Developing Scientific Software, Part 2

Judith Segal, Open University • Chris Morris, Science and Technology Facilities Council

This special issue on developing scientific software is a continuation of one in the July/August 2008 issue. The earlier issue featured six articles, three of which were case studies narrating particular “war stories.” The other three focused on characterizations: of the scientists’ perceptions and management of software development risk, of the high-performance-computing community, and of workflow systems. The three articles in this issue focus on practices for improving the relationships between users, developers, and software.

In the guest editorial of the earlier issue,1 we discussed some fundamental differences between (most) commercial software development and (most) scientific software development. These differences include the greater complexity and uncertainty of the scientific domain, the impossibility of full up-front requirements specification therein, and the difficulties of testing scientific software. Such disparity accounts for the fact that not every practice used in developing commercial software is also useful for developing scientific software.

Some attempts to bring software engineering techniques to scientific software development have generated light, whereas others have generated heat. The three articles we present here show, in the context of particular projects, how you can apply some software engineering practices (in particular, those pertaining to user engagement) in support of scientific software development.

Sarah Thew and her colleagues describe the establishment of requirements within an e-science context, epidemiology. The authors address the impossibility of specifying requirements up front by enabling a continuous design-discovery dialogue between users and developers. David De Roure and Carole Goble also discuss the importance of the relationship between users and developers in the context of developing the Taverna workbench and a site for sharing Taverna scripts. They describe six user-engagement principles and six design principles that promote the adoption of scientific software. Catriona Macaulay and her colleagues discuss the problems of integrating usability and user-centered design in a software development project to support microscopy.

As we mentioned in our earlier editorial, one of our aims in publishing these special issues is to promote a community of people interested in the issues of scientific software development. If you have any comments or ideas about how we might advance this aim further, we’d be pleased to hear from you (j.a.segal@open.ac.uk, chris.morris@stfc.ac.uk).

Reference

About the Authors

Judith Segal is a lecturer in computing at the Open University, working in the Empirical Studies of Software Development group. She studies software development by nonprofessional software developers such as financial mathematicians, earth and space scientists, and molecular biologists. She’s also interested in how to bridge the gap between the academic and practitioner communities. Segal received her PhD in algebra from the University of Warwick. Contact her at j.a.segal@open.ac.uk.

Chris Morris is a software developer in the UK government’s Daresbury Lab, where he leads a multidisciplinary team developing a laboratory information management system for molecular biology. Morris received his MA in pure mathematics from the University of Oxford. Contact him at c.morris@stfc.ac.uk.