Feedback in Scrum: Data-Informed Retrospectives

Christoph Matthies
Hasso Plattner Institute
University of Potsdam, Germany
christoph.matthies@hpi.de

***Abstract*—Improving the way that teams work together by reflecting and improving the executed process is at the heart of agile processes. The idea of iterative process improvement takes various forms in different agile development methodologies, e.g. Scrum Retrospectives. However, these methods do not prescribe how improvement steps should be conducted in detail. In this research we investigate how agile software teams can use their development data, such as commits or tickets, created during regular development activities, to drive and track process improvement steps. Our previous research focused on data-informed process improvement in the context of student teams, where controlled circumstances and deep domain knowledge allowed creation and usage of specific process measures. Encouraged by positive results in this area, we investigate the process improvement approaches employed in industry teams. Researching how the vital mechanism of process improvement is implemented and how development data is already being used in practice in modern software development leads to a more complete picture of agile process improvement. It is the first step in enabling a data-informed feedback and improvement process, tailored to a team's context and based on the development data of individual teams.**

***Index Terms*—agile, software development, Scrum, retrospective, software process improvement**

I. Background & Motivation

A major aspect of modern, agile software development processes is the focus on iterative development [1] and adaptation to the given context [2].

A. Feedback Cycles in Agile Methods

Continuously repeating the stages of analysis, design, implementation, and evaluation enables frequent deliveries and timely feedback cycles [2]. Feedback is not only collected regarding the developed software, i.e. whether the requirements were correctly implemented, but also regarding the process itself. By critically examining the development process and how it was executed, problems within a team can be identified and improvements can be proposed for the next iteration, making it more effective and enjoyable [3]. In most software development methodologies, this concept is assigned a dedicated term, such as *Kaizen*, the idea of continuous improvement in Lean or the *Inspect & Adapt* approach of Empirical Process Control in Scrum [4].

B. Feedback in Scrum

Scrum, the agile process framework most often employed in industry [5], explicitly provides meetings for the collection of feedback on iteration outcome and the enacted process. While during the sprint review meeting, the team looks at *what* was built, the retrospective focuses on *how* it was built. Figure 1 gives an overview of the process and its different meetings.

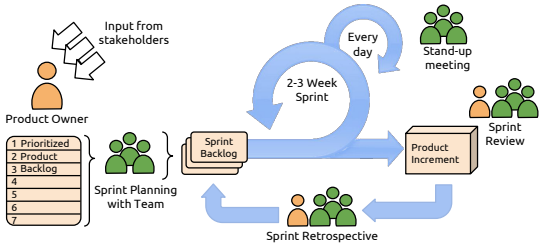

Fig. 1. Generalized overview of the Scrum development process.

During retrospectives, the software development team discusses what aspects of the performed work and collaboration worked well during the last iteration, and should consequently be continued, as well as what issues hindered a smooth workflow and posed problems. The team then collectively decides which improvements could be made in the next development iteration. While effectively reflecting on one's own executed process and finding concrete action items that should be tackled is already challenging, a possibly even larger challenge is tracking the identified issues and their possible resolutions over time.

C. Progress Measurement

In order to allow insights into which improvements were made in the last iteration, some form of progress measurement must exist, i.e. the *check* stage of the Plan-Do-Check-Act cycle [6]. The Scrum Guide itself states that "Scrum is founded on empirical process control theory", with "empiricism asserting that knowledge comes from experience and making decisions based on what is known" [4]. Traditionally, the team decides, based on personal appraisals of the last sprint, whether headway has been made on an action item and whether it can, therefore, be considered resolved or needs to be further worked on in the following iteration [2]. Additionally, an external third party, with knowledge of the chosen development methodology and the teams can be consulted in order to identify process improvement steps or determine their success. However, it is a challenge to instinctively gauge whether an issue has improved. To tackle this issue improvement actions can be based on data produced during the last iteration, relying on a common understanding of the produced development artifacts [7], i.e. "artifact transparency" [4], in order to continuously improve team processes and collaboration.

C. Matthies, "Feedback in Scrum: Data-Informed Retrospectives," 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 2019, pp. 198-201. https://doi.org/10.1109/ICSE-Companion.2019.00081.

II. Research Objectives

The main objective of this research is to gain a better understanding of the role of development data in process improvement approaches of agile software development teams. In particular, we are interested in how teams can be supported in their continuous software process improvement steps by employing measurements based on their own development data.

A. Research Hypothesis

Our research hypothesis, motivated by preliminary interviews with agile coaches and Scrum Masters in industry, is as follows:

> The development data already created by agile software teams during regular development activities holds extensive information on how team members work and collaborate. Teams can use analyses of this data to inform and track their process improvement steps.

B. Previous Work

Our previous work has focused on supporting process improvement efforts of student teams in university capstone software engineering courses, employing analyses of development data to gauge the conformance to agile processes. In university project courses, gaining insights into student teams can be a challenge for educators due to the large number of course attendees. However, giving feedback and providing support for software process improvement steps within teams is vital for learning success. Furthermore, student teams repeatedly face similar issues and challenges, which allowed us to iteratively improve the utilized analyses and measurements [8]. We employed surveys, tutors as well as custom software to ascertain student behavior [9]. The developed software analyzed the large amounts of development data produced by student teams, identifying those areas and artifacts, where the executed process deviated from the prescribed one [10]. The identified artifacts could then be used to discuss the executed process and possible improvements on the basis of concrete data points. This approach of process conformance is especially applicable as the environment that students worked in was set up and controlled by the course educators. The problems and challenges that arise in these software engineering courses are well understood. This is not the case in industry, where teams have various contexts, team members have different levels of experience and there are many organizational constraints.

C. Next Steps

We propose studying the status quo of how professional development teams currently understand and practice process improvement in their individual contexts, focusing on how improvement possibilities are detected and how decided upon actions are tracked over time. By collecting the experiences of developers in a multitude of companies, working on different problems in different domains we can gain insights into how process improvement is executed in industry, what roles and meetings are involved and how teams can be further supported. In particular, the ratio of identified action items that are related and can be tracked using development data to those identified items not related to development data, e.g. human issues, is of interest. Furthermore, we aim to form further hypotheses on how exactly development teams could best be supported with additional tools for process improvement.

III. Research Methodology

An interview study with professional Scrum Masters, agile coaches or other roles, tasked with process conformance, process guidance or process improvement within multiple companies in different contexts will be performed, focusing on the executed process improvement process in the company. Surveys of professional development teams have shown that agile methodologies, particularly Scrum, are widely understood and employed in a multitude of projects [11]. The vast majority of teams practicing Scrum also holds retrospective meetings after each sprint [5]. In fact, in the 2017 *State of Agile* survey, more respondents used agile project tools, such as task boards, than relied on spreadsheets for project management tasks [11]. Many companies following agile development processes rely on employees in Scrum Master or agile coach roles to mentor and advise development teams on agile methodologies and their implementation. These individuals have deep process knowledge, often know multiple development teams, and understand the challenges of the company's context. Interviews with these individuals are most suited to gain insights into how a company or individual teams implement development processes. Interviews will be performed in a half-structured manner in order to allow exploring beliefs and attitudes as well as observing non-verbal indicators [12] and allowing questions to be appropriately adapted to different companies contexts. The main goal of the interviews is to establish how process improvement is handled in the company by the interviewee and to what degree development data is being used or considered useful. Interviews will take a maximum of 30 minutes, in order to not overly disturb the working day and allow easier scheduling. However, interviewees will be asked beforehand to collect, if available, the minutes or protocols of the last three meetings that process improvement was discussed in. After the interviewing researcher has introduced themselves and has stated the goal of the study, i.e. learning about the implementation of agile process improvement in the teams the interviewee is knowledgeable in, the following topics will be discussed:

- What is the general idea of process improvement in the company? Do teams actively implement these ideas?
- What is the agenda for process improvement meetings? What is regularly discussed? Can you share or summarize the minutes of these meetings?
- If action items for improvements are decided upon, how are these tracked over time?
- Are tools or frameworks for process conformance measurement employed? If so, which ones?

Following the advice in the literature on half-structured interviews [13], the audio of interviews will be recorded if the interviewee is comfortable with this, otherwise, notes will be taken by the interviewer. To build trust, the goal of the research will be explained at the start of the interview. Additionally, the interview questions will be made available to interviewees beforehand. All personal data will be anonymized, although we will report on the metadata of interviews, such as the how long the interviewee has been with the company, details of the interviewee selection process and the employed interview guides. After the interview, a summary of the meeting will be made available to the interviewee in order to allow feedback and correct possible communication errors. As in related literature, interviews will be transcribed [14] and coded with topics in an iterative fashion. We will analyze the frequency of topics mentioned in interviews and group them by company, team age, and other factors, dependant on interviewee selection. We plan to present our analysis to the research community in the form of a conference paper, focusing on analyses of collected improvement action items and the applicability of development data analyses for observed teams.

A. Research Timeline

We expect to finish the interview study with industry partners within the next six months. We have already conducted initial interviews with interested Scrum Masters from select companies, who showed interest in the research as a means of supporting teams and gaining insights into the executed processes of teams. After the initial phase of participants recruitment, scheduled for two months, we will begin interviews, iteratively refining the interview guide. Interviews are expected to be completed within an additional two months, depending on interviewee availability and their schedules. The last two months will be used to code interview outcomes, analyze the data and summarizing the findings in an initial research paper. Based on the conclusions drawn from the gathered data, we will form further hypotheses on how to best support teams with software solutions, that will be evaluated in future work.

IV. Related Work

A. Software Development Artifacts

Part of our research hypothesis is that development data is already present in development teams. Due to the nature of modern software development, this is a necessity. Software engineers practically digitally ”self-document“ continuously, producing data about the development process and their steps while they work. Not only does the version control system log which change (the diff) was made at what point in time (the timestamp), but also who authored and committed the changes and what the goal of the change was (the commit message). Furthermore, there is likely more detailed information in a ticket within an issue tracker, clarifying the scope and context of the change, which can be referenced in the commit message. Other tools used on a daily basis by programmers, such as test frameworks, Continuous Integration services, static code analysis tools or code coverage analytics provide data on the status of the software at a particular point in time.

Especially interesting is the fact, that this information not only meticulously includes successes and progress but also documents failures, setbacks, and conflicts, something that is otherwise unlikely to be communicated.

Based on these data sources, a range of metrics can be constructed that might be used to guide process improvement in a development team. These can range from traditional software metrics, such as McCabe's cyclomatic complexity [15], to those of the agile context, such as effort estimation vs. real invested effort [16]. In the context of agile process improvement, especially the combinations of these metrics can be of interest. For example, it can be detected if the *Test-Driven Development* (TDD) process was followed [17].

B. Software Repository Mining

This research heavily draws from software repository mining techniques to analyze the development data of agile teams. Multiple frameworks and tools have been proposed in related literature to extract insights from software development repositories [18]–[20]. However, these tools mostly focus on analyses of large amounts of software repositories on GitHub or similar collaboration platforms. These tools excel at supporting analyses to answer questions such as “what is the most widely used open source license?” [20]. However, little research has focused on how insights from single a repository can be used by the people that created the data and thus have intricate knowledge of it and are able to interpret analyses extremely well. Furthermore, little research thus far has focused on how insights gained from repository analyses are used by development teams, e.g. for process improvement.

C. Software Process Improvement

Work related to this research can furthermore be found in the field of software process improvement (SPI) [21]. However, as Santana et al. point out, organizing software process improvement activities in environments employing agile methodologies is quite different than in more traditional ones [22]. The authors conducted a systematic literature review, including 423 papers published between 2001 and 2013. They conclude that it is necessary to adapt existing SPI approaches or to create new methods for agile contexts. Kuhrmann et al. note that there is a growing interest in agile methods and adopting agile principles in SPI [23]. However, in accordance with our own research to date, they note that research mostly focuses on student labs or a single case study in industry. The authors conducted a comprehensive systematic mapping study of publications related to SPI over the past 25 years. They identified agility becoming more relevant for SPI as companies adopt agile methods as a major research direction.

V. Summary

In previous work, we showed that analyzing the development data of university students provided actionable insights into the problems of agile development teams made up of students [8]–[10], [24]. Measurements, in this case, were derived from years of institutional knowledge and experience running the university course as well as from literature. However, these learning are not transferable to industry, with experienced professional developers working full-time in development teams, employing custom development processes. Encouraged by previous research results, we aim at understanding the methods and approaches as well as the difficulties and challenges involved in improving work processes in industry. To this end, we propose an interview study with those roles in companies, tasked with process improvement, conformance, and coaching, focusing on the currently implemented processes and the development data being produced in teams. We aim at gaining an understanding of how process improvement is conducted in teams, as a starting point for supporting teams in these activities. A core component of agile methodologies is the idea of self-organizing teams, i.e. teams that do not need management to give them structure and processes, but who create structures that work best for them [25].

Our vision is that this idea could also apply to software improvement processes, where teams can autonomously decide how their improvement approach should work. In order to make the first steps in this direction, the problem domain needs to first be better understood using empirical methods.

References

[1] K. Schwaber, "Scrum development process," in *Business Object Design and Implementation*. Springer, 1997, pp. 117–134.

[2] H. Kniberg, *Scrum and XP from the Trenches*. C4Media, 2007.

[3] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004, vol. 7.

[4] K. Schwaber and J. Sutherland, "The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game," Tech. Rep., 2017. [Online]. Available: http://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[5] Scrum Alliance, "The State of Scrum Report 2017 Edition," Scrum Alliance, Tech. Rep., 2017. [Online]. Available: https://www.scrumalliance.org/scrum/media/ScrumAllianceMedia/FilesandPDFs/StateofScrum/State0fScrum_2016_FINAL.pdf

[6] International Organization for Standardization, "ISO 9001:2015 Quality management systems," 2015.

[7] D. M. Fernández, W. Böhm, A. Vogelsang, J. Mund, M. Broy, M. Kuhrmann, and T. Weyer, "Artefacts in Software Engineering: What are they after all?" *International Journal on Software and Systems Modeling*, may 2018. [Online]. Available: http://arxiv.org/abs/1806.00098

[8] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner, "Agile metrics for a university software engineering course," in *IEEE Frontiers in Education Conference (FIE)*. Erie, PA: IEEE, oct 2016, pp. 1–5. [Online]. Available: https://doi.org/10.1109/FIE.2016.7757684

[9] C. Matthies, T. Kowark, K. Richly, M. Uflacker, and H. Plattner, "How surveys, tutors, and software help to assess Scrum adoption in a classroom software engineering project," in *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE)*. New York, New York, USA: ACM Press, 2016, pp. 313–322.

[10] ——, "ScrumLint: Identifying Violations of Agile Practices Using Development Artifacts," in *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '16*. New York, New York, USA: ACM Press, 2016, pp. 40–43. [Online]. Available: https://doi.org/10.1145/2897586.2897602

[11] VersionOne Inc., "The 11th Annual State of Agile Report," VersionOne Inc., Tech. Rep., 2017. [Online]. Available: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

[12] K. Louise Barriball and A. While, "Collecting data using a semi-structured interview: a discussion paper," *Journal of Advanced Nursing*, vol. 19, no. 2, pp. 328–335, feb 1994. [Online]. Available: https://doi.org/10.1111/j.1365-2648.1994.tb01088.x

[13] S. Hove and B. Anda, "Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research," in *11th IEEE International Software Metrics Symposium (METRICS'05)*, no. Metrics. IEEE, 2005, pp. 23–23. [Online]. Available: https://doi.org/10.1109/METRICS.2005.24

[14] I. Richter, F. Raith, and M. Weber, "Problems in Agile Global Software Engineering Projects especially within Traditionally Organised Corporations," in *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering - C3S2E '16*. New York, New York, USA: ACM Press, 2016, pp. 33–43. [Online]. Available: https://doi.org/10.1145/2948992.2949019

[15] T. J. McCabe, "A complexity measure," *Software Engineering, IEEE Transactions on*, no. 4, pp. 308–320, 1976.

[16] S. H. Kan, *Metrics and Models in Software Quality Engineering*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[17] A. Ju and A. Fox, "TEAMSCOPE: measuring software engineering processes with teamwork telemetry," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. New York, New York, USA: ACM Press, 2018, pp. 123–128. [Online]. Available: https://doi.org/10.1145/3197091.3197107

[18] J. Bevan, E. J. Whitehead, S. Kim, and M. Godfrey, "Facilitating software evolution research with kenyon," in *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering - ESEC/FSE-13*. New York, New York, USA: ACM Press, 2005, p. 177. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1081706.1081736

[19] E. Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi, "Sourcerer: mining and searching internet-scale software repositories," *Data Mining and Knowledge Discovery*, vol. 18, no. 2, pp. 300–336, apr 2009. [Online]. Available: http://link.springer.com/10.1007/s10618-008-0118-x

[20] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen, "Boa: A language and infrastructure for analyzing ultra-large-scale software repositories," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, may 2013, pp. 422–431. [Online]. Available: http://ieeexplore.ieee.org/document/6606588/

[21] S. Zahran, *Software Process Improvement: Practical Guidelines for Business Success*. Essex, UK, UK: Addison-Wesley Longman Ltd., 1998.

[22] C. Santana, F. Queiroz, A. Vasconcelos, and C. Gusmao, "Software Process Improvement in Agile Software Development A Systematic Literature Review," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, aug 2015, pp. 325–332. [Online]. Available: https://doi.org/10.1109/SEAA.2015.82

[23] M. Kuhrmann, C. Konopka, P. Nellemann, P. Diebold, and J. Münch, "Software process improvement: where is the evidence?: initial findings from a systematic mapping study," in *Proceedings of the 2015 International Conference on Software and System Process - ICSSP 2015*. New York, New York, USA: ACM Press, 2015, pp. 107–116. [Online]. Available: https://doi.org/10.1145/2785592.2785600

[24] C. Matthies, "Scrum2Kanban: Integrating Kanban and Scrum in a University Software Engineering Capstone Course," in *SEEM'18: IEEE/ACM International Workshop on Software Engineering Education for Millennials*. Gothenburg, Sweden: ACM, New York, NY, USA, 2018, p. 8. [Online]. Available: https://doi.org/10.1145/3194779.3194784

[25] R. Hoda, J. Noble, and S. Marshall, "Organizing self-organizing teams," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, vol. 1. New York, New York, USA: ACM Press, 2010, p. 285. [Online]. Available: https://doi.org/10.1145/1806799.1806843