Agile Software Development Meets Corporate Deployment Procedures: Stretching the Agile Envelope

Olly Gotel[1] and David Leip[2]

[1] Department of Computer Science, Pace University, New York
ogotel@pace.edu
[2] *ibm.com* Chief Innovation Dude and Agile Methods Advocate, IBM Hawthorne
leip@us.ibm.com

Abstract. This paper describes a process initiative within IBM to make the Corporate Portal (*ibm.com*) development practices more responsive to changing customer needs and explains the bottlenecks that arose with application deployment when this agile approach was not initially extended throughout the wider solution delivery lifecycle. The paper details the simple process changes that were adopted to expand the agile philosophy beyond development.

Keywords: Agile Deployment, Agile Development, Extreme Programming.

1 Introduction

The IBM Corporate Portal (*ibm.com*) is an application-driven website that advertises and markets the products and services of IBM. Prior to 2004, the team responsible for its development followed a "waterfall-like" approach, attempting to capture and prioritize requirements for a new release before design and development. They found that, while straightforward to agree on the main requirements, reaching agreement on the complete set was problematic. Negotiations would introduce delays and slow down releases to customers. A more agile approach was viewed as a pragmatic way to tackle this issue; development would proceed from key requirements and additional requirements would be determined and prioritized as the product evolved.

In 2004, the IBM Corporate Webmaster's team adopted an agile approach to software development. The use of eXtreme Programming was trialed in the development of a major release of *ibm.com*, culminating in its successful roll-out in November 2004 [2]. The agile process has since been streamlined to develop and deliver software that addresses evolving customer needs within a reduced timescale. However, once a new release has been developed, it still has to be deployed to a production environment to deliver a fully operational solution. The deployment environment is maintained by a geographically distinct team. This team orchestrates the deployment of multiple applications from across IBM, so their work is subject to organizational-wide scheduling requirements and policies; the team has to ensure any new deploys do not impact other deployment schedules and live applications.

Limited improvements can be gained in end-to-end agility if there is a bottleneck between the development of a product and its eventual deployment. While there is

G. Concas et al. (Eds.): XP 2007, LNCS 4536, pp. 24–27, 2007.

advice on how to align agile with non-agile practices [3], there is less focus on deployment [1]. This paper reports on the issues surrounding the alignment of agile development practices with corporate deployment procedures and describes the agile-spirited end-to-end process adopted for *ibm.com*.

2 Agile Development for *ibm.com*

The Corporate Webmaster's team is responsible for developing and maintaining *ibm.com*. The requirements for *ibm.com* change frequently, driven by new business offerings, improvements in search engine technology, etc. Part of IBM Sales and Distribution, the team comprises some 20 personnel, skilled in Java and XML/XSL development, open and technical standards, and project management. The majority of the team is based in New York State, with other members based in Asia and Europe.

The deployment team is drawn from roughly 100 personnel within IBM Global Services. This team runs the technology for IBM-sponsored events, like Wimbledon, in addition to being responsible for the *ibm.com* infrastructure and operations. The Webmaster's team are hence one of a number of customers for the deployment team's services. The deployment team is responsible for estimating demand on servers, understanding network traffic requirements for new or changing applications, network settings and permissions, etc. They are also responsible for testing that new applications meet non-functional requirements, do not compromise the performance or availability of existing applications, and checking that applications are compliant with organizational security policies. The team is based in Raleigh, North Carolina.

The agile development process required the development team to work 7 week release cycles, each cycle comprising 3 iterations of 2 week duration followed by a week for deployment. Customer groups from across IBM submit high-level requirements to be reviewed and prioritized based on business value. Selected requirements are split into and/or reformulated as stories by customer representatives and written on index cards. The stories are sized by the development team according to the time they estimate they need to build the story and returned to the customer, together with their estimated velocity for the iteration. Since the development team is distributed, the velocity is for the extended team. The sizing is based on the expected development effort (Java and XML/XSL), not on the associated deployment effort. The customer selects stories to implement in the forthcoming iteration. This selection process takes place every first Monday in a meeting at the start of each 2 week cycle. Further interaction with the customer occurs as stories are clarified and developed, and any elaboration is written on the back of the card. The extended development team communicates via telephone and instant messaging during this time. Towards the end of the iteration, the team performs user/customer acceptance testing.

During this process, the deployment team should (ideally) be notified of any story that may have a deployment implication. One scheduled call takes place every Tuesday to give the deployment team an alert as to what may be coming through the process and some indication of timeline. The first formal notification the team has is when the development team submits a work request for a code review as a precursor to deploying into staging. This occurs around the beginning of the third iteration (i.e. week 5 in the cycle) but, as the development team began to take on more of the

responsibility for code review, this removed the necessity to put in such a request. Once the application has been deployed, any issues are dealt with by the joint team.

3 Problem Description

A retrospective was undertaken at the end of 2004 and the benefits from the new process were seen to be the increased level of communication between the customer representatives and development team. Since the customers are able to determine and prioritize requirements on a regular basis, they had more control over the product's direction. This retrospective revealed tensions, however, at the bottlenecks in deployment. From the deployment team's perspective, the last minute hand-over of applications and expectation of a fast release into production indicated the developers were not cognizant of the other demands on their time and corporate constraints.

With the prior "waterfall-based" approach, the deployment team would receive a requirements document and use this to plan and estimate their work for scheduled deploys. Following the move to agile, this was replaced by paper-based story cards and the team would often only find out about deployment requirements with the request to deploy into staging. When this request came late, the seventh week in the overall cycle would get stretched, leading to delays; the deployment team could not abandon other work to accommodate this. With the move to agile there had been little focus on the processes required to manage the transition of products from the development environment to the corporate production environment. It relied on ad-hoc communications between individuals and tacit institutional knowledge.

4 End-to-End Agile

Both teams have demands on their time and conflicting priorities, and are in separate parts of the IBM organizational chart. It is therefore critical to gain an awareness of the wider culture, working practices, constituency and remit of each other to understand what is and is not feasible. A model of the prior end-to-end process was constructed to clarify the tasks, timelines and information needs of both teams. This included the role of artifacts that mediate communications and so help to synchronize efforts (e.g. meetings, phone calls, requirements documents, etc.) This was compared with a model of the agile process to get an idea of where there were significant changes in the cross-team communication and possible information gaps. It was found that the deployment team did not need intricate details about the application, just specific and quite regular information pertaining to deployment. It was anticipated the deployment team could provide an information checklist for development (acting like triggers to talk), thereby affording some lead time for planning and decision making.

The notion of who is a customer and who is a service provider changes throughout the end-to-end process. The development team effectively assumes a dual role as intermediary – supplier to the business customer and customer for the deployment team's services. The very nature of this shift in relationship means that the development team needs to rethink their interaction point: when they are customers, do they behave in an agile manner or does their agile thinking come to a stop? Examining

how to reduce cycle times within this wider chain led to the introduction of time-boxes (with velocities) for the deployment team, which the development team could apportion in a "lock-and-load" manner. All the *ibm.com* deploys were thus scheduled to take place on a Monday/Thursday cycle (i.e. Monday to deploy to staging and Thursday to deploy to production). On a Friday, the development team would submit work requests for applications to be deployed in the following week's cycle. The deployment team would expect the code at 9am on Monday else release their time to other customers. Monday through Wednesday the development team would be in testing and on Wednesday the production request would go in for deployment on the Thursday. Extending the metaphor such that the development team received a set amount of scheduled effort, and making them responsible for deciding how to prioritize their demands and use this resource, extended the agile envelope.

5 Future Considerations

This paper has highlighted how deployment can easily be an afterthought in agile process initiatives. Since May 2006, the simple changes described above have resulted in a more agile end-to-end process for product development and solution delivery within *ibm.com*. A number of outstanding questions remain:

Scalability. Only one of the deployment team's customers works in an agile manner, while the other teams plan and pre-schedule releases up to a few months in advance. Would this model scale if all customers were to work in this way?

Whole team velocity. Is it more useful, in terms of end-to-end agility, to consider the teams separately or as one whole team with a single velocity?

Story structure. Does it make additional sense to augment customer stories with deployment aspects or to create separate deployment stories to chain with stories?

Accounting for the REAL end-to-end process. This initiative focuses on the latter stages of an evolving product's lifecycle. Are there initial upstream stakeholders and processes that can also be brought into better alignment for further agility?

Acknowledgments. We would like to thank the IBM staff who assisted in this work.

References

1. Ambler, S.: One Piece at a Time: Just because agile practitioners deliver working software weekly doesn't mean that they deploy it into production at the same rate. Dr. Dobbs Portal, November 9th (2004)
2. Grossman, F., Bergin, J., Leip, D., Merritt, S., Gotel, O.: One XP Experience: Introducing Agile (XP) Software Development into a Culture that is Willing but not Ready. In: Proceedings of CASCON, Markham, Ontario, Canada, pp. 242–254 (2004)
3. McMahon, P.E.: Bridging Agile and Traditional Development Methods: A Project Management Perspective. CrossTalk: The Journal of Defense Software Engineering (May 2004)