Adopting iterative development: the perceived business value

Caryna Pinheiro, Frank Maurer, Jonathan Sillito

University of Calgary
Calgary, Alberta, Canada
{capinhei, frank.maurer, sillito}@ucalgary.ca

Abstract. Iterative development is a common characteristic of agile methods. It is important to understand how the adoption of an iterative process provides business value, and how this value can be used to buy management support to implement other agile techniques. This paper exposes to the community an experience report of a large government agency's migration from a Waterfall process to an iterative methodology, the Rational Unified Process (RUP). Through field observations and semi-formal interviews with key business partners, we found five main areas of improvement: reestablishment of business involvement, better distribution of acceptance testing effort, introduction of a testing team, less pushback on necessary changes, improved communication and management of expectations.

Keywords: iterative development, agile techniques, business value, Rational Unified Process, acceptance testing.

1. Background

This paper contributes to an understanding of the business advantages in adopting an iterative development practice in a bureaucratic industrial setting. Such an understanding is important as many business leaders prefer to adopt processes that have been successfully implemented by others, to reduce the risk of failure [1]. The company under study is a large Oil & Gas government agency that lacked the initial management support to adopt mainstream agile methods. This agency has a workforce of 900+ employees, with a large IT department comprised of over 10 different IT Programs. This study collected field observations and interviews from the key business representatives of the largest IT Program in the corporation, focusing on a set of three existing applications, and two newly developed systems. These multi-million dollar projects support business critical functions, such as the digital submission of information, internal processing of such information, and the publishing of the results to the public.

The original IT vision was to develop a simple solution to provide a central data management point to the business partners. The first release commenced in early 2001, with a group of 4-6 developers. The development team did not formally adopt a development methodology, but was following a Waterfall approach: gather all the system requirements, then develop the entire application, which is at last handed off to the business partners for testing and approval. After the first few releases, a new vision for a workflow system, that would allow digital submission of data for quicker turn-around times, was born. By 2004, the team increased to over 15 developers, with a total of 40+ team members (including business analysts, technical support, and managers). The former Waterfall process was not able to support the increasing pace of development, and many releases were delayed, resulting in poor software quality and cost overruns. Late in 2004, the corporation decided to adopt the IBM Rational Unified Process (RUP) [2], as it provided an iterative development approach as well as the degree of formality and traceability desired by the top-level management. Many Agilists consider the Rational Unified framework heavyweight, but since its inception in 1998, the RUP framework has been customized to fit more agile environments [4, 5] and the company adopted such a lighter version.

The adoption stages were identified as: pre-RUP, transition to RUP, and partial RUP adoption. IBM suggests an iterative approach to the RUP implementation, "adoption through execution" [2]. The company's current execution state includes: the iterative RUP lifecycle (inception, elaboration, construction, and transition), Rational Tools, role sets, and selected work products (Design and Use-Case Models, Software Architecture Document, Iteration Plan and Assessment, Risk List, Issues List, Test Case, amongst others). The partial RUP adoption refers to the pre-existing projects, as they did not benefit from the iterative approach since inception, missing the majority of the exercises that result in the above mentioned work products.

The Rational tools that were adopted during the transition stages included software for source code repository management, requirements gathering, and bug logging. The pre-existing systems moved to a spiral approach, where development was conducted in mini waterfall cycles of analysis, development, and testing, with release dates being booked according to business needs. The team later moved to scheduled releases, which are comprised of time boxed 6 week iterations. The two new projects followed the adopted RUP framework since inception.

2. Findings and Observations

Reestablishment of business involvement

During the first system release, the small team atmosphere allowed the business partners to have an active role in the requirements gathering stages of the system development. The project manager would set up business meetings with the involved stakeholders to gather requirements. Some requirements were documented in Word or Excel, others were only verbally communicated to the development team. Later, the development team would create screen mock-ups of the application, and present them to the business partners for feedback during meetings. Although acceptance testing did not occur until development was completed, business partners found the screen shoots extremely useful: *"even though we didn't get to test until the end, when we got the application, it was not about testing the screens and see how they looked like, it was testing to see if they worked, if they met the requirements."* They were very pleased with the first release of the system, which took approximately one year to be production ready. As the number of requirements increased, so did the IT team size.

More rigorous management procedures were put into place. Developers needed to follow the project plan more closely, in some cases resulting in frustration, as the plan was quickly outdated. Business partners were used to contacting developers with requests, who would in turn implement the requirements, causing a delay to the defined project plan, also found by Blotner [3]. As a result, managers prohibited business partners from contacting developers directly: *"we got cut off by management: 'that's it, no more talking to the developers!'"* It got to the point where management would complain about e-mails sent to developers by business: *"don't be seen talking to a developer, [...] and really, that environment was not good. For us that doesn't work!"* Business partners felt that they lost the element of teamwork, causing friction and *"blaming games"* between IT management and Business, which was *"very disruptive to everyone involved."*

The introduction of the six week iterations has helped business partners become more involved in the iteration planning, by prioritizing which items need to be worked on first, and which ones require more analysis. They feel more ownership and accountability over the decisions made, which has helped rebuild the teamwork [1]. They are now allowed to contact developers: *"a developer came and sat with me [to discuss a task] and mocked it up in paper, and asked if it was ok with me, which was fantastic."* Still, involvement with developers is limited, as most of the communication goes through the project leaders and business analysts. Perhaps this can be attributed to the responsibilities defined in the RUP roles. Business feels that this "*middleman*" approach to communication has advantages, when dealing with developers that lack interpersonal skills, and drawbacks, as information gets "*lost in translation.*" To mitigate this issue, key developers are invited to business meetings.

Business partners feel that the most visible gains come from the new systems that started development using the iterative RUP process, as they were involved in the process since inception. They were not given functional parts of the system to test until the construction stages, but they had iteration assessment meetings where demos were provided, allowing feedback on system functionality. As a result, the first full iteratively implemented system was the first project in more than six years to be delivered on-time and on-budget: *"which is significant for the organization, the first in years, [laughs] that says a lot. Our executive was very happy, from our perspective [it] is great."*

Better distribution of Acceptance Testing effort

The three business managers, corresponding section leads, and a few senior end-users conduct acceptance testing. The interviewed business partners felt that the original development process did not provide reasonable time for testing the system: "*you would get it [the application] for two days, and you need to approve it and its gotta go.*" They felt rushed and uncomfortable by having to sign-off on a system that took over 10 months to develop, and only a few days to test. At the end of the development cycle, business had compounded testing to do, which caused an overwhelming workload: *"[testing] is not my full time job. I need to deal with core business. Testing work is supposed to be on the side, but [at that point] becomes fulltime work. I am basically doing two fulltime jobs, which makes things difficult."* Iterative development has time boxed the testing effort required by business to two weeks per iteration. Testing is not compounded, but it can still feel rushed based on the number of changes implemented during the iteration. The business partners see the organized and scheduled acceptance testing effort as a big improvement: *"it is better to plan that way, even from a personal life perspective. It is just way more organized than it used to be."* Some interviewees actually stated that this organized schedule is the major improvement provided by the process changes made to the existing projects.

Introduction of testing team
A Quality Assurance (QA) team was not available in the pre-RUP stage, as management perceived formal testing as peripheral in comparison with other more pressing deliverables. The code would go from the developers who did not implement any automated tests, to the business partners for testing: *"we used to joke around saying what is the point? I open it [the application] and get the 'yellow screen of death*[1]*', so you are just wasting my time!"* As suggested by the six key RUP principles for business-driven development, management hired a full-time testing team at the end of the third transitional iteration. After the introduction of the testing team, all code goes through a round of formal testing before getting into the hands of the business users, and as a result the business partners find fewer fatal errors during acceptance testing. They can also focus on the areas that have been changed or included, as the testing team is responsible for the regression testing, which is considered a big time saver: *"it is night and day."* However, the testing required by the QA is complex and time consuming. The QA team is shared between all projects, and may not have enough resources to provide the appropriate levels of manual regression testing. That, in addition to the lack of unit tests, has been a sore spot for pre-existing systems, having problems reappear in production after being fixed. New systems are now implementing unit tests, which allow developers to regression test the application even before it is handed off to the formal testing team.

Less pushback on necessary changes
In the former Waterfall process, IT management would push back to implement changes: *"so you get stuck with it."* It is very difficult for business partners to define the project's scope to the degree of granularity needed at the initial requirements gathering stages: *"it is virtually impossible to foresee all the details and functionality of an application to define a hard scope document. To expect that when creating a scope document is unreasonable and shortsighted."* Business partners would have to make go-no-go decisions close to the production date, and many releases were delayed as much as a year due to poor testing results, and essential requirements being missed in the original scope document.

Iterative development has provided business with a set release schedule that are 6 weeks apart from each other, allowing critical items to be negotiated, prioritized and included in the next release. Also, for new development projects, the iteration assessments and demos allowed business partners to provide the feedback necessary to avoid major changes later on in the process. A visible result of that is the number of

[1] This refers to the fatal application errors in .Net, which display the error message in a yellow screen.

bugs[2] in production for the first system developed using RUP, which is less than a dozen compared to the hundreds found in the former ad-hoc projects.

Improved communication and management of expectations
In the former process, issues were logged in Excel spreadsheets, discussed in business meetings, prioritized and put away in a place only accessible to managers. The RUP adoption involved the adoption of Rational Tools including a bug and enhancement logging software. Business partners have access to these tools, being able to view what is outstanding, which is very important to assist them in negotiations of shared resources, and to have more realistic expectations of what and when changes will be delivered. They also feel that overall the projects are much more organized, and due to the iterations, they are in constant communication with the team, which helps reduce *"surprises"* at the end of a release cycle.

3. Implications for practice

The interviewed business partners see the adoption of the iterative RUP process as a definite benefit to the organization, with particular improvements in the areas of organization, communication, accountability, teamwork, and acceptance testing effort. In a bureaucratic governmental environment, bound to set regulations and continuous auditing, the patented IBM RUP framework provided the initial social change required to open the doors to other agile techniques. Management is now providing support for test and continuous integration automation, and new projects are holding daily stand-up meetings with the presence of involved business partners.

References

1. Hartman, Francis. Don't Park Your Brain Outside. Project Management Institute 2000.
2. Barnes, Joshua. Implementing the IBM rational unified process and solutions: a guide to improving your software development capability and maturity. IBM Press 2007.
3. Blotner, J. A. 2002. Agile techniques to avoid firefighting at a start-up. In OOPSLA 2002 Practitioners Reports (Seattle, Washington, November 04 - 08, 2002). OOPSLA '02. ACM, New York, NY, 1-ff.
4. Hirsch, M.. 2002. Making RUP agile. In OOPSLA 2002 Practitioners Reports (Seattle, Washington, November 04 - 08, 2002). OOPSLA '02. ACM, New York, NY, 1-ff.
5. Ambler, S., Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process, John Wiley & Sons, Inc., New York, NY, 2002

[2] Interviewees used the words "bug" and "defect" interchangeably, and both refer collectively to faults and failures. Enhancement requests were logged separately.