Neglecting Agile Principles and Practices: A Case Study

Patrícia Vilain
Departament de Informatics and Statistics (INE)
Federal University of Santa Catarina
Florianópolis, Brazil
vilain@inf.ufsc.br

Alexandre Jonatan B. Martins
NATTEX Sistemas
Florianópolis, Brazil
alexandre@nattex.com.br

Abstract—Agile processes are expected to follow a set of well known agile principles and practices. This paper analyses the utilization of an agile process in a project that imposes certain difficulties in meeting such principles and practices in their totality, particularly those that refer to daily face-to-face communication among team members and frequent delivery of new versions of the product in short periods of time within a predictable schedule. A process based on Scrum was adapted to be utilized in a project which had its members distributed in different locations and that required the utilization of emergent technologies not familiar to the developers.

Keywords—Agile principles; agile methods; agile practices; Scrum

I. INTRODUCTION

Since the appearance of agile methods [1], several software development companies that refused to adopt traditional software processes (e.g. the Unified Process [2]), started to adopt agile methods. Such methods value “individuals and interactions over processes and tools” and “working software over comprehensive documentation”. What can be observed in practice is that the development of low complexity applications, when performed by small teams, does not require a bureaucratic software process, which explains why agile methods seem like a good fit in those situations.

Agile methods are defined according to agile principles [1]. The processes described by such methods include the so called agile practices, that is, practices that help developers follow those principles accordingly. Some of these principles are focused on having people working together and meeting in a daily basis. In the agile method Scrum [3], for example, these principles are followed by the practice of daily meetings. Other agile principles state that working software must be delivered frequently, preferably in short periods of time within a predictable schedule. That can be accomplished in Scrum by simple practices such as having Sprints restricted to a specific duration. However, constraints eventually imposed to the development process may prevent the team from adopting even simple practices such as daily meetings or time-boxed Sprints.

This paper analyses the utilization of an agile process in a project (case study) that imposes certain difficulties in meeting agile principles and practices in their totality, particularly the ones mentioned above. The case study in question refers to the development of a web application for storing and managing electronic documents. One of the difficulties faced in this project was that the development team, though small, did not work every day at the same place, which prevented the adoption of daily meetings. Another difficulty was that the application being developed, though of relatively low complexity, involved the use of emerging technologies that the development team was not familiar with. This generated certain uncertainty around iteration planning in the sense that iterations might not be set to a fixed duration or offer the guarantee to produce working software at the end, thus neglecting the agile principles and practices already mentioned. This process utilized in the referred project was based in Scrum, adapted with practices that try to compensate for the difficulties in following agile principles and practices in their totality.

The paper is organized as follows. In Section 2, agile methods are revisited, along with the agile practices proposed in Scrum. Section 3 briefly describes the case study. Section 4 presents the constraints imposed to following all agile principles and practices for the case study, along with adaptations made to compensate for those constraints. Section 5 presents the process utilized in the case study in more detail. Section 6 contains a discussion about the adaptations necessary to the agile process. Section 7 concludes the paper.

II. AGILE METHODS AND SCRUM

Agile methods are software development methods which apply the iterative and evolutionary development, employ adaptive planning, promote incremental delivery, and include other values and practices that encourage agility. They are best suited for projects characterized by changing, speed and turbulence.

In beginning of 2001, a group of agile methods followers created the agile manifesto containing agile principles that all agile methods should follow [1]. These principles are reproduced below:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity--the art of maximizing the amount of work not done--is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

A development method is considered agile if it provides agile practices that follow the principles listed above. Among the most common agile methods are: Scrum, XP, DSDM, Lean, Crystal, FDD and ASD [4]. The agile method Scrum is detailed next for being the basis of the agile process utilized in this work.

A. Scrum

The method Scrum was created by Ken Schwaber and Jeff Sutherland in the 90’s and its purpose is to manage the system development process [3]. As it is generally devoted to process management, practices related to more specific software development activities such as requirements gathering, software design and programming are not detailed in Scrum. As observed in [5], those practices are usually borrowed from XP, and are out of the scope of this paper.

Scrum defines the following agile practices:

- **Scrum Master.** The person who is responsible for ensuring that Scrum values, practices, and rules are enacted and enforced.

- **Product backlog.** A repository containing all system requirements (features, functions, technologies, enhancements, and bug fixes). These requirements are organized in a priority list.

- **Scrum Team.** A group of people that will develop a set of items from the Product Backlog during an iteration (Sprint). It is recommended to include seven people, plus or minus two, but teams with 3 people are also allowed.

- **Daily Scrum Meetings.** 15-minute meetings that promote the face-to-face communication among Scrum Team members. During this meeting, members of the team are supposed to explain what they have accomplished since their last meeting, what they intend to do before the next meeting, and what difficulties they are facing.

- **Sprint.** This refers to one development iteration. The development must be divided into fixed sprints that take, each one, 30 days. Once project members have developed more experience with Scrum, adjustments can be made to the duration of the sprints.

- **Sprint Planning Meeting.** This is the meeting to plan the next sprint. At this meeting, the Scrum Team decides what functionalities and technologies will be developed during the sprint.

- **Sprint Review Meeting.** 4-hour meeting that happens in the last day of the sprint. This is the meeting for the team to present and review the product increment developed during the sprint. It must be coordinated by the Scrum Master.

- **Sprint Backlog.** A repository containing those requirements selected from the Product Backlog to be developed during the next sprint.

Fig. 1 briefly shows the Scrum process. Initially, all functionalities, features, and technologies are included as items in the Product Backlog. During the project, instead of being static, the Product Backlog evolves along with the product. In each sprint, the Scrum Team selects as many items from the Product Backlog as they think they can develop during the iteration, and these items are included in the Sprint Backlog as a list of tasks. Throughout the entire sprint, a short meeting conducted by the Scrum Master, called Daily Scrum Meeting, happens to review the progress and identify impediments. At the end of each sprint, an increment of the product including new functionalities is delivered; members of Scrum Team all gather for a meeting, the Sprint Review Meeting, where that product increment is inspected. Finally, the Product Backlog is reorganized and prioritized, and the items with higher priority are selected for the next sprint.

![Figure 1. Scrum Process](http://www.scrumalliance.org/learn_about_scrum)
Scrum members can play the following roles:

- **Scrum Master**: the person responsible for the success of the project. As already mentioned, it is the Scrum Master who has to ensure that the values, practices and rules of Scrum are performed and enforced accordingly;
- **Product Owner**: the one person who is responsible for managing and controlling the product backlog;
- **Team Members**: those responsible for achieving the target goal set for a sprint.

### III. Case Study

The case study utilized in this paper refers to the development of a web application for storing and managing electronic documents. It consists essentially of a web-based user interface to a core document service already implemented. That core service was made available to team members as an API for the Java language. A non-functional requirement was that the implementation of such a web application should rely on the latest version of the JSF framework (i.e., Java Server Faces version 2.0) that had been recently released.

The team was composed by four people. All of project members had significant experience with Java. The Scrum Master had good experience in project management and could spend four hours a week in this project. The Product Owner also played the role of a team member (i.e., developer). This person had developed the code of the document service back end to be utilized in this project, thus having good knowledge about the application domain, which is why was assigned the role of Product Owner. This person could spend ten hours a week in this project: four hours as Product Owner, and six hours as a development team member. The remaining two people were regular team members and they were considered beginners in regard to the technology utilized in the implementation, particularly the JSF-2 framework. They could dedicate twenty hours a week to this project.

The entire project should not exceed four months due to budget restrictions, which is why adopting some sort of agile process was considered imperative by all project members.

### IV. Constraints to the Agile Process

The development process utilized in the case study had to be defined with the following constraints in mind:

- **Distributed Team**: Team members would not be working at the same place every day and face-to-face meetings could be held only once a week.
- **New Development Technologies**: Team members were not very familiar with some of the development technology utilized in the project.

It is important to note that, although all the participants did not work at the same place, they were all in the same city. In fact, they could meet once a week. They could also schedule sporadic face-to-face meetings in case some crucial issue had to be addressed. Therefore, this project did not have most of the difficulties commonly found in distributed agile development such as culture, time zone, communication, customer collaboration, trusting, training and technical issues [6].

The process utilized in the case study was based on Scrum. However, some of its practices had to be modified in order to cope with the constraints mentioned above. Therefore, the following decisions regarding the adaptations necessary to the process were made:

- **Utilize a web-based tool to support the process.** As project members would be working at different places in the same city, they decided to utilize a web-based tool to support the process, not only to allow all members to monitor the execution of sprints, but also to serve as a means of communication between them. In addition to providing support for managing the Product Backlog and monitoring the development of sprints, the tool in question should have some sort of wiki or blog associated with Product Backlog items (stories and their tasks) with the capability of notifying (preferably through e-mail) those involved about new entries in the wiki.
- **Eliminate daily meetings.** Despite daily meetings being one of the most important practices of Scrum [3], it was necessary to remove them from the software process because project members were unable to meet daily. The possibility of having daily meetings through video web-conferencing was considered, but there was the additional difficulty of team members having disparate working hours on certain days of the week. So, they decided to adopt only weekly meetings for monitoring the sprints. It would be up to the Scrum Master and other project members to monitor and update the entries in the wiki daily in order to identify and resolve any issues that might come up during the development of sprints.
- **Permit sprints with extendable length.** Most of the programming to be carried out in sprints was expected to utilize a new technology not yet dominated by developers. Initially, the sprints had their duration set to two weeks. However, some of the first sprints showed a delay between 10% and 25% compared to the estimated time. Project members realized that the main reason for the error in estimating sprints was the difficulty in predicting the time developers needed for learning the new technology. In a situation like that, it is recommended that the Scrum Master speak with people who understand the technology [3]. However, no one with that knowledge was available to contribute to the project. It was then decided that the Scrum Master would have the power to increase the duration of the sprints by 25% in case the delay had been caused by the lack of experience with the technology involved. Therefore, instead of being sent back to the product backlog in order to be incorporated into a new sprint (as recommended in [3]), the tasks that could not be completed by the end of a given sprint would remain in
that same sprint, which would have its length increased by up to 25% (the exact percentage should be set by the Scrum Master during the weekly meeting). The idea behind that decision was to prevent a newly created sprint from causing team members to lose their focus on the issues related to learning the new technology, and also to save the team from unnecessary frustration resulting from a significant number of sprints not being completed.

These adaptations introduced into the process were shown to be adequate to the project in question and to the profile of the project team. Table I summarizes the agile principles and practices that had to be neglected due to such adaptations and why they were considered necessary.

<table>
<thead>
<tr>
<th>TABLE I. AGILE PRINCIPLES AND PRACTICES NEGLECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Principles/Practices</td>
</tr>
<tr>
<td>----------------------</td>
</tr>
<tr>
<td>Agile Principle:</td>
</tr>
<tr>
<td>Scrum Practice:</td>
</tr>
<tr>
<td>Daily Scrum Meeting</td>
</tr>
<tr>
<td>Agile Principle:</td>
</tr>
<tr>
<td>Business people and developers must work together daily throughout the project</td>
</tr>
<tr>
<td>Scrum Practice:</td>
</tr>
<tr>
<td>Time Boxed Sprints</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

V. DEFINING A NEW AGILE PROCESS

The agile process that resulted from the modifications mentioned above is generically described by the steps below. Note that the roles played by project team members are the same ones defined in Scrum: Scrum Master, Product Owner and Team Member.

A. Definition of the Product Backlog

This step aims to define an overview of the application, and it consists of a list with an initial description of the stories and a list of non-functional requirements. The stories are included in the Product Backlog, just like in Scrum, with the exception that, in this case, it assumes the utilization of a workgroup tool (preferably web-based) that provides support to creating and managing the backlog and monitoring the development process. As new stories are identified, they are included into the Product Backlog.

B. Sprint Planning

The duration of each sprint should be 1 to 3 weeks. The sprints can have their time length increased by 25% in case of error in the estimate due to the lack of experience with new technologies. Sprints of 4 weeks are not permitted because a 25% increase would result in a sprint of five weeks, inconsistent with the recommendations of Scrum.

The stories of the Product Backlog that will be part of the sprint should be selected according to the time estimated for its development, following the method Planning Poker [7] with the participation of all project members. However, considering that team members may not be familiar with the technology to be utilized in application development, or difficulties imposed by the distance between team members, it is acceptable to have only the Scrum Master and Product Owner participate in Planning Poker time estimations.

After stories have been allocated to the Spring Backlog, it is necessary to verify that the description of each selected story is detailed enough to be understood by team members. If necessary, the description of the story should be more detailed. These details could include the sketch of a screen layout, an operating example, or even part of the programming code. This is important to avoid fundamental questions about the story descriptions that would normally be resolved during the daily meetings.

Once reviewed, each story is divided into tasks. The tasks, in turn, may also be detailed, and each task is assigned an estimate of effort and time for execution. It is not required that a story is sub-divided into tasks, but it may be necessary to define a task within the story if the tool utilized to support the process requires it.

Finally, tasks are distributed to developers. This distribution is so that developers can benefit from the knowledge acquired in previous tasks regarding the technology (or component) necessary to perform those tasks.

C. Daily Notification Monitoring

As this process does not include the daily meetings originally proposed in Scrum, it is necessary to monitor notifications triggered by the Scrum Master or other team members. This assumes that the Scrum Master and team members are not always at the same place, and the execution of this step requires the utilization of a workgroup tool that allows team members to submit their questions and complaints to the Scrum Master.
D. **Sprint Design and Coding**

This step is devoted to applying any design techniques considered appropriate for a given project, such as sequence diagrams and design patterns, and writing the corresponding implementation code to be compiled and run in the target platform.

E. **Weekly Meeting**

Participants must schedule a weekly meeting to be held by all project members. Whenever possible, members should meet in person, otherwise through video conference. If the meeting is held at the end of the sprint (Sprint Meeting), it may last up to three hours. In such a Sprint meeting, the Scrum Master can evaluate how the stories that were not completed will be handled and whether or not the sprint will have its duration augmented. If it is just a regular meeting to monitor the sprint, the Sprint Backlog must be partially reviewed and discussed, and it can last no longer than 2 hours.

F. **Code Delivery**

After each sprint, the code must be made available to both the Product Owner and the Scrum Master so they can verify the functionality implemented in that sprint.

VI. **DISCUSSION**

The development of an application that utilizes emerging technologies not well known by the developers brought up some important aspects related to utilizing agile methods for that task, which are discussed next.

When a new technology is being utilized, the time spent to learn this technology has to be computed in the time allocated for the development of the stories selected in a sprint. However, it is often quite difficult to determine the time required to learn that technology, which normally leads to errors in estimation. One solution to avoid that problem is to allocate a period of training for developers prior to development. However, this can cause an unnecessary delay to the start of the project as developers can be more efficient to learn about the new technology while they are trying to address specific issues related to the tasks of each sprint. For this reason, the process described here assumed that the time required to learning that technology should be estimated as part of each sprint.

Another solution to the problem of incorrectly estimating the time allocated to each sprint would be to assign the stories and respective unfinished tasks back to the Product Backlog, as suggested originally in Scrum [3]. This solution was not adopted here to prevent team members from being held responsible for unfinished sprints. Project team members found better to allow the length of the sprint to be increased in up to 25%, provided that the delay was related to technologies that developers were not familiar with.

It is important to point out that as developers become familiar with new technology, it is no longer considered new and therefore the time that was once required for learning that technology can no longer be included in the sprint time estimation.

In the case study described earlier, the time originally allocated for development was 3 months (13 weeks), divided in 8 sprints. As a considerable number of sprints have had their duration time increased, the development ended up taking 16 weeks to complete. Fig. 2 presents a graphic showing the estimated time and the actual time spent in each sprint. Note that this project involved the development of a graphical user interface (UI). Some agile processes suggest that the design of all UI must be done prior to regular iterations [8] and some suggest that UI design is performed in parallel to product development [9]. Since the project in question could not count on a UI specialist in usability, as suggested by [10], and considering that the UI was relatively simple, it was up to the Product Owner to sketch some of the UI screens, which were utilized to complement the description of the stories.

Another point of discussion is the amount of description included in each story and task. As suggested in [11], the stories can be detailed to make them more understandable for the developers. Usually, this description should be concise. However, when developers have no prior knowledge of a technology being utilized or lack a complete understanding of application requirements, detailing the stories and tasks increases their productivity in development. That also helps in situations where project members are physically apart.

As suggested in [12], it is important to modify or replace a practice, rather than simply delete it. Regarding daily meetings, an agile practice defined in Scrum, it seems possible to replace them by electronic communication without large apparent losses, provided that an appropriate tool is utilized. When there is no possibility of having daily meetings, the communication through a tool seems to be the best option to keep track of problems that are occurring daily as opposed to address these problems only after the weekly meeting.
A common practice of Scrum is to use a task board. Since the development team was not working at the same place, the use of a task board to record tasks not started, in progress and completed seemed useless. However, that was compensated by the web tool selected, which allowed to have tasks assigned to each team member and keep track of the status of each task (not started, in progress and completed).

VII. CONCLUSIONS

This paper presented the development of an application using an agile process. This process was based on the agile method Scrum, but it was adapted according to some development and project team constraints. Although neglecting some of the most known agile principles and practices, the resulting process was successfully applied to a software project without affecting the agility of its development.

The agile process defined in this work neglects two agile principles defined in the Agile Manifesto [1], particularly those that say that development teams must work together daily and that information should circulate among team members in face to face conversation. However, it was possible to observe that agile processes can also be applied to projects that do not allow such agile principles and practices to be completely followed, provided these are compensated by appropriate practices or tools. In the case study presented here the daily meeting practice had to be replaced by a weekly meeting. The lack of a daily face to face communication was compensated by the adoption of a web based tool that notifies project members about questions or issues faced throughout the sprint.

One important adaptation incorporated to the development process described here was to allow extendable sprints. Instead of having time-boxed sprints, the time allocated for each sprint could increase by 25% provided that the delay was caused by errors in estimating the time required for the developers to become familiar with the new technologies utilized in development.

REFERENCES