Chapter 8
Developing Software for a Scientific Community:
Some Challenges and Solutions

Judith Segal
The Open University, UK

Chris Morris
STFC Daresbury Laboratory, UK

ABSTRACT

There are significant challenges in developing scientific software for a broad community. In this chapter, we discuss how these challenges are somewhat different both from those encountered when a scientist end-user developer develops software to address a very specific scientific problem of his/her own, and from those encountered in many commercial developments. However, many developers of scientific community software are steeped in the culture of either scientific end-user or commercial development. As we shall discuss herein, neither background provides sufficient experience so as to meet the challenges of developing software for a scientific community. We make various proposals as to which development approaches, methods, techniques and tools might be useful in this context, and just as importantly, which might not.

INTRODUCTION

Many scientific software projects intended for a broad scientific community succeed in that they make a significant contribution to the science. Many, however, fail. Some of these fail for scientific reasons (the underlying science was imperfectly understood), or because of coding problems (for example, an inappropriate choice of implementation language). Another less obvious cause of failure is the differences in the behaviour, knowledge, values, assumptions and goals between three different groups of people involved in such projects. These three groups are

DOI: 10.4018/978-1-61350-116-0.ch008

Table 1. Two snapshots from the first author's field studies:

Scientist: Anyone can develop software. Why should we listen to the advice of a professional software developer?	(Professional software developer is deeply offended)
Professional software developer: We need to start off with a clear document of your requirements, and then we'll draw up a requirements specification document which you can check.	*Scientist:* But that simply isn't how we work.

scientists; scientific end-user developers, that is to say, scientists who are developing software for their own use or for that of their close colleagues; and professional software developers, to whom the science is just another user domain.

In writing this chapter, we draw heavily on the field studies conducted by the first author, an academic, in a variety of scientific settings, and on the many years' experience developing scientific software of the second author, a professional software developer.

Our aims in writing this chapter are:

- To articulate some specific challenges facing scientific software developers. These challenges have their origins either in the culture of scientific end-user development or in the nature of science itself.
- To suggest ways in which these challenges might be addressed.

In what follows, we shall firstly articulate the behaviour, knowledge, values, assumptions and goals that characterize much scientific end-user development and then discuss the challenges which these characteristics pose when the context of the development is broadened. We then go on to discuss which development approaches, methods/techniques and tools might be useful in scientific software development, and, equally importantly, identify some which will not. Finally, we discuss how this identification of effective ways of supporting scientific software development can be progressed.

Throughout this paper, we stress the importance of context. A couple of examples give a flavour of this importance:

- A particular tool which is useful in a commercial development context might not be so useful in a scientific;
- Assumptions which are perfectly justified in a setting where a scientist is developing software for himself/herself to explore a particular scientific question might not be justified in other development settings.

This emphasis on the importance of context means that it is difficult to set any hard-and-fast rules along the lines of 'scientific software developers should apply *this* testing technique to their software'. We hope rather that this chapter might provide the means by which you might recognise the challenges in your particular development context, and suggest some ways by which you might address such challenges.

There is a caveat which we should stress here. One chapter cannot possibly say all there is to say about the challenges facing developers of software for a scientific community. We focus here on the challenges posed by the culture of scientific end-user development, as revealed by our field studies. These studies did not include FLOSS developments (free libre open source software), see the later section on future research directions. We also took little cognisance of CSCW (computer supported cooperative work) literature. We comment further on this literature in the additional reading section.