

Day 3 Discussion: Collegeville 2021  
Cultural 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](mailto:mheroux@csbsju.edu))

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

Name, affiliation, and GitHub ID (if available), as you would like it listed in the blog post

1. Sarah Osborn, Lawrence Livermore National Laboratory (osborn9)
2. Anshu Dubey, Argonne National Laboratory (adubey64)
3. Ben Cowan (benc303) - Formerly with Tech-X Corporation, moving to Pilot AI
4. Stan Tomov, University of Tennessee, Innovative computing Laboratory (stomov)
5. Sarah Knepper, Intel Corporation, @sknepper
6. Charles Ferenbaugh, Los Alamos National Laboratory
7. Vadim Dyadechko, ExxonMobil, @vdyadechko

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

- Recognition
  - Monetary (at least for industry)
  - Publicly praising people
  - Recommend people (especially younger ones) for opportunities (like serving on program committees) that bring broader recognition - indirect means
  - Spotlight awards at meetings - verbal call out. Technical or non-technical contributions.
  - Can make an impact in yearly evaluation.
- Perception that software is a technician's work (valued less), or less intellectually demanding than what a domain scientist does
  - Citations; should SW citations be counted, or to the extent that other research citations are counted
  - Of course these citations/library usage should be counted; how to overcome this prejudice
  - Citations aren't a good metric for software quality

- Citations are one way to try to close the gap of knowing who uses your software and for what
- But can't also have citations; e.g., proprietary or classified code/projects
- Example to get around this: user group meetings (look at attendance)
- What can we do for things that are external to a team? Example: Mars rover recently landed; at the launch, GitHub put a badge on every bit of software that played a part (like SciPy) - <https://github.blog/2021-04-19-open-source-goes-to-mars/>
- Important for management to understand contributions even if there aren't citations; may be easier in industry than academia
- Original paper may be cited, but current development team may be significantly different
- Encouraged not to cite just the software but also the version
- LAPACK lists new contributors for a release
- Outreach - press releases when software is used for some exciting new thing
- Software development inherently a team work. Small projects have little value; to create something useful requires collaborating with others. Scientific work can be more individual. Culture of individualism can erode the process.
  - Peers as competitors
  - No sense of "we're all in this boat together!"
  - Metrics used to evaluate performance were strictly individual
  - No informal team building
  - Culture of sharing the credit for work done is very important for the success of the project!!
  - Standard practice in milestone reviews: at beginning or end, include slide with all team members who worked on it (even though there may be just one presenter) - especially important for public presentations
  - "Hidden effort" - not seen as main business. Facilitators who lay the groundwork to make it easier for developers. Sys admins or tech writers may feel like second-tier citizens.
- What are some cultural things to help SW team work better internally?
  - Team lunches can be effective
- Individuals working on multiple teams
  - Can cost more mentally due to context switching
  - Benefit: may need a few people to cover needed skill sets - brings collaboration
  - Cross-fertilization between teams
  - Good time scale for context switching? Every hour too often; every few months too infrequent.
- Meetings!!
  - Good meeting practices: in or out concept. If you're in the meeting, you're fully involved and responsible; otherwise, you're out of the meeting.
  - Bad meeting practices: calling or cancelling a meeting with insufficient notice (e.g., 20 minutes beforehand).
  - Good: everyone bring agenda items BEFORE meeting

- Good: provide slides before meeting starts (not just abstract)
- Good: collect feedback at end of meeting if it was helpful, if scope was proper (conversational and whether the right people were attending)
- Common thread of a good project meetings (e.g., twice-weekly syncs, planning every 3-4 weeks, couple ad hoc meetings for various issues): meetings were never just for spreading information; they were interactive in some way
- Frequency for recurring meetings is important. Daily sync up meetings may not be helpful for tasks that take a while to complete (not much day-to-day change).
- Cameras on or off for virtual meetings
- Bring laptops to in-person meetings (or not)
- Meeting anti-pattern: meeting to bring manager up-to-speed on what everyone is doing
- A day with 2 intense (interactive, talk a lot) meetings may wipe out development work for that day
- Long-term considerations should be taken into account when making plans (introducing a new feature, etc.). People have tendency to want to minimize their own effort, but need to ensure the implementation will be extensible, portable, supportable.
- A culture of giving credit is imperative!!
  - Can be used to get people on your side who may otherwise resist your idea.

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

1. **Make sure team's work is recognized.** There needs to be a good way of judging the overall success of a software project. Citations can be a part of it, but you also need to account for the fact that you don't know all of the users. Collecting information from a variety of sources is useful, like emails/private communications or user group meetings. Ensure people higher up in institutions as well as funding agencies use these metrics. Make sure managers share that recognition with team members. "Wall of fame?"
2. **Make sure meetings are used effectively.**
3. **Have a culture of "all jobs are important."** Communicators, technical writers, sysadmins, software engineers, scientists, everyone. Full- and part-time. "Boring" work (documentation) as much as "exciting" (speeding up a function).