Merging Agile Methodologies
The Case for DSDM and XP

Andrew M Goulding and Rory V O’Connor
Dublin City University
Ireland

Abstract

This paper presents the proposal that while the agile approach to developing software is proving to be a success, there is value in merging some of the best practices of more than one approach. In this paper, two specific agile approaches - Dynamic Systems Development Methodology (DSDM) and Extreme Programming (XP) - are examined. This paper focuses on the proposition that if an organisation has implemented one approach (DSDM) then is it possible to bring elements of good practices from another approach (XP) in order to add value to the overall process. The main objective of this research was to establish where common ground exists between these two approaches and to explore how they may best work together. A case study within an organisation was conducted to enable further investigation and to evaluate the underlying research proposal.

Keywords

Agile Methodologies, Extreme programming, DSDM, Process Improvement
1 Introduction

An organisation may be motivated to define and improve its software development processes due to a variety of forces affecting software development and maintenance. As a result there is a need for organisations to continuously explore new approaches to software development, which incorporate not just the development of new methodologies, but exploring new ways of using existing methodologies. DSDM and XP are both established agile methodologies. As a framework for business centred development, DSDM focuses very strongly on the full project lifecycle and describes an iterative and incremental process for software development in the context of new and established business processes, whereas XP has a narrower more rigorous software developer focus. At a simplistic level, DSDM can be viewed as a business focused systems development framework, where XP may be viewed as an approach to coding.

Although they are distinct approaches, DSDM and XP have much in common [1]. However, many see DSDM and XP as competitive methods for software development and therefore assess which methodology to adopt based upon the strengths and weaknesses as perceived from the firms perspective, normally resulting in the selection of one method over the other.

This paper is focused on the proposition that if an organisation has implemented one agile approach then is it possible to bring elements of good practices from another approach in order to add value. Specifically, in this paper we will examine the case for two agile approaches – DSDM and XP. The main objective of this research being to establish where common ground exists between these two approaches and to explore how they may best work together. A case study within an organisation was conducted to enable further investigation and to evaluate the underlying research proposal.

1.1 Case Study Organisation

The organisation that participated in this research is a large Irish financial institution with an in-house software development department which employs approximately 250 software developers. The developed systems have been a major factor in the success of the organisation with the more traditional approaches to development projects being used. In the mid-nineties the need for change was recognised, driven by the fact that the existing IT approach was in danger of impeding rapid change. In answer to this an internal group composed of senior IT and business management along with an external expert advice conducted an investigation of industry best practices and the main outcome was the selection and implementation of DSDM. To date, DSDM has been a proven success within the company and has led to many implementations that have met business and IT requirements.

2 DSDM and XP – A Comparison

In general it appears that XP and DSDM share many of the same ideas on how projects should be run. XP attempts to simplify how teams work while also introducing some ‘extreme’ ways of coding and testing. Visibility and communication are also central to XP and these can only help ensure that the project is delivering what the customer expects. Many organisations have realised that XP and DSDM are not merely competitive methods for application development but can in fact complement each other [1]. The DSDM lifecycle puts more emphasis on the pre-project set-up in terms of feasibility studies and prioritisation. It also takes into account Implementation and Post-Project Reviews which are not apparent in XP. XP appears to focus more on the actual development phase once it is up and running but not really on what happens before or after this. XP has been described as being too light on project controls where this is a known strength for DSDM. It would appear that combining the two would give a controlled framework with robust programming practices [1].
Based on comparisons made [2] the following show how XP practices could be practically applied to a DSDM project:

- User stories can be used for requirements gathering, estimation and for building test cases. This is a relatively easy concept to introduce with minimum preparation and training required. User stories to be publicly displayed within the project team for constant reference especially during stand-up meetings.
- Stand-up meetings can be used for discussions between business users and developers – user stories to be used as the main focus for these discussions.
- Simple design and re-factoring to be applied where possible. This can be employed regardless of whether DSDM or XP is used.
- Pair programming to be used as a means to improve code quality and enhance developer knowledge transfer. This will challenge traditional thinking but once the improved quality is proven then it could become an accepted development technique.
- Test-driven development to be used with suitable tools to be made available, where possible, to the developer. This will require a significant amount of training for the developer and an acceptance that improved quality is the output leading to less user acceptance testing time required. Test-first approach for user testing to be used as standard.
- Continuous integration can be implemented but at a cost to the business in terms of test hardware and time taken to re-build the test system and re-run tests on a regular basis.
- Collective code ownership has been around in different forms. The ability to make this work may depend heavily on the environment and the types of source code management tools in operation. The concept should be considered for each individual project.

3 Case Study

This project was the development of a new system to track the activities of the administration section of a business area where requests for action are taken on electronic forms and processed on a daily basis. The main driver for this new system was due to the fact that the business department lacked management information about the volumes and the time taken to process these requests as well as the information on the error rate during the processing of these requests.

Prior to starting the pilot project a set of three focus group sessions were conducted within the organization to present and discuss ideas on implementing XP practices within the DSDM environment and to gain views all stakeholders. These focus groups were chaired by one of the researchers and included senior management, development staff, a DSDM consultant (DSDM consortium) and an experienced XP developer. All stakeholders agreed that user stories, refactoring, test-first design, continuous integration and pair programming would be evaluated during a pilot project as additional practices to the DSDM approach. The idea of fully utilising automated testing was put aside, and the XP approach to testing using the normal execution of tests for both unit and acceptance testing.

The agreement was to use the following XP practices in addition to the normal DSDM process:

- The planning game - User stories were to be used during the requirements gathering phase. They were to be used for stand-up meetings and as a means of communication, feedback and build of test cases, with developers choosing user stories and estimate effort per story
- Test-First Design - Business testers to take the user stories and build acceptance test cases before development and developers to build unit test cases before coding.
- Refactoring - Developers were to use refactoring during project where possible or sensible, with continuous Integration and daily integration of new code.
- Pair Programming - Developers were to work as a pair.
3.1 Project Phases

- **Design Phase** - The project then began with the initial design workshop involving all stakeholders, where roles and responsibilities were defined as per DSDM. The main outputs of the were the story cards and these were the focus for discussion throughout the project. User stories were then used for requirements gathering and were used as part of a prioritisation session, with the MoSCow approach for prioritisation being used. Time-boxes were agreed- two time-boxes with three weeks per time-box with a live delivery at the end of each time-box.

- **Development Phase** - The design and coding phase began with agreement between the developers on ownership of story cards, which led to a clear set of objectives for both developers with the more experienced developers taking ownership of the more complex functions. As test-first development was to be used the programmers firstly wrote unit test cases for each user story or function to be written. Pair programming was the agreed format for coding and the developers set up with this in mind. All work was logged to the usual system for recording project time and this would be used to determine actual effort against planned effort for all phases of development. At the end of each working day all code developed was merged with previous latest version and the test system re-built.

- **Test Phase** - The test coordinator also took the story cards to create the user acceptance test cases in advance of the development. This led to discussion, particularly at the early stages, on the clarification of the requirements and the system functions. Traditionally user acceptance test documentation takes place independently of the development team and is based on the requirements specification which is usually contained in one single document. The significant change here was that the individual user story cards were used as opposed to the requirements specification and was carried out before any development began. As automated testing was not available for this project the user testing took place in a traditional manner.

4 Results and Analysis

The case study was completed with a post-implementation review. Key issues identified were:

- Project initiation and definition of roles and responsibilities took place as per normal for a DSDM project. Resources were assigned as required with full commitment from all stakeholders to the project plan.

- Time-boxes were decided up-front and delivered on time (two time-boxes with 3 weeks per time-box where a live delivery took place for each one).

- Priority needs were delivered early in first release with the reports coming from this phase being produced for the department some 6 weeks after project initiation.

- Requirements were fully captured – primarily during the initial workshop and a small number of changes were made subsequent to this. Subsequent prioritisation of requirements did not change significantly from the decisions at the workshop.

- Development and test phases ran to the expected time-scales with no significant overhead seen from the use of XP.

- User acceptance testing sign-off was achieved for both time-boxes in a shorter period to the planned time-frame.

The use of user stories during this project was considered successful. In particular it made the requirements gathering easier for both customer and developers and brought a personal touch to requirements, in that business users felt that they owned that story. Gaps and complexity were very clear in when stories were on a board and everyone could see them and all parties involved felt that this was an improvement from the DSDM approach where flip-charts of requirements were not as easy to view. Also prioritisation was easier to achieve as cards could be moved around without difficulty. User stories were used for stand-up meetings and as a means of communication, feedback and build of test cases, which was accepted by all involved as an improved way to discuss requirements.
and issues compared to the normal DSDM approach where reference to requirements specifications was not so regular or easy. Accordingly user stories appear to work better than the DSDM approach. It is worth noting that in the pilot project there was no obvious improvement in requirements coverage as compared to previous DSDM projects.

Developers worked as a pair constantly throughout project as opposed to DSDM where developers work independently. The developers had no issues in working together though they commented that a strict time schedule needed to be enforced to ensure hand-over or there was potential for one person to dominate the coding. The cost of pair programming was calculated to be approximately 30% longer to code compared to the normal output. This was based on the estimate which was originally derived in the normal way as against the time recorded on the tracking system used for all IT projects. However, complexity becomes a factor here as complex issues were definitely resolved faster (developers estimate that solutions were found twice as quick as in the normal scenario). Given that approximately 20% of the code was complex then the actual increase in effort works out as 17% effort in total extra effort regardless of the size of program.

Test first design was split into two parts with the programmer writing unit tests before coding and the business tester writing acceptance tests as soon as possible after the requirements are identified. Test-first improved communication during the project, with business and IT staff working much closer than in traditional projects. The developers build unit test cases before coding and found that they could code with more confidence once this had happened and effort improvements would certainly be realised. It was evident that effort was not significantly higher for actual set-up of unit tests but there was a reduction in time taken to the overall development and unit test phase together compared to a traditional project of similar size. The overall reduction in defects through pair programming helped reduce user acceptance test effort significantly. Generally in the organisation experience has shown that 40% of all changes made after delivery to test are for bug fixing – a greater proportion by some 20% than for requirements changes. In this case a reduction in bug-fix testing was seen compared to a traditional project with 57% less defects identified after unit testing.

5 Conclusions and Discussion

The main finding of the research and pilot project is that the XP practices can truly add real additional quality assurance to the lifecycle of a project, regardless of what other methodology is being used. In this case, during the pilot project in the organisation, the benefits of DSDM and XP are realised with DSDM continuing to bring a strong framework and ensuring that the requirements are found and met. XP development and testing practices have helped achieve both improved quality and introduced some potential around effort savings when pair programming and automated testing are utilised.

The research presented in this paper is primarily based on the experience of a single case study experiment carried out within a financial institution. While the findings are positive and show good promise, the small scale of this experiment must be acknowledged. A larger scale investigation would need to be undertaken to validate the results of this early study.

With the recent launch of DSDM version 4.2 – which contains guidance for those wishing to use XP in conjunction with DSDM - the DSDM consortium have acknowledged the need to explore the potential for combining these two separate methodologies. However at this early stage there is a lack of any real experience of using XP and DSDM.

6 References


7 Author CVs

Andrew M Goulding

Andrew Goulding is an IT Manager with a large financial institution based in Ireland. He has over 15 years of experience in information systems development. Andrew received a B.Sc. in Computing and Mathematics from University College Dublin and an M.Sc. in Computer Applications from Dublin City University.

Rory V O'Connor

Dr. Rory O’Connor is a lecturer in the School of Computing at Dublin City University. He received a PhD. in Computer Science from City University (London) and an M.Sc. in Computer Applications from Dublin City University. He has previously held research positions at both the National Centre for Software Engineering and the Centre for Teaching Computing, and has also worked as a software engineer and consultant for several Irish and European technology organisations. His research interests are centred on the processes whereby software intensive systems are designed, implemented and managed, in particular, methods, techniques and tools for supporting the work of software project managers and software developers in relation to software process improvement, software project planning and management of software development projects. Dr. O’Connor is a member of the Irish Software Engineering Research Consortium (ISERC). He can be contacted by email at: roconnor@computing.dcu.ie