Exploring the Individual Project Progress of Scrum Software Developers

Ezequiel Scott and Dietmar Pfahl

University of Tartu, Tartu, Estonia
{ezequiel.scott, dietmar.pfahl}@ut.ee

Abstract. Scrum based software development has become increasingly popular in recent years. Scrum requires teams following agile practices and their principles. One of them includes having room for the reflection of the team on how to become more effective. In this context, measuring and enhancing the performance of teams is still an area of interest for the Scrum community. Traditional Scrum metrics have often been used to measure the performance and productivity; however, individual contributions of team members to the project are often shaded by the team overall performance. In this paper, we propose a metric for measuring individual differences in project progress based on the traditional Burndown chart. We also show preliminary results of applying it in a particular training context, highlighting how learning-styles based instruction can improve the individual project progress of students.

Keywords: Agile Software Development, Scrum, Agile Metrics

1 Introduction

Scrum based software development has become increasingly popular in recent years [3]. In fact, the latest State of Agile Survey reports that Scrum and its variants are used by more than the 75% of respondents [14]. This acceptance stems from the fact that many Scrum teams have reported relevant results regarding the quality of the software, the working synergy, the user satisfaction, and the enjoyable working environment [12].

To put Scrum into practice, teams not only have to follow the Scrum practices but also adhere to the agile manifesto and their principles. One of them includes having room for the reflection of the team on how to become more effective, in order to tailor their behavior accordingly[1]. In this context, measuring and enhancing the performance of teams is still an area of interest for Scrum practitioners. As a result, many metrics for measuring performance in Scrum have been proposed such as the *Velocity* of the team and their *Burndown chart*, among many others [4].

These metrics have been successfully used for monitoring how certain data points affect the progress of teams. However, individual contributions are often shaded by the team overall performance since Scrum stresses teamwork and collective responsibility for the final outcome of a project [6]. Knowing the individual project progress could help teams in different ways. For example, teams can

get a better understanding of the progress of the project and use this knowledge as a first indicator of possible problems. In addition, training contexts can take advantage of individual metrics since they can serve as an indicator of students' performance. Thus, studies on metrics for assessing individual project progress are beneficial for the Scrum community.

In this study, we aim to make a two-fold contribution. First, we propose a metric for measuring individual project progress based on Scrum. Second, we show the preliminary results of applying the metric in a particular training context. Our results highlight how learning-styles based instruction can improve the individual project progress of students.

2 Related Work

There are many metrics often used to deal with project monitoring and control activities in Scrum. All of them usually rely on the use of agile artifacts such as *Backlogs* and *User Stories*. Although these metrics are popular, the agile community agrees with the idea of improving the measurement in Scrum. Dawney and Sutherland [4] have defined a set of Scrums metrics which aim at improving the traditional ones. The goal of these metrics, like many of the traditional ones, is to measure the team productivity.

The team productivity and the factors which correlate with it have been studied by many authors. For example, performance has been studied in their relationship with the stakeholder-driven process [9], and the stress and empowerment of the teams [8]. In addition, researchers and practitioners seem to agree about the importance of both technical and non-technical skills of developers regarding productivity [13]. So far, however, there has been little discussion about individualized metrics in Scrum.

In particular, studies focused on individuals have been connected with the educational field. For example, Gamble and Hale [6] have defined four individualized metrics: Contribution, Influence, Impact, and Impression. These metrics are based on the interaction of the students with a collaborative tool. In contrast, this study is more focused on the indivudals' amount of work done during the software development than in the level of social engagement. Other studies have proposed metrics that rely on self-reporting activity, project evaluation rubrics, and grades based on individual submissions related to the project [7,2]. Surprisingly, the analysis of metrics derived from traditional agile artifacts like the Burndown chart have not been closely examined.

Exploring the individual characteristics of team members allows for studying their achievements in Scrum from a training point of view. In this line, Scott et al. [11] have explored how to improve the students' understanding of Scrum topics when learning preferences are used. Measuring the individual project progress in these contexts could be useful for determining the effectiveness of training approaches.

3 Method

The method used to explore the individual project progress mainly comprises three steps. The first step involves an initial training in Scrum. In the second step, the trainees put the Scrum framework into practice by developing a small software product. Finally, their individual project progress is analyzed. We describe these steps in the following sections.

3.1 Initial training in Scrum

The initial Scrum training is based on previous experiences using the Felder-Silverman learning style model [5] in capstone projects. This model has been widely used in Computer Science and proposes classifying students into different learning styles which have their corresponding teaching style. Previous studies have shown that students can improve their understanding of Scrum concepts when they are exposed to instructional methods tailored according to their learning style [11].

To analyze the individual project progress when students receive the instructional method according to their learning styles, we rely on the recommendations of Pashler et al. [10]. The authors propose several guidelines to design a controlled experiment and obtain thoroughly evidence from it when learning styles are involved. According to these guidelines, we organize the initial training as follows.

First, students are assigned to two different groups according to their learning style. The students' learning styles are collected through an online questionnaire that allows for classifying the students into *active* or *reflective* students. Briefly speaking, *active* students prefer doing tasks or talking about concepts whereas *reflective* students are likely to manipulate and examine the information introspectively [5].

Once students are grouped by learning style, we randomly assign the students to two different instructional-method groups related to the *active* and *passive* teaching styles. These groups determine the instructional methods received by the students and the random assignment guarantees that both instructional-method groups have the same numbers of *active* and *reflective* students.

Finally, students receive the training in Scrum on the basis of their assigned instructional method. One group of students receive the instructional method according to one learning style (i.e. the *active* method) whereas the second group is trained using the remaining instructional method (i.e. the *passive* method). Thus, both groups receive the same topics, yet in different ways. The topics and the instructional methods used are described in more detail in previous research [11].

After receiving the training in Scrum, the students are expected to put the Scrum concepts into practice through the development of a small software project. We organize the software development as follows.

3.2 Software development

To put the Scrum concepts into practice, the students are allocated to different teams. Each team is asked for developing the same software product: a small software application that allows teachers to manage courses and their topics. They are also asked for following the Scrum practices they learned before and track the project monitoring and control information into an online spreadsheet. Figure 1 shows an example of a user story in the spreadsheet.

#User Story	User Story Description	Story Points (SP)	#Task	Task Description	Resp.	Task SP
1	As a teacher, I want to manage the topics of my course to easily define the syllabus online.	5	1	Allow for adding new topics to the course	X	1
			2	Allow for removing topics from the course	Y	1
			3	Develop the printing module	Z	1
			4	Support linking topics	X	1
			5	Allow for listing all the topics of the syllabus	Y	1

Fig. 1: Example of the spreadsheet used to monitor and control the projects.

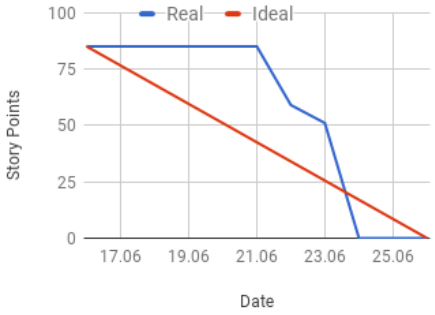

Fig. 2: Example of Burndown chart.

In this context, the students have to create User Stories from the requirements given by the Product Owner (the teacher) to build the Product Backlog. Then, they have to define the Sprint Backlog and estimate the User Stories to be done during the Sprint by using Planning Poker. The students also have to split User Stories into Tasks and allocate them into a Release Plan. Following these practices, the User Stories belong to the team whereas Tasks are self-assigned to team members.

During the development, the students record all the data about User Stories, Story Points, Tasks, and their statuses in the spreadsheet. At the end of the Sprint, the students are expected to generate a product increment and integrate it into the working product. Then, they present the product and receive feedback from the Product Owner during the Sprint Review. Finally, the students are encouraged to carry out the Scrum ceremonies within the team such as the Sprint Retrospective.

At the end of the Sprint, it is also possible to calculate several traditional Scrum metrics such as Velocity, Technical debt, and the Burndown chart, if the students use the spreadsheet properly. In particular, we focus on studying the individual project progress of the students through the data recorded on the spreadsheet.

3.3 Project progress measurement

To analyze the project progress of a team, we can analyze the number of Story Points they have done during the Sprint. The spreadsheet used by the teams allow us to identify the completed User Stories on each day of the Sprint. This way, we can build a Burndown chart for each team. This chart is a commonly used measurement tool for planning and monitoring the progress in agile methods [4]

The Burndown chart represents the amount of work remaining that needs to be accomplished till the end of the Sprint. The horizontal axis shows the days of a Sprint whereas the vertical axis shows the number of remaining Story Points. The trend line of remaining Story Points, also known as *ideal*, indicates whether the Team will accomplish the tasks committed by the end of the Sprint. Figure 2 shows an example of Burndown chart.

However, this chart is not suitable for studying the individual contribution of the team members to the overall project progress. This is because the chart shows the remaining Story Points of the User Stories, and User Stories belong to all the members of the team. To deal with it, we apply a straightforward strategy that is focused on the Tasks done by the team members and not in the User Stories.

The strategy consists in dividing the number of Story Points of each User Story among their Tasks. For example, if a User Story has been estimated with 5 Story Points and it has 5 linked Tasks, we consider that the amount of work of each linked Task is 1 Story Point. Figure 1 shows this example. This way, it is possible to measure the size of a Task in Story Points based on its User Story. In consequence, we can calculate the number of spent Story Points and build an individual Burndown chart representing the contribution of the team member to the project progress.

Using the individual Burndown chart, we study how far the individual project progress is from the average expected rate determined by the *ideal* line. To address this question, we calculate the area of the region between the *real* and the *ideal* progress lines of the Burndown chart. This area is calculated following the Trapezoidal Rule (Eq. 1) since its result is exact when the integrand is a linear function.

$$A = \int_a^b f(x)dx \approx \Delta x(\frac{y_0}{2} + y_1 + \ldots + \frac{y_n}{2}) \tag{1}$$

In our case, a represents the first day of the Sprint and b the last one, $f(x)$ is the function of interest (i.e. the difference between the *ideal* and the *real* progress lines of the Burndown chart), y_n represents the value of $f(x)$ at day n and we determine the value of Δx according to the intersection of both lines. Thus, the area of the region between the real and the ideal lines A can be considered as a measure of the individual progress of a team member. Values closer to zero indicate better performance whereas further values indicate worst performance.

4 Results

We explored the data from a pilot study which involved 35 students. The students received the initial training in Scrum according to their learning styles by following the aforementioned procedure. Among the total number of students, 18 were assigned to the active instructional method (12 of them were *active* students and 6 *reflective* students). The remaining 17 students were assigned to the passive instructional method (12 of them were *active* and 5 *reflective*).

After receiving the tailored instruction, the students were allocated to 8 teams for developing the same software product. At the end of the first Sprint, we processed all the data from the spreadsheets. We removed the data of those students who were not able to record the spent Story Points. We also removed data outliers of those who spent an excessive number of Story Points (more than 100) since these values are unrealistic. As a result, we analyzed the data about 27 students in total, 13 who received the active instructional method (10 of them were *active* and 3 *reflective*) and 14 who received the passive one (10 of them were *active* and 4 *reflective*).

We computed 27 Burndown charts in total, one chart per student. We also calculated the value of the area A for each one. We analyzed the arithmetic mean of these differences with regard to the different learning and teaching style groups. According to Pashler et al. [10], the evidence about using learning styles is reliable if the experiment reveals what is commonly known as a cross-over interaction between the learning style and the instructional method. In this case, the cross-over interaction occurs when the value of A of students being taught with a suitable instructional method is lower than students being taught with an unsuitable method. Table 1 shows these results in terms of arithmetic mean ($\bar{x}$) and standard deviation (s^2) and Figure 3 depicts the crossover interaction.

We also studied the statistical significance of the difference in the mean areas of both groups: students who were taught with suitable instructional methods (Group A) and students who were taught in a way that did not correspond to their learning style (Group B). Therefore, we conducted a *t-test* for independent samples. Although the mean of Group A ($\bar{x}_A = 27.041$) is lower than the mean of Group B ($\bar{x}_B = 37.216$), the results of the *t-test* show that there is no statistically significant difference between both means ($T = -0.623$, $p = 0.537$). A possible explanation for this results could be the small samples used.

Table 1: Mean and standard deviation of the area between real and ideal.

	Active students ($\bar{x} \pm s^2$)	Reflective students ($\bar{x} \pm s^2$)
Active method	34.89 ± 28.11	54.80 ± 35.43
Passive method	38.24 ± 41.11	57.02 ± 49.58

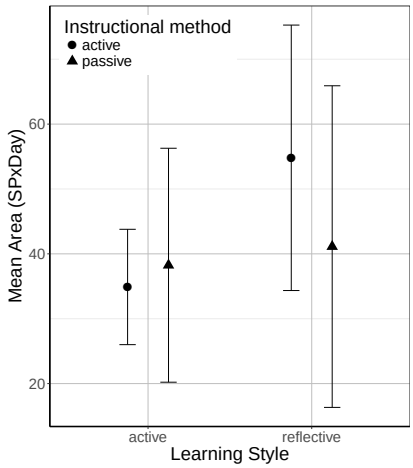

Fig. 3: Distribution of individual project progress according to the instructional methods and the students' learning style.

5 Discussion, Limitations, and Conclusion

In this paper, we explored a metric for measuring the individual project progress of Scrum developers. The metric is mainly based on adapting the traditional Burndown chart to the individual and then studying the area of the region between the real progress and the ideal one. We also showed the utility of applying this metric in a training context, pointing out how learning-styles based instruction can improve the students' individual project progress.

Using the proposed metric, we have found interesting differences in the individual project progress of students who were exposed to different instructional methods. Although these results are not statistically significant yet, they suggest that these differences can be measured in a traditional Scrum environment and how the individual project progress can be interpreted as a learning outcome.

There are threats to validity that should be carefully evaluated in future research. As for construct validity, more indicators have to be considered to determine the individual project progress. We suggest to use this metric as a first indicator of possible problems and analyze the context to find the root causes of the problems. Regarding external validity, there are factors that jeopardize the generalization of the results. In educational contexts, we suggest to use the metric carefully, bearing in mind that it is only one learning outcome among many others. Moreover, different groups of individuals are affected by their history, cultural background, and previous experience differently, and it can conduct to different conclusions.

In terms of directions for future research, further work could explore the previous knowledge on Scrum as well as different metrics that allow for measuring different aspects of the software development process. These aspects are not only performance indicators such as the number of incomplete tasks but also

psychological constructs of the developer. In addition, including communication metrics of the team is a research line worth to explore in order to understand the problems that can arise in a project.

Acknowledgements. This research was supported by the institutional research grant IUT20-55 of the Estonian Research Council.

References

1. Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. The agile manifesto, 2001.
2. WL Cooley. Individual student assessment in team-based capstone design projects. In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages F1G–1. IEEE, 2004.
3. Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development, 2012.
4. Scott Downey and Jeff Sutherland. Scrum metrics for hyperproductive teams: how they fly like fighter aircraft. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4870–4878. IEEE, 2013.
5. Richard M Felder and Linda K Silverman. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681, 1988.
6. Rose F Gamble and Matthew L Hale. Assessing individual performance in agile undergraduate software engineering teams. In *Frontiers in Education Conference, 2013 IEEE*, pages 1678–1684. IEEE, 2013.
7. Jane Huffman Hayes, Timothy C Lethbridge, and Daniel Port. Evaluating individual contribution toward group software engineering projects. In *Proceedings of the 25th International Conference on Software Engineering*, pages 622–627. IEEE Computer Society, 2003.
8. Maarit Laanti. Agile and wellbeing–stress, empowerment, and performance in scrum and kanban teams. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 4761–4770. IEEE, 2013.
9. V Mahnic and Ivan Vrana. Using stakeholder driven process performance measurement for monitoring the performance of a scrum based software development process. *Electrotechnical Review*, 74(5):241–247, 2007.
10. Harold Pashler, Mark McDaniel, Doug Rohrer, and Robert Bjork. Learning styles concepts and evidence. *Psychological Science in the Public Interest*, 9(3):105–119, 2008. SAGE Publications.
11. Ezequiel Scott, Guillermo Rodríguez, Álvaro Soria, and Marcelo Campo. Towards better scrum learning using learning styles. *Journal of Systems and Software*, 111:242–253, 2016.
12. Andrew Stellman and Jennifer Greene. *Learning agile: Understanding scrum, XP, lean, and kanban*. O'Reilly Media, Inc., 2014.
13. Adam Trendowicz and Jürgen Münch. Factors influencing software development productivity – state-of-the-art and industrial experiences. *Advances in computers*, 77:185–241, 2009.
14. VersionOne. 11th annual state of agile survey, 2017. https://explore.versionone.com/state-of-agile.