What Open Source Software Development Teaches Us about Scientific Governance

An NSF EarthCube White Paper

Dan Bedard, Ray Idaszak, Howard Lander, Stan Ahalt
Renaissance Computing Institute, University of North Carolina at Chapel Hill

Abstract—In this whitepaper, we posit that an ***open science governance model*** will enable scientists to address emerging scientific Grand Challenges and to develop the infrastructure required to conduct Grand Challenge research. We present the open source software development model as a starting point for discussion and ***make the distinction between the commonly-held perception of open source and the actual mechanics of open source that have permanently changed the software development landscape.*** Further, we explain how open source mechanics can be applied to ***make science more efficient while enhancing opportunities for constructive competition, in concert with the existing scientific governance superstructure***. Finally, we present future challenges—unanswered questions, submitted in the spirit of open science—for the community to discuss in pursuit of the next generation of scientific discovery.

A recent NSF Task Force Report on Grand Challenges analyzes a set of emerging research priorities and identifies commonalities among them in an effort to determine the cyberinfrastructure developments required to address the challenges. Among the report's findings are that the challenges, which include such topics as climate change prediction and mitigation, managing greenhouse gases, and hazard analysis and management will all demand **transformative discovery and innovation across disciplines; advanced computational models and algorithms that operate at multiple scales; and the collection, management, and analysis of massive, heterogeneous datasets (NSF 2011).**

The existing model that governs scientific research is not equipped to enable the scale of collaboration, data sharing, and software interoperability that will be required to pursue Grand Challenge research problems. The shortcomings of the prevalent model occur throughout the sponsored research cycle, and include:

Static Collaboration The formal proposal process typically enforces *static* or *semi-static* collaborations that form before research begins, typically among a pre-arranged group of collaborators. Because grants are designed to carefully scope deliverable and budgets, it is logistically challenging to expand and evolve the circle of collaborators as projects develop. The barrier to entry into existing, previously established collaborations is high for all scientists who are not part of the original collaboration, and even more challenging for less experienced scientists.

Review Lag Research is typically carefully reviewed within a closed community on a yearly basis, and peer review only occurs after a lengthy publication process. This timescale is not rapid enough to detect the need for course corrections before problems become ingrained, nor does this timescale permit rapid implementations of course corrections in the research plan.

Incentives Researcher tenure and stature are heavily weighted on journal publications and strongly-reviewed conference publications. However, these types of scholarly works rarely include source code or source data, making rigorous peer review—and follow-on research—unwieldy, if not impossible. Beyond counting citations, it is difficult to ascertain the impact of a researcher's work or their body of work.

Sustainability The current infrastructure does not provide a standard framework for publishing and maintaining data and source code for others to replicate and use in future projects.

The NSF's EarthCube program represents a tremendous opportunity to transform scientific research into a more open and agile activity. The timing is right for this transformation: the driving problems are increasingly salient, the Internet provides a versatile medium for the instant, broad dissemination of information and interactions, and the modern global playing field has been "flattened," in large part due to community participation in online projects through open source software (Friedman 2007). As a result of these developments, a governance model now exists that can be used to guide the practice of science in the open.

We propose that the open source software model can address several of the shortcomings of the usual scientific governance paradigm by stimulating dynamic, productive collaboration and review, and by providing the mechanisms and incentives for contributing knowledge—in the form of software, data, or other media—to the commons, while still preserving constructive competition within the EarthCube community.

Open Source May Not Mean What You Think It Means

The term "open source" is a victim of its own success, in that it has been applied as a selling point to projects that have failed to live up to the promise the model holds. To be clear, open source mechanics encompass much more than simply applying a relaxed form of licensing to source code and hoping that volunteers will clamor to contribute to it. Rather, the guiding principle of open source is the expectation that each contributor will transparently review, evaluate, and build upon the previous body of work. In this sense, open source software development is analogous to the current practice of scientific publication.

Open Source Software Development

Figure 1 depicts how the open source software development cycle guides the creation of a software project or a collection of software projects:

1. The cycle begins in service of the requirements of:
    - a client-oriented project,
    - a lone developer or small group of developers "scratching an itch,"
    - a proprietary product released into the open source arena, or
    - a fork of an existing project (Tucker 2011).

2. Using search tools and repositories, it is determined if existing open source software meets the need or comes close to meeting the need.

3. If no appropriate project exists, the user begins a new project, selects a license, and recruits volunteers. Recruiting volunteers is generally more successful if the user can start with a working prototype. If an existing project comes close, the user works with the project community to request help meeting the need.

4. The workgroup evaluates approaches to the problem and, via domain analysis, creates a workplan, while continuing to evaluate if existing projects might meet the need. User stories and use cases are created.

5. According to the workplan, the team sets up a collaborative development environment and develops and tests software code. Roles and processes are defined. Contributions from workgroup members are stored in a central repository.

6. A subset of the workgroup reviews the submitted code. "Stable versions" of the code are packaged and released to the public on a set schedule according to a release plan.

7. Some open source projects evolve to have broader impact. They mature through an incubation stage, building a broader participant/contributor community around the project in order achieve self-sustainability, with perpetual governance based on democratic meritocracy. Foundations often assist in this process.

8. Self-sustaining open source projects often either become stand-alone foundations or seek membership in a larger organization to ensure continuity and sustenance (Tucker 2011).

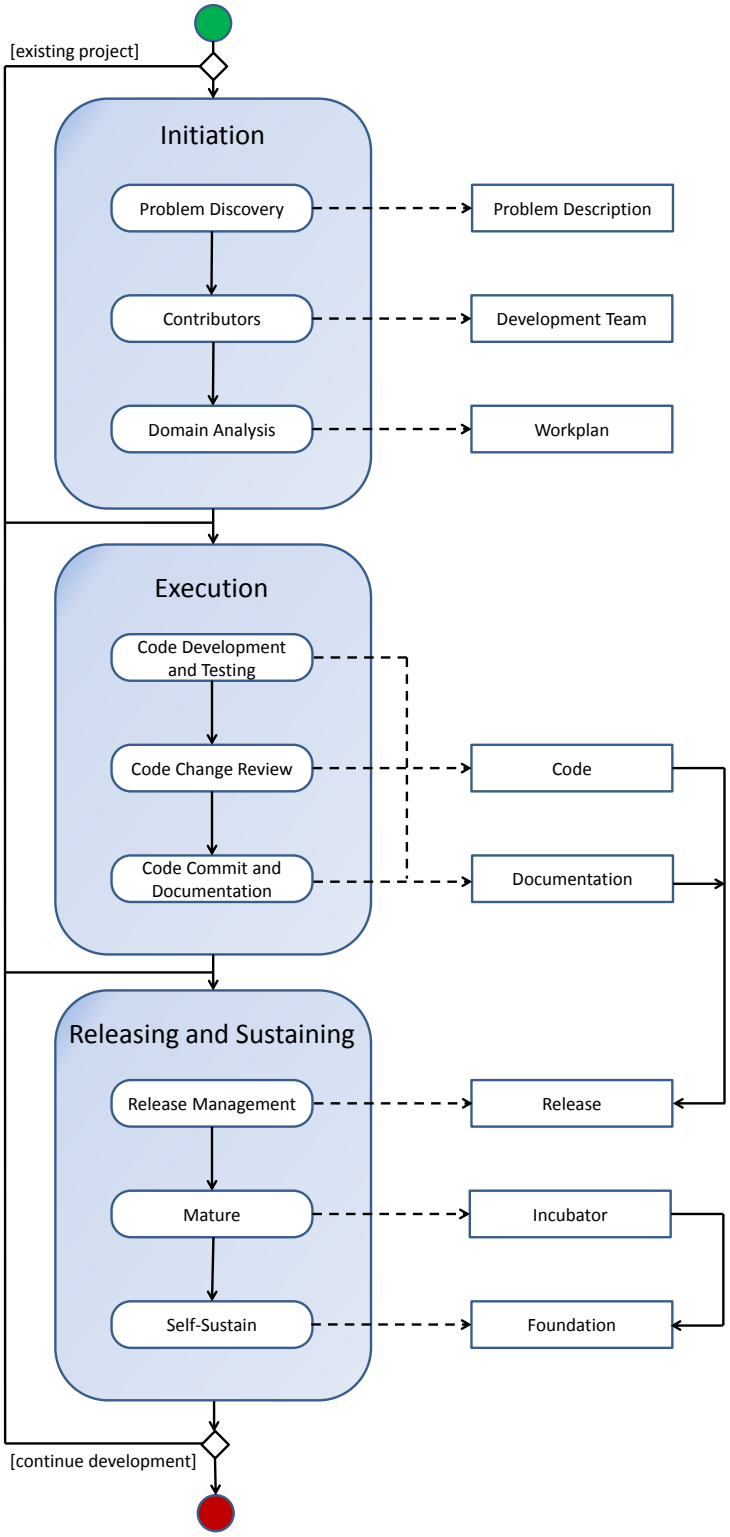
[existing project]
Initiation
Problem Discovery
Problem Description
Contributors
Development Team
Domain Analysis
Workplan
Execution
Code Development and Testing
Code Change Review
Code
Code Commit and Documentation
Documentation
Releasing and Sustaining
Release Management
Release
Mature
Incubator
Self-Sustain
Foundation
[continue development]

Figure 1

It is important to note that a release committee makes the determination of the value and validity of code submitted to a project. Release committees tend to draw from the most prolific contributors to a project, and they often include representatives of projects' most influential stakeholders. As such, successful open source governance rarely exists on its own—it is *supported by* other governance structures that enjoy the advantages it provides.

There are numerous examples of successful open source software development projects, including the Linux kernel, the Apache web server, and the WordPress blogging platform. Key aspects of the open source governance model which are critical to all successful open source projects include:

Dynamic Collaboration Potential developers can join project teams at any point during the project life cycle. The barrier to entry for a new developer is kept low by making the code modular with robust software engineering and by factoring documentation into review of all developers' contributions (Fitzgerald 2011).

Robust Competition Open source development allows contributing organizations to agree upon and distinguish between areas of cooperation and areas of competition. Resources can be focused cooperatively or competitively on solving the most dynamic and challenging problems within a common, stable framework; the user community reaps the benefits (Tiemann 2011).

Modularity, Accessibility, and Inclusivity By dividing functions into discrete modules, each developer can make contributions without threatening the stability of the entire project's codebase (Upton and Staats 2008).

Central Repositories Projects and corresponding individual code modules are stored in a persistent, searchable, and publicly accessible location.

"Release Early and Often" Reviews and updates of the codebase on a frequent schedule helps to quickly identify emerging development priorities and locate errors (Raymond 2001).

Metrics and Incentives By tracking developers' contributions and how they meet the needs of the user public (e.g., through download counters and bug tracking), it is possible to quantify the impact of individual contributors. A developer's stature within the community is tied to the merit of his contributions. Further incentives within the development community also exist.

Assessment Projects are evaluated on key open source software characteristics, including functionality, usability, quality, community, documentation, security, support, scalability and performance, architecture, adoption, and quality and professional integrity of governance (Tucker 2011).

Bonus Contributions When private companies have funded open source development, they have generally realized benefits beyond the work they have directly sponsored, due to the intellectual contributions of unpaid volunteers, or from contributions by developers outside the expected project community.

Scientific Governance Based on Open Source Mechanics

These important aspects of open source software mechanics (Hellekson 2010) apply quite readily to scientific research. The EarthCube blog and charrette already represent a step in the direction of open science, providing a forum for discussion among the broader community early in the development process.

Recall that open source governance works *in concert* with an overarching community governance model. Thus, it lends itself well to—in fact, thrives in—such organizations as large centers, university consortia, foundations, and corporations seeking the increased collaboration, interoperability, and agility it enables. Figure 2 illustrates the EarthCube community of communities administering open source governance in support of a move toward open science.

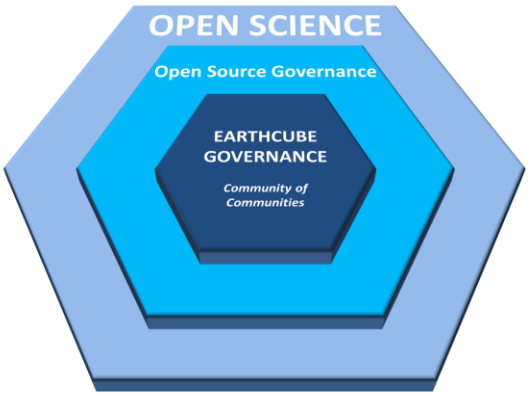

Figure 2

While the open source community has developed numerous technologies that the research community can use to facilitate open science, the onus lies on the scientific governance model to provide policies that do the same:

1. Novel funding mechanisms must be developed to permit rapid reconfiguration of project teams as project priorities are developed. This may include base funding that employs graduate students or post-docs, along with supplemental funding of many smaller short-term projects instead of a small number of large projects. Awarding bonuses for heavily used modules is another possible mechanism to incent development that can be repurposed and sustained. This degree of flexibility may require a PI, or an Executive Committee, to hire new contributors on short sub-contracts in order to seed or replenish the community.

2. A project's data and software should be developed in anticipation of *direct* peer review and further collaboration. Indeed, inclusion of one or more modules into a release should be considered a form of peer-review, and accordingly "counted" as a scientific contribution. Community standards should be drafted for documentation and metadata, publication, and maintenance.

3. Metrics of the impact of a researcher's software, data, blog posts, and other contributions should be collected and published in a format that can be used for such applications as funding and tenure panels.

Open Problems

We note that there are open problems associated with open source scientific governance, and we encourage the EarthCube community to discuss and debate these and other potential challenges with this model. These include:

What are the right funding mechanisms? Additional investigation is required to determine the optimal balance of stable and flexible funding and to develop the appropriate administration and accounting mechanisms for non-traditional funding strategies.

How can "infinite loops" be avoided? Open source software development depends on dividing large projects into smaller, bounded problems. Some scientific research objectives may not have tractable short-term outcomes. The governance model will require methods to detect and mitigate this condition.

When should research remain closed? In some cases, there are legitimate reasons to delay the sharing of specific research outcomes, or of data. Because this condition implies that the research is sensitive, it is critically important to accommodate these constraints while still ensuring that progress is hindered as little as possible.

Making education and outreach a priority. A recent NSF-funded workshop on cyberinfrastructure software sustainability and reusability has suggested that "the computer science, computational science and computationally-oriented scientific discipline would benefit from the widespread adoption of one or a very few standard excellent textbooks or other learning materials in software engineering" (Stewart, et al. 2010). It may be beneficial to apply a type of open source release cycle to educational materials. The text *Software Carpentry* is a highly successful example of an educational text on software engineering created and sustained via open source (Software Carpentry 2011).

Exactly how will metrics be determined and implemented? Metrics, including funding, publication, use of modules within and beyond target communities, inclusion in releases, identification of bugs or errors, maintenance, contributions to educational resources, etc. need to be both harnessed and harmonized across the open source science governance team.

In the short term, introduction of the open source governance approach will result in software projects that sustain themselves, producing continual returns to the originating community as they evolve in concert. In the long term, the intent of this approach is to change the culture of the community and the NSF to embrace sustainable open source and software engineering practices. The application of open source mechanics has repeatedly resulted in unforeseen innovation. In the context of the NSF, this translates into translational new open science that is continual and sustainable.

Works Cited

Fitzgerald, B. (2011). "Open Source Software: Lessons from and for Software Engineering." Computer **44**: 25-30.

Friedman, T. L. (2007). The World is Flat: A Brief History of the Twenty-First Century, Picador USA.

Hellekson, G. (2010). Freedom Isn't Free: Open Source Mechanics. NSF Workshop on Creating a Scientific Software Innovation Institute (S2I2) for Environmental Observatory Communities.

NSF. (2011). "Advisory Committee for Cyberinfrastructure Task Force on Grand Challenges, Final Report, March 2011." from http://www.nsf.gov/od/oci/taskforces/TaskForceReport_GrandChallenges.pdf.

Raymond, E. S. (2001). The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly & Associates, Inc.

Software Carpentry. (2011). "Software Carpentry." from http://software-carpentry.org/.

Stewart, C. A., G. T. Almes, et al. (2010). Cyberinfrastructure Software Sustainability and Reusability: Report from an NSF-funded workshop, Indiana University, Bloomington, IN**:** 5.

Tiemann, M. (2011). "Competition and Sustainability: Two Sides of the Same Coin." February 10, 2011. Fidelity Investments "Leadership in Technology" Executive Speakers Series, from http://www.youtube.com/watch?v=9HrnnpGx6T8.

Tucker, A. M., R; De Silva, Chamindra. (2011). Software Development: An Open Source Approach, CRC Press.

Upton, D. M. and B. R. Staats (2008). "Radically simple IT." Harvard Business Review **86**(3): 118.