scientists and mathematicians to work more efficiently and effectively. No matter our roles, our team hopes to be an incubator and an accelerator for impactful ideas in software engineering. The ways in which our team impacts developer productivity are numerous, but some can be subtle. Consider the case of a typical scientist-developer in our center:

- She is able to track her team's work through Jira and Confluence which we offer and maintain.
- Her ongoing research into next-generation algorithms is directly supported by an embedded developer from our department.
- To manage her repository, she uses a git workflow that we set up and trained her on.
- Whenever she submits a pull request, our tools detect this and launch build and test jobs on our server farm to ensure her code performs as intended.
- Her software relies upon a complex system of library dependencies, all of which we seamlessly provision through our environment module system.
- Recently she reached out to our team for consultation on software design, and we are in the process of compiling peer-reviewed literature and industry best practices to inform her design-related decisions.
- She sends an email to a colleague about an interesting talk she attended by a leading expert in software engineering. We invited him.
- When she encounters a software-related problem, she submits a ticket to the CCR Help Center. We operate that.
- She has a project website to advertise her research papers and document her code. We built that.

Taken altogether, she spends less time worrying about the minutiae of software development and maintenance, and more time more time delivering on cutting-edge science. Prior to the creation of our department, one of the research staff members in the Center commented "It's crazy how much of my career has become devoted to making and maintaining software since I got here. I've had to learn a lot on the job. Sometimes I feel like I'm going at this alone". Addressing that concern is precisely why our department exists today.

Challenges

One disadvantage to forming a department was weakening the organizational connections between the software-focused staff and the developers of the scientific software projects they contribute to. This can be mitigated through intentional communication, but to some degree there is simply a trade-off associated with the types of staff that are organizationally near the members of the Software Engineering and Research department.

A significant challenge for the department has been that the demand for software engineers has been far greater than the available supply. This puts the department manager in a position where she must prioritize requests, all of which are important to the respective requester. In an attempt to better meet this demand, the department has nearly doubled in size since its inception. This has helped, but also creates the challenge of training new staff and integrating them into the collaborative culture the department is trying to foster.

The high demand for support and services from the department has in some ways limited the staff time available for research efforts. This has not come as a surprise. We have been working to integrate research

into the culture of the department, encouraging those who do not have a primary interest in research to contribute their experiences to the community and participate in research studies.

As with many new organizational appoaches, during the first year plus of the existence of the new department we have collectively needed to better define our role in the Center and beyond with respect to departments and teams that provide similar or complimentary services. Getting to know one another has also been important, but has taken time. These activities are important because they can help to create a healthy work environment, and also so that we can better leverage our collective knowledge, and are comfortable doing so. These activites can be challenging enough, but have been perhps moreso because the department was formed with staff from across the Center and has brought on many new faces in a short time. Going forward, we plan to leverage these activities for higher satisfaction and productivity, and to help us more clearly carve out the unique role of our department in the Center for Computing Research and the broader computational science community.

Conclusion

In this white paper, we summarized the ongoing work of the Software Engineering and Research Department at Sandia National Laboratories. We hold that cross-cutting, software-focused teams like ours are an effective way of bringing software engineering expertise into scientific software organizations.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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.

SAND2020-6685 C