The craftsperson and the scholar

Author(s)

J. Hetherington | SSI fellow

Posted on 9 November 2012

Estimated read time: 4 min

Sections in this article

Posted by s.hettrick on 9 November 2012 - 9:04am

By James Hetherington, **Research Software Development** Team Leader at University College London.

At Digital Research 2012, I presented a position paper with colleagues regarding the role of the Research Software Engineer. This paper followed on from a discussion I led at the Collaborations Workshop and some very interesting blog posts by Dirk Gorissen and Ilian Todorov. Rather than repeat these discussions, I've written this post for those who think the Research Software Engineer role could be for them.

A quick note about the Research Software Development Team at UCL

With the establishment of the Research Software Development Team at UCL, I hope we're on the way towards establishing a successful home for scientific programmers. If you love learning about cutting edge research, and enjoy crafting robust, readable and efficient code, then please apply to **join the UCL team.**

Bringing together the best of two archetypes

A good scientific coder combines two characters: the scholar and the craftsperson.

The first of these, the scholar, is the archetypical researcher who is driven by a desire to understand things to their fullest capability. Intellectually demanding problems attract, rather than deter, scholars. Thorough exploration of the research literature is part of their preparation for work. The scholar's curiosity is insatiable, and they prefer not to spend time on a topic once understanding has been acquired, though they enjoy passing on that understanding through teaching.

The craftsperson, on the other hand, desires to create and leave behind an artefact which reifies their efforts in a field. They feel pain when the things they make are fragile or ugly. They prefer to make things which explain themselves. Design for usability is a puzzle with a human factor. The craftsperson's work requires patience, and pride in doing a job well. They prefer not to complicate things beyond necessity, because complex systems are more likely to break.

Scientific software doesn't require a balance between these archetypes, it requires individuals who combine the best of both roles.

One can't create or maintain software without thoroughly understanding the system it describes. Scientific software requires the scholar's drive and ability to understand the research field. To be a research software engineer is extremely intellectually demanding, since it requires the ability to understand research literature across a huge range of research topics. Familiarity with

the needs and attitudes of the research user-community is also vital.

However, the construction and maintenance of research software doesn't stop when you have code that works. Indeed, the *program 'till it works* attitude, where the point is curiosity about results, is responsible for the sorry state of much research software. The goal must be the creation of something which lasts, something which others can use. Sustainable software will produce research results faster in the long run, but it requires a craftsperson's emotional commitment to making things that last.

The role of research software engineer might be the solution for software engineers who are frustrated at the lack of intellectual meat in the problems they address, or researchers who are frustrated by the pressure to move on to the next research paper. You might not get quite the freedom or recognition you would as a researcher, or be as well rewarded financially as you would as a corporate software engineer, but if you get your kicks from understanding the complex and then making a robust, clear and efficient tool, you should consider becoming a research software engineer.

Whatever your role, you should still let your inner craftsperson and scholar express themselves. If you're a researcher, insist on time to make code that you're proud of, whether the incentives make it difficult or not. If you're an engineer, then take the time to study the research literature, and then implement that ever-so-clever algorithm that makes feasible a new feature that will delight your users.

Share on blog/article:

Subscribe >

Data management

Privacy policy

Web accessibility

Contact

Site map