Development of VTK-m During ECP

Date: 25 September 2024
Presented by: Ken Moreland (Oak Ridge National Laboratory)

(The slides are available via the link in the page's sidebar.)

Q: (Alfred Tang) most research teams have small number of workers compared to Apple, Google and Microsoft that can invest a lot of manpower and resources on logistics and management. Given the limited manpower and resources that a research team has, how does it strike a balance between research and management?

A: Start small. There are lots of little things you can do that take little effort but have a big impact on software development. Pick, choose, and adapt features from agile formal methods that provide the most benefit to you. That won't get you the full benefit that an enormous software company can provide, but you can get close enough. Here are some simple things that we did that are straightforward to implement.

1. Use git (or a competitor like mercurial if you want to be a contrarian) for revision control and get everyone using a branching workflow (where all development starts as a topic branch, gets implemented to a working state, and then merged back into the main branch. It's a very small effort on developer's part, but helps isolate developers from each other's intermediate code.
2. Host the central repository from a web-based system like GitHub, GitLab, or Bitbucket. These give you lots of incredibly valuable tools for free: pull/merge request system, CI resources, issue tracker, wiki, web page hosting, and more. If security is a concern, many of the DOE labs have internal enterprise editions of one of these systems to ensure proper access controls.
3. Set up a testing suite in your build system. If you are using CMake for your build system, that makes it straightforward to add and run tests. Establish to developers that all features should be touched by at least one test. I won't lie, adding tests can be onerous. Often I find myself spending more time making the test than the feature it is supposed to be testing. But in my experience the drudgery of writing tests is nothing compared to the headache of finding bugs.
4. Establish a process for merging changes to the main branch. Protect the main branch in the central repository (to make it impossible to push commits directly there). Have all developers submit changes as a pull/merge request. I suggest having a second developer review all code before it is accepted (made easy by the web-based pull request). This gives you a lot of the benefit of pair

programming without halving your effective developers. Also, make running the full suite of tests for each change mandatory before merging it in.

5. Have regular informal meetings. It's helpful for everyone to get a short status update on what is happening. It's also a helpful tool for the PI to manage the progress of the project.
6. Establish continuous integration (CI) testing as part of the pull request process. This is probably the hardest thing to do in the list of items here, but there are lots of resources for setting this up with GitHub or GitLab. It is extremely helpful to have the testing integrated into the code review process.

Q: Is there a way to combine the Doxygen developer manual with the User guide (i.e., embed the User guide in the code)? Do you think it may help with maintaining it?

A: Doxygen allows you to establish pages of documentation that are not directly associated with a particular class/function, so you could use that to establish freeform documentation like tutorials and user's guide. That said, organization of these pages is not Doxygen's strong suite. For any software project that required more than a few pages of explanation, I would be reluctant to try to cram a large amount of document format into a tool that is not designed for that.

Q: When precisely did ECP end? The website just says 2024 and that it "has concluded" already.

A: It ended at the end of 2024: 31 December 2024