Sustainable Cyberinfrastructure Software Through Open Governance

Marlon Pierce[1], Suresh Marru[1], Chris Mattmann[2,3]

[1] University Information Technology Services, Indiana University, Bloomington IN 47408
[2] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109
[3] Department of Computer Science, University of Southern California, Los Angeles, CA 90089

Introduction: The Need for Open Governance

Sustainable software depends on communities who are invested in the software's success. These communities need rules that guide their interactions, that encourage participation, that guide discussions, and that lead to resolutions and decisions -- we refer to these rules as a community's *governance*. *Open governance* provides well-defined mechanisms executed through open communications that allow stakeholders from diverse and even competing backgrounds to interact in neutral forums in a collaborative manner encouraging growth and transforming passive users into active stakeholders. Our position is that cyberinfrastructure software sustainability benefits from these open governance methods.

Software sustainability has many aspects. As a first principle, the software must obviously fulfill a need for a community of users. In addition, some portion of the project's members must acquire funding to develop and improve the software, to fix bugs, and to maintain compatibility with new computing platforms, with compilers, with run time environments, etc. The software needs to be well-engineered, documented, and supported so that it can survive the departure of early developers and add new developers. All these activities are performed by the project *stakeholders.* Greater diversity of stakeholders increases the resiliency and sustainability of the project in the face of uncertain funding and developer turnover, but this diversity also increases the likelihood of conflicts. Projects must therefore have well defined governance to balance the need for attracting new committers with the dangers of community splintering.

While the practices above bear fruit for sustainability of many types of software (even closed source), in particular they are especially true for *Open source software*. In open source software projects competitors agree that it is to everyone's advantage to work on a common code base, joining forces as stakeholders in a common community. Famous examples include the Apache HTTPD server, the Linux operating system, many programming languages such as Perl, Ruby, and Python, and (more recently) domain specific environments, tools, and platforms such as Apache Hadoop, Apache Cassandra, OpenStack, etc. Many of these efforts have commercial (or else explicitly anti-commercial) origins, but open source principles also apply to academic software supporting scientific research. It is our contention that much of these open source principles have been inadequately applied or misapplied by the research community.

Cyberinfrastructure (CI) software is developed to support e-Science research. CI is software typically operated as network accessible services, that supports large scale distributed computing and scientific data management. It is primarily funded by government agencies.

"Open" CI software is often taken to mean various things. The software may have an open source or free license, but how this is enforced or ensured internally within the project is not clear. The software may be openly available on the Web through online version repositories such as GitHub, SourceForge, Google Code, etc, where it can be viewed, branched, and so on, but how one contributes back to the trunk of this open source code (and get credit) may not be clearly defined. Technologies may help, but accepting patches and granting full access to the trunk are ultimately human decisions. The code may even implement open, community standards, but the value of these standards, in our judgement, is often misunderstood. A common assertion is that open standards create an environment within CI software that avoids "vendor lock-in" because there can be multiple implementations of the same standard; presumably a customer of one vendor can chose another if the customer is dissatisfied with the standard implementation. In our experience, this is not appropriate for CI software: the community of available developers would be better off collaborating on a single reference implementation of the foundational software and competing with each other on value-added capabilities built on top of the standards - as is the practice in open source communities such as e.g., Apache Hadoop wherein which vendors innovate Hadoop's core, but also make their own "distributions" e.g., Cloudera Distribution for Hadoop; or Hortonworks Data Platform.

Our ideal for CI software is represented in Figure 1: CI-enabled scientific research, CI research itself, and CI operations are all mutually supportive. The core of our position is the center of Figure 1: open community software built by contributions of stakeholders of the three concerns.

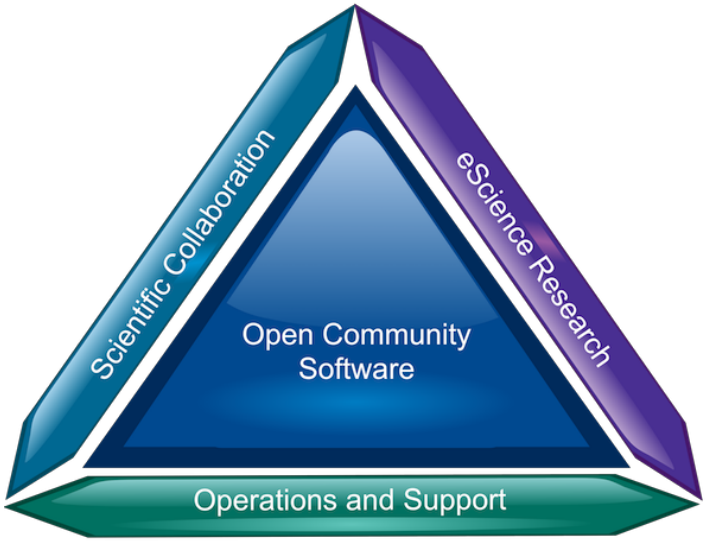

Figure 1: Open community software supports scientific applications, cyberinfrastructure research, and operations.

Governance Functions, Stakeholders, and Implementations

Software governance is used to make decisions about the project. Typical decisions include the following: a) deciding if a new stakeholder should be added; b) deciding who has write access to the main version of the code base; c) deciding when to make a software release, what is in the

release, who will be responsible for putting the release together, and if the released software artifacts meet the project's standards for functionality, packaging, and licensing; d) making major project decisions such as changing the software's APIs, adding new features, removing obsolete features, and significantly revising existing capabilities. These correspond to major and minor version changes in semantic versioning schemes.

As previously discussed, we use the term *stakeholder* to mean anyone involved in a software community who can participate in the above governance functions. A stakeholder may be a developer with write-access to the code trunk, but this is not required. Stakeholders may also include funders of the software, important users of the software, champions of the software, and volunteers who contribute by producing documentation, tutorials, and outreach material. Stakeholders interact with each other through the project's governance mechanisms.

Governance can be implemented in a number of ways. Decisions may be made at specific locations or asynchronously. Deliberations may be open or closed. Resolution on issues can be done by stakeholder vote, although the weighting of the votes may not be equal. Veto mechanisms may be explicitly defined or implicit in the voting process (that is, consensus may be a prerequisite).

We assert the need for *open governance* in federally funded CI software projects to make them actually open. Open governance is characterized by project deliberations on open, archivable forums. Resolutions are made through open voting using the same open forums, with votes carried out asynchronously over a period of time that allows all stakeholders the chance to express an opinion and cast a vote. Resolutions may pass with simple majority, although it will be common to seek consensus to avoid community splintering.

Open Governance and the Apache Software Foundation

There are numerous approaches to software governance [1,2]. Many of these, such as the "benevolent dictator for life" model, are closely associated with the project's history. It is also certainly possible for a set of software stakeholders to invent a workable governance model *ex novo*. Previous efforts within the CI software and related academic communities are summarized in [3]. In our own experiences, we have found it useful to align our software projects with the Apache Software Foundation (ASF). The ASF is an organization factory: it is an organization that creates other organizations. The advantage of the ASF approach to CI software over alternatives is that the ASF has proven its viability as an umbrella organization for projects at a wide range of scales and a neutral field for collaboration between corporations and individuals who otherwise compete directly in the market. There is no need to invent a new approach.

The ASF is described more thoroughly in [4, 5], although the ASF processes are better understood as the governance mechanisms of society of individuals rather than an algorithmic set of rules. New projects come under the ASF umbrella by first going through a period of incubation. Incubated projects have a champion (usually a well known ASF member) and at least three mentors drawn from members of other, established ASF projects. Incubating

projects, with guidance from their mentors, must demonstrate that they understand and embrace the ASF open governance mechanisms as well as show a vibrant and active community. Incubating projects are expected to make software releases following Apache guidelines for packaging, licensing, and approving the release but are not expected to have “1.0” caliber software before graduation. Graduating from incubation into a fully fledged top level project indicates instead the the community is “version 1.0”.

Experiences and Conclusions

Our initial experiences with the ASF come from co-founding several projects (Apache Airavata, Apache Rave, Apache Tika, and Apache OODT) that have all graduated from incubation as top level projects. A significant advantage we have derived from our Apache involvement is the valuable mentoring relationships that we established during our incubation with ASF members who have expertise in open source software methodology, running commercial software companies based on open source software, and fostering communities of scientific software developers. Apache provides a community of communities and much cross-pollination.

Our projects show the potential value of applying Apache governance to CI software, and Apache Hadoop, Cassandra, jClouds, Cloudstack and others are (intentionally or otherwise) becoming key parts of the cyberinfrastructure landscape. However, open governance remains elusive for a significant portion of CI software. Academic groups still need to see compelling evidence that giving up dictatorial ownership of their software is compatible with winning grants in highly competitive environments. We still need to find the balance between collaboration and competition. It is possible that the change in attitude will come from the next generation of developers, who will see the value in moving seamlessly from one CI software project to another, as well as the value of proving their programming chops to future employers through membership in Apache projects and clearly measurable, documented software skills.

Cloud computing models such as “Platform as a Service” and “Software as a Service” present new challenges to open governance. The software that powers these services may be open source, but the actual operations--the configurations used, the deployment environment and settings, and so forth--may be treated as closely guarded, proprietary secrets. Open governance approaches must extend themselves into what we may term “open operations”.

References

1. Raymond, Eric. "The cathedral and the bazaar." Knowledge, Technology & Policy 12, no. 3 (1999): 23-49.
2. Ljungberg, Jan. "Open source movements as a model for organising." European Journal of Information Systems 9, no. 4 (2000): 208-216.
3. Stewart, Craig A., Guy T. Almes, and Bradley C. Wheeler. "Cyberinfrastructure Software Sustainability and Reusability: Report from an NSF-funded workshop." (2010).
4. “How the ASF Works”: http://www.apache.org/foundation/how-it-works.html
5. Fielding, Roy T. "Shared leadership in the Apache project." Communications of the ACM 42.4 (1999): 42-43.