Journal of
open research software

Trainer, E H et al 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, pp. 1-5, DOI: http://dx.doi.org/10.5334/jors.bc

ISSUES IN RESEARCH SOFTWARE

The Big Effects of Short-term Efforts: Mentorship and Code Integration in Open Source Scientific Software

Erik H. Trainer*, Chalalai Chaihirunkarn* and James D. Herbsleb*

Scientific progress relies crucially on software, yet in practice there are significant challenges to scientific software production and maintenance. We conducted a case study of a bioinformatics software library called Biopython to investigate the promise of Google Summer of Code (GSoC), a program that pays students to work on open-source projects for the summer, for addressing these challenges. We find three positive outcomes of GSoC in the Biopython community: the addition of new features to the Biopython codebase, training, and personal development. We also find, however, that mentors face several challenges related to GSoC project selection and ranking. We believe that because GSoC provides an occasion to extend the software with capabilities that can be used to produce new knowledge, and to train successive generations of potential contributors to the software, it can play a vital role in the sustainability of open-source scientific software.

Keywords: scientific software; open source; scientific software; mentorship; Summer of Code

1. Introduction

Simply releasing the source code to scientific software is inadequate if it is to remain useful to scientists beyond initial publication [1]. First, scientists often develop "kleenex-code" that they intend to use once and throw away [5]. This software is therefore unlikely to be accessible to other scientists. Second, scientists do not receive obvious benefits from contributing to open-source scientific software, unless those contributions directly result in publication [5, 6]. Third, scientists often lack the expertise to develop production quality software [4, 7]. Fourth, because there tend not to be funding lines for the maintenance of scientific software, the software may quickly become outdated and unusable once project grants are exhausted [9].

Google Summer of Code (GSoC)[1], an annual program that pays students who successfully complete short-term open-source coding projects, may hold promise for addressing these concerns. As a first step, we sought to understand the kinds of features that get integrated, and participants' impressions of the program. We therefore conducted a case study of Biopython [2], a scientific software community with a history of participation in GSoC. Bringing in new contributors and retaining them over the long term are of course primary objectives, but we found a number of less expected effects as well.

* Institute for Software Research, School of Computer Science, Carnegie Mellon University, USA
etrainer@cs.cmu.edu, cchaihir@cs.cmu.edu, jdh@cs.cmu.edu

2. Case study

2.1 Biopython

Biopython comprises a set of free and open-source bioinformatics software libraries upon which researchers in the biological sciences can develop their own applications [2]. The first release of Biopython was in 1999, and the project remains active today. At the time of this writing, the Biopython master branch on GitHub has 9,267 commits, 46 releases, 66 contributors, and 251 active forks[2].

Biopython has participated in GSoC every year from 2009 to 2013 via one of two nonprofit research organizations: The Open Bioinformatics Foundation (O|B|F) or The National Evolutionary Synthesis Center (NESCent). Each year, one of these organizations applies to GSoC on behalf of Biopython. If the application is successful, students then create proposals describing features they wish to develop for Biopython. They also submit their project ideas to the Biopython mailing list. Biopython developers often seed discussions with a list of their own project ideas as well. Afterward, Biopython participants assign possible mentors to each of the projects. The mentors rank all proposals and decide which projects to pursue. They use a competitive interview process to select the students for each project. Google ultimately decides how many projects to fund.

2.2 Data Collection

We collected data for all 10 Biopython GSoC projects from 2009 to 2013, from interviews with Biopython community members, the wiki, and the source code repository.

We began data collection by conducting interviews with 11 Biopython developers. We grounded the interviews

2009	
Project A**	Automated the searching, downloading, and parsing of specimens' geographic location records.
Project B*	Parsing, generation, and manipulation of phyloXML files to store information about phylogenetic trees.
2010	
Project C*	Improved data-mining biomolecular structure databases (Bio. PDB) by adding polar hydrogens to macromolecule structures and interfaces for model validation.
2011	
Project D**	Added a transparent Python plug-in interface to Mocapy++, a machine learning toolkit for parameter learning and inference in dynamic Bayesian networks.
Project E**	Developed a Python interface to Mocapy++ to enable biomolecular structure prediction and simulation.
Project F***	Added large-scale protein analysis capabilities to Bio.PDB and extended it to operate with protein-DNA and protein-RNA complexes (Enhancements to Project C).
2012	
Project G**	Added capabilities to represent sequence variation objects, convert them to and from common human and file representations, and provide manipulations on them.
Project H*	Developed a standard Python interface called SearchIO that supports pairwise sequence search file input/output file formats used by a variety of bioinformatics tools.
2013	
Project I*	Added phylogenetic tree construction algorithms and functions for the consensus of multiple trees.
Project J*	Extended Biopython to support new data types and analyses for codon alignment.

Note. *integrated **not integrated ***ongoing integration.

Table 1: Biopython Google Summer of Code project list.

by e-mailing the top 15 Biopython developers ranked by number of commits to the Biopython master branch on GitHub. Of these 15 developers, 8 responded to our recruitment e-mail, including 1 previous leader (Brad Chapman) and the current leader (Peter Cock) of Biopython. We conducted semi-structured interviews with developers using either Google Hangouts or Skype. Each interview lasted an average of 45 minutes. We also sent e-mails to four former Biopython GSoC students who are not now actively involved in the project. We asked them several open-ended questions about their experiences. All four participants responded.

Furthermore, we extracted from the Biopython wiki entries for each GSoC project. Each entry contains i) the project's name, ii) an abstract, iii) the student's name, iv) the mentors' names, v) a link to the project's branch on GitHub, and vi) a status indicating whether (or not) the project has been integrated into the Biopython master branch.

2.3 Data Analysis

All interviews were recorded, transcribed, and prepared for analysis in the Dedoose qualitative data analysis software [3]. We began analysis by conducting open-coding on all transcripts with instructions to identify statements by participants about GSoC. We then unified and collapsed codes where there was commonality. Moreover, we triangulated statements in the interviews, such as references to mentors, students, and project status with data collected about GSoC projects from the Biopython wiki. In the next phase of analysis, we wrote, shared, and discussed descriptive memos about the role of GSoC.

3. Results

3.1 Project Integration

We triangulated information from the Biopython wiki with the directory structure of the source code repository and found that each GSoC project is in one of three states: *integrated*, *not integrated*, or *ongoing integration*. *Integrated* means that the project is part of Biopython's master branch; *not integrated* means that it is not; *ongoing integration* means that the mentor is still working with the student to integrate the project.

Table 1 shows that 5 out of 10 (50%) GSoC projects have been integrated into the Biopython master branch. As we illustrate in the following subsections, however, there are several positive outcomes of GSoC, regardless of whether the project becomes part of Biopython.

3.2 Learning New Skills

Our findings suggest that GSoC enables Biopython students to learn new software engineering skills. Prior to GSoC, some students had only limited software development experience (P1, P6, P11). Other students had no software development background at all; one student characterized GSoC as his *"introduction to unit testing, online APIs, and object-oriented programming"* (P6). A different student commented on the benefits of working with an established project:

> *"It was an immersion in a large code base with lots of practice reading others' code. It improved my coding skills and habits noticeably."* (P1)

Another student learned the steps required to submit a feature to an open source project using GitHub:

> *"…what I learned from my Summer of Code was when I really wanted to develop a new feature, basically you make a new branch of your repository. Then you work on it and then you submit a pull request."* (P11)

Besides helping students develop general software engineering skills, GSoC provides students with ways to learn about Biopython's code base as well as how to contribute to it. After his GSoC, one student realized that he could unify Biopython's existing parsers into an interface for writing phylogenetic trees in formats used by other bioinformatics tools (P8). Another student learned that Biopython lacked support for conducting systematic analyses of protein structures (P11). This student started a GSoC project the following summer (Project F) to provide this support.

Students also learned how to coordinate their code changes with other Biopython developers through the mailing list and pull requests (P4, P11). For example, one student recalled how he used the mailing list to discuss code changes that could impact parts of the codebase:

> *"…whenever I tried to change other people's code, I try to communicate. I try to tell them if I change their functions. And I try. And we have a dis-- we have a mailing list and try to discuss that, that we changed this, is it okay."* (P4)

Students were also mindful about testing their code. When describing new features that he planned to add to his GSoC project, one student told us:

> *"I also am preparing a new [pull request] as well, but it's not completely done yet. I need to write some mini tests for it."* (P11)

GSoC provides Biopython students with generally useful software engineering skills and helps them learn the particulars of the code base. Moreover, students recognize these skills as being important to their work.

3.3 Promoting Student Development

Both students and mentors agree that participating in GSoC positively impacts students' professional development. One mentor commented that a line on students' CV that shows that they went through a selective interview process is more impressive to potential employers than the average summer job (P3). Indeed, one student said that it was the most valuable output of her project (P1). Another student attributed his current job to the skills he learned in his year of GSoC (P4).

GSoC may also help students understand their own scientific interests. Proposing a GSoC project helped some students to define the research areas that they would pursue for their graduate degrees (P8, P11). Another student's GSoC experience helped him realize that his scientific interests were actually *"outside the scope of the Biopython library"* (P2). Although this student's code was never integrated into Biopython, he worked with his mentor to integrate it into another project for his master's thesis.

Mentoring can be far-ranging, with topics beyond programming for Biopython (P3, P7). As one mentor said, *"We talked a bit about [different topics], for example, career choices and sample applications for jobs, that kind of stuff"* (P3). Mentors were pleased to find that their former students entered into bioinformatics careers full time, became active contributors to Biopython, or both (P3). Moreover, training students to be comfortable contributing to open-source is an important part of GSoC mentorship, sometimes even ahead of project integration:

> *"…one thing I learned is to focus more on training than getting something accomplished for Biopython. I think of [GSoC] as a way to help people get involved with Biopython, work on a project, feel welcome in the community and understand the code base while learning how to build open source code. This changed from thinking of the major goal of GSoC as accomplishing a specific project."* (P7)

In addition to helping young scientists learn software development skills, GSoC contributes to Biopython students' personal growth. GSoC also plays a role in shaping their scientific interests. Mentors find these outcomes to be highly rewarding.

3.4 Facing Challenges Associated with the GSoC Process

Integrating new features, learning new skills, and promoting student development have been positive outcomes for Biopython. Our findings also indicate, however, that GSoC has presented challenges to the mentors.

Uncertainty of Mentoring Organization Eligibility. Google accepts a mentoring organization (i.e., a group running an active free/open source software project,), not the software project itself. Biopython therefore typically applies with the O|B|F, the mentoring organization for various open source bioinformatics projects. One frustration that mentors expressed was that there is no automatic eligibility for the organization. For example, in 2013, Google rejected O|B|F, despite its successful track record with GSoC since 2010. A Biopython mentor explained to us how this created uncertainty among mentors and students:

> *"So the last few years we were applying as a group as Bioinformatics-- Open Bioinformatics Foundation. And then suddenly this year they said, 'No, sorry. We don't want you.'… the BioRuby people got in touch with the Scientific Ruby Foundation to get accepted, and we put some students through that instead… but there was a period of quite a lot of confusion about whether or not we would be able to offer any students just this year. And for potential students, that's disconcerting as well, 'cause they're trying to plan their summer."* (P3)

Stress of Proposal Submission and Ranking. After Google accepts a mentoring organization, the student application period opens. In an iterative process, students write proposals and obtain feedback from potential mentors:

> *"And then what happens over sort of a two-week period is the mentor tries to get them to get it to be the best possible proposal they can have. So it's a lot of stuff like filling out more specifics in the proposal; making sure you have detailed timelines from week to week what you're going to accomplish."* (P7)

From the mentor's perspective, giving feedback is stressful because the mentor must balance producing a good quality proposal with giving equal attention to all applications:

> *"Then there's the whole writing the proposal stage, getting feedback...and so actually, that's really quite stressful for the students and for the mentors as well. When you're trying to help a student improve the proposal. Trying to be fair to all the students and not give it any one particular advantage."* (P3)

After the student proposal deadline passes, Google allocates to the mentoring organization a number of project slots. The mentoring organization reviews and ranks the student proposals. This process is often challenging, as members must make difficult decisions about which projects to mentor and the students who will lead them.

> *"And then as a group, you're ranking these people. You're going to affect, not just the next six months of their life, but potentially it's career-changing. That's kind of scary."* (P3)

A participant summarized the process of giving feedback and ranking proposals as a "real compressed period" with "tons of e-mails and discussions going on." (P7)

Mentor Un(availability). When ranking projects, mentors mentioned that the tie-breaker is mentor availability:

> *"And this year we didn't have very many volunteers for the Biopython side. So we actually had more students than we had mentors. And we had to turn students away because we didn't have enough people to look after them. So when you have two students who want to do similar or different projects, but the one's not mentored, we have to just pick the stronger student....we probably could have one or two more students if we had the mentors."* (P3)

Mentor availability depends on the amount of time mentors have. Many open-source developers contribute to projects in their spare time, not as part of their regular jobs. Becoming a GSoC mentor is yet another commitment. As one former Biopython mentor explained to us:

> *"I have a family and I don't have the time to do that. And it's like you have a certain amount of hours for open source stuff; and you're like why? And I prefer to code I guess. You know?"* (P7)

In sum, although GSoC has benefitted Biopython in several ways, it is not without its challenges. Mentors expressed concerns with the uncertainty of their organization's eligibility, the stress of creating and ranking proposals, and the availability of mentors as a limiting factor.

4. Conclusion

We see three positive outcomes of GSoC in Biopython: the addition of new features to the Biopython codebase, training, and personal development. The impact of these outcomes in Biopython suggests that GSoC is a promising solution to address critical issues of scientific software sustainability.

The first outcome emphasizes the value of integration of new features into the Biopython codebase. As Table 1 illustrated, these features may extend existing functionality, add additional functionality, or some combination of both. A successful GSoC project not only improves the software, but also benefits scientists in generating new knowledge, and can lead to traditional publication. For instance, Bio. Phylo [8] grew out of the work on Project B (see **Table 1**).

The second outcome reflects the utility of student training. According to mentors, this outcome can be just as important as the integration of GSoC projects into the codebase. Training may also help provide scientists with the software development expertise that they sorely lack [4, 7]. In addition, the perceived impact on student's professional development may offset some of the aforementioned disincentives associated with developing scientific software [5, 6].

The third outcome indicates the importance of student development during GSoC. We observed that, in the Biopython community, mentors sometimes focused more on helping students with their academic progress and future careers than project integration. Although some projects are not integrated in the Biopython codebase (see Table 1), this could benefit scientific communities by producing new generations of scientific software developers with development experience and skills. Moreover, we found that many students have strong relationships with their mentors due to their intensive support beyond GSoC-related matters. These relationships enhance students' commitment and become a substantial factor that causes students to come back to contribute after GSoC ends, especially when they receive requests from their mentors.

In order to gain the benefits of GSoC, however, other open-source scientific software communities should expect to have an adequate supply of dedicated volunteers with enough spare time to mentor students, and tools to support community discussion around project ranking and selection. One way for communities to address mentor availability might be to provide a pathway for students interested in research careers to become mentors the following year, with publication as a possible additional outcome of the project. Such

a pathway might be explained to students as an extension of their participation, in which they can gain strong leadership skills, and the ability to draw the best from their students in order to achieve their overall research agendas. Students who have recently completed GSoC projects are in a good position to become mentors, since they can draw on the positive and negative aspects of their own experiences, and are familiar with the GSoC process and practices.

To support the proposal ranking process, we suggest that communities use a tool that provides a variation of the popular reddit[3] format where community members can easily organize, discuss, and up/down vote proposed projects in a centralized location. When selecting students for these projects, mentors might weigh accomplishing a specific project against contributing to students' intellectual and professional development. If the goal is to accomplish a specific project, mentors might choose a student who has a history of participation in the community, or who is a skilled developer. If the goal is to substantially contribute to a student's development, mentors might choose a student who is less skilled but is interested in research as a career, and sees scientific software development as a skill they wish to explore and cultivate.

Future work should of course test the representativeness of our results in other scientific software communities, but nuances in the impact of other types of interventions should be studied as well. For example, in GSoC, the intensive mentoring process facilitates the creation of strong ties between mentor and student, and contributes to the student's training and development. In contrast, the social events at hackathons may expose participants to more networking opportunities but do less to enrich individuals' skillsets. Both outcomes—training and relationship building—are important components of community building. Developers, scientists, and funding agencies would do well to understand the benefits and limitations that each type of engagement brings to bear on the development and sustainability of scientific software.

Acknowledgements

This work was supported by a grant from the Alfred P. Sloan Foundation, NSF awards 0943168, 1111750, and 1064209, and the Google Open Source Programs Office. We thank the Biopython developers for collaborating with us.

Notes

[1] https://developers.google.com/open-source/soc/
[2] https://github.com/biopython/biopython
[3] http://www.reddit.com

References

1. **Cock, P** Opening up NCBI BLAST? Available at: http://blastedbio.blogspot.co.uk/2011/08/opening-up-ncbi-blast.html [Last accessed 23 August 2013].
2. **Cock, P J A, Antao, T, Chang, J T, Chapman, B A, Cox, C J, Dalke, A, Friedberg, I, Hamelryck, T, Kauff, F, Wilczynski, B** and **de Hoon, M J L** 2009 Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics,* 25(11): 1422–1423. DOI: http://dx.doi.org/10.1093/bioinformatics/btp163
3. **Dedoose** 2013 *Dedoose Version 4.5, web application for managing, analyzing, and presenting qualitative and mixed method research data*. Los Angeles, CA: SocioCultural Research Consultants, LLC. Available at: http://www.dedoose.com/
4. **Hatton, L** and **Roberts, A** 1994 How Accurate is Scientific Software? *Trans. Soft Eng.* 20(10): 785–797. DOI: http://dx.doi.org/10.1109/32.328993
5. **Howison, J** and **Herbsleb, J D** 2011 Scientific Software Production: Incentives and Collaboration. In: Proceedings of the 2011 Conference on Computer Supported Cooperative Work, Hangzhou, China, CSCW'11, pp. 513–522.
6. **Howison, J** and **Herbsleb, J D** 2013 Incentives and Integration in Scientific Software Production. In: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, San Antonio, Texas, CSCW'13, pp. 459–470.
7. **Segal, J** 2009 Some Challenges Facing Software Engineers Developing Software for Scientists. In: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Scientists and Engineers, Vancouver, BC, SECSE '09, pp. 9–14.
8. **Talevich, E, Invergo, B M, Cock, P J A** and **Chapman, B A** 2012 Bio.Phylo: A unified toolkit for processing, analyzing and visualizing phylogenetic trees in Biopython. *BMC Bioinformatics,* 13(1): 209. DOI: http://dx.doi.org/10.1186/1471-2105-13-209
9. **Wren, J** 2008 URL Decay in MEDLINE—a 4-Year Follow-up Study. *Bioinformatics,* 24(11): 1381–1385. DOI: http://dx.doi.org/10.1093/bioinformatics/btn127

How to cite this article: Trainer, E H, Chaihirunkarn, C and Herbsleb, J D 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, pp. 1-5, DOI: http://dx.doi.org/10.5334/jors.bc

Published: 9 July 2014

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS