Sprint Planning with a Digital Aid Tool: Lessons Learnt

Erlend Agøy Engum
National Oilwell Varco
Stavanger, Norway
eaengum@yahoo.no

Zornitza Racheva, Maya Daneva
University of Twente
Enschede, The Netherlands
{z.racheva, m.daneva}@utwente.nl

Abstract - **Managing the product's backlog is a major task in agile projects. This case study reports on one organization's experiences from the transition to a backlog management tool and its contribution to improving sprint planning.. Our key lessons learnt are that a tool is particularly appropriate to organize and specify backlog items in a transparent manner and to handle dependencies. However, we also observed an overhead in backlog management and in reporting during meetings. The concrete project settings play the paramount role in whether such a tool helps or harms the process.**

Keywords: agile development, sprint planning, process improvement

I. INTRODUCTION

Agile software development became a prominent solution coping with two weaknesses of the 'traditional' development methods, namely, intolerance to changes and long development cycles. Agile methods are incremental and iterative and promise the clients fast delivery of value and ability to accommodate changing requirements during the development [1]. Although the core agile practices [6,12] seem intuitive and easy to apply, empirical studies [3,8] indicate that their implementation requires experience, discipline and motivation. This poses a serious challenge especially to inexperienced teams [13]. For example, when the process is not well tuned, and developers lack agile experience, it is difficult to achieve the efficiency and flexibility that the agile followers claim to be the particularity of this development method. This is specifically true for requirements management. The requirements, the so-called features in agile project, are organized as a list, named Product Backlog (PB). Unlike in the traditional development, the PB in an agile project changes often, which necessitates frequent (re-) prioritization of the requirements. As the development is structured in the form of short cycles, it is not possible to include many features in an iteration. This, in turn, means that those features left aside from each iteration should be prioritized again in the next iterations. Moreover, the appearance of new features on the list also triggers prioritization. Therefore, an effective procedure is needed helping both the product owner and the project manager to keep track of the requirements and the changes in the PB. In young and small agile companies, a common practice is to use for this purpose a spreadsheet complemented with the use of hand-written post-it notes. While simple and easy to apply, this method brings some serious problems: (i) it renders developers inefficient because of not enough information available to them, which leads to misunderstandings about the exact task to be developed, (ii) it creates ambiguity issues as individual interpretations dominate over a shared understanding across project participants, which, in turn, makes the exchange of tacit knowledge difficult, and (iii) it is overstrained when there are dependencies between teams and/or team members, where coordination and knowledge sharing is instrumental to project success.

This paper reports on a case study in a North-European small agile company transitioning to a tool for managing the PB and the Sprint Backlog (SB). We attempted to distill lessons learnt from the case study company's experiences. We make the note that previous studies by other authors [7] indicate the necessity of providing more empirical research and giving more attention to management-oriented approaches in agile context. As Lindvall et al. [7] says, "collection and analysis of empirical evidence of the effectiveness and classification of appropriate environments for Agile projects has not been conducted". We consider our study a step in this direction, as we (i) collect empirical evidence; and (ii) give an insight about the concrete environment. In what follows we describe the research process (Sect. 2), provide the background and the context of the project (Sect. 3), present the application of the digital tool to improve the prioritization process (Sect 4) and report on our lessons learnt (Sect 5).

II. RESEARCH APPROACH

We carried out an explorative case study by using the qualitative research practices recommended in [4,15]. The goals of our explorative study were: (i) to observe the state of the practice when using two different process approaches, namely with and without tool support; (ii) to compare the observations concerning the challenges the particular team faced; (iii) to formulate hypotheses based on the observations, that can be validated in further case studies. In this paper, we report on those research results pertaining to the first two goals. We make the note that the results related to the third goal are out of the scope of this paper. Our research process included (i) collecting experiences about what worked and what did not in requirements reprioritization, (ii) categorizing these experiences, (iii) discerning key themes and concepts, and (iv) sense-making of our leanings. We used the constant comparison and coding techniques of Grounded Theory [4]. Because this paper is focused on the lessons learnt, we do not discuss in detail the theoretical foundations of the case study approach we followed. We deliberately put in the foreground what we distilled as lessons which other practitioners in other agile companies might find applicable to their practice. We are set out to present problems or challenges encountered in

978-0-7695-3784-9/09 $26.00 © 2009 IEEE
DOI 10.1109/SEAA.2009.68

practice, to relate success and failure stories, and to report on industrial practice. Therefore, the focus is on 'what' and on lessons learnt, not on an in-depth analysis of 'why'. We will describe the agile practices used and provide information about the context, so that readers should be able to draw conclusions for their own practice.

III. Background

Our case study site is a small and growing IT company serving the travel industry by providing to them web sites, payment solutions and search services. The first author (Engum) held a scrum master role for a development team in this organization. Experiences are gathered from a total of ten sprints of different duration, with and without a tool to assist the planning process.

Project context: The scrum team included six to seven developers, half of them working part-time, as the company hires students in part-time positions along with more experienced developers. There is no trained tester, no technical writer and no other non-development roles filled. One team member holds the role as scrum master. The product owner is not part of the development team but is responsible for handling customer contact, the requirements specification and the PB. When the company introduced Scrum it did not send product owners, scrum masters or the scrum team to formal training. The process introduced was based on the reading of books, e.g. [12] and a package with other materials. In the remainder of this paper we refer to scrum terminology as defined in these books.

The duration of each sprint was set individually, based on the work demanded for implementing the features and on the customers' and other development teams' deadlines. In practice, sprints were varying from one day to three weeks.

Initial state: We report on the development of a new, custom built content management system (CMS) and a related web service used for powering web sites. The demands are high as the successful implementation of new features is used to automate the building of several existing and new web sites.

The PB and the SB were both initially handled by spreadsheets. Tracking tasks during the sprint was done on a scrum board where the tasks were written on magnetic stickers which had the size of post-its. The description of issues on the PB and SB were poorly written and most of the information was kept with those professionals issuing requests and/or performing tasks. A review and retrospective meeting are held at the end of each sprint, which are to identify two types of experiences: "what went well" and "what can we improve" (impediments) from the last sprint. The latter get divided into two categories, "team" and "organization", and are ordered by priority. Prioritization is done by pair-wise comparison of one impediment to another. The team discusses the experienced or perceived effect and allocates higher priority to higher impact impediments. The process is continued iteratively until all impediments are dealt with. We make the note that we allow for equal priority impediments and for merging of very similar impediments. The Sprint 4 retrospective meeting of "sprint 4" identified a set of nine impediments that needed to be overcome, in order to improve the efficiency of development.

An important part of this process is that the reviews encouraged both the management and the team to look for solutions to improve the process. After analyzing the problematic processes, the project manager was clear on that better task description and tracking tool was needed. A decision was made to introduce the generic issue management tool, Redmine [10], in order to help organize, specify and prioritize the PB and track development and time usage. Based on their collective experience, the team expected that the tool would provide improvement to three out of the nine identified impediments described in detail below:

Priority 1: *Poor or insufficient specification of tasks during sprint planning.* The members of the team differ in experience and in knowledge of the system being built. This means, much tacit knowledge is involved and the person writing the one or two line task definition often omits information of great importance to the performing team member. This generated misunderstandings which lead to an inferior product and frustration inside the team.

Priority 2: *Lack of identification of dependencies to other teams.* A particular aspect of the company's context are the dependencies between different teams, similar to a supplier network situation [2]. For example, several of the tasks the team performs include interaction and testing together with other teams. Our experience shows that failure to identify and communicate these cooperation needs lead to poor resource allocation and time consuming workarounds.

Priority 3: *Team dependencies on part-timers.* The inclusion of developers, employed with part-time work contracts, in the project team added up to tedious and complex coordination. While part-timers are frequently not attending the planning meetings and they possess less general system knowledge, they often serve as the key resources for specific modules and possess key knowledge for the success of a sprint. The team needed a means to better communicate information to the part-timers, as well as to be able to access their knowledge at all times.

IV. The Application of the Tool

This section describes how the Redmine tool was used in the company's context in support of the activities mentioned earlier. We refer interested reader to [10] for more information on its functionality and look-and-feel features. We also make the note that the tool was selected by the senior management team of the company without any involvement of the developers and the scrum master. In our case study we did not have access to the senior team nor to documentation explaining the selection process and the selection criteria. Therefore, our research did not include investigation on this topic. We use the tool to support the four main activities:

Activity 1: Add PB item. Any stakeholder in the project can add requirements to the PB as the tool provides sufficient space to specify information on the new PB item. Compared to the initial situation, we observed two advantages: (i) those

team members adding and modifying an issue are clearly identified helping to determine dependencies; (ii) the amount of explicitly stated information on an issue has improved. Earlier, the issue description was rarely longer then one sentence, while, in Readmine it is often a few paragraphs long and occasionally diagrams or documents are added. However, an adopting team should also consider side effects. We learnt, for example, that when not all stakeholders are trained in entering the information needed, it is harder to ensure the relevance of what they enter. We observed that the PB grows including several duplicate issues and issues regarding already implemented features. We have also experienced issues of no relevance to the CMS being added to the backlog. The maintenance effort for the backlog has, thus, increased.

Activity 2: Prioritize PB. The Product Owner is responsible for prioritizing the PB. He does so with input from the stakeholders before, during and after the "estimate PB" activity. The PB is initially prioritized by an auction scheme where the product owner sells time and priority to the stakeholders. Changes to priorities may happen at any time due to events such as customers change of mind or new issues entering the PB. The tool provides a field for assigning priority to an issue. We assign priority in the range 1 to 9 using priority 9 for new, non-prioritized issues. Priority 1 issues will finally go into the next SB.

Compared to the initial situation, we experienced that Readmine changed in many ways how we dealt with priorities: it made it easier to search, filter and sort issues so that related issues can be found and compared. The increased information concerning new requirements can also help in setting initial priority. On the other hand, we also observed that the amount of backlog items has increased significantly which adds complexity to the prioritization process.

Activity 3. Estimate PB. Estimation of the PB is carried out by the product owner, the scrum master and the scrum team during the "sprint planning 1" meeting. Stakeholders, such as the costumer or members of other teams are also invited to attend. A starting point for the estimations is a prioritized PB. Typically, as the PB is long and the process rather time consuming, only high priority issues are chosen for estimation. All meeting participants exchange knowledge and reason about the effort needed for each issue on the agenda. The purpose is to collect everyone's knowledge about the issue. After the discussion, one or more rounds of planning poker [6] are played where the participants express their estimates. The final estimate, in terms of shells (measure of complexity compared to a commonly understood task) is recorded in Redmine. In this estimation process, the tool has multiple functions: besides recording the shells, the issues get specified in greater detail by adding notes and attachments coming out of the discussion. This helps mitigate our highest priority impediment. Also, as the meeting involves many stakeholders, dependencies are more easily identified and these are recorded. This helps us with the other impediments.

Activity 4. Create SB. This activity is about splitting a subset of the PB into manageable and explicit tasks and is done during the "sprint planning 2" meeting led by the scrum master and attended by the scrum team. The product owner is available to answer questions during this meeting. The SB contains tasks, bugs, support and feature issues, all handled as issues. Creating a task issue for the SB is done the same way as other issues. If team members are not able to attend the meeting, they are expected to review the SB and add their knowledge to it later. Below we will describe three procedures used during the "create SB" activity, namely task specification, risk assessment and dependency handling.

Sub-activity 4.1. Task specification. The tool allows to link tasks to the respective features each task implements as well as to team's discussions on task execution. This turns the planning process into a knowledge-sharing session whereby knowledge is captured in writing. This mitigates the priority 1 impediment described in Sect. 3. An unexpected drawback is that the secretary of the group gets confused during the discussion and spends too much time writing during the meeting. This experience shows that a practice should be introduced to capture only some keywords and important notes during the meeting and have someone specify the task in more detail just after the meeting ends.

Sub-activity 4.2. Risk assessment. As the tool made possible for the team to write a risk assessment in the details field, they estimated risk of events that can prevent them from sprint success, e.g. the lack of resources in a cooperating team or failure to make a server available. The risk assessment procedure is as follows: (i) the risk is specified including who or what is affected and what is particular about it; (ii) the likelihood that this risk poses a threat to the sprint is estimated on a scale of high, medium and low; (iii) severity, if this problem occurs, is estimated on a scale of high, medium and low. This method is a simplification of an industrial hazard analysis [5] and risk control method. Risk estimation helped improve the cooperation and decreased the stress factor in the development process, as the risk information was used to: (i) communicate team's concerns to stakeholders, (ii) ask for change in requirements or re-prioritization if the risk is too high, (iii) perform expectation management with the stakeholders, (iv) work as an insurance policy for the team, as risks that could not be mitigated are explicitly communicated ahead of time and the team feels "off the hook".

Sub-activity 4.3. Handling inter-team dependencies. As part of sprint planning, we identify tasks for which the team either needs input from other teams or for which someone else needs our output to successfully complete their tasks. The team decided to set task deadlines in the middle of the sprint in such cases. By explicitly stating deadlines it is easier to work against them and to perform expectation management of external parties. This mitigates the priority 2 impediment described in Sect. 3.To illustrate it, we provide one of our experiences as an example: The team created a menu structure to be used by a site being developed abroad. It was identified that, to meet the deadline, there was a need for implementation on our side and a proof of concept on the costumer side. To achieve this, it was necessary to set an agreed upon deadline for when to hand over implementation details from our team to customer. When they received the documentation they had

already set aside resources and implemented the proof of concept in time for the sprint review.

V. LESSONS LEARNT

The findings of the case study were used twofold: (1) to catalogue existing lessons learnt and (2) to compare them and identify areas where the lessons overlap or diverge with earlier published experiences. In our comparison, we also checked for each lesson the context of its intended use. We included the project manager of the company in this analysis. The analysis revealed the following characterizing features of our learning:

In our experience, the tool helps the sprint planning process in at least five ways: (i) it provides a single, easily accessible and transparent interface for the PB and SB, (ii) it greatly increases the amount of explicit information, (iii) inter-team dependencies are more easily identified, (iv) risk analysis and communication is made possible and (v) knowledge and experience is better shared between the team members. We will note that Redmine only provided the opportunity for these improvements and that the key to successful impediment mitigation is the processes and not the tool.

All impediments (see Sect 3) have been resolved. The company is happy with the new task specification process and believes it ensures more efficient development and a better product. The inter-team cooperation has improved due to early detection of dependencies and earlier request for resources. When it comes to the dependency on part-timers we find it hard to draw any conclusions. On one side the knowledge sharing has increased due to the processes described here but some turnover, interpersonal issues and other initiated processes prevents us from saying that the impediment is removed because of the tool.

As already indicated earlier, introducing and using a new tool did not come for free. Below we point out to what we found problematic: (i) we did have a slowed down introduction in the organization as all users were new to the program, (ii) several developers reported to have lost overview of the SB and its progress because of the removal of the scrum board, (iii) the PB and its management increased due to some duplicated or non-relevant issues as well as increased number of added relevant issues and (iv) meetings got longer due to heavy workload on the secretary.

Using a tool to organize backlogs can be a good solution when the context is appropriate. In our experience, an appropriate context is one characterized by: (i) a great deal of dependencies between product lines and teams and (ii) the team members possessing very different knowledge about the system and the interpersonal knowledge sharing is difficult.

Last, when comparing our lessons with previously published reports [9,11,13] on topics similar to ours, we found that our conclusions converge with observations made by other agile practitioners. For example, Silva et al. [13] found that the interaction between the programmers is affected throughout the project when not everyone is able to keep regular face-to-face meetings. This agrees with our experiences regarding the part-timers' participation. Read [9] observes that the agile techniques could be customized to collaborate into an effective solution. Ruhnow [11] reports that "At first the team struggled putting into practices the ideas from books and papers on agile development. We eventually made headway by focusing on making changes in a very purposeful and incremental fashion, which I call "Conscious Evolution". The matter that our experiences overlapped with previously published ones indicated that the lessons we present go beyond the context of our case study company and that they can be of relevance to other practitioners.

VI. CONCLUSION

This paper reported on lessons learnt from one company's experiences in adopting an agile requirements management tool. We applied case-study-research techniques and observed ten iterations of a project, where Scrum was used. The first four sprints were performed using spreadsheets to organize SBs, during the remaining sprints Redmine was used. After the first introduction pains, it became clear to the company that the tool will continue to support the sprint planning process. We make the note that there are many products e.g. [14] available for the purpose of agile project management. Here we don't put the focus on the concrete tool, but moreover on the difference in the project management practice with or without tool support.

Our future work is to use the experiences we collected in a more rigorous analytical process aimed at formulating research hypotheses based on the observations. Our long term plan includes further empirical studies to find evidence supporting or refuting these hypotheses.

REFERENCES

[1] Agile manifesto http://agilemanifesto.org/

[2] Assmann, D., T. Punter, Towards Partnership in Software Subcontracting Source, Computers in Industry 54(2), 2004, pp.137-150.

[3] Baker, T.J.C., Establishing an Agile Portfolio to Align IT Investments with Business Needs, AGILE'08, pp. 252-258.

[4] Charmaz, K., Constructing Grounded Theory, Sage, 2008.

[5] Clifton A. Ericson, Hazard Analysis Techniques for System Safety, Wiley, 2005.

[6] Cohn, M., "Agile Estimating and Planning", (November 2005).Mountain Goat Software.

[7] Lindvall, M., Basili, V., Boehm, B., Costa, P., Shull, F., Tesoriero, R., Williams, L., Zelkowitz, M., Empirical Findings in Agile Methods, XP/Agile Universe Conference'02, Springer, pp. 197– 207.

[8] Paasivaara, M., Durasiewicz, S., Lassenius M., Using scrum in a globally distributed project: a case study, J of Software Process: Improvement and Practice 13(6), Nov/Dec 2008, pp. 527-544.

[9] Read, D., Going Agile – A Case Study Software Process Consultant Strategic Systems, 19. Australian Software Engineering Conference, 2008, Perth.

[10] Redmine [www.redmine.org]

[11] Ruhnow, A. Consciously Evolving an Agile Team, AGILE 2007, pp. 130-135.

[12] Schwaber, K and Beedle, M, "Agile Software Development with Scrum", ISBN 0-13-067634-9, Prentice Hall, 2001

[13] Silva L, Santana C., Rocha F., Paschoalino M., Falconieri G, Ribeiro L., Medeiros R.,Soares S., Gusmão C., Applying XP to an Agile-Inexperienced Software Development Team, Springer, 2008, pp. 114-126

[14] Tinypm http://www.tinypm.com/

[15] Yin, R.K.: Case study research, Sage Publications, 2003.