Evaluation Of A Retrospective Practice

Olivia Denbu Vilhelmsson, Azer Hojlas
CINTE-19
Modern mjukvaruutveckling, IV1303

ABSTRACT:	(Azer)
SECTION I: Introduction	(Azer)
SECTION II: Background	(Olivia)
SECTION III: Research Method	(Azer)
SECTION IV: Project Results	(Olivia)
SECTION V: Analyze your results and evaluate your project work	(Azer)
SECTION VI: Conclusions and Future Work	(Azer and Olivia)

Abstract

Even though implementing retrospectives is viewed as mandatory for any self-respecting group of programmers, little is known about it's effectiveness. In this report we will evaluate this practice with respect to a few chosen criteria to shed light on just how effective it is, and whether or not anyone should be considering it's use. We have come to the conclusion of a resounding yes! Even though retrospectives are not always beneficial depending on the group dynamic and how one implements it, they are in most cases perceived as a constructive addition to any project worth its mettle.

Keywords: group dynamic, implementation, constructive, agile.

Section I: Introduction

The purpose of a retrospective is, as Jeff Dalton mentions in his book (Great Big Agile, 2012), "for each team or functional group to reflect on actions, results, and behaviors from the current sprint, and identify potential improvements for the next. Retrospectives align to one of the core principles from the Agile Manifesto, which states "at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."" ([3] Dalton, Jeff. (2018). Great Big Agile, page 219). He then proceeds to state that there are many types of retrospectives, but that they all are similar in purpose regardless of way of conduct, which is: to learn and improve.

In order to identify potential problems, one must gather data on the completed iteration ([13] Matthies, Christoph; (2020). *Playing with Your Project Data in Scrum Retrospectives*. Proceedings of the ACM/IEEE 42nd international conference on software engineering: Companion proceedings. 113-115). There is however no clear-cut way that prescribes how to gather the aforementioned data and as a matter of fact, several types of doing this exist, namely, *sprint retrospective, enterprise retrospective* and *heartbeat retrospective,* just to name a few. Because of these variations, the question of which is better naturally arises, and even more importantly, what is the overall benefit of retrospectives? Because "Little research deals with the quality and nature of identified problems or how progress towards removing issues is measured" ([12] Matthies, Christoph; (2019). *Agile process improvement in retrospectives.* Proceedings of the 41st international conference on software engineering: Companion proceedings. 150-152.), there have been few instances of qualitative research and even fewer quantitative ones, which is the purpose of this report. It is more clearly illustrated in the citation: "The impact of project retrospectives on process improvement initiatives has been questioned for many years. It is an important question because project retrospectives can be demanding in terms of time and organizational resources. Therefore, measuring whether retrospectives are an efficient methodology for instigating continuous process improvements is extremely important. Many have been committed to the notion that retrospectives are essential to continuous process improvement in organizations but unfortunately could not prove their efficacy in a scientific manner." ([6] Hayes, Darren; Grossman, Fred; Knapp, Constance; Rising, Linda. (2011). The impact of project retrospectives on process improvement initiatives: A case study. IEEE Long Island Systems, Applications and Technology Conference. IEEE)

In the following sections, we will try to evaluate this practice and uncover the elusive synergies of retrospectives. The report is structured in the following: Following this general introduction, section 2 will give a more detailed and specific one, section 3 explains our way of conduct and research methods, whilst section 4 is where the raw data is compiled. Section 5 deals with the analysis of the raw data, and finally in section 6 our conclusions will be presented.

Section II: Background

2.1 - Introduction to software development methods

In this section, we briefly give an introduction to traditional- and agile software development methods. The methods are presented in Section 2.1.1 and Section 2.1.2, respectively and in Section 2.1.3 we point out the major differences between them.

2.1.1 - Traditional development methods:

The traditional models have an approach which assumes that it is possible at the beginning of a project to plan and predict what the coming time will look like. It assumes that the requirements can be fully set at an early stage and that not a lot will change with it. One example of a traditional software development method is the *waterfall model*.
The working process is done sequentially and contains different phases. Initially there is an *analysis phase* which is used to summarize the requirements. This is then followed by the phase which focuses on the system design; the *design phas*e. When the design is set the project moves over to the *coding phase* which later on is followed by the *testing phase.* The purpose of testing the system is to make sure that the created software matches the initial requirements set by the customer. All of these phases are then completed by the *deployment* and now the product is given to the customer[15].

2.1.2 - Agile development methods:

The translation of the word Agile is a combination of for example the words fast, flexible and well coordinated. There are four important aspects and core values for Agile developments;

- *Value the individual and interaction* rather than processes and tools
- *Value well functioning software* rather than extensive documentation
- *Value cooperation with customers* rather than negotiation of contracts
- *Value adaptation of changes* rather than following a settled plan

Even though there is value in the right cornered points as well, the Agile-movement values the left sided more. For example the tools and processes are important, but they can never be in the way of a meeting between individuals. If that would be the case the conclusion is that the process has got to change[2].

Scrum is one way of working agile and conducts all work in cycles which are called *sprints,* also referred to as *iterations*. A sprint is a fixed, in forehand decided, time period normally with the span of two, three or four weeks. It starts off with a *sprint planning meeting* where you discuss and plan which functionalities to target and work on the coming iteration. You choose the highest prioritised items from the product-backlog, which is a strictly prioritized to-do list of the product, and this is then followed by a discussion where you break these bigger items into smaller pieces. The goal is to make sure that everyone in the team understands what is expected to be built[1].

Daily scrum meetings are used to continuously keep track of the team and the work to be done. Each team member gives a brief update on the status of his work and the team compares the overall situation with the *sprint backlog*[5].

The iteration ends with the team presenting a working, tested software to the product owner and the stakeholders. During this demonstration it is requested by everyone to give opinions on what is being monitored and to suggest improvements[1].

Finally there is a special session used for reflection and learning from the past sprint. The aim is to adapt this new knowledge into the coming iteration. These meetings are called retrospectives[5].

Another known Agile-method is Kanban. Scrum is known for describing *how the working tasks are being driven*, whereas Kanban describes *an improvement of the process*. These two methods are possible to combine and it is common to see Kanban as a version of Scrum only without the sprints. Working by the Kanban model allows the work to go in a constant flow and can be compared to the manufacturing of Toyota cars. This in contrast to Scrum where the work is coming in batches and is being checked off by its iterations. Nothing within Kanban prohibits working with iterations, but at the same time there is nothing that advocates it[2].

2.1.3 - Major differences between traditional- and agile development methods

A Stack Overflow study done in 2017 showed that more than 80% of the developers in the survey were using Agile practices. Although, there were indications that certain problems came with this transformation into the modern Agile methodology. Not everyone agrees to the benefits of changing into this new way of working and it resulted in another group of developers who only adapted the Agile in a superficial way. This division of interest in the new methodology created tension among the two opponents[7].

Examples of situations when it can be hard to introduce agile methods is:

- If the acceptance of changing the organisation is low
- If the organisation has no desire for collaboration between individuals or different departments
- If the key people in a group prefer to work individually instead of together[1].

Concerning decision making there is a major difference between the two approaches. Within the modern model decisions are being made as late as possible. The later the stage the more information and knowledge can be provided. It is better to know than to guess everything from the beginning.

The traditional approach has been getting critique for the fact that this model assumes that it is possible to predict and have a full understanding of the requirements at the absolute beginning of the process. The downside of this approach is that it is mostly common that the stakeholders within software development processes do not know what they are searching for and cannot formulate the exact requirements this early in the process. As a reaction to this methodology the Agile way of working was created[15].

The big difference is to now have a smaller team that spends a short amount of time on building a small item instead of having a large group working on something big during a long time period[9].

2.2 - Software development practices

In this section, we present software development practices. In Section 2.2.1 we provide a general description of practices and in Section 2.2.2 we provide a detailed description of the one we have chosen to evaluate in this report, *retrospectives.*

2.2.1 - General description of practices

Modern software development practices are *process tools* that will allow the work to become more effective. Depending on the situation different tools are better suited and therefore one does not

necessarily exclude the other one. It is highly common to mix and match between the different variations that exist. The purpose of these modern software practices is to provide you with certain limitations & guidelines. One value of these tools is that it *limits your options*. A practice that is lacking in constraints and allows you to do anything is not that useful.
Using the right modern software practice will help the process, but there is no guarantee of success just because you use it. A project can fail despite the utilisation of a great tool and likewise the project can succeed despite a lousy tool [9].

Some of the most common modern software practices are:

- Daily standup meetings (a Scrum practice)
- Limitation of queue sizes (a Kanban practice)[9]
- Just-In-Time (a Lean practice)
- Stop-the-Line-Culture (a Lean practice)
- The five why (a Lean practice)[1]

Several practices have been proposed but of the above written list this report will only focus on one specific method; *Retrospectives.*

2.2.2 - Detailed description of retrospectives:
Retrospectives is a practice used for improvements within the software development process. Within Scrum retrospectives are being held at the end of every sprint. The product owner discusses together with the team how they should improve to the next sprint and these improvements are then followed up at the end of the next coming iteration[1].

The standard format of an agile retrospective meeting starts off by *setting the stage* where the team is being welcomed and the aim of today's meeting is being described. This is followed by the stage *collecting data* which involves a review and participants sharing their experiences from the previous sprint and also a short discussion between the members. After this there is the stage of *creating insight* which involves further discussions and agreements on what issues to prioritize and creating solutions on how to solve them in the *settling on what to do* stage where the team is deciding on what areas that needs to be ameliorated. At last there is the stage of *closing the meeting* where the retrospective is being summarized and appreciation is shown to the participants[16].

Action items that have been detected as issues are defined as the outcome of the retrospective meeting. This analysis has a need for measurements, both of the current situation but also of the coming statuses to later make the comparison with. Examples of measurements are velocity, number of stories fulfilled and the total of refactored code[11].

The general format of a retrospective can vary but a common way is to have a whiteboard for visualizing the reflections of the meeting. It consists of three columns which can vary, but one example is: "*Good*", "*Could have done better*" and "*Improvements*". The two first columns focus on the past and the third one on the future. The team brainstorms on the whiteboard using Post-its and later the team will decide on which items to prioritize for the coming sprint. One way to do this is letting each team member have three "dot vots" where he or she votes on what is the most important. Based on this a few improvements are being chosen to focus on for the coming sprint[8].

Retrospective games are another way of facilitating reflection within the team. The aim for these games is to help the team to think together and is a time boxed process[13].

Section III: Research Method

In this section we describe our research method. Section 2.1 we account for our research phases. In section 2.2 we present our evaluation criteria.

2.1 Research phases

Our research consists of four phases, illustrated in figure 1. These are: (1) Procure Appropriate Literature, (2) Create Evaluation Criteria, (3) Literature Evaluation With Regards To Criteria and (4) Compile Results.

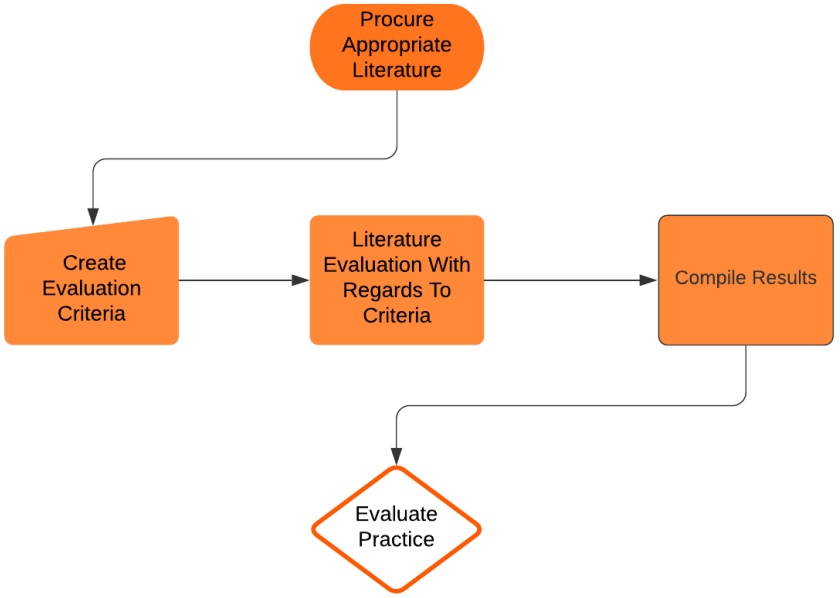

Figure 1, our research phases

Literature study phase

In this phase we searched for appropriate literature that would be relevant with regards to our criterias. In ca 25 hits (articles, essays, books etc), we chose the 15 most relevant. These are stated below:

Create Evaluation Criteria phase

In this phase we discussed what criteria would best illustrate the impacts of retrospectives, and proceeded to write these down. Even though most of the criteria are qualitative in nature, we found these the most suitable for shedding light on the matter at hand.

Literature Evaluation With Regards To Criteria Phase

In this phase we re-read all our material, whilst keeping in mind our criteria. We then proceeded to document our findings with regards to these aforementioned criteria. The research was conducted by

skimming through the material until we stumbled upon information that was relevant to our study, then compiling these paragraphs or sometimes pages into a separate document. We collected as much data as possible in order to have a satisfactory selection of backing arguments when compiling our findings and presenting our conclusions.

Compile Results Phase

Here we compiled our results and presented our conclusions in an easy to read and digestible form. This is important as to the necessity of the reader having to understand the research that would otherwise be in vain.

2.2 Evaluation criteria

Due to the scarcity of quantitative criteria, we had to make due with only one. The other (qualitative) two chosen criteria however, have proven more than enough with regards to our research.

Evaluation criteria
Impact on time-frame
Comparative benefits
Effect on group dynamic and satisfaction

Figure 2, our criteria

Our model consists of six components. As illustrated in Figure 2, these are (1) Impact on time-frame, (2) Comparative benefits and (3) Effect on group dynamic and satisfaction. Below we present why we chose the specified criteria:

- **Impact on time-frame:**
  This was selected because of it's quantitative properties and practical ease of measurement. The premise is simple, if retrospectives really are as efficient as they imply, then there should be a particular amount of time saved by implementing them instead of randomly reflecting about the states of the processes.

- **Comparative benefits:**
  This criteria regards the fact as to why one should participate in this process in the first place. We felt that this criteria was necessary due to the fact that some readers of this report might not be familiar with the use of, and more importantly, the benefits of retrospectives. Hopefully this criteria will remedy that.

- **Effect on group dynamic and satisfaction:**
  Depending on the group, the observations under the scope of this criteria can seem arbitrary, however, as this practice is entirely a group activity, the need for such a criteria becomes evident. We found it important that all the possible effects of retrospectives on group dynamic and satisfaction be recorded, regardless if they are positive or otherwise.

Section IV: Project Results

In this section we present the results of our research. Section 4.1 describes the results of the data collection connected to the first evaluation criteria *Impact on time-frame*. Section 4.2 focuses on results to the evaluation criteria *Comparative benefits* and Section 4.3 presents data for evaluation criteria *Effect on group dynamic and satisfaction.*

4.1 - Impact on time-frame

4.1.1 - Repetitive meetings, bad preparations and lack of improvement

When these meetings tend to be too repetitive it leads to affecting the team motivation in a bad way. The members get bored and tired of listening to the same things over and over again.

If the preparations before a meeting are not done properly and the agenda of the session is not clear there is a risk that the participants leave the meeting feeling lost and with a dissatisfaction that their time has not been valued.

A failure of deciding action items for the coming sprint will lead to a situation of “All talk - No Action”. The purpose of aiming for a clearer path towards ameliorations will then lose its impact and value[4].

There can also be the opposite situation where the team is good in identifying weaknesses and strong practice but the problem is that there is no real analysis done of the situation. The problem here is that there is no follow-up mechanism and there is no plan created to reduce the current problem state[14].

4.1.2 - Better estimation of tasks

By lifting discussions of unfinished tasks it has shown to give a better understanding of the time estimation for assignments.
When for example three team members explain that they have not been able to finnish their given tasks from the last sprint the team can see a common pattern. By surfacing this obstacle it has shown to be helpful in the process of actually understanding how much time and effort that is needed for finishing it. This has also been shown to generate more discussions about *why* these tasks have been challenging which then helped to understand the reason behind it[16].

4.1.3 - Early problem identification and lower risk of repeatedly executed mistakes

Retrospective meetings have shown to result in an earlier stage of problem identification and it also adds up to better improvements within the team[12]. By critically reflecting on the past period there is a lower risk of making the same errors with a repeatable pattern. It has shown that only to identify a problem could be enough for the problem to be solved during the coming iteration[8].

By sharing problems to other team members it generates a better awareness among the team connected to the problem area. By listening to other team members the individuals could get a better understanding for their own tasks which then helps the team as a whole to solve the problem[16].

4.2.3 - Playground retrospective

If the team doesn't really understand the reason for hosting these retrospectives there is a risk of it turning into a session of playful exercises. Agile games and team-building activities might take place to release some stress from the previous iteration but building on the actual goals and improvements

will not take place. The agenda for the meetings might be constructed from what has been found in literature but the actors do not see the purpose of doing it[14].

4.2 - Comparative benefits

4.2.1 - Valuable information coming through

During retrospective meetings information comes up that is very valuable. For example the team could come up with the conclusion that they find it hard to concentrate because of the fact that the sales manager is asking this team for participants during their meetings and this results in problems for the software developers.
The retrospective meeting could then lead to coming up with a conclusion on how to manage this. For example to invite the sales team manager to their meetings sometimes[8].

4.2.2 - Higher acceptance of team members ideas

Even without retrospective meetings it is possible that team members come up with good ideas. Although the planning in sessions with retrospectives proves that it is a higher chance that the team will accept and be on board with the idea during these sessions. The fact that it is an open platform where everyone is allowed to pitch in makes this process better[8].

4.3 - Effect on group dynamic and satisfaction

4.3.1 - Complaining game and negative focus

Depending on how the meetings are being framed there is a risk that the analyzing and focus on improvements leads to an approach where the negativity is centered rather than underlining what the positive outcomes were from the previous sprint. The team might lose their team spirit due to a negative focus.

Also the retrospective meetings hazards can turn into a complaining game. Participants could see this as an opportunity to complain and relieve negative energy rather than a chance to talk about improvements[4].

4.3.2 - Harmony valued higher than having critical discussions

The team risks having a negative effect when harmony is valued higher than having critical discussions. Aiming for a constant state of ease within the group could block the part where you as a group really get down to the bottom of the problem. This type of “Group thinking” will not make the team develop[4].

4.3.3 - Game of pointing fingers and blaming others

By critically analysing the executed sprint there is a risk of creating a bad atmosphere within the team. Depending on how this process is done there is a risk of targeting certain team members in a negative way. If not done correctly it might make it more into a game of pointing fingers and finding ways to blame people rather than constructively trying to find solutions.

The retrospective meetings are based on aiming for an open communication within the team. As a cause of this constructive process members can get triggered to take things personally. Statements of other stakeholders might result in team members feeling offended or hurt.

If team members do not feel comfortable there is a risk of being reluctant to openly speak about their experiences and problems concerning the group[4].

4.3.4 - Increased focus, clearer objectives and increased motivation

According to studies the use of gamification in retrospective meetings increase the participants' focus on development tasks and to construct clearer objectives. Several researchers have indicated that it also gains the motivation of the team and to have a great impact on social behaviour[10].

4.3.5 - Sharing feelings

Besides sharing practical issues and obstacles, there is also a part of retrospectives where the agile teams share their emotional experiences from the past sprint. The sharing of feelings can be visualized in different ways but for example by drawings. There have been examples of where the shardom of positive feelings has had a great impact on the productivity of the group[16].

4.3.6 - Leader taking over too much

Situations where the analysis of the previous sprint is done mainly by the coaches and managers has shown to give a bad influence on the team. The team will not feel as if they are in charge of their own process. The problem diagnosis, identification of impediments and proposals of solutions should be done by the team as a whole[14].

Section V: Analysis

In this section we conduct a thorough analysis of the data presented in the results section. This allows for the inclusion of our own opinions and should be viewed as such.

Our studies have shown that generally, retrospectives make a fine and constructive addition to any team-oriented project. There are however some objections to this, namely that “by critically analysing the executed sprint there is a risk of creating a bad atmosphere within the team. Depending on how this process is done there is a risk of targeting certain team members in a negative way. If not done correctly it might make it more into a game of pointing fingers and finding ways to blame people rather than constructively trying to find solutions”. It would seem that a retrospective approach doesn’t always yield the desired results. This depends primarily on two variables, one dependent and one independent, i.e one where the outcome depends on how the practice has been conducted, and one where previous inter-group relations play a large part.

If we first address the matter of bad execution, that retrospectives can have a negative impact if not correctly implemented, as with anything, this process will yield bad results if approached as intended. This says more about the team implementing the practice than it says about the practice itself. Our sources have stated, that as long as the practice is intelligently applied, that there should be no negative outcome. Therefore we consider any faulty use of this process to be just that, an anomaly that should not be repeated for the sake of the interests of the stakeholders, and that any such faulty use cannot be seen as indicative for what a retrospective process should yield. If we use the analogy of a person without a driving license or any past experience driving a car just to moments later crash, our first impulse would not be to judge the car for crashing. Our conclusion is that it requires a special kind of skill or experience to be using this practice, just as one would assume out of any driver on the highway.

The second touches more on the subject of group dynamics, described by the following: “The retrospective meetings are based on an open communication within the team. As a cause of this constructive process members can get triggered to take things personally. The statements of other stakeholders might result in team members feeling offended or hurt”. We are of the belief that regardless of the intelligence and capabilities of the various participants, that the practice is doomed to fail if the team members are not comfortable working with each other, or more importantly, if they are not comfortable with criticizing each other (which most tight-knitted groups should and can do). For a lack of better words, criticism is good, it is an integral part of retrospective practices, and no matter how well presented and polite this criticism is, it will remain just that. Thus, we believe that retrospectives are only warranted under the condition that there exists some form of mutual trust and comfortability between the participants. Under any other conditions, a retrospective process might be regarded as counter-productive, as would any other process, task or project that isn’t conducted by a fully matured group.

Finally, when we have dealt with the potential issues of the practice, we want to analyze the implied benefits of the retrospectives. All the conceivable benefits that one might have expected from the practice have been confirmed by our various sources. These are namely: Valuable information being targeted and spread around the team; Higher acceptance of team members' ideas; Problem identification and lower risk of repeatedly executed mistakes. Team members also achieve a better

grasp of the time-frame for various projects. Consequently teams become more effective at managing their time and completing projects before deadlines, which directly translates to greater stakeholder satisfaction.

Section VI: Conclusions

We hope that this report will answer the question of whether it is useful to implement retrospectives in a software process or not. Our answer is a clear yes due to the fact the impact of retrospectives is universally positive. When it is not it depends on exterior factors such as an undesirable group dynamic or bad knowledge of the actual practice.

Our sources have consistently shown that even though the retrospective process is a time consuming activity, that it in fact does make the work more effective, i.e it results in greater productivity and lesser time spent on projects.

In addition to this, we believe ourselves to have proven a slew of other benefits, viz., valuable information being targeted and spread around the team, higher acceptance of team members' ideas, problem identification and lower risk of repeatedly executed mistakes and so on.

As we have stated before, retrospectives are only warranted under the condition that there exists some form of mutual trust and comfortability between the participants. If a team avoids the aforementioned pitfalls and utilizes retrospectives correctly as a well-functioning team, then all these previously mentioned benefits are ripe for the picking.

Used references

[1] Björkholm, Tomas; Brattberg, Hans. (2013). *Prioritera, Fokusera, Leverera*. Crisp. 1.2edn. ISBN 978-91-978630-5-6

[2] Björkholm, Tomas; Brattberg, Hans. (2016). *Prioritera, Fokusera, Leverera*. Crisp. 1.4edn. ISBN 978-91-978630-5-6

[3] Dalton, Jeff. (2018). Great Big Agile. Online 1st edition. Apress, Berkeley, CA. 978-1-4842-4206-3

[4] Ernst, Alexander; Dobrigkeit, Franziska; Matthies, Christoph. (2019). *Counteracting agile retrospective problems with retrospective activities.* Communications in computer and information science book series (CCIS, volume 1060). 532-545.

[5] Freykamp, Frank; Rubart, Jessica. (2009). *Supporting daily scrum meetings with change structure.* Proceedings of the 20th ACM conference on Hypertext and hypermedia. 57-62.

[6] Hayes, Darren; Grossman, Fred; Knapp, Constance; Rising, Linda. (2011). The impact of project retrospectives on process improvement initiatives: A case study. IEEE Long Island Systems, Applications and Technology Conference. IEEE. 978-1-4244-9878-9

[7] Ivanov, Vladimir; Masyagin Sergey; Rogers Alan; Succi Giancarlo; Tormasov Alexander; Yi Jooyong; Zorin Vasily. *Comparison of Agile, Quasi-Agile and Traditional Methodologies.* (2019). Advances in intelligent systems and computing book series (AISC, volume 925). 128-137.

[8] Kniberg, Henrik. (2015). *Scrum and XP from the trenches*. 2edn. ISBN 978-1-392-22427-8

[9] Kniberg, Henrik; Skarin, Mattias. (2009). *Kanban and Scrum, making the most of both*. ISBN 978-0-557-13832-6

[10] Mas Antònia; Poth Alexander; Sasabe Susumu. (2018). *SPI with Retrospectives: A Case Study.* Communications in computer and information science book series (CCIS, volume 896). 456-466.

[11] Matthies, Christoph; Dobrigkeit, Franziska; Hesse, Guenter. (2019). *An additional set of (automated) eyes: Chatbots for agile retrospectives.* Proceedings of the 1st International Workshop on Bots in Software Engineering. 34-37.

[12] Matthies, Christoph; (2019). *Agile process improvement in retrospectives.* Proceedings of the 41st international conference on software engineering: Companion proceedings. 150-152.

[13] Matthies, Christoph; (2020). *Playing with Your Project Data in Scrum Retrospectives*. Proceedings of the ACM/IEEE 42nd international conference on software engineering: Companion proceedings. 113-115.

[14] Medinilla, Ángel. (2014). *Retrospectives and Kaizen Events.* Agile Kaizen. 37-58.

[15] Ur Rehman, Israr; Ullah, Sajid; Rauf, Abul; Shahid, Arshad Ali. (2010). *Scope management in agile versus traditional software development methods*. Proceedings of the 2010 National Software Engineering Conference. 1-6.

[16] Andriyani, Yanti; Hoda, Rashina; Amor, Robert. (2017). *Reflection in Agile Retrospectives*. Lecture notes in business information processing book series (LNBIP, volume 283). 3-19.