How Should We Tailor Software Development Practices for the Scientific Domain?

Archer L. Batcheller
University of Michigan
archerb@umich.edu

ABSTRACT
This position paper is about the process of creating software that enables new scientific practices. I take the position that creating software for scientists is different from creating software in other contexts, and that it is important to understand the modes and particularities of software engineering for scientists. I focus on the implications for requirements analysis, where various stakeholders in the project seek to negotiate and communicate their desires. I believe that our methodology for designing software and engaging stakeholders needs to vary with the specific project.

Author Keywords
Cyberinfrastructure, infrastructure, software engineering, participatory design.

ACM Classification Keywords
K.4.3 [Organizational Impacts]: Computer-supported collaborative work; D.2 [Software Engineering]; J.2 [Physical Sciences and Engineering]: Earth and atmospheric sciences

INTRODUCTION
Software to support science work is produced in different ways and for different purposes. Sometimes it's a top-down product of visionaries in the field. Other times it comes about as a result of someone producing a local solution that ends up being a big hit. The software can support monitoring and generating data, or support modeling efforts, or facilitate storage and access to scientific memory – either data or publications. Or software can facilitate connections and collaborations between scientists. To support these diverse purposes and origins, a number of different software development methodologies are employed. Here we investigate some of the ways that software is being written for climate science and citizen science work. Individuals working on this strategically choose diverse design methodologies – varying both among the various efforts and differing from software development in non-science domains. Better understanding the choices and implications of design methodologies is an important part of producing successful technologies.

Domain scientists writing code
One mode for developing software to support scientific work is when scientists within a domain take on their own software development. This may entail translating a mathematical model into computer code, cleaning and manipulating data, writing software for archiving observations, or a variety of other large and small projects. Such software endeavors may be side projects or attended to only as necessary, since rewards for scientists revolve around research production and publications in particular.

Computer scientists writing code
At other times, computer scientists may partner with researchers within a specific domain to help develop software. This typically involves incorporating computer scientists' research goals into the project to address some particularly technically challenging problem. Yet as both Weedman and Lawrence have each pointed out, this often leads to tension within the project as computer scientists receive little reward for fine-tuning and debugging software to make it the production quality that the domain scientists need [7, 11].

Software engineers writing code
Recent attention to developing cyberinfrastructure to support scientific work has resulted in more direct and focused work on building software. In practice, this often means assigning or hiring software developers to be responsible for writing the necessary code. These workers may have more professional training in managing software projects and are paid to do the tough bug-hunting work necessary when trying to produce production quality systems. This separation of roles, such that scientists are not responsible for (as much) software development work represents a deliberate strategy to split both the technical architecture and the work. Non-scientists can be responsible for the underlying architecture, which scientists may then build upon as they assemble tools and do tasks. Scientists may still write some lesser amount of code, which is then supported by the underlying infrastructure (or middleware).

As Conway has long pointed out, the communication structure of a team is reflected in the design of the architecture [3]. By splitting the architecture of scientific software, we hope that scientists don't have to communicate with or be part of a large chunk of the software development efforts.

However, as Segal has highlighted, software engineers do still need to coordinate with scientists about the infrastructural tools being developed and the interaction between their two different "cultures" can present a challenge [10]. Cultural differences are only compounded by affects of distributed teams and inadequate methodological tools for cyberinfrastructure projects [12].

DEVELOPING FOR SCIENTISTS

Software development for scientific cyberinfrastructure will face many of the challenges typical for both scientific collaboration [8] and for software development [9]. But it also face issues specific to the intersection of software development for science. A recent series of International Conference on Software Engineering workshops on software engineering for high-performance computing and computational science and engineering have highlighted these special challenges [1-2]. For instance, while requirements for all software projects shift, requirements changes are an inherent part of discovery as science explores different possibilities. Other ways that software development in science is unique include:

- A focus on optimization, potentially impacting whether a program is usable
- "Kleenex code" where one correct run is enough
- Software that needs to implement complex mathematical models, and use complex hardware
- Quality assurance is both important and especially challenging when correct outcomes may not be known beforehand
- Scientific funding depends on short-term grants

SOFTWARE FOR CLIMATE SCIENTISTS

The following projects that I am studying describe themselves as developing infrastructure for scientists to use. Their immediate goals are focused around engineering outcomes, not scientific discoveries. In positioning themselves as teams to develop infrastructure, they have deliberately separated a set of concerns away from scientists for which they will take ownership. This is intended to shift software development burden away from scientists to professional software developers. So scientists end up being more users than end-user developers like we see in other areas of software innovation in science. This makes it more straightforward to compare dynamics of software development in these teams to teams in industry, where there is also often a user-developer dichotomy.

My research inquiry focuses around the software requirements engineering of the following groups, detailing the interaction between scientists and developers. I am particularly interested in the methodological approaches that the project leaders and developers use, and how those compare to ones typically used in industry.

Earth System Modeling Framework

The Earth System Modeling Framework (ESMF) group is producing a single, open source software product managed in one CVS source code repository. An advisory board and executive committee establish project priorities, and a change review board meets quarterly to determine the exact customer bug and feature requests that should be addressed and when. It is led by a project manager, who oversees approximately ten full-time developers. Developers are scattered geographically, and may be based at other institutions or from their homes as contractors.

The ESMF organizers' goal is an ambitious one, to "unite climate, weather and data assimilation groups under a common framework" and to "fundamentally change the culture of Earth system modeling" [4]. ESMF aspires to be a project, a product, and a standard used by the community [6]. The ESMF is an example of an effort to deliberately allocate some software development to a group of software engineers, with the hope that it can be reused and serve as a building block for modeling groups that write components that are to be linked together under the framework.

The ESMF keeps track of its users using customer management software, and provides support and feature request services for the scientist users. The team is conscious of a need to satisfy modeling groups to lead to further adoption and momentum for the project.

Earth System Grid

The Earth System Grid (ESG) group is led by several principal investigators who make up a small executive board. There are a set of tools that facilitate the sharing of datasets, coordinated via a web portal. These portals can be used to access data, and the various tools simplify the process of discovering content, authorizing access, and downloading data. ESG developers are also scattered geographically, but tend to have clustered teams at different institutions who are all working on the same or related tools.

ESG's efforts at adoption were bolstered by the designation of one of its sites (Lawrence Livermore National Laboratory's Program for Climate Model Diagnosis and Intercomparison) as a host of climate model data for the Intergovernmental Panel on Climate Change (IPCC)'s 4th Assessment Report. With this mark of, ESG established a foothold as a major data distribution system, serving more than 130 TB to about 4,000 different users [5].

The ESG team has been deliberate about trying to get usability feedback from scientists, at times recognizing that

they need to pay scientists as part of the project to ensure they will be adequately involved to give quality feedback. As the project has grown, they have also recognized that they need to mature in robustness too, and have hired developers with industry experience to enhance their product as one that is of production quality.

Earth System Curator

The Earth System Curator (ESC) project is an effort to advance work on metadata for climate model datasets. There is no single code repository resulting from this project. Developers are also scattered geographically, and different institutions may or may not ultimately contribute their code back to a single resource. For instance, one institution may use its ESC funds to further development of metadata tools for itself. The most concrete products will feed back into the ESG and ESMF code repositories, improving their capacities with respect to metadata.

The ESC has faced some of the typical challenges of involving computer scientists, where computer scientists do good development work but are motivated by their own set of research questions. Answering those research questions is usually possible by building prototypes, not concrete products. This project may end up pushing the frontier of technological possibilities, which can then be selectively implemented by more permanent projects.

PARTICIPATORY DESIGN AND CITIZEN SCIENCE

I have also been involved in the development of a web portal for lay people to access lake data. The Lake Sunapee Protective Association maintains a buoy which reports real-time data about air and water temperatures, water oxygen levels, wind speed, and solar radiation. The web portal is intended to provide a resource for those interested in the lake, and to help them to think about the science related to the lake. This sort of engagement of lay persons is often termed "citizen science."

To design for this special case of citizen scientists, our team used a participatory design methodology. Lake association members worked with the design team in a series of 3-4 day-long workshops – quite a few for a relatively small project. Philosophically, broadening participation in science is well aligned with the goal of opening participation in the design process. As we prepare to launch the web portal, we suspect that this design approach has helped to gather buy-in to the product, which they helped create, and has helped give the organization both a time and structure to start thinking about how they can integrate the web portal into their existing activities.

It is important to have a range of different design methodologies available, and I believe that participatory design has been a particularly good fit for this project. I suspect that it would entail too much commitment for professional scientists, who may not be willing to devote the time necessary to be engaged in the design work regularly.

DISCUSSION POINTS

Of course I am interested in feedback about my work, but there are a few items that I am particularly interested in discussing at the workshop on "The Changing Dynamics of Scientific Collaborations." These are primarily focused around the methodology of designing systems for scientists.

1. Which roles and types of people are involved in innovation around scientific collaboration? How do scientists relate to developers and negotiate product specifications?
2. How is the context of software development for science similar and different from industry and open source?
3. What design methodologies are appropriate for scientific cyberinfrastructure? How should methodologies vary for the targeted project?

REFERENCES

1. Carver, J.C. Third international workshop on Software Engineering for High Performance Computing Applications. In *Proc. of the 29th International Conf. on Software Engineering*, Minneapolis, 2007.
2. Carver, J.C. The second international workshop on Software Engineering for Computational Science and Engineering. In *Proc. of the 31st International Conf. on Software Engineering*, Vancouver, 2009.
3. Conway, M.E. How Do Committees Invent? *Datamation*, 14, 5 (1968), 28-31.
4. da Silva, A. and et al. *Future Directions for the Earth System Modeling Framework*. 2004.
5. ESG. *ESG II Final Report: Turning Climate Datasets into Community Resources*. U.S. Department of Energy Office of Science, 2006.
6. ESMF. *ESMF Draft Project Plan 2005-2010*. 2005.
7. Lawrence, K.A. Walking the Tightrope: The Balancing Acts of a Large e-Research Project. *Computer Supported Cooperative Work*, 15, 4 (2006), 385-411.
8. Olson, J.S., Hofer, E.C., Bos, N., Zimmerman, A., Olson, G.M., Cooney, D. and Faniel, I. *A Theory of Remote Scientific Collaboration*. Scientific Collaboration on the Internet, G. M. Olson, A. Zimmerman and N. Bos. MIT Press, Cambridge, 2008.
9. Sangwan, R., Mullick, N., Bass, M., Paulish, D. and Kazmeier, J. *Global software development handbook*. CRC Press, 2006.
10. Segal, J. When Software Engineers Met Research Scientists: A Case Study. *Empirical Software Engineering*, 10, 4 (2005), 517-536.
11. Weedman, J. The Structure of Incentive: Design and Client Roles in Application-Oriented Research. *Science Technology & Human Values*, 23, 3 (1998), 315-345.
12. Zimmerman, A. and Nardi, B.A. Whither or whether HCI: requirements analysis for multi-sited, multi-user cyberinfrastructures. In *Proc. of the CHI '06 extended abstracts on Human factors in computing systems*. ACM Press, Montreal, 2006.