# **CHALLENGES AND LESSONS LEARNED FROM 10 YEARS OF OPENMC DEVELOPMENT**

P.K. ROMANO1, S.M. HARPER2, P.C. SHRIWISE1, A.E. JOHNSON3, B. FORGET4

1Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL 60439, United States

2TerraPower, LLC, Bellevue, WA, United States

3Ultra Safe Nuclear Corporation - Technologies, Seattle, WA, United States

4Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, United States

Email contact of corresponding author: promano@anl.gov

OpenMC [1] was founded in 2011 primarily as a tool for methods research and development with a focus on high-performance computing. In its 10th year of development, the code has expanded and evolved greatly since its inception and now boasts a thriving community of users and developers. OpenMC represents the first attempt at a community-developed Monte Carlo (MC) code in stark contrast to the usual development model, whereby codes are controlled by a single institution and are under export control and licensing restrictions. As the first such open MC code, the development team has faced many challenges; this paper reflects on some of the challenges of the open source development model and lessons learned over the last decade.

**Challenges**

Many of the challenges of the open source development model stem from the fact that the established incentive structures are not well-aligned with the ever-growing costs of maintaining the code and supporting the userbase.

Most developers of OpenMC are either full-time employees at US research institutions or students at universities pursuing a doctoral degree, and their funding is geared specifically towards R&D and science goals. However, a significant amount of the OpenMC development work is instead focused on maintenance and user support. These activities are crucial for the success of the code, but they only indirectly support the funded research objectives. Thus, developers carry out much of this work in their spare time.

This funding issue is not unique to OpenMC, but it is more acute due to the lack of organizational ownership. Research institutions will often choose to invest in software maintenance and user support for strategic reasons. For example, they can better compete for funding by developing an in-house tool with unique capabilities. These institutions might also invest in user support in the hopes of developing strategic relationships that may lead to additional funding. However, no organization has sole stewardship of OpenMC, and thus no organization has a clear strategic motivation to fund the code, advocate on its behalf, and ensure its success.

Securing funding from a sponsor is also complicated by the fact that as an open source project, some of the traditional metrics used in the nuclear energy field for measuring code adoption are not readily available, such as the number of registered users, the number of downloads, and the number of institutions where the code is being used. However, there are a wide variety of ways that the impact of the code can be measured, including the number of active users on OpenMC’s discussion forum, “clones” (repository downloads from GitHub), contributors, contributions over time, GitHub stars, and citations to major papers describing OpenMC.

The time required for the unsupported work is significant, and the open source model brings some unique demands. This includes the challenge of incorporating sporadic, unexpected contributions from the community. Although contributions from newcomers are highly valued, surprise code changes nevertheless increase the workload on the core developers. Sometimes, these contributions conflict with planned changes or interfere with important parts of the code design. Advance planning of the changes would reduce the time required to review and merge the code, but it is difficult to execute this planning with such a decentralized community.

In addition to direct code contributions, a significant time commitment is required to handle communication and requests from users. As the code has grown, so has the number of inquiries and requests that come in from users. This is a double-edged sword. On one hand, continual user interactions help uncover bugs, improve code quality, and encourage better documentation habits (as it is easier to refer to documentation that has already explained the code interfaces well). On the other hand, responding to questions from users takes time away from developers toward adding new features and carrying out the R&D tasks that are their main responsibility. Over time, the positive user interactions have been observed to create a feedback loop, generating further interest in the code and also sometimes yielding very competent users who themselves are able to answer inquiries and make contributions to the code, thereby lessening some of the burden on the core development team.

Some of the challenges faced by OpenMC developers are common issues that are shared by most large software projects. As the number of users increases over time, breaking changes in the code that would disrupt users and cause confusion become increasingly consequential. While this might be seen as a negative, it does force developers to put more forethought into the design of user-facing APIs and file formats. As the code itself grows over time, so does the amount of technical debt, which can impede the pace of future development. Code refactoring that reduces technical debt but otherwise doesn’t increase functionality can be hard to justify in a research enterprise, where the main focus is on achieving R&D goals and advancing science. Along similar lines, maintaining performance in a large codebase that is constantly changing with new features being added can be quite challenging. This is especially relevant for OpenMC, where one of the focal points for R&D is application on advanced computing architectures and parallelization.

Another challenge resulting from the growth of OpenMC is the technical expertise necessary for newcomers to meaningfully engage in development. The codebase is a mix of Python and C++, thus requiring that developers be sufficiently fluent in those two languages (which have very different conventions and idiosyncrasies). Developers may also need knowledge of the CMake build system, parallelization with OpenMP and MPI, and continuous integration testing. In addition to the sheer technical skills necessary, the maturity level of the code has led to fewer opportunities to implement simple features that are missing from the code for new developers to address. Nevertheless, this hasn’t stopped the code from attracting a multitude of contributors from many institutions in recent years.

**Benefits and Lessons Learned**

Having had 10 years of experience building an open source code with a thriving community, we truly believe there are many crucial advantages that come with open source development. Our development team has also learned many lessons over the years that may be useful for others considering starting an open source software project or open sourcing an existing project.

One of the key benefits of the open source model is that it empowers anyone, regardless of institutional affiliation, to make meaningful contributions to a code if they are willing to invest their own time. Under OpenMC’s governance model, anyone who makes a non-trivial contribution is given permission to review and merge code change requests. Giving others equity in the project means that they too are vested in the success of the project. Despite the funding-related challenges associated with the lack of organizational ownership mentioned earlier, in our observation the community nature of the code helps to disassociate it from the natural competition that occurs between similar organizations (e.g., national laboratories in the US). Whereas a staff member at one laboratory would be unlikely to contribute to a code owned and controlled by a different laboratory, they may instead contribute to a community code where they have an equal opportunity to gain ownership.

In addition to easier collaboration, open source projects benefit from a wide variety of free resources. This includes repository hosting, continuous integration computing resources, web hosting, DOI assignment for code releases, web forums, and more. Together, all of these resources make for a seamless developer and user experience.

A perhaps lesser appreciated benefit of open source development is the higher return on government R&D investments. Nearly all of OpenMC’s funding was derived from US government programs funded by taxpayers. The code itself is one of the most valuable artifacts produced from these R&D programs, and allowing others to freely use and build upon the code maximizes the value of the original investment.

One of the most important lessons is that while it is very easy to open source a new project, it can take years to build enough momentum to have a sustainable community of developers and users. One good metric for considering whether a software project is sustainable is the so-called bus factor—the number of developers that would have to get hit by a bus to completely halt development. For OpenMC, the bus factor was 1, or at most 2, for the first few years of its development. At present, there are enough developers with broad knowledge of the codebase that even if one or two of the core developers were to cease all activity, the others would be able to keep the momentum going.

Another important lesson is that although the founder and project lead for the code, P. Romano, has been responsible for overall stewardship of the software project, many of the most important contributions to the code over its history have come from other developers (e.g., depletion [2], photon transport [3], the Python API, and CAD-based geometry[4]). This underscores another key benefit of the open source model; of the four examples given here, three of them were unsolicited contributions to the code.

Finally, although it takes a substantial investment of time to work with new developers and respond to user inquiries and requests, that investment can pay off in ways that are hard to measure. A user who has a positive experience working with the code may convince their colleagues to consider using it. A new developer who feels welcomed to the community may make a substantial contribution to the code that could expand potential use cases. While the technical decisions made by the development team are an important aspect of OpenMC’s success, the sense of community is an equally important aspect.

**Conclusions**

We have highlighted here many of the challenges, benefits, and lessons learned over the course of the development of OpenMC, one of the most visible open source projects used in nuclear energy modeling and simulation. We hope that sharing our perspective will help others understand the many benefits of open source software development while being realistic about the associated challenges.

**Acknowledgments**

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The submitted manuscript was also supported in part by the U.S. Department of Energy, Office of Science, under contract No. DE-AC02-06CH11357.

**References**

1. ROMANO, P.K., HORELIK, N.E., HERMAN, B.R., NELSON, A.G., FORGET, B., SMITH, K. OpenMC: A state-of-the-art Monte Carlo code for research and development. Ann. Nucl. Energy **82** (2015) 90–97.
2. ROMANO, P.K., JOSEY, C.J., JOHNSON, A.E., LIANG, J. Depletion capabilities in the OpenMC Monte Carlo particle transport code. Ann. Nucl. Energy **152** (2021) 107989.
3. LUND, A.L., ROMANO, P.K. Implementation and validation of photon transport in OpenMC. Technical report, Argonne National Laboratory, ANL/MCS-TM-381, 2018.
4. SHRIWISE, P. Xiaokang, Z. Davis, A. DAG-OpenMC: CAD-Based Geometry in OpenMC. American Nuclear Society Annual Meeting **122** (2020) 395-398