

Fixations in Agile Software Development Teams

Klara Borowa, Sebastian Kamoda, Piotr Ogrodnik, Andrzej Zalewski *

Abstract. Cognitive biases influence every human being, including the individuals that take part in the software development process. Fixation is a cognitive bias that occurs when one focuses too much on certain items, events, obstacles or activities. In this study, we examine whether agile team members fixate on any particular agile practices. Through a set of semi-structured interviews, we investigated the source of these fixations, their consequences, and then propose possible countermeasures. We found that practitioners tend to fixate on practices that give them a sense of being in control over the project (such as meetings or Scrum events), while neglecting the Agile Principles of self-organising teams and working at a sustainable pace. This resulted in a series of problems, such as futile attempts to control team members, oversharing information with the client, meetings becoming a form of interrogation, and others.

Keywords: Agile software development, Software Development, Fixation, Cognitive Bias

1. Introduction

It has been more than 20 years since the original Agile Manifesto was published [2]. Since then, practitioners' approach to agile software development has changed significantly [1]. Nowadays, teams usually employ certain specific agile methodologies, rather than the bare Manifesto itself [13]. Additionally, agile methodologies are often tailored to suit the team's needs [5].

The values and principles of the original Manifesto are still relevant to the problems and the challenges that most teams face [13]. However, teams may start adopting certain aspects of agile methodologies, without putting sufficient further thought as to whether their actions are fully rational [14].

*Warsaw University of Technology, Institute of Control and Computation Engineering, Warsaw, Poland, klara.borowa@pw.edu.pl

So far, there have been numerous studies on the human aspects [10] affecting agile software development. The challenges to the adoption of agile methodologies are also well recognised [8].

However, the idea that cognitive biases may influence agile teams has only been touched upon in two studies so far [18, 11]. Therefore, a significant knowledge gap exists as cognitive biases significantly influence software development [16].

In this work, we investigate the impact of fixation [15] on agile teams. Fixation, is a cognitive bias manifesting in a disproportionate, irrational focus on certain aspects of reality, objects, actions, problems, and the like [17]. To delve further into this topic, we defined the following research questions:

- RQ1: Do the agile team members fixate on any particular agile practices?
- RQ2: What kind of problems do these fixations cause?

Our study resulted in a set of agile practices that practitioners fixated on being identified. The way in which these practices were implemented was not necessarily in line with the agile movement's values and principles. Unconsciously, practitioners were driven towards focusing on actions that gave them a sense of control over the project.

We also found the agile principles of self-organising teams and working at a sustainable pace were neglected due to these fixations. In addition, we identified a set of problems that resulted from these fixations. Finally, we proposed a set of debiasing actions that could help to mitigate the negative effects of fixation.

2. Related Work

The basic concept of cognitive biases revolves around the dual nature of the human mind, namely, fast and intuitive (called System 1) and slow and logical (System 2) [12]. Cognitive biases influence a decision that is solely based on System 1 logic without being corrected by System 2.

The idea that cognitive biases may impact software development is not new. This observation was already made by Stacy and Macmillan in the nineties [23]. Since then, researchers have discovered numerous influences of cognitive biases [16] on various aspects of software development. Examples can be found in architecture decision-making [24], requirements [26], testing [4] or time estimation [9].

Cognitive biases affect any individual that takes part in the development process – an example being developers [6] and architects [27].

In the context of agile software development, so far, only the egocentric bias [18] and the combination of anchoring and confirmation biases impacting pair programming [11] have been studied, to the best of the authors' knowledge. Ramin [18] shows that the egocentric bias is more likely to appear in a new agile team that has no experience in working together. This means that new team members overestimate their impact on the overall project. Jain et al. [11] show that, in the setting of pair programming, experts are less likely to search documentation due to the influence of the confirmation bias, but novices are more likely to anchor on initial solutions.

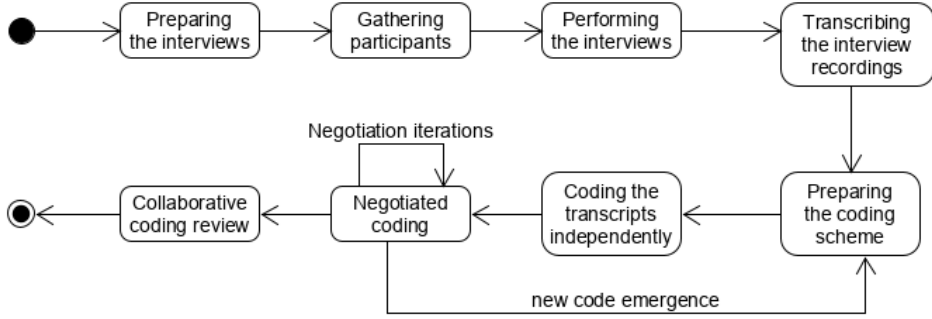


Figure 1. Research process

Fixation is a cognitive bias that can be defined as the “disproportionate focus on one aspect of an event, object or situation, especially self-imposed or imaginary obstacles” [17]. So far, it has been observed in the context of design decisions, where it has been proven to hinder designers’ creativity [15].

Through our literature search on cognitive biases, we found no work on the influence of fixation on agile software development. The main purpose of this case study is to explore the phenomenon of fixation and its impact on agile software development teams.

3. Research method

The research method follows the case study research guidelines by Runeson et al. [19]. Data was gathered from a series of semi-structured interviews with 13 software practitioners whose teams used agile methodologies in their everyday work. The interviews were transcribed and coded. The coding process was started by creating a preliminary set of codes, then two authors coded the transcripts independently, and following that they discussed the differences between their coding during a negotiation until reaching a full consensus. Since new codes could emerge during the analysis, it was necessary to update the coding scheme and code the transcripts with the new codes in additional analysis iterations. The negotiations themselves were considered an iterative process since the coders sometimes noticed an inconsistency during negotiations that required revisiting previously coded transcripts. Ultimately, another author reviewed the coding and discussed possible errors in the coding with the initial coders. The whole process is illustrated in Figure 1.

To answer the research questions, we focused on investigating the following: (1) Points of fixation: agile practices on which our participants or their colleagues focused on irrationally; (2) Points of neglect: agile values, principles or practices that were neglected due to fixations. (3) Problems caused by the first two areas.

3.1. Interview procedure

Table 1. Interview plan

No.	Question	Question objective
Consent	[Ask for consent to record the interview and use the data for research purposes. Answer any questions from the participant]	Obtaining consent
Intro	What is your position? How many years have you worked in an agile team? How big were the companies you worked in? How were you introduced to agile software development?	Obtaining demographic data
1	What does 'Agile' mean to you?	RQ1
2	What kinds of agile teams have you worked with? Number of people/used agile methodologies/practices?	Participant data, RQ1
3	What elements make a team agile?	RQ1
4	Has your perception of agile software development changed since you first started working in an agile team? How?	RQ1
5	Have you ever encountered a situation when the pressure to 'do something the Agile way' was seemingly nonsensical? Please give us some specific examples.	RQ2
6	What were the biggest advantages of using agile software development in your projects?	RQ1
7	What were the biggest disadvantages of using agile software development in your projects?	RQ2

The second author interviewed three participants, the third author interviewed eight participants, and the second and third authors jointly interviewed two participants. We designed a set of interview questions in order to obtain data relevant to our research questions and to obtain demographic data about our participants, such as: age, years of experience in software development, years of experience with agile software development, position, team size, agile methodologies used, company size and company sector. The interview plan is described in detail in Table 1. As with any semi-structured interview, in order not to disrupt the flow of the conversation, it was not strictly limited by the questions. Participants were encouraged to give additional insights and the interviewer sometimes asked clarifying questions.

The participants were not informed that the goal of this study was to explore the influences of cognitive biases on agile software development, though they did know we were researching problems connected with agile teams. The participants did not have a chance to discuss their experiences with each other prior to the interview.

3.2. Participants

Table 2. Participants

No.	Position	Experience (yrs)	Experience with Agile (yrs)	Agile methodology	Team size	Company size	Sector
1	Consultant/ Developer	3	2	Scrum	9	250	Software House
2	Software Developer	2	1	SAFe	12	500	Finance
3	Software Developer	1	1	SAFe	6	200	IT Services
4	Architect	4.5	4	Scrum	9-10	20 000	Retail
5	Consultant/ Developer	5.5	2.5	Scrum	9	250	Software House
6	Technical Consultant	1	0.9	Custom tailored approach	9	4 000	Software House
7	Software Developer	2	2	No specific methodology	9	20 000 (worldwide)	Geoinformatics
8	Data Engineer	2	0.3	No specific methodology	8	950	Data Analytics
9	Architect	20	10	Scrum	8-9	150 000 (worldwide)	Enterprise software, Fuel and Energy
10	Consultant/ Developer	4	4	Scrum	8-9	100	Software House
11	Software Developer	3	3	No specific methodology	6	300	Fintech
12	CTO	20	10	Scrum	9	350	Banking
13	Software Developer	2.5	1	Scrum	8	284 000 (worldwide)	Consulting

The study participants were recruited from the authors' private network. Basic information about the participants, including their years of experience (working in software development, and using agile software development methods), company size and domain, are presented in Table 2. To obtain high quality data, we gathered a diverse set of participants – from small companies to international giants, from novices to very experienced practitioners.

3.3. Analysis procedure

All the interviews were recorded and subsequently transcribed. The transcripts were analysed through coding and memoing. We created an initial set of codes based on the research questions, thus using the hypothesis coding method [20]. Fragments of the transcripts that fit the code definition were marked accordingly.

The coding of each transcript was firstly performed separately by two of the authors. Subsequently, using the negotiated agreement approach [7], the authors met and discussed the coding in order to reach a full consensus.

Table 3. Coding table

Code category	Code	Definition	Interviews with occurrences	Total occurrences
Introduction by work	I-W	The participant was introduced to agile software development for the first time in his workplace.	11	11
Introduction by theory	I-T	The participant was first introduced to agile software development through a theoretical course - during their college education or on other courses.	2	2
Performed debiasing	Auto-debiasing	Information about our participants debiasing themselves by studying agile software development further. By self-study or taking part in additional courses.	2	2
Fixation	F	A single occurrence fixation on any specific agile practice by the participant.	9	24
Misunderstood Agile	Bad Agile	The participant's or his colleague's mistaken belief about agile software development, or particular agile methodologies.	7	12
Fixation awareness	AW-F	Information about fixations resulting in bad practices by the participant's colleagues.	4	8
Agile Neglect	NAC	The participant or his colleague neglected certain aspects that are key to agile software development or the particular agile methodology that they were supposed to be using.	4	8
Performed debiasing	Debiasing	Information about interventions that were suggested by our interviewees, or performed by their colleagues.	2	2
Building silos	SILO	When the team was divided into silos (smaller, tightly-knit groups).	3	5

During the coding procedure, the need to add new codes emerged, in order to perform a more detailed analysis. In this way, the authors updated the coding scheme to add those codes and then performed another round of coding in order to obtain additional data. Table 3 contains the final coding scheme with the codes' descriptions.

All the coded transcripts were then reviewed by an additional author, one that had not performed the previous steps of the coding. Any ambiguities that were found had to be resolved during a final three-person negotiation round.

Ultimately, the occurrences of the codes were counted (see Table 3) and the information related to them was aggregated. This data was then analysed and discussed by the authors, in order to find points of fixation, points of neglect, and the resulting problems.

4. Results

In this section, we describe in detail the results of our enquiry. In Section 4.1 we present the overall codes that were found. In Section 4.2 we describe the participants' theoretical knowledge on agile and its influence on fixations. Finally, we demonstrate practices on which practitioners fixate (Section 4.3), which Agile Principles

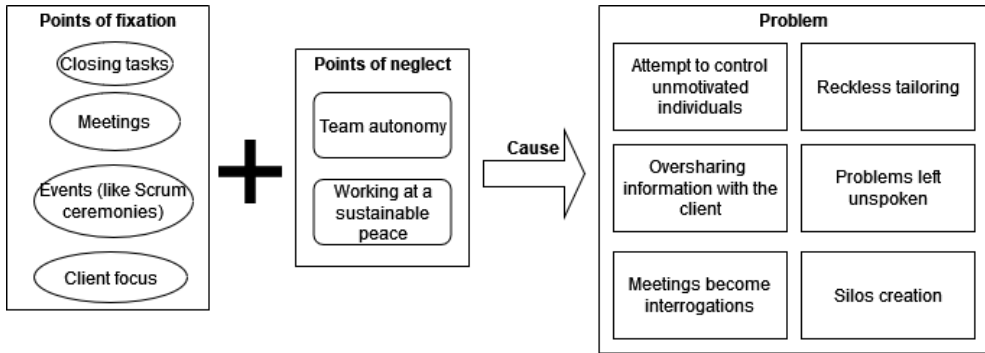


Figure 2. Fixations influence on agile teams

they neglect due to fixation (Section 4.4) and what problems these fixations cause (Section 4.5). The influence of fixations is presented overall in Figure 2.

4.1. Code occurrences

The occurrences of the codes in the transcripts are summed up in Table 3. Codes “I-W” and “I-T” presents data on how our participants were first introduced to agile software development, whether at their workplace or during their education. Additionally, code “Auto-debiasing” provides information about how many participants, after their first experiences, continued to expand their knowledge about agile software development - and which could possibly provide a debiasing effect.

All other codes could emerge multiple times during the same interview, since our participants could have described many earlier experiences, and any number of their own observations.

4.2. Fixations and theoretical knowledge of agile

The majority of the participants were introduced to agile software development when assigned to a project in their workplace. They were expected to learn about the agile methodology used by their team during their work ‘as they go’. Only one of these 11 participants decided to educate themselves (by reading a book about the specific agile methodology and attending courses). The majority did not have a theoretical grasp over the basic concepts of the Agile Manifesto and the Principles. This kind of shallow knowledge may make it more likely for practitioners to misunderstand the fundamental values of agile software development.

Participants with no theoretical foundation to their knowledge about agile software development provided overall 15 cases of their own fixations, the median being one fixation occurrence per interview. Participants that possessed theoretical knowledge, gave five cases of their fixations, the median being two occurrences.

Additionally, both groups provided only four examples overall of an awareness of others' fixations, with the median being none per interview.

Considering the limited number of participants and the minimal code occurrence differences between both groups, we could not conclude whether theoretical knowledge about Agile actually had a debiasing effect. Our study does not provide any evidence that would support such a debiasing effect, but it does not completely exclude the possibility either.

However, since the awareness of others' fixations was significantly smaller than the participants own fixations, it is possible to draw the assumption that developers are less likely to notice their own fixations, than the fixations of their colleagues.

4.3. Fixation points

By analysing the interviews, we found a set of practices that were common fixation points in agile teams:

4.3.1. Closing tasks

At first glance, this is the expected activity that gives a positive impression of the team's work. The mere action of closing a task can be motivating. As Participant No. 2 stated, "It was a weird pleasure to close a task, especially one that had been hanging around for quite a while".

However, what looks good on paper may not be as valuable in practice. In some teams, tasks were often not defined or assigned collectively by the team, but by a subset of the team's members, manipulating the information about tasks to "look good on paper". Participant No. 3 described such a situation, "Basically, regarding the work tracking in terms of closing the tasks in correct sprints, there are a couple of senior developers who were redistributing them and planning them across the sprint, they also had a 'magic spreadsheet' that keeps track of what was finished or not, making some adjustments as necessary. And that was the source of truth for all the work on the project."

In addition, in such a setting, the team members with the authority to determine the 'truth' about the current state of the project backlog created a silo, making it harder for other team members to communicate their problems or doubts regarding the current state of the project.

Furthermore, this drive towards having 'good numbers on paper' did not take into account the quality of the software produced. The only crucial matter was maximising the measured team's performance.

4.3.2. Attending meetings

An excessive emphasis on meetings, is another common fixation point. Participants often perceived their presence during meetings as unnecessary. Meetings often had

no agenda and the topics discussed were of no importance to their work.

Participant No. 1 stated, “Yes. I could say that I was being held [in meetings]. As I mentioned, there were issues that could have been handled by a smaller group of people, but instead the whole team was called in. There were those huge meetings – a dozen or so people – during which most of the people were just sitting there and you could say they were wasting their time there.”

One of the reasons behind meetings taking too much time may be the excessively high number of attendees during meetings. Many agile methodologies are supposed to be applied by small teams. For example, the Scrum Guide [21] suggests that a development team should consist of 10 or fewer members. Despite our participants’ teams being smaller than 10 members, they reported a significantly larger number of attendees during their recurrent meetings.

Another problem related to the meetings was that the organiser (commonly the Scrum Master or Project Owner) did not provide an agenda for the meeting set beforehand. Often no agenda existed, because meetings were simply a thing that “were supposed to happen” every day or every week.

4.3.3. Methodology specific events

Methodology specific events, such as the Scrum Retrospective, were practices on which agile teams were frequently fixated.

Two participants who reportedly used Scrum, had counterproductive Sprint Retrospective meetings. The teams would mention the problems that occurred during the last Sprint, but they never made any improvement plans.

Another example came from Participant No. 13, who stated that in their company, several teams had to take part in the backlog refinement. However, the items in the backlog usually were connected with the work of only one team, making the attendance of the others an unnecessary ritual.

4.3.4. Client focus

Client focus was a value that often turned into simply meeting all of the client’s whims. Although delivering value to the client is a priority in agile software development [3], in many cases the blind acceptance of the client’s requirements may be in conflict with the quality of the software.

However, some of our participants had to work in settings where the client’s requests were always final. This not only negatively impacted the quality of the solution, but also undermined the team’s motivation. As Participant No. 5 stated, “It went directly against the quality of the work delivered, how much it had to be revisited or reworked or changed.”

To achieve the best possible quality, the client must trust the development team. Sometimes a change that is requested by the client may have a negative impact on the end product – for example, it may require breaking key architectural constraints.

4.4. Neglected Agile Principles

We identified a subset of the original Agile Principles [3] that were neglected due to fixations: self-organising teams and working at a sustainable pace.

4.4.1. Teams and team members deprived of their autonomy

We found that it was common for the decision-making process to be monopolised by one or two team members. This usually concerned a manager, the Scrum Master, Product Owner, or the client. Developers were forced to attend meetings, during which they were not given the chance to actively take part in the decision-making process.

Participant No. 1 provided us with the following example, “During the Sprint Planning, we were supposed to choose tasks that we were going to deliver during the Sprint. But this was not entirely our [the developers’] choice to make. Ultimately, the tasks were assigned according to what the project manager and scrum master decided. We were not allowed to tell them, for example, that we will not be able to deliver this Sprint, because we were already at 95% capacity, and that unplanned ad hoc tasks and meetings always appear during the week. Despite knowing that we cannot deliver, we had to declare that we would take this work upon ourselves.”

Additionally, participants reported that, due to the excessive time that they have to spend in meetings, they often lacked time to informally contact each other for any help or knowledge sharing. This made the communication between team members overly formal and less effective.

4.4.2. A sustainable pace of work was often foregone

When problems arose, it was not unusual to abandon all previous practices (such as meetings or Scrum events) during the drive to deliver before deadlines. Participant No. 8 mentioned that, “Sometimes the end of the Sprint is not enjoyable. What I mean is that they are pushing you from every direction to close your tasks, even though it happens that the work is seriously underestimated, meaning that on some Fridays (or when a Sprint is coming to an end) we have to work overtime.”

Teams were sometimes forced to work at an absolutely unrealistic pace. Participant No. 4 told us a story that illustrated an extreme case of this problem. It was the case of a team that was expected to work in ‘day-long sprints’, that included the deployment of new features to production on every single day.

4.5. Resulting Problems

By analysing the interviews, we extracted a set of common problems and misconceptions that plagued agile teams, due to their fixations.

1. **Attempt to control individuals through certain meetings/events.** *Causes: Fixation on meetings and events combined with the neglect of the principles of team autonomy and working at a sustainable pace.*
Through these attempts, a team member of higher authority attempted to push the team and force them to perform better, while ignoring valuable information from the team members.
2. **Oversharing information with the client.** *Causes: Fixating on customer focused development and meetings while neglecting the principle of team autonomy.*
While client satisfaction is a key Agile principle [3], this does not have to be achieved by informing the client about absolutely everything. Since the client does not necessarily have in-depth technical knowledge in every aspect, there is no benefit from involving him in certain decisions. However, due to fixation, some teams involved the client in every aspect of their work, making him the main decision-maker in absolutely all aspects.
3. **Meetings as a form of interrogation.** *Cause: Fixating on meetings and closing tasks, while ignoring the principles of team autonomy and working at a sustainable pace.* Our participants often felt that, during meetings, they had to prove that they had been actively working on their tasks. If a task was not closed during the last development iteration, they were suspected of not working hard enough. This means that the meetings, instead of giving them tools to solve problems, often felt like a hunt to find a scapegoat. This approach made team members likely to avoid performing less predictable and more challenging tasks, while focusing on smaller easier ones, just to have something to report during the meetings.
4. **Unspoken problems.** *Causes: Fixating on methodology specific events while neglecting the principle of team autonomy.* While the Agile Manifesto stresses the importance of individuals and their interactions [2], we found many cases of unexpressed problems when team autonomy was neglected, in favour of blindly performing methodology specific events. If a problem did not fit into any of the events' agenda, it was left unspoken. Team members, stripped of their sense of agency, kept their observations, opinions and suggestions to themselves.
5. **Reckless agile tailoring.** *Causes: Fixating on closing tasks, meetings or methodology specific events, while neglecting the principle of team autonomy.* While agile tailoring can be performed intentionally, with great benefit to agile teams and organisations [5], we found many cases of unintentional reckless agile tailoring. For example, while many participants stated that their team used Scrum, they gave various examples of actions that were contradicted the Scrum Guide [21]. Instances included having large teams, having the Scrum Master and Product Owner acting as traditional managers, and taking away the developers' ability to influence the Sprint Planning. None of these modifications were performed deliberately or consulted with other team members.

6. **The creation of team silos.** *Causes: Fixating on meetings.* When all information was to be shared at meetings, informal communication between team members became minimal. As a result, teams broke down into small, specialised groups of people that closely collaborated together. However, these groups encountered problems when cooperating with team members that were not part of their silo.

5. Debiasing methods

Based on our findings about fixation points, neglected principles and the resulting problems, as well as our participants' suggestions and experiences, we propose a set of countermeasures that can be employed to improve agile teams' everyday work by limiting the influence of fixations.

1. Educate the parties involved on the basics of agile software development – the Agile Manifesto [2] and Agile Principles [3]. Although this knowledge may not be enough to avoid fixations, this basic knowledge is necessary for any agile team to consciously make informed decisions during their everyday work. It is important to emphasise the importance of the principles of team autonomy and working at a sustainable pace.
2. If a specific methodology is used, or a tailored version thereof, make sure that all the team members know all the implications of that choice.
3. Make decisions about tailoring consciously, after understanding the specific original agile methodology.
4. Let the team self-organise. If certain team members dominate the decision-making, work towards giving other team members a chance to influence the project.
5. Integrate the team by breaking down any team silos.
6. Motivate the team by encouraging their participation in decision-making.
7. Check what kind of periodical meetings and ceremonies the team has. Discuss as a team: Are all of them necessary? How can they be made more useful?
8. Make agendas for meetings and make sure that all the team members understand the goal of the meeting.

6. Threats to validity

When evaluating the validity of this research, the set of its limitations has to be taken into account. Runeson et al. [19] describes four basic types of validity, that have to be assessed.

1. **Construct validity** – how the measures used in the study represent what was intended by the researchers. In this study, we did not inform the participants in advance that we were researching fixations. As such, we simply ascertained whether the real-life examples given by the participants fit the definition of fixation. Because of that, the full understanding of this concept by the participants was unnecessary.
2. **Internal validity** – whether factors not researched could impact the research. We relied on our participants recollections of past events. It is natural for an individual to forget or have faulty memories of the past. Additionally, we did research the impact of other factors than fixation on the observed problems. It may be possible that if taken into account, fixation may not have been the most impactful factor behind them. Future work could include the observation of agile teams during their everyday work, in order to to maximise internal validity.
3. **External validity** – whether the findings can be generalised. To maximise external validity, we attempted to gather a varied group of participants. However, especially taking into account the rather small sample, there is the possibility that, with a different set of individuals, slightly differing or additional findings may have emerged. Additionally, our study did not include participants representing certain agile methodologies, such as Kanban or DSDM. Furthermore, since we did not want to focus on one specific agile methodology, we interviewed only participants with experience as software developers, which is the only role necessary in any agile team. However, this made us omit methodology specific roles such as Scrum Masters or Product Owners. Future quantitative and replication studies, made in a different setting, could further validate our findings.
4. **Reliability validity** – whether the data or analysis depend on the researcher. The interviews and analysis procedure are explained in detail in Section 3. no part of the data gathering and analysis depended solely on one researcher. Interviews were performed by the second and third authors, and the coding was performed, reviewed and analysed by the first, second and third authors.

Finally, the debiasing methods that we proposed in Section 5 have not been validated in an empirical setting and require verification.

7. Discussion

Research on cognitive biases affecting agile teams is scarce (see Section 2). We have not found any prior research on fixations affecting agile teams. This means that our results can only be discussed in the context of previous research about the problems experienced by agile teams.

The fixations that we identified have one common theme – *control*. The developers felt in control when 'closing tasks'. Managers used 'magic spreadsheets' to determine

the ‘truth’ about the state of the project. Scrum Masters forced developers to take part in poorly organised Scrum ceremonies. The client forced the developers to perform tasks that negatively impacted the product’s quality.

Similarly, the neglected Agile Principles were associated with insufficient trust in the team’s capability. It was often preferred to impose control on the team’s activities instead of relying on its self-organisation potential.

Previous work shows that the change from planning-driven methods to agile software development, takes away the feeling of predictability [25] that was associated with precise planning. This change can become a source of fear for practitioners who have grown used to planning-driven methods [8].

The fear of losing control is the subconscious driving force behind the occurrence of disruptive fixations. Practitioners affected by this problem simply cherry-pick a small subset of agile practices that give them the feeling of projects remaining under control, while ignoring core Agile Principles that they are not so comfortable with.

8. Conclusions

Our paper answered the research questions in the following way:

- *RQ1: Do the agile team members fixate on any particular agile practices?*
In Section 4.3 we describe the main fixations that occur in agile teams. A common theme between these factors is ‘control’. Participants and their teams were likely to fixate on performing certain actions that gave them a sense of control over the project (meetings, methodology specific events, managing tasks using ‘magic spreadsheets’, etc.). Additionally, as described in Section 4.4, while focusing on these points of fixation, teams were likely to neglect two basic Agile Principles: self-organising teams and working at a sustainable pace.
- *RQ2: What kind of problems do these fixations cause?*
Team members that have a biased perception of agile practices cannot fully benefit from the use of agile methods. We described a set of problems that result from these fixations in Section 4.5. Notably, many teams operated under the false belief that through certain ceremonies, individuals can be more easily controlled and would work more effectively.

In Section 5 we proposed a set of debiasing actions that can be performed to counter the negative impact of fixations. The most basic one being: education regarding the basic agile values, principles and practices.

Future research should focus on two areas: the further validation of our findings, and establishing effective debiasing treatments. A debiasing treatment through workshops that heighten awareness of cognitive biases may be effective [22].

Our study could also be additionally validated by further empirical research: replication studies, as well as qualitative studies that would make use of data from various sources: such as focus group discussion or shadowing agile teams. Debiasing treatments could be tested in both controlled experiments, as well as in real-life settings.

References

- [1] Babb J. S., Nørbjerg J., Yates D., and Waguespack L. J. The Empire Strikes Back: The end of Agile as we know it? *Selected Papers of the IRIS*, 8(44-59):17, 2017.
- [2] Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R. C., Mellor S., Schwaber K., Sutherland J., and Thomas D. Manifesto for Agile Software Development. <https://agilemanifesto.org/>, 2001. Accessed 2021-07-05.
- [3] Beck K., Beedle M., van Bennekum A., Cockburn A., Cunningham W., Fowler M., Grenning J., Highsmith J., Hunt A., Jeffries R., Kern J., Marick B., Martin R. C., Mellor S., Schwaber K., Sutherland J., and Thomas D. Principles behind the Agile Manifesto. <https://agilemanifesto.org/principles.html>, 2001. Accessed 2021-07-05.
- [4] Çalikli G. and Bener A. B. Influence of confirmation biases of developers on software quality: An empirical study. *Software Quality Journal*, 21(2):377–416, 2013.
- [5] Campanelli A. S. and Parreiras F. S. Agile methods tailoring - A systematic literature review. *Journal of Systems and Software*, 110:85–100, 2015.
- [6] Chattopadhyay S., Nelson N., Au A., Morales N., Sanchez C., Pandita R., and Sarma A. A Tale from the Trenches : Cognitive Biases and Software Development. In *International Conference on Software Engineering (ICSE)*, pages 654–665, 2020.
- [7] Garrison D. R., Cleveland-Innes M., Koole M., and Kappelman J. Revisiting methodological issues in transcript analysis: Negotiated coding and reliability. *Internet and Higher Education*, 9(1):1–8, 2006.
- [8] Gregory P., Barroca L., Sharp H., Deshpande A., and Taylor K. The challenges that challenge: Engaging with agile practitioners’ concerns. *Information and Software Technology*, 77:92–104, 2016.
- [9] Halkjelsvik T. and Jørgensen M. *Time Predictions: Understanding and Avoiding Unrealism in Project Planning and Everyday Life*. Springer Nature, 2018.
- [10] Hoda R., Salleh N., Grundy J., and Tee H. M. Systematic literature reviews in agile software development: A tertiary study. *Information and software technology*, 85:60–70, 2017.
- [11] Jain R., Muro J., and Mohan K. A cognitive perspective on pair programming. In *AMCIS 2006 Proceedings*, 2006.
- [12] Kahneman D. *Thinking, fast and slow*. Macmillan, 2011.

- [13] Kiv S., Heng S., Kolp M., and Wautelet Y. Agile manifesto and practices selection for tailoring software development: A systematic literature review. In *Lecture Notes in Computer Science*, volume 11271, pages 12–30, 2018.
- [14] Kruchten P. Voyage in the agile memplex. *Queue*, 5(5):38–44, 2007.
- [15] Mohanani R., Ralph P., and Shreeve B. Requirements fixation. *International Conference on Software Engineering (ICSE)*, pages 895–906, 2014.
- [16] Mohanani R., Salman I., Turhan B., Rodriguez P., and Ralph P. Cognitive Biases in Software Engineering: A Systematic Mapping Study. *IEEE Transactions on Software Engineering*, 5589(c), 2018.
- [17] Ralph P. Toward a theory of debiasing software development. *Lecture Notes in Business Information Processing*, 93:92–105, 2011.
- [18] Ramin F. The Role of Egocentric Bias. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 122–124, 2020.
- [19] Runeson P., Höst M., Rainer A., and Regnell B. *Case Study Research in Software Engineering: Guidelines and Examples*. 2012.
- [20] Saldaña J. *The coding manual for qualitative researchers*. Sage Publications, 2021.
- [21] Schwaber K. and Sutherland J. The scrum guide. <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>, 2020. Accessed 2021-07-02.
- [22] Shepperd M., Mair C., and Jørgensen M. An Experimental Evaluation of a Debiasing Intervention for Professional Software Developers. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1510–1517, 2018.
- [23] Stacy W. and Macmillan J. Cognitive Bias in Software Engineering. *Communications of the ACM*, 38(6):57–63, 1995.
- [24] Tang A. Software designers, are you biased? In *Proceedings of the 6th international workshop on sharing and reusing architectural knowledge*, 2011.
- [25] Van Waardenburg G. and Van Vliet H. When agile meets the enterprise. *Information and Software Technology*, 55:2154–2171, 2013.
- [26] Zalewski A., Borowa K., and Kowalski D. On cognitive biases in requirements elicitation. In *Integrating Research and Practice in Software Engineering*, pages 111–123. Springer, 2020.
- [27] Zalewski A., Borowa K., and Ratkowski A. On cognitive biases in architecture decision making. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10475, pages 123–137, 2017.