NACCQ Fostering Computing Education in NZ

Bulletin of Applied Computing and Information Technology

Invited:

05:02
December 2007

Extreme Research: Applying Extreme Programming Practices to Research

Mark Toleman, University of Southern Queensland, Australia
markt@usq.edu.au

Toleman, M. (2007, Dec), Extreme Research: Applying Extreme Programming (XP) Practices To Research. *Bulletin of Applied Computing and Information Technology* Vol. 5, Issue 2. ISSN 1176-4120. Retrieved May 1, 2008 from http://www.naccq.ac.nz/bacit/0502/2007Toleman_invited.htm

1. INTRODUCTION

A few years ago, my colleagues and I embarked on a journey that had multiple rationales and multiple outcomes. We had all been software developers at some point in our professional lives and we noticed a growing interest in the software developer community about so-called Agile methods, and eXtreme Programming (XP) in particular (Beck, 1999; Beck & Andres, 2005). None of us had explicitly applied these methods in the projects in which we had been a part, but we did have some other colleagues who had and still were. As academics, these Agile methods interested us because our developer colleagues were reporting significant benefits of using them. Many of the practices seemed to reflect the way these developers preferred to work and since traditional software development methods had a relatively low adoption rate, perhaps these new methods provided a more relevant approach. We also saw a research opportunity, a gap, with many practitioner reports but little academic research focussing on Agile methods. Another rationale was to build research capacity by mentoring young researchers in designing, conducting and reporting research.

Our efforts culminated in a number of research outputs (Ally, Darroch & Toleman, 2005; Darroch, Toleman & Ally, 2004; Toleman, Ally & Darroch, 2004; Toleman, Ally & Darroch, 2005; Toleman, Darroch & Ally, 2006) with several more still to be written. The research team has moved on and we may return to the original agenda when other circumstances change. However, my point in this article is not to describe or detail our research; that can be found in the articles referenced. The purpose of this article is to describe our research approach, which actually took on a form reminiscent of XP with its practices and problems. Did we start out to use a form of XP as our research approach? No, but it did seem to us that many of our research practices reflected XP practices.

XP was originally centred on twelve core practices (Ally, Darroch & Toleman, 2005), also known now as *Xp Xtudes*, which guide the software development process (Table 1). An updated version of XP is available (Beck, 1999) which is less restrictive in terms of defining practices, but the essential features and practices are still relevant. These practices reflect the sentiment/intent of the twelve principles underpinning the Agile Manifesto.

Table 1. XP Core Practices (Source: adapted from [7])

XP Core Practices	Description
The Planning Game	The customer decides on the functionality (called stories) to be implemented while the developers estimate the time required, and through negotiation, workloads for the next cycle (project-dependent but perhaps week) are planned.
Small Releases	Production software is released to customers regularly, usually weekly, but potentially daily.
System Metaphor	The team has an overarching view or model of the system being developed. At the very least a common vocabulary is required.
Simple Design	Developers avoid unnecessary complication with respect to software architecture and coding, staying with the stories agreed with the customer each cycle.
Test Driven Development	Tests are written prior to code development.
Design Improvement (was Refactoring)	There is a process of continual improvement of the code as the developer's understanding of the system grows. Functionality is not altered so all tests should still operate effectively.
Pair Programming	Two programmers collaborate on the same program code. .Every code fragment is developed by a team of two programmers sitting together working at the same workstation. There are two roles, namely, a driver controlling the mouse, keyboard or other input device to write the code and unit tests, and a navigator, observing and quality assuring the code, asking questions, considering alternative approaches, identifying defects, and thinking strategically. The partners are considered equals and will regularly swap roles and partners.
Collective Code Ownership	Developers were free to work on all code. Any code may be changed provided it is done by pairs of developers, complying with coding standards and subject to a satisfactory run of all tests.
Continuous Integration	This involves the integration of new code into the project with consequential system building and testing.
Sustainable Pace (was 40-Hour Work Week)	Developers are restricted to about 40 hours of work per week.
Whole Team (was On-site Customer)	Customer availability in project gives developers continuous access thereby lessening the need for extensive requirements documents. They can ask the customer about functionality, test cases, interfaces, etc, at any time.
Coding Standards	Coding standards are developed for each project. Since code may be worked on by any programmer at any time, coding standards are essential and must be rigorous.

So how did our research approach match these practices? Below are each of the twelve core practices and a brief description of our how our approach matched or might be considered to match.

2. The Planning Game

Each week we met to plan or map out research activities for the following week. These meetings ranged up to one hour and included allocation of various research-related tasks including research design, data collection, data analysis and writing tasks. We always had

multiple projects in progress. There was no actual customer involved in these meetings. However, if one of our customers could be considered as an upcoming conference deadline or a book chapter or journal submission schedule, then they were certainly deciding on the 'functionality to be implemented'. Our outputs were not software systems but research reports and articles, and they needed to address the expectations of the various forums with which we engaged. Other customers could be our developer colleagues. Again these were not explicitly involved in our planning meetings but we did regularly make them aware of our research explorations.

3. Small Releases

Each week there was an expectation on each member of the team to undertake the tasks allocated. We built our research outputs in small manageable chunks, each of us concentrating on those aspects of the output that reflected our expertise. We could not say that we released our outputs to customers weekly, but we did communicate with developer colleagues regularly, sometimes informally and sometimes formally via an established 'Developers on the Downs' group. Deliverables for conferences, books and journals were obviously less frequent than weekly, thank goodness!

4. System Metaphor

We did not really establish a system metaphor for our research approach. This is a not uncommon failing of XP projects. We did, however, establish a common vocabulary even becoming known in our academic department as 'the Agile group'. In retrospect, XP itself, or some version of XP, might be considered the metaphor for our approach to research.

5. Simple Design

For several reasons we chose simple approaches to research design and methodology. Our first few studies, for example (Darroch, Toleman & Ally, 2004; Toleman, Ally & Darroch, 2005) were reports on cases, some successful and others not so successful. Not all of the cases examined have been written up, for various reasons, which is perhaps typical of an Agile approach in itself. We also conducted designed experiments on pairing and pair programming (still to be reported) and even developed theoretical frameworks (Ally, Darroch & Toleman, 2005); Toleman, Ally & Darroch, 2004) relevant to Agile approaches to software development. In all these we avoided complexity and complication. Our goal was to produce research outputs that met the needs of our customers with the minimum of fuss.

6. Test Driven Development

Not all of our research outputs met the needs of customers the first time. Indeed some were rejected outright. However, we took each rejection as a message about the suitability of the research output for the intended audience and considered ways to improve the output for the next customer. No customers were happy the first time and if the reader is a software developer, then they will know this story well.

7. Design Improvement

Testing indicated design flaws in our outputs so improvements were necessary if we were to have eventual success. It has to be said that, at times, this testing did show some significant design faults but due to the simple designs chosen the correction was never a major concern. So far, every output has found a 'home'.

8. Pair Programming

We designed experiments to examine pairing and pair programming, thus far unpublished, and even enhanced an existing framework about pair programming success (Ally, Darroch & Toleman, 2005). Did we use pairing in our research? On a couple of occasions, we found it useful to allocate tasks to pairs. We even found situations where, as a group, we could achieve a level of knowledge and understanding about some issue better than as individuals. However, we did not apply pairing routinely to all tasks. Like the case of 'system metaphor', this is a not uncommon outcome for XP projects

9. Collective Code Ownership

We were free to work on any part of any research output at any time. Initially we used an approach that involved passing a token around so that we did not accidentally over-ride someone's work. Later we used a central repository to house the various outputs in their various stages. However, contamination of output was rarely an issue because of the task allocation mechanism we used at planning meetings. Research these days is so rarely an individual pursuit. It is imperative that a system is in place to allow maximum involvement and activity at all phases in the projects. Would our system have worked so successfully in a larger team? Our team will never know as we never desire nor expect to be much bigger, but I am sure there are large research teams out there, particularly in biotechnology or laboratory experiment situations, where similar issues apply.

10. Continuous Integration

At times, we needed considerable and consolidated effort on a particular research output. This was over and above our usual tasks for the week and most often involved an afternoon or full day. We would use a computer linked to a projector and systematically review, evaluate and update a particular output for a specific purpose. Usually such efforts were driven by a pending (or overdue) deadline.

11. Sustainable Pace

With multiple conflicting requirements on our time, fitting our research into a typical 40-hour week was just not possible. Inevitably, we completed research tasks outside normal working hours, at night and on weekends; indeed I am writing this on the Monday of a long weekend. Most academic researchers will concur that research is so often the last thing that is done in a working week, after teaching and administration are completed (if they are ever completed either). However, in terms of some of the cases of XP use that we examined, our situation is no different to the developer.

12. Whole Team

The definition of a customer is perhaps the most problematic issue for my XP metaphor for our research approach. Whether the customer is a software developer or a publication outlet, they were certainly not continuously available. Sometimes that level of availability would have been extremely helpful and avoided the need for trips down many blind alleys but the reality of our situation was that this was not a possible scenario.

13. Coding Standards

Our coding standards evolved over time so that eventually our written work was receiving less criticism and was being accepted more readily. A major component of research is identifying that part of the research that is necessary and sufficient for the audience. Another is effectively communicating the findings. These coding standards help external communication as well as internal communication for the research group; we established a common vocabulary even becoming known for this in our academic department.

14. Conclusion

We learned much conducting research on Agile methods and XP in particular. We learned about new approaches to software development, and about the people using the approaches and the situations in which they found themselves. We learned about designing, conducting and reporting research and about building a research agenda, and about ourselves, how we interacted within such an environment.

Were we being Agile? I think so. In the sense of the dictionary meaning of 'agile' we were certainly active and lively in producing relevant outputs efficiently. Did we really follow the XP practices? Certainly some were more obviously visible than others but this seems to be the practitioner experience of XP use too. Was it a successful approach to research management? You can be the judge but my view, based on relevant research outputs and our original goals to target a research gap and build research capacity, is a firm yes.

Acknowledgement

I want to thank my colleagues Mustafa Ally and Fiona Darroch without whom this research would not have been possible

References

1. Ally, M., Darroch, F. & Toleman, M.(2005). A framework for understanding the factors influencing pair programming success. In H. Baumesiter (Ed.), *Proceedings of the Sixth International Conference on Extreme Programming and Agile Processes in Software Engineering—XP 2005*, Lecture Notes in Computer Science LNCS 3556 (pp. 82-91). Berlin, Germany: Springer-Verlag.
2. Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison Wesley.
3. Beck, K. & Andres, C. (2005). *Extreme Programming Explained: Embrace Change*. Second edition, Boston, MA, USA: Pearson Publishing.
4. Darroch, F., Toleman, M. & Ally, M. (2004). Exploring the domain suitability of Agile system development methodologies in aged and community care: A system user perspective.In *Proceedings of the Aged and Community Care Informatics Conference (ACCIC'04).*
5. Toleman, M., Ally, M. A. & Darroch, F. (2004). Aligning adoption theory with Agile system development methodologies. In *Proceedings of the Eighth Pacific-Asia Conference on Information Systems (PACIS 2004)* (pp. 458–471).
6. Toleman, M., Ally, M. A. & Darroch, F. (2005). WEB publishing—an eXtreme, Agile experience. In R. L. Baskerville, L. Mathiassen, J. Pries-Heje & J. I. DeGross (Eds.), *Business Agility and Information Technology Diffusion*, pp. 245-256. New York, USA; Springer.
7. Toleman, M., Darroch, F. & Ally, M. (2006). The impact of Agile methods on managing IT professionals. In P. Yoong & S. Huff (Eds.), *Managing IT professionals in the Internet Age*, pp. 233-253. Hershey, PA, USA: Idea Group Publishing.