Agile Software Development for e-Science

Agile Software Development for e-Science

Yuwei Lin, Meik Poschen, Rob Procter, Alex Voss, Carole Goble, Jitenkumar Bhagat, David De Roure, Don Cruickshank, Mark Rouncefield

New research challenges increasingly demand collaborative and cross-disciplinary methods. In recognition of this, substantial resources have been invested in developing virtual research environments (VREs) based on Grid technologies and standards. However, it seems that these solutions do not necessarily address the different needs of scientists across the full range of research areas and disciplines. That, at least, is one interpretation of the steadily accumulating evidence that the scientific research community is leveraging Web 2.0 tools and technologies to facilitate collaboration through the creation of social networking sites for the sharing and re-use of research findings, knowledge and artefacts.

In this paper, we focus on the experiences of myExperiment, a project to create a Web 2.0 social networking site for scientists with the aim of reflecting on the principles and practices for developing them. myExperiment aims to enable scientists to share digital resources associated with their research and, in particular, to share and execute scientific workflows (De Roure, Goble and Stevens, 2007). It aims to help scientists do research in a distributed community, to share, re-use and repurpose experiments, to reduce time-to-experiment, share expertise and avoid reinvention. Much of this is enabled through increasingly familiar social networking and social collaboration features.

In a recent paper, De Roure and Goble (2008) set out six key principles of scientific software design derived over several years of experience of developing e-Science applications and further evolved during the course of the myExperiment project: *fit in, don't force change*; *jam today and more jam tomorrow*; *just in time and just enough*; *act local, think global*; *enable users to add value*; and *design for network effects*. This paper briefly revisits these principles and then explores some of the challenges of realising them in practice, including dealing all the usual contingencies and constraints to which project work is subject (Button and Sharrock, 1996).

Conventional software engineering methodologies are recognised as being too inflexible for dealing with imprecise and volatile requirements. 'Agile' approaches have become quite popular as an answer to the rigidities of traditional phased methodologies. One of the most notable of these is extreme programming (Beck, 2000). Among its key elements are a focus on working code, involving early release and short release cycles and an incremental planning approach that allows changes to be made according to evolving circumstances and user requirements, and it is this approach that is being followed in myExperiment, such that the site is up and running as an almost 'perpetual beta' (De Roure and Goble, 2008).

In this paper, using myExperiment as a case study, we focus on the sorts of practical actions necessary to create and maintain the conditions wherein an agile development approach can flourish, striking a balance between responsiveness to often rapidly evolving requirements while ensuring the project is kept 'on track'.

Drawing on fieldwork, we will describe how the project team, whose members are distributed across multiple sites (and often mobile), exploits a repertoire of coordination mechanisms, communication modes and tools (face-to-face meetings, workshops, 'hackfests', teleconferencing, Skype-based talk and instant messaging (IM), wikis, mailing lists), artefacts and structuring devices (sketches, mock-ups, (throwaway) prototypes, software architectures, code) in order to achieve the orderly and smooth running of the project while following an agile, user driven approach.

Finally, we will consider the lessons learned, examining what has worked and where there is room for further improvement and conclude with some general recommendations for software development practices in this important application field.

References

Beck, K. (2000). Extreme Programming Explained: Embracing Change. Addison Wesley.

Button, G.. and Sharrock, W. (1996). Project work: the organisation of collaborative design and development in software engineering, Computer Supported Cooperative Work, v.5 n.4, pp.369-386.

De Roure D, Goble CA and Stevens R. (2007). Designing the myExperiment Virtual Research Environment for the Social Sharing of Workflows. In Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, Bangalore, India, 10-13 December, pp. 603-610.

De Roure, D. and Goble, C. (2008). Six Principles of Software Design to Empower Scientists. IEEE Software.