Pierce, M E et al 2015 Patching It Up, Pulling It Forward. *Journal of Open Research Software*, 3: e12, DOI: http://dx.doi.org/10.5334/jors.bz

ISSUES IN RESEARCH SOFTWARE

Patching It Up, Pulling It Forward

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

[1] University Information Technology Services, Indiana University, Bloomington, IN 47408
marpierc@iu.edu, smarru@iu.edu

[2] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109
chris.a.mattmann@jpl.nasa.gov

[3] Department of Computer Science, University of Southern California, Los Angeles, CA 90089

An important reason for making any software open source is to encourage code and other community contributions, resulting in more diverse developer communities coalescing around valuable software efforts. We believe the full picture of open developer communities is underappreciated by scientific and cyberinfrastructure open source software efforts. Free and open source licensing is popular in scientific and cyberinfrastructure software, and Web-based tools for source code management (such as GitHub and Bitbucket) are in common use, but community building efforts and associated governance models that foster these communities need improvement. We propose here a simple mechanism to address this problem: developers should be given incentives to submit patches and to make other measurable contributions to code bases that they use but are not otherwise connected to, and projects should be given incentives to accept these outside contributions. As an example implementation, we outline a contest system with small monetary rewards for individuals and recognition for both individuals and projects. The goal is to change the mindset of scientific and cyberinfrastructure developers, converting them from passive downstream users to active contributors. We hypothesize that this easily measurable concrete action will contribute to the sustainability of many projects and also create a more flexible scientific workforce. Building this effort on currently available, federally funded software will establish a foundation of public data that can be used to verify our hypothesis. More broadly, the effort will demonstrate the benefits for scientific and cyberinfrastructure projects that adopt workable governance models that are already well established in the broader open source software ecosystem.

Keywords: cyberinfrastructure; open source software; software governance

Introduction: Open Source and Open Governance

Sustainable software depends on communities of users and developers who are invested in the software's success [1]. These communities need rules that guide their interactions, that encourage participation, that guide discussions, and that lead to resolutions and decisions [2, 3, 4]; we refer to these rules as a community's governance. Open governance [4, 5, 6, 7] provides well-defined mechanisms executed through open communications that allow individuals from diverse and even competing organizations to interact in neutral forums in a collaborative manner that encourages growth and transforms passive users into active contributors and project members. For a general overview of open source software governance mechanisms, which may take different forms, see [7].

These principles are well-established in the general open source software ecosystem, but it is our experience that the adoption of a well-articulated governance model by scientific and cyberinfrastructure software efforts primarily funded by federal agencies is particularly inadequate; these projects compare unfavorably to a host of cyberinfrastructure-like software efforts such as Apache Hadoop, Apache Spark, and Apache Mesos that have open, Apache-style [5] governance. The growing distinction between being "open source" and being community managed via a governance model is summarized in [4].

Our hypothesis is that scientific and cyberinfrastructure software sustainability would benefit from using open governance methods, which would create more resilient developer communities. This position is supported by the statistical analysis of 352 open source projects as reported in [8]: openly governed projects were significantly more efficient at what the authors call "enhancive maintenance" (adding new features, improving performance) but less efficient at corrective maintenance (bug fixes). We extrapolate that enhancive maintenance is vital to software's long term viability as it moves the software onto new platforms, increases its capabilities, and generally keeps the project moving forward. Similarly, Shah [9] found that openly governed open source software projects attracted more engaged contributors who worked for longer periods of time on the project voluntarily. Openly governed software projects would thus be better able to create and sustain a pipeline of developers and contributors at all levels, and individuals would be able to publicly demonstrate their coding skills and adaptability to the specific developer cultures of multiple projects. They would also be properly credited for their work. Thus openly governed projects are designed to increase the number of constituent stakeholders. Stakeholders are responsible for making decisions, developing software, fixing bugs, creating software releases, and writing documentation. Greater stakeholder diversity (that is, not all stakeholders come from the same organization or are paid to work on the project by the same funding source) increases the resiliency and sustainability of the project in the face of uncertain funding and developer turnover.

In open source software projects, competitors take the unusual step of agreeing that it is to everyone's advantage to work on a common code base, joining forces as stakeholders in a common community with well-defined governance rules. Famous examples include the Apache HTTPD server, the Linux operating system, many programming languages such as Perl, Ruby, and Python, and (more recently) platforms such as Apache Hadoop, Apache Cassandra, Apache Spark, OpenStack, and others. These projects do not exist strictly to support science and most are not academic, but the same open source principles that they use also apply to academic cyberinfrastructure software supporting scientific research. It is our contention that much of these open source principles have been inadequately applied or misapplied by the academic research community.

Cyberinfrastructure software is developed to support e-Science research. It is software that supports large scale distributed computing, scientific computing, and scientific data management. Cyberinfrastructure software is primarily funded by government agencies to enable scientific research and may also be the subject of computer science research itself. "Open" cyberinfrastructure software is often taken by its would-be practitioners to mean various things. The software may have an open source or free license, but how this is enforced or ensured by the project is often not clear. The software may be openly available on the Web through online version repositories such as GitHub, Bitbucket, SourceForge, and Google Code, where it can be viewed, branched, and so on, but how one contributes back to the trunk of this open source code and gets credit may not be clearly defined. Code management technologies offered by online resources like GitHub may help, but accepting patches and granting full access to the code trunk are ultimately human decisions.

The code may even implement open, community standards, but the value of these standards, in our judgment, is misunderstood. A common assertion is that open standards create an environment within cyberinfrastructure 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. In our experience, this is not appropriate for cyberinfrastructure software with its limited developer pools and smaller communities: 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 Apache Hadoop wherein vendors such as Cloudera and Hortonworks innovate on Hadoop's core and also make their own distributions.

Our ideal for cyberinfrastructure software is represented in **Figure 1**: cyberinfrastructure enabled scientific research, cyberinfrastructure (e-Science) research itself, and cyberinfrastructure operations (that is, organizations that operate cyberinfrastructure on behalf of scientists) are all mutually supportive and dependent. Open community software built by contributions of all three types of stakeholders is at the core. Since each of the stakeholding groups has a different mission or goals, different funding sources, and different members, it is essential that the software at the center has a defined governance model.

Governance Functions, Stakeholders, and Implementations

Project members use software governance to make decisions about the project. Example 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 and internal components.

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.

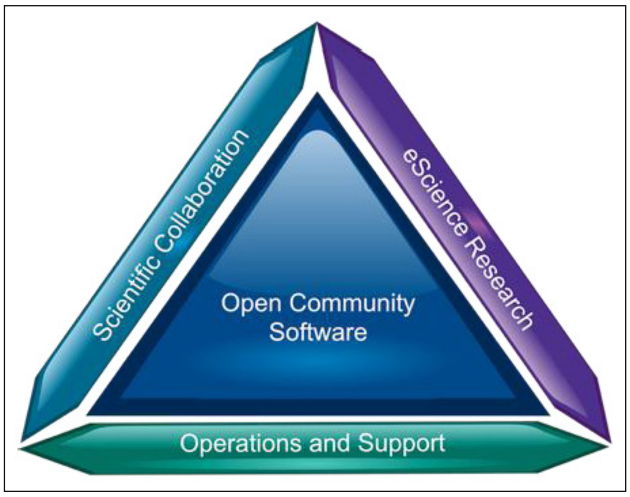

Figure 1: Open community software supports scientific applications, cyberinfrastructure research, and operations.

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. Issue resolution 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). The Apache Software Foundation provides a well known example for open governance: project membership is not limited to a particular organization (technically, all members are part of the Apache Software Foundation and act as autonomous individuals), and all but a few decisions are made by voting on publicly available, archived mailing lists; discussions of new candidate project management committee members and committers are the main exceptions, but these discussions are done on archived private lists. Apache can be viewed as an organization factory that creates and supports other organizations. After an incubation period, projects that demonstrate that they have implemented Apache governance mechanisms can graduate to full project status. Graduation means that the software is backed by a "1.0" community, not necessarily that the software itself is "1.0" quality yet.

We assert that there is a need for the greater adoption of open governance in scientific cyberinfrastructure software projects to make them truly open and accountable. In summary, open governance is characterized by project deliberations on open, archived 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 is common to seek consensus to avoid community splintering.

Call to Action

In our view, the litmus test for open governance is the ability of a project to absorb a software contribution from a non-member and to properly acknowledge the contributor. We therefore make the following call to action:

Stop just taking. Stop being passive. Contribute to projects that you like or depend upon. The best way to contribute is by submitting patches and pull requests to improve the code base and fix bugs. So pick your favorite open source project, find something that needs your help, submit a patch, and see what happens next.

Governance: Give and Take

We next discuss a possible implementation of this call. Among our goals is to implement the call in a measurable fashion that can be used to substantiate our hypotheses about the value of open governance to scientific and cyberinfrastructure software.

Open source software lives or dies over the long term by the number of people who are able to make significant contributions to the code base. This reverses the usual give and take relationships between software providers and their user communities; we must foster the attitudes of both giving back and accepting within the scientific and cyberinfrastructure software communities through code contributions. To accomplish our goal, we call upon members of these communities to start submitting software patches as part of a community-wide experiment. These contributions must be measurable, requiring both Web-based source code management tools as well as public, archived discussion forums.

The "submit a patch" activity is designed to foster community growth and a giving mindset in individual developers, but it is also a way to expose and hopefully correct flaws in the management and governance of the targeted software projects. Examples of possible outcomes, both good and bad, that this activity may expose are summarized in **Table 1**.

Implementation and Incentives

The call to action by itself is unlikely to convince a critical quorum of developers to start voluntarily submitting patches to other projects. It is thus necessary to consider incentives that will be needed to implement this call to action. We will assume that only small amount of funding is available for incentives, so we must use a cost-efficient strategy.

Scope: This call to action can be implemented by a national funding agency (such as the National Science Foundation in the United States) or through a collaboration of funding agencies. Eligible projects would be those that are at least partially funded by the funding agency. A specific implementation may choose to target a smaller subset, such as current award winners of the National Science Foundation's Software Infrastructure for Sustained Innovation (SI2) program.

Prerequisites: For a project to participate, it must have the following:

- **An open source or free source license.** The source code must obviously be licensed in a way that allows contributions.
- **Publicly available source code in an online source code management system.** Having code in a public repository enables indexing and analysis tools such as OpenHub (formerly Ohloh) to measure user contributions. Most public repositories have similar tools. This will be essential for measuring impact of the proposed effort in the long term.
- **Public forums for discussion.** Our proposed effort requires a publicly accessible, archived and searchable forum for contributors to interact with project developers, submit patches, and discuss contributions.

Developer Incentives: Although we anticipate that there will be many long-term benefits to individual developers (such as publicly demonstrated coding ability and ability to interact productively with other developers), this will most likely not be enough incentive to initiate the project. We therefore propose launching a series of one-year contests with recognition at a major community venue (such as Supercomputing). Awards (plaques, medals,

Scenario	Outcome
A willing volunteer dives into the project but cannot see how to get started submitting a patch for anything.	The project is not well documented, is not modularly designed, has a broken build and test system, is not using issue tracking systems, has no easy way to communicate with developers directly, etc.
The volunteer creates a patch but then does not know what to do with it.	The project does not have a way to accept patches (by Jira issue, through a developer mailing list, etc).
A volunteer submits a patch, but it is ignored.	The project does not actually want contributions; the project members are unaccustomed to receiving a patch and do not know what they should do with it; the developers decide to appropriate the patch ideas for themselves and not share credit (hopefully a rare outcome); the project is no longer active, so no one receives the patch.
The patch is discussed but never applied.	The patch may be deemed unacceptable after public discussion and iterations with the contributor; the project may not have (or think they have) resources to apply the patch; the project may not want the patch.
The patch is applied but the volunteer later doesn't feel properly credited.	The project may not have thought through intellectual property and copyright issues.
The patch is applied (typically after some iterations) and incorporated into the release.	The project is a mature open source project with open governance.
The contributor submits several more patches and is eventually given write access to the main code base and the ability to participate in major project decisions.	The project has a governance model that it uses to make these decisions.

Table 1: Possible outcomes from patch submissions and pull requests that reflect the health of project governance.

small honoraria) will be given for the following individual achievements to external developers who

- Submits the most patches in released software of participating projects.
- Contributes to the most participating projects.
- Are given trunk write access to the most software repositories of participating projects.

In all cases, the external developer cannot be paid directly by the project owners (that is, the owners of the project's license or copyright) and does not begin the project with write access to the code base.

Project Incentives: As with developers, we anticipate that software projects will have many long-term benefits from participating in our proposed efforts, but short-term incentives will be needed to initiate the effort. We propose that awards of recognition be given to projects for the following achievements.

- Most patches by external developers accepted in released software.
- Most external developers contributing patches.
- Most new, (formerly) external developers given write access to the project's trunk (release) code base.

Again, "external" developers are developers who are not paid directly to work on the project by the owners of the project's license or copyright. Projects will receive the awards at a prominent community venue (such as Supercomputing). Projects will not be given monetary awards, but they may apply for funding to organize developer workshops. Awards may be categorized according to the size of a project's stakeholder community (i.e., small, medium or large).

Related Work

The authors are practitioners of cyberinfrastructure research and development, and we base our positions in this paper on empirical evidence and extended observations of our own and other software efforts, including both academic and non-academic projects. This paper is an extension of two previous white papers [10, 11]. In [10], we describe our involvement in the Apache Software Foundation and provide a longer discussion of the application of the foundation's governance mechanisms to cyberinfrastructure software.

The Computer-Supported Cooperative Work (CSCW) conference series (most recently, [12]) provides a survey of the field via direct and rigorous research on open source and other software development communities. [13] provides a survey of research into open source and free software communities. Our paper looks at targeted incentives for open source developers that we believe will spur more transparent cyberinfrastructure project governance, but general issues with incentives in scientific software are explored in [14, 15]. From our point of view, [15] describes anti-patterns resulting from improper governance.

Our proposed incentive model targets individual developers with monetary awards as well as recognition. Projects receive only awards of recognition. An alternative award system that gave monetary awards to projects is Mellon Awards for Technology Collaboration (http://old.arl.org/news/pr/mellontechawards06~print.shtml). We believe that targeting individual developers is more cost effective and also more likely to provide objective metrics.

Google Summer of Code (GSOC) and OpenHatch are two activities aimed particularly at students and younger developers. GSOC pays students a small stipend to work for several months with a well-known open source community. Google selects the projects that can participate and sets a barrier that may be too high for many of the scientific and cyberinfrastructure projects that we consider. GSOC splits the stipend over three payments, giving students an initial payment but reserving the final two payments for mid-term and final evaluations. In our proposed model, the barrier for projects to participate would be much lower (meeting the requirements described above would be sufficient, similar to the requirements for projects to participate in OpenHatch), and the number of awards would be much lower. We don't anticipate that developers would earn a living primarily through the contest. The awards are provided mainly as a recognition for outstanding achievement. To our knowledge, there is no full scale academic study of GSOC, although [16] examines its impact on one project.

Conclusion: Measuring Long-Term Impact

Our working hypothesis is that encouraging more outside contributions will benefit both scientific software projects and individual developers and scientists. Projects will potentially become better organized and more transparently governed in order to accept code contributions and grow to include a broader stakeholder base. This will contribute to better sustainability in a number of ways: new project staff members can be recruited and carefully vetted over a longer process, and the project will be more resilient to the loss of key individuals. Individual developers and scientists will likewise benefit from the proposed effort by publicly demonstrating their coding abilities, their ability to translate between science and software, and their interpersonal and communication skills. This may serve to keep more people interested in academic software projects. A young developer can, for example, increase his or her publicly demonstrable skills and reputation while remaining in scientific or cyberinfrastructure software development. This may lead to better and more secure employment both within and outside academia.

These are subjective conclusions that need to be more carefully measured over a number of years. The award-based incentive program that we have outlined can be undertaken for low-cost for a number of years by an interested funding agency. The result will be measurable data that can be used to test our hypothesis.

Competing Interests

The authors declare that they have no competing interests.

References

1. **Stewart, C A, Guy, T A** and **Bradley, C W** 2010 Cyberinfrastructure Software Sustainability and Reusability: Report from an NSF-funded workshop.
2. **Raymond, E** 1999 The cathedral and the bazaar. Knowledge, Technology & Policy, 12(3): 23–49. DOI: http://dx.doi.org/10.1007/s12130-999-1026-0
3. **Ljungberg, J** 2000 Open source movements as a model for organising. European Journal of Information Systems, 9(4): 208–216. DOI: http://dx.doi.org/10.1057/palgrave.ejis.3000373
4. **O'Mahony, S** 2007 The governance of open source initiatives: what does it mean to be community managed? Journal of Management & Governance, 11(2): 139–150. DOI: http://dx.doi.org/10.1007/s10997-007-9024-7
5. **How the ASF Works?** Avavilable at: http://www.apache.org/foundation/how-it-works.html.
6. **Fielding, R T** 1999 Shared leadership in the Apache project. Communications of the ACM, 42.4: 42–43. DOI: http://dx.doi.org/10.1145/299157.299167
7. **Markus, M L** 2007 The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance,* 11(2): 151–163. DOI: http://dx.doi.org/10.1007/s10997-007-9021-x
8. **Midha, V** and **Bhattacherjee, A** 2012 Governance practices and software maintenance: A study of open source projects. Decision Support Systems, 54(1): 23–32. DOI: http://dx.doi.org/10.1016/j.dss.2012.03.002
9. **Shah, S K** 2006 Motivation, governance, and the viability of hybrid forms in open source software development. Management Science, 52(7): 1000–1014. DOI: http://dx.doi.org/10.1287/mnsc.1060.0553
10. **Pierce, M, Suresh, M** and **Mattmann, C** 2013 Sustainable Cyberinfrastructure Software Through Open Governance. figshare. DOI: http://dx.doi.org/10.6084/m9.figshare.790761. Retrieved 19:01, Jul 18, 2014 (GMT).
11. **Pierce, M, Suresh, M** and **Mattmann, C** 2014 WSSSPE2: Patching It Up, Pulling It Forward. figshare. DOI: http://dx.doi.org/10.6084/m9.figshare.1112540. Retrieved 19:49, Mar 06, 2015 (GMT).
12. **Pierce, M, Suresh, M** and **Mattmann, C** 2014 Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing. ACM, New York, NY, USA.
13. **Crowston, K, Wei, K, Howison, J** and **Wiggins, A** 2012 Free/Libre open-source software development: What we know and what we do not know. ACM Computing Surveys (CSUR), 44(2). DOI: http://dx.doi.org/10.1145/2089125.2089127
14. **Howison, J** and **James, D H** 2011 Scientific software production: incentives and collaboration. In Proceedings of the ACM 2011 conference on Computer supported cooperative work. ACM, pp. 513–522. DOI: http://dx.doi.org/10.1145/1958824.1958904
15. **Howison, J** and **James, D H** 2013 Incentives and integration in scientific software production. In Proceedings of the 2013 conference on Computer supported cooperative work. ACM, pp. 459–470. DOI: http://dx.doi.org/10.1145/2441776.2441828
16. **Trainer, E H, Chaihirunkarn, C** and **James, D H** 2014 The Big Effects of Short-term Efforts: Mentorship and Code Integration in Open Source Scientific Software. *Journal of Open Research Software,* 2(1): e18. DOI: http://dx.doi.org/10.5334/jors.bc

How to cite this article: Pierce, M E, Marru, S and Mattmann, C 2015 Patching It Up, Pulling It Forward. *Journal of Open Research Software* 3: e12, DOI: http://dx.doi.org/10.5334/jors.bz

Published: 19 November 2015