Chapter Proposal for Software Engineering for Science

Tom Crick[1], Benjamin A. Hall[2] and Samin Ishtiaq[3]

[1]Department of Computing & Information Systems, Cardiff Metropolitan University, UK
[2]MRC Cancer Unit, University of Cambridge, UK
[3]Microsoft Research Cambridge, UK
[1]tcrick@cardiffmet.ac.uk
[2]bh418@mrc-cu.cam.ac.uk
[3]samin.ishtiaq@microsoft.com

Abstract

Scientific software development represents a borderland in the explosion of code. It combines problems and people in a unique way; the scientists involved are not programmers, and do not typically have the skill sets and broader awareness of standard approaches to ensure code reliability. Fixing this is however not just issue of education or culture; in contrast to the shifting specifications an industry programmer may face, the specifications for scientific software may simply not exist. People will approach problems from multiple angles, effectively panning for scientific gold, unsure of their tools and reliant on their own judgement. The fundamental problem — not knowing what the "right" answer should be, whilst attempting to make reproducible discoveries — requires a modified approach. Here we detail the pressures and problems of working in this rarefied environment, and highlight lessons which are valuable for both scientific and software cultures.

Outline

In this chapter, we will discuss not only the issues which may arise in interdisciplinary work but also to draw out the characteristics of research scientific software development and how it is distinguished from conventional software engineering. We will highlight the issues which arise that are unique to research software and how we have encountered and (sometimes) successfully overcome them. We will draw from our wide range of experiences, but specifically focusing on two key scientific software projects which we have been involved with: the BioModelAnalyzer[1] [1]; and the more recent, related Cellular Dynamics Engine [2]. We intend to cover specifically:

- The challenges of testing in an environment where behaviours are not known *a priori*;
- Balancing scientific requirements (such as implementing alternative approaches rapidly) against common engineering needs of code optimisation, readability, etc;
- user driven discovery: the unexpected identification of scientific/algorithmic issues;
- Issues encountered when separating models and code;
- The linking of *in vivo* and *in silico* work flows;
- Issues encountered in maintaining a scientific codebase in a diverse, cross-disciplinary project;
- Issues encountered in maintaining and extending a model database in a diverse, cross-disciplinary project;
- User interface development to help in cross-disciplinary work;

[1]http://biomodelanalyzer.research.microsoft.com/

- Practices that we used but should be more widely adopted in scientific environments, for example: version control, unit testing, continuous integration, code review, robust documentation, etc.

We envisage this chapter sitting in *Section I: Examples of the Application of Traditional Software Engineering Practices to Science*, as it will describe generalisable experiences with applying modern software engineering practices to the development, particularly, of scientific software. We will illustrate our narrative with appropriate case studies and real world examples.

Authors

The three authors have wide experience of scientific software engineering, from real-world tool development across a number of application domains, through to associated policy. They have previously published work in this space, focusing on reproducibility and sharing of scientific software models [3, 4], as well as proposing e-infrastructure and workflow models for encouraging and supporting reproducibility in computational science and engineering[2].

Dr Tom Crick[3] is a Senior Lecturer in Computing Science at Cardiff Metropolitan University, having completed his PhD and post-doctoral research at the University of Bath. His research interests cut across computational science: knowledge representation and reasoning, optimisation, big data, social network analysis and high performance computing. He is the Nesta Data Science Fellow, a 2014 Fellow of the Software Sustainability Institute (EPSRC) and a member of *HiPEAC*, the European FP7 Network of Excellence on High Performance and Embedded Architecture and Compilation.

Dr Benjamin A. Hall[4] is a Royal Society University Research Fellow, developing hybrid and formal models of carcinogenesis and biological signalling at the MRC Cancer Unit, University of Cambridge. He previously worked at Microsoft Research Cambridge (on BioModelAnalyzer with Jasmine Fisher), UCL and the University of Oxford. As part of his role at Oxford, he was one of two Apple Laureates, awarded by Apple and the Oxford Supercomputing Centre for the project *A biomolecular simulation pipeline*. Benjamin has an MBiochem and DPhil from the University of Oxford.

Dr Samin Ishtiaq[5] is Principal Engineer in the Programming Principles and Tools group at Microsoft Research Cambridge. He currently works on the SLAyer (Separation Logic-based memory safety for C programs), TERMINATOR (program termination) and BMA projects. Samin joined MSR in April 2008. Before that (2000-2008), he worked in CPU modelling and verification at ARM, helping to tape-out the Cortex A8, Cortex M3 and SC300 processors, and the AMBA bus protocol checker. Samin has an MEng from Imperial and a PhD in dependent type theory from Queen Mary.

References

[1] David Benque, Sam Bourton, Caitlin Cockerton, Byron Cook, Jasmin Fisher, Samin Ishtiaq, Nir Piterman, Alex Taylor, and Moshe Y Vardi. BMA: visual tool for modeling and analyzing biological networks. In *Proceedings of the 24th International Conference on Computer Aided Verification (CAV 2012)*, volume 7358 of *Lecture Notes in Computer Science*, pages 686–692. Springer, 2012.

[2] Benjamin A. Hall, Nir Piterman, Alex Hajnal, and Jasmin Fisher. Emergent Behaviours of Stem Cells in Organogenesis Demonstrated by Hybrid Modelling. *Biophysical Journal*, 108(2):365a, 2015.

[3] Tom Crick, Benjamin A. Hall, and Samin Ishtiaq. "Can I Implement Your Algorithm?": A Model for Reproducible Research Software. In *Proceedings of 2nd International Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2)*, 2014.

[4] Tom Crick, Benjamin A. Hall, Samin Ishtiaq, and Kenji Takeda. "Share and Enjoy": Publishing Useful (and Usable) Scientific Models. In *Proceedings of 1st International Workshop on Recomputability*, 2014.

[2] http://arxiv.org/a/crick_t_1.html
[3] http://drtomcrick.com
[4] http://www.mrc-cu.cam.ac.uk/hall.html
[5] http://research.microsoft.com/en-us/people/sishtiaq/