End-User Software Engineering and Professional End-User Developers

Judith Segal

j.a.segal@open.ac.uk

Professional end-user developers

By the term 'professional end-user developers' is meant professionals working in a highly technical, knowledge-rich domain who develop their own software in order to further their professional work. I have conducted empirical studies of such developers working in the domains of financial mathematics ([1], [2]), earth and space sciences ([2], [3]), and, currently, structural biology. These developers are distinguished from other end-user developers in two ways. The first is that, consistent with their being familiar with formal notations and logical scientific reasoning, they tend to have few problems with coding per se. The second is that, as a class, they have a history of developing their own software which long predates the advent of the PC.

My empirical studies reveal that professional end-user developers work within a culture which, while recognising that software failure can have devastating consequences, nevertheless does not appear to value the knowledge, effort and skill required for software development. There is a widespread perception that software development is something that 'anyone can do' – it is regarded as just part of the armoury of skills that all such professionals are considered to have, or to be able easily to acquire (it was compared to glass-blowing by one earth scientist). The studies also demonstrate that professional end-users typically develop software in a highly iterative manner and requirements largely emerge, rather than being specified upfront.

In common with many other types of end-user development, professional end-user development is generally regarded as simply being a matter of coding. The consequence of this is that the products of professional end-user development can often be less than robust, and the underlying code is not written with a view to being comprehensible or easy to modify and maintain. This is especially problematic as software originally intended for individual, exploratory use might later be used by other scientists and take on a more formal role within the organisation.

Some problems of professional end-user development

The culture of professional end-user development poses some significant problems such as:

1. How does the professional end-user developer acquire the software development knowledge that it is assumed that 'everybody' has (or can easily acquire)? One exacerbating factor is the reluctance of some professional end-user organisations to expend resources on formal training. Another is that the community of practice of professional end-user developers can be small in any particular organisation and is inherently unstable. Those research scientists who develop software tend to be on short-term contracts and/or at the beginning of their careers. In the latter case, as they advance up their career ladder, they are likely to concentrate on their scientific endeavours, leaving others to develop their software for them. In either case, their knowledge of software development becomes lost to the community.

2. How can awareness that software development is more than merely coding, become embedded in the culture of professional end-user development? If this awareness were so embedded, then the quick construction of a piece of software that appears to address the domain problem at hand (rather than expending more resources on the construction of software which is robust and maintainable) would be the result of a conscious decision rather than the unthinking norm, as at present.

How might these problems be addressed?

There are various suggestions in the end-user literature as to how the product of end-user development might be improved. These suggestions include: end-user developers should adopt a software development methodology; software engineers might provide professional end-user developers with a library of customisable components, and the professional end-user developers themselves should develop some sort of library of reusable customisable components.

Each of these suggestions requires some change in work culture and practices. Given that such changes are very difficult to effect and especially so where software development is not the main focus of the organisation, I argue in [3] that we should 'cherry pick' methods, tools and practices which have proved effective in the context of professional software development and which support – or only slightly perturb – the way that professional end-user developers work naturally. Given the additional difficulty that professional end-user developers have in acquiring and sharing knowledge of software development, I argue that we should also look for methods, tools and practices which tend to strengthen their community of practice.

Agile methods appear to be a likely source of practices etc. which both support the way professional end-users currently develop software and strengthen the community of practice. However, the effect of the introduction of agile practices into a professional end-user developer community is still to be investigated.

References

[1] Segal J., 2001, 'Organisational Learning and Software Process Improvement: A Case Study', in *Advances in Learning Software Organizations,* K-D Althoff, R.L. Feldmann, W. Muller (Eds.), Lecture Notes in Computer Science, Vol. 2176, Springer, 68-82.

[2] Segal, J.,2005, 'Two principles of end-user software engineering research', Proceedings of the 1st Workshop on End User Software Engineering, WEUSE, International Conference of Software Engineering, St. Louis Missouri, May 2005.

[3] Segal J., 2005, 'When software engineers met research scientists: a case study', *Empirical Software Engineering,* 10, 517-536.