A note on our software development working environment

Solveig Albrand, Peter Sherwood

Much literature exists describing various ways of improving software quality.

Some authors recommend procedures that appear to be somewhat rigid to many physicists. Do any of the recommendations have any value for our working environment?

Below we list what we see as the "normal" approach to software development, and some changes that we feel would be of value to the community.

Some comments on commonly recommended procedures

- Development is performed by relatively large team. Although numbers of people in development n teams are not often mentioned, one can infer from such activities as task allocation that the minimum number of people in a team is about five people, and may extend to many more than this.

  This contrasts with our the majority of our team sizes lying between one (common) and three physicists.

- With large teams of people one can rely less on informal communication.

- The subject material may be completely unknown to the software development team. In this case procedures for establishing user requirements become important. In contrast, physicists often know their subject matter well. When this is not the case, the material is often of a nature that the physicist–developer can learn the subject domain more easily than a developer who is completely new to the field.

- Not all team members of a software development team are highly motivated. A formal structure helps individuals to understand what is expected of them, and for managers to determine whether the individuals are performing adequately. In contrast, researchers are often highly motivated. Approaching deadlines (beam!) can induce individual researchers to make heroic efforts.

Some comments procedures we often follow

We live in a community where people work independently. The individuals (or small teams) take sole responsibility for quality aspects of their work. Formal quality measures and goals are often non–existent. Instead, intuition is used, often to perform consistency

checks of of program outputs. These checks are often known as validating, or physics validating a program.

Documentation is often considered to be an add-on. When it does exist, it often consists only of a user guide.

Organisation of larger teams of people, while in principle possible, does not happen naturally. This is partly due to the non-hierarchical nature of the collaboration. Indeed, individuals may well be constrained to conform to the interests of their home institutes rather than coming together and participating in a structured, collaboration–based development team.

What is in it for us?

- Real benefit can be obtained from peer review, if the reviews are held in a constructive spirit, and are implemented in a lightweight manner (see notes on reviews). The very fact that material is to be reviewed often means that more care is taken immediately, rather than put off to some indefinite time in the future. Documents used for reviews allow focused discussions on technical points. Requirements documents and reviews are powerful ways to overcome the problem, which is perhaps even more common when all concerned have familiarity of the problem domain, that come about from assumptions that "everyone else understands the issues just as I do".

- Some of our core software will be used by many people, and will have (presumably with changes) a long lifetime. Working methods which improves maintainability and the ease with which it can be developed and extended are desirable. The approach which leads to these qualities are often the opposite of those used for quick prototyping.

- We are moving to writing more of our code in C++. While this language allows us to write object-oriented code, this is certainly not an automatic result of moving to this language. Indeed, it could be argued that procedural code written in C++ is harder to maintain and develop than code written in Fortran. The nature of OO code is such that comprehensibility improves enormously with a moderate amount of documentation (UML sequence and class diagrams in particular).

It is clear that it is neither possible nor desirable to convert our current social structure from the one where we work either as individuals or in very small teams where each the individual has sole responsibility for all aspects of software development to one where we work in a large, rigidly organised teams slavishly following some How-To book of rules. It is not desirable as it would squander our essential resources of independence, versatility and commitment.

It is also clear that each individual working in his own manner may not be working optimally given the demands of a product that is satisfies many users, and is maintainable and extensible.

Our vision is that, although some of the QA suggestions will be onerous, and seem to be of limited value in the short term (which is the main concern of people developing private or prototype code), that clear cut procedures, based on the a number of well defined steps, and monitored by a reviews will have important long term benefits. Further, the use of various software tools should be made available to support such procedures.