When Not to Use Agile in Scientific Software Development

Anshu Dubey

June 29, 2020

Over the last few years it has almost become an orthodoxy among better informed practitioners of scientific software productivity that *Agile* is the way to go. There are some very good reasons for this, including the fact that the models that dominated enterprise software development before development and adoption of agile methods had far too many process overheads to be even remotely feasible in science environments. In most science domains software development process resembled dysfunctional agile in the sense that the development was driven by results needed for the next science paper. So it was all quick development and availability to the "client" which were often the developers themselves. The flip side was the amount of technical debt accrued through such completely unplanned development which led to untimely demise of many software projects. So, ironically enough, the introduction of agile methods in these domains was doing the exact opposite of what it was doing to the enterprise software. That is it was slowing down the next "delivery" of software to incorporate some process where none existed before.

Over the course of two major version revisions in a large and long lived scientific software I have come to the conclusion that slowing down with agile helped some aspects of software development, but did not go far enough for others. This intuition comes from the realization that many multiphysics application software projects rely on a very robust framework backbone to enable flexibility and extensibility of the software. As with anything that needs to be robust and has to last a long time, a great deal of thought needs to go into the design. The exploration of the design space itself can happen with processes that have quick turn around in some instances, but that is not always possible.

In both instances of major architectural revisions of FLASH, the software that I have been associated with for a long time, The design phase took 6+ months of discussions, white boarding, prototyping, refining ideas, and even discarding ideas. Both times it took more than two years from start before the first simplest application instance could be configured. This was possible because a quickly stitched together version existed that was used to continue science research in the interim. Use of agile would have been entirely appropriate for the first functional version. However, once a tool was available to do science, it was equally important to redesign and refactor. The longevity of the code and its ability to pivot to several science domains would not have been possible without this upfront long investment in requirements gathering and understanding the design space quite thoroughly. Agile would not have served us well, it would have got in the way. However, once we have the backbone in place, agile methods are the best way to move forward with building or transitioning physics capabilities.