


# Experience vs Data: A Case for More Data-informed Retrospective Activities

Christoph Matthies<sup>[0000-0002-6612-5055]</sup>, Franziska  
Dobrigkeit<sup>[0000-0001-9039-8777]</sup>

Hasso Plattner Institute  
University of Potsdam, Germany  
`christoph.matthies@hpi.de`, `franziska.dobrigkeit@hpi.de`

**Abstract.** Effective Retrospective meetings are vital for ensuring productive development processes because they provide the means for Agile software development teams to discuss and decide on future improvements of their collaboration. Retrospective agendas often include activities that encourage sharing ideas and motivate participants to discuss possible improvements. The outcomes of these activities steer the future directions of team dynamics and influence team happiness. However, few empirical evaluations of Retrospective activities are currently available. Additionally, most activities rely on team members experiences and neglect to take existing project data into account. With this paper we want to make a case for data-driven decision-making principles, which have largely been adopted in other business areas. Towards this goal we review existing retrospective activities and highlight activities that already use project data as well as activities that could be augmented to take advantage of additional, more subjective data sources. We conclude that data-driven decision-making principles, are advantageous, and yet underused, in modern Agile software development. Making use of project data in retrospective activities would strengthen this principle and is a viable approach as such data can support the teams in making decisions on process improvement.

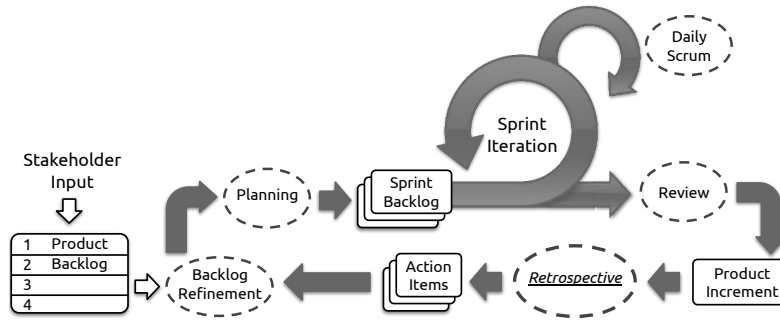
**Keywords:** Retrospective, Scrum, Agile Methods, Data-driven Decision Making, Data-informed Processes

## 1 Introduction

Agile development methods, particularly Scrum, which focus on managing the collaboration of self-organizing, cross-functional teams working in iterations [1], have become standards in industry settings. The most recent survey of Agile industry practitioners by *Digital.ai*<sup>1</sup>, conducted between August and December 2019, showed that Scrum continued to be the most widely-practiced Agile

<sup>1</sup> formerly *CollabNet VersionOne*

method: 75% of respondents employed Scrum or a Scrum hybrid [2]. In the survey, which included 1,121 full responses, both of the top two Agile techniques employed in organizations were focused on communication and gathering feedback: the Daily Standup (85%) and Retrospective meetings (81%). The importance of these meetings was also stated in a previous, similar survey by the *Scrum Alliance*. A vast majority of respondents (81%) to this 2018 survey stated that their teams held a Retrospective meeting at the end of every Sprint, while 87% used Daily Scrum meetings [3]. A prototypical, generalized flow through the Scrum method, depicting the different prescribed meetings and process artifacts, i.e. the context of Retrospectives, is represented in Figure 1.



**Fig. 1.** Prototypical flow through the Scrum process, based on [4]. Process meetings are represented by circles, process artifacts and outcomes as squares. Scrum’s process improvement meeting, the Retrospective, is highlighted.

In this research, we focus on the popular Retrospective meeting, which forms the core practice of process improvement approaches in the Scrum method, and the activities that are used in it. Retrospectives are a realization of the “inspect and adapt” principle [1] of Agile software development methods [5].

### 1.1 Retrospective Meetings

Recent research has pointed to Retrospective meetings as crucial infrastructure in Scrum [6]. Similarly, Retrospectives have also been recognized as one of the most important aspects of Agile development methods by practitioners [7]. The seminal work on Scrum, the *Scrum Guide*, defines the goal of Retrospectives to ascertain “how the last Sprint went” regarding both people doing the work, their relationships, the employed process, and the used tools [1]. As such, Retrospectives cover improvements of both technical and social/collaboration aspects. Teams are meant to improve their modes of collaboration and teamwork, thereby also increasing the enjoyment in future development iterations [8]. The Scrum framework prescribes Retrospectives at the end of each completed iteration. Teams are meant to generate a list of improvement opportunities, i.e. “action

items” [8], to be tackled in the next iteration. Retrospective meetings focus less on the quality of the produced product increment, but more on how it was produced and how that process can be made smoother and more enjoyable for all involved parties in the next iteration.

While Scrum is a prescriptive process framework, suggesting concrete meetings, roles, and process outcomes, the Scrum Guide also points out: “Specific tactics for using the Scrum framework vary and are described elsewhere” [1]. For Retrospectives, this means that while the meeting’s goal of identifying improvement opportunities is clear, the concrete steps that teams should follow are not and are up to the individual, self-organizing Scrum teams [9, 10]. One of the easiest and most effective ways to generate the types of process insights that Retrospectives require is by relying on those most familiar with the teams’ executed processes: team members themselves. Their views and perceptions of the previous, completed development iteration are, by definition, deeply relevant as inputs for process improvement activities. Furthermore, these data points are collectible with minimal overhead, e.g. by facilitating a brainstorming session in a Retrospective meeting, and they are strongly related to team satisfaction [11].

## 1.2 Data Sources used in Retrospective Meetings

Most of the data that forms the basis of improvement decisions in current Retrospectives is, at present, based on the easily collectible perceptions of team members. However, modern software development practices and the continuing trend of more automated and integrated development tools have opened another avenue for accessing information on teams’ executed process: their *project data* [12, 13, 14]. This project data includes information from systems used for such diverse purposes as version control (what was changed, why, when?), communication (what are other working on?), code review (feedback on changes), software builds (what is the testing status?), or static analysis (are standards met?). The data is already available, as modern software engineers continuously document their actions as part of their regular work [15, 16]. The development processes of teams, their successes as well as their challenges, are “inscribed” into the produced software artifacts [17]. This type of information, which can be used in Retrospectives, in addition to the subjective assessments of team members, has been identified as “a gold-mine of actionable information” [18]. More comprehensive, thorough insights into teams’ process states, drawn from activities that make use of both project data and team members’ perceptions, can lead to even better results in Retrospectives [8].

## 1.3 Research Goals

In this research, we focus on the integration of project data sources into Agile Retrospective meetings. In particular, we investigate to which extent project data analyses are already provided for in Retrospective activities and how more of them could benefit from data-informed approaches in the future. We provide an overview of popular activities and review the types of data being employed

to identify action items, i.e. possible improvements. We highlight those activities that already rely on software project data in their current descriptions as well as those that could be augmented to take advantage of the information provided by project data sources. We argue that the principles of data-driven decision making, which have already been adopted in many business areas [19], are suitable and conducive, yet underused, in the context of modern Agile process improvement.

## 2 Retrospective Activities

The core concept of Retrospectives is not unique to Scrum. These types of meetings, focusing on the improvement of executed process and collaboration strategies, have been employed since before Agile methods became popular. Similarly, team activities or “games” that meeting participants can play to keep sessions interesting and fresh have been used in Retrospectives since their inception [20]. These, usually time-boxed, activities are interactive and designed to encourage reflection and the exchange of ideas in teams. Derby and Larsen describe the purpose of Retrospective activities as to “help your team think together” [8]. Retrospective games have been shown to improve participants’ creativity, involvement, and communication as well as make team members more comfortable participating in discussions [10]. The core idea is that meeting participants already have much of the information and knowledge needed for future process improvements, but a catalyst is needed to start the conversation.

In 2000, Norman L. Kerth published a collection of Retrospective activities [20]. Additional collections were published in the following years by different practitioners as well as researchers [21, 22, 23, 24]. Table 1 presents an overview of the literature containing collections of Retrospective activities.

**Table 1.** Sources of Retrospective activities in literature.

Year	Reference Name of Reference
2006	[8] Agile Retrospectives - Making Good Teams Great
2006	[21] Innovation Games
2013	[22] The Retrospective Handbook
2014	[23] Getting value out of Agile Retrospectives
2015	[24] Agile Retrospective Kickstarter
2015	[11] Fun Retrospectives
2018	[25] Retromat: Run great agile retrospectives!

A generalized meeting agenda for Retrospective was proposed in 2006 by Derby and Larsen. It features five consecutive phases: (i) *set the stage* (define the meeting goal and giving participants time to “arrive”), (ii) *gather data* (create a shared pool of information), (iii) *generate insight* (explore why things happened, identify patterns within the gathered data), (iv) *decide what to do*

(create action plans for select issues), and (v) *close* (focus on appreciations and future Retrospective improvements) [8]. This plan has since established itself and has been accepted by other authors [25]. Retrospective activities remain an open area of investigation and continued learning, with current research further exploring the field [26, 27, 4].

While research articles and books offer extensive collection efforts regarding Retrospective activities, Agile practitioners rely on up-to-date web resources in their daily work, rather than regularly keeping up with research literature [27, 28]. The *Retromat*<sup>2</sup> [25] is a popular, comprehensive and often referenced [29, 27, 7], online repository of Retrospective activities for meeting agendas. It currently contains 140 different activities for five Retrospective meeting phases<sup>3</sup>.

### 3 Review of Retrospective Activities

As the Retromat repository represents the currently best updated, most complete list of Retrospective activities in use by practitioners [29, 27, 30], we employ its database as the foundation of our review. Our research plan contains the following steps:

- Extract activities that provide or generate inputs for discussion in Retrospectives
- Identify the specific data points being collected
- Categorize data points by their origin
- Study those activities in detail which already (or are close to) taking project data into account

#### 3.1 Activity Extraction

The Retromat, following Derby and Larsen’s established model [8], features activities and games for the five Retrospective phases<sup>4</sup> *set the stage*, *gather data*, *generate insight*, *decide what to do* and *close the Retrospective*. As this research focuses on the types of gathered inputs employed for meetings, we initially collected all activities classified by the Retromat as suitable for the *gather data* phase. This meeting phase aims to help participants remember and reflect and is aimed at collecting the details of the last iteration, in order to establish a shared understanding within the team. We extracted 35 activities intended for the *gather data* from the Retromat repository. These activities are listed in Table 4 in the Appendix.

Additionally, we reviewed the Retromat activities prescribed for all other phases to ensure that we did not miss any activities that gathered data as part of their proceedings. These could have been classified under different phases, as data gathering and analysis steps are often intertwined, or because the activity’s

<sup>2</sup> available at <https://retromat.org>

<sup>3</sup> <https://retromat.org/blog/history-of-retromat/>

<sup>4</sup> <https://retromat.org/blog/what-is-a-retrospective/>

main focus is broader than data collection. This step yielded an additional four activities that, at least partly, base their procedures on collected data: “3 for 1 - Opening” (assessments of iteration results and number of communications), “Last Retro’s Actions Table” (collecting assessments of previous action items), “Who said it?” (collecting memorable quotes), and “Snow Mountain” (using the Scrum burndown chart)<sup>5</sup>. The first three of these were classified in the Retromat under the *set the stage* phase, the last as *generate insights*.

### 3.2 Identification of Retrospective Inputs

We analyzed the textual descriptions provided within the Retromat collection for each of the extracted activities of the previous research step. We manually tagged each of the activities with labels regarding the specific data points that are collected and used as inputs for the following actions. Many activity descriptions featured subsequent aggregation and synthesis actions, e.g. dot-voting or clustering, from which we abstracted. The generated short data labels describe the specific outcomes of the initial data acquisition within activities. Examples include “numerical ratings of performed meetings”, “notes on what team members wish the team would learn”, or “collection of all user stories handled during the iteration”. Multiple activity descriptions contained mentions of physical representations of collected data points, which we generalized. For example, we consider “index cards” and “sticky notes” filled by meeting participants with their ideas to be instances of the more general “notes”. The results of this tagging step are shown in Table 2.

### 3.3 Classification of Retrospective Data Sources

We categorized activities based on the origins of their data inputs, using the generated descriptions. We distinguish whether the gathered data is (i) drawn solely from team members’ perceptions (no mention/reliance on software project data), (ii) is directly extracted from project data sources, or (iii) is ambiguous, i.e. could be drawn from either source, depending on team context and interpretation. We consider the term “project data” as an overarching collection of software artifacts. We follow Fernández et al.’s definition of the term “software artifacts” [31], in that we consider them “deliverables that are produced, modified, or used by a sequence of tasks that have value to a role”. These artifacts are often subject to quality assurance and version control and have a specific type [31].

The vast majority, i.e. 86% (30 of 35), of proposed *gather data* activities in the Retromat collection make no mention of software project data and do not take advantage of it. It should be noted that most of these *gather data* activities are very similar in terms of the type of collected data. They tend to deal with team members’ answers to varying prompts or imagined scenarios, aimed at starting discussions. Examples of such prompts include “mad, sad, glad“, “start,

<sup>5</sup> <https://retromat.org/en/?id=70-84-106-118>

**Table 2.** Overview of the types of inputs employed in the selected Retrospective activities. Activities above the divide are part of the *gather data* phase, those below are included after reviewing the activities of other phases. The categories of activities that gather data through specific prompts are italicized.

Shortened name	#	Type of activity input (regarding last iteration)
Activities from the <i>gather data</i> Retrospective phase		
<i>Timeline</i>	4	List of memorable/personally significant events
<i>Analyze Stories</i>	5	Collection of user stories handled during the iteration
<i>Like to like</i>	6	Notes on things to <i>start doing</i> , <i>keep doing</i> and <i>stop doing</i>
<i>Mad Sad Glad</i>	7	Notes on events when team members felt <i>mad</i> , <i>sad</i> or <i>glad</i>
<i>Speedboat/Sailboat</i>	19	Notes on what drove the team <i>forward</i> & what <i>kept it back</i>
<i>Proud &amp; Sorry</i>	33	Notes of instances of <i>proud</i> and <i>sorry</i> moments
<i>Self-Assessment</i>	35	Assessments of team state regarding Agile checklist items
<i>Mailbox</i>	47	Reports of events or ideas collected during the iteration
<i>Lean Coffee</i>	51	List of topics team members wish to be discussed
<i>Story Oscars</i>	54	Physical representations of completed user stories
<i>Expectations</i>	62	Text on what team members expect from each other
<i>Quartering</i>	64	Collection of everything the team did during iteration
<i>Appreciative Inquiry</i>	65	Answers to positive questions, e.g. best thing that happened
<i>Unspeakable</i>	75	Text on the biggest unspoken taboo in the company
<i>4 Ls</i>	78	Notes on what was <i>loved</i> , <i>learned</i> , <i>lacked</i> & <i>longed for</i>
<i>Value Streams</i>	79	Drawing of a value stream map of a user story
<i>Repeat &amp; Avoid</i>	80	Notes on what practices to <i>avoid</i> and which to <i>repeat</i>
<i>Comm. Lines</i>	86	Visualization of the ways information flows in the process
<i>Satisfaction Hist.</i>	87	Numerical (1-5) ratings of performed meetings
<i>Retro Wedding</i>	89	Notes on categories something <i>old</i> , <i>new</i> , <i>borrowed</i> & <i>blue</i>
<i>Shaping Words</i>	93	Short stories on iteration, including a 'shaping word'
<i>#tweetmysprint</i>	97	Short texts/tweets commenting on the iteration
<i>Laundry Day</i>	98	Notes on <i>clean</i> (clear) & <i>dirty</i> (unclear/confusing) items
<i>Movie Critic</i>	110	Notes on movie critic-style categories: <i>Genre</i> , <i>Theme</i> , <i>Twist</i> , <i>Ending</i> , <i>Expected?</i> , <i>Highlight</i> , <i>Recommend?</i>
<i>Genie in a Bottle</i>	116	Notes on 3 wishes: for <i>yourself</i> , <i>your team</i> and <i>all people</i>
<i>Hit the Headlines</i>	119	Short headlines on newsworthy aspects of the iteration
<i>Good, Bad &amp; Ugly</i>	121	Notes on categories <i>good</i> , <i>bad</i> & <i>ugly</i> concerning the iteration
<i>Focus Principle</i>	123	Assessments on relative importance of Agile Manifesto principles
<i>I like, I wish</i>	126	Notes on <i>likes</i> and <i>wishes</i> concerning the iteration
<i>Delay Display</i>	127	Notes on team <i>destination</i> , <i>delay</i> & <i>announcement</i>
<i>Learning Wish List</i>	128	Text on what team members wish the team would learn
<i>Tell me something I don't know</i>	133	Facts and questions, in game show fashion, on something that only one team member knows and most others do not
<i>Avoid Waste</i>	135	Notes on the <i>7 categories of waste</i> in the process
<i>Dare, Care, Share</i>	137	Notes on <i>bold wishes</i> , <i>worries</i> & <i>feedback/news</i>
<i>Room Service</i>	139	Notes on the prompts <i>Our work space helps me/us...</i> and <i>Our work space makes it hard to...</i>
Activities from phases <i>set the stage</i> and <i>generate insights</i>		
<i>3 for 1</i>	70	Points in coordinate plane of satisfaction with results and communication
<i>Retro Actions Table</i>	84	List of last Retrospectives action items
<i>Who said it?</i>	106	Quotes collected from project artifacts
<i>Snow Mountain</i>	118	Burndown chart of problematic Sprint

stop, continue“, “good, bad, ugly” or “proud and sorry”. All of these activities are, by default, drawn from the individual perceptions and experiences of team members.

The nine activities that we identified in our review as featuring (possible) connections to development data—five from the *gather data*, three from *set the stage* and a single one from the *generate insights* phase—are shown in Table 3 and are discussed in the following two sections.

**Table 3.** Overview of Retromat activities not solely reliant on team members’ perceptions. Activities which could be connected to project data, depending on how they are executed, are marked as *Possible*.

Activity #	Name	Data used as (partial) input for the activity and subsequent steps	Project Data
5	Analyze Stories	Collection of all user stories handled during the iteration	<i>Yes</i>
54	Story Oscars	Physical representation of all stories completed in the last iteration	<i>Yes</i>
84	Last Retro’s Actions Table	List of outcomes of the last Retrospective, i.e. action items/improvement plans	<i>Yes</i>
106	Who said it?	Literal quotes of team members extracted from communication channels, e.g. emails, chat logs or ticket discussions	<i>Yes</i>
35	Agile Self-Assessment	Assessments of team state regarding Agile checklist items	<i>Possible</i>
64	Quartering - Identify boring stories	Collection of “everything” the team did in the last iteration	<i>Possible</i>
70	3 for 1	Number of times team members coordinated in the last iteration	<i>Possible</i>
79	Value Stream Mapping	Drawing of a value stream map concerning a particular user story	<i>Possible</i>
118	Snow Mountain	The shape of the Scrum Burndown chart of a problematic iteration	<i>Possible</i>

## 4 Activities Already Reliant on Project Data

Of the overall nine activities identified in this research that feature (possible) connections to project data, four make direct mentions of specific development artifacts in their descriptions on Retromat:

- *Analyze Stories*
- *Story Oscars*
- *Last Retro’s Actions Table*
- *Who said it?*



These are marked as *Yes* regarding the use of project data in Table 4. Of these four activities, two employ the user stories of the last iteration as inputs, which are analyzed and graded by meeting participants in the following steps. The other two are concerned with the outcomes of the last Retrospective meeting and an extract of intra-team communications. The user stories/work items of modern Agile teams are usually contained in an issue tracker system [32] or can be acquired in printed form from a shared workspace or board [7]. Persisting the outcomes of Retrospectives, i.e. making note of the resulting action items and documenting meeting notes, is a common practice of Agile processes [6] and enables the tracking of progress towards these goals. Furthermore, digital communication tools, e.g. bug reports, mailing lists, or online forums, and the artifacts that result from their usage form a core part of modern software development [33]. The fact that these project artifacts are already present and are produced as part of the regular tasks of modern software developers, means that they can be collected with minimal overhead [34].

The four Retrospective activities we identified in this review as already employing project data represent only a small fraction of the 140 overall activities included in the Retromat. However, these are the activities that explicitly follow Derby and Larsen’s principle of having the *gather data* phase of Retrospective meetings “start with the hard data” [8]. The authors consider this “hard data” to include iteration events, collected metrics, and completed features or user stories. They point out that while it “may seem silly to gather data for an iteration that lasted a week or two”, being absent for a single day of a week-long iteration already results in missing 20% of events. As such, reflecting on the completed iteration through the lens of project data can ensure a more complete overview for all team members. Furthermore, even when nothing was missed through absence, perceptions of iteration events vary between observers and different people exhibit different perspectives and understandings regarding the same occurrences [8]. Lastly, by focusing on project data, in addition to the “soft data” usually employed, teams can optimize their Retrospective meetings. The roles in teams tasked with facilitating Retrospectives are able to prepare the inputs for meeting activities beforehand, without relying on the presence of others. Team members are able to focus their attention on interpreting data instead of trying to remember the details of the last iteration. The time gained by reviewing, e.g. an already existing list of user stories rather than having to reconstruct it collaboratively, frees up more time for the actual Retrospective work of reflecting on process improvements using Retrospective activities.

## 5 Towards Data-informed Retrospective Activities

The activities that we identified, depending on interpretation and context, as having a *possible* connection to project data, i.e. depending on concrete execution in teams, are “Agile Self-Assessment”, “Quartering - Identify boring stories”, “3 for 1”, “Value Stream Mapping” and “Snow Mountain”, see Table 3. In the

following paragraphs, we discuss these activities and their relations with software project data in detail.

*Agile Self-Assessment* involves assessments of team members regarding the state of their own team, based on a checklist of items. Depending on the employed checklist, these assessments might involve quantifiable measurements, e.g. “time from pushing code changes until feedback from a test is received”<sup>6</sup> or can rely on entirely team members’ perceptions, e.g. “the team delivers what the business needs most”<sup>7</sup>. By switching to a checklist featuring measurements based on Agile practice usage and project data [35], this activity can be modified to present a more objective, data-based process view.

*Quartering - Identify boring stories* assumes a collection of “everything a team did” in the last iteration. The activity’s description does not mention how this overview is achieved or how the data points are collected. By brainstorming all their activities, this overview can be collaboratively reconstructed from the memories of participants. Relying on project data could significantly speed up this (error-prone) method of data collection. Dashboards featuring all interactions with the version control system by team members, e.g. using GitHub<sup>8</sup>, can present activity audits with minimal overhead, leaving more time in Retrospectives for discussion. Furthermore, the goal of quartering is to identify boring stories. While the “boringness” of a story/work item is, by definition, in the eye of the beholder, data from project issue trackers could provide an additional level of analysis: Stories with no discussion that were closed rapidly, needing only a few commits by a single author, might be ideal candidates to be discussed for this Retrospective activity.

*3 for 1* combines, as the name suggests, the assessments of meeting participants regarding three categories: iteration results, team communication, and mood. Team members are asked to mark their spot in a coordinate plane using the axes “satisfaction with iteration result” and “number of times we coordinated” with an emoticon representing their mood. While satisfaction with iteration results and mood are hard to gauge using project data, the frequency of communication within a team can be extracted from the team’s employed communication tools. As more communication moves to digital tools, such as chat or ticket systems, the wealth of information in this domain is steadily increasing [36]. If a digital tool is used, the number of contacts and touch points between team members can be counted and quantified. The input for one axis of the *3 for 1* activity can therefore be automated or augmented with project data analyses. Furthermore, variations of this exercise include varying the employed categories, such as replacing communication frequency with the frequency of pair programming [37] in the team. Relying more heavily on project data analyses for this activity can simplify both data collection and substitution of employed categories.

<sup>6</sup> <https://finding-marbles.com/2011/09/30/assess-your-agile-engineering-practices/>

<sup>7</sup> <https://www.crisp.se/gratis-material-och-guider/scrum-checklist>

<sup>8</sup> <https://github.blog/changelog/2018-08-24-profile-activity-overview/>

*Value Stream Mapping* attempts to create a *value stream map* (VSM) [38, 39] of a team’s process based on the perspective of a single user story. While the details of the story might still be in participants’ memories, gathering additional data, based on project artifacts, can provide additional context to improve the map’s accuracy. One of the main goals of a VSM is to identify delays, choke points, and bottlenecks in the process. In a software development process, these are measurable using project data, e.g. by calculating the time it took from pushing code for a story until the code was reviewed or by assessing its *lead time* [40]. A more complete VSM can be generated by relying on these metrics, leading to improved subsequent analysis and improvement steps in a team.

*Snow Mountain* uses the shape of the Scrum Burndown chart regarding a problematic iteration to draw an image that is used as a reflection prompt. Using the metaphor of a snowy mountain ridge, meeting participants describe their perceptions of the iteration with kids sledging down the slopes. The Burndown chart is a measurement tool for planning and monitoring of progress in Scrum teams [41]. They are based on the amount of work left to do versus remaining time during an iteration. Depending on the team, the amount of outstanding work can be represented by time units, story points or other effort measures (e.g. “gummy bear” [42]). If sophisticated project management software is used by the team and work items are entered into it with the required level of detail, burndown charts can be created and extracted from the project data<sup>9</sup>. These digital images can then be printed or otherwise transformed into the snowy mountains required for the activity, without expending team members’ time in creating them.

## 6 Conclusion

We present a review and analysis of current Retrospective activities, with a focus on the *gather data* meeting phase. We discuss the role of software project data, i.e. development artifacts produced by developers in their day-to-day work, within existing Retrospective meeting structures. This type of data has previously been identified in the literature as an extremely valuable source of insight and actionable information [34, 18]. However, we show that the vast majority, i.e. 86%, of activities explicitly proposed for the *gather data* phase in a popular Retrospective agenda collection [25], lack explicit connections to this software project data. Of these data-gathering activities, many share a similar process of collecting participant perceptions and improvement ideas through structured prompts in the general form of *start, stop, continue*. Most current Retrospective activities rely on the perceptions of meeting participants as their sole inputs. However, software project data, in particular requirements information or insights from version control systems, show promise as additional data sources for Retrospective techniques. Integrating the principles of data-driven decision-

<sup>9</sup> <https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-burndown-chart/>

making, based on project data, into Agile processes enables “evidence-based decision making” [38] in Retrospective meetings.

These concepts are not foreign (or new) to Agile methods but seem to have fallen by the wayside recently. The Scrum Guide states, “Scrum is founded on empirical process control theory [...] knowledge comes from experience and making decisions based on what is known. [...]” [1]. We argue that these concepts are still important, yet underused, in current implementations of Agile methods in general and Retrospectives in particular.

We identify four meeting activities in the Retromat collection that already explicitly take project data into consideration. Of these, only two are listed for the *gather data* phase of Retrospectives. We then focus on employing software project data in additional activities to augment Retrospective meetings, decreasing manual efforts by Agile development teams, and process facilitators.

We propose modifications to five other activities, which are suited to take advantage of the process knowledge contained within project data. These proposals present initial steps towards more evidence-based, data-informed decision making by participants of Retrospectives.

## References

- [1] Ken Schwaber and Jeff Sutherland. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game*. Tech. rep. scrumguides.org, 2017, p. 19.
- [2] Digital.ai (formerly CollabNet VersionOne). *14th Annual State of Agile Report*. Tech. rep. 2020, p. 19.
- [3] Scrum Alliance. *State of Scrum 2017-2018: Scaling and Agile Transformation*. Tech. rep. 2018, p. 36.
- [4] Christoph Matthies and Franziska Dobrigkeit. “Towards Empirically Validated Remedies for Scrum Retrospective Headaches”. In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*. 2020. ISBN: 978-0-9981331-3-3. DOI: 10.24251/HICSS.2020.762.
- [5] Yanti Andriyani, Rashina Hoda, and Robert Amor. “Reflection in Agile Retrospectives”. In: *Lecture Notes in Business Information Processing*. Vol. 283. 2017, pp. 3–19. ISBN: 9783319576329. DOI: 10.1007/978-3-319-57633-6\_1.
- [6] Torgeir Dingsøy et al. “Learning in the Large - An Exploratory Study of Retrospectives in Large-Scale Agile Development”. In: *International Conference on Agile Software Development*. Springer International Publishing, 2018, pp. 191–198. ISBN: 978-3-319-91602-6. DOI: 10.1007/978-3-319-91602-6\_13.
- [7] Henrik Kniberg. *Scrum and XP From the Trenches*. 2nd. C4Media, 2015, p. 183. ISBN: 9781430322641.
- [8] Derby Esther and Diana Larsen. *Agile retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, 2006, p. 200. ISBN: 0-9776166-4-9.

- [9] Christoph Matthies, Thomas Kowark, and Matthias Uflacker. “Teaching Agile the Agile Way — Employing Self-Organizing Teams in a University Software Engineering Course”. In: *American Society for Engineering Education (ASEE) International Forum*. ASEE, 2016.
- [10] Adam Przybyłek and Dagmara Kotecka. “Making agile retrospectives more awesome”. In: *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems*. Vol. 11. 2017, pp. 1211–1216. ISBN: 9788394625375. DOI: 10.15439/2017F423.
- [11] Paulo Caroli and Taina Caetano. *Fun Retrospectives-Activities and ideas for making agile retrospectives more engaging*. Leanpub.com, 2016.
- [12] Christoph Matthies, Franziska Dobrigkeit, and Guenter Hesse. “Mining for Process Improvements: Analyzing Software Repositories in Agile Retrospectives”. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. ACM, 2020, pp. 189–190. ISBN: 9781450379632. DOI: 10.1145/3387940.3392168.
- [13] Anna Zaitsev, Uri Gal, and Barney Tan. “Coordination artifacts in Agile Software Development”. In: *Information and Organization* 30.2 (May 2020), p. 100288. ISSN: 14717727. DOI: 10.1016/j.infoandorg.2020.100288.
- [14] Rebekka Wohlrab. “Living Boundary Objects to Support Agile Inter-Team Coordination at Scale”. PhD thesis. 2020. ISBN: 978-91-7905-269-0.
- [15] Annie T T Ying, James L Wright, and Steven Abrams. “Source code that talks: an exploration of Eclipse task comments and their implication to repository mining”. In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), p. 1. ISSN: 0163-5948. DOI: 10.1145/1082983.1083152.
- [16] Christoph Matthies et al. “ScrumLint: Identifying Violations of Agile Practices Using Development Artifacts”. In: *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, 2016, pp. 40–43. ISBN: 9781450341554. DOI: 10.1145/2897586.2897602.
- [17] Cleidson de Souza, Jon Froehlich, and Paul Dourish. “Seeking the Source: Software Source Code as a Social and Technical Artifact”. In: *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*. ACM Press, 2005, p. 197. ISBN: 1595932232. DOI: 10.1145/1099203.1099239.
- [18] Jin Guo et al. “Cold-start software analytics”. In: *Proceedings of the 13th International Workshop on Mining Software Repositories*. ACM Press, 2016, pp. 142–153. ISBN: 9781450341868. DOI: 10.1145/2901739.2901740.
- [19] Christoph Matthies and Guenter Hesse. “Towards using Data to Inform Decisions in Agile Software Development: Views of Available Data”. In: *Proceedings of the 14th International Conference on Software Technologies*. SciTePress, 2019, pp. 552–559. ISBN: 978-989-758-379-7. DOI: 10.5220/0007967905520559.
- [20] Norman L Kerth. “The ritual of retrospectives: how to maximize group learning by understanding past projects”. In: *Software Testing & Quality Engineering* 2.5 (2000), pp. 53–57.

- [21] Luke Hohmann. *Innovation Games: Creating Breakthrough Products through Collaborative Play*. Addison-Wesley, 2006, p. 176. ISBN: 9780132702225.
- [22] Patrick Kua. *The Retrospective Handbook: A guide for agile teams*. Leanpub.com, 2013. ISBN: 978-1480247871.
- [23] Luis Gonçalves and Ben Linders. *Getting Value out of Agile Retrospectives - A Toolbox of Retrospective Exercises*. Leanpub.com, 2014, p. 71. ISBN: 9781304789624.
- [24] Alexey Krivitsky. *Agile Retrospective Kickstarter*. Leanpub.com, 2015.
- [25] Corinna Baldauf. *Retromat - Run great agile retrospectives!* Leanpub.com, 2018, p. 239.
- [26] Miloš Jovanović et al. “Agile retrospective games for different team development phases”. In: *Journal of Universal Computer Science* 22.12 (2016), pp. 1489–1508. DOI: 10.3217/jucs-022-12-1489.
- [27] Marc Loeffler. *Improving Agile Retrospectives: Helping Teams Become More Efficient*. Addison-Wesley Professional, 2017, p. 270. ISBN: 978-0134678344.
- [28] Sarah Beecham et al. “Making Software Engineering Research Relevant”. In: *Computer* 47.4 (2014), pp. 80–83. ISSN: 0018-9162. DOI: 10.1109/MC.2014.92.
- [29] Chris Northwood. “Planning Your Work”. In: *The Full Stack Developer*. Apress, 2018, pp. 11–46. ISBN: 978-1-4842-4152-3. DOI: 10.1007/978-1-4842-4152-3.2.
- [30] Alf Magnus Stålesen and Bjørn Dølvik. “Agile Retrospectives: An Empirical Study of Characteristics and Organizational Learning”. Master Thesis. Norwegian University of Science and Technology, 2015.
- [31] D. Méndez Fernández et al. “Artefacts in Software Engineering: What are they after all?” In: (2018). arXiv: 1806.00098.
- [32] Sonja Dimitrijević, Jelena Jovanović, and Vladan Devedžić. “A comparative study of software tools for user story management”. In: *Information and Software Technology* 57.1 (Jan. 2015), pp. 352–368. ISSN: 09505849. DOI: 10.1016/j.infsof.2014.05.012.
- [33] Najam Nazar, Yan Hu, and He Jiang. “Summarizing Software Artifacts: A Literature Review”. In: *Journal of Computer Science and Technology* 31.5 (Sept. 2016), pp. 883–909. ISSN: 1000-9000. DOI: 10.1007/s11390-016-1671-1.
- [34] Marco Ortu et al. “The JIRA Repository Dataset”. In: *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM Press, 2015, pp. 1–4. ISBN: 9781450337151. DOI: 10.1145/2810146.2810147.
- [35] Christoph Matthies et al. “Agile metrics for a university software engineering course”. In: *2016 IEEE Frontiers in Education Conference*. IEEE, 2016, pp. 1–5. ISBN: 978-1-5090-1790-4. DOI: 10.1109/FIE.2016.7757684.
- [36] Viktoria Stray and Nils Brede Moe. “Understanding coordination in global software engineering: A mixed-methods study on the use of meetings and Slack”. In: *Journal of Systems and Software* 170 (Dec. 2020), p. 110717. ISSN: 01641212. DOI: 10.1016/j.jss.2020.110717.

- [37] Henrik Kniberg. *Scrum and XP from the Trenches*. C4Media, 2007, pp. 1–105. ISBN: 1430322640. DOI: 978-1-4303-2264-1.
- [38] Brian Fitzgerald, Mariusz Musiał, and Klaas-Jan Stol. “Evidence-based decision making in lean software project management”. In: *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*. New York, New York, USA: ACM Press, 2014, pp. 93–102. ISBN: 9781450327688. DOI: 10.1145/2591062.2591190.
- [39] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. “Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies”. In: *Information and Software Technology* 62.1 (June 2015), pp. 143–163. ISSN: 09505849. DOI: 10.1016/j.infsof.2015.02.005.
- [40] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. “Kanban in software development: A systematic literature review”. In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, Sept. 2013, pp. 9–16. ISBN: 978-0-7695-5091-6. DOI: 10.1109/SEAA.2013.28.
- [41] Ezequiel Scott and Dietmar Pfahl. “Exploring the Individual Project Progress of Scrum Software Developers”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10611 LNCS. 2017, pp. 341–348. ISBN: 9783319699257. DOI: 10.1007/978-3-319-69926-4\_24.
- [42] Bertrand Meyer. *Agile! The Good, the Hype and the Ugly*. 1st ed. Cham: Springer International Publishing, 2014, p. 170. ISBN: 978-3-319-05154-3. DOI: 10.1007/978-3-319-05155-0.

## Appendix

**Table 4.** List of activities extracted from the Retromat repository [25] for the *gather data* phase of Retrospectives, as of Oct. 2020.

#	Name & Activity Tagline
4	<i>Timeline</i> : Write down significant events and order them chronologically
5	<i>Analyze Stories</i> : Walk through a team’s stories and look for possible improvements
6	<i>Like to like</i> : Match quality cards to their own Start-Stop-Continue-proposals
7	<i>Mad Sad Glad</i> : Collect events of feeling mad, sad, or glad and find the sources
19	<i>Speedboat/Sailboat</i> : Analyze what forces push you forward and pull you back
33	<i>Proud &amp; Sorry</i> : What are team members proud or sorry about?
35	<i>Agile Self-Assessment</i> : Assess where you are standing with a checklist
47	<i>Empty the Mailbox</i> : Look at notes collected during the iteration
51	<i>Lean Coffee</i> : Use the Lean Coffee format for a focused discussion of the top topics
54	<i>Story Oscars</i> : The team nominates stories for awards and reflects on the winners
62	<i>Expectations</i> : What can others expect of you? What can you expect of them?
64	<i>Quartering</i> : Categorize stories in 2 dimensions to identify boring ones
65	<i>Appreciative Inquiry</i> : Lift everyone’s spirit with positive questions
75	<i>Writing the Unspeakable</i> : Write down what you can never ever say out loud
78	<i>4 Ls</i> : Explore what people loved, learned, lacked and longed for individually
79	<i>Value Stream Mapping</i> : Draw a value stream map of your iteration process
80	<i>Repeat &amp; Avoid</i> : Brainstorm what to repeat and what behaviours to avoid
86	<i>Lines of Communication</i> : Visualize information flows in, out and around the team
87	<i>Meeting Satisfaction Histogram</i> : Create a histogram on how well ritual meetings went during the iteration
89	<i>Retro Wedding</i> : Collect examples for something old, new, borrowed and blue
93	<i>Tell a Story with Shaping Words</i> : Each participant tells a story about the last iteration that contains certain words
97	<i>#tweetmysprint</i> : Produce the team’s twitter timeline for the iteration
98	<i>Laundry Day</i> : Which things are clear & feel good and which feel vague & implicit?
110	<i>Movie Critic</i> : Imagine your last iteration was a movie and write a review about it
116	<i>Genie in a Bottle</i> : Playfully explore unmet needs
119	<i>Hit the Headlines</i> : Which sprint events were newsworthy?
121	<i>The Good, the Bad, and the Ugly</i> : Collect what team members perceived as good, bad and non-optimal
123	<i>Find your Focus Principle</i> : Discuss the 12 agile principles & pick one to work on
126	<i>I like, I wish</i> : Give positive, as well as non-threatening, constructive feedback
127	<i>Delay Display</i> : What’s the current delay? And where are we going again?
128	<i>Learning Wish List</i> : Create a list of learning objectives for the team
133	<i>Tell me something I don’t know</i> : Reveal hidden knowledge with a game show
135	<i>Avoid Waste</i> : Tackle the 7 Wastes of Software Development
137	<i>Dare, Care, Share</i> : Collect topics in three categories: ‘Dare’, ‘Care’ and ‘Share’
139	<i>Room Service</i> : Take a look at the team room: Does it help or hinder?