Sustainable & Productive: Improving Incentives for Quality Software

Michael A. Heroux
Sandia National Laboratories, Albuquerque, NM, E-mail:maherou@sandia.gov
Saint John's University, Collegeville, MN, E-mail:mheroux@csbsju.edu

***Abstract*—Computational Science and Engineering (CSE) software can benefit substantially from an explicit focus on quality improvement. This is especially true as we face increased demands in both modeling and software complexities. At the same time, just desiring improved quality is not sufficient. We must work with the entities that provide CSE research teams with publication venues, funding, and professional recognition in order to increase incentives for improved software quality. In fact, software quality is precisely calibrated to the expectations, explicit and implicit, set by these entities. We will see broad improvements in sustainability and productivity only when publishers, funding agencies and employers raise their expectations for software quality. CSE software community leaders, those who are in a position to inform and influence these entities, have a unique opportunity to broadly and positively impact software quality by working to establish incentives that will spur creative and novel approaches to improve developer productivity and software sustainability.**

I. Introduction

Funding agencies, publishers and employers play an essential role in determining expected behavior from the computational science and engineering (CSE) research community. Publishers use a peer review process to ensure that published content is accurate, complete and relevant. Funding agencies place expectations on how research is conducted, requiring, for example, that research teams perform work within certain ethical norms. Employers determine promotion, rank and tenure and other professional recognition. The value, both implicit and explicit, that these entities place on software quality plays a crucial role in the thousands of decisions that CSE community members make on a daily basis regarding the real value of software quality.

CSE software teams have strong desires to improve developer productivity and software sustainability. At the same time, competition to publish, receive funding, satisfy project goals and attain professional promotion and recognition means that the amount of time and resources they can apply to improve quality is determined by how much value they can obtain from the investment as part of this competition. In order to increase software quality[1], competitive elements must be added that are correlated with increased quality.

Fortunately, direct evaluation of software artifacts by rank and tenure committees, program managers and publishers is not required. There are other metrics that, when assessed, will provide incentive for increasing quality:

- **Funding agencies:** For several years, funding agencies have requested the submission of data management plans [1], [2]. More recently, a first-of-a-kind software productivity and sustainability plan was included in a US Department of Energy Funding Opportunity Announcement. This new required element is a simple request for information, asking proposal teams to describe how they produce and sustain their software products.
- **Publishers:** Reproducibility of scientific results has become a major concern for many publishers. Expectations for reproducibility are increasing, as are the number of journals and conference proceedings that offer artifact and results reviews. Any increased expectations for reproducible computational results have direct impact on the importance of higher quality software processes, documentation, source code management and more; all of which have a strong positive impact on programmer productivity and software sustainability when measured over a sufficiently long time span.
- **Employers:** Employers can provide incentives recognizing those whose work results in high-quality software. The exact metrics need to be adapted depending on the situation. Metrics for industry, research labs and universities will necessarily be distinct, and can either be observed directly (most feasible for industry) or indirectly (such as the acceptance of a peer-reviewed paper to a software journal) for research labs and universities.

II. Creating Incentives for Sustainability and Productivity

Expectations from CSE software clients provide a means to give intrinsic value to virtuous behaviors that are otherwise

[1]Many articles have been written on the characteristics of software quality. The ISO 9126-1 software quality model definition (see http://www.sqa.net/iso9126.html for details) lists six main characteristics. For our purposes, we do not analyze correlations between specific quality characteristics and the incentives we discuss. Instead we argue that improving incentives will have a positive impact on all quality characteristics since the incentives represent a holistic approach to improvement.

difficult to reward. CSE software teams certainly want to do a good job of being productive and producing sustainable software tools. But the drive to produce science results on a short time scale often leads to shortcuts and lack of investment up front that is necessary for sustainable software and long-term productivity. Given the demands to provide a computational answer as soon as possible, especially in a competitive environment, other activities such as careful design, efficient implementation and thorough software testing environments are often compromised.

The connection between these clients and sponsors (publishers, funding agencies and employers) and science teams who use computation as part of their scientific endeavors can and must be leveraged to provide intrinsic value to other activities that can improve sustainability and productivity.

A. Funding Agencies

Funding agencies have typically placed very high value on scientific productivity and impact. In principle this emphasis should provide strong incentive to produce high quality software. However, often the time cycles for producing science results are short, and the competitive drive to produce results makes software investments that have longer time cycles out of scope. This reality was recognized by the US National Science Foundation and spurred its development of the SI2 program [3]. More recently the US Department of Energy (DOE) has developed several funding opportunities that explicitly focus on software productivity and sustainability:

- The 2014 Funding Opportunity Announcement (FOA) for Scientific Software Developer Productivity, which resulted in the establishment of the IDEAS Project [4]. While many DOE projects use funds to improve software quality, the work is always done as part of achieving another goal. The IDEAS project is a first-of-a-kind effort that enables dedicated research funding for software methodologies as a pursuit in its own right.
- The DOE Biological and Environment Research (BER) Office FOA on the Energy-Water-Land Nexus from February 2016 [5], included a first-of-a-kind requirement for proposals to contain a software productivity and sustainability plan.
- The BER Accelerated Climate Modeling for Energy (ACME) Software Engineering Initiative is a large project whose express goal is to improve the quality of the ACME source code base, train ACME scientists in more effective software development, and introduce improve software processes and tools into the ACME project.

B. Publishers

At first glance, the role of publishers in improving software quality may not be apparent, but by raising expectations for independent reproducibility of computational results, publishers provide one of the strongest incentives for CSE software teams to invest in new and effective software quality methodologies. Furthermore, the community incentive at the present time for improving reproducibility is extremely high [6], [7].

Reproducibility of scientific results has become a major concern for many publishers. Numerous highly visible articles [8], [9], [10], [11] have highlighted the poor result of reproducibility efforts. Some community members [12], [13] argue that openness and transparency are important ways to provide incentives for scientists to improve reproducibility. Such a focus also placed increased value on the quality of software artifacts and processes. Other articles argue that repeatability as a core value is the fundamental issue in CSE publications [14], [15].

The ACM Transactions on Mathematical Software (TOMS) initiated a Replicated Computational Results (RCR) review process during which a dedicated reviewer is tasked with replicating the computational results of a submitted manuscript [16], [17], [18]. One of the first author responses when preparing for the RCR review was a remark that the version of software used to generate the computational results for the paper was tagged so that it could be clearly identified for the RCR review, as well as any time in the future.

C. Employers

Industrial employers already have a reward structure in place that, at least indirectly and evidenced by the large volume of industry-focused software literature and highly-sought speakers, recognizes the value of software quality. This is less true the closer a software business is to the scientific and engineering domains. From the author's experience working with commercial, scientific and engineering software companies, the majority have no explicit software lifecycle model and seldom give recognition, except through informal means, to developers who produce higher quality software products. Research laboratories and university software teams place even less emphasis on employee recognition for software quality. In fact, there is often a negative opinion about staff members who spend too much time working on software [14], with more recognition going to staff who use software to do research but spend little time producing or reviewing software products.

Development of employee incentives for quality software can be challenging. Any reward based on direct software metrics can lead to skewed behavior that may actually be counter-productive. For example, rewarding a developer for having few defects can lead to low code generation rates for fear of introducing a defect, avoiding challenging programming assignments or similar behaviors that reduce overall productivity. Instead, employers can look to recognition from funding agencies and publishers, and assess the overall impact of a faculty member's or research scientist's software contributions, with the same regard as for other publication and professional activities.

III. A Time of Disruptive Change

The CSE community is presently in an era of disruptive changes. Similar to the time period when we transitioned from sequential and vector computers to distributed memory parallel computers, system architecture is changing, driven by the stagnation of hardware clock speeds, resulting in exponential increases in available concurrency. Performance sensitive applications must expose proportional levels of concurrency in order to stay on the new commodity curves of threading and vectorization scalability.

In addition, the tremendous growth in performance potential leads to increased opportunities for multi-scale and multi-physics coupling. Coupling of previously independent codes demands increased collaboration across historical separate efforts. Projects become more inter-disciplinary, distributed (since areas of expertise are typically not all in one group or geographic location), and larger.

All of these disruptions can be better managed and exploited by having more deliberate efforts to improve software sustainability and productivity. In fact, now is the best time to invest in software quality. The stakes are higher than ever.

IV. Toward Sustainable and Productive

Numerous studies have shown that activities such as code review and test driven development [19] are universally valuable. Furthermore, explicit life cycle models [20], including models for refactoring legacy codes [21], [22], are also extremely beneficial, especially for larger teams. Productivity models [23], which attempt to characterize and measure the value and cost attributes of a project, can also be extremely helpful, especially in the face of disruptive changes in computing platforms and application requirements.

The CSE software community can benefit greatly from an improved focus on sustainability and productivity, especially at this time of increased complexity and disruptive change. There are many valuable methodologies, tools and best practices that the CSE community can explore, adapt, adopt and develop to improve developer productivity and software sustainability. However, for many teams the incentive for investment is not strong enough when compared to other demands; other priorities rank higher.

Funding agencies can provide incentives by explicitly requesting software productivity and sustainability plans in proposals, and by funding projects that pursue software methodologies research and a fundamental activity, not just as part of pursuing another goal. Publishers can play a critical role by increasing expectations for reproducibility, and transparency of software processes used to generate computational results. Employers, especially research laboratories and universities, must start to explicitly value software contributions with the same respect as other research output, and give recognition for faculty and staff who receive publication and funding agency recognition for quality software.

Funding agencies, publishers and employers can and must play a key and truly unique role in improving developer productivity and software sustainability by increasing quality expectations. In fact, it is not too strong to say that software quality will be calibrated precisely to the expectations of these entities. Because of the important roles these entities play in providing incentives for improved software quality, CSE software leaders–those who are engaged in editorial work, informing and shaping funding policy and establishing employer compensation and recognition systems–have a unique opportunity to influence progress. If we want to improve software quality, we must change expectations from funding agencies, publishers and employers.

Acknowledgment

Sandia is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

The author thanks the United States Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) and Biological and Environmental Research (BER) Programs for IDEAS Project funding.

References

[1] US National Science Foundation, "Data Management Guidance for CISE Proposals and Awards," http://www.nsf.gov/cise/cise_dmp.jsp.

[2] U. S. Department of Energy Office of Science Office of Scientific Computing Research, "Statement on Digital Data Management," http://science.energy.gov/funding-opportunities/digital-data-management/.

[3] US National Science Foundation, "Software Infrastructure for Sustained Innovation (SI2: SSE & SSI)," http://www.nsf.gov/pubs/2016/nsf16532/nsf16532.htm.

[4] "The IDEAS Project," http://ideas-productivity.org, 2016.

[5] U. S. Department of Energy Office of Science Office of Biological and Environmental Research, "Regional and Global Climate Modeling and Integrated Assessment Research: An Integration Framework for Multi-Model, U.S. Regional Climate Evaluation that Incorporates Local Human Influences for Research at the Energy-Water-Land Nexus," http://science.energy.gov/~/media/grants/pdf/foas/2016/SC_FOA_0001531.pdf, Funding Opportunity Number: DE-FOA-0001531.

[6] American Association for the Advancement of Science, "III Arnold workshop: Modeling and code," http://www.aaas.org/event/iii-arnold-workshop-modeling-and-code.

[7] Supercomputing 2016, "Student Cluster Competition reproducibility initiative winner," http://sc16.supercomputing.org/studentssc/scc-reproducibility-initiative-winner/.

[8] C. G. Begley, G. Robertson, K. Kaitin, K. Asadullah, L. Ellis, P. Sharp, and S. Begley, "In cancer science, many discoveries don't hold up — Reuters," 2014. [Online]. Available: http://www.reuters.com/article/2012/03/28/us-science-cancer-idUSBRE82R12P20120328

[9] D. H. Bailey, "Fooling the masses: Reproducibility in high-performance computing," 2014. [Online]. Available: https://www.xsede.org/documents/659353/703287/xsede14_bailey.pdf

[10] J. Arrowsmith, "Trial watch: Phase II failures: 2008–2010," *Nat Rev Drug Discov*, vol. 10, no. 5, pp. 328–329, May 2011. [Online]. Available: http://dx.doi.org/10.1038/nrd3439

[11] B. Owens, "Reliability of 'new drug target' claims called into question : Nature News Blog," 2011. [Online]. Available: http://blogs.nature.com/news/2011/09/reliability_of_new_drug_target.html

[12] D. H. Bailey and J. M. Borwein, "Set the Default to "Open": Reproducible Science in the Computer Age—David H. Bailey," 2014. [Online]. Available: http://www.huffingtonpost.com/david-h-bailey/set-the-default-to-open-r_b_2635850.html

[13] V. Stodden, D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider, and W. Stein, "Setting the Default to Reproducible: Reproducibility in Computational and Experimental Mathematics," 2013. [Online]. Available: http://icerm.brown.edu/html/programs/topical/tw12_5_rcem/icerm_report.pdf

[14] C. Collberg and T. A. Proebsting, "Repeatability in computer systems research," *Commun. ACM*, vol. 59, no. 3, pp. 62–69, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2812803

[15] S. Krishnamurthi and J. Vitek, "The real software crisis: Repeatability as a core value," *Commun. ACM*, vol. 58, no. 3, pp. 34–36, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2658987

[16] M. A. Heroux, “Editorial: ACM TOMS replicated computational results initiative,” *ACM Trans. Math. Softw.*, vol. 41, no. 3, pp. 13:1–13:5, Jun. 2015. [Online]. Available: http://doi.acm.org/10.1145/2743015

[17] J. M. Willenbring, “Replicated computational results (RCR) report for BLIS: A framework for rapidly instantiating BLASx functionality,” *ACM Trans. Math. Softw.*, vol. 41, no. 3, Sep. 2015.

[18] F. G. Van Zee and R. A. van de Geijn, “Blis: A framework for rapidly instantiating blas functionality,” *ACM Trans. Math. Softw.*, vol. 41, no. 3, Sep. 2015.

[19] K. Beck, *Test Driven Development*. Addison Wesley, 2003.

[20] R. A. Bartlett, M. A. Heroux, and J. M. Willenbring, “Overview of the TriBITS lifecycle model: A lean/agile software lifecycle model for research-based computational science and engineering software,” in *E-Science (e-Science), 2012 IEEE 8th International Conference on*, Oct 2012, pp. 1–8.

[21] M. Feathers, *Working Effectively with Legacy Code*. Addison Wesley, 2005.

[22] M. Fowler, *Refactoring (Improving the Design of Existing Code)*. Addison Wesley, 1999.

[23] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, *Goal Question Metric (GQM) Approach*. John Wiley and Sons, Inc., 2002.