OAK RIDGE National Laboratory

Open Sourcing Your Software is Not a Sustainability Plan – Until it Is!

David E. Bernholdt <bernholdtde@ornl.gov>

Oak Ridge National Laboratory

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

(Loosely) based on the whitepaper:
https://collegeville.github.io/CW3S19/WorkshopResources/WhitePapers/bernholdt-open-source-v02.pdf

License, Citation, and Acknowledgments

License

- Requested citation: David E. Bernholdt, Open Sourcing Your Software is Not a Sustainability Plan – Until it Is!, invited talk, Collegeville Workshop on Sustainable Scientific Software (CW3S19), Collegeville, MN, July 2019.

Acknowledgements

- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and 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.

- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Open Source and Sustainability

Q: How do you plan to sustain your software?

A: We'll make it open source

- I believe this response is fairly common…
- Among naïve software development teams

What's the Problem with OSS as a Sustainability Strategy?

- Is it rare that project reaps significant benefits from the broader community
- The signal-to-noise is low: how will your package get noticed?
- Many more potential users than developers
- Many incentives/excuses to rewrite instead of reusing

Addressing the Challenges Takes Work

- Getting noticed takes work
  - Quality technical publications,
  - Capability and generality to attract interest
  - Quality of code
  - "Trustworthy"
- Supporting users and contributors takes work
  - Most contributors start as users; but many users never contribute
  - Users are needy, and can be a distraction
  - But how you support your users will influence their interest in contributing
  - Contributors will also need support, especially initially
- Producing code that others want to reuse takes work
  - Ensure code is high quality
  - Lower barrier to understanding
  - Strong tendency, many excuses. Sometimes you won't overcome them

Users and Contributors

- Users are likely to file bug reports and "suggest" enhancements
  - It may actually take a fair amount of work to get an *actionable* bug report out of them
  - Ideas for enhancements may be couched much less politely
  - This is a "cost" of opening up a code
- How many have the skills, experience, and willingness to contribute?
- Intended contributions may not be aligned with the core developers' plans
  - How will these contributions be supported and maintained?
- Contributors may not be familiar with preferred development practices

Technical Approaches to Facilitate Crossing the Chasm to a Truly Sustainable Open Source Software Project

- Critically assess the "market" to set your expectations
  - What features and capabilities will you offer?
  - Who needs them?
  - How many prospective users will have the skills, experience, and willingness to contribute?
- Make a thoughtful choice of license
  - What does your community expect?
  - How will commercial entities use and contribute?
  - Patent considerations?
  - Contributor license agreement?

Technical Approaches (2)

- Provide quality documentation
  - Technical publications that are clearly associated with your code
  - User-level documentation
  - Developer-level documentation
  - Development roadmap
  - Contributor guidelines, development practices, style guides
  - On-boarding support
- Ensure code quality and understandability
  - Set and enforce a coding style
  - Practice code review
  - Tests (existence and application)
  - Are there other "software quality metrics" worth targeting?

Technical Approaches (3)

- Provide tools to facilitate communication
  - Between users and developers
  - Among developers (including “core” and “outside” contributors)
  - Issue tracking
  - Mailing lists, chat tools
  - Web site

Technical Approaches: Necessary, but Not Sufficient

- Creating a welcoming, supportive, and responsive environment takes more than just technical tools
  - A strong social aspect, not independent of technical choices
  - Consider GCC vs LLVM
- Code needs to be supported and maintained over time
  - How does this happen? For "core" capabilities? For non-core contributions?
  - Companies can make a business case for investing in open source
  - Many research sponsors still have trouble with this…
  - So the research software community has trouble with this

Summary

- It is *not* common for open source software projects to become sustainable through their broader community
- It does happen sometimes…
- But not by accident
  - It requires conscious effort in both the technical and social realms
- It could be facilitated and promoted by the policies and $ of research sponsors
  - But today, not so much