Day 2 Discussion: Collegeville 2021
Technical Approaches to Improved Software Teams

Overview: The content your discussion group creates in this document will be synthesized in a blog posting for https://bssw.io

Instructions:

1. Pick one person in your discussion group to create a new copy of this Google Doc
2. Make a copy of this template in a new Google Doc (the person from step 1)
3. Share the edit link to the document in step 2 with others (copy and paste into Zoom chat)
4. Co-edit the document: Can have one lead writer with others modifying, or another approach
5. Send the document to Mike Heroux at the end of the session by email (mheroux@csbsju.edu)

Add group member names for anyone who wants attribution in the blog post:

1. Reed Milewicz, Sandia National Laboratories, git: rmmilewi
2. Todd Munson, Argonne National Laboratory, git: tmunson
3. David Moulton, Los Alamos National Laboratory, git: jd-moulton
4. Sam Yates, Swiss National Supercomputing Centre, git: halfflat
5. Nur Fadel, Swiss National Supercomputing Centre

Discuss as a group the most promising technical approaches you see as opportunities for scientific software teams. Summarize discussion in outline form.

- Recognize that the social/”soft” sciences can be technical
    - Todd: A lot of the issues we have with scientific software teams can be recognized and addressed by including the soft sciences. A lot of our problems are communication-related. Social scientists like sociologists and anthropologists are very good at teasing out these kinds of information.
    - Sam: For one of our teams, we had a student anthropologist on our team for three years, and the things she brought up were problems that were very real, and her insights are very helpful (providing people are willing to listen)
    - Todd: And on the cognitive and behavioral side, there’s a lot to offer there.
    - Sam: When you’re talking about this, do you mean people who are specifically skilled in these areas, or having domain scientists reading up on the literature and applying it themselves?
    - Reed: As a software engineering researcher, I directly apply social science research and collaborate with social scientists. Developing solutions people will actually use requires understanding them and their culture. That’s so critical to success.
    - Todd: an industrial psychologist in house at the lab may also be helpful.
- Curated relevant literature, made available and accessible to teams.

    - Evidence based approach to guidance: Reed's example of Requirements Gathering is a great example.
- Use of cool tools like https://www.mural.co (ideation)
- Use of collaborative tools that can support peer programming when not in person
- Use of a "Business Model Canvas" adopted for DOE / Software projects: https://en.wikipedia.org/wiki/Business_Model_Canvas
    - Todd has the template for DOE-related projects
- Adoption of tools like GitHub (version control), Kanban boards (team organization/planning), and Gitter (communication)
    - David M: We've found that Kanban boards are things that you have to get everyone to buy into, which can be challenging
    - Sam: We have a small team, and we aren't always consistent in using it, but we pick it back up whenever we slack off.
    - Sam: More recently, we've been moving from Slack to Gitter for the sake of our German colleagues who really want to use open, non-commercial tools wherever possible.
- Workflow automation tools and techniques from code review to continuous integration to code coverage analysis
    - Nur: To be clear, in academia not everyone is onboard yet, there are plenty of researchers who are still using tarballs and emails instead of version control.
    - Reluctance can be driven by code shame — making code public can highlight its quality. But this in turn can be a tool: make it public, so that shame drives code improvement.
- Test case prioritization and selection to reduce inefficiencies in test cycles, giving teams faster and more useful feedback on their code changes.
- Feature branch management:
    - How long can they be open without drifting from main (even if merging frequently - it may start to break).
    - What happens when you need Feature A and Feature B and those are different feature branches.
- Code Review - as part of PRs, but how deeply do teams review, how well do people accept criticism?
- Containerization for CI/CD
- Adoption of better refactoring techniques and tools
- Better time management strategies

About 20 prior to the end of the session, around 1:40 pm CDT, try to reach consensus on 3 - 5 high-level technical approaches your team identified

- **Approach 1**: The technical and social aspects of software development go hand-in-hand. Understanding those social aspects is critical to the adoption of new technologies. As a community, we should be working closer with social scientists to understand how we do our work and how we could do it better.
- **Approach 2**: The adoption of modern software engineering tools and techniques will empower scientific software teams to do their best work. This includes tried-and-true

solutions like version control tools and planning and organization tools (e.g. Kanban), but also emerging technologies like test case prioritization and containerization. In the same way we look outside our immediate field of study to learn new things, we can look to software engineering practice in industry for inspiration.

- **Approach 3**: Evidence-based best practices (EBP) should be used to guide decision-making in critical areas. EBP here means integrating current best evidence from research with practical experience and human values to improve decision-making related to software development and maintenance. While research can’t replace our professional expertise, we can leverage peer-reviewed evidence to inform how we do our work.