Challenges and Lessons Learned from 10 Years of OpenMC Development

Paul K. Romano
Computational Scientist, Argonne National Laboratory

ONCORE Meeting
June 23, 2022

Motivation

- OpenMC has been under continuous development for 11 years and has undergone substantial change
- Arguably the first successful open-source transport code in the nuclear engineering space
- The goal of this paper is to highlight **challenges** and **lessons learned** over the course of a decade for the benefit of others in the nuclear OSS community
- Specifically focused on *community* codes

Challenges

Takeaway

Most challenges stem from the fact that the incentive structures in the R&D establishment are not aligned with the needs of a growing, OSS community

Staffing

- Most developers are staff/postdocs/students at research institutions and universities
- Funding is geared toward R&D and science goals
- Maintenance and user support are usually not called out specifically in work scope — crucial for success and sustainability

User Interaction

- Significant time commitment is necessary to support user/developer community
- Challenges:
    - Intermittent nature of developer contributions
    - Constant onslaught of user questions / requests
- Opportunities:
    - Users help uncover bugs and improve code quality
    - Incentivizes better documentation habits
    - Positive interactions eventually yield competent users who themselves can contribute to the community

Organizational Ownership

- If a community code is successful, ownership is not centered at a single institution
- Any given institution has less incentive to invest in the code and advocate for the developer community if they don’t view it as “their” code
- Governance can potentially become more difficult — clear need to formalize governance structure of project

Metrics

- For OSS, some of the traditional metrics (number of registered users, e.g.) are not readily available
- Lack of metrics can make it difficult to show impact of the software, but there are ways:
  - Number of active users on user's forum
  - Number of GitHub clones
  - Number of contributors/contributions
  - GitHub stars
  - Citations to major papers*
- Need to be careful because it's easy to "game" most metrics

Software Challenges

- As software grows, it’s more difficult to make breaking changes
- Hyrum’s Law — any observable behavior of an API will be relied on by someone
- Technical debt can, and does, accumulate over time; it becomes increasingly expensive to pay down technical debt (and hard to justify to a sponsor)
- Maintaining performance can be difficult when new features are constantly added

Technical Expertise

- As OpenMC has grown, so has the minimum skill set needed to meaningfully engage
- **Skills:** C++, Python, CMake, git, MPI/OpenMP, testing, CI/CD
- As code matures, there are fewer opportunities for low-hanging fruit to be addressed by newcomers

Benefits and Lessons Learned

Takeaway

Despite the challenges, there are many crucial advantages that come with open source development and I truly believe the OSS model is here to stay for nuclear

Meritocracy

- Under our governance model, anyone—regardless of affiliation—can make a meaningful contribution if they are willing to invest time
- Giving others equity means they too are vested in the success of the project
- Community nature of code helps to engender collaborations across multiple institutions

Free Resources

- For an OSS project, you can easily secure free:
    - Repository hosting
    - CI/CD services
    - Web hosting
    - DOI assignment for code releases
    - Discussion forum
- All these resources improve user/developer experience

Return on investment

- For sponsors (particularly government), allowing code to be open sourced leads to a higher return on investment since code artifacts can be easily reused
- Responsible stewardship of taxpayer dollars

Building sustainability

- While it’s easy to open source a project, it can take years to build a true community
- User/developer growth happens slowly and to really reach sustainability, you need one or more people to be in it for the long haul
- Bus factor: How many people would need to get hit by a bus in order to effectively stop development?

Code leadership

- Although I have been the de facto / official lead developer of the code for its entire life, most of the substantial recent contributions in the code were not my work:
    - Depletion (Colin Josey, MIT)
    - Photon Transport (Amanda Lund, ANL)
    - Python API (Will Boyd, MIT)
    - CAD-based geometry (Patrick Shriwise, ANL)
- Of the above examples, three of them were unsolicited

Community building

- Although time spent interacting with users/developers might be seen as a burden, it pays off in unexpected ways that are hard to measure
    - A user who has a positive experience may convince their colleagues to consider using it
    - A developer who feels welcomed may make contributions that could expand the potential use cases and scope of the code
- Technical decisions and good software design are important, but building a sense of commuity is equally important

Conclusions

Conclusions

- OSS development comes with many challenges, some of which are unique to the nuclear community
- Many of these challenges are unresolved, particularly funding and alignment with R&D goals
- OpenMC has benefitted from constant development over a decade, allowing it to grow a substantial user/developer community
- Despite challenges, the open-source model has been hugely beneficial for OpenMC

Thank you!