2019 Workshop: Software Engineering for Heliophysics

Long title
Software Engineering for Heliophysics
Conveners
Michael Hirsch
Matthew Zettergren
Guy Grubbs
Description

This session targets those developing software and doing analysis, modeling or data collection every day (students and early-mid career) as well as more senior scientists interested in the best trends and techniques from industry as applied to heliophysics. We will discuss intermediate to advanced geospace software engineering at a level **accessible and useful to all**.

Scope includes **coding languages commonly used in heliophysics**, including: C++, Fortran, Julia, Python, R

Use cases we address include:

- scripting languages to analyze large data sets quickly
- model developers reduce the time spent tutoring new users in building / modifying / using the model
- reduce debugging effort by adding automated self-tests
- ensure code will be usable on most current computing platforms and easily adaptable to future systems

We intend that most participants will be able to apply industry best-practices to their own work the same day or week.

Agenda

- **Intro**: What aspects of software engineering are most important to heliophysics community (pdf)(Hirsch 13:30-13:35)

- **Version Control**: working effectively in large and small teams with Git and GitHub (pdf) (A. Ridley / Hirsch 13:35-13:45)
- **Continuous Test & Integration**: catching and tracking problems before you know they exist (pdf) (A. Ridley / Hirsch 13:45-14:00)
- **Software / Data connections**: Making heliophysics data accessible via lightweight standard: HAPI (pdf) (R. Weigel, 14:00-14:15)
- **OpenMPI physics model**: Fortran 2018 design patterns and Python integration (pdf) (Zettergren 14:15-14:30)
- **Mobile / Web app development**: crowd-sourced science - Aurorasaurus (pdf) (E. MacDonald 14:30-14:45)
- **Scientific software workflow**: The science software stack--going from ideation to simulation to publication (M. Young 14:45-15:05) [PYTHON] [Full Stack] [Public Data Archiving] (pdfs)
- **Parallel Python**: introductory Python examples (asyncio, ProcessPool, ThreadPool) (pdf)(Hirsch 15:05-15:15)
- **Room discussion**: what topics did we miss? What should we do more of? Should we have side sessions this year? (15:15-15:30)

Justification

ST #5: Fuse the Knowledge Base across Disciplines

1. Good software engineering practices expedite reliable, repeatable science results and encourage diverse outside collaborator participation. Repeatable, traceable science analyses are better trusted and solidify community and stakeholder confidence in published results.
2. Software sharing / collaboration sites like GitHub have matured and are widely used by the heliophysics community. We address minor changes in practice that reduce time to science closure.
3. Progress is readily measured by semi-automated metrics such as:
    - quantity and diversity (institutional, geographical, participant) of code contributions and issues opened
    - an increase in the use of continuous integration facilities such as Travis-CI
    - increased use of public data sharing such as Zenodo

ST #6: Manage, Mine, Manipulate Geoscience Data and Models

1. Increased scientific computing efficiencies are essential for computer-aided discovery of the growing petabytes of data collected. Extracting value from the

decades of diverse existing data sources can be greatly aided by more efficient software engineering practices
2. Effective use of software toolchains can be a significant force multiplier in avoiding mistakes and repeated or manual work.
3. Progress might be measured by mining papers for citations / keywords used such as links to software repos used, which can themselves be mined for use of continuous integration tools, build system type and specific software libraries

Summary

Peak attendance about 35, with people in and out. Student percentage seems 20% or so. Wondered why higher student percentage didn't show up. At both this session and Python library session following, group generally seemed supportive of a third session in 2020 just for Python performance development.

It's highly useful to have slides usable offline. All of this session's talks did that.

That is, a possible 2020 CEDAR software session lineup including:

- general software engineering best practices (Git, build systems, cloud, docker, HPC)
- Python performance development (parallel, big data, PEP8, type hinting, setuptools-agnostic, Python 3 transition, Numba, Pycuda
- Python libraries (this has been going for a few years)
- hackathon for Python (specific list of issues to tackle)
- data science sessions (Sunday + two like 2019?)
- participatory data science tutorial (like 2019 InGeo)

View PDF