Moving towards agility in a bureaucratic environment – RUP as a bridge between Waterfall and Agile processes

Caryna Pinheiro, Frank Maurer, Jonathan Sillito
Department of Computer Science, University of Calgary
{capinhei,frank.maurer,sillito}@ucalgary.ca

Abstract

Transitioning from waterfall to agile processes can be difficult especially for organizations lacking top-level management support. To explore this topic, we collected qualitative and quantitative data about three projects at a government agency transitioning from a waterfall process to the IBM Rational Unified Process. The results of our analysis of this data identify some of the challenges organizations face in making process changes, and demonstrate that the iterative aspect of RUP together with investment into test teams and re-factoring can succeed in improving software quality and software stability. Based on our data we also argue that RUP can provide gentle ways to introduce even lighter agile approaches into bureaucratic, trust challenged environments that initially are lacking management support for agility.

1. Introduction

Agile methods foster an environment of collaboration, work ownership, and accountability. Recent literature provides many experience reports on the move from waterfall to Scrum, supported and promoted in a top-down fashion, from leaders to initially skeptical engineering groups [3,4,9,10]. But how can we move towards agility in a risk adverse bureaucratic environment without the strong commitment of the top management? The answer seems straightforward: gain their support! The real question lies in how to earn that support, and more over, how to change a trust challenged culture into a collaborative trust-driven environment.

This paper reports on a longitudinal case study of a large government agency's migration from an ad-hoc waterfall process to an iterative development framework, the Rational Unified Process (RUP) [2]. Many Agilists question the agility of the RUP framework. Perhaps this can be attributed to the early heavyweight out-of-the-box implementations of the RUP framework before the Agile methods became widely used [5], leading to a common misunderstanding of the RUP philosophy. On the other side, since its inception in 1998, the RUP framework has been customized to fit more agile environments [1,5].

This paper - through the analysis of the quantitative and qualitative data gathered - provides an empirical perspective of the improvements to software quality and software stability resulted by the adoption of RUP. We also discuss how these improvements have provided gentle ways to introduce Agile-related concepts and practices into an environment that lacked the initial management support for agility.

2. Company Background

We conducted a case study in a large Oil & Gas government agency. This agency has a large IT department comprised of 10+ different IT programs. Our study collected data from three complex projects within the largest IT Program in the corporation: two on-line applications, and one client-server application. The client representatives for these applications were the internal Business Area Leaders, from herein referred to as business clients. These leaders held regular focus groups with industry users.

A group of 4-6 developers commenced development for the first release of the client-server application and the external publishing site in early 2001. The development team informally followed a waterfall development approach: gather all the system requirements, develop the entire application, and hand it off to the business clients for testing and approval.

By 2004, the small team evolved to over 12 developers, with a total of 40+ team members (including business clients, business analysts, managers, and technical support). The former waterfall

process was not able to support the growing and fast pace of development, with many releases being delayed or rushed, resulting in poor and unstable software quality and low team morale. The business clients communicated to IT management the need for better time estimates, better quality, and more reliable testing schedules. After considering many options, and consulting with external vendors, top-level management decided to adopt the IBM Rational Unified Process. IBM's branding in combination with the suite of supporting tools and on-line training provided enough confidence in the process.

3. Case Study

This study covers three different process stages, thus the data was grouped into three time periods: waterfall, which we refer to as to as Pre-RUP, Transition, and Partial RUP adoption. The company followed the IBM iterative approach to the RUP implementation: "adoption through execution" [2].

During the Transition stage, the existing projects, such as the ones under study, only adopted the Rational tools and moved to a spiral approach, conducing development in mini waterfall cycles of analysis, development, and manual user acceptance testing. There was no formal testing by a test team or automated testing, and release dates were booked according to clients needs. After the first Transition release, the team moved to non-fixed iterations (defined by the number of tasks). By the seventh release of the new code base, considered by our interviewees as the first Partial RUP release, the team adopted time boxed 6-week iterations, with tasks being prioritized to fit this timeframe. While new systems were mandated to follow the emerging RUP framework since inception, including: the iterative RUP lifecycle, rational tools, role sets, and selected work products, the RUP adoption of the projects under study principally consisted of the implementation of an iterative development process to an evolving code base.

Data Collection. We collected both quantitative and qualitative data. Quantitative data gathering included collecting information from the bug databases and source control repositories for a three-year time span for three projects. The word "bug" and "defect" are used interchangeably, and both refer collectively to faults and failures [6]. Bug reports were extracted from IBM Rational ClearQuest, which also contained reports migrated from older logging systems. Code activity data was extracted from Microsoft Visual Source Safe and IBM Rational ClearCase.

We define bug-density as the number of bugs found per one thousand lines of code changed, added or deleted (KLOC) in the source control repository, to take into account the varied levels of effort per release, as suggested by Mohagheghi *et al.* [6].

Qualitative data gathering included several months of field observations and seven semi-formal interviews. The interviews were designed to acquire further insights into the quantitative bug-density results and to identify the perceived business value brought by the process changes. Interviews were conducted using the responsive interviewing technique [8] and the interview and observation data was analyzed using a grounded theory approach. Field observations and interviews provided valuable help in understanding anomalies in the bug-density data.

Data Limitations. The industrial data collected had limitations and data gaps. Reliable and complete bug data was unavailable prior to July 2004. Thus, the data sample used for the Pre-RUP period starts with release six (R6) of the systems, dated from July 2004. It is also important to note that no formal logging process was in place in 2004, and developers indicated the possibility that not all bugs were being consistently logged.

Another data limitation includes the loss of code activity for five months of the first Transition RUP release due to technical issues with the source control repository. The lost data comprised mostly proof of concept code. To deal with this limitation, we used the number of lines of code of new files as the activity data for that period. Since both limitations favor the waterfall process – potentially less bug reports, and potentially higher bug-density for the first RUP Transition release – they don't pose a strong threat to the positive findings in this paper.

4. Findings and Observations

4.1. Iterative development, aided by code re-factoring and formal testing, improved software quality and software stability

As mentioned in the previous section, we calculated the bug-density for all available releases and grouped the releases into three stages: waterfall (Pre-RUP), Transition, and Partial RUP adoption. Figure 1 presents the bug-density distribution and severity breakdown for all releases, grouped by stage.

Pre-RUP. The interviewees indicated that the last three releases of the Pre-RUP period (R14, R15, R16) only included minor patches to the client application, and are not representative of a standard release, thus

they were not used in our data analysis. The standard deviation in the bug-density was 6.26 between the releases in the Pre-RUP period. This standard deviation is an indication of the instability of the software being delivered, which was confirmed by the interviews with developers and business clients. This instability generated issues of trust and blaming between IT and business clients. The highest bug-density encountered in this stage was 15.96 bugs/KLOC, which is also the last "standard" release of this legacy and mature code-base.

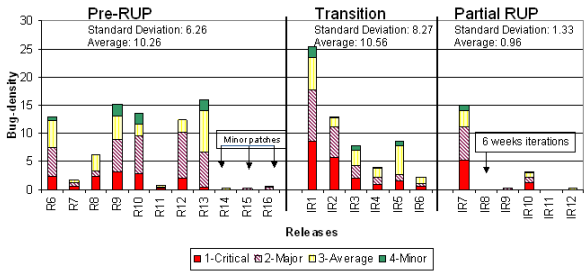

Figure 1. Bug-density and severity distribution.

RUP Transition. The standard deviation of 8.27 in bug-density, as well as the highest data spike (25.33 bugs/KLOC) indicate that things actually got worse during the transition period, which corresponds to many of the experiences of large companies migrating to agile [3,4,9,10]. From our semi-formal interviews and field observations, we found that in addition to challenges involved in transitioning to a new process, the following factors also contributed to these results:

Adoption of new technologies and implementation of new complex functionality. On top of the organizational adoption of the IBM Rational tools and selected RUP guidelines, an Enterprise initiative required that the high profile systems move to the .NET framework, to follow a new OOAD development approach, supported by an in-house development framework created by the architects.

Increased development team size. The development team more than doubled in size during the first Transition RUP release. Due to the lack of business knowledge, new developers would fix an issue, but create new bugs. Accordingly, the first release of the new code base, which really did not follow an iterative approach, took over ten months to develop and had the production date delayed four times. The team attributed this failure to the limited RUP adoption, *"honestly, it felt like nothing had changed, other than we had to do more documentation."* As a result, the transition approach for existing systems was reviewed. During the second RUP Transition release (IR2) a formal testing team was hired and a formal bug logging procedure was introduced, to ensure that all bugs, even minor cosmetic issues, were logged and visible to all stakeholders.

Partial RUP. The standard deviation of 1.33 in bug-density between the releases after the Partial RUP adoption indicates an increase to software stability. The lowest average of 0.96 also indicates an increase in the overall software quality. It is important to discuss some interesting points in the data: the developers attribute the high bug-density of the first Partial adoption release (IR7) to a major system re-factoring done without automated unit tests. The re-factoring was undertaken to deal with design flaws in the system, especially in the security transactions and in-house development framework usage. The developers saw this effort as an outlier to the standard releases, thus it is not included in our data analysis. They believe the re-factoring was a very positive exercise that ultimately added to the overall system reliability and quality. Prior to RUP, management considered re-factoring and the development of automated tests peripheral activities. The developers believe that management came to support the re-factoring due to improved communication channels supported by the iterations. On the other hand, they were disappointed by the lack of management support to introduce automated unit tests to the development suite during the RUP Transition. The team attributes the spike seen in IR10 to another upgrade of the system to support a new in-house framework version.

4.2. Iterative development increased customer satisfaction

On figure 2, we isolated the bug-density of bugs logged by business clients. The quantitative analysis again points out that the Transition stage presented challenges. However, after the first Transition release the bug-density logged by business clients dropped. The Partial RUP adoption stage has the lowest average of bug-density (0.35), against the Transition (3.40) and Pre-RUP (4.37) stages, as well as the lowest standard deviation (0.35), indicating an increase in the quality and stability of the software being delivered to user acceptance testing.

During the initial Pre-RUP development stages, the communication channels were open between clients and the small development team. However, after the team increased and the waterfall process broke down, management prohibited business clients from contacting developers: *"at one point it was so bad that, if we were seen talking to a developer, we would get a*

nasty e-mail from management." Business clients missed the small team atmosphere and felt that they had *"lost the team work."*

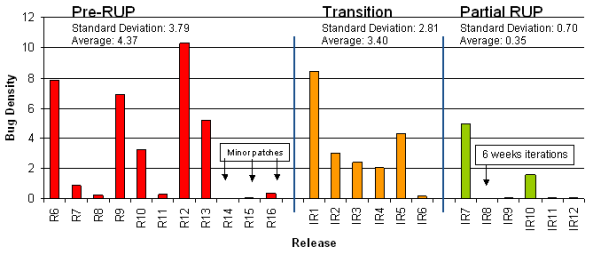

Figure 2. Bug-density logged by business clients.

When iterative development was adopted, the communication channels were slowly re-opened through prioritization meetings, iteration planning sessions, iteration assessments, and iteration user acceptance testing. Any new system being developed at the company is now using a more complete RUP process, thus providing constant opportunities for business clients' feedback, and welcoming business clients to drop by and view the progress at any time.

Our interviews with the business clients and field observations revealed that iterative development provided: better distribution of acceptance testing effort, less pushback on necessary changes, reestablishment of business involvement, introduction of a testing team, improved communication and management of expectations [7].

5. Implications for adopting other Agile concepts and practices

Our findings indicated that the transition to RUP was not smooth. Lack of direction, compounded by new technologies and growing development teams, resulted in a temporary decrease of software quality. However, three things supported by the new RUP process assisted the team to get back on track: iterative development, the introduction of a testing team, and the major re-factoring of the code base. We asked team members what they think would have happened if these challenges arouse from a direct transition to an agile methodology: *"we would be back to waterfall at no time."* However, since management supported RUP, it survived the transition stage. And due to its perceived success, management is now more open to other Agile-related concepts. Currently, more agile techniques are being adopted: automated functional unit testing (including TDD practices by some teams), the hiring of a continuous integration expert, the holding of daily stand-up meetings, and some teams are replacing use cases with user stories. There is also an effort to minimize the amount of required work products, which has been surprisingly well received by upper management.

6. References

[1] S. Ambler. The agile unified process. June 2006. http://www.ambysoft.com/unifiedprocess/agileUP.html Downloaded: February 27 18:00pm.

[2] J. Barnes. Implementing the IBM rational unified process and solutions: a guide to improving your software development capability and maturity. IBM Press, 2007.

[3] P. A. Beavers. Managing a large "agile" software engineering organization. AGILE 2007, pages 296–303. IEEE 2007.

[4] C. Fry and S. Greene. Large scale agile transformation in an on-demand world. AGILE 2007, pages 136–142. IEEE 2007.

[5] M. Hirsch. Making RUP agile. OOPSLA 2002 Practitioners Reports, pages 1–ff. ACM Press 2002.

[6] P. Mohagheghi, R. Conradi, and J. A. Borretzen. Revisiting the problem of using problem reports for quality assessment. In Proceeding of the 2006 international workshop on Software quality, pages 45–50. ACM 2006.

[7] C. Pinheiro, F. Maurer, and J. Sillito. Adopting iterative development: the perceived business value. In Proceeding of the 9th Agile Processes and Extreme Programming, pages 185–189. Springer, 2008.

[8] H. J. Rubin and I. S. Rubin. Qualitative Interviewing: The
Art of Hearing Data (2nd Edition). SAGE Publications, 2005.

[9] A. Ruhnow. Consciously evolving an agile team. AGILE 2007, pages 130–135. IEEE 2007.

[10] T. R. Seffernick. Enabling agile in a large organization our journey down the yellow brick road. AGILE 2007, pages 200–206. IEEE 2007.