Implementing eResearch Projects using Agile Development

Nicholas May,
RMIT University, Melbourne, Australia, Nicholas.May@rmit.edu.au

INTRODUCTION

It is important to take the expertise of stakeholders into account when developing software systems. In the case of e-research systems, the roles of product owner and domain expert are usually filled by researchers: however, researchers are often inexperienced in the application of software development methodologies.

An active area of discussion in field of eResearch relates to the benefits that can be realised by using agile development, as demonstrated by a workshop at a previous eResearch Australia [1]. Agile development encompasses a family of methodologies that are well suited for projects where the requirements are poorly understood. They promote the inclusion of users in the development process and the elicitation or requirements from user feedback.

In this presentation, we discuss why agile methodologies should be suited to e-research projects, the processes we used to implement several e-research projects, and our experience using agile development. Finally, we draw some conclusions about the when it is appropriate to use agile with an e-research projects.

AGILE DEVELOPMENT AND ERESEARCH

The origin of the term 'agile', as applied to software development, stems from a meeting of like-minded people intent on exploring alternatives to heavyweight development processes. Problems they were trying to overcome include: delivered software that is not useful, features that are not used and a lack of visibility during the development of the software. This meeting produced a manifesto which outlined the values associated with agile development [2], which can be summed up as follows;

- Iteration: a working software is released at regular intervals during the development project.
- Communication: face to face meetings are the principle method of determining how to proceed.
- Collaboration: the product owners and users should be involved in decisions throughout the development project.
- Flexibility: priorities should be open to change based on current requirements.

As can be seen from these values, agile methodologies favour change over pre-planning. This is important in e-research projects because it facilitates an ongoing process of requirement discovery. At each iteration, the researchers get to explore the working software, provide feedback to the developers and set priorities for the next iteration. This helps the researchers to think in terms of their future practices and leads to a delivered software system that contains only the most important and useful features.

Whilst we have been discussing their benefits, agile methodologies are not always the most appropriate for a development project. As the size of the development team grows the overhead of coordination and face to face meetings becomes a burden. And without regular input from the product owners and users, the developers can get the requirements and priorities wrong. Hence, agile methodologies are best suited to small development teams working closely with product owners and users.

OUR AGILE PROCESS

There is a growing list of agile methodologies, of which we have drawn upon the following three for our process: Extreme Programming, Scrum and Kanban [3,4,5,6]. A set of practices, meetings and task types was selected from these methodologies to form an initial process. The practices include;

- Iterations of fixed number of weeks (called a Sprint).
- Defined roles for each person involved in the project (i.e. Product Owner, User, Developer, Manager).
- Common ownership of the source code by all the developers.
- Pair programming used to transfer skills, when required.
- An appropriate metaphor that guides the system language and layout.
- All project tasks are defined and monitored in a hosted development application suite called JIRA Studio [7].

All project tasks are described and allocated to one of three prioritized lists: product backlog (features described by the users), technical backlog (development tasks required to implement the features), and impediment backlog (issues outside the control of the development team that prevent implementation of features). Items in each backlog can be assigned an effort and allocated to a particular sprint. The management of items were controlled within the following project meetings;

- Planning: users describe the features, and the product owner prioritizes the product backlog.

- Estimation: the development team discusses and estimates the backlog items, and selects what they can complete in the current sprint.
- Scrum: the development team reviews and updates the status of items in the current sprint.
- Review: the development team demonstrates the working software to the users, who explore and provide feedback on the features.
- Retrospective: the development team reviews the sprint and adjust the methodology.

As the projects progressed, the schedule of meetings evolved to include; one retrospective, planning and estimation (in week one), one review (in the last week), and regular scrums (at least one per week).

EXPERIENCES FROM IMPLEMENTING ERESEARCH PROJECTS USING OUR PROCESS

Two e-research projects were implemented using the above process: however, each was managed separately, with different timelines and project members. Both were projects for the Australian National Data Service (ANDS) [8], one was a 'data capture' project and the other a 'seeding the commons' project. An initial problem was encountered with the mismatch in schedules between development sprints and the deliverable timeline for ANDS. This was resolved by assigning the responsibility for allocating milestone tasks to the appropriate sprint to the team manager.

The development teams consisted of 2 to 3 developers per project, all of whom were relatively new to agile development. This led to the decision to start with a sub-set of agile practices which were then expanded as the developers and researchers became more comfortable with the new way of working. In addition, the access to relevant researchers was intermittent at the start of each project: however, we were able to schedule more technical tasks in the first couple sprints to solve some technical questions, and as the projects evolved and the benefits became apparent, the necessary researcher involvement was satisfied. Also, a core value of agile development is collaboration, a concept with which researchers are usually quite familiar. This means that they were quick to pick up on this aspect of the process and forthcoming with valuable feedback.

As has been discussed, the agile process helps researchers to discover their requirements for the software. The practice of selecting a metaphor for the system enhances this by defining a vocabulary that makes it easier for the developers to understand what the user requires and by suggesting possibly unforeseen features. Finally and most importantly, the agile process should be flexible. Within each sprint the process was static, but the purpose of the retrospective meeting was to review and evolve the process based on the team's experience with the previous sprint. Thus, the process was optimized as the developers and researchers experience grew.

CONCLUSION

In this presentation, we discussed the benefits of using agile development methodologies when implementing e-research software. We described an agile process that evolved over the lifetime of two e-research projects and some of the lessons that were learned.

We hoped to show that agile development is suited to e-research projects, but only if the development team is not too large and the intended users are actively involved in the development process. However, what we found was that agile development is more suited to projects where the users are researchers, because their existing skills fit well with the agile practices.

Finally, our agile process is evolving with each new sprint. In the future, we intend to look at specific practices that can be included to manage and evolve the architecture of the systems.

REFERENCES

1. *Where Agile meets eResearch*, Georgina Edwards and Rodney Harrison, Workshop, eResearch Australia, 2010, abstract available from: http://ocs.arcs.org.au/index.php/eraust/2010/paper/viewFile/42/2, accessed 30 June 2011.
2. *Manifesto for Agile Software Development*, available from: http://agilemanifesto.org/, accessed 30 June 2011.
3. *Extreme Programming: A gentle introduction*, available from: http://www.extremeprogramming.org/, accessed 30 June 2011.
4. *Scrum and XP from the Trenches - How we do Scrum*, Henrik Kniberg, C4Media Inc., 2007, available online: http://www.infoq.com/minibooks/scrum-xp-from-the-trenches/, accessed 29 June 2011.
5. *Kanban and Scrum - making the most of both*, Henrik Kniberg and Mattias Skarin, C4Media Inc., 2009, available online: http://www.infoq.com/minibooks/kanban-scrum-minibook/, accessed 29 June 2011.
6. *Your Scrum Checklist'*, Boris Gloger, bor!sgloger Wien. Baden-Baden, 2011, ISBN-13: 978-3-000-32112-2.
7. *JIRA Studio*, Atlassian, see http://www.atlassian.com/hosted/studio, accessed 30 June 2011.
8. *Australian National Data Service*, see http://www.ands.org.au, accessed 30 June 2011.

ABOUT THE AUTHOR(S)

Nicholas May is a software engineer with the eResearch Office at RMIT University, with responsibility for setting up and monitoring the agile development process. He is a certified professional member of the Australian Computer Society, and has more than twenty years of development experience across the software development lifecycle. In addition, he is a PhD Candidate in the School of Computer Science & Information Technology, at RMIT University, with research interests in the fields of service-oriented computing and fault tolerance.