The effect of HEP software culture and practices on diversity, inclusion, and retention

Deborah Bard[1] Matt Bellis[2]

[1]National Energy Research Scientific Computing Center (NERSC), LBNL, CA
[2]Siena College, Loudonville, NY

August 31, 2020

Abstract

The Snowmass 2021 effort provides an opportunity to critically examine the data analysis practices and culture in our community. While there is little doubt that past efforts have led to some truly amazing scientific discoveries, we suggest that practices could be improved, particularly in analysis software, that leads to more productive and efficient workflows while also creating a more inclusive environment. Snowmass offers us an opportunity for us to re-think how we approach our software and how we design the training and on-boarding for new colleagues.

Introduction

Big Science often involves experiments that take many years, even decades, to plan and run. These large, complex experiments result in large, complex software stacks that have often been designed by a committee of hundreds, if not thousands. Code is usually written by a core group of experts, who have a wealth of knowledge and expertise. This knowledge filters down to new collaborators whose nearest-neighbor experts (usually senior students and post-docs in their own group) do not always share this knowledge. The experiments have a long timeline and so their software stacks will span shifts and improvements in software and hardware and will incorporate those changes in "real time". These improvements save time and increase experimental sensitivity. However, these improvements are often rolled out in a way that make it even more difficult for students to adapt to.

Because documentation is often the last thing to be written and because most grad students and post-docs who write the documentation are usually not trained as teachers or mentors, the documentation does not always serve the needs of the boots-on-the ground analysts who can be distributed all over the world. This results in graduate students spending the first 2-3 years of their PhD simply learning an evolving and changing framework just to access the data and MC in their experiment. This is the norm in most areas of HEP, and so PIs and others tend to accept this environment as simply an immutable fact of nature/the system.

This environment can be particularly frustrating for women, students of color, and other under-represented groups. The barrier to entry for a new researcher trying to understand and contribute to such a complex stack is often viewed as a rite of passage, something that everyone has to go through. Newcomers are given complex code to copy and edit, pointed to confusing or non-existant documentation (often in the form of commented code), and told to "go ask someone" how to proceed. This is a system that exacerbates imposter syndrome - the feeling that you are not as good as everyone around you. If a young researcher doesn't understand what their code is doing, how can they consider themselves to be a credible scientist? The way we structure software development in our collaborations favors personality types that are comfortable asking for help, often at risk of being patronised in very public forums (e.g. Slack, hypernews) or those who have the confidence to barrel through without really understanding what they're doing.

While there can be a real sense of achievement in understanding even a corner of the software used in a large collaboration like LSST, DUNE, or any of the LHC experiments , too often the experience leaves a researcher demoralised and disillusioned about the process of science. This stress can be mitigated by an involved and supportive mentor or PI, but not every group has that. Promising scientists are leaving the field as a direct result of the software culture, and anecdotal evidence suggests that this is more likely to happen to members of underrepresented groups in HEP. We are personally aware of students, mostly women, who left the field after their PhD (or even before completion, opting to take a Masters instead) in large part because of their experience with software in their experiments. And even students who "push through" express frustration at their experiences, especially when they are told by more senior members of their collaboration that "this is just how things are".

Ways Forward

We plan on formally collecting these anecdotes from individuals who have left the field as well as those remain to share with the community. We will also develop a set of suggestions and guidelines for future software UI/UX and culture that can contribute to more efficient analyses and a more supportive and inclusive environment for analysts.