Addressing Bad Feelings in Agile Contexts

K. Madampe
Monash University, Australia

R. Hoda
Monash University, Australia

J. Grundy
Monash University, Australia

***Abstract*—Like all people, software developers feel several emotions – positive and negative – when working on projects. But what are the underlying reasons for these emotions? Are their emotions linked to their work satisfaction and productivity? Can we boost developer satisfaction and productivity by remedying the causes of negative emotions? Agile is the most common software development approach currently used. Based on our empirical industry studies, we propose several solutions to overcome the causes of negative emotions in agile contexts.**

Improving productivity in teams has always been a priority in organisations, given their motives to amplify business performance[1]. This is no less for software teams, where they are expected to be productive throughout the project and timely deliver the software they work on. Team welfare has become an increasingly important concern, especially developer mental health. Work satisfaction has always been important, but increasingly so in recent years.

A single Google search on "how to be productive" results in thousands of articles, where the majority talk about work habits. Many of these habits are closely aligned with ways to stop one from getting burnout – physical, emotional, or mental exhaustion that are accompanied by decreased motivation, lowered performance, and negative attitudes toward oneself and others [1]. One could consider following these habitual improvements to uplift their productivity, but it might not always work.

Expecting software teams to be constantly high functioning and constantly productive is unrealistic. A key factor impacting productivity and satisfaction are developer *emotions* [2], [3], [4]. We have been exploring what causes negative agile developer emotions and looking for remedies to overcome them. In this article, we suggest some possible approaches to remedy causes of developer negative emotions in agile contexts discovered through our empirical research. Some of solutions may appear obvious, but act as strong

[1]https://www.pwc.com.au/digitalpulse/perform-business-productivity.html

reminders to developers to gain the maximum use of agile practices and artefacts, and agile's flexible nature.

We first summarise the State of Agile Report findings about productivity and agile practices. We briefly summarise what the literature says about developer productivity and their emotions, and then present our empirical research findings. We use these to suggest several solutions to addressing negative developer emotions from an agile perspective. Throughout this article, we use the terms *developer* and *practitioner* interchangeably, as developers play multiple roles in agile software contexts.

Productivity and agile practices

Over the past 16 years, the State of Agile Report (SOA) has presented many insights into adopting and scaling agile in enterprises, conducting an annual survey of agile practitioners worldwide. It has especially shed light on understanding the reasons for adopting agile, how agile has positively impacted several factors in organisations, and significant barriers to adopting and scaling agile practices. According to the 15th SOA [5], adopting agile methods within software teams has skyrocketed from 37% in 2020 to 86% in 2021. One of the top three reasons for *agile adoption* reported is to increase productivity. 60% of the participants believed that agile positively impacted team productivity. However, the most significant barriers to *adopting and scaling agile practices* is reported to be inconsistent use of processes and adaption of practices across teams, lack of organisational culture supporting agile methods, and lack of experience and expertise with agile practices [5]. In other words, teams adopt agile to increase productivity and agile positively impacts productivity. However, inconsistent processes and practices across teams, lack of expertise with agile practices and lack of organisational support hinder the adoption and scaling of agile practices [5]. **Overcoming these issues is crucial to getting the maximum benefit from practising agile**.

Developer productivity and their emotions

Several empirical studies in software engineering show a significant linkage between developer productivity and developer emotions. Through a study focusing on the impact of emotions on productivity during software development, Wrobel [6] found that *frustration* is felt often by developers, lowering their productivity. *Enthusiasm* and, interestingly, *anger* tend to increase productivity. Emotions transit from frustration $\rightarrow$ anger $\rightarrow$ contentment $\rightarrow$ enthusiasm. While it is interesting that negative emotions such as anger increase productivity, it is necessary to know the level of anger to understand if a high or low level of anger boosts productivity, and possibly, if and how this might be sustainable or even desirable. Giradi et al.'s [7] study provides evidence on the connection between emotions and productivity. They found that valence is positively correlated with perceived productivity with stronger connections in the afternoon whereas correlation between dominance and productivity is stronger in the morning for the developers.

Our empirical studies demonstrate that practitioners feel diverse emotions when they are working on software projects [8], [9], [10]. In one study, 201 practitioners shared with us how they feel when working on projects with numerous requirements changes (RCs) such as feature requests – an obvious phenomenon in agile contexts. We provided them with 20 emotions (positive and negative) using a well-established emotion scale, Job Affective Wellbeing Scale [11]. The practitioners told us when they felt those emotions. Their feedback indicated that their feelings vary at different milestones due to various reasons. Fig. 1 summarises these key milestones, the emotions they felt, and the reasons. The 'smileys' in the figure represent two levels of high and low emotions in each. High^2 represents emotions: *Energetic, Excited, Ecstatic, Enthusiastic, Inspired*, High^1 represents emotions: *At-ease, Calm, Content, Satisfied, Relaxed*, Low^1 represents emotions: *Angry, Anxious, Disgusted, Frightened, Furious*, and Low^2 represents emotions: *Bored, Depressed, Discouraged, Gloomy, Fatigued.* Key findings are the reasons behind practitioners feeling negative emotions at the project milestones of project commencement, during development, and when deadline is approached. Some of the experiences shared by our participants, were:

At project commencement: unstable require-

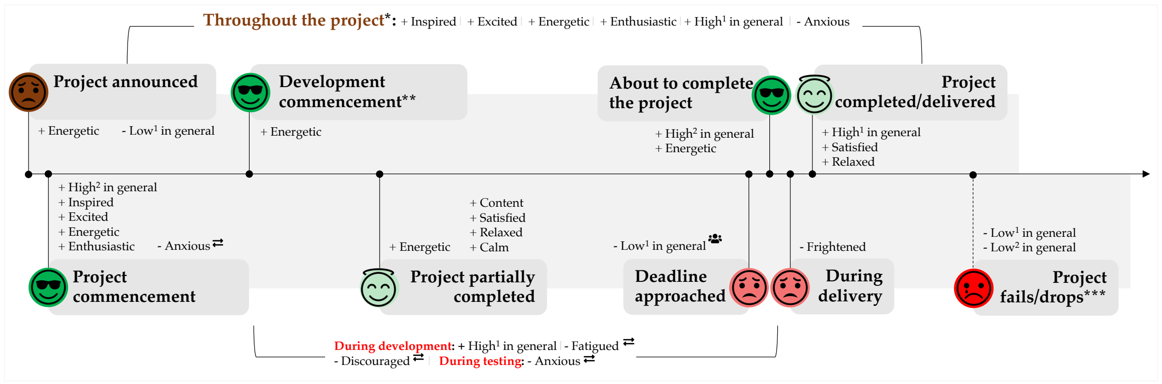
Throughout the project*: + Inspired + Excited + Energetic + Enthusiastic + High¹ in general - Anxious
Project announced
+ Energetic - Low¹ in general
Development commencement**
+ Energetic
About to complete the project
+ High² in general
+ Energetic
Project completed/delivered
+ High¹ in general
+ Satisfied
+ Relaxed
+ High² in general
+ Inspired
+ Excited
+ Energetic
+ Enthusiastic - Anxious ⇄
Project commencement
+ Energetic
+ Content
+ Satisfied
+ Relaxed
+ Calm
Project partially completed
- Low¹ in general 👥
Deadline approached
- Frightened
During delivery
- Low¹ in general
- Low² in general
Project fails/drops***
During development: + High¹ in general - Fatigued ⇄
- Discouraged ⇄ During testing: - Anxious ⇄

Figure 1. Emotion Dynamics of Practitioners in Project Life Cycle (*in general while specific emotions are triggered at specific milestones; **may overlap with the beginning of the project; ***may happen at any time; Emoji: Dominating emotion sub-scale; 😎: High^2; 😊: High^1; ☹: Low^1; 😠: Low^2; ●: both high/low emotions exist; Stimuli: ⇄: RC, 👥: Team)

ments. Unstable requirements result in *anxiety* at the commencement of the project:

> *"I was anxious when we first started and the requirements were still not settled on." – Tester*

During development: Requirements changes introduced after design and implementation. During development, when RCs are introduced after design and implementation, practitioners are *discouraged* and *fatigued*.

> *"I was fatigued in one such project (startup project), where almost every feature (3 out of 5 feature sets) had a change request in the midst of implementation phase. The most discouraging part is the time / phase when the requirement change happens. Most of the time it happens after the design and implementation phase." – Agile Coach/Scrum Master, Developer*

However, when an RC is introduced during testing, *anxiety* is felt, as reported by another participant:

> *"During online trading system development after the development phases is completed and we are in testing phase of our product client called to roll back the whole segment of project which make me and my team anxious." – Business Analyst.*

Deadline approached: None of the team members could solve the unresolved issues. As mentioned by a participant, as none of the practitioners in the team could solve unresolved issues, emotions such as boredom, depression, discouragement, gloom, and fatigue occur *"The above emotions happened to me when the deadline of the project approached. At the last day of the release there were some issues which couldn't be sorted out by anyone" – Agile Coach/Scrum Master, Developer*

We linked the State of Agile report findings on productivity and practices, literature on emotions and productivity in agile, and our research findings on emotions, causes and possible solutions through an agile lens. This is depicted in Fig. 2.

Overcoming causes of negative emotions

From our empirical studies [12], [10], we identified that practitioners use numerous agile practices and artefacts, and modifying and adopting some of these artefacts and practices may better support challenges in an agile environment. In Table 1, we highlight some possible solutions to the above mentioned issues that our studies suggest are used and/or positively viewed by practitioners. However, to verify that these solutions are more generally useful and significant, further empirical research needs to be done. It is also important to note that the possibility of choice and execution of the solutions may change from context to context. For example, for one team, it might be possible to conduct frequent customer demos whereas it might not be possible

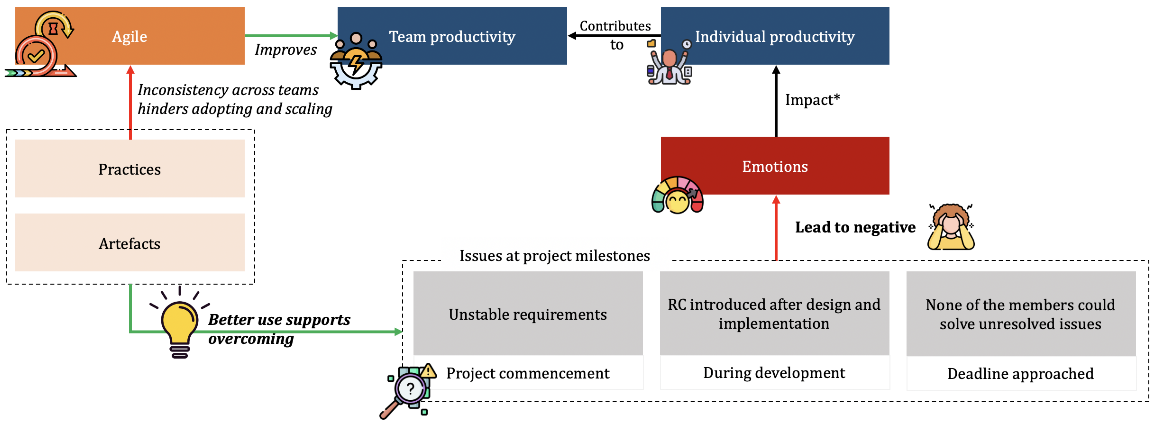
Agile
Improves
Team productivity
Contributes to
Individual productivity
Inconsistency across teams hinders adopting and scaling
Impact*
Practices
Artefacts
Emotions
Lead to negative
Issues at project milestones
Better use supports overcoming
Unstable requirements
RC introduced after design and implementation
None of the members could solve unresolved issues
Project commencement
During development
Deadline approached

Figure 2. Better use of agile practices and artefacts (green arrows) will support overcoming causes leading to negative emotions in practitioners, which will eventually contribute to a productive agile team (*italic*: State of Agile Survey findings; *: literature; **bold**: our survey findings; ***bold and italic***: possible agile-based solutions we suggest). The icons are from https://www.flaticon.com/

for some other team due to their circumstances.

At project commencement: Unstable requirements: Practitioners should consider actively labelling unstable, incomplete or unclear requirements in their product backlog if they clearly seen as vague or subject to change [12]. The same can be done with Scrum/ Kanban boards. User stories and use cases of such requirements should also be tagged. We suggest defining stability and an acceptable level or criterion to consider a requirement to be 'stable' so that such definitions and measurements can be used to decide whether to include them in the sprint backlog. Consideration should be given to marking such unstable requirements as low-priority items until further refinement with customers. Review meetings should be utilised to discuss. Practitioners may consider discussing unstable requirements with the team during daily standups/ team meetings. Having clarity through such discussions [13], or at least knowing that requirements may change by having them labeled in artefacts, will help reduce feeling negative emotions.

During development: RC introduced after design and implementation: Even though practitioners claim that they practice agile, it may not be possible to practice 'pure' agile [14] – – i.e., practising agile by following formative literature such as Scrum guide or Scrum primer [14]. However, [14] has shown that in the real world, many teams do not consistently follow all of these practices all the time. Teams may limit sprints to sub-phases of the software development life cycle (SDLC), as in the waterfall model. Introducing an RC to such an environment may make the triple constraints of cost, time, and scope suffer. RCs introduced after design and implementation tends to make practitioners feel negative emotions [9]. We suggest not limiting sprints to phases of the SDLC, but practising agile – as it was intended – as much as possible.

Another example is that the Scrum method if practised as intended prohibits RCs during a sprint [15]. We found that many teams find this practice overly prescriptive in practice [12]. However, this introduction of RCs during a sprint we found can have negative developer emotional consequences [9]. Late introduction of RCs could be avoided by conducting more frequent customer demos, and discussing need for refinements with customers during review meetings. Variations to agile practice is common, but some variations are harmful deviations – that are neither temporary nor justified – and can lead to losing the core benefits of using agile methods [14]. Introducing RCs once a sprint commences and especially later in the sprint, is one such deviation that leads to negative developer emotions.

Deadline approaching – Unresolved issues: There may be issues left unresolved when a project deadline is reached and our participants claimed they feel very negative when this happens [9], [10]. Proactive solutions, such as having an

Table 1. Possible Solutions from an Agile Perspective to Overcome Causes of Negative Emotions

Artefact*/ practice	Project commencement: Unstable requirements	During development: RC introduced after design and implementation	Deadline approached: None of the members could solve unresolved issues
Product backlog* Scrum/ Kanban board* User stories* Use cases*	Label unstable requirements where possible		
Sprint backlog	If an acceptable level of stability is not met, do not include in sprint backlog/ mark low priority until confirmed after discussion with customer	Do not limit sprints to phases of SDLC, practise agile	Have an empty backlog item: Allocate time for unforeseen issues
Daily standup/ team meeting	Discuss unstable requirements with the team		Discuss unresolved issues often during the standups, get help from others early without waiting till last minute help
Retrospectives			Discuss what went wrong and ways of improving for future sprints
Review meetings	Discuss unstable requirements with customer until acceptable level of stability reached get management support to defer RC to next Sprint	Discuss the requirements with the customer	
Collective estimation			Improve estimations
Customer demos		Conduct frequent customer demos	
Release planning			Avoid planning to release features with unresolved issues
Pair programming			Improve pair programming

empty sprint backlog item and allocating time for unforeseen issues, may help practitioners avoid stressful last minute rushes. Practitioners could improve their estimations through collective estimations. Better estimations come with experience, and if estimations are done collectively, practitioners with less experience be enabled to better manage workloads to deliver when a deadline is approaching [9]. Practitioners need to get the maximum use of daily standups to frequently discuss unresolved issues, and to proactively get help from others. Having unresolved issues near a deadline demands discussion during retrospectives so that such occurrences will not happen again in future sprints. Pair programming can also be better utilised to resolve some issues. Our practitioners report that pair programming is not a popular practice in many agile teams. However, pair programming can be an excellent way to identify and address unresolved issues that no single developer can solve last-minute.

Consistent Agile Practices: Our practitioners report that inconsistent use (using agile practices on and off during their project –sometimes using them, but sometimes not) of agile practices across teams hinders adopting and scaling agile [12]. Since software development is not a task of a single team, but a collaborative effort of multiple teams, it is vital to have the consistency of using practices and artefacts, which helps improve productivity.

Summary

Feeling a variety of emotions during software development is normal. But it is necessary not to allow negative emotions to impact individual productivity, which in turn contributes to overall team productivity. One of the main reasons why software teams adopt agile is that they believe it improves team productivity. Smart use of agile practices and artefacts can help avoid the causes of negative emotions when working on a software project.

ACKNOWLEDGMENTS

We thank all the participants who participated in our studies. This work was partially supported by the Monash Faculty of IT Postgraduate Publications Award. Madampe and Grundy are supported by ARC Laureate Fellowship FL190100035.

REFERENCES

1. G. R. VandenBos, *APA Dictionary of Psychology*. American Psychological Association, 2007.
2. R. Colomo-Palacios, A. Hernández-López, García-Crespo, and P. Soto-Acosta, "A study of Emotions in Requirements Engineering," in *Communications in Computer and Information Science*, 2010.
3. D. Graziotin, X. Wang, and P. Abrahamsson, "Do Feelings Matter? On the Correlation of Affects and the Self-Assessed Productivity in Software Engineering," *Journal of Software: Evolution and Process*, 2015.
4. A. Kolakowska, A. Landowska, M. Szwoch, W. Szwoch, and M. R. Wrobel, "Emotion Recognition and Its Application in Software Engineering," in *2013 6th International Conference on Human System Interactions, HSI 2013*, 2013.
5. "15 th State of Agile Report," tech. rep., https://info.digital.ai/rs/981-LQX-968/images/SOA15.pdf.
6. M. R. Wrobel, "Emotions in the Software Development Process," in *2013 6th International Conference on Human System Interactions, HSI 2013*, 2013.
7. D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and Perceived Productivity of Software Developers at the Workplace," *IEEE Transactions on Software Engineering*, pp. 1–1, 6 2021.
8. K. Madampe, R. Hoda, and P. Singh, "Towards Understanding Emotional Response to Requirements Changes in Agile Teams," in *IEEE/ACM 42nd International Conference on Software Engineering New Ideas and Emerging Results (ICSE-NIER'20)*, (Seoul, Republic of Korea), p. 4, ACM, New York, NY, USA, 2020.
9. K. Madampe, R. Hoda, and J. Grundy, "The Emotional Roller Coaster of Responding to Requirements Changes in Software Engineering," *IEEE Transactions on Software Engineering*, 2022.
10. K. Madampe, R. Hoda, and J. Grundy, "A Framework for Emotion-oriented Requirements Change Handling in Agile Software Engineering," *IEEE Transactions on Software Engineering*, pp. 1–20, 2023.
11. P. T. Van Katwyk, S. Fox, P. E. Spector, and K. Kelloway, "Using the Job-Related Affective Well-Being Scale (JAWS) to Investigate Affective Responses to Work Stressors.," *Journal of occupational health psychology*, vol. 5, no. 2, pp. 219–230, 2000.
12. K. Madampe, R. Hoda, and J. Grundy, "A Faceted Taxonomy of Requirements Changes in Agile Contexts," *IEEE Transactions on Software Engineering*, 2021.
13. K. Madampe, R. Hoda, and J. Grundy, "The role of emotional intelligence in handling requirements changes in software engineering," *arXiv preprint arXiv:2206.11603*, 2022.
14. Z. Masood, R. Hoda, and K. Blincoe, "Real World Scrum A Grounded Theory of Variations in Practice," *IEEE Transactions on Software Engineering*, pp. 1–1, 9 2020.
15. "The Scrum Guide." https://scrumguides.org/index.html.

Kashumi Madampe is a Research Fellow at Monash University, Melbourne, Australia. Her research interests are developer productivity and experience, software analytics and tools, AI for software engineering, product management, and privacy engineering. More details about her research can be found at https://kashumim.com. Contact her at kashumi.madampe@monash.edu.

Rashina Hoda is an Associate Professor in Software Engineering at the Faculty of Information Technology, Monash University, Melbourne. Her research focuses on human and socio-technical aspects of software engineering and empirical software engineering. She has introduced "socio-technical grounded theory (STGT)" for qualitative and mixed methods research and theory development. She serves as an Associate Editor for IEEE Transactions on Software Engineering, ICSE2023 SEIS co-chair, and ICSE2024 Workshops co-chair. For details see www.rashina.com. Contact her at rashina.hoda@monash.edu.

John Grundy received the BSc (Hons), MSc, and PhD degrees in computer science from the University of Auckland, New Zealand. He is an Australian Laureate fellow and a professor of software engineering at Monash University, Melbourne, Australia. He is an associate editor of the IEEE Transactions on Software Engineering, the Automated Software Engineering Journal, and IEEE Software. His current interests include domain–specific visual languages, model–driven engineering, large-scale systems engineering, and software engineering education. More details about his research can be found at https://sites.google.com/site/johncgrundy/. Contact him at john.grundy@monash.edu.